Solution Guide

MayaData

Deploying Elasticsearch on Kubernetes using OpenEBS

OpenEBS

MayaData

Contents

Overview

Elasticsearch is a distributed, free and open search and analytics engine for all types of data, including textual, numerical, geospatial, structured, and unstructured. Elasticsearch is built on Apache Lucene and was first released in 2010 by Elasticsearch N.V. (now known as Elastic). Known for its simple REST APIs, distributed nature, speed, and scalability, Elasticsearch is the central component of the Elastic Stack, a set of free and open tools for data ingestion, enrichment, storage, analysis, and visualization. Commonly referred to as the ELK Stack (after Elasticsearch, Logstash, and Kibana), the Elastic Stack now includes a rich collection of lightweight shipping agents known as Beats for sending data to Elasticsearch.[1].

This guide explains the basic installation for Elasticsearch operators on OpenEBS Local PV devices using KUDO. We will be installing Fluentd and Kibana to form the EFK stack. The guide will also provide a way to monitor the health of Elasticsearch using Prometheus and Grafana.

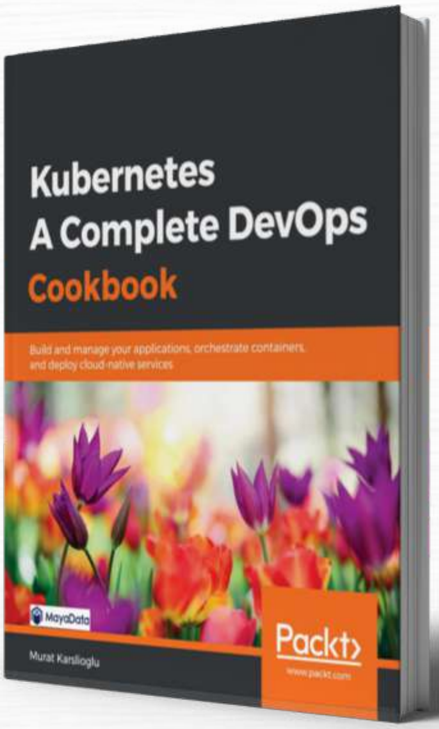
Before starting

You require an existing Kubernetes cluster. Kubernetes provides platform abstraction, cloud-native software runs, and behaves the same way on a managed Kubernetes service like AWS EKS, Google Cloud GKE, Microsoft AKS, DigitalOcean Kubernetes Service, or self-managed based on Red Hat OpenShift and Rancher. You can also use kubeadm, kubespray, minikube. Since you made it here, we assume you already have one configured.

MayaData team has proudly over 50 CKAs, years of experience building for enterprises, and running Kubernetes in production. If you need professional help to decide, we can connect you with one of our trusted partners. In case you want to learn more, just schedule a call (https://calendly.com/mayadata/15min) with us and we will send you a best-selling “Kubernetes - A Complete DevOps Cookbook,” also written by one of our own experts.

Free Book:

Kubernetes A Complete DevOps Cookbook


Kubernetes
A Complete DevOps
Cookbook
Build and manage your applications, orchestrate containers, and deploy cloud-native services
MayaData
Murat Karslioglu
Packt›
www.packt.com

Schedule a 15-minute call today to speak with one of our partners and receive a FREE copy of our new book.

Perform pre-configuration

We will use GKE, where we will install Elasticsearch stack (EFK) with OpenEBS storage engine. The Local PV volume will be provisioned on a node where elastic pod is getting scheduled and uses one of the matching unclaimed block devices, which will then use the entire block device for storing data. No other application can use this device. If users have limited blockdevices attached to some nodes, they can use nodeSelector in the application YAML to provision applications on particular nodes where the available block device is present. The recommended configuration is to have at least three nodes and three unclaimed external disks to be attached per node.

Let's review our setup used for the configuration.

Our Setup

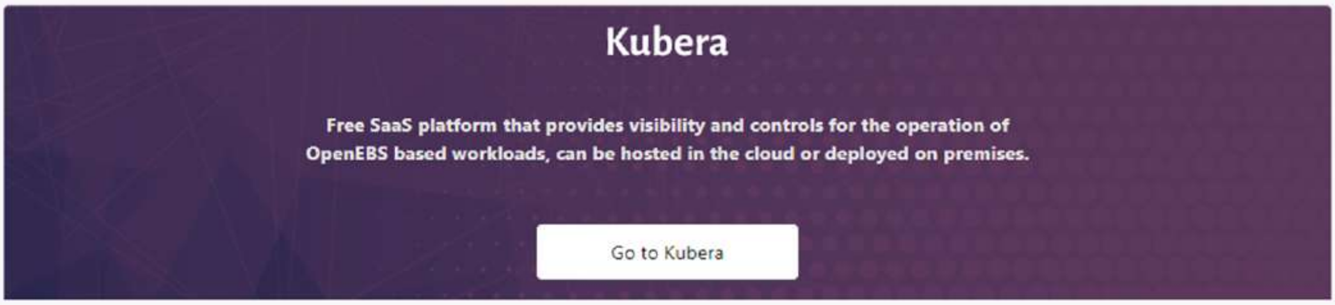
- 3 Nodes in GKE
- 4 vCPUs / node
- Ubuntu 18.04
- 16 GB memory / node
- 3 Local SSD(100Gi) / node
- GCP instance type: e2-standard-4
- Kubernetes version: v1.18

Getting Started with OpenEBS

Let's start the installation of OpenEBS using the Kubera platform.

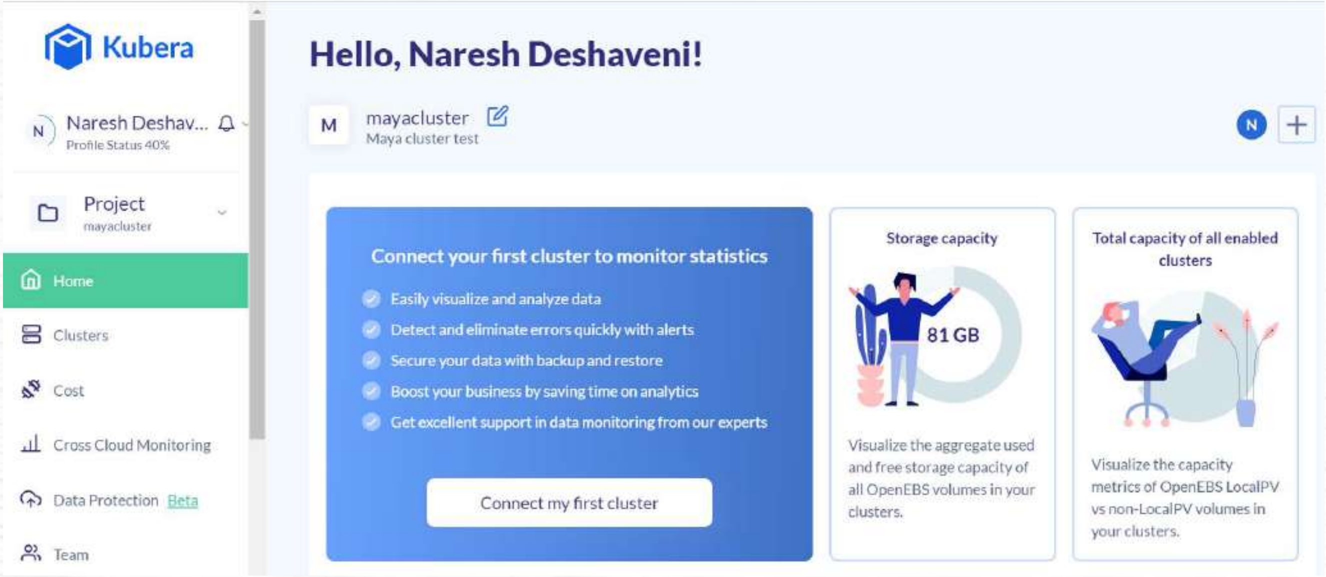
Installing OpenEBS using Kubera

Signup here for your free Kubera account. Then click on **Go to Kubera.**



Kubera

Free SaaS platform that provides visibility and controls for the operation of OpenEBS based workloads, can be hosted in the cloud or deployed on premises.



Hello, Naresh Deshaveni!

Connect your first cluster to monitor statistics

- Easily visualize and analyze data
- Detect and eliminate errors quickly with alerts
- Secure your data with backup and restore
- Boost your business by saving time on analytics
- Get excellent support in data monitoring from our experts

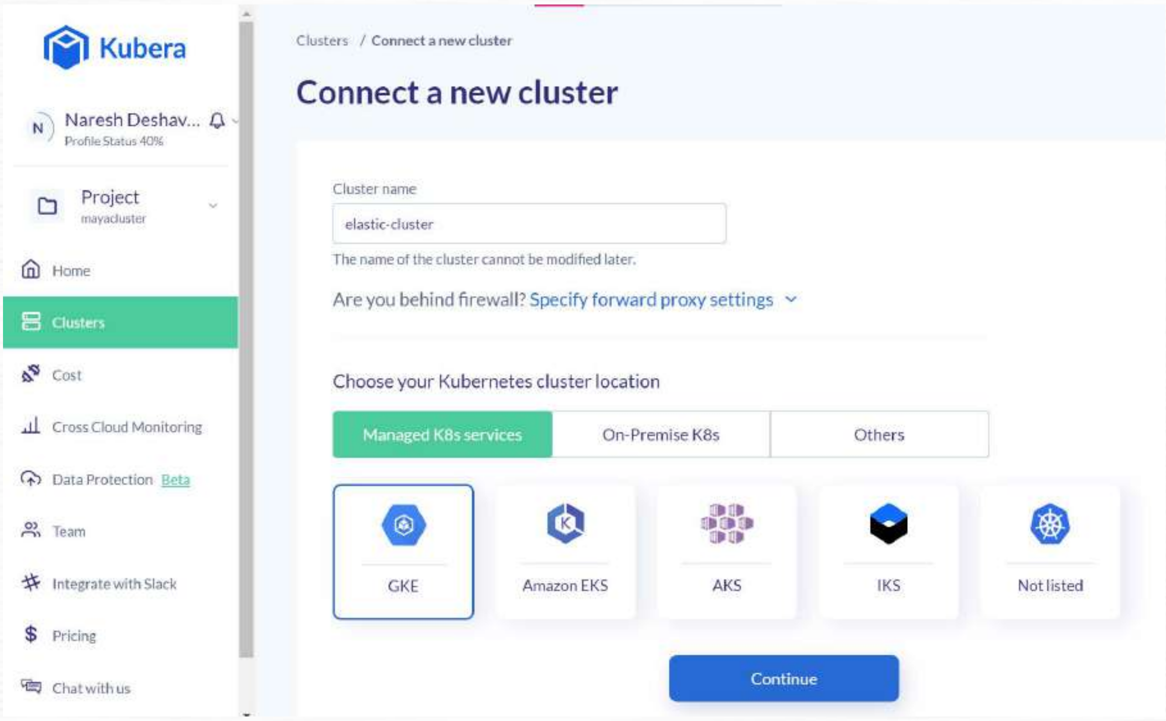
Storage capacity

Visualize the aggregate used and free storage capacity of all OpenEBS volumes in your clusters.

Total capacity of all enabled clusters

Visualize the capacity metrics of OpenEBS LocalPV vs non-LocalPV volumes in your clusters.

Follow the instructions to connect your cluster to your Kubera account.



Connect a new cluster

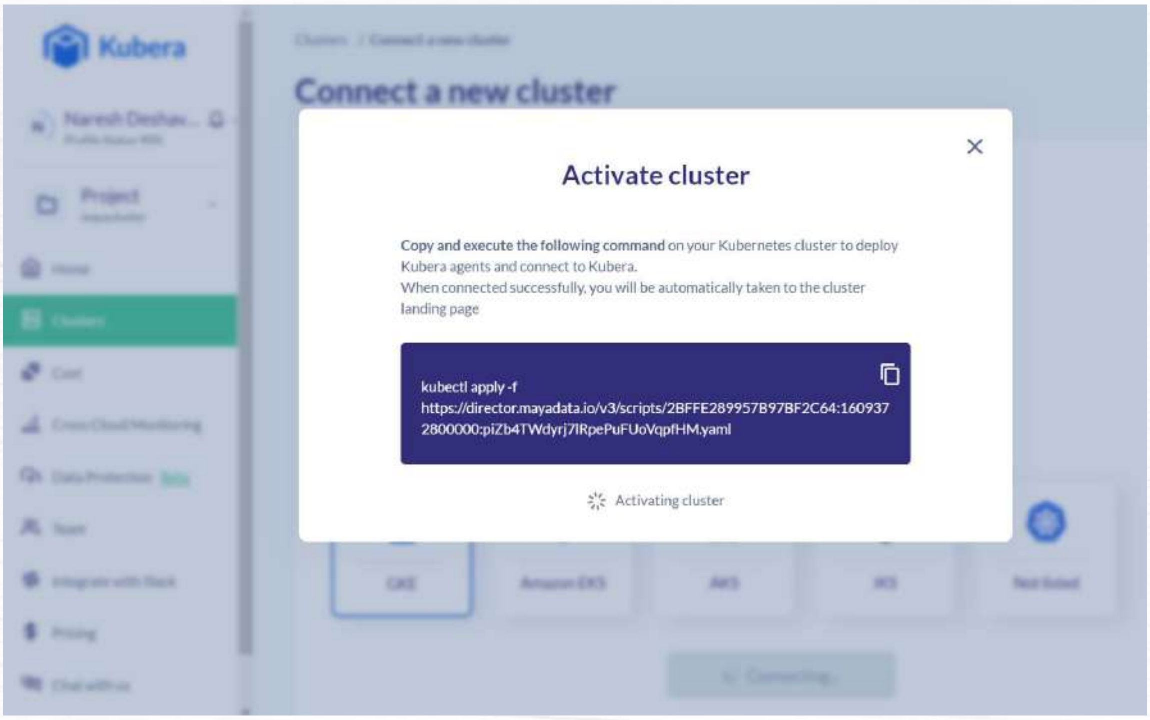
Cluster name: elastic-cluster

The name of the cluster cannot be modified later.

Are you behind firewall? Specify forward proxy settings

Choose your Kubernetes cluster location: Managed K8s services

It will open a window with the command to connect your K8s cluster with the Kubera SaaS version. Copy and execute the command on your own Kubernetes cluster.

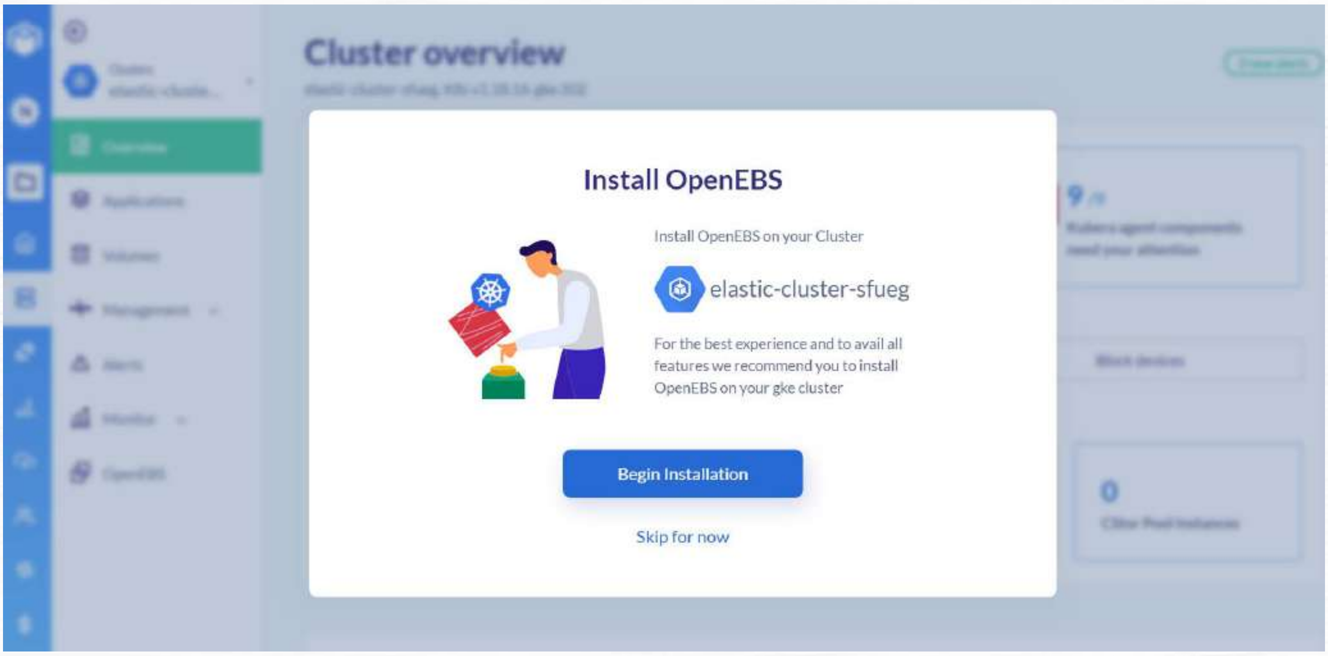


Activate cluster

Copy and execute the following command on your Kubernetes cluster to deploy Kubera agents and connect to Kubera.
When connected successfully, you will be automatically taken to the cluster landing page

```
kubectl apply -f
https://director.mayadata.io/v3/scripts/2BFFE289957B97BF2C64:160937
2800000:piZb4TWdyrj7IRpePuFUoVqpfHM.yaml
```

If OpenEBS was already installed using Kubera in your cluster, skip this process. If OpenEBS was not installed using Kubera, then click on **Begin Installation**, which will lead to a page where you can choose how to install OpenEBS.



Cluster overview

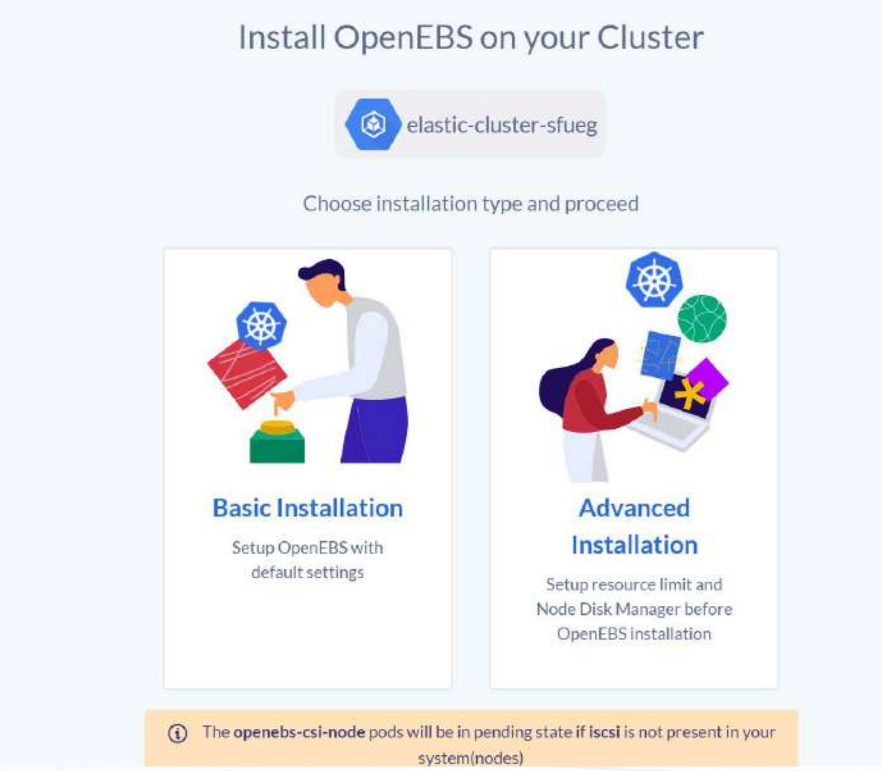
Install OpenEBS

Install OpenEBS on your Cluster

elastic-cluster-sfueg

For the best experience and to avail all features we recommend you to install OpenEBS on your gke cluster

Follow the on-screen instructions titled **Basic Installation** for the default installation of OpenEBS Enterprise Edition on your K8s cluster.



Install OpenEBS on your Cluster

elastic-cluster-sfueg

Choose installation type and proceed

Basic Installation

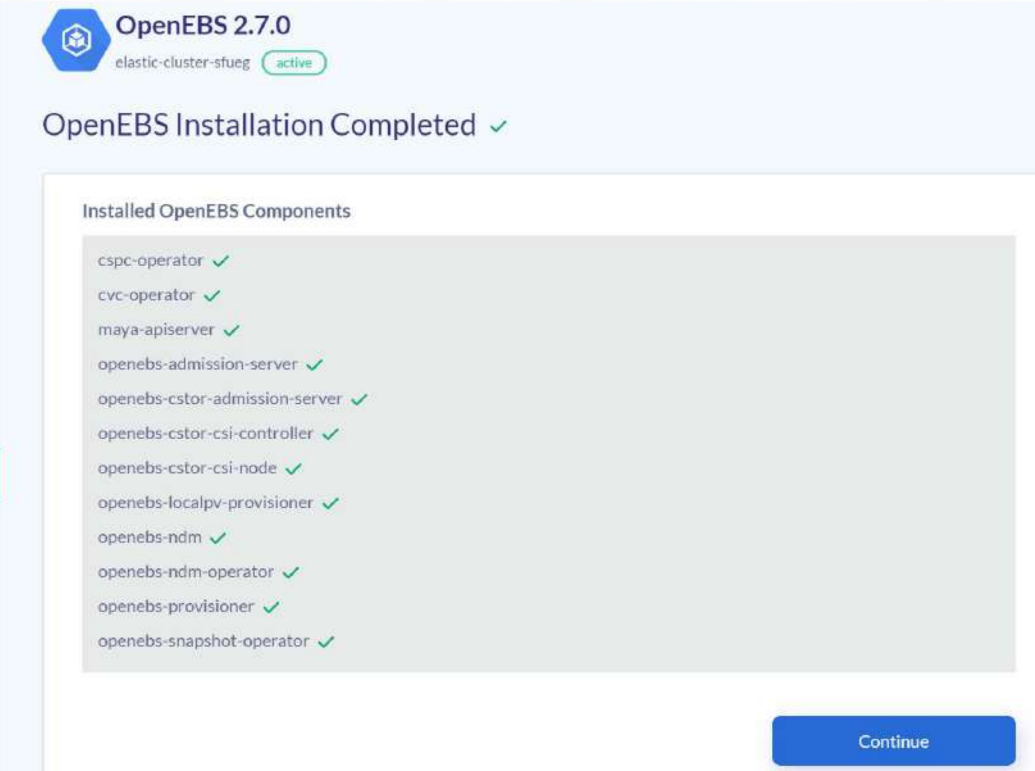
Setup OpenEBS with default settings

Advanced Installation

Setup resource limit and Node Disk Manager before OpenEBS installation

ⓘ The **openebs-csi-node** pods will be in pending state if **iscsi** is not present in your system(nodes)

Click on **Deploy OpenEBS** on the next screen and verify the installation status from the next screen. After successful installation of OpenEBS, click on Continue. If you run into any errors or have questions, community support for Kubera is available on Slack.



OpenEBS 2.7.0

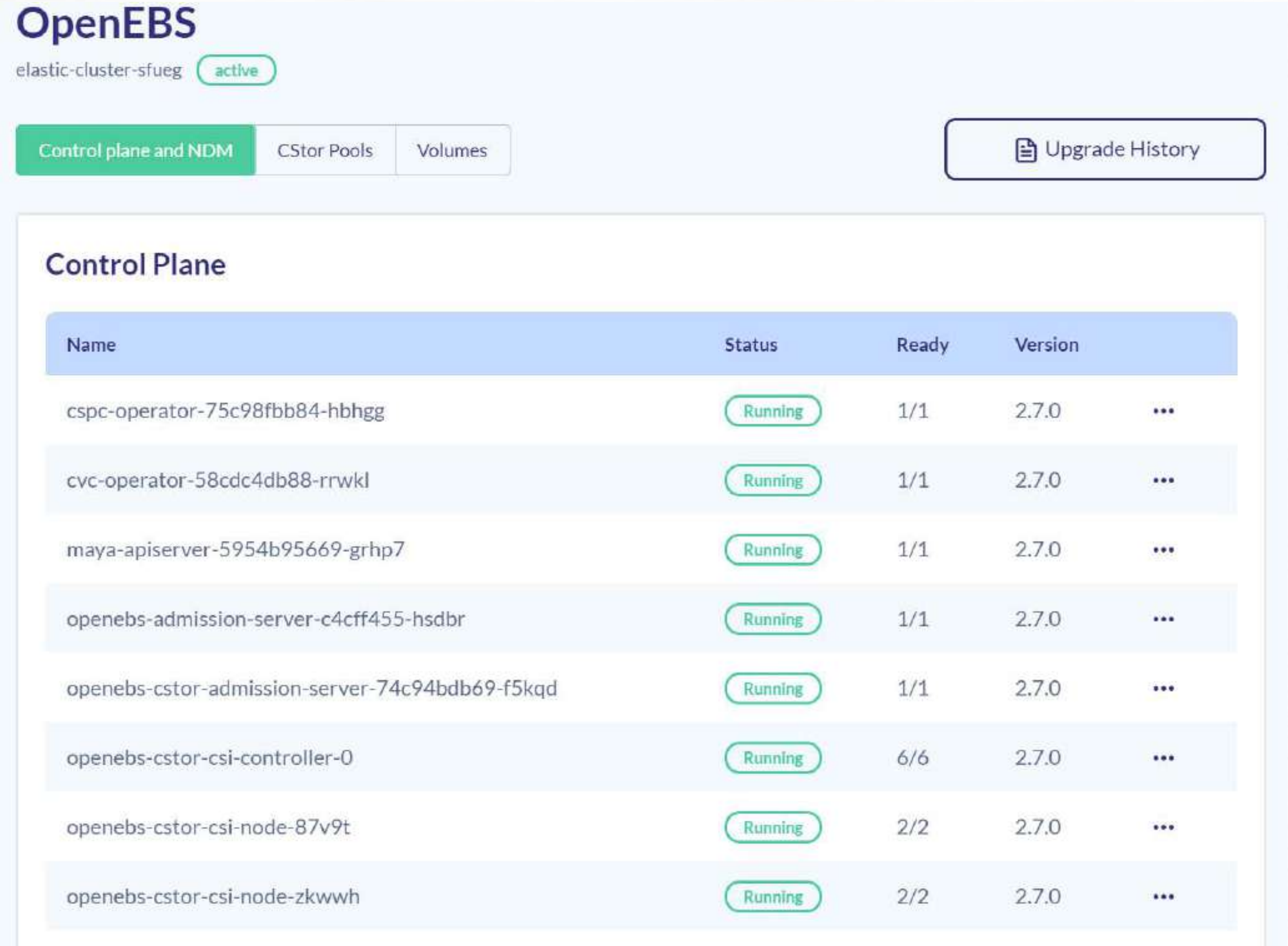
elastic-cluster-sfueg active

OpenEBS Installation Completed ✓

Installed OpenEBS Components

- cspc-operator ✓
- cvc-operator ✓
- maya-apiserver ✓
- openebs-admission-server ✓
- openebs-cstor-admission-server ✓
- openebs-cstor-csi-controller ✓
- openebs-cstor-csi-node ✓
- openebs-localpv-provisioner ✓
- openebs-ndm ✓
- openebs-ndm-operator ✓
- openebs-provisioner ✓
- openebs-snapshot-operator ✓

Now, you will see OpenEBS control-plane has been enabled on your Kubernetes cluster.



OpenEBS

elastic-cluster-sfueg active

Control Plane

Name	Status	Ready	Version	
cspc-operator-75c98fbb84-hbhgg	Running	1/1	2.7.0	•••
cvc-operator-58cdc4db88-rrwkl	Running	1/1	2.7.0	•••
maya-apiserver-5954b95669-grhp7	Running	1/1	2.7.0	•••
openebs-admission-server-c4cff455-hsdbr	Running	1/1	2.7.0	•••
openebs-cstor-admission-server-74c94bdb69-f5kqd	Running	1/1	2.7.0	•••
openebs-cstor-csi-controller-0	Running	6/6	2.7.0	•••
openebs-cstor-csi-node-87v9t	Running	2/2	2.7.0	•••
openebs-cstor-csi-node-zkwwh	Running	2/2	2.7.0	•••

Configuring GCP Project

If you are on GCP, you need to select your project before you can attach disks to the nodes.

```
$ gcloud config set project <your-project-name-here>
```

Create three 100Gi disks for each node.

```
$ gcloud compute disks create disk-1 disk-2 disk-3 disk-4 disk-5 disk-6 disk-7 disk-
8 disk-9 --size=100G --zone=us-central1-c
```

Note: Provide the required size initially as currently Local PV volume will not allow you to expand the capacity later.

Attaching disks to each Node

Now, we will add 3 additional devices to each node. Disks will be later consumed by Elastic. This step can be done through your cloud vendor's web user interface, or if you are running in a VM, you can use your hypervisor to add 3 additional virtual devices to each node. In this example, we have used GCP and added the disks using the gcloud CLI tool.

Get list of Instance IDs per each Zone

```
$ gcloud compute instances list --zones us-central1-c
NAME  ZONE  MACHINE_TYPE   PREEMPTIBLE  INTERNAL_IP  EXTERNAL_IP  STATUS
gke-cluster-2-default-pool-c8c74720-65nf  us-central1-c  e2-standard-4
10.128.15.193  34.70.58.139    RUNNING
gke-cluster-2-default-pool-c8c74720-90r9  us-central1-c  e2-standard-4
10.128.15.194  35.188.38.187   RUNNING
gke-cluster-2-default-pool-c8c74720-cjjc  us-central1-c  e2-standard-4
10.128.15.196  35.224.255.199  RUNNING
```

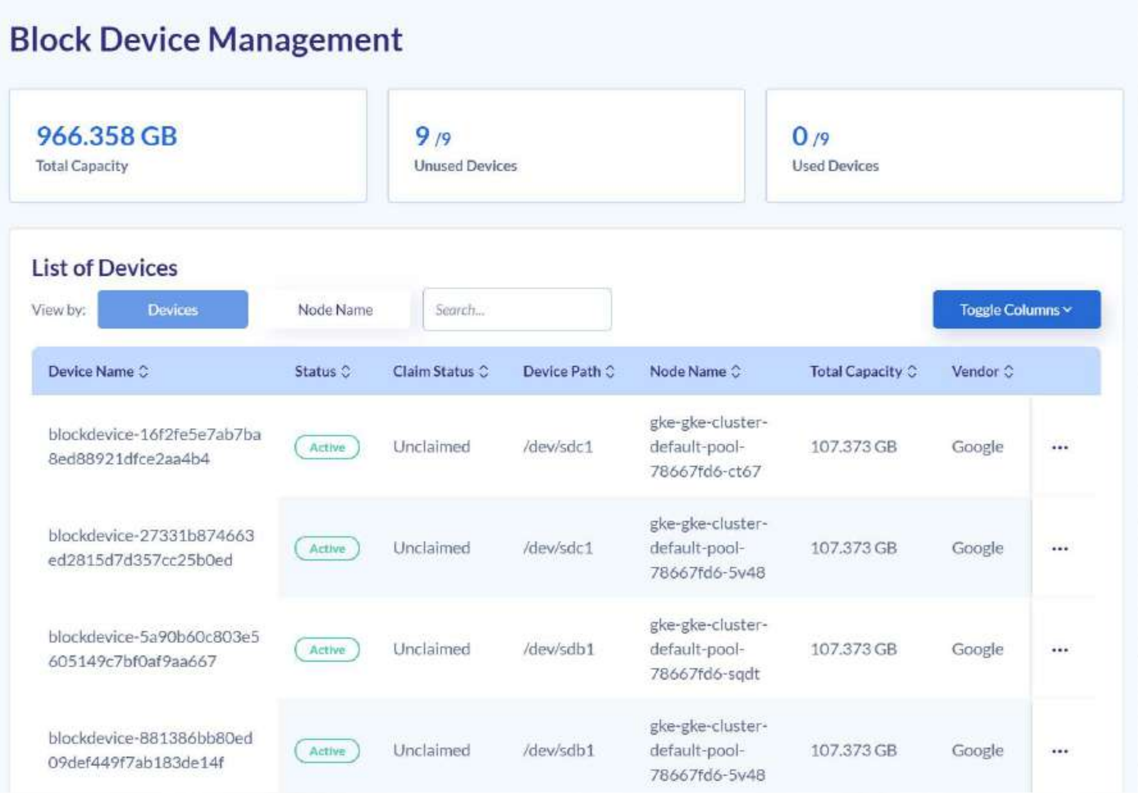
Now, attach the disks to each node.

```
$ gcloud compute instances attach-disk gke-cluster-2-default-pool-c8c74720-65nf  --
disk disk-1 --device-name disk-1 --zone us-central1-c
$ gcloud compute instances attach-disk gke-cluster-2-default-pool-c8c74720-65nf  --
disk disk-2 --device-name disk-2 --zone us-central1-c
$ gcloud compute instances attach-disk gke-cluster-2-default-pool-c8c74720-65nf  --
disk disk-3 --device-name disk-3 --zone us-central1-c
$ gcloud compute instances attach-disk gke-cluster-2-default-pool-c8c74720-90r9  --
disk disk-4 --device-name disk-4 --zone us-central1-c
```

```
$ gcloud compute instances attach-disk gke-cluster-2-default-pool-c8c74720-90r9  --
disk disk-5 --device-name disk-5 --zone us-central1-c
$ gcloud compute instances attach-disk gke-cluster-2-default-pool-c8c74720-90r9  --
disk disk-6 --device-name disk-6 --zone us-central1-c
$ gcloud compute instances attach-disk gke-cluster-2-default-pool-c8c74720-cjjc  --disk
disk-7 --device-name disk-7 --zone us-central1-c
$ gcloud compute instances attach-disk gke-cluster-2-default-pool-c8c74720-cjjc  --disk
disk-8 --device-name disk-8 --zone us-central1-c
$ gcloud compute instances attach-disk gke-cluster-2-default-pool-c8c74720-cjjc  --disk
disk-9 --device-name disk-9 --zone us-central1-c
```

Verify the Block Device information

You can verify the attached Block Device information from Kubera portal under **Management > Block Devices** from the corresponding cluster page.



Block Device Management

966.358 GB
Total Capacity

9 /9
Unused Devices

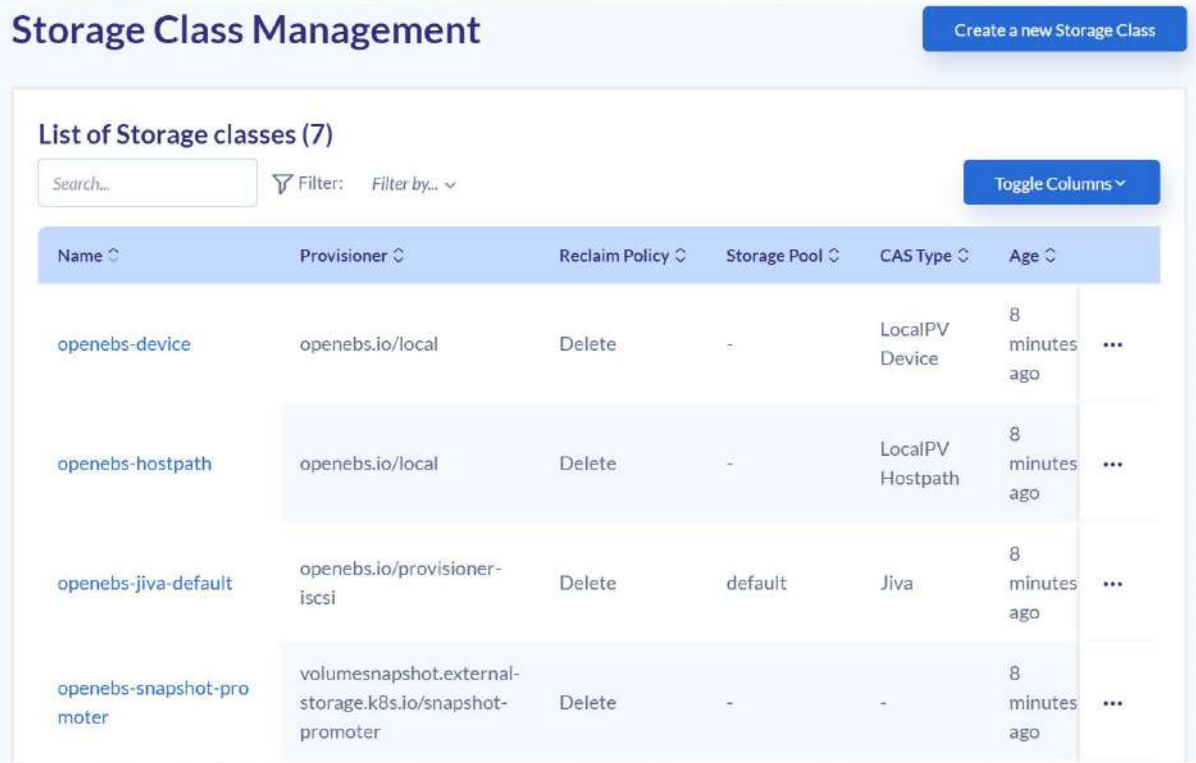
0 /9
Used Devices

List of Devices

Device Name	Status	Claim Status	Device Path	Node Name	Total Capacity	Vendor	
blockdevice-16f2fe5e7ab7ba8ed88921dfce2aa4b4	Active	Unclaimed	/dev/sdc1	gke-gke-cluster-default-pool-78667fd6-ct67	107.373 GB	Google	...
blockdevice-27331b874663ed2815d7d357cc25b0ed	Active	Unclaimed	/dev/sdc1	gke-gke-cluster-default-pool-78667fd6-5v48	107.373 GB	Google	...
blockdevice-5a90b60c803e5605149c7bf0af9aa667	Active	Unclaimed	/dev/sdb1	gke-gke-cluster-default-pool-78667fd6-sqdt	107.373 GB	Google	...
blockdevice-881386bb80ed09def449f7ab183de14f	Active	Unclaimed	/dev/sdb1	gke-gke-cluster-default-pool-78667fd6-5v48	107.373 GB	Google	...

Verify default Storage Class

You can verify the installed Storage Class information from Kubera portal under **Management > Storage Classes** from the corresponding cluster page.



Storage Class Management

List of Storage classes (7)

Name	Provisioner	Reclaim Policy	Storage Pool	CAS Type	Age
openebs-device	openebs.io/local	Delete	-	LocalPV Device	8 minutes ago
openebs-hostpath	openebs.io/local	Delete	-	LocalPV Hostpath	8 minutes ago
openebs-jiva-default	openebs.io/provisioner-iscsi	Delete	default	Jiva	8 minutes ago
openebs-snapshot-promoter	volumesnapshot.external-storage.k8s.io/snapshot-promoter	Delete	-	-	8 minutes ago

From the default StorageClass, we use openebs-device for using persistent storage for running EFK pods.

Installing KUDO Operator

In this section, we will install the KUDO operator. We will later deploy the latest available version of elasticsearch applications through KUDO operator. Check the release section for getting the latest Kudo version.

You need **go** utility to be installed in your setup as a prerequisite. Install **go** utility in your environment if it is not installed.

Check the version of Go using the following command:

```
$ go version
go version go1.15.7 linux/amd64
```

Ensure the following ENV variable are set correctly:

- GOROOT
- GOPATH
- PATH

The above environments have been configured in our setup using the following way:

```
$ export GOROOT=/usr/local/go
$ export GOPATH=$HOME/gopath
$ export PATH=$GOPATH/bin:$GOROOT/bin:$PATH
```

The following is a sample output to verify the configured ENV variables:

```
$ echo $GOROOT
/usr/local/go
$ echo $GOPATH
/home/k8s/gopath
$ echo $PATH
/home/k8s/maya/elastic/es-mon/google-cloud-sdk/bin:/home/k8s/gopath/bin:/usr/local/go/
bin:/home/k8s/.local/bin:/home/k8s/bin:/usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin
```

Use the latest stable version of KUDO CLI. KUDO v0.18.2 was the latest stable version when we updated this article. The same version will be used for installing KUDO server as well. The latest version of KUDO can be checked from here.

```
$ VERSION=0.18.2
$ OS=$(uname | tr '[:upper:]' '[:lower:]')
$ ARCH=$(uname -m)
$ wget -O kubectl-kudo
https://github.com/kudobuilder/kudo/releases/download/v${VERSION}/kubectl-
kudo_${VERSION}_${OS}_${ARCH}
$ chmod +x kubectl-kudo
```

Now, add the downloaded kubectl-kudo binary to your path.

```
$ sudo mv kubectl-kudo /usr/local/bin/kubectl-kudo
```

Verify the version of kubectl-kudo CLI using the following command:

```
$ kubectl-kudo version
KUDO Version: version.Info{GitVersion:"0.18.2", GitCommit:"43929950",
BuildDate:"2021-01-20T08:26:44Z", GoVersion:"go1.13.4", Compiler:"gc",
Platform:"linux/amd64", KubernetesClientVersion:"v0.19.2"}
```

Verify if Cert-manager is installed

For installing KUDO operator, the Cert-manager must be already installed in your cluster. If not, install the Cert-manager. The instruction can be found from here. Since our K8s version is v1.18.12, we have installed Cert-manager using the following command.

```
$ kubectl apply -f https://github.com/jetstack/cert-
manager/releases/download/v1.2.0/cert-manager.yaml
```

```
$ kubectl get pods --namespace cert-manager
NAME                                        READY     STATUS        RESTARTS
AGE
cert-manager-7747db9d88-7qjnm                1/1      Running       0
 81m
cert-manager-cainjector-87c85c6ff-whnzr       1/1     Running       0
81m
cert-manager-webhook-64dc9fff44-qww8s        1/1      Running       0
 81m
```

Installing KUDO operator into cluster

Once prerequisites are installed you will need to initialize the KUDO operator. The following command will install KUDO v0.18.0.

```
$ kubectl-kudo init --version 0.18.2
$KUDO_HOME has been configured at /home/k8s/.kudo

✓   Installed Cards
✓   Installed Namespace
✓   Installed Service Account
✓   Installed Webhook
✓   Installed Kudo Controller
```

Verify pods in the kudo-system namespace:

```
$ kubectl get pod -n kudo-system
NAME                                      READY   STATUS     RESTARTS   AGE
kudo-controller-manager-0                 1/1       Running    0            25s
```

Setting Open EBS Storage Class as default

Change the default storage class from your current setting to OpenEBS LocalPV Device. For example, in this tutorial default storage class is used as **openebs-device from standard.**

```
$ kubectl patch storageclass standard -p '{"metadata":
{"annotations":{"storageclass.kubernetes.io/is-default-class":"false"}}}'
```

```
$ kubectl patch storageclass openebs-device -p '{"metadata":
{"annotations":{"storageclass.kubernetes.io/is-default-class":"true"}}}'
```

Verify default Storage Class

List the storage classes and verify openebs-device is set to default.

```
$ kubectl get sc
NAME                   PROVISIONER                                   RECLAIMPOLICY
VOLUME BINDING MODE      ALLOW VOLUME EXPANSION                AGE
openebs-device (default)    openebs.io/local
Delete        WaitForFirstConsumer    false                                4h30m
openebs-hostpath            openebs.io/local
Delete        WaitForFirstConsumer    false                                4h30m
openebs-jiva-default        openebs.io/provisioner-iscsi
Delete        Immediate               false                                4h30m
openebs-snapshot-promoter    volumesnapshot.external-
storage.k8s.io/snapshot-promoter    Delete        Immediate
```

```
false             4h30m
premium-rwo                pd.csi.storage.gke.io
Delete        WaitForFirstConsumer   true                                   5h13m
standard                kubernetes.io/gce-pd
Delete        Immediate           true                                      5h13m
standard-rwo               pd.csi.storage.gke.io
Delete        WaitForFirstConsumer   true                                   5h13
```

Install Elastic - Get Elastic Up and Running

In this section, we will install the Elastic operator using KUDO.

Installing Kudo Elastic operator

Set instance and namespace variables:

```
$ export instance_name=elastic
$ export namespace_name=default

$ kubectl-kudo install elastic --namespace=$namespace_name --instance
$ instance_name
operator default/elastic created
operatorversion default/elastic-7.0.0-0.2.1 created
instance default/elastic created
```

Verifying Elastic Pods

```
$ kubectl get pods -n $namespace_name
NAME                    READY       STATUS       RESTARTS      AGE
elastic-coordinator-0   1/1         Running      0             31s
elastic-data-0          1/1         Running      0             56s
elastic-data-1          1/1         Running      0             44s
elastic-master-0        1/1         Running      0             2m31s
elastic-master-1        1/1         Running      0             119s
elastic-master-2        1/1         Running      0             90s
```

Verifying Services

```
$ kubectl get svc -n $namespace_name
NAME                      TYPE         CLUSTER-IP    EXTERNAL-IP    PORT(S)     AGE
elastic-coordinator-hs    ClusterIP    None          <none>         9200/TCP    62s
elastic-data-hs           ClusterIP    None          <none>         9200/TCP    87s
elastic-ingest-hs         ClusterIP    None          <none>         9200/TCP    50s
elastic-master-hs         ClusterIP    None          <none>         9200/TCP    3m2s
kubernetes                ClusterIP    10.48.0.1     <none>         443/TCP     5h18m
```

Verifying Elastic instance status

```
$ kubectl kudo plan status --namespace=$namespace_name \
 --instance $instance_name

Plan(s) for "elastic" in namespace "default":
✓   Elastic (Operator-Version: "elastic-7.0.0-0.2.1" Active-Plan: "deploy")

✓   Plan deploy (serial strategy) [COMPLETE], last updated 2021-02-22 16:18:17

✓   Phase deploy-master (parallel strategy) [COMPLETE]

✓   Step deploy-master [COMPLETE]

✓   Phase deploy-data (parallel strategy) [COMPLETE]

✓   Step deploy-data [COMPLETE]

✓   Phase deploy-coordinator (parallel strategy) [COMPLETE]

✓   Step deploy-coordinator [COMPLETE]

✓   Phase deploy-ingest (parallel strategy) [COMPLETE]

✓   Step deploy-ingest [COMPLETE]
```

We have upgraded all elastic StatefulSets images to elasticsearch:7.10.1 from elastic search:7.0.0. You can find the latest release of ElasticSearch from here.

Accessing Elastic instance

Enter into one of the master pod using exec command:

```
$ kubectl exec -it elastic-master-0 -- bash
[root@elastic-master-0 elasticsearch]#
```

Run below command inside Elastic master pod:

```
$ curl -X POST "elastic-coordinator-hs:9200/twitter/_doc/" -H 'Content-Type:
application/json' -d'
 {
     "user" : "openebs",
     "post_date" : "2021-03-02T14:12:12",
     "message" : "Test data entry"
 }
 '
{"_index":"twitter","_type":"_doc","_id":"LoliyXcBg9iVzVnOj5QL","_version":1,"result":"cre
ated","_shards":{"total":2,"successful":1,"failed":0},"_seq_no":0,"_primary_term":1}[root@
elastic-master-0 elasticsearch]#
```

Get test results for the above test:

```
$ curl -X GET "elastic-coordinator-
hs:9200/twitter/_search?q=user:openebs&pretty"
```

Get the details of ElasticSearch cluster:

```
$ curl localhost:9200
{
"name" : "elastic-master-0",
"cluster_name" : "elastic-cluster",
"cluster_uuid" : "A0qErYmCS2OpJtmgR_j3ow",
"version" : {
"number" : "7.10.1",
"build_flavor" : "default",
"build_type" : "docker",
"build_hash" : "1c34507e66d7db1211f66f3513706fdf548736aa",
"build_date" : "2020-12-05T01:00:33.671820Z",
```

continued to the next page..

```
"build_snapshot" : false,
"lucene_version" : "8.7.0",
"minimum_wire_compatibility_version" : "6.8.0",
“minimum_index_compatibility_version" : "6.0.0-beta1"
},
"tagline" : "You Know, for Search”
}
```

Installing Kibana

First, add helm repository of Elastic

```
$ helm repo add elastic https://helm.elastic.co & helm repo update
```

Install Kibana deployment using helm command. Ensure to meet required prerequisites corresponding to your helm version. Fetch the Kibana values.yaml:

```
$ wget https://raw.githubusercontent.com/elastic/helm-
charts/master/kibana/values.yaml
```

Edit the following parameters:

edit `elasticsearchHosts` as "http://elastic-coordinator-hs:9200" # service name of Elastic search

edit `service.type` as "NodePort"

edit `service.nodePort` as "30295" # since this port is already added in our network firewall rules.

edit `imageTag` as "7.10.1" , it should be the same image tag of ElasticSearch. In our case, ElasticSearch image tag is 7.10.1

Now install Kibana using Helm:

```
$ helm install  kibana -f values.yaml elastic/kibana
```

Verifying Kibana Pods and Services

```
$ kubectl get pod
NAME                              READY   STATUS    RESTARTS   AGE
elastic-coordinator-0             1/1     Running   0          12m
elastic-data-0                    1/1     Running   0          12m
elastic-data-1                    1/1     Running   0          12m
elastic-master-0                  1/1     Running   0          11m
elastic-master-1                  1/1     Running   0          11m
elastic-master-2                  1/1     Running   0          12m
kibana-kibana-74cbc4d654-h8djr    1/1     Running   0          6m33s
```

```
$ kubectl get svc
NAME                    TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)          AGE
elastic-coordinator-hs  ClusterIP   None           <none>        9200/TCP         18m
elastic-data-hs         ClusterIP   None           <none>        9200/TCP         19m
elastic-ingest-hs       ClusterIP   None           <none>        9200/TCP         18m
elastic-master-hs       ClusterIP   None           <none>        9200/TCP         20m
kibana-kibana           NodePort    10.48.12.146   <none>        5601:30295/TCP   7m1s
kubernetes              ClusterIP   10.48.0.1      <none>        443/TCP          5h36m
```

Accessing Kibana dashboard

Verify Kibana service is accessible over web using

```
<any_node_external-ip>:<NodePort>
```

Example:

```
http://34.67.160.246:30295/
```

Installing Fluentd-ES

Fetch the values.yaml:

```
$ wget
https://raw.githubusercontent.com/bitnami/charts/master/bitnami/fluentd/values.yaml
$ helm repo add bitnami https://charts.bitnami.com/bitnami & helm repo update
```

Replace the following section in the values.yaml file with new content:

Old:

```
# Send the logs to the standard output
      <match **>
        @type stdout
      </match>
```

New:

```
# Send the logs to the elasticsearch output
      <match **>
        @type elasticsearch
        include_tag_key true
        host elastic-coordinator-hs
        port 9200
        logstash_format true

        <buffer>
          @type file
          path /opt/bitnami/fluentd/logs/buffers/logs.buffer
          flush_thread_count 2
          flush_interval 5s
        </buffer>
      </match>
```

Install Fluentd- ElasticSearch DaemonSet using the new values:

```
$ helm install fluentd -f values.yaml bitnami/fluentd
```

Verify Fluentd Daemonset, Pods and Services:

```
$ kubectl get ds
NAME     DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE
SELECTOR   AGE
fluentd     3          3            3          3                3              <none>
74m
```

```
$ kubectl get pod
NAME                                 READY       STATUS          RESTARTS     AGE
pod/elastic-coordinator-0            1/1         Running         0            67m
pod/elastic-data-0                   1/1         Running         0            66m
pod/elastic-data-1                   1/1         Running         0            67m
pod/elastic-master-0                 1/1         Running         0            66m
pod/elastic-master-1                 1/1         Running         0            66m
pod/elastic-master-2                 1/1         Running         0            66m
pod/fluentd-0                        1/1         Running         0            6m5s
pod/fluentd-4sbs4                    1/1         Running         2            6m5s
pod/fluentd-5mvv9                    1/1         Running         2            6m5s
pod/fluentd-z2sxt                    1/1         Running         2            6m5s
pod/kibana-kibana-74cbc4d654-h8djr   1/1         Running         0            9m46s
```

Getting logs from the indices

Goto Kibana dashboard.

Click on `Management->Stack Management` which is placed at left side bottom most.

Click on `index patterns` listed under `Kibana` and then click on `Create Index pattern`.

Provide `logstash-*` inside the index pattern box and then select `Next` step.

In the next step, inside the `Time Filter field name`, select the `@timestamp` field from the dropdown menu, and click `Create index pattern`.

Now click on the `Discover` button listed on the top left of the side menu bar.

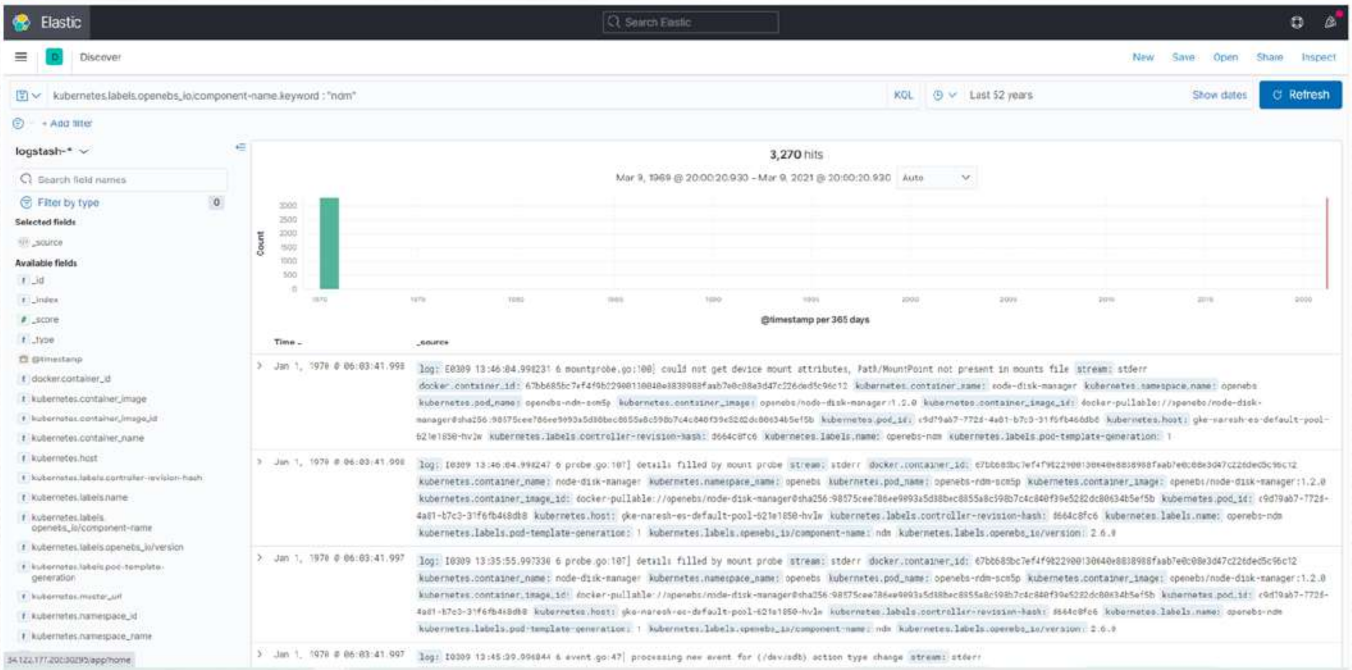
There will be a dropdown menu where you can select the available indices. In this case, select the indices created in the above step.

In this case, you have to select `logstash-*` from the dropdown menu.

continued to the next page

Now let's do some tests: If you want to get the logs of NDM pods, type the following text inside the `Filters` field.
`kubernetes.labels.openebs_io/component-name.keyword : "ndm"`
and then choose the required date and time period. After that, click `Apply`.

You will see the OpenEBS NDM pod logs listed on the page.



kubernetes.labels.openebs_io/component-name.keyword : "ndm"

```
> Jan 1, 1970 @ 06:03:41.998  log: E0309 13:46:04.998231 6 mountprobe.go:100] could not get device mount attributes, Path/MountPoint not present in mounts file stream: stderr
                              docker.container_id: 67bb685bc7ef4f9b22900130840e8838988faab7e0c08e3d47c226ded5c96c12 kubernetes.container_name: node-disk-manager kubernetes.namespace_name: openebs
                              kubernetes.pod_name: openebs-ndm-scm5p kubernetes.container_image: openebs/node-disk-manager:1.2.0 kubernetes.container_image_id: docker-pullable://openebs/node-disk-
                              manager@sha256:98575cee786ee9993a5d38bec8855a8c598b7c4c840f39e5282dc80634b5ef5b kubernetes.pod_id: c9d79ab7-772d-4a81-b7c3-31f6fb468db8 kubernetes.host: gke-naresh-es-default-pool-
                              621e1850-hvlw kubernetes.labels.controller-revision-hash: d664c8fc6 kubernetes.labels.name: openebs-ndm kubernetes.labels.pod-template-generation: 1
> Jan 1, 1970 @ 06:03:41.998  log: I0309 13:46:04.998247 6 probe.go:107] details filled by mount probe stream: stderr docker.container_id: 67bb685bc7ef4f9b22900130840e8838988faab7e0c08e3d47c226ded5c96c12
                              kubernetes.container_name: node-disk-manager kubernetes.namespace_name: openebs kubernetes.pod_name: openebs-ndm-scm5p kubernetes.container_image: openebs/node-disk-manager:1.2.0
                              kubernetes.container_image_id: docker-pullable://openebs/node-disk-manager@sha256:98575cee786ee9993a5d38bec8855a8c598b7c4c840f39e5282dc80634b5ef5b kubernetes.pod_id: c9d79ab7-772d-
                              4a81-b7c3-31f6fb468db8 kubernetes.host: gke-naresh-es-default-pool-621e1850-hvlw kubernetes.labels.controller-revision-hash: d664c8fc6 kubernetes.labels.name: openebs-ndm
                              kubernetes.labels.pod-template-generation: 1 kubernetes.labels.openebs_io/component-name: ndm kubernetes.labels.openebs_io/version: 2.6.0
> Jan 1, 1970 @ 06:03:41.997  log: I0309 13:35:55.997330 6 probe.go:107] details filled by mount probe stream: stderr docker.container_id: 67bb685bc7ef4f9b22900130840e8838988faab7e0c08e3d47c226ded5c96c12
                              kubernetes.container_name: node-disk-manager kubernetes.namespace_name: openebs kubernetes.pod_name: openebs-ndm-scm5p kubernetes.container_image: openebs/node-disk-manager:1.2.0
                              kubernetes.container_image_id: docker-pullable://openebs/node-disk-manager@sha256:98575cee786ee9993a5d38bec8855a8c598b7c4c840f39e5282dc80634b5ef5b kubernetes.pod_id: c9d79ab7-772d-
                              4a81-b7c3-31f6fb468db8 kubernetes.host: gke-naresh-es-default-pool-621e1850-hvlw kubernetes.labels.controller-revision-hash: d664c8fc6 kubernetes.labels.name: openebs-ndm
                              kubernetes.labels.pod-template-generation: 1 kubernetes.labels.openebs_io/component-name: ndm kubernetes.labels.openebs_io/version: 2.6.0
> Jan 1, 1970 @ 06:03:41.997  log: I0309 13:45:39.996844 6 event.go:47] processing new event for (/dev/sdb) action type change stream: stderr
```

Monitoring ElasticSearch

Set up Prometheus and Grafana

In this section, we will install Prometheus Operator and use ElasticSearch Service Monitor to add a ElasticSearch Dashboard to Grafana.

Installing Prometheus Operator

Installation of the Prometheus operator using Helm, will install both Prometheus and Grafana. Download Prometheus operator using Helm v3.

```
$ helm repo add prometheus-community https://prometheus-
community.github.io/helm-charts
$ kubectl create ns monitoring
$ helm repo update
```

This following command will install both Prometheus and Grafana components.

```
$ helm install prometheus prometheus-community/kube-prometheus-stack --
namespace monitoring
```

Note: Check compatibility for your Kubernetes version and Prometheus stack from here.

Verify if Prometheus related pods are installed successfully under monitoring namespace:

```
$ kubectl get pods -n monitoring
NAME                                                      READY   STATUS    RESTARTS   AGE
alertmanager-prometheus-kube-prometheus-alertmanager-0   2/2     Running   0
2m17s
prometheus-grafana-7666764f44-v58cp                      2/2     Running   0          2m21s
prometheus-kube-prometheus-operator-8bfdd5bcf-7dt98      1/1     Running   0
2m21s
prometheus-kube-state-metrics-6bfcd6f648-mw48q           1/1     Running   0
2m21s
prometheus-prometheus-kube-prometheus-prometheus-0       2/2     Running   1
2m16s
prometheus-prometheus-node-exporter-fxhdj                1/1     Running   0          2m21s
prometheus-prometheus-node-exporter-mn72f                1/1     Running   0          2m21s
prometheus-prometheus-node-exporter-v782r                1/1     Running   0          2m21s
```

Verify if Prometheus related services are installed successfully under monitoring namespace:

```
$ kubectl get svc -n monitoring
NAME                                       TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)
AGE
alertmanager-operated                      ClusterIP   None            <none>
9093/TCP,9094/TCP,9094/UDP   2m50s
prometheus-grafana                         ClusterIP   10.112.23.251   <none>        80/TCP
2m54s
prometheus-kube-prometheus-alertmanager   ClusterIP   10.112.25.4     <none>
9093/TCP                     2m54s
prometheus-kube-prometheus-operator        ClusterIP   10.112.17.237   <none>
443/TCP                      2m54s
prometheus-kube-prometheus-prometheus      ClusterIP   10.112.16.119   <none>
9090/TCP                     2m54s
prometheus-kube-state-metrics              ClusterIP   10.112.23.8     <none>        8080/TCP
2m54s
prometheus-operated                        ClusterIP   None            <none>        9090/TCP
2m49s
prometheus-prometheus-node-exporter        ClusterIP   10.112.22.35    <none>
9100/TCP                     2m54s
```

Change `prometheus-prometheus-oper-prometheus` service to LoadBalancer/NodePort from ClusterIP. This change is for accessing Prometheus service from your Web browser.

```
$ kubectl patch svc prometheus-kube-prometheus-prometheus -n monitoring -p
'{"spec": {"type": "NodePort"}}'
```

Change `prometheus-grafana` service to LoadBalancer/NodePort from ClusterIP. This change is for accessing Grafana service from your Web browser

```
$ kubectl patch svc prometheus-grafana -n monitoring -p '{"spec": {"type":
"NodePort"}}'
```

Note: If the user needs to access Prometheus and Grafana outside the network, the service type can be changed or a new service should be added to use LoadBalancer or create Ingress resources for production deployment.

For ease of simplicity in testing the deployment, we are going to use NodePort. Please be advised to consider using LoadBalancer or Ingress, instead of NodePort, for production deployment.

Sample output after changing above 2 services:

```
$ kubectl get svc -n monitoring
NAME                                    TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)
AGE
alertmanager-operated                   ClusterIP   None            <none>
9093/TCP,9094/TCP,9094/UDP   5m19s
prometheus-grafana                      NodePort    10.112.23.251   <none>
80:30029/TCP                 5m23s
prometheus-kube-prometheus-alertmanager   ClusterIP   10.112.25.4     <none>
9093/TCP                     5m23s
prometheus-kube-prometheus-operator     ClusterIP   10.112.17.237   <none>
443/TCP                      5m23s
prometheus-kube-prometheus-prometheus   NodePort    10.112.16.119   <none>
9090:31784/TCP               5m23s
prometheus-kube-state-metrics           ClusterIP   10.112.23.8     <none>        8080/TCP
5m23s
prometheus-operated                     ClusterIP   None            <none>        9090/TCP
5m18s
prometheus-prometheus-node-exporter     ClusterIP   10.112.22.35    <none>
9100/TCP                     5m23s
```

Sample output details of Nodes:

```
$ kubectl get node -o wide
NAME                                    STATUS   ROLES    AGE     VERSION            INTERNAL-IP
EXTERNAL-IP     OS-IMAGE            KERNEL-VERSION   CONTAINER-RUNTIME
gke-cluster-2-default-pool-c8c74720-65nf   Ready    <none>   5h48m   v1.18.12-gke.1210
10.128.15.193   34.70.58.139     Ubuntu 18.04.5 LTS   5.4.0-1030-gke   docker://19.3.6
gke-cluster-2-default-pool-c8c74720-90r9   Ready    <none>   5h48m   v1.18.12-gke.1210
10.128.15.194   35.188.38.187    Ubuntu 18.04.5 LTS   5.4.0-1030-gke   docker://19.3.6
gke-cluster-2-default-pool-c8c74720-cjjc   Ready    <none>   5h48m   v1.18.12-gke.1210
10.128.15.196   35.224.255.199   Ubuntu 18.04.5 LTS   5.4.0-1030-gke   docker://19.3.6
```

Verify if Prometheus service is accessible over the web using `<any_node_external-ip>:<NodePort>` if Service Type of Prometheus is NodePort.

Example:

```
http://34.70.58.139:31784
```

Installing ElasticSearch service monitor

This is how your **`servicemonitor.yaml`** should look like:

```
$ cat servicemonitor.yaml
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  labels:
    release: prometheus
  name: elastic-cluster-monitor
spec:
  endpoints:
  - interval: 10s
    path: /metrics
    targetPort: 9114
  namespaceSelector:
    matchNames:
    - default
  selector:
    matchLabels:
      app: elasticsearch-exporter
```

Create a Service Monitor:

```
$ kubectl apply -f servicemonitor.yaml
```

Deploy ElasticSearch exporter to get the metrics:

Get the values.yaml:

```
$ wget https://raw.githubusercontent.com/prometheus-community/helm-charts/main/charts/prometheus-elasticsearch-exporter/values.yaml
```

Edit the following parameters in values.yaml:

```
service.httpPort: 9114
es.uri:  http://elastic-coordinator-hs:9200   # Provide the service name of Elastic coordinator
es.cluster_settings: true
serviceAccount.create: true
podSecurityPolicies.enabled:true
```

Now, install Elastic Search exporter:

```
$ helm install es-monitor -f values.yaml prometheus-community/prometheus-elasticsearch-exporter
```

Launch Grafana using External IP of prometheus-grafana with port 80 on your browser, similar to the format here http://:<80>. This is applicable if the service is being created using Load Balancer. If it is NodePort, then use :< Nodeport of prometheus-grafana>.

Example:

```
http://34.67.160.246:30029
```

Username: admin Password: prom-operator

Password can be get using the command

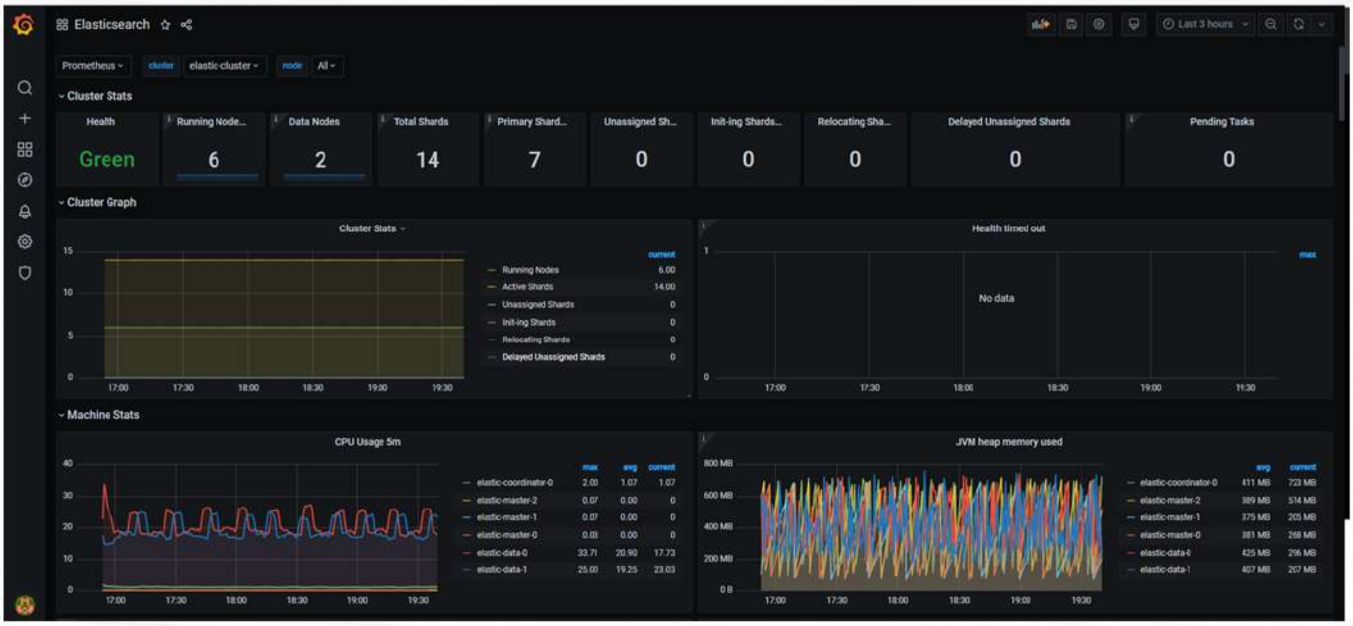
```
(kubectl get secret \
   --namespace monitoring prometheus-grafana \
   -o jsonpath="{.data.admin-password}" \
   | base64 --decode ; echo
)
```

Adding ElasticSearch Dashboard

Meanwhile, either you can download the grafana json file for Elasticsearch from the following link and upload into Grafana or you can mention Apache grafana dashboard id 7259 in Import via grafana.com section.

```
https://grafana.com/api/dashboards/7259/revisions/1/download
```

If you download the dashboard, then in your Grafana Console, goto + sign and click Import. Then, use the downloaded elasticsearch_rev1.json file into the form and click on Load. In the next page, select Prometheus as a data source against the Prometheus data field and then import it. Now, you can see the Monitoring dashboard of Elasticsearch workload.

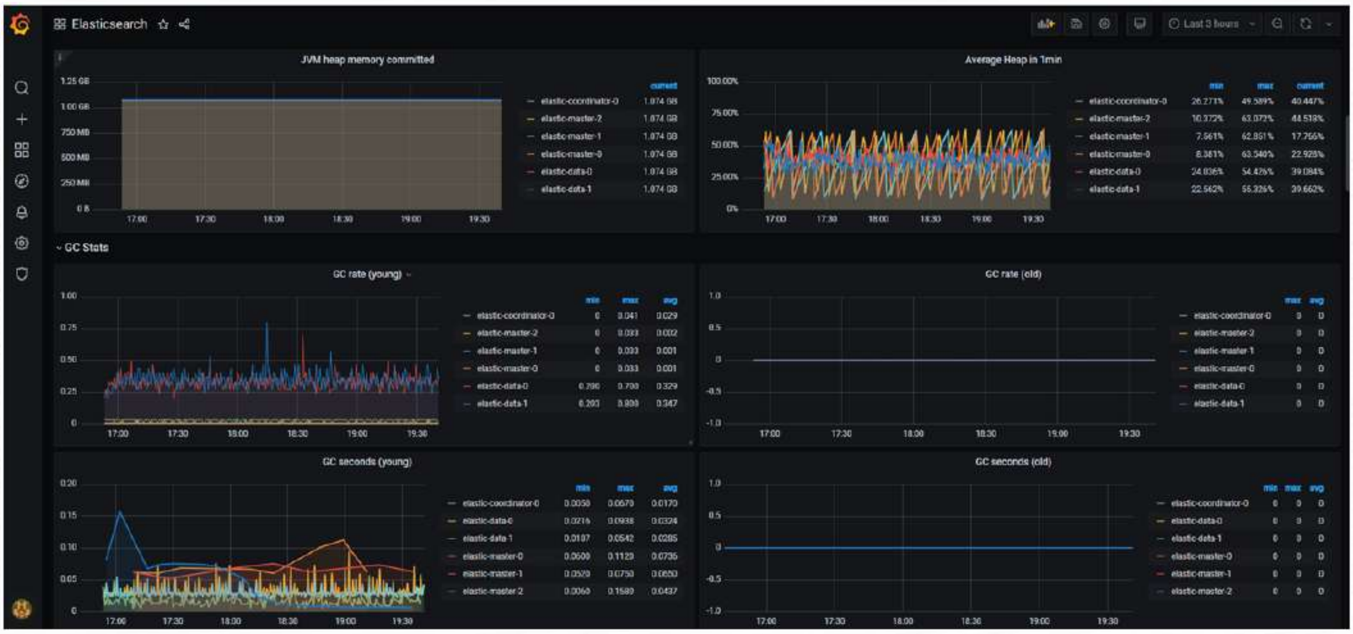


Health	Running Node...	Data Nodes	Total Shards	Primary Shard...	Unassigned Sh...	Init-ing Shards...	Relocating Sha...	Delayed Unassigned Shards	Pending Tasks
Green	6	2	14	7	0	0	0	0	0

	current
Running Nodes	6.00
Active Shards	14.00
Unassigned Shards	0
Init-ing Shards	0
Relocating Shards	0
Delayed Unassigned Shards	0

	max	avg	current
elastic-coordinator-0	2.00	1.07	1.07
elastic-master-2	0.07	0.00	0
elastic-master-1	0.07	0.00	0
elastic-master-0	0.03	0.00	0
elastic-data-0	33.71	20.90	17.73
elastic-data-1	25.00	19.25	23.03

	avg	current
elastic-coordinator-0	411 MB	723 MB
elastic-master-2	389 MB	514 MB
elastic-master-1	375 MB	205 MB
elastic-master-0	381 MB	268 MB
elastic-data-0	425 MB	296 MB
elastic-data-1	407 MB	207 MB



JVM heap memory committed

	current
elastic-coordinator-0	1.074 GB
elastic-master-2	1.074 GB
elastic-master-1	1.074 GB
elastic-master-0	1.074 GB
elastic-data-0	1.074 GB
elastic-data-1	1.074 GB

Average Heap in 1min

	min	max	current
elastic-coordinator-0	26.271%	49.589%	40.447%
elastic-master-2	10.372%	63.072%	44.518%
elastic-master-1	7.561%	62.851%	17.766%
elastic-master-0	8.381%	63.540%	22.928%
elastic-data-0	24.036%	54.426%	39.084%
elastic-data-1	22.562%	55.326%	39.662%

GC rate (young)

	min	max	avg
elastic-coordinator-0	0	0.041	0.029
elastic-master-2	0	0.033	0.002
elastic-master-1	0	0.033	0.001
elastic-master-0	0	0.033	0.001
elastic-data-0	0.290	0.700	0.329
elastic-data-1	0.203	0.800	0.347

GC rate (old)

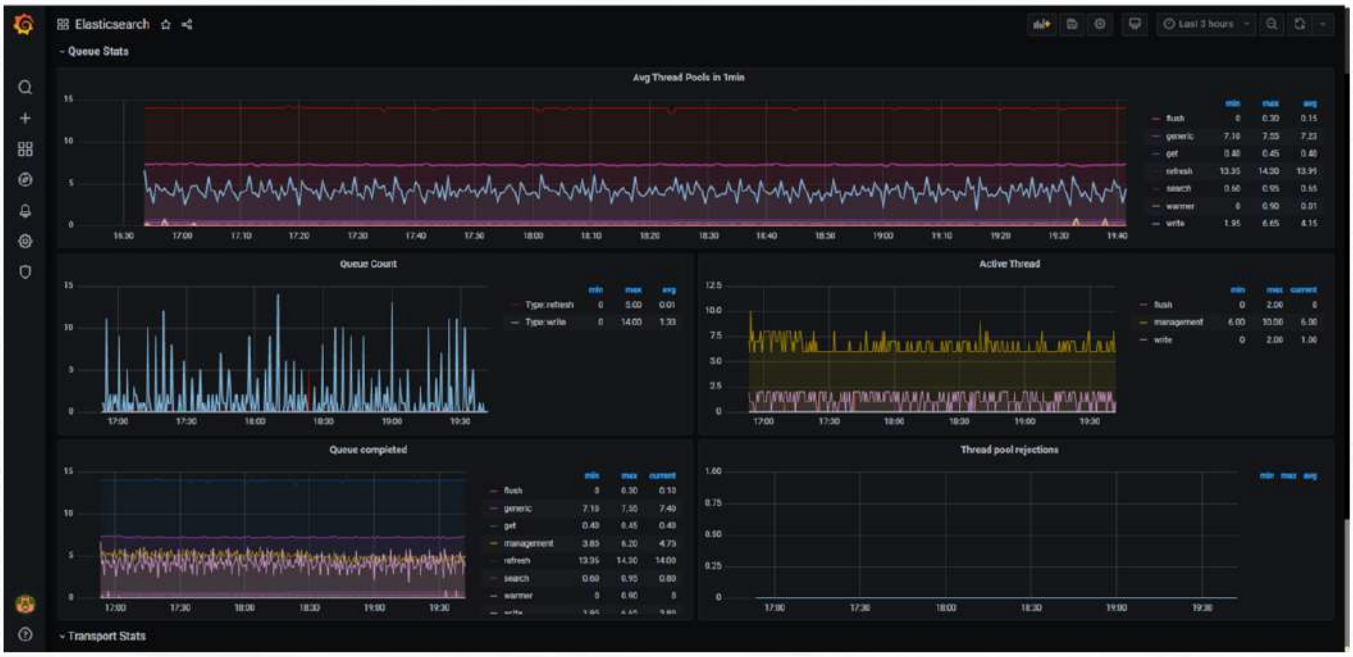
	max	avg
elastic-coordinator-0	0	0
elastic-master-2	0	0
elastic-master-1	0	0
elastic-master-0	0	0
elastic-data-0	0	0
elastic-data-1	0	0

GC seconds (young)

	min	max	avg
elastic-coordinator-0	0.0050	0.0670	0.0170
elastic-data-0	0.0216	0.0938	0.0324
elastic-data-1	0.0107	0.0542	0.0285
elastic-master-0	0.0600	0.1129	0.0735
elastic-master-1	0.0520	0.0750	0.0650
elastic-master-2	0.0060	0.1580	0.0437

GC seconds (old)

	min	max	avg
elastic-coordinator-0	0	0	0
elastic-data-0	0	0	0
elastic-data-1	0	0	0
elastic-master-0	0	0	0
elastic-master-1	0	0	0
elastic-master-2	0	0	0



Avg Thread Pools in 1min

	min	max	avg
flush	0	0.30	0.15
generic	7.10	7.55	7.23
get	0.40	0.45	0.40
refresh	13.35	14.30	13.91
search	0.60	0.95	0.66
warmer	0	0.90	0.01
write	1.95	6.65	4.15

Queue Count

	min	max	avg
Type: refresh	0	5.00	0.01
Type: write	0	14.00	1.33

Active Thread

	min	max	current
flush	0	2.00	0
management	6.00	10.00	6.00
write	0	2.00	1.00

Queue completed

	min	max	current
flush	0	0.30	0.10
generic	7.10	7.50	7.40
get	0.40	0.45	0.40
management	3.85	6.20	4.75
refresh	13.35	14.30	14.00
search	0.60	0.95	0.80
warmer	0	0.90	0
write	[illegible]	[illegible]	[illegible]

Thread pool rejections

min	max	avg

CONCLUSION

Elasticsearch provides distributed, scalable, multitenant-capable full text search engines for all kinds of documents. In this tutorial, we have deployed EFK stack using openEBS Local PV devices storage engine to provide a persistent storage solution for Elasticsearch, Fluentd and Kibana in Kubernetes environment. We used Kubera to deploy openEBS on the k8 cluster. We showed how to check metrics and monitoring of elasticsearch instances using Prometheus and Grafana.

www.mayadata.io

MayaData

linkedin.com/company/mayadata/

twitter.com/MayaData

blog.mayadata.io/