This document belongs to MicroK8s, if you are using another Kubernetes cluster, you must make slight changes to the manifest files provided in this documentation.

Enable following modules:

```
microk8s enable dns
microk8s enable hostpath-storage
microk8s enable ingress
microk8s enable metrics-server
```

Create namespace:

vi name-space-kube-logging-space.yml

```
apiVersion: v1
kind: Namespace
metadata:
  name: kube-logging-space
```

This will create a namespace called `kube-logging-space`. This namespace will be used to organize and control access to the resources that are deployed in it.

Create Service file for elastic search:

vi es-service.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: elasticsearch-logging
  namespace: kube-logging-space
  labels:
    k8s-app: elasticsearch-logging
    kubernetes.io/cluster-service: "true"
    addonmanager.kubernetes.io/mode: Reconcile
    kubernetes.io/name: "Elasticsearch"
spec:
  ports:
    - name: db
      port: 9200
      protocol: TCP
      targetPort: 9200
    - name: transport
      port: 9300
      protocol: TCP
      targetPort: 9300
  publishNotReadyAddresses: true
  selector:
    k8s-app: elasticsearch-logging
  sessionAffinity: None
  type: NodePort
```

- The name of the service is `elasticsearch-logging`.
- The namespace is `kube-logging-space`.
- The service has two ports: 9200 and 9300. Port 9200 is the HTTP port for Elasticsearch, and port 9300 is the transport port.
- The service will publish the addresses of pods that are not yet ready, so that clients can connect to them even if they are not yet fully initialized.
- `labels`: Labels are key-value pairs used to identify and categorize resources. In this Service, several labels are defined, such as `k8s-app`, `kubernetes.io/cluster-service`, `addonmanager.kubernetes.io/mode`, and `kubernetes.io/name`. These labels can be used for selecting and filtering resources.
- `sessionAffinity`: It's set to `None`, which means that the Service doesn't maintain session affinity. Each incoming request can be routed to any available Pod without regard to previous requests from the same client.
- `type`: This defines the Service type. It's set to `NodePort`, which means that the Service will be accessible on each node's IP address at a specific port (randomly assigned by Kubernetes) in addition to its Cluster IP.

Create Deployment file for elastic search:

vi es-statefulset.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: elasticsearch-logging
  namespace: kube-logging-space
  labels:
    k8s-app: elasticsearch-logging
```

A ServiceAccount is a way to identify and authenticate pods. It allows pods to access Kubernetes resources without having to use a user account.

The `apiVersion` field specifies the version of the Kubernetes API that the ServiceAccount definition is using. In this case, it is `v1`.

The `kind` field specifies the type of Kubernetes resource that the definition is creating. In this case, it is a ServiceAccount.

The `metadata` field contains the name and labels for the ServiceAccount. In this case, the name is `elasticsearch-logging` and the labels are `k8s-app: elasticsearch-logging` and `addonmanager.kubernetes.io/mode: Reconcile`.

The `spec` field is empty, which means that there are no specific configuration options for this ServiceAccount.

```
---
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: elasticsearch-logging
  labels:
    k8s-app: elasticsearch-logging
    addonmanager.kubernetes.io/mode: Reconcile
rules:
  - apiGroups:
      - ""
    resources:
      - "services"
      - "namespaces"
      - "endpoints"
    verbs:
      - "get"
```

A ClusterRole is a way to grant permissions to a group of users or service accounts.

The `apiVersion` field specifies the version of the Kubernetes API that the ClusterRole definition is using. In this case, it is `rbac.authorization.k8s.io/v1`.

The `kind` field specifies the type of Kubernetes resource that the definition is creating. In this case, it is a ClusterRole.

The `metadata` field contains the name and labels for the ClusterRole. In this case, the name is `elasticsearch-logging` and the labels are `k8s-app: elasticsearch-logging` and `addonmanager.kubernetes.io/mode: Reconcile`.

The `rules` field specifies the permissions that are granted to the ClusterRole. In this case, the ClusterRole is granted the permission to `get` services, namespaces, and endpoints.

```
---
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  namespace: kube-logging-space
  name: elasticsearch-logging
  labels:
    k8s-app: elasticsearch-logging
    addonmanager.kubernetes.io/mode: Reconcile
subjects:
  - kind: ServiceAccount
    name: elasticsearch-logging
    namespace: kube-logging-space
    apiGroup: ""
roleRef:
  kind: ClusterRole
  name: elasticsearch-logging
  apiGroup: ""
```

A ClusterRoleBinding is a way to bind a ClusterRole to a group of users or service accounts.

The `apiVersion` field specifies the version of the Kubernetes API that the ClusterRoleBinding definition is using. In this case, it is `rbac.authorization.k8s.io/v1`.

The `kind` field specifies the type of Kubernetes resource that the definition is creating. In this case, it is a ClusterRoleBinding.

The `metadata` field contains the name and labels for the ClusterRoleBinding. In this case, the name is `elasticsearch-logging` and the labels are `k8s-app: elasticsearch-logging` and `addonmanager.kubernetes.io/mode: Reconcile`.

The `subjects` field specifies the subjects that are bound to the ClusterRole. In this case, the subject is the `elasticsearch-logging` ServiceAccount in the `kube-logging-space` namespace.

The `roleRef` field specifies the ClusterRole that is bound to the subjects. In this case, the ClusterRole is the `elasticsearch-logging` ClusterRole.

```
---
# Elasticsearch deployment itself
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: elasticsearch-logging
  namespace: kube-logging-space
  labels:
    k8s-app: elasticsearch-logging
    version: v7.4.3
    addonmanager.kubernetes.io/mode: Reconcile
spec:
  serviceName: elasticsearch-logging
  replicas: 1
  selector:
    matchLabels:
      k8s-app: elasticsearch-logging
      version: v7.4.3
  template:
    metadata:
      labels:
        k8s-app: elasticsearch-logging
        version: v7.4.3
    spec:
      serviceAccountName: elasticsearch-logging
      containers:
        - image: quay.io/fluentd_elasticsearch/elasticsearch:v7.10.2
          name: elasticsearch-logging
          imagePullPolicy: Always
          resources:
            # need more cpu upon initialization, therefore burstable class
            limits:
              cpu: 1000m
              memory: 3Gi
            requests:
              cpu: 100m
              memory: 3Gi
          ports:
            - containerPort: 9200
              name: db
              protocol: TCP
            - containerPort: 9300
              name: transport
              protocol: TCP
          livenessProbe:
            tcpSocket:
```

```
            port: transport
          initialDelaySeconds: 5
          timeoutSeconds: 20
          failureThreshold: 10
        readinessProbe:
          tcpSocket:
            port: transport
          initialDelaySeconds: 5
          timeoutSeconds: 20
          failureThreshold: 10
        volumeMounts:
          - name: elasticsearch-logging
            mountPath: /data
        env:
          - name: ES_JAVA_OPTS
            value: "-Dlog4j2.formatMsgNoLookups=true"
          - name: "NAMESPACE"
            valueFrom:
              fieldRef:
                fieldPath: metadata.namespace
          - name: "MINIMUM_MASTER_NODES"
            value: "1"
      volumes:
        - name: elasticsearch-logging
          hostPath:
            path: /home/vishal/es-data    #path on host for storing data of es
      # Elasticsearch requires vm.max_map_count to be at least 262144.
      # If your OS already sets up this number to a higher value, feel free
      # to remove this init container.
      initContainers:
        - image: alpine:3.6
          command: ["/sbin/sysctl", "-w", "vm.max_map_count=262144"]
          name: elasticsearch-logging-init
          securityContext:
            privileged: true
```

A StatefulSet is a Kubernetes resource that manages the deployment and scaling of a set of Pods that are meant to be identical.

StatefulSets are used for applications that require their Pods to have unique identities and persistent storage. For example, a StatefulSet could be used to deploy a cluster of database servers, where each server has a unique name and its own persistent storage volume.

I m using stateful set here because I wish to have unique pod name for it

The `kind` field specifies the type of Kubernetes resource that the definition is creating. In this case, it is a StatefulSet.

The `metadata` field contains the name, namespace, and labels for the StatefulSet. In this case, the name is `elasticsearch-logging`, the namespace is `kube-logging-space`, and the

labels are `k8s-app: elasticsearch-logging`, `version: v7.4.3`, and `addonmanager.kubernetes.io/mode: Reconcile`.

The `spec` field specifies the configuration for the StatefulSet.

- **serviceName:** The name of the service that the StatefulSet will create. In this case, the name is `elasticsearch-logging`.
- **replicas:** The number of replicas that the StatefulSet will create. In this case, the number of replicas is 1.
- **selector:** The selector that is used to match the Pods that are managed by the StatefulSet. In this case, the selector matches Pods with the labels `k8s-app: elasticsearch-logging` and `version: v7.4.3`.
- **template:** The template that is used to create the Pods that are managed by the StatefulSet.
- **metadata:** The metadata that is applied to the Pods that are created by the template.
- **spec:** The spec that is applied to the Pods that are created by the template.
    - **serviceAccountName:** The name of the ServiceAccount that is used by the Pods. In this case, the name is `elasticsearch-logging`.

containers: The list of containers that are run in the Pods.

- **image:** The image that is used to create the container. In this case, the image is `quay.io/fluentd_elasticsearch/elasticsearch:v7.10.2`.
- **name:** The name of the container. In this case, the name is `elasticsearch-logging`.
- **imagePullPolicy:** The policy for pulling the image. In this case, the policy is `Always`, which means that the image will always be pulled from the registry, even if it is already present in the local cache.
- **resources:** The CPU and memory resources that the container is allowed to use. In this case, the container is allowed to use up to 1000m of CPU and 3Gi of memory.
- **ports:** The ports that the container exposes. In this case, the container exposes ports 9200 and 9300.
- **livenessProbe:** A probe that is used to check if the container is alive. The probe connects to port 9300 and expects to receive a response within 20 seconds. If the probe fails 10 times, the container will be restarted.
- **readinessProbe:** A probe that is used to check if the container is ready to serve requests. The probe connects to port 9300 and expects to receive a response within 20 seconds. If the probe fails 10 times, the container will be removed from the load balancer.
- **volumeMounts:** The list of volumes that are mounted into the container. In this case, the container mounts the `elasticsearch-logging` volume at `/data`.

```
[root@elasticsearch-logging-0 elasticsearch]# grep -rnw -e  "/data"
config/elasticsearch.yml:10:path.data: /data
bin/run.sh:26:chown -R elasticsearch:elasticsearch /data
[root@elasticsearch-logging-0 elasticsearch]#
```

- **volumes:** The list of volumes that are used by the Pods.
  - **name:** The name of the volume. In this case, the name is `elasticsearch-logging`.
  - **hostPath:** The hostPath volume type. In this case, the hostPath volume type is used to mount a volume from the host node to the Pod.
    - **path:** The path on the host node where the volume is located. In this case, the path is `/home/vishal/es-data`.

The `initContainers` section defines an init container that is run before the Pods are created. This init container sets the `vm.max_map_count` sysctl to 262144. This is necessary for Elasticsearch to function properly and to avoid java errors.

The `privileged` field is set to `true`, which means that the init container will have all the privileges of the host node. This will help us to collect logs that needs root privileges to access host files.

Here are some of the things that a privileged container can do:

- Mount any filesystem, including the root filesystem.
- Access devices, such as network interfaces and block devices.
- Run any command, including root-privileged commands.

Whole es-statefulset.yaml looks like below,

vi es-statefulset.yaml

```
# RBAC authn and authz
apiVersion: v1
kind: ServiceAccount
metadata:
  name: elasticsearch-logging
  namespace: kube-logging-space
  labels:
    k8s-app: elasticsearch-logging
    addonmanager.kubernetes.io/mode: Reconcile
---
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: elasticsearch-logging
  labels:
    k8s-app: elasticsearch-logging
    addonmanager.kubernetes.io/mode: Reconcile
rules:
  - apiGroups:
      - ""
    resources:
      - "services"
      - "namespaces"
      - "endpoints"
    verbs:
      - "get"
---
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  namespace: kube-logging-space
  name: elasticsearch-logging
  labels:
    k8s-app: elasticsearch-logging
    addonmanager.kubernetes.io/mode: Reconcile
subjects:
  - kind: ServiceAccount
    name: elasticsearch-logging
    namespace: kube-logging-space
    apiGroup: ""
roleRef:
  kind: ClusterRole
  name: elasticsearch-logging
  apiGroup: ""
---
# Elasticsearch deployment itself
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: elasticsearch-logging
```

```
  namespace: kube-logging-space
  labels:
    k8s-app: elasticsearch-logging
    version: v7.4.3
    addonmanager.kubernetes.io/mode: Reconcile
spec:
  serviceName: elasticsearch-logging
  replicas: 1
  selector:
    matchLabels:
      k8s-app: elasticsearch-logging
      version: v7.4.3
  template:
    metadata:
      labels:
        k8s-app: elasticsearch-logging
        version: v7.4.3
    spec:
      serviceAccountName: elasticsearch-logging
      containers:
        - image: quay.io/fluentd_elasticsearch/elasticsearch:v7.10.2
          name: elasticsearch-logging
          imagePullPolicy: Always
          resources:
            # need more cpu upon initialization, therefore burstable class
            limits:
              cpu: 1000m
              memory: 3Gi
            requests:
              cpu: 100m
              memory: 3Gi
          ports:
            - containerPort: 9200
              name: db
              protocol: TCP
            - containerPort: 9300
              name: transport
              protocol: TCP
          livenessProbe:
            tcpSocket:
              port: transport
            initialDelaySeconds: 5
            timeoutSeconds: 20
            failureThreshold: 10
          readinessProbe:
            tcpSocket:
              port: transport
            initialDelaySeconds: 5
            timeoutSeconds: 20
            failureThreshold: 10
          volumeMounts:
            - name: elasticsearch-logging
              mountPath: /data
          env:
            - name: ES_JAVA_OPTS
              value: "-Dlog4j2.formatMsgNoLookups=true"
            - name: "NAMESPACE"
```

```
            valueFrom:
              fieldRef:
                fieldPath: metadata.namespace
          - name: "MINIMUM_MASTER_NODES"
            value: "1"
      volumes:
        - name: elasticsearch-logging
          hostPath:
            path: /home/vishal/es-data    #path on host for storing data of es
      # Elasticsearch requires vm.max_map_count to be at least 262144.
      # If your OS already sets up this number to a higher value, feel free
      # to remove this init container.
      initContainers:
        - image: alpine:3.6
          command: ["/sbin/sysctl", "-w", "vm.max_map_count=262144"]
          name: elasticsearch-logging-init
          securityContext:
            privileged: true
```

Apply manifest files:

```
kubectl apply -f name-space-kube-logging-space.yml
kubectl apply -f es-service.yaml
kubectl apply -f es-statefulset.yaml
```

`vi kibana-service.yaml`

```
apiVersion: v1
kind: Service
metadata:
  name: kibana-logging
  namespace: kube-logging-space
  labels:
    k8s-app: kibana-logging
    kubernetes.io/cluster-service: "true"
    addonmanager.kubernetes.io/mode: Reconcile
    kubernetes.io/name: "Kibana"
spec:
  ports:
    - port: 5601
      protocol: TCP
      targetPort: ui
  selector:
    k8s-app: kibana-logging
```

The `kind` field specifies the type of Kubernetes resource that the definition is creating. In this case, it is a Service.

The `metadata` field contains the name, namespace, and labels for the Service. In this case, the name is `kibana-logging`, the namespace is `kube-logging-space`, and the labels are `k8s-app: kibana-logging`, `kubernetes.io/cluster-service: "true"`, `addonmanager.kubernetes.io/mode: Reconcile`, and `kubernetes.io/name: "Kibana"`.

The `spec` field specifies the configuration for the Service.

- **ports:** The list of ports that the Service exposes. In this case, the Service exposes port 5601.
- **selector:** The selector that is used to match the Pods that are exposed by the Service. In this case, the selector matches Pods with the label `k8s-app: kibana-logging`.

The `ports` section specifies the ports that the Service exposes. In this case, the Service exposes port 5601. This is the port that the Kibana web interface is listening on.

The `selector` section specifies the selector that is used to match the Pods that are exposed by the Service. In this case, the selector matches Pods with the label `k8s-app: kibana-logging`. This means that the Service will only expose Pods that have this label.

vi kibana-deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: kibana-logging
  namespace: kube-logging-space
  labels:
    k8s-app: kibana-logging
    addonmanager.kubernetes.io/mode: Reconcile
spec:
  replicas: 1
  selector:
    matchLabels:
      k8s-app: kibana-logging
  template:
    metadata:
      labels:
        k8s-app: kibana-logging
    spec:
      securityContext:
        seccompProfile:
          type: RuntimeDefault
      containers:
        - name: kibana-logging
          image: docker.elastic.co/kibana/kibana-oss:7.10.2
          resources:
            # need more cpu upon initialization, therefore burstable class
            limits:
              cpu: 1000m
            requests:
              cpu: 100m
          env:
            - name: ELASTICSEARCH_HOSTS
              value: http://elasticsearch-logging:9200
            - name: SERVER_NAME
              value: kibana-logging
            - name: SERVER_REWRITEBASEPATH
              value: "false"
          ports:
            - containerPort: 5601
              name: ui
              protocol: TCP
          livenessProbe:
            httpGet:
              path: /api/status
              port: ui
            initialDelaySeconds: 5
            timeoutSeconds: 10
          readinessProbe:
            httpGet:
              path: /api/status
              port: ui
            initialDelaySeconds: 5
            timeoutSeconds: 10
```

The kind field specifies the type of Kubernetes resource that the definition is creating. In this case, it is a Deployment.

The metadata field contains the name, namespace, and labels for the Deployment. In this case, the name is kibana-logging, the namespace is kube-logging-space, and the labels are k8s-app: kibana-logging and addonmanager.kubernetes.io/mode: Reconcile.

The spec field specifies the configuration for the Deployment.

- **replicas:** The number of replicas that the Deployment will create. In this case, the number of replicas is 1.
- **selector:** The selector that is used to match the Pods that are managed by the Deployment. In this case, the selector matches Pods with the label k8s-app: kibana-logging.
- **template:** The template that is used to create the Pods that are managed by the Deployment.
    - **metadata:** The metadata that is applied to the Pods that are created by the template.
    - **spec:** The spec that is applied to the Pods that are created by the template.
        - **securityContext:** The security context for the container. The seccompProfile field is set to RuntimeDefault, which means that the container will use the default seccomp profile.

containers: The list of containers that are run in the Pods.

- **name:** The name of the container. In this case, the name is kibana-logging.
- **image:** The image that is used to create the container. In this case, the image is docker.elastic.co/kibana/kibana-oss:7.10.2.
- **resources:** The CPU and memory resources that the container is allowed to use. In this case, the container is allowed to use up to 1000m of CPU and 100m of memory.
- **env:** The environment variables that are set for the container. In this case, the environment variables ELASTICSEARCH_HOSTS, SERVER_NAME, and SERVER_REWRITEBASEPATH are set.
- **ports:** The ports that the container exposes. In this case, the container exposes port 5601.
- **livenessProbe:** A probe that is used to check if the container is alive. The probe connects to port 5601 and expects to receive a response within 10 seconds. If the probe fails 10 times, the container will be restarted.
- **readinessProbe:** A probe that is used to check if the container is ready to serve requests. The probe connects to port 5601 and expects to receive a response within 10 seconds. If the probe fails 10 times, the container will be removed from the load balancer.

Apply manifest files:

```
kubectl apply -f kibana-service.yaml
kubectl apply -f kibana-deployment.yaml
```

vi fluentd-es-configmap.yaml

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: fluentd-es-config-v0.2.0
  namespace: kube-logging-space
  labels:
    addonmanager.kubernetes.io/mode: Reconcile
data:
  system.conf: |-
    <system>
      root_dir /tmp/fluentd-buffers/
    </system>

  containers.input.conf: |-
    # This configuration file for Fluentd / td-agent is used
    # to watch changes to Docker log files. The kubelet creates symlinks that
    # capture the pod name, namespace, container name & Docker container ID
    # to the docker logs for pods in the /var/log/containers directory on the host.
    # If running this fluentd configuration in a Docker container, the /var/log
    # directory should be mounted in the container.
    #
    # These logs are then submitted to Elasticsearch which assumes the
    # installation of the fluent-plugin-elasticsearch & the
    # fluent-plugin-kubernetes_metadata_filter plugins.
    # See https://github.com/uken/fluent-plugin-elasticsearch &
    # https://github.com/fabric8io/fluent-plugin-kubernetes_metadata_filter for
    # more information about the plugins.
    #
    # Example
    # =======
    # A line in the Docker log file might look like this JSON:
    #
    # {"log":"2014/09/25 21:15:03 Got request with path wombat\n",
    #  "stream":"stderr",
    #   "time":"2014-09-25T21:15:03.499185026Z"}
    #
    # The time_format specification below makes sure we properly
    # parse the time format produced by Docker. This will be
    # submitted to Elasticsearch and should appear like:
    # $ curl 'http://elasticsearch-logging:9200/_search?pretty'
    # ...
    # {
    #      "_index" : "logstash-2014.09.25",
    #      "_type" : "fluentd",
    #      "_id" : "VBrbor2QTuGpsQyTCdfzqA",
    #      "_score" : 1.0,
    #      "_source":{"log":"2014/09/25 22:45:50 Got request with path wombat\n",
    #                 "stream":"stderr","tag":"docker.container.all",
    #                 "@timestamp":"2014-09-25T22:45:50+00:00"}
    #    },
    # ...
    #
    # The Kubernetes fluentd plugin is used to write the Kubernetes metadata to the
log
    # record & add labels to the log record if properly configured. This enables users
    # to filter & search logs on any metadata.
    # For example a Docker container's logs might be in the directory:
    #
```

```
    #
/var/lib/docker/containers/997599971ee6366d4a5920d25b79286ad45ff37a74494f262e3bc98d909
d0a7b
    #
    # and in the file:
    #
    #  997599971ee6366d4a5920d25b79286ad45ff37a74494f262e3bc98d909d0a7b-json.log
    #
    # where 997599971ee6... is the Docker ID of the running container.
    # The Kubernetes kubelet makes a symbolic link to this file on the host machine
    # in the /var/log/containers directory which includes the pod name and the
Kubernetes
    # container name:
    #
    #
synthetic-logger-0.25lps-pod_default_synth-lgr-997599971ee6366d4a5920d25b79286ad45ff37
a74494f262e3bc98d909d0a7b.log
    #    ->
    #
/var/lib/docker/containers/997599971ee6366d4a5920d25b79286ad45ff37a74494f262e3bc98d909
d0a7b/997599971ee6366d4a5920d25b79286ad45ff37a74494f262e3bc98d909d0a7b-json.log
    #
    # The /var/log directory on the host is mapped to the /var/log directory in the
container
    # running this instance of Fluentd and we end up collecting the file:
    #
    #
/var/log/containers/synthetic-logger-0.25lps-pod_default_synth-lgr-997599971ee6366d4a5
920d25b79286ad45ff37a74494f262e3bc98d909d0a7b.log
    #
    # This results in the tag:
    #
    #
var.log.containers.synthetic-logger-0.25lps-pod_default_synth-lgr-997599971ee6366d4a59
20d25b79286ad45ff37a74494f262e3bc98d909d0a7b.log
    #
    # The Kubernetes fluentd plugin is used to extract the namespace, pod name &
container name
    # which are added to the log message as a kubernetes field object & the Docker
container ID
    # is also added under the docker field object.
    # The final tag is:
    #
    #
kubernetes.var.log.containers.synthetic-logger-0.25lps-pod_default_synth-lgr-997599971
ee6366d4a5920d25b79286ad45ff37a74494f262e3bc98d909d0a7b.log
    #
    # And the final log record look like:
    #
    # {
    #   "log":"2014/09/25 21:15:03 Got request with path wombat\n",
    #   "stream":"stderr",
    #   "time":"2014-09-25T21:15:03.499185026Z",
    #   "kubernetes": {
    #     "namespace": "default",
    #     "pod_name": "synthetic-logger-0.25lps-pod",
    #     "container_name": "synth-lgr"
    #   },
    #   "docker": {
    #     "container_id":
"997599971ee6366d4a5920d25b79286ad45ff37a74494f262e3bc98d909d0a7b"
    #   }
    # }
```

```
    #
    # This makes it easier for users to search for logs by pod name or by
    # the name of the Kubernetes container regardless of how many times the
    # Kubernetes pod has been restarted (resulting in a several Docker container IDs).

    # Json Log Example:
    # {"log":"[info:2016-02-16T16:04:05.930-08:00] Some log text
here\n","stream":"stdout","time":"2016-02-17T00:04:05.931087621Z"}
    # CRI Log Example:
    # 2016-02-17T00:04:05.931087621Z stdout F [info:2016-02-16T16:04:05.930-08:00]
Some log text here
    <source>
      @id fluentd-containers.log
      @type tail
      path /var/log/containers/*.log
      pos_file /var/log/es-containers.log.pos
      tag raw.kubernetes.*
      read_from_head true
      <parse>
        @type multi_format
        <pattern>
          format json
          time_key time
          time_format %Y-%m-%dT%H:%M:%S.%NZ
        </pattern>
        <pattern>
          format /^(?<time>.+) (?<stream>stdout|stderr) [^ ]* (?<log>.*)$/
          time_format %Y-%m-%dT%H:%M:%S.%N%:z
        </pattern>
      </parse>
    </source>

    # Detect exceptions in the log output and forward them as one log entry.
    <match raw.kubernetes.**>
      @id raw.kubernetes
      @type detect_exceptions
      remove_tag_prefix raw
      message log
      stream stream
      multiline_flush_interval 5
      max_bytes 500000
      max_lines 1000
    </match>

    # Concatenate multi-line logs
    <filter **>
      @id filter_concat
      @type concat
      key message
      multiline_end_regexp /\n$/
      separator ""
    </filter>

    # Enriches records with Kubernetes metadata
    <filter kubernetes.**>
      @id filter_kubernetes_metadata
      @type kubernetes_metadata
    </filter>

    # Fixes json fields in Elasticsearch
    <filter kubernetes.**>
      @id filter_parser
      @type parser
```

```
      key_name log
      reserve_data true
      remove_key_name_field true
      <parse>
        @type multi_format
        <pattern>
          format json
        </pattern>
        <pattern>
          format none
        </pattern>
      </parse>
    </filter>

  system.input.conf: |-
    # Example:
    # 2015-12-21 23:17:22,066 [salt.state       ][INFO    ] Completed state
[net.ipv4.ip_forward] at time 23:17:22.066081
    <source>
      @id minion
      @type tail
      format /^(?<time>[^ ]* [^ ,]*)[^\[]*\[[^\]]*\]\[(?<severity>[^ \]]*) *\]
(?<message>.*)$/
      time_format %Y-%m-%d %H:%M:%S
      path /var/log/salt/minion
      pos_file /var/log/salt.pos
      tag salt
    </source>

    # Example:
    # Dec 21 23:17:22 gke-foo-1-1-4b5cbd14-node-4eoj startupscript: Finished running
startup script /var/run/google.startup.script
    <source>
      @id startupscript.log
      @type tail
      format syslog
      path /var/log/startupscript.log
      pos_file /var/log/es-startupscript.log.pos
      tag startupscript
    </source>

    # Examples:
    # time="2016-02-04T06:51:03.053580605Z" level=info msg="GET /containers/json"
    # time="2016-02-04T07:53:57.505612354Z" level=error msg="HTTP Error" err="No such
image: -f" statusCode=404
    # TODO(random-liu): Remove this after cri container runtime rolls out.
    <source>
      @id docker.log
      @type tail
      format /^time="(?<time>[^"]*)" level=(?<severity>[^ ]*) msg="(?<message>[^"]*)"(
err="(?<error>[^"]*)")?( statusCode=($<status_code>\d+))?/
      path /var/log/docker.log
      pos_file /var/log/es-docker.log.pos
      tag docker
    </source>

    # Example:
    # 2016/02/04 06:52:38 filePurge: successfully removed file
/var/etcd/data/member/wal/00000000000006d0-00000000010a23d1.wal
    <source>
      @id etcd.log
      @type tail
      # Not parsing this, because it doesn't have anything particularly useful to
```

```
      # parse out of it (like severities).
      format none
      path /var/log/etcd.log
      pos_file /var/log/es-etcd.log.pos
      tag etcd
    </source>

    # Multi-line parsing is required for all the kube logs because very large log
    # statements, such as those that include entire object bodies, get split into
    # multiple lines by glog.

    # Example:
    # I0204 07:32:30.020537    3368 server.go:1048] POST /stats/container/:
(13.972191ms) 200 [[Go-http-client/1.1] 10.244.1.3:40537]
    <source>
      @id kubelet.log
      @type tail
      format multiline
      multiline_flush_interval 5s
      format_firstline /^\w\d{4}/
      format1 /^(?<severity>\w)(?<time>\d{4} [^\s]*)\s+(?<pid>\d+)\s+(?<source>[^
\]]+)\] (?<message>.*)/
      time_format %m%d %H:%M:%S.%N
      path /var/log/kubelet.log
      pos_file /var/log/es-kubelet.log.pos
      tag kubelet
    </source>

    # Example:
    # I1118 21:26:53.975789       6 proxier.go:1096] Port "nodePort for
kube-system/default-http-backend:http" (:31429/tcp) was open before and is still
needed
    <source>
      @id kube-proxy.log
      @type tail
      format multiline
      multiline_flush_interval 5s
      format_firstline /^\w\d{4}/
      format1 /^(?<severity>\w)(?<time>\d{4} [^\s]*)\s+(?<pid>\d+)\s+(?<source>[^
\]]+)\] (?<message>.*)/
      time_format %m%d %H:%M:%S.%N
      path /var/log/kube-proxy.log
      pos_file /var/log/es-kube-proxy.log.pos
      tag kube-proxy
    </source>

    # Example:
    # I0204 07:00:19.604280       5 handlers.go:131] GET /api/v1/nodes: (1.624207ms)
200 [[kube-controller-manager/v1.1.3 (linux/amd64) kubernetes/6a81b50]
127.0.0.1:38266]
    <source>
      @id kube-apiserver.log
      @type tail
      format multiline
      multiline_flush_interval 5s
      format_firstline /^\w\d{4}/
      format1 /^(?<severity>\w)(?<time>\d{4} [^\s]*)\s+(?<pid>\d+)\s+(?<source>[^
\]]+)\] (?<message>.*)/
      time_format %m%d %H:%M:%S.%N
      path /var/log/kube-apiserver.log
      pos_file /var/log/es-kube-apiserver.log.pos
      tag kube-apiserver
    </source>
```

```
    # Example:
    # I0204 06:55:31.872680       5 servicecontroller.go:277] LB already exists and
doesn't need update for service kube-system/kube-ui
    <source>
      @id kube-controller-manager.log
      @type tail
      format multiline
      multiline_flush_interval 5s
      format_firstline /^\w\d{4}/
      format1 /^(?<severity>\w)(?<time>\d{4} [^\s]*)\s+(?<pid>\d+)\s+(?<source>[^
\]]+)\] (?<message>.*)/
      time_format %m%d %H:%M:%S.%N
      path /var/log/kube-controller-manager.log
      pos_file /var/log/es-kube-controller-manager.log.pos
      tag kube-controller-manager
    </source>

    # Example:
    # W0204 06:49:18.239674       7 reflector.go:245]
pkg/scheduler/factory/factory.go:193: watch of *api.Service ended with: 401: The event
in requested index is outdated and cleared (the requested history has been cleared
[2578313/2577886]) [2579312]
    <source>
      @id kube-scheduler.log
      @type tail
      format multiline
      multiline_flush_interval 5s
      format_firstline /^\w\d{4}/
      format1 /^(?<severity>\w)(?<time>\d{4} [^\s]*)\s+(?<pid>\d+)\s+(?<source>[^
\]]+)\] (?<message>.*)/
      time_format %m%d %H:%M:%S.%N
      path /var/log/kube-scheduler.log
      pos_file /var/log/es-kube-scheduler.log.pos
      tag kube-scheduler
    </source>

    # Example:
    # I0603 15:31:05.793605       6 cluster_manager.go:230] Reading config from path
/etc/gce.conf
    <source>
      @id glbc.log
      @type tail
      format multiline
      multiline_flush_interval 5s
      format_firstline /^\w\d{4}/
      format1 /^(?<severity>\w)(?<time>\d{4} [^\s]*)\s+(?<pid>\d+)\s+(?<source>[^
\]]+)\] (?<message>.*)/
      time_format %m%d %H:%M:%S.%N
      path /var/log/glbc.log
      pos_file /var/log/es-glbc.log.pos
      tag glbc
    </source>

    # Example:
    # I0603 15:31:05.793605       6 cluster_manager.go:230] Reading config from path
/etc/gce.conf
    <source>
      @id cluster-autoscaler.log
      @type tail
      format multiline
      multiline_flush_interval 5s
      format_firstline /^\w\d{4}/
```

```
      format1 /^(?<severity>\w)(?<time>\d{4} [^\s]*)\s+(?<pid>\d+)\s+(?<source>[^
\]]+)\] (?<message>.*)/
      time_format %m%d %H:%M:%S.%N
      path /var/log/cluster-autoscaler.log
      pos_file /var/log/es-cluster-autoscaler.log.pos
      tag cluster-autoscaler
    </source>

    # Logs from systemd-journal for interesting services.
    # TODO(random-liu): Remove this after cri container runtime rolls out.
    <source>
      @id journald-docker
      @type systemd
      matches [{ "_SYSTEMD_UNIT": "docker.service" }]
      <storage>
        @type local
        persistent true
        path /var/log/journald-docker.pos
      </storage>
      read_from_head true
      tag docker
    </source>

    <source>
      @id journald-container-runtime
      @type systemd
      matches [{ "_SYSTEMD_UNIT": "{{ fluentd_container_runtime_service }}.service" }]
      <storage>
        @type local
        persistent true
        path /var/log/journald-container-runtime.pos
      </storage>
      read_from_head true
      tag container-runtime
    </source>

    <source>
      @id journald-kubelet
      @type systemd
      matches [{ "_SYSTEMD_UNIT": "kubelet.service" }]
      <storage>
        @type local
        persistent true
        path /var/log/journald-kubelet.pos
      </storage>
      read_from_head true
      tag kubelet
    </source>

    <source>
      @id journald-node-problem-detector
      @type systemd
      matches [{ "_SYSTEMD_UNIT": "node-problem-detector.service" }]
      <storage>
        @type local
        persistent true
        path /var/log/journald-node-problem-detector.pos
      </storage>
      read_from_head true
      tag node-problem-detector
    </source>

    <source>
```

```
      @id kernel
      @type systemd
      matches [{ "_TRANSPORT": "kernel" }]
      <storage>
        @type local
        persistent true
        path /var/log/kernel.pos
      </storage>
      <entry>
        fields_strip_underscores true
        fields_lowercase true
      </entry>
      read_from_head true
      tag kernel
    </source>

  forward.input.conf: |-
    # Takes the messages sent over TCP
    <source>
      @id forward
      @type forward
    </source>

  monitoring.conf: |-
    # Prometheus Exporter Plugin
    # input plugin that exports metrics
    <source>
      @id prometheus
      @type prometheus
    </source>

    <source>
      @id monitor_agent
      @type monitor_agent
    </source>

    # input plugin that collects metrics from MonitorAgent
    <source>
      @id prometheus_monitor
      @type prometheus_monitor
      <labels>
        host ${hostname}
      </labels>
    </source>

    # input plugin that collects metrics for output plugin
    <source>
      @id prometheus_output_monitor
      @type prometheus_output_monitor
      <labels>
        host ${hostname}
      </labels>
    </source>

    # input plugin that collects metrics for in_tail plugin
    <source>
      @id prometheus_tail_monitor
      @type prometheus_tail_monitor
      <labels>
        host ${hostname}
      </labels>
    </source>
```

```
  output.conf: |-
    <match **>
      @id elasticsearch
      @type elasticsearch
      @log_level info
      type_name _doc
      include_tag_key true
      host elasticsearch-logging
      port 9200
      logstash_format true
      <buffer>
        @type file
        path /var/log/fluentd-buffers/kubernetes.system.buffer
        flush_mode interval
        retry_type exponential_backoff
        flush_thread_count 2
        flush_interval 5s
        retry_forever
        retry_max_interval 30
        chunk_limit_size 2M
        total_limit_size 500M
        overflow_action block
      </buffer>
    </match>
```

A ConfigMap in Kubernetes is used to store configuration data that can be used by other resources like pods. Let's break down this ConfigMap:

- `kind` and `apiVersion`: These fields specify the type and version of the Kubernetes resource. In this case, it's a `ConfigMap` with `v1` version.
- `metadata`: This section contains metadata about the ConfigMap, including its name, namespace, and labels for identification and categorization purposes.
- `data`: This section contains the actual configuration data. It consists of various configuration files, each under a different key:
    1. `system.conf`: This contains configuration for Fluentd's system settings, such as the root directory for Fluentd buffers.
    2. `containers.input.conf`: This is a configuration file for Fluentd that defines how to collect and parse container logs from Docker. It includes settings for tailing log files, parsing log formats (including JSON and CRI log formats), and adding metadata like Kubernetes namespace, pod name, and container name to the log records.
    3. `system.input.conf`: This section defines how Fluentd should collect system-level logs, such as logs from the kubelet, kube-proxy, and other Kubernetes components.
    4. `forward.input.conf`: This section specifies how Fluentd should handle messages sent over TCP using the forward protocol. It's a placeholder for receiving logs from other Fluentd instances.
    5. `monitoring.conf`: This section sets up Fluentd's Prometheus Exporter Plugin, which exports metrics. It configures sources for collecting monitoring metrics.

6. `output.conf`: This is the output configuration for Fluentd. It specifies how Fluentd should send log data to Elasticsearch. It uses the Elasticsearch output plugin to send logs to an Elasticsearch instance running at `elasticsearch-logging` on port `9200`.

vi fluentd-es-ds.yaml

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: fluentd-es
  namespace: kube-logging-space
  labels:
    k8s-app: fluentd-es
    addonmanager.kubernetes.io/mode: Reconcile
---
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: fluentd-es
  labels:
    k8s-app: fluentd-es
    addonmanager.kubernetes.io/mode: Reconcile
rules:
  - apiGroups:
      - ""
    resources:
      - "namespaces"
      - "pods"
    verbs:
      - "get"
      - "watch"
      - "list"
---
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: fluentd-es
  labels:
    k8s-app: fluentd-es
    addonmanager.kubernetes.io/mode: Reconcile
subjects:
  - kind: ServiceAccount
    name: fluentd-es
    namespace: kube-logging-space
    apiGroup: ""
roleRef:
  kind: ClusterRole
  name: fluentd-es
  apiGroup: ""
---
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: fluentd-es-v3.1.0
  namespace: kube-logging-space
  labels:
    k8s-app: fluentd-es
    version: v3.1.0
    addonmanager.kubernetes.io/mode: Reconcile
spec:
  selector:
    matchLabels:
      k8s-app: fluentd-es
      version: v3.1.0
  template:
```

```
    metadata:
      labels:
        k8s-app: fluentd-es
        version: v3.1.0
    spec:
      securityContext:
        seccompProfile:
          type: RuntimeDefault
      priorityClassName: system-node-critical
      serviceAccountName: fluentd-es
      containers:
        - name: fluentd-es
          image: quay.io/fluentd_elasticsearch/fluentd:v3.1.0
          env:
            - name: FLUENTD_ARGS
              value: --no-supervisor -q
          resources:
            limits:
              memory: 500Mi
            requests:
              cpu: 100m
              memory: 200Mi
          volumeMounts:
            - name: varlog
              mountPath: /var/log
            - name: varlibdockercontainers
              mountPath: /var/lib/docker/containers
              readOnly: true
            - name: config-volume
              mountPath: /etc/fluent/config.d
          ports:
            - containerPort: 24231
              name: prometheus
              protocol: TCP
          livenessProbe:
            tcpSocket:
              port: prometheus
            initialDelaySeconds: 5
            timeoutSeconds: 10
          readinessProbe:
            tcpSocket:
              port: prometheus
            initialDelaySeconds: 5
            timeoutSeconds: 10
      terminationGracePeriodSeconds: 30
      volumes:
        - name: varlog
          hostPath:
            path: /var/log
        - name: varlibdockercontainers
          hostPath:
            path: /var/snap/microk8s/common/var/lib/containerd
        - name: config-volume
          configMap:
            name: fluentd-es-config-v0.2.0
```

DaemonSets are used to run applications that need to be running on all Nodes, such as logging daemons or monitoring agents.

For example, a DaemonSet could be used to ensure that a logging daemon is running on all Nodes in a cluster. This would ensure that all of the logs from all of the Pods are being collected and stored.

A DaemonSet ensures that all (or some) Nodes run a copy of a Pod. The YAML code you provided creates a DaemonSet that ensures that a Pod named `fluentd-es-v3.1.0` is running on all Nodes in the `kube-logging-space` namespace.

The DaemonSet has the following configuration:

- **Selector:** The selector that is used to match the Pods that are managed by the DaemonSet. In this case, the selector matches Pods with the label `k8s-app: fluentd-es` and `version: v3.1.0`.

Containers: The DaemonSet creates a Pod that has a single container named `fluentd-es`.

  - The container runs the `quay.io/fluentd_elasticsearch/fluentd:v3.1.0` image.
  - **Env:** The container has an environment variable named `FLUENTD_ARGS` that is set to the value `--no-supervisor -q`. This tells the fluentd container to run in a non-supervisored mode and to be quiet.
  - **Resources:** The container is allowed to use up to 500Mi of memory.
  - **VolumeMounts:** The container mounts three volumes:
    - `varlog`: This volume is mounted at the path `/var/log` in the container. It contains the logs from all the containers running on the Pod.
    - `varlibdockercontainers`: This volume is mounted at the path `/var/lib/docker/containers` in the container. It contains the Docker container information for all the containers running on the Pod.
    - `config-volume`: This volume is a configMap that contains the configuration for the fluentd container.
  - **Ports:** The container exposes a port named `prometheus` on the host network. This port is used by Prometheus to scrape metrics from the fluentd container.
  - **LivenessProbe:** The container has a liveness probe that checks if the container is alive by connecting to port `prometheus`. If the probe fails 10 times, the container will be restarted.
  - **ReadinessProbe:** The container has a readiness probe that checks if the container is ready to serve requests by connecting to port `prometheus`. If the probe fails 10 times, the container will be removed from the load balancer.
- **TerminationGracePeriodSeconds:** The DaemonSet specifies that the container should be allowed to run for 30 seconds after it is gracefully terminated.

- **Volumes:** The DaemonSet uses three volumes:
  - `varlog`: This volume is a hostPath volume that is mounted at the path `/var/log` in the container. It contains the logs from all the containers running on the Pod.
  - `varlibdockercontainers`: This volume is a hostPath volume that is mounted at the path `/var/lib/docker/containers` in the container. It contains the Docker container information for all the containers running on the Pod.
  - `config-volume`: This volume is a configMap volume that contains the configuration for the fluentd container.

Apply manifest files:

```
kubectl apply -f fluentd-es-configmap.yaml
kubectl apply -f fluentd-es-ds.yaml
```

Test it :

Enable required modules:

```
vishal@vishal-HP-245-G8:~/vishal/efk-kubernetes$ microk8s enable dns
microk8s enable hostpath-storage
microk8s enable ingress
microk8s enable metrics-server
Infer repository core for addon dns
Addon core/dns is already enabled
Infer repository core for addon hostpath-storage
Addon core/hostpath-storage is already enabled
Infer repository core for addon ingress
Enabling Ingress
ingressclass.networking.k8s.io/public unchanged
ingressclass.networking.k8s.io/nginx unchanged
namespace/ingress unchanged
serviceaccount/nginx-ingress-microk8s-serviceaccount unchanged
clusterrole.rbac.authorization.k8s.io/nginx-ingress-microk8s-clusterrole unchanged
role.rbac.authorization.k8s.io/nginx-ingress-microk8s-role unchanged
clusterrolebinding.rbac.authorization.k8s.io/nginx-ingress-microk8s unchanged
rolebinding.rbac.authorization.k8s.io/nginx-ingress-microk8s unchanged
configmap/nginx-load-balancer-microk8s-conf unchanged
configmap/nginx-ingress-tcp-microk8s-conf unchanged
configmap/nginx-ingress-udp-microk8s-conf unchanged
daemonset.apps/nginx-ingress-microk8s-controller unchanged
Ingress is enabled
Infer repository core for addon metrics-server
Enabling Metrics-Server
serviceaccount/metrics-server unchanged
clusterrole.rbac.authorization.k8s.io/system:aggregated-metrics-reader unchanged
clusterrole.rbac.authorization.k8s.io/system:metrics-server unchanged
rolebinding.rbac.authorization.k8s.io/metrics-server-auth-reader unchanged
clusterrolebinding.rbac.authorization.k8s.io/metrics-server:system:auth-delegator unchanged
clusterrolebinding.rbac.authorization.k8s.io/system:metrics-server unchanged
service/metrics-server unchanged
deployment.apps/metrics-server configured
apiservice.apiregistration.k8s.io/v1beta1.metrics.k8s.io unchanged
clusterrolebinding.rbac.authorization.k8s.io/microk8s-admin unchanged
Metrics-Server is enabled
vishal@vishal-HP-245-G8:~/vishal/efk-kubernetes$
```

- I have already enabled

Enabled all manifest files:

```
kubectl apply -f name-space-kube-logging-space.yml
kubectl apply -f es-service.yaml
kubectl apply -f es-statefulset.yaml
kubectl apply -f kibana-service.yaml
kubectl apply -f kibana-deployment.yaml
kubectl apply -f fluentd-es-configmap.yaml
kubectl apply -f fluentd-es-ds.yaml
```

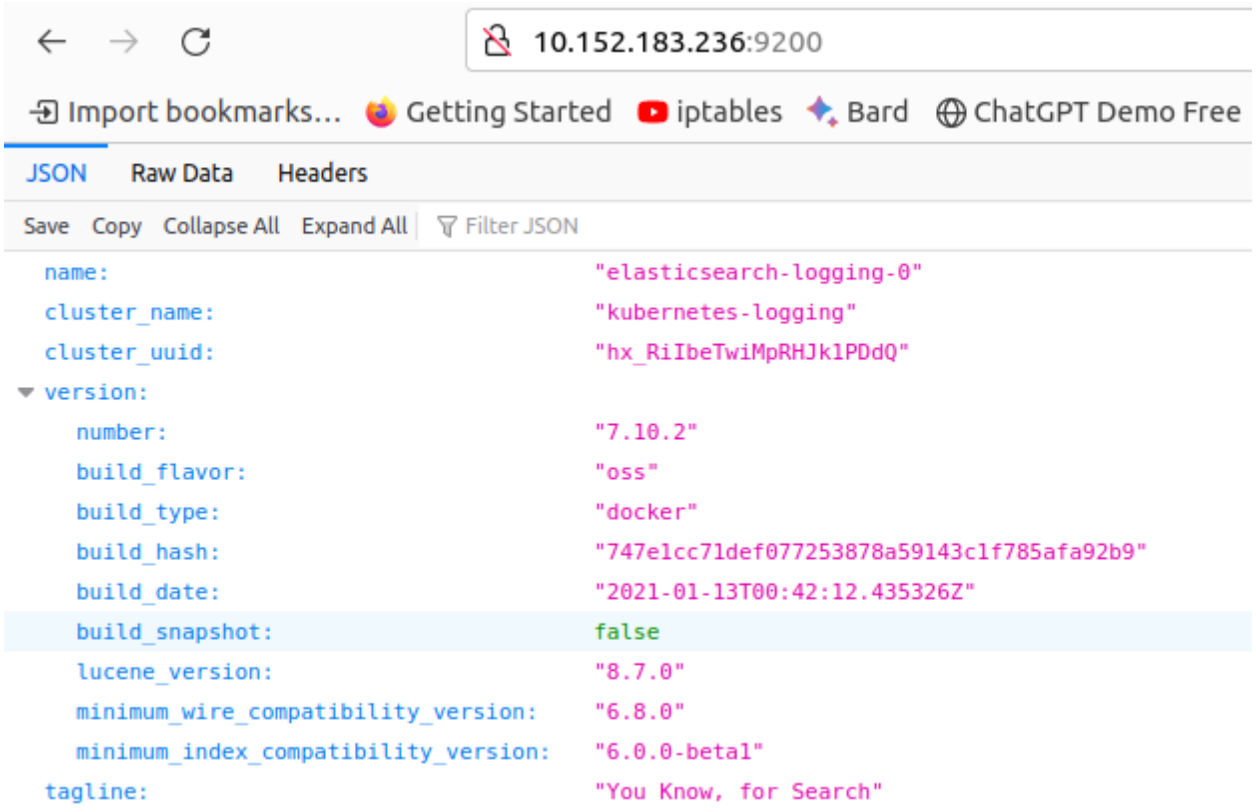
```
vishal@vishal-HP-245-G8:~/vishal/efk-kubernetes$ kubectl apply -f name-space-kube-logging-space.yml
kubectl apply -f es-service.yaml
kubectl apply -f es-statefulset.yaml
kubectl apply -f kibana-service.yaml
kubectl apply -f kibana-deployment.yaml
kubectl apply -f fluentd-es-configmap.yaml
kubectl apply -f fluentd-es-ds.yaml
namespace/kube-logging-space created
service/elasticsearch-logging created
serviceaccount/elasticsearch-logging created
clusterrole.rbac.authorization.k8s.io/elasticsearch-logging created
clusterrolebinding.rbac.authorization.k8s.io/elasticsearch-logging created
statefulset.apps/elasticsearch-logging created
service/kibana-logging created
deployment.apps/kibana-logging created
configmap/fluentd-es-config-v0.2.0 created
serviceaccount/fluentd-es created
clusterrole.rbac.authorization.k8s.io/fluentd-es created
clusterrolebinding.rbac.authorization.k8s.io/fluentd-es created
daemonset.apps/fluentd-es-v3.1.0 created
vishal@vishal-HP-245-G8:~/vishal/efk-kubernetes$
```

check all things in space of kube-logging-space

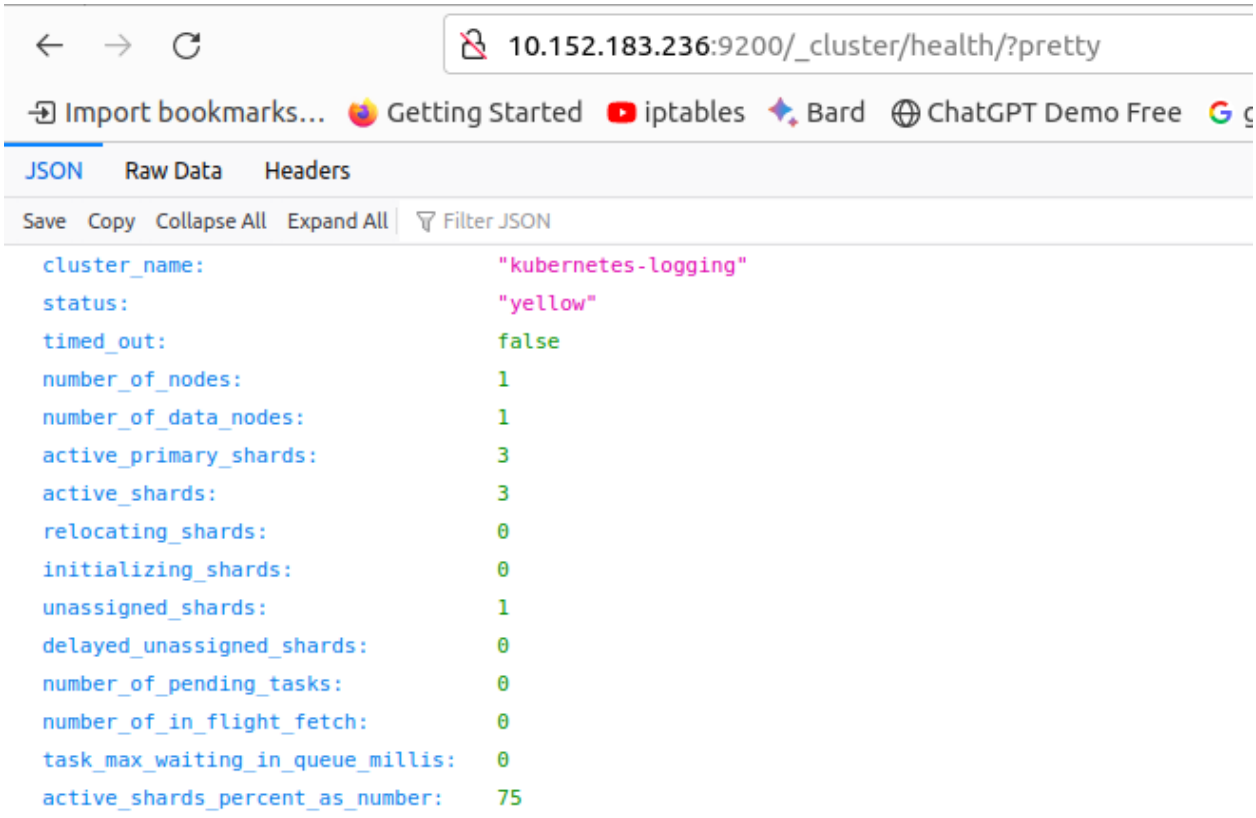
```
vishal@vishal-HP-245-G8:~/vishal/efk-kubernetes$ kubectl get all -n kube-logging-space
NAME                                READY   STATUS    RESTARTS      AGE
pod/elasticsearch-logging-0         1/1     Running   0             86s
pod/kibana-logging-54996f69d8-l8vcx 1/1     Running   1 (53s ago)   85s
pod/fluentd-es-v3.1.0-472hj         1/1     Running   1             83s

NAME                            TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)                         AGE
service/elasticsearch-logging   NodePort    10.152.183.236   <none>        9200:31015/TCP,9300:31785/TCP   86s
service/kibana-logging          ClusterIP   10.152.183.71    <none>        5601/TCP                        85s

NAME                               DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
daemonset.apps/fluentd-es-v3.1.0   1         1         1       1            1           <none>          83s

NAME                              READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/kibana-logging    1/1     1            1           85s

NAME                                         DESIRED   CURRENT   READY   AGE
replicaset.apps/kibana-logging-54996f69d8    1         1         1       85s

NAME                                     READY   AGE
statefulset.apps/elasticsearch-logging   1/1     86s
vishal@vishal-HP-245-G8:~/vishal/efk-kubernetes$
```

Check elasticsearch service:

http://<ip-of-es-service>:9200/
In my case it is :
http://10.152.183.236:9200/



```
http://<ip-of-es-service>:9200/_cluster/health/?pretty

In my case here curl
http://10.152.183.236:9200/_cluster/health/?pretty
```



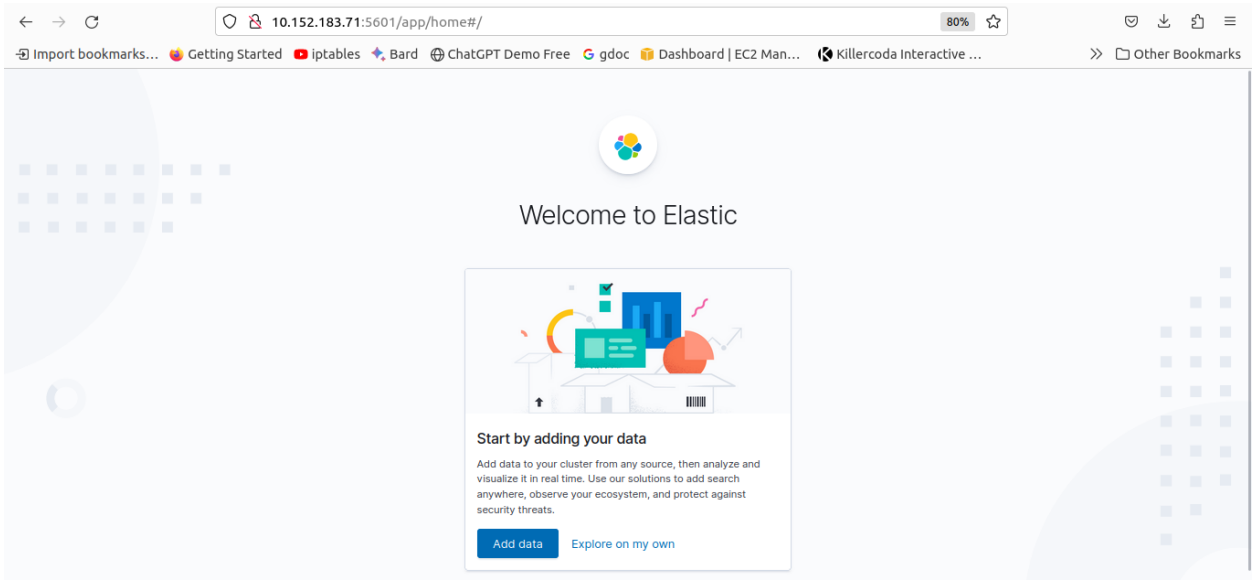
Check kibana :

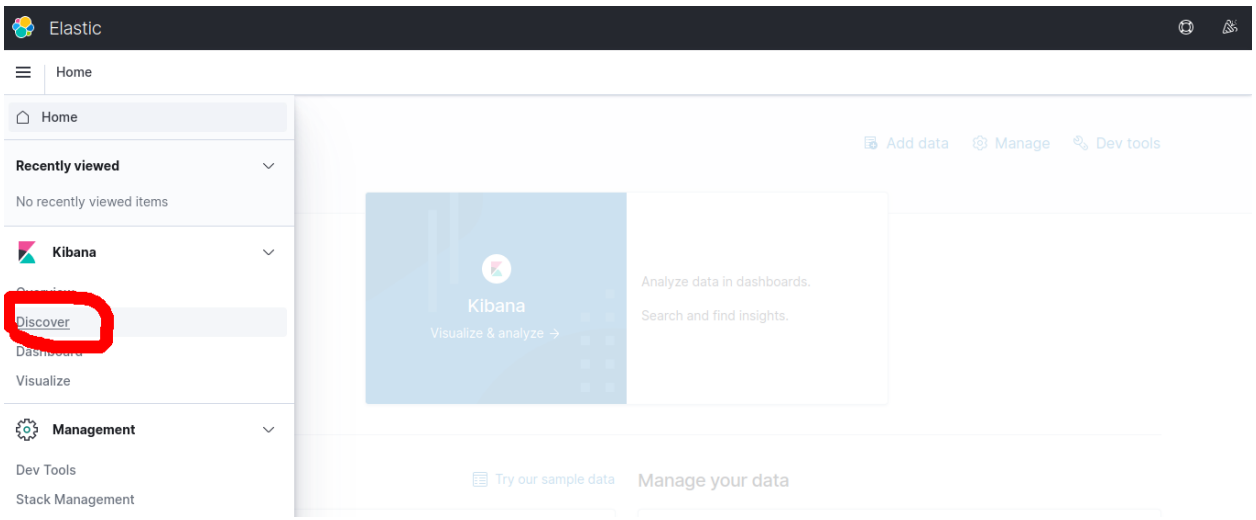
Url would be:

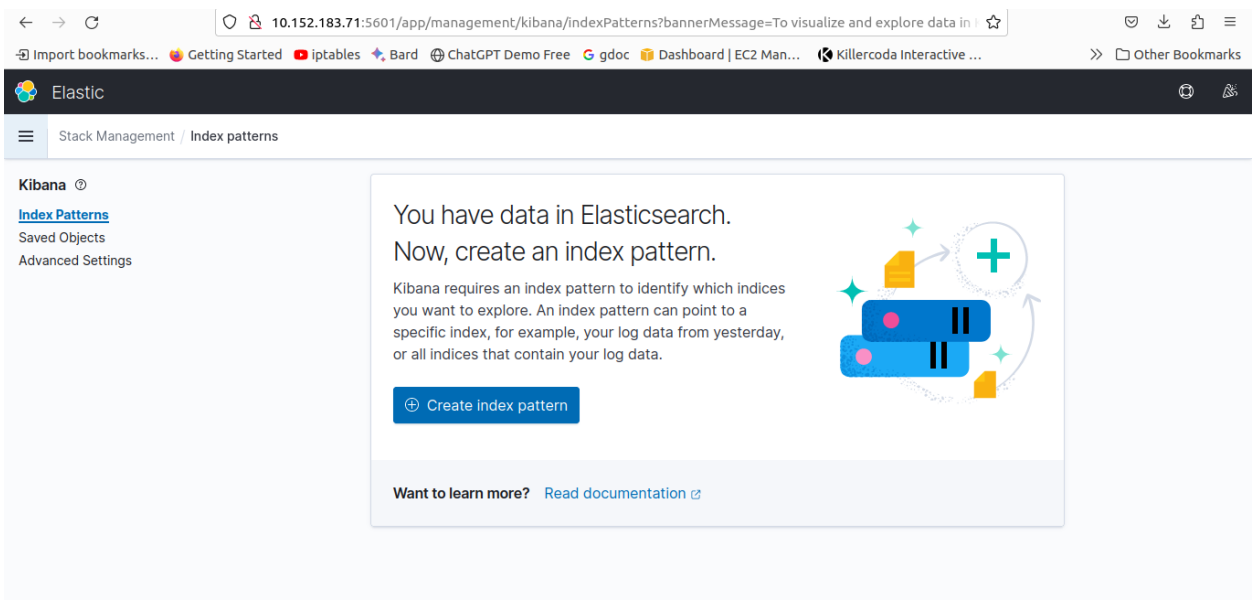
Kibana access point http://<ip-of-kibana-service>:5601

In my case it is ,

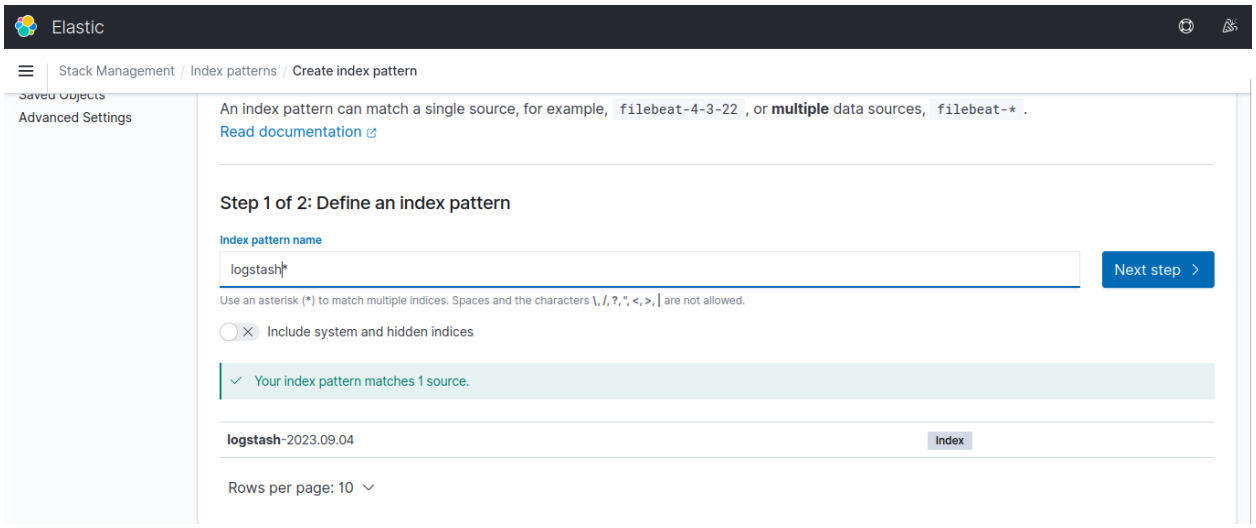
http://10.152.183.71:5601



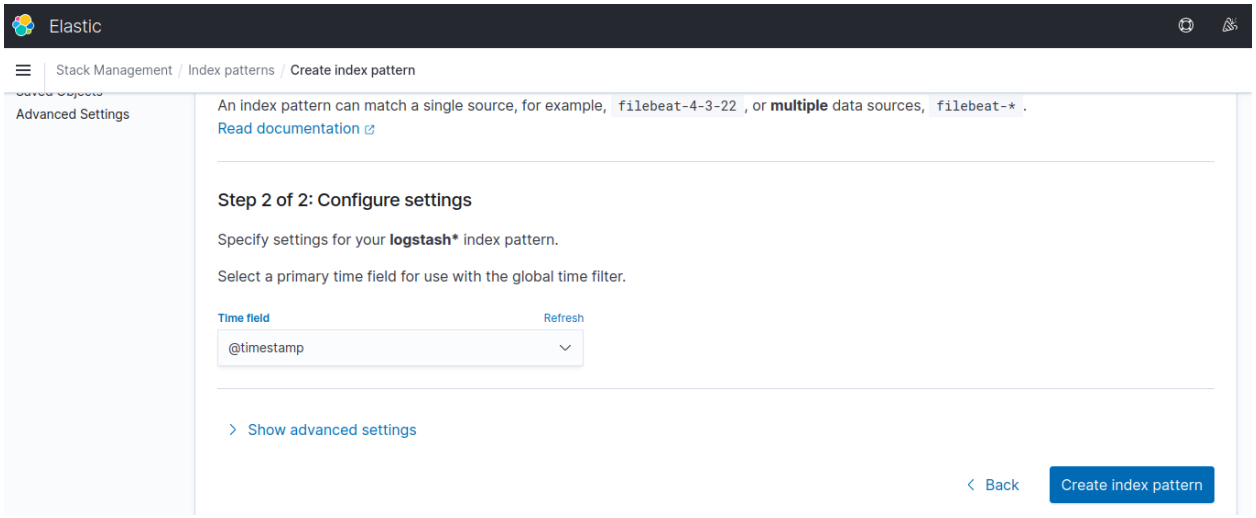




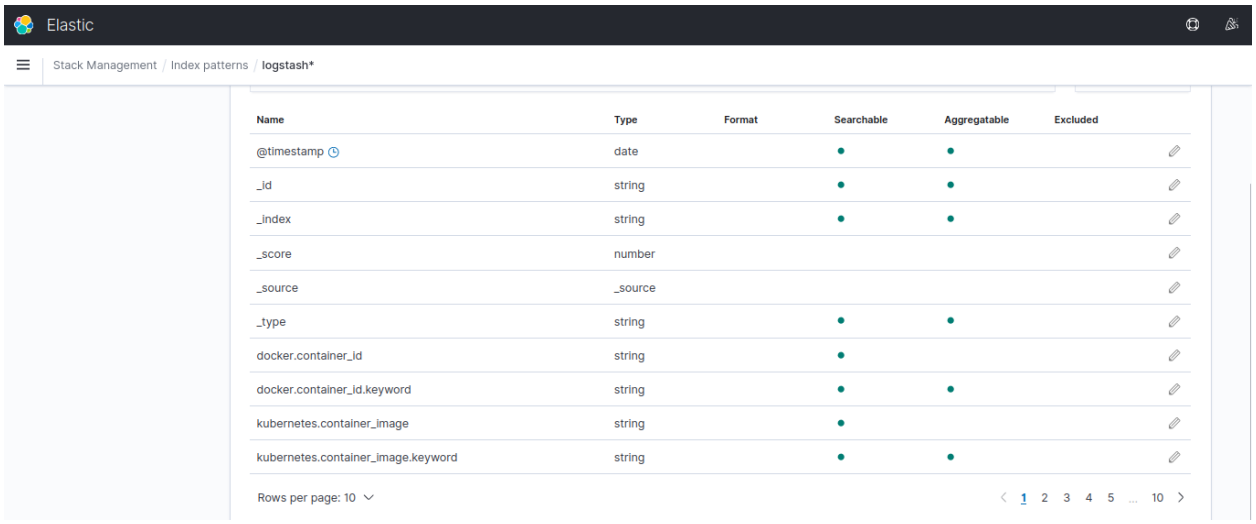
- Click on create index pattern



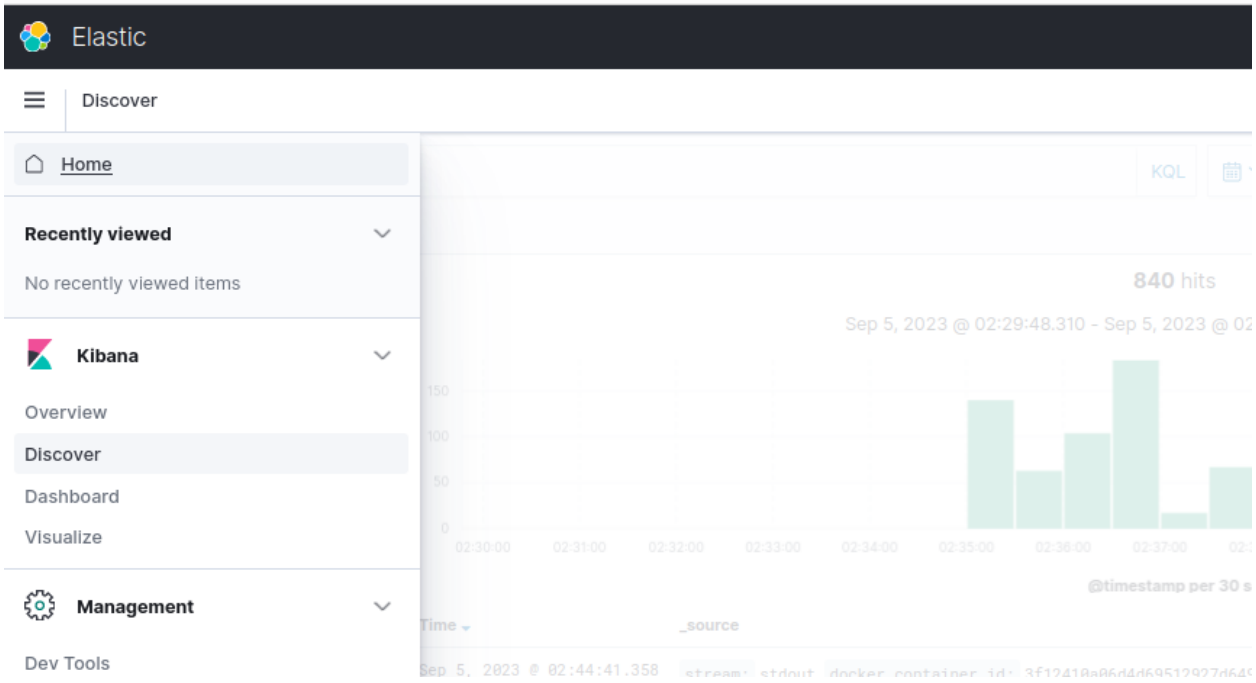
- Type a name matching with index pattern and proceed to next step.

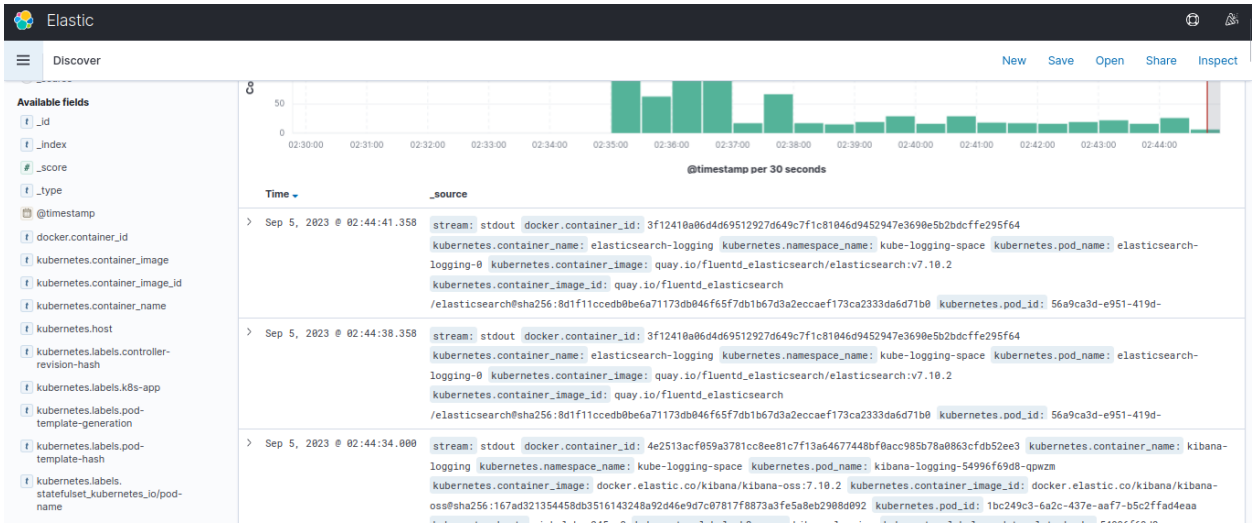


Select @ timestamp then proceed further and click on create index pattern



- You will see like this
- Again click on discover , you should see as per below screenshot





- Here are logs

My devops repo :

- https://github.com/vishalk17/devops

My telegram channel:

- https://t.me/vishalk17_devops

Contact:

Telegram : t.me/vishalk17

vishalk17 My youtube Channel :

- YouTube https://www.youtube.com/@vishalk17