LIS Nepal

Lalitpur, Nepal

LIS NEPAL

A YÓMARÍ COMPANY

EFK Stack: Documentation

Documented by

Aayam Adhikari

Software Intern - 2024

1. What is the EFK stack?

EFK, which stands for Elasticsearch, Fluentd, and Kibana, is a popular open-source stack used for collecting, processing, and visualizing logs.

Elasticsearch is a distributed search and analytics engine that can store and index large amounts of data. Fluentd is a data collector that can gather data from a variety of sources and send it to Elasticsearch for indexing. Kibana is a data visualization tool that allows users to interact with data stored in Elasticsearch and create custom dashboards and visualizations. Together, these three components provide a comprehensive logging solution that can handle a large amount of data and provide real-time insights into the behavior of complex systems. Below is a brief intro to each of these:

Elasticsearch: Elasticsearch is a distributed search and analytics engine designed for storing and indexing large volumes of data. It provides fast search capabilities and is highly scalable, making it ideal for use cases such as logging, search, and analytics within the EFK stack.

Fluentd: Fluentd is a versatile data collector that gathers logs from various sources and sends them to Elasticsearch for indexing. It supports a wide range of inputs and outputs, processes logs in real-time, and can enrich logs with metadata before sending them to Elasticsearch.

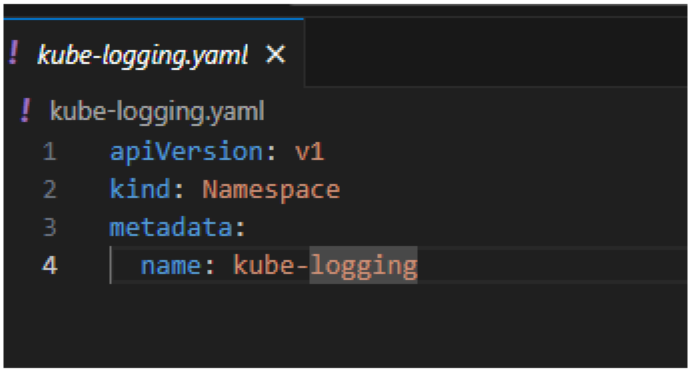
Kibana: Kibana is a powerful data visualization tool that allows users to interact with data stored in Elasticsearch. It enables users to create custom dashboards, charts, and visualizations to analyze and monitor logs in real-time. Kibana is essential for gaining insights into system behavior and monitoring application performance within the EFK stack.

2. Steps to set-up EFK stack for your logging solution

Step-1: Creating a Namespace

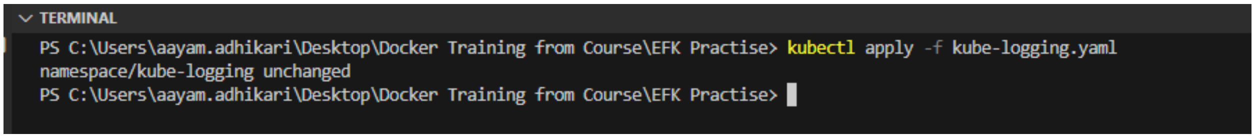
Before we roll out an Elasticsearch cluster, we'll first create a Namespace into which we'll install all of our logging instrumentation. Kubernetes lets you separate objects running in your cluster using a "virtual cluster" abstraction called Namespaces.

To begin, we'll create a Namespace called **kube-logging**. To do so, create a configuration file in your working directory (project folder) named as **kube-logging.yaml**, and write the following configuration code in that file:



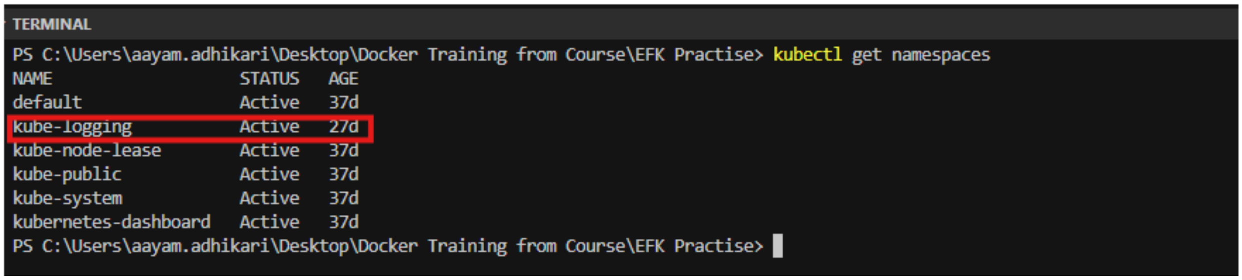
```
apiVersion: v1
kind: Namespace
metadata:
  name: kube-logging
```

Now, apply this configuration file using **kubectl apply -f kube-logging.yaml** in the terminal as:



```
PS C:\Users\aayam.adhikari\Desktop\Docker Training from Course\EFK Practise> kubectl apply -f kube-logging.yaml
namespace/kube-logging unchanged
PS C:\Users\aayam.adhikari\Desktop\Docker Training from Course\EFK Practise>
```

Then, confirm if the Namespace was successfully created:

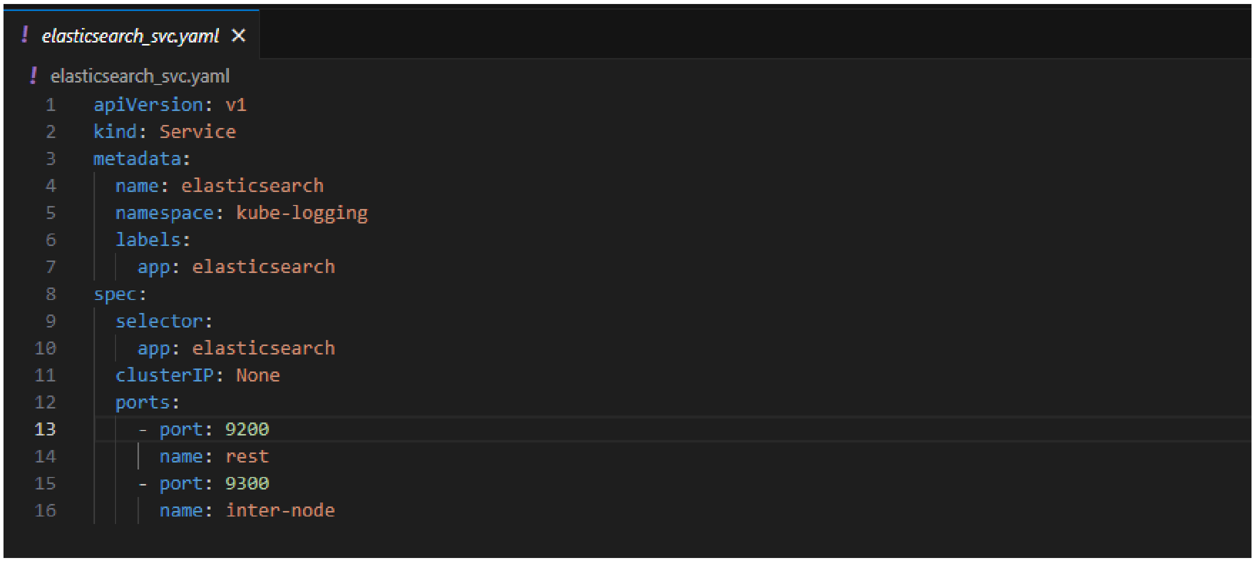


```
PS C:\Users\aayam.adhikari\Desktop\Docker Training from Course\EFK Practise> kubectl get namespaces
NAME                   STATUS   AGE
default                Active   37d
kube-logging           Active   27d
kube-node-lease        Active   37d
kube-public            Active   37d
kube-system            Active   37d
kubernetes-dashboard   Active   37d
PS C:\Users\aayam.adhikari\Desktop\Docker Training from Course\EFK Practise>
```

Step - 2: Creating the Elasticsearch Statefulset.

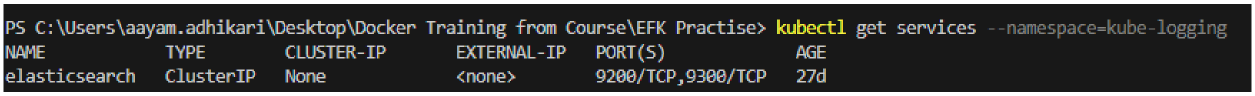
To start, we'll create a headless Kubernetes service called **elasticsearch** that will define a DNS domain for the 3 Pods. A headless service does not perform load balancing or have a static IP; to learn more about headless services.

To do so, create a file named **elasticsearch_svc.yaml** and write the following kubernetes service YAML:



```yaml
apiVersion: v1
kind: Service
metadata:
  name: elasticsearch
  namespace: kube-logging
  labels:
    app: elasticsearch
spec:
  selector:
    app: elasticsearch
  clusterIP: None
  ports:
    - port: 9200
      name: rest
    - port: 9300
      name: inter-node
```

Create the service using **kubectl apply -f elasticsearch_svc.yaml,** and check that the service was successfully created using **kubectl get services —namespace=kube-logging** as:

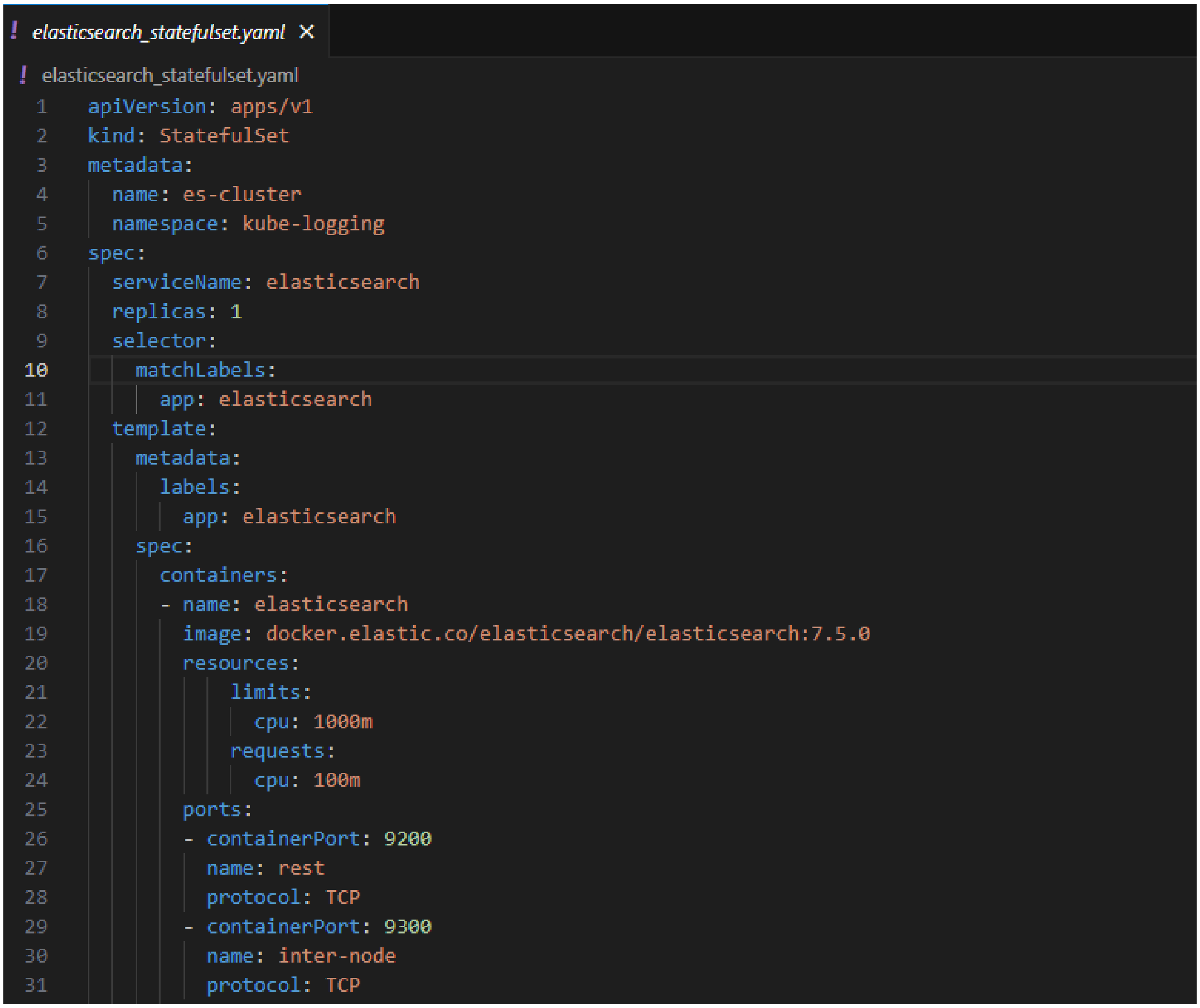


```
PS C:\Users\aayam.adhikari\Desktop\Docker Training from Course\EFK Practise> kubectl get services --namespace=kube-logging
NAME            TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)             AGE
elasticsearch   ClusterIP   None         <none>        9200/TCP,9300/TCP   27d
```

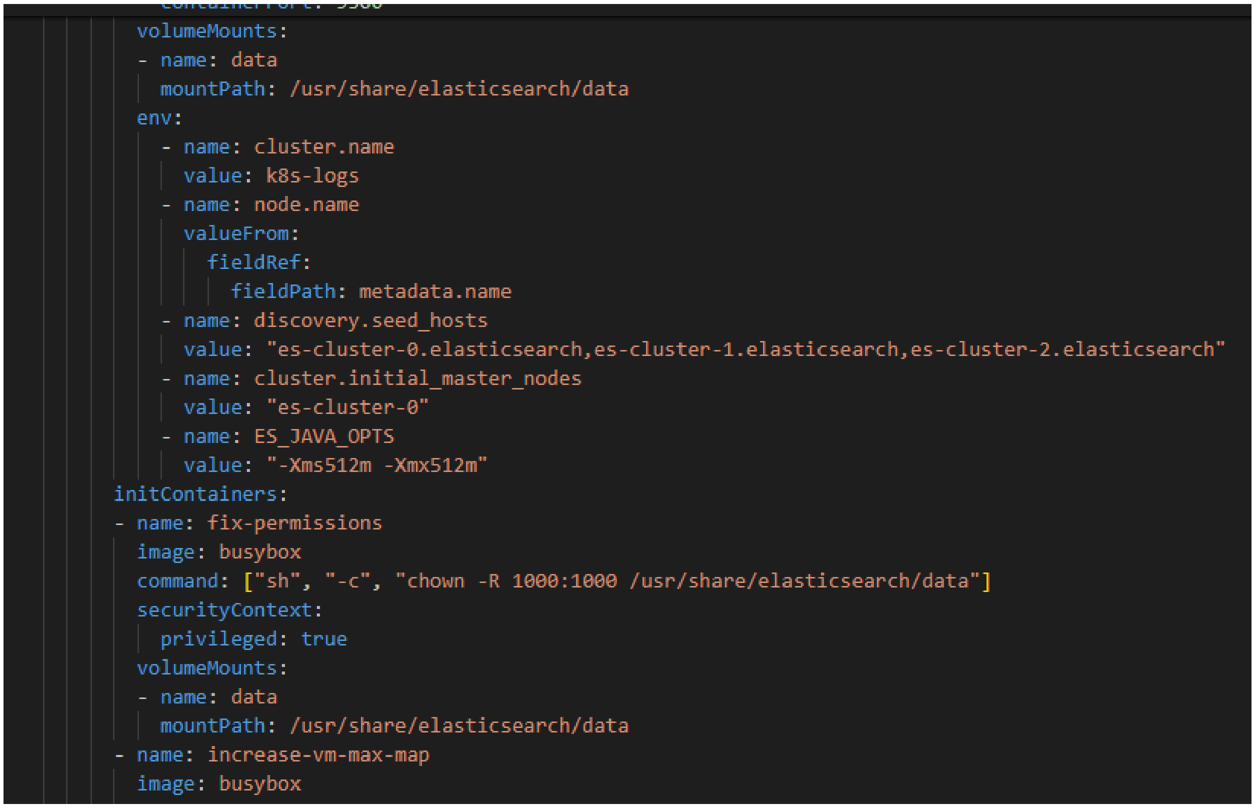
Now, create the Statefulset:

A Kubernetes StatefulSet allows you to assign a stable identity to Pods and grant them stable, persistent storage. Elasticsearch requires stable storage to persist data across Pod rescheduling and restarts.

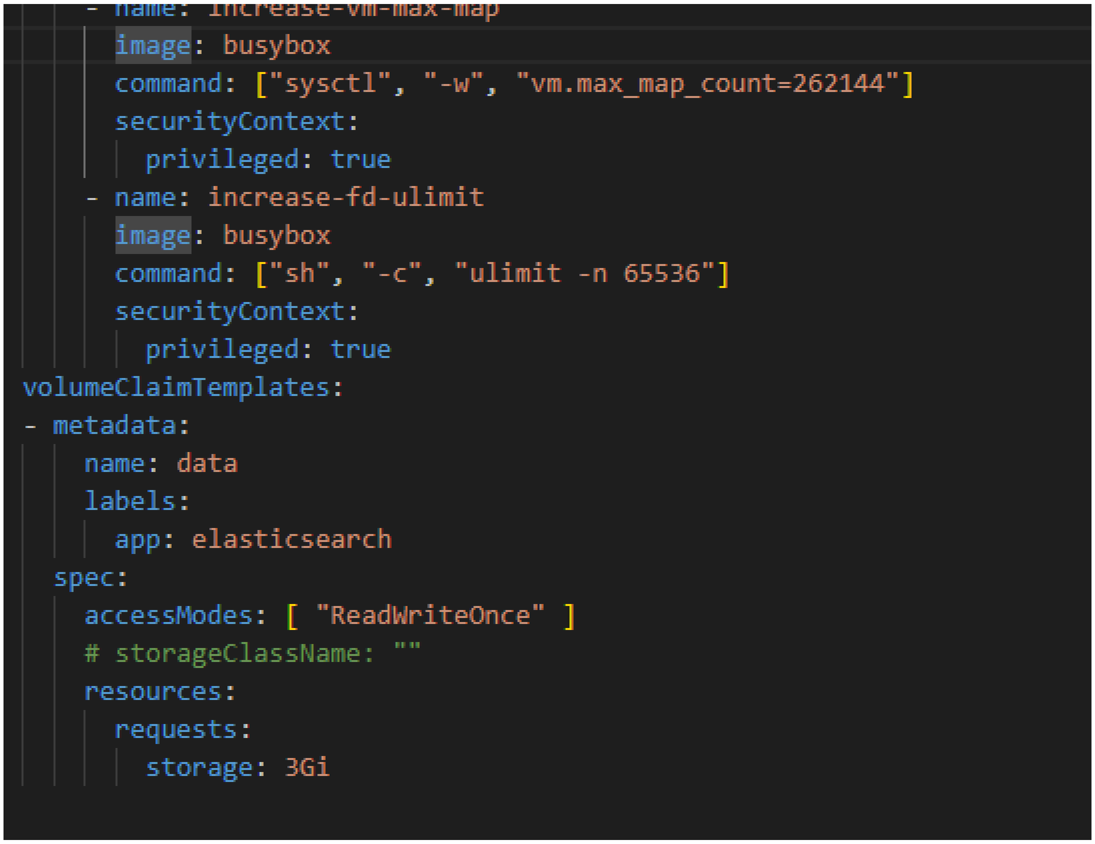
First, open a file named **elasticsearch_stetefulset.yaml,** and write the following kubernetes yaml configuration in it:


elasticsearch_statefulset.yaml
elasticsearch_statefulset.yaml

```yaml
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: es-cluster
  namespace: kube-logging
spec:
  serviceName: elasticsearch
  replicas: 1
  selector:
    matchLabels:
      app: elasticsearch
  template:
    metadata:
      labels:
        app: elasticsearch
    spec:
      containers:
      - name: elasticsearch
        image: docker.elastic.co/elasticsearch/elasticsearch:7.5.0
        resources:
            limits:
              cpu: 1000m
            requests:
              cpu: 100m
        ports:
        - containerPort: 9200
          name: rest
          protocol: TCP
        - containerPort: 9300
          name: inter-node
          protocol: TCP
```

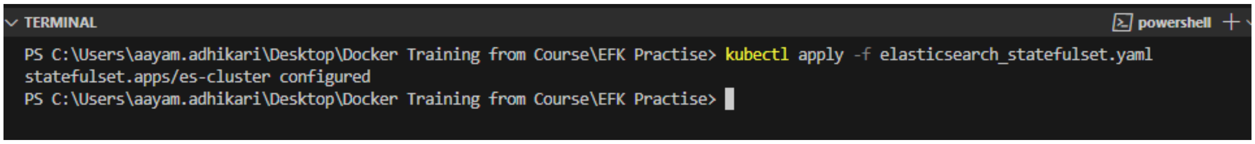


```yaml
    containerPort: 9300
  volumeMounts:
  - name: data
    mountPath: /usr/share/elasticsearch/data
  env:
    - name: cluster.name
      value: k8s-logs
    - name: node.name
      valueFrom:
        fieldRef:
          fieldPath: metadata.name
    - name: discovery.seed_hosts
      value: "es-cluster-0.elasticsearch,es-cluster-1.elasticsearch,es-cluster-2.elasticsearch"
    - name: cluster.initial_master_nodes
      value: "es-cluster-0"
    - name: ES_JAVA_OPTS
      value: "-Xms512m -Xmx512m"
initContainers:
- name: fix-permissions
  image: busybox
  command: ["sh", "-c", "chown -R 1000:1000 /usr/share/elasticsearch/data"]
  securityContext:
    privileged: true
  volumeMounts:
  - name: data
    mountPath: /usr/share/elasticsearch/data
- name: increase-vm-max-map
  image: busybox
```

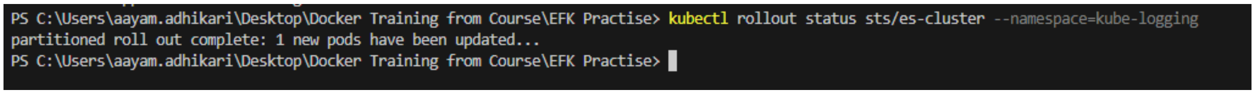


```yaml
    - name: increase-vm-max-map
      image: busybox
      command: ["sysctl", "-w", "vm.max_map_count=262144"]
      securityContext:
        privileged: true
    - name: increase-fd-ulimit
      image: busybox
      command: ["sh", "-c", "ulimit -n 65536"]
      securityContext:
        privileged: true
volumeClaimTemplates:
- metadata:
    name: data
    labels:
      app: elasticsearch
  spec:
    accessModes: [ "ReadWriteOnce" ]
    # storageClassName: ""
    resources:
      requests:
        storage: 3Gi
```

Now, deploy the stateful set using command **kubectl apply -f elasticsearch_statefulset.yaml** and monitor the StatefulSet as it is rolled out using **kubectl rollout status** as follows:



```
PS C:\Users\aayam.adhikari\Desktop\Docker Training from Course\EFK Practise> kubectl apply -f elasticsearch_statefulset.yaml
statefulset.apps/es-cluster configured
PS C:\Users\aayam.adhikari\Desktop\Docker Training from Course\EFK Practise> 
```

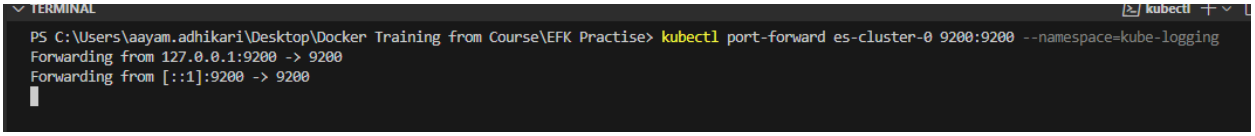


```
PS C:\Users\aayam.adhikari\Desktop\Docker Training from Course\EFK Practise> kubectl rollout status sts/es-cluster --namespace=kube-logging
partitioned roll out complete: 1 new pods have been updated...
PS C:\Users\aayam.adhikari\Desktop\Docker Training from Course\EFK Practise> 
```

Here, I've only deployed 1 pod, due to the memory limitation and also, because the minikube cluster only includes 1 node (which acts as both the master node and worker node).

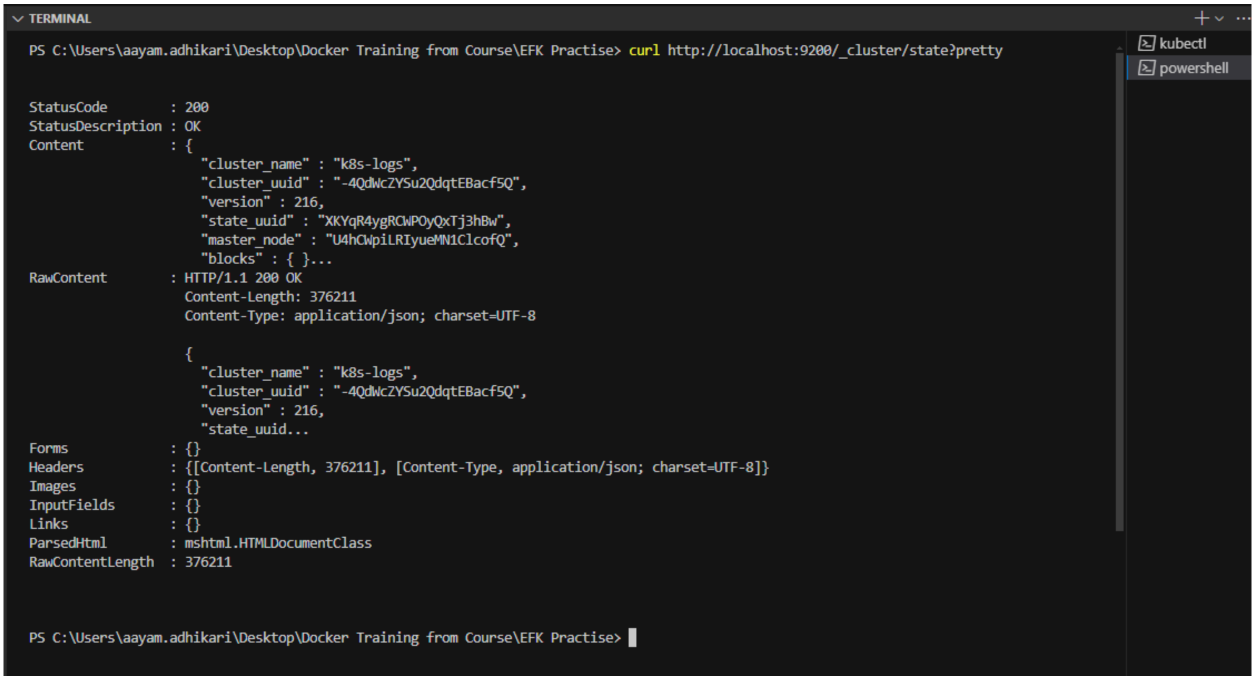
Once the Pods have been deployed, you can check that your Elasticsearch cluster is functioning correctly by performing a request against the REST API.

To do so, first forward the local port 9200 to the port 9200 on one of the Elasticsearch nodes (es-cluster-0) using kubectl port-forward:



```
PS C:\Users\aayam.adhikari\Desktop\Docker Training from Course\EFK Practise> kubectl port-forward es-cluster-0 9200:9200 --namespace=kube-logging
Forwarding from 127.0.0.1:9200 -> 9200
Forwarding from [::1]:9200 -> 9200
```

Now, In a separate terminal window, perform a curl request against the REST API as:

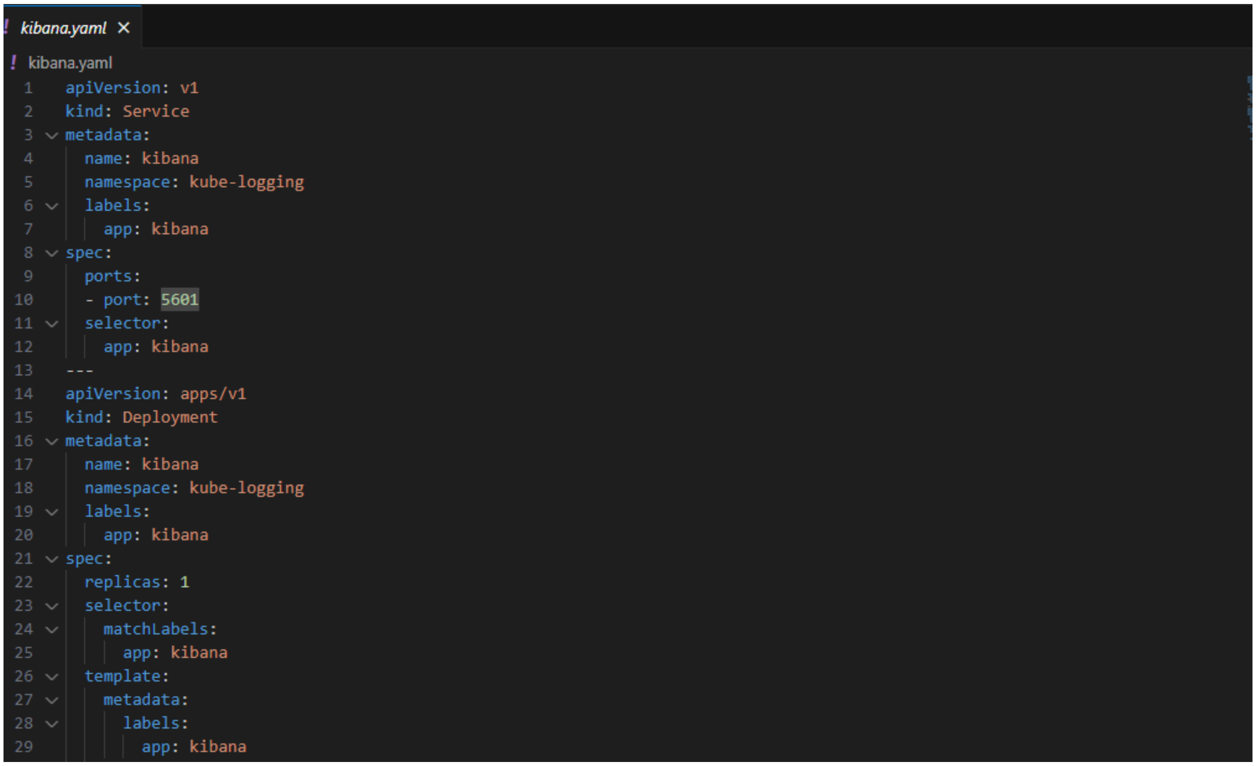


```
PS C:\Users\aayam.adhikari\Desktop\Docker Training from Course\EFK Practise> curl http://localhost:9200/_cluster/state?pretty


StatusCode        : 200
StatusDescription : OK
Content           : {
                      "cluster_name" : "k8s-logs",
                      "cluster_uuid" : "-4QdWcZYSu2QdqtEBacf5Q",
                      "version" : 216,
                      "state_uuid" : "XKYqR4ygRCWPOyQxTj3hBw",
                      "master_node" : "U4hCWpiLRIyueMN1ClcofQ",
                      "blocks" : { }...
RawContent        : HTTP/1.1 200 OK
                    Content-Length: 376211
                    Content-Type: application/json; charset=UTF-8

                    {
                      "cluster_name" : "k8s-logs",
                      "cluster_uuid" : "-4QdWcZYSu2QdqtEBacf5Q",
                      "version" : 216,
                      "state_uuid...
Forms             : {}
Headers           : {[Content-Length, 376211], [Content-Type, application/json; charset=UTF-8]}
Images            : {}
InputFields       : {}
Links             : {}
ParsedHtml        : mshtml.HTMLDocumentClass
RawContentLength  : 376211



PS C:\Users\aayam.adhikari\Desktop\Docker Training from Course\EFK Practise>
```

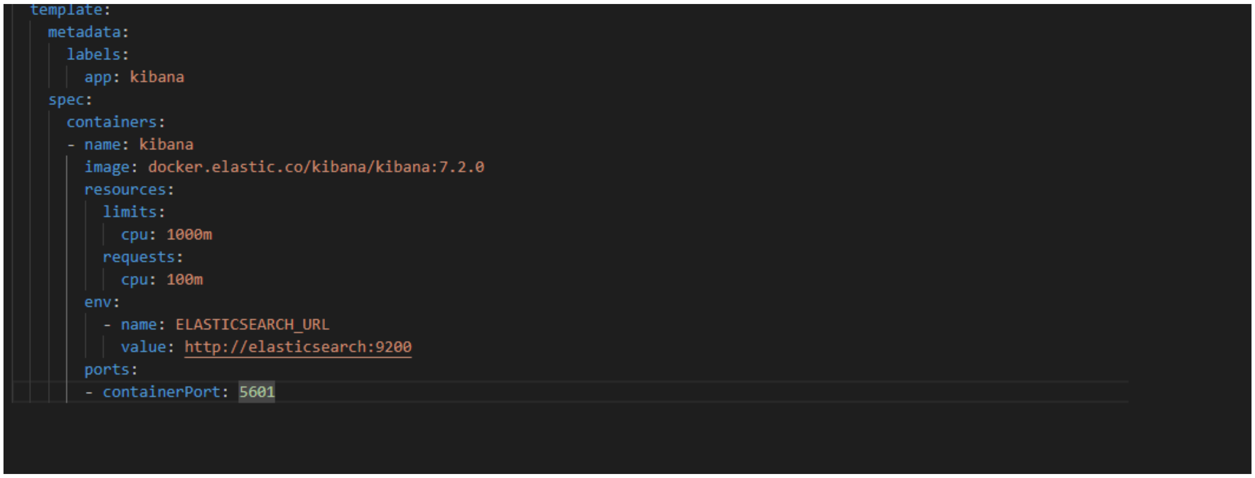
Step - 3: Creating the Kibana Deployment and Service.

To launch Kibana on Kubernetes, we'll create a Service called **kibana**, and a Deployment consisting of one Pod replica. You can scale the number of replicas depending on your production needs, and optionally specify a **LoadBalancer** type for the Service to load balance requests across the Deployment pods.

This time, we'll create the Service and Deployment in the same file named **kibana.yaml** as:

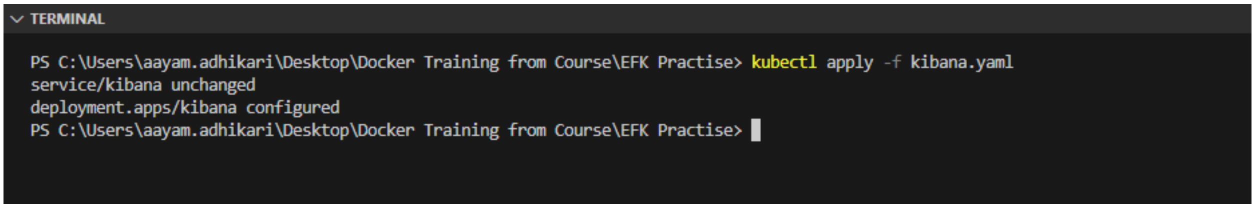


```yaml
apiVersion: v1
kind: Service
metadata:
  name: kibana
  namespace: kube-logging
  labels:
    app: kibana
spec:
  ports:
  - port: 5601
  selector:
    app: kibana
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: kibana
  namespace: kube-logging
  labels:
    app: kibana
spec:
  replicas: 1
  selector:
    matchLabels:
      app: kibana
  template:
    metadata:
      labels:
        app: kibana
```

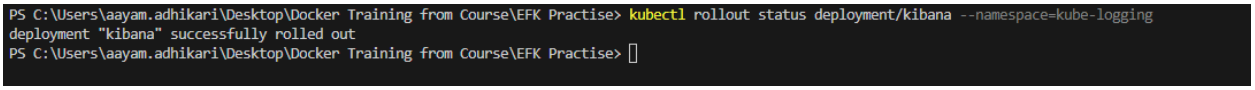


```yaml
  template:
    metadata:
      labels:
        app: kibana
    spec:
      containers:
      - name: kibana
        image: docker.elastic.co/kibana/kibana:7.2.0
        resources:
          limits:
            cpu: 1000m
          requests:
            cpu: 100m
        env:
          - name: ELASTICSEARCH_URL
            value: http://elasticsearch:9200
        ports:
        - containerPort: 5601
```

Now, roll out the Service and Deployment using **kubectl** as follows:

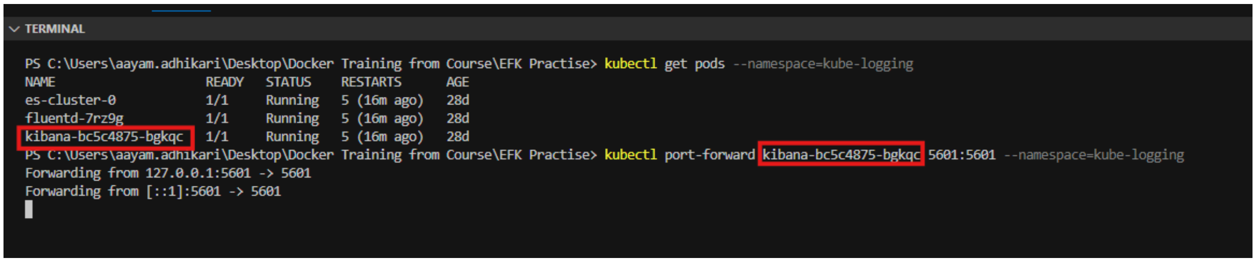


```
PS C:\Users\aayam.adhikari\Desktop\Docker Training from Course\EFK Practise> kubectl apply -f kibana.yaml
service/kibana unchanged
deployment.apps/kibana configured
PS C:\Users\aayam.adhikari\Desktop\Docker Training from Course\EFK Practise> 
```



```
PS C:\Users\aayam.adhikari\Desktop\Docker Training from Course\EFK Practise> kubectl rollout status deployment/kibana --namespace=kube-logging
deployment "kibana" successfully rolled out
PS C:\Users\aayam.adhikari\Desktop\Docker Training from Course\EFK Practise> 
```

Now, To access the Kibana interface, we'll once again forward a local port to the Kubernetes node running Kibana. Grab the Kibana Pod details using **kubectl get** as and also forward the port **5601** to port **5601** on this pod:

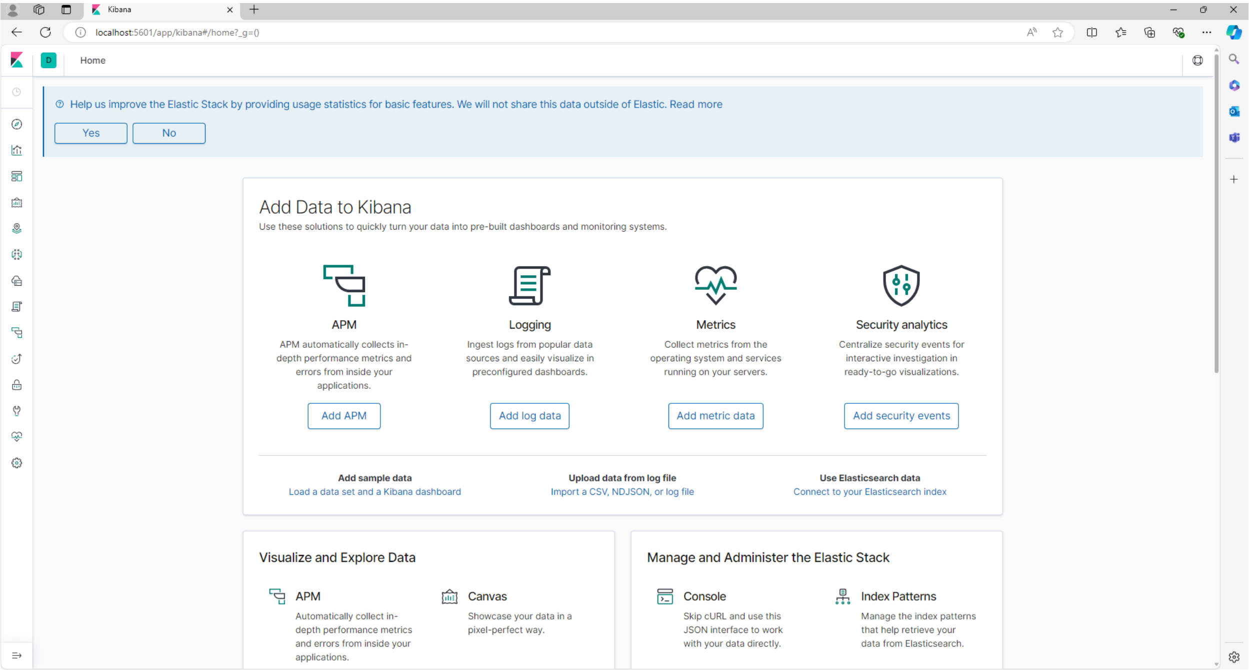


```
PS C:\Users\aayam.adhikari\Desktop\Docker Training from Course\EFK Practise> kubectl get pods --namespace=kube-logging
NAME                    READY   STATUS    RESTARTS      AGE
es-cluster-0            1/1     Running   5 (16m ago)   28d
fluentd-7rz9g           1/1     Running   5 (16m ago)   28d
kibana-bc5c4875-bgkqc   1/1     Running   5 (16m ago)   28d
PS C:\Users\aayam.adhikari\Desktop\Docker Training from Course\EFK Practise> kubectl port-forward kibana-bc5c4875-bgkqc 5601:5601 --namespace=kube-logging
Forwarding from 127.0.0.1:5601 -> 5601
Forwarding from [::1]:5601 -> 5601
```

Then, In your web browser, visit the following URL:

http://localhost:5601

If you see the following Kibana welcome page, you've successfully deployed Kibana into your Kubernetes cluster:



Help us improve the Elastic Stack by providing usage statistics for basic features. We will not share this data outside of Elastic. Read more

Yes No

Add Data to Kibana

Use these solutions to quickly turn your data into pre-built dashboards and monitoring systems.

APM

APM automatically collects in-depth performance metrics and errors from inside your applications.

Add APM

Logging

Ingest logs from popular data sources and easily visualize in preconfigured dashboards.

Add log data

Metrics

Collect metrics from the operating system and services running on your servers.

Add metric data

Security analytics

Centralize security events for interactive investigation in ready-to-go visualizations.

Add security events

Add sample data
Load a data set and a Kibana dashboard

Upload data from log file
Import a CSV, NDJSON, or log file

Use Elasticsearch data
Connect to your Elasticsearch index

Visualize and Explore Data

APM

Automatically collect in-depth performance metrics and errors from inside your applications.

Canvas

Showcase your data in a pixel-perfect way.

Manage and Administer the Elastic Stack

Console

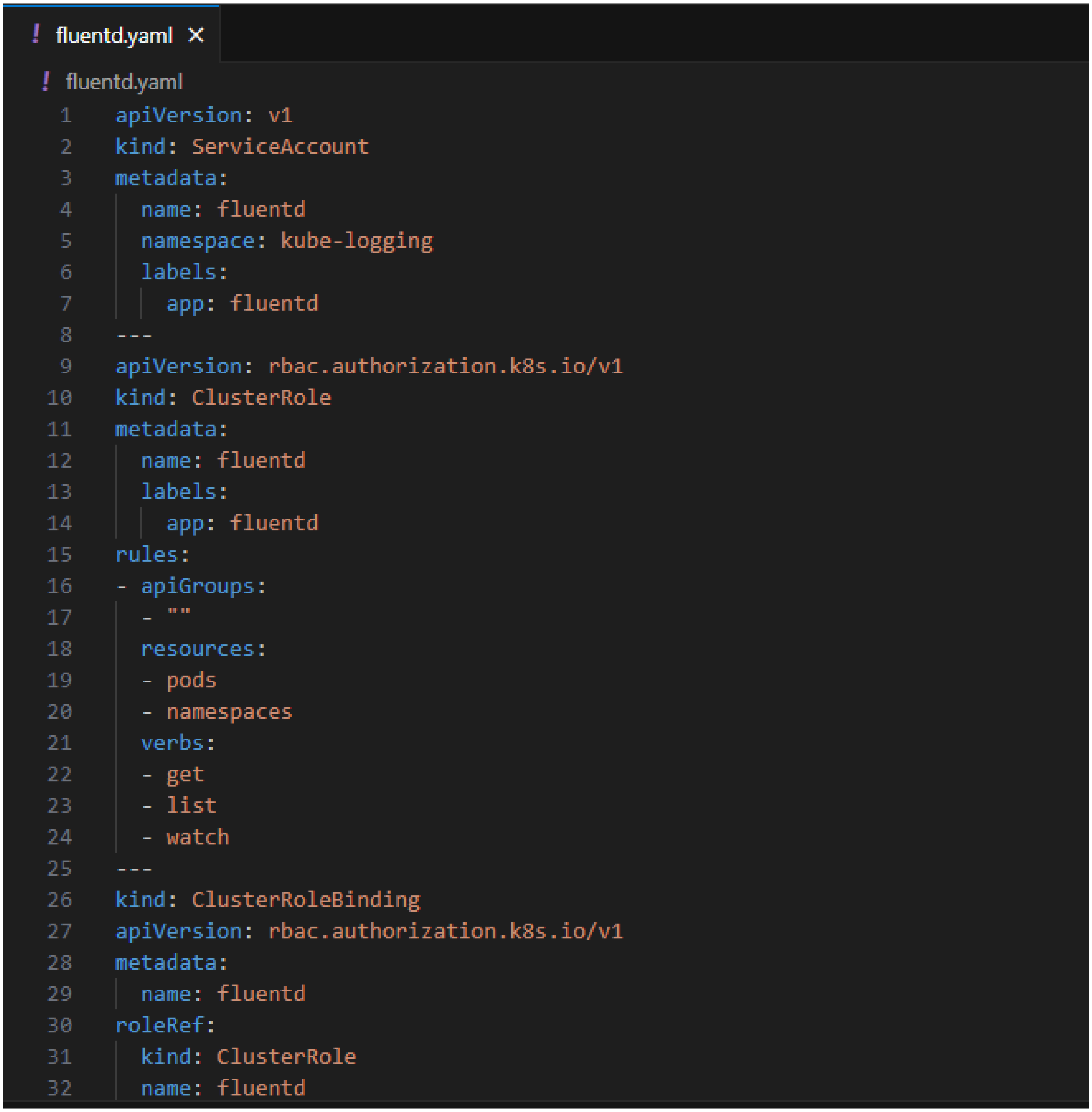
Skip cURL and use this JSON interface to work with your data directly.

Index Patterns

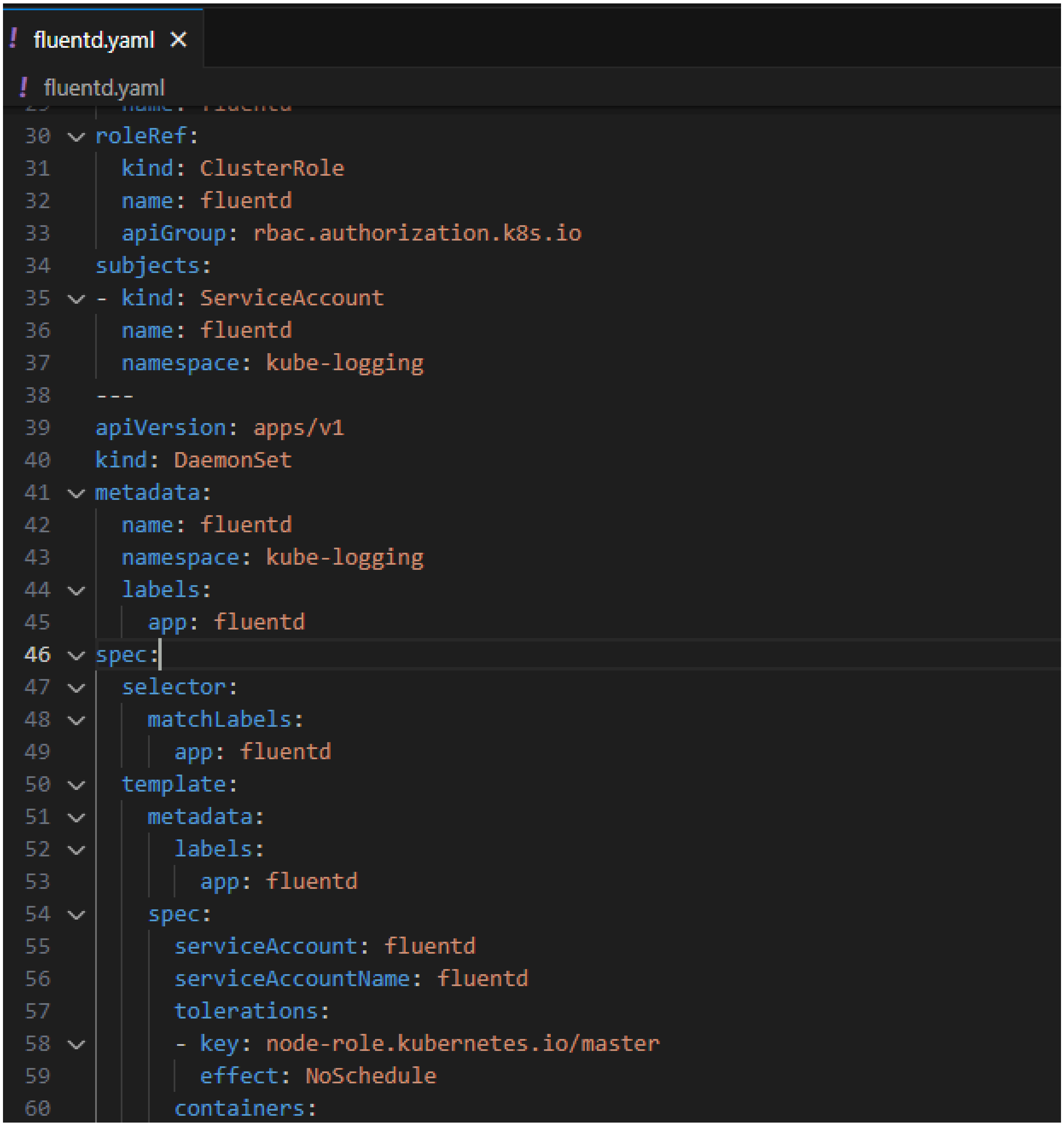
Manage the index patterns that help retrieve your data from Elasticsearch.

Step - 4: Creating the Fluentd DaemonSet.

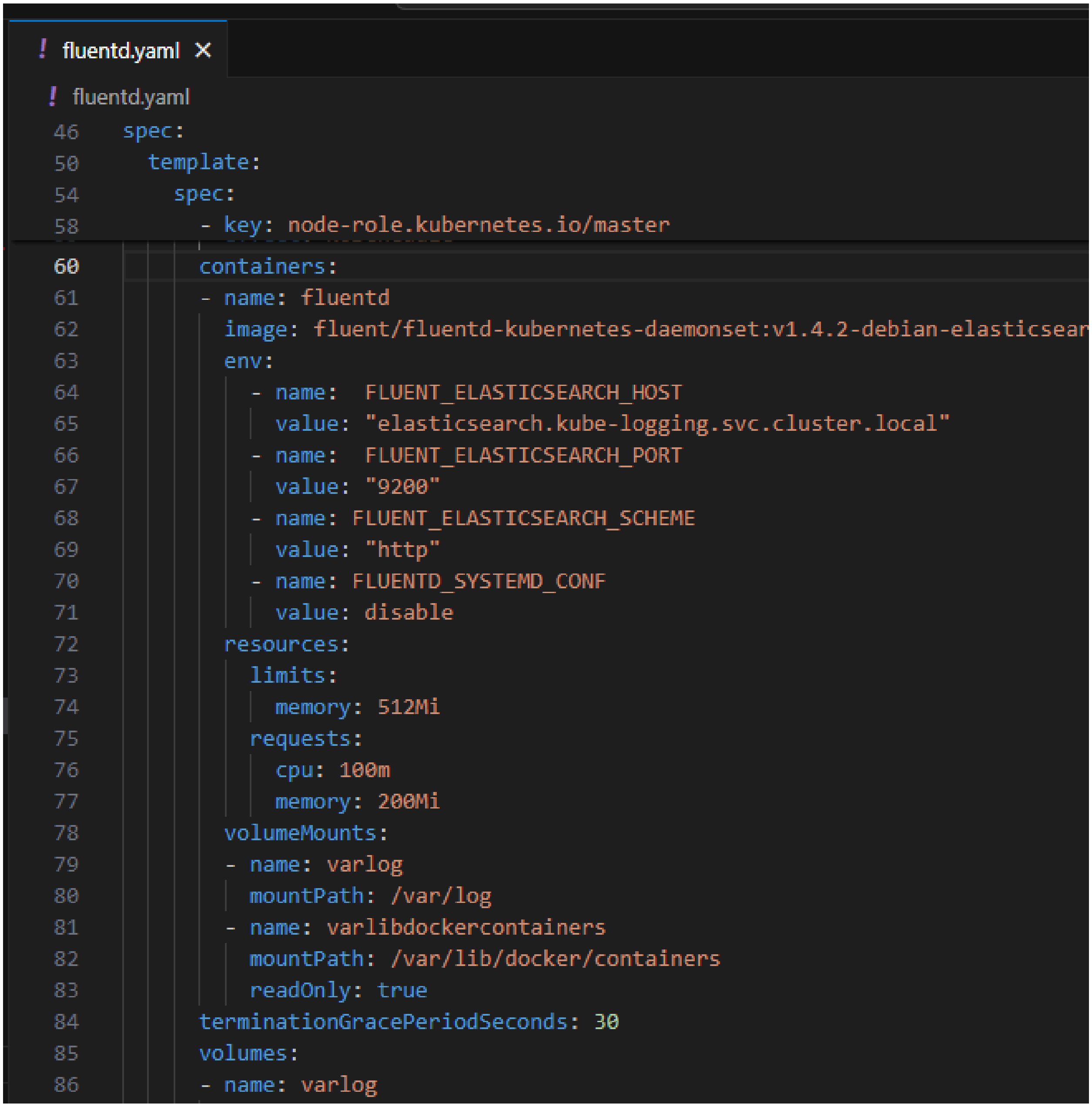
Create the file named **fluentd.yaml** and write the following configuration as:



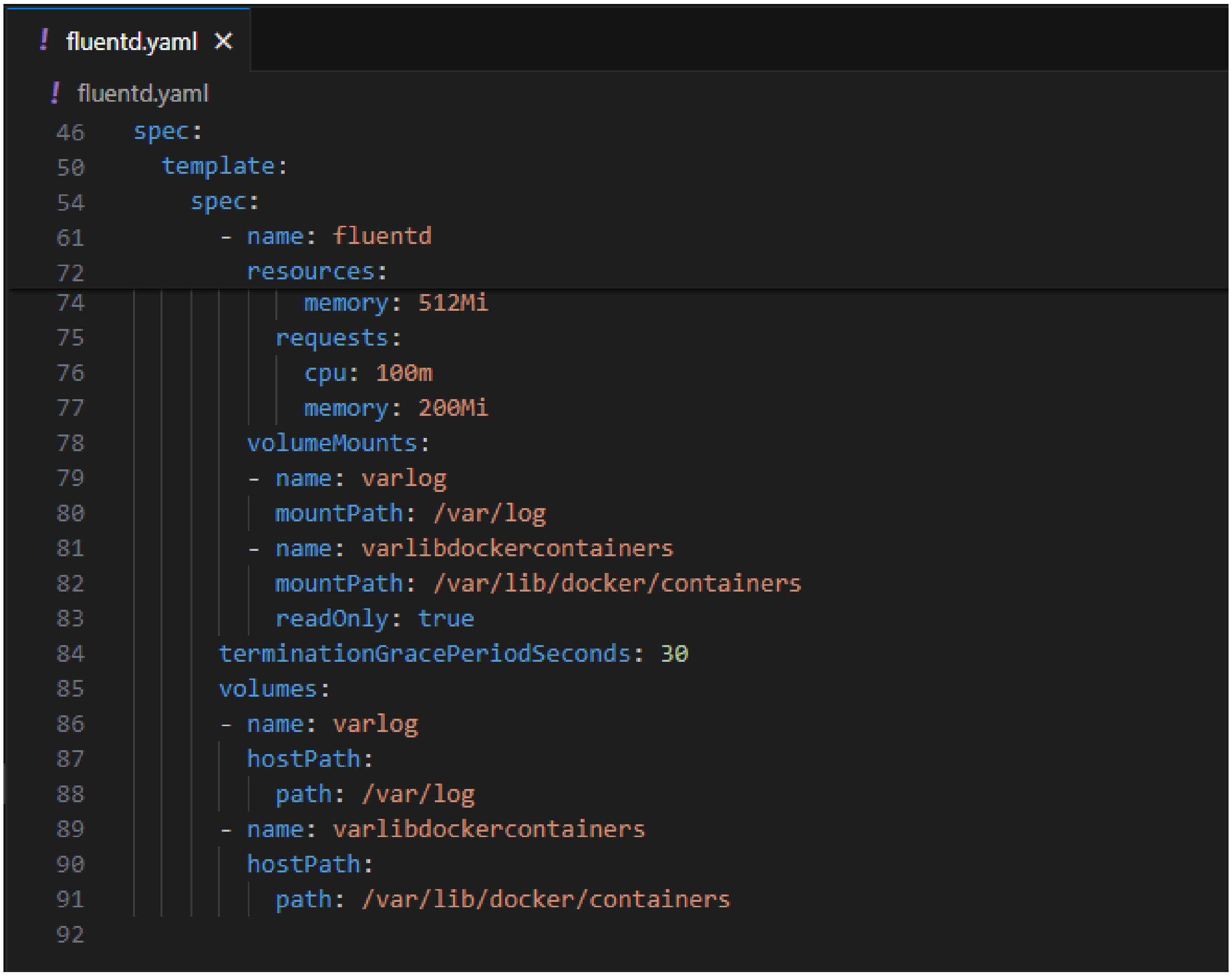
```yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  name: fluentd
  namespace: kube-logging
  labels:
    app: fluentd
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: fluentd
  labels:
    app: fluentd
rules:
- apiGroups:
  - ""
  resources:
  - pods
  - namespaces
  verbs:
  - get
  - list
  - watch
---
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: fluentd
roleRef:
  kind: ClusterRole
  name: fluentd
```



```yaml
  name: fluentd
roleRef:
  kind: ClusterRole
  name: fluentd
  apiGroup: rbac.authorization.k8s.io
subjects:
- kind: ServiceAccount
  name: fluentd
  namespace: kube-logging
---
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: fluentd
  namespace: kube-logging
  labels:
    app: fluentd
spec:
  selector:
    matchLabels:
      app: fluentd
  template:
    metadata:
      labels:
        app: fluentd
    spec:
      serviceAccount: fluentd
      serviceAccountName: fluentd
      tolerations:
      - key: node-role.kubernetes.io/master
        effect: NoSchedule
      containers:
```

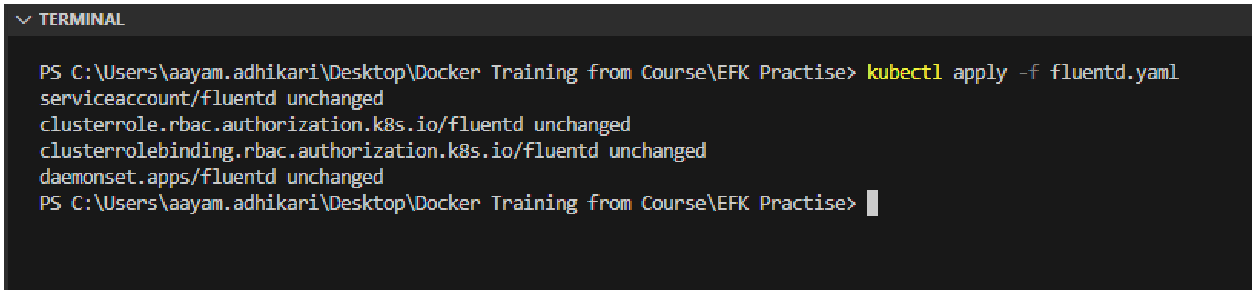


```yaml
spec:
  template:
    spec:
      - key: node-role.kubernetes.io/master
      containers:
      - name: fluentd
        image: fluent/fluentd-kubernetes-daemonset:v1.4.2-debian-elasticsear
        env:
          - name:  FLUENT_ELASTICSEARCH_HOST
            value: "elasticsearch.kube-logging.svc.cluster.local"
          - name:  FLUENT_ELASTICSEARCH_PORT
            value: "9200"
          - name: FLUENT_ELASTICSEARCH_SCHEME
            value: "http"
          - name: FLUENTD_SYSTEMD_CONF
            value: disable
        resources:
          limits:
            memory: 512Mi
          requests:
            cpu: 100m
            memory: 200Mi
        volumeMounts:
        - name: varlog
          mountPath: /var/log
        - name: varlibdockercontainers
          mountPath: /var/lib/docker/containers
          readOnly: true
      terminationGracePeriodSeconds: 30
      volumes:
      - name: varlog
```



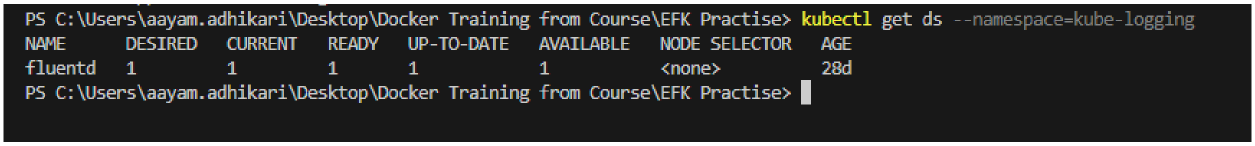
```yaml
spec:
  template:
    spec:
      - name: fluentd
        resources:
            memory: 512Mi
          requests:
            cpu: 100m
            memory: 200Mi
        volumeMounts:
        - name: varlog
          mountPath: /var/log
        - name: varlibdockercontainers
          mountPath: /var/lib/docker/containers
          readOnly: true
      terminationGracePeriodSeconds: 30
      volumes:
      - name: varlog
        hostPath:
          path: /var/log
      - name: varlibdockercontainers
        hostPath:
          path: /var/lib/docker/containers
```

Now, roll out the DaemonSet using kubectl:



```
PS C:\Users\aayam.adhikari\Desktop\Docker Training from Course\EFK Practise> kubectl apply -f fluentd.yaml
serviceaccount/fluentd unchanged
clusterrole.rbac.authorization.k8s.io/fluentd unchanged
clusterrolebinding.rbac.authorization.k8s.io/fluentd unchanged
daemonset.apps/fluentd unchanged
PS C:\Users\aayam.adhikari\Desktop\Docker Training from Course\EFK Practise>
```

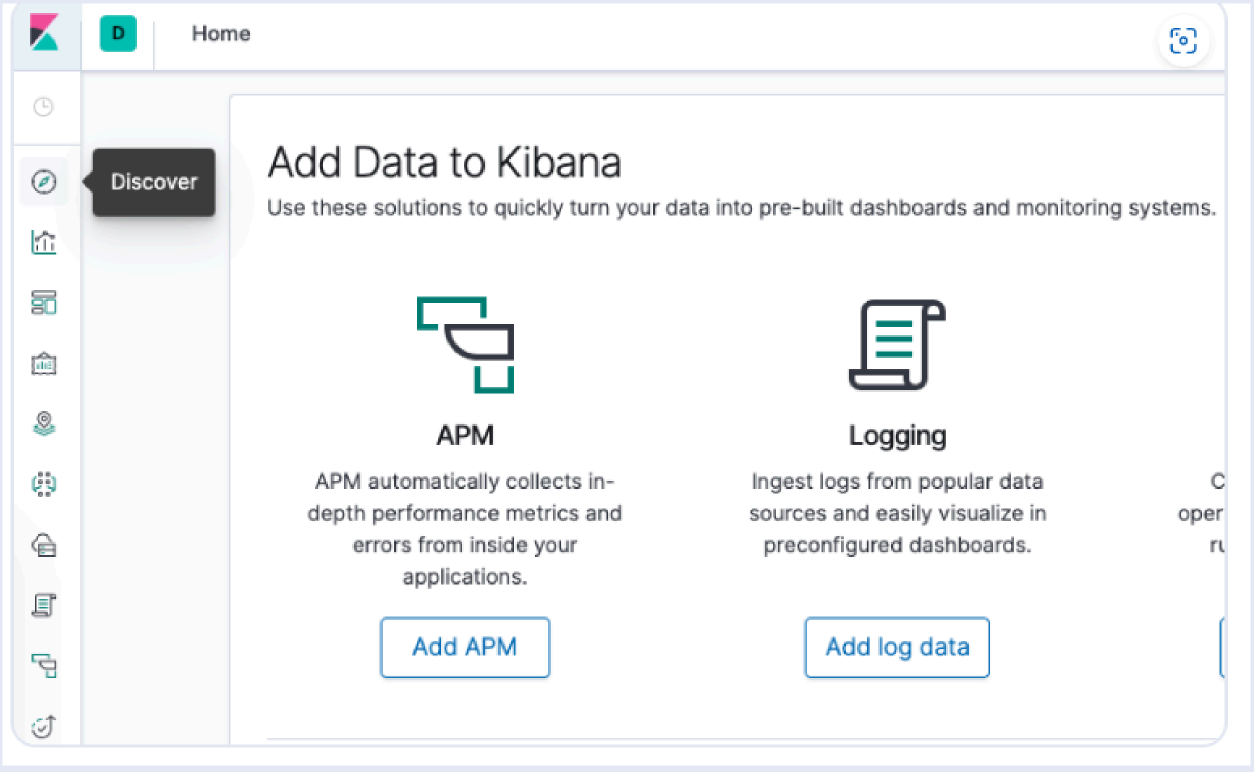
Verify that your DaemonSet rolled out successfully using **kubectl** as:



```
PS C:\Users\aayam.adhikari\Desktop\Docker Training from Course\EFK Practise> kubectl get ds --namespace=kube-logging
NAME      DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR   AGE
fluentd   1         1         1       1            1           <none>          28d
PS C:\Users\aayam.adhikari\Desktop\Docker Training from Course\EFK Practise>
```

We can now check Kibana to verify that log data is being properly collected and shipped to Elasticsearch.

With the kubectl port-forward still open, navigate to http://localhost:5601 and perform the following:



Add Data to Kibana

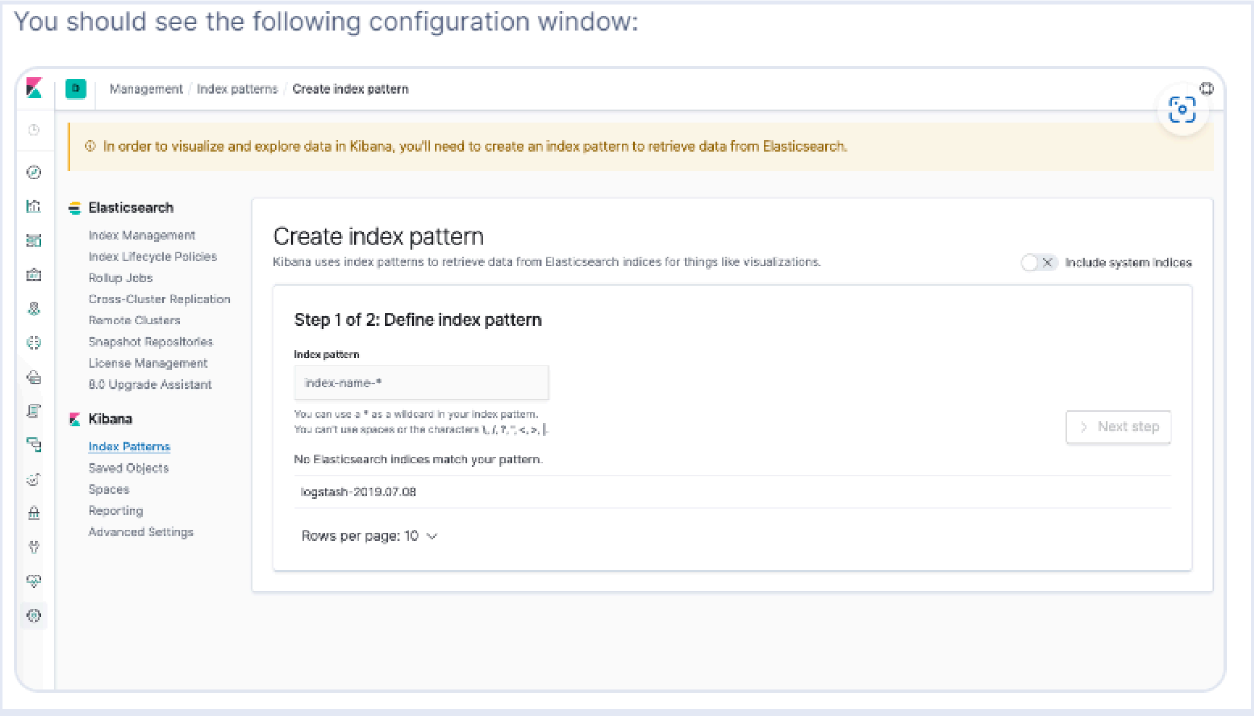
Use these solutions to quickly turn your data into pre-built dashboards and monitoring systems.

APM

APM automatically collects in-depth performance metrics and errors from inside your applications.

Logging

Ingest logs from popular data sources and easily visualize in preconfigured dashboards.



You should see the following configuration window:

Management / Index patterns / Create index pattern

ⓘ In order to visualize and explore data in Kibana, you'll need to create an index pattern to retrieve data from Elasticsearch.

Create index pattern

Kibana uses index patterns to retrieve data from Elasticsearch indices for things like visualizations.

☒ Include system indices

Step 1 of 2: Define index pattern

Index pattern

index-name-*

You can use a * as a wildcard in your index pattern.
You can't use spaces or the characters \, /, ?, ", <, >, |.

No Elasticsearch indices match your pattern.

logstash-2019.07.08

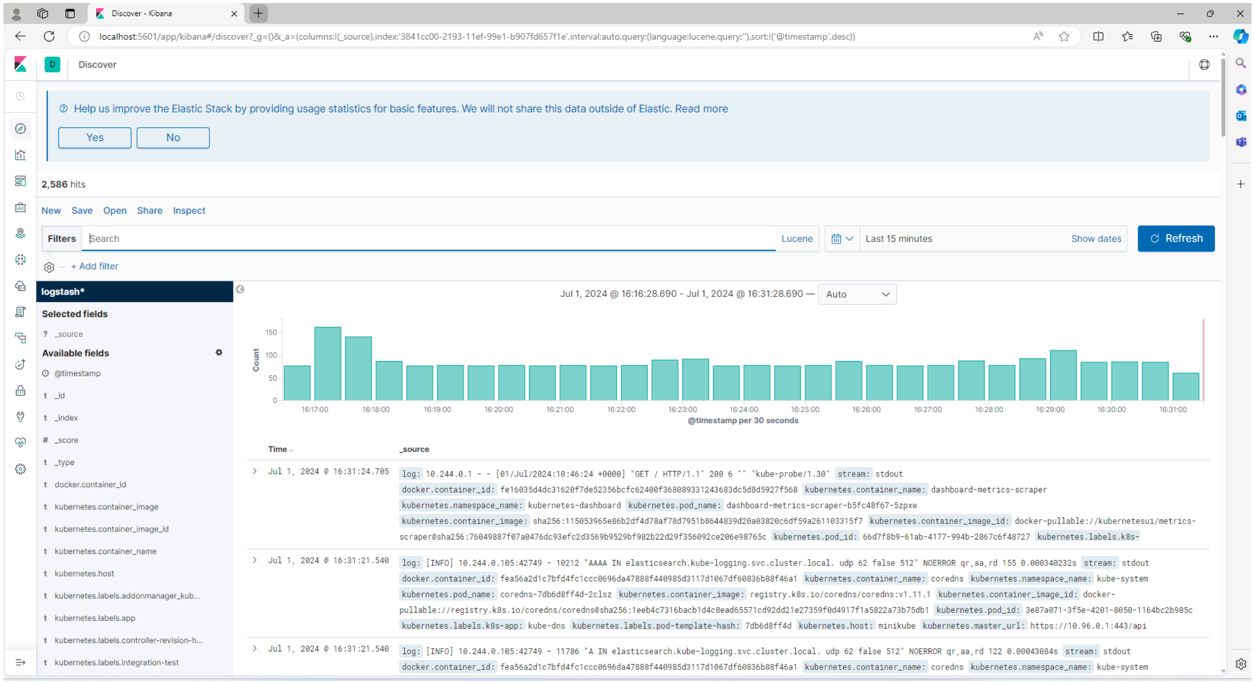
Rows per page: 10

This allows you to define the Elasticsearch indices you'd like to explore in Kibana. For now, we'll just use the **logstash-*** wildcard pattern to capture all the log data in our Elasticsearch cluster. Enter **logstash-*** in the text box and click on Next step.

This allows you to configure which field Kibana will use to filter log data by time. In the dropdown, select the **@timestamp** field, and hit Create index pattern.

Now, hit Discover in the left hand navigation menu.

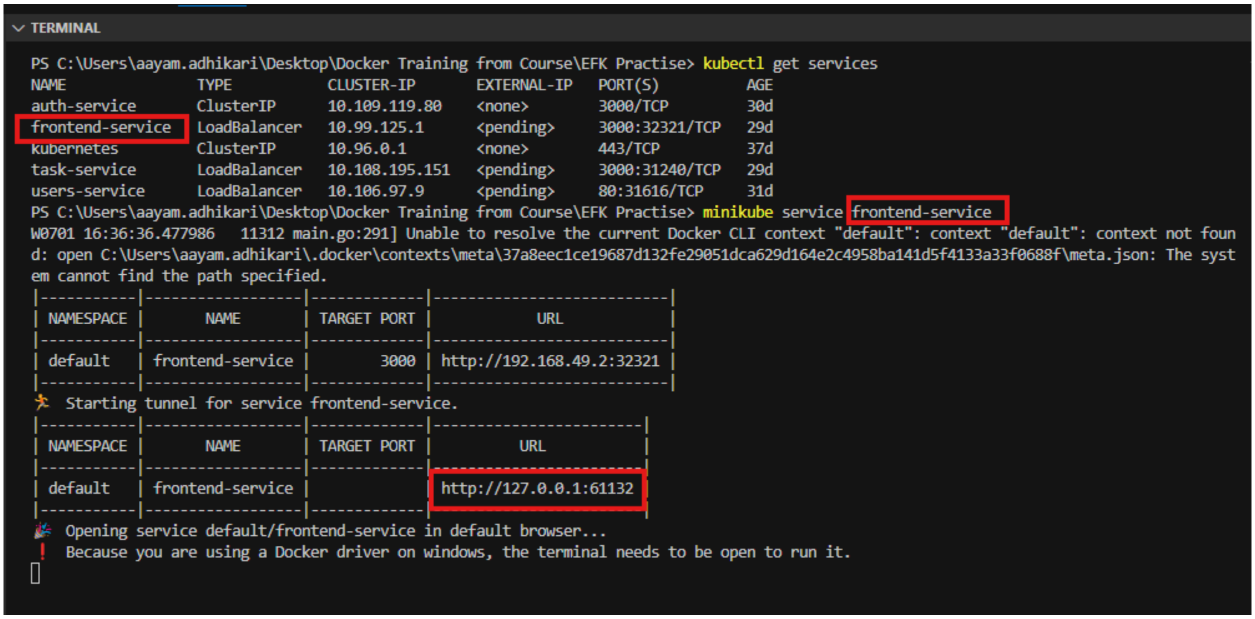
You should see a histogram graph and some recent log entries.



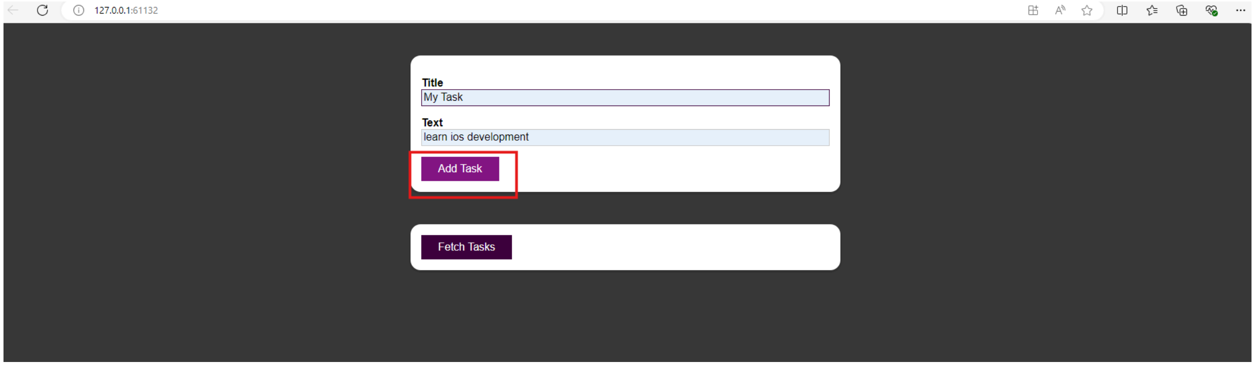
Discover - Kibana
localhost:5601/app/kibana#/discover?_g=()&_a=(columns:!(_source),index:'3841cc00-2193-11ef-99e1-b907fd657f1e',interval:auto,query:(language:lucene,query:''),sort:!('@timestamp',desc))
Discover
Help us improve the Elastic Stack by providing usage statistics for basic features. We will not share this data outside of Elastic. Read more
Yes
No
2,586 hits
New
Save
Open
Share
Inspect
Filters
Search
Lucene
Last 15 minutes
Show dates
Refresh
+ Add filter
logstash*
Selected fields
? _source
Available fields
@timestamp
t _id
t _index
# _score
t _type
t docker.container_id
t kubernetes.container_image
t kubernetes.container_image_id
t kubernetes.container_name
t kubernetes.host
t kubernetes.labels.addonmanager_kub...
t kubernetes.labels.app
t kubernetes.labels.controller-revision-h...
t kubernetes.labels.integration-test
Jul 1, 2024 @ 16:16:28.690 - Jul 1, 2024 @ 16:31:28.690 —
Auto
Count
150
100
50
0
16:17:00
16:18:00
16:19:00
16:20:00
16:21:00
16:22:00
16:23:00
16:24:00
16:25:00
16:26:00
16:27:00
16:28:00
16:29:00
16:30:00
16:31:00
@timestamp per 30 seconds
Time
_source

```
Jul 1, 2024 @ 16:31:24.705  log: 10.244.0.1 - - [01/Jul/2024:10:46:24 +0000] "GET / HTTP/1.1" 200 6 "" "kube-probe/1.30" stream: stdout
                            docker.container_id: fe16035d4dc31620f7de52356bcfc62400f368089331243683dc5d8d5927f568 kubernetes.container_name: dashboard-metrics-scraper
                            kubernetes.namespace_name: kubernetes-dashboard kubernetes.pod_name: dashboard-metrics-scraper-b5fc48f67-5zpxw
                            kubernetes.container_image: sha256:115053965e86b2df4d78af78d7951b8644839d20a03820c6df59a261103315f7 kubernetes.container_image_id: docker-pullable://kubernetesui/metrics-
                            scraper@sha256:76049887f07a0476dc93efc2d3569b9529bf982b22d29f356092ce206e98765c kubernetes.pod_id: 66d7f8b9-61ab-4177-994b-2867c6f48727 kubernetes.labels.k8s-

Jul 1, 2024 @ 16:31:21.540  log: [INFO] 10.244.0.105:42749 - 10212 "AAAA IN elasticsearch.kube-logging.svc.cluster.local. udp 62 false 512" NOERROR qr,aa,rd 155 0.000340232s stream: stdout
                            docker.container_id: fea56a2d1c7bfd4fc1ccc0696da47888f440985d3117d1067df60836b88f46a1 kubernetes.container_name: coredns kubernetes.namespace_name: kube-system
                            kubernetes.pod_name: coredns-7db6d8ff4d-2clsz kubernetes.container_image: registry.k8s.io/coredns/coredns:v1.11.1 kubernetes.container_image_id: docker-
                            pullable://registry.k8s.io/coredns/coredns@sha256:1eeb4c7316bacb1d4c8ead65571cd92dd21e27359f0d4917f1a5822a73b75db1 kubernetes.pod_id: 3e87a071-3f5e-4201-8050-1164bc2b905c
                            kubernetes.labels.k8s-app: kube-dns kubernetes.labels.pod-template-hash: 7db6d8ff4d kubernetes.host: minikube kubernetes.master_url: https://10.96.0.1:443/api

Jul 1, 2024 @ 16:31:21.540  log: [INFO] 10.244.0.105:42749 - 11786 "A IN elasticsearch.kube-logging.svc.cluster.local. udp 62 false 512" NOERROR qr,aa,rd 122 0.00043084s stream: stdout
                            docker.container_id: fea56a2d1c7bfd4fc1ccc0696da47888f440985d3117d1067df60836b88f46a1 kubernetes.container_name: coredns kubernetes.namespace_name: kube-system
```

Now, for example, to see the logs of a specific **pod (or container)**, I have created a task listing project with a frontend and a backend, which is using React and Node. I have deployed this project using Kubernetes configuration files in the minikube cluster (as you can see below), and I have opened the webapp and tried to save a task as shown:



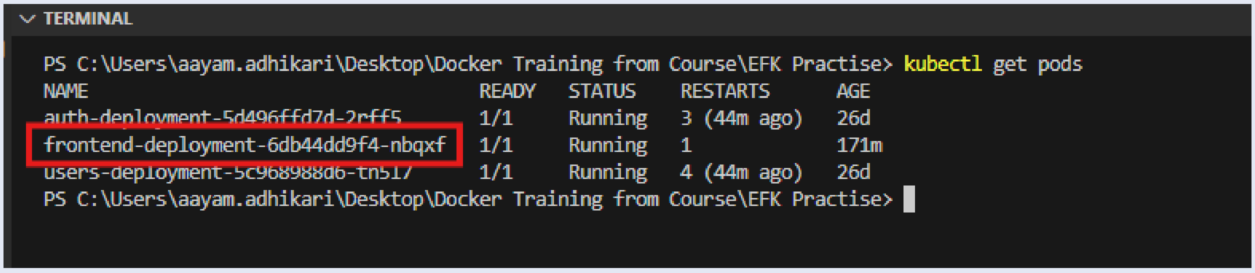
```
PS C:\Users\aayam.adhikari\Desktop\Docker Training from Course\EFK Practise> kubectl get services
NAME               TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
auth-service       ClusterIP      10.109.119.80    <none>        3000/TCP         30d
frontend-service   LoadBalancer   10.99.125.1      <pending>     3000:32321/TCP   29d
kubernetes         ClusterIP      10.96.0.1        <none>        443/TCP          37d
task-service       LoadBalancer   10.108.195.151   <pending>     3000:31240/TCP   29d
users-service      LoadBalancer   10.106.97.9      <pending>     80:31616/TCP     31d
PS C:\Users\aayam.adhikari\Desktop\Docker Training from Course\EFK Practise> minikube service frontend-service
W0701 16:36:36.477986   11312 main.go:291] Unable to resolve the current Docker CLI context "default": context "default": context not foun
d: open C:\Users\aayam.adhikari\.docker\contexts\meta\37a8eec1ce19687d132fe29051dca629d164e2c4958ba141d5f4133a33f0688f\meta.json: The syst
em cannot find the path specified.
|-----------|------------------|-------------|---------------------------|
| NAMESPACE |       NAME       | TARGET PORT |            URL            |
|-----------|------------------|-------------|---------------------------|
| default   | frontend-service |        3000 | http://192.168.49.2:32321 |
|-----------|------------------|-------------|---------------------------|
🏃  Starting tunnel for service frontend-service.
|-----------|------------------|-------------|------------------------|
| NAMESPACE |       NAME       | TARGET PORT |          URL           |
|-----------|------------------|-------------|------------------------|
| default   | frontend-service |             | http://127.0.0.1:61132 |
|-----------|------------------|-------------|------------------------|
🎉  Opening service default/frontend-service in default browser...
❗  Because you are using a Docker driver on windows, the terminal needs to be open to run it.
```



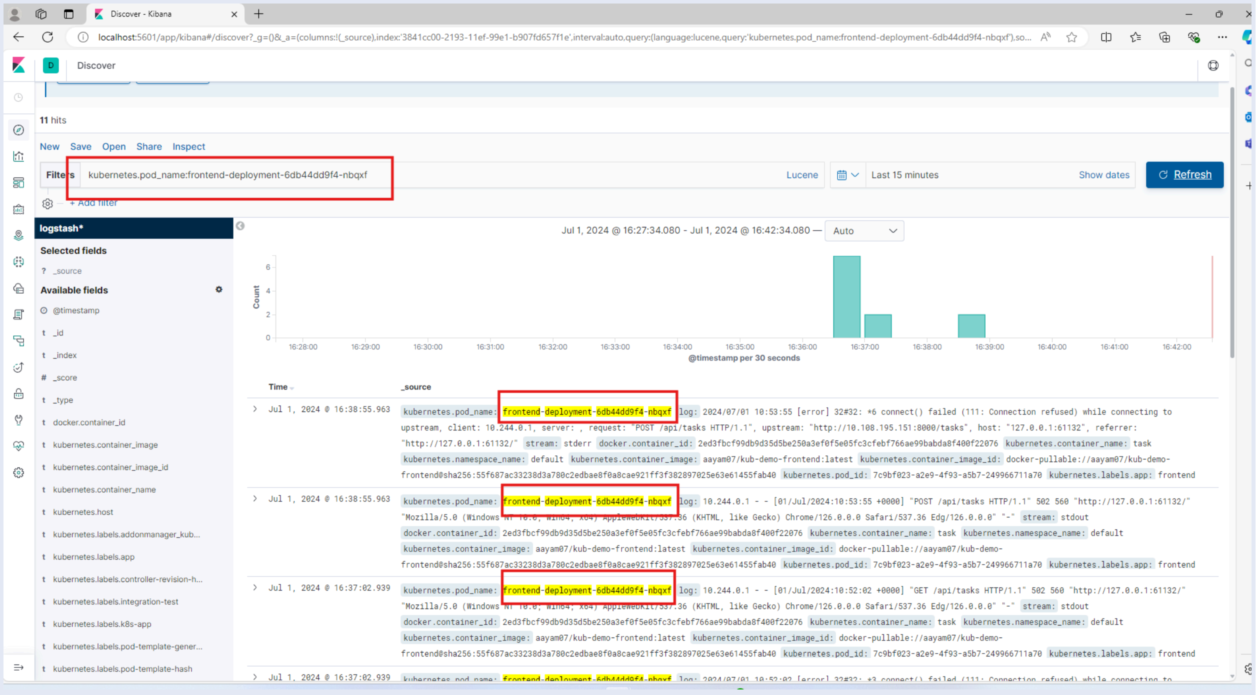
Title
My Task

Text
learn ios development

Now, navigate back to your Kibana dashboard. From the Discover page, in the search bar enter `kubernetes.pod_name:name-of-your-pod`. This filters the log data for Pods named **name-of-your-pod**. This is shown below:



```
PS C:\Users\aayam.adhikari\Desktop\Docker Training from Course\EFK Practise> kubectl get pods
NAME                                   READY   STATUS    RESTARTS      AGE
auth-deployment-5d496ffd7d-2rff5       1/1     Running   3 (44m ago)   26d
frontend-deployment-6db44dd9f4-nbqxf   1/1     Running   1             171m
users-deployment-5c968988d6-tn517      1/1     Running   4 (44m ago)   26d
PS C:\Users\aayam.adhikari\Desktop\Docker Training from Course\EFK Practise>
```


Discover - Kibana
localhost:5601/app/kibana#/discover?_g=()&_a=(columns:!(_source),index:'3841cc00-2193-11ef-99e1-b907fd657f1e',interval:auto,query:(language:lucene,query:'kubernetes.pod_name:frontend-deployment-6db44dd9f4-nbqxf'),so...
Discover
11 hits
New
Save
Open
Share
Inspect
Filters
kubernetes.pod_name:frontend-deployment-6db44dd9f4-nbqxf
Lucene
Last 15 minutes
Show dates
Refresh
+ Add filter
logstash*
Selected fields
? _source
Available fields
@timestamp
t _id
t _index
# _score
t _type
t docker.container_id
t kubernetes.container_image
t kubernetes.container_image_id
t kubernetes.container_name
t kubernetes.host
t kubernetes.labels.addonmanager_kub...
t kubernetes.labels.app
t kubernetes.labels.controller-revision-h...
t kubernetes.labels.integration-test
t kubernetes.labels.k8s-app
t kubernetes.labels.pod-template-gener...
t kubernetes.labels.pod-template-hash
Jul 1, 2024 @ 16:27:34.080 - Jul 1, 2024 @ 16:42:34.080 —
Auto
Count
6
4
2
0
16:28:00
16:29:00
16:30:00
16:31:00
16:32:00
16:33:00
16:34:00
16:35:00
16:36:00
16:37:00
16:38:00
16:39:00
16:40:00
16:41:00
16:42:00
@timestamp per 30 seconds
Time
_source

```
> Jul 1, 2024 @ 16:38:55.963  kubernetes.pod_name: frontend-deployment-6db44dd9f4-nbqxf log: 2024/07/01 10:53:55 [error] 32#32: *6 connect() failed (111: Connection refused) while connecting to
                              upstream, client: 10.244.0.1, server: , request: "POST /api/tasks HTTP/1.1", upstream: "http://10.108.195.151:8000/tasks", host: "127.0.0.1:61132", referrer:
                              "http://127.0.0.1:61132/" stream: stderr docker.container_id: 2ed3fbcf99db9d35d5be250a3ef0f5e05fc3cfebf766ae99babda8f400f22076 kubernetes.container_name: task
                              kubernetes.namespace_name: default kubernetes.container_image: aayam07/kub-demo-frontend:latest kubernetes.container_image_id: docker-pullable://aayam07/kub-demo-
                              frontend@sha256:55f687ac33238d3a780c2edbae8f0a8cae921ff3f382897025e63e61455fab40 kubernetes.pod_id: 7c9bf023-a2e9-4f93-a5b7-249966711a70 kubernetes.labels.app: frontend

> Jul 1, 2024 @ 16:38:55.963  kubernetes.pod_name: frontend-deployment-6db44dd9f4-nbqxf log: 10.244.0.1 - - [01/Jul/2024:10:53:55 +0000] "POST /api/tasks HTTP/1.1" 502 560 "http://127.0.0.1:61132/"
                              "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.0.0 Safari/537.36 Edg/126.0.0.0" "-" stream: stdout
                              docker.container_id: 2ed3fbcf99db9d35d5be250a3ef0f5e05fc3cfebf766ae99babda8f400f22076 kubernetes.container_name: task kubernetes.namespace_name: default
                              kubernetes.container_image: aayam07/kub-demo-frontend:latest kubernetes.container_image_id: docker-pullable://aayam07/kub-demo-
                              frontend@sha256:55f687ac33238d3a780c2edbae8f0a8cae921ff3f382897025e63e61455fab40 kubernetes.pod_id: 7c9bf023-a2e9-4f93-a5b7-249966711a70 kubernetes.labels.app: frontend

> Jul 1, 2024 @ 16:37:02.939  kubernetes.pod_name: frontend-deployment-6db44dd9f4-nbqxf log: 10.244.0.1 - - [01/Jul/2024:10:52:02 +0000] "GET /api/tasks HTTP/1.1" 502 560 "http://127.0.0.1:61132/"
                              "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.0.0 Safari/537.36 Edg/126.0.0.0" "-" stream: stdout
                              docker.container_id: 2ed3fbcf99db9d35d5be250a3ef0f5e05fc3cfebf766ae99babda8f400f22076 kubernetes.container_name: task kubernetes.namespace_name: default
                              kubernetes.container_image: aayam07/kub-demo-frontend:latest kubernetes.container_image_id: docker-pullable://aayam07/kub-demo-
                              frontend@sha256:55f687ac33238d3a780c2edbae8f0a8cae921ff3f382897025e63e61455fab40 kubernetes.pod_id: 7c9bf023-a2e9-4f93-a5b7-249966711a70 kubernetes.labels.app: frontend

> Jul 1, 2024 @ 16:37:02.939  kubernetes.pod_name: frontend-deployment-6db44dd9f4-nbqxf log: 2024/07/01 10:52:02 [error] 32#32: *3 connect() failed (111: Connection refused) while connecting to
```

As you can see above, we're successfully getting logs from the pod that contains the frontend React application. Similar process can be applied for any project application.