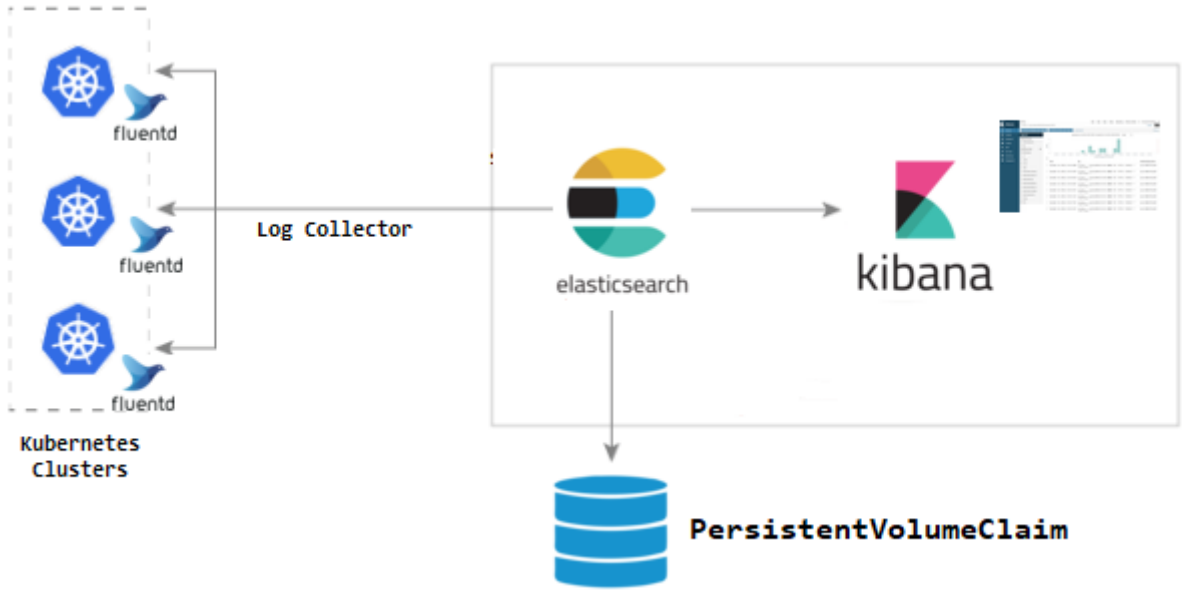
Kubernetes Advocate Follow
Jul 22, 2020 · 7 min read · · Listen

How to Deploy an EFK stack to Kubernetes



EFK

Introduction to EFK stack

When running multiple services and applications on a Kubernetes cluster, a centralized, cluster-level work stack will assist you quickly type through and analyze the serious volume of log knowledge made by your Pods. One well-liked centralized solution is that the Elasticsearch, Fluentd, and Kibana (EFK) stack.

Elasticsearch could be a period, distributed, and a scalable computer program that

Here we will use Fluentd to gather, transform, and ship logs knowledge to the Elasticsearch backend.

Hola, Let's start with the Lab :

We will be working with the list of contents :

```
Step1 — How to create a Namespace?
Step2 — How to Create the Elasticsearch StatefulSet?
Step3 — How to create Kibana deployments and services?
Step4 — How to Create the Fluentd DaemonSet in the cluster?
```

Step 1 — Creating a Namespace

So once we tend to begin with Creating cluster, we want to form namespace wherever we are going to have our work directions

To start, we want to 1st see the present Namespaces in our existing cluster kubectl:

```
kubectl get namespaces
```

Then we must always see the subsequent 3 Namespaces, which require to be pre implemented together with your K8s cluster:

```
Output
NAME          STATUS    AGE
default       Active    5m
kube-system   Active    5m
kube-public   Active    5m
```

So to make the Kube-log Namespace we'll initial open and edit a file known as `Kube-log.yaml `using nano or VI:

Inside our editor, we'll paste the desired Namespace object YAML:

```
kind: Namespace
```


```
Then, save and close the file.
```

Once we tend to have created the **Kube-log.yaml** Namespace object file, we'd like to produce the Namespace with **kubectl create** with the -f filename flag:

```
kubectl create -f kube-logging.yaml
```

We should see the subsequent output:

```
namespace/kube-log created
```

we can then ensure that the Namespace was created:

Now, we should always see the new Kube-log Namespace:

```
NAME                STATUS   AGE
default             Active  12m
kube-log            Active  2m
kube-public         Active  13m
kube-system         Active  12m
```

We can now deploy an Elasticsearch cluster here

Step 2 — How to Create the Elasticsearch StatefulSet

Now that we've created a Namespace to our stack, we will begin rolling out its various parts.

we will begin by deploying a 3-node ES cluster.

Let's start with Creating the Headless Service

we'll produce a headless Kubernetes service referred to as elastic search which will

Then, save the file.

we currently set clusterIP: None, that shows the service headless. Finally, we tend to outline ports 9200 and 9300 which can act with the API, and for inter-node communication within the cluster

Now, create the service using kubectl:


```
service/elasticsearch created
```

Finally, the service was successfully created using `kubectl get`:

```
kubectl get services --namespace=kube-logging
```

You should see the following ES now:


logging.svc.cluster.local domain for our Pods

Step 3 — How to Create the Kibana Deployment and Service

To launch Kibana on Kubernetes, we'll produce a Service known as kibana, and a readying consisting of 1 Pod duplicate. you'll scale the number of replicas looking on your production desires, and optionally specify a LoadBalancer kind for the Service to load balance requests across the readying pods.

This time, we'll produce the Service and readying within the same file. Open up a file known as kibana.yaml in your favorite editor:

Paste within the following service spec:


Then, save the file.

In this specification, we've outlined a service referred to as kibana within the Kube-logging namespace and gave it the app: kibana label.

We've conjointly specified that it ought to be accessible on port 5601 and use the app: kibana label to pick out the Service's target Pods.

In the preparation specification, we tend to outline a preparation referred to as kibana and specify that we'd like one Pod duplicate.

We use the custom .elastic.co/kibana/kibana:7.2.0 image. At now you'll substitute your non-public or public Kibana image to use.

We specify that we'd like at the least zero.1 vCPU bound to the Pod, exploding up to a limit of one vCPU. you'll amendment these parameters looking on your anticipated load and obtainable resources.

Next, we tend to use the ELASTICSEARCH_URL atmosphere variable to line the end and port for the Elasticsearch cluster. exploitation Kubernetes DNS, this end corresponds to its Service name elastic search. This domain can resolve to a listing of information processing addresses for the three Elasticsearch Pods. to be told a lot of concerning Kubernetes DNS, consult DNS for Services and Pods.

Finally, we tend to set Kibana's instrumentation port to 5601, to that the kibana Service can forward requests.

Once you are glad together with your Kibana configuration, you'll be able to roll out the Service and preparation using kubectl:

You should see the subsequent output:

```
service/kibana created deployment.apps/kibana created
```

You can ensure the rollout succeeded by running the subsequent command:


To access the Kibana interface, we will forward the local port to the Kubernetes node running Kibana. Grab the Kibana Pod details
using **kubectl get:**

```
kubectl get pods – namespace=kube-logging
```

```
kubectl port-forward kibana-9cfcnhb7-lghs2 5601:5601 —
namespace=kube-logging
```

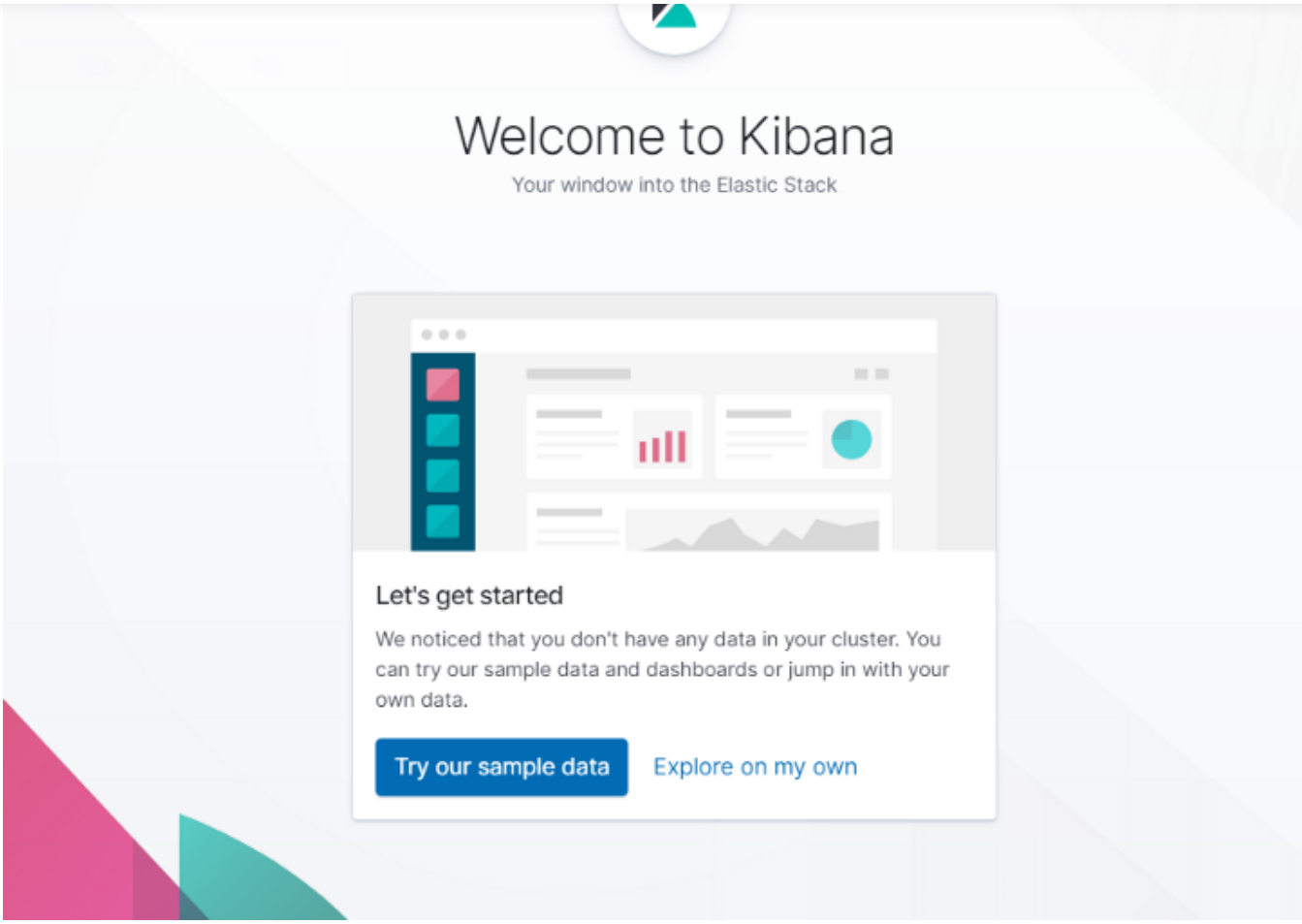
You should see the subsequent output:

```
Forwarding from 127.0.0.1:5601 -> 5601 Forwarding from [::1]:5601 ->
5601
```

visit the following URL in your web browser:

```
http://localhost:5601
```

If you see Kibana welcome page, you have correctly deployed Kibana into your Kubernetes cluster:

You can currently go to rolling out the last part of the EFK stack: the log collector, Fluentd.

Step 4 — How to Create the Fluentd DaemonSet in the cluster

Begin by gap a file referred to as fluentd.yaml in nano .we use the Fluentd DaemonSet specification provided by the Fluentd maintainers.
Another useful resource provided by the Fluentd maintainers is Kubernetes Fluentd.

First, paste in the ServiceAccount definition:

Here, we tend to produce a Service Account referred to as fluentD that the Fluentd Pods can use to access the Kubernetes API.

Next, paste in ClusterRole block:

Now, paste in the following `ClusterRoleBinding` block:

Here, we tend to outline a DaemonSet known as fluentd within the Kube-logging Namespace and provides it the app: fluentd label.

Next, paste within the following section:

Here, we tend to match the app: fluentd label in .metadata.labels and assign the DaemonSet the fluentd Service Account.

Finally, paste within the following section:

The entire Fluentd spec should reflect like this :

Now we've got finished configuring the Fluentd DaemonSet, Save it

Now, roll out the DaemonSet via kubectl:

You should see the subsequent output:

```
serviceaccount/fluentd created
clusterrole.rbac.authorization.k8s.io/fluentd created
clusterrolebinding.rbac.authorization.k8s.io/fluentd created
daemonset.extensions/fluentd created
```

Confirm that your DaemonSet extended with success via kubectl:
You should see the subsequent standing output:

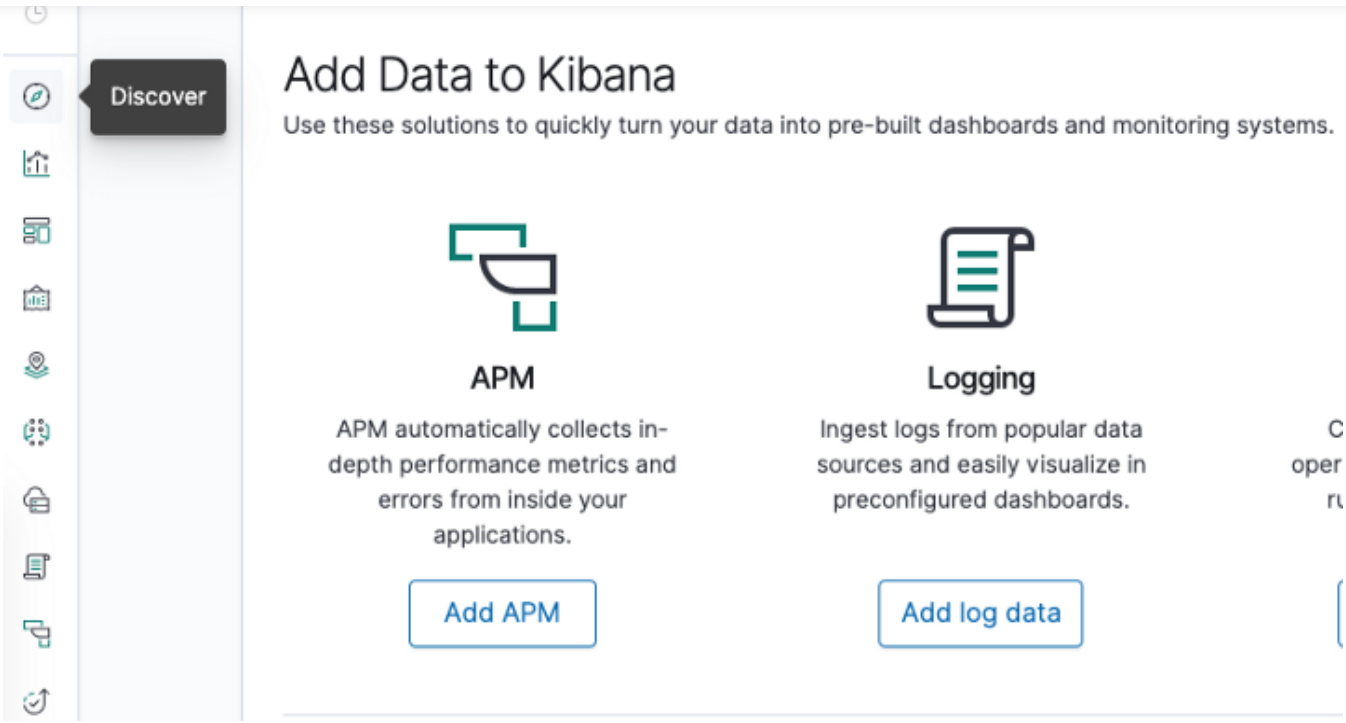
```
NAME DESIRED CURRENT READY UP-TO-DATE AVAILABLE NODE SELECTOR AGE
fluentd 3 3 3 3 3 <none> 58s
```

This currently indicates that three fluentd Pods are running within the cluster

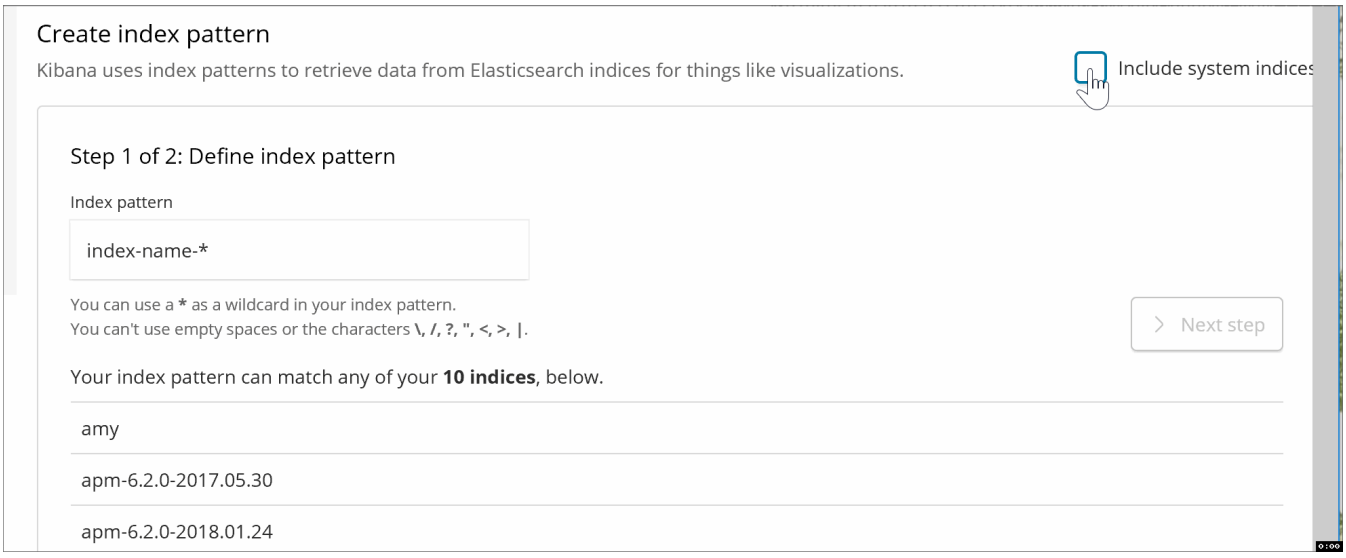
We can currently check Kibana to verify that log information or data is being properly collected and shipped to Elasticsearch.

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

Click

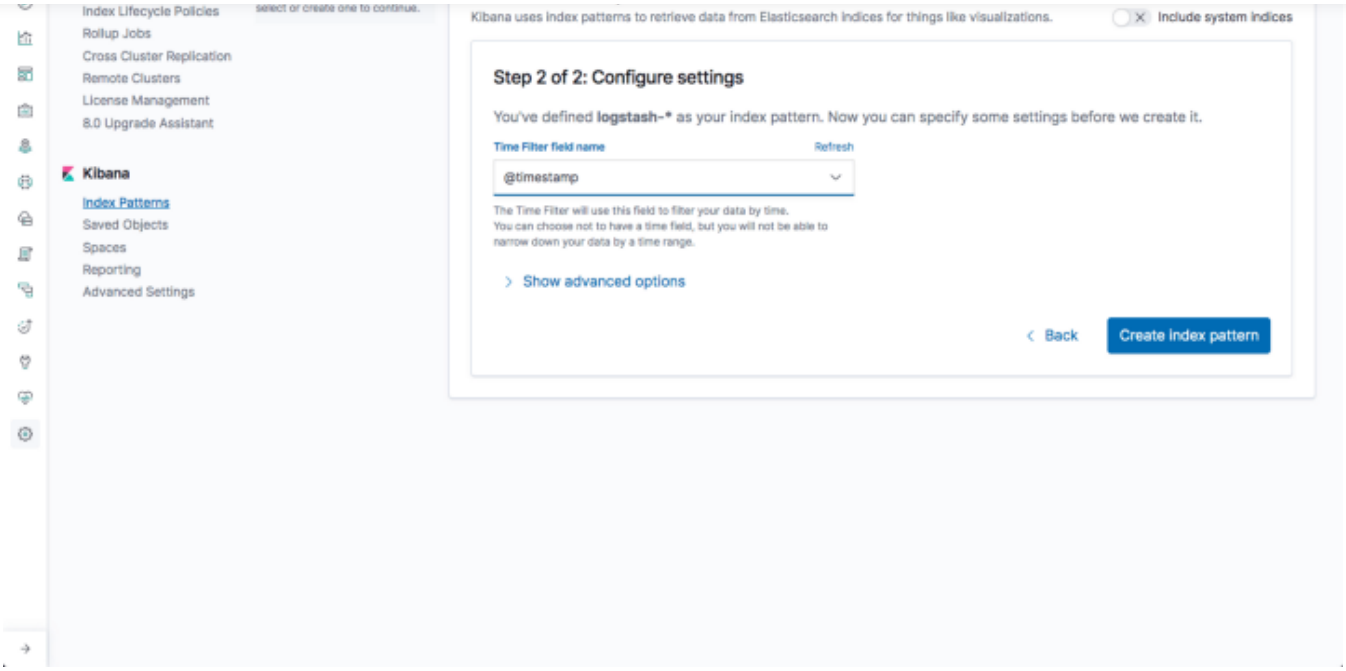



you will see this window :

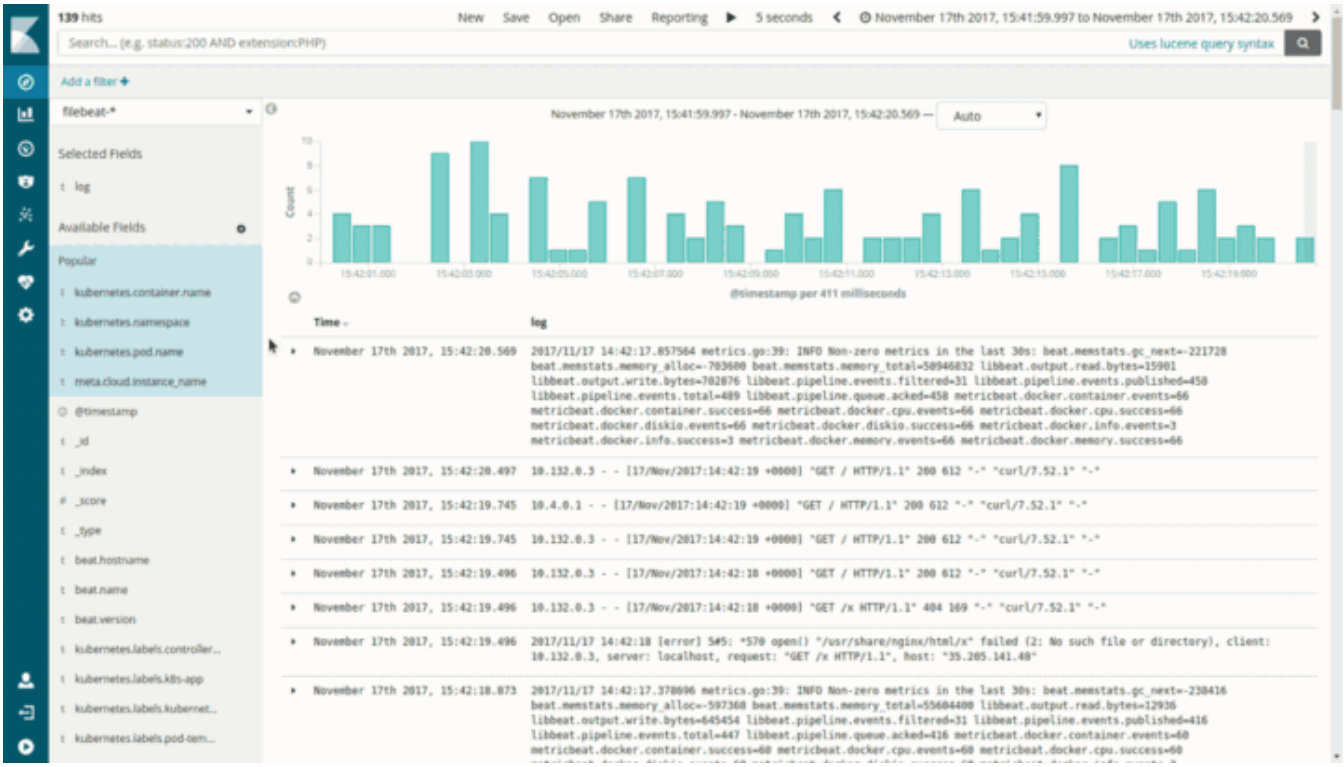


After the window, this section will appear :

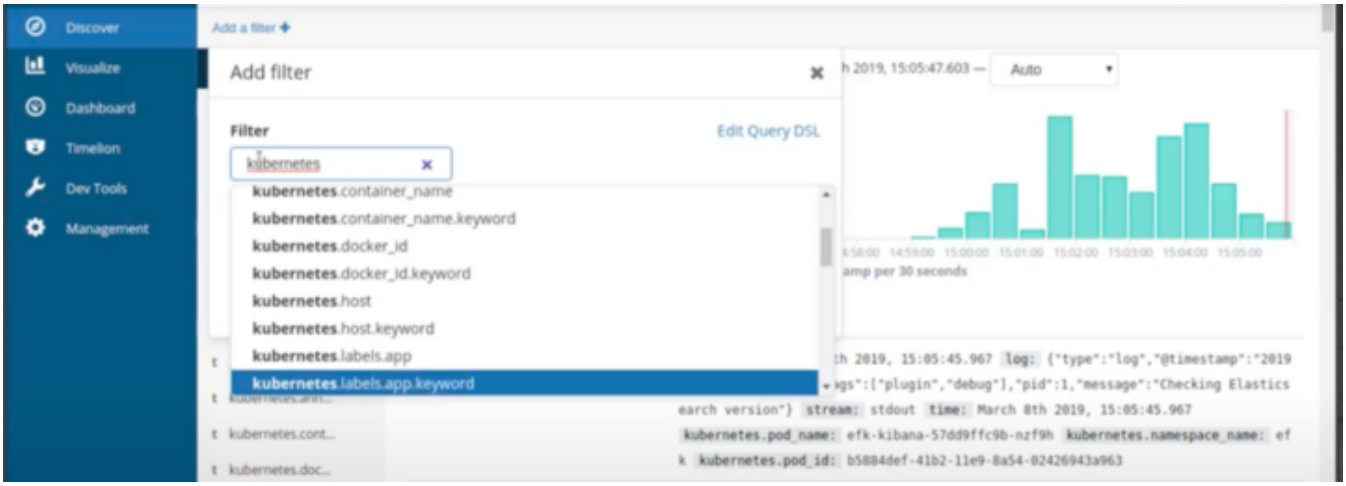
Here you will select the @timestamp field for Stack, and hit the Create index pattern as mentioned



You should see a bar chart graph and a few latest Log entries here :



Search Kubernetes here in this section below and you will see the latest logs

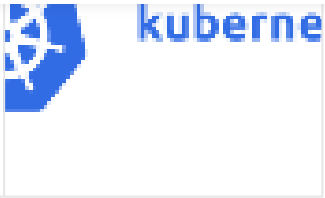


At now you've got with success designed and extended the EFK stack on your Kubernetes cluster.

Thanks

EFK stack? Like ELK? You may have heard of ELK (Elasticsearch [https://www.elastic.co/products/elasticsearch], Logstash...

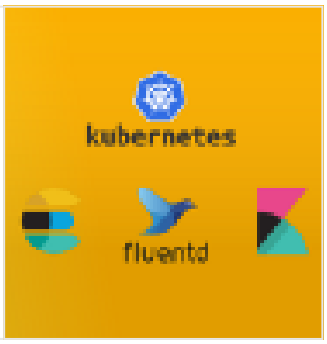
blog.ptrk.io



Installing the EFK Stack with Kubernetes with GKE | Logz.io

The ELK Stack (Elasticsearch, Logstash, and Kibana) is the weapon of choice for many Kubernetes users looking for an...

logz.io



How To Set Up an Elasticsearch, Fluentd, and Kibana (EFK) Logging Stack on Kubernetes | DigitalOcean

When running multiple services and applications on a Kubernetes cluster, a centralized, cluster-level logging stack can...

www.digitalocean.com

👋 Join us today !!

Follow us on LinkedIn, Twitter, Facebook, and Instagram

https://avmconsulting.net/