Kubernetes Logging and Monitoring: The Elasticsearch, Fluentd, and Kibana (EFK) Stack – Part 2: Elasticsearch Configuration

September 12, 2018
By Sachin Manpathak

This is a 3-part series on Kubernetes monitoring and logging:

1. *Requirements and recommended toolset*
2. *EFK Stack – Part 1: Fluentd Architecture and Configuration*
3. *EFK Stack – Part 2: Elasticsearch Configuration (this article)*

---

elasticsearch

In the previous posts in this series, we've reviewed the architecture and requirements for a logging and monitoring system for Kubernetes, as well as the configuration of fluentd, one of the components in the Elasticsearch, fluentd, Kibana (EFK) stack. In this part, we'll review the installation of the other two components: Elasticsearch and Kibana.

Elasticsearch is a robust search engine and document store. Kibana is the UI companion of Elasticsearch, simplifying visualization and querying.

From the rich feature set of Elasticsearch, the following ones are most useful for log management:

- It is schema-less, and can store data from various log sources
- It can automatically guess data types, unlike traditional data stores, and does not require schema definitions.
- It can store very large datasets (Petabyte Scale)
- It can efficiently search across a very large dataset
- It scales horizontally, and can tolerate node failures.

As you might have guessed, Elasticsearch is the most complex piece in our EFK stack for Kubernetes log aggregation and monitoring solution.

Elasticsearch architecture:

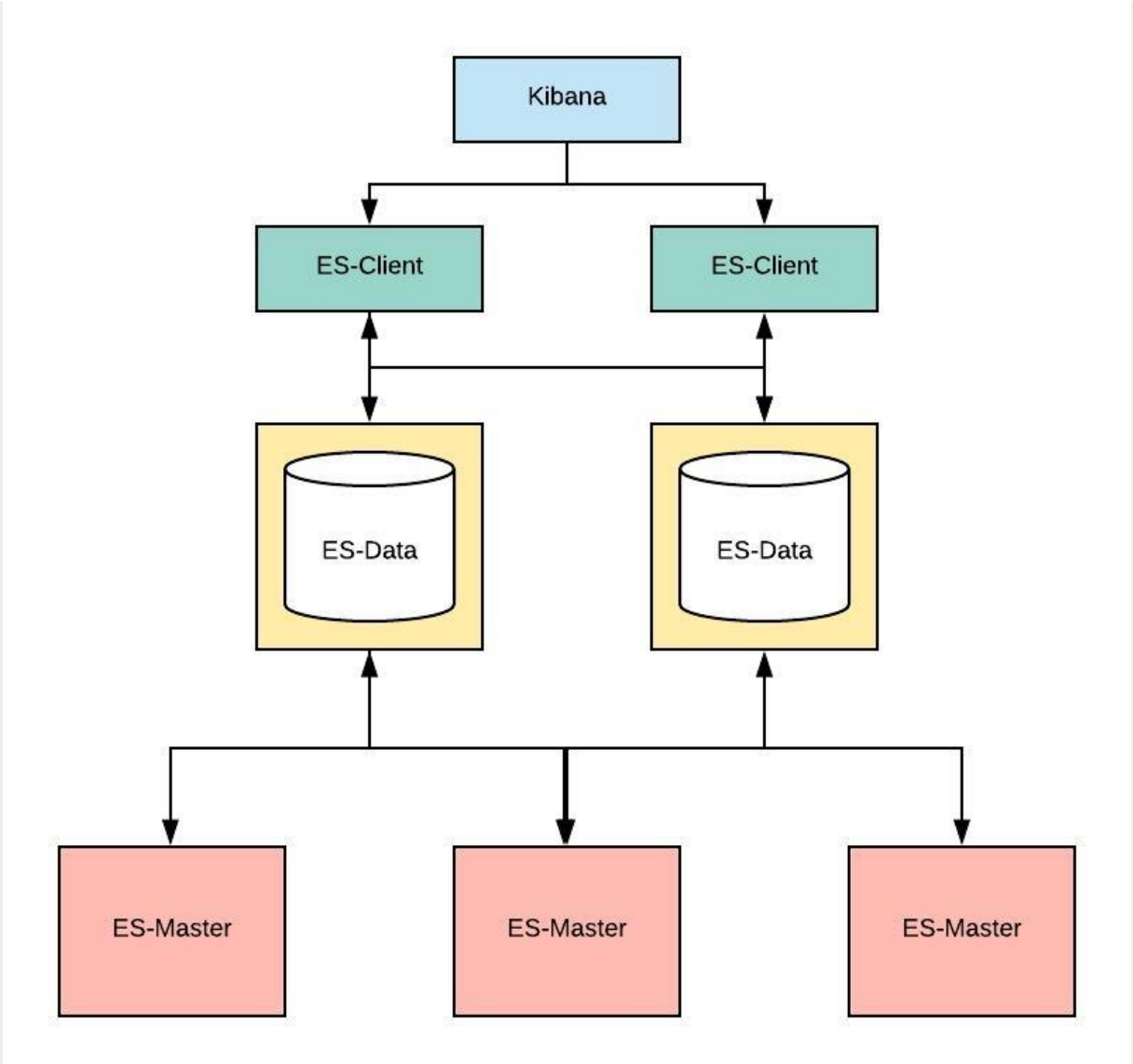
Let's review the Elasticsearch architecture and key concepts that are critical to the EFK stack deployment:

1. **Cluster:** Any non-trivial Elasticsearch deployment consists of multiple instances forming a cluster. Distributed consensus is used to keep track of master/replica relationships.
2. **Node:** A single Elasticsearch instance.
3. **Index:** A collection of documents. This is similar to a database in the traditional terminology. Each data provider (like fluentd logs from a single Kubernetes cluster) should use a **separate index** to store and search logs. An index is **stored across multiple nodes** to make data highly available.
4. **Shard:** Because Elasticsearch is a distributed search engine, an index is usually split into elements known as shards that are distributed across multiple nodes.(Elasticsearch automatically manages the arrangement of these shards. It also re-balances the shards as necessary, so users need not worry about these.)
5. **Replica:** By default, Elasticsearch creates five primary shards and one replica for each index. This means that each index will consist of five primary shards, and each shard will have one copy.

Deployment:

ElasticSearch deployment consists of three node types:

1. **Client:** These nodes provide the API endpoint and can be used for queries. In a Kubernetes-based deployment these are deployed a service so that a logical dns endpoint can be used for queries regardless of number of client nodes.
2. **Master:** These nodes provide coordination. A single master is elected at a time by using distributed consensus. That node is responsible for deciding shard placement, reindexing and rebalancing operations.
3. **Data:** These nodes store the data and inverted index. Clients query Data nodes directly. The data is sharded and replicated so that a given number of data nodes can fail, without impacting availability.



Elasticsearch Components

Installing Elasticsearch cluster with fluentd and Kibana

Now that we understand the basics of Elasticsearch, let us set up an Elasticsearch cluster with fluentd and Kibana on Kubernetes.

For this purpose, we will need a Kubernetes cluster with following capabilities.

1. Ability to run **privileged containers.**
2. **Helm and tiller** enabled.
3. Statefulsets and dynamic volume provisioning capability: Elasticsearch is deployed as **stateful set** on Kubernetes. It's best to use latest version of Kubernetes (v 1.10 as of this writing)

Step by Step Guide:

1. Make sure you have **incubator repo enabled** in helm:

```
helm repo add incubator https://kubernetes-charts-incubator.storage.g
oogleapis.com/
```

2. **Update repos:**

```
helm repo update
```

3. As a good practice, lets **keep logging services in their own namespace**:

```
kubectl create ns logging
```

4. **Install Elasticsearch**

```
helm install incubator/elasticsearch --namespace logging --name elast
icsearch --set data.terminationGracePeriodSeconds=0
```

Note that the install is customized. As Elasticsearch can use replicas, the individual processes can terminate immediately, without the risk of data loss.

5. **Install Fluentd**

```
helm install --name fluentd --namespace logging stable/fluentd-elasti
csearch --set elasticsearch.host=elasticsearch-client.logging.svc.clu
ster.local,elasticsearch.port=9200
```

Note that above command configured Fluentd so that it can send logs to right Elasticsearch endpoint. This deployment **does not use explicit authentication.** The fluentd-elasticsearch chart injects the right fluentd configuration so that it can pull logs from all containers in the Kubernetes cluster and forward them to ElasticSearch in logstash format.

6. **Install Kibana**

```
helm install --name kibana --namespace logging stable/kibana --set en
v.ELASTICSEARCH_URL=http://elasticsearch-client.logging.svc.cluster.l
ocal:9200,env.SERVER_BASEPATH=/api/v1/namespaces/logging/services/kib
ana/proxy
```

Same as with fluentd, Kibana chart variables are set to point it to the deployed Elasticsearch. The SERVER_BASEPATH is a quirk in Kibana UI. When deployed on Kubernetes, we need to **set it to an internal endpoint.**

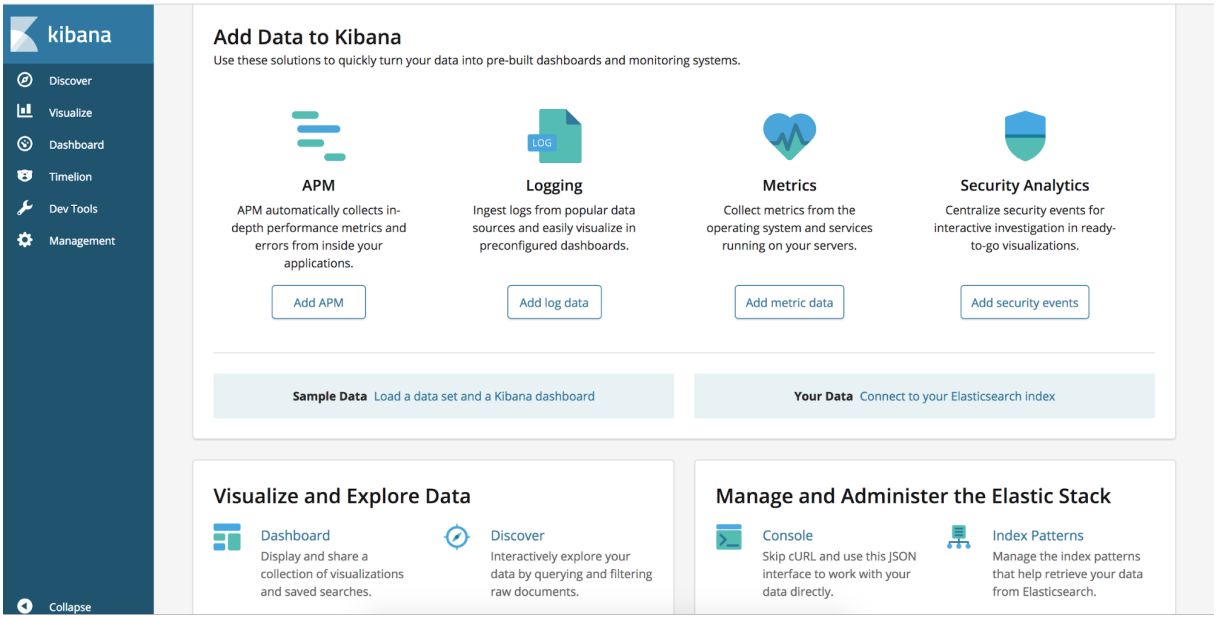
That's it!

Once these services come up, the Kibana UI can be accessed by forwarding the port of the Kibana pod to your machine or with a Kubernetes proxy command:

```
kubectl proxy 8001
```

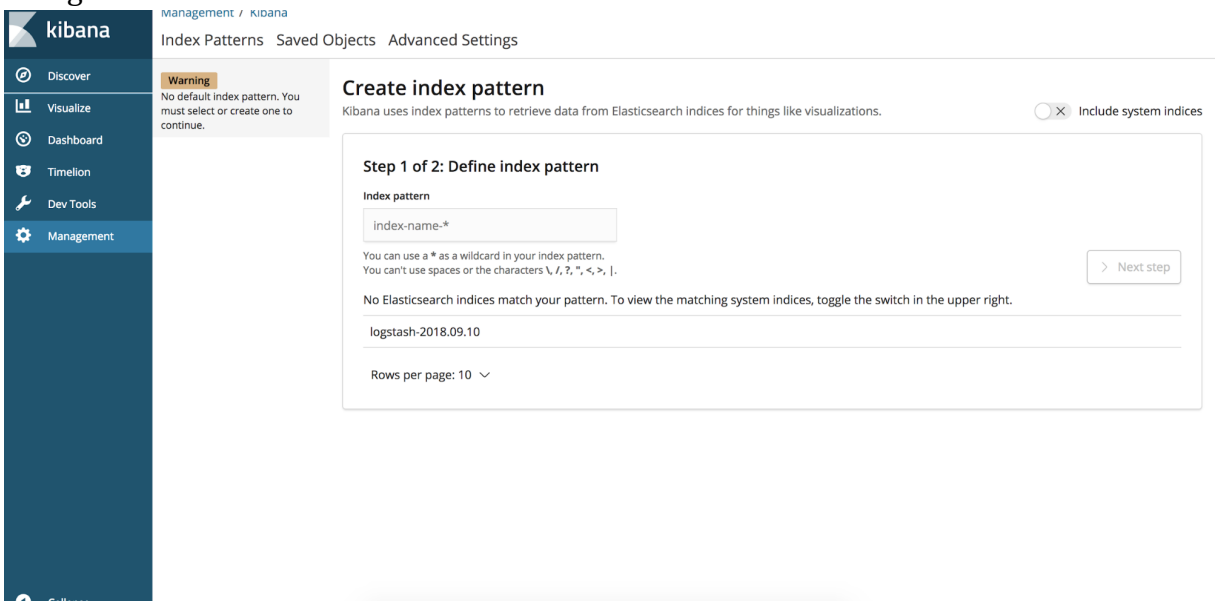
Then go to http://localhost:8001/api/v1/namespaces/logging/services/kibana:443/proxy/

You should see the following screen:

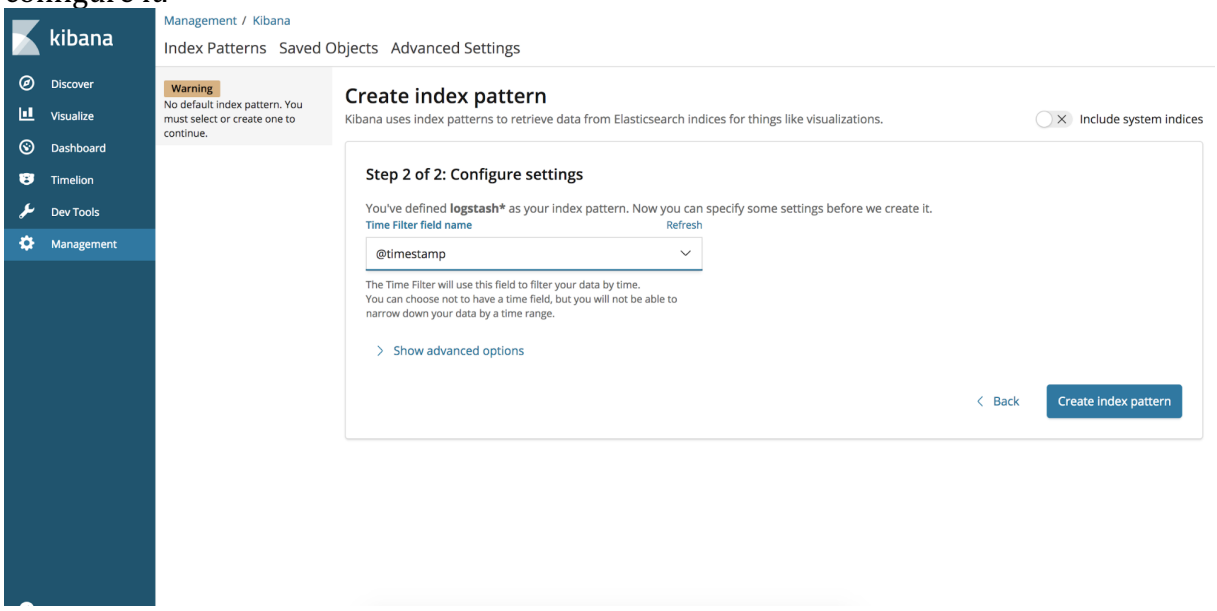


Now, let's add data for Kibana to display

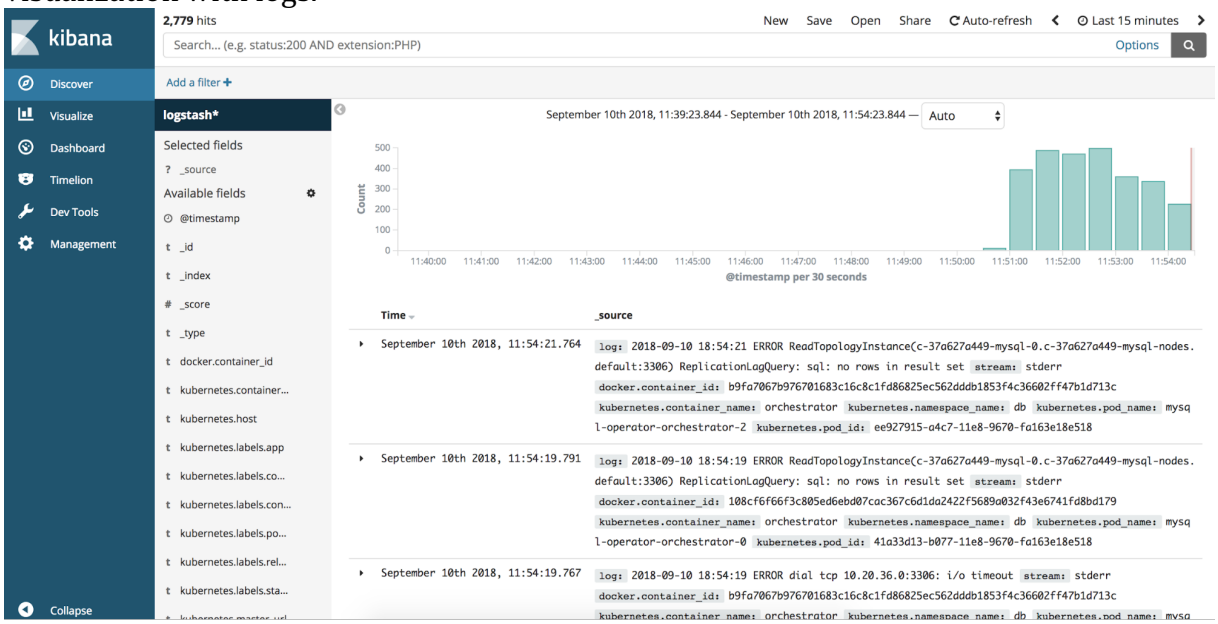
1. Navigate to “**Discover**” tab.



2. In the [previous article on fluentd](), we discussed the fluentd config to send log data in logstash **format** to Elasticsearch. The logstash format is also recognized by Kibana.
   Type **logstash*** in the index pattern dialog box.
3. Timestamp-based filtering is usually a good idea for logs, Kibana lets you configure it.



4. After creating the index pattern, navigate to “discover” tab again to view basic visualization with logs.



You’re DONE!

Congratulations! you know have a working **ELF stack for logging and monitoring of Kubernetes.**

For longer-term management we need a few more things like Cerebro, Elasticsearch Curator, and Elastalert. We will discuss those in subsequent articles.