LAB: Logging in Kubernetes with ELK (Elasticsearch, Logstash and Kibana)

Pre-requisites

For this lab you at least the following software being installed:

- Docker
- Windows Powershell
- Boxstarter
- Chocolatey

Make a copy of the sig-kubernetes/SIG02-logging repository on Github.

If you already have installed minikube, please remove all (`minikube delete`, `rd -r ~\.kube`, `rd -r -force ~\.minikube`) of it and start over.

Open a boxterstarter shell.

1. Type: `mkdir ~\.kube` (this will create a empty folder .kube in your home directory).
2. Type: `choco install minikube -y --force` (this will re-install minikube)
3. Create a new minikube cluster by typing:

```
minikube start --vm-driver hyperv --hyperv-virtual-switch "AMIS Virtual Switch" --memory
6144 --disk-size 25g --cpus 4
```

4. Retrieve the minikube ip and cluster information after succesfull installation:
    1. `minikube ip`
    2. `kubectl config use-context minikube`
    3. `kubectl cluster-info`

Kubernetes dashboard

Start a new boxtershell shell.

1. Type: `minikube addons enable dashboard`
2. Type: `minikube dashboard`

This will enable the Kubernetes dashboard and open it in a new browser window. Keep the shell open to keep the dashboard running.

Elasticsearch

Go back to the first boxstarter shell.

Navigate to the yaml subfolder.

Now we will install elasticsearch.

Type:

```
kubectl create namespace logging
kubectl create -f .\elastic.yaml -n logging --save-config=true
kubectl get pods -n logging
```

Now you will see a name which looks like: `elasticsearch-7b4cc779b8-8sds7`. Check also the status of the pod. Wait until it is fully started.

Type:

```
kubectl get service -n logging
```

Mark the port number on which elasticsearch is exposed.

> Note: In Kubernetes a service gets a random port number on creation time.

Lets try if we can access the pod using cluster ip and the exposed port number:

`curl MINIKUBE_IP:ELASTICSEARCH_EXPOSED_PORT`

Kibana

Now we will add Kibana to the cluster.

Type:

```
kubectl create -f .\kibana.yaml -n logging --save-config=true
kubectl get service -n logging
kubectl get pods -n logging
```

Wait until the pods are fully started.

Type:

```
minikube service list --namespace logging
```

You will see 2 services:

- elasticsearch
- kibana

And the URL on which they are exposed.

Start Kibana either by copying the URL and paste it in a new browser tab or type `minikube service kibana -n logging`.

Logstash and Beats

For logging there are 2 types of products in the Elastic Stack: Logstash, which is JVM based and Beats (FileBeat, MetricBeat, etc.) which are written in Go and are more lightweight.

The current best practice is to use Beats as data harvesters and Logstash for transformation and filtering.

Beats get the data from the running pods and send them to Logstash.

Filebeat is one of the best log file shippers out there today - it’s lightweight, supports SSL and TLS encryption, supports back pressure with a good built-in recovery mechanism, and is extremely reliable. It cannot, however, in most cases, turn your logs into easy-to-analyze structured log messages using filters for log enhancements. That’s the role played by Logstash.

Logstash acts as an aggregator - pulling data from various sources before pushing it down the pipeline, usually into Elasticsearch but also into a buffering component in larger production environments.

Filebeat

Navigate to the folder `hello-node-k8s`.

To be able to make the application (simple REST API) available in Kubernetes we have to make sure that the Minikube cluster can pull the image from a Docker registry.

Until now we have used public available Docker images. One way to add our own Docker image is to use the Docker registry which is part of Minikube. But this runs on the VM. So we need to push the image to the Docker registry in the VM.

One way to do this is by using the Docker daemon of Minikube.

Type the following:

```
minikube docker-env
```

You will now see something like:

```
$Env:DOCKER_TLS_VERIFY = "1"
$Env:DOCKER_HOST = "tcp://192.168.43.247:2376"
$Env:DOCKER_CERT_PATH = "C:\Users\Henk Jan van Wijk\.minikube\certs"
$Env:DOCKER_API_VERSION = "1.35"
# Run this command to configure your shell:
# & minikube docker-env | Invoke-Expression
```

As stated in the last line, type:

```
& minikube docker-env | Invoke-Expression
```

If you perform a `docker ps` you should see all kinds of Kubernetes container. Now you know you are using the correct Docker daemon.

Next we will build a new Docker container using this daemon (and this will add the build image in the Minikube Docker registry).

Type:

```
docker build -t hello-node:1.0 .
```

You can use a different version, but it should match the version in the hello-node.yml file.

We can test the created container locally by typing:

```
docker run --rm hello-node:1.0
```

This will start the Node.js/Express application and will about a log message.

You can test the api in the browser :

```
http://localhost:8080/hello/name
```

This will return a JSON response and a log message on the console.

Stop the running application using ctrl+c (this will also remove the Docker container, because of the --rm argument).

Now deploy the application to the Minikube cluster:

```
kubectl apply -f .\hello-node.yml
```

Check that a new service is added:

```
minikube service list --namespace logging
```

You can use the shown URL to test the API.

Type:

```
kubectl get pods -n logging
```

to see the name of the hello-node pod.

Type:

```
kubectl logs hello-node-848fb656b9-hdt4w -n logging
```

where `hello-node-848fb656b9-hdt4w` is the name of the pod.

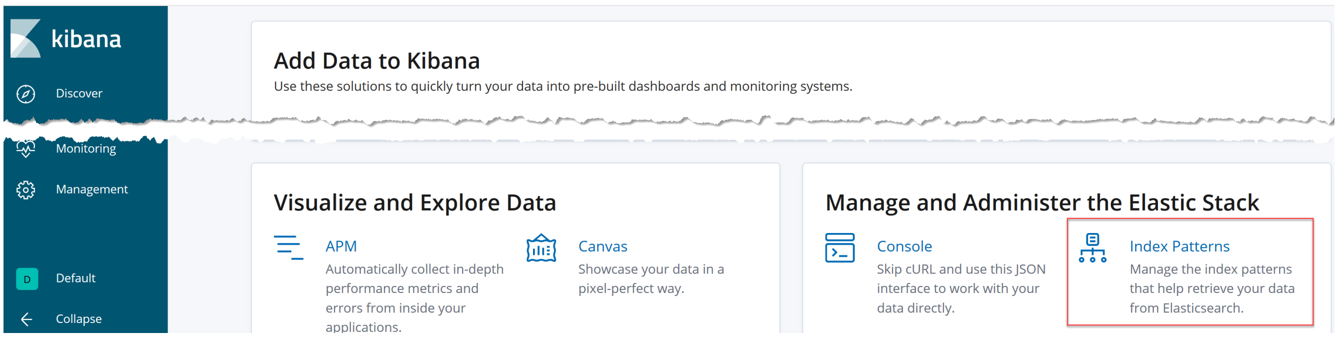
Next we will add Filebeat.

Navigate back to the `yaml` folder and type:

```
kubectl create -f .\filebeat.yaml --save-config=true
```

Open Kibana and check if there filebeat entries in Elasticsearch.

Go the home page by clicking on the Kibana logo and select Index Patterns.



Add Data to Kibana

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

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.

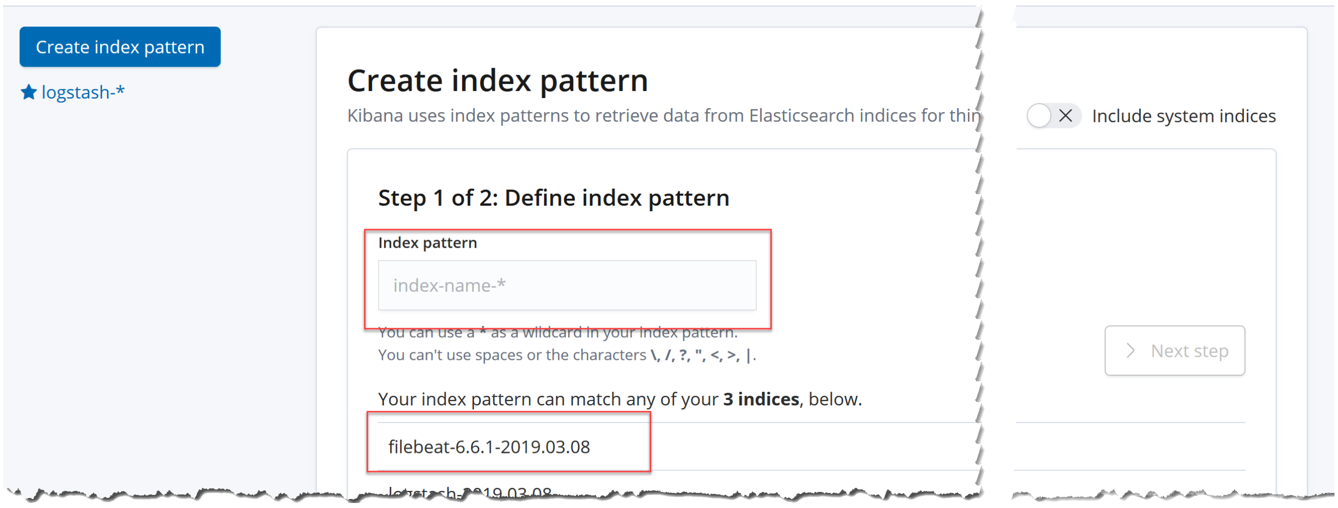
Manage and Administer the Elastic Stack

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.

Press Create index pattern.

You will see already that there is a filebeat-... index suggestion.



Create index pattern

Kibana uses index patterns to retrieve data from Elasticsearch indices for thi

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 \, /, ?, ", <, >, |.

Your index pattern can match any of your **3 indices**, below.

filebeat-6.6.1-2019.03.08

logstash-2019.03.08

Enter as index pattern: filebeat-*

Press Next step.

Select `@timestamp` as Time filter field.

Press Create index pattern.

Click on Discover and select the new index pattern.

You can use filters to query the data.

Call the hello-node API several times and see if you can find the entries in the logging.

Logstash

As we see, there are a lot of messages coming from Filebeat.

The way to fix this is to place Logstash in between.

Let's start by deploying logstash to our Kubernetes cluster:

```
kubectl create -f .\logstash.yaml --save-config=true
```

Wait until logstash is deployed:

```
kubectl get deployments -n logging
kubectl get pods -n logging
```

Now we need to change the output from Filebeat from directly sending to Elasticsearch to sending it to Logstash.

Open the `filebeat.yaml` and disable the output to Elasticsearch and check that the output to Logstash is enabled:

```
...
data:
  filebeat.yml: |-
    filebeat.config:
      inputs:
        # Mounted `filebeat-inputs` configmap:
        path: ${path.config}/inputs.d/*.yml
        # Reload inputs configs as they change:
        reload.enabled: false
      modules:
        path: ${path.config}/modules.d/*.yml
        # Reload module configs as they change:
        reload.enabled: false
    filebeat.inputs:
    - type: log
      enabled: true
      paths:
        - /var/log/*.log
    #  - logstash-tutorial.log
    output.logstash:
      hosts: ["logstash.logging:5044"]

    # To enable hints based autodiscover, remove `filebeat.config.inputs` configuration and
uncomment this:
    #filebeat.autodiscover:
    #  providers:
    #    - type: kubernetes
    #      hints.enabled: true

    processors:
      - add_cloud_metadata:
```

```
    #cloud.id: ${ELASTIC_CLOUD_ID}
    #cloud.auth: ${ELASTIC_CLOUD_AUTH}

    #output.elasticsearch:
    #  hosts: ['${ELASTICSEARCH_HOST:elasticsearch}:${ELASTICSEARCH_PORT:9200}']
    #  username: ${ELASTICSEARCH_USERNAME}
    #  password: ${ELASTICSEARCH_PASSWORD}
...
```

Save the file and apply the changes:

```
kubectl apply -f .\filebeat.yaml
```

Perform some calls to the hello-node rest service and check if you can still find them using Kibana.

You can perform queries on the Discover tab. Add some filters so you only see the messages from the hello-node service.

Filtering / enhancing logging

Within the logstash configuration we can add a filter component to extract information from the message into separate tags.

Uncomment the sample in the `logstash.yml` file and apply the changes.

```
   filter {
      grok {
        match => {
          "message" => "\[%{GREEDYDATA:timestamp}\]\ \[%{WORD:loglevel}\]\ %{WORD:logtype}\
-\ %{GREEDYDATA:msg}"
        }
      }
    }
```

Perform some new requests on the hello-node rest api and check the results in Kibana.

See http://grokconstructor.appspot.com if you want some help constructing grok match expressions.