YAML Files in the ELK Stack

Of course, you might also be here because you are trying to keep your YAML configurations straight specifically for the ELK Stack (or another monitoring tool, whether or not related to Docker and/or Kubernetes).

The *`elasticsearch.yml`* file—like similar files in the ELK Stack and Beats—will be by default located in different places depending on the way you install ELK. In general, this is where you will find them:

Linux:

```
/etc/elasticsearch/elasticsearch.yml
/etc/kibana/kibana.yml
/etc/filebeat/filebeat.yml
/etc/metricbeat/metricbeat.yml
```

Configure the elasticsearch.yml File

To configure *`elasticsearch.yml`*, enter the file.

```
sudo vim /etc/elasticsearch/elasticsearch.yml
```

Copy

Hit i in order to edit the file. You will want to uncomment lines for the following fields:

1. `cluster.name`
2. `node.name`
3. `path.data`
4. `path.logs`
5. `network.host #Depending on your situation, this should usually be 127.0.0.1`

6. http.port 9200 *(When you uncomment this, make sure this is already set at 9200. Otherwise, set it yourself and type it in.)*

Advanced YAML: Elasticsearch Cluster Configuration

You will need more advanced settings for Elasticsearch clusters, including disabling swapping unused memory.

```
bootstrap.memory_lock: true OR bootstrap.mlockall: true
```
Copy

And:

```
MAX_LOCKED_MEMORY=unlimited
```
Copy

Configure the kibana.yml File

Besides the default file locations mentioned above, if you installed Kibana from a tar.gz or .zip distribution, look for the file in KIBANA_HOME/config.

Uncomment the following lines and/or make sure the settings match:

```
server.port: 5601
elasticsearch.url: "http://localhost:9200"
```

An alternative *kibana.yml* example might look like this:

```
server.port: 127.0.0.1:5601
elasticsearch.url: "http://elasticsearch:9200"
```

In general, Elasticsearch should be located at localhost:9200 in all ELK Stack configuration files for system-hosted ELK, unless of course you have a different location.

Intermediate/Advanced Kibana configurations

You can point to an X.509 server certificate and their respective private keys using different options. These configurations are possible for both Elasticsearch input and Kibana itself. Both sets of configurations, however, would be in the *`kibana.yml`* configuration file.

Kibana uses all of these options to validate certificates and create a chain of trust with SSL/TLS connections from end users coming into Kibana.

For Kibana:

```
server.ssl.keystore.path:
```

Copy

For Elasticsearch:

```
elasticsearch.ssl.keystore.path:
```

Alternatively, the server.ssl.certificate and server.ssl.key configurations can be used. These together function as an alternative because they *cannot be used in conjunction with the server.ssl.keystore.path* configuration.

For Kibana:

```
server.ssl.certificate:
```

```
server.ssl.key:
```

For Elasticsearch:

```
elasticsearch.ssl.certificate:
```

```
elasticsearch.ssl.key:
```

Copy

Elasticsearch or Kibana will use these chains, respectively, when PKI authentication is active.

Additionally, you can enable the following configuration to encrypt the respective ssl.key configs:

For Kibana:

```
server.ssl.keyPassphrase:
```

For Elasticsearch:

```
elasticsearch.ssl.keyPassphrase:
```

Configure the logstash.conf and logstash.yml Files

You will mainly configure Logstash in its .conf file, which is in JSON. However, the *logstash.yml* file is still relevant.

The logstash.conf file is actually in JSON. However, while this post obviously focuses on YAML configurations, it would be a disservice not to include the basics for the .conf file.

Here is a basic Logstash configuration example for the file's three main sections: input, filter, and output:

logstash.conf Configuration

```
input {

  file {
```

```
        path => "/var/log/apache2/access.log"
start_position => "beginning"
    sincedb_path => "/dev/null"
  }
}
filter {
    grok {
      match => { "message" => "%{COMBINEDAPACHELOG}" }
    }
    date {
    match => [ "timestamp" , "dd/MMM/yyyy:HH:mm:ss Z" ]
  }
  geoip {
      source => "clientip"
    }
}
output {
  elasticsearch {
    hosts => ["localhost:9200"]
}
}
```

Copy

logstash.yml Configuration

Specify Logstash modules:

```
modules:

  - name: MODULE_NAME1

    var.PLUGIN_TYPE1.PLUGIN_NAME1.KEY1: VALUE

    var.PLUGIN_TYPE1.PLUGIN_NAME1.KEY2: VALUE

    var.PLUGIN_TYPE2.PLUGIN_NAME2.KEY1: VALUE

    var.PLUGIN_TYPE3.PLUGIN_NAME3.KEY1: VALUE

  - name: MODULE_NAME2

    var.PLUGIN_TYPE1.PLUGIN_NAME1.KEY1: VALUE

    var.PLUGIN_TYPE1.PLUGIN_NAME1.KEY2: VALUE
```

Configure the filebeat.yml Files

Configure filebeat.inputs for type: log. Identify separate paths for each kind of log (Apache2, nginx, MySQL, etc.)

```
filebeat.inputs:

- type: log

  #Change value to true to activate the input configuration

  enabled: false

  paths:

    - "/var/log/apache2/*"
```

```
    - "/var/log/nginx/*"
    - "/var/log/mysql/*"
```

Copy

Then define processors *within* filebeat.inputs. This example defines the *drop_fields* processor:

```
filebeat.inputs:
- type: log
  paths:
- "/var/log/apache2/access.log"
  fields:
    apache: true
  processors:
  - drop_fields:
      fields: ["verb","id"]
```

Then define the Filebeat output. Uncomment or set the outputs for Elasticsearch or Logstash:

```
output.elasticsearch:
  hosts: ["localhost:9200"]


output.logstash:
  hosts: ["localhost:5044"]
```

Configuring Filebeat on Docker

The most common method to configure Filebeat when running it as a Docker container is by bind-mounting a configuration file when running said container. To do this, create a new *filebeat.yml* file on your host. This example is for a locally hosted version of Docker:

```
filebeat.inputs:

- type: log

  paths:

  - '/var/lib/docker/containers/*/*.log'

  json.message_key: log

  json.keys_under_root: true

  processors:

  - add_docker_metadata: ~


output.elasticsearch:

  hosts: ["localhost:9200"]
```

To see further examples of advanced Filebeat configurations, check out our other Filebeat tutorials::

What is Filebeat Autodiscover?

Using the Filebeat Wizard in Logz.io

Musings in YAML—Tips for Configuring Your Beats

Configure the metricbeat.yml File

metricbeat.yml will list a number of modules (Apache, system, nginx, etc.). Make sure to identify the module, the metricsets, the interval, processes, hosts and enabled: true.

Here is an example configuration of two modules in Metricbeat, one for your system and another for Apache metrics:

```
metricbeat.modules:
- module: system
  metricsets: ["cpu","memory","network"]
  enabled: true
  period: 15s
  processes: ['.*']
- module: apache
  metricsets: ["status"]
  enabled: true
period: 5s
  hosts: ["http://172.20.11.7"]
```

If you are setting up a Metricbeat Docker module, it’s advisable to mark the following metricsets:

metricsets: [“container”, “cpu”, “diskio”, “healthcheck”, “info”, “memory”, “network”]

Metricbeat Output

Just as with Filebeat, uncomment or set the outputs for Elasticsearch or Logstash:

```
output.elasticsearch:
  hosts: ["localhost:9200"]

output.logstash:
  hosts: ["localhost:5044"]
```