**SElastic Stack on Docker**

**Introduction**

Elastic Stack (**ELK**) Docker Composition, preconfigured with **Security**, **Monitoring**, and **Tools**; Up with a Single Command.

Suitable for Demoing, MVPs and small production deployments.

Stack Version: [8.3.2](https://www.elastic.co/blog/whats-new-elastic-8-3-0) 🎉 - Based on [Official Elastic Docker Images](https://www.docker.elastic.co/)

You can change Elastic Stack version by setting ELK\_VERSION in .env file and rebuild your images. Any version >= 8.0.0 is compatible with this template.

**Main Features 📜**

* Configured as a Production Single Node Cluster. (With a multi-node cluster option for experimenting).
* Security Enabled By Default.
* Configured to Enable:
  + Logging & Metrics Ingestion
  + APM
  + Alerting
  + Machine Learning
  + SIEM
  + Enabling Trial License
* Use Docker-Compose and .env to configure your entire stack parameters.
* Persist Elasticsearch's Keystore and SSL Certifications.
* Self-Monitoring Metrics Enabled.
* Prometheus Exporters for Stack Metrics.
* Collect Docker Host Logs to ELK via make collect-docker-logs.
* Embedded Container Healthchecks for Stack Images.
* [Rubban](https://github.com/sherifabdlnaby/rubban) for Kibana curating tasks.

**More points**

And comparing Elastdocker and the popular [deviantony/docker-elk](https://github.com/deviantony/docker-elk)

One of the most popular ELK on Docker repositories is the awesome [deviantony/docker-elk](https://github.com/deviantony/docker-elk). Elastdocker differs from deviantony/docker-elk in the following points.

* Security enabled by default using Basic license, not Trial.
* Persisting data by default in a volume.
* Run in Production Mode (by enabling SSL on Transport Layer, and add initial master node settings).
* Persisting Generated Keystore, and create an extendable script that makes it easier to recreate it every-time the container is created.
* Parameterize credentials in .env instead of hardcoding elastich:changeme in every component config.
* Parameterize all other Config like Heap Size.
* Add recommended environment configurations as Ulimits and Swap disable to the docker-compose.
* Make it ready to be extended into a multinode cluster.
* Configuring the Self-Monitoring and the Filebeat agent that ship ELK logs to ELK itself. (as a step to shipping it to a monitoring cluster in the future).
* Configured tools and Prometheus Exporters.
* The Makefile that simplifies everything into some simple command

**Requirements**

* [Docker 20.05 or higher](https://docs.docker.com/install/)
* [Docker-Compose 1.29 or higher](https://docs.docker.com/compose/install/)
* 4GB RAM (For Windows and MacOS make sure Docker's VM has more than 4GB+ memory.)

**Setup**

1. Clone the Repository

git clone https: <https://github.com/amanshrivastava1410/Continous-Monitoring-on-Docker>

Initialize Elasticsearch Keystore and TLS Self-Signed Certificates

$ make setup

**For Linux's docker hosts only**. By default virtual memory [is not enough](https://www.elastic.co/guide/en/elasticsearch/reference/current/vm-max-map-count.html) so run the next command as root sysctl -w vm.max\_map\_count=262144

1. Start Elastic Stack

$ make elk <OR> $ docker-compose up -d <OR> $ docker compose up -d

1. Visit Kibana at [https://localhost:5601](https://localhost:5601/) or https://<your\_public\_ip>:5601

Default Username: elastic, Password: changeme

* + Notice that Kibana is configured to use HTTPS, so you'll need to write https:// before localhost:5601 in the browser.
  + Modify .env file for your needs, most importantly ELASTIC\_PASSWORD that setup your superuser elastic's password, ELASTICSEARCH\_HEAP & LOGSTASH\_HEAP for Elasticsearch & Logstash Heap Size.

Whatever your Host (e.g AWS EC2, Azure, DigitalOcean, or on-premise server), once you expose your host to the network, ELK component will be accessible on their respective ports. Since the enabled TLS uses a self-signed certificate, it is recommended to SSL-Terminate public traffic using your signed certificates.

🏃🏻‍♂️ To start ingesting logs, you can start by running make collect-docker-logs which will collect your host's container logs.

**Additional Commands**

Expand

**Configuration**

* Some Configuration are parameterized in the .env file.
  + ELASTIC\_PASSWORD, user elastic's password (default: changeme *pls*).
  + ELK\_VERSION Elastic Stack Version (default: 8.3.2)
  + ELASTICSEARCH\_HEAP, how much Elasticsearch allocate from memory (default: 1GB -good for development only-)
  + LOGSTASH\_HEAP, how much Logstash allocate from memory.
  + Other configurations which their such as cluster name, and node name, etc.
* Elasticsearch Configuration in elasticsearch.yml at ./elasticsearch/config.
* Logstash Configuration in logstash.yml at ./logstash/config/logstash.yml.
* Logstash Pipeline in main.conf at ./logstash/pipeline/main.conf.
* Kibana Configuration in kibana.yml at ./kibana/config.
* Rubban Configuration using Docker-Compose passed Environment Variables.

**Setting Up Keystore**

You can extend the Keystore generation script by adding keys to ./setup/keystore.sh script. (e.g Add S3 Snapshot Repository Credentials)

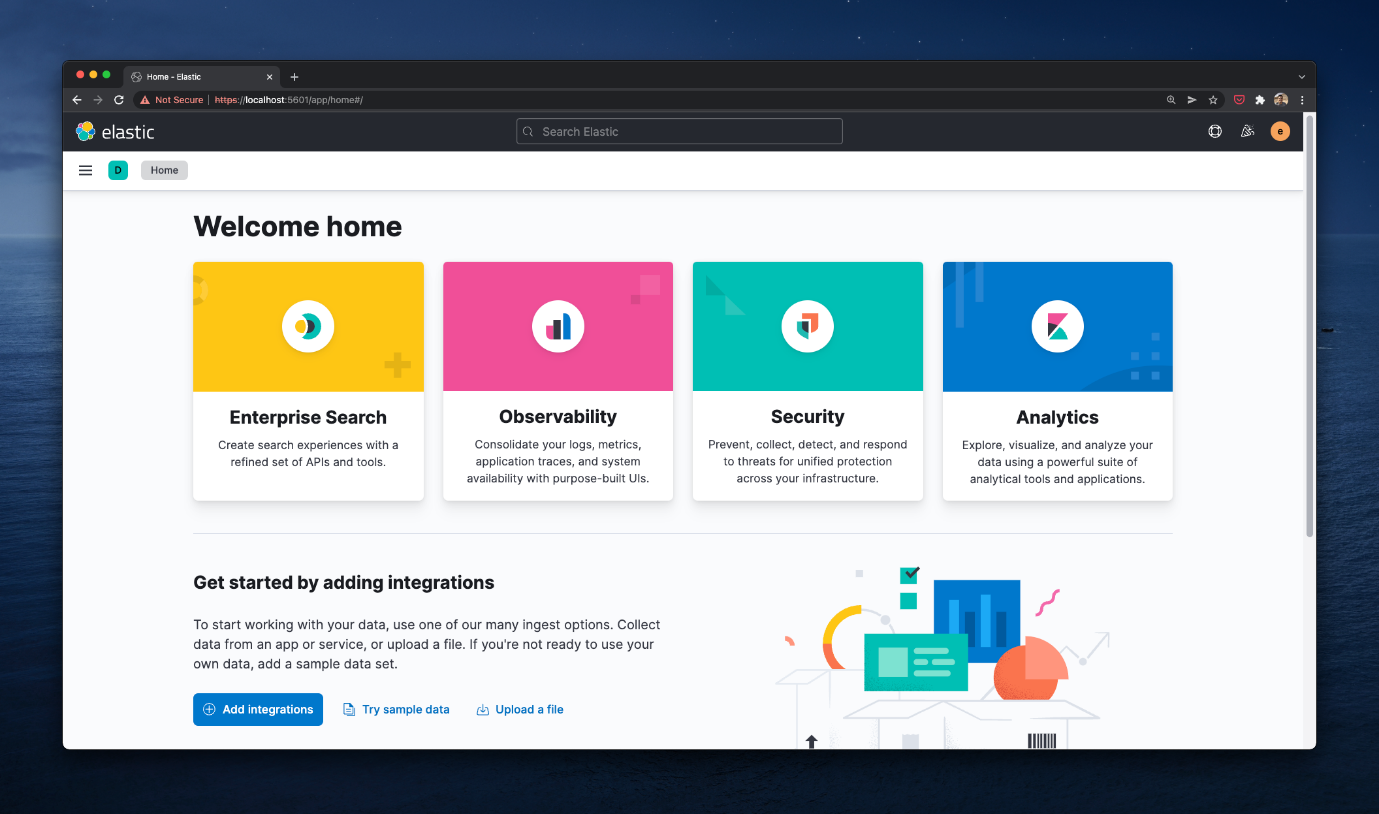
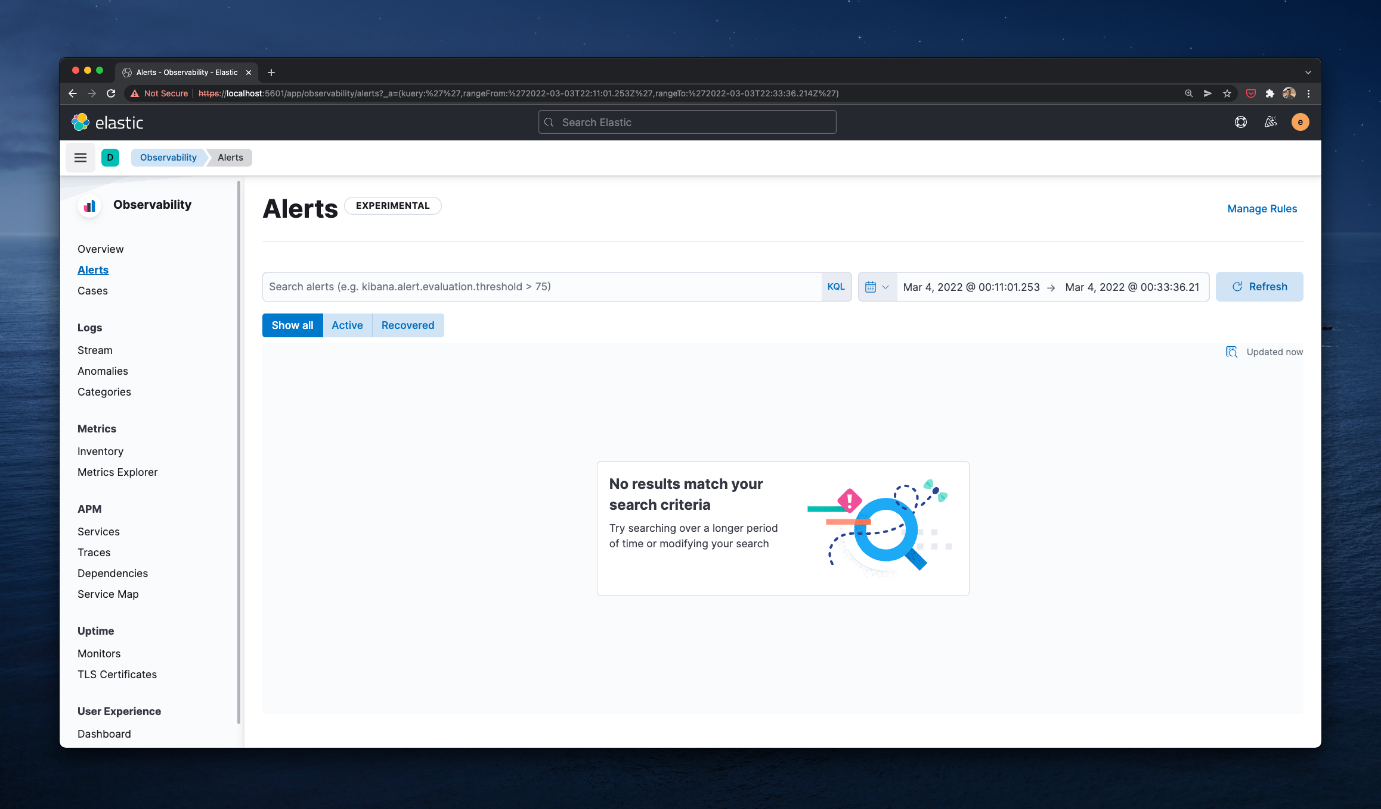
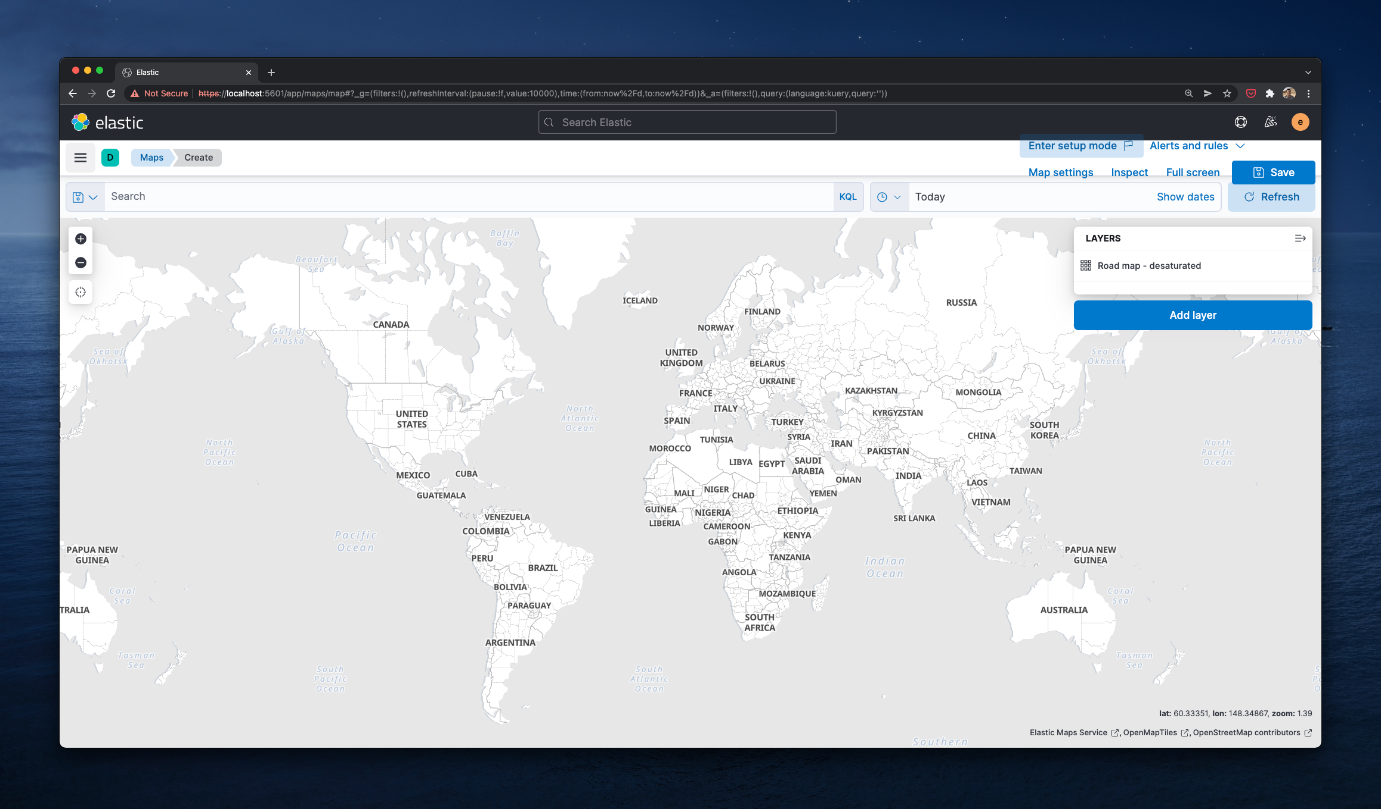
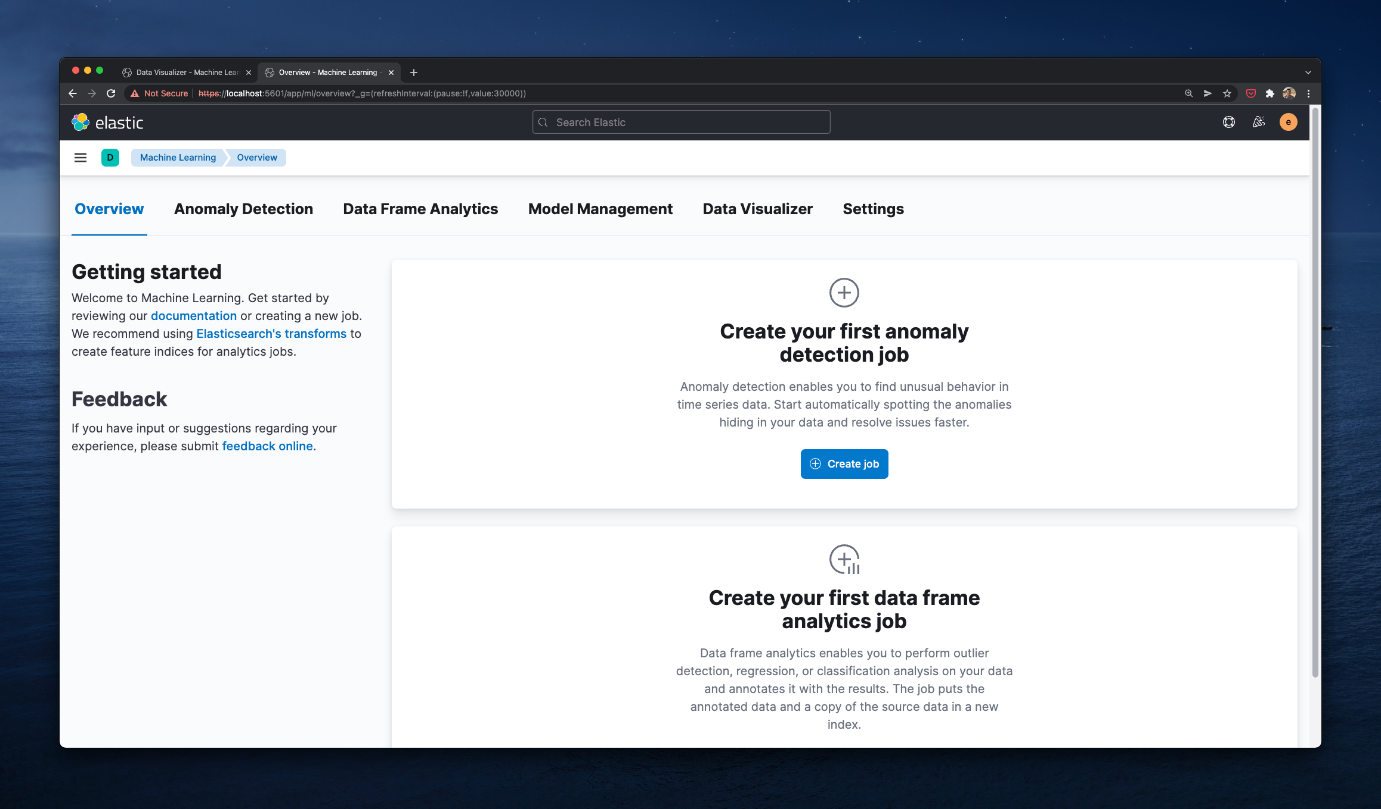
To Re-generate Keystore:

make keystore

**Notes**

* ⚠️ Elasticsearch HTTP layer is using SSL, thus mean you need to configure your elasticsearch clients with the CA in secrets/certs/ca/ca.crt, or configure client to ignore SSL Certificate Verification (e.g --insecure in curl).
* Adding Two Extra Nodes to the cluster will make the cluster depending on them and won't start without them again.
* Makefile is a wrapper around Docker-Compose commands, use make help to know every command.
* Elasticsearch will save its data to a volume named elasticsearch-data
* Elasticsearch Keystore (that contains passwords and credentials) and SSL Certificate are generated in the ./secrets directory by the setup command.
* Make sure to run make setup if you changed ELASTIC\_PASSWORD and to restart the stack afterwards.
* For Linux Users it's recommended to set the following configuration (run as root)
* sysctl -w vm.max\_map\_count=262144

By default, Virtual Memory [is not enough](https://www.elastic.co/guide/en/elasticsearch/reference/current/vm-max-map-count.html).

[](https://user-images.githubusercontent.com/16992394/156664447-c24c49f4-4282-4d6a-81a7-10743cfa384e.png) [](https://user-images.githubusercontent.com/16992394/156664848-d14f5e58-8f80-497d-a841-914c05a4b69c.png) [](https://user-images.githubusercontent.com/16992394/156664562-d38e11ee-b033-4b91-80bd-3a866ad65f56.png) [](https://user-images.githubusercontent.com/16992394/156664695-5c1ed4a7-82f3-47a6-ab5c-b0ce41cc0fbe.png)

**Working with Elastic APM**

After completing the setup step, you will notice a container named apm-server which gives you deeper visibility into your applications and can help you to identify and resolve root cause issues with correlated traces, logs, and metrics.

**Authenticating with Elastic APM**

In order to authenticate with Elastic APM, you will need the following:

* The value of ELASTIC\_APM\_SECRET\_TOKEN defined in .env file as we have [secret token](https://www.elastic.co/guide/en/apm/guide/master/secret-token.html) enabled by default
* The ability to reach port 8200
* Install elastic apm client in your application e.g. for NodeJS based applications you need to install [elastic-apm-node](https://www.elastic.co/guide/en/apm/agent/nodejs/master/typescript.html)
* Import the package in your application and call the start function, In case of NodeJS based application you can do the following:

const apm = require('elastic-apm-node').start({

serviceName: 'foobar',

secretToken: process.env.ELASTIC\_APM\_SECRET\_TOKEN,

// https is enabled by default as per elastdocker configuration

serverUrl: 'https://localhost:8200',

})

Make sure that the agent is started before you require any other modules in your Node.js application - i.e. before express, http, etc. as mentioned in [Elastic APM Agent - NodeJS initialization](https://www.elastic.co/guide/en/apm/agent/nodejs/master/express.html#express-initialization)

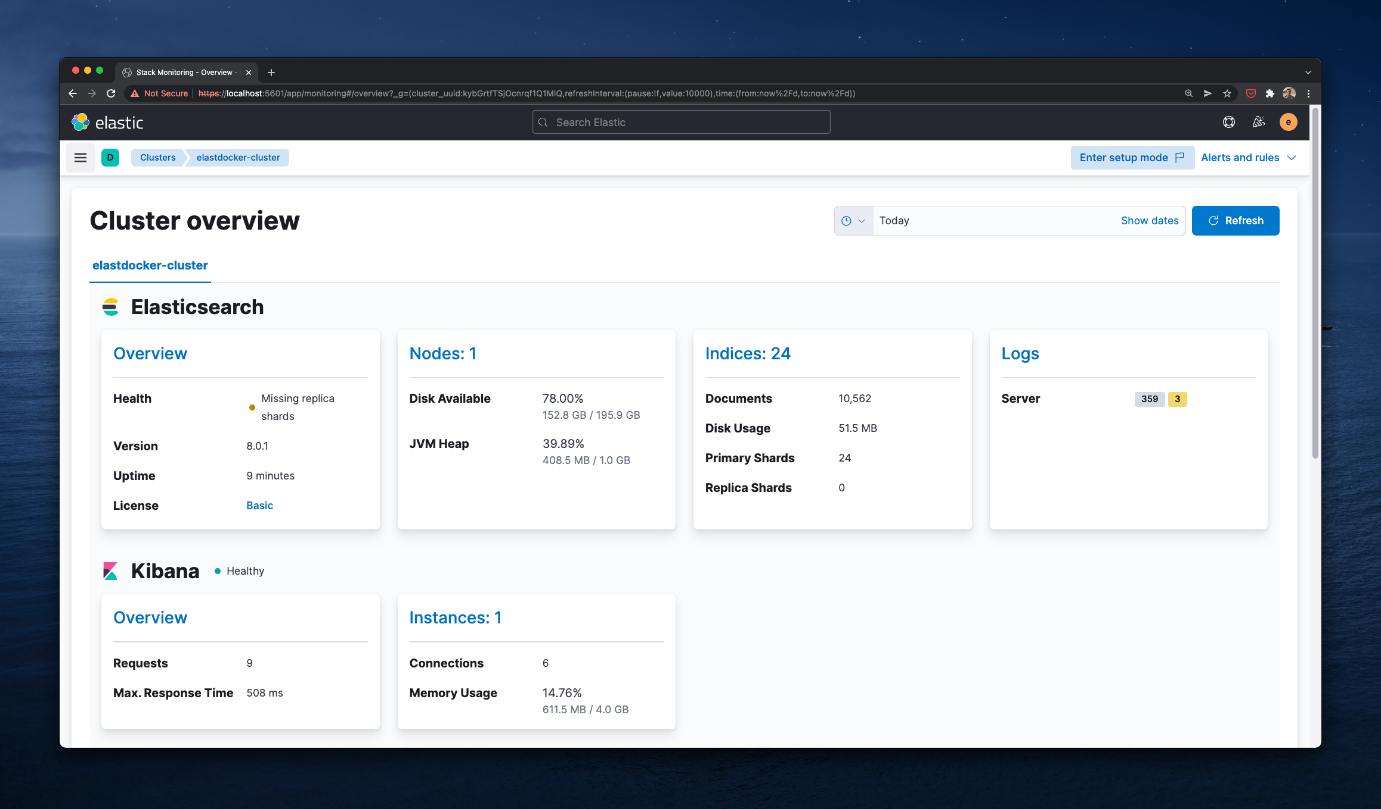
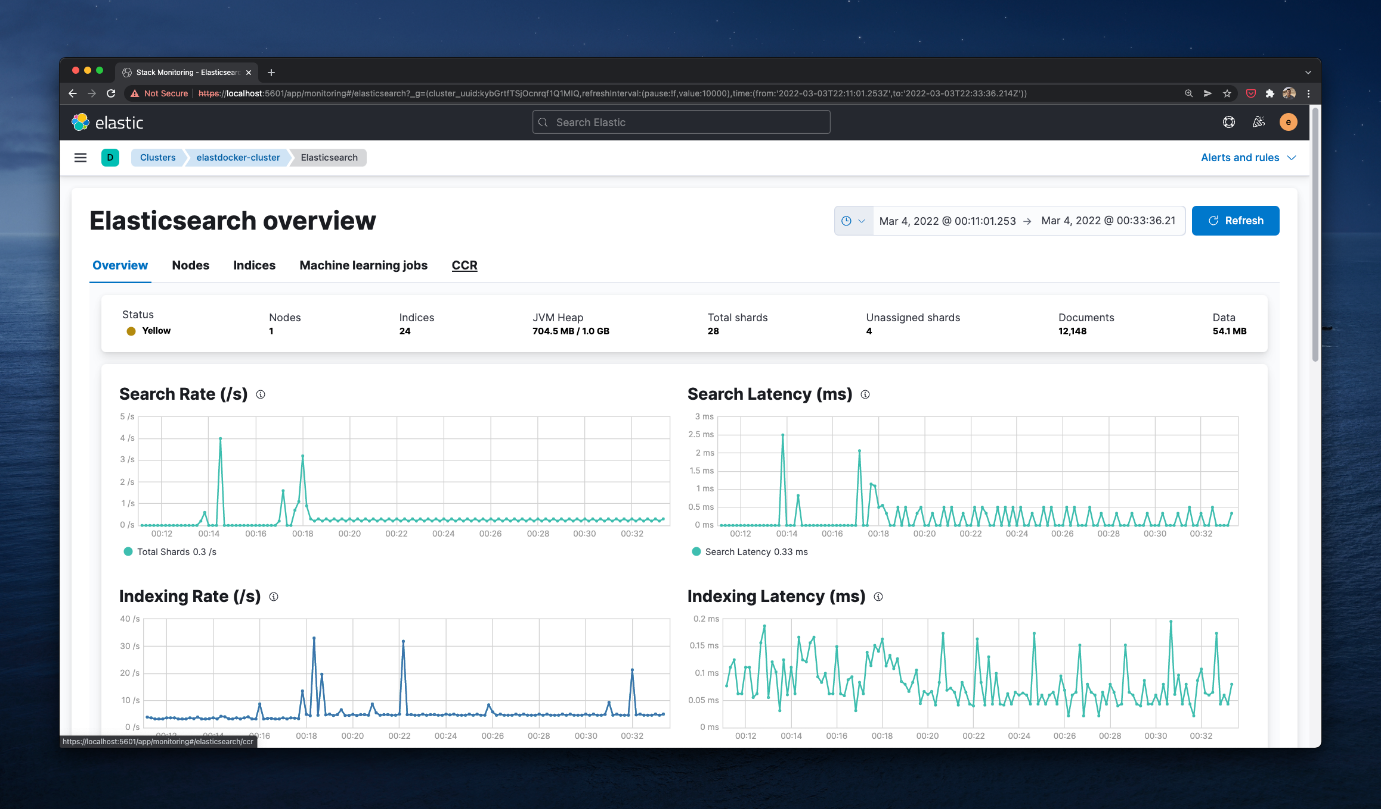
For more details or other languages you can check the following:

* [APM Agents in different languages](https://www.elastic.co/guide/en/apm/agent/index.html)

**Monitoring The Cluster**

**Via Self-Monitoring**

Head to Stack Monitoring tab in Kibana to see cluster metrics for all stack components.

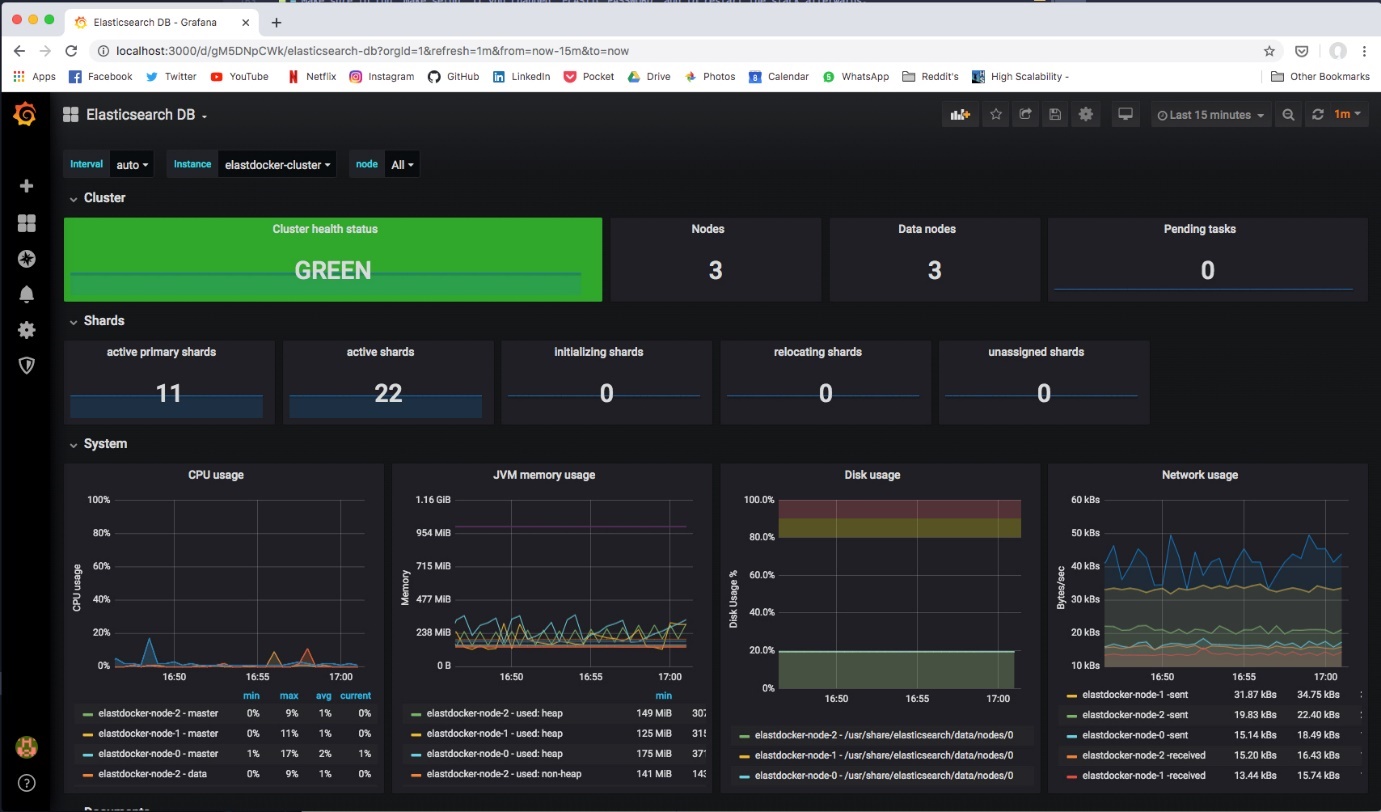
[](https://user-images.githubusercontent.com/16992394/156664539-cc7e1a69-f1aa-4aca-93f6-7aedaabedd2c.png) [](https://user-images.githubusercontent.com/16992394/156664647-78cfe2af-489d-4c35-8963-9b0a46904cf7.png)

In Production, cluster metrics should be shipped to another dedicated monitoring cluster.

**Via Prometheus Exporters**

If you started Prometheus Exporters using make monitoring command. Prometheus Exporters will expose metrics at the following ports.

| **Prometheus Exporter** | **Port** | **Recommended Grafana Dashboard** |
| --- | --- | --- |
| elasticsearch-exporter | 9114 | [Elasticsearch by Kristian Jensen](https://grafana.com/grafana/dashboards/4358) |
| logstash-exporter | 9304 | [logstash-monitoring by dpavlos](https://github.com/dpavlos/logstash-monitoring) |

[](https://user-images.githubusercontent.com/16992394/78685076-89a58900-78f1-11ea-959b-ce374fe51500.jpg)

**About**

🐳 Elastic Stack (ELK) v8+ on Docker with Compose. Pre-configured out of the box to enable Logging, Metrics, APM, Alerting, ML, and SIEM features. Up with a Single Command.