CONTINUOUS MONITORING ON DOCKER WITH ELK STACK PROJECT

Prepared by: Shubham Mehta

Project Objective :

CM on Docker with ELK stack is the process that helps developers to monitor the application in real-time using Kibana.

The App Should Have The Following Features :

- The application and its versions should be available on GitHub
- Commit the code multiple times and track their versions on GitHub
- Build the application in Docker, and host it in Docker Hub
- Deploy ELK stack on Docker and push application logs to it
- Automate Docker build and deployment using Jenkins pipeline code

Following tools should be used:

- **Docker**
- **Docker Compose**
- **Elasticsearch**
- **Logstash**
- **Kibana**
- **Spring Boot application**

Background of the problem statement:

XYZ Technology Solutions hired you as a DevOps Engineer. The company is undergoing an infrastructural change regarding the tools used in the organization. The company decides to implement DevOps to develop and deliver the products. Since XYZ is an agile organization, they follow Scrum methodology to develop the projects incrementally. They decide to dockerize their applications so that they can deploy them on Kubernetes. Each application when deployed and exposed, will have a unique URL and port, using which we can access that

Step by step process :

This is a project for monitoring the performance of servers and network traffic using ELK (ElasticSearch, LogStash and Kibana) as well as grafana and a single file was created for creating multi-container application (docker-compose). Thus, using a single file makes possible to create a full-deployed stack required by OpenStack monitoring solution. It has two main part to be monitored in openstack such as the server performance where its samples were produced by ceilometer and network flow where its samples were produced by softflowd.

The list of containers :

logstash

- IP: 172.26.36.4
- It has 3 main volumes for their own basic requirements such as storing the data and configuration files. (logstash-data, logstash-config, logstash-pipe) Furthermore, it has 2 additional volumes (mapping-res, mappingprod-res) for indexing the entries. Actually, these are the files which were generated by containers (mapping, mappingprod)
  The java was set to 16G memory. Otherwise, it was crushed because of insufficient memory.

The list of containers :

- elasticsearch

    - IP: 172.26.36.2, 172.26.36.3 and 172.26.36.7
    - The cluster has 3 containers. Even all of them are located on the same host, it would be separated over the multiple host in order to get high availability.
    - It has 2 main volumes for storing data and its configuration.
    - The logstash ip and memory size for java was set properly.

- kibana

    - IP: 172.26.36.5
    - The elasticsearch ip was set in order to visualize their data.

The list of containers :

- grafana
  - IP: 172.26.36.6
  - The elasticsearch ip was not initialized because the configuration is applied on the web application. However, the username and password were set here.

- mapping and mappingprod
  - IP: 172.26.36.10, 172.26.36.11
  - These are ubuntu containers which runs regularly self-developed python codes in order to index the entries in a proper format using the translate functionality of the logstash. There are number of ip addresses which belong to same tenant so that these ip addresses were accumulated under a single title and indexed properly. The xml files were generated and updated for each hour. The file were used by logstash because the volume were attached these both containers.
  - The name of the executable python as well as the output files were set here.

Multi-container for an application - docker-compose.yaml :

The volumes and network were seperated because make any change on the configuration files can be easy without modifying the compose files and adding/removing another container can be applied properly by adding just another ip address.
We have deployed 6 containers on a single computer which has 48 cores and 128 GB RAM. Furthermore, 2 additional containers were also deployed running a self-developed python code regulary. It is not possible to monitor a tenant network traffic since ceilometer does not provide the information. On the other hand, sotfflowd can capture the network traffic based on the ip adressed. However, we need to visualize the network traffic for each tenant separately. (floating ip and router gw interface belong to specific tenant). Mapping container was simply constructed (Dockerfile) because of this purpose.