**Steps To Install and run the logging module on local system**

Note: Tested on ubuntu 18 and my MacBook Pro

1. Clone the code from git repository

2. cd logging

3. Install and setup docker and docker-compose on local machine.

https://www.digitalocean.com/community/tutorials/how-to-install-docker-compose-on-ubuntu-18-04

Follow the steps given in above url for docker and docker-compose installation.

4. Once docker-compose installed run the following command within logging directory

a) **docker plugin install grafana/loki-docker-driver:latest --alias loki --grant-all-permissions**

b) **docker plugin ls**

**c) docker-compose -f docker-compose.yml up**

if you get permission error use sudo

s**udo docker-compose -f docker-compose.yml up**

Note: If you get an error for node-exporter in MacBook Pro please use below configuration in docker-compose.yml

volumes:

- /private/var/db/timezone:/etc/timezone:ro

Instead of below in mon\_node-exporter: section

volumes:

- /etc/localtime:/etc/localtime:ro

- /etc/timezone:/etc/timezone:ro

5. Can verify all the containers are up and running using the command **docker ps**.

A picture containing graphical user interface

Description automatically generated

Once all the containers are up and running please check the url and login `**http://localhost:3000**`

6. Login in Grafana

username: **admin**

password: admin

change the password if required.

7. Go to explore on left panel and select a data source Loki/Prometheus

Graphical user interface, application

Description automatically generated

By default Loki and Prometheus are configured through config itself. if no data source is visible there please add by going to settings-> data source option

7.1 Add Loki as data source URL http://loki:3100

7.2 Add Prometheus as data source URL http://prometheus:9090

8. Now go to explore option in left panel and check available logs

9. For adding dashboards for visualization

Go to + icon on left panel click on import

Type the ID

Docker monitoring dashboard Id: 193

and select data source Prometheus and load the dashboard.

url: https://grafana.com/grafana/dashboards/193

Graphical user interface, application

Description automatically generated A screenshot of a computer

Description automatically generated Graphical user interface, text, application

Description automatically generated

Chart

Description automatically generated

If we need to add monitor more metrics, we can create panels with appropriate queries.

Already added node exporter as container which will pull the system metrics.

**Details on services/plugins used:**

---------------------------------

**Grafana:**

**Grafana** is open source visualization and analytics software. It allows you to query, visualize, alert on, and explore your metrics no matter where they are stored. In plain English, it provides you with tools to turn your time-series database (TSDB) data into beautiful graphs and visualizations.

**Prometheus**:

**Prometheus** is a free software application used for event monitoring and alerting. It records real-time metrics in a time series database (allowing for high dimensionality) built using a HTTP pull model, with flexible queries and real-time alerting.

**Node Exporter** :

**Node Exporter**  is a **Prometheus exporter** for server level and OS level metrics with configurable metric collectors. It helps us in measuring various server resources such as RAM, disk space, and CPU utilization.

**Loki-promtail:**

**Grafana Loki** is a tool that gives you a panel for indexing of your systems' logs and visualizing them on a dashboard. ... Promtail, just like Prometheus, is a log collector for Loki that sends the log labels to Grafana Loki for indexing.

**Cadviser:**

**cAdvisor** (Container Advisor) provides container users an understanding of the resource usage and performance characteristics of their running containers. It is a running daemon that collects, aggregates, processes, and exports information about running containers.

Important urls:

Promtail/loki: https://grafana.com/docs/loki/latest/clients/promtail/

Grafana: https://grafana.com/docs/

Prometheus: https://prometheus.io/docs/introduction/overview/

cadvisor: https://hub.docker.com/r/google/cadvisor/

Note: I have implemented and tested this central-logging solution stack on docker conatiner.

Below are sample prometheus Queries.

a)Memory Utilization

100 \* (1 - ((avg\_over\_time(node\_memory\_MemFree[5m]) + avg\_over\_time(node\_memory\_Cached[5m]) + avg\_over\_time(node\_memory\_Buffers[5m])) / avg\_over\_time(node\_memory\_MemTotal[5m])))

b)CPU Utilization

1. - (avg by (instance) (rate(node\_cpu[5m])) \* 100)

c)Disk Utilization

100 - ((node\_filesystem\_avail\_bytes{mountpoint="/",fstype!="rootfs"} \* 100) / node\_filesystem\_size\_bytes{mountpoint="/",fstype!="rootfs"})

d)Process state

node\_systemd\_unit\_state{name="abc.service",state="active"}

**Note: nginx and ctovena/logger are the sample application containers which we can use as reference from prometheus.yml config to scrape the metrics.**

**Answer for the below Question**

**""Assume that this is a development server under load in which the developers are doing a variety of tasks, like running database and web servers, transferring files,  
opening sockets and exposing ports to the outside world.  
What would you additionally monitor to make their life easier and debug the problems  
they might be having more efficiently?""**

**Answer**: For the ease of developers we can implement alerts using Alertmanager and tools like PagerDuty,Opsgenie etc for critical metrics like cpu utilisation, memory usage, instances availability, disk utilisation, http server logs, etc using Prometheus on various channels like slack, email, telegram etc. Loki/Promtail aggregates and avails application's log which can be used to get insight of what's going on in an application. If the application or a script fails to run, we can go ahead and can check it on Grafana rather than exec into container and checking through commands. Also can monitor unused port usage, outside world security dashboards etc.

KPI is the important key to improve monitoring efficiency.

KPI - In simpler terms, you have a database call, or your micro-services request/response should be within 1 minute or in milliseconds or should be in the accepted timeline. If you collect all data based on your KPI, you can easily identify hot spots that need attention and fix it before it gets blast in the production. For a successful monitoring environment, you have to structure KPI for all of the components in your environment.

In turn, these data can be converted into useful metrics and can be measured easily using dashboards, charts, and so on.