

Logging System

Logs are somehow irritating to comb through, especially when the logs are scattered around. A good logging system would definitely help alleviate some of this hassle. The standard recommendation seemed to be the ELK stack, Loki was the new kid of the block. So the two logging tools I tried out were: ELK stack, Loki

ELK Stack:

The ELK stack stands for Elastic Search, Logstash and Kibana - One for metric collection, metric ingestion and then distribution/visualization. I setup the system with docker containers and it took a lot of work in order to even get the system to start properly. I think the root cause is that the ELK stack is not meant to be run on a single host. It took up significantly more resources than the Melton API it was monitoring, doesn't really make sense. It was also quite hard to configure for basic log collection. Since it took more resources than it was worth, I decided to keep looking.

Loki:

As with other stuff the Grafana guys create, Loki is pure gold. `Loki is a horizontally-scalable, highly-available, multi-tenant log aggregation system inspired by Prometheus. It is designed to be very cost effective and easy to operate.` Sounds like someone was reading my requirement list :D

So I decided to set up Loki+Promtail(metric collector) in a docker container, and I added `/var/log` as a volume mount. This volume mount means Loki would already get access to system level logs. The config(defined in `loki-promtail-config.yaml`) then scrapes the defined logs and ingests it in a format that is query-able.

Loki is also set up to be able to be used as a log driver directly by docker, this means all docker logs (when specified) are directly labelled and sent over to Loki. Loki is also horizontally scalable, so multiple hosts can forward their logs to the Loki push endpoint.

In our setup, Loki is limited to localhost, listening on port 3100. One can add Loki as the logging driver by using the below code in their docker-compose file:

```
logging:
  driver: loki
  options:
    loki-url: "http://localhost:3100/api/prom/push"
    loki-retries: "5"
    loki-batch-size: "400"
```

I should eventually set up HTTPS support too :P

Now that we have logs ingested by Loki (its quite efficient in resource usage too), then we just have to add it as a datasource on Grafana and we instantly have access to the logs. We can search logs by tags and labels, and even create a custom dashboard that constantly displays the log rate and log themselves.

The dashboard looks like this:

grafana.png

The log frequency is plotted in the graph on top, the App logs are in the second box and the last one defines the requests received by NGINX. The request received may help figure out the source of a crash.

Revision #3

Created 2 August 2020 11:35:36 by pari

Updated 2 August 2020 15:13:42 by pari