Modern infrastructure monitoring using OSS tools

DevStaff meetup
13 January 2022
Istvan Gyori

Who am I?

- Cloud/DevOps engineer at a Danish Fintech company (with Linux Sysadmin background)
- My “work-stack”: AWS, Kubernetes (EKS), IaC (Terraform), Prometheus, Grafana, GitLab, Docker, Ansible
- Working remotely since 2012, based in Budapest, Hungary

Infrastructure monitoring?

- Why? I’m a Developer, I don’t care about Infra...
- Good news! All of these can be used to monitor your applications as well!

What is OSS?

- Open-source software tools:
  - released under a license in which the copyright holder grants users the rights to use, study, change, and distribute the software and its source code to anyone and for any purpose

Why do we monitor things?

- “You **only** need to monitor and measure what is important”
- SLA: Service-level agreement (what kind of services are provided by a service provider, and when can we say that the quality of the service is “good enough”)
- SLO: Service-level objective is a key part of the SLA (e.g. The application will be available 99.99% of the time)

We need metrics to measure if the SLOs are met

- OSS (Open-source software) tools to the rescue
- Expose metrics that can be collected (Node-exporter), or expose a /metrics endpoint in your application
- Collect and Store the metrics: Prometheus (and Thanos for long-term storage)
- Visualize the metrics (Grafana)
- Send alerts to Developers/DevOps/Ops/Your manager if there is an issue

Node exporter or /metrics application endpoint

- Expose the metrics by using:
  - Node exporter: to monitor physical or virtual machines
  - Custom application /metrics endpoint (Java, NodeJS, etc.)

Prometheus

- Collect (scrape) and store metrics
- Prometheus is a free software application used for event monitoring and alerting
- It records real-time metrics in a time series database (TSDB) (allowing for high dimensionality)
- built using a HTTP pull(!) model, with flexible queries and real-time alerting
- Scalable and HA (High availability)
- Store the entire configuration as YAML in git, store metrics on an AWS EBS or local disk
- A collection of Prometheus alerts: https://awesome-prometheus-alerts.grep.to/rules.html
- URL: https://prometheus.io/

Why pull the metrics instead of pushing them?

- When you push metrics to a metrics DB/service, your alerting can give you false-positive “application is down”, “call the Boss” alerts when:
  - There are network issues between the application and the metrics DB/service
  - Difficult to create a HA/redundant setup without duplicating metrics data
  - You will lose metrics when you do maintenance on your metrics DB/service
  - Application Developers needs to take care about sending the metrics (use 3rd party client libraries, cache metrics and re-send them if needed; this increases application complexity)
- When pulling the metrics from an application:
  - Beside exposing a /metrics endpoint and data on it, you (Developers) don’t need to worry about having to send it anywhere
  - you can be sure that the application is working, even if you do maintenance/upgrades on Prometheus (if one of the nodes of your HA Prometheus cluster can access the application, likely your customers can also use the application
  - Easy to create a HA solution, without duplicating metrics data
  - Easy to do rolling updates of the monitoring cluster

Grafana

- Visualize the metrics
- Grafana is a multi-platform open source analytics and interactive visualization web application
- It provides charts, graphs, and alerts, and a WYSIWYG editor
- Supports different data sources (Prometheus, AWS CloudWatch, etc.)
- Export, store and apply the created dashboards as JSON in git (or just use the WYSIWYG editor)
- Lots of pre-built dashboards: https://grafana.com/grafana/dashboards/
- URL: https://grafana.com/

Alertmanager

- Send alerts
- Alertmanager handles alerts sent by the Prometheus server
- It takes care of deduplicating, grouping, and routing alerts to the correct receiver integration such as email, Slack, PagerDuty, or OpsGenie
- Store configured alerts as YAML in git
- It also takes care of silencing and inhibition of alerts
- URL: https://prometheus.io/docs/alerting/latest/alertmanager/

Demo time (Demo stack)

- Prometheus: http://localhost:9090/
- Grafana: http://localhost:3000/ (Docker host)
- Alertmanager: http://localhost:9093/

Some interesting projects

- Monitor your smoker: https://dev.to/focusedlabs/i-built-a-rube-goldburg-machine-to-monitor-my-smoker-using-k8s-prometheus-and-grafana-5c59
- Monitor your beer fermentation: https://www.brewbench.co/
- Grafana dashboard: https://grafana.com/grafana/dashboards/3957

This sounds like a hard thing to try...

- If you have Docker, docker-compose installed, just clone this repo, and follow the Installation steps: https://github.com/igyori/Docker-Compose-Prometheus-and-Grafana
- And run it using docker-compose up in MacOS/Linux/WSL
- Access Prometheus at: localhost:9091 (user/pass: admin/admin)
- Access Grafana at: localhost:3000 (user/pass: admin/admin)
- Disclaimer: this is intended for local testing purposes only. If you would like to deploy it to a machine that is accessible from the public internet, change the default passwords, and make sure that you have properly configured firewall or Security Groups (in AWS).

Next steps

- Check the existing Grafana dashboards at: https://grafana.com/grafana/dashboards/
  - You might want to start using some of them
- Pull the Prometheus-Grafana demo stack from GitHub, start it, and experiment with it
- Convince fellow Developers, DevOps engineers and managers in your company to start using some of these tools :)
- Check out the Monitoring section of the Google SRE workbook: https://sre.google/workbook/monitoring/

• Want to connect with me?

- DevStaff Slack profile name: igy
- LinkedIn: https://www.linkedin.com/in/istvan-gyori-4459ba21/
- … GitHub: https://github.com/igyori
- … or Email: devops@gyori.org

- Thank you for your attention!

Q&A

- Questions?