DOI: https://doi.org/10.36910/6775-2524-0560-2021-43-31
УДК: 004.71
Костючко Сергій Миколайович, к.т.н., доцент
https://orcid.org/0000-0002-1262-6268
Кирилюк Людмила Миколаївна, асистент
https://orcid.org/0000-0002-8279-3133
Протасюк Андрій Русланович, студент
Кривдік Олег Андрійович, студент
Романюк Денис Сергійович, студент
Луцький національний технічний університет

МОНІТОРИНГ ПРОГРАМ НА КЛАСТЕРІ RASPBERRY PI

Костючко С., Кирилюк Л., Протасюк А., Кривдік О., Романюк Д. Моніторинг програм на кластері Raspberry Pi. У цій статті йдеться про програми моніторингу, розгорнуті в кластері Kubernetes, що працює на Raspberry Pi, зокрема про інфраструктуру моніторингу. Важливо зазначити, що, хоча у цій статті основна увага приділяється платформі Raspberry Pi, більша частина описаного, крім спеціальних областей ARM, застосовується до інших апаратних платформ.
Ключові слова: Raspberry Pi, кластер, Prometheus, Elasticsearch, Kibana, Telegraf.

Костючко С., Кирилюк Л., Протасюк А., Кривдик А., Романюк Д. Мониторинг программ на кластере Raspberry Pi. В этой статье речь идет о программах мониторинга, развернутые в кластере Kubernetes, работающий на Raspberry Pi, в частности об инфраструктуре мониторинга. Важно отметить, что, хотя в этой статье основное внимание уделяется платформе Raspberry Pi, большая часть описанного, кроме специальных областей ARM, применяется к другим аппаратных платформ.
Ключевые слова: Raspberry Pi, кластер, Prometheus, Elasticsearch, Kibana, Telegraf.

Kostiuchko S., Kyryliuk L., Protasyuk A., Kryvdik O., Romaniuk D. Monitoring of programs on the Raspberry Pi cluster. This article discusses the monitoring programs deployed in the Kubernetes cluster running on the Raspberry Pi, including the monitoring infrastructure. It is important to note that, although this article focuses on the Raspberry Pi platform, much of what is described applies to other hardware platforms in addition to special ARM areas.
Keywords: Raspberry Pi, cluster, Prometheus, Elasticsearch, Kibana, Telegraf.

Вступ

Будь-яка виробнича програма потребує хорошого стека моніторингу. Для більшості програм це включатиме поєднання журналів та показників.

Метрики реєструються програмою для відстеження таких речей, як частота запитів, частота помилок та тривалість запиту. Метрики корисні для швидкого виявлення проблем - наприклад, надмірного рівня помилок або неприйнятної тривалості запиту. Метрики не визначають, що спричинило конкретну проблему, лише те, що сталася подія проблеми. Prometheus – це фреймворк, який надає додатку можливість записувати та зберігати метрики за відносно низькою вартістю.

Реєстрація, як правило, включає запис у файл або stdout. Записи журналу зазвичай містять такі речі, як мітки часу та деталі події, що сталася. Наприклад, запис журналу може містити інформацію про неправильно сформований HTTP-запит і містити інформацію про клієнта-запитувача, а також інформацію про те, що саме було не так із запитом. Таким чином, журнали дуже хороші для розслідування та вирішення проблем.

І метрики, і ведення журналу мають місце в моніторингу додатків. Метрики мають мінус щодо входу, оскільки вони не містять контексту, а лише інформацію про те, що щось сталося. На відміну від цього, журнали можуть надавати контекст, необхідний для розслідування та усунення неполадок. Недоліком журналювання є те, що контекст є дорогим порівняно з метриками. Показники дуже малі, часто це лише мітка, число та позначка часу. Журнали можуть бути досить об'ємними, інциденти може бути порівняно важко попередити з журналу.

Існує ряд ресурсів, що охоплюють філософію та механіку моніторингу. Два чудові ресурси – це посібник Google SRE, зокрема Глава 6: Моніторинг розподілених систем . Weaveworks має чудові результати щодо використання метрик з Prometheus та Grafana.

Огляд статей

Ця стаття охоплює набір інструментів, які зазвичай використовуються як для показників, так і для ведення журналу. Журнали можна захоплювати, індексувати та візуалізувати за допомогою так званого стеку EFK: Elasticsearch, Fluentd та Kibana. EFK – це модифікація ELK, де Logstash використовується для пересилання журналів замість Fluentd. Метрики можна запитувати, візуалізувати та сповіщати про них через Prometheus та Grafana.

Діаграма нижче ілюструє стек Elasticsearch-Fluentd-Kibana.

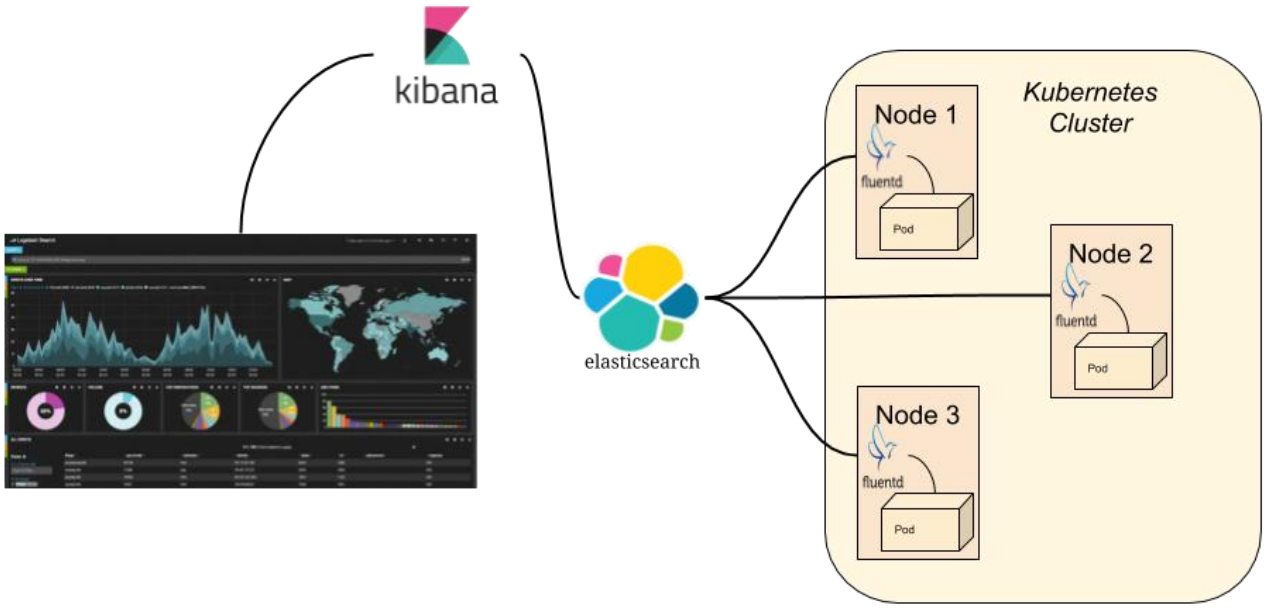

Рис. 1 – Стек EFK

Загалом, потік повідомлень журналів виглядає так:

1. Контейнер реєструється в stdout.
2. Fluentd працює на тому самому вузлі, що і контейнер, і фіксує потік повідомлень журналу.
3. Fluentd може бути налаштований для пересилання повідомлень журналу на цілу низку кінцевих сторінок, включаючи Elasticsearch.
4. Elasticsearch зберігає та індексує, наскільки це можливо, повідомлення журналу, які він отримує від Fluentd. Структурований журнал значно покращує здатність Elasticsearch індексувати повідомлення журналу.
5. Kibana може запитувати базу даних Elasticsearch і представляти результати як візуалізацію.

На наведеній нижче схемі показано, як Prometheus та Grafana вписуються в архітектуру програми:

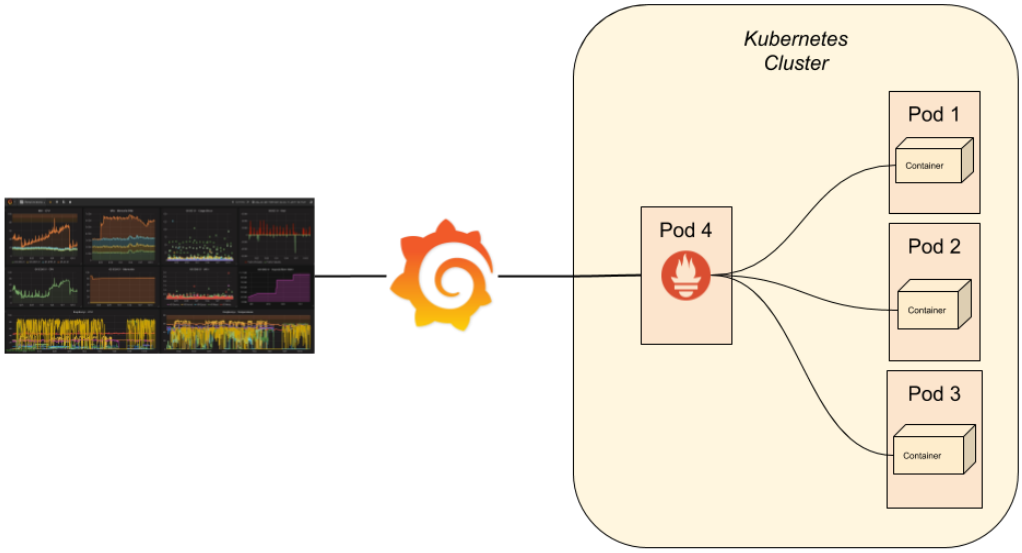

Рис. 2 – Prometheus та Grafana

Потік даних метрик такий:

1. Prometheus *викреслює* (тобто *витягує*) дані метрик із програми. Програми виявляють кінцеву точку, як правило, на порту 5000, яку Prometheus періодично опитує.
2. Prometheus зберігає дані метрик у власній базі даних часових рядів.
3. Grafana використовує PromQL, мову запитів Prometheus, для створення інформаційних панелей, які можна використовувати для візуалізації даних метрик. Grafana також може використовуватися як джерело попередження. У Grafana можна встановити порогові значення для спрацьовування попередження, коли дана метрика поза межами (наприклад, запити тривають занадто довго).

Проникливі спостерігачі відзначать, що і Kibana, і Grafana є інструментами візуалізації даних. Причиною потреби обох є поточна підтримка візуалізації журналів у Grafana (обмежена та в бета-версії), тоді як Kibana та Elasticsearch можуть керувати та відображати дані Prometheus.

Prometheus

Prometheus - це повний "набір інструментів для моніторингу та попередження систем". До його компонентів належать:

- Сервер, який вилучає дані метрик із увімкнених програм, а також є базою даних, що зберігає дані часових рядів, отримані з цих програм
- PromQL, мова запитів Prometheus, яка використовується для запитів до бази даних
- Клієнтські бібліотеки для різних мов (підтримується більшість основних мов)
- І інші компоненти (документація Prometheus розглядає їх більш детально)

Для встановлення Prometheus потрібні Helm та контролер Traefik Ingress.

Кроки встановлення Prometheus та розгортання-перевірки такі:

1. Клонуйте сховище.
2. Запустіть helm install ...команду, щоб розгорнути Prometheus.
3. Перевірте розгортання, відкривши інформаційну панель Prometheus у веб-браузері.

Prometheus також доступний із helm/charts/prometheus сховища. Зазвичай це хороше джерело для діаграм Helm, які можна використовувати для розгортання програм у кластері Kubernetes.

Grafana

Grafana - це платформа, яка дозволяє "запитувати, візуалізувати, оповіщати та розуміти ваші показники незалежно від того, де вони зберігаються". Ця можливість включає підтримку Prometheus як джерела даних. Grafana також може бути використана для того, щоб зробити те саме з метриками, виробленими Telegraf.

Щоб перевірити версію Grafana, необхідно протестувати за допомогою розгортання Prometheus. Для цього виконуються наступні дії:

1. Відкрийте інформаційну панель Grafana та увійдіть до системи. **Примітка** . Ймовірно, доведеться змінити адресу інформаційної панелі Grafana на щось на зразок

http://xxx.xxx.xxx.xxx:3000 IP-адреси Raspberry Pi, де була встановлена Grafana. Потрібно встановити Grafana на Raspberry Pi, до якої можна дістатись поза кластером Kubernetes. У даній роботі URL-адреса інформаційної панелі Grafana є http://10.0.0.100:3000.

2. Перевірка доступності Prometheus, проводиться налаштуванням джерела даних Prometheus.

Elasticsearch, Fluentd та Kibana

У цій роботі використано Fluentd замість Logstash. На підставі деяких обмежень Fluentd з точки зору її інфраструктури та безперешкодної інтеграції в Docker. Я також Fluentd замість Fluent Bit.

Як і у випадку з Grafana, розгорнуто Elasticsearch та Kibana в кластері Kubernetes.

Elasticsearch

Багато джерел пропонують встановлювати Elasticsearch за допомогою Homebrew.

Важливо зазначити, що Elasticsearch - це програма Java, для якої потрібна Java 8. Java 8 можна завантажити та встановити з сайту Oracle Java для завантаження. Завантаження - це .dmg файл, тому встановити його досить просто.

Запуск Elasticsearch є простим - запустіть /usr/local/elasticsearch-7.6.0/bin/elasticsearch (шлях може бути різним, залежно від завантаженої версії та місця, де він був фактично витягнутий).

Kibana

Як і у випадку з Elasticsearch, багато джерел також рекомендують встановлювати Kibana через Homebrew. Запуск Kibana - запустіть /usr/local/kibana-7.6.0-x86_64/bin/kibana (шлях може бути іншим, залежно від завантаженої версії та місця, де вона була фактично видобута).

Telegraf

Telegraf підтримує моніторинг машинних ресурсів, таких як використання процесора та диска. Оскільки Telegraf вимагає InfluxDB, вам доведеться його встановити. Оскільки Telegraf контролює машинні ресурси, він не встановлюється в кластер Kubernetes. Його слід встановити безпосередньо на хості Raspberry Pi, як це було зроблено з Grafana вище. Його потрібно буде встановити на кожному вузлі кластера, якщо ви хочете контролювати використання ресурсів у кластері.

Рис. 3 – Інформаційна панель Telegraf

Висновки.

У цій статті встановлено та протестовано Prometheus разом із Grafana. Встановили стек EFK і перевірено встановлення. Розгорнуто Telegraf на машинах, на яких розміщений кластер Kubernetes, і встановлено інформаційну панель Grafana для відображення метрик хосту Telegraf.

References.

1. S. Kostiuchko and V. Tchaban, "Variational Method of Auxiliary Equations in Nonlinear Systems Analysis and Synthesis Problems," *2019 IEEE 20th International Conference on Computational Problems of Electrical Engineering (CPEE)*, Lviv-Slavske, Ukraine, 2019, pp. 1-5, doi: 10.1109/CPEE47179.2019.8949123.
2. S. Kostiuchko, O. Kuzmych, A. Aitouche, S. Grinyuk and O. Mekush, "Application of Parametric Sensitivity Method to Analysis of Automatic Mooring Winch with Electric Drive System," *2019 4th Conference on Control and Fault Tolerant Systems (SysTol)*, Casablanca, Morocco, 2019, pp. 294-299, doi: 10.1109/SYSTOL.2019.8864751.
3. Luigi Atzori, Antonio Iera, and Giacomo Morabito. The Internet of Things: A survey. Computer Networks, 54(15):2787–2805, 2010, doi:10.1016/j.comnet.2010.05.010.
4. Algirdas Avizienis. Fault-Tolerant Systems. IEEE Trans. Computers, 25(12):1304–1312, 1976, doi:10.1109/TC.1976.1674598.
5. Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Timothy L. Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the Art of Virtualization. In Proceedings of the 19th ACM Symposium on Operating Systems Principles, SOSP 2003, Bolton Landing, NY, USA, October 19-22, pages 164–177, 2003, doi:10.1145/945445.945462.
6. Luiz André Barroso, Jimmy Clidaras, and Urs Hölzle. The Datacenter as a Computer: An Introduction to the Design of WarehouseScale Machines, Second Edition. Morgan & Claypool Publishers, 2013. doi:10.2200/S00516ED2V01Y201306CAC024, ISBN: 9781627050098.
7. David Bernstein. Containers and Cloud: From LXC to Docker to Kubernetes. IEEE Cloud Computing, 1(3):81–84, 2014, doi:10.1109/MCC.2014.51.
8. Rajkumar Buyya. High Performance Cluster Computing: Architectures and Systems. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1999. ISBN: 0130137847.
9. Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, and Ivona Brandic. Cloud Computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. Future Generation Comp. Syst., 25(6):599–616, 2009, doi:10.1016/j.future.2008.12.001.
10. Marcos Dias de Assunção, Rodrigo N. Calheiros, Silvia Bianchi, Marco A. S. Netto, and Rajkumar Buyya. Big Data computing and clouds: Trends and future directions. J. Parallel Distrib. Comput., 79-80:3–15, 2015, doi:10.1016/j.jpdc.2014.08.003.