ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Впровадження масштабованої системи моніторингу як засіб підвищення продуктивності інформаційних систем»

на здобуття освітнього ступеня магістра
зі спеціальності 126 Інформаційні системи та технології
освітньо-професійної програми Інформаційні системи та технології

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ _____
(підпис) *Ім'я, ПРІЗВИЩЕ здобувача*

Виконав: здобувач вищої освіти гр. ІСДМ-63
Олександр ГРЕЧУХА

Керівник: доктор технічних наук, професор
Ігор СІНІЦИН

Рецензент:

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ

ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут Інформаційних технологій

Кафедра *Інформаційних систем та технологій*

Ступінь вищої освіти *Другий (магістерський)*

Спеціальність *126 Інформаційні системи та технології*

Освітньо-професійна програма *Інформаційні системи та технології*

ЗАТВЕРДЖУЮ

Завідувач кафедрою _____

_____

«____» _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

Гречусі Олександру Павловичу

(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: *Впровадження масштабованої системи моніторингу як засіб підвищення продуктивності інформаційних систем*

керівник кваліфікаційної роботи *Ігор Сініцин, доктор технічних наук, професор*

(Ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)

затверджені наказом Державного університету інформаційно-комунікаційних технологій від «___» _____ 20__ р. № ___

2. Строк подання кваліфікаційної роботи «___» _____ 20__ р.

3. Вихідні дані до кваліфікаційної роботи: *1. Загальна архітектура побудови системи моніторингу IT-інфраструктури*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. *Вступ до моніторингу IT-інфраструктури.* **Error! Bookmark not defined.**

2. *Порівняльний аналіз систем моніторингу*

3. *Процес побудови системи моніторингу на базі Grafana, Loki та VictoriaMetrics*

5. Перелік ілюстративного матеріалу: презентація

6. Дата видачі завдання

«___» _____ 20__ р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційного проекту (роботи)	Строк виконання етапів (роботи)	Примітка
1.	*Опрацювання необхідної літератури*	3.10.2024	
2.	*Написання та подання на перевірку 1-го розділу роботи*	21.10.2024	
3.	*Написання та подання на перевірку 2-го розділу роботи*	8.11.2024	
4.	*Написання та подання на перевірку 3-го розділу роботи*	6.12.2024	
5.	*Реалізація практичної частини та подання на перевірку*	6.12.2024	
6.	*Подання кваліфікаційної роботи на відгук*	25.12.2024	
7.	*Подання кваліфікаційної роботи на рецензію*	26.12.2024	

Здобувач(ка) вищої освіти _____ _____

(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник

кваліфікаційної роботи _____ _____

(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра: 85 стор., 21 рис., 3 табл., 31 джерел.

Мета роботи – аналіз та розробка системи моніторингу IT-інфраструктури на основі сучасних технологій збору та візуалізації даних.

Об'єкт дослідження – системи моніторингу IT-інфраструктури та їх компоненти.

Предмет дослідження – методи та інструменти збору метрик, логів та їх візуалізації.

Короткий зміст роботи: У роботі проведено дослідження існуючих рішень для моніторингу IT-інфраструктури, таких як Nagios, Zabbix, Prometheus, VictoriaMetrics, а також хмарних рішень, таких як Datadog і New Relic. У роботі розглянуто проєктування архітектури системи моніторингу на базі Grafana, Loki та VictoriaMetrics. Наведено процес інсталяції, налаштування компонентів та інтеграції системи з іншими сервісами для досягнення максимальної ефективності моніторингу.

Проведено перевірку працездатності системи моніторингу. Створено дашборди, які візуалізують інформацію про завантаженість серверів, включаючи метрики використання CPU, пам'яті та дискового простору. Також налаштовано систему сповіщень (алертів), які автоматично надсилають повідомлення на електронну пошту у разі перевищення встановлених порогових значень або втрати зв'язку із сервером. Алерти були протестовані на предмет коректності роботи, що підтвердило їхню здатність забезпечити оперативне інформування про інциденти в IT-інфраструктурі.

КЛЮЧОВІ СЛОВА: моніторинг, IT-інфраструктура, Grafana, Loki, VictoriaMetrics, Prometheus, ELK, LGTM, масштабованість, продуктивність.

ABSTRACT

The text part of the qualification thesis for obtaining a Master's degree: 85 pages, 21 figures, 3 tables, 31 references.

The purpose of the work is to analyze and develop an IT infrastructure monitoring system based on modern technologies for data collection and visualization.

The object of the research is IT infrastructure monitoring systems and their components.

The subject of the research is methods and tools for collecting metrics, logs, and their visualization.

Summary of the work:

The paper investigates existing solutions for IT infrastructure monitoring, such as Nagios, Zabbix, Prometheus, VictoriaMetrics, as well as cloud-based solutions like Datadog and New Relic. It considers the design of the monitoring system architecture based on Grafana, Loki, and VictoriaMetrics. The installation process, component configuration, and integration of the system with other services are provided to achieve maximum monitoring efficiency.

The functionality of the monitoring system has been tested. Dashboards have been created to visualize server load information, including CPU usage, memory, and disk space metrics. An alerting system has also been configured to automatically send email notifications in case of exceeding set threshold values or loss of server connectivity. Alerts were tested to confirm their ability to provide prompt notification of incidents in the IT infrastructure.

KEYWORDS: monitoring, IT infrastructure, Grafana, Loki, VictoriaMetrics, Prometheus, ELK, LGTM, scalability, performance.

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
Навчально-науковий інститут Інформаційних технологій

ПОДАННЯ
ГОЛОВІ ЕКЗАМЕНАЦІЙНОЇ КОМІСІЇ
ЩОДО ЗАХИСТУ КВАЛІФІКАЦІЙНОЇ РОБОТИ
на здобуття освітнього ступеня магістра

Направляється здобувач(ка) Гречуха О. П. до захисту кваліфікаційної роботи
(прізвище та ініціали)
за спеціальністю 126 Інформаційні системи та технології
(код, найменування спеціальності)
освітньо-професійної програми Інформаційні системи та технології
(назва)
на тему: «*Впровадження масштабованої системи моніторингу як засіб підвищення продуктивності інформаційних систем*».
Кваліфікаційна робота і рецензія додаються.
Директор ННІ _____ _____
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Висновок керівника кваліфікаційної роботи

Здобувач_____
_____
_____
_____

Все це дозволяє оцінити виконану кваліфікаційну роботу здобувача _____ на оцінку «_____» та присвоїти йому кваліфікацію _____.

Керівник кваліфікаційної роботи _____ _____
(підпис) *(Ім'я, ПРІЗВИЩЕ)*
«____» _____ 20___ року

Висновок кафедри про кваліфікаційну роботу

Кваліфікаційна робота розглянута. Здобувач Гречуха О.П. допускається до захисту даної роботи в Екзаменаційній комісії.

Завідувач кафедрою Інформаційних Систем та Технологій

_____ _____
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

ВІДГУК РЕЦЕНЗЕНТА
на кваліфікаційну роботу
на здобуття освітнього ступеня магістра

здобувача вищої освіти Гречухи Олександра Павловича
(прізвище, ім'я, по батькові)
на тему «*Впровадження масштабованої системи моніторингу як засіб підвищення продуктивності інформаційних систем*»

Актуальність.

_____
_____
_____

Позитивні сторони.

1.
2.
3.

Недоліки.

1.
2.

Відзначені зауваження не впливають на загальну позитивну оцінку кваліфікаційної роботи магістерської.

Висновок: кваліфікаційна робота на здобуття ступеня магістра заслуговує оцінку "_____", а здобувач Гречуха О. П. заслуговує присвоєння кваліфікації: _____

Рецензент:
науковий ступінь, вчене звання

_____ _____
підпис *Ім'я, ПРІЗВИЩЕ*

ЗМІСТ

ВСТУП

Актуальність теми. Сучасна IT-інфраструктура вимагає високого рівня стабільності, надійності та продуктивності, що можливе лише за наявності ефективних інструментів моніторингу. У світі, де розподілені системи та мікросервіси стають стандартом, моніторинг відіграє ключову роль у швидкому виявленні інцидентів, їх усуненні та забезпеченні безперебійної роботи критичних бізнес-сервісів. У цьому контексті тема кваліфікаційної роботи є надзвичайно актуальною, адже впровадження сучасних систем моніторингу дає можливість вирішити низку проблем, пов'язаних із продуктивністю та стабільністю IT-сервісів.

Актуальність дослідження зумовлена необхідністю створення масштабованих, гнучких і економічно ефективних рішень для моніторингу. З розвитком цифровізації в Україні важливість таких систем ще більше зростає, оскільки вони дають змогу оптимізувати ресурси, підвищувати продуктивність та мінімізувати втрати від простоїв. Використання open-source рішень, таких як Grafana, Loki, VictoriaMetrics, дає змогу забезпечити якісний моніторинг за відносно низькою вартістю, що є вагомим фактором для малого та середнього бізнесу. Крім того, зберігання даних про роботу інфраструктури дозволяє фіксувати та аналізувати інциденти, які можуть виникати у процесі її функціонування, що створює можливості для їх подальшого дослідження та запобігання.

Метою дослідження є розробка комплексної системи моніторингу IT-інфраструктури з використанням сучасних open-source рішень, що відповідає вимогам масштабованості, надійності та функціональності.

Завданням дослідження є аналіз існуючих інструментів моніторингу та визначення їх переваг і недоліків. Розробити архітектуру системи моніторингу на основі Grafana, Loki, VictoriaMetrics. Створення механізму для збору та аналізу даних про роботу IT-інфраструктури підприємства. Налаштування сповіщень (алертів) для своєчасного інформування користувачів про інциденти.

Об'єкт дослідження – системи моніторингу IT-інфраструктури та їх компоненти.

Предмет дослідження – методи та інструменти збору метрик, логів та їх візуалізації.

Методи дослідження. Для вивчення сучасних систем моніторингу використовується аналітичний метод. Експериментальний метод використовується для тестування компонентів системи. Метод порівняльного аналізу – для оцінки ефективності використаних рішень. Для розробки архітектури системи – метод моделювання, а емпіричний метод використовується для оцінки роботи системи у реальних умовах.

Наукова новизна та практична значущість. Наукова новизна роботи полягає у створенні інтегрованої системи моніторингу, яка базується на open-source рішеннях та відповідає сучасним вимогам до продуктивності та масштабованості. Робота також демонструє нові підходи до інтеграції компонентів для збору та аналізу інцидентів, у той же час практична значущість полягає у можливості застосування розробленої системи на підприємствах різного масштабу для забезпечення стабільності IT-сервісів, а також збереження даних про інциденти для подальшого аналізу.

РОЗДІЛ 1: АНАЛІЗ ТА ДОСЛІДЖЕННЯ ІСНУЮЧИХ РІШЕНЬ ДЛЯ МОНІТОРИНГУ ІТ-ІНФРАСТРУКТУРИ

1. Вступ до моніторингу ІТ-інфраструктури. Значення моніторингу для підвищення продуктивності інформаційних систем

Моніторинг – це один з ключових інструментів, що забезпечує продуктивність та стабільність інформаційних систем. Основна роль моніторингу полягає в постійному зборі інформації про стан систем. Саме це дає змогу оперативно реагувати на можливі інциденти, виявляти потенційні проблеми, а також забезпечувати надійність роботи інформаційної інфраструктури.

Також моніторинг дає можливість впливає на швидкість прийняття рішень. Завдяки інформації, що міститься в моніторингу, а саме: наявності актуальних даних про стан мереж, серверів та додатків, адміністратори, на основі реальної картини, можуть приймати рішення, знижуючи ймовірність помилок та простоїв.

1.1. Основні вимоги до сучасних систем моніторингу

Сучасні системи моніторингу повинні відповідати переліку важливих вимог.

По-перше, це масштабованість та продуктивність. Система моніторингу повинна швидко та легко адаптуватися до зростання обсягів даних та збільшення інфраструктури.

Друга вимога – гнучкість та адаптивність. Для системи моніторингу важливо, щоб вона могла працювати з різними типами середовищ (локальними серверами, хмарними рішеннями тощо).

Третя вимога – інтеграція з різними технологіями та протоколами. Моніторингові системи повинні підтримувати широкий спектр технологій, протоколів і систем управління, наприклад, SNMP, NetFlow та інші.

1.1.1. Критерії оцінки та вибору систем моніторингу

Під час вибору системи моніторингу, щоб забезпечити ефективну роботу, варто враховувати певні критерії.

Однією з найголовніших характеристик системи моніторингу є її технічні можливості. Система має забезпечувати повноцінний збір і обробку даних, бути сумісною з вже існуючими технологіями.

Не менш важливою є вартість впровадження та підтримки цієї системи. Важливо враховувати не тільки початкову вартість на встановлення системи, але й довгострокові витрати на її підтримку.

Також слід звертати увагу на простоту використання та налаштування. Працювати із системою буде зручніше та ефективніше, якщо вона буде інтуїтивно зрозумілою і не вимагати великих ресурсів для навчання персоналу.

1.2. Традиційні інструменти моніторингу

Nagios

Одним з найстаріших та найпоширеніших інструментів для моніторингу IT-інфраструктури є Nagios. Його архітектура базується на клієнт-серверній моделі, де сервер збирає дані від агентів або інших сервісів через різні протоколи, такі як SNMP чи NRPE. Nagios може моніторити роботу мережевих пристроїв, серверів, додатків та інших компонентів IT-інфраструктури. Під час збору даних Nagios використовує клієнт-серверну архітектуру, де основний сервер (Nagios Core) відповідає за управління моніторингом, а агенти (Nagios Remote Plugin Executor, NRPE) надають дані про стан систем. Сервер отримує інформацію через плагіни або інші джерела, такі як SNMP.

Серед основних компонентів Nagios є такі:

- Nagios Core – головний сервер моніторингу.
- NRPE (Nagios Remote Plugin Executor) – агент для віддаленого виконання плагінів на серверах.
- Pnp4Nagios – інструмент для графічного відображення даних моніторингу.
- Nagios Plugins – набір скриптів для збору метрик (наприклад, статус дискового простору, CPU).

Nagios дозволяє моніторити стан серверів, мережевих пристроїв та додатків. Оповіщення про проблеми можна отримувати через електронну пошту або SMS.

Серед переваг Nagios є те, що вона підтримує широкий спектр протоколів для збору даних та має високу гнучкість налаштувань через конфігураційні файли.

Проте, Nagios має і недоліки, а саме: він не містить в собі вбудованого графічного інтерфейсу для спрощеного налаштування. А також у ній досить обмежені можливості масштабування для великих мереж.

Проілюструємо приклад використання Nagios. Ця система використовується багатьма організаціями для базового моніторингу серверів та сервісів. Ця система підійде, наприклад, освітнім установам, малим бізнесам або іншим організаціям, де важливий мінімальний моніторинг інфраструктури.

Zabbix

На відміну від Nagios, Zabbix є потужнішим інструментом моніторингу. Він забезпечує збір та аналіз великої кількості даних в реальному часі. Zabbix підтримує різноманітні джерела даних, таких як мережеві пристрої, сервери, додатки та інтернет-речі (IoT). Для Zabbix характерна велика гнучкістю в налаштуванні метрик і попереджень. Саме це робить її одним з найпопулярніших рішень для підприємств.

Щодо архітектури та принципів роботи Zabbix. Ця мережа використовує розподілену архітектуру, яка складається з таких компонентів:

- головного компоненту, що відповідає за збір даних – Zabbix Server.
- агентів, що працюють на моніторингових пристроях – Zabbix Agents.
- додаткового інструмента, який використовується для масштабування і зменшення навантаження на головний сервер – Zabbix Proxy.
- програмного інтерфейсу, що забезпечує можливість інтеграції з іншими системами Zabbix API.

Zabbix має досить широкі функціональні можливості. Як от:

- збір метрик з використанням агентів та безагентних методів (SNMP, IPMI, JMX тощо).
- автоматизація процесів моніторингу через тригери та правила оповіщень.
- інтеграція з іншими системами для візуалізації даних та управління інцидентами.

Оскільки Zabbix є однією з найпоширеніших систем моніторингу, він має чимало переваг. Найголовніше, що це відкрите програмне забезпечення з активною спільнотою та частими оновленнями. Ще однією з позитивних характеристик Zabbix є висока масштабованість та підтримка великої кількості пристроїв.

Але є й недоліки цієї системи. Наприклад, високі вимоги до ресурсів для великих середовищ та складність налаштування для нетехнічних користувачів.

Наводимо приклад впровадження Zabbix в роботу організацій для забезпечення ефективного моніторингу. Ця система успішно використовується у великих корпораціях, де необхідно моніторити тисячі пристроїв та серверів. Такими корпораціями можуть бути, наприклад, банки, провайдери IT-послуг, датацентри та інші.

1.3. Хмарні рішення для моніторингу

Datadog

Серед провідних хмарних рішень для моніторингу серверів, додатків та мережевих пристроїв у реальному часі є Datadog. Цей сервіс орієнтований на компанії, що працюють з великими обсягами даних у хмарних середовищах, таких як AWS, Microsoft Azure та Google Cloud.

Сервіс Datadog надає багато можливостей для ефективної роботи з ним. Наприклад:

- Моніторинг інфраструктури. З допомогою Datadog можна збирати метрики з різних систем та пристроїв, забезпечуючи глибоку інтеграцію з багатьма хмарними платформами та сервісами.

- APM (Application Performance Monitoring). Сервіс надає детальну інформацію щодо продуктивності додатків, включно з трасуванням запитів та аналізом роботи мікросервісів.
- Моніторинг журналів. Також Datadog підтримує збір, обробку та аналіз журналів додатків. Завдяки цій характеристиці користувачі сервісу мають змогу відстежувати проблеми на всіх рівнях додатку.
- Оповіщення та інтеграції. Платформа підтримує більше 400 інтеграцій з різними сервісами та мовами програмування. Наприклад, з сервісами для оповіщення: Slack, PagerDuty та інші. Мовами програмування: Python, Java, Node.js тощо. Провайдерами – AWS, Azur та іншими. Системами контейнеризації: Docker, Kubernetes. Ця особливість дозволяє використовувати Datadog у багатьох сферах і додатках.

Завдяки широким інтеграціям з хмарними платформами та сервісами, потужними інструментами для моніторингу продуктивності додатків, гнучкій системі оповіщень та звітності, високій масштабованості та надійності у хмарних середовищах Datadog є доволі популярним сервісом, яким користуються розробники програмного забезпечення та інші користувачі.

Проте, Datadog має певні обмеження. А саме:

- висока вартість для великих інфраструктур та довгострокового зберігання даних.
- цей сервіс вимагати спеціальних знань для налаштування та оптимізації.

Закцентуємо на реальних кейсах використання Datadog. Цим сервісом активно використовується компаніями для моніторингу масштабних хмарних інфраструктур. Наприклад, Datadog використовують такі компанії, як Samsung та Airbnb, аби забезпечити стабільність та продуктивність своїх сервісів.

New Relic

New Relic – це потужна платформа для моніторингу продуктивності додатків (APM). У ній важливе місце займають аналіз продуктивності, пошук проблемних ділянок і оптимізація роботи додатків у хмарних і локальних середовищах.

New Relic має хороший функціонал. Серед основних характеристик платформи є:

- моніторинг продуктивності додатків. New Relic дає можливість проводити глибокий аналіз роботи додатків у реальному часі, з можливістю відстежувати час відгуку, кількість запитів і трасування запитів через різні сервіси та мікросервіси.
- аналіз використання ресурсів. Платформа дає зможу відстежувати використання ресурсів, таких як CPU, пам’ять і дисковий простір, з допомогою якого можна виявити вузькі місця у роботі додатків.
- трасування запитів. New Relic надає детальне трасування запитів від початку до кінця. За допомогою цього є можливість швидко ідентифікувати проблеми з продуктивністю.
- моніторинг користувацького досвіду. Платформа відслідковує, як користувачі взаємодіють із додатком, що дозволяє аналізувати їх досвід і продуктивність додатка з погляду кінцевого користувача.

Проаналізувавши основні характеристики можемо виокремити переваги платформи New Relic. Це:

- глибокий аналіз продуктивності додатків у реальному часі.
- інтеграція з різними мовами програмування та платформами (Java, .NET, Ruby, Python, Node.js тощо).
- можливості моніторингу користувацького досвіду.

Проте, попри усі можливості, які надає New Relic, платформа має і недоліки. Це вартість. Вона може бути високою для великих підприємств. А також вона може вимагати великої кількості ресурсів для повного моніторингу додатків з багатьма компонентами.

Платформа New Relic є ефективною у використанні для компаній, що моніторять продуктивність веб-додатків та мобільних додатків. Наприклад, компанія GitHub використовує New Relic, щоб аналізувати продуктивність свого інструментарію для розробників, що забезпечує стабільну роботу їхніх сервісів на глобальному рівні.

1.4. Стек ELK (Elasticsearch, Logstash, Kibana)

Elasticsearch

Elasticsearch – це основний компонент стека ELK що виконує роль пошукового та аналітичного рушія для великих обсягів даних у реальному часі. Основою Elasticsearch є Apache Lucene, що забезпечує швидкий пошук, індексацію та аналітику. У Elasticsearch є одна є ключових особливостей – здатність обробляти та зберігати дані у структурованому та неструктурованому вигляді. Це дає можливість застосовувати цей компонент у багатьох галузях.

Elasticsearch має чимало переваг, що робить його ефективним у роботі. Перша перевага – це пошук та аналіз великих обсягів даних.

Завдяки його здатності здійснювати складні пошукові запити з мінімальною затримкою Elasticsearch можна використовувати для різних типів даних – від журналів подій до текстових документів та метрик продуктивності.

Ще одна з переваг Elasticsearch – швидкий повнотекстовий пошук. Це можливість у великих масивах даних знаходити інформацію на основі ключових слів або фраз.

Наступна перевага – агрегація. Це здатність здійснювати складні обчислення та аналіз на основі отриманих даних (середні, медіани, мінімальні та максимальні значення тощо).

Завдяки розподіленій архітектурі Elasticsearch може бути масштабованим (як по горизонталі, так і по вертикалі) шляхом додавання нових вузлів у кластері, що забезпечує надійність та безперебійність роботи навіть у великих інфраструктурах. Це робить його ідеальним для обробки терабайтів даних у розподілених середовищах.

Logstash

Logstash – це інструмент для збору, обробки та переадресації даних, зокрема журналів, подій, метрик та інших типів інформації з різних джерел до Elasticsearch або інших сховищ даних. Завдяки своїй гнучкості Logstash дозволяє працювати з різноманітними форматами даних, включно зі структурованими та неструктурованими логами, JSON, CSV тощо.

Logstash може збирати дані з різноманітних джерел – файлових систем, мережевих пристроїв, баз даних – та обробляти їх за допомогою фільтрів. Наприклад, фільтри дають змогу перетворювати дані, очищати їх, додавати нові поля чи структури для подальшого аналізу. Після обробки ці дані можуть бути переадресовані до Elasticsearch з метою індексації або інших систем.

Аби працювати з Logstash важливо знати його основні компоненти:

- Input: це компонент для збирання даних з різних джерел.
- Filter: модуль, що виконує обробку, перетворення та нормалізацію даних.
- Output: компонент для відправки оброблених даних до Elasticsearch або інших систем зберігання.

Logstash часто використовується для збору логів з веб-серверів, мережевих пристроїв та додатків. З його допомогою можна нормалізувати журнали перед їх передачею до Elasticsearch. Це забезпечує єдину структуру для всіх типів логів.

Kibana

Kibana – це інструмент для візуалізації даних, що використовується разом з Elasticsearch. З її допомогою створюються інтерактивні дашборди, графіки, звіти та аналізи на основі даних, які попередньо були зібрані у Elasticsearch. Завдяки Kibana користувачі можуть легко знаходити аномалії у даних, виявляти тенденції або відслідковувати ключові показники у реальному часі.

Щодо стосовновних інструментів візуалізації та аналітики в Kibana. Вона підтримує широкий спектр типів візуалізацій, таких як лінійні графіки, гістограми, кругові діаграми та інші. Також можна створювати дашборди, що поєднують кілька візуалізацій на одній сторінці для кращого розуміння даних.

Крім візуалізації, Kibana дає можливість створювати звіти, щоб надалі проводити аналіз та ділитися результатами цього аналізу з колегами. У Kibana можна швидко знаходити потрібні дані та змінювати параметри запитів у реальному часі завдяки її інтерфейсу, який підтримує інтерактивні фільтри та пошукові запити.

1.4.1. Переваги та недоліки ELK стека

Як і будь-який стек, ELK має свої переваги та недоліки. Щодо переваг:

Вагомим плюсом є масштабованість. Завдяки розподіленій архітектурі, кожен компонент ELK стека може бути масштабованим. Це дозволяє використовувати його в масштабних інфраструктурах з великими обсягами даних. Швидка індексація та пошук Elasticsearch забезпечує високу швидкість індексації та пошуку навіть при великих обсягах даних. Гнучкість і кастомізація у Logstash дає можливість обробляти дані за допомогою фільтрів, що робить його придатним для роботи з різними форматами даних. Зручні дашборди в Kibana надають зручні та ефективні засоби для візуалізації даних, що полегшує аналіз та прийняття рішень на основі отриманих результатів.

Серед недоліків ELK стека є складність налаштування. Налаштування та підтримка ELK стека можуть бути досить складними, особливо під час роботи з великими даними або при необхідності інтеграції з різними джерелами. А також високі вимоги до ресурсів, адже використання ELK стека вимагає значних ресурсів, особливо під час роботи з великими обсягами даних і складними запитами.

1.4.2. Приклади застосування ELK стека

ELK стек широко використовується в корпоративних середовищах з метою моніторингу безпеки, аналізу логів та подій в реальному часі. Наприклад, він допомагає оперативно виявляти порушення безпеки або аномальні події в системах, що використовуються в корпоративних мережах. Завдяки своїй здатності інтегруватися з багатьма джерелами даних ELK стек є оптимальним для великих

підприємств, яким необхідно відслідковувати тисячі серверів та мережевих пристроїв.

Наведемо приклади використання ELK:

- Банківський сектор: Використання ELK стека для відстеження операцій у реальному часі допомагає виявляти шахрайські дії, що підвищує безпеку системи.
- Телекомунікаційні компанії: ELK використовується для моніторингу мережевих подій. Це допомагає вчасно виявляти проблеми і реагувати на них, зменшуючи час простоїв.

1.5. OpenTelemetry

Концепція та призначення

OpenTelemetry – це набір інструментів з відкритим вихідним кодом, що призначений для стандартизації збору метрик, логів і трасувань у розподілених системах. Він розроблений, аби надавати єдину екосистему для спостереження за станом додатків. Таке його призначення дозволяє підвищити прозорість і контроль за роботою мікросервісних і розподілених архітектур.

Завдяки стандартизації збору метрик, логів та трасувань OpenTelemetry забезпечує єдиний підхід до збору даних, дозволяючи централізовано отримувати інформацію про метрики продуктивності, логи та трасування запитів між сервісами. Це суттєво спрощує моніторинг складних архітектур, своєю чергою забезпечуючи можливість легко інтегрувати систему з іншими інструментами для збору та аналізу даних.

OpenTelemetry підтримує різні мови програмування та платформи, такі як Java, Python, JavaScript, Go, .NET та інші. Саме ця характеристика забезпечує універсальність і дає можливість інтегрувати OpenTelemetry у різні платформи та додатки, включно з хмарними, серверними, мобільними та настільними рішеннями. OpenTelemetry також інтегрується з багатьма популярними системами спостереження, такими як Prometheus, Jaeger, Grafana та інші.

Можливості та інтеграція з іншими системами

OpenTelemetry дозволяє збирати, обробляти та передавати дані про метрики, логи та трасування у форматі, сумісному з різними інструментами для спостереження за додатками та інфраструктурою.

OpenTelemetry сумісна з Prometheus, Grafana, Jaeger тощо. Розглянемо детальніше.

- Сумісність з Prometheus. OpenTelemetry може збирати метрики з додатків і передавати їх до Prometheus. Це дозволяє використовувати існуючі дашборди та метрики у Grafana для аналізу продуктивності.
- Сумісність з Grafana. Завдяки інтеграції з OpenTelemetry, Grafana може візуалізувати зібрані дані у вигляді графіків та дашбордів, що дозволяє легко аналізувати продуктивність систем.
- Сумісність з Jaeger. Трасування запитів у розподілених системах є ключовою функцією OpenTelemetry. І Jaeger використовується для відображення цих трасувань. Він дозволяє візуалізувати шлях запиту через мікросервіси та виявляти вузькі місця в продуктивності.

Інтеграція з іншими системами дозволяє створити єдину платформу для моніторингу й аналізу продуктивності додатків та інфраструктури.

Переваги та недоліки OpenTelemetry

Серед переваг системи OpenTelemetry є:

- Універсальність. Вона зумовлена тим, що завдяки такій особливості OpenTelemetry як підтримка різних мов програмування та платформ можна інтегрувати її практично в будь-який додаток або систему.
- гнучкість. Система легко налаштовується для збору саме тих даних, які необхідні для моніторингу конкретних компонентів додатка.
- стандартизація. Завдяки єдиному стандарту збору даних, можна уникнути проблем з різнорідними рішеннями для моніторингу.
- інтеграція з популярними інструментами. OpenTelemetry можна використовувати разом з такими системами, як Prometheus, Grafana і Jaeger. Це спрощує інтеграцію в існуючі процеси DevOps та моніторингу.

Недоліки OpenTelemetry:

- новизна. OpenTelemetry є відносно новою технологією, тому документація може бути не повною або складною для розуміння.
- недостатня підтримка деяких старих систем. Хоча OpenTelemetry активно розвивається, проте, деякі старі системи можуть не мати повної підтримки для цієї технології.
- потреба у конфігураціях. OpenTelemetry вимагає налаштування та інтеграції для кожної конкретної системи або додатка, а це може зайняти певний час.

Приклади використання OpenTelemetry

OpenTelemetry широко використовується у компаніях, що працюють з мікросервісами, оскільки дозволяє відстежувати запити через кілька сервісів, виявляючи проблеми з продуктивністю на будь-якому етапі обробки запиту. Наприклад, OpenTelemetry можуть використовувати великі хмарні сервіси для моніторингу сотень або тисяч мікросервісів, кожен з яких обробляє запити в різних середовищах.

Серед компаній, що активно використовують OpenTelemetry є такі:

- Shopify. Компанія використовує OpenTelemetry для збору трасувань між мікросервісами, аби виявляти затримки у виконанні запитів та швидко вирішувати проблеми з продуктивністю.
- GitHub. Використовує OpenTelemetry для моніторингу роботи своїх систем, з метою аналізу продуктивності під час пікових навантажень і покращувати ефективність обробки запитів.

1.6. Стек LGTM від Grafana (Loki, Grafana, Tempo, Mimir)

Стек LGTM (Loki, Grafana, Tempo, Mimir) є ефективним набором інструментів від Grafana Labs, що збирає, аналізує та візуалізує дані, логи, метрики і трасування. Цей стек відзначається інтегрованістю компонентів, що спрощує

процес моніторингу й дає можливість отримати всебічний огляд стану IT-інфраструктури.

Grafana

Grafana є одним із найпопулярніших інструментів для візуалізації та аналізу даних. Основні особливості Grafana:

- Інтуїтивний, простий інтерфейс для створення дашбордів: користувачі можуть створювати персоналізовані дашборди з різними графіками, таблицями, картами та іншими візуалізаційними компонентами.
- Підтримка безлічі джерел даних: Grafana підтримує інтеграцію з різними базами даних і системами моніторингу, такими як Prometheus, InfluxDB, VictoriaMetrics, Loki, Elasticsearch та іншими.
- Оповіщення та моніторинг в реальному часі: з Grafana можливо налаштовувати оповіщення, які спрацьовують на основі заданих умов, і надсилати сповіщення через різні канали, такі як email, Slack або Telegram.
- Підтримка метрик PromQL: Grafana дає можливість використовувати запити PromQL для детального аналізу даних, що надходять із Prometheus і інших сумісних систем.

Loki

Loki – це система для зберігання та пошуку логів, що оптимізована для моніторингу та легко інтегрується з Grafana.

Як і кожна система для збору та зберігання логів, Grafana має певні особливості. Опишемо основні з них:

- Модель зберігання, що імітує Prometheus. Loki зберігає логи як потоки, пов’язані з часовими мітками. Але на відміну від традиційних рішень для логів, він не індексує весь вміст логів. Саме це робить його більш продуктивним і економічним.
- Масштабованість. Loki підтримує горизонтальне масштабування, завдяки чому можна обробляти великі обсяги логів без значних витрат на обчислювальні ресурси.

- Запити LogQL. Grafana надає можливість виконувати запити LogQL для аналізу логів, що дозволяє швидко знаходити потрібні події або патерни.

Tempo

Tempo – це інструмент для розподіленння трейсингу, що використовується для аналізу та відстеження запитів у розподілених системах.

- Безіндексна архітектура: Tempo дозволяє зберігати трейсинг-запити без використання традиційного індексування, що суттєво знижує витрати на зберігання.
- Підтримка OpenTelemetry: Tempo повністю підтримує OpenTelemetry, що дозволяє збирати дані трасування з різних мов програмування та платформ.
- Інтеграція з Grafana: трейсинги можна відображати в Grafana поруч з метриками і логами для отримання комплексної картини роботи системи.

Mimir

Mimir є системою для масштабованого зберігання метрик, спеціально розробленою для роботи в умовах великих навантажень.

Основні характеристики Mimir:

- Висока масштабованість. Mimir може зберігати значні обсяги метрик і підтримувати зберігання на рівні кількох десятків мільярдів часових рядів.
- Підтримка стандарту Prometheus. Mimir працює зі стандартними Prometheus-експортерами. Завдяки цьому йому вдається легко інтегруватися в існуючі інфраструктури, які використовують Prometheus.
- Швидкий пошук метрик. Оптимізовані алгоритми обробки дають можливість швидко виконувати запити, навіть у високонавантажених середовищах.

1.6.1. Переваги та недоліки LGTM стека

Розглянемо основні переваги та недоліки LGTM стека. Отож, переваги:

- Інтегрованість компонентів для комплексного моніторингу.
- Масштабованість і висока продуктивність у великих середовищах.
- Підтримка різних форматів даних для збору та аналізу.
- Гнучкість та адаптивність під потреби бізнесу.
- Можливість налаштування сповіщень і оповіщення в реальному часі.

Недоліки:

- Високі вимоги до ресурсів для повноцінного функціонування.
- Відносна складність налаштування і підтримки.
- Потреба у високій кваліфікації адміністраторів для налаштування.
- Можливі обмеження в специфічних сценаріях.
- Підтримка специфічних конфігурацій може вимагати додаткових зусиль.

LGTM стек надає всебічне рішення для моніторингу IT-інфраструктури. Однак він потребує значних ресурсів для повноцінного функціонування та високої кваліфікації адміністраторів для налаштування.

1.6.2. Реальні кейси застосування LGTM стека

LGTM стек застосовують у різних середовищах з високими навантаженнями. Наведемо приклади галузей, де ю доречно застосовувати LGTM стек:

1. Моніторинг фінансових транзакцій у банках: банки та фінансові організації використовують LGTM стек з метою моніторингу та аналізу транзакцій. Loki дає змогу зберігати логи транзакцій, а Tempo – відстежувати виконання транзакцій, а Grafana допомагає візуалізувати основні показники.
2. Обслуговування високонавантажених вебсайтів: великі веб-платформи, що обслуговують мільйони користувачів, застосовують LGTM

стек для моніторингу продуктивності сайтів та виявлення проблем із часом відгуку або помилками в логах. Mimir забезпечує зберігання великого обсягу метрик, Tempo, своєю чергою, дозволяє відстежувати шляхи запитів.

3. Хмарні платформи та DevOps: платформи, що забезпечують хмарні сервіси, інтегрують LGTM стек у свої DevOps-процеси. Loki та Tempo дозволяють відстежувати історію та взаємодію запитів між різними мікросервісами, що допомагає під час виявлення проблем і для покращення взаємодії між компонентами.

Загалом, впровадження LGTM стека допомагає суттєво зменшити час на виявлення і вирішення проблем, покращити ефективність та стабільність роботи систем, що позитивно впливає на користувацький досвід.

1.7. Prometheus

Prometheus – це відкрита система моніторингу та збору метрик, що розроблена для забезпечення надійного моніторингу систем і додатків. Prometheus став стандартом у галузі моніторингу завдяки своїй продуктивності, гнучкості та підтримці широкого спектру експортерів.

Архітектура та принципи роботи

Prometheus базується на pull-моделі збору метрик, що дозволяє серверу самостійно запитувати дані з метрик-експортерів або сервісів.

Основні характеристики Prometheus:

- Архітектура. Prometheus складається з основного сервера для збору та зберігання метрик, окремих експортерів для різних служб і компонентів, а також з Alertmanager для оповіщень.
- Метрики та часові ряди. Prometheus зберігає дані як часові ряди, які складаються з мітки часу і значення.
- Запити PromQL. Для збору, фільтрації і обробки метрик використовується мова запитів PromQL, що дає змогу гнучко налаштовувати запити.

Збір та зберігання метрик

Prometheus збирає метрики за допомогою експортерів, що надають дані про стан різних систем або служб.

- Експортери – це спеціалізовані програми або компоненти, які витягують метрики з додатків (наприклад, Node Exporter для збору метрик сервера, Blackbox Exporter для перевірки доступності сервісів).
- Зберігання. Для зберігання метрик Prometheus використовує локальну базу даних. Це дозволяє отримати доступ до даних для короткочасного зберігання, але обмежує можливості для довготривалого зберігання.

Інтеграція з Grafana

Prometheus широко використовується в комбінації з Grafana, що дозволяє створювати візуалізації та дашборди.

Візуалізація даних відбувається таким чином: Grafana інтегрується з Prometheus для відображення метрик у вигляді графіків, таблиць, гістограм тощо.

Користувачі можуть налаштовувати дашборди під специфічні потреби для відображення ключових показників продуктивності та стану системи.

1.7.1. Переваги та недоліки Prometheus

Переваги:

- Простота налаштування та масштабування завдяки pull-моделі.
- Гнучкість у використанні експортерів для збору метрик із різноманітних джерел.
- Підтримка мови запитів PromQL для створення складних запитів.

Недоліки:

- Обмеження в довгостроковому зберіганні даних через використання локальної бази даних.
- Не підходить для глибинного аналізу логів та трасування, фокусуючись лише на метриках.

- Високі вимоги до ресурсів у великих середовищах з великим обсягом метрик.

1.7.2. Приклади використання Prometheus

Prometheus є надзвичайно популярним серед організацій для моніторингу контейнеризованих додатків, зокрема в середовищах Kubernetes, де Prometheus використовує Kubernetes API для автоматичного виявлення сервісів і збору метрик. Наприклад, для моніторингу стану подів, вузлів і сервісів у кластері Kubernetes.

Реальні кейси:

- Netflix: компанія Netflix використовує Prometheus для моніторингу своїх хмарних мікросервісів, таким чином забезпечуючи швидке виявлення проблем та оповіщення.
- SoundCloud: як один із перших користувачів Prometheus, SoundCloud використовує його для моніторингу своїх мікросервісів, налаштовуючи алерти для контролю навантаження та забезпечення стабільності сервісів.

Prometheus зарекомендував себе як надійний і масштабований інструмент для моніторингу сучасних розподілених додатків та інфраструктури, особливо в поєднанні з Grafana для візуалізації даних.

1.8. VictoriaMetrics

VictoriaMetrics є масштабованим рішенням для зберігання та аналізу метрик, яке підтримує високу продуктивність та оптимізоване для обробки великих обсягів даних. Цей інструмент відомий своєю сумісністю з екосистемою Prometheus, що дозволяє використовувати його як альтернативне сховище метрик у великих середовищах.

Архітектура та особливості VictoriaMetrics

VictoriaMetrics підтримує мультиорендність, що дозволяє обробляти і зберігати метрики з кількох середовищ або користувачів одночасно. Це особливо

корисно для великих компаній або SaaS-рішень, де важливо забезпечити сегментацію даних.

Особливістю VictoriaMetrics є сумісність з Prometheus та його експортерами. VictoriaMetrics повністю підтримує Prometheus-експортери та PromQL, що забезпечує легку інтеграцію з існуючими конфігураціями та інфраструктурою Prometheus.

Масштабованість та продуктивність

VictoriaMetrics використовує спеціальні методи стиснення і зберігання, що знижує вимоги до дискового простору і забезпечує швидку обробку запитів. Таким чином ця особливість оптимізує зберігання даних.

Система підтримує великі обсяги даних завдяки горизонтальному масштабуванню, що робить її придатною для обробки метрик від великих хмарних або мікросервісних додатків.

1.8.1. Переваги та недоліки VictoriaMetrics

Серед переваг системи є:

- висока продуктивність та ефективне стиснення даних, що забезпечує економію ресурсів.
- підтримка Prometheus-експортерів і PromQL, що спрощує інтеграцію з існуючими системами.
- можливість роботи як у кластерному, так і в окремому режимах. А це дозволяє адаптувати рішення під різні потреби.

Але для користування цією системою збору та зберігання метрик необхідно мати високу кваліфікацію, особливо під час налаштування в кластерному режимі. Адже робота VictoriaMetrics залежить від специфічної конфігурації. Тобто, якщо налаштування не коректні, інтеграція з нестандартними інструментами або метриками може бути ускладнена.

1.8.2. Інтеграція з іншими інструментами

VictoriaMetrics інтегрується з Grafana, що дозволяє візуалізувати метрики в знайомому інтерфейсі Grafana і налаштовувати дашборди для відображення ключових показників.

Система може використовувати існуючі експортери від Prometheus, що знижує витрати на налаштування і забезпечує сумісність зі стандартами, прийнятими в екосистемі Prometheus.

1.8.3. Приклади використання VictoriaMetrics

Система VictoriaMetrics може використовуватися великими SaaS-платформами, де необхідно забезпечити мультиорендність і масштабованість. Наприклад, одна з хмарних платформ використовує VictoriaMetrics для збору метрик з декількох регіонів і клієнтів, забезпечуючи ізоляцію даних і високу доступність.

Також цю систему експлуатують компанії, які обробляють великі обсяги даних з пристроїв IoT, для швидкої обробки та зберігання метрик. Завдяки оптимізації запитів і стисненню даних система дає змогу аналізувати показники IoT-пристроїв у реальному часі.

VictoriaMetrics часто використовується в поєднанні з Kubernetes для моніторингу продуктивності мікросервісів. Наприклад, компанія може використовувати дану систему для збору метрик від тисяч мікросервісів, що працюють у кластері Kubernetes, та інтегрувати їх з Grafana для візуалізації.

Можемо підсумувати, що VictoriaMetrics має здатність обробляти значні обсяги даних, забезпечуючи швидкий доступ до метрик і економію ресурсів завдяки ефективному стисненню.

1.9. Моніторинг баз даних

Безумовно, критично важливим компонентом забезпечення стабільності і продуктивності IT-інфраструктури є моніторинг баз даних. Завдяки ньому можна

своєчасно виявляти та усувати проблеми, пов’язані з продуктивністю, забезпечуючи ефективну роботу додатків і сервісів.

Важливість моніторингу баз даних

Бази даних відіграють важливу роль в IT-структурі, адже є центральними сховищами для зберігання й обробки інформації, яку використовують додатки та сервіси в різних сферах, від фінансів до охорони здоров’я.

Їх ефективна робота має значний вплив на загальну продуктивність системи. Своєю чергою, продуктивність баз даних впливає на швидкість обробки запитів, доступність сервісів і загальний користувацький досвід. Низька продуктивність може призвести до зниження швидкості відгуку, затримок і навіть простоїв системи, що може відчутно вплинути на бізнес-результати.

Інструменти для моніторингу баз даних

1. LogicMonitor

LogicMonitor є платформою для моніторингу, яка забезпечує автоматичне виявлення ресурсів і аналіз стану баз даних.

Основною особливістю LogicMonitor є те, що вона підтримує моніторинг для широкого спектру баз даних, включаючи MySQL, PostgreSQL, Oracle та інші.

2. AppOptics

Основні особливості AppOptics

Можливості моніторингу в реальному часі. Завдяки моніторингу баз даних у режимі реального часу, використання платформи AppOptics дозволяє користувачам виявляти аномалії та оперативно реагувати на них.

А завдяки тому, що AppOptics легко інтегрується з іншими системами моніторингу, такими як Grafana і Prometheus, є можливість розширити можливості аналітики.

3. ScienceLogic

Однією з основних властивостей платформи є автоматизація моніторингу та управління. ScienceLogic забезпечує автоматизоване налаштування і управління процесом моніторингу баз даних.

Ще одна характеристика – це аналітика та звітність. Платформа надає розширені можливості аналітики та звітності для оцінки продуктивності баз даних і оптимізації їх роботи.

1.9.1. Відстеження продуктивності запитів та ресурсів

Моніторинг баз даних охоплює ключові аспекти продуктивності, що впливають на швидкість та стабільність системи, зокрема:

- моніторинг використання ресурсів (CPU, пам'ять, дисковий I/O). Моніторинг цих ресурсів дає змогу виявляти вузькі місця, які можуть впливати на продуктивність баз даних.
- аналіз продуктивності SQL-запитів. Оцінка часу виконання SQL-запитів допомагає виявити проблемні запити, які споживають забагато ресурсів або мають надмірний час виконання.
- оптимізація індексів та схем баз даних. Суттєво покращити продуктивність, зменшуючи навантаження на систему та прискорюючи час відгуку, може регулярний аналіз індексів і структур баз даних.

1.9.2. Переваги та недоліки інструментів моніторингу баз даних

Серед переваг інструментів моніторингу баз даних можемо виокремити такі:

- Здатність порівнювати функціональні можливості. Іінструменти для моніторингу баз даних часто пропонують розширені функції аналітики. За допомогою цих функцій є змога більш точно оцінити продуктивність і ефективність баз даних.
- Легкість налаштування та використання. Зручний інтерфейс для налаштування платформ для моніторингу баз даних, дає можливість швидко адаптувати інструменти до потреб системи.
- Інтеграція з іншими сервісами. Завдяки цій особливості є можливість вивантаження даних та візуалізації їх у Grafana або інших системах.

Недоліки:

Для ефективної роботи розширені інструменти для моніторингу баз даних можуть мати високу вартість ліцензії та вимагати додаткових налаштувань і ресурсів, особливо для великих організацій.

1.9.3. Приклади використання інструментів моніторингу баз даних

Наведемо приклади використання інструментів моніторингу баз даних

1. Електронна комерція. У системах електронної комерції завдяки моніторингу баз даних є можливість відстежувати час відгуку запитів, що важливо для обробки замовлень та транзакцій у режимі реального часу. Наприклад, одна з платформ e-commerce використовує ScienceLogic для постійного контролю стану баз даних, завдяки чому можна оперативно виявляти затримки і покращувати час відгуку.
2. Фінансові організації. Банки і фінансові компанії користуються LogicMonitor для моніторингу баз даних, що обробляють великі обсяги транзакційних даних. Завдяки тому, що ці інструменти можуть автоматично виявляти проблеми і в них є можливість налаштувавати оповіщення, компанія може своєчасно вирішувати проблеми, забезпечуючи стабільну роботу сервісів.
3. Системи IoT. У середовищах IoT часто використовують AppOptics для моніторингу баз даних, тому що це рішення забезпечує моніторинг у реальному часі. Компанія, що обслуговує тисячі IoT-пристроїв, використовує AppOptics для контролю ефективності роботи баз даних і збору метрик у масштабованій архітектурі.

Отож, підсумуємо: застосування інструментів для моніторингу баз даних допомагає підвищити продуктивність і стабільність роботи систем, забезпечуючи швидкий доступ до даних і зменшуючи час реагування на інциденти. У реальних кейсах моніторинг баз даних дозволяє компаніям знизити затримки в обробці даних та підвищити загальну ефективність. А це, своєю чергою, позитивно впливає на користувацький досвід і забезпечує стабільну роботу додатків.

1.10. Моніторинг додатків

Моніторинг продуктивності додатків (Application Performance Monitoring, APM) є критично важливим інструментом для забезпечення високої якості роботи додатків і задоволення користувацького досвіду. APM дозволяє виявляти проблеми продуктивності, відстежувати основні метрики, а також забезпечувати відповідність рівня обслуговування (SLA).

1.10.1. Значення моніторингу додатків

Продуктивність додатків безумовно впливає на досвід користувачів. Ключовими факторами, що впливають на задоволеність користувачів та їхню взаємодію з продуктом є швидкий час відгуку, відсутність помилок і стабільність додатків.

Також важливим фактором є забезпечення відповідності рівню обслуговування (SLA). Моніторинг додатків є важливим для забезпечення дотримання умов SLA, що містить в собі показники доступності, продуктивності і швидкості реакції на інциденти. Компанії, аби забезпечити відповідність заявленим зобов’язанням, можуть використовувати APM для контролю ключових метрик.

1.10.2. Інструменти APM (Application Performance Monitoring)

Коротко охарактеризуємо системи, в яких доступна функція Application Performance Monitoring

1. New Relic

Завдяки інструментам APM New Relic забезпечує моніторинг продуктивності додатків у реальному часі, що надає можливість відстежувати ключові показники продуктивності, як-от час завантаження та використання ресурсів. Також, з допомогою інструментів APM вебдодаток New Relic може здійснювати аналіз продуктивності та помилок. Він допомагає виявляти проблеми продуктивності та помилки, забезпечуючи детальний аналіз проблемних запитів і трасувань.

2. AppDynamics

Для цього рішення, завдяки Application Performance Monitoring доступний глибокий аналіз транзакцій. AppDynamics надає можливість глибоко аналізувати транзакцій. Це дає змогу відстежувати шлях кожного запиту через систему, виявляючи проблемні місця.

За допомогою інструментів AppDynamics компанії можуть оцінювати користувацький досвід і бачити, як користувачі взаємодіють із додатком, відстежуючи точки входу, переходи між сторінками та інші показники.

3. Dynatrace

Dynatrace використовує штучний інтелект з метою автоматичного виявлення аномалій та аналізу продуктивності. Це зменшує навантаження на DevOps-команди.

Завдяки тому, що Dynatrace надає повний огляд усієї інфраструктури, що містить аналіз взаємодії додатків з базами даних, серверами і мережею, користувачі мають можливість бачити повну картину роботи системи.

1.10.3. Моніторинг досвіду користувачів та продуктивності додатків

Перерахуємо основні показники, які можна отримати під час експлуатації систем моніторингу продуктивності додатків.

Моніторинг продуктивності додатків містить в собі контроль часу відгуку додатка, завантаження сторінок і обробки запитів. Ці характеристики є важливими для забезпечення швидкості роботи додатків.

APM інструменти дають змогу відстежувати взаємодію користувачів із додатками, що дає можливість розуміти, як користувачі взаємодіють із системою і які компоненти потребують покращення. Таким чином здійснюється аналіщ поведінки користувачів.

Аналіз продуктивності додатків дозволяє виявляти місця, які сповільнюють роботу додатків, такі як проблемні SQL-запити, надмірне використання ресурсів або низька пропускна здатність мережі.

1.10.4. Переваги та недоліки інструментів APM

Виокремимо основні переваги інструментів APM:

По-перше, це порівняння можливостей та функціональності. APM інструменти, такі як New Relic, AppDynamics та Dynatrace, забезпечують широкий набір функцій для детального аналізу продуктивності, відстеження помилок та покращення користувацького досвіду.

Наступна перевага – інтеграція з DevOps процесами. Адже більшість інструментів APM інтегруються з CI/CD-процесами. Це дозволяє виявляти проблеми на ранніх етапах розробки та сприяє їх швидшому вирішенню.

Не менш важливою особливістю є використання штучного інтелекту та автоматизації Деякі інструменти, такі як Dynatrace, використовують AI для автоматизації аналізу продуктивності, що прискорює процес виявлення та усунення проблем.

Серед недоліків інструментів APM є такі:

- Вартість володіння та впровадження. Для організацій, в яких моніторинг розгорнуто на багатьох серверах або віртуальних машинах, інструменти APM можуть мати високу вартість ліцензії.
- Деякі інструменти можуть вимагати високої кваліфікації для налаштування і підтримки, особливо під час інтеграції з нестандартними додатками або складними середовищами.

1.10.5. Приклади використання інструментів APM

Досить часто ці інструменти використовує фінансовий сектор. Наприклад, одна з фінансових компаній використовує New Relic для моніторингу банківських додатків у режимі реального часу. Це дає можливість миттєво відслідковувати аномалії та забезпечувати високу продуктивність сервісів для клієнтів.

Також AppDynamics активно використовується у телекомунікаційній сфері з метою моніторингу складних транзакцій та аналізу користувацького досвіду. Завдяки цьому компанії можуть забезпечувати стабільність зв’язку та відповідність умовам SLA.

Наступний приклад – E-commerce платформи. Відомий інтернет-магазин впровадив Dynatrace, щоб забезпечити високу швидкість завантаження сторінок і швидку обробку запитів. Завдяки автоматичному аналізу, компанія змогла виявити та усунути проблеми продуктивності. Це, безумовно, позитивно вплинуло на задоволеність користувачів та конверсію.

Отож, можемо зробити висновок, що застосування APM інструментів для моніторингу додатків дає можливість компаніям забезпечувати стабільність і високу продуктивність додатків. А це своєю чергою призводить до підвищення задоволеності користувачів та покращення бізнес-результатів. Успішні кейси використання APM демонструють значні переваги, такі як зменшення часу простоїв, підвищення швидкості роботи додатків і відповідність заявленим умовам обслуговування (SLA).

1.11. Аналізатори трафіку

Досліджувати мережеві пакети, виявляти проблеми з мережею, а також забезпечувати безпеку IT-інфраструктури можна завдяки аналізаторам трафіку. До найпопулярніших інструментів для аналізу трафіку відносяться Wireshark, NetFlow та sFlow.

Wireshark

Wireshark є одним із ефективних інструментів для аналізу мережевого трафіку, який дозволяє захоплювати і досліджувати пакети на різних рівнях мережевої моделі.

Особливостями цього інструменту є:

- Деталізація пакетів. Wireshark дає можливість бачити всі деталі мережевих пакетів, включно із заголовками, IP-адресами відправника та отримувача, портами та вмістом переданих даних.
- Підтримка різних протоколів. Wireshark підтримує широкий спектр мережевих протоколів. Це дозволяє аналізувати мережі різних типів.

- Фільтри для аналізу. Інструмент надає фільтри, з допомогою яких можна обмежувати видимість тільки до певних типів пакетів, що спрощує процес аналізу.

NetFlow та sFlow

NetFlow і sFlow – це протоколи, що розроблені для збору статистики мережевого трафіку та моніторингу продуктивності мережі.

NetFlow надає такі можливості:

- Збір інформації про потоки. NetFlow дозволяє збирати інформацію про IP-пакети, що проходять через маршрутизатор або комутатор, зберігаючи такі параметри, як IP-адреси джерела і призначення, порти, розмір пакета та час передачі.
- Підтримка Cisco. NetFlow був розроблений компанією Cisco і часто використовується на її мережевих пристроях.

Своєю чергою sFlow забезпечує семплування трафіку та підтримку багатьох вендорів. Щодо семплування: sFlow відрізняється від NetFlow тим, що використовує семпли пакетів замість запису всіх потоків. Це знижує навантаження на процесор і дає можливість зберігати менше даних, що зручно для великих мереж. Також sFlow є відкритим стандартом, що підтримується багатьма мережевими виробниками, включаючи Cisco, Juniper та інші.

1.11.1. Переваги та обмеження аналізаторів трафіку

Аналізатори трафіку мають ряд значних переваг, таких як:

- Деталізація даних.Ааналізатори трафіку, наприклад, такі як Wireshark, забезпечують глибокий аналіз мережевих пакетів. Це дає можливість отримувати точні дані про кожен пакет та виявляти проблеми.
- Інтеграція з іншими інструментами. Протоколи NetFlow і sFlow легко інтегруються із системами моніторингу, що дозволяє отримувати загальну картину мережевої активності.
- Моніторинг у реальному часі. Завдяки моніторингу в реальному часі можна спостерігати за мережею та своєчасно виявляти проблеми.

Але також є й обмеження. Для користувача без належного рівня кваліфікації використання інструментів на рівні пакетів, таких як Wireshark, дані, зібрані аналізаторами трафіку, можуть бути складними для інтерпретації. Також у великих мережах детальний збір даних (особливо повний захват трафіку, як у Wireshark) може вимагати значних ресурсів і знизити продуктивність.

1.11.2. Приклади застосування аналізаторів трафіку для моніторингу

За допомогою Wireshark мережеві адміністратори можуть ідентифікувати затримки в передачі пакетів, помилки та інші проблеми. Наприклад, у випадку підвищеного часу відгуку мережі Wireshark допомагає виявити, на якому рівні або на якому маршрутизаторі виникає проблема.

Також NetFlow і sFlow використовуються для моніторингу аномальної активності, наприклад, такої як підозрілі обсяги трафіку або непередбачені IP-з’єднання. Наприклад, у разі можливих DoS-атак NetFlow може допомогти виявити надмірну кількість з’єднань з певного IP.

Тож, варто розуміти що аналізатори трафіку є необхідним інструментом для мережевих фахівців, адже з їх допоиогою можна виявляти проблеми продуктивності та загрози безпеці, а також забезпечувати високу якість роботи мережевих сервісів.

Висновок

У першому розділі здійснено комплексний огляд сучасних рішень для моніторингу IT-інфраструктури, що охоплює як класичні інструменти, так і новітні хмарні й модульні платформи. Інформація, викладена у розідлі, підтверджує наявність широкого спектра інструментів моніторингу, кожен з яких пропонує специфічні функціональні можливості та підходи до аналізу даних. Це різноманіття рішень дає можливість вибудовувати системи моніторингу з урахуванням унікальних потреб організацій, водночас ускладнюючи вибір через необхідність глибокого розуміння характеристик і обмежень кожного інструмента.

Можемо зробити висновок, що вибір оптимального інструмента моніторингу залежить від численних чинників, серед яких масштабованість, продуктивність, функціональні можливості, сумісність з іншими системами та вартість володіння. У кожного підходу є свої переваги та недоліки, що зумовлюють вибір конкретного рішення для окремих сценаріїв використання. Для підприємств, які працюють з розподіленими середовищами, значну перевагу становлять модульні платформи з підтримкою масштабування та високим рівнем інтеграції, тоді як класичні рішення можуть залишатись актуальними для невеликих інфраструктур з обмеженими потребами.

Отже, можемо сформулювати такий висновок: ринок інструментів моніторингу пропонує широкий вибір, однак визначення оптимального стека технологій потребує врахування конкретних потреб та архітектурних особливостей об’єкта моніторингу.

Детальний порівняльний аналіз функціональних можливостей і характеристик кожного рішення буде проведено в наступному розділі, що дасть змогу зробити обґрунтований висновок про переваги і недоліки кожного підходу для застосування у різних умовах.

РОЗДІЛ 2: ПОРІВНЯЛЬНИЙ АНАЛІЗ ТЕХНОЛОГІЙ МОНІТОРИНГУ

2.1. Методологія порівняльного аналізу

Для ефективного порівняння рішень з моніторингу IT-інфраструктури важливо визначити ключові критерії, що дадуть можливість об’єктивно оцінити кожен інструмент, а також обрати методи для глибокого аналізу та тестування. Цей розділ окреслює критерії порівняння та методи оцінки, що використовуватимуться в аналізі рішень.

2.1.1. Визначення критеріїв порівняння

Першим критерієм для порівняння є масштабованість. За цією характеристикою оцінюється здатність системи адаптуватися до змін у масштабі — як вертикальному (підвищення обсягів даних в межах одного вузла), так і горизонтальному (додавання нових вузлів у кластер). Цей критерій важливий для великих IT-інфраструктур, де моніторинг потребує значних ресурсів та швидкого масштабування.

Наступний критерій – продуктивність. Продуктивність системи моніторингу визначається її здатністю обробляти великий обсяг даних з мінімальною затримкою. Для цього важливі швидкість збору та зберігання метрик, реакція на запити та здатність системи підтримувати стабільність під високим навантаженням.

Також важливо, щоб система підтримувала широкий набір функцій для моніторингу — від збору даних і зберігання до візуалізації та створення сповіщень. Беруться до уваги можливості налаштування метрик, інтеграція з іншими системами, підтримка різних форматів даних та додаткові можливості, такі як автоматизація та налаштування інтерфейсу.

Такий критерій як легкість інтеграції оцінюється за здатністю інструмента легко інтегруватися з іншими компонентами системи. Це містить підтримку стандартних протоколів і форматів (наприклад, Prometheus, OpenTelemetry), можливість використання існуючих дашбордів або експорт даних до інших аналітичних платформ.

Не менш важливим критерієм є вартість володіння та підтримки Він враховує загальні витрати на придбання, встановлення, обслуговування та масштабування системи. До вартості володіння входять ліцензія на програмне забезпечення, ресурси, необхідні для роботи системи, а також витрати на підтримку і навчання персоналу.

Останні з критеріїв – це спільнота та підтримка. Відкрита й активна спільнота розробників є важливим фактором для проектів з відкритим кодом, оскільки забезпечує стабільний розвиток, регулярні оновлення та зворотний зв'язок. Також важливою є доступність офіційної технічної підтримки, що сприяє оперативному вирішенню проблем під час впровадження і роботи системи.

2.1.2. Опис методів оцінки

Опишемо основні методи оцінки. Перший метод – кількісний аналіз метрик. Він містить в собі вимірювання основних метрик системи, таких як час відгуку, затримка збору даних, обсяг збережених даних та використання ресурсів. Кожна система тестується за однаковими параметрами, щоб забезпечити об'єктивну оцінку продуктивності та масштабованості.

Другий метод – якісний аналіз функціональності. Він охоплює огляд функціональності та можливостей налаштування інструменту. Розглядаються, зокрема, гнучкість налаштувань, підтримка різних джерел даних, інтуїтивність інтерфейсу та зручність користування. Додатково оцінюються можливості автоматизації та роботи з нестандартними метриками.

Третім методом є тестування в реальних умовах. Аби максимально об'єктивно оцінити продуктивність та надійність системи проводиться тестування в умовах, максимально наближених до реальних. Це допомагає визначити, як система поводиться під час пікових навантажень, як реагує на інциденти і чи стабільно працює за умов зміни параметрів.

Враховується і огляд документації та відгуків користувачів. Він допомагає зрозуміти рівень підтримки продукту та наявність інструментів для інтеграції й налаштування. З оглядом відгуків користувачів можна оцінити реальний досвід

експлуатації системи, особливо в контексті складності впровадження, швидкості вирішення проблем та досвіду роботи з підтримкою.

2.2. Порівняння систем збору та зберігання метрик

Існує багато баз даних часових рядів, таких як Prometheus, M3DB, TimescaleDB, OpenTSDB, InfluxDB тощо. У першому розділі коротоко були описані Prometheus VictoriaMetrics, Mimir, у другому ж розділі ми познайомимося з ними більш детально, тож – це бази даних часових рядів із відкритим кодом, які забезпечують надійні рішення для моніторингу та оповіщення в складних IT-середовищах. Однак вони розроблені по-різному та пропонують унікальні функції, які можуть вплинути на їх продуктивність, масштабованість і простоту використання для моніторингу робочих навантажень. Цей розділ має на меті проаналізувати відмінності між Prometheus, VictoriaMetrics та Mimir таким чином надаючи розуміння, який продукт є найбільш прийнятним рішенням для моніторингу та спостереження або усунення несправностей у своїх системах.

Prometheus, який розпочався як проєкт у межах SoundCloud, є потужним інструментом моніторингу та попередження, створеним спеціально для роботи з даними часових рядів у багатовимірному середовищі. Він набув надзвичайної популярності серед спільноти SRE та DevOps завдяки вбудованій підтримці збору багатовимірних даних, запитів і створення сповіщень.

Prometheus розроблено в рамках Cloud Native Computing Foundation (CNCF). Сервер Prometheus, клієнтські бібліотеки, Alertmanager та інші пов’язані компоненти можна знайти в організації Prometheus GitHub.

VictoriaMetrics, з іншого боку, є високопродуктивною, економічно ефективною та масштабованою базою даних часових рядів, яку можна використовувати як довгострокове віддалене сховище для Prometheus. Він може похвалитися чудовим стисненням даних і високою швидкістю прийому даних, що робить його привабливою альтернативою для великомасштабних завдань моніторингу.

Mimir, створений компанією Grafana Labs, є високоефективною та масштабованою системою зберігання часових рядів, що була розроблена для роботи з великими обсягами метрик у складних розподілених середовищах. Mimir є результатом подальшого розвитку досвіду, отриманого з Cortex, і містить в собі значні покращення щодо зберігання, масштабування та керування часовими рядами. Також продукт орієнтований на довгострокове зберігання метрик і інтегрується з Prometheus, забезпечуючи повну підтримку PromQL для гнучких запитів. Він оптимізований для роботи в хмарних середовищах, забезпечуючи відмовостійкість, горизонтальне масштабування та зниження витрат на зберігання великих обсягів даних.

Архітектура Prometheus

Prometheus використовує модель на основі витягування для збору показників, тобто він отримує показники з контрольованих систем і може обробляти до мільйонів активних часових рядів. Ця архітектура спрощує розгортання контрольованих служб. Тим не менш, керування великими динамічними середовищами може бути складним завданням, оскільки екземпляри Prometheus повинні знати цілі, з яких їм потрібно працювати. На рис. 2.1 зображено архітектуру Prometheus:

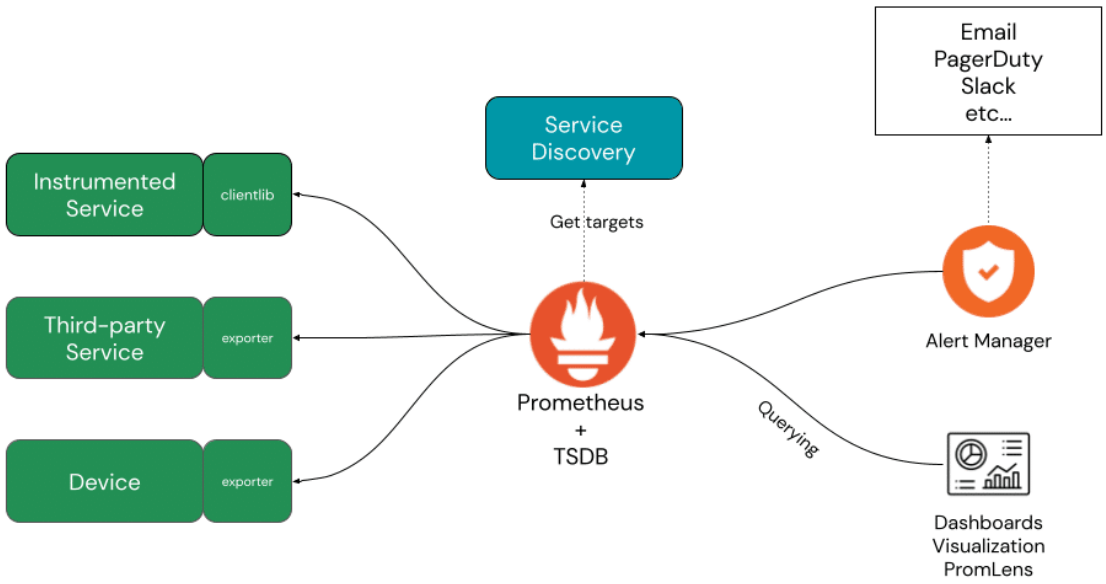


Рис. 2.1

Prometheus створено за допомогою поєднуваної архітектури, яка складається з кількох ключових компонентів, зокрема:

- Сервер Prometheus. Серце системи, сервер Prometheus, відповідає за збирання та зберігання даних часових рядів.
- Клієнтські бібліотеки. Ці бібліотеки надають показники з коду програми. Prometheus надає клієнтські бібліотеки для кількох мов, включно з Go, Java, Python тощо.
- Експортери. Ці служби HTTP надають показники у форматі, який Prometheus може отримати. Експортери доступні для систем сторонніх виробників, таких як HAProxy, MySQL або таких, як системна статистика Linux.
- Pushgateway. Для послуг, які неможливо позбутися (наприклад, короткочасні роботи), Prometheus надає Pushgateway. Це дозволяє ефемерним і пакетним завданням надавати свої показники Prometheus.
- Alertmanager. Цей компонент керує сповіщеннями, видаленням дублікатів і групами, а також надсилає сповіщення електронною поштою, PagerDuty або OpsGenie. В його функціонал входять також вимикання та блокування сповіщень.
- PromQL. Це вбудована гнучка мова запитів Prometheus для дослідження даних і приладної дошки, відмінна від SQL.
- Виявлення служб. Prometheus підтримує різноманітні механізми виявлення служб, які допомагають знаходити цілі, які потрібно скинути.

Архітектура VictoriaMetrics

VictoriaMetrics, навпаки, підтримує як pull, так і push модель. Його здатність обробляти великі обсяги даних і розширені мережеві сценарії (завдяки підтримці моделі push) робить його масштабованим і гнучким. Він кластеризується нативно, що спрощує тривале зберігання та широкомасштабне розгортання. На рис. 2.2 зображено архітектуру VictoriaMetrics:

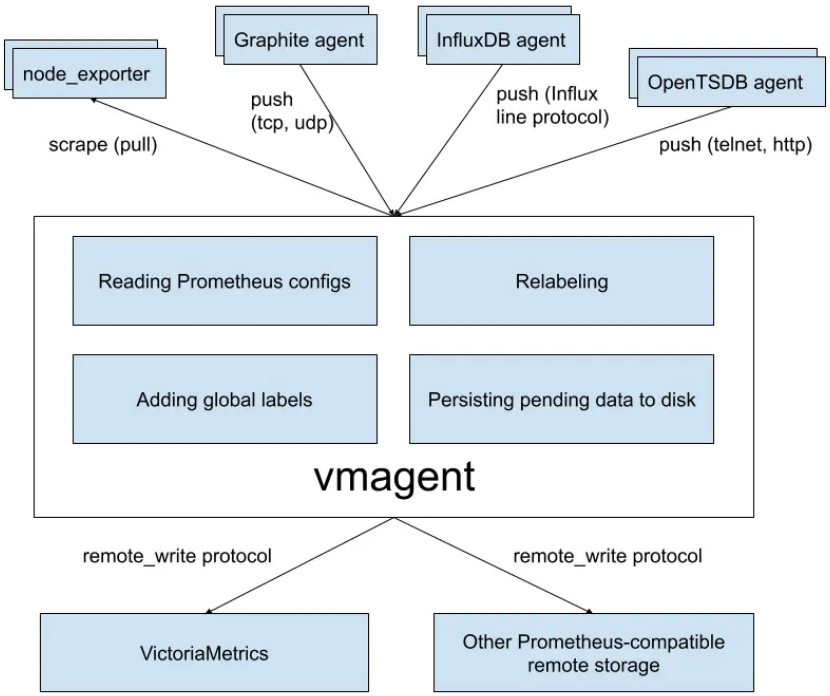


Рис. 2.2

VictoriaMetrics, зберігаючи простішу архітектуру, також включає кілька основних компонентів:

- VictoriaMetrics з одним вузлом – це базовий будівельний блок VictoriaMetrics. Він містить в собі базу даних часових рядів і HTTP-сервер для прийому та запиту даних.
- Кластер VictoriaMetrics – є кластерною версією VictoriaMetrics, яка розроблена для широкомасштабного розгортання, включає додаткові компоненти, такі як вузли VMSelect, VMInsert і VMStorage для виконання запитів, прийому даних і довготривалого зберігання відповідно.
- Vmagent – це крихітний, але потужний інструмент для збирання даних, який може отримувати дані з різних джерел і надсилати їх до VictoriaMetrics або будь-якого іншого віддаленого сховища, яке підтримує протокол віддаленого запису InfluxDB або Prometheus.

- Інструмент для налаштування правил спрацювання сповіщень – vmalert. Він оцінює провила стовіщення і запису щодо VictoriaMetrics або будь-якої іншої сумісної TSDB.
- Інструмент CLI – vmctl, який переносить дані з різних TSDB до VictoriaMetrics.
- Бекенд зберігання для VictoriaMetrics - vmstorage.
- Рівень інтерфейсу користувача - vmui, який поставляється з VictoriaMetrics.
- Утиліта для резервного копіювання даних – vmbackup. Вона дозволяє створювати повні або інкрементні резервні копії даних у сумісне сховище, забезпечуючи безпечне зберігання даних та можливість їх відновлення у разі збою системи.

Архітектура Mimir

Mimir також підтримує обидві моделі збору метрик, перша на основі витягування (pull), друга штовхання (push), що забезпечує гнучкість у виборі способу отримання даних. Завдяки своїй кластерній архітектурі, Mimir пропонує високу масштабованість і здатність обробляти великі обсяги даних часових рядів. Це робить його релевантним для широкомасштабних розгортань, особливо у середовищах з високою динамікою, де кількість джерел даних може швидко змінюватися. На відміну від Prometheus, Mimir оптимізовано для довготривалого зберігання даних і розширеного аналізу, забезпечуючи високу продуктивність і зручність у роботі з великими наборами часових рядів. Його архітектура містить додаткові компоненти для розподіленого зберігання та запитів, що полегшує інтеграцію в складні середовища та забезпечує надійність навіть під час високих навантаженнях. На рис. 2.3 зображено архітектуру Mimir:

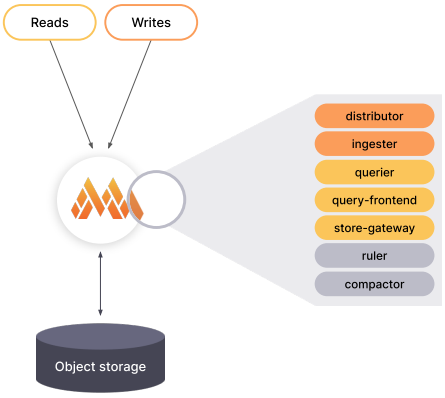


Рис. 2.3

Основні компоненти архітектури Mimir містять:

- Ingestors – приймають метрики від Prometheus, клієнтів remote_write та інших джерел, зберігають їх і розподіляють по інших компонентах для обробки та зберігання.
- Distributors – відповідають за маршрутизацію запитів і розподіл навантаження між іншими компонентами для забезпечення масштабованості та високої продуктивності.
- Compactors – періодично об’єднують старіші блоки даних для зменшення обсягу зберігання та оптимізації запитів, підтримуючи ефективне довготривале зберігання метрик.
- Store-gateways – зберігають та обробляють запити до історичних даних, забезпечуючи низьку затримку запитів навіть при великому обсязі інформації.
- Querier та Query Frontend – дають змогу виконувати запити до збережених метрик, забезпечуючи швидкий доступ до даних для аналізу та візуалізації в реальному часі.

Ефективність стиснення та зберігання даних

Prometheus також має ефективну систему зберігання, яка не збігається з VictoriaMetrics щодо довготривалого зберігання даних та ефективності пошуку.

Проте, існують інші альтернативи, такі як Levitate, Thanos, Cortex і Mimir, які також забезпечують тривале збереження даних.

Однією з основних переваг VictoriaMetrics перед Prometheus є його можливості стиснення даних. Він використовує ефективніший алгоритм стиснення даних, що значно зменшує вимоги до пам’яті. VictoriaMetrics стверджує, що забезпечує до 10 разів більше стиснення даних, ніж Prometheus, що є важливою перевагою для тривалого збереження даних і оптимізації витрат.

Mimir, зі свого боку, пропонує високу масштабованість і стабільність у великих середовищах, поєднуючи переваги розподіленого зберігання з оптимізацією для роботи з великими наборами часових рядів. Його архітектура спрямована на забезпечення високої продуктивності запитів і можливості ефективного довготривалого зберігання, що робить його конкурентоспроможним рішенням серед інших систем для роботи з даними часових рядів.

Prometheus, VictoriaMetrics та Mimir використовують комбінацію обробки даних у пам’яті та зберігання на диску для керування даними часових рядів.

Prometheus використовує сховище в пам’яті для негайного доступу до останніх даних часового ряду. Цей сегмент бази даних відомий як «головний блок». Щодо дискового зберігання після досягнення певного віку або розміру дані в головному блоці переміщуються на диск за допомогою процедури, відомої як режим checkpointingреального часу. Ця база даних складається з «постійних блоків», призначених для довгострокового зберігання. Prometheus оптимізовано для моніторингу в реальному часі за допомогою цієї комбінації пам’яті та дискового сховища. Однак він не призначений для тривалого тривалого зберігання.

Подібно до Prometheus, VictoriaMetrics використовує сховище в пам’яті для буферизації вхідних даних перед їх записом на диск. Цей підхід допомагає оптимізувати продуктивність запису. Він також кешує дані, до яких часто звертаються, для швидшого пошуку. Основна частина даних у VictoriaMetrics зберігається на диску. У системі використовується формат зберігання, що економить простір і забезпечує значне стиснення даних.

Щодо Mimir – ця платформа використовує обробку даних у пам’яті для забезпечення швидкого доступу до недавніх часових рядів. Це забезпечує високу швидкість обробки запитів і дає змогу швидко реагувати на зміни метрик. Завдяки своїй розподіленій архітектурі, Mimir використовує сховища на диску для тривалого зберігання даних. Завдяки цьому системі вдається обробляти масштабні набори часових рядів у великих інфраструктурах. Його інтеграція з обчислювальними платформами, такими як Kubernetes, підвищує масштабованість і відмовостійкість системи.

Мова запитів

Prometheus використовує PromQL (мову запитів Prometheus). PromQL дає змогу вибирати та агрегувати дані часових рядів у режимі реального часу. Це уможливлює роботу розробників з показниками з високою гнучкістю. За допомогою PromQL користувачі можуть фільтрувати та агрегувати показники, обчислювати ставки, коефіцієнти, середні значення та процентилі, а також прогнозувати тенденції. Після освоєння PromQL, користувачі розуміють, що це дуже виразна мова, яка дає можливість виконувати складні запити для збору значущих даних зі своїх показників.

VictoriaMetrics, з іншого боку, зворотно сумісна з PromQL. Він може виконувати будь-який запит, дійсний у PromQL. Однак він також представляє розширення PromQL під назвою MetricsQL. MetricsQL розроблено для розширення можливостей запитів, наданих PromQL. Він представляє нові функції, оператори та полегшену синтаксичну форму. Це спрощує та покращує взаємодію з користувачем, особливо для складних запитів та агрегацій.

Mimir також підтримує запити на основі PromQL, що робить його повністю сумісним із екосистемою Prometheus. Крім цього, Mimir оптимізує виконання запитів за допомогою механізмів кешування та розподілу обчислень. Завдяки розподіленій архітектурі Mimir робить можливим виконання запитів високої складності з мінімальними затримками навіть у великих середовищах. На відміну від VictoriaMetrics, яка вводить MetricsQL для розширення функціоналу PromQL,

Mimir зберігає основний фокус на стандарті PromQL, гарантуючи максимальну сумісність із існуючими інструментами та дашбордами, розробленими для Prometheus.

Висока доступність і надійність

Prometheus за своєю суттю не підтримує кластеризацію, що означає, що він не забезпечує власну високу доступність. Високої доступності можна досягти шляхом запуску дублікатів, але цей процес вимагає ручних зусиль і координації.

На відміну від цього, VictoriaMetrics розроблено з урахуванням високої доступності. Він використовує реплікацію та кластеризацію, щоб гарантувати, що дані не будуть втрачені в разі збою екземпляра, що робить його більш надійним варіантом для критичних програм.

Mimir за замовчуванням підтримує кластеризацію та високу доступність, що є його однією з ключових переваг. Він використовує горизонтальне масштабування та реплікацію даних для забезпечення відмовостійкості. Це дозволяє Mimir працювати ефективно навіть у великих середовищах з високими вимогами до продуктивності та доступності.

Таблиця порівняння ключових характеристик.

Порівняння	**Prometheus**	**Victoriametrics**	**Mimir**
Збір даних	На основі тяги	На основі Pull та Push	На основі Pull
Поглинання даних	До 240 000 вибірок в секунду	До 360 000 вибірок в секунду	Залежить від масштабування
Запит даних	До 80 000 запитів в секунду	До 100 000 запитів в секунду	Оптимізовано для великого навантаження
Використання пам’яті	До 14 ГБ оперативної пам’яті	До 4,3 ГБ оперативної пам’яті	Вимагає більше пам'яті в залежності від масштабування

Порівняння	Prometheus	Victoriametrics	Mimir
Стиснення даних	Використовує стиснення LZF	Використовує миттєве стиснення	Використовує схему зберігання "chunks"
Частота запису на диск	Частіше записує дані на диск	Рідше записує дані на диск	Залежить від конфігурації
Використання дискового простору	Потрібно більше місця на диску	Вимагає менше місця на диску	Оптимізовано для розподіленого зберігання
Мова запитів	PromQL	MetricsQL (зворотно сумісний із PromQL)	PromQL

Таб. 2.1

Результати порівняння

Аналіз показує, що кожна з розглянутих систем зберігання метрик має свої унікальні особливості та підходить для певних сценаріїв.

- Prometheus. Найбільш підходить для невеликих та середніх середовищ, де немає потреби у високій масштабованості. Його простота у використанні, сумісність з PromQL та широка підтримка спільноти роблять його популярним для багатьох DevOps-практик. Однак, при значному збільшенні обсягу даних Prometheus може не впоратися з навантаженням, і це стає його основним обмеженням.
- VictoriaMetrics. Підходить для середніх і великих середовищ, особливо там, де є потреба в ефективному використанні ресурсів. Її архітектура забезпечує стиснення даних, що дає можливість зберігати великі обсяги метрик з мінімальним використанням пам’яті. Наявність VMTools для

спрощеного управління метриками робить VictoriaMetrics гнучким і зручним рішенням для різноманітних середовищ, сумісних з Prometheus.

- Mimir. Створений для великих середовищ з високими вимогами до масштабованості та продуктивності. Завдяки розподіленій архітектурі, Mimir може забезпечувати низькі затримки для запитів і обробляти великі обсяги метрик навіть у масштабних хмарних інфраструктурах. Однак його розгортання вимагає значних ресурсів і може бути дорогим в обслуговуванні, що обмежує його використання для невеликих компаній.

Вибір системи зберігання метрик залежить від розміру інфраструктури та вимог до масштабованості. Для малих і середніх середовищ Prometheus є оптимальним вибором завдяки простоті та низькій вартості. VictoriaMetrics підходить для середовищ з великим обсягом даних завдяки ефективності стиснення та оптимізованому зберіганню. Mimir ідеально підходить для великих компаній та розподілених систем з високими вимогами до продуктивності, хоча й вимагає більше ресурсів для обслуговування.

2.3. Порівняння систем збору та аналізу логів

Два популярні інструменти, які часто використовуються для збору та аналізу логів – це ELK стек та Loki. Кожне із цих рішень має свої сильні сторони та унікальні функції, що задовольняють різні потреби та вподобання.

У цьому підрозділі буде надано детальне порівняння цих інструментів, щоб обрати найрелевантніше рішення для побудови системи моніторингу.

Як вже було описано у першому розділі, ELK стек – це добре відоме рішення для журналювання, що складається з трьох основних інструментів: Elasticsearch, Logstash і Kibana. Це тріо працює в тандемі, щоб збирати, аналізувати та візуалізувати дані журналу. Elasticsearch обслуговує можливості зберігання та пошуку, Logstash відповідає за обробку та прийом журналів, а Kibana надає потужний функціонал для візуалізації отриманих даних.

У той же час, Loki відносно новий інструмент у сфері журналювання, представлений Grafana Labs у 2018 році. Loki розроблений як економний і масштабований, зосереджений на індексуванні журналів у більш ефективний спосіб. На відміну від традиційних систем керування журналами, які індексують повний текст журналів, Loki індексує лише метадані журналу та використовує сховища об'єктів, такі як Amazon S3, для зберігання фрагментів даних журналу. Це робить його більш ефективним з використанням ресурсів і легшим для управління в масштабі.

Архітектура Loki

Архітектура Loki створена на базі Prometheus. Він використовує подібний підхід до збору та зберігання журналів, що робить його природним вибором для користувачів, які вже знайомі з Prometheus. Архітектура Loki зображена на рисунку 2.4.

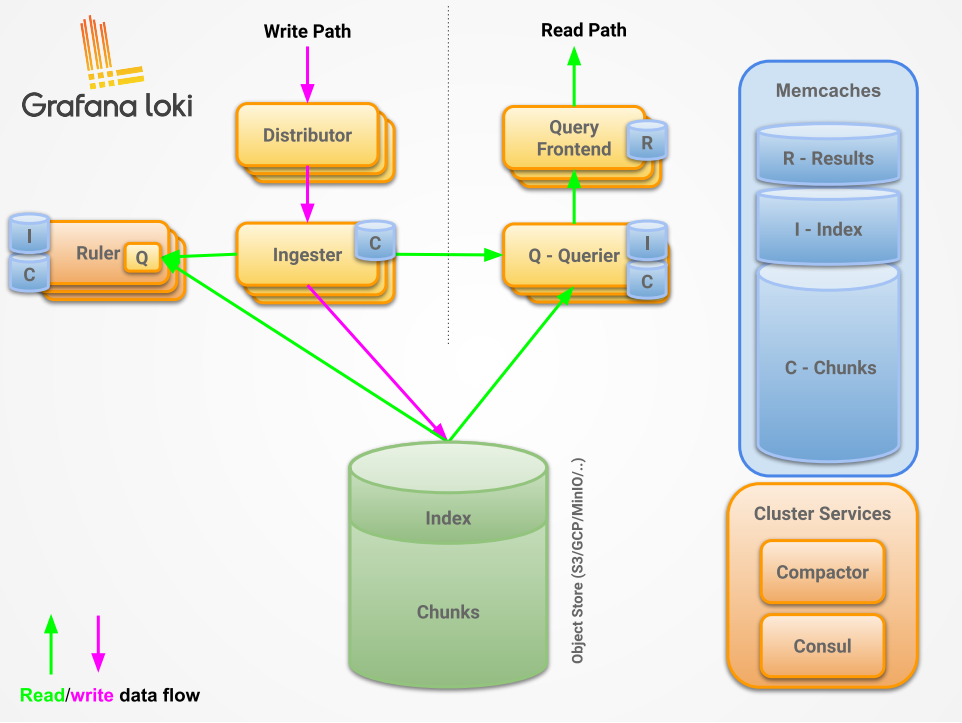


Рис. 2.4

Основними компонентами Loki є:

- Клієнти – це агенти, які збирають журнали із системи. Основним прикладом є Promtail, але Loki також підтримує інтеграцію з іншими клієнтами, такими як Fluentd або syslog.

- Distributor, який приймає вхідні журнали від клієнтів і виконує попередню обробку, таку як перевірка та парсинг даних. Також він розподіляє записи логів між іншими компонентами кластеру за допомогою механізму хешування.
- Ingester, своєю чергою, зберігає отримані журнали в оперативній пам’яті для швидкого доступу та обробки, а для довготривалого зберігання, у певні інтервали часу, здійснює збереження даних на дисковий простір.
- Сховище чанків відповідає за зберігання даних журналів у вигляді стиснених блоків, які забезпечують ефективність використання дискового простору.
- Querier виконує обробку запитів користувачів або системи, витягуючи дані журналів зі сховища або з пам’яті інджестера.
- Query Frontend виконує роль посередника між користувачами та компонентами Loki. Оптимізує та розподіляє запити між декількома серверами Querier, що підвищує продуктивність системи.

Ця архітектура забезпечує масштабованість і високу доступність Loki, даючи змогу ефективно обробляти великі обсяги даних журналів у розподілених середовищах.

Архітектура ELK stack

Через те, що ELK Stack складається з трьох окремих компонентів, відповідно, він має більш складну архітектуру ніж Loki. Основними компонентами є:

- Elasticsearch: розподілена пошукова система, що зберігає журнали та надає можливості пошуку.
- Logstash: інструмент конвеєра журналу, який отримує дані з різних джерел, обробляє їх та надсилає в Elasticsearch.
- Kibana: інструмент візуалізації та дослідження для аналізу та візуалізації даних, що зберігаються в Elasticsearch.

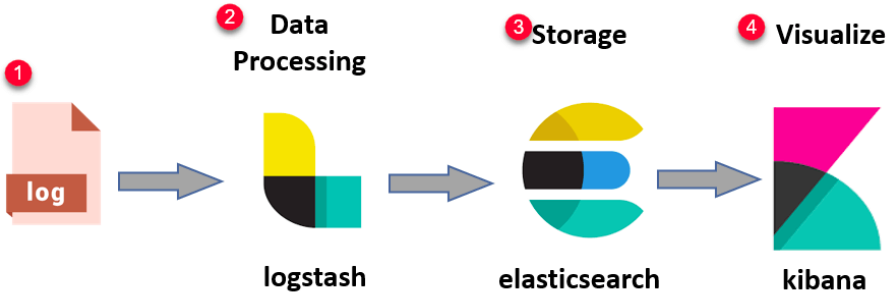


Рис. 2.5

Кожен із цих компонентів був більш детально описаний в першому розділі. Підсумовуючи, можна сказати, що ELK Stack є потужним та універсальним набором інструментів для збирання, обробки й аналізу даних, але через складність налаштування та високі вимоги до ресурсів, може бути важчим у впровадженні, порівняно з альтернативами.

Індексація та зберігання логів

Унікальний підхід Loki до індексування лише метаданих робить його високоефективним з точки зору зберігання та використання ресурсів. Такий вибір конструкції зменшує накладні витрати на роботу та робить Loki придатним для середовищ із великим об’ємом журналу. Журнали зберігаються у спосіб, подібний до показників Prometheus, у фрагментах та індексуються мітками, які є парами ключ-значення, визначеними користувачем.

Стек ELK, навпаки, індексує повний текст журналів, що забезпечує більш детальний пошук, але ціною вищих вимог до пам’яті та споживання ресурсів. Elasticsearch використовує інвертований індекс для полегшення швидких пошукових операцій, а Logstash може обробляти та перетворювати журнали до їх індексування.

Мова запитів

Loki використовує власну мову для запитів журналів, яка називається LogQL. Кожен запит складається з двох частин: селектора потоку журналу та виразу фільтра. Продуктивність виконання запиту залежить від того, скільки міток вибрано для фільтрації потоків журналів.

Якщо ви використовуєте Kibana для візуалізації даних журналу з Elasticsearch, ви можете використовувати мову запитів Kibana (KQL). Ви також можете вимкнути Kibana Query Language і замість цього використовувати синтаксис Lucene query syntax(формат для створення запитів).

І LogQL, і KQL мають певну криву навчання, і від користувачів залежить, наскільки швидко вони звикнуть до них. Оскільки KQL набагато старший, ви можете знайти більше прикладів в Інтернеті, щоб вивчити та застосувати їх для свого випадку використання.

Ефективність витрат

Ефективна модель індексування та зберігання даних Loki означає зниження витрат на інфраструктуру. Індексуючи лише метадані, Loki зменшує обсяг необхідних сховища та обчислювальних ресурсів, що робить його економічно ефективним рішенням для керування великими журналами.

Щодо ELK стеку, його повнотекстове індексування забезпечує більш детальні можливості пошуку, але за більшу вартість. Збільшення вимог до сховища та обчислень може призвести до вищих операційних витрат. Це може призвести до неідеальної ситуації при масштабуванні інфраструктури та ресурсів.

Тож вибір між Loki та Elasticsearch залежить від конкретних випадків використання та доступних ресурсів. Хоча Loki ефективно використовує ресурси, він не надає широкі пошукові можливості, які, натомість, доступні в Elasticsearch. Під час вибору потрібно також врахувати вартість і вимоги до ресурсів. Elasticsearch потребує більше ресурсів порівняно з Loki.

У Таб. 2.2 порівняно основі аспекти Loki та ELK стеку.

Характеристика	**Loki**	**ELK**
Архітектура	Спрощена, орієнтована на логування з мінімальним	Багатокомпонентна, з індексацією даних у Elasticsearch і потоковою

Характеристика	Loki	ELK
	індексуванням	обробкою в Logstash
Зберігання даних	Зберігає лише метадані, що знижує потреби у зберіганні	Індексує та зберігає усі дані, що вимагає значних ресурсів
Продуктивність	Висока продуктивність завдяки оптимізації для потокового логування	Потужна, але може бути ресурсомісткою через індексацію
Масштабованість	Добре масштабується для великих обсягів логів	Масштабування можливе, але вимагає налаштування для великих інфраструктур
Функціональність	Орієнтований на легку інтеграцію з Grafana для візуалізації	Повний стек з візуалізацією в Kibana і аналізом у Logstash
Інтеграція	Легка інтеграція з Grafana для побудови дашбордів	Підтримує інтеграцію з багатьма інструментами, але потребує налаштування
Спільнота та підтримка	Розвивається спільнотою Grafana Labs	Велика спільнота та корпоративна підтримка Elastic

Таб. 2.2

Отож зробимо висновок до цього підрозділу. Loki є більш легким і економічно ефективним рішенням для зберігання та аналізу логів, що ідеально підходить для середовищ із великими обсягами даних. Водночас ELK Stack пропонує ширші можливості пошуку та аналізу, однак вимагає значних ресурсів і складніших налаштувань, що може бути критичним фактором для організацій із великими потребами в деталізації даних журналів. Вибір між цими рішеннями залежить від пріоритетів щодо ресурсів, масштабу інфраструктури та потреб у функціональності.

2.4. Порівняння інструментів візуалізації та аналітики

Grafana — це популярний інструмент для візуалізації та аналітики, за допомогою якого можна створювати дашборди для моніторингу різних систем та інфраструктур.

Grafana підтримує широкий спектр джерел даних: вона може інтегруватися з багатьма джерелами, такими як Prometheus, VictoriaMetrics, Elasticsearch, InfluxDB, Graphite, та інші. Це робить його гнучким інструментом, який можна адаптувати до різноманітних середовищ.

Завдяки налашуванню дашбордів користувачі можуть створювати індивідуальні дашборди, які відображають усі необхідні метрики у зручному форматі. Інтерфейс дає можливість швидко налаштовувати таблиці, графіки, гістограми, кругові діаграми та інші типи візуалізацій.

Також Grafana підтримує оповіщення (alerting), які можуть надсилатися через різні канали, такі як електронна пошта, Slack, Telegram тощо. Оповіщення налаштовуються на основі заданих умов і можуть сповіщати про критичні зміни у значеннях метрик.

Grafana підтримує запити PromQL та інші типи запитів для Prometheus, LogQL для Loki та інші мови запитів для специфічних джерел даних.

А завдяки широким можливостям аналітики користувачі можуть аналізувати дані у реальному часі, налаштовуючи фільтри, часові проміжки та порівнювати дані за різні періоди часу. З цим функціоналом можна більш глибоко аналізувати показники продуктивності.

Отож, бачимо, що Grafana є дуже зручним інструментом візуалізації, який допомагає оптимізувати роботу підприємств та інших користувачів. Серед його переваг такі:

- Гнучкість та універсальність. Grafana підтримує багато джерел даних і надає можливості для створення інтерактивних дашбордів, що робить його зручним для різних типів інфраструктур і задач.

- Масштабованість. Grafana може бути розгорнута на великих інфраструктурах і підтримувати великий обсяг даних. Це робить її придатною для використання у великих компаніях.
- Розвинена система оповіщень. Завдяки можливості налаштування сповіщень на основі умов сприяє своєчасному інформуванню команди про можливі проблеми в інфраструктурі.
- Інтеграція з іншими інструментами моніторингу. Легка інтеграція з Prometheus, Loki, VictoriaMetrics дає можливість використовувати Grafana як єдиний центр для візуалізації різних метрик та логів.
- Активна спільнота та підтримка. Grafana має велику спільноту користувачів і активну підтримку від розробників, що забезпечує швидке вирішення проблем і постійне оновлення функціональності.

Але разом з перевагами є й певні обмеження.

Хоча Grafana й надає широкі можливості для візуалізації, аналітичні можливості залежать від джерел даних, які мають обробляти метрики або логи на своєму рівні. Проте, сама по собі Grafana не є аналітичною платформою і не має можливості зберігає дані.

Ще одним обмеженням є складність у налаштуванні для нових користувачів. Хоч інтерфейс Grafana інтуїтивний, але створення складних дашбордів може бути викликом для нових користувачів.

Також під час роботи з великими обсягами даних Grafana може споживати значні ресурси, особливо якщо йдеться про інтерактивні та оновлювані в реальному часі дашборди.

Обмеження на складні обчислення: Grafana не призначена для глибоких математичних розрахунків чи обробки великих масивів даних. У Grafana тільки візуалізувати підсумкові показники, а роботу з даними краще делегуваи іншим інсрументам.

Отже, Grafana є потужним та гнучким інструментом для візуалізації даних та аналітики в реальному часі. Завдяки підтримці багатьох джерел даних та розвиненим можливостям налаштування, вона підходить для багатьох сценаріїв

моніторингу IT-інфраструктури. Однак під час вибору Grafana важливо враховувати її обмеження, особливо щодо аналітичної обробки даних, яка часто залежить від можливостей інтегрованих джерел.

У **стеку ELK** (Elasticsearch, Logstash, Kibana) Kibana є основним інструментом для візуалізації даних та забезпечує широкі можливості для моніторингу та аналітики логів.

Перерахуємо основні характеристики Kibana. Оскільки Kibana побудована для тісної інтеграції з Elasticsearch. Це дає можливість швидко та ефективно створювати візуалізації та дашборди на основі даних з Elasticsearch.

Kibana підтримує різні типи графіків та візуалізацій, такі як гістограми, лінійні графіки, кругові діаграми, картограми та інші, що допомагає створити інформативні дашборди.

Завдяки детальній аналітиці логів користувачі можуть виконувати пошук по логах, фільтрувати дані та створювати візуалізації для аналізу зібраної інформації.

Kibana підтримує вбудовані можливості машинного навчання (Machine Learning (ML)) для виявлення аномалій і прогнозування на основі даних з Elasticsearch. Ця функціональність дозволяє автоматично відстежувати відхилення від нормальних показників.

Інструменти для дашбордів та спільної роботи в Kibana дозволяють створювати дашборди, що легко налаштовуються та надають деталізовану інформацію. Користувачі також можуть ділитися дашбордами з іншими учасниками команди.

Також Kibana підтримує систему оповіщень, яка працює разом з Elasticsearch Watcher для створення умовних сповіщень, що надсилаються через електронну пошту або інші канали.

Хоч Kibana є ефективним інтрументом, але є певні недоліки, які ускладнюють роботу з нею.

По-перше – це залежність від Elasticsearch. Оскільки візуалізація та пошук даних повністю залежать від Elasticsearch, Kibana не може використовуватися

окремо. Це може бути обмеженням для організацій, які використовують інші сховища даних.

Для ефективного функціонування ELK стека, включаючи Kibana, потрібні значні ресурси, що може бути дорогим рішенням для великих середовищ або обробки великих обсягів даних.

Складність у налаштуванні та підтримці. У масштабованих середовищах, де необхідна оптимізація продуктивності Kibana, як і весь ELK стек, може бути складною для налаштування та обслуговування

Kibana оптимізована для аналізу логів, проте, в порівнянні з інструментами, спеціалізованими на моніторингу метрик, такими як Grafana, вона має обмежену функціональність для роботи з метриками.

Ще одним суттєвим недоліком є те, що Kibana не зберігає дані, а виконує лише функцію візуалізації на основі Elasticsearch, тому потребує відповідної бази даних.

Отже, Kibana є ідеальним інструментом для аналізу логів та потужним інструментом для візуалізації даних з Elasticsearch. Однак її ефективність повністю залежить від Elasticsearch, а вимоги до ресурсів можуть обмежувати використання у великих середовищах.

Для зручності, сформуємо таблицю 2.3 – «Порівняння результатів аналізу систем візуалізації даних», в якій опишемо всі характеристики Grafana й Kibana та порівняємо їх.

Характеристика	**Grafana**	**Kibana**
Основна функціональність	Візуалізація метрик, алерти	Візуалізація логів, алерти
Інтеграція з джерелами	Підтримка багатьох джерел даних (Prometheus, VictoriaMetrics, Loki тощо)	Тісна інтеграція з Elasticsearch
Типи візуалізацій	Гнучкі налаштування для метрик, графіки, гістограми, heatmaps,	Графіки, гістограми, кругові діаграми,

Характеристика	Grafana	Kibana
	алерти	картограми
Можливості оповіщення	Розширені, інтеграції з популярними сервісами	Інтеграція з Elasticsearch Watcher
Машинне навчання	Відсутнє	Підтримка вбудованого ML
Продуктивність	Висока продуктивність при роботі з великим обсягом метрик	Висока продуктивність при роботі з логами
Гнучкість налаштуванн	Висока, підтримка запитів PromQL	Висока, гнучкі фільтри для аналізу логів
Залежність від сховища	Підтримує різні джерела	Потребує Elasticsearch
Спільнота та підтримка	Активна спільнота та розробка	Велика спільнота завдяки ELK стеку
Ресурсні вимоги	Оптимізовано для метрик	Високі вимоги для ELK стека

Таб. 2.3

Grafana та Kibana мають різні переваги, залежно від завдання візуалізації та аналізу даних.

- Grafana забезпечує широку гнучкість для роботи з метриками. Це і робить її універсальним рішенням для моніторингу IT-інфраструктури, оскільки вона підтримує різні джерела даних і має потужні можливості для побудови аналітичних дашбордів. Серед її переваг – інтеграція з популярними системами, як-от Prometheus і VictoriaMetrics, що надає їй широку функціональність і адаптивність. Grafana також відзначається меншою залежністю від єдиного сховища даних, а це дає можливість легко змінювати джерела даних за потреби.

- Kibana ж, своєю чергою, є найкращим вибором для аналізу логів завдяки тісній інтеграції з Elasticsearch та підтримці детальної фільтрації та пошуку. З Kibana дає змогу швидко знаходити аномалії й аналізувати тренди, а її інструменти для ML дають змогу автоматично відстежувати аномальні значення. Але ця платформа потребує великих ресурсів для зберігання й аналізу даних, що може стати недоліком у великих середовищах з високими обсягами логів.

Тож, підсумуємо. Для організацій, які потребують гнучкості у візуалізації метрик та планують працювати з різними джерелами, Grafana стане ефективним вибором. Тим часом Kibana є релевантним інструментом для аналізу логів завдяки її можливостям фільтрації та машинного навчання. Вибір між Grafana та Kibana варто здійснювати на основі специфічних потреб у моніторингу: чи орієнтовано рішення на метрики, чи на роботу з логами й аналітику безпеки.

2.5. Обґрунтування вибору Grafana, Loki та VictoriaMetrics

Відповідність вимогам системи

Поєднання Grafana, Loki та VictoriaMetrics забезпечує високу масштабованість у зборі, обробці та візуалізації як метрик, так і логів. VictoriaMetrics оптимізовано для обробки великих обсягів метрик з низькими вимогами до ресурсів, що робить її придатною для масштабованих систем. Loki ефективно зберігає логи за рахунок компресії, що знижує обсяг необхідного сховища та, як наслідок, підвищує продуктивність.

Кастомізований стек, що містить Grafana, Loki та VictoriaMetrics забезпечує гнучкість та адаптивність. Оскільки, Grafana, яка виступає як інструмент візуалізації, має вбудовану підтримку для великої кількості джерел даних, що дає можливість легко додавати нові сервіси до моніторингу. А Loki та VictoriaMetrics мають гнучкі інтерфейси для інтеграції з інфраструктурою, що сприяє їх адаптації під різні середовища та платформи.

Grafana є популярним інструментом з підтримкою багатьох плагінів для інтеграції з іншими системами моніторингу, такими як Prometheus, InfluxDB та Elasticsearch. Це забезпечує безперебійну інтеграцію з уже розгорнутими компонентами, спрощуючи налаштування і використання існуючих даних у нових дашбордах.

Переваги обраних технологій

Щоб пояснити, чому обрані технології є ефективними та релевантними для багатьох користувачів – коротко охарактеризуємо їх переваги.

Синергія між компонентами. Поєднання Grafana, Loki та VictoriaMetrics створює ефективний моніторинговий стек, де кожен компонент доповнює функціональність іншого. Loki відповідає за ефективне зберігання логів, VictoriaMetrics зосереджується на метриках, а Grafana об’єднує ці дані для зручного аналізу, надаючи користувачам цілісну картину інфраструктури.

Кожен із компонентів стека оптимізований для швидкої обробки та зберігання великих обсягів даних. VictoriaMetrics забезпечує високу швидкість збору та обробки метрик, а Loki зменшує використання ресурсів, зберігаючи лише часові мітки для кожного потоку логів. Це робить цей стек високопродуктивним та ефективним рішенням для великих систем.

Оскільки кожен із компонентів стека має активну спільноту та регулярні оновлення, це забезпечує постійне вдосконалення, швидке вирішення проблем та появу нових функцій. А також надає можливості доступу до широкого спектру документації та підтримки. Наприклад, обраний стек підтримує OpenTelemetry, тож, якщо виникне такий запит, до запропонованої системи моніторингу можна додати його. Це допоможе легко інтегрувати дані з різних сервісів та додатків, при цьому стандартизуючи процес збору метрик і логів. Відповідно, це спрощує масштабування інфраструктури та інтеграцію з іншими системами моніторингу.

Ще однією характеристикою стеку є його можливість розширення функціональності. Grafana має безліч додаткових плагінів для інтеграції з різними джерелами даних, а також може створювати кастомні візуалізації. Loki і

VictoriaMetrics теж мають функції розширення, що робить цей стек придатним для специфічних завдань та налаштувань під потреби підприємства.

Висновок

У другому розділі проведено порівняльний аналіз різних систем моніторингу для збору, зберігання та аналізу метрик і логів. Також розглянуто можливості інтеграції та візуалізації даних. Ми визначили, що обраний стек з Grafana, Loki та VictoriaMetrics надає оптимальний баланс між масштабованістю, продуктивністю, гнучкістю та ефективністю.

Проаналізувавши такі системи як Prometheus, VictoriaMetrics, ELK і Loki, ми виокреили сильні сторони та обмеження кожного з інструментів, зокрема щодо функціональності, адаптивності до інфраструктур різних масштабів та рівня підтримки спільноти. Результати проведеного порівняння підтвердили, що стек Grafana, Loki та VictoriaMetrics найкраще відповідає критеріям, оскільки даєможливість забезпечити надійний моніторинг та управління даними в реальному часі, залишаючись при цьому гнучким для інтеграцій.

Обґрунтований вибір Grafana, Loki та VictoriaMetrics відкриває перспективи для подальшого розширення функціональності за рахунок підтримки OpenTelemetry, плагінів та активного розвитку. Це забезпечує можливість адаптувати систему моніторингу до різних вимог бізнесу та зростання інфраструктури, що сприяє підвищенню стабільності та продуктивності IT-середовища.

РОЗДІЛ 3: ПРОЦЕС ПОБУДОВИ СИСТЕМИ МОНІТОРИНГУ НА БАЗІ GRAFANA, LOKI ТА VICTORIAMETRICS

3.1. Проектування архітектури системи моніторингу

Вимоги до системи

Для планування архітектури системи моніторингу з обраними мною компонентами для створення та дослідження цієї системи, варто окреслити вимоги.

Функціональні вимоги

Система повинна забезпечувати збір метрик та логів у реальному часі з серверів, додатків і мережевих пристроїв. Візуалізація даних повинна відображатися у вигляді графіків, дашбордів і звітів для аналізу. Обов’язково має бути реалізовано налаштування алертів з можливістю відправки сповіщень (наприклад, через email, Slack, Telegram). Також має бути забезпечена інтеграція з існуючими інструментами (Prometheus експортери). Запитів та фільтрів для аналізу метрик і логів мають бути гнучкі у налаштуванні.

Нефункціональні вимоги

Система повинна адаптуватися до зростання кількості пристроїв і обсягів даних, що збираються, має підтримуватися як горизонтальне так і вертикальне масштабування. За можливості необхідно мінімізувати простої і забезпечення високої доступності системи. Також системо має обробляти великий обсяг даних із мінімальною затримкою.

Архітектурна схема рішення

Опис компонентів системи:

VictoriaMetrics – відповідає за збір, зберігання та аналіз метрик.

Loki – використовується для збору та аналізу логів.

Grafana – основний інструмент для візуалізації метрик і логів, інтеграція з VictoriaMetrics і Loki.

Prometheus експортери – забезпечують збір метрик з систем, додатків та інфраструктури.

Схема взаємодії між компонентами:

- VictoriaMetrics та Loki збирають дані з експортерів.
- Grafana інтегрується з Loki та VictoriaMetrics для візуалізації даних.
- Дані відображаються на інтерактивних дашбордах, а налаштування алертів здійснюється через Grafana.

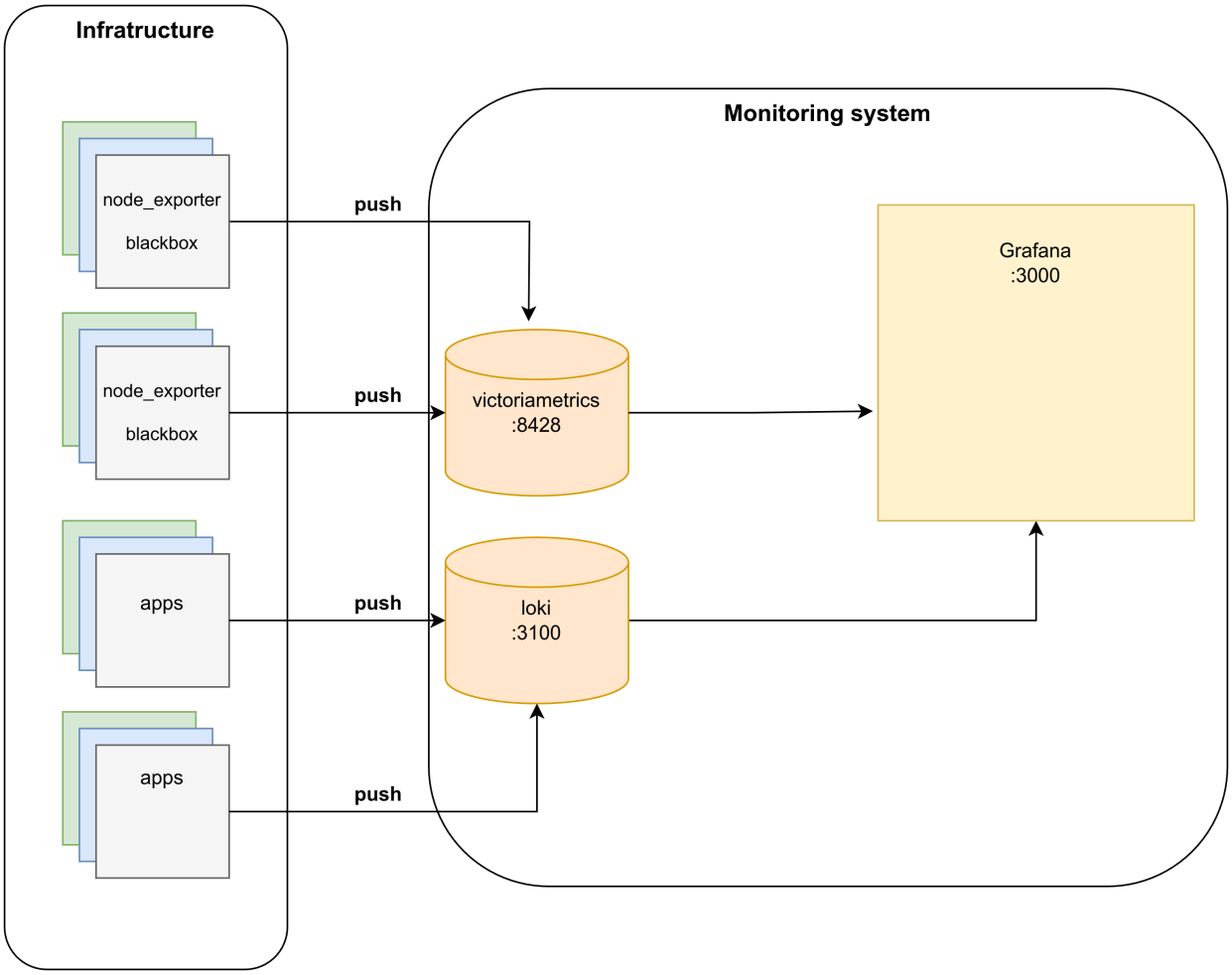


Рис. 3.1

Вибір інфраструктури та середовища розгортання

Тестова система система буде розгорнута на віртуальній машині, ОС Ubuntu 24.04, 2 ядрама CPU, 4 ГБ RAM, 50 ГБ дискового простору. Щодо робочого середовища – вибір між локальними серверами та хмарною платформою залежить від розмірів підприємства. Наприклад, для малого підприємства із невеликою кількістю систем локальний сервер може бути достатнім, натомість для великих

організацій із тисячами метрик доцільно використовувати хмарну платформу через її масштабованість і надійність.

Технології контейнеризації:

У тестовому середовищі Docker та Kubernetes не використовуватимуться для спрощення конфігурації.

Для робочої системи контейнеризація може забезпечити швидке розгортання компонентів, масштабування за потреби та зручність оновлення та управління.

Особливості вибору технічних параметрів серверів:

Технічні характеристики серверів залежать від розміру підприємства та кількості систем для моніторингу:

- Мале підприємство: Сервер з 4 ядрами CPU, 16 ГБ RAM і 500 ГБ дискового простору може бути достатнім.
- Середнє підприємство: 8 ядер CPU, 32 ГБ RAM, SSD на 1 ТБ.
- Велике підприємство: Високопродуктивні сервери або хмарна платформа з автоматичним масштабуванням ресурсів.

Ці параметри визначаються очікуваним навантаженням – кількість метрик за секунду, обсяг логів, частота запитів і глибина історичних даних.

3.2. Встановлення та налаштування компонентів

Підготовка сервера

Для розгортання системи моніторингу було обрано дистрибутив Ubuntu Server 24.04, який характеризується стабільністю, сучасними оновленнями та легкістю в налаштуванні.

Після встановлення ОС, проводено базові налаштування середовища для розгортання системи моніторингу:

1. Створено користувача для процесів системи моніторингу, оскільки призначення спеціальних користувачів з обмеженим доступом підвищує безпеку системи:

sudo useradd --no-create-home --shell /usr/sbin/bass victoriametrics

2. Створено окремі директорії під кожен компонент, тому що це забезпечує ізоляцію даних та зручність обслуговування

- VictoriaMetrics - /mnt/operational/victoriametrics
- Loki - /mnt/operational/loki
- Grafana - /mnt/operational/grafana
- Blackbox - /mnt/operational/blackbox
- Node exporter - /mnt/operational/node-exporter

Розгортання VictoriaMetrics

Для розгортання VictoriaMetrics було підготовлено середовище у директорії /mnt/operational/victoriametrics. Всі компоненти системи, включно з бінарними файлами, дані та логи були розміщені у відповідних підпапках: /bin, /data та /log.

1. Завантаження та встановлення:

Бінарний файл VictoriaMetrics було завантажено з офіційного GitHub репозиторію, розпаковано та переміщено до /mnt/operational/victoriametrics/bin.

2. Налаштування конфігураційного файлу:

Конфігураційний файл prometheus.yml був створений для налаштування збору метрик. У ньому було визначено інтервали збору даних та джерела метрик, зокрема, для Node Exporter. На рис. 3.2 зображено вміст конфігураційного файлу.

```
global:
  scrape_interval: 15s
  scrape_timeout: 10s

scrape_configs:
  - job_name: 'node_exporter'
    static_configs:
      - targets:
        - 'localhost:9100'
```

Рис. 3.2

3. Створення сервісного файлу:

Для забезпечення автоматичного запуску VictoriaMetrics було створено файл системного сервісу у /etc/systemd/system/victoriametrics.service. У цьому файлі було налаштовано:

- Шлях до бінарного файлу.
- Шляхи до конфігурацій, логів та даних.
- Параметри автозапуску та перезапуску у разі помилки.

На рис. 3.3 зображено вміст сервісного файлу.

```
[Unit]
Description=VictoriaMetrics
After=network.target

[Service]
User=victoriametrics
Group=victoriametrics
ExecStart=/mnt/operational/victoriametrics/victoria-metrics-prod \
-promscrape.config=/mnt/operational/victoriametrics/prometheus.yml\
-storageDataPath=/mnt/operational/victoriametrics/data\
-retentionPeriod=7
Restart=always

[Install]
WantedBy=multi-user.target
```

Рис. 3.3

4. Після додавання сервісного файлу було виконано запуск VictoriaMetrics через systemd та налаштовано його автозапуск під час старту системи та проведено перевірту статусу сервісу. На рис. 3.4 показана результат виконання команди, що перевіряє статус сервісу.

```
ubuntu@monitoiring /m/o/victoriametrics> sudo systemctl status victoriametrics.service
● victoriametrics.service - VictoriaMetrics
     Loaded: loaded (/etc/systemd/system/victoriametrics.service; enabled; preset: enabled)
     Active: active (running) since Mon 2024-12-02 09:26:59 UTC; 4s ago
   Main PID: 1639 (victoria-metric)
      Tasks: 8 (limit: 4614)
     Memory: 41.6M (peak: 42.0M)
        CPU: 142ms
     CGroup: /system.slice/victoriametrics.service
             └─1639 /mnt/operational/victoriametrics/victoria-metrics-prod -storageDataPath=/mnt/operational/
```

Рис. 3.4

Встановлення Node exporter для збору метрик

Для розгортання Node exporter було підготовлено середовище у директорії /mnt/operational/node-exporter. Усі компоненти системи, включно з бінарними файлами, даними та логами, були розміщені у відповідних підпапках: /bin, /data та /log.

1. Завантаження бінарного файлу:

Архів із бінарним файлом Node Exporter завантажено з офіційного GitHub репозиторію, розпаковано та переміщено до /mnt/operational/node-exporter/bin.

2. Налаштування systemd сервісу:

Створено сервісний файл для Node Exporter, який було додано до /etc/systemd/system. У сервісному файлі вказано шлях до бінарного файлу, а також параметри запуску(Рис. 3.5).

```
[Unit]
Description=Node Exporter
Requires=node_exporter.service

[Service]
User=victoriametrics
Group=victoriametrics
ExecStart=/mnt/operational/node-exporter/node_exporter --collector.textfile.directory=/mnt/operational/node-exporter/textfile_collector
StandardOutput=append:/mnt/operational/node-exporter/logs/stdout.txt
StandardError=append:/mnt/operational/node-exporter/logs/stderr.txt
Restart=on-failure

[Install]
WantedBy=multi-user.target
```

Рис. 3.5

3. Після додавання сервісного файлу було виконано запуск Node Exporter через systemd та налаштовано його автозапуск під час старту системи та проведено перевірку статусу сервісу(Рис. 3.6).

```
ubuntu@monitoiring /m/o/node-exporter> sudo systemctl status node_exporter.service
● node_exporter.service - Node Exporter
     Loaded: loaded (/etc/systemd/system/node_exporter.service; enabled; preset: enabled)
     Active: active (running) since Mon 2024-12-02 09:22:36 UTC; 11min ago
   Main PID: 762 (node_exporter)
      Tasks: 5 (limit: 4614)
     Memory: 21.0M (peak: 21.8M)
        CPU: 619ms
     CGroup: /system.slice/node_exporter.service
             └─762 /mnt/operational/node-exporter/node_exporter --collector.textfile.directory=/mnt/operational/node-exporter/textfile_collector

Dec 02 09:22:36 monitoiring systemd[1]: Started node_exporter.service - Node Exporter.
```

Рис. 3.6

Після успішного запуску Node Exporter почав збирати базові метрики сервера, які доступні для подальшого використання в системі моніторингу через VictoriaMetrics.

Розгортання Loki

Для розгортання Loki було підготовлено середовище у директорії /mnt/operational/loki. Усі компоненти системи, що містять бінарні файли, дані та логи, були розміщені у відповідних підпапках: /bin, /data та /log.

1. Завантаження бінарного файлу:

Архів із бінарним файлом Loki завантажено з офіційного GitHub репозиторію, розпаковано та переміщено до /mnt/operational/loki/bin.

2. Створення конфігураційного файлу:

Підготовлено файл конфігурації loki-local-config.yml (Рис.3.7), у якому задано основні параметри:

- Мережеві налаштування: HTTP-інтерфейс (192.168.50.214:3100), gRPC (9096).
- Шляхи до зберігання даних:

- Логи: /mnt/operational/loki/data/chunks
- Компресор: /mnt/operational/loki/data/compactor
- Параметри схеми: сховище tsdb, об'єкти filesystem, індекси з префіксом index_, період зберігання даних — 30 днів.

```
auth_enabled: false

server:
  http_listen_address: 192.168.50.214
  http_listen_port: 3100
  grpc_listen_port: 9096

common:
  ring:
    instance_addr: 127.0.0.1
    kvstore:
      store: inmemory
  replication_factor: 1
  path_prefix: /mnt/operational/loki/data

schema_config:
  configs:
  - from: 2024-11-22
    store: tsdb
    object_store: filesystem
    schema: v13
    index:
      prefix: index_
      period: 24h

storage_config:
  filesystem:
    directory: /mnt/operational/loki/data/chunks

compactor:
  working_directory: /mnt/operational/loki/data/compactor
  delete_request_store: filesystem
  retention_enabled: true

limits_config:
  max_query_lookback: 30d
  retention_period: 30d
```

Рис. 3.7

3. Створено сервісний файл для Loki та розміщено у /etc/systemd/system.

Сервісний файл містить вказівки щодо запуску процесу (Рис. 3.8), використання конфігураційного файлу та логування.

```
[Unit]
Description=Loki log aggregation system
After=network.target

[Service]
User=victoriametrics
Group=victoriametrics
ExecStart=/mnt/operational/loki/loki-linux-amd64 \
-config.file=/mnt/operational/loki/loki-local-config.yml \
-log.level=debug \
-legacy-read-mode=true
StandardOutput=append:/mnt/operational/loki/logs/stdout.txt
StandardError=append:/mnt/operational/loki/logs/stderr.txt
Restart=always

[Install]
WantedBy=multi-user.target
```

Рис. 3.8

4. Сервіс Loki було запущено через systemd, налаштовано автозапуск під час старту системи. На Рис. 3.9 показано статус сервісу.

```
ubuntu@monitoiring /m/o/loki> sudo systemctl status loki.service
● loki.service - Loki log aggregation system
     Loaded: loaded (/etc/systemd/system/loki.service; enabled; preset: enabled)
     Active: active (running) since Mon 2024-12-02 09:22:40 UTC; 19min ago
   Main PID: 993 (loki-linux-amd6)
      Tasks: 9 (limit: 4614)
     Memory: 66.7M (peak: 147.7M)
        CPU: 10.299s
     CGroup: /system.slice/loki.service
             └─993 /mnt/operational/loki/loki-linux-amd64 -config.file=/mnt/operational/loki/loki-local-config.yml -log.level=debug -legacy-read-mode=true
```

Рис. 3.9

Після успішного налаштування Loki почав приймати логи від додатків та сервісів.

Встановлення Promtail для збору логів

Для розгортання Promtail було підготовлено середовище у директорії /mnt/operational/promtail . Всі компоненти системи, що містять бінарні файли, дані та логи, були розміщені у відповідних підпапках: /bin, /data та /log.

1. Завантаження бінарного файлу:

Архів із бінарним файлом Promtail завантажено з офіційного GitHub репозиторію, розпаковано та переміщено до /mnt/operational/promtail/bin.

2. Створення конфігураційного файлу:

Підготовлено файл promtail_config.yaml (Рис. 3.10). У конфігурації зазначено:

- Файли для збору логів (вказано шляхи до логів системи);
- Адреса сервера Loki (http://192.168.50.214:3100) для надсилання зібраних логів.

```
server:
  http_listen_port: 9080
  grpc_listen_port: 0

positions:
  filename: /mnt/operational/promtail/data/positions.yaml

clients:
  - url: 'http://192.168.50.214:3100/loki/api/v1/push'

scrape_configs:
  - job_name: services
    static_configs:
      - targets:
          - localhost
        labels:
          service: loki
          visible_name: local
          __path__: /mnt/operational/loki/logs/*.txt
```

Рис. 3.10

3. Створено сервісний файл для Promtail (Рис. 3.11) та розміщено у /etc/systemd/system. Сервісний файл містить вказівки щодо запуску процесу.

```
[Unit]
Description=Promtail service
After=network.target

[Service]
User=victoriametrics
Group=victoriametrics
ExecStart=/mnt/operational/promtail/promtail-linux-amd64 -config.file /mnt/operational/promtail/promtail_config.yaml
StandardOutput=append:/mnt/operational/promtail/logs/stdout.txt
StandardError=append:/mnt/operational/promtail/logs/stderr.txt

[Install]
WantedBy=multi-user.target
```

Рис. 3.11

4. Сервіс Promtail було запущено через systemd, налаштовано автозапуск під час старту системи. На рис. 3.12 паказано статус сервісу.

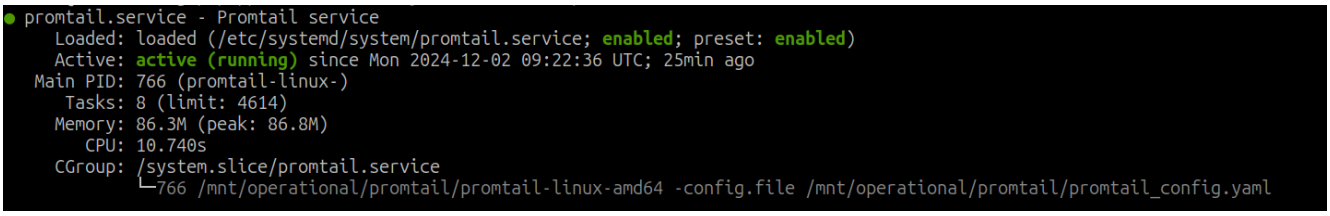


Рис. 3.12

У результаті Promtail розпочав збір логів із зазначених джерел і передає їх до Loki.

Розгортання Grafana

1. Для зберігання .deb файлів створено підпапку /bin В директорії /mnt/operational/promtail.

2. Переходимо до папки /bin, викачуємо .deb файл з офіційного порталу Grafana та встановлюємо:

sudo dpkg -i grafana-enterprise_11.3.1_amd64.deb

3. За необхідності вносимо зміни в конфігураційний файл grafana, який знаходиться /etc/grafana/grafana.ini. Основні налаштування, які можуть бути змінені:

- Порт для підключення (за замовчуванням 3000);
- Адреса доступу Grafana (http://<ip>:<port>);
- Налаштування авторизації (логін та пароль адміністратора);
- Налаштування з’єднання з джерелами даних.

4. Запускаємо сервіс та перевіряємо статус (Рис. 3.13):

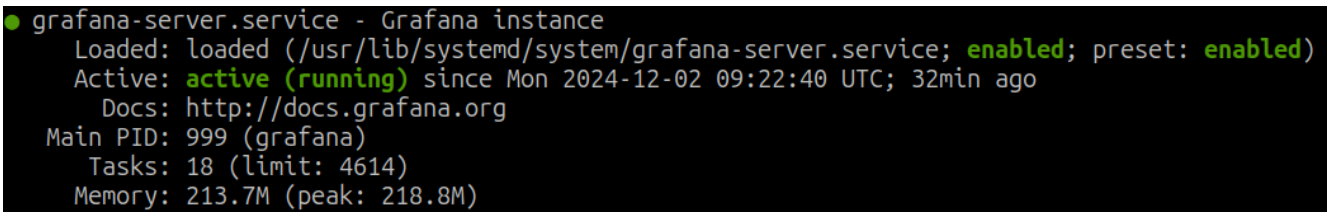


Рис. 3.13

Grafana успішно запущена і доступна для налаштування через веб-інтерфейс за адресою http://<ip>:3000. Подальше налаштування включає підключення джерел даних (VictoriaMetrics, Loki) та створення дашбордів.

Налаштування Grafana

Після встановлення всіх необхідних компонентві, переходимо до налаштування візуалізації та алертів

1. Підключення Loki та VictoriaMetrics до Grafana здійснюється через розділ Data Sources у Grafana (Рис. 3.14). Для цього необхідно вказати адресу сервера та порт, на якому розгорнуті відповідні компоненти. Зокрема:

- Для VictoriaMetrics використовується http://localhost:8428
- Для Loki — http://localhost:3100

З'єднання між компонентами здійснюється через протокол HTTP, забезпечуючи обмін даними для моніторингу та аналізу.

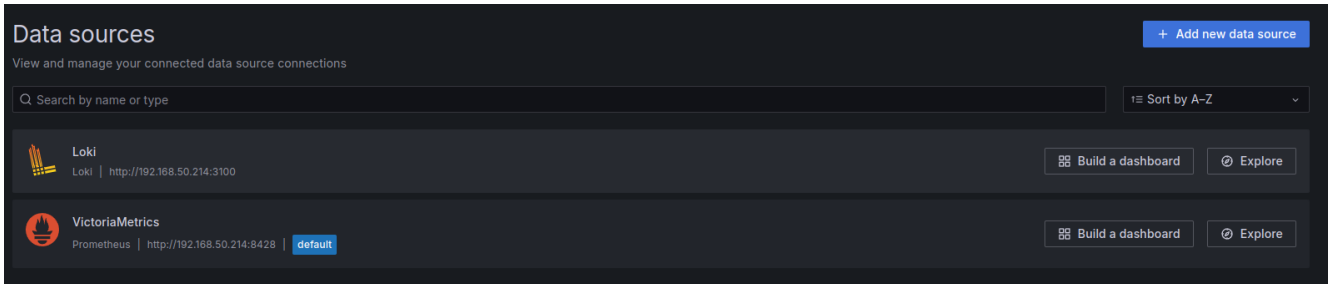


Рис. 3.14

2. Grafana пропонує готові шаблони панелей для різноманітних експортерів, які можна імпортувати у форматі JSON. Для тестової системи було використано шаблон для Node Exporter (Рис. 3.15) та Loki, що значно спрощує налаштування і візуалізацію метрик.

Рис. 3.15

3. Grafana пропонує функціонал для налаштування алертів, що дозволяє оперативно реагувати на критичні події у системі. Для налаштування алертів на тестовій системі виконано наступні дії:

1) Перехід до розділу Alerting:

- В інтерфейсі Grafana перейдіть до розділу Alerting та відкрийте вкладку Contact points. Тут налаштовуються контакти для сповіщень.

2) Налаштування Contact point:

- У якості каналу зв’язку було вибрано Email:
- Введено адресу поштового сервера SMTP.
- Налаштовано облікові дані для автентифікації на поштовому сервері.
- Зазначено отримувача (email, на який надходитимуть сповіщення).
- За необхідності можна додати інші контактні точки, такі як Telegram, Slack, або сторонні сервіси через вебхуки.

3) Створення правила для алертів:

- Перейдіть до вкладки Alert rules.
- Створіть нове правило, яке моніторить зв’язок між серверами:
- Зазначено джерело даних (наприклад, VictoriaMetrics чи Node Exporter).
- Налаштовано умови тригера (наприклад, server_up == 0, де server_up відображає стан сервера).
- Вказано період перевірки стану (наприклад, кожні 30 секунд).
- Додано логіку сповіщення:
- Вибрано раніше налаштований Contact point (наприклад, Email).

4) Тестування алерта:

- Для перевірки коректності налаштувань вручну ініційовано алерт (імітація втрати зв’язку сервера).
- На пошту надійшло повідомлення, яке містить деталі інциденту(рис):
- Назва алерта.
- Час спрацювання.
- Опис проблеми та метрики, які викликали спрацювання.

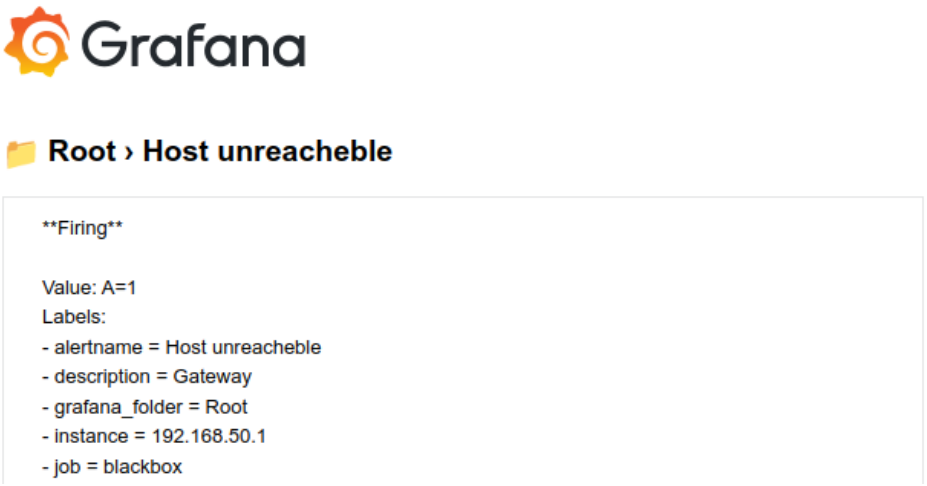


Рис. 3. 16

Завдяки налаштуванню алертів система моніторингу автоматично повідомляє адміністратора про потенційні або поточні проблеми, що значно зменшує час реакції на інциденти та забезпечує стабільність роботи інфраструктури.

3.3. Тестування та оптимізація системи

Для перевірки здатності створеної системи моніторингу витримувати інтенсивне використання, проводиться тестування з метою збору таких показників як максимальна кількість оброблених метрик за секунду, швидкість індексації логів, час відгуку Grafana при великій кількості запитів. Ці показники можна проалазівувати за допомогою візуалізації Grafana. Під кожен компонент системи моніторингу є можливість створити дашборд, який буде візуалізувати ключові показники.

На тестовій системі було створено дашборд для моніторингу показників Loki. Для імітації навантаження на кожен компонент системи моніторингу можемо використовувати спеціальні інструменти. Для генерації запитів до VictoriaMetrics використовується Promtool, а також спеціальні скрипти для масового надсилання метрик через Node Exporter. Для Loki – Promtail може бути налаштований для

симуляції інтенсивного потоку логів. Щодо Grafana – k6 або Apache JMeter можуть використовуватись для навантаження запитами до дашбордів.

Після проведення навантажувальних тестувань проводиться аналіз продуктивності компонентів, а саме: визначення вузьких місць(аналіз часу обробки метрик, використання пам’яті та CPU, перевірка швидкості індексації логів і доступності запитів до архівованих даних, оцінка часу завантаження дашбордів і роботи алертів). Також перевіряється використання ресурсів CPU, пам’ять, швидкості читання/запису дисків.

Перевірка проводиться у разі потреби, наприклад, якщо спостерігаються затримки в роботі сервісів, або у випадках, коли планується масштабування системи для підтримки більшої кількості сервісів, що вимагає гарантій продуктивності та стабільності роботи.

У разі отримання негативних результатів тестувань в першу чергу розгядаємо можливість оптимізації компонентів. У VictoriaMetrics, наприклад, є можливість налаштування буферів для оптимізації збору метрик. Також доречно підкорегувати показники retentionPeriod (період зберігання метрик) і maxConcurrentInserts (максимальна кількість одночасних вставок). У Loki оптимізація chunk_size (розмір сегмента логів, що зберігається) і index_period (період створення індексів для пошуку) для зменшення навантаження на дискову систему. У той же час у Grafana – це коригування кешу для швидшого завантаження дашбордів, обмеження кількості одночасних запитів. Усі ці опції безпосередньо впливають на взаємодію з базами даних, оптимізуючи їх використання, що в результаті підвищує загальну швидкість і продуктивність системи.

У разі неможливості оптимізації компонентів, доречно роглянути горизонтальне або вертикальне масштабування.

3.4 Забезпечення відмовостійкості та резервування

Кожен компонент створеної системи моніторингу має інструменти та функціонал для забезпечення відмовостійкості та резервування даних.

Victoriametrics підтримує налаштування кластерного режиму з реплікацією даних, також містить інструмент описаний раніше, а саме vmbackup, який в свою чергу дозволяє зберігати резервні копії на локальний дисках або у зовнішніх сховищах, наприклад S3.

Loki має функцію реплікації логів для збереження їх у кількох копіях, що дозволяє забезпечити їх доступність у випадку збою одного з вузлів. Також Loki підтримує зберігання логів у S3-подібних сховищах, що в свою чергу зменшує навантаження на локальну дискову систему та дозволяє масштабувати обсяг збереження логів.

Для забезпечення відмовостійкості Grafana можна налаштувати резервні які автоматично активуються у разі виходу з ладу основного інстансу. Щодо резервування даних, Grafana має інструмент під назвою grafana-backup, який дозволяє створювати резервні копії дашбордів, джерел даних та загальних налаштувань та зберігати їх у локальних або хмарних сховищах.

3.5. Результати впровадження системи моніторингу

Система моніторингу, побудована на базі VictoriaMetrics, Loki та Grafana, повністю відповідає встановленим функціональним і нефункціональним вимогам. Основні цілі, зокрема, збір метрик і логів у реальному часі, надання доступу до інтерактивних дашбордів для візуалізації даних, можливість швидкого виявлення проблем та аналізу системних інцидентів**,** було успішно реалізовано.

Завдяки інтерактивним дашбордам у Grafana та централізованому збору метрик і логів, а також налаштованим алертам виявляти інциденти стало значно простіше. Це дозволить оперативно реагувати на критичні ситуації та мінімізувати їхній вплив на бізнес.

Постійний моніторинг ресурсів серверів і додатків дозволяє уникати перевантажень, своєчасно оптимізувати роботу компонентів, а аналітика зібраних даних сприяє точнішому прогнозуванню навантаження і потреб у ресурсах.

3.6. Обговорення та перспективи розвитку

У разі необхідності розширення функціоналу системи моніторингу, обраний стек підтримує інтеграцію з численними інструментами, що відкриває нові можливості для масштабування та вдосконалення. Наприклад:

- **Tempo**. Додавання системи трасування запитів дозволить краще розуміти взаємодію між компонентами системи, що особливо важливо для мікросервісної архітектури. Це значно спрощує пошук причин збоїв і аналіз продуктивності.
- **Zabbix**. Інтеграція із Zabbix відкриває можливість моніторити специфічні мережеві пристрої та вузли, які не підтримують Prometheus-експортери. Це робить систему більш гнучкою та універсальною.
- **OpenTelemetry**. Використання OpenTelemetry як стандарту для збору метрик і трасування сприяє спрощенню інтеграції з іншими моніторинговими рішеннями. Це дає змогу підвищити сумісність і забезпечити легкий перехід до нових інструментів у майбутньому.
- **Хмарні сервіси**. Підключення до хмарних провайдерів, таких як AWS CloudWatch або Google Cloud Monitoring, забезпечує моніторинг у багатохмарних середовищах. Це особливо корисно для компаній, що використовують гібридну або багатохмарну інфраструктуру.
- **CI/CD процеси**. Інтеграція з конвеєрами розгортання дозволяє оперативно виявляти та усувати проблеми на етапі розробки та тестування. Це значно знижує ризик помилок у продуктивному середовищі та покращує загальну якість розгортання.

Подальше вдосконалення системи моніторингу дасть змогу збільшити її функціональність і адаптивність, підтримуючи сучасні стандарти моніторингу IT-інфраструктури. Це, своєю чергою, забезпечить кращу масштабованість, відмовостійкість і можливість інтеграції з новими технологіями.

Висновок

У третьому розділі було детально описано процес створення та налаштування системи моніторингу IT-інфраструктури, починаючи від проєктування архітектури та вибору компонентів до тестування й оптимізації. Було спроєктовано та реалізовано архітектурну схему, яка містить такі ключові компоненти системи — VictoriaMetrics, Loki та Grafana, що взаємодіють для збору, аналізу та візуалізації даних. Виконано налаштування сервера, встановлено та інтегровано основні модулі системи, забезпечено збір метрик і логів у реальному часі. Описано методику проведення навантажувальних тестів, які можуть підтвердити здатність системи працювати стабільно за значних навантажень та надано аналіз потенційних вузьких місць та рекомендації щодо їх усунення. Висвітлено рекомендації щодо оптимізації системи. Також представлено механізми забезпечення відмовостійкості та резервування даних, що, в свою чергу, підвищують стабільність системи у разі збоїв.

Запропонована система моніторингу є ефективним рішенням для середніх і великих підприємств завдяки її гнучкості, масштабованості та інтеграційним можливостям. Реалізований підхід дає можливість забезпечити не лише якісний моніторинг, але й подальше розширення функціональності відповідно до зростаючих потреб організації.

Результати, отримані у цьому розділі, створюють основу для практичного впровадження системи у реальних умовах, з можливістю її адаптації до специфічних бізнес-вимог.

ВИСНОВКИ

У моїй кваліфікаційній роботі було проведено комплексний аналіз існуючих технологій і рішень для моніторингу IT-інфраструктури, розглянуто їх архітектурні особливості, функціональні можливості та методи інтеграції. На основі отриманих результатів було обґрунтовано вибір оптимального стеку технологій для масштабованої системи моніторингу, який поєднує VictoriaMetrics для збору та зберігання метрик, Loki для обробки логів та Grafana для візуалізації і аналітики.

У ході роботи вдалося:

по-перше, систематизувати інформацію про переваги та недоліки класичних і хмарних рішень моніторингу, а також порівняти функціональність різних інструментів щодо їх масштабованості, продуктивності та простоти інтеграції;

по-друге, на основі проведеного порівняльного аналізу сформувати вимоги до майбутньої системи моніторингу, яка має відповідати потребам середніх та великих IT-інфраструктур, забезпечувати ефективний збір, зберігання та візуалізацію даних у реальному часі, а також підтримувати механізми оперативного реагування на інциденти;

по-третє, спроєктувати та реалізувати модель масштабованої системи моніторингу, що об’єднує VictoriaMetrics, Loki та Grafana, інтегровано налаштувати алерти, дашборди, механізми резервування й оптимізації для досягнення надійності, гнучкості та розширюваності системи.

Загалом, поставлена мета була досягнута. Результати роботи можуть бути використані як основа для впровадження подібних систем моніторингу у реальних умовах експлуатації – як у мережах спеціального призначення, так і у цивільних IT-середовищах. Це сприятиме підвищенню продуктивності, надійності та стабільності інформаційних систем, а також швидшому реагуванню на потенційні інциденти та оптимізації витрат на утримання інфраструктури.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ahmed D. Kora and Moussa Moindze Soidridine, “Nagios based enhanced IT management system,” Interntional Journal of Engineering Science and Technology, vol. 4, no. 3, pp. 818–822, 2012.
2. Barroso L. A., Holzle U. The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines. – Morgan & Claypool, 2019. – 180 с.
3. Datadog Docs. [Електронний ресурс]. – URL: https://docs.datadoghq.com/
4. Davis J., Devers C., Sable T. (2021). Modern System Administration: Building and Maintaining Reliable Systems. 300 p. Available at: https://bigl.ua/ua/p2026141934-modern-system-administration (accessed 23 January 2024).
5. Elastic Documentation. [Електронний ресурс]. – URL: https://www.elastic.co/docs
6. Favour Daniel, Ankit Anand, (2024) Kibana vs. Grafana - A Scenario-Based Decision Guide. [Електронний ресурс]. – URL: https://signoz.io/blog/kibana-vs-grafana/
7. Grafana docs. [Електронний ресурс]. – URL: https://grafana.com/docs/grafana/latest/
8. Grafana Loki documentation. [Електронний ресурс]. – URL: https://grafana.com/docs/loki/latest/
9. Grafana – [Електронний ресурс]. Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Grafana
10. Infrastructure Monitoring Overview. [Електронний ресурс]. – URL: https://www.datadoghq.com/knowledge-center/infrastructure-monitoring/
11. Kreps J., (2015) I Heart Logs Event Data, Stream Processing, and Data Integration. [Електронний ресурс]. – URL: https://github.com/sprabhuonline/Kafka-books/blob/master/I%20Heart%20Logs%20Event%20Data%2C%20Stream%20Processing%2C%20and%20Data%20Integration.pdf

12. Latkin Igor, (2023) Grafana Loki Configuration Nuances. [Електронний ресурс]. – URL: https://medium.com/lonto-digital-services-integrator/grafana-loki-configuration-nuances-2e9b94da4ac1
13. Md Aftab, (21.05.2024) Understanding Grafana's LGTM Stack: Components, Features, Benefits, and Applications. [Електронний ресурс]. – URL: https://www.linkedin.com/pulse/understanding-grafanas-lgtm-stack-components-features-md-aftab-lsxpc?utm_source=share&utm_medium=member_ios&utm_campaign=share_via
14. Nagios Core Documentation. [Електронний ресурс]. – URL: https://library.nagios.com/products/nagios-core/documentation/
15. New Relic docs. [Електронний ресурс]. – URL: https://docs.newrelic.com/
16. OpenTelemetry Documentation. [Електронний ресурс]. – URL: https://opentelemetry.io/docs/what-is-opentelemetry/
17. Oracle VirtualBox: User Guide for Release 7.1 (2024). [Електронний ресурс]. – URL: https://www.virtualbox.org/manual/
18. Prometheus docs. [Електронний ресурс]. – URL: https://prometheus.io/docs/introduction/overview/
19. Scott G. Chaplowe & J. Bradley Cousins. (2016). Monitoring and Evaluation Training: A Systematic Approach. Thousand Oaks, CA: Sage. 464
20. Turnbull J., (2016) The Art of Monitoring. [Електронний ресурс]. – URL: https://www.slideshare.net/slideshow/the-art-of-monitoring-2016pdf/252142218
21. Urooj Abbasi, (2023) Best practices for data visualization in Monitoring and Evaluation. [Електронний ресурс]. – URL: https://www.activityinfo.org/blog/posts/2023-06-29-best-practices-for-data-visualization-in-monitoring-and-evaluation.html
22. VictoriaMetrics Docs. [Електронний ресурс]. – URL: https://docs.victoriametrics.com/
23. Warley's CatOps Warley's CatOps, (2024) ELK Stack: The Essentials of Elasticsearch, Logstash, and Kibana. [Електронний ресурс]. – URL:

https://medium.com/@williamwarley/elk-stack-the-essentials-of-elasticsearch-logstash-and-kibana-3a09ff647352

24. WESEEK, Quickly Trace and Understand The Key Concepts of VictoriaMetrics. [Електронний ресурс]. – URL: https://weseek-inc.medium.com/quickly-trace-and-understand-the-key-concepts-of-victoriametrics-9294327fd0bc
25. Zabbix documentation. by Zabbix SIA. All rights reserved 2001-2024. [Електронний ресурс]. – URL: https://www.zabbix.com/documentation/current/en/
26. Данилюк І. В. Технологія проведення комплексного іт-моніторингу компанії / Данилюк Ірина Вадимівна, Будник Людмила Андріївна // Галицький економічний вісник. — Т. : ТНТУ, 2024. — Том 87. — № 2. — С. 40–49. — (Економіка).
27. Метрики продуктивності для системи моніторингу [Електронний ресурс]. – URL: https://www.site24x7.com/help/network-metrics/network-monitor.html
28. Найкращі застосунки для аналізу трафіку та пропускної здатності [Електронний ресурс]. – https://www.pcwdld.com/router-monitoringsoftware#wbounce-modal
29. «Проблема обробки великої кількості даних». [Електронний ресурс]. – URL: https://www.jetinfo.ru/bolshie-dannye-bolshayaproblema/
30. ТОП 20 систем моніторингу IT інфраструктури: веб-сайт. [Електронний ресурс]. – URL: https://blog.iteducenter.ua/ratings/monitoring_tools/ (дата звернення: 20.10.2024).
31. Що таке моніторинг мережі? [Електронний ресурс]. – URL: https://avinetworks.com/glossary/network-monitoring/

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА НА ТЕМУ:

ВПРОВАДЖЕННЯ МАСШТАБОВАНОЇ СИСТЕМИ МОНІТОРИНГУ ЯК ЗАСІБ ПІДВИЩЕННЯ ПРОДУКТИВНОСТІ ІНФОРМАЦІЙНИХ СИСТЕМ

Виконав: здобувач вищої освіти гр. ІСДМ-63
Гречуха Олександр Павлочив

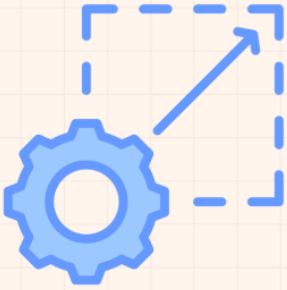
Вступ

МЕТА РОБОТИ – АНАЛІЗ ТА РОЗРОБКА СИСТЕМИ МОНІТОРИНГУ IT-ІНФРАСТРУКТУРИ НА ОСНОВІ СУЧАСНИХ ТЕХНОЛОГІЙ ЗБОРУ ТА ВІЗУАЛІЗАЦІЇ ДАНИХ.

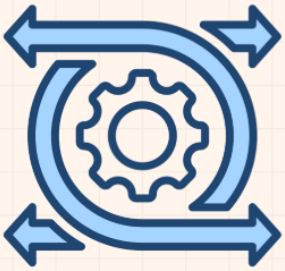
ОБ'ЄКТ ДОСЛІДЖЕННЯ – СИСТЕМИ МОНІТОРИНГУ IT-ІНФРАСТРУКТУРИ ТА ЇХ КОМПОНЕНТИ.

ПРЕДМЕТ ДОСЛІДЖЕННЯ – МЕТОДИ ТА ІНСТРУМЕНТИ ЗБОРУ МЕТРИК, ЛОГІВ ТА ЇХ ВІЗУАЛІЗАЦІЇ.

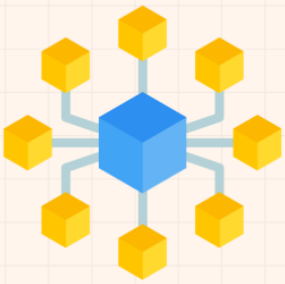
ОСНОВНІ ВИМОГИ ДО СУЧАСНИХ СИСТЕМ МОНІТОРИНГУ



МАСШТАБОВАНІСТЬ ТА ПРОДУКТИВНІСТЬ



ГНУЧКІСТЬ ТА АДАПТИВНІСТЬ



ІНТЕГРАЦІЯ З РІЗНИМИ ТЕХНОЛОГІЯМИ ТА ПРОТОКОЛАМИ

АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

Zabbix

Nagios

DATADOG

New Relic

ELK stack

Grafana

Prometheus

VictoriaMetrics

ПОРІВНЯННЯ КОМПОНЕНТІВ

Характеристика	Prometheus	VictoriaMetrics	Mimir
Модель зберігання	Локальна база даних	Оптимізоване стиснення даних	Об'єктне сховище
Довгострокове зберігання	–	+	+
Масштабованість	–	+	+
Кластеризація	–	+	+
Підтримка PromQL	+	+ (розширення MetricsQL)	+
Інтеграція з Grafana	+	+	+
Продуктивність запису	Середня	Висока	Висока
Стиснення даних	–	+	+
Простота налаштування	+	+	+
Open-source	+	+	+

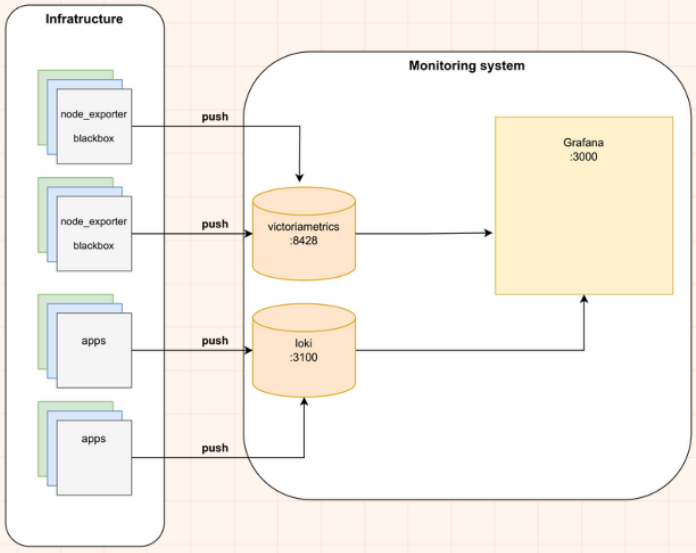
ПОРІВНЯННЯ КОМПОНЕНТІВ

Характеристика	Loki	ELK Stack
Модель зберігання	Індексування лише метаданих	Повнотекстове індексування
Масштабованість	+ (ефективна для великих обсягів)	+ (розподілене зберігання даних)
Ресурсоємність	+ (низькі вимоги до ресурсів)	– (високі вимоги до ресурсів)
Потужність пошуку	– (обмежений пошук за метаданими)	+ (детальний пошук завдяки повнотекстовому індексуванню)
Простота налаштування	+	– (складніша інсталяція та налаштування)
Візуалізація	+ (через Grafana)	+ (через Kibana)
Система індексування	+ (метадані, легка)	– (повнотекстова, важка)
Вартість інфраструктури	+ (низька)	– (висока через ресурсозатратність)
Open-source	+	+

ПОРІВНЯННЯ КОМПОНЕНТІВ

Характеристика	Grafana	Kibana
Інтерфейс користувача	+ (інтуїтивний і гнучкий)	+ (візуально насичений, але складніший)
Можливості візуалізації	+ (підтримка багатьох джерел даних)	+ (зосереджена на Elasticsearch)
Плагіни та розширення	+ (широка підтримка плагінів)	+ (обмежена підтримка)
Масштабованість	+	+
Інтеграція з іншими інструментами	+	+
Пошук даних	+ (фокус на візуалізації, пошук менш потужний)	+ (розширений пошук із Elasticsearch)
Аналітика та функціональність	+ (широкий набір опцій)	+ (аналітика спеціалізована для логів)
Open-source	+	+

АРХІТЕКТУРА СИСТЕМИ МОНІТОРИНГУ

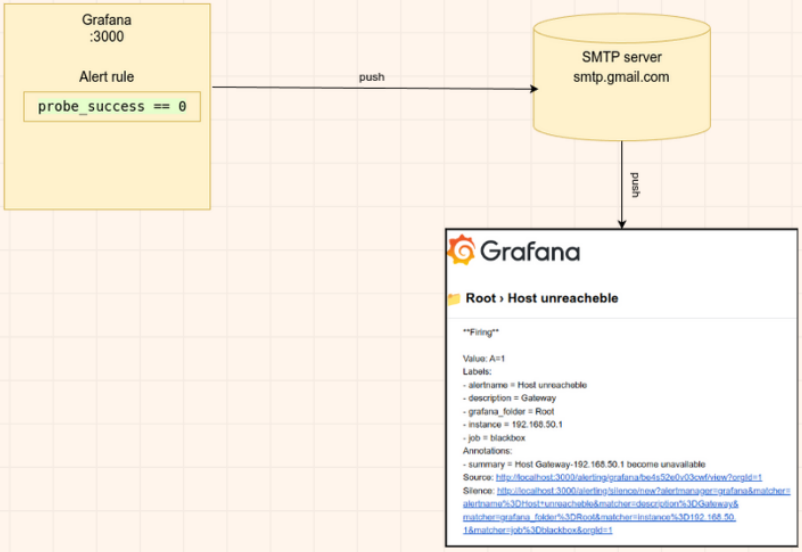


VictoriaMetrics – працює на порту 8428, отримує та зберігає метрики від експортерів

Loki – працюєна на порту 3100, отримує та зберігає логи від експортерів

Grafana – працює на порту 3000, надає веб інтерфейс для візуалізації даних. Збирає дані для візуалізації з victoriametrics та loki

СХЕМА ВІДПРАЦЮВАННЯ СПОВІЩЕННЯ



Протокол аналізу звіту подібності науковим керівником

Заявляю, що я ознайомився (-лась) з Повним звітом подібності, який був згенерований Системою виявлення і запобігання плагіату щодо роботи:
Автор: Олександр ГРЕЧУХА

Назва: Впровадження масштабованої системи моніторингу як засіб підвищення продуктивності інформаційних систем
Координатор: Ігор СІНІЦИН

Підрозділ: ННІІТ

Коефіцієнт подібності 1:0.1

Коефіцієнт подібності 2:0

Тривога: 3

Після аналізу Звіту подібності констатую наступне:

☐ виявлені в роботі запозичення є сумлінними і не мають ознак плагіату. Тому робота визнається самостійною і допускається до захисту;

☐ виявлені в роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і самостійності її автора. Роботу направити на доопрацювання;

☐ виявлені в роботі запозичення є недобросовісними і мають ознаки плагіату або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень. У зв'язку з чим, робота не допускається до захисту.

…… ……

Дата *Підпис Наукового керівника*