МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут автоматики і електротехніки

Кафедра комп'ютерних технологій та інформаційної безпеки

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних технологій та інформаційної безпеки, к.т.н., доцент
_____ С.М. Нужний
«19» ____12____ 2024 р.

Кваліфікаційна робота
на правах рукопису

КВАЛІФІКАЦІЙНА РОБОТА

Дослідження системи моніторингу Grafana

Ступінь вищої освіти: магістр
Галузь знань: 14 Електрична інженерія
Спеціальність: 141 «Електроенергетика, електротехніка та електромеханіка»
Освітньо-професійна програма: Системи технічного захисту інформації, автоматизація її обробки

Виконав: студент 6 курсу групи 6367м
Подгайний Олег Миколайович _____

Керівник:
к.т.н., доцент кафедри КТІБ

Сірівчук Андрій Сергійович _____

Миколаїв – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут автоматики і електротехніки

ЗАТВЕРДЖУЮ
Гарант освітньої програми, к.т.н., доцент, доцент кафедри КТІБ
_____ Турти М.В.
«_19_» ____12____ 2024 р.

ЗАВДАННЯ

на кваліфікаційну роботу

Студент: Подгайний Олег Миколайович

Курс:6

Група: 6366м

Ступінь вищої освіти: магістр

Галузь знань: 14 Електрична інженерія

Спеціальність: 141 «Електроенергетика, електротехніка та електромеханіка»

Освітньо-професійна програма: Системи технічного захисту інформації, автоматизація її обробки

1. Тема роботи: Дослідження системи моніторингу Grafana

затверджена наказом НУК від «_14__» жовтня 2024 р. № 1075 - уч

2. Вихідні дані до роботи: об'єкт дослідження – функціональні можливості системи моніторингу Grafana для систем захисту інформації

призначення та область застосування систем моніторингу; огляд Open telemetry protocol; автоматизація аналізу телеметрії та оповіщення на базі системи Grafana.

4. Терміни виконання завдання:
термін видачі завдання: «_04_» вересня 2024 р.
термін подання роботи: «_19_» грудня 2024 р.

6. План-графік виконання кваліфікаційної роботи

№ п/п	Заходи	Дата		Готовність	Відмітка про виконання
		Навч. тиждень	Контрольна дата		
1.	Наукове стажування	1 - 8	25.10.2024	Звіт з наукового стажування	
2.	Організаційні збори	9	31.10.2024	Попередній варіант змісту КР	
3.	Рубіжний контроль	10	05.11.2024	Попередній варіант оглядових розділів	
4.	Рубіжний контроль	13	28-29.11.2024	1. Попередній варіант повної КР. 2. Попередній варіант презентації.	
5.	Рубіжний контроль	15	12.12.2024	Доповідь на 12-ій Всеукраїнській науково-технічній конференції з міжнародною участю «СПІБТ–2024»	
6.	Перевірка на плагіат	15	12-13.12.2024	Електронний варіант кваліфікаційної роботи	
7.	КЕ на рівень « магістра»	16	17-18.12.2024	Здавання державного екзамену зі спеціальності на ступінь бакалавра	
8.	Кафедральний захист	16	19.12.2024	1. Оформлена КР (в форматі pdf) (передається студентом на кафедрі для внутрішньої рецензії). У разі отримання позитивної внутрішньої рецензії, КР подається на зовнішню рецензію. 2. Оформлена презентація (в форматі pdf).	
9.	Подання документів на захист	16	20.12.2024	1. Подання щодо захисту КР: тема, успішність, висновок керівника та висновок кафедри про КР. 2. Зовнішня рецензія (окремо від КР). Резюме на КР. 3. Електронний варіант та роздрукована презентація в кількості 5 примірників. 4. Електронний варіант КР на диску	
10.	Захист ДР	17	24,26,27, 28.12.2024	1. Приєднання студента до засідання кваліфікаційної комісії (конференції) у встановлений час для проведення процедури дистанційного захисту 2. Процедура захисту КР.	

ПРИМІТКА: Завдання додається до виконаної роботи та подається до атестаційної комісії.

Керівник
К.т.н., доцент _____ А.С. Сірівчук

Студент _____ О.М. Подгайний

АНОТАЦІЯ

Подгайний О.М. Дослідження системи моніторингу *Grafana*

Кваліфікаційна робота: 63 с.; 21 рис.; 36 джерел та 1 додаток.

Актуальність роботи визначається необхідністю постійного моніторингу компютерних систем. Засоби моніторингу надають можливість попередити збої в інформаційнфй безпеці, а не боротися з її наслідками.

Метою роботи є дослідження можливостей системи *Grafana*. *Grafana* є безкоштовною системою моніторингу, що постійно розвивається та має підтримку великої спільноти.

В кваліцікаційні роботі проведено аналіз джерел даних та засобів їх обробки. В роботі було розгорнуто систему *Grafana* для моніторингу мережі в операційній системі *Ubuntu*. Також в системі було створено спеціалізований dashboards який забезпечу візуалізацію лише корисної поточної інформації.

Ключові слова: *Grafana*, моніторинг мережі, *OTLP*.

ANNOTATION

Podhayny O.M. Research of the Grafana monitoring system

Qualification work: 63 p.; 21 fig.; 36 sources and 1 appendix.

The relevance of the work is determined by the need for constant monitoring of computer systems. Monitoring tools provide an opportunity to prevent failures in information security, rather than to deal with its consequences.

The purpose of the work is to study the capabilities of the Grafana system. Grafana is a free monitoring system that is constantly developing and has the support of a large community.

In the qualification work, an analysis of data sources and means of their processing was carried out. In the work, the Grafana system for network monitoring was deployed in the Ubuntu operating system. Also, a specialized dashboard was created in the system that will provide visualization of only useful current information.

Keywords: Grafana, network monitoring, OTLP.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

FTP – File Transfer Protocol

IMAP – Internet Message Access Protocol

NNTP – Network News Transfer Protocol

NRDP – Nagios Remote Data Processor

NRPE – Nagios Remote Plugin Executor

OTLP – OpenTelemetry Protocol

POP – Post Office Protocol

SIEM – Security information and event management

SMTP – Simple Mail Transfer Protocol

SNMP – Simple Network Management Protocol

ЗП – законне перехоплення

ОС – операційна система

ВСТУП

Комп’ютерні системи життєво важливі для роботи будь-якої великої компанії. Хоча персональні і принтери для працівників є важливими, зазвичай основними «гвинтами» комп’ютерної системи є сервери, системи зберігання даних, мережеве обладнання, система резервного копіювання тощо, які знаходяться за лаштунками.

Моніторинг цих важливих систем надзвичайно важливий і є ключовим компонентом будь-якої комп’ютерної системи. Моніторинг комп’ютерної системи зазвичай включає встановлення програмного забезпечення для керування на пристрої, яке надсилає сповіщення системному адміністратору про будь-які проблеми, які можуть виникнути на пристрої. Хоча сама важливість цих систем заслуговує на моніторинг, безпека, збір даних і проактивне реагування є додатковими причинами для моніторингу комп’ютерної мережі.

Мета роботи дослідження системи моніторингу *Grafana* для впровадження її як системи моніторингу та менеджменту мережевих ресурсів.

Для виконання поставленої мети необхідно виконати наступні завдання:

– аналіз підходів до моніторингу ресурсів автоматизованих систем;

– озанйомитись з *Open telemetry protocol*;

– провести аналіз інструментів *Grafana*;

– провести розгортання та налаштування системи.

Результати роботи частково опубліковані в наступному джерелі:

Подгайний О. М. Аналіз інструметів системи моніторингу *Grafana*. Матеріали: Всеукраїнська науково-технічна конференції з міжнародною участю «Сучасні проблеми інформаційної безпеки на транспорті». Миколаїв: НУК, 2024.

Результати роботи можуть бути використанні при підготовці студентів в рамках дисциплін: «Безпека інформаційних систем» та «Управління інформаційної безпеки».

1. ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ СИСТЕМ МОНІТОРИНГУ

1.1 Аналіз проблем забезпечення спостережності автоматизованих систем

Моніторинг комп’ютерної системи так само важливий, як і сама система. Моніторинг забезпечує проактивне реагування, безпеку та збір даних і загальний стан комп’ютерної системи. Хоча моніторинг не вирішує проблеми, він призводить до більш стабільних і надійних комп’ютерних систем.

Звичайно, ви можете запускати свої критично важливі IT-системи та бізнес-системи, не знаючи, чи працюють вони зараз чи ні. У сучасному бізнесі, якщо «критичне» не означає «треба працювати 24/7».

Однією з найважливіших причин для моніторингу мережі є те, що він дозволяє системному адміністратору діяти проактивно, а не реагувати та інцеденти. Деякі компанії вирішили використовувати методологію «виправлення поломок», що означає, що якщо щось зламалося, виправте це. Це призводить до тривалого простою, що компанії не можуть собі дозволити. Моніторинг допомагає сповіщати системного адміністратора, перш ніж щось зламається, і дозволяє йому вирішити проблему до того, як вона стане збоєм. Наприклад, моніторинг може сповістити системного адміністратора про погіршення роботи жорсткого диска на сервері. Системний адміністратор отримує сповіщення про це та замінює пошкоджений жорсткий диск на новий. Без моніторингу пошкоджений жорсткий диск може перетворитися на несправний жорсткий диск, що спричинить тривалий збій у роботі та можливу втрату даних.

Надійна безпека комп’ютерної системи в даний час на умі більшості керівників компаній, оскільки дані компанії є одним із її найцінніших активів. Компанії хочуть бути впевненими, що їх дані зберігаються в безпечному та

недоступному місці. Через це системні адміністратори наполегливо працюють над тим, щоб їхні дані мали доступ лише ті люди, яким потрібен до них доступ. Це досягається шляхом впровадження захищених систем програмного та апаратного забезпечення, які забезпечують цю функціональність.

Однак люди, які намагаються отримати доступ до даних компанії без авторизації, завжди винаходять нові методи. Моніторинг дозволяє системному адміністратору забезпечити безпеку даних компанії. Залежно від типу програмного забезпечення моніторингу, яке використовується, програмне забезпечення може надавати інформацію про доступ до файлів, вторгнення в мережу тощо. Володіння цією інформацією не тільки дозволяє системному адміністратору зупинити будь-які поточні порушення безпеки, але й дозволить йому працювати над тим, щоб цього не сталося.

Поширене помилкове уявлення про моніторинг полягає в тому, що він потрібен для негативних речей; проблеми з обладнанням, порушення безпеки тощо. Хоча це правда, стабільна комп'ютерна система не завжди має ці проблеми. Ще однією перевагою моніторингу комп'ютерної системи є збір даних. Хороша система моніторингу також може зберігати історичну інформацію. Ця інформація дозволяє системному адміністратору переглядати тенденції продуктивності будь-якої системи. Це може допомогти прийняти рішення щодо додаткових ресурсів або компонентів, які можуть знадобитися для забезпечення більш надійної системи.

Особли увагу в системах моніторингу необхідно приділяти серверам, віртуальним машинам чи набіру веб-служб. Це критично важливі для бізнесу системи але вони зазвичай не містять окремих моніторів тому спостереження за їх роботою відводиться на спеціальні служби.

1.2 Аналіз підходів до моніторингу ресурсів автоматизованих систем

Без моніторингу мережі неможливо отримати перегляд у реальному часі підключеного середовища. Але за допомогою звітів *Network Monitoring* ви

можете озирнутися назад, щоб виявити проблеми та тенденції. Не менш важливими є журнали, які покращують вигляд «дзеркала заднього виду», оскільки вони містять усі дані для всіх елементів, які ви контролюєте.

Щоб підтримати криміналістичну експертизу безпеки та по-справжньому зрозуміти мережу, IT професіонали впроваджують найкращі методи керування мережевими журналами (які починаються з реєстрації та аналізу в першу чергу).

1.2.1 Сніфінг.

Сніфінг – це процес моніторингу та захоплення всіх пакетів, що проходять через дану мережу, за допомогою інструментів сніфінгу. Це форма «прослуховування телефонних проводів» і отримання інформації про розмову. Його також називають прослуховуванням телефонних розмов у комп'ютерних мережах.

Існує велика ймовірність того, що якщо набір портів корпоративного комутатора відкритий, то один із їхніх співробітників зможе перехопити весь трафік мережі. Будь-хто, хто знаходиться в тому самому фізичному місці, може підключитися до мережі за допомогою кабелю *Ethernet* або бездротового з'єднання з цією мережею та прослухати загальний трафік.

Іншими словами, сніфінг дозволяє вам бачити всі види трафіку, як захищені, так і незахищені. У відповідних умовах і за допомогою правильних протоколів атакуюча сторона може зібрати інформацію, яка може бути використана для подальших атак або для створення інших проблем для власника мережі чи системи.

З мережі можна перехопити конфіденційну інформацію таку як: трафік електронної пошти, *FTP* паролі, веб-трафіки, паролі *Telnet,* Конфігурація роутера, сеанси чату, трафік *DNS*.

Сніффер зазвичай перемикає мережевий адаптер системи в безладний режим, щоб він прослуховував усі дані, що передаються в його сегменті.

Безладний режим відноситься до унікального способу обладнання *Ethernet*, зокрема мережевих інтерфейсних плат (*NIC*), який дозволяє *NIC* отримувати весь трафік у мережі, навіть якщо він не адресований цьому *NIC*.

За замовчуванням мережевий адаптер ігнорує весь трафік, який не адресований йому, що робиться шляхом порівняння адреси призначення пакета *Ethernet* з апаратною адресою (відомою також як *MAC*) пристрою. Хоча це цілком доцільно для роботи в мережі, безладний режим ускладнює використання програмного забезпечення для моніторингу та аналізу мережі для діагностики проблем з підключенням або обліку трафіку.

Сніффер може безперервно контролювати весь трафік до комп'ютера через мережевий адаптер, декодуючи інформацію, інкапсульовану в пакетах даних.

Снiфінг може бути активним або пасивним за своєю природою.

Під час пасивного аналізу трафік перехоплюється але жодним чином не змінюється. Пасивне перехоплення дозволяє лише слухати. Він працює з концентраторами. На пристрої-концентраторі трафік надсилається на всі порти. У мережі, яка використовує концентратори для підключення систем, усі хости в мережі можуть бачити трафік. Тому зловмисник може легко перехопити трафік, що проходить.

Хороша новина полягає в тому, що хаби сьогодні майже застаріли. Більшість сучасних мереж використовують комутатори. Отже, пасивне нюхання не є більш ефективним.

Під час активного перехоплення трафік не лише перехоплюється та відстежується, але також може бути змінений певним чином, як це визначається атакою. Активний аналіз використовується для аналізу мережі на основі комутатора. Він передбачає введення пакетів розпізнавання адреси у цільову мережу для затоплення таблиці адресованої пам'яті комутатора. Комутатор відстежує, який хост підключено до якого порту.

Такі протоколи, як випробуваний і справжній *TCP/IP*, ніколи не розроблялися з урахуванням безпеки, тому не створюють особливого опору потенційним зловмисникам. Декілька правил дозволяють легко перехопити та проаналізувати трафік: −

– HTTP – використовується для надсилання інформації у вигляді відкритого тексту без будь-якого шифрування, тому є справжньою знахідкою;

– *SMTP* (*Simple Mail Transfer Protocol*) − *SMTP* в основному використовується для передачі електронних листів. Цей протокол є ефективним, але він не містить жодного захисту від прослуховування;

– *NNTP* (*Network News Transfer Protocol*) − використовується для всіх типів зв'язку, але його основним недоліком є те, що дані та навіть паролі надсилаються через мережу як відкритий текст;

– *POP* (*Post Office Protocol*) − *POP* використовується лише для отримання електронних листів із серверів. Цей протокол не включає захист від прослуховування, оскільки його можна перехопити;

– *FTP* (*File Transfer Protocol*) − *FTP* використовується для надсилання та отримання файлів, але він не пропонує жодних функцій безпеки. Усі дані надсилаються як чіткий текст, який можна легко проаналізувати;

– *IMAP* (*Internet Message Access Protocol*) − *IMAP* за своїми функціями такий самий, як *SMTP*, але він дуже вразливий для перехоплення;

– *Telnet* − *Telnet* надсилає все (імена користувачів, паролі, натискання клавіш) через мережу як відкритий текст, і, отже, його можна легко пронюхати.

Сніфери - це не тупі утиліти, які дозволяють переглядати лише поточний трафік. Якщо ви дійсно хочете проаналізувати кожен пакет, збережіть запис і переглядайте його, коли це дозволить час.

Законне перехоплення (ЗП) визначається як дозволений законом доступ до даних комунікаційної мережі, таких як телефонні дзвінки чи повідомлення електронної пошти. ЗП завжди повинен керуватися законними повноваженнями з метою аналізу чи доказів. Отже, ЗП – це процес безпеки, у якому оператор мережі або постачальник послуг надає правоохоронним органам дозвіл на доступ до приватних комунікацій окремих осіб або організацій.

Майже всі країни розробили та ввели в дію законодавство для регулювання процедур законного перехоплення; групи стандартизації створюють специфікації технології ЗП. Зазвичай діяльність ЗП здійснюється з метою захисту інфраструктури та кібербезпеки. Однак оператори інфраструктур приватних мереж можуть підтримувати можливості ЗП у своїх власних мережах як невід'ємне право, якщо інше не заборонено.

1.2.2 *Simple Network Management Protocol*.

Simple Network Management Protocol (*SNMP*) – це широко використовуваний протокол для керування мережею, який забезпечує стандартизовану структуру для моніторингу та керування мережевими пристроями, такими як маршрутизатори, комутатори, сервери, принтери, брандмауери та балансувальник навантаження. Він працює на прикладному рівні набору Інтернет-протоколів і дозволяє мережевим адміністраторам керувати продуктивністю мережі, знаходити та вирішувати проблеми мережі, а також планувати розвиток мережі.

SNMP – це стандартний протокол Інтернету, який використовується для керування та моніторингу підключених до мережі пристроїв у *IP*-мережах. *SNMP* — це протокол прикладного рівня , який використовує номер порту *UDP* 161/162. *SNMP* використовується для моніторингу мережі, виявлення збоїв у мережі та іноді навіть для налаштування віддалених пристроїв.

Архітектура *SNMP* складається з трьох основних компонентів:

– менеджер SNMP: це централізована система, яка використовується для моніторингу мережі. Він також відомий як станція керування мережею (NMS). Маршрутизатор , який запускає програму сервера SNMP, називається агентом, а хост, який запускає програму клієнта SNMP, називається менеджером;

– агент SNMP: це програмний модуль керування програмним забезпеченням, встановлений на керованому пристрої. Менеджер отримує доступ до значень, що зберігаються в базі даних, тоді як агент підтримує інформацію в базі даних. Наприклад, щоб визначити, перевантажений

маршрутизатор чи ні, менеджер може перевірити відповідні змінні, які зберігає маршрутизатор, наприклад, кількість отриманих і переданих пакетів;

– інформаційна база управління: містить інформацію про ресурси, якими потрібно керувати. Ця інформація організована ієрархічно. Він складається з екземплярів об'єктів, які по суті є змінними.

Повідомлення *SNMP*:

– *GetRequest*: він просто використовується для отримання даних від агентів *SNMP*. У відповідь на це агент *SNMP* відповідає запитуваним значенням у вигляді відповідного повідомлення;

– *GetNextRequest*: щоб отримати значення змінної, менеджер надсилає агенту повідомлення *GetNextRequest*. Значення записів у таблиці витягуються за допомогою такого типу зв'язку. Менеджер не зможе отримати доступ до значень, якщо йому не відомі індекси записів. Повідомлення *GetNextRequest* використовується для визначення об'єкта за певних обставин;

– *SetRequest*: використовується менеджером SNMP для встановлення значення екземпляра об'єкта в агенті SNMP;

– *Request*: надсилаючись у відповідь на повідомлення *Set*, воно міститиме нове встановлене значення як підтвердження того, що значення встановлено.

– *InformRequest*: його було додано до *SNMPv2c* і використовується для визначення того, отримав менеджер повідомлення перехоплення чи ні. Це те саме, що і перехоплення, але додає підтвердження, яке не надає перехоплення.

Рівні безпеки SNMP:

– *noAuthNoPriv*: цей рівень безпеки (без автентифікації, без конфіденційності) використовує рядок спільноти для автентифікації та без шифрування для конфіденційності;

– *authNopriv*: цей рівень безпеки (автентифікація , без конфіденційності) використовує *HMAC* з *Md*5 для автентифікації, а шифрування не використовується для конфіденційності;

– *authPriv*: цей рівень безпеки (автентифікація, конфіденційність) використовує *HMAC* з *MD*5 або *SHA* для автентифікації, а шифрування використовує алгоритм *DES*-56.

*SNMPv*1: він використовує рядки спільноти для автентифікації та використовує лише *UDP*. *SNMPv*1 є першою версією протоколу. Це описано в *RFC* 1155 і 1157 і його легко налаштувати.

*SNMPv*2*c*: для автентифікації використовуються рядки спільноти. Він використовує *UDP*, але його можна налаштувати на використання *TCP*. Транспортні відображення та типи пакетів протоколів включені в цю оновлену версію. Однак він також використовує поточну адміністративну структуру *SNMPv*1 «на основі спільноти», тому версія називається *SNMPv*2*c*. *RFC* 1901, *RFC* 1905 і *RFC* 1906 описують його.

*SNMPv*3: він використовує *MAC* на основі хешу з *MD*5 або *SHA* для автентифікації та *DES*-56 для конфіденційності. Ця версія використовує *TCP*. *NMPv*3 забезпечує віддалену конфігурацію об’єктів SNMP. Це найбезпечніша версія на сьогоднішній день, оскільки вона також включає автентифікацію та шифрування, які можна використовувати окремо або в комбінації. *RFC* 1905, *RFC* 1906, *RFC* 2571, *RFC* 2572, *RFC* 2574 і *RFC* 2575.6 є *RFC* для *SNMPv*3.

Задачі *SNMP*:

–використовується для моніторингу мережі;

– виявляє будь-які збої в мережі;

– можна використовувати для налаштування віддалених пристроїв;

– стандартизований спосіб збору інформації про всі типи пристроїв від різних виробників серед мережевої галузі.

1.2.3 Журнали подій.

Журнали мережевих пристроїв дають уявлення про події, що відбуваються на пристроях, програмах і мережевому трафіку. Вони допомагають у вирішенні проблем і реагуванні на інциденти.

Кожен маршрутизатор, комутатор, програма та інший пристрій, зберігає журнали мережевої активності та внутрішніх подій. Ці журнали корисні для

усунення несправностей, реагування на інциденти та цифрової криміналістики та навіть можуть використовуватися для розробки програм.

Різні типи журналів фіксують різні типи подій. Вони можуть бути журнали трафіку, журнали аудиту та протокол системного журналу. Ці журнали є одними з найпростіших та інформативних, які можна знайти на будь-якому пристрої.

Журнали трафіку містять інформацію про мережевий трафік, включаючи початок і кінець з'єднання, спроби з'єднання, запити та іншу інформацію, що стосується зв'язку одного пристрою з іншим.

Ці журнали є чудовим джерелом інформації та можуть використовуватися кінцевими користувачами, мережевими адміністраторами та аналітиками безпеки для виявлення проблем. Журнали часто зберігаються в базових форматах, з якими неприємно працювати, але доступно багато безкоштовних і платних інструментів, які допоможуть візуалізувати та організувати мережевий трафік.

Першим кроком до аналізу мережевого трафіку є збір і зберігання трафіку. Більшість мережевих пристроїв зроблять це за замовчуванням і зберігають журнали десь у файловій системі. Крім того, багато організацій налаштовують свої мережеві пристрої для пересилання цих журналів до інструментів, спеціально розроблених для допомоги в аналізі мережевого трафіку.

Після збору журнали слід класифікувати, застосовуючи теги на основі конкретних критеріїв, таких як тип події, позначка часу, джерело та призначення, а також будь-які інші атрибути, які можуть бути корисними. Тепер ви можете використовувати ці журнали трафіку для вирішення проблем і покращення продуктивності.

Аналітики безпеки також можуть використовувати журнали трафіку для виявлення аномальної та потенційно зловмисної активності. Деякі ключові елементи для пошуку в журналах включають вихідні з'єднання в нестандартний час, трафік до підозрілих місць призначення та з них, а також

раптове збільшення обсягу вихідного трафіку. Це може свідчити про зловмисну активність у вашій мережі, і вам може знадобитися використовувати інші журнали для визначення джерела.

Журнал аудиту фіксує дані про такі дії, як події на рівні програми. Наприклад, ви щойно підійшли до свого робочого столу й намагаєтесь увійти в робочий інструмент керування проектами. Ви забули свій пароль і провалили першу спробу, але потім вгадали правильно й успішно ввійшли з другої спроби.

Щоб цього не повторилося, змініть пароль на новий, який вам буде легше запам’ятати. Кожна з цих дій реєструється як окрема подія в журналах аудиту програми.

Журнали аудиту можна використовувати під час аудитів відповідності, щоб підтвердити, що певні події відбуваються чи ні. Наприклад, ви можете використовувати журнали аудиту, щоб визначити, хто отримує доступ до системи чи сховища, підтвердити, що резервні копії системи створюються через певні проміжки часу, або підтвердити, що з’єднання припиняються після визначеного періоду.

Припустімо, що ми порівнюємо мітку часу цих підозрілих з’єднань із діяльністю журналу аудиту приблизно в той самий час. Можливо, ми зможемо віднести це підключення до конкретного користувача або облікового запису служби. Якщо ми потім переглянемо журнали аудиту на наявність аномальної активності, пов’язаної з цим користувачем, ми можемо виявити кілька невдалих спроб вводити пароль протягом кількох годин або навіть кількох днів, а потім успішну спробу, що вказує на атаку грубою силою. Ми можемо виявити, що користувач завантажив і запустив програмне забезпечення незадовго до початку підозрілих з’єднань, що вказує на виконання фішингової атаки .

Журнали аудиту надають величезну кількість інформації та безцінне розуміння діяльності користувачів організації та програм. З цієї причини багато організацій вирішують пересилати свої журнали аудиту або журнали

подій до інструменту *SIEM* (*Security information and event management*) для моніторингу та попередження в реальному часі.

Протокол *syslog* допомагає передавати ці дані до централізованого інструменту журналювання та моніторингу.

Кожне повідомлення системного журналу складається з ключових компонентів, включаючи заголовок, структуровані дані та повідомлення. Заголовок містить інформацію про пріоритет, ім'я хоста, мітку часу тощо. Компонент структурованих даних складається з блоків даних у форматі пари «ключ:значення». Компонент повідомлення описує подію, про яку повідомляється. Приклад повного повідомлення системного журналу може виглядати так:

timestamp device-id severity code detailed-information

Ця інформація буде перенаправлена в централізований інструмент, наприклад *SIEM*. Деякі з найбільш популярних SIEM, про які ви можете почути, включають *Splunk*, *ELK Stack* і *LogRhythm*.

Крім того, більшість постачальників хмарних послуг пропонують власні інструменти для внутрішнього керування журналами, такі як *AWS CloudTrail* і *CloudWatch* .

Можливість переглядати та досліджувати дані про мережеву активність, дані програм і події на системному рівні в одному інструменті робить моніторинг у реальному часі більш ефективним. Те, що часто називають «єдиною скляною панеллю», також допомагає гарантувати, що сповіщення не залишаться непоміченими.

Щоб досягти такого типу пересилання журналів, вам знадобиться сервер *syslog* , який складається з кількох компонентів:

– прослуховувач системного журналу: це дозволить серверу приймати вхідний трафік із кількох джерел журналу;

– база даних: оскільки дані надходять, їх потрібно буде зберігати впорядковано;

– програмне забезпечення для фільтрації: сервери Syslog містять багато даних, тому вам знадобиться ефективний спосіб їх фільтрації та пошуку саме того, що ви шукаєте.

Наскільки простим чи складним буде розгортання вашого сервера syslog, залежатиме від вибраного вами інструменту та від того, чи отримуєте ви підтримку від постачальника. На щастя, для більшості інструментів доступна достатня кількість документації, а будь-які запитання, які у вас можуть виникнути, ймовірно, можна знайти на технічному онлайн-форумі на веб-сайті постачальника.

Навіть якщо ви вважаєте, що все налаштовано правильно, час від часу варто вручну переглядати журнали. Ви б зробили це з кількох причин, наприклад, щоб переконатися, що ваш інструмент правильно подає сповіщення на основі встановлених вами критеріїв, а також щоб виявити будь-яку аномальну поведінку, достатньо підозрілу, щоб вимагати подальшого дослідження, але не настільки дивну, щоб викликати сповіщення в інструменті.

Організації зі зрілими мережевими та захисними командами можуть навіть проводити регулярні перевірки журналів, які в галузі називають «полюванням на загрози». Іноді ці пошуки загроз можуть передбачати пошук індикаторів компрометації, виявлених під час нещодавніх порушень безпеки, або ймовірних дій, пов’язаних із нещодавно виявленими *CVE* (загальні вразливості та експозиції), наприклад дивні *SSH* або *SMB*-з’єднання. Рекомендується регулярно перевіряти журнали, щоб переконатися, що нічого не пропущено через технічну проблему.

У більшості випадків ви матимете пристойний контроль над тим, яка інформація реєструється. Чим надійніший пристрій або програма, тим детальнішими можуть бути ваші журнали, і тим більше інформації вам неминуче знадобиться проаналізувати.

Якщо ваша організація вирішить, ви можете сповіщати на основі таких критеріїв, як ідентифікатор події або серйозність, пересилаючи лише ту інформацію, яку вважаєте корисною. Будьте обережні, налаштовуючи пересилання журналу. Занадто мало журналів збільшує ризик не побачити потенційних загроз, тоді як занадто багато журналів збільшує ризик пропуску очевидних сповіщень, значного споживання ресурсів і наражає аналітиків на «втому від сповіщень», коли вони переглядають велику кількість сповіщень.

Найкращий спосіб для вашої організації отримати максимальну віддачу від керування журналами – це встановити та підтримувати чіткі політики, які визначають, яку інформацію з яких систем слід реєструвати, як довго слід зберігати журнали, хто може отримати доступ до журналів і які відповіді мають надходити з різних типів сповіщень. Це допомагає гарантувати, що кожен знає свою роль у більш широкому процесі керування журналами.

Керування журналами також є процесом, що постійно розвивається, і потребує постійного оновлення. Найімовірніше, хтось, відповідальний за підтримку централізованого інструменту керування журналами, координуватиме роботу з аналітиками безпеки, які працюють із інструментом, щоб точно налаштувати критерії сповіщень.

У міру того, як ваше середовище змінюється, а нові вразливості та експлойти загрожують вашій мережі, вам потрібно буде налаштувати свої інструменти, щоб вони стали точнішими, то м’якшими, щоб знайти правильний баланс між надто великою та надто малою кількістю сповіщень. Стійкість вашої організації до ризику також буде чинником у тому, як ви налаштуєте ці критерії попередження.

1.2.4 Засоби діагностики мережі.

Засоби діагностики мережі – це програмні рішення, призначені для моніторингу, діагностики та усунення проблем із мережами. Вони аналізують продуктивність, надійність і безпеку комп’ютерних мереж і можуть автоматизувати виправлення мережевих помилок. Інструменти діагностики мережі мають вирішальне значення для запобігання простою мережі та

кібератакам. Вони допомагають адміністраторам швидко визначити, чи має місце порушення безпеки або *DDoS*-атака.

Основні функції включають моніторинг мережі в режимі реального часу, політику конфігурації для компонентів пристроїв і відповідності, сповіщення про продуктивність і відображення топології. Вони дозволяють командам контролювати продуктивність мережі та збирати дані для криміналістичного аналізу мережевої активності.

Інструменти діагностики мережі використовуються мережевими адміністраторами, IT-фахівцями та розробниками для виявлення помилок мережі, оптимізації продуктивності мережі та усунення проблем. Вони допомагають значно пришвидшити процес виявлення та усунення проблем із безпекою мережі та зробити його масштабованішим у міру зростання вашої організації.

Інструменти мережевої діагностики постійно аналізують і відстежують мережі для виявлення та діагностики мережевих проблем. Інструменти можуть працювати різними способами, включаючи аналіз трафіку, тестування підключення до мережі та моніторинг налаштувань конфігурації.

Ці інструменти працюють за допомогою поглибленого аналізу пакетів для збору та перевірки даних у комп’ютерній мережі. Вони збирають дані з мережевих пристроїв для моніторингу та аналізу всього мережевого трафіку, що дозволяє їм виявляти проблеми та позначати їх адміністраторам мережі. Проблеми можуть включати збої в мережі, проблеми з продуктивністю або вразливі місця в безпеці.

Коли виявляється проблема, ці рішення також можуть допомогти вирішити проблему, сповіщаючи адміністраторів, автоматизуючи робочі процеси та надаючи вказівки або повідомляючи про проблеми. Наприклад, інструмент може сповістити адміністраторів про проблему з продуктивністю мережі або безпекою, а потім запропонувати кроки для її усунення, як-от оновлення програмного або апаратного забезпечення.

Інструменти мережевої діагностики – це служби, які можуть охоплювати широкий спектр випадків використання для різних організацій. Загальні ключові функції, які надають ці служби, включають:

– моніторинг мережі: безперервний моніторинг мережі для виявлення проблем з продуктивністю та скорочення часу простою мережі;

– автоматичне сповіщення: сповіщення адміністраторів про проблеми продуктивності в мережі;

– діагностика та оцінка: прискорення виправлення, пропонуючи аналіз із діагностикою та рекомендаціями;

– інформаційна панель адміністратора: поглиблені звіти та аналітика мережі з відстеженням ефективності за попередні періоди;

– спеціальні звіти та сповіщення: команди можуть налаштовувати спеціальні звіти з попередженнями для виявлення проблем.

1.3 Інтегровані системи моніторингу

1.3.1 *Nagios.*

Nagios – це інструмент моніторингу IT-системи з відкритим кодом. Він був розроблений для роботи в операційній системі *Linux* і може контролювати пристрої під керуванням ОС *Linux, Windows* і *Unix* .

Програмне забезпечення *Nagios* проводить періодичні перевірки критичних параметрів програми, мережі та ресурсів сервера. Наприклад, *Nagios* може контролювати використання пам’яті , використання диска та завантаження мікропроцесора, а також кількість запущених процесів і файлів журналу. *Nagios* також може контролювати такі служби, як *Simple Mail Transfer Protocol*, *Post Office Protocol* 3 , *Hypertext Transfer Protocol* та інші поширені мережеві протоколи. *Nagios* ініціює активні перевірки, тоді як пасивні перевірки надходять із зовнішніх програм, підключених до інструменту моніторингу.

Спочатку випущений у 1999 році як *NetSaint, Nagios* був розроблений Ітаном Галстадом і згодом вдосконалений численними учасниками як проект з

відкритим кодом. *Nagios Enterprises*, компанія, заснована на технології *Nagios Core*, пропонує різноманітні продукти, такі як *Nagios XI*, сервер журналу, мережевий аналізатор і *Fusion*.

Користувачі можуть вибрати роботу в інтерфейсі командного рядка або вибрати веб-графічний інтерфейс користувача в деяких версіях *Nagios* і сторонніх розробників. Інформаційна панель *Nagios* надає огляд критичних параметрів, які відстежуються на активах.

На основі визначених параметрів і порогових значень *Nagios* може надсилати сповіщення про досягнення критичних рівнів. Ці сповіщення можна надсилати електронною поштою та текстовими повідомленнями. Система авторизації дозволяє адміністраторам обмежувати доступ.

Nagios запускає як агентні, так і безагентні конфігурації. Незалежні агенти встановлюються на будь-якому апаратному чи програмному забезпеченні для збору даних, які потім передаються на сервер керування. Безагентний моніторинг використовує існуючі протоколи для емуляції агента. Обидва підходи можуть контролювати використання файлової системи, показники ОС, стани служб і процесів. Приклади агентів *Nagios* включають *Nagios Remote Data Processor (NRDP), Nagios Cross Platform Agent* і *NSClient++*.

Nagios також може запускати віддалені сценарії та плагіни за допомогою агента *Nagios Remote Plugin Executor* (*NRPE*). *NRPE* дозволяє віддалено контролювати системні показники, такі як завантаження системи, використання пам’яті та диска. Він складається з плагіна check_nrpe, який зберігається на локальній машині моніторингу, і *NRDP*, який працює на віддаленій машині. *Nagios* використовує плагін для консолідації даних від агента *NRPE* перед тим, як вони надходять на сервер керування для обробки. *NRPE* також може спілкуватися з агентами *Windows* для моніторингу машин *Windows* .

Nagios підтримує плагіни, які є самостійними доповненнями та розширеннями, щоб користувачі могли визначати цілі та параметри цілі для моніторингу. Плагіни *Nagios* обробляють аргументи командного рядка та обмінюються командами з *Nagios Core*.

Існує близько 50 плагінів, розроблених і підтримуваних *Nagios*, а понад 3000 від спільноти. Ці плагіни класифікуються за списками, включаючи обладнання, програмне забезпечення, хмару, операційну систему (ОС), безпеку, файли журналів і мережеві підключення. Наприклад, якщо використовується в поєднанні з системами вимірювання навколишнього середовища, плагін *Nagios* може обмінюватися даними про змінні навколишнього середовища, такі як температура, вологість або барометричний тиск.

Nagios виявився популярним серед малого та великого бізнесу, а також серед постачальників послуг Інтернету, навчальних закладів, державних установ, закладів охорони здоров'я, виробничих компаній і фінансових установ.

Користувачі можуть вибирати між безкоштовними та платними варіантами, залежно від необхідних послуг та підтримки.

Служба, яка спочатку була відома як *Nagios*, тепер називається *Nagios Core*. *Core* є у вільному доступі як програмне забезпечення для моніторингу IT-систем, мереж та інфраструктури з відкритим кодом. Core містить широкий спектр засобів моніторингу інфраструктури за допомогою плагінів для розширення можливостей моніторингу. Є основою для платних систем моніторингу *Nagios*.

Nagios Core має додатковий веб-інтерфейс, який відображає статус мережі, сповіщення та файли журналу. Core може повідомити свого користувача про проблеми з сервером або хостом. Крім того, Core може контролювати мережеві служби, такі як *SMTP, HTTP* і *Ping* .

Nagios XI – це розширений інтерфейс *Nagios Core*, призначений як версія інструменту моніторингу на рівні підприємства. *XI* діє як програмне забезпечення для моніторингу, менеджер конфігурації та інструментарій. Хоча *Nagios Core* безкоштовний, *XI* потрібно придбати в *Nagios Enterprises*. Окрім тих самих функцій, що й у *Core, XI* додає попередньо налаштовані віртуальні машини (ВМ), інтерфейс користувача веб-конфігурації , графік продуктивності, мобільний додаток, інформаційні панелі, заплановані звіти та технічну підтримку електронною поштою.

Nagios XI відстежує такі компоненти IT-інфраструктури, як програми, ОС, мережі та системні показники. Для цих компонентів інфраструктури підтримуються плагіни для розширення можливостей моніторингу XI.

Nagios Log Server – це інструмент моніторингу та керування журналами, який дає змогу організації переглядати, сортувати та налаштовувати журнали своєї IT-інфраструктури, включаючи журнали подій *Windows*. Сервер журналу може аналізувати, збирати та зберігати зареєстровані дані на основі настроюваних і попередньо призначених специфікацій. Адміністратори можуть налаштувати сповіщення для сповіщення користувачів сервера журналу про потенційну загрозу або несправність активу, що контролюється. Наприклад, сповіщення надходить адміністратору *Microsoft Exchange*, коли є три невдалі спроби входу до сервера *Exchange Server*, що означає, що хтось може невиправдано спробувати вгадати пароль до системи.

Nagios Network Analyzer відстежує мережевий трафік і використання пропускної здатності. *Network Analyzer* може вирішувати збої в мережі, аномалії та загрози безпеці. Функції включають автоматичні сповіщення безпеки, настроюваний моніторинг додатків, інтеграцію з *Nagios IX* і калькулятор використання пропускної здатності.

Nagios Fusion — це служба агрегації для серверів *Nagios Core* та *Nagios XI,* яка показує кілька систем в одному поданні. *Fusion* узагальнює керування мережею, централізуючи функції та дані з *Nagios XI* і *Core* в одному місці, створюючи детальне уявлення про мережеву інфраструктуру. За допомогою *Fusion* адміністратори можуть визначати, які сервери *XI* і *Core* відображаються, а також контролювати, яким користувачам дозволено переглядати ці сервери. Крім того, користувачі *Fusion* можуть увійти на будь-який керований сервер і використовувати кешовані або живі дані для налаштування діаграм та інших графічних елементів для відображення на інформаційних панелях.

1.3.2 *Zenoss*

Zenoss — це IT-моніторингове програмне забезпечення для хмарних, віртуальних і фізичних IT-середовищ. *Zenoss* відстежує сервери, мережі,

віртуальні машини, бази даних та інші апаратні та програмні засоби в IT-інфраструктурі. Подібно до *Nagios, Zenoss* доступний як версія з відкритим вихідним кодом під назвою *Zenoss Core* або більш широкі платні підтримувані опції, включаючи *Zenoss Service Dynamics* і *Zenoss as a Service*. *Service Dynamics* — це локальна версія програмного забезпечення, тоді як *Zenoss as a Service* — це варіант програмного забезпечення як послуги. Подібно до продуктів *Nagios*, продукти *Zenoss* надають плагіни під назвою *ZenPacks*, які розширюють можливості моніторингу.

1.3.3 *Zabbix*

Zabbix – це інструмент моніторингу з відкритим кодом для *ОС Linux, Unix* і *Windows*, який покладається на агентів для збору даних моніторингу. Він також може використовувати загальні протоколи для роботи без агентів. Технологія контролює фізичні та хмарні активи, віртуальні машини, служби та програми. *Zabbix* розвивається для хмарного розгортання, а також локального.

Microsoft SCOM дозволяє користувачам налаштовувати, керувати та контролювати пристрої та програми за допомогою однієї консолі. *SCOM* відстежує серверне обладнання, системні служби, ОС, гіпервізори та програми. *SCOM*, як і *Nagios*, покладається на агентів або моніторинг без агентів для збору даних і підтримує плагіни.

Ми часто використовуємо *Zabbix*, коли хочемо контролювати ресурси наших серверів і не маємо часу переглядати журнали – наприклад, коли ми працюємо з великою кількістю серверів і хочемо переконатися, що все працює належним чином. У цій ситуації *Zabbix* здійснюватиме моніторинг і інформуватиме нас у разі виникнення проблеми, дозволяючи нам налаштувати кілька шляхів зв’язку. *Zabbix* інформує нас, наприклад, якщо певна «кінцева точка» не працює або не вистачає місця на диску, або якщо ЦП перевантажений. *Zabbix* автоматично надсилає нам всю інформацію, яка може вплинути на якість роботи програми.

Zabbix складається з кількох основних програмних компонентів. Їхні обов'язки описані нижче:

– сервер Zabbix є центральним компонентом, якому агенти повідомляють інформацію та статистику щодо доступності та цілісності. Сервер є центральним репозиторієм, де зберігаються всі конфігураційні, статистичні та робочі дані;

– зберігання бази даних. Уся конфігураційна інформація та дані, зібрані Zabbix, зберігаються в базі даних;

– Веб-інтерфейс. Надається веб-інтерфейс, який забезпечує легкий доступ до *Zabbix* з будь-якого місця чи платформи. Інтерфейс є частиною сервера *Zabbix* і зазвичай працює на тій же фізичній машині, що й сервер;

– *Zabbix* проксі. Проксі діє як проксі та може збирати дані про продуктивність і доступність від імені сервера *Zabbix*. Це необов'язкова частина розгортання *Zabbix*, але вона може бути дуже корисною для розподілу навантаження на один сервер *Zabbix*.

– агент *Zabbix* – це спеціальна програма, яка забезпечує зв'язок із головним сервером. Він активно відстежує локальні ресурси та програми та повідомляє зібрані дані на сервер *Zabbix*.

– *Zabbix* може контролювати тисячі даних із серверів, віртуальних машин, програм і мережевих пристроїв у режимі реального часу. Це дозволяє виявляти проблеми до того, як вони привернуть увагу користувачів.

Агент Zabbix збирає дані, які нас цікавлять, наприклад, поточне використання процесора, оперативна пам'ять, потік даних у мережі, завантаження бази даних та ефективність окремих служб (*HTTP, SSH, FTP*).

Потім він надсилає зібрану інформацію на головний сервер, формулюючи її у зрозумілі та зручні для читання таблиці чи діаграми.

Дані зберігаються в реляційних базах даних (*MySQL, Oracle* або *PostgreSQL*), і до них можна отримати доступ через інтуїтивно зрозумілий веб-інтерфейс.

Агент *Zabbix* підтримує як пасивне (опитування), так і активне (перехоплення) запити. *Zabbix* може виконувати перевірки на основі інтервалів, але також можна запланувати конкретні години для опитування елементів.

Пасивні (опитувальні) перевірки:

– *Zabbix*-сервер (або проксі) запитує значення від *Zabbix*-агента;

– агент обробляє запит і повертає значення на сервер *Zabbix* (або проксі);

Активні елементи керування (ловлення):

– агент *Zabbix* запитує список активних перевірок із сервера *Zabbix* (або проксі);

– агент періодично надсилає результати.

Починаючи з версії 3.0, *Zabbix* підтримує зашифрований зв’язок між сервером і агентами, тому ми гарантуємо безпеку всіх даних під час «подорожі».

Zabbix може підтримувати моніторинг тексту *Windows* і журналів подій. Він також має вбудовану підтримку *Windows Management Instrumentation* (*WMI*), що збільшує можливість легкого отримання та моніторингу системної інформації та показників продуктивності в реальному часі з серверів і робочих станцій *Windows*. Це лише деякі з послуг, які він може виконувати завдяки тому, що він є гнучким інструментом, який адаптується до програм.

Інші функції Zabbix включають моніторинг таких речей, як:

– навантаження на процесори, мережеві карти та пам'ять;

– обсяг місця на жорстких дисках;

– мережеві протоколи;

– термін дії сертифікатів SSL на веб-сайтах;

– температура компонентів.

Zabbix повідомить нас про збій різними способами. Однією з найпопулярніших є відправка інформації електронною поштою; Ви також можете налаштувати відправку *SMS*. Це дуже зручне і ефективне рішення дозволяє дізнатися про проблему і швидко на неї відреагувати. У наших командах розробників сповіщення налаштовані таким чином, щоб і електронна пошта, і канал програми *Slack* повідомляли нам про незвичайні ситуації, пов’язані з сервером.

Zabbix можна налаштувати відповідно до власних потреб і вподобань. Хоча кожна проблема, навіть найменша, певним чином важлива, вона

підпорядковується певній ієрархії. Завдяки тригерам, які ми встановили в *Zabbix,* інструмент повідомить нас про проблему відповідно до її важливості.

Приклад: під час конфігурації ми налаштували *Zabbix* на перегляд менше ніж 20% вільного місця на диску як проблему середньої серйозності. Якщо місця на диску менше 10%, проблема має високу серйозність. Система реагує відповідним чином на основі вибраних тригерів. Якщо ми домовилися, що звичайна проблема не потребує негайної відповіді, тоді ми можемо налаштувати *Zabbix* на надсилання нам електронного листа з попередженням, який надійде нам у зручний час, щоб ми не турбувалися про це без потреби. Краще дізнаватися про катастрофічні новини негайно, тому для серйозних проблем ми можемо наказати *Zabbix* вибрати *SMS* як форму зв'язку або, як ми згадували вище, увімкнути екстрену сирену.

У *Zabbix* ви визначаєте рівень пріоритету для нагадування або групи нагадувань і встановлюєте його для відповідних сповіщень. Якщо ви хочете отримувати сповіщення лише про критичні ситуації, таким чином ви встановлюєте пріоритет. Інженери, відповідальні за технічні аспекти проекту, як правило, отримують сповіщення нижчої категорії, тому вони можуть швидко виправляти менші проблеми. З іншого боку, значні збої вирішуються відділом *DevOps*.

1.3.4 *Grafana*

Grafana – це рішення з відкритим вихідним кодом для проведення аналізу даних за допомогою показників, які дають нам зрозуміти складну інфраструктуру та величезний обсяг даних, з якими працюють наші служби, за допомогою настроюваних інформаційних панелей.

Grafana з'єднується з усіма можливими джерелами даних, такими як *Graphite, Prometheus, Influx DB, ElasticSearch, MySQL, PostgreSQL* тощо. Природа рішення з відкритим вихідним кодом допомагає нам альтернативно писати спеціальні плагіни для підключення до будь-якого джерела даних за нашим вибором.

Інструмент допомагає нам вивчати, аналізувати та контролювати дані за певний період часу, який технічно називається аналітикою часових рядів. Це допомагає нам відстежувати поведінку користувачів, поведінку додатків, частоту появи помилок у робочому, передвиробничому або будь-якому іншому середовищі, типи помилок, що виникають, і контекстні сценарії, надаючи відносні дані.

Великий плюс проекту полягає в тому, що він може бути розгорнутий локально організаціями, які не хочуть, щоб їхні дані передавалися в хмару постачальника з міркувань безпеки.

Згодом цей фреймворк набув великої популярності в індустрії та використовується великими зброєю, такими як *PayPal, eBay, Intel* та багато інших. Я обговорю галузеві випадки використання вище в статті.

Крім основного рішення з відкритим вихідним кодом, команда *Grafana* пропонує ще дві послуги під назвою *Grafana Cloud* і *Enterprise*.

Інформаційні панелі *Grafana* отримують дані з підключених джерел даних, таких як *Graphite, Prometheus, Influx DB, ElasticSearch, MySQL, PostgreSQL* тощо. Це лише деякі з багатьох джерел даних, які *Grafana* підтримує за замовчуванням.

Інформаційні панелі містять набір опцій візуалізації, таких як геокарти, теплові карти, гістограми та різноманітні діаграми та графіки, які зазвичай потрібні компанії для вивчення даних.

Приладова панель містить кілька різних окремих панелей на сітці. Кожна панель має різні функції.

Екземпляри програми було розгорртаються як контейнери *Docker*, якими керує *docker swarm*. Були випадки, коли екземпляри не працювали або критична проблема спричиняла збій системи. Усі ці сценарії відстежувалися на інформаційній панелі *Grafana*, що значно полегшило моє життя.

Дані можна отримати з *Prometheus*, який було підключено до інформаційної панелі *Grafana* як джерело даних. З інформаційної панелі надсилалися запити з різними виразами, такими як *min, avg* тощо. І *Prometheus* витягував дані з *cAdvisor* .

Grafana піклується про всю аналітику нашого додатка. Ми можемо легко надсилати запити, візуалізувати, налаштовувати сповіщення та розуміти дані за допомогою показників. Інформаційна панель оснащена різноманітними функціями та постійно вдосконалюється, що допомагає нам зрозуміти складні дані. Від відображення графіків до теплових карт, гістограм, географічних карт тощо. Інструмент має безліч опцій візуалізації для розуміння даних відповідно до нашого випадку використання.

Попередження налаштовуються та спрацьовують, як сповіщення, щоразу, коли відбувається очікуваний сценарій. Про ці події можна повідомити на *Slack* або будь-якому іншому комунікаційному інструменті, який використовує команда моніторингу.

Grafana має вбудовану підтримку прибл. десяток баз даних із великою кількістю плагінів . Розмістіть його локально або на будь-якій хмарній платформі на ваш вибір.

Він має вбудовану підтримку для *Graphite* і таких виразів, як *add*, *filter, avg, min, max* функції тощо для користувацького отримання даних. Він також має вбудовану підтримку *Influx DB, Prometheus, ElasticSearch і CloudWatch*.

Grafana Cloud – це хмарна, високодоступна, продуктивна повністю керована відкрита *SaaS* (програмне забезпечення як послуга) платформа для вимірювання показників. Досить корисно для тих, хто не хоче брати на себе навантаження з локального розміщення рішення та хоче не хвилюватися про керування всією інфраструктурою розгортання.

Він працює на кластерах *Kubernetes*. Сервер сумісний з *Prometheus* і *Graphite*. Корпоративний сервіс містить усі функції *Grafana Cloud*, а також преміум-плагіни, джерела даних і преміальну підтримку від основної команди. Ми отримуємо відповіді на *SLA*, навчання та багато іншого. Для отримання додаткової інформації відвідайте .

Інформаційні панелі *Grafana* розгорнуті в усій індустрії, будь то ігри, Інтернет речей, *Fintech* або *eComm*. *StackOverflow* використовує цей інструмент, щоб дозволити своїм розробникам і командам із забезпечення надійності сайтів

створювати спеціальні інформаційні панелі для візуалізації даних і оптимізації продуктивності серверів.

Digital Ocean використовує *Grafana* для обміну даними візуалізації між своїми командами та має спільну платформу обміну візуальними даними.

Prometheus – це інструмент моніторингу даних з відкритим кодом. Комбінація *Prometheus* і *Grafana* – це фактична комбінація, яка використовується в галузі для розгортання налаштування візуалізації даних. Інформаційна панель *Grafana* використовується для візуалізації даних, тоді як бекенд працює на основі *Prometheus* .

Хоча *Prometheus* також має функції візуалізації даних, для візуалізації даних краще використовувати *Grafana*. Запити запускаються з інформаційної панелі *Grafana*, а дані витягуються з *Prometheus*.

Він діє як ідеальна модель даних із відкритим кодом для зберігання даних часових рядів.

Graphite Grafana, знову ж таки, є інструментом моніторингу. Це полегшує зберігання та візуалізацію даних часових рядів. В ідеалі *Graphite* використовується як джерело даних для інформаційної панелі *Grafana* в налаштуваннях моніторингу даних.

У *Grafana* є досить просунутий редактор запитів *Graphite*, який дозволяє нам взаємодіяти з даними за допомогою виразів і функцій.

Висновки до розділу

Моніторинг роботи комп’ютерної систми є невід’ємною частиною забезпечення її безперебійної роботи. Джерелами даних для системи моніторингу можу виступати як повідомленнві від компютерів так і від мережевого обладнання. Наразі існують готові рішення для зберігання та відображення інформації з різних джерел, що забезпечує більш простий спосіб моніторингу

2. СТРУКТУРА *GRAFANA*

2.1 Огляд *Open telemetry protocol*

Протокол *OpenTelemetry Protocol* (*OTLP*) виступає як маяк, який направляє вас до єдиного підходу до збору та експорту телеметричних даних. *OTLP* діє як універсальна мова, що дозволяє додаткам легко обмінюватися показниками з будь-яким сумісним сервером, незалежно від постачальника чи формату. Уявіть собі, як легко мати всі свої показники, акуратно організовані на одній платформі, готові для аналізу!

OpenObserve легко інтегрується з *OTLP*, надаючи розширені можливості зберігання, візуалізації та аналізу даних.

OTLP (*OpenTelemetry Protocol*) – це протокол, призначений для збору та передачі телеметричних даних. Це частина проекту *OpenTelemetry*, метою якого є надання єдиної платформи для збору й обробки телеметричних даних із різних джерел.

OTLP відіграє вирішальну роль у серіалізації, десеріалізації та транспортуванні телеметричних даних. Він забезпечує стандартизований спосіб серіалізації та десеріалізації телеметричних даних, що робить можливим передачу даних між різними системами та службами.

Буфери протоколу є ключовим компонентом *OTLP*. Вони забезпечують спосіб серіалізації та десеріалізації телеметричних даних, що робить можливим передачу даних між різними системами та службами. Буфери протоколу використовуються для визначення структури телеметричних даних, гарантуючи, що дані правильно відформатовані та можуть бути легко оброблені різними системами.

OTLP підтримує два транспортні механізми: *gRPC* і *HTTP*. *gRPC* – це високопродуктивна структура *RPC*, яка забезпечує швидкий і ефективний спосіб передачі телеметричних даних. *HTTP* – це широко поширений протокол, який

забезпечує простий і легкий у використанні спосіб передачі телеметричних даних.

Специфікації *OTLP* стосуються стандартизованих протоколів і форматів, які використовуються для збору та передачі телеметричних даних. OTLP забезпечує дві операційні структури: *OTLP/HTTP* і *OTLP/gRPC* (рис. 2.1). Ці структури дозволяють передавати телеметричні дані між різними системами та службами.

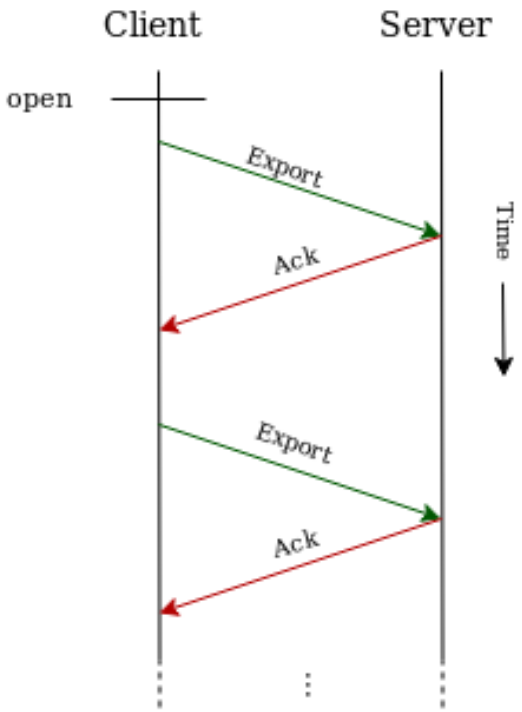

Рисунок 2.1 Операційні інфраструктури *OTLP*

OTLP/HTTP – це протокол, який використовує *HTTP* як транспортний механізм для передачі телеметричних даних.

Основні характеристики:

– *HTTP*-запит/відповідь : *OTLP/HTTP* використовує *HTTP*-запити та відповіді для передачі телеметричних даних.

– корисне навантаження *JSON* : корисне навантаження запиту/відповіді *HTTP* має формат *JSON*.

Обробка помилок : *OTLP/HTTP* забезпечує механізми обробки помилок для обробки помилок під час передачі даних.

OTLP/gRPC – це протокол, який використовує *gRPC* як транспортний механізм для передачі телеметричних даних.

Основні характеристики:

– запит/відповідь *gRPC* : *OTLP/gRPC* використовує запити та відповіді *gRPC* для передачі телеметричних даних.

– буфери протоколу : корисне навантаження запиту/відповіді *gRPC* має формат буфера протоколу.

– обробка помилок : *OTLP/gRPC* забезпечує механізми обробки помилок для обробки помилок під час передачі даних.

Структура запиту та відповіді:

OTLP визначає структуру запитів і відповідей для передачі телеметричних даних. Структура запиту

– заголовок: Заголовок запиту містить такі метадані, як ідентифікатор запиту, мітка часу та назва служби;

– корисне навантаження : корисне навантаження запиту включає телеметричні дані для передачі;

– помилка : помилка запиту містить інформацію про помилку, таку як код помилки та повідомлення.

Структура відповіді:

– заголовок: Заголовок відповіді містить такі метадані, як ідентифікатор відповіді, мітка часу та назва служби;

– корисне навантаження: корисне навантаження відповіді включає телеметричні дані, отримані від сервера;

– помилка: Помилка відповіді містить інформацію про помилку, таку як код помилки та повідомлення.

Буфери протоколу забезпечують ефективну серіалізацію телеметричних даних. Це досягається за рахунок:

1) буфери протоколів:

– буфери протоколів використовують двійковий формат для серіалізації телеметричних даних;

– буфери протоколу забезпечують компактне представлення телеметричних даних, зменшуючи розмір даних;

– буфери протоколу забезпечують швидку серіалізацію телеметричних даних, забезпечуючи ефективну передачу даних;

2) Сумісність:

– буфери протоколу забезпечують міжмовну підтримку, що дозволяє передавати телеметричні дані між різними мовами програмування;

– буфери протоколу забезпечують незалежність від платформи, дозволяючи передавати телеметричні дані між різними платформами.

OTLP дотримується моделей даних *OpenTelemetry* та вносить свій внесок у них. Моделі даних визначають структуру та формат телеметричних даних.

Дані телеметрії: моделі даних визначають структуру та формат даних телеметрії, включаючи показники, журнали та трасування.

Серіалізація даних: моделі даних визначають формат серіалізації телеметричних даних, включаючи буфери протоколів.

Передача даних: моделі даних визначають формат передачі телеметричних даних, включаючи *HTTP* і *gRPC*.

Специфікації *OTLP* розвиваються в рамках проекту *OpenTelemetry*. Це досягається за рахунок:

– специфікації *OTLP* покращено на основі відгуків і пропозицій спільноти;

– нові функції додано до специфікацій *OTLP* для покращення його функціональності;

– виправлено помилки, щоб забезпечити стабільність і надійність специфікацій *OTLP*.

Проект *OpenTelemetry* є спільним зусиллям для розробки та підтримки специфікацій *OTLP*. Проект співпрацює зі спільнотою для збору відгуків і пропозицій щодо специфікацій *OTLP*. Проект також містить дорожню карту для розробки та підтримки специфікацій *OTLP*.

Дотримуючись цих специфікацій, *OTLP* забезпечує ефективну та надійну передачу телеметричних даних між різними системами та службами.

Щоб збирати показники за допомогою *OTLP*, потрібно правильно налаштувати агентів і експортерів. Агенти несуть відповідальність за збір показників із вашої програми чи служби, тоді як експортери відповідають за надсилання зібраних показників до системи моніторингу. Щоб використовувати *OTLP* для збору показників, вам потрібно налаштувати агентів і експортерів на використання протоколу OTLP.

Що стосується прийому метрик, у вас є два основні варіанти: *Prometheus* і *API* моніторингу. *Prometheus* – це популярна система моніторингу, яка використовує підхід на основі витягування для збору показників, тоді як *Monitoring API* — це підхід на основі *push*, який надсилає показники в систему моніторингу. Обидва підходи мають свої сильні та слабкі сторони, і вибір між ними залежить від конкретного випадку використання.

Налаштування правильного *API* для ефективного моніторингу має вирішальне значення для збору та аналізу показників. Вибраний вами *API* визначатиме, як збираються, обробляються та зберігаються ваші показники. Якщо ви виберете неправильний *API*, у вас можуть виникнути проблеми з якістю даних, масштабованістю та продуктивністю. Тому важливо вибрати правильний *API* для конкретного випадку використання.

Після того, як ви зібрали свої показники за допомогою *OTLP*, ви можете запитувати та аналізувати їх у своїй системі моніторингу. Це передбачає використання запитів для отримання певних показників, агрегування даних і візуалізацію результатів. Аналізуючи свої показники, ви можете отримати уявлення про продуктивність своєї програми чи служби, виявити проблеми та прийняти рішення на основі даних.

Збираючи показники *OTLP*, слід пам’ятати про кілька операційних міркувань. До них відносяться відмінності в ціноутворенні, квотах і структурі показників. Ціноутворення стосується вартості збору та зберігання показників, тоді як квота стосується обмежень на обсяг даних, які можна зібрати. Відмінності

в структурі показників стосуються способу структурування та форматування показників. Розуміння цих операційних міркувань є важливим для забезпечення ефективного та економічно ефективного збору показників.

2.2 Збір телеметрії в екосистемі *Grafana*

Grafana Agent Flow можна налаштувати для збору даних, сумісних з *OpenTelemetry*, і пересилання їх до будь-якої кінцевої точки, сумісної з OpenTelemetry.

Перш ніж компоненти зможуть отримувати дані *OpenTelemetry*, ви повинні мати компонент, відповідальний за експорт даних *OpenTelemetry*. Компонент експортера *OpenTelemetry* відповідає за запис (експорт) даних *OpenTelemetry* до зовнішньої системи.

Щоб налаштувати *otelcol.exporter.otlp* компонент для експорту даних *OpenTelemetry* за допомогою *OTLP*, виконайте такі дії:

1. Додайте наступний *otelcol.exporter.otlp* компонент до файлу конфігурації:

```
otelcol.exporter.otlp "<EXPORTER_LABEL>" {
  client {
    url = "<HOST>:<PORT>"
  }
}
```

Код має наступні параметри:

– <*EXPORTER_LABEL*>: Мітка для компонента, наприклад default. Мітка, яку ви використовуєте, має бути унікальною для всіх *otelcol.exporter.otlp* компонентів в одному конфігураційному файлі;

– <*HOST*>: ім’я хоста або IP-адреса сервера для надсилання запитів *OTLP;*

– <*PORT*>: порт сервера для надсилання запитів *OTLP*.

2. Якщо ваш сервер потребує базової автентифікації, виконайте такі дії:

а) додайте наступний *otelcol.auth.basic* компонент до файлу конфігурації:

```
otelcol.auth.basic "<BASIC_AUTH_LABEL>" {
  username = "<USERNAME>"
  password = "<PASSWORD>"
}
```

Код має наступні параметри:

– <*BASIC_AUTH_LABEL*>: Мітка для компонента, наприклад *default*. Мітка, яку ви використовуєте, має бути унікальною для всіх *otelcol.auth.basic* компонентів в одному конфігураційному файлі;

– <*USERNAME*>: базове ім’я користувача для автентифікації;

– <*PASSWORD*>: базовий пароль автентифікації або ключ *API*.

б) додайте наступний рядок усередині *client* блоку вашого *otelcol.exporter.otlp* компонента:

```
auth = otelcol.auth.basic.<BASIC_AUTH_LABEL>.handler
```

<*BASIC_AUTH_LABEL*>: Мітка для *otelcol.auth.basic* компонента.

3. Якщо у вас є кілька серверів для експорту показників, створіть новий *otelcol.exporter.otlp* компонент для кожного додаткового сервера. *otelcol.exporter.otlp* надсилає дані за допомогою *OTLP* через *gRPC* (*HTTP*/2). Щоб надіслати на сервер за допомогою *HTTP*/1.1, виконайте попередні кроки, але натомість використовуйте компонент *otelcol.exporter.otlphttp*.

У наступному прикладі демонструється налаштування *otelcol.exporter.otlp* за допомогою автентифікації та компонента, який пересилає йому дані:

```
otelcol.exporter.otlp "default" {
  client {
    endpoint = "my-otlp-grpc-server:4317"
```

```
    auth     = otelcol.auth.basic.credentials.handler
  }
}

otelcol.auth.basic "credentials" {
  // Retrieve credentials using environment variables.
  username = env("BASIC_AUTH_USER")
  password = env("API_KEY")
}

otelcol.receiver.otlp "example" {
  grpc {
    endpoint = "127.0.0.1:4317"
  }

  http {
    endpoint = "127.0.0.1:4318"
  }

  output {
    metrics = [otelcol.exporter.otlp.default.input]
    logs    = [otelcol.exporter.otlp.default.input]
    traces  = [otelcol.exporter.otlp.default.input]
  }
}
```

Конфігурації *Grafana Agent Flow* не повинні надсилати дані *OpenTelemetry* безпосередньо експортеру для доставки. Замість цього дані зазвичай надсилаються до одного або кількох компонентів процесора, які виконують різні перетворення даних.

Забезпечення пакетної обробки даних є етапом готовності до роботи, щоб покращити стиснення даних і зменшити кількість вихідних мережевих запитів до зовнішніх систем.

Щоб налаштувати *otelcol.processor.batch* компонент, виконайте такі кроки:

1. Дотримуйтеся налаштувань експортера протоколу *OpenTelemetry* описаного вище, щоб забезпечити можливість запису отриманих даних у зовнішню систему.

2. Додайте наступний *otelcol.processor.batch* компонент у файл конфігурації:

```
otelcol.processor.batch "<PROCESSOR_LABEL>" {
  output {
    metrics = [otelcol.exporter.otlp.<EXPORTER_LABEL>.input]
    logs    = [otelcol.exporter.otlp.<EXPORTER_LABEL>.input]
    traces  = [otelcol.exporter.otlp.>EXPORTER_LABEL>.input]
  }
}
```

Замініть наступні параметри:

– <*PROCESSOR_LABEL*>: Мітка для компонента, наприклад default. Мітка, яку ви використовуєте, має бути унікальною для всіх otelcol.processor.batchкомпонентів в одному конфігураційному файлі;

– <*EXPORTER_LABEL*>: Мітка для наявного otelcol.exporter.otlp компонента.

- Щоб вимкнути один із типів телеметрії, установіть у блоці відповідний тип *outputy* порожньому списку, наприклад metrics = [].
- Щоб надіслати пакетні дані на інший процесор, замініть компоненти в *output* списку компонентами процесора, які потрібно використовувати.

У наступному прикладі показано налаштування послідовності *otelcol.processor* компонентів перед експортом.

```
otelcol.processor.memory_limiter "default" {
  check_interval = "1s"
  limit          = "1GiB"
  output {
    metrics = [otelcol.processor.batch.default.input]
    logs    = [otelcol.processor.batch.default.input]
    traces  = [otelcol.processor.batch.default.input]
  }
}
otelcol.processor.batch "default" {
  output {
    metrics = [otelcol.exporter.otlp.default.input]
    logs    = [otelcol.exporter.otlp.default.input]
    traces  = [otelcol.exporter.otlp.default.input]
  }
}
otelcol.exporter.otlp "default" {
  client {
    endpoint = "my-otlp-grpc-server:4317"
  }
}
```

Ви можете налаштувати *Grafana Agent Flow* на отримання метрик, журналів і трасування *OpenTelemetry*. Компонент приймача *OpenTelemetry* відповідає за отримання даних *OpenTelemetry* із зовнішньої системи.

У цьому прикладі будете використовуватися компонент *otelcol.receiver.otlp* для отримання даних *OpenTelemetry* через мережу за допомогою протоколу *OpenTelemetry*. Ви можете налаштувати компонент

приймача для пересилання отриманих даних іншим компонентам *Grafana Agent Flow*.

Зверніться до списку доступних компонентів, щоб отримати повний список *otelcol.receiver* компонентів, які можна використовувати для отримання даних, сумісних з *OpenTelemetry*.

Щоб налаштувати *otelcol.receiver.otlp* компонент для отримання даних *OTLP*, виконайте такі дії:

1. Дотримуйтеся налаштувань експортера протоколу *OpenTelemetry*, щоб забезпечити можливість запису отриманих даних у зовнішню систему.

2. Необов’язково: виконайте налаштування пакетної обробки, щоб покращити стиснення та зменшити загальну кількість мережевих запитів.

3. Додайте наступний otelcol.receiver.otlp компонент до файлу конфігурації.

```
otelcol.receiver.otlp "<LABEL>" {
  output {
    metrics = [<COMPONENT_INPUT_LIST>]
    logs    = [<COMPONENT_INPUT_LIST>]
    traces  = [<COMPONENT_INPUT_LIST>]
  }
}
```

Замініть наступне:

– *<LABEL>*: Мітка для компонента, наприклад *default*. Мітка, яку ви використовуєте, має бути унікальною для всіх *otelcol.receiver.otlp* компонентів в одному конфігураційному файлі;

– *<COMPONENT_INPUT_LIST>*: розділений комами список входів компонентів для пересилання отриманих даних. Наприклад, щоб надіслати дані до існуючого компонента пакетного процесора, використовуйте *otelcol.processor.batch.PROCESSOR_LABEL.input*. Щоб надіслати дані

безпосередньо до існуючого компонента експортера, використовуйте *otelcol.exporter.otlp.EXPORTER_LABEL.input*.

1. Щоб дозволити програмам надсилати дані *OTLP* через *gRPC* на порт 4317, додайте наступне до свого *otelcol.receiver.otlp* компонента.

```
grpc {
  endpoint = "<HOST>:4317"
}
```

Замініть <*HOST*>: Хост для прослуховування трафіку. Використовуйте вузьку адресу прослуховування, коли це можливо. Щоб прослуховувати всі мережеві інтерфейси, замініть <*HOST*>на 0.0.0.0.

2. Щоб дозволити програмам надсилати дані *OTLP* через *HTTP*/1.1 на порт 4318, додайте наступне до свого otelcol.receiver.otlpкомпонента.

```
http {
  endpoint = "<HOST>:4318"
}
```

Замініть <*HOST*>: хост для прослуховування трафіку. Використовуйте вузьку адресу прослуховування, коли це можливо. Щоб прослуховувати всі мережеві інтерфейси, замініть <*HOST*>на 0.0.0.0.

3. Щоб вимкнути один із типів телеметрії, установіть у блоці відповідний тип *outputy* порожньому списку, наприклад *metrics* = [].

У наступному прикладі демонструється налаштування *otelcol.receiver.otlp* та надсилання його експортеру:

```
otelcol.receiver.otlp "example" {
  grpc {
    endpoint = "127.0.0.1:4317"
  }
```

```
  http {
    endpoint = "127.0.0.1:4318"
  }
  output {
    metrics = [otelcol.processor.batch.example.input]
    logs    = [otelcol.processor.batch.example.input]
    traces  = [otelcol.processor.batch.example.input]
  }
}
otelcol.processor.batch "example" {
  output {
    metrics = [otelcol.exporter.otlp.default.input]
    logs    = [otelcol.exporter.otlp.default.input]
    traces  = [otelcol.exporter.otlp.default.input]
  }
}
otelcol.exporter.otlp "default" {
  client {
    endpoint = "my-otlp-grpc-server:4317"
  }
}
```

2.3 Візуалізація телеметрії

Однією з основних задач, що накладаються на систему моніторингу є візуалізація даних. В системі *Grafana* це забезпечується за допомогою спеціалізованих інформаційних панелей (*dashboards*).

Інформаційна панель *Grafana* складається з панелей, які відображають дані у вигляді красивих графіків, діаграм та інших візуалізацій. Ці панелі створюються за допомогою компонентів, які перетворюють необроблені дані з

джерела даних у візуалізації. Процес передбачає передачу даних через три шлюзи: плагін, запит і додаткове перетворення (рис. 2.2).

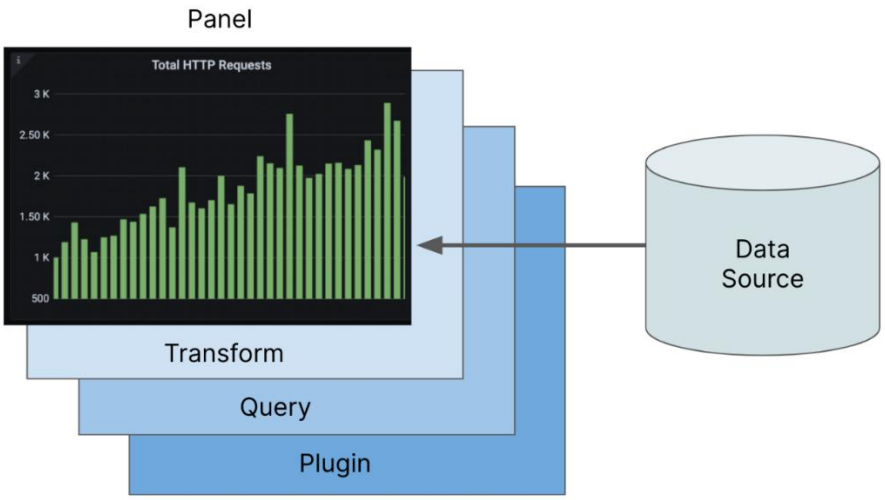

Рисунок 2.2 – Етапи відображення даних

Джерело даних відноситься до будь-якої сутності, яка складається з даних. Це може бути база даних *SQL*, *Grafana Loki*, *Grafana Mimir* або *API* на основі *JSON*. Це може бути навіть простий файл *CSV*.

Першим кроком у створенні візуалізації інформаційної панелі є вибір джерела даних, яке містить потрібні вам дані. Може бути важко зрозуміти відмінності між різними джерелами даних, оскільки кожен має власну структуру та потребує різних методів запиту. Однак на інформаційних панелях ви можете бачити різні джерела даних, візуалізовані в одному поданні, що полегшує загальне розуміння даних.

Плагін *Grafana* – це програмне забезпечення, яке додає нові можливості *Grafana*. Їх існує багато типів, але зараз ми розглянемо плагіни джерела даних. Завдання плагіна джерела даних *Grafana* полягає в тому, щоб прийняти запит, на який ви хочете отримати відповідь, отримати дані з джерела даних і узгодити відмінності між моделлю даних джерела даних і моделлю даних панелей інструментів *Grafana*. Для цього використовується уніфікована структура даних, яка називається кадром даних.

Дані, які надходять до плагіна з джерела даних, можуть мати різні формати (наприклад, *JSON*, рядки та стовпці або *CSV*), але коли вони залишають плагін і переміщуються через інші ворота до візуалізації, вони завжди знаходяться в кадри даних.

Наразі *Grafana* пропонує широкий вибір із 155 джерел даних, якими ви можете скористатися. Опції, які найчастіше використовуються, уже попередньо встановлені та доступні. Перш ніж досліджувати інші варіанти, знайдіть існуюче джерело даних, яке відповідає вашим вимогам. *Grafana* постійно оновлює список, але якщо ви не знайдете відповідного джерела даних, ви можете переглянути каталог плагінів або створити плагін.

Запити дозволяють скоротити всі ваші дані до певного набору даних, забезпечуючи більш керовану візуалізацію. Вони допоможуть відповісти на ваші запитання щодо системи та операційних процесів. Наприклад, компанія з інтернет-магазином може захотіти визначити кількість клієнтів, які додають продукти до своїх кошиків для покупок. Цього можна досягти за допомогою запиту, який агрегує показники доступу до служби кошика для покупок, показуючи кількість користувачів, які отримують доступ до служби за секунду.

Працюючи з джерелами даних, дуже важливо розуміти, що кожне з них має свою окрему мову запитів. Наприклад, джерела даних Prometheus використовують *PromQL*, тоді як *LogQL* використовується для журналів, а окремі бази даних використовують *SQL*. Запит є основою кожної візуалізації в *Grafana*, а приладна панель може використовувати низку мов запитів.

Редактор запитів (рис. 2.3), пов’язаний із джерелом даних *Prometheus*. Запит *node_cpu_seconds_total* написаний у *PromQL* і запитує лише одну метрику:

Якщо формат даних у візуалізації не відповідає вашим вимогам, ви можете застосувати перетворення, яке маніпулює даними, повернутими запитом. Можливо, вам не потрібно буде трансформувати дані, коли ви тільки починаєте роботу, але вони потужні та варті згадки.

Перетворення даних корисно в таких ситуаціях:

– ви хочете об’єднати два поля разом, наприклад, конкатенацію *Given Name*та *Family Name*в *Full Name* поле;

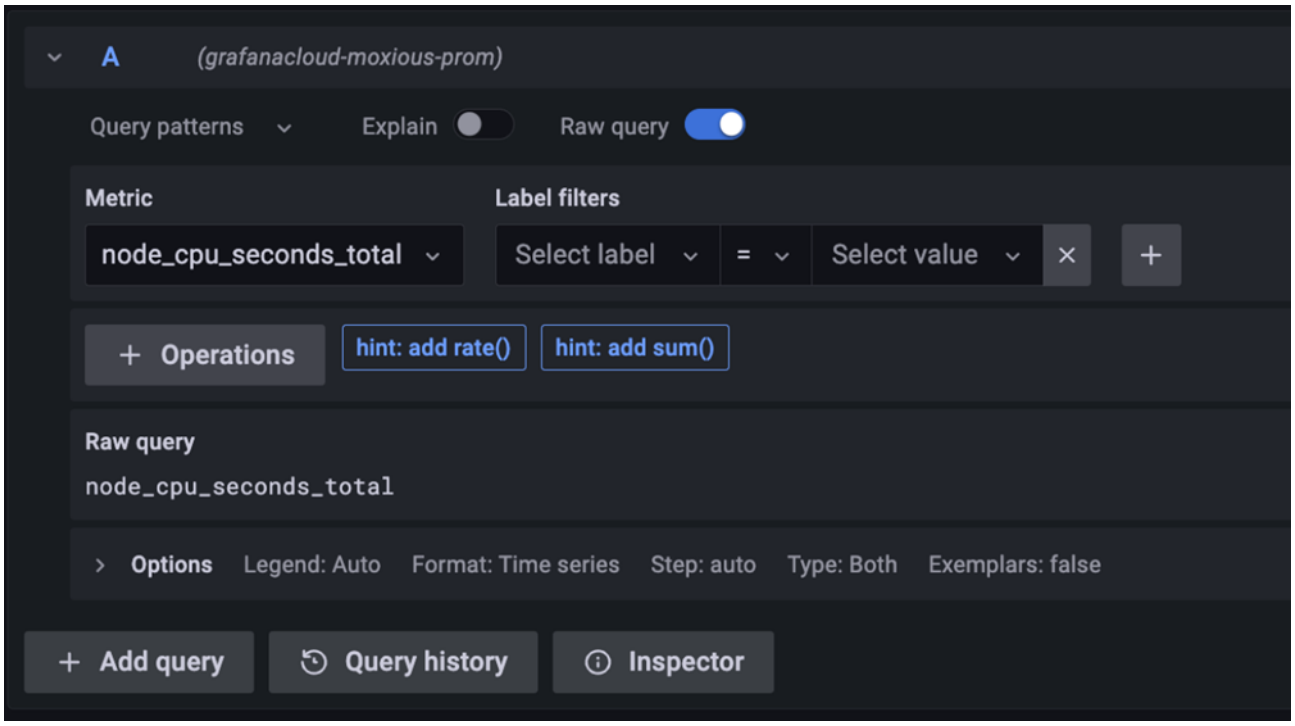

Рисунок 2.3 – Налаштування джерела Prometheus

– у вас є дані *CSV* (увесь текст), і ви хочете перетворити тип поля (наприклад, розібрати дату або число з рядка).

Ви хочете фільтрувати, об’єднувати, об’єднувати або виконувати інші *SQL*-подібні операції, які можуть не підтримуватися основним джерелом даних або мовою запитів.

Трансформації розташовані на вкладці «Трансформація» в діалоговому вікні редагування панелі. Виберіть потрібне перетворення та визначте перетворення. На наступному зображенні показано, що ви можете мати скільки завгодно перетворень, як і запитів. Наприклад, ви можете об’єднати низку перетворень, які вносять зміни до типу даних, фільтрують результати, упорядковують стовпці та сортують результат в одному конвеєрі даних. Щоразу, коли інформаційна панель оновлюється, трансформація застосовується до останніх даних із джерела даних.

Після отримання даних, їх запитів і трансформації вони переходять на панель, яка є останніми воротами на шляху до візуалізації *Grafana*. Панель – це контейнер, який відображає візуалізацію та надає різні елементи керування для керування нею. У конфігурації панелі ви вказуєте, як ви хочете бачити дані. Наприклад, ви використовуєте спадне меню у верхньому правому куті панелі, щоб вказати тип візуалізації, який ви хочете побачити, як-от гістограма, секторна діаграма або гістограма.

Параметри панелі дозволяють налаштувати багато аспектів візуалізації, і параметри відрізняються залежно від візуалізації, яку ви вибрали. Панелі також містять запити, які визначають дані, які панель візуалізує.

На рисунку 2.4 показано панель таблиці, яка редагується, параметри панелі, що показують запит унизу, і параметри панелі праворуч. На цьому зображенні ви можете побачити, як поєднуються джерело даних, плагін, запит і панель.

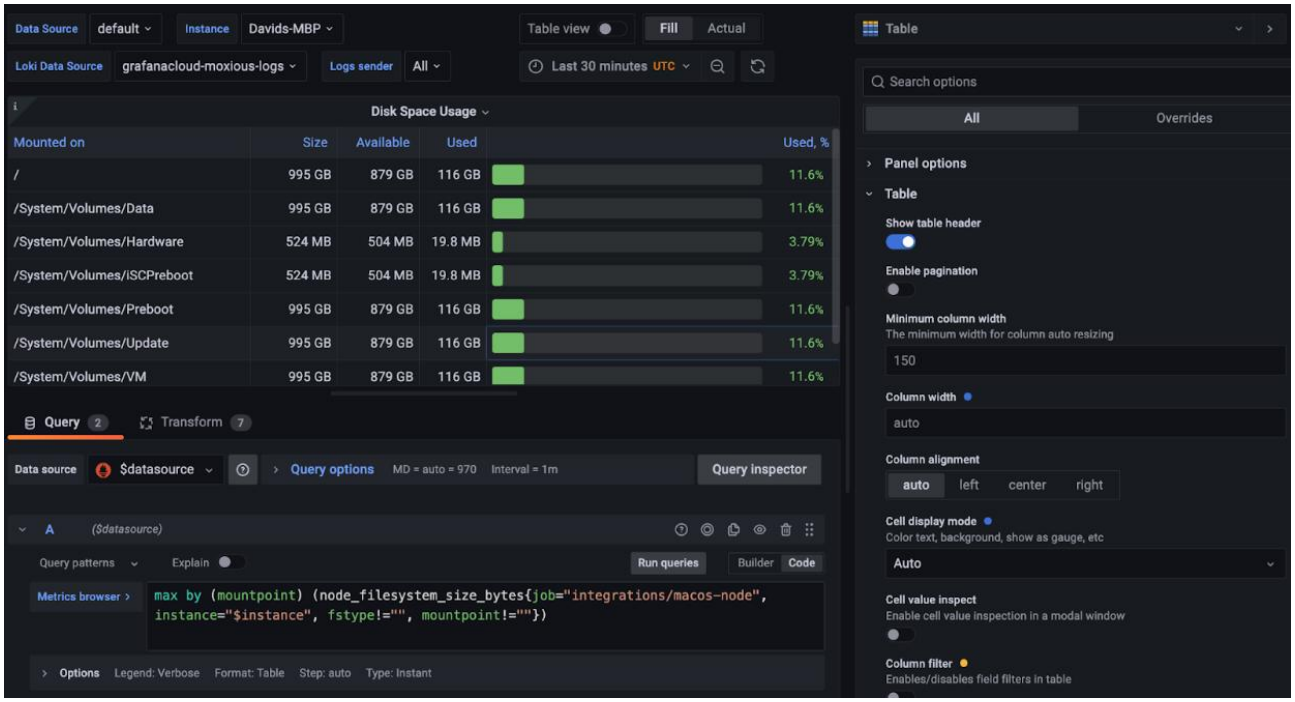

Рисунок 2.4 – Приклад настільної панелі

Вибір найкращої візуалізації залежить від даних і того, як ви хочете їх представити. Щоб побачити приклади інформаційних панелей в одному місці, які ви можете переглядати та перевіряти, зверніться до *Grafana Play*, де є демонстрації функцій і різноманітні приклади.

Висновки по розділу

Створення інформаційної панелі Grafana — це процес, який починається з визначення вимог до інформаційної панелі та визначення джерел даних, які підтримують ці вимоги. Ця компонентна архітектура є частиною того, що робить *Grafana* настільки потужним і загальним. Враховуючи плагін джерела даних і абстракцію кадру даних, будь-яке джерело даних, до якого ви можете отримати доступ, може працювати з тим самим загальним підходом.

3. РОЗРОБКА РЕКОМЕНДАЦІЙ ЗАХИСТУ

3.1 Побудова моделі дослідження

В якості моделі дослідження обрано віртуальну машину з операційною системою *Ubuntu*, параметри якої зображено на рисунку 3.1.

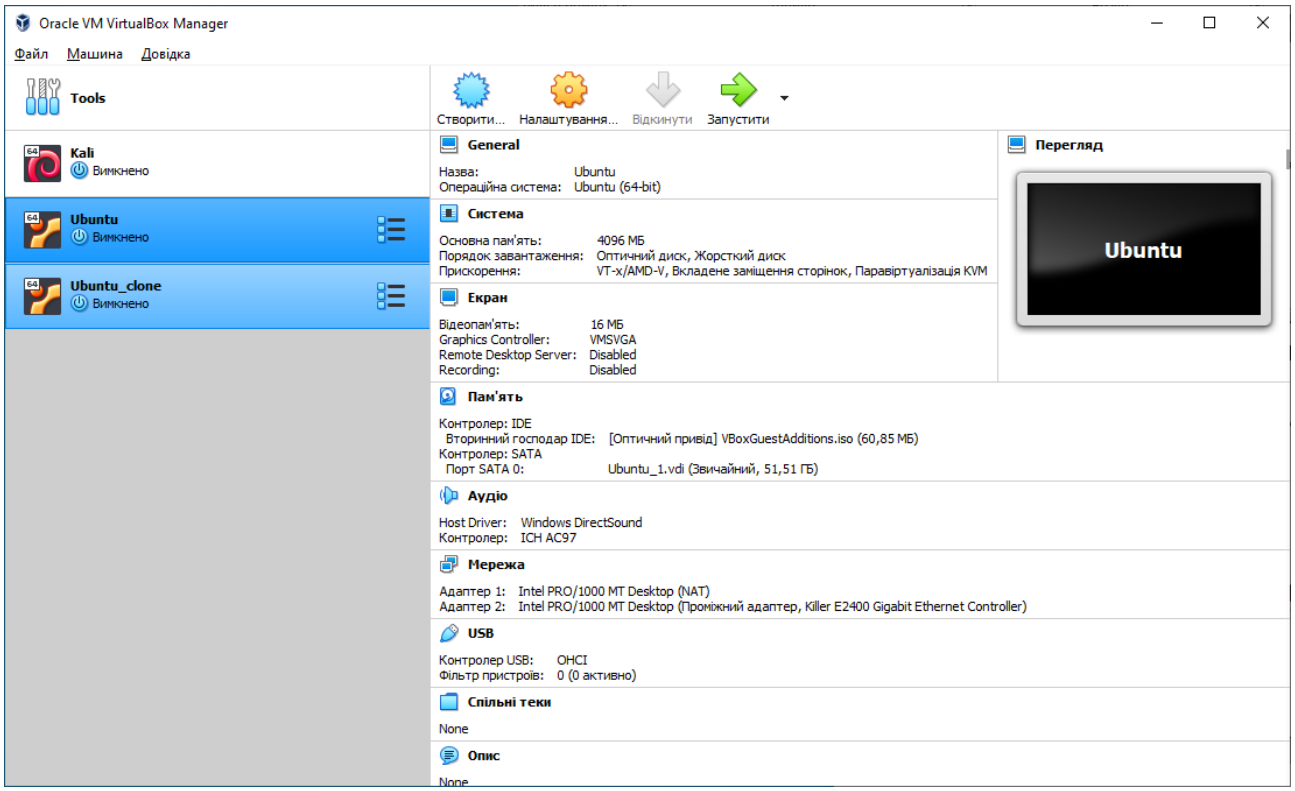

Рисунок 3.1 – Параметри віртуальної машини

Перед початком роботи створюємо файли конфігурації для системи *docker* та *Grafana* мовою*YAML* : *docker-compose.yaml*, *prometheus.yml* та *blackbox.yml*.

docker-compose.yaml надає опис налаштувань серверу *Grafana*, *Prometheus* та модулю *blackbox_exporter*:

```
services:
  prometheus_server:
    image: prom/prometheus:latest
```

```
    command: "--web.enable-admin-api --storage.tsdb.retention.time=1y --config.file=/etc/prometheus/prometheus.yml"
    volumes:
      - type: volume
        source: prometheus_data
        target: /prometheus
      - /opt/monitoring/config/prometheus.yml:/etc/prometheus/prometheus.yml
    ports:
      - 9090:9090
    privileged: true
    restart: unless-stopped
    networks:
      - monitoring
  grafana_server:
    image: grafana/grafana:latest
    volumes:
      - type: volume
        source: grafana_configuration
        target: /etc/grafana
      - type: volume
        source: grafana_data
        target: /var/lib/grafana
    ports:
      - 3000:3000
    privileged: true
    depends_on:
      - prometheus_server
      - loki-query-frontend
    restart: unless-stopped
    networks:
```

```
      - monitoring
  blackbox_exporter:
    image: quay.io/prometheus/blackbox-exporter:latest
    volumes:
      - /opt/monitoring/config/blackbox_exporter.yml:/config/blackbox.yml:ro
    ports:
      - 9115:9115
    command: --config.file=/config/blackbox.yml
    cap_add:
      - CAP_NET_RAW
    restart: unless-stopped
    networks:
      - monitoring
networks:
  monitoring:
    driver: bridge
    ipam:
      driver: default
      config:
      - subnet:  172.24.0.0/16
volumes:
  prometheus_data:
  grafana_configuration:
  grafana_data:
```

prometheus.yaml імпортує основні параметри середовища *Prometheus*:

```
global:
  scrape_interval:     15s # Set the scrape interval to every 15 seconds. Default is every 1 minute.
```

```
    evaluation_interval: 15s # Evaluate rules every 15 seconds. The default is every
1 minute.
    alerting:
     alertmanagers:
     - static_configs:
       - targets:
    rule_files:
    scrape_configs:
     - job_name: 'prometheus'
       static_configs:
       - targets: ['localhost:9090']
     - job_name: node
       static_configs:
       - targets:
         - 'node.monitoring.local:9100'
     - job_name: 'blackbox'
       metrics_path: /probe
       params:
         module: [icmp_prober]
       static_configs:
         - targets:
           - google.com
           - www.youtube.com
       relabel_configs:
         - source_labels: [__address__]
           target_label: __param_target
         - source_labels: [__param_target]
           target_label: instance
         - target_label: __address__
```

replacement: blackbox_exporter:9115 # The blackbox exporter's real hostname:port.

- job_name: 'blackbox_exporter' # collect blackbox exporter's operational metrics.

static_configs:

- targets: ['blackbox_exporter:9115']

blackbox.yml імпортує налаштування модулю *blackbox:*

modules:

icmp_prober:

prober: icmp

timeout: 5s

icmp:

preferred_ip_protocol: "ip4"

Переміщуємо налаштування в системну папку *opt*. Оскільки папка є системною то перміщення відбувається через консоль з правами суперадміністратора. Для цього виконуємо команди:

root@user-VirtualBox:/opt# mkdir -p monitoring/config

root@user-VirtualBox:/opt# cp /home/user/docker-compose.yaml /opt/monitoring/

root@user-VirtualBox:/opt# cp -R /home/user/config /opt/monitoring/

Далі встановлюємо систему для підтримки контейнеризації *docker*:

root@user-VirtualBox:/opt/monitoring# apt install docker.io

Для коректної роботи системи *Grafana* встановлюємо системку роботи з ключами в *docker* для цього необхідно виконати ряд інструкцій (рис. 3.2):

sudo apt-get update

sudo apt-get install ca-certificates curl

sudo install -m 0755 -d /etc/apt/keyrings

sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc

sudo chmod a+r /etc/apt/keyrings/docker.asc

 # Add the repository to Apt sources:

*echo *

*"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu *

*$(. /etc/os-release && echo "$VERSION_CODENAME") stable" | *

sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt-get update

sudo apt-get update

sudo apt-get install docker-compose-plugin

sudo /opt/monitoring# docker compose up -d

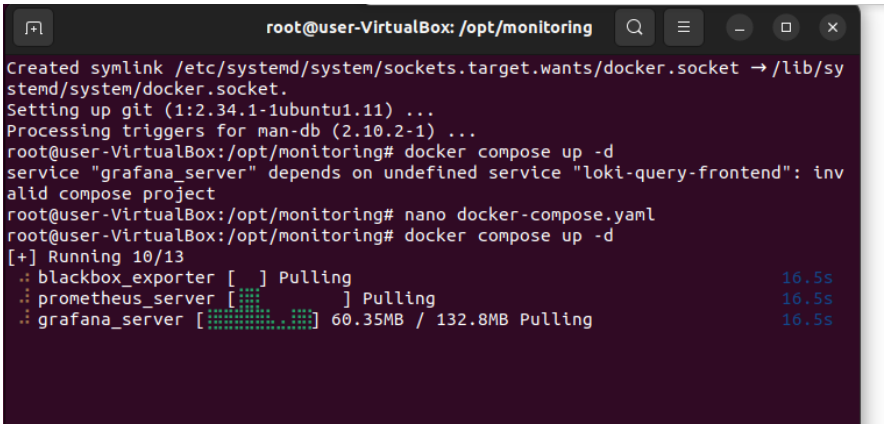

Рисунок 3.2 – Закінчення встановлення та налаштування системи

3.2 Налаштування системи

Для запуску системи *Grafana* в браузері вводимо в адресну строку http:/localhost:3000 (рис. 3.3). Номер порту вказано в файлі *docker-compose.yaml*. При першому заході логін – *admin* та пароль – *admin.*

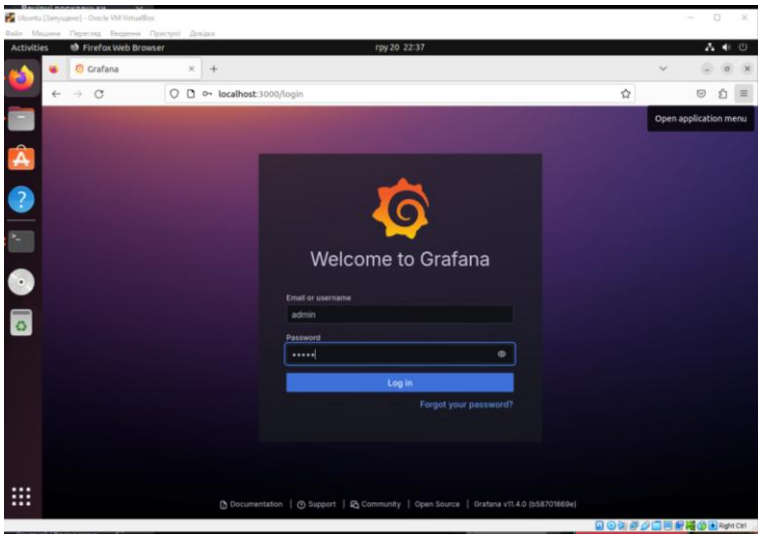

Рисунок 3.3 – Перший вхід в систему

Після першої авторизації система пропує змінити пароль для користувача *admin* (рис. 3.4).

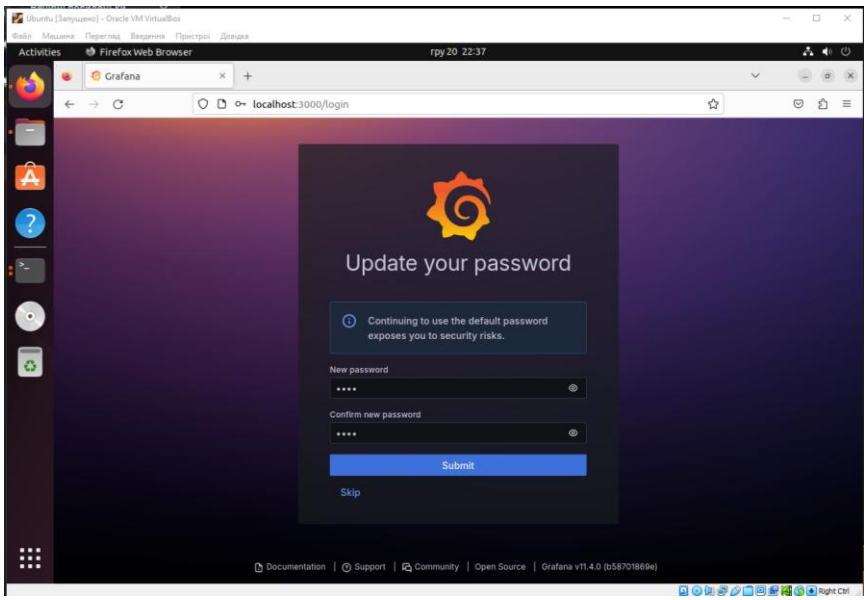

Рисунок 3.4 – Зміна стартового паролю

Після виконання стартових налаштувань користувача необхідно обрати джерело даних для в *Grafana* нашому випадку це – *Prometeus* (рис. 3.5).

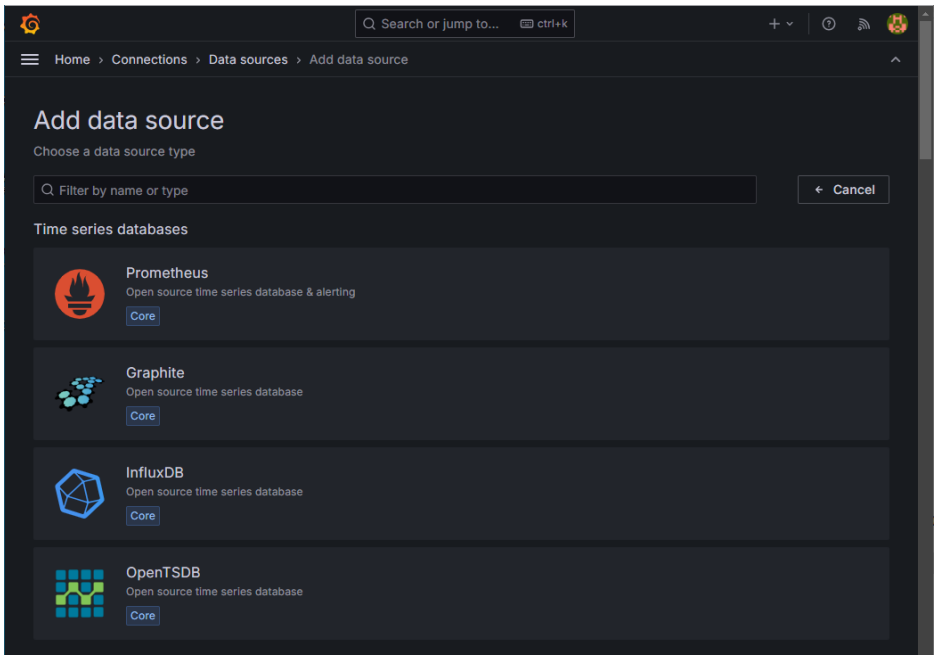

Рисунок 3.5 – Джерела даних

Обравши джерела даних можна налаштувати назву, серевер, та інші параметри (рис. 3.6).

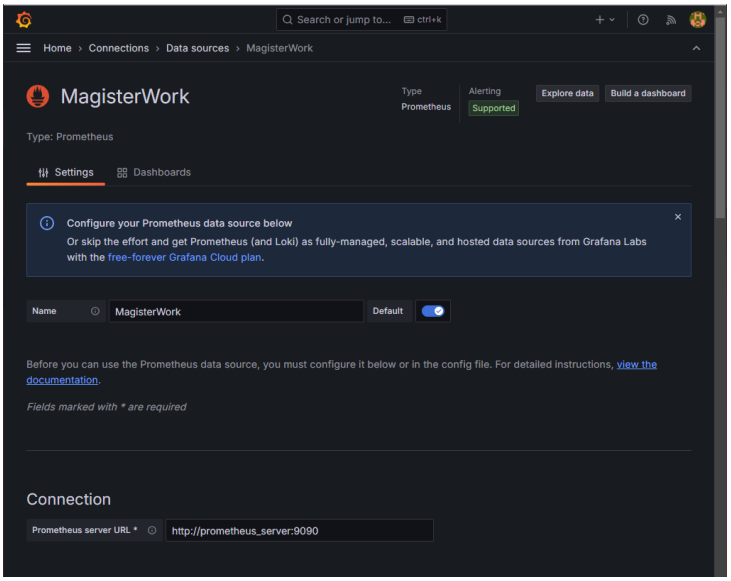

Рисунок 3.6 – Вікно налаштувань даних

Для перегляду параметрів які зараз надаються в систему необхідно перейти в меню *Metrics-> Select metrics* (рис. 3.7).

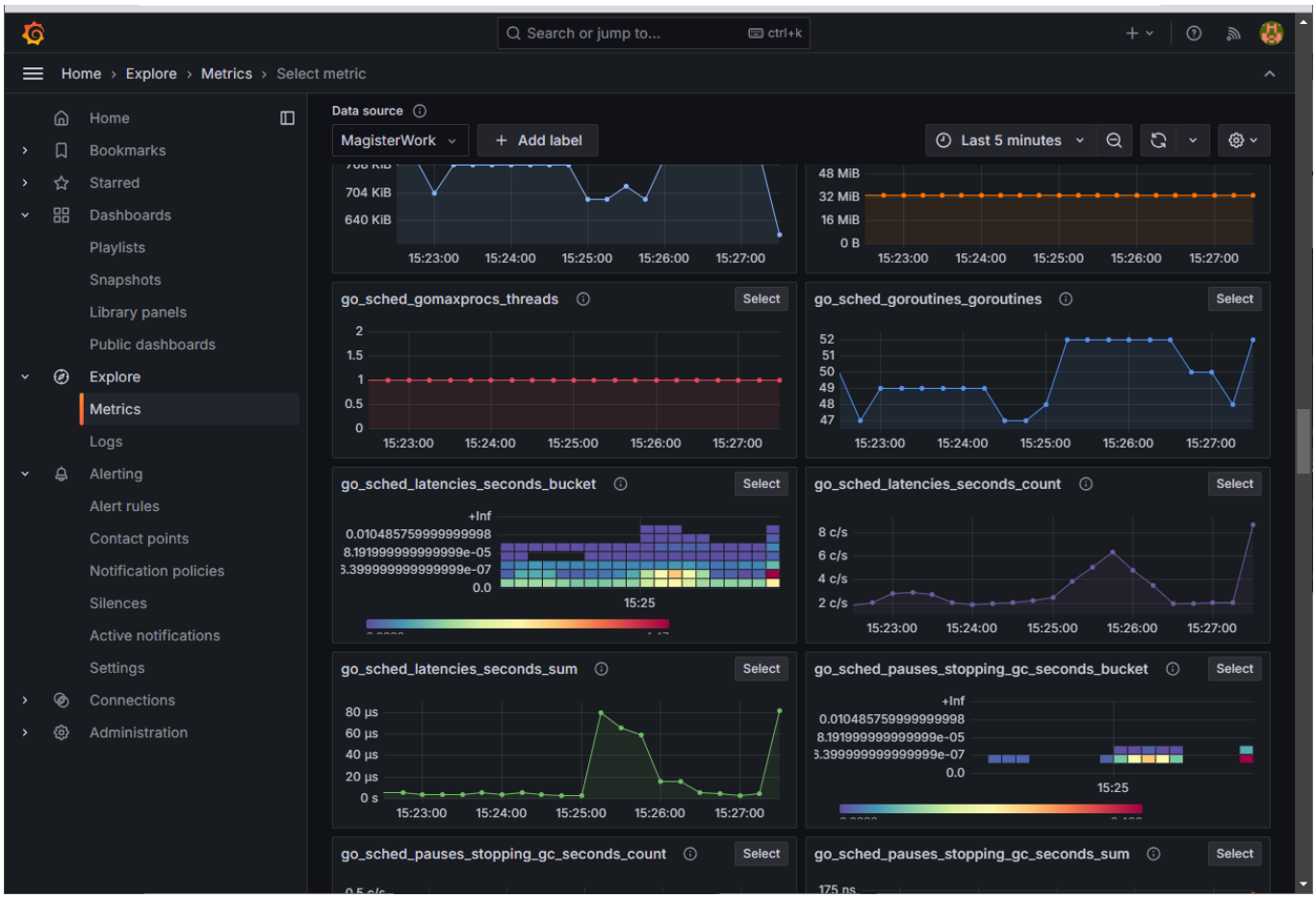

Рисунок 3.7 – Перегляд данних для моніторингу

Перегляд одночасно всих вимірювальних параметрів є досить незручним. Тому в системі *Grafana* є спеціалізовані механізми для відображення лише тих функцій, що вам потрібні, а саме – *Dashboards*. В них ви можете обрати метрики та налаштувати їх в ручному режимі за допомогою спеуціального конструктора (рис. 3.7) або задати параметри в режимі *JSON*.

Для моніторингу якості зв’язку Інтернет використовуємо стандарті налаштування *ICMP -BLACKBOX* в форматі *JSON* (Додаток А), що були попередньо налаштовані в [35].

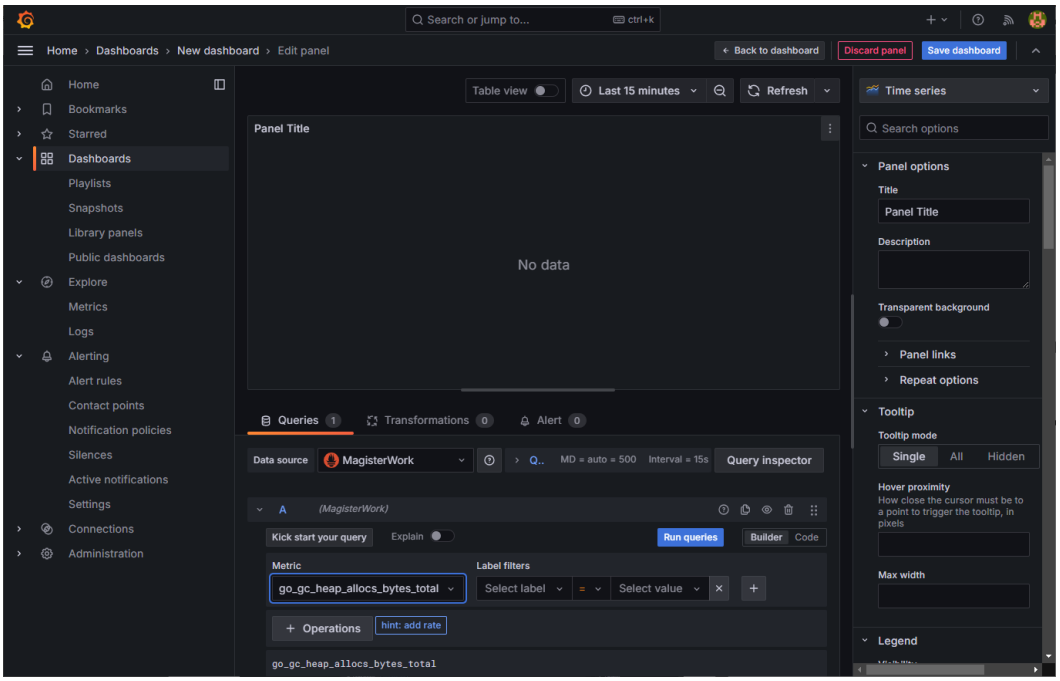

Рисунок 3.7 – Вікно налаштування візуалізації

Отримане вікно налаштувань *ICMP -BLACKBOX* (рис. 3.8) показує що зв'язок з системою *google.com* є значно кращим ніж з *yotube.com* (джерела налаштовані в *prometheus.yml*).

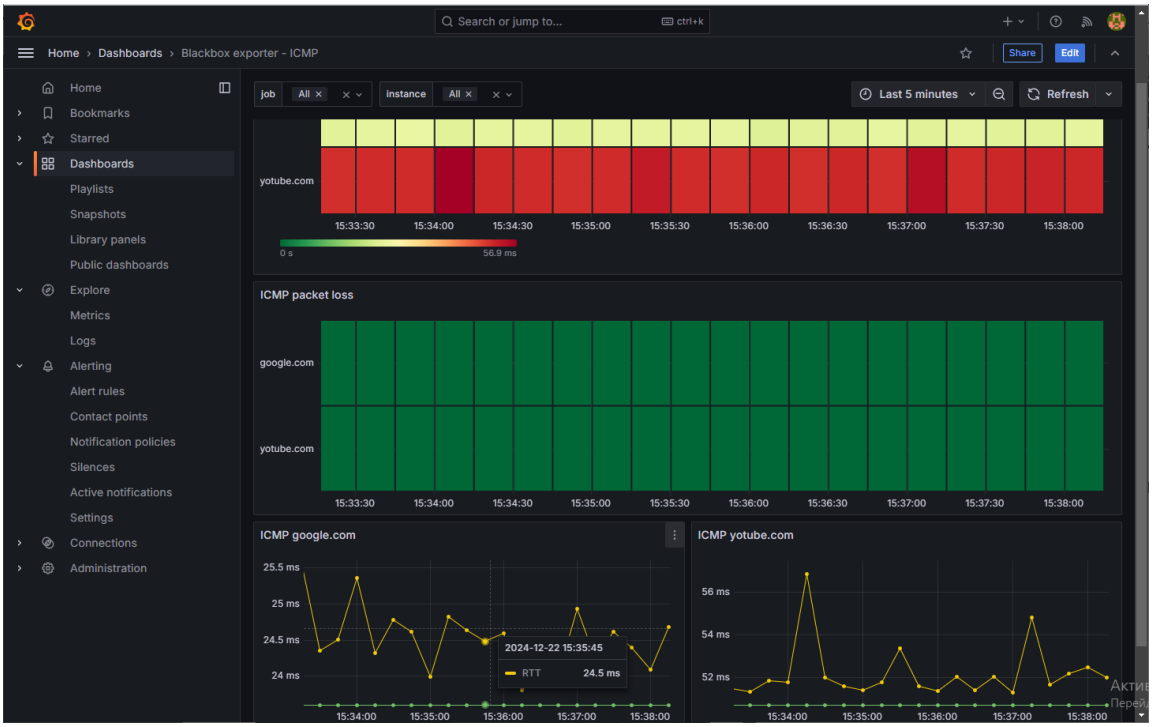

Рисунок 3.8 – *Dashboards ICMP -BLACKBOX*

Висновки по розділу

Було розгорнуто систему *Grafana* для діагностики мережі віртуального комп’ютера на базі операційної системи Ubuntu. Діагностика проводилась для визначення якості з’єднання з мережею Інтернет. Для цього використовувалися моніторинг зв’язку з сайтами *google.com* та *yotube.com.*

ВИСНОВОК

В ході кваліфікаційної роботи підтверджено необхідність системи соніторингу навіть, якщо не виникає загроза кібератак. Було проаналізовано джерела інформації для різних систем моніторингу та описано системи візуалізації даних.

В роботі для систем моніторингу пропонується використання Grafana, як система відображення даних та *Prometheus,* як система збору даних. Grafana є безкоштовним програмним засобом, тому її використання не завдасть значних фінансових затрат для фірми де вона буде використовуватися, а також буде корисною в навчальному процесі, як одна з можливих *SIEM* систем.

Для демонстрації роботи було розгорнуто систему *Grafana* на віртуальній машині з операційною системою *Ubuntu* на налаштовано систему відображення якості зв’язку.

ПЕРЕЛІК ПОСИЛАНЬ

1. Андрощук Г.О. Кібербезпека: тенденції в світі та Україні Кібербезпека та інтелектуальна власність: проблеми правового забезпечення: матеріали Міжнародної науково-практичної конференції. – К.: Вид-во «Політехніка», 2017. – С. 30-36.

2. Левчук А.С. Огляд програмних засобів для аналізу мережевого трафіку Інформаційні технології в освіті та науці: зб. наук. праць. – Мелітополь: Вид-во МДПУ ім. Б. Хмельницького, 2015. Вип. 7.С. 93-97.

3. Жилін А. В., Шаповал О. М., Успенський О. А. Технології захисту інформації в інформаційно-телекомунікаційних системах: навч. посібник. Київ : КПІ ім. Ігоря Сікорського, Вид-во «Політехніка», 2021. – 213 с.

4. Аналізатори. *URL: https://thk.kz/index.php/informatsiya/stati/318-20-besplatnykh-programm-administratora*.

5. 11 найкращих нюхачів *WiFi* - бездротові нюхачі пакетів у 2021 році. *URL: https://uk.myservername.com/11-best-wifi-sniffers-wireless-packet-sniffers-2021.*

6. *Khokhar, Varsha & Khan, Shehnaz & Muppuri, Priyanka & Chaudhary, Prachi. (2014). Sniflyzer: A Network Sniffer. Open Journal of Information Security and Applications. 2014. 1-8. 10.15764/ISA.2014.02001.*

7. *MB. Ezerski, SNMP completes the TCPIIP solution, Networking Management, 1991.*

8. *T. Drevers, R. van de Meent, and A. Pras, "Prototyping Web Services based Network Monitoring," in Proc. 10th Open European Summer School (EUNICE 2004) and IFIP WG 6.3 Workshop, Tampere, June 2004, pp. 135–142.*

9. *J. V. Hansen. Audit considerations in distributed processing systems. Communi-cations of the ACM, 26(8):562 – 569, August 1983.*

10. *B. Malin and L. Sweeney. How (Not) to Protect Genomic Data Privacy in a Dis-tributed Network: Using Trail Re-identification to Evaluate and Design*

AnonymityProtection Systems. Journal of Biomedical Informatics, 37(3):179–192, 2004.

11. *G. Spanoudakis and K. Mahbub. Non intrusive monitoring of service based sys-tems. International Journal of Cooperative Information Systems, 15(3):325–358,2006.*

12. *Matt Kafami What are Network Device Logs? URL: https://www.cbtnuggets.com/blog/technology/networking/what-are-network-device-logs.*

13. *NDT (Network Diagnostic Tool) URL: https://www.measurementlab.net/tests/ndt.*

14. *Dugan, J., Elliott, S., Mah, B. A., Poskanzer, J., & Prabhu, K. iPerf - The ultimate speed test tool for TCP, UDP and SCTPTest the limits of your network Internet neutrality test. URL: https://iperf.fr/*

15. *Solarwinds Worldwide, LLC. (2019). Network Diagnostic Tool - Performance Diagnostics Software. Retrieved from diagnostics-tool URL: https://www.solarwinds.com/network-performance-monitor/use-cases/network-.*

16. *Zabbix LLC. Network Monitoring. URL: https://www.zabbix.com /network_monitoring #why_zabbix .*

17. *Adam kucera, Petr Glos, and Tomas Pitner, Fault detection in building management system networks, in IFAC Proceedings, 2013,pp. 416–421.*

18. *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.*

19. Thomas Davis, David Skinnes, Software monitoring using Nagios, Cacti and Prism, in CUG Proceedings, 2005, pp. 1–5.

20. *Nagios XI Documentation URL: http://www.nagios.com/products/nagiosxi*

21. *M. Badger, Zenoss Core Network and System Monitoring. Packt Publishing Ltd., 2008*

22. *A. Clemm, Network management fundamentals, Indianapolis,USA, Cisco Press, 2007.*

23. *Katonová, Erika & Džubák, J. & Feciľak, P.. (2023). Automated Monitoring of Network Infrastructures Based on the Zabbix Solution. 283-288.*

24. *Zabbix documentation. URL: https://www.zabbix.com/documentation/current/start.*

25. *Zabbix appliance URL: https://www.zabbix.com/documentation/current/manual/appliance.*

26. *Service monitoring URL: https://www.zabbix.com/documentation/current/ manual/it_services.*

27. *Edupuganti, Geetha & Chetana, D. (2023). Using GitHub and Grafana Tools: Data Visualization (DATA VIZ) in Big Data. 10.1007/978-981-19-7892-0_38.*

28. *What is Grafana? URL: https://medium.com/@MetricFire/what-is-grafana-8de44d241765*

29. *How Grafana Works. URL: https://kodekloud.com/blog/how-grafana-works/*

30. *Understanding OpenTelemetry Protocol (OTLP) Specifications and Metrics URL: https://openobserve.ai/resources/otlp-specifications-metrics*

31. *OpenTelemetry Protocol URL: https://opentelemetry.io/docs/specs/otel/protocol/*

32. *Install Grafana on Debian or Ubuntu URL: https://grafana.com/docs/grafana/latest /setup-grafana/installation/debian/*

33. *How to Install Prometheus on Ubuntu 22.04 URL: https://www.cherryservers.com/blog/install-prometheus-ubuntu*

34. *Grafana dashboards overview. URL: https://grafana.com/docs/grafana/latest/fundamentals/dashboards-overview/*

35. *Dashboards URL: https://grafana.com/docs/grafana/latest/dashboards/*

36. Blackbox exporter - ICMP URL: https://github.com/prometheus/blackbox_exporter

ДОДАТОК А

Стандартні налаштування *Dashboards* для перевірки якості з’єднання з зовнішніми адресами.


```
{
  "annotations": {
    "list": [
      {
        "builtIn": 1,
        "datasource": {
          "type": "grafana",
          "uid": "-- Grafana --"
        },
        "enable": true,
        "hide": true,
        "iconColor": "rgba(0, 211, 255, 1)",
        "name": "Annotations & Alerts",
        "type": "dashboard"
      }
    ]
  },
  "description": "Blackbox exporter - ICMP https://github.com/prometheus/blackbox_exporter",
  "editable": true,
  "fiscalYearStartMonth": 0,
  "graphTooltip": 0,
  "id": 1,
  "links": [],
```

```
"panels": [
 {
  "datasource": {
   "type": "prometheus",
   "uid": "de7itdzfl8y68d"
  },
  "fieldConfig": {
   "defaults": {
    "custom": {
     "hideFrom": {
      "legend": false,
      "tooltip": false,
      "viz": false
     },
     "scaleDistribution": {
      "type": "linear"
     }
    }
   },
   "overrides": []
  },
  "gridPos": {
   "h": 8,
   "w": 24,
   "x": 0,
   "y": 0
  },
  "id": 1,
  "options": {
   "calculate": false,
```

"cellGap": 2,

"cellValues": {

"unit": "s"

},

"color": {

"exponent": 0.5,

"fill": "dark-orange",

"min": 0,

"mode": "scheme",

"reverse": false,

"scale": "exponential",

"scheme": "RdYlGn",

"steps": 128

},

"exemplars": {

"color": "rgba(255,0,255,0.7)"

},

"filterValues": {

"le": 1e-9

},

"legend": {

"show": true

},

"rowsFrame": {

"layout": "unknown"

},

"tooltip": {

"maxHeight": 600,

"mode": "single",

"showColorScale": false,

```
        "yHistogram": true
      },
      "yAxis": {
        "axisPlacement": "left",
        "reverse": true,
        "unit": "percentunit"
      }
    },
    "pluginVersion": "11.4.0",
    "targets": [
      {
        "datasource": {
          "type": "prometheus",
          "uid": "de7itdzfl8y68d"
        },
        "editorMode": "code",
        "expr":
"sum(probe_icmp_duration_seconds{phase=\"rtt\",job=~\"$job\",instance=~\"$instance\"}) by (instance)",
        "instant": false,
        "legendFormat": "{{instance}}",
        "range": true,
        "refId": "A"
      }
    ],
    "title": "ICMP RTT",
    "type": "heatmap"
  },
  {
    "datasource": {
```

```
  "type": "prometheus",
  "uid": "de7itdzfl8y68d"
},
"fieldConfig": {
  "defaults": {
    "custom": {
      "hideFrom": {
        "legend": false,
        "tooltip": false,
        "viz": false
      },
      "scaleDistribution": {
        "type": "linear"
      }
    }
  },
  "overrides": []
},
"gridPos": {
  "h": 8,
  "w": 24,
  "x": 0,
  "y": 8
},
"id": 2,
"options": {
  "calculate": false,
  "cellGap": 2,
  "color": {
    "exponent": 0.5,
```

```
  "fill": "dark-orange",
  "mode": "scheme",
  "reverse": false,
  "scale": "exponential",
  "scheme": "RdYlGn",
  "steps": 2
},
"exemplars": {
  "color": "rgba(255,0,255,0.7)"
},
"filterValues": {
  "le": -1
},
"legend": {
  "show": false
},
"rowsFrame": {
  "layout": "unknown"
},
"tooltip": {
  "maxHeight": 600,
  "mode": "single",
  "showColorScale": false,
  "yHistogram": true
},
"yAxis": {
  "axisPlacement": "left",
  "reverse": true,
  "unit": "percentunit"
}
```

```
  },
  "pluginVersion": "11.4.0",
  "targets": [
   {
    "datasource": {
     "type": "prometheus",
     "uid": "de7itdzfl8y68d"
    },
    "editorMode": "code",
    "expr": "1-probe_success{job=~\"$job\",instance=~\"$instance\"}",
    "instant": false,
    "legendFormat": "{{instance}}",
    "range": true,
    "refId": "A"
   }
  ],
  "title": "ICMP packet loss",
  "type": "heatmap"
 },
 {
  "datasource": {
   "type": "prometheus",
   "uid": "de7itdzfl8y68d"
  },
  "fieldConfig": {
   "defaults": {
    "color": {
     "mode": "palette-classic"
    },
    "custom": {
```

"axisBorderShow": false,

"axisCenteredZero": false,

"axisColorMode": "text",

"axisLabel": "",

"axisPlacement": "auto",

"barAlignment": 0,

"drawStyle": "line",

"fillOpacity": 0,

"gradientMode": "none",

"hideFrom": {

"legend": false,

"tooltip": false,

"viz": false

},

"insertNulls": false,

"lineInterpolation": "linear",

"lineStyle": {

"fill": "solid"

},

"lineWidth": 1,

"pointSize": 5,

"scaleDistribution": {

"type": "linear"

},

"showPoints": "auto",

"spanNulls": false,

"stacking": {

"group": "A",

"mode": "none"

},

```
    "thresholdsStyle": {
      "mode": "off"
    }
  },
  "fieldMinMax": false,
  "mappings": [],
  "thresholds": {
    "mode": "absolute",
    "steps": [
      {
        "color": "green",
        "value": null
      },
      {
        "color": "red",
        "value": 80
      }
    ]
  },
  "unit": "s"
},
"overrides": [
  {
    "matcher": {
      "id": "byName",
      "options": "packet loss"
    },
    "properties": [
      {
        "id": "custom.axisPlacement",
```

```
          "value": "hidden"
        },
        {
          "id": "unit",
          "value": "percentunit"
        },
        {
          "id": "custom.axisSoftMax",
          "value": 1
        }
      ]
    }
  ]
},
"gridPos": {
  "h": 9,
  "w": 24,
  "x": 0,
  "y": 16
},
"id": 3,
"maxPerRow": 2,
"options": {
  "legend": {
    "calcs": [
      "min",
      "max",
      "mean",
      "last"
    ],
```

```
      "displayMode": "table",
      "placement": "bottom",
      "showLegend": true
     },
     "tooltip": {
      "maxHeight": 600,
      "mode": "single",
      "sort": "none"
     }
    },
    "repeat": "instance",
    "repeatDirection": "h",
    "targets": [
     {
      "datasource": {
       "type": "prometheus",
       "uid": "de7itdzfl8y68d"
      },
      "editorMode": "code",
      "exemplar": false,
      "expr": "1-avg_over_time(probe_success{job=~\"$job\",instance=~\"$instance\"}[$__interval])",
      "hide": false,
      "instant": false,
      "legendFormat": "packet loss",
      "range": true,
      "refId": "A"
     },
     {
```

```
            "datasource": {
              "type": "prometheus",
              "uid": "de7itdzfl8y68d"
            },
            "editorMode": "code",
            "expr":
"probe_icmp_duration_seconds{phase=\"rtt\",job=~\"$job\",instance=~\"$instance\
"} > 0",
            "hide": false,
            "instant": false,
            "legendFormat": "RTT",
            "range": true,
            "refId": "B"
          }
        ],
        "title": "ICMP $instance",
        "type": "timeseries"
      }
    ],
    "preload": false,
    "refresh": "",
    "schemaVersion": 40,
    "tags": [],
    "templating": {
      "list": [
        {
          "current": {
            "text": [
              "All"
            ],
```

```
      "value": [
        "$__all"
      ]
    },
    "datasource": {
      "type": "prometheus",
      "uid": "de7itdzfl8y68d"
    },
    "definition": "label_values(probe_icmp_duration_seconds,job)",
    "includeAll": true,
    "label": "job",
    "multi": true,
    "name": "job",
    "options": [],
    "query": {
      "qryType": 1,
      "query": "label_values(probe_icmp_duration_seconds,job)",
      "refId": "PrometheusVariableQueryEditor-VariableQuery"
    },
    "refresh": 1,
    "regex": "",
    "type": "query"
  },
  {
    "current": {
      "text": [
        "All"
      ],
      "value": [
        "$__all"
```

```
      ]
    },
    "datasource": {
      "type": "prometheus",
      "uid": "de7itdzfl8y68d"
    },
    "definition":
"label_values(probe_icmp_duration_seconds{job=~\"$job\"},instance)",
    "includeAll": true,
    "label": "instance",
    "multi": true,
    "name": "instance",
    "options": [],
    "query": {
      "qryType": 1,
      "query":
"label_values(probe_icmp_duration_seconds{job=~\"$job\"},instance)",
      "refId": "PrometheusVariableQueryEditor-VariableQuery"
    },
    "refresh": 1,
    "regex": "",
    "type": "query"
   }
  ]
 },
 "time": {
  "from": "now-5m",
  "to": "now"
 },
 "timepicker": {},
```

```
  "timezone": "",
  "title": "Blackbox exporter - ICMP",
  "uid": "fcb1bdc8-fa9d-4bfd-b725-003b174473ad",
  "version": 1,
  "weekStart": ""
}
```