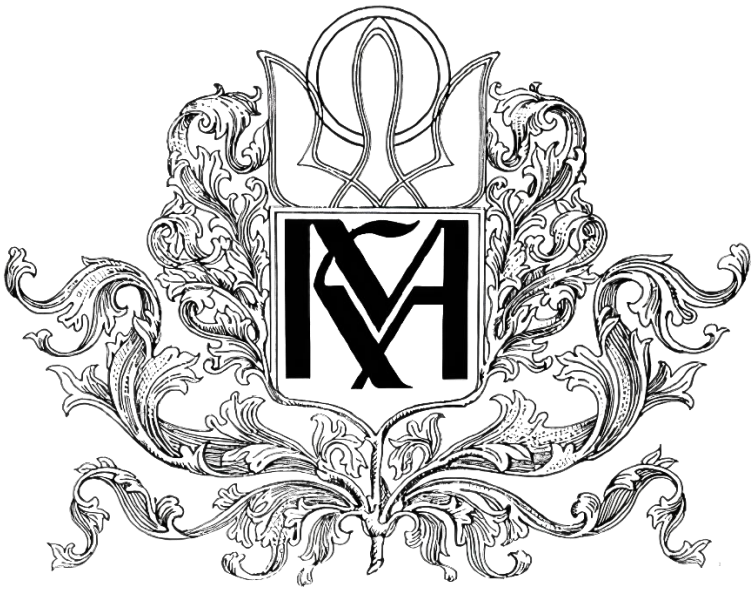
Міністерство освіти і науки України НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ» Кафедра мережних технологій факультету інформатики



«Система моніторингу для мережі підприємства»

Текстова частина до курсової роботи за спеціальністю «Інженерія програмного забезпечення» 121

Керівник курсової роботи

старший викладач Черкасов Д.І.

_____

(підпис)

“____” _____ 2024 р.

Виконав студент

Крутінь І.С.

“____” _____ 2024 р.

Календарний план виконання роботи

№	Назва етапу курсової роботи	Термін виконання етапу	Примітка
1.	Визначення теми кваліфікаційної роботи	Жовтень 2023	
2.	Отримання завдання на кваліфікаційну роботу	Жовтень 2023	
3.	Огляд літератури	Листопад - Грудень 2023	
4.	Написання теоретичної частини кваліфікаційної роботи	Січень 2024	
5.	Написання програмної реалізації	Лютий 2024	
6.	Написання практичної частини кваліфікаційної роботи	Березень - Квітень 2024	
7.	Захист кваліфікаційної роботи	Травень 2024	

Студент _____

Керівник _____ “_____”_____ 2024

Анотація

Ціллю роботи було знаходження найкращої, на даний момент, системи моніторингу для підприємства. Було проведено аналіз поточної ситуації на ринку даних систем і виявлено три найкращих кандидати : Zabbix, Prometheus, Nagios. У результати порівняльного аналізу, було виявлено, що саме Zabbix є найкращою системою моніторингу.

Через виявлення недоліків у роботі репортингової системи Zabbix було створено додаток, який значно її покращує і дозволяє отримувати посерверні звіти у форматі PDF.

Для демонстрації можливостей роботи з API Zabbix , було додатково створено Telegram Bot, який сповіщає користувача про нові сповіщення, які виникають у системі.

Зміст

Вступ

Будь-яке сучасне підприємство має вибір із десятків різних можливих конфігурацій та архітектур, які можуть бути застосовані до поставленої задачі. Більше того ми живемо у дому технологічної доступності, тому розширення таких підприємств відбувається доволі часто.

Тенденція до швидкого розширення системи, а також збільшення компонентів у ній призводить до того, що контроль над усіма її аспектами стає експоненціально складнішим і вимагає великої кількості уваги. Не достатня кількість часу та ресурсів, приділених до цієї проблеми, може призвести до таких ситуацій як критичний збій у роботі бази даних чи, наприклад API, який залишиться непоміченим. Оскільки майже будь-яка компанія потребує, щоб їх сервіси були доступні постійно, тобто безвідмовно, це питання стає все більш актуальним.

Проте, рішення уже існує і це – системи моніторингу. Вони являють собою програмне забезпечення, з або без інтерфейсу, яке дозволяє налаштувати певний перегляд тої, чи іншої інформації, якої потребують підприємства для того, щоб упевнитися у роботоздатності застосунку, сервісу і тд. Дані засоби дозволяють не тільки бачити теперішній стан системи, а й сповіщати про проблеми у разі їх виникнення, що є ключовою їх роллю.

Тим не менш, хоча на даний момент ми маємо велику кількість різноманітних систем моніторингу, багато з них є спеціалізованими, наприклад, Grafana, яка в основному, використовується у випадках, де саме візуалізація даних є основною потребою, а не можливість порівнювати дані з різних ресурсів, чи інші функції. Або ж Datadog, який є одним із лідерів для моніторингу саме хмарних рішень.

Таким чином, можемо побачити, що хоча ми і маємо вибір із багатьох засобів моніторингу, існує потреба дізнатися, який із них є найкращим для загального користування, тобто, такий, що дозволить моніторити якомога більше ресурсів - від серверів до баз даних та сайтів, а не лише специфічні речі, такі як мережа. А також розглянути, як можна покращити його роботу чи функціонал.

Отже, задачею курсової роботи є знаходження найкращого, гнучкого рішення для моніторингу підприємства, та можливості його покращення.

Це завдання можна розбити на два різні етапи :

1. Вибір найкращої, загальної системи моніторингу серед уже існуючих.

2. Знаходження її можливих вад та виправлення їх за допомогою розробки покращуючого рішення.

Розділ 1. Аналіз існуючих рішень моніторингу для виявлення лідера для загального використання.

За останні декілька років ринок систем моніторингу значно розширився, що сприяло збільшенню конкуренції та, як результат, гонці для покращенні систем, щоб виділятися серед різних моніторингових засобів. Та не зважаючи на спроби більшості рішень створити щось нове, або додати більшу кількість функціоналу, було виявлено, що деякі системи моніторингу утримують свої позиції на ринку і навіть часто піднімаються вище. Такими системами виявилися : Zabbix , Nagios та Prometheus. Вони мають найбільшу кількість функціоналу серед конкурентів, і відповідно, найбільші спільноти у просторі моніторингових рішень.

Для того щоб чітко перевірити, яка з цих системи, на даний момент, є найкращою, потрібно спочатку окрему розглянути можливості кожної із них, а потім порівняти результати. Отже, визначимо критерії, за якими будемо їх порівнювати :

- Доступність – чи система є платною, та чи є open-source рішенням.
- Крива опанування – на скільки складно зрозуміти та використовувати функціонал рішення.
- Система сповіщень («alerts») – кожний моніторинговий засіб має мати хоча б якийсь спосіб для сповіщення про відхилення у роботі інфраструктури чи сервісів.
- Зручність використання – чи має система графічний інтерфейс, та на скільки інтуїтивним він є для використання. Наскільки є безболісною робота з додатком, наприклад, процес його встановлення та початкового налаштування, тощо.

- Візуалізація даних – чи надає система можливість створювати, або переглядати уже готові графіки для аналізу зібраних метрик та загальної інформації.
- Архітектура – як побудована система, які є її складові.
- Гнучкість – можливість безболісного розширення, у разі збільшення підприємства, та кількість способів налаштування системи.
- Обробка та аналіз зібраних даних – до такого функціоналу входить наприклад звітування, яке, показує поведінку конкретних сервісів чи хостів протягом певного періоду часу.

Цих критеріїв має бути достатньо, для того щоб об’єктивно порівняти та знайти найкращу систему моніторингу для загального користування, на сьогоднішній день.

1.1 Аналіз системи моніторингу Nagios

На огляд візьмемо саме продукт Nagios XI, оскільки сама компанія надає низку інших специфічних сервісів, але вони не підходять для загального використання.

1.1.1 Загальні відомості про доступність та відкритість

Nagios XI – частково безкоштовне рішення, яке позиціонує себе як професійний застосунок для моніторингу стану мережі та серверів. Частково open-source, оскільки працює на базі Nagios Core, який є у відкритому доступі. Для моніторингу Nagios XI використовує систему вузлів («nodes»), де кожен вузол це окремий пристрій, наприклад, комутатор, роутер чи сервер. Безкоштовна версія надає можливість моніторити дуже обмежену кількість вузлів, тому для навіть маленького підприємства потрібно буде купувати хоча б версію на 100 вузлів, яка є найменшою за обсягом після безкоштовної, але може помітно вдарити по бюджету невеликих компаній.

1.1.2 Складність роботи із системою

Крива опанування для Nagios XI є далеко не стабільною, оскільки в залежності від кількості вузлів та сервісів, база потрібних знань змінюється. Наприклад, Nagios XI є відомим, перед усе, за свою обширну систему плагінів, яка дозволяє прискорити роботу та налаштування моніторингу в загальному. Плагіни знаходяться між самим Nagios та вузлом, тож можливість створювати свої власні плагіни розширює способи моніторингу різних сервісів.

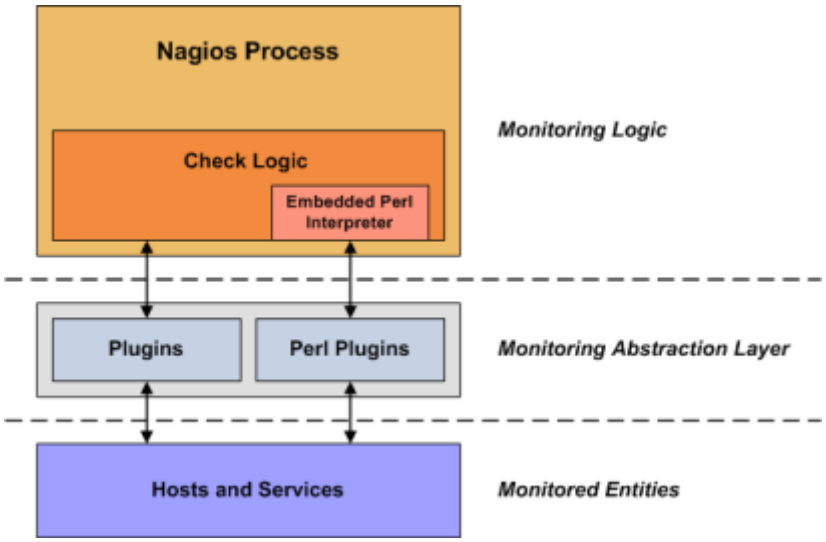


Рис 1.1.1 Спрощена архітектура системи Nagios з виділенням шару плагінів

Проте, з відносно малою кількістю вузлів, скажімо 10-15, достатньо буде використовувати лише стандартні плагіни для більшості ресурсів та вузлів, але зрозуміти цю систему доволі складно, оскільки окрім цього Nagios вимагає правильного налаштування зв'язку між собою та вузлами та відповідного налаштування вузлів у самому інтерфейсі, що на початку роботи із цією системою не є легким завданням, через не найкращий UI інтерфейс та документацію. Після цього початкового кроку, робота з Nagios має бути відносно простою, так як додавання нових вузлів чи плагінів для сервісів не має складати проблеми. Проте, для великих підприємств, які постійно розширяються, робота з Nagios може призвести до так званого «пекла конфігів» («config-hell»), при якому неймовірна кількість власно написаних плагінів, та навіть тих, що доступні у відкритому доступі, потребують постійних змін та контролю їх налаштувань, під різні нові вузли чи сервіси, які часто потрапляють до таких компаній.

1.1.3 Система сповіщень та її можливості

Nagios XI дозволяє налаштувати різноманітні сервіси для роботи із повідомленнями. Самі повідомлення можуть мати наступні значення :

Значення	Опис
PROBLEM	Сервіс або хост мають проблеми з доступом. Тобто для сервісу це може означати завершення роботи, помилка роботи, або ж неможливість до нього достукатися. Для хоста це може означати, що він перестав працювати, або ж є не доступним, наприклад, змінився його IP.
RECOVERY	Якщо сервіс або хост стали доступними після винекнення проблем. Для сервісу – стан OK, а для хоста – UP.
ACKNOWLEDGEMENT	Загальне повідомлення, яке показує інформацію про певну помилку, попередження чи просто певну інформацію, яка з’являється при використанні власно налаштованих плагінів.
FLAPPINGSTART	Сервіс чи хост почали «мерехтіти» (flapping). Позначає часту зміну у стані сервісу чи хоста, наприклад, доступність/недоступність сервісу, або різкі підскоки у вокористанні процесора.
FLAPPINGSTOP	Сервіс чи хост перестали мерехтіти.
FLAPPINGDISABLED	Сервіс чи хост перестали мерехтіти, оскільки перевірка на мерехтіння була вимкнена.

DOWNTIMESTART	Хост або сервіс знаходяться під технічними роботами, відправка повідомленнь, що виникають під час будь-яких проблем буде зупинена.
DOWNTIMESTOP	Технічні роботи над хостом або сервісом було завершено, повідомлення, що відносяться до сервісу чи хоста, знову почнуть приходити.
DOWNTIMECANCELLED	Технічні роботи були відміннені власноруч, повідомлення, що відносяться до сервісу чи хоста, знову почнуть приходити.

Nagios XI дозволяє велику кількість способів відправки вище зазначених повідомлень, до них входять :

- Email
- Pager, маленький пристрій для приняття сигналів у вигляді, наприклад, звукових чи текстових повідомленнь.
- SMS
- Yahoo, ICQ, або MSN
- Аудіо сповіщення, у вбудованному інтерфейсі
- Налаштування свого способу використовуючи Nagios XI API.

1.1.4 Зручність користування.

Nagios XI не вимагає високого рівня знань для початкового налаштування його роботи, оскільки надає чітку інструкцію, до якої входить налаштування бази даних, для зберігання даних системи, а також веб «помічника» (configuration wizard), який допомагає налаштувати конфігурацію системи прямо через веб-інтерфейс, а не власноруч на сервері, на якому буде встановлено Nagios XI.

Веб-інтерфейс, хоча і наявний, але все ж, поступається своїм конкурентам, і на те є дві основні причини:

- Комплексність
- Дизайн

Після початкового налаштування системи, перше враження від інтерфейсу може сильно вплинути на рішення використовувати систему. Навіть без додаткових налаштувань, Nagiox XI має дуже заповнений інтерфейс, який містить купу вкладок та їх розгалужень, а з розширенням підприємства та, відповідно сервісів чи хостів, стає складно навігувати і використовувати його.

Дизайн рішення також має бути покращене, оскільки він і досі містить багато елементів, які безпосередньо впливають на UX складову інтерфейсу, як, наприклад, стандартні, малі, HTML селектори, які важко виділити серед купи інших вкладок та кнопок на екрані.

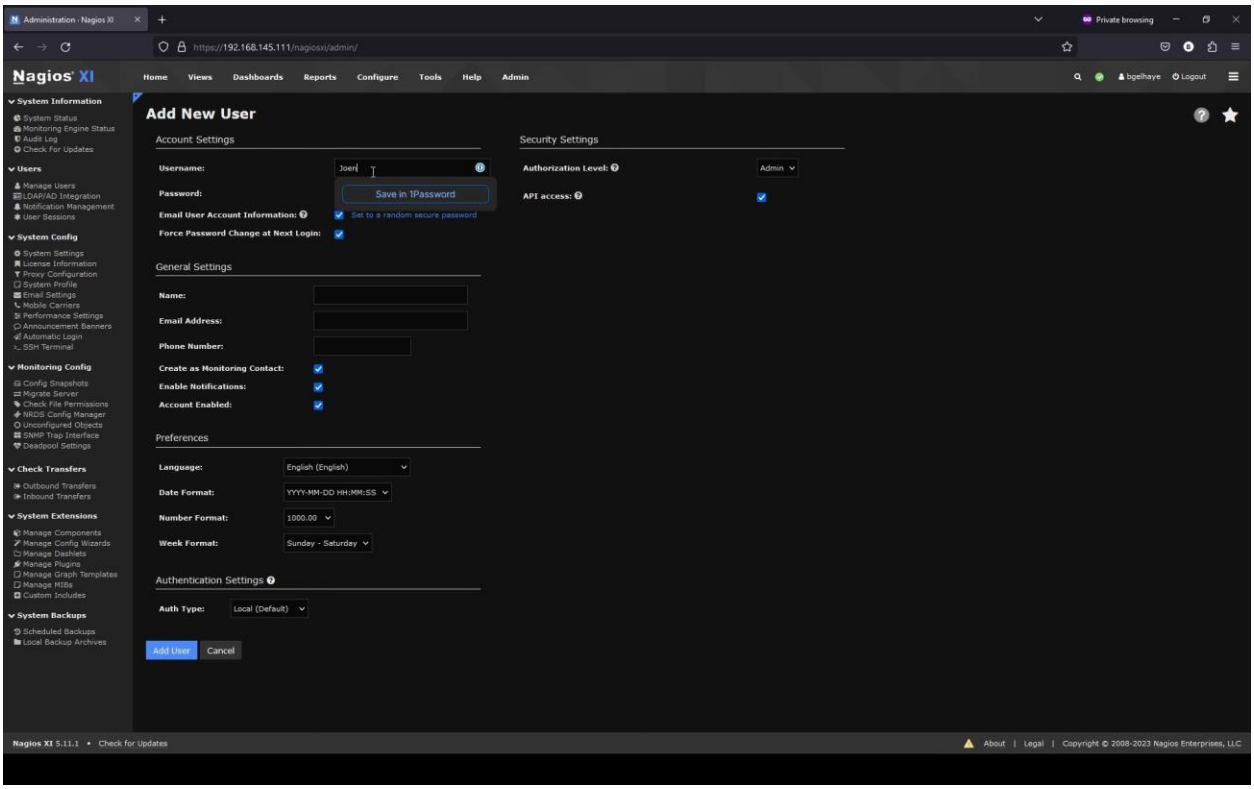


Рис 1.1.2 UI Інтерфейс Nagios XI

1.1.5 Візуалізація даних

Nagios XI надає безмежні варіанти візуалізації своїх даних, а все через систему плагінів, які надають додаткові компоненти для створення графіків. Навіть при стандартному наборі засобів, можна створювати непогані графіки, як для дашборду, так і для збору інформації на кожному вузлі. Та при додаванні, наприклад, компоненту Nagios XI Graph Explorer, який доволі просто налаштовується, можна створювати набагато кращі та різноманітні,

продвинуті, графіки.

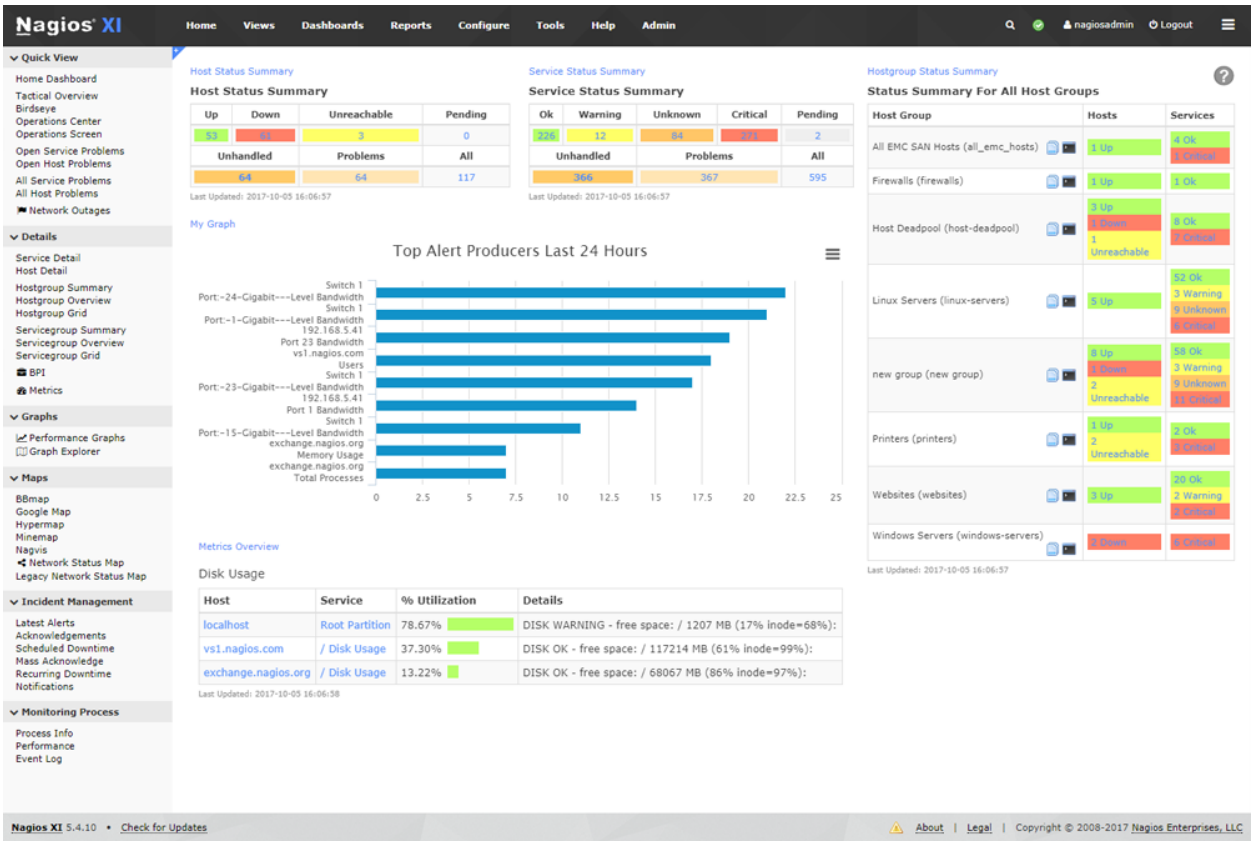


Рис 1.1.3 Візуалізація даних у Nagios XI. Приклад 1.

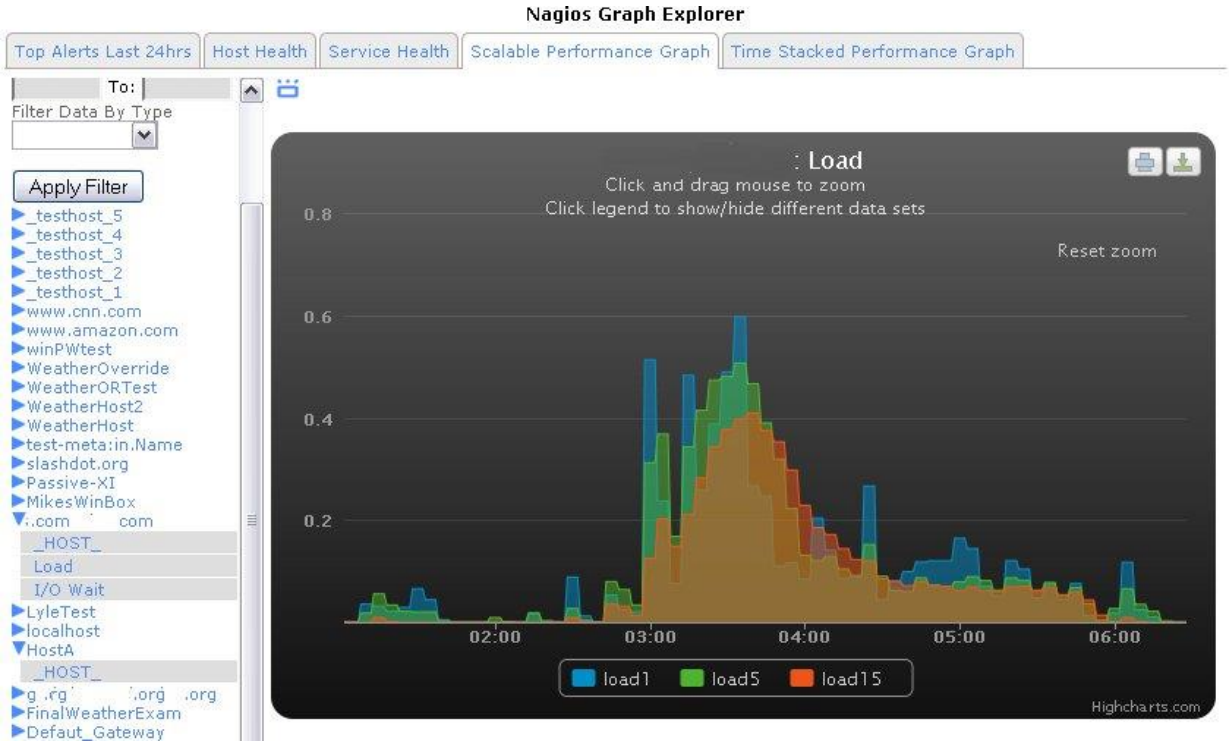


Рис 1.1.4 Візуалізація даних у Nagios XI. Приклад 2.

1.1.6 Архітектура

Nagios XI має просту, але в водночас ефективну і широку клієнт-сервер архітектуру.

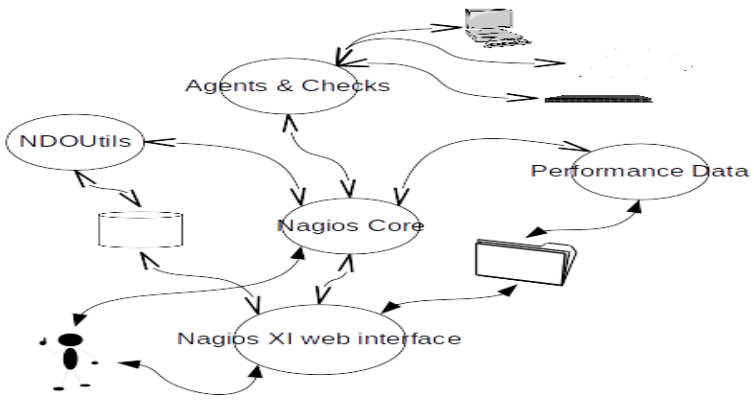


Рис 1.1.5 Архітектура Nagios XI. Приклад 1

Для повноцінної роботи він потребує лише серверу, якому безпосередньо і буде працювати, та базу даних, яка не обов'язково має бути на тому ж сервері. Далі достатньо використати один із можливих засобів з'єднання до віддалених нод, моніторинг яких і буде налаштовано. При налаштуванні віддаленої ноди і додаються ті плагіни, які будуть проводити перевірки. Спочатку відповідні плагіни виконують оцінювання стану сервісу, для якого вони налаштовані, а потім обробляють результат, після того як Nagios хост надішле сигнал через свій планувальник процесів. Статус цих тестів може мати один з чотирьох станів: OK, WARNING, CRITICAL або UNKNOWN.

Цього достатньо, щоб почати роботу з ним, але на цьому структура не закінчується. Nagiox XI базується на Nagios Core, який і є центром рішень і налаштувань усієї системи. Це корисно, оскільки люди, які раніше працювали напряму з Nagios Core, можуть завантажити їхні вже готові моніторингові системи, з їх плагінами та конфігураціями (config files) , без жодних проблем.

Не мало важливою частиною загальної архітектури Nagios XI є можливість розширюватися, наприклад, за допомогою доповнення NagiosQL можна налаштовувати конфігурацію всієї системи напряму, через веб-інтерфейс, без нього це б було не можливо. І таких плагінів та доповнень , які використовуються можливості архітектури Nagios XI для того, щоб покращити роботу із системою та кастомізувати її, існує багато.

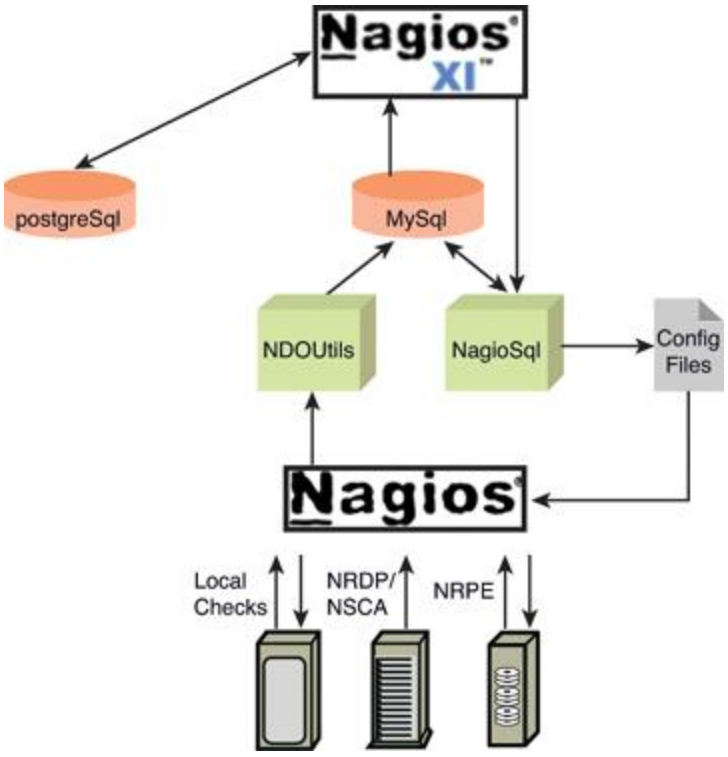


Рис 1.1.6 Архітектура Nagios XI. Приклад 2

1.1.7 Гнучкість

Хоча кастомізація Nagios XI досить хороша, проте його можливості розширення не є найкращими. Він, зазвичай, використовується для роботи над 1-3 проектами, оскільки велика кількість конфігурацій та плагінів, які налаштовуються під специфічні проекти робить розширення, з точки зору практичності, доволі складним. Тому часто і можна побачити Nagios, який налаштований та кастомізаний до найменших деталей, але працює лише з одним проектом чи задачею.

1.1.8 Обробка та аналіз зібраних даних

Звітність щодо роботи сервісів та хостів є дуже важливим аспектом роботи з будь-якою системою моніторингу. Для цього Nagiox XI надає окрему

вкладинку у своєму інтерфейсі, яка дозволяє створювати обширні та точні звіти.

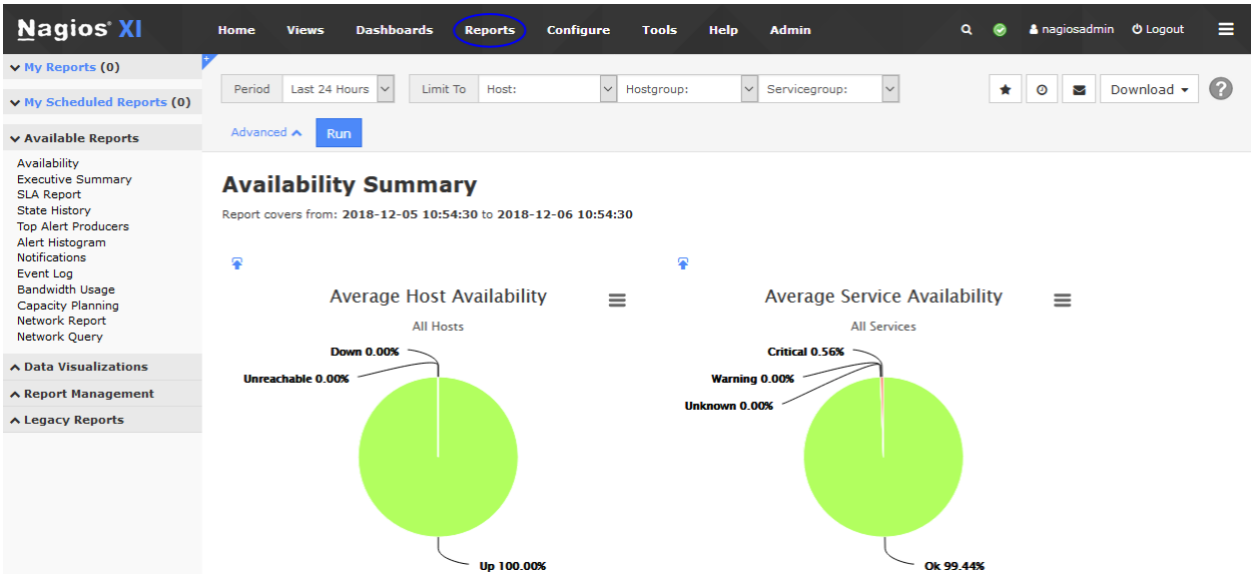


Рис 1.1.7 Вкладка для звітів у Nagios XI

Наприклад, можна фільтрувати їх по часу, датам, розміру, метрикам і тд. Також є можливість генерувати звіти як на групи хостів чи сервісів так і для і кожного індивідуально. Не мало важливо, що вберегти їх можна і великій кількості різних форматів, від PDF до JPEG.

Якщо стандартних засобів створення репортингу не достатньо, завжди можна доповнити Nagiox XI сторонніми засобами.

1.2 Аналіз системи моніторингу Prometheus

1.2.1 Загальні відомості про доступність та відкритість

Prometheus – open-source система моніторингу, яка є доступною для всіх. Створювався він із ідею роботи з мікросервісами, які є зараз дуже популярним архітектурним рішенням для розробки програмного забезпечення, що також посприяло його росту.

1.1.2 Складність роботи із системою

Для того, щоб розібратися як працює Prometheus та створити на базі нього моніторингову систему - багато часу не потрібно. Він є легким як в розумінні так і використанні, оскільки має велику базу користувачів та обширну документацію, яка надає купу прикладів налаштування системи на свій смак. Та і сама система не потребує постійного навчання для її розуміння, а просто набудовує новий функціонал на вже готовій основі. Це є великим плюсом роботи з Prometheus.

1.1.3 Система сповіщень та її можливості

У Prometheus сповіщення працюють через Prometheus Alertmanager, який є дуже потужним інструментом. Він дозволяє групувати сповіщення на ваш вибір, наприклад, у разі падіння бази даних, не лише вона перестає працювати, а і багато сервісів, які від неї залежать, тому, людині, яка наглядає за системою можуть прийти десятки чи сотні алертів, в залежності він розміру підприємства. У в таких випадках можна згрупувати всі ці сповіщення в одне, яке буде приходити після падіння бази даних і містити у собі дані не тільки про стан бази, а і всі уражені сервіси.

Самі сповіщення створюються в ручну та підлягають обширному рівню кастомізації. Наприклад, є такі налаштування:

- alert : Тип сповіщення, наприклад, HighRequestLatency
- expr : Вираз, який буде діставати метрики із хоста чи сервісу
- for : Змушує Prometheus почекати надану кількість часу перед відправкою сповіщення.
- severity : Описує на скільки серйозним є алерт, за замовчуванням таких значеннь немає, їх надає користувач, наприклад High, Disaster, Normal, Ignore і тд.

І таких можливостей кастомізації сповіщень є велика кількість. Також Prometheus дозволяє використовувати шаблонування, для того, щоб робити алерти більш загальними і не повторно використовувати їх для різних хостів чи сервісів, наприклад :

- summary: "Instance {{ $labels.instance }} down"

Тут назва хоста, який перестав працювати, буде автоматично підставлятися, що дозволить створити лише один загальний алерт, а не додавати його для кожного хоста окремо.

Подальші налаштування відбуваються саме через Alertmanager, де також можна налаштувати запуск певних скриптів при надходженні алертів, наприклад, для перезапуску сервісів. Alertmanager надає можливість фільтрувати та «замикати» алерти, тобто якщо, наприклад, певний алерт не має приходи, бо вже приходить якийсь інший, можна його фільтрувати. «Замикання» - коли потрібно вимкнути алерти з певної причини, але тимчасово і не видаляти їх, наприклад, під час технічних робіт.

Останнім кроком у всьому процесі сповіщень є їх відправка і Prometheus надає необмежену кількість можливостей для цього. Від звичайних SMS та Email до повністю кастомізовних рішень, таких як власні боти чи дзвінки особам, які мають дізнатися про алерт. Це є можливим, оскільки Alermanager має свій окремий API, який має різноманітні можливості, які направлених на кастомізацію як і алертів так і їх доставки.

1.2.4 Зручність користування.

Для встановлення системи Prometheus та її початкового налаштування була створена детальна документація, яка чітко описує, що потрібно виконати для правильної її роботи. Також, що не мало важливо, вона є обширною, оскільки надає різні способи запуску Prometheus, такі як Docker контейнери, чи використання систем конфігурацій, наприклад, Ansible, Puppet, Chef і тд. Проте досить вагомим мінусом є те, що більшість цієї роботи потрібно буде виконувати вручну, оскільки засобів, які пришвидшують, чи полегшують встановлення системи Prometheus не надає.

Щодо графічного інтерфейсу, то тут можна наглядно розглянути два варіанти :

- Вбудований
- Сторонній (Grafana)

Вбудованим веб-інтерфейсом є той, який можна одразу побачити після встановлення системи, він є сучасним та інтуїтивним, що дозволяє доволі просто пересуватися по ньому і працювати із потрібним матеріалом.

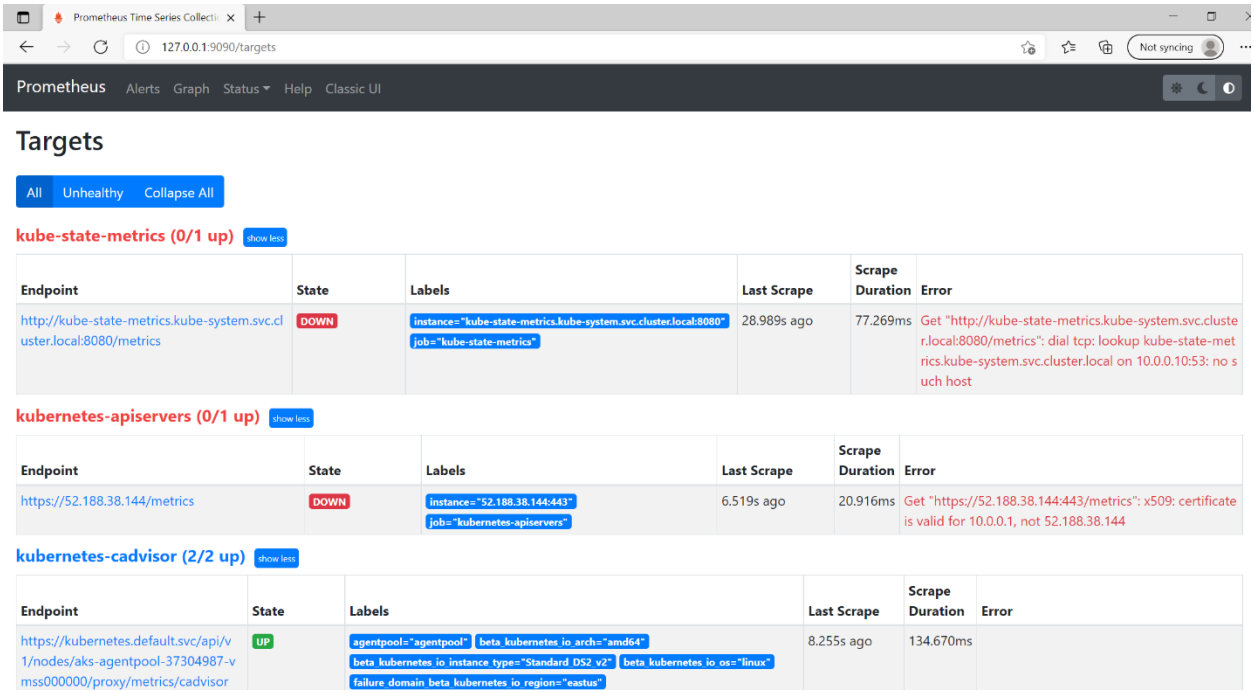


Рис 1.2.1 Вбудований графічний інтерфейс Prometheus

Проте, хоча вбудованого інтерфейсу і достатньо для роботи із системою, оскільки він надає можливість працювати та переглядати усі основні компоненти моніторингових систем, такі як сповіщення, прості графіки, статуси сервісів та серверів, та інше, багато кому було не достатньо вбудованих можливостей.

1.2.5 Візуалізація даних.

Саме простота відображення цих метрик , тобто графіків з ними, та загальної статусної інформації, породили бажання до чогось, що буде краще з цим працювати.

Grafana – система візуалізації та загального моніторингу, яка підтримує досить обширну кількість різних сервісів, які можуть надавати їй дані, так звані «DataSource». Одними із перших цих сервісів був Prometheus, який і досі є найпопулярнішим рішенням при роботі із Grafana. Причиною є те, що під’єднати Prometheus до неї дуже просто, а налаштувати збір даних, також не

вимагає багато зусиль. Так вирішуються одразу дві проблеми з UI Prometheus – кастомізація візуалізації даних, та їх комплексність. Grafana дозволяє комбінувати результати з різних метрик, порівнювати їх, підключати відповідні сповіщення, та навіть створювати окремі шаблони під окремі задачі, що робить таку зв'язку чи не найпотужнішою, при роботі з Grafana.

Рис 1.2.2 Графічний інтерфейс Grafana

1.2.6 Архітектура

Prometheus доволі сильно, за архітектурою, відрізняється від конкурентів :

- Він не потребує сторонніх баз даних а використовує свою - TSDB, де після збереження даних, вони утримуються на пристрої, де розміщений prometheus, він позначається як вузол («Node»).

- Навколо цього «ядра» набудовується Alertmanager, PromQL, який аналізує дані для перетворення їх у прийнятний для UI, та не тільки, вигляд.
- Також Pushgateway, який виступає своєрідним шлюзом, який передає метрики до самого ядра.

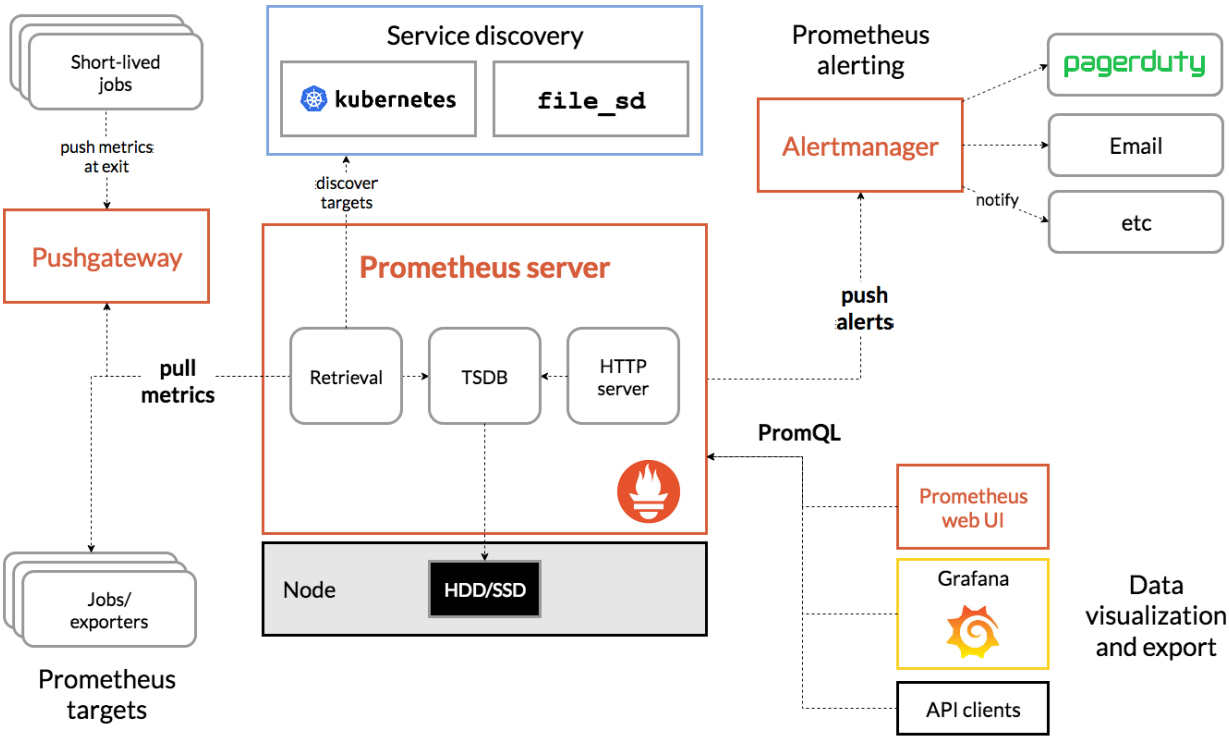


Рис 1.2.3 Розширена схема архітектури Prometheus

Для самого збору даних Prometheus використовує «Jobs», тобто певні інструментовані завдання, з яких він потім напряму, або через Pushgateway, якщо це короткотривалі задачі, збирає дані на обробку.

Самі метрики побудовані як «часові», тобто такі, де кожна метрика має тайм код, коли була зібрана, разом із опціональною парою «ключ-значення».

1.2.7 Гнучкість

Архітектура Prometheus, за означенням, робить його доволі різностороннім засобом, що і зробило його популярним при роботі з мікросервісною архітектурою, або для швидкого та ефективного моніторингу у вигляді метрик як «ключ-значення».

Проте, він ще дозволяє інтегрувати сторонні рішення, більшість з них є детальними і експортується від відповідних батьківських компаній, тому позначаються як офіційні, але можна створювати і свої інтеграції, які користувач може також встановлювати, але вже без гарантії на апдейти, виправлення помилок, та інше.

1.2.8 Обробка та аналіз зібраних даних

На жаль, окрім стандартизованої обробки даних перед їх виведенням в UI, Prometheus не має можливостей для обробки даних. Тобто він не містить системи репортингу, яка дозволить зібрати дані чи то за фільтром, чи якимось іншим критерієм, і використати даний репорт для аналізу потрібної задачі - це потрібно робити самому, або ж використовувати сторонні засоби, такі як Grafana.

1.3 Аналіз системи моніторингу Zabbix

1.3.1 Загальні відомості про доступність та відкритість

Zabbix – найстарша, з уже розглянутих, open-source система моніторингу, яка має обширні можливості моніторингу : від сервісів та хостів, до клауд рішень і IOT пристроїв.

1.3.2 Складність роботи із системою

Налаштувати свій перший моніторинг у цій системі – просто. Це зумовлено тим, що Zabbix використовує блочну схему налаштування моніторингу у своєму середовищі. Наприклад, для того, щоб просто поставити хоста на моніторинг, достатньо встановити десь Zabbix сервер і на хості Zabbix агент, який і буде працювати над моніторингом хоста, і це все. При такому дуже простому і швидкому налаштуванні уже можна отримувати певні загальні метрики, але якщо хочеться більшого, то можна додати додатковий рівень безпеки на це з’єднання, або моніторити цілі групи хостів, а не по-одному. Можна навіть зробити збір ще безпечнішим, закривши самі хости у приватну мережу і встановивши перед ними Zabbix проксі сервер, який буде переймати запити від основного Zabbix серверу і діставати із відповідних серверів метрики для нього.

Не мало важливим є також те, що документація Zabbix є найбільшою серед трьох конкурентів та зрозумілою навіть тим, хто перший раз бачить системи моніторингу. Вона містить як приклади налаштування певного функціоналу, так і графічні доповнення для кращого розуміння і порівняння.

На офіційному сайті Zabbix можна також замовити тренування, та сертифікацію, що значно полегшує поріг входу.

1.3.3 Система сповіщень та її можливості

Zabbix надає 6 типів для своїх алертів :

- Not Classified
- Information
- Warning
- Average
- High
- Disaster

Самі по собі вони мають лише свій специфічний колір та звук, який їх супроводжує, але при цьому не мають певного значення, воно обирається користувачем. Тобто якщо сервіс раптово перестав працювати, то сам користувач вибирає чи це Average чи Disaster.

Способи відправки сповіщень Zabbix має такі ж , що і конкуренти і також надає можливість використовувати API для створення кастомізованих рішень. Але окрім цього є вбудована можливість додавати свої «Media Type» та навіть створювати шаблони повідомлень для них.

Найбільш цікавою складовою системи алертів у Zabbix є можливість використовувати «Actions» , які і відправляють сповіщення, для того, щоб запускати команди чи цілі скрипти на віддалених хостах для швидкого вирішення проблеми, а також, навіть повертати результат виконання цього

запуску, наприклад у форматі email, де буде надано результат : успішно , чи не успішно.

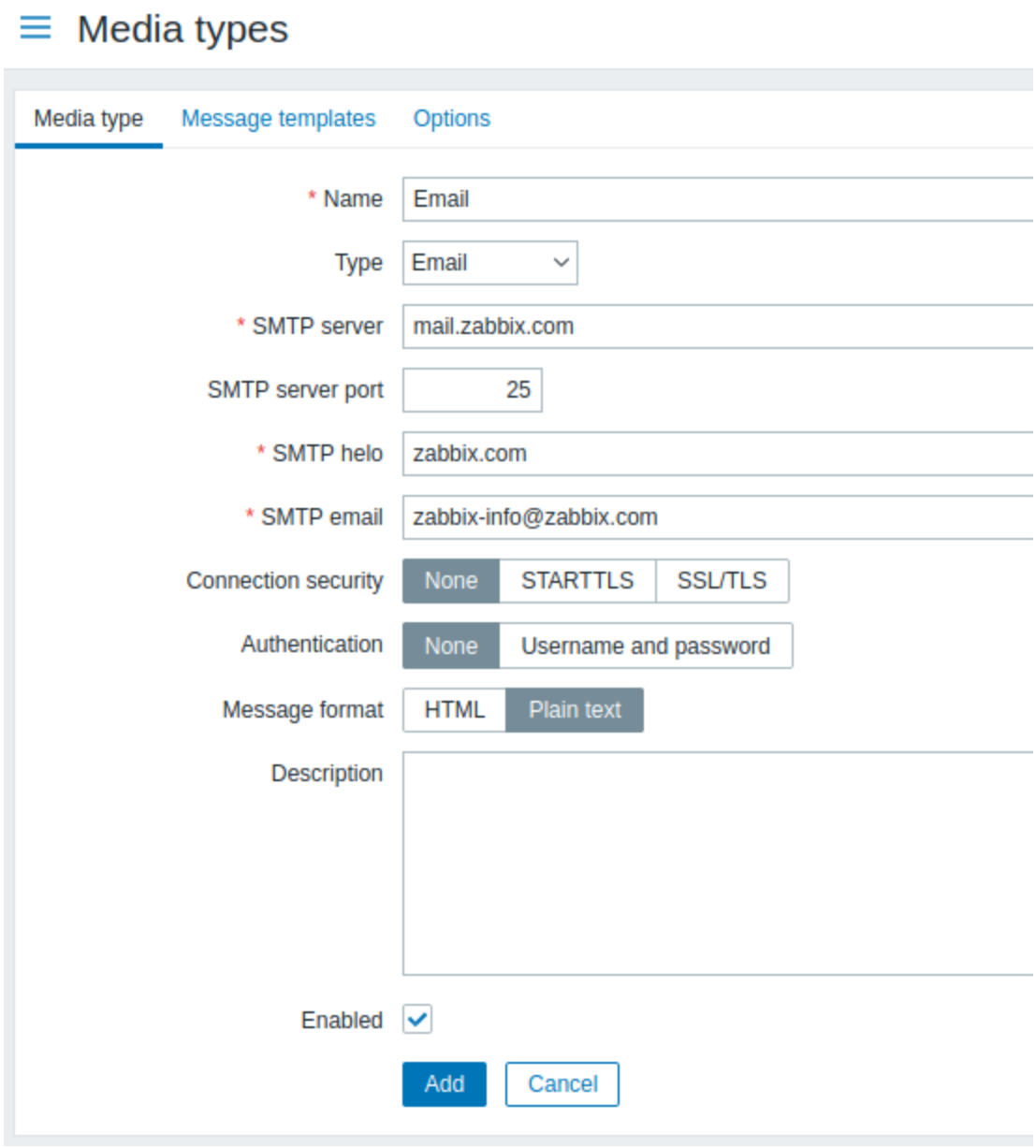


Рис 1.3.1 Типи ресурсів через, які можна відправляти повідомлення про алерти.

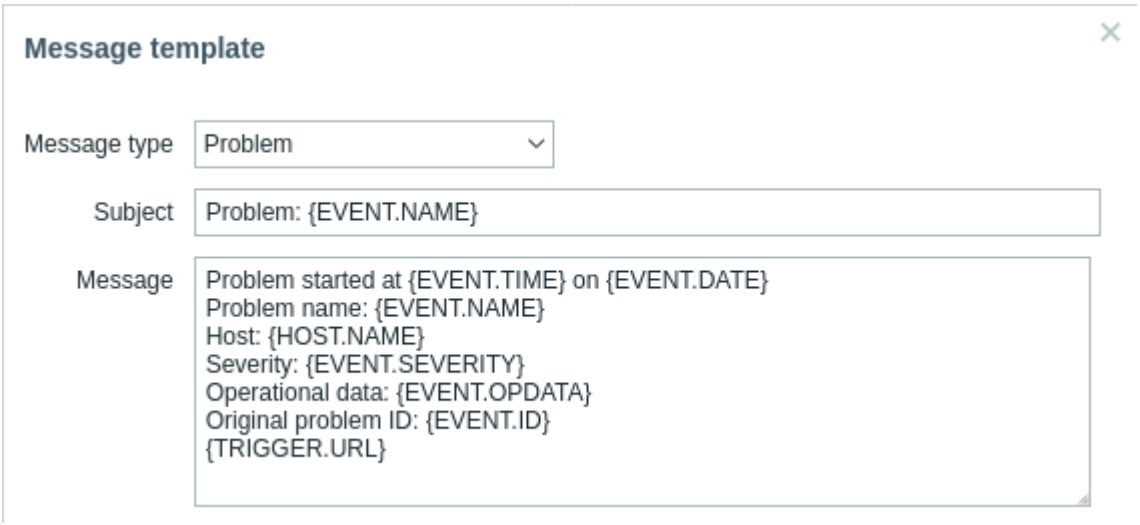


Рис 1.3.2 Приклад роботи із шаблонами повідомлення у Zabbix

1.3.4 Зручність користування.

Zabbix, на відмінну від Prometheus, потребує налаштування через файли конфігурацій лише на початку, а уся інша роботи виконується через сучасний веб-інтерфейс.

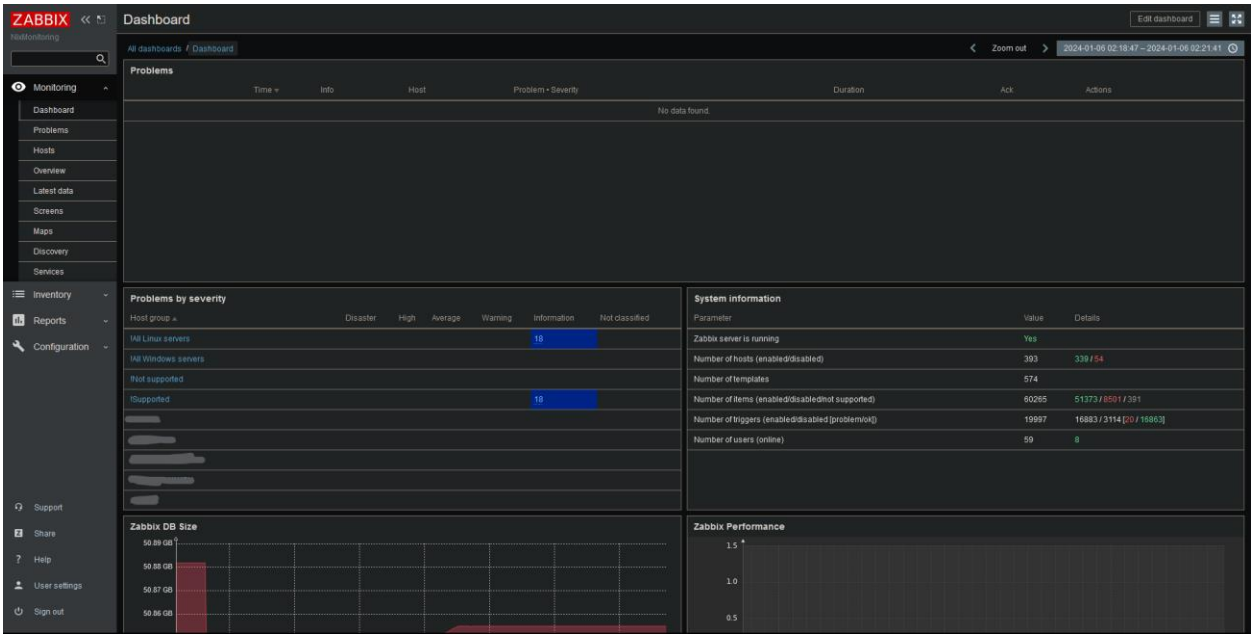


Рис 1.3.3 Інтерфейс zabbix

Окрім можливостей налаштовувати увесь потрібний функціонал, веб-інтерфейс також надає можливість кастомізації : зміна теми, розміщення вкладинок, горизонтальний чи вертикальний види, та додаткові віджети, які можна розміщувати на вхідній панелі.

У разі виникнення проблем, можна звернутися до вкладинок «Support» чи «Help» , що не тільки допоможе користувачеві швидше вирішити проблему, а і сповістити про її наявність, щоб у новій версії її вже не було.

1.3.5 Візуалізація даних.

Zabbix має вбудовану систему графіків, які збирають певну інформацію обрану користувачем. При бажанні переглянути специфічний графік, можна перейти до відповідного хоста та обрати останню інформацію про нього, де і можна буде побачити усі доступні графіки, які йому належать.

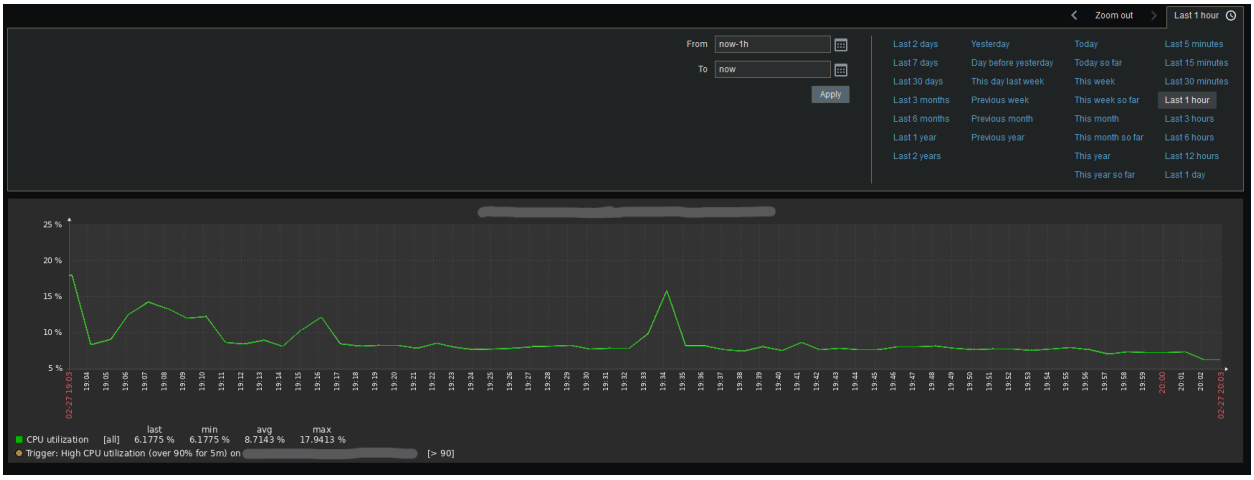


Рис 1.3.4 Приклад графіку у Zabbix

Також є можливість створювати свої власні графіки, які можуть містити декілька датасетів, а не використовуються для лише одного значення.

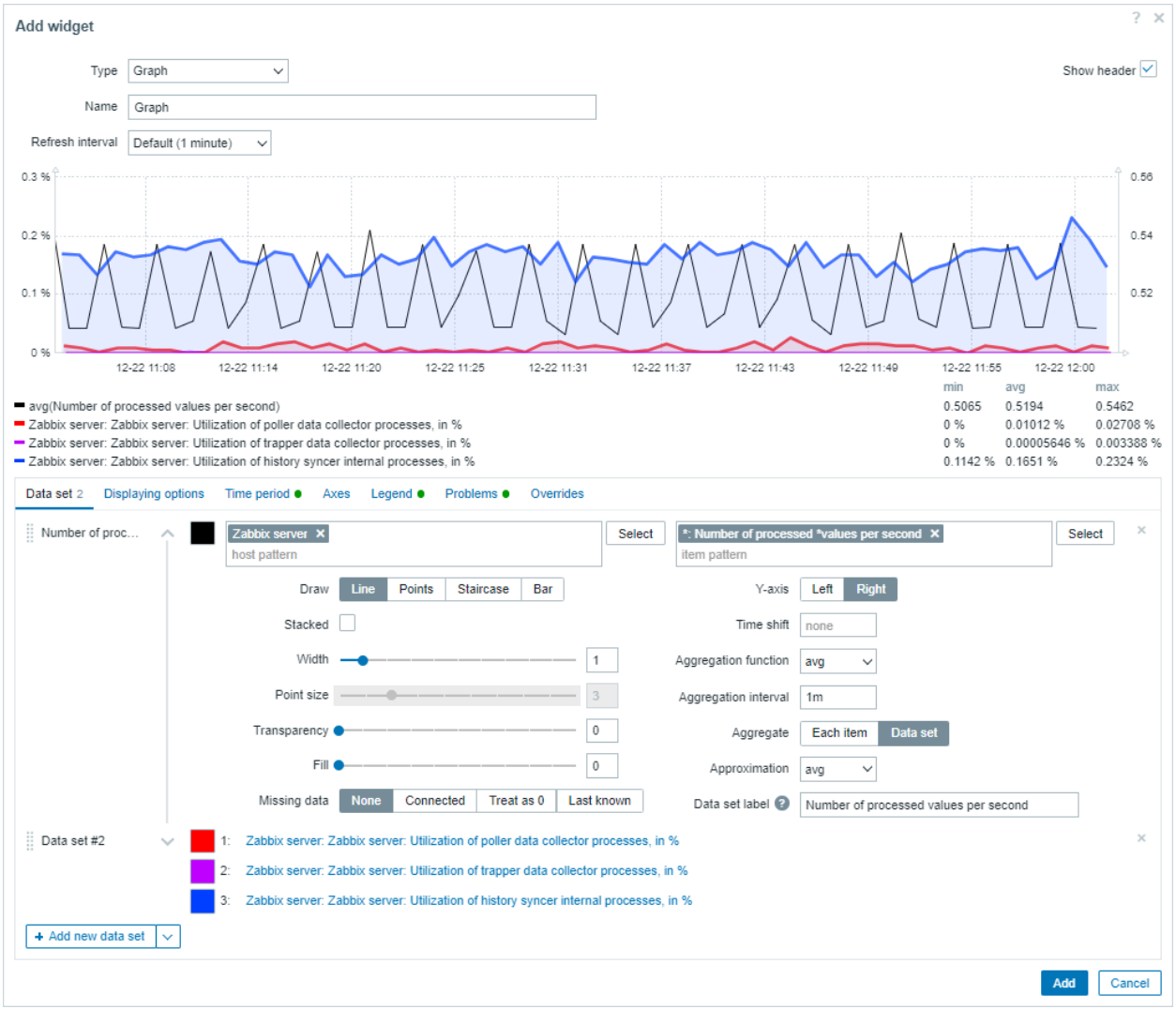


Рис 1.3.5 Меню створення власних графіків у Zabbix

1.3.6 Архітектура.

Zabbix, як і всі зазначені системи моніторингу, складається із декількох частин :

- Сервер – центральний та основний компонент системи, якому «агенти» відправляють усю зібрану інформацію.

- База даних – для успішної обробки та зберігання інформації наданої агентами, використовується SQL база даних, всього їх три на вибір : MySQL(MariaDB), PostGres, SQLite.
- Веб інтерфейс – доступний через будь-який популярний веб-сервер, наприклад Apache чи Nginx, за допомогою перенаправлення на відповідний порт Zabbix серверу.
- Проксі – про цей компонент уже згадувалося, він є не обов’язковим, але за потреби моніторити приватні мережі, можна його використовувати.
- Агент – «працівники» Zabbix, які встановлюються на пристрій, який буде моніторитися. Вони відповідно і збирають всю потрібну інформацію і після відправляють на Zabbix сервер.

Така архітектура дозволяє збирати інформацію ефективно і навіть не лише прості дані. Справа у тому, що так звані ітеми («item»), за визначенням це є звичайна пара ключ-значення, збирають якусь метрику на хості, але вони не обов’язково збираються їх на пряму : кожен Zabbix агент має змогу запускати скрипти і отримувати з них результати, будь-то Python чи Bash скрипт. До поки результатом є одне із можливих типів, які дозволяє визначити Zabbix, наприклад, JSON, float, string, то будь-яке значення ключа можна прив’язати до цього скрипта і таким чином динамічно діставати навіть комплексні метрики.

Групи ітемів називаються шаблонами. Шаблони можна створювати та імпортувати ззовні, це є своєрідним аналогом плагінів у Nagios, хоча і не є наскільки гнучким, оскільки для правильної робити цих шаблонів часто потрібно до налаштовувати певні речі на самих хостах.

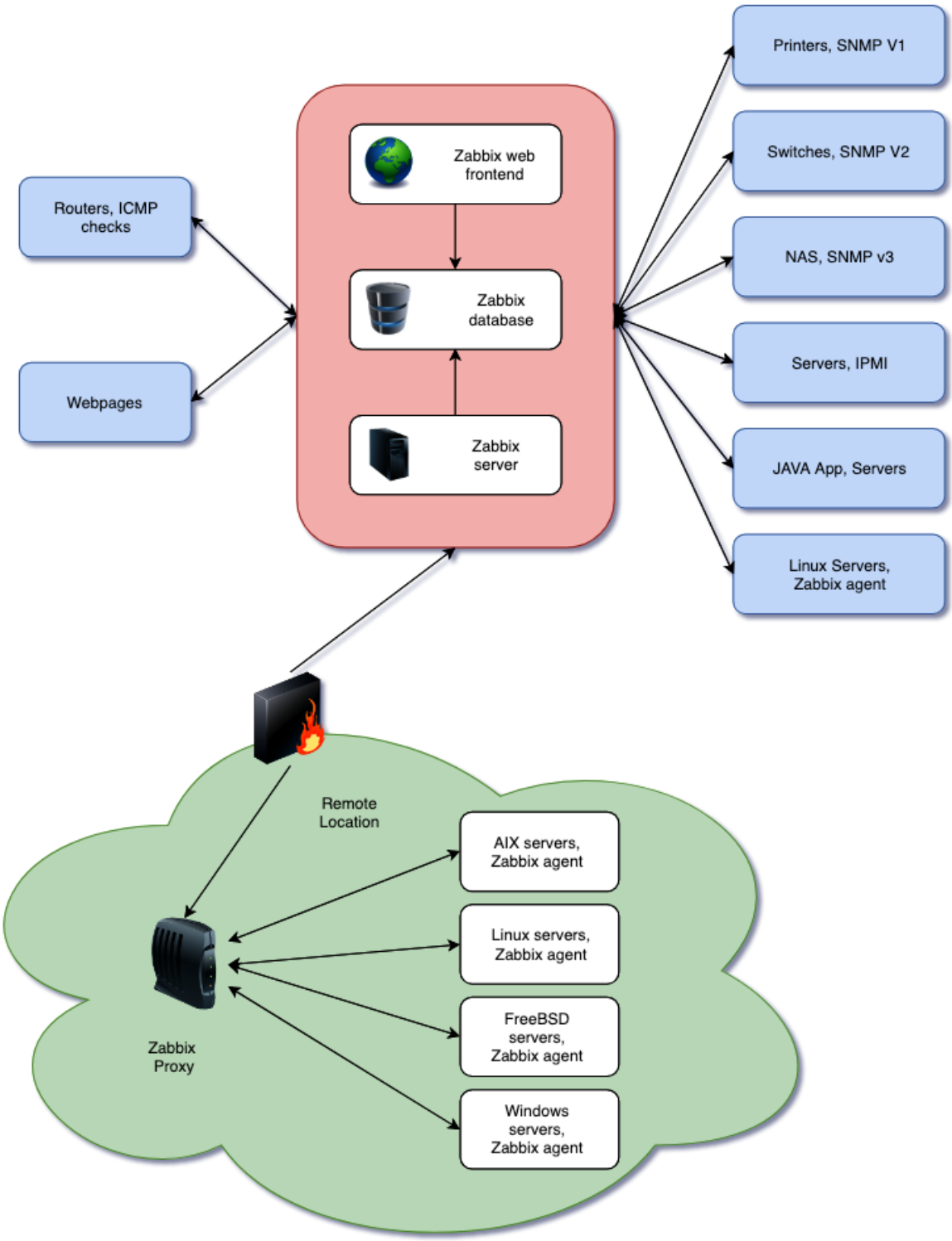


Рис 1.3.6 Архітектура Zabbix

1.3.7 Гнучкість

Завдяки шаблонами Zabbix просто і швидко масштабується. Наприклад, можна створити шаблон, який містить усі речі, які будуть моніторитися на кожному хості. Після цього достатньо встановити Zabbix агент на хост і додати шаблон до цього хоста у Zabbix сервері – це все, сервер повноцінно моніторить усі потрібні речі, і за бажання, до нього можна буде додати нові ітеми у цьому шаблоні, оскільки Zabbix дозволяє робити повні комії шаблонів і модифікувати їх для нових хостів.

Для того, щоб знайти більшість готових шаблонів достатньо відвідати офіційний репозиторій Zabbix для шаблонів, де містяться тисячі уже готових рішень.

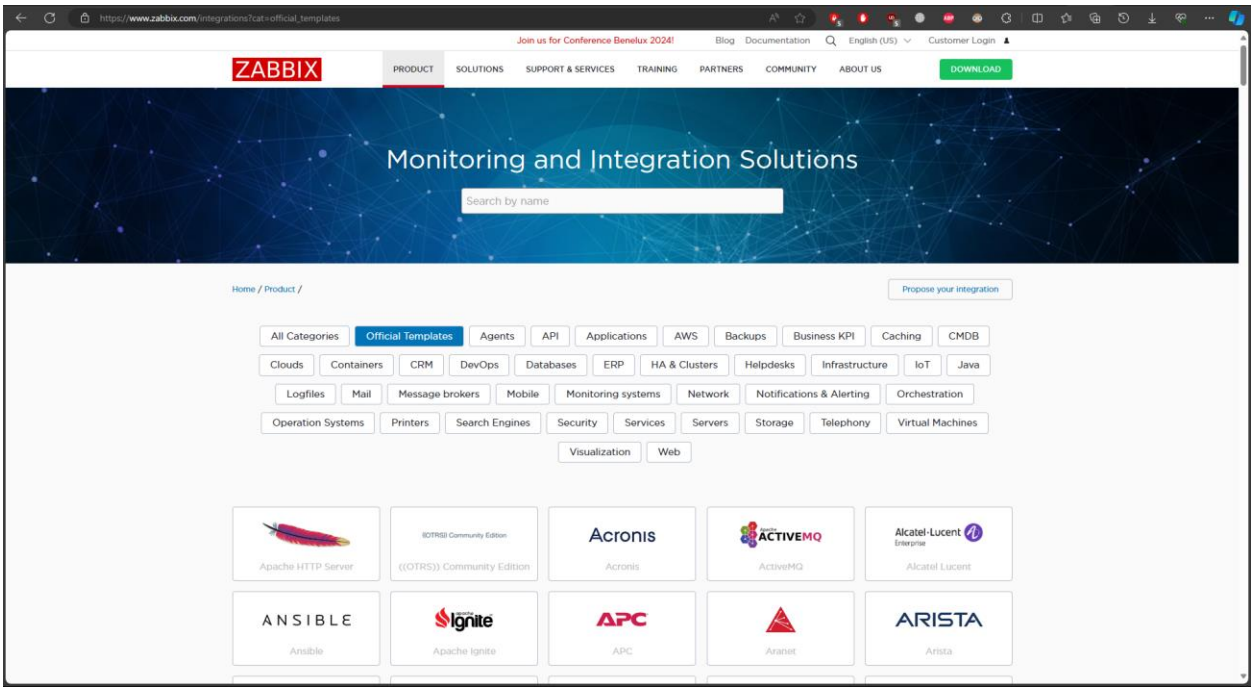


Рис 1.3.7 Онлайн репозиторій із готовими рішеннями для Zabbix

1.3.8 Обробка та аналіз зібраних даних

Zabbix надає декілька можливостей для аналізу та обробки отриманої інформації :

- Availability report – даний звіт містить інформацію про доступність усіх хостів у системі. Інформацію можна фільтрувати за часом чи групами хостів, але вміст самої інформації не змінюється.
- Triggers top 100 – надає, з можливістю фільтрації, 100 найчастіше використовуваних трігерів, тобто тих, які найбільше реагують на якусь зміну у сервісі чи хості.
- Notifications – список нотифікації, які було відправлено, з відповідною інформацією про їх кількість, хоста, та від якого користувача вони відправлялися.

Таким чином Zabbix хоча і надає функцію репортингу та аналізу даних на рівні Nagios, і краще ніж Prometheus, але усе одно сам репортинг є обмеженим готовими шаблонами, без можливості будь-якої фільтрації чи налаштувань власноруч.

1.4 Порівняльний аналіз роглянутих систем моніторингу

Так як більшість індивідуальних можливостей кожної із систем було розкрито, залишилося лише порівняти основні деталі і визначити найкращу систему моніторингу для загального використання.

	Zabbix	Prometheus	Nagios
Документація	Обширна, чудовий рівень деталізації та велика кількість прикладів.	На рівні із Zabbix, але при цьому доволі складно читається, оскільки має дуже багато прикладів для мануального налаштування Prometheus.	Виглядає застаріло та складно читається, кількість прикладів могла б бути більшою.
Час на встановлення системи	Завдяки чіткій документації – усе просто встановлюється та налаштовується.	Встановлюється найшвидше серед конкурентів через свою централізованість і незалежність від сторонніх баз даних.	Має швидкий процес встановлення та навіть самої конфігурації завдяки автоматичним засобам налаштування(wizard)

Графічний інтерфейс	Сучасний, простий та зрозумілий, не потребує додаткових засобів, щоб розібратися.	Має чудовий UI рівень, але є доволі обмеженим у функціоналі.	Складний у розумінні, але є найбільш гнучким і надає багато можливостей для конфігурації.
Розширення іншими готовими рішеннями	Має цілу екосистему, на кшталт dockerhub чи npm, шаблонів для хостів та сервісів.	Обмежений у цьому аспекті, оскільки потрібно усе робити в ручну, хоча готові конфігурації можна знайти у відкритому доступі.	Серед усіх моніторингових рішень має найбільшу таку систему. Вона містить тисячі плагінів, якими можна покращувати своє рішення.
Обробка та візуалізація даних	Має чудові можливості створення графіків, як уже готових, так і власних, але є доволі обмеженим у можливостях	Примітивний, але достатній рівень візуалізації. Не має рішень для аналізу та обробки готових даних.	Має багато можливостей візуалізації, але обмежений у обробці даних.

	обробки даних, наприклад, репортинг.		
Гнучкість	Окрім можливості збирати дані через агентів, можна також використовувати ssh, ping, SNMP та інші засоби. Також надає можливість працювати з проксі, що дозволяє моніторити навіть віддалені, закриті мережі хостів.	Працює лише на базі HTTP ендпоїнтів, з яких він і збирає дані. Не має вбудованих можливостей для моніторингу через проксі.	Має можливості моніторингу через ssh, ping, WMI, SNMP та агентів. Підтримує роботу через проксі.
Ресурси для оптимальної робити	CPU – 2 cores RAM – 8GB	CPU – 2 cores RAM – 4GB	CPU – 2 cores RAM – 4GB

Сповіщення не було розглянуто, оскільки кожна система має достойну їх кастомізацію, а також можливість навіть створювати власні рішення через API.

Як ми бачимо після порівняльного аналізу трьох рішень, Zabbix або перемагає у більшості випадків, або стоїть на рівні з конкурентами, хоча і вимагає не набагато більше ресурсів, тому саме він буде обраний як система, яка буде покращеною.

Zabbix поділяв одну проблему з усіма іншими рішеннями – доволі слабка обробка отриманих даних, вбудовані можливості репортингу обмеженні і не дають багато інформації. Отже для того, щоб позбутися цієї проблеми було вирішено створити власну систему репортингу для Zabbix, яка буде використовувати можливості його API. Також для того, щоб продемонструвати інші методи відправки даних, які можна створювати з використанням вище зазначеного API, було створено Telegram бота, який буде сповіщати про усі алерти, які надходять у Zabbix, та їх статус.

Розділ 2. Створення репортингової системи для Zabbix та власного рішення для сповіщень на основі Telegram.

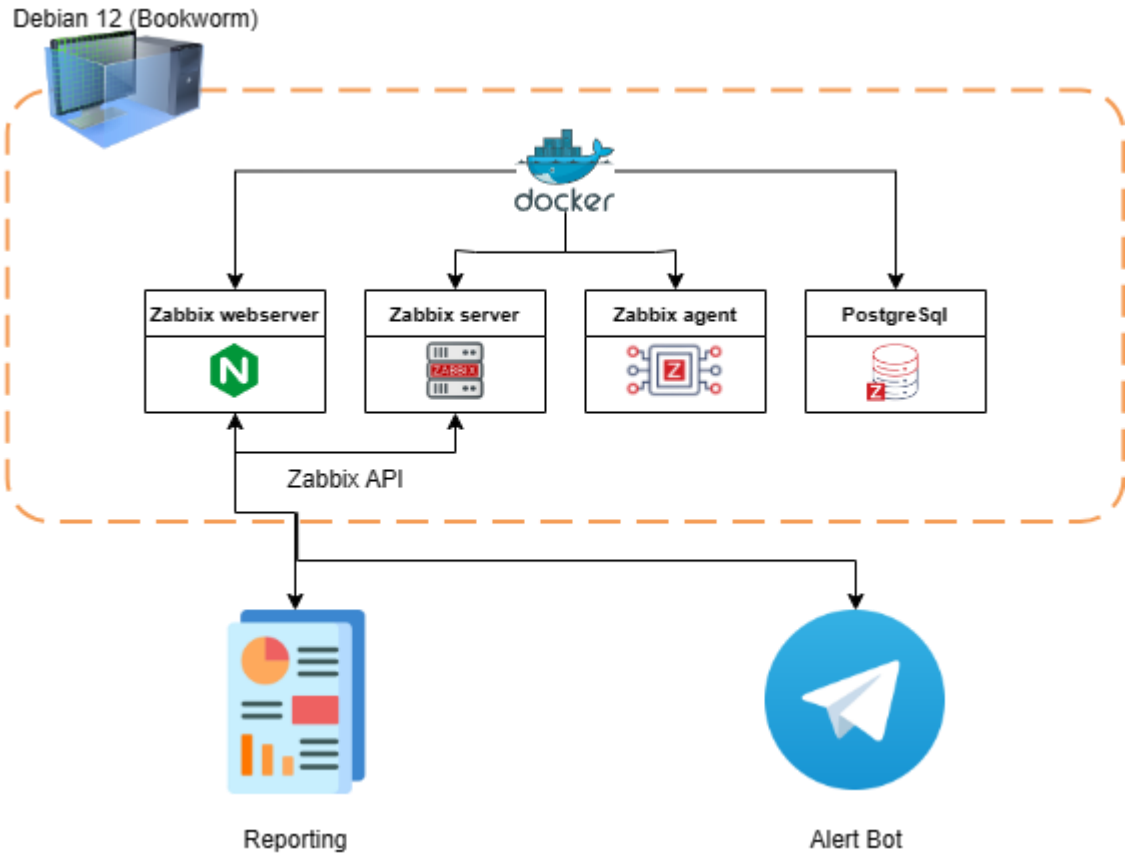
Перед тим як розглядати безпосередньо практичну частину, важливо розібрати архітектуру відповідних рішень, а також чому саме вони були обрані.

2.1 Архітектура хоста

Основою обох застосунків є взаємодія з API Zabbix. Для цього було створено віртуальну машину на основі Debian 12 (Bookworm), яка виступає у ролі хоста для Zabbix. Для запуску самого Zabbix використовувався Docker, оскільки в такому випадку усі компонентні складові будуть ізольовані, що

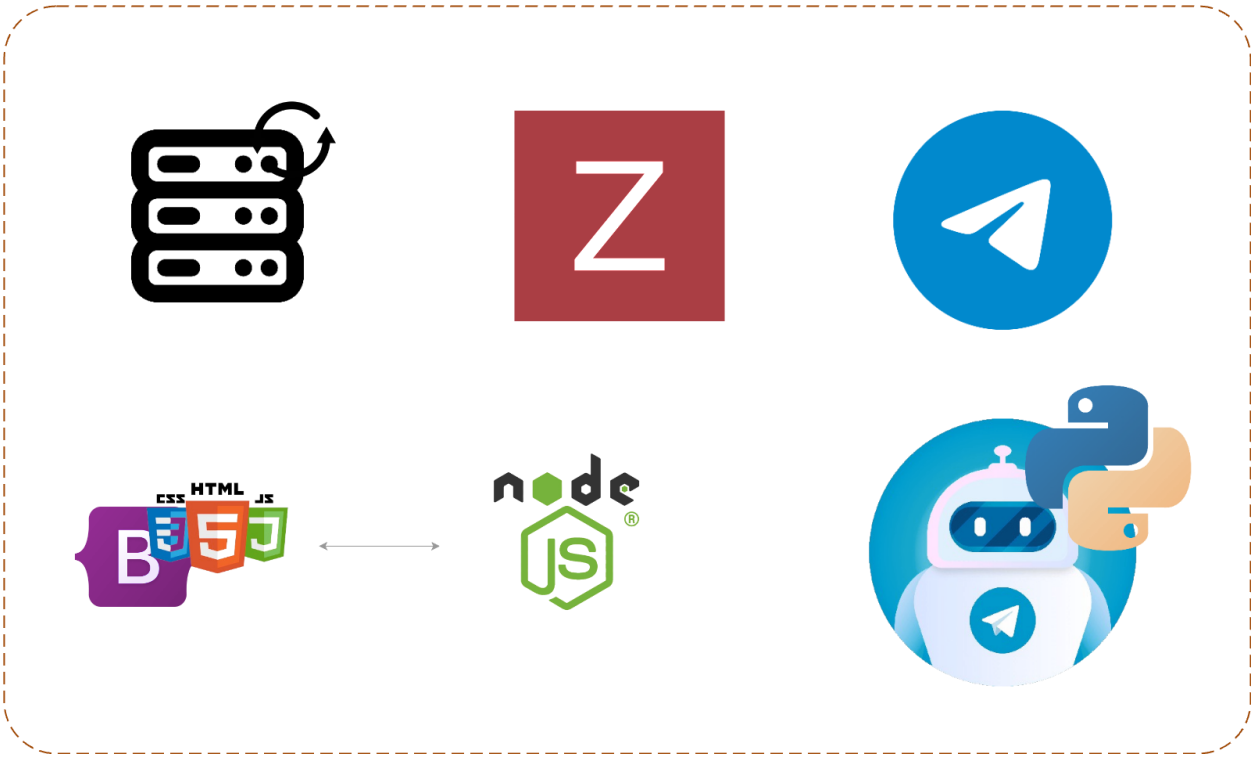
дозволило набагато швидше запустити робочий та налаштований Zabbix, ніж якби це було зроблено напряму на хості. Усього було використано 4 Docker контейнерів :

- Веб сервер для доступу до веб інтерфейсу
- Postgresql як основна база даних
- Zabbix server як основа, яка обробляє всі запити і власне функціонал Zabbix
- Agent, який був доданий для того, щоб Zabbix server міг сам себе моніторити.



Варто зазначити, що це архітектурне рішення використовувалося, тому що його доволі легко створити та воно не потребує ніяких затрат, окрім ресурсів хоста. Воно чудово підходить для демонстрації роботи з Zabbix, але на рівні підприємства цього може бути замало. У такому разі, для розширення архітектури можна використовувати ресурси клауд провайдерів, таких як AWS чи Azure, або ж використовувати готові фізичні пристрої, на яких будуть налаштовані віртуальні машини.

2.2 Архітектура застосунків



2.3 Репортингова система

Застосунок було створено за допомогою HTML/CSS/JS для фронтенду, та NodeJS для бекенду.

2.3.1 Принцип роботи

Застосунок використовує API Zabbix для збору даних про існуючі хости. Після того як користувач вибере відповідного хоста буде створена папка для нього, у яку будуть завантажені і закешовані усі графіки на даний момент. Причини того, що було вибрано саме завантаження графіків, а не їх відтворення на основі даних, які можна отримувати за допомогою API у тому, що даних для кожного графіку доволі багато, а таких графіків може бути в середньому 15-30 на хост, в залежності від його призначення. Це дуже сповільнює роботу застосунку і вимагає чимало ресурсів для обробки всіх графіків. Завантаження графіків займає набагато менше часу і при цьому, кожен з них має вагу 150-200Kb, тобто для кожного хоста така папка буде важити всього декілька Мб, що є малим значенням у сучасній індустрії, де більшість серверів мають десятки чи сотні гігабайт вмістимості.

Після того як користувач переглянув свій звіт для хоста, він може або оновити графіки до поточних, оскільки вони кешуються за замовчуванням, або відправити репорт на свою адресу, де він зможе переглянути його у PDF форматі.

Цей застосунок є демонстрацією можливостей Zabbix і роботою з ним. Створення додатку пройшло ефективно та швидко завдяки чітко описаній документації, яка постійно покращується і оновлюється. Сам API також виявився обширним, оскільки він надає можливість працювати з абсолютно усіма метриками, які доступні у Zabbix, тим самим даючи змогу створювати на його основі будь-які розширення чи окремі додатки, які допоможуть у моніторингу готових систем.

2.3.1 Демонстрація

Рис 2.3.1 Головне меню вибору хоста

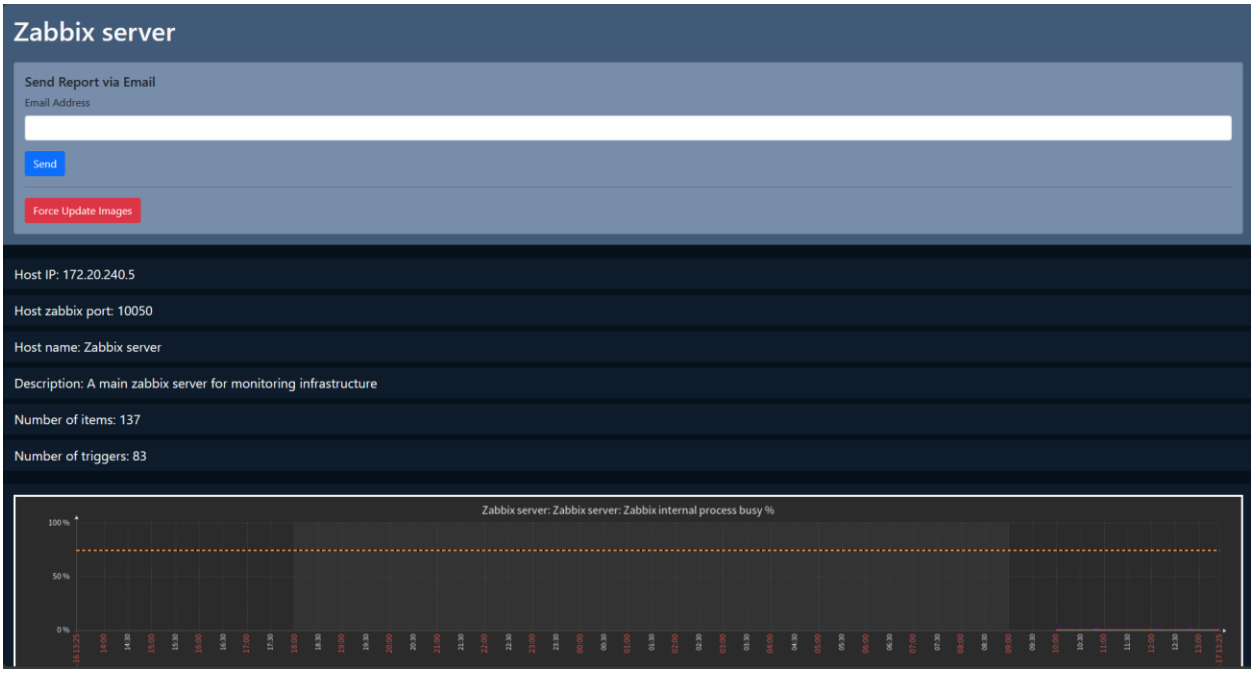


Рис 2.3.2 Інформація та графіки обраного хоста

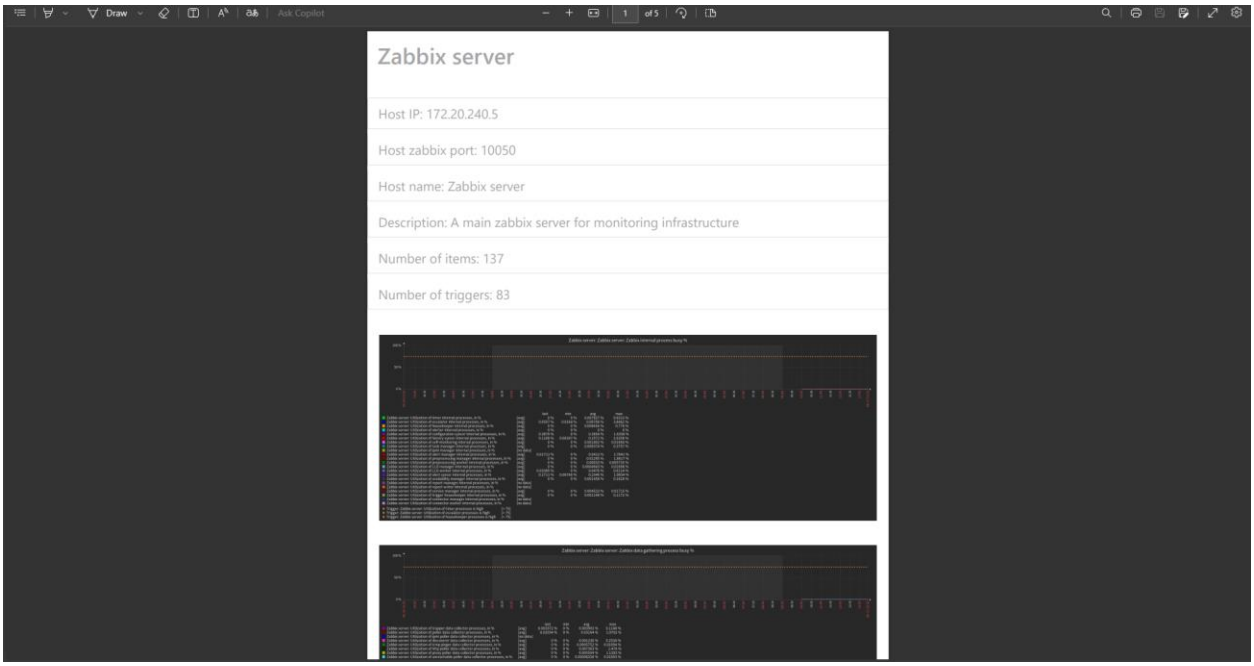


Рис 2.3.3 Вигляд отриманого PDF репорту

2.4 Telegram Bot для обробки сповіщень

Бот працює на основі двох API : офіційний API Telegram для роботи з ботом, і Zabbix API для збору інформації про поточні сповіщення та їх статус, для створення боту використовувалась мова Python.

2.4.1 Принцип роботи

Через певний інтервал часу, бот запитує нові дані про поточні сповіщення у Zabbix, після чого він обробляє отримані дані і групує їх у читабельний вигляд для користувача. Далі бот надає дані про нове сповіщення і очікує до поки воно не зникне у Zabbix, після чого буде надіслано повідомлення про те, що алерт зник.

2.4.1 Демонстрація

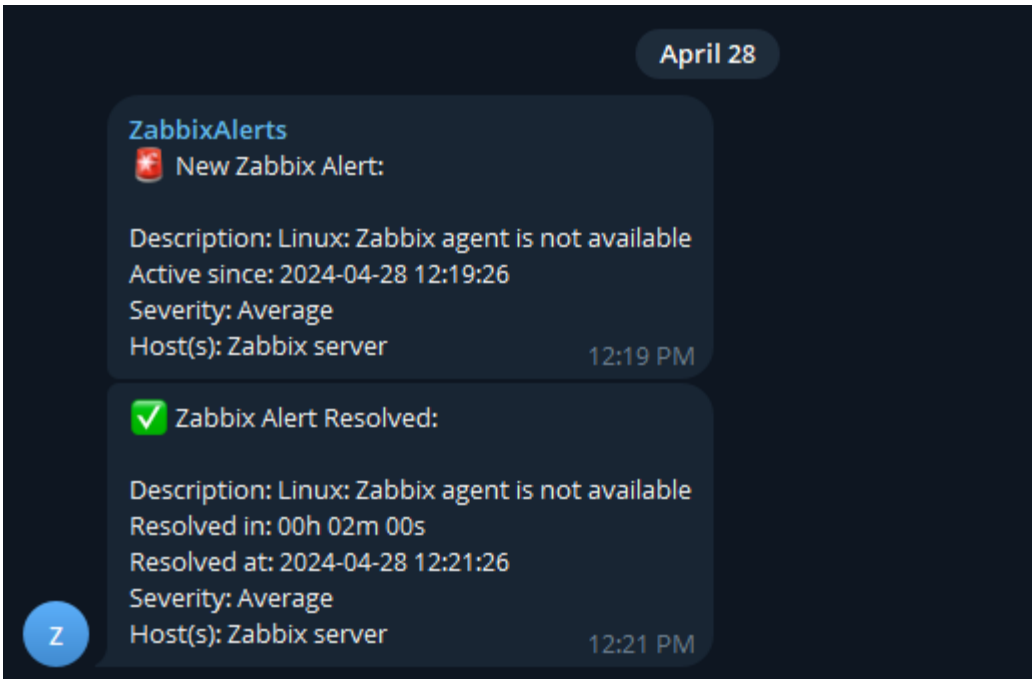


Рис 2.4.1 Роботи бота на прикладі одного з алертів

Висновки

Основною метою курсової робити було дослідження наявних рішень систем моніторингу для підприємства, їх порівняльний аналіз та вибір найкращого варіанту. У результаті цієї частини було виявлено наступні факти :

- На даний момент Zabbix є найкращою системою моніторингу, яку можна обрати у будь-якій ситуації.
- Деякі системи моніторингу кращі за інші, але лише в певному аспекті, якому вони спеціалізуються (Grafana, DataDog).
- Серед трьох найрозвинутіших систем : Zabbix, Prometheus, Nagios – саме Nagios має найкращу систему розширення (плагіни), але вона є складною для опанування та часто не є у потребі. Також матеріали по цій системі часто застарілі та нерелевантні.
- Prometheus – чудова система, яка є другою після Zabbix. Проте, головною проблемою Prometheus є велика кількість мануальної роботи. Це напряму впливає на такі невід'ємні складові як розширення системи для більшої кількості хостів та збільшення об'єму даних, які знаходяться під моніторингом.
- Незважаючи на рейтинг розглянутих систем, кожна з них має серйозний потенціал для розширення, будь-то через вбудований API, чи сторонні ресурси, такі як Zabbix solutions, з готовими рішеннями для великої кількості операційних систем чи сервісів, або ж плагіни (Nagios) і тд.

Для практичної частини було обрано розширення слабкого аспекту усіх оглянутих систем – репортингу. Також, як демонстрацію можливостей роботи

API та розвинутої системи сповіщень, створено телеграм бота для отримання сповіщень про поточні алерти у системі. Під час цієї частини роботи було отримані наступні результати :

- Додаток для репортингу неабияк спростив роботу і системою, оскільки тепер, щоб переглянути основну інформацію про хоста, а також кожен його важливий графік, не потрібно відкривати декілька вкладинок на перегляду потрібною інформації. Також він надає змогу передати цю інформацію іншій людині у зручному форматі. За допомогою саме цього додатку було продемонстровано можливості роботи з API Zabbix.
- Бот, який повідомляє про надходження нових сповіщень також підняв ефективність роботи, оскільки тепер, людина не має знаходитися біля серверу, щоб дізнатися про погіршення роботи хоста, або ж перебирати певне повідомлення на пошті, а одразу побачити усі алерти у спеціально виділеному під це чаті.

При цьому проблем при роботі не виникало, окрім форматування часу, який визначає скільки пройшло з моменту початку алерта, причиною цього було те, що Zabbix API повертав timestamp, який трохи відрізнявся під того, який python може оброблювати за замовченням.

Додаток для репортингу хоча і працює чудово, але все ж має потенціал для покращення і перетворення його у повноцінний комерційний застосунок, або розширення до Zabbix. Можливі покращення :

- Надсилання репорту на декілька різних пошт
- Вибір чіткого періоду, за який надсилається репорт

- Можливість обирати, які саме графіки та інформацію про хост буде відправлено
- Перевірка статусу хоста за допомогою кастомного скрипта (набір стандартних перевірок на здоров'я хоста, такі як вільне місце, навантаження пам’яті чи процесора і тд.)
- Категорування репортів, наприклад, один з них буде описувати сам сервер та його ресурси, в той час як інший буде показувати результат роботи певного скрипта, який буде проводити тести присутніх на сервері сервісів.

Джерела

1. Zabbix API [Електронний ресурс]
2. Documentation (zabbix.com) [Електронний ресурс]
3. Nagios XI Documentation - Nagios Library [Електронний ресурс]
4. Overview | Prometheus [Електронний ресурс]
5. Exploring Prometheus Architecture | by Rapidcode Technologies | Medium [Електронний ресурс]
6. Bot API Library Examples (telegram.org) [Електронний ресурс]
7. python-telegram-bot [Електронний ресурс]
8. Zabbix Prometheus and Nagios : Key Differences to Know [Електронний ресурс]
9. Nagios Exchange [Електронний ресурс]
10. Grafana and Prometheus [Електронний ресурс]