Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інфокомунікацій
(повна назва)
Кафедра Інфокомунікаційної інженерії ім. В. В. Поповського
(повна назва)

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

другий (магістерський)
(рівень вищої освіти)

ГЮІК.ХХХХХХ.005ПЗ
(позначення документа)

Аналіз застосування ELK STACK в системах управління контентом

Analysis of ELK STACK Application in Content Management Systems
(тема)

Виконав: студент 2 курсу, групи ІКІм-19-1
спеціальності 172 Телекомунікації та радіотехніка
(код і повна назва напряму)
освітньої програми Інфокомунікаційна інженерія
(шифр і назва напряму, спеціальності)

Лябах А. С.
(прізвище, ініціали)

Керівник доц. Токар Л. О.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ О. В. Лемешко
(підпис) (прізвище, ініціали)

2020 р.

Харківський національний університет радіоелектроніки

Факультет Інфокомунікацій

Кафедра Інфокомунікаційної інженерії ім. В.В. Поповського

Рівень вищої освіти другий (магістерський)

Спеціальність 172 Телекомунікації та радіотехніка

Спеціалізація Інфокомунікаційна інженерія

(код і повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____»_____ 20 ___ р.

ЗАВДАННЯ

НА АТЕСТАЦІЙНУ РОБОТУ

студентові Лябаху Артему Сергійовичу

(прізвище, ім'я, по батькові)

1.Тема роботи: «Аналіз застосування ELK STACK в системах управління контентом», затверджена наказом по університету від « 20 » жовтня 2020 р. № 1396

2. Термін подання студентом роботи до екзаменаційної комісії 24.12.2020р.

3. Вихідні дані до роботи:

1. Модель Erlang B з параметрами - навантаження контактного центру A = 5 Erl, 60 вхідних дзвінків за одну годину, середнім часом - 5 хвилин (0,084 години), ймовірність відхилення викликів P B від 0 до 80%;

2. Модель Erlang C з параметрами - навантаження контактного центру A = 5 Erl, 60 вхідних дзвінків за одну годину, середнім часом - 5 хвилин (0,084 години), ймовірність відхилення викликів P B від 0 до 60%;

3. Система білінгу ELK Stack з інструментами Ansible, Vagrant, Visual Studio Code

4. Перелік питань, що потрібно опрацювати в роботі:

1. Огляд технологій які використовуються для білінгу в Телекомунікаціях

2. Аналіз трафіку в контактних центрах

3. Аналіз моделі Erlang B та Erlang C

4. Аналіз інструментів для розробки ELK в системах управління контентом

5. Огляд технологій які використовуються в ELK

6. Розробка програмного забезпечення для автоматичного розгортання універсальної системи білінгу ELK

7. Аналіз ефективності програмного забезпечення ELK Stack в порівнянні з існуючими рішеннями

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів): презентація на слайдах

1. Вступ
2. Тема та мета
3. Модель Erlang B
4. Результати аналізу моделі Erlang B
5. Модель Erlang C
6. Результати аналізу моделі Erlang C
7. Необхідність ведення журналів системи у сучасному світі
8. Принцип роботи ELK
9. Результаты аналізу інструментів для розробки ELK в системах управління контентом
10. Результати швидкості розгортання ELK
11. Розгортання тестового середовища за допомогою Vagrant
12. Елементи Ansible
13. Ansible playbooks
14. Приклад Ansible ролі EK
15. Автоматичне розгортання ELK
16. Сервіс Kibana, як результат успішного встановлення та конфігурації
17. Висновки

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Основна частина	доцент Токар Л.О.		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Огляд технологій які використовуються для білінгу	18.10.2020р.	Виконано
2	Аналіз трафіку в контактних центрах	21.10.2020р.	Виконано
3	Аналіз моделі Erlang B та Erlang C	3.11.2020р	Виконано
4	Аналіз інструментів для розробки ELK в системах управління контентом	15.11.2020р.	Виконано
5	Огляд технологій які використовуються в ELK	1.12.2020р.	Виконано
6	Розробка програмного забезпечення для автоматичного розгортання ELK	22.12.2020р.	Виконано

Дата видачі завдання 01 09 2020р.

Студент _____
(підпис)

Керівник роботи _____ к.т.н., доц. Токар Л.О
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 78 с., 27 рис., 10 табл., 13 джерел.

ELK, ANSIBLE, АВТОМАТИЗАЦІЯ, МОНІТОРИНГ, VAGRANT

Об'єкт дослідження – системи білінгу та моніторингу.

Предмет дослідження – методи автоматичного розгортання ELK Stack.

Мета роботи – розробка та аналіз ефективності програмного забезпечення, за допомогою якого можливо автоматичне встановлення та налаштування базової конфігурації ELK Stack для необхідних серверів.

Методи досліджень – аналіз, формалізація, порівняння та розробка програми для автоматичного розгортання ELK.

Для ведення статистики, моніторингу, аналізу даних, виставлення рахунків, та пошуку неполадок, ведуться журнали системи та іншого програмного забезпечення. Одним з найкращих інструментів для цього є ELK. Проте налаштування цього стеку є не простою задачею.

Таким чином, актуальною є задача розробки програмного забезпечення для автоматичного встановлення та конфігурації ELK Stack, яке дозволить заощадити значну частину часу та грошей.

У роботі розроблено програмне забезпечення для швидкого встановлення та налаштування ELK Stack. Проаналізовано набір компонентів ELK Stack та доведено, що ці інструменти мають місце в системах білінгу. Проведено аналіз ефективності розробленого програмного забезпечення в порівнянні з існуючими.

ABSTRACT

The report contains: 78 p., 27 fig., 10 tables, 13 sources.

ELK, ANSIBLE, AUTOMATION, MONITORING, VAGRANT

A research object is billing and monitoring systems.

The subject of research is the methods of automatic deployment of ELK Stack.

An aim of work is to develop and efficiency analysis software that can automatically install and configure the basic configuration of ELK Stack for the required servers.

Methods of researches are analysis, formalization, modeling and program development for ELK automatic deployment.

System logs and other software are maintained for statistics, monitoring, data analysis, billing, and troubleshooting. One of the best tools for this is ELK. However, setting this stack is not an easy task.

Thus, the development of software for the automatic installation and configuration of ELK Stack is a topical task, which will save a significant amount of time and money.

Software for quick installation and configuration of ELK Stack has been developed. The ELK Stack component set has been analyzed and it has been shown that these tools occur in billing systems. The analysis of efficiency of the developed software in comparison with existing is carried out.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

АТС - автоматична телефонна станція
КСХ – коефіцієнт стоячої хвилі
ЦК – контакт-центр
ACD – automatic call distribution
API – application programming interface
AWS – amazon web services
CC – call control
CDR – call detail record
CRM – customer relationship management
DWH - data warehouse
ELB – elastic load balancing
ELK – elasticsearch, logstash, kibana
ERP – enterprise resource planning
FTP – file transfer protocol
GoS – grade of service
HTTP – hypertext transfer protocol
IDS – intrusion detection system
IMSI – international mobile subscriber identity
IN – intelligent network
ISDN – integrated services digital network
IVR – interactive voice response
IoT – internet of things
MMS – multimedia messaging service
MSISDN – mobile subscriber integrated services digital number
NIS – network information service
OSS – operation support system
QoS – quality of service
SEO – search engine optimization

TAP3 - transferred account procedure 3

TCP – transmission control protocol

UDP – user datagram protocol

ВСТУП

На сьогоднішній день вже не можливо уявити своє життя без різних технологій та сервісів, які полегшують нам життя. Беручи просту статичну веб сторінку, до інтернет магазинів та різного виду програмного забезпечення Ці сервіси являються достатньо не простими системами, які розробляють та підтримують велика кількість людей. Також, цими сервісами користується надзвичайно велика кількість людей. Звичайно, самостійне існування без підтримки є неможливим. Для ведення статистики, збору корисних даних, виставлення рахунків, та пошуку неполадок, ведуться журнали кожного програмного забезпечення. Кожна подія, повинна бути записана в журнал, кожен рух користувача. Це означає, що за години може бути згенеровано десятки терабайт цих журналів. З цього випливає, що журнали стають громіздкими. Швидкий пошук необхідної інформації стає дедалі важчим.

Існують декілька допоміжних програм, які оперативно можуть працювати з такою кількістю даних. Одним з найпопулярніших рішень є ELK Stack. Але основна проблема полягає в тому, ще цей комплекс програм потребує достатньо високий рівень знань для встановлення та конфігурації.

Мета роботи – розробка та аналіз ефективності програмного забезпечення, за допомогою якого можливо автоматичне встановлення та налаштування базової конфігурації ELK Stack для необхідних серверів.

1 ОГЛЯД ТЕХНОЛОГІЙ ЯКІ ВИКОРИСТОВУЮТЬСЯ ДЛЯ БІЛІНГУ В ТЕЛЕКОМУНІКАЦІЇЯХ

1.1 Необхідність використання білінгу с системах надання послуг

Телекомунікації та ділові функції сьогодні йдуть рука об руку. Це є причиною того, що більшість компаній застосовують ефективне програмне забезпечення для виставлення рахунків для телекомунікацій, щоб поліпшити щоденні ділові операції. Програмне забезпечення відіграє важливу роль в управлінні витратами та витратами. Сила рішень щодо виставлення рахунків у сфері телекомунікацій полягає в управлінні витратами. Крім того, вони можуть включати функції, спеціально розроблені для забезпечення унікальних вимог вашого бізнесу.

Рішення щодо виставлення рахунків за телекомунікаційні послуги є основою фінансових операцій у галузі бездротових телефонів. Ці рішення допомагають відстежувати загальний час використання клієнта. Отже, замовник виставляє рахунок лише за той час, який він використовує. Програмне забезпечення також створює рахунки, що надаються клієнтам. Включення різних модулів в єдину програму допомагає компанії управляти всіма своїми грошовими ресурсами.

Використання рішень OSS для виставлення рахунків у телекомунікаційних компаніях не обмежується лише телекомунікаційною галуззю. Будь-яка компанія, яка використовує телекомунікаційні послуги тим чи іншим способом, може використовувати ці рішення для відстеження активів, використання та витрат. Ви також можете підготувати інклюзивний звіт на основі параметрів, які вказує користувач. Аналіз та експертиза використання телекомунікаційних послуг, а також підготовка звітів дозволяє зацікавленим сторонам переглянути результати. Переглянувши результати, можна внести необхідні зміни для підвищення прибутку компанії.

Однією з важливих переваг телекомунікаційних рішень для виставлення рахунків є їх здатність зменшувати або усувати повторювані помилки, що виникають під час виставлення рахунків. Ми не можемо заперечувати, що навіть найменші

помилки можуть в кінцевому підсумку коштувати компанії чимало. Іноді помилки в рахунках виникають через поганий зв’язок або невідповідну інформацію. Крім того, використання різних програм для фінансової діяльності призводить до більшої кількості помилок.

Інтеграція різних фінансових операцій в єдину програму надає однорідності процесу. Це також полегшує захист окремих модулів, оскільки вони мали б обмежений доступ. Об’єднання різних модулів виставлення рахунків у межах одного програмного забезпечення допомагає забезпечити зв’язок між цими модулями. Це не тільки призводить до більш точного виставлення рахунків, але й зменшує ймовірність помилок.

Утримання клієнтів часто визначає найефективніші бізнес-стратегії. Сучасна телекомунікаційна компанія робить усе можливе, щоб покращити свій імідж в очах своїх клієнтів.

Справжня мета програми виставлення рахунків для телекомунікацій - врахувати кожен байт інформації, що протікає по мережі. У світлі нинішньої ситуації важливо зрозуміти, чому телекомунікаційні компанії повинні бути конкретними щодо своїх вимог щодо виставлення рахунків. Деякі важливі фактори включають:

- зміни у телекомунікаційній галузі: послуги, що надаються телекомунікаційною компанією, більше не обмежуються вашими стаціонарними зв’язками або стільниковими телефонами. В даний час кожен клієнт має смартфон, який підтримує різноманітні програми. Тому постачальнику послуг потрібно відстежувати обсяг даних, які використовуються кожним клієнтом. Крім того, кількість даних має пряме відношення до рахунку клієнта та, зрештою, доходу компанії;
- потреба у більшій робочій силі: кількість послуг, що надаються телекомунікаційними компаніями, продовжує зростати. Отже, компаніям також потрібно збільшити робочу силу, зайняту ними. Зрештою, вам потрібно керувати клієнтською базою, яка складається з клієнтів, розташованих в різних частинах світу. Більше того, вимоги кожного замовника унікальні. У такій ситуації використання телекомунікаційних програм виставлення рахунків допомагає організації контролювати свої доходи та підтримувати точну базу даних;

- забагато постачальників послуг: минули часи, коли конкретна телекомунікаційна компанія контролювала та монополізувала свою територію. В даний час не рідкість того, що нова компанія починає свою діяльність через день. Таким чином, клієнт цілком розуміє, що телекомунікаційна компанія потрібна йому більше за все. Ось чому він може перейти до нового постачальника послуг, навіть коли стикається з найменшими проблемами. Все, що йому потрібно зробити, - це зв’язатись із телефонним кол-центром.

Нові учасники телекомунікаційного домену стикаються з жорсткою конкуренцією як з боку аналогів, так і з боку провідних постачальників послуг.

Якщо клієнт не задоволений наданою йому послугою, він може розглянути інші доступні йому варіанти. Компанія може максимально використати рішення для виставлення рахунків за допомогою телекомунікацій, увімкнувши багатомовну підтримку та виставлення рахунків у реальному часі. Нарешті, але не менш важливим є те, що компанії повинні пропонувати видатні послуги з виставлення рахунків, щоб вижити [1].

1.2 Білінг в системах надання послуг

Надсилання голосу, даних, зображення, факсу тощо з однієї точки в іншу за допомогою електронних носіїв називається телекомунікацією і, коротше, "телекомунікацією". Приклади включають телефон, радіо, телебачення та Інтернет. До середовища передачі належать провід (мідний), волоконна оптика, ефір (безпроводовий), радіовежі, мікрохвильова піч, супутник тощо.

Плата за оренду - це плата, яку щомісяця беруть із клієнтів за надану послугу. Наприклад, щомісячна плата за телефон складе 5,00 доларів США незалежно від того, використовуєте ви це чи ні.

Плата за користування - це плата, що береться з клієнтів на основі використання послуги. Наприклад, з вас платитимуть за всі дзвінки, які ви зробили, або дані, завантажені за допомогою вашого телефону.

Окрім щомісячної плати за оренду та користування, оператори можуть також стягувати плату за ініціювання, встановлення, припинення або припинення послуги.

Телекомунікаційний білінг - це процес збору використання, його агрегування, застосування необхідних платежів за користування та оренду та, нарешті, формування рахунків-фактур для клієнтів. Процес виставлення рахунків за телекомунікації також включає отримання та запис платежів від клієнтів

Можуть існувати дуже тяжкі сценарії надання даних, які важко було б обробити вручну. На ринку програмного забезпечення доступні найсучасніші системи виставлення рахунків, які дозволяють дуже ефективно вирішувати завдання з виставлення рахунків та надають велику кількість можливостей постачальникам послуг пропонувати свої послуги з різною структурою цін.

Системи виставлення рахунків часто розглядаються як дебіторська заборгованість, оскільки система виставлення рахунків допомагає у збиранні (отриманні) грошей від клієнтів. Системи виставлення рахунків також є частиною кредиторської заборгованості (для розрахунків між операторами), оскільки клієнти часто використовують послуги інших компаній, такі як безпроводовий роумінг, міжміські та завершення дзвінків через інші мережі.

Системи виставлення рахунків - це висококласні, надійні та дорогі програмні засоби, що забезпечують різні функціональні можливості. Ось список найважливіших функцій:

- оцінка та виставлення рахунків - це включає оцінку використання продуктів чи послуг та складання щомісячних рахунків;
- обробка платежів - передбачає розміщення платежів клієнта на його / її рахунку;
- кредитний контроль та стягнення - він передбачає переслідування непогашених платежів та вжиття відповідних дій для їх стягнення;
- спори та коригування - він включає в себе реєстрацію суперечок клієнта щодо своїх рахунків та створення коригування для повернення спірної суми з метою врегулювання суперечок;
- послуги передоплати та післяплати - це передбачає підтримку баз клієнтів з передплатою та післяплатою;
- багатомовні та декілька валют - підтримка багатомовних та кількох валют потрібна, якщо бізнес розповсюджений по всьому світу та має багатонаціональних клієнтів, або якщо це вимагають державні норми;

- розрахунки між операторами - це передбачає розподіл доходу між перевізниками, які надають послуги споживачам один одного;
- товари та послуги - це передбачає надання гнучкого способу обслуговування різноманітних товарів та послуг та їх продажу окремо або в упаковці;
- заявки на знижки - це передбачає визначення різних схем знижок з метою зменшення відтоку клієнтів та залучення та збільшення клієнтської бази.

Коли ви детально визначаєте предмет рахунку, він ускладнюється. Давайте розглянемо широко поширені типи виставлення рахунків:

- виставлення рахунків перед оплатою - механізм виставлення рахунків, коли клієнт сплачує заздалегідь, а після цього починає користуватися послугою. Зазвичай передплачені клієнти не отримують рахунків-фактур, і плата за них здійснюється в режимі реального часу за допомогою високодоступних систем виставлення рахунків під назвою „IN" (Інтелектуальна мережа);
- виставлення рахунків після оплати - це звичайна оплата, яка надходить протягом багатьох років. Тут клієнти купують товари та послуги та використовують їх протягом місяця, а до кінця місяця рахунки-фактури формуються постачальником послуг та надсилають ці рахунки клієнтам для здійснення належної оплати;
- виставлення рахунків за взаємозв'язок: оператор мережі, як правило, несе фінансову відповідальність за послуги, що надаються своїм клієнтам іншими мережами, незалежно від того, оплачує клієнт послугу чи ні. Виставлення рахунків за взаємозв'язок пов'язане з взаємозв'язком між операторами або десь так званими розрахунками партнерів;
- плата за роумінг - коли клієнт переходить з зони покриття одного оператора мережі в зону покриття іншого оператора, перший оператор сплачує незначні збори другому оператору за надання послуг своїм клієнтам. Такі збори сплачуються за допомогою рахунків у роумінгу. Це врегулювання здійснюється відповідно до протоколу TAP3, про що ми поговоримо у наступних розділах;
- конвергентний білінг - конвергентний білінг - це інтеграція всіх платежів за послуги в єдиний рахунок-фактуру клієнта. Конвергентний білінг означає створення єдиного уявлення про клієнта та всі послуги (мобільні, стаціонарні, IP, тощо), що надаються цьому клієнту [2].

1.3 Архітектура систем білінгу в телекомунікаціях

На рис. 1.1 показано типову архітектуру системи виставлення рахунків. У цьому розділі буде коротко представлено всі системи взаємодії, починаючи зверху вниз.

Рисунок 1.1 – Архітектура систем білінгу в телекомунікаціях

Це перша система, де замовлення клієнта фіксується, а клієнт створюється в системі. CRM означає «Управління відносинами з клієнтами», а OMOF - «Управління замовленнями» та «Виконання замовлення».

Існують такі системи, як Siebel, яка забезпечує модулі для CRM, а також OMOF. Система CRM зберігає інформацію, пов'язану з клієнтами, разом із продуктами та послугами. Модуль OMOF відповідає за відстеження замовлення, починаючи від його створення і закінчуючи його завершенням. Тут ми маємо два варіанти:

- CRM (управління взаємовідносинами з клієнтами) / OMOF (управління замовленнями та виконання замовлення) контактує із системою виставлення рахунків, а система виставлення рахунків - системою забезпечення, щоб надавати послуги та систему інвентаризації мережі, а також призначати номери телефонів або IP-адреси тощо;
- другою можливістю може бути те, що система CRM / OMOF сама контактує із системою забезпечення для надання послуг та системи інвентаризації мережі, а також для призначення номерів телефонів або IP-адрес тощо.

Ця система приймає команди з Платіжної системи або CRM / OMOF для активації, деактивації та призупинення послуг. Обидві архітектури є дійсними і залежать від того, як архітектор розробляє всю установку.

Після прийому команд забезпечення, ця система контактує з основною мережевою системою, щоб активувати, деактивувати або призупинити послуги. Після успішного забезпечення ця система надсилає відповідь назад або до системи виставлення рахунків, або до системи CRM залежно від того, хто надіслав їй останню команду.

Ця система підтримує всі мережеві ідентифікатори, такі як телефонні номери, MSISDN, IP-адреси, адреси електронної пошти тощо, і технічно вона називається Network Inventory System.

Залежно від архітектури системи, CRM / OMOF або Billing System звертаються до NIS для отримання необхідного ідентифікатора мережі та присвоюють його клієнту під час створення замовлення.

Ця система відповідає за підтримку життєвого циклу мережевих ідентифікаторів, який починається з доступних, а потім проходить через різні етапи, такі як активація, призупинення, припинення, карантин і знову доступність.

Система посередництва збирає CDR з різних елементів мережі в різних форматах. Різні мережеві елементи генерують CDR у форматі ASN.1, а деякі мережеві елементи мають власний формат CDR.

Система медіації обробляє всі CDR та перетворює їх у формат, сумісний із низхідною системою, яка, як правило, є системою рахунків. Система медіації застосовує різні правила щодо CDR для їх обробки; наприклад, система посередництва позначає міжнародні дзвінки на основі набраного номера (B-Number), так само, як система посередництва позначає дзвінки в мережі на основі A-Number та B-Number.

Може бути вимога відфільтрувати всі дзвінки, тривалість яких триває менше 5 секунд, найкраще відфільтрувати такий тип дзвінків буде на рівні Медіаційної системи. Точно так само, якщо в CDR потрібна додаткова інформація, що є критично важливою для виставлення рахунків, тоді Mediation System допоможе у наданні такої інформації на основі деяких інших атрибутів, доступних у CDR.

Після того, як зібрані CDR обробляються, система посередництва надсилає всі CDR до системи виставлення рахунків за допомогою FTP, оскільки зазвичай системи посередництва та виставлення рахунків працюють на різних машинах.

Ця система для виставлення рахунків є нижчою за ціною та зазвичай зберігає багато історичних даних, пов’язаних із клієнтами. Система виставлення рахунків передає різну інформацію про клієнтів у систему DWH. Ця інформація включає використання послуг, рахунки-фактури, платежі, знижки та коригування тощо

Вся ця інформація використовується для формування різних типів управлінських звітів, а також для бізнес-аналітики та прогнозування.

Система DWH завжди призначена для роботи над масовими та величезними даними, і якщо є потреба у будь-якому невеликому звіті, то завжди варто генерувати його із системи виставлення рахунків, а не зловживати DWH для невеликих завдань [3].

1.4 Білінг, продукт як послуга

Припустимо, такий оператор зв'язку, як Airtel, хоче створити власну систему виставлення рахунків. Тоді Airtel спочатку повинен був би визначити свої продукти

та послуги своїм відділом продажів та маркетингу, перш ніж перейти до створення системи виставлення рахунків.

Продукт - це логічна або фізична особа, яку оператори можуть продати кінцевому споживачу. Це може бути мобільний телефон, підключення до Інтернету, підключення голосових дзвінків, VPN, відео на вимогу, підключення цифрового телебачення тощо.

Продукт може мати щомісячну оренду, яку ми також називаємо періодичною оплатою. Продукт може бути продуктом, що генерує використання, або продуктом, що не генерує використання. Продукт, що генерує використання, іноді називають продуктом, що генерує події, а продукт, що генерує невикористання, - товаром, що не генерує події.

Наприклад, голосовий дзвінок, який постачається разом із номером телефону, є продуктом, що генерує використання, оскільки він генерує використання, коли кінцевий споживач використовує цей продукт для здійснення голосового дзвінка. Простий телефонний апарат без підключення - продукт, що не генерується, і він може бути наданий клієнту лише на основі щомісячної орендної плати. Тож навіть якщо клієнт не користується ним, він повинен платити щомісячну оренду.

Коли ми говоримо про них з маркетингової точки зору, як такої немає різниці між товарами та послугами, оскільки більшість випадків обидва вони взаємозамінні між собою різними експертами з питань білінгу та маркетингу.

Простіше кажучи, оператор використовує свою продукцію для надання голосових послуг своїм клієнтам. Міжнародний дзвінок можна назвати послугою, що надається за допомогою з'єднання голосового дзвінка. Іншим прикладом може бути дзвінок на 800 номерів, який може бути доступний чи не доступний через певного оператора, очікування дзвінка, переадресація дзвінка, можна сказати, послуга, що надається моделлю телефонного апарату або оператором.

Продукти - це предмети, які клієнти можуть купити як безпосередньо, так і здати в оренду. продукти можуть бути:

- реальні предмети (наприклад, мобільний телефон);
- послуги (наприклад, послуга очікування дзвінка в телефонній системі);
- більш абстрактні поняття (наприклад, угода про рівень обслуговування).

Супутні товари можна об’єднати у сімейство продуктів. Можливо декілька рівнів продуктів, тому товар може бути одночасно батьком і дитиною.

Крім того, кожна родина продуктів може мати більше одного батьківського товару, якщо це потрібно. Приклади сімейств товарів:

- послуги телефонії;
- кабельне телебачення;
- інтернет;
- виділена лінія.

Багато разів оператори об’єднують більше однієї продукції в одну групу і продають їх у вигляді цілої упаковки. Існують системи виставлення рахунків, які підтримують поєднання різних видів продуктів у вигляді упаковки. Його можна запропонувати за зниженою ціною.

Пакети дозволяють пропонувати товар покупцеві за зниженою ціною, якщо він береться як частина пакету. Кожна упаковка може складатися з будь-якої кількості продуктів, і ці продукти можуть бути взяті з більш ніж однієї групи продуктів.

Цей тарифний план на упаковку товару зазвичай відрізняється від порівняльного (тобто безпакетного) цінового плану, оскільки таким чином компанія пропонує клієнту знижку на придбання повного пакета. Однак це не є обов’язковим, оскільки для товару може бути призначений один із його звичайних тарифних планів у межах упаковки.

Продукт може мати ряд пов’язаних з ними ознак. Атрибути товару дозволяють зберігати інформацію про окремі екземпляри товару там, де відповідна інформація відрізняється між типами товару. Наприклад, продукт платного телебачення може мати атрибут, що записує номер його телевізійної приставки.

Крім того, продукт мобільного телефону може потребувати атрибутів для запису Міжнародної ідентифікації абонента мобільного зв'язку (IMSI) та Міжнародного ISDN-номера мобільної станції (MSISDN).

Продукт може мати кілька типів подій, пов’язаних з ним. Ці типи подій регулюють події, які може генерувати продукт. Наприклад, мобільний телефон може мати такі події, як голосові дзвінки та послуги обміну повідомленнями. З одним телефонним пристроєм може бути набагато більше типів подій, і оператор може стягувати плату з кінцевого споживача за кожну подію, згенеровану клієнтом.

Після того, як ваш відділ маркетингу доопрацює всі продукти, послуги, пакети та супутні ціни, вони будуть налаштовані в системі виставлення рахунків. Різні системи виставлення рахунків забезпечують різні рівні гнучкості визначення продуктів та їх ієрархій з точки зору батьків, дітей та онуків. Деякі системи є досить гнучкими, щоб підтримувати пакети та пакети, а деякі забезпечують обмежені функції, пов'язані з пакетами та зниженими цінами.

Деякі системи ведуть каталоги товарів окремо від каталогів цін, щоб забезпечити кращий модульний підхід, а деякі системи виставлення рахунків взагалі поєднують опис товарів, їх особливості та пов'язані ціни.

Після того, як усі продукти, послуги, пакети та їх події будуть налаштовані в системі виставлення рахунків, наступним кроком є визначення їх ціни оренди та використання, про що ми розповімо в наступному розділі [3].

1.5 Планування тарифів

Відділ маркетингу компанії телекомунікаційного оператора наполегливо працює над визначенням плати за оренду та користування різними продуктами та послугами. Ці збори визначаються з урахуванням інших конкурентів та правил. Взагалі кажучи, існує два типи тарифів, які також називаються тарифними або ціновими планами, залежно від термінології, що використовується в різних системах виставлення рахунків. Можуть існувати різні типи платежів, що застосовуються за товар і пов'язані послуги. Для даного товару оператор може визначити одну або декілька з наведених нижче тарифів, але вони не обмежуються лише цими зборами, можуть існувати деякі інші види збору, залежно від країни, місцезнаходження та ділової ситуації:

- плата за ініціювання продукту - це одноразова плата, яку клієнт може взяти як частину встановлення, активації, обслуговування або встановлення з'єднання;
- періодичні платежі за товар - це плата, яка може застосовуватися щомісяця, раз на два місяці або щороку як оренда продукту та наданих послуг;
- збори за припинення використання товару - це плата, яка може застосовуватися при припиненні дії товару та послуги;

- плата за призупинення продукту - це збори, які можуть застосовуватися, якщо продукт зупинено з якихось причин; наприклад, несплата;
- періодичні збори за призупинення дії продукту - може існувати вимога періодично стягувати плату з клієнта, навіть якщо з якоїсь причини клієнта призупинено;
- плата за повторну активацію продукту - якщо припустити, що продукт було призупинено з якоїсь причини, і тепер йому потрібна його активація, оператор може застосувати плату за повторну активацію цієї послуги;
- плата за використання продукту - це найважливіший вид стягнення, який застосовується на основі використання послуги. Наприклад, дзвінок за хвилину або за секунду, завантаження даних на МБ тощо.

Усі вищезазначені збори визначаються (тобто конфігуруються) в різних каталогах тарифів, що включають або виключають відповідний податок, залежно від законодавства. Ці каталоги відрізняються від системи виставлення рахунків до системи виставлення рахунків. Деякі білінгові системи зберігають усі ціни в єдиному каталозі, а деякі білінгові системи зберігають плату за користування окремо від інших платежів. Ці каталоги зберігаються в білінговій системі, але вони також доступні для зовнішньої системи, так що до різних тарифів можуть застосовуватися Усі ціни визначаються залежно від продуктів та їх упаковки. Можуть бути різні комбінації продуктів, що дають різні ціни в різних упаковках [4].

1.6 Вимоги до системи виставлення рахунків

Система виставлення рахунків повинна складатися з ряду незалежних додатків, які, коли працюють разом, називаються системою виставлення рахунків. Хороша система виставлення рахунків повинна забезпечувати наступні основні функції з глибокою гнучкістю

- управління клієнтським інтерфейсом - система виставлення рахунків повинна мати можливість обробляти ініційовані клієнтом контакти, контролювати вихідні контакти із клієнтами та керувати життєвим циклом контактів;
- управління замовленнями - це основна функціональність, яка повинна бути доступна у типовій системі виставлення рахунків. Система виставлення

рахунків повинна бути достатньо здатною фіксувати замовлення на товари та послуги та управляти життєвим циклом введення замовлення та контролювати життєвий цикл виконання замовлення;

- продажі та маркетинг - задовільна система виставлення рахунків повинна відповідати на запити замовника, обробляти комісії, надавати підтримку продажів, відстежувати перспективи, керувати кампаніями, аналізувати ефективність товару та купувати кілька житлових одиниць;
- тарифні плани та рейтинг - білінгові системи повинні керувати різноманітними товарами та послугами, різними тарифними планами, пов'язаними з цими продуктами та послугами, і повинні забезпечувати гнучкі способи оцінки використання, створеного цими продуктами та послугами.
- знижка - білінгова система повинна мати можливість надавати різні види знижок при різному використанні та оренді;
- виставлення рахунків - важливо, щоб система виконувала запит на виставлення рахунків, формувала рахунки, обробляла депозити, здійснювала адміністрування рахунків, зберігала інформацію про податки та збори та обробляла фінансову інформацію;
- кредитний контроль та інкасо - система виставлення рахунків повинна контролювати використання та дохід, присвоюючи різні класи кредитування різним клієнтам. Система повинна підтримувати збір платежів та застосовувати їх до різних рахунків-фактур;
- багатомовна підтримка - багатомовна підтримка передбачає надання рахунків-фактур та обслуговування клієнтів різними мовами;
- кілька валют - кілька валют, що використовуються в різних країнах, можуть ускладнити систему виставлення рахунків, оскільки система виставлення рахунків та обслуговування клієнтів повинна мати можливість реєструвати та обробляти в одиницях декількох валют;
- управління доходами партнерів - управління доходами партнерів - це розподіл доходу між перевізниками, які надають послуги споживачам один одного;
- обробка проблем - система виставлення рахунків повинна також мати можливість керувати входом квитка про несправності, координувати закриття квитка про неполадки та відстежувати хід вирішення проблеми з білетами;

- звітність про результативність - задовільна система забезпечить звітність про ефективність, забезпечить звітність про якість послуг (QoS), створить звіти про управління та генерує нормативні звіти;
- встановлення та обслуговування - система також повинна забезпечувати планування робочої сили та керувати діяльністю, що виконується в приміщенні замовника;
- аудит та безпека - система виставлення рахунків повинна проводити аудит даних та перевірку цілісності. Безпечна система завжди бажана для оператора [4].

Окрім вищезазначених функцій, хорошою системою виставлення рахунків повинна бути

- прискорення часу випуску на ринок нових запусків послуг;
- забезпечення конвергентного перегляду клієнтів та продуктів;
- підтримка економічної архітектурної масштабованості;
- забезпечення управління та врегулювання відносин з партнерами;
- зниження загальної вартості володіння.

1.7 Процеси оцінки білінгу в телекомунікаційній сфері

Ставка - це ціна за настання події. Приклади тарифів включають плату за час телефонного дзвінка: Наприклад: "0,40 INR за 1 хвилину" або певну кількість. Наприклад: 10,00 індійських рупій за завантаження 1 МБ або за плату за якість послуги.

Подія - це єдиний випадок використання продукту або послуги. Події фіксуються елементами мережі у вигляді CDRS чи UDR і передаються в систему виставлення рахунків для оцінки та виставлення рахунків.

Рейтинг - це процес визначення нарахування чи ціни окремих подій. Наприклад, ціна дзвінка за 2 хвилини становить 0,80 INR за тарифом 0,40 INR за хвилину.

Система оцінки, яка є частиною системи виставлення рахунків, виконує цю функцію оцінки.

Рейтинговий механізм отримує події у вигляді записів даних, які називаються Записами детальних викликів (CDR) або Детальними записами використання (UDR), що описують використання товару / послуги. CDR - це рядок даних, що містить інформацію про дзвінок, таку як дата та час дзвінка, тривалість дзвінка, виклик, сторона, що викликається тощо, які використовуються для оцінки подій.

- прийняття CDR від системи посередництва або інших постачальників послуг або роумінгових партнерів у разі використання роумінгу;
- перевірка CDR та усунення дублікатів записів. Ці повторювані події зберігаються в таблиці бази даних для подальшої перевірки;
- визначити рахунок клієнта, з якого потрібно стягнути плату за подію. Тут процес Rate визначає джерело події (номер мобільного телефону або IP-адресу тощо) та перевіряє базу даних, щоб перевірити, чи пов'язане це джерело події з яким-небудь обліковим записом. Цей етап називається керівництвом подіями;
- якщо подією неможливо керувати, ця подія буде відхилена і може бути віднесена до категорії напружень. Ці відхилені події зберігаються в таблиці бази даних для подальшої перевірки;
- для розрахунку вартості / ціни події відповідно до рейтингового тарифу (також називаного тарифним планом);
- застосувати будь-які відповідні знижки за рейтинговий час. Це може бути перші п’ять хвилин безкоштовно, а після цього дзвінок буде стягуватися за звичайною ставкою. Такий вид знижок називають рейтинговими знижками за часом;
- щоб зберегти оцінену подію в базі даних з метою оплати або надіслати її зовнішній системі для виставлення рахунків.

На рис. 1.2 показано огляд механізму оцінки та пов'язаних з ним функцій.

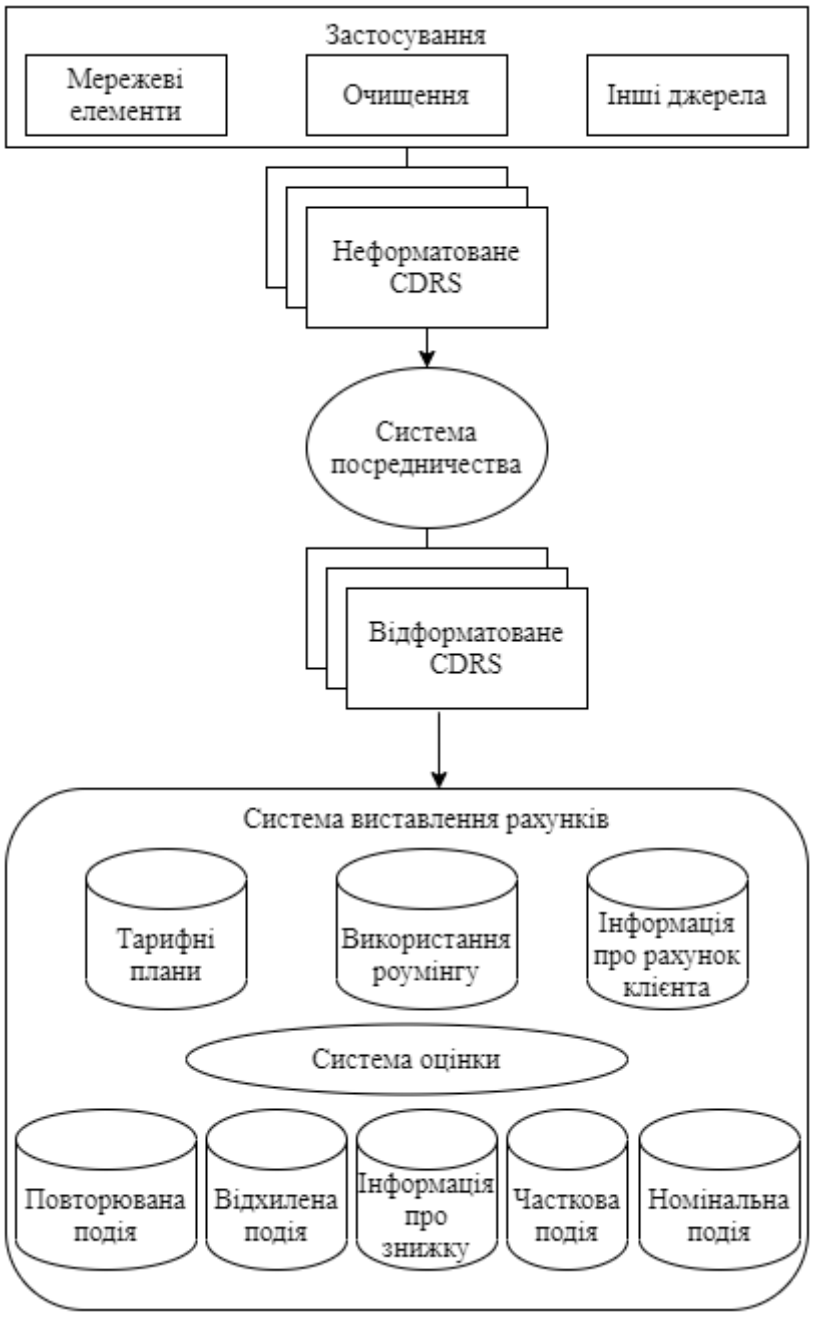

Рисунок 1.2 – Механізм оцінки

Інформація замовника визначає тарифний план (рейтинговий тариф), який слід використовувати для розрахунку плати / ціни. Механізм рейтингу використовує рейтингові таблиці та інформацію про подію з CDR (наприклад, відстань, час доби, місце дзвінка, тривалість або обсяг події тощо) для розрахунку фактичної плати за кожен дзвінок [4].

2 АНАЛІЗ ТРАФІКУ В КОНТАКТНИХ ЦЕНТРАХ

2.1 Основні принципи роботи контактного центру

Сьогодні межа між успіхом і невдачею в бізнесі дуже крихітна. Безперервний контакт з клієнтами є ключем успіху. Будівництво безпроблемного контактного центру допомагає підтримувати контакти з існуючими та, крім того, потенційними клієнтами. Контакт-центр (ЦК) - це складна система зв'язку, яка виступає посередником зв'язку між корпорацією та її клієнтами. Це програмне забезпечення та апаратна надбудова АТС, яка є головним шлюзом для голосового зв'язку для організації. СС поєднує технічні компоненти з людським фактором, напр. агенти ЦК, менеджери, керівники, адміністратори та інші співробітники. Складність технології СС дозволяє застосовувати різні способи її використання.

Корпоративне впровадження СС призводить до кращого наближення до клієнтів, і тому воно має великі заслуги для обох сторін. Клієнт спростив доступ до продукту та послуги центр приносить різні витрати з економічної точки зору. Ці витрати можна розділити на три основні групи загалом: плата за телекомунікації, витрати на людські ресурси та витрати на технічне рішення.

Значна частина цих витрат виробляє витрати на людські ресурси (понад 50%) заробітну плату, витрати на навчання, премії та інші. Ці витрати періодично повторюються, отже, важливо мати відповідну кількість агентів у найкоротший час разом із найменшими витратами. Метою є обслуговування найбільшої кількості клієнтів з найменшою кількістю агентів.

Для керівників ЦК зусилля щодо зменшення витрат, підвищення якості та підвищення ефективності означає пошук відповідей на наступні запитання:

- скільки агентів нам потрібно;
- які навички вони повинні мати;
- як організувати свій робочий час (зміни, перерви, обіди, тренінги);
- скільки, коли і якого типу дзвінків ми очікуємо;
- як можна перенаправити виклик для найбільш ефективного використання агентів;

- як можна передбачити трафік контакт-центру.

Відповіді на ці питання можна знайти за допомогою різних інструментів, моделей та моделювання.

2.2 Принципи моделювання контактного центру

Загалом ми маємо три суперечливі та обмежуючі параметри трафіку QoS, задоволеність працівників та витрати на контактні центри. Для поліпшення якості обслуговування необхідно використовувати більшу кількість агентів, тому витрати зростають. Якщо ми хочемо зменшити витрати, заробітна плата агентів повинна бути зменшена, якщо нам потрібна така ж хороша якість обслуговування. Але зменшення заробітної плати спричиняє невдоволення працівників, а також якість їх роботи впаде. Правильне розміщення контактного центру означає пошук системи, в якій усі ці параметри будуть збалансованими. Робота контактного центру базується на наступному принципі: клієнти звертаються до контактного центру з різними запитами. Перший контакт із клієнтом здійснюється за допомогою IVR (інтерактивна голосова відповідь), яка відповідає за ідентифікацію клієнтів і здатна обробляти деякі запити клієнтів. Якщо обробка запиту потребує допомоги агента, IVR перенаправить виклик на модуль ACD (автоматичне розподіл викликів), який підключить виклик до найбільш відповідних агентів за допомогою спеціальних алгоритмів маршрутизації. Якщо кількість абонентів перевищує кількість агентів, черги очікування збільшаться. ACD також керує цими чергами очікування. Нарешті, агент контактного центру обробить запит клієнта.

Клієнти випадково телефонують до контакт-центру. Деякі з них будуть чекати на лінії через весь процес (контакт з IVR, перенаправлення через ACD і очікування в черзі), інші зависають після контакту з IVR або під час очікування в черзі. Якщо ми хочемо мати повністю функціональну модель контакт-центру , важливо точно імітувати його трафік. Необхідно визначити основні параметри, які полегшать моніторинг трафіку контактного центру:

- відсоток успішно відповідених дзвінків;
- відсоток перерваних дзвінків дзвінки, перервані клієнтами до того, як їх обслужив агент;

- середня швидкість відповіді;
- година максимального транспортного навантаження;
- середня швидкість обробки дзвінків;
- GoS Ступінь обслуговування означає очікуваний рівень запитуваних послуг; гарним GoS може бути, наприклад, якщо 80% вхідних дзвінків обслуговуються до 20-30 секунд (решта дзвінків чекає довше).

Зі статистичної точки зору вхідні дзвінки є неоднорідними змінними, їх розподіл ймовірностей може бути добре описаний функцією розподілу ймовірності Пуассона / функцією щільності ймовірності дискретної випадкової величини із середнім значенням. Для вимірювання трафіку в телекомунікаціях ми використовуємо Erlangunit. Отже, для моделювання трафіку в контактних центрах використовується модель Erlang, де джерела запитують певні послуги, а сервери пропонують ці послуги. У контактних центрах джерелами трафіку є клієнти, а сервери - агенти, призначені для обслуговування дзвінків клієнтів. Телефонний термінал, як правило, обслуговується одним агентом.

2.3 Аналіз моделі Erlang B

Для аналізу трафіка в контактних центрах досліджено модель Erlang Bmodel. Вхідні дзвінки моделюються розподілом Пуассона із середнім вхідним значенням $\mathcal{Z}$. Усі посилання контактного центру зайняті з імовірністю P_B (ймовірність блокування дзвінків), це означає, що вхідні дзвінки вважаються втраченими. Ймовірність успішного виклику до контакт-центру та послідовного зв’язку з агентом за допомогою ACD становить $1- P_B$. Модель Erlang B не враховує жодних черг очікування для взаємозв'язку викликів із агентом, показано на рис. 2.1.

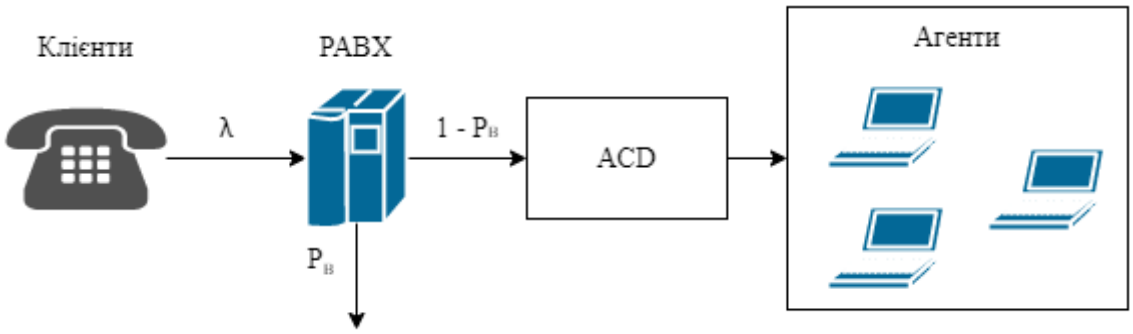

Рисунок 2.1 - Модель Erlang B

Для розрахунків параметрів контактного центру за допомогою Ерланга Bmodel B використовується формула Erlang B:

$$P_B = \frac{\frac{A^N}{N!}}{\sum_{i=0}^{N} \frac{A^i}{i!}} \qquad (2.1)$$

Формула 2.1 Erlang B використовує 3 основні параметри:

- A – транспортне навантаження в Ерлангу;
- N – кількість агентів / ліній / магістралей (запитується одночасне підключення);
- P_B – ймовірність блокування дзвінків.

Завдяки формулі Erlang B можна обчислити невідомий параметр, якщо знаємо решту два. Для успішного роботи з контактним центром дуже важливо визначити точну кількість агентів (N), необхідних для обробки всього трафіку контактного центру (A).

Припустимо, що модель для розрахунку кількості агентів наступна: змодельований контактний центр працює з навантаженням $A = 5$ Erl. Це навантаження дорівнює в середньому 60 вхідних дзвінків за одну годину, середній час подачі - 5 хвилин (приблизно 0,084 години), а контакт-центр працює 8 годин на день, це показано на рис. 2.2.

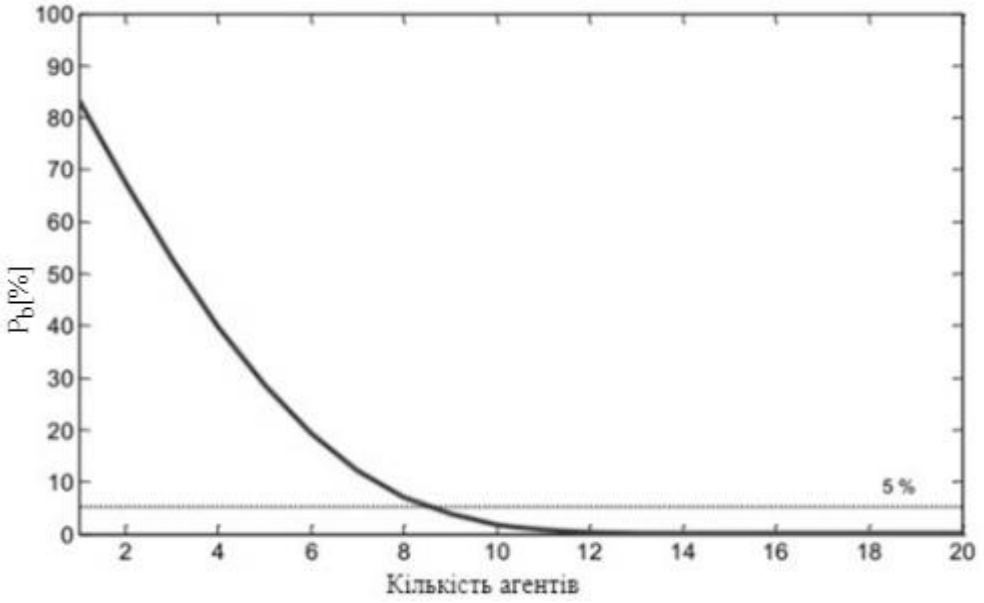

Рисунок 2.2 - Зв'язок між P_B та кількістю агентів, якщо навантаження трафіку $A = 5$.

З рис. 2.2 можна зробити висновок, що зі збільшенням кількості агентів ймовірність відхилення виклику P_B різко зменшується до конкретного рівня. Після досягнення 5% це різко зменшується зупинками. У цьому випадку це після досягнення 8 агентів у контактному центрі. Після досягнення 12 агентів ймовірність відхилення P_B майже дорівнює нулю. Отже, між цими двома значеннями лежить потрібна кількість агентів.

Для визначення точної кількості агентів необхідно вибрати максимально можливу ймовірність відхилення виклику P_B, Контакт-центр не повинен мати вищу ймовірність відхилення виклику визначається з наступної таблиці 2.1.

Таблиця 2.1 - Результати з моделі Erlang B

A[Erl]	*N*	P_B*[%]*
5	5	28,49
5	6	19,18
5	7	12,05
5	8	7,00
5	9	3,75
5	10	1,84
5	11	0,83
5	12	0,34

У цьому випадку максимальне значення ймовірності відхилення виклику становить P_B = 2,5%. Це значення вибирається щодо запитуваних параметрів контактного центру. Це залежить від окремого менеджера контактного центру.

Розрахунок максимального навантаження з визначеною кількістю агентів Модель Erlang B також використовується для розрахунку максимального навантаження контактного центру, не змінюючи кількість агентів. Зв'язок між P_B та навантаженням A з кількістю агентів = 10 показано на рис. 2.3.

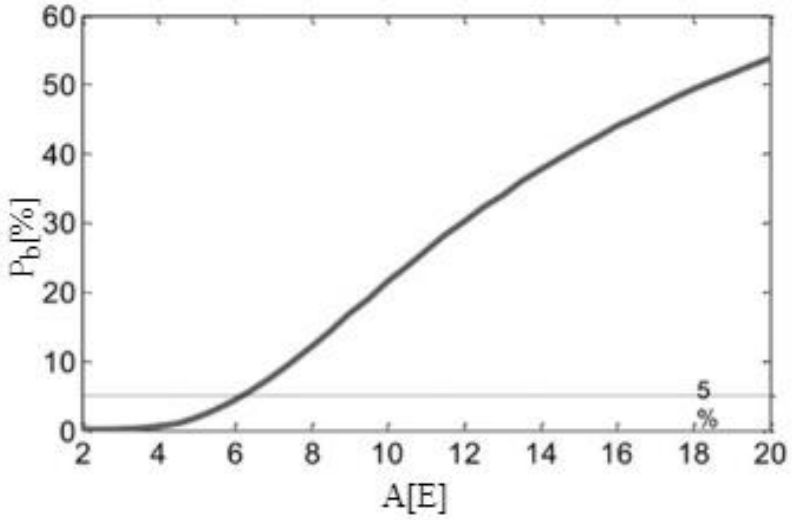

Рисунок 2.3 - Зв'язок між P_B та навантаженням A з кількістю агентів = 10

З рис. 2.3 можна зробити висновок, що зі збільшенням завантаженості трафіку та незмінною кількістю агентів ймовірність відхилення виклику P_B різко зростає. 5% кордону P_B досягає при завантаженні 6 Erl, при 10Erl ймовірність відхилення дзвінка становить 20%, а при 18 Erl - 50%, тобто кожен другий дзвінок (в середньому) буде відхилений.

Визначення кількості агентів щодо навантаження на транспорт при постійній P_B Зі збільшенням навантаження на транспорт і постійною кількістю агентів контакт-центр відносно швидко перевищує максимальну ймовірність відхилення виклику P_B. Це значення підбирається індивідуально, в процесі аналізу максимальне значення P_B становить 2,5%. Максимальний P_B = 2,5%, залишається тим самим. Це співвідношення показано на рис. 2.4.

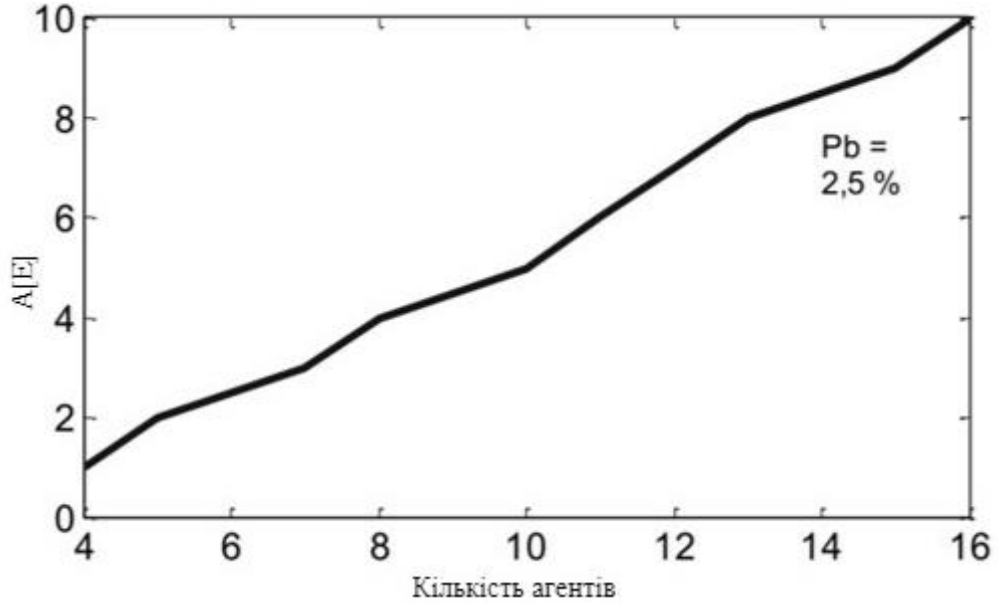

Рисунок 2.4 - Зв'язок між транспортним навантаженням A та кількістю агентів константою Nat P_B = 2,5%

Можна зробити висновок, що кількість агентів, необхідних для забезпечення таких P_B, є майже лінійною. Також на рис 2.4 показано, що при навантаженні A = 5 Erl, для досягнення максимальної ймовірності відторгнення P_B = 2,5% потрібні N = 10 агентів, як було розраховано раніше [5].

Аналіз результатів моделі Erlang B З розрахункових значень і діаграм модель Erlang B може бути використана для розрахунків наступних параметрів:

- кількість агентів, необхідних для обслуговування певного навантаження трафіку при певному P_B;
- максимально можливе навантаження транспортного центру при постійній кількості агентів;
- ймовірність відхилення виклику при визначеному навантаженні та кількості агентів.

2.4 Аналіз моделі Erlang C

Відмінність від моделі B полягає в тому, що в моделі Erlang C відхилені дзвінки не вважаються втраченими, але затримуються до тих пір, поки транк не буде звільнений і ці виклики не будуть обслуговуватися. Модель Erlang C розглядає чергу

очікування протягом нескінченного часу очікування. Жоден дзвінок не втрачається, абонент, що телефонує, залишається на лінії, доки його не обслуговують, це показано на рис. 2.5.

Рисунок 2.5 - Модель Erlang C

Для розрахунків параметрів контактного центру за допомогою моделі Erlang C краще використовувати розширену формулу 2.2 Erlang C, яка містить, ймовірно, відхилення виклику, визначену формулою Erlang B.

$$P_C = \frac{N * P_B}{N - A + + * P_B} \tag{2.2}$$

Формула 2.2 містить 4 основні параметри:

- A – транспортне навантаження контактного центру в Ерлангах;
- N – кількість агентів;
- P_B – ймовірність відхилення дзвінка;
- Pc – ймовірність очікування дзвінка в черзі очікування.

Визначивши три з цих параметрів, можна обчислити четвертий з формули 2.2. Найважливіші параметри контактного центру можна розрахувати, якщо додати п'ятий параметр GoS і після комбінування з формулою 2.2:

$$GoS = 1 - P_C * e^{-\frac{(N-A)t}{h}} \tag{2.3}$$

Формула 2.3 включає такі важливі параметри:

- *GoS* – ймовірність прийняти дзвінок агентом до часу *t*;
- *t* – максимальний час очікування на обслуговування дзвінка, що виконує Божу умову;
- *h* – середній час обробки дзвінків за агентом.

Для точного підрахунку кількості агентів нам потрібно знати очікуваний трафік контактного центру. Нехай A = 5 Erl, що дорівнює 60 вхідним дзвінкам за один час і середній час обробки дзвінків 5 хвилин. Можливість відхилення виклику P_B повинна бути максимум 5% і ймовірність очікування в черзі на обслуговування дзвінків також 5%, так що лише 3 дзвінки очікують в черзі із загальної кількості 60 вхідних дзвінків. Зв'язок P_B і Pc з кількістю агентів при транспортному навантаженні S = 5 Erl показано на рис. 2.6.

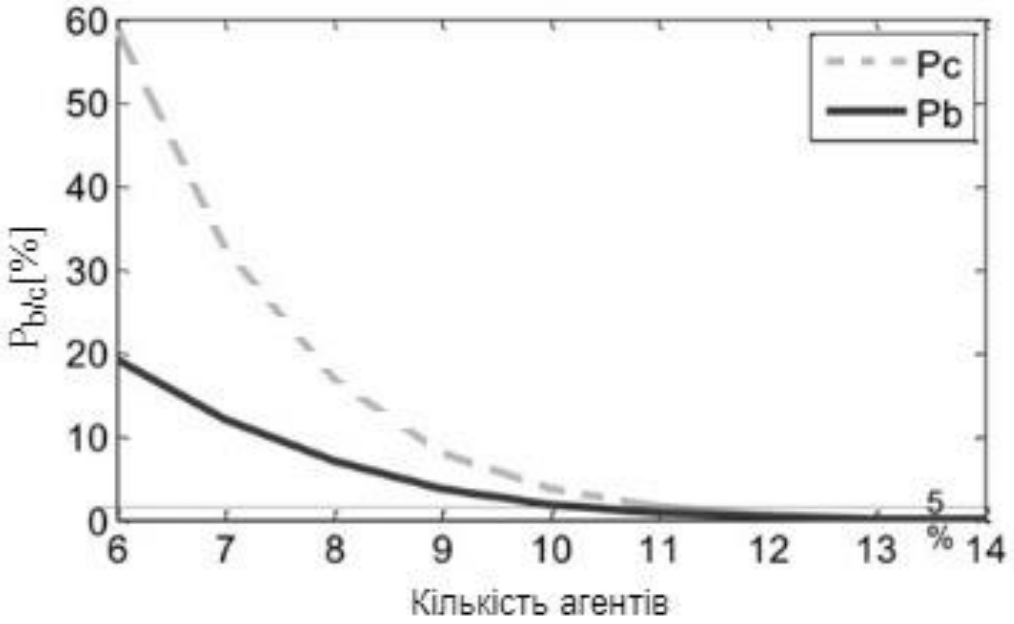

Рисунок 2.6 - Зв'язок P_B і Pc з кількістю агентів при транспортному навантаженні S = 5 Erl

З рис. 2.6 можемо зробити висновок, що зі збільшенням кількості агентів ймовірність P_B і Pc зменшується. Імовірність P_B досягає обраного значення у 9 агентів, імовірність Pc у 10 агентів. Точні номери наведені в таблиці 2.1. З таблиці 2.1 видно, що для 10 агентів GoS становить майже 98%, що означає, що лише 2% вхідних дзвінків чекають черги довше, ніж 30 секунд. Збільшуючи кількість агентів

до 11, GoS зростає до 99%, тобто підвищення лише на 1%, а витрати на цей агент підвищують значно більше 1%. Отже, з цих розрахунків можна зробити висновок, що додавати одинадцятий агент не потрібно.

Таблиця 2.2 - Результати з моделі Erlang C

A	*N*	P_B*[%]*	*Pc[%]*	*Gos[%]*
5	6	19,18	58,75	46,83
5	7	12,05	32,41	73,46
5	8	7,00	16,72	87,60
5	9	3,75	8,05	94,60
5	10	1,84	3,61	97,81
5	11	0;83	1,50	99,17
5	12	0,34	0,58	99,70
5	13	0,13	0,21	99,90

Зв'язок між GoS (t = 30 с) і кількістю агентів при постійному завантаженні трафіку показано на рис 2.7. Зі збільшенням кількості агентів GoS різко зростає, це збільшення зупиняється на кількості 10 агентів, де GoS становить майже 98%. Подальше збільшення кількості агентів впливає на вартість GoS лише в мінімальному аспекті.

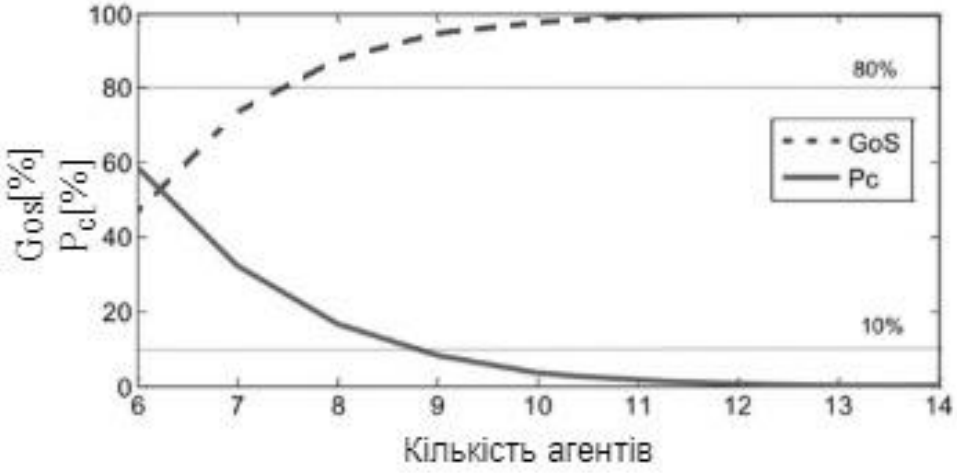

Рисунок 2.7 - Зв'язок GoS і *Pc* з кількістю агентів при навантаженні A = 5 Erl

Значення 80% досягає GoS вже у 8 агентів; у цьому випадку ймовірність переадресації викликів на чергу очікування становить більше 10%. У 9 агентів

обидва значення підходять для ефективності роботи контактного центру. З рис. 2.7 видно, що відношення GoS за кількістю агентів протилежне відношенню між ймовірністю очікування в черзі *Pc* та кількістю агентів.

Стабільність контактного центру означає рівень продуктивності контактного центру при визначеному транспортному навантаженні. Цей рівень позначається різними параметрами. Найважливішим параметром є значення GoS на обраному *t* параметр, іншими параметрами є ймовірність переадресації виклику до черги *Pc*, що очікує, кількість дзвінків, що очікують у черзі, що буде обслуговуватися, та ймовірність відхилення виклику через блокування посилань P_B. Зв'язок між P_B та *Pc* та навантаженням трафіку *A* за кількістю агентів $N = 10$ показано на рис. 2.8.

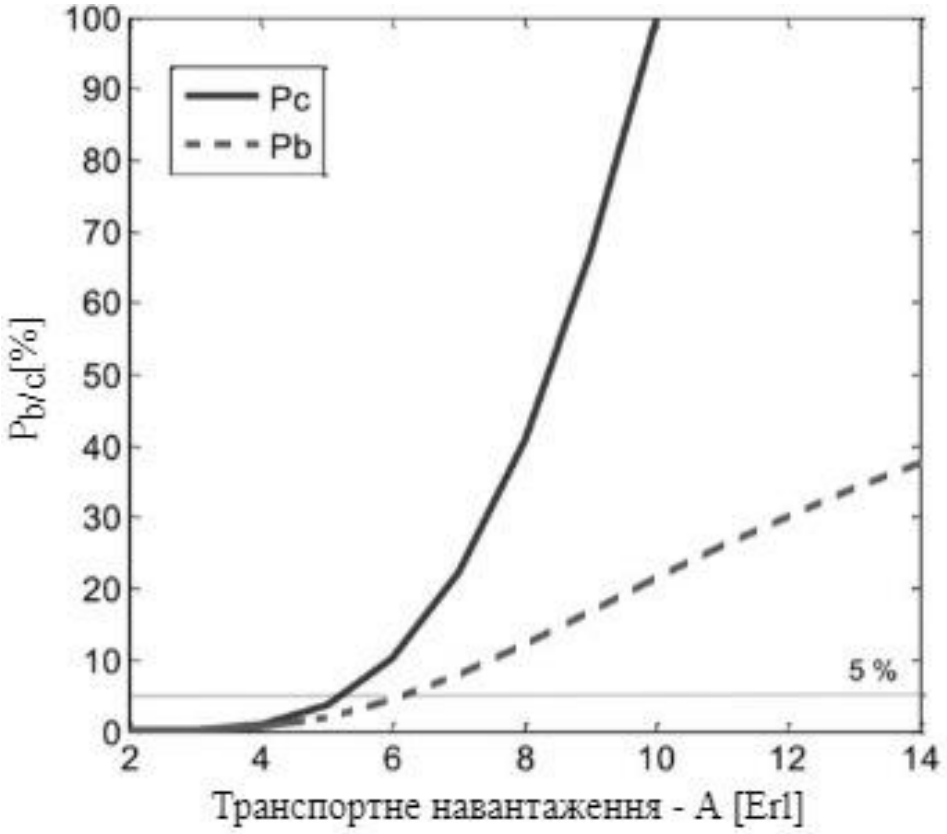

Рисунок 2.8 - Зв'язок між P_B та *Pc* та навантаженням трафіку *A* за кількістю агентів $N = 10$

Для навантаження трафіку 5 Erl, розрахункова кількість агентів, необхідних для обробки такого навантаження, дорівнює 10. При цьому значенні контактний центр досягає GoS 97,81%, максимальний час очікування виклику в черзі, що

заповнює цей GoS, становить 30 секунд, а кількість відхилених викликів 1,84% на вхідних дзвінках (буде заблоковано). Форма рис. 2.8 можна визначити максимальне навантаження на трафік, при якому контакт-центр буде працювати добре. Очевидно, що зі збільшенням навантаження на трафік різко зростає ймовірність переадресації викликів до черги очікування. Вже при завантаженні 5 Erl Pc перевищує 5%, при завантаженні 10 Erl контактний центр заповнений, ймовірність перенаправлення виклику в чергу очікування становить 100%, і кожен п'ятий дзвінок блокується [7].

GoS і збільшення навантаження впливає на середній час обслуговування дзвінків. На підставі попереднього аналізу, зробленого на прикладі моделі, для хорошої роботи контакт-центру при навантаженні A = 5 Erl нам потрібні N = 10 агентів, на випадок, якщо середній час обслуговування виклику агентом становить h = 5 хвилин (300 секунд). і h були обрані, а кількість агентів була розрахована за моделлю Erlang C. З цими параметрами контакт-центр працює зі значенням GoS 97,81%, виконуючи умови t = 30 с.

Збільшення завантаженості контактного центру з початкового значення A=5 Erl (60 вхідних дзвінків на годину), середній час обслуговування дзвінків h= 5 хвилин; при цьому навантаженні. Зв'язок між P_B та Pc та навантаженням трафіку A за кількістю агентів N = 10 показано на рис. 2.9.

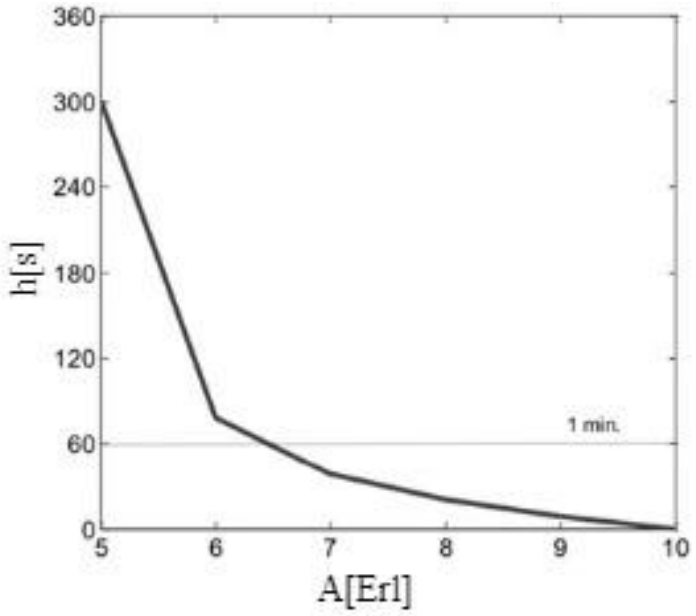

Рисунок 2.9 - Зв'язок між середнім часом обслуговування дзвінка та навантаженням трафіку A при однаковому значенні GoS

З рис. 2.9 видно, що нам потрібно любити значення середнього часу дзвінка зі збільшенням кількості вхідних дзвінків. Для того, щоб зберегти той самий GoS при збільшенні навантаження трафіку, агенти повинні працювати набагато швидше. Тільки збільшення завантаженості трафіку на 1 Ерланг спричиняє майже нестійкість GoS при тій же кількості агентів.

Аналіз результатів моделі Erlang C. З розрахункових значень і діаграм модель Erlang C (після розширення за параметром GoS та модифікація Erlang Cformula за формулою Erlang B) може бути використана для розрахунків за наступними параметрами контактного центру:

- кількість агентів, необхідних для обробки навантаження трафіку A при визначених максимальних значеннях для Pc а P_B;
- GoS при визначеній кількості агентів, визначеному навантаженні трафіку та середньому часі обслуговування дзвінків;
- максимально можливе навантаження на транспорт при визначеній кількості агентів (стійкість контактного центру);
- максимальний середній час обслуговування дзвінка щодо значення GoS;
- імовірність відхилення виклику (P_B), при визначеному навантаженні та кількості агентів;
- ймовірність перенаправлення виклику на чергу очікування (Pc), при визначеному завантаженні та кількості агентів.

Таким чином, досліджено, що передумови моделі Erlang B для контактного центру недостатні (відсутність черги очікування), Erlang Bmodel не є оптимальним для розрахунків кількості необхідних агентів. Але ця модель може бути використана для розрахунку параметрів за допомогою моделі Erlang C, де вона може обчислити ймовірність блокування посилання або кількість необхідних телефонних ліній у центрі контактів.

Аналіз моделі Erlang C показує, що її можна використовувати для розрахунків усіх необхідних і важливих параметрів руху контактного центру. Основним недоліком моделі Erlanh C є розрахунок з нескінченною чергою очікування та нескінченне терпіння клієнта. Іншим недоліком є відсутність розгляду IVR у контактному центрі.

Перевага моделі Erlang C полягає в простоті та широкому масштабі розрахунків для майже всіх важливих параметрів контактного центру [6].

3 ОГЛЯД ТЕХНОЛОГІЙ ЯКІ ВИКОРИСТОВУЮТЬСЯ В ELK STACK

3.1 Необхідність ведення журналів системи у сучасному світі

Практично всі види обчислювальних пристроїв, систем і додатків випромінюють свого роду журнал для вказівки стану системи. Простіше кажучи, журнал - це просто потік повідомлень в часі Послідовність. Вони можуть бути спрямовані в файли і збережені на диску або спрямовані в реєстратор журналу. Первинні журнали - це просто дані, але коли вони обробляються і аналізуються, вони дають корисну інформацію. Аналізатори журналів беруть за вхідні дані масу даних, що виробляються нашими брандмауерами, маршрутизаторами, IDS та програмами, і перетворюють ці дані на дієвий інтелект.

Всякий раз, коли розробник або адміністратор системи стикається з проблемою із системою, першим інстинктом є перегляд журналів. Протягом тривалого часу ми використовували основні інструменти, такі як grep, awk або perl, щоб виконувати аналіз журналів. Однак із зміною часу та масштабних додатків попередніх методів вже не вистачає. Уявіть собі систему з десятками, сотнями або тисячами хостів. На всіх цих хостах працює кілька екземплярів різних програм. Щоб зробити речі цікавішими, ці хости можуть розташовуватися не в одному місці, і нам, можливо, доведеться мати справу з мільйонами рядків у журналах. У такому світі неможливо вирішити проблеми за допомогою раніше використовуваних інструментів або просто переглянувши одного конкретного хоста. Додайте до цього той факт, що журнали можуть створюватися в різних часових поясах, форматах і навіть різними мовами. Що вимагає цілісного рішення створення, аналізу, зберігання та аналізу журналів.

Все більше і більше IT-інфраструктури переходить до загальнодоступних хмар, таких як Amazon Web Services та Microsoft Azure, що робить платформи аналізу журналів все більш критичними. Ізоляція продуктивності не є тривіальною в хмарних інфраструктурах. Для цього існує багато факторів, таких як коливання навантаження на віртуальних машинах, динамічна кількість користувачів та зміни в середовищі. Ці проблеми може контролювати лише платформа управління

журналами наступного покоління, яка може сканувати різні джерела, такі як системні журнали, журнали веб-сервера, журнали програм та журнали ELB та S3 на AWS. Правильний аналіз журналів може допомогти інженерам DevOps, системним адміністраторам, інженерам із надійності веб-сайтів та розробникам приймати кращі рішення.

Журнали надають нам необхідну інформацію про те, як поводиться наша система. Однак зміст і формат журналів різняться залежно від різних служб або, скажімо, від різних компонентів однієї системи. Наприклад, сканер може реєструвати повідомлення про помилки, пов'язані зі зв'язком з іншими пристроями; з іншого боку, веб-сервер реєструє інформацію про всі вхідні запити, вихідні відповіді, час приймається для відповіді тощо. Подібним чином журнали програм веб-сайту електронної комерції реєструватимуть журнали, що стосуються конкретного бізнесу.

Оскільки журнали залежать від їх змісту, змінюватиметься і їх використання. Наприклад, журнали зі сканера можуть бути використані для усунення несправностей або для простої перевірки стану або звітування, тоді як журнал веб-сервера використовується для аналізу моделей трафіку для багатьох продуктів. Аналіз журналів із веб-сайту електронної комерції може допомогти з'ясувати, чи повертаються пакети з певного місця повторно, та ймовірні причини того самого.

Нижче наведено кілька типових випадків використання, коли аналіз журналу є корисним, що показано на рис. 3.1.

Рисунок 3.1 – Випадки, коли використання аналіз журналу є корисним

Налагодження - одна з найпоширеніших причин увімкнення журналювання у вашій програмі. Найпростішим і найчастішим використанням журналу налагодження є grep для конкретного повідомлення про помилку або виникнення

події. Якщо системний адміністратор вважає, що програма аварійно завершила роботу через збій мережі, він або вона спробує знайти повідомлення про перерване з'єднання або подібне повідомлення в журналах сервера, щоб проаналізувати, що спричинило проблему. Щойно помилку або проблему виявлено, рішення для аналізу журналів допомагають збирати інформацію про програми, а знімки того конкретного часу можуть бути легко передані командам розробників для подальшого її аналізу.

Аналіз журналів допомагає оптимізувати або налагодити роботу системи та надати суттєві дані щодо вузьких місць у системі. Розуміння продуктивності системи часто полягає у розумінні використання ресурсів у системі. Журнали можуть допомогти проаналізувати використання індивідуальних ресурсів у системі, поведінку декількох потоків у програмі, потенційні тупикові умови тощо. Журнали також містять інформацію про мітки часу, що є важливим для аналізу поведінки системи з часом. Наприклад, журнал веб-сервера може допомогти дізнатись, як працюють окремі служби, залежно від часу відгуку, кодів відповідей HTTP тощо.

Журнали відіграють життєво важливу роль в управлінні захистом програм будь-якої організації. Вони особливо корисні для виявлення порушень безпеки, зловживання додатками, зловмисних атак тощо. Коли користувачі взаємодіють із системою, вона генерує події журналу, які можуть допомогти відстежувати поведінку користувачів, виявляти підозрілі дії та викликати тривоги або інциденти безпеки через порушення.

Процес виявлення вторгнень включає реконструкцію сеансу з самих журналів. Наприклад, події входу в систему SSH у системі можуть бути використані для виявлення будь-яких порушень на машинах.

Прогностичний аналіз є однією з найгарячіших тенденцій останнього часу. Дані журналів та подій можна використовувати для дуже точного прогнозного аналізу. Моделі прогнозного аналізу допомагають визначити потенційних споживачів, планувати ресурси, керувати запасами та оптимізувати, ефективність навантаження та ефективне планування використання ресурсів. Це також допомагає керувати маркетинговою стратегією, націлюванням на сегменти користувачів, стратегією розміщення реклами, і так далі.

Коли справа стосується пристроїв IoT (пристроїв або машин, які взаємодіють між собою без будь-якого втручання людини), життєво важливо, щоб система

контролювалась та управляла зведенням до мінімуму простоїв і швидко вирішувала всі важливі помилки або проблеми. Оскільки ці пристрої повинні працювати з незначним втручанням людини і можуть існувати у великому географічному масштабі, дані журналів, як очікується, зіграють вирішальну роль у розумінні поведінки системи та скороченні простоїв [8].

3.2 Характеристики інструментів ELK Stack, Elasticsearch, Logstash, Kibana

Стек ELK (Elasticsearch, Logstash та Kibana) складається з проектів з відкритим кодом, які беруть дані з будь-якого джерела та будь-якого формату, а потім здійснюють пошук, аналіз та візуалізацію в режимі реального часу. Він пропонує платформу управління журналами наступного покоління, яка вирішує проблеми, пов’язані з неоднорідністю та масштабом журналів. В основі стеку ELK лежить Elasticsearch, який є розподіленим механізмом пошуку та аналітики з відкритим кодом. Він заснований на Apache Lucene і розроблений для горизонтальної масштабованості, надійності та простоти управління. Logstash - це трубопровід збору даних, збагачення та транспортування. Можливість інтеграції з'єднувачів із загальною інфраструктурою дає Logstash можливість обробляти різні типи журналів, подій та неструктурованих джерел даних для розподілу серед різноманітних результатів, включаючи Elasticsearch. Стек ELK доповнений Kibana, платформою візуалізації даних, яка забезпечує взаємодію з даними за допомогою приголомшливої потужної графіки. Kibana може оживити дані за допомогою інформаційних панелей, використовуючи різноманітні візуальні ефекти від гістограм до географічних карт.

Стек ELK працює над концепцією поєднання одного компонента з іншим. Це створює конвеєр даних, що показано на рис. 3.2.

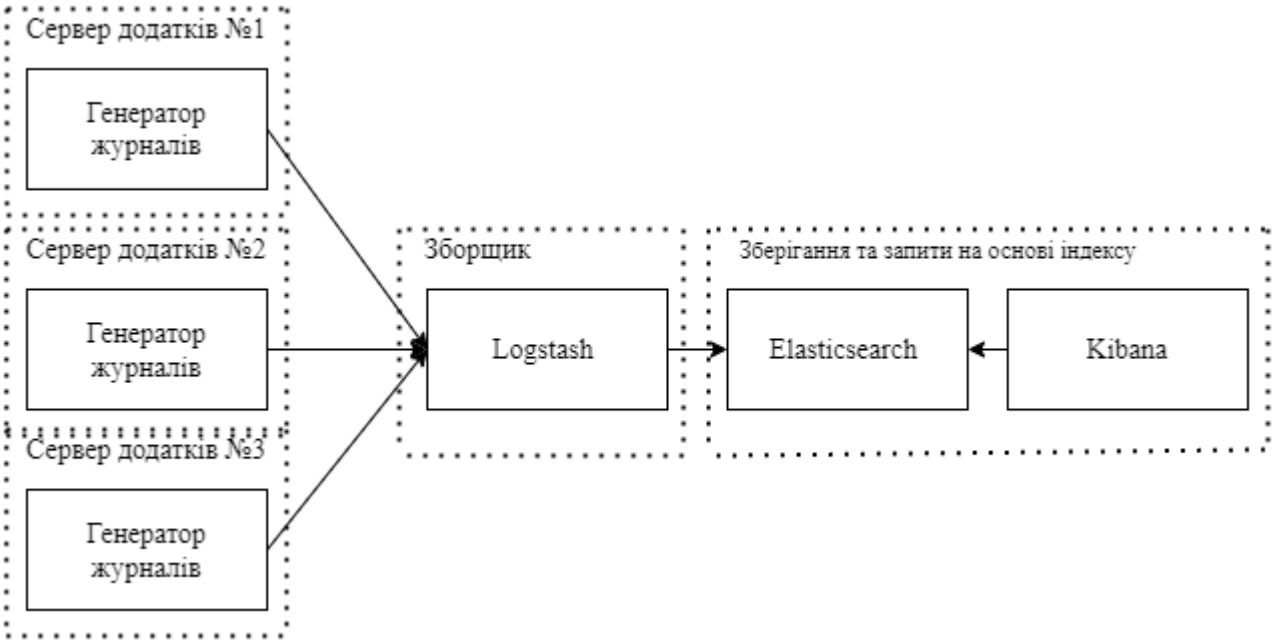

Рисунок 3.2 – Приклад конвеєра даних ELK

Кожна програма запускає в кінці відправника logstash, який надсилає журнали до центрального сервера Logstash, відомого як індексатор. Використання вантажовідправника в кінці заявки дозволяє уникнути необхідності повного встановлення Logstash скрізь. Як бачите, центральний індексатор Logstash може отримувати журнали від декількох програм. Звідти журнали передаються в кластер Elasticsearch. Kibana можна використовувати для запиту кластера Elasticsearch для відображення чудових діаграм та побудови інформаційних панелей.

Elasticsearch - це пошуковий сервер, заснований на Apache Lucene. Він надає можливість розподіленого в повному обсязі повнотекстового пошукового механізму в режимі реального часу. Він надає RESTful API, використовуючи документи JSON. Він може бути використаний для повнотекстового пошуку, структурованого пошуку, аналітики або комбінації всіх трьох. Elasticsearch розроблений на Java і випущений як відкритий код на умовах ліцензії Apache 2.0. Однією з його ключових особливостей є можливість швидкого пошуку шляхом індексації тексту, який потрібно шукати.

Багато пошукових систем доступні протягом тривалого часу з можливістю пошуку на основі мітки часу або точних значень. Отже, в чому проблема

Elasticsearch? Він диференціюється, виконуючи повнотекстовий пошук, обробляючи синоніми та оцінюючи документи за релевантністю. Більше того, він також може генерувати аналітику та агрегування з тих самих даних у режимі реального часу. Тут Elasticsearch виграє порівняно з іншими пошуковими системами. Elasticsearch змусить вас закохатися у ваші дані.

Elasticsearch досить популярний у багатьох великих компаніях. Нижче наведено кілька випадків використання.

Netflix використовує Elasticsearch для доставки мільйонів повідомлень клієнтам у будь-який день по кількох каналах, таких як електронна пошта, push-сповіщення, текст, голосові дзвінки тощо.

Salesforce створив власний плагін поверх Elasticsearch, який дозволяє збирати дані журналу Salesforce, полегшуючи розуміння тенденцій використання організації та поведінки користувачів.

New York Times використовує Elasticsearch для розміщення всіх 15 мільйонів своїх статей, опублікованих за останні 160 років. Це забезпечує чудову можливість пошуку в архівах.

Microsoft використовує Elasticsearch для пошуку та аналітичних можливостей у різних продуктах, таких як MSN, Microsoft Social Listening та Azure Search.

EBay використовує Elasticsearch для створення гнучкої платформи пошуку і надалі використовує її для аналізу даних.

Якщо вищезазначені випадки використання створюють враження, що Elasticsearch призначений лише для великих корпорацій, то дозвольте запевнити вас, що він використовується також у багатьох стартапах та малих корпораціях. Краса Elasticsearch полягає в тому, що ви можете запустити його на своєму ноутбуці або масштабувати до сотень серверів і петабайт даних.

Тепер давайте розглянемо деякі ключові особливості Elasticsearch:

- він забезпечує пошук та аналіз даних у режимі реального часу;
- Elasticsearch - це справді розподілена система, яка може працювати від скромного ноутбука до тисяч вузлів;
- він може бути розгорнутий як високодоступні кластери з підтримкою багатонаціональності. Після додавання нового вузла або відмови вузла він автоматично реорганізовує та врівноважує дані;

- Elasticsearch надає зручний інтерфейс RESTful, використовуючи JSON через HTTP. Усі дані чи інформація зберігаються як структуровані документи JSON;
- Elasticsearch побудований поверх Apache Lucene і доступний як програмне забезпечення з відкритим кодом за ліцензією Apache 2.

Замість того, щоб використовувати традиційні способи генерації та аналізу журналів, що має свої підводні камені, набагато краще використовувати Logstash, який є системою ведення журналів наступного покоління. Logstash - це, по суті, інтегрована структура для збору журналів, централізації, аналізу, зберігання та пошуку. Це програмне забезпечення з відкритим кодом, яке може динамічно об'єднувати дані з різних джерел та нормалізувати дані за вибраними вами напрямками.

Logstash дозволяє збагачувати та трансформувати будь-який тип подій за допомогою широкого набору плагінів введення, фільтрації та виведення, а багато вбудованих кодеків ще більше спрощують процес прийому. Logstash надає інформацію, використовуючи більший обсяг та різноманітність даних. Logstash може приймати вхідні дані з різних механізмів введення, таких як файли, Syslog, TCP / UDP, stdin та багато інших. Існує великий набір фільтрів, які можна застосувати до зібраних журналів для перетворення подій. Logstash не розчаровує під час виведення даних, оскільки підтримує безліч варіантів, таких як TCP / UDP, файли, електронну пошту, HTTP, Nagios та багато інших мережевих служб.

Logstash має розширювану архітектуру та зручну для розробників екосистему плагінів. Logstash - це найпопулярніша система збору подій для споживання даних, що передаються з мобільних пристроїв до інтелектуальних будинків, підключених транспортних засобів, датчиків охорони здоров'я та багатьох інших галузевих програм. Він пропонує статистику майже в реальному часі відразу за індексом або часом виведення. Logstash пропонує безліч агрегацій та мутацій, а також можливості узгодження шаблонів, географічного відображення та динамічного пошуку. Переадресація цих журналів з Logstash на Elasticsearch дозволяє виконувати різноманітний спектр зіставлення, агрегування та пошуку.

Kibana - це платформа для аналітики та візуалізації з відкритим кодом, яка працює над Elasticsearch. Він може використовуватися для пошуку, перегляду та взаємодії з будь-якими структурованими або неструктурованими даними, що

зберігаються в Elasticsearch. Це полегшує розширений аналіз даних і забезпечує візуалізацію даних на різноманітних гістограмах, діаграмах, графіках, таблицях і картах.

Розуміння великих обсягів даних досить інтуїтивно зрозуміло з Kibana. Це увімкнено за допомогою простого інтерфейсу на основі браузера, який дозволяє швидко створювати та обмінюватися динамічними інформаційними панелями, які можуть відображати зміни до запитів Elasticsearch у реальному часі.

Деякі ключові особливості Кібани:

- безшовна інтеграція з Elasticsearch дозволяє візуалізувати будь-які структуровані або неструктуровані дані. Дані, що надходять у Elasticsearch з будь-якого джерела, можна легко візуалізувати;
- краще розуміння даних, представляючи їх у різних формах, таких як стовпчасті діаграми, графіки ліній та розсіювання, гістограми, секторні діаграми та карти;
- інтеграція з потужними аналітичними можливостями Elasticsearch допомагає аналізувати дані з різних сторін;
- це легко налаштувати та використовувати. Гнучкий інтерфейс Kibana дозволяє легко створювати, зберігати, ділитися, експортувати та вбудовувати візуалізовані дані для подальшого спілкування [9].

3.3 Приклад використання ELK Stack

Стек ELK найчастіше використовується як інструмент аналізу журналів. Його популярність полягає в тому, що він забезпечує надійний та відносно масштабований спосіб узагальнення даних з різних джерел, їх зберігання та аналіз. Таким чином, стек використовується для різноманітних випадків використання та цілей, починаючи від розробки, моніторингу, безпеки та відповідності, до SEO та BI.

Перш ніж налаштувати стек, спочатку треба зрозумійте свій конкретний випадок використання. Це безпосередньо впливає майже на всі кроки, реалізовані на цьому шляху - де і як встановити стек, як налаштувати кластер Elasticsearch і які ресурси виділити йому, як побудувати конвеєри даних, як захистити установку - список нескінченний .

Журнали сумно відомі тим, що вони були в нагоді під час кризи. Перше місце, на яке слід звернути увагу, коли виникає проблема, - це ваші журнали помилок та винятки. Проте журнали стають в нагоді набагато раніше у життєвому циклі програми.

Впевнений в розробці, керованій журналами, де реєстрація починається з першої написаної функції, а потім інструментується протягом усього додатка. Реалізація входу у ваш код додає міри спостережливості у ваші програми, що стане в нагоді під час вирішення проблем.

Незалежно від того, розробляєте ви моноліт або мікросервіс, стек ELK з’являється на початку, як засіб для розробників співвідносити, виявляти та усувати помилки та винятки, що мають місце, переважно під час тестування або інсталяції, і до початку запуску коду. Використовуючи безліч різних додатків, фреймворків, бібліотек та вантажовідправників, журнальні повідомлення передаються в стек ELK для централізованого управління та аналізу.

Потрапивши у виробництво, інформаційні панелі Kibana використовуються для моніторингу загального стану програм та конкретних служб. Якщо проблема має місце, і якщо ведення журналу було структуровано структуровано, наявність усіх даних журналу в одному централізованому місці допомагає зробити аналіз та усунення неполадок більш ефективним і швидким процесом. На рис. 3.3 показано загальну структуру використання ELK Stack.

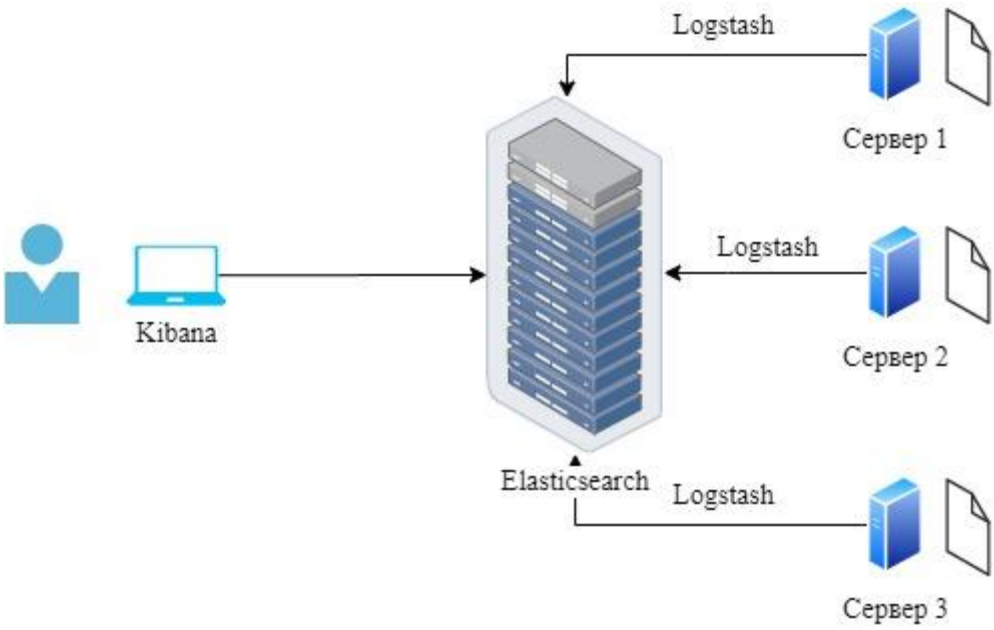

Рисунок 3.3 – Загальна структура використання ELK

Сучасні IT-середовища багатошарові та розподілені за своєю суттю, що створює величезну проблему для команд, відповідальних за їх експлуатацію та моніторинг. Моніторинг усіх різних систем та компонентів, що складаються з архітектури програми, надзвичайно забирає багато часу та ресурсів.

Щоб мати змогу точно вимірювати і контролювати стан та загальний стан навколишнього середовища, командам DevOps та IT-операцій потрібно враховувати наступні ключові міркування: як отримати доступ до кожної машини, як зібрати дані, як додати контекст до дані та обробляти їх, де зберігати дані та як довго їх зберігати, як аналізувати дані, як захистити дані та як створити їх резервну копію.

Стек ELK допомагає, надаючи організаціям засоби для вирішення цих питань, надаючи майже все-в-одному рішення. Beats можна розгорнути на машинах, щоб діяти як агенти, що пересилають дані журналу до екземплярів Logstash. Logstash можна налаштувати на агрегування даних та їх обробку перед індексацією даних у Elasticsearch. Потім Kibana використовується для аналізу даних, виявлення аномалій, аналізу основних причин та побудови чудових інформаційних панелей моніторингу.

І це не просто журнали. Хоча Elasticsearch спочатку був розроблений для повнотекстового пошуку та аналізу, він все частіше використовується для аналізу метрик. Моніторинг показників продуктивності для кожного компонента у вашій

архітектурі є ключовим для отримання видимості операцій. Збір цих показників можна здійснити за допомогою сторонніх агентів аудиту або моніторингу або навіть за допомогою деяких доступних ритмів (наприклад, Metricbeat, Packetbeat), а Kibana тепер постачається з новими типами візуалізації, що допомагають аналізувати часові ряди (Timelion, Visual Builder).

Моніторинг продуктивності додатків, він же APM, - один із найпоширеніших методів, який сьогодні використовують інженери для вимірювання доступності, часу відгуку та поведінки додатків та служб.

Elastic APM - це система контролю ефективності додатків, яка побудована поверх стеку ELK. Подібно до інших рішень APM на ринку, Elastic APM дозволяє відстежувати ключову інформацію, пов'язану з продуктивністю, таку як запити, відповіді, транзакції бази даних, помилки тощо.

Подібним чином, розподілені інструменти трасування з відкритим кодом, такі як Zipkin та Jaeger, можуть бути інтегровані з ELK для глибокого занурення у продуктивність програми.

Безпека завжди була вирішальною для організацій. Проте протягом останніх кількох років, як через збільшення частоти атак, так і через вимоги дотримання вимог (HIPAA, PCI, SOC, FISMA тощо), використання механізмів та стандартів безпеки стало головним пріоритетом.

Оскільки дані журналу містять безліч цінної інформації про те, що насправді відбувається в режимі реального часу в процесі запущених процесів, мало би бути несподіванкою, що безпека швидко стає вагомим варіантом використання стеку ELK.

Незважаючи на те, що як самостійний стек, ELK не має вбудованих функцій безпеки, той факт, що ви можете використовувати його для централізації реєстрації з вашого середовища та створення моніторингу та орієнтованих на безпеку інформаційних панелей, призвів до інтеграції стека з деякі визначні стандарти безпеки.

Ось два приклади того, як стек ELK може бути реалізований як частина першого розгортання системи безпеки.

Після встановлення DDoS-атаки час має найбільше значення. Швидка ідентифікація є ключовим фактором для мінімізації школи, а саме тут з моніторингу

журналів. Журнали використовують незароблений слід, що створюється запущеними процесами, і таким чином пропонують велику кількість інформації про те, що відбувається в реальному часі.

Використовуючи стек ELK, організація може побудувати систему, яка створює дані для різних рівнів в IT-середовищі (веб-сервер, бази даних, брандмауери тощо), створюючи дані для більш легкого аналізу та візуалізації даних про потужних панелях моніторингу.

SIEM - це підхід до управління безпекою підприємства, який прагне надавати цілісне уявлення про IT-безпеку організації. Основна мета SIEM - забезпечити одночасне та всебічне уявлення про вашу IT-безпеку. Підхід SIEM включає консолідовану інформаційну панель, яка дозволяє легко визначати діяльність, тенденції та закономірності. Якщо їх правильно впровадити, SIEM може запровадити законний загроза шляхом їхнього попереднього виявлення, моніторингу активності в Інтернеті, надавши звітів про відповідь та підтримку команди реагування на інцидентів.

Стек ELK може сприяти досягненню SIEM. Візьмемо для прикладу середовище на основі AWS. Організації, що обслуговують службу AWS, мають велику кількість засобів аудиту та ведення журналу, які генерують дані журналу, інформацію про аудит та деталі змін, внесених до служб конфігурації. Якщо розподілити джерела даних можна використовувати та використовувати разом, щоб отримати хороший та централізований огляд стека.

Business Intelligence (BI) - це використання програмного забезпечення, інструментів та додатків для аналізу вихідних даних організації з метою оптимізації рішень, поліпшення співпраці та підвищення загальної ефективності.

Процес передбачає збір та аналіз великих наборів даних з різних джерел даних: баз даних, ланцюгів поставок, обліку персоналу, даних виробництва, продажів та маркетингових кампаній тощо. Самі дані можуть зберігатися у внутрішніх сховищах даних, приватних хмарах або загальнодоступних хмарах, а інженерія, що займається вилученням та обробкою даних (ETL), породила низку технологій, як власних, так і з відкритим кодом.

Як і у попередніх випадках використання, описаних тут, стек ELK стане в нагоді для збору даних з цих різноманітних джерел даних в одне централізоване

місце для аналізу. Наприклад, ми можна витягувати журнали доступу до веб-серверів, щоб дізнатись, як наші користувачі отримують доступ до нашого веб-сайту; можна скористатися нашою системою CRM, щоб дізнатись більше про потенційних клієнтів та користувачів, або можна перевірити дані, які надає наш інструмент автоматизації маркетингу.

Існує ціла купа власних інструментів, які використовуються саме для цієї мети. Але стек ELK - дешевший варіант із відкритим кодом для виконання майже всіх дій, які надають ці інструменти.

Технічне SEO - ще один варіант використання стека ELK, але, тим не менш, важливий. Яке відношення має SEO до ELK? Ну, загальним знаменником, звичайно, є журнали.

Журнали доступу до веб-сервера (Apache, nginx, IIS) відображають точну картину того, хто надсилає запити на ваш веб-сайт, включаючи запити, зроблені ботами, що належать пошуковим системам, які сканують сайт. Експерти з SEO використовуватимуть ці дані для відстеження кількості запитів, зроблених Baidu, BingBot, GoogleBot, Yahoo, Yandex та іншими.

Технічні експерти SEO використовують дані журналу для моніторингу, коли боти востаннє сканували сайт, а також для оптимізації бюджету сканування, помилок веб-сайту та несправних переадресацій, пріоритету сканування, повторного сканування та багато іншого.

4 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ АВТОМАТИЧНОГО РОЗГОРТАННЯ УНІВЕРСАЛЬНОЇ СИСТЕМИ БІЛІНГУ ELK

4.1 Інструмент автоматизації розгортання ELK Stack

Ansible - це простий, гнучкий та надзвичайно потужний інструмент, який надає вам можливість автоматизувати загальні завдання інфраструктури, запускати спеціальні команди та розгортати багатозадачні додатки, що охоплюють кілька машин. Незважаючи на те, що ви можете паралельно використовувати команди Ansible для запуску команд на багатьох хостах, справжня сила полягає в управлінні ними за допомогою playbook.

Як системний інженер, інфраструктура, яку ми зазвичай потребуємо для автоматизації, містить складні багаторівневі додатки. Кожен із них представляє клас серверів, на прикладі, балансувальники навантаження, веб-сервери, сервери баз даних, програми кешування та черги проміжного програмного забезпечення. Оскільки багато з цих програм мають працювати в тандемі, щоб надавати послугу, тут також задіяна топологія. Наприклад, навантажувач підключається до веб-серверів, які, у свою чергу, читають / записують у базу даних і підключаються до сервера кешування для отримання об’єктів у пам'яті. Більшу частину часу, коли ми запускаємо такі стеки програм, нам потрібно налаштовувати ці компоненти в дуже конкретному порядку. На рис 4.1 показано схему поширеної трирівневої веб-програми, на якій запущено балансир навантаження, веб-сервер та серверну базу даних.

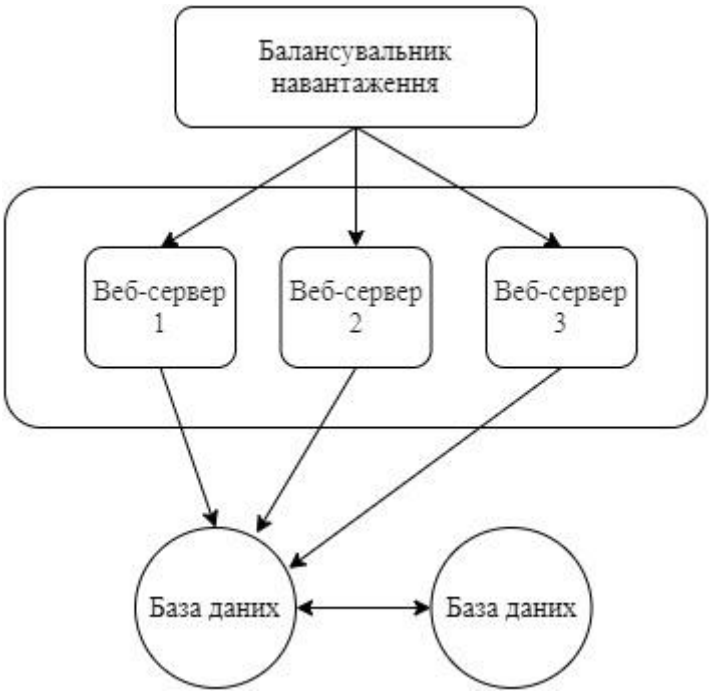

Рисунок 4.1 - Поширена трирівнева веб-програма, на якій запущено балансир навантаження, веб-сервер та серверну базу даних

Ansible дозволяє перекласти цю діаграму в проект, який визначає а=послідовність кроків для розгортання інфраструктури. Формат, що використовується для вказівки такої політики, - це те, що таке playbooks.

Послідовність політик, які застосовано у роботі, вказані в наступних кроках:

- для початку, встановлено, налаштовано та запущено службу MySQL на серверах баз даних;
- встановлено та налаштовано веб-сервери, на яких запущено Nginx, з прив'язками PHP;
- розгорнуто програму Wordpress на веб-серверах та додано відповідні конфігурації до Nginx;
- запущено службу Nginx на всіх веб-серверах після розгортання Wordpress, та встановлено, налаштовано та запущено службу haproxy на балансах навантаження. Оновлено конфігурації haproxy за допомогою імен хостів усіх веб-серверів, створених раніше.

На рис. 4.2 наведено зразок посібника, який перетворює план інфраструктури у політику, що застосовується Ansible:

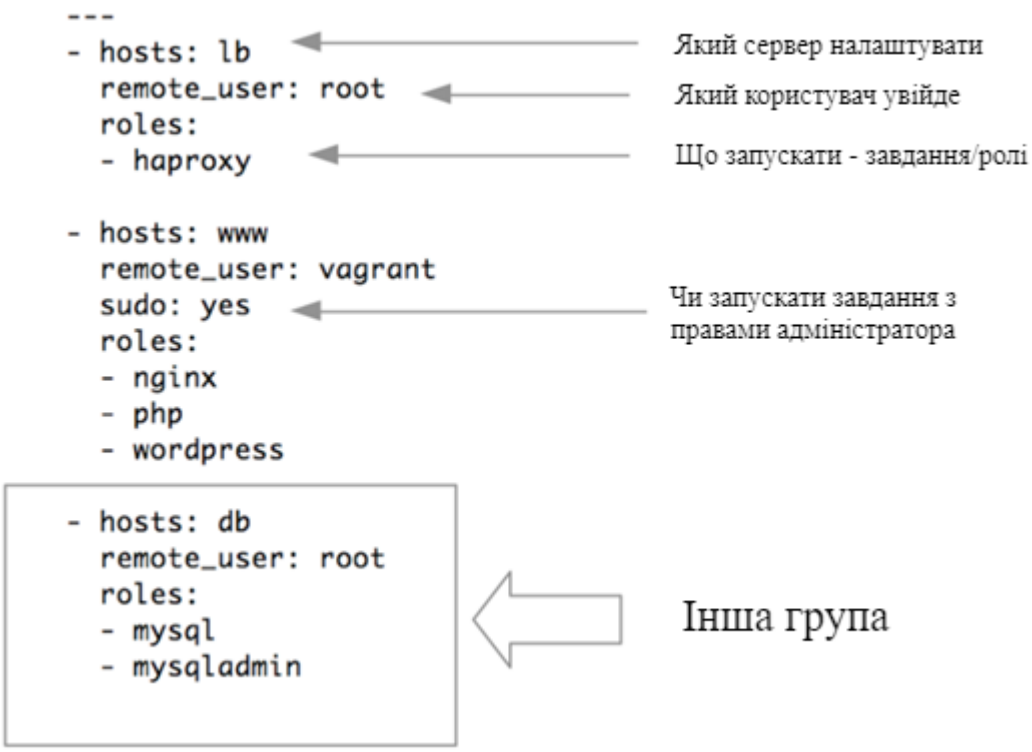

Рисунок 4.2 – Структура Ansible playbook

Playbook складається з однієї або декількох п'єс, які відображають групи ведучих до чітко визначених завдань. Попередній приклад містить три відтворення, кожна для конфігурації одного рівня в багатоурожайній веб-програмі. П'єси також визначають порядок конфігурування завдань, що дозволяє нам організувати розгортання багаторівневості. Наприклад, налаштуйте балансири навантаження лише після запуску веб-серверів або виконайте двофазне розгортання, де перша фаза лише додає ці конфігурації, а друга фаза запускає послуги у бажаному порядку.

Те, що було описано вище більше нагадує конфігурацію тексту, ніж фрагмент коду. Це тому, що творці Ansible вирішили використовувати простий, зручний для читання та звичний формат YAML для створення друку інфраструктури. Це додає привабливості Ansible, оскільки користувачі цього інструменту не повинні вивчати жодної спеціальної мови програмування, щоб розпочати. Код, що відповідає, за своєю суттю пояснює себе і самодокументує. Для розуміння основного синтаксису достатньо швидкого курсу збоїв на YAML. Деякі моменти, які потрібно знати про YAML, щоб розпочати роботу зі сво:

- перший рядок playbook повинен починатися з "---" (три дефіси), що вказує на початок документа YAML;
- списки в YAML представлені дефісом, після якого залишається пробіл. Playbook містить список п'єс; вони представлені знаком "-". Кожна гра є асоціативним масивом, словником або картою з точки зору пар ключ-значення.
- відступи є важливими;
- усі члени списку повинні мати однаковий рівень вказівки;
- кожна гра може містити пари ключ-значення, розділені символом ":" для позначення хостів, змінних, ролей, завдань тощо[10].

4.2 Розробка Ansible playbook

Щоб протестувати програмне забезпечення, було розгорнуто тестове середовище за допомогою Vagrant.

Vagrant - це інструмент для побудови та управління середовищами віртуальних машин за один робочий процес. Завдяки простому у використанні робочому процесу та зосередженню на автоматизації, Vagrant скорочує час налаштування середовища розробки, збільшує паритет виробництва та робить "роботи на моїй машині" виправданням пережитком минулого. Інфраструктура, яка необхідна, зображена на рис. 4.3.

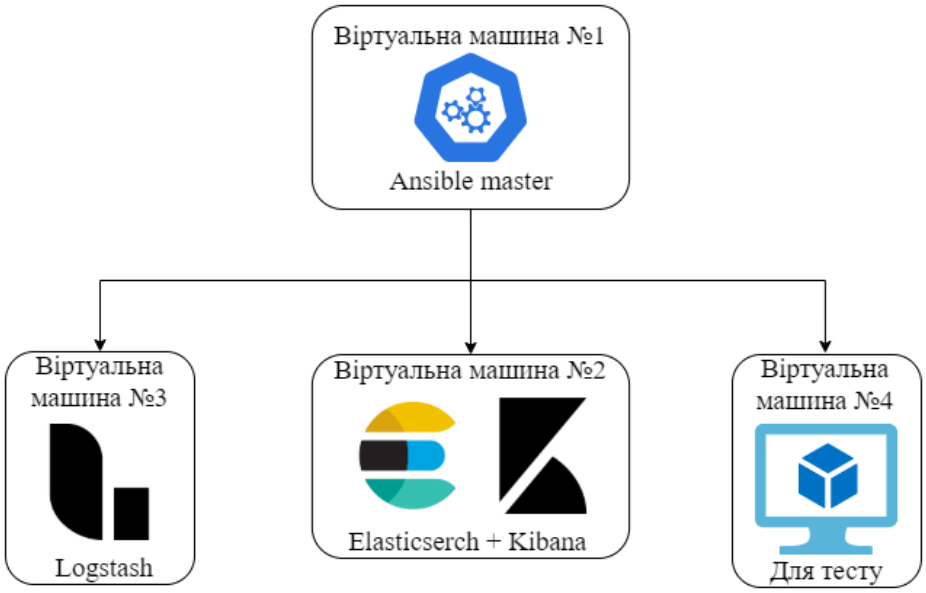

Рисунок 4.3 – Схема віртуального середовища

Далі наведено файл конфігурації, за допомогою якого було розгорнуто необхідну структуру.

```
Vagrant.configure("2") do |config|
  config.vm.define "ansible" do |ansible|
    ansible.vm.box = "ubuntu/bionic64"
    ansible.vm.box_check_update = false
    ansible.vm.hostname = "ansible"
  end
  config.vm.define "ek" do |ek|
    ek.vm.box = "ubuntu/bionic64"
    ek.vm.box_check_update = false
    ek.vm.hostname = "ek"
  end
  config.vm.define "logstash" do |logstash|
    logstash.vm.box = "ubuntu/bionic64"
    logstash.vm.box_check_update = false
    logstash.vm.hostname = "logstash"
  end
  config.vm.define "server1" do |server1|
```

```
    server1.vm.box = "ubuntu/bionic64"
    server1.vm.box_check_update = false
    server1.vm.hostname = "server1"
  end
end
```

За допомогою цього невеликого конфігураційного файлу, було розгорнуто чотири віртуальних машини. Master машина, з якої будуть проводитися всі налаштування. На другу віртуальну машину було встановлено Elasticserch та Kibana. На третю віртуальну машину, було встановлено Logstash. Четверта віртуальна машина використовувалася як тестова, з якої ми отримували логи.

4.2.1 Конфігурація необхідних компонентів для розгортання ELK

Для самостійного встановлення та конфігурації ELK Stack потрібно досить багато часу, а що більш критичніше людина, яка має достатньо великий досвід роботи з цією технологією та ОС.

Було розроблено сценарій, який використовується Ansible для автоматичного розгортання ELK на будь яку операційну систему за декілька хвилин. Для використання цього програмного забезпечення не потрібно мати багато досвіду, достатньо лише дотримуватися інструкції.

Перше, що необхідно зробити, встановити Ansible на хост, який буде використовуватися для менеджменту Elasticsearch, Logstash, and Kibana. В моїй інфраструктурі, такою машиною є master. Всі інші віртуальні машини є slave. В моєму тестовому середовищі, використано Ubuntu 18.04 як master.

Щоб встановити Ansible на Ubuntu, було виконано наступні команди:

- sudo apt-add-repository ppa:ansible/ansible;
- sudo apt update;
- sudo apt install ansible.

Встановлено все необхідне програмне забезпечення, необхідне для адміністрування серверів через Ansible.

Ansible відстежує всі сервери, про які йому відомо, за допомогою файлу «hosts», також його ще називають «inventory» файл. Спочатку потрібно налаштувати

цей файл, перш ніж ми зможемо почати спілкуватися з іншими нашими комп'ютерами.

Файл hosts досить гнучкий і може бути налаштований кількома різними способами. Синтаксис, який було використано, зображено на рис. 4.4.

```
[group_name]
alias ansible_ssh_host=your_server_ip
```

Рисунок 4.4 – Приклад host файла

[group_name] - це організаційний тег, який дозволяє посилатися на будь-які перелічені під ним сервери одним словом. Псевдонім - це просто назва для посилання на цей сервер.

Файл host, який було використано, зображено на рис.4.5.

```
ek ansible_host=192.168.43.186 ansible_user=vagrant ansible_ssh_private_key_file=/home/vagrant/.ssh/id_rsa
logstash ansible_host=192.168.43.204 ansible_user=vagrant ansible_ssh_private_key_file=/home/vagrant/.ssh/id_rsa
server1 ansible_host=192.168.43.252 ansible_user=vagrant ansible_ssh_private_key_file=/home/vagrant/.ssh/id_rsa
```

Рисунок 4.5 – Ansible host файл

Для доступу до віртуальних машин, master повинен мати SSH доступ до всіх slaves. Ключи SSH були згенеровані та покладені в відповідне місце на всіх машинах.

Не слід використовувати паролі. Хоча Ansible, безсумнівно, має можливість обробляти SSH-аутентифікацію на основі паролів, SSH-ключі допомагають зробити все просто.

Хости можуть бути в декількох групах, а групи можуть налаштовувати параметри для всіх своїх членів.

Було налаштовано всі хости, вони мають всі необхідні параметри, щоб успішно підключитися до хостів. Щоб провірити доступність всіх хостів, використовується модуль ping. Команда має наступний вигляд:

- ansible -m ping all.

Ключ «-m» вказує на Ansible модуль, який повинен використовуватися при виконанні команди. Результати цієї команди показано на рис.4.6.

```
vagrant@ansible:/vagrant/ansible$ ansible -m ping all -i inventory.txt
 [WARNING] Ansible is in a world writable directory (/vagrant/ansible),
logstash | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
ek | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
server1 | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
```

Рисунок 4.6 – Результати роботи Ansible модуля ping

Це базовий тест, щоб переконатися, що Ansible має зв’язок з усіма своїми хостами.

Частина команди -m ping - це інструкція Ansible використовувати модуль “ping”. В основному це команди, які можна запускати на віддалених хостах. Модуль ping працює багато в чому, як звичайна утиліта ping в Linux, але замість цього він перевіряє підключення Ansible.

На даний момент налаштовано сервер Ansible для зв’язку із серверами, над якими буде здійснюватися керування. Можемо переконатися, що Ansible може спілкуватися з кожним хостом, який знає, як використовувати команду ansible для віддаленого виконання простих завдань.

Також, потрібно згадати про role в Ansible. Адже це являється важливим елементом роботи. Ролі дозволяють автоматично завантажувати пов'язані файли vars_files, завдання, обробники та інші артефакти Ansible на основі відомої структури файлів. Згрупувавши вміст за ролями, дає можливість легко використовувати його повторно та надати йому доступ до інших користувачів.

Роль Ansible має визначену структуру каталогів із сімома основними стандартними каталогами. Необхідно включити принаймні один із цих каталогів у кожну роль. Дозволяється опустити будь-які каталоги, які роль не використовує. Це зображено на рис. 4.7.

```
# playbooks
site.yml
webservers.yml
fooservers.yml
roles/
    common/
        tasks/
        handlers/
        library/
        files/
        templates/
        vars/
        defaults/
        meta/
    webservers/
        tasks/
        defaults/
        meta/
```

Рисунок 4.7 – Структура Ansible ролі

За замовчуванням Ansible шукатиме у кожному каталозі в ролі файл main.yml для відповідного вмісту (також main.yaml та main):

- tasks/main.yml - основний перелік завдань, які виконує роль;
- handlers/main.yml - обробники, які можуть використовуватися в межах або поза цією роллю;
- library/my_module.py - модулі, які можуть використовуватися в рамках цієї ролі (див. розділ Вбудовування модулів та плагінів у ролі для отримання додаткової інформації);
- defaults/main.yml - змінні за замовчуванням для ролі (докладнішу інформацію див. у розділі Використання змінних). Ці змінні мають найнижчий пріоритет серед усіх доступних змінних і можуть бути легко замінені будь-якою іншою змінною, включаючи змінні запасів;
- - vars/main.yml - інші змінні для ролі (див. Використання змінних для отримання додаткової інформації);
- - files/main.yml - файли, які роль розгортає;

- - templates/main.yml - шаблони, які роль розгортає;
- - meta/main.yml - метадані про роль, включаючи рольові залежності.

Мною було створено декілька ролей, які необхідні для розгортання ELK. Для створення шаблону для ролі, було використану вбудовану команду Ansible:

- ansible-galaxy init «назва ролі».

Ця команда створює необхідну структуру директорій, яка була описана вище та файли які використовує Ansible. Розглянемо структуру однієї з ролей, яка була створена.

```
.
|-- elasticsearch
    |-- README.md
    |-- defaults
    |   `-- main.yml
    |-- files
    |   `-- elasticsearch.yml
    |-- handlers
    |   `-- main.yml
    `-- tasks
        |       `-- main.yml
```

Було створено декілька основних yml файлів, які використовується Ansible як інструкція, для розгортання ELK. Розглянемо ці файли.

Перший, використовується для встановлення Elasticsearch та Kibana в тестовому середовищі.

```
---
- name: Install Elasticsearch
  hosts: ek
  become: True
  roles:
    - java
    - elasticsearch
    - kibana
```

Розглянемо його більш детально. Кожен Ansible playbook повинен починатися з «---», також дуже важливо дотримуватися відступів між різними елементами сценарію. Це є дуже важливим правилом, інакше при запуску сценарію отримаємо синтаксичну помилку. Основні елементи цього сценарію:

- name – назва завдання яка буде виконуватися, використовується в вихідному журнал при виконанні сценарію;
- hosts – назва хосту або групи хостів на яких потрібно виконати даний playbook;
- become – якщо значення є true, всі команди будуть виконуватися з привілеями рута;
- roles – список ролів, котрі повинні визватися та виконатися в даному сценарії. В нашому випадку, використовується java, elasticsearch та kibanaю.

Розглянемо більш детально роль elasticsearch.

```
---
# tasks file for elasticsearch
- name: Update
  apt:
    name: aptitude
    update_cache: yes
- name: import the Elasticsearch public GPG key into APT
  apt_key:
    url: https://artifacts.elastic.co/GPG-KEY-elasticsearch
    state: present
- name: add the Elastic source list to the sources.list.d
  shell: echo "deb https://artifacts.elastic.co/packages/6.x/apt stable main" | sudo tee -
a /etc/apt/sources.list.d/elastic-6.x.list
- name: Update
  apt:
    name: aptitude
    update_cache: yes
- name: Install elasticsearch
  apt:
```

```
    name: elasticsearch
    state: latest
- name: Copy file with configuration
  copy:
    src: "{{ config_file }}"
    dest: "{{ elasticsearch_folder }}"
    owner: root
    group: elasticsearch
    mode: 0660
- name: Starting and enable elasticsearch
  service:
    name: elasticsearch
    state: started
    enabled: yes
```

Вище було наведено такий елемент ролі як elasticsearch/task/main.sh. Він має вигляд звичайного Ansible playbook. Головним завданням цього сценарію є встановлення та налаштування elasticsearch.

В файлі /elasticsearch/defaults/main.yml знаходяться параметри, які необхідні для успішно виконання даного сценарію.

```
---
config_file:          elasticsearch.yml
elasticsearch_folder: /etc/elasticsearch/elasticsearch.yml
```

Ці параметри вказують на назву конфігураційного файлу та шлях куди його потрібно перемістити. В директорії elasticsearch/files/ знаходяться файли, які білу будуть скопійовані на цільову машину.

Конфігураційний файл зображено на рис.4.8.

```
# Path to directory where to store the data (separate multiple locations by comma):
path.data: /var/lib/elasticsearch
#
# Path to log files:
path.logs: /var/log/elasticsearch
```

Рисунок 4.8 – Конфігураційний файл Elasticsearch

Цей конфігураційний файл було скопійовано на ek хост в /etc/elasticsearch/elasticsearch.yml.

4.2.2 Автоматичне розгортання та конфігурація ELK

Розгорнута мінімальна інфраструктура, яка необхідна для тестування ELK. Використовуючи Ansible playbook було встановлено та налаштовано ELK. Для цього були виконано декілька кроків.

Спершу, було запущено встановлення та конфігурація Elasticsearch і Kibana, це показано на рис.4.9.

```
vagrant@ansible:/vagrant/ansible$ ansible-playbook install_ek.yml -i invento
ry.txt
 [WARNING] Ansible is in a world writable directory (/vagrant/ansible), igno
ring it as an ansible.cfg source.

PLAY [Install Elasticsearch] ***************************************************
****

TASK [Gathering Facts] *********************************************************
****
ok: [ek]

TASK [java : Update] ***********************************************************
****
```

Рисунок 4.9 – Запуск конфігурації Elasticsearch і Kibana

Цією командою було успішно встановлено Elasticsearch і Kibana на один з серверів, який був створений раніше. На рис 4.10 продемонстровано успішне закінчення даного процесу.

```
TASK [kibana : Starting and enable nginx] **************************************
****
ok: [ek]

TASK [kibana : service] ********************************************************
****
changed: [ek]

PLAY RECAP *********************************************************************
****
ek                         : ok=21   changed=17   unreachable=0    failed=0
```

Рисунок 4.10 – Журнал успішного встановлення Elasticsearch та Kibana

Далі необхідно встановити Logstash на інший сервер. Цей процес продемонстровано на рис. 4.11.

```
vagrant@ansible:/vagrant/ansible$ ansible-playbook install_logstash.yml -i inventory.txt
 [WARNING] Ansible is in a world writable directory (/vagrant/ansible), ignoring it as an

PLAY [Install Logstash] ********************************************************************

TASK [Gathering Facts] *********************************************************************
ok: [logstash]

TASK [java : Update] ***********************************************************************
changed: [logstash]

TASK [java : Install java] *****************************************************************
ok: [logstash]

TASK [logstash : Update] *******************************************************************
ok: [logstash]

TASK [logstash : Add the GPG key to install signed packages] *******************************
```

Рисунок 4.11 – Встановлення Logstash

Команда, яка використовується для встановлення Logstash має схожий вигляд з минулою, єдина різниця, було використано install_logstash.yml playbook, який має інший набір ролів.

Останній крок, який зроблено, встановлено filebeat. Filebeat - це легкий відправник для пересилання та централізації даних журналу. Встановлений як агент на серверах, Filebeat контролює вказані файли журналів або розташування, збирає події журналу та передає їх до Elasticsearch або Logstash для індексації.

Ось як працює Filebeat: коли запускається Filebeat, він запускає один або кілька входів, які шукають у вказаних місцях для даних журналу. Для кожного журналу, який Filebeat знаходить, Filebeat запускає комбайн. Кожен комбайн читає один журнал для нового вмісту та надсилає нові дані журналу до libbeat, який агрегує події та надсилає агреговані дані до виводу, який налаштований для Filebeat.

Процес встановлення Filebeat зображено на рис.4.12. Для цього було використано інший шаблон за набором необхідних ролей.

```
vagrant@ansible:/vagrant/ansible$ ansible-playbook install_filebeat.yml -i inventory.txt
 [WARNING] Ansible is in a world writable directory (/vagrant/ansible), ignoring it as an

PLAY [Install Filebeat] ********************************************************************

TASK [Gathering Facts] *********************************************************************
ok: [server1]

TASK [filebeat : Run update machine] *******************************************************
ok: [server1]

TASK [filebeat : Import the Elasticsearch public GPG key into APT] *************************
ok: [server1]

TASK [filebeat : Add the Elastic source list to the sources.list.d] ************************
changed: [server1]

TASK [filebeat : Install Filebeat] *********************************************************
ok: [server1]

TASK [filebeat : Run update machine] *******************************************************
changed: [server1]

TASK [filebeat : Install filebeat] *********************************************************
ok: [server1]
```

Рисунок 4.12 – Процес встановлення Filebeat

Все працює, як очікувалося, все, що залишилося зробити, це отримати доступ до Kibana за такою URL-адресою:

- http://<serverhost>:5601.

Відкриється Kibana рис.4.13, яка буде мати дані з тестової машини.

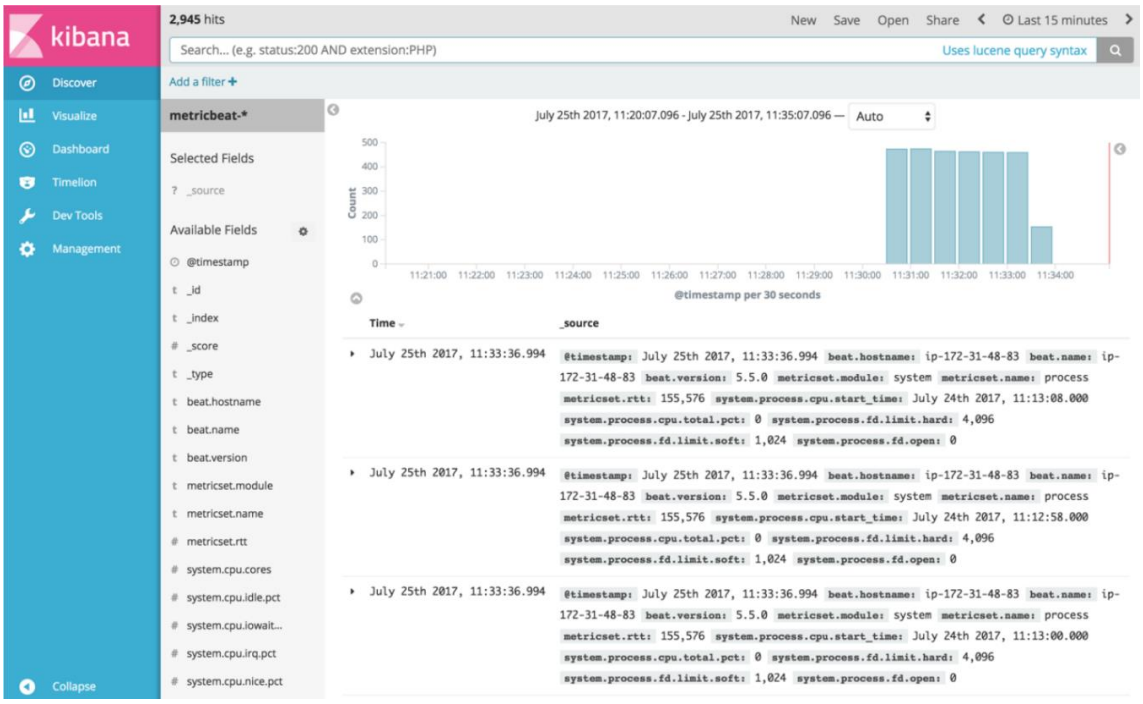

Рисунок 4.13 - Інтерфейс Kibana

Ось так, лише запустивши декілька команд, було налаштовано ELK Stack. Застосування такого підходу дає можливість використовувати найновіші функції Elasticsearch, Logstash, Kibana не марнувати час на ручну конфігурацію.

4.3 Аналіз інструментів для розробки ELK в системах управління контентом

Щоб розгорнути ELK, було налаштовано відповідну інфраструктуру за допомогою програми Vagrant. Існують інші альтернативи, такі як Chef та Puppet.

Три інструменти конфігурації прості у використанні, але при цьому надають надійні можливості для автоматизації складних багаторівневих IT-середовищ.

Відмінності між Ansible, Chef та Puppet зображуються на основі різних факторів, включаючи доступність, мову конфігурації, налаштування та встановлення, простоту управління, масштабованість та сумісність,.

Наявність трьох інструментів конфігурації у випадку відмови основного майстра або сервера дуже висока. Кожен інструмент має резервний сервер або альтернативний майстер для надання підтримки у разі відмови основного сервера. В таблиці 4.1 наведено результати порівняння.

Таблиця 4.1 – Порівняння доступності інструментів

Інструмент	Доступність (у разі відмови сервера)
Chef	Резервний сервер
Puppet	Альтернативний майстер
Ansible	Вторинна машина

Мова конфігурації відіграє помітну роль у визначенні застосувань інструменту управління конфігурацією, результати наведено в таблиці 4.2.

Таблиця 4.2 – Порівняння складності мов конфігурації

Інструмент	Мова конфігурації	Складність навчання
Chef	Ruby DSL	Важко
Puppet	Ruby, Puppet DSL, Embedded Ruby (ERB), DSL	Важко
Ansible	Python, YAML	Простий

Що стосується простоти налаштування та встановлення, Ansible домінує над двома іншими інструментами, оскільки має архітектуру без агента. Chef та Puppet слідують архітектурі ведучого-агента або ведучого-раба, результати наведено в таблиці 4.3.

Таблиця 4.3 – Порівняння простоти налаштування

Інструмент	Architecture	Простота встановлення та встановлення
Chef	Master-Agent	Складний через Chef Workstation
Puppet	Master-Agent	Складно через підписання сертифіката між master та agent
Ansible	Тільки Master	Легко

Існує два типи конфігурацій, "pull" та "push". Конфігурація pull передбачає витягування всіх конфігурацій з центрального сервера на підлеглі вузли без будь-яких команд. Тоді як у push-конфігурації всі конфігурації на сервері будуть передані до вузлів за допомогою певних команд. З точки зору мови конфігурації, YAML вважається найпростішою, оскільки вона схожа на англійську та читається людиною. Знову Ansible демонструє своє домінування над іншими з точки зору управління, оскільки підтримує мову YAML і дотримується як конфігурацій push і pull, результати наведено в таблиці 4.4.

Таблиця 4.4. – Порівняння простоти управління

Інструмент	Конфігурація	Простота управління
Chef	Pull	Важко
Puppet	Pull	Важко
Ansible	Push and Pull	Легко

Масштабованість інструментів конфігурації є одним з основних факторів, що враховуються підприємствами перед вибором інструменту.

Chef, Puppet та Ansible здатні управляти великими інфраструктурами, одночасно справляючись із тягарем у масштабуванні конфігурацій. Однак між ними є невелика різниця щодо масштабованості через складність мови конфігурації, результати наведені в таблиці 4.5.

Таблиця 4.5 – Порівняння масштабованості

Інструмент	Масштабованість
Chef	Висока
Puppet	Висока
Ansible	Дуже високо

Огляд можливостей продукту кожного інструменту управління конфігурацією може допомогти у виборі найбільш підходящого інструменту для потреб. Кожен інструмент має свій набір можливостей, які є кращими по-своєму. Порівняльна характеристика наведена в таблиці 4.6.

Таблиця 4.6 – Порівняння можливостей інструментів

Можливості інструменту		
Chef	Puppet	Ansible
- безперервна доставка з автоматизованим робочим процесом; - відповідність та управління безпекою; - Автоматизація інфраструктури.	- Оркестровка; - Автоматизоване забезпечення; - Управління кодами та вузлами; - Автоматизація конфігурації; - Проста візуалізація та звітування; - Висока прозорість; - Рольовий контроль доступу.	- Проста оркестровка; - Оптимізоване забезпечення; - Безперервна доставка з автоматизованим робочим процесом; - Розгортання програми; - Інтеграція безпеки та відповідності в автоматизовані процеси.

Всі три інструменти, включаючи Chef, Puppet та Ansible, мають свої переваги і є кращими по-своєму. Кожен інструмент має дещо інший підхід до автоматизації та підходить для певних сегментів користувачів в одному цільовому просторі.

Та враховуючи різні технічні та ділові аспекти, включаючи архітектуру, функції, зручність використання та підтримку, найкращим інструмент управління конфігурацією ELK є Ansible. Він простий у використанні та має всі необхідні функції.

4.4 Аналіз ефективності розробленого програмного забезпечення в порівнянні з існуючими технологіями

Встановити та налаштувати ELK можна різними способами. Одне з базових рішень, це конфігурувати кожний компонент окремо. Це займає багато часу і потрібно мати досить високі технічні навички. Крім того, часто можна наштовхнутися на несумісність необхідних пакетів з вже встановленими. Також, якщо сервери мають різну операційну систему, це може лише ускладнити і так не просту задачу. Якщо підсумувати, без висококваліфікованого спеціаліста зробити це н можливо.

Інший спосіб налаштувати ELK, є встановлення його в контейнері. Контейнеризація за допомогою docker дозволяє вирішити більшу частину проблем, які мав попередній підхід. Додаток розбивається на компоненти за функціями, які мають індивідуальну упаковку з залежностями, а потім можуть бути розгорнуті на архітектурі відмінною від стандартної. Дана можливість спрощує відновлення окремих компонент і масштабування додатки в цілому.

Використання даного методу має наступні плюси:

- абстрагування додаток від хоста, повна стандартизація. Контейнер з'єднується з хостом певним інтерфейсом, контейнерізорований додаток не залежить від архітектури або ресурсів хоста. Для хоста же контейнер якийсь «чорний ящик», не має значення що в ньому;
- масштабування, в продовження переваги абстрагування додатка від хоста, є можливість простого і лінійного маштабування. Тобто на одній машині може бути запущено декілька контейнерів в той же час вони можуть бути запущені і на тестовому сервері. В продуктивної середовищі вони так само можна їх маштабувати;
- управління версіями і залежностями, завдяки використанню контейнерів, розробник прив'язує всі компоненти і залежності до додатка, що дозволяє працювати як цілісним об'єктом. На хості не потрібні установки додаткових компонент або залежність для запуску програми, яка знаходиться всередині контейнера, досить можливості запуску докер контейнера;

- ізолювання середовища, ізоляція в контейнері не досягає такого ж рівня як в віртуалізації, але тим не менше має легку середовищем виконання і відноситься до ізоляції на рівні процесів. При цьому контейнер працює на тому ж самому ядрі, що позвоялет йому дуже швидше запускатися. Запуск сотні контейнерів на робочій машині не буде проблемою.

Проте крім значних переваг, даний метод має наступні недоліки:

- тонке налаштування, при великих масштабованості і навантаженні необхідно дуже чітка і якісна настройка систем. Комплекс ELK у великих системах потребує значну кількість ресурсів;
- зворотна сумісність, докер швидко розвивається і одним з мінусів такого розвитку буває обмежена зворотна сумісність з деякими напрямки. Таким чином може бути несумісність деяких компонентів ELK з новою версією докера;
- продуктивність, додаткові надбудови на системі в будь-якому випадку призводить до збільшення навантаження і витраті ресурсів;
- архітектура, контейнеризація являє собою надбудову над ОС, тим самим ускладнюючи реалізацію завдання;
- підтримка, для підтримки і супроводу докер контейнерів необхідно не тільки навички системного адміністратора, але і хороші знання docker.

Використання розробленого програмного забезпечення вирішує всі вище перелічені недоліки, і є чудовим рішенням для не великих проектів. Ansible в порівнянні з іншими інструментами є досить простим, проте має весь необхідний функціонал.

Звісно, програмне забезпечення та підхід який я пропоную не є ідеальним і також має свої недоліки. Одним з них є те, що потребує хоч не значної але все ж таки первісної конфігурації. Потрібно задати ip адреса або DNS імена машин, на які потрібно встановити ELK Stack. Також, необхідно налаштувати SSH ключи. Звісно, можно обійтися без цього і використовувати паролі, але набагато краще та безпечніше використовувати ключи.

Нижче приведено таблицю 4.7, де показано затрачений часу на встановлення та конфігурацію ELK за допомогою Docker.

Таблиця 4.7 – Результати затраченого часу на встановлення та конфігурацію ELK за допомогою Docker

№	Завдання	Витрачений час
1	Завантаження базового образа ELK	10 хвилин
2	Перезбірка контейнерів з необхідними конфігураційними файлами	30 хвилин
3	Час на запуск контейнерів	10 хвилин
	Сумарний час	40 хвилин

Також наведено таблицю 4.8 , де показано затрачений часу на встановлення та конфігурацію ELK за допомогою розробленого програмного забезпечення. Для отримання цих даних використовувалося теж саме обладнання та мережа інтернет, що й попередніх результатах.

Таблиця 4.8 – Результати затраченого часу на встановлення та конфігурацію ELK за допомогою розробленого програмного забезпечення

№	Завдання	Витрачений час
1	Створення hosts файлу	2 хвилини
2	Додавання SSH ключів	5 хвилин
3	Час за який відбувається розгортання	8 хвилин
4	Сумарний час	15 хвилин

Можна зробити висновок, що використання розробленого підходу є значно ефективнішим ніж використання Docker контейнерів та потребує на багато менше часу. Крім того, сам процес контейнерізації потребує додаткових ресурсів від хостової машини, що збільшує плату за використання.

ВИСНОВКИ

Розглянуто типову архітектуру систем білінгу в телекомунікаціях, де проаналізовано роль кожного елементу системи взаємодії. Виявлено, що існують два варіанти включення першого рівня CRM/OMOF, які використовуються розробниками. Особливістю приведеної архітектури є використання CDR від різних комутаторів, що мають унікальний формат.

Розглянуто принцип роботи контактного центру та проведено аналіз трафіку з використанням моделей Erlang B та Erlang C. Доведено, що модель Erlang C можна використовувати для розрахунків усіх необхідних і важливих параметрів руху контактного центру, однак її недоліком є відсутність розгляду IVR у контактному центрі, а в моделі Erlang B зі збільшенням навантаження і постійною кількістю агентів контакт-центр відносно швидко перевищує максимальну ймовірність відхилення виклику зі значенням 2,5 відсотка.

Розглянуто характеристики інструментів стеку ELK Stack, що складається з проектів Elasticsearch, Logstash та Kibana, з відкритим кодом для пошуку, аналізу та візуалізації в режимі реального часу та управління журналами наступного покоління. З огляду на загальну структуру ELK Stack встановлено, що його робота необхідна для різноманітних випадків використання та цілей, а це - розробка, моніторинг, безпека у відповідності до SEO та BI.

В атестаційній роботі розроблено програмне забезпечення для автоматичного розгортання універсальної системи білінгу ELK. Вказано послідовність політик, що застосовано у роботі, а саме: встановлення служби Filebeat, Java, розгортання програм Elasticsearch, Logstash та Kibana.

Розроблено сценарій для автоматичного розгортання ELK на будь яку операційну систему за десять хвилин, який використовує інструмент Ansible. Приведено переваги цього інструменту в порівнянні з існуючими: Puppet, Chef. Доведено, що вказані модулі потребують доопрацювання. А велика кількість модулів дає втрату в часі при налаштуванні.

Проаналізовано використання інструментів для розробки ELK в системах управління контентом щодо доступності, використання мов конфігурації, простоти

налаштування та управління, а також можливості кожного з них. Доведено, що деякі модулі потребують доопрацювання. А велика кількість модулів дає втрату в часі при налаштуванні.

В атестаційній роботі було проведено автоматичне розгортання та налаштування конфігурації ELK, про що свідчать журнал успішного встановлення системи та заключний процес встановлення Filebeat, а також інтерфейс Kibana з параметрами тестової машини.

Для аналізу ефективності даної розробки проведено порівняння способів встановлення та налаштування в системах управління контентом. Доведено, що використання розробленої програми дає виграш в 62% сумарного часу на розгортання ELK за меншу кількість кроків у порівнянні з методами контейнеризації.

В роботі приведено конфігураційні та додаткові файли для встановлення та налаштування програмного забезпечення ELK Stack для систем управління контентом (Додатки А-К).

Окремі результати роботи доповідались на трьох Міжнародних наукових конференціях [11, 12, 13].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Joshila Grace L.K., Maheswari V. and Nagamalai Dhinaharan Web Log Data Analysis and Mining / L.K. Joshila Grace, V. Maheswari and Dhinaharan Nagamalai // International Scientific and Technical Conference Computer Sciences and Information Technologies (CCSIT-2011), Vol. 2, 2011. – P. 459-469.

2. Shen H., Huang J., Lee Ch. Forecasting and Dynamic Updating of Uncertain Arrival Rates to a Contact Center / H. Shen, J. Huang, Ch. Lee // In: IEEE International Conference on Service Operations and Logistics and Informatics, 2007. – P. 105-108.

3. Miguth T., Barofak I. Performance Forecast of Contact Centre with Differently Experienced Agents / T. Miguth, I. Barofak // In: Elektrorevue, ISSN 1213-1539, Vol. 15, 2010. – 57 p.

4. Nayan B., Ruparelia A. Software Development Lifecycle Models // B. Nayan, A. Ruparelia // International Scientific Conference, 2010. – 11 p.

5. Диби В. Н. Моделювання контакт-центрів з відкладеним обслуговуванням заявок на інформаційні послуги / В. Н. Диби // Изв. Вузів Росії, Радіоелектроніка, СПБ .: № 3, 2010. – С. 55- 57.

6. Steinset B., January K. Traffic a Data Collection in Botswana / B. Steinset, K. January // Fact Finding Appraisal report. NPRA, 2008. – P. 193-198.

7. Ваняшин С. Математичні моделі центрів обслуговування викликів / С. Ваняшин, А. В. Росляков. – М .: Іріас, 2006. – 48 с.

8. Крестьянинов С.В. Міжнародна стандартизація комп'ютерної телефонії / С.В. Крестьянинов // Технології і засоби зв'язку, №5, 2000. – С. 82-85.

9. Офіційний сайт інструмента ELK [Електронний ресурс]. – 2014. –Режим доступу до ресурсу: https://www.elastic.com.

10. Офіційний сайт інструмента Ansible [Електронний ресурс]. – 2016. – Режим доступу до ресурсу: https://docs.ansible.com.

11. Лябах А.С. Науковий керівник – доц., к.т.н. Токарь Л.А. Порівняльна характеристика протоколів IP-телефонії / А.С.Лябах, Л.О. Токар // Матеріали 23-го Міжнародного молодіжного форуму «Радіоелектроніка та молодь у XXI столітті». Зб. Матеріалів форуму. Т.4. – Харків: ХНУРЕ. – 2018. –С. 38-39.

12. Лябах А.С. Хачиров Е.Ф. Науковий керівник – доц., к.т.н. Токарь Л.А. Особливості використання комплексу програм ELK STACK / А.С.Лябах, Е.Ф.Хачиров, Л.О. Токар // Материалы 24-го Міжнародного молодіжного форуму «Радіоелектроніка та молодь у XXI столітті». Зб. Матеріалів форуму. Т.4. – Харків: ХНУРЕ. – 2020. – С. 20-21.

13. Токар Л.О., Красноженюк Я.О., Лябах А.С., Кодаченко Р.Ю. Особливості роботи стандарту 802.11 ac з використанням технології Airtime Fairness / Л.О. Токар, Я.О. Красноженюк, А.С. Лябах, Р.Ю. Кодаченко // Матеріали шостої Міжнародної науково-технічної конференції «Проблеми електромагнітної сумісності перспективних бездротових мереж зв'язку ЕМС- 2020». – Харків, ХНУРЕ. Том 4. – 2020. – С. 66-68.