Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних систем та мереж
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

Магістр
(назва освітнього ступеня)

на тему: **Методи та засоби розгортання артефактів екосистеми опрацювання великих даних Hadoop**

Виконав: студент(ка) 6 курсу, групи СІм-62
спеціальності 123 «Комп'ютерна інженерія»
(шифр і назва спеціальності)

(підпис) Мельник Н. О. (прізвище та ініціали)

Керівник (підпис) Луцків А. М. (прізвище та ініціали)

Нормоконтроль (підпис) Луцик Н. С. (прізвище та ініціали)

Завідувач кафедри (підпис) Осухівська Г. М. (прізвище та ініціали)

Рецензент (підпис) Бревус В. М. (прізвище та ініціали)

Тернопіль
2024

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних систем та мереж
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ *Осухівська Г.М.*
(підпис) (прізвище та ініціали)
«11» листопада 2024 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня *магістр*
(назва освітнього ступеня)

за спеціальністю *123 «Комп'ютерна інженерія»*
(шифр і назва спеціальності)

студенту *Мельнику Назарію Олександровичу*
(прізвище, ім'я, по батькові)

1. Тема роботи *Методи та засоби розгортання артефактів екосистеми опрацювання великих даних Hadoop*

Керівник роботи *Луцків Андрій Мирославович, к.т.н., доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « *29* » *листопада 2024* року № 4/7-1144
2. Термін подання студентом завершеної роботи *16.12.2024р.*
3. Вихідні дані до роботи

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ
1 Огляд екосистеми опрацювання великих даних Hadoop та аналіз методів розгортання її артефактів
2 Вибір методів та інструментів розгортання артефактів екосистеми Hadoop. Теоретичне обґрунтування
3 Практичне дослідження та реалізація розгортання артефактів екосистеми Hadoop
4 Охорона праці та безпека в надзвичайних ситуаціях
Висновки
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Актуальність і мета дослідження. завдання, об'єкт і предмет, наукова новизна і практична цінність дослідження. 2. Загальний порівняльний аналіз методів розгортання артефактів екосистеми Hadoop. 3. Високорівнева архітектура Apache Ambari.
4. Порівняння основних характеристик інструментів Ansible, Puppet та Chef.
5. Схема розгортання артефктів екосистеми опрацювання великих даних Hadoop.
6. Переваги та недоліки методів налаштування артефактів екосистеми Hadoop з використанням Apache Bigtop. 7. Хости та розгорнуті на них артефакти екосистеми Hadoop.
8. Висновки

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	*Осухівська Г.М., зав.каф. КС*	*05.12.2024*	*20.12.2024*
Безпека в надзвичайних ситуаціях	*Стручок В. С., ст. викладач каф. обладнання харчових технологій*	*10.12.2024*	*19.12.2024*

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	*Аналіз існуючих методів та засобів розв'язання науково-технічних проблем, що відповідають тематиці кваліфікаційної роботи.*	*20.11.2024р. – 30.11.2024р.*	*Викон.*
2.	*Огляд екосистеми опрацювання великих даних Hadoop та аналіз методів розгортання її артефактів*	*01.12.2024р. – 02.12.2024р.*	*Викон.*
3.	*Вибір методів та інструментів розгортання артефактів екосистеми Hadoop. Теоретичне обґрунтування*	*03.12.2024р. – 06.12.2024р.*	*Викон.*
4.	*Практичне дослідження та реалізація розгортання артефактів екосистеми Hadoop*	*07.12.2024р. – 12.12.2024р.*	*Викон.*
5.	*Охорона праці та безпека в надзвичайних ситуаціях*	*13.12.2024р.*	*Викон.*
6.	*Оформлення пояснювальної записки і графічного матеріалу*	*14.12.2024р. – 15.12.2024р.*	*Викон.*
7.	*Попередній захист кваліфікаційної роботи магістра*	*16.12.2024р.*	*Викон.*
8.	*Захист кваліфікаційної роботи магістра*	*23.12.2024р.*	*Викон.*

Студент _____ *Мельник Н. О.*
(підпис) (прізвище та ініціали)

Керівник роботи _____ *Луцків А. М.*
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Мельник Н.О. Методи та засоби розгортання артефактів екосистеми опрацювання великих даних Hadoop: робота на здобуття кваліфікаційного ступеня магістра: спец. 123 — комп'ютерна інженерія / наук.кер. А. М. Луцків. Тернопіль: Тернопільський національний технічний університет імені Івана Пулюя, 2024. — 71с.

Ключові слова: Великі Дані, Hadoop, артефакт, розгортання, Bigtop, Ambari

Кваліфікаційна робота присвячена дослідженню методів і засобів розгортання артефактів екосистеми опрацювання великих даних Hadoop. Робота включає в себе аналіз існуючих досліджень у сфері розгортання екосистеми Hadoop, а також акцентує увагу на можливостях використання комбінованого методу для розгортання та тестування власних артефактів сервісів Hadoop.

Під час виконання кваліфікаційної роботи проведено детальний теоретичний огляд, аналіз та порівняння наявних на даний момент методів і інструментів розгортання. На основі зробленого теоретичного аналізу запропоновано новий метод розгортання екосистеми Hadoop, що використовує поєднання декількох з інструментів, спрямованих на виконання поставленого завдання.

Проведено експериментальні дослідження розгортання артефактів, використовуючи засоби та інструменти, які було обрано при теоретичному аналізі для реалізації запропонованого комбінованого методу розгортання.

Результати проведеного дослідження можуть мати велике теоретичне та практичне значення для полегшення розгортання сформованих артефактів, що налаштовані під специфічні вимоги поставлених завдань, а також забезпечення високої гнучкості, масштабованості екосистми Hadoop та зручності використання.

ANNOTATION

Melnyk N.O. Methods and tools for deploying artifacts in the Hadoop Big Data processing ecosystem. Master's Graduation Thesis: speciality 123 - Computer engineering/supervisor A.M. Lutskiv. Ternopil: Ternopil Ivan Puluj National Technical University, 2024. — 71p.

Keywords: Big Data, Hadoop, artifact, Deployment, Bigtop, Ambari

The Master's graduation thesis is dedicated to to researching methods and tools for deploying artifacts within the Hadoop big data processing ecosystem. The work includes an analysis of existing studies in the field of Hadoop ecosystem deployment and emphasizes the potential of utilizing a combined approach for deploying and testing custom Hadoop service artifacts.

During the preparation of the thesis, a detailed theoretical review, analysis, and comparison of currently available deployment methods and tools were conducted. Based on the theoretical analysis, a new Hadoop ecosystem deployment method was proposed, which combines several tools to address the defined objectives.

Experimental studies were carried out to deploy artifacts using the tools and methods selected during the theoretical analysis to implement the proposed combined deployment method.

The results of the study may have significant theoretical and practical value by facilitating the deployment of tailored artifacts configured to meet specific task requirements. Additionally, they contribute to enhancing the flexibility, scalability, and usability of the Hadoop ecosystem.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

CI/CD (англ. Continuous Integration/Continuous Delivery) неперервна інтеграція/неперервна доставка

HDFS (англ. Hadoop Distributed File System) Розподілена файлова система Hadoop

SQL (англ. Structured Query Language) мова структурованих запитів

YARN (англ. Yet Another Resource Negotiator) платформа керування обчислювальними ресурсами кластера для розподілених обчислень

ВСТУП

Актуальність роботи. У світі стрімкого розвитку технологій та безпрецедентного обсягу інформації, управління та обробка даних стають критично важливими завданнями. В цьому контексті Apache Hadoop, як розподілена система обробки даних, виходить на передній план як потужний інструмент для вирішення завдань, пов'язаних із великими обсягами інформації. Сьогодні обробка великих даних є важливим компонентом розвитку інновацій у багатьох галузях, таких як фінанси, охорона здоров'я, телекомунікації, маркетинг і наукові дослідження. Кожна з цих галузей вимагає певні конкретні налаштування або використання специфічних сервісів для виконання різних завдань. У цьому випадку на передній план виходять додаткові налаштування сервісів або створення власних артефактів (сформованих інсталяційних пакетів сервісів) екосистеми Hadoop.

Попри широке застосування, розгортання компонентів екосистеми Hadoop залишається складним і ресурсозатратним процесом. Він потребує врахування багатьох факторів, таких як конфігурація кластерів, забезпечення узгодженості даних, інтеграція з іншими системами та моніторинг продуктивності. Ручне налаштування часто є джерелом помилок і затримок, особливо у великих кластерах з сотнями або тисячами вузлів.

Окрім цього, у зв’язку з динамічним розвитком технологій автоматизації, організації прагнуть мінімізувати людське втручання у процеси розгортання, щоб забезпечити стабільність і знизити ризик помилок.

З огляду на це, обрана тема дослідження є актуальною, адже її результати дозволять значно спростити та прискорити впровадження таких систем у реальних умовах, що сприятиме підвищенню їхньої ефективності та економічної доцільності.

Метою кваліфікаційної роботи виступає забезпечення високої продуктивності, надійності, більшої незалежності від зовнішніх компаній та розробників сервісів екосистеми опрацювання великих даних Hadoop, що пропонують власні програмні рішення, а також забезпечення можливості повного

налаштування кожного компонента кластера Hadoop під власні потреби компаній для виконання конкретних поставлених завдань і оптимізації їх виконання.

Завдання кваліфікаційної роботи:

– провести огляд літературних джерел та проаналізувати методи та засоби розгортання артефактів екосистеми Hadoop;

– дослідити наявні програмні рішення та інструменти для побудови, пакування, тестування та розгортання артефактів екосистеми Hadoop;

– оптимізувати процес розгортання власних збудованих артефактів екосистеми Hadoop, використовуючи вибрані наявні програмні рішення та інструменти;

Відповідно до мети та завдань кваліфікаційної роботи визначено її об'єкт та предмет.

Об'єкт дослідження: процеси та технології розгортання артефактів екосистеми опрацювання великих даних Hadoop.

Предмет дослідження: методи та засоби автоматизованого розгортання власних артефактів екосистеми опрацювання великих даних Hadoop.

Методи дослідження: У ході виконання використано низку методів дослідження, які дозволили досягти поставленої мети. Для аналізу існуючих рішень щодо розгортання артефактів екосистеми Hadoop застосовано метод теоретичного аналізу літературних джерел, документації та інструментів автоматизації. На етапі вибору оптимальних методів та інструментів для розгортання використано метод систематизації й порівняння, що дозволило оцінити ефективність різних підходів. Практичну реалізацію виконано за допомогою експериментального методу, що включав створення тестового середовища, впровадження автоматизованого розгортання артефактів та аналіз отриманих результатів. Такий підхід забезпечує логічність і прийнятність обраних методів у контексті вирішення поставлених завдань.

Наукова новизна дослідження. Запропоновано новий підхід для автоматизації розгортання та налаштування конфігурацій власних артефактів екосистеми Hadoop з урахуванням особливостей налаштування та створення нових сервісів.

Практичне значення результатів кваліфікаційної роботи. Досліджувані методи та засоби, а також новий запропонований підхід для розгортання власних артефактів екосистеми Hadoop можуть бути використані в підприємствах та організаціях, які працюють з великими обсягами даних та мають на меті використовувати власні збудовані артефакти сервісів екосистеми Hadoop для виконання специфічних потреб і завдань та (або) для забезпечення більшої незалежності від третіх осіб при розгортанні та налаштуванні власного кластера Hadoop.

Публікації. Результати дослідження апробовано на VII міжнародній студентській науково-технічній конференції «Природничі та гуманітарні науки. Актуальні питання», V міжнародній науково-практичній конференції учених та студентів «Цифрова економіка як фактор інновацій та сталого розвитку суспільства», у вигляді тез конференцій:

1. Методи та засоби розгортання артефактів екосистеми Hadoop.
2. Розгортання власних артефактів екосистеми Hadoop.

Структура роботи. Робота складається з пояснювальної записки та графічної частини. Пояснювальна записка складається із вступу, 4 розділів, висновків, списку використаних джерел та 2 додатків. Обсяг роботи: пояснювальна записка – 71 арк. формату А4, графічна частина – 8 аркушів формату А1.

РОЗДІЛ 1
ОГЛЯД ЕКОСИСТЕМИ ОПРАЦЮВАННЯ ВЕЛИКИХ ДАНИХ HADOOP ТА АНАЛІЗ МЕТОДІВ РОЗГОРТАННЯ ЇЇ АРТЕФАКТІВ

1.1. Огляд екосистеми опрацювання великих даних Hadoop

Екосистема Hadoop – це сукупність інструментів, платформ, бібліотек і сервісів, що забезпечують зберігання, обробку та аналіз великих обсягів даних. Hadoop є основою для роботи з великими даними і складається з кількох основних компонентів, а також додаткових інструментів, які спрощують роботу з різноманітними завданнями.

Екосистема Hadoop допомагає зберігати та обробляти масиви інформації, готувати її для вивантаження в інші сервіси, збирати статистику. Найкраще Hadoop підходить для роботи з неструктурованими даними – невпорядкованою інформацією без певної структури, яку складно класифікувати та розбити на групи. Наприклад, із файлами документів, повідомленнями, аудіо- та відеозаписами, зображеннями. Система може шукати необхідні відомості у великому архіві, отримувати з масиву «порожньої» інформації – інформацію корисну для підприємства. Наприклад, підрахувати унікальних користувачів у трафіку з мільйонів IP-адрес [9].

З розвитком технологій виникає потреба в удосконаленні методів та засобів розгортання артефактів екосистеми Hadoop, що дозволяє покращити продуктивність, надійність та гнучкість систем. Дослідження у цій сфері спрямовані на розробку нових підходів до автоматизації процесів розгортання, налаштування кластерів, моніторингу та управління ресурсами.

До основних та одних із найбільш популярних компонентів екосистеми Hadoop відносяться:

- Hadoop;
- ZooKeeper;
- Hive;
- Spark;

– Kafka.

Apache Hadoop – це розподілена обчислювальна платформа з відкритим кодом, яка була створена для обробки великих обсягів даних (Big Data). Вона забезпечує масштабованість, надійність і стійкість до відмов. Логотип Apache Hadoop зображено на рис. 1.1.

Рис. 1.1. Офіційний логотип Apache Hadoop

В основі Hadoop лежить модель обчислень MapReduce та розподілена файлова система HDFS. Він призначений для масштабування від одного сервера до тисяч машин, кожен з яких пропонує локальні обчислення та зберігання. Хоч платформу Hadoop і можна розгорнути локально на одній машині, проте основним її завданням є виконання поставлених задач опрацювання великих даних кластером. [10]

Кластер – це декілька незалежних обчислювальних машин, які використовуються спільно та працюють як одна система для підвищення продуктивності, забезпечення надійності та відмовостійкості, спрощення адміністрування.

Обчислювальний кластер необхідний та переважно використовується для збільшення швидкості обрахунків, використовуючи при цьому принципи паралелізації.

Основними компонентами Apache Hadoop є HDFS, YARN та MapReduce.

HDFS – це розподілена файлова система, яка зберігає дані на багатьох вузлах. Вона оптимізована для запису та читання великих файлів. Дана файлова система має високу відмовостійкість і забезпечує високу пропускну здатність доступу до даних додатків. Підходить для додатків, які мають великі набори даних.

Основні проблеми та цілі, які вирішує HDFS:

– підтримка паралельної обробки;

– ефективне управління метаданими;

– забезпечення надійного та ефективного розподіленого збереження великих обсягів даних;

– виявлення апаратних несправностей і швидке автоматичне відновлення після них;

– висока доступність та зменшення ризиків втрати даних;

– масштабування обсягів даних;

– захист даних та керування доступом.

Основними елементами HDFS є:

– NameNode. Обчислювальна машина, яка відповідає за метадані та управління доступом до файлів. Виконує операції простору імен файлової системи, такі як відкриття, закриття та перейменування файлів і каталогів. Він також визначає відображення блоків на DataNodes;

– DataNode. Відповідає за обслуговування запитів на читання та запис від клієнтів файлової системи. Вузли даних також виконують створення, видалення та реплікацію блоків за вказівкою NameNode.

– Secondary NameNode. Додаткова обчислювальна машина, яка синхронізується із основною NameNode, створює резервні копії метаданих для відновлення у разі збоїв.

Архітектуру HDFS графічно зображено на рис. 1.2.

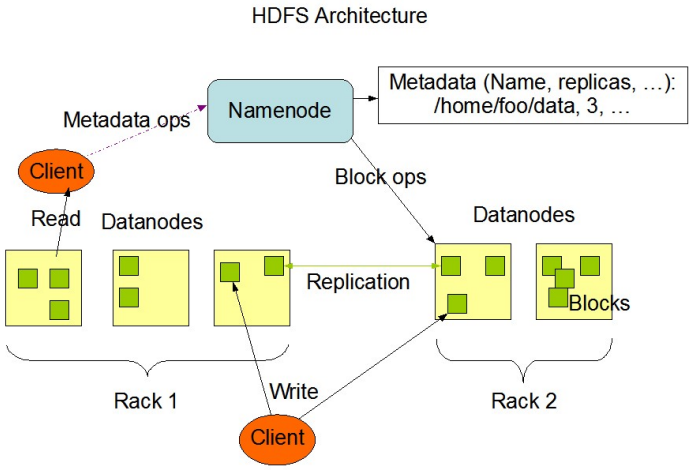

Рис. 1.2. Архітектура HDFS

HDFS підтримує традиційну ієрархічну організацію файлів. Ієрархія простору імен файлової системи подібна до більшості інших існуючих файлових систем. Можлива реалізація квоти користувачів та налаштування прав доступу, проте не підтримуються жорсткі або м'які посилання. Однак, архітектура не виключає і їх реалізації [11].

YARN – це система управління ресурсами в Hadoop, яка дозволяє запускати різні обчислювальні завдання на розподілених ресурсах. Вона розподіляє ресурси між програмами та керує плануванням завдань [10].

MapReduce – це модель обробки даних, яка складається з двох основних етапів: Map та Reduce. Map відповідає за обробку вхідних даних і їх розбиття на пари ключ-значення. Reduce відповідає за агрегування результатів із етапу Map для створення фінального результату [12].

Як правило, обчислювальні вузли та вузли зберігання даних є однаковими, тобто фреймворк MapReduce та розподілена файлова система Hadoop працюють на одному наборі вузлів. Ця конфігурація дозволяє фреймворку ефективно планувати завдання на вузлах, де вже присутні дані, що призводить до дуже високої сукупної пропускної здатності по всьому кластеру.

Незважаючи на те, що фреймворк Hadoop реалізований на Java, додатки MapReduce не обов'язково повинні бути написані на Java.

Окремо варто відзначити і інші додаткові компоненти та сервіси, що використовуються для розширення функціональності Hadoop. До найбільш популярних відносяться:

– ZooKeeper – розподілена система координації, створена для управління метаданими та координації в розподілених системах. Вона забезпечує надійний механізм синхронізації, управління конфігураціями, координування процесів та групових служб для додатків, які потребують високої доступності та узгодженості.

До основних функцій ZooKeeper належать координація вузлів, управління конфігурацією, виявлення сервісів, реалізація розподілених блокувань та синхронізації потоків, а також відстеження членства сервісів в групах вузлів [13].

– Hive – система для роботи з великими даними на Hadoop за допомогою SQL-подібної мови запитів (HiveQL). Вона надає інтерфейс для аналітики, що дозволяє користувачам працювати з даними за допомогою запитів, замість написання коду MapReduce [14].

– Spark – надшвидкісний, високопродуктивний кластерний обчислювальний фреймворк для обробки даних, що зберігаються в кластері Hadoop. Являє собою розподілену обчислювальну систему, яка підтримує обробку даних у пам'яті, що значно прискорює виконання аналітичних завдань у порівнянні з MapReduce.

Даний фреймворк має також власну інтерактивну оболонку SparkShell для розробки та експериментів з Spark. Забезпечує можливість використання Spark API через командний рядок, використовуючи команди на таких мовах як Scala, Java, Python та R.

Spark також додатково підтримує модулі для роботи з потоковими даними, машинним навчанням (MLlib), графовими обчисленнями (GraphX) та обробкою структурованих даних (Spark SQL) [15].

Spark є у сотні разів продуктивнішим порівняно з Hadoop. Він включає в себе оптимізатор на основі витрат, генерацію коду та стовпчасте зберігання, щоб зробити запити гнучкими разом із обчисленням тисяч вузлів за допомогою механізму Spark, який забезпечує повну відмовостійкість – mid-query. Наприклад, продуктивність Spark SQL порівняно з Hadoop зображено на рис. 1.3.

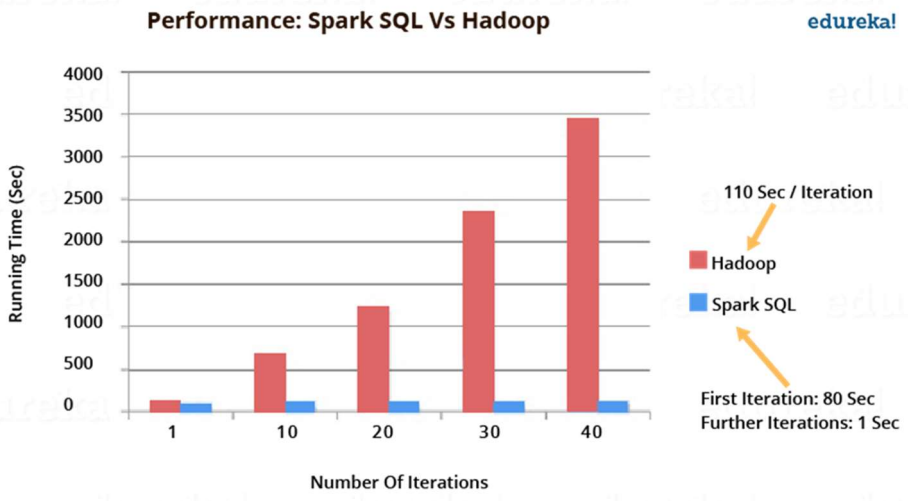

Рис. 1.3. Продуктивність Spark SQL порівняно з Hadoop

– Kafka – розподілене сховище подій і платформа для їх багатопотокового оброблення. Метою проєкту є створення уніфікованої високопродуктивної платформи з низькою затримкою для обробки потоків даних у реальному часі.

В основі концепції Kafka лежить абстракція «набір повідомлень», яка передбачає групування повідомлення разом, щоб зменшити накладні витрати на транспортування по мережі.

Kafka зберігає повідомлення у форматі ключ-значення, які приходять від необмеженої кількості клієнтів, що називаються «продюсерами» (producers). Дані можуть бути розділені на окремі «розділи» (partitions) залежно від «теми» (topic). Всередині розділу, повідомлення суворо впорядковуються за відступом (offsets) — позицією повідомлення в розділі, індексуються і зберігаються разом з позначкою часу. Інші клієнти, які називаються «споживачами» (consumers), можуть читати повідомлення з розділів [16].

1.2. Дослідження та аналіз проблематики розгортання екосистеми Hadoop

Зі зростанням обсягу та складності даних, які створює сучасне суспільство, і які охоплюються терміном "Big Data", традиційні методи управління та обробки даних часто стають недостатніми. Як результат, виникли нові підходи до вирішення цих викликів. Незважаючи на те, що всі ці підходи об’єднуються під єдиний термін, вони охоплюють різноманітні, часто дуже відмінні сценарії використання [20]. Однак усі вони мають спільну рису — значний акцент на масштабованості та портативності впроваджених рішень [21–23]. Крім того, у багатьох сценаріях важливо забезпечити високий рівень гнучкості, що впливає на дизайн і розробку таких рішень.

Для реалізації проєктів у галузі великих даних обов’язковим є використання високотехнологічних та масштабованих технологій, здатних впоратися з викликами, які виникають через характеристики великих даних.

Однак, попри підвищений інтерес до виконання різних задач опрацювання великих даних, на сьогодні спостерігається брак загальних знань. Це значною мірою обумовлено постійно зростаючим ринком технологій і інструментів, що ускладнює

підтримку актуальних знань щодо їхнього розвитку. Це було детально описано та досліджено у [24], де було представлено комплексну онтологію технологій великих даних, яка охоплює існуючі властивості, необхідні знання та зв’язки з іншими технологіями. Використання концепції мапування технологій для всієї екосистеми великих даних показує, що вибір, поєднання та реалізація технологій є складним процесом, який вимагає виконання численних етапів.

Дії, пов’язані з плануванням, проєктуванням і розробкою систем для роботи з великими даними, часто об’єднують під терміном "інженерія великих даних" [25]. На основі структурованого аналізу та планування цільового проєкту визначаються вимоги, специфікації, рішення щодо дизайну системи, а також проводяться тестування та розгортання. Зрештою, це приводить до створення ефективної комбінації технологій для роботи з великими даними, що формують архітектуру великих даних. Точніше, її можна визначити як "архітектуру, яка забезпечує структуру для роботи з усіма формами даних. Це логічна структура основних елементів, які використовуються для зберігання, доступу та управління великими даними" [26].

У багатьох випадках ці архітектури можуть бути не лише дуже складними у створенні, але й у розгортанні та управлінні. Тому, враховуючи динамічний характер цієї галузі, який відображається у постійному виникненні нових сфер застосування, технологій чи архітектур, доцільно автоматизувати принаймні останні етапи, а саме конфігурування та розгортання, які є досить затратними за часом.

Оскільки програми для роботи з великими даними зазвичай добре підходять для розгортання на фізичних чи віртуальних серверах, а також в хмарних середовищах [27] і потребують при цьому високої масштабованості, для їх реалізації часто використовуються інструменти автоматизації. Окрім цього, для забезпечення високої гнучкості важливо правильно визначати початкову конфігурацію сервісів екосистеми або і зовсім використовувати власні сформовані та попередньо налаштовані артефакти, що забезпечать виконання специфічних задач опрацювання великих даних.

Тому для визначення найкращого методу розгортання, що забезпечуватиме високу масштабованість, виключення великої кількості помилок при розгортанні, простоту конфігурування та гнучкість сервісів, проведено огляд та аналіз існуючих методів розгортання екосистеми Hadoop.

1.3. Огляд та аналіз існуючих методів та засобів розгортання екосистеми опрацювання великих даних Hadoop

До методів та засобів розгортання артефактів екосистеми Hadoop можна віднести наступні:

– традиційний метод розгортання, або розгортання вручну;

– використання автоматизаційних інструментів;

– використання власних збудованих артефактів.

1.3.1. Традиційний метод розгортання кластера Hadoop.

Традиційно розгортання Hadoop включає ручне налаштування кластерів, встановлення необхідного програмного забезпечення та налаштування конфігураційних файлів усіх сервісів. Цей підхід вимагає значних затрат часу та ресурсів, оскільки кожен вузол кластера повинен бути налаштований окремо.

Основними етапами традиційного розгортання є:

– встановлення необхідної операційної системи та налаштування базової конфігурації на кожному з вузлів кластера;

– інсталяція компонентів Hadoop та необхідних пакетів для їх роботи;

– налаштування конфігураційних файлів. Один з найбільш трудомістких етапів, адже налаштування необхідно виконувати для кожного компонента Hadoop;

– запуск та тестування кластера;

– моніторинг та управління кластером.

Після успішного розгортання та запуску необхідно налаштувати інструменти для моніторингу кластера, системи сповіщень та логування.

До основних переваг традиційного методу можна віднести:

– повний контроль над усіма аспектами конфігурації та управління кластером;

– гнучкість та можливість адаптації конфігурацій під специфічні вимоги проєкту.

До основних недоліків даного методу відносяться:

– великі часові витрати;

– складність масштабування. Додавання нових вузлів до кластера вимагає великих зусиль та може призвести до помилок в роботі;

– великий ризик помилок, що може безпосередньо вплинути на стабільність та продуктивність у роботі.

1.3.2. Використання інструментів автоматизації. Для спрощення процесу автоматизації розгортання сервісів Hadoop розроблено різноманітні автоматизаційні інструменти, які дозволяють зменшити витрати часу та підвищити точність налаштувань. Ці інструменти дозволяють автоматизувати більшість рутинних завдань, знижуючи ймовірність помилок і прискорюючи процес розгортання [9]. Серед найпопулярніших інструментів варто відзначити:

– Apache Ambari;

– Cloudera Manager;

– Hortonworks Data Platform.

1.3.3. Використання власних збудованих артефактів. Використання власних збудованих артефактів Hadoop дозволяє організаціям адаптувати Hadoop до специфічних вимог їхньої інфраструктури та робочих процесів. Хоча на ринку доступні готові рішення для розгортання екосистеми Hadoop, створення та використання власних артефактів надає декілька важливих переваг.

Основні переваги цього підходу включають:

– адаптивність: власні артефакти дозволяють враховувати специфічні вимоги проєкту та оптимізувати налаштування системи для досягнення максимального результату;

– контроль та безпека: організації мають повний контроль над процесом розробки та впровадження артефактів, що знижує ризики, пов'язані з використанням сторонніх рішень;

– гнучкість: власні артефакти забезпечують більшу гнучкість у налаштуванні та оптимізації системи, що дозволяє швидко реагувати на зміни в навантаженні та вимогах до системи.

Для формування власних артефактів екосистеми Hadoop розробниками Apache реалізований проєкт Bigtop.

Отже, у результаті огляду екосистеми Hadoop та її основних сервісів стверджується, що вона є ключовим інструментом для опрацювання великих даних завдяки своїй масштабованості, високій доступності, гнучкості та підтримці розподілених обчислень для виконання різних завдань у багатьох галузях.

У процесі дослідження та аналізу розглянуто три основні методи розгортання артефактів Hadoop, кожен з яких має свої переваги та недоліки.

Традиційний метод є доцільним для невеликих середовищ або навчальних цілей, але він неефективний для складних та масштабованих проєктів.

Метод розгортання з використанням інструментів автоматизації є оптимальним рішенням для середніх і великих проєктів, що потребують масштабованості та швидкого розгортання.

Розгортання власних артефактів є ефективним методом для унікальних проєктів, які вимагають високого рівня оптимізації, а також для забезпечення більшої незалежності від сторонніх виробників чи надавачів послуг.

Загальний порівняльний аналіз розглянутих методів розгортання екосистеми Hadoop наведено у таблиці 1.1.

Таблиця 1.1

Загальний порівняльний аналіз методів розгортання екосистеми Hadoop

Метод	Переваги	Недоліки
Традиційний метод	Гнучкість, повний контроль	Низька швидкість, високий ризик помилок

Продовж. табл. 1.1

Інструменти автоматизації	Швидке розгортання, мінімізація помилок	Потреба в початковому налаштуванні
Власні артефакти	Гнучкість, висока оптимізація під завдання	Трудомісткість, складність підтримки, забезпечення сумісності між сервісами

На основі аналізу переваг та недоліків кожного з методів можна зробити висновок, що комбінація методу використання інструментів автоматизації та методу розгортання власних артефактів є найефективнішою стратегією для проєктів, що працюють із великими даними використовуючи сервіси екосистеми опрацювання великих даних Hadoop.

РОЗДІЛ 2
ВИБІР МЕТОДІВ ТА ІНСТРУМЕНТІВ РОЗГОРТАННЯ АРТЕФАКТІВ ЕКОСИСТЕМИ HADOOP. ТЕОРЕТИЧНЕ ОБҐРУНТУВАННЯ.

2.1. Вибір методу розгортання екосистеми Hadoop

Відповідно до дослідження та аналізу наявних методів розгортання артефактів Hadoop, проведеного у попередньому розділі, пропонується комбінація методу використання інструментів автоматизації та методу розгортання власних артефактів. Комбінація цих методів виступає найефективнішою стратегією для проєктів, що працюють із великими даними використовуючи сервіси екосистеми опрацювання великих даних Hadoop і при цьому бажають отримати високу швидкість розгортання, забезпечення масштабування, мінімізацію помилок та достатню гнучкість для виконання специфічних задач. Також даний метод надасть достатню незалежність від зовнішніх виробників чи комерційних провайдерів, що може позитивно вплинути на витрати підприємств.

2.2. Вибір інструментів автоматизації для формування та розгортання артефактів екосистеми Hadoop

Комерційні провайдери, такі як Amazon і Google, які частково дозволяють повне або напівавтоматизоване розгортання популярних технологій великих даних у своїх хмарних середовищах, у багатьох випадках надають власні розроблені рішення для роботи з великими даними, які можна використовувати з відносно невеликими зусиллями. Однак такі рішення мають певну вартість і зобов'язання.

На сьогодні відомо лише декілька некомерційних рішень, які дозволяють автоматизоване розгортання популярних сервісів опрацювання великих даних:

- фреймворк DICE;
- Apache Bigtop;
- Apache Ambari;

– дистрибутив Cloudera;

– дистрибутив Hortonworks Data Platform.

Дистрибутиви Cloudera та Hortonworks Data Platform є багатофункціональними наборами інструментів, що надають широкий вибір популярних інструментів. У свою чергу, фреймворк DICE та Apache Bigtop є рішеннями для розгортання, які дозволяють налаштувати конкретні інструменти та сервіси під конкретні вимоги. Apache Ambari виступає окремо як інструмент для розгортання вже сформованих артефактів, конфігурування та повного управління кластером Hadoop.

Bigtop – це проєкт з відкритим вихідним кодом, який забезпечує комплексне рішення для створення, тестування та розгортання дистрибутивів екосистеми Hadoop. Bigtop надає інструменти для автоматизації створення артефактів та їхнього розгортання на різних платформах.

До основних можливостей Apache Bigtop відносяться:

– підтримка різних платформ: Bigtop дозволяє створювати пакети для різних операційних систем, включаючи Debian, Ubuntu, CentOS та інші. Це забезпечує сумісність з різними середовищами розгортання та знижує витрати на підтримку;

– автоматизація процесів: Bigtop автоматизує процеси створення та тестування артефактів, що дозволяє швидко розгортати оновлення або нові компоненти Hadoop. Це забезпечує стабільність та передбачуваність у роботі кластера;

– масштабованість: завдяки своїй архітектурі, Bigtop підтримує масштабованість розгортання, що дозволяє ефективно управляти великими кластерами з різними конфігураціями;

– інтеграція з CI/CD: Bigtop легко інтегрується з інструментами безперервної інтеграції та доставки, що дозволяє автоматизувати процеси розробки, тестування та впровадження нових артефактів у кластер Hadoop;

– розширюваність: проєкт підтримує створення власних шаблонів та конфігурацій для специфічних потреб, що робить його ідеальним для розгортання власних версій екосистеми Hadoop.

Apache Bigtop значно спрощує процес створення власних артефактів, надаючи засоби для їхнього автоматизованого тестування та розгортання. Це особливо

важливо для організацій, які прагнуть розробляти індивідуальні рішення на базі Hadoop, адаптовані до їхніх унікальних потреб.

Bigtop також дозволяє організаціям створювати та підтримувати власні дистрибутиви Hadoop, що включають лише ті компоненти, які потрібні для конкретних завдань, тим самим підвищуючи ефективність і зменшуючи загальну складність управління кластером [17]. На відміну від зазначеного Bigtop, фреймворк DICE виник у рамках проєкту ЄС, фінансованого за програмою Horizon 2020, метою якого було "Створення інструментів DevOps для роботи з великими даними, що нативно підтримують ці технології великих даних" [28]. У межах цього проєкту було створено плагін для інтегрованого середовища розробки Eclipse, який допомагає крок за кроком реалізовувати програми для роботи з великими обсягами даних. За допомогою інтеграції UML-діаграм та специфік технологій можна виконувати комплексні та деталізовані завдання з інженерії великих даних, включно з плануванням, дизайном, розробкою, тестуванням і розгортанням.

Для розгортання пов’язаних технологій у DICE використовується інструмент управління конфігурацією Chef, який виконує подібні функції до Ansible і Puppet. Завдяки додатковому використанню стандарту TOSCA (Topology and Orchestration Specification for Cloud Application) забезпечується хмарне розгортання прототипів, а також безперервна доставка та тестування.

Проте, проєкт DICE завершився у 2018 році. З того часу жодних значних оновлень або розширень не було. Тому тривале використання цього рішення не рекомендується, оскільки зміни в екосистемі великих даних більше не враховуватимуться.

Проєкт Apache Ambari – це проєкт з відкритим вихідним кодом, спрямований на спрощення управління Hadoop шляхом розробки програмного забезпечення для розгортання, управління та моніторингу кластерів Apache Hadoop [18].

Ambari надає інтуїтивно зрозумілий, простий у використанні вебінтерфейс управління Hadoop, підкріплений своїми RESTful API (це архітектурний стиль інтерфейсу прикладного програмування, який використовує HTTP-запити для доступу та використання даних).

Ambari дозволяє системним адміністраторам:

– розгорнути кластери Hadoop використовуючи покроковий майстер для встановлення та розподілу сервісів на великій кількості вузлів;

– налаштовувати конфігурацію сервісів Hadoop;

– використовувати централізоване управління для запуску, зупинки та переналаштування сервісів Hadoop у всьому кластері;

– проводити моніторинг кластера Hadoop.

Високорівневу архітектуру Apache Ambari зображено на рис. 2.1.

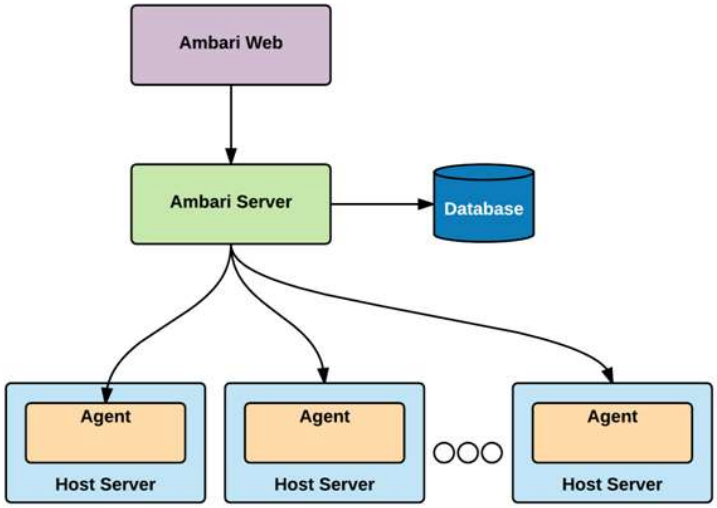

Рис. 2.1. Високорівнева архітектура Apache Ambari

Ambari надає інформаційну панель для моніторингу здоров'я та стану кластера Hadoop, використовуючи при цьому систему Ambari Metrics System для збору, агрегування та зберігання показників (метрик). Для відображення даних інформаційних панелей також використовується сервіс Grafana. Для системного оповіщення та їх налаштування Ambari використовує компонент Alert Framework.

Окремо варто зазначити, що через проблеми із припиненням підтримки проєкту Ambari від Apache у січні 2022 року, даний проєкт було поставлено на паузу, внаслідок чого з’явилось та залишились невирішеними багато недоліків. До основних проблем, які досі не вирішено, відносяться:

– використання застарілих JDK версії 8 та Python версії 2, які більше не підтримуються. Основна проблема, над якою активно працюють розробники;

– підтримка операційних систем, які також вже не підтримуються, а саме CentOS 7, Ubuntu 14;

– використання пакетів керування Ambari (Ambari Management Packs), які розгортають застарілі версії сервісів екосистеми Hadoop (наприклад сервіс Spark2).

На даний момент активно розробляється та готується реліз Apache Ambari версії 3.0.0, який вирішить усі наведені вище проблеми. Натомість, ці проблеми було вирішено компанією Clemlab, які взяли проєкт Ambari та підтримують його відгалуження на GitHub, починаючи з версії 2.7.5 [30]. Основні переваги проєкту Ambari від Clemlab:

– реалізовано використання Python версії 3;

– підтримка запуску та роботи на новіших версіях операційних систем, а саме RHEL 7/8/9 та Ubuntu 22.04;

– розроблено та активно підтримується власний стек ODP (OpenSource Data Platform), який дозволяє розгортати одні з останніх підтримуваних версій сервісів екосистеми Hadoop;

– внесено багато дрібних патчів для забезпечення сумісності сервісів.

Недоліком використання Clemlab Ambari все ще залишається відсутність гнучкості при налаштуванні безпосередньо сервісів екосистеми Hadoop, адже стек сервісів суворо визначений у конкретній версії стеку ODP, а самі артефакти програмних рішень завантажуються з репозиторіїв Clemlab.

Проєкти Cloudera та Hortonworks Data Platform [29] – комерційні інструменти для управління кластерами Hadoop, що надають широкий спектр функцій для автоматизації інсталяції, налаштування та моніторингу.

Серед переваг даних проєктів можна виділити, що вони не лише являються повноцінними рішеннями, що мають усі необхідні інструменти для управління кластером Hadoop, а також надають постійну комерційну підтримку. Це забезпечує надійність, своєчасне оновлення та вирішення проблем роботи кластера та сервісів.

В свою чергу основним недоліком обох платформ являється вартість даних комерційних рішень, тому вони можуть бути дорогими для малих та середніх організацій.

Відповідно до наведеної вище інформації про наявні на даний момент інструменти розгортання, конфігурування, повного управління та моніторингу кластера Hadoop, для реалізації методу розгортання з використанням автоматизаційних інструментів та власних сформованих артефактів пропонується використання комбінації інструментів Ambari (Apache Ambari починаючи з 2.8.0 або 3.0.0, наведеного прикладу вище Clemlab Ambari або інший підтримуваний проєкт Ambari) та Apache Bigtop. Проєкт Ambari забезпечить зручне та просте розгортання кластера Hadoop, використовуючи вебінтерфейс, а проєкт Bigtop надасть інструментарій для формування власних артефактів сервісів екосистеми Hadoop, забезпечуючи можливість налаштування цих сервісів під власні потреби та більшої незалежності від зовнішніх розробників чи комерційних рішень.

2.3. Вибір додаткового інструменту автоматизації для виконання невеликих допоміжних програм при розгортанні

При розгортанні кластера Hadoop важливою є автоматизація рутинних завдань, таких як налаштування конфігурацій, розгортання компонентів, оновлення системи чи виконання інших допоміжних операцій. Для цього використовуються інструменти автоматизації, які дозволяють ефективно управляти конфігурацією вузлів і підтримувати узгодженість стану кластерів. Серед найпопулярніших таких інструментів — Ansible, Puppet та Chef.

Puppet — це потужний інструмент автоматизації, який використовується для управління конфігурацією та автоматизованого забезпечення ресурсів у великих інфраструктурах. Puppet використовує власну мову опису станів системи, що дозволяє описати бажаний стан конфігурації вузлів у вигляді декларативних маніфестів. Головною перевагою Puppet є його орієнтація на роботу з великими кластерами, де важливо підтримувати узгодженість конфігурацій. Інструмент працює

за допомогою клієнт-серверної архітектури: сервер Puppet Master координує вузли через Puppet Agent, встановлений на кожному клієнті. Це забезпечує високий рівень автоматизації та контроль стану. Однак, для виконання невеликих завдань або одноразових операцій Puppet може бути менш зручним через складність налаштування та необхідність у серверній інфраструктурі [31].

Chef — це інструмент автоматизації, який дозволяє описувати інфраструктуру як код за допомогою сценаріїв, написаних мовою Ruby. Chef використовує клієнт-серверну архітектуру, але також підтримує режим Chef Solo, що дозволяє виконувати завдання без сервера, що робить його придатним для невеликих проєктів. Основною одиницею роботи в Chef є «рецепт» (recipe) — сценарій, який визначає бажаний стан системи. Рецепти об'єднуються у «книги рецептів» (cookbooks), які можна повторно використовувати для різних завдань. Chef забезпечує гнучкість у налаштуванні та добре інтегрується з різними платформами та хмарними середовищами. Проте, його поріг входження є вищим через необхідність знань Ruby і складності створення складних конфігурацій. Chef добре підходить для автоматизації процесів розгортання та управління конфігурацією, але для невеликих допоміжних завдань може бути надмірно складним [32].

Ansible — це простий механізм автоматизації, який використовується для автоматизації забезпечення хмарних ресурсів, управління конфігурацією, розгортання додатків, оркестрації між сервісами та інших завдань. Використовуючи Ansible, можна підключатися до вузлів (серверів, контейнерів або віртуальних машин) і створювати невеликі програми, які називаються "модулями Ansible". Ці модулі є моделями ресурсів бажаного стану, у якому повинна перебувати система. Ansible виконує ці модулі, а потім видаляє їх після завершення завдання. Порівняно з іншими подібними інструментами, такими як Puppet або Chef, Ansible є ефективним і компактним, оскільки не потребує активного сервера, демона або бази даних для виконання модулів чи підтримки станів [33].

Кожен із них має свої особливості, переваги та недоліки, які слід враховувати залежно від специфіки завдань, масштабу інфраструктури та рівня технічної підготовки команди.

Порівняння основних характеристик інструментів Ansible, Puppet та Chef наведено у таблиці 2.1.

Таблиця 2.1

Порівняння основних характеристик інструментів Ansible, Puppet та Chef

Характеристика	Ansible	Chef	Puppet
Мова конфігурації	YAML (Ansible Playbooks)	Ruby (Chef Recipes)	Puppet DSL (Puppet Manifests)
Інфраструктура	Без агентів (можливість використання агентів)	Потрібні агенти на всіх вузлах	Потрібні агенти на всіх вузлах
Простота використання	Висока, простий синтаксис	Складніший, потребує знань Ruby	Середня, потребує знань DSL Puppet
Масштабованість	Легка масштабованість з інвентарними файлами	Добре масштабуються за допомогою Chef Server	Добре масштабуються через Puppet Master
Продуктивність	Швидке виконання, мінімальні вимоги до ресурсів	Залежить від конфігурації і розміру інфраструктури	Залежить від розміру кластера та конфігурації
Використання в компаніях	Широко використовується для швидкого розгортання	Використовується в великих компаніях з комплексними інфраструктура-ми	Популярний в середніх і великих організаціях
Підтримка хмарних платформ	Легко інтегрується з хмарними платформами через модулі	Інтеграція через бібліотеки Chef для хмар	Підтримує інтеграцію з хмарними платформами

2.3. Поєднання інструментів Ambari та Apache Bigtop

Поєднання інструментів автоматизації Ambari та Apache Bigtop дозволяє досягти високого рівня автоматизації та узгодженості при розгортанні й обслуговуванні екосистеми Hadoop. Кожен із цих інструментів виконує специфічні завдання, а їх взаємодія забезпечує ефективність процесів на всіх етапах життєвого циклу кластеру:

– Формування пакетів. Apache Bigtop використовується для створення узгоджених дистрибутивів Hadoop-компонентів у вигляді пакетів, готових до встановлення. Це дає змогу забезпечити сумісність із різними операційними системами, такими як CentOS, Ubuntu чи Debian, та інтеграцію з контейнерними середовищами, наприклад Docker. Дані пакети можна завантажити на відповідні репозиторії, після чого неоднократно використовувати їх для інсталяції необхідних сервісів при розгортанні кластера Hadoop.

– Розгортання екосистеми:

а) автоматизована установка. Побудовані за допомогою Bigtop пакети інтегруються у процеси розгортання, що здійснюються через Apache Ambari. Ambari дозволяє автоматично встановлювати необхідні компоненти Hadoop на всіх вузлах кластеру, оптимізуючи час і зусилля адміністратора;

б) масштабування. У разі додавання нових вузлів у кластер або розширення існуючої інфраструктури, Bigtop забезпечує створення додаткових пакетів, а Ambari автоматично інтегрує нові вузли в кластер із правильними налаштуваннями.

– Управління конфігураціями:

а) централізоване налаштування. Ambari надає єдиний інтерфейс для керування конфігураціями всіх компонентів Hadoop. Завдяки цьому забезпечується узгодженість параметрів на всіх вузлах кластера, незалежно від складності його архітектури;

б) швидке внесення змін. Якщо необхідно змінити параметри конфігурації для окремого компонента або всього кластера, Ambari автоматично розповсюджує ці

зміни по всіх вузлах. Це значно спрощує адміністрування великих інфраструктур, розгорнутих за допомогою Bigtop;

в) версіонування. Інтеграція з Bigtop дозволяє легко застосовувати оновлення конфігурацій або повертатися до попередніх версій у разі виникнення проблем із сумісністю.

– Моніторинг і підтримка:

а) моніторинг стану кластеру. Ambari забезпечує візуалізацію стану вузлів, використання ресурсів та продуктивності компонентів Hadoop. Це дозволяє вчасно виявляти та усувати потенційні проблеми;

б) оновлення та патчі. У разі виходу нових версій компонентів Hadoop Bigtop забезпечує створення оновлених пакетів. Ambari, у свою чергу, дозволяє централізовано розгортати ці оновлення на всіх вузлах кластеру з мінімальними простоями;

в) тестування стабільності. Завдяки інтеграції з Bigtop автоматично перевіряється стабільність кластера після оновлень або внесення змін до конфігурації. Це дозволяє підтримувати безперебійну роботу системи.

– Оптимізація процесів адміністрування:

а) автоматизація рутинних завдань. Використання Ambari спрощує виконання таких завдань, як налаштування параметрів HDFS або запуск сервісів Spark, що дозволяє адміністраторам зосередитися на більш важливих аспектах;

б) інтеграція з CI/CD. Bigtop можна використовувати для автоматизації побудови й тестування пакетів у конвеєрах безперервної інтеграції, а Ambari — для автоматичного розгортання змін у кластері.

– Гнучкість і адаптивність:

а) підтримка різних середовищ. Завдяки можливостям Bigtop створювати пакети для різних операційних систем і платформ, а також універсальності Ambari у налаштуванні конфігурацій, можна адаптувати кластер до специфічних потреб організації;

б) швидке розгортання тестових середовищ. Для розробки та тестування можна швидко створювати тестові середовища з використанням Bigtop і Ambari, що пришвидшує цикл розробки.

2.4. Побудова власних артефактів за допомогою Apache Bigtop

Apache Bigtop надає гнучкі можливості для створення та тестування власних артефактів екосистеми Hadoop. Даний інструмент виступає на даний момент єдиним та безальтернативним рішенням для адміністраторів і розробників, які прагнуть адаптувати екосистему Hadoop до специфічних потреб своїх організацій. Проєкт дозволяє створювати власні дистрибутиви, що забезпечують індивідуальні налаштування, які відповідають технічним і бізнесовим вимогам. Це дозволяє значно підвищити ефективність інфраструктури, оскільки надається можливість налаштовувати пакети з урахуванням конкретних умов експлуатації.

Крім того, Bigtop надає можливості для тестування нових функціональностей, що є критично важливим при впровадженні інновацій або внесенні змін у кластер. Інструмент включає засоби автоматизованого тестування, що дозволяють оцінити стабільність і сумісність нових версій компонентів у реальному середовищі, що знижує ймовірність виникнення проблем після впровадження змін у продуктивне середовище. Це дозволяє не лише зберегти високий рівень стабільності системи, але й підвищити надійність виконуваних операцій.

Ще однією важливою перевагою проєкту є можливість інтеграції нестандартних компонентів. Адміністратори та розробники можуть додавати до екосистеми власні модулі чи змінювати існуючі, тестуючи їхню роботу перед розгортанням у prod-середовищі. Це відкриває нові можливості для розширення функціональності Hadoop, особливо при використанні інноваційних або експериментальних рішень.

Особливо корисною є можливість створювати середовища для розробки. Завдяки автоматизації процесу тестування та швидкому формуванню пакетів, Bigtop сприяє створенню ізольованих середовищ для розробки і тестування нових рішень.

Це значно прискорює цикл розробки програмного забезпечення, одночасно зменшуючи ризики, пов'язані з потенційним впливом на основний кластер.

Не менш важливим аспектом є оптимізація підтримки. Використання Bigtop для формування власних артефактів дозволяє централізовано документувати всі пакети та налаштування, що значно полегшує процес підтримки інфраструктури. Всі компоненти контролюються в рамках єдиного процесу, що підвищує прозорість і зручність управління.

Система також забезпечує розширені можливості для відлагодження, що включають детальне трасування та аналіз виконуваних процесів при формуванні пакетів та тестуванні. Це дозволяє ефективно відслідковувати виконання завдань, виявляти помилки на різних етапах їх виконання і детально аналізувати причини несправностей. Завдяки гнучким інструментам для відлагодження, розробники та адміністратори можуть не лише ідентифікувати та усувати проблеми в реальному часі, але й здійснювати глибоке тестування компонентів у різних умовах. Інтеграція Apache Bigtop з Ambari дозволяє налаштувати детальну реєстрацію та відслідковування подій як при формуванні пакетів, так і тестуванні і розгортанні самого кластера Hadoop. Це підвищує ефективність налагодження і оптимізації продуктивності всього кластера.

Підсумовуючи викладені аспекти у даному розділі, стає очевидним, що знайдене рішення перевершує використання окремих поодиноких проєктів для розгортання кластера Hadoop в кількох аспектах. У такому швидко змінюваному середовищі, як великі дані, запропоновано довготривале та адаптивне рішення, яке дозволяє системним інженерам швидко розгортати не лише найновіші сервіси опрацювання великих даних Hadoop, а й формування та використання власних артефактів для їхніх сценаріїв використання. Проте, переваги цього рішення досягаються за рахунок рівня деталізації конфігурації, яку потрібно інвестувати до початкового розгортання.

У даному розділі запропоновано новий, достатньо гнучкий та автоматизований метод, який полягає у використанні певних автоматизаційних інструментів для

формування власних артефактів та їх розгортання, що дозволяє дослідникам та практикам розгортати свої архітектури великих даних у різних середовищах.

РОЗДІЛ 3
ПРАКТИЧНЕ ДОСЛІДЖЕННЯ ТА РЕАЛІЗАЦІЯ РОЗГОРТАННЯ АРТЕФАКТІВ ЕКОСИСТЕМИ HADOOP

3.1. Вибір версій сервісів та формування власних артефактів екосистеми Hadoop

3.1.1. Опис середовища виконання та вибір версії проєкту Ambari.

Для проведення досліджень використано кластер, що складається з віртуальних машин, запущених використовуючи сервіс OpenStack. Використовувана операційна система на віртуальних машинах – Ubuntu 22.04.

Кластер складається з наступних віртуальних машин:

– edge-вузол: isv-s-hdp-edge-01. На ньому встановлюється Ambari Server для розгортання та керування усім кластером Hadoop;

– майстер-вузол: isv-s-hdp-mst-01. Виступатиме як namenode в кластері HDFS. Окрім цього на ньому встановлено сервіси Tez, Hive та Spark;

– керуючий вузол: isv-s-hdp-mng-01. На ньому встановлено сервіс Ambari Metrics Collector для збирання, зберігання та агрегування метрик зі всіх вузлів кластера;

– робочий вузол: isv-s-hdp-w-01. Виступає як datanode та виконавча одиниця для основної обробки даних.

Для розгортання кластера Hadoop обрано Ambari, що представлений компанією Clemlab. Він являється відгалуженням від оригінального репозиторію Apache Ambari. Обрана релізна версія – 2.7.8. До переваг даної версії можна віднести:

– постійна підтримка проєкту Ambari в той період, коли Apache Ambari мали проблеми з цим;

– можливість розгортання на операційних системах CentOS 8/9, а також Ubuntu 22.04, які досі підтримуються, в той час як Apache Ambari підтримував розгортання лише на операційній системі CentOS 7;

– підтримка власного стеку для розгортання екосистеми Hadoop, який пропонує актуальні версії популярних сервісів;

– виправлення та покращення програмних кодів проєкту.

3.1.2. Вибір версії проєкту Apache Bigtop. Дослідження його структури та опис основних команд. Для формування артефактів сервісів екосистеми Hadoop використано проєкт Bigtop версії 3.2.1. Дана версія підтримує актуальні версії сервісів, а також у ній виправлені важливі помилки, допущені у версії 3.2.0, а саме невідповідність python залежностей в Ubuntu 22.04, яка викликає провал smoke-тестів Ambari, gpdb, Hive, Ycsb та Zeppelin, як це зазначено самими розробниками на офіційному вебсайті.

Перелік сервісів екосистеми Hadoop та їх версії, які підтримує для розгортання Bigtop v3.2.1 наведено у таблиці 3.1.

Таблиця 3.1

Сервіси екосистеми Hadoop та їх версії, які підтримує Bigtop v3.2.1

Сервіс	Версія
Bigtop-ambari-mpack	v2.7.5
Flink	v1.15.3
Hadoop	v3.3.5
Hbase	v2.4.13
Hive	v3.1.3
Kafka	v2.8.1
Phoenix	v5.1.2
Solr	v8.11.2
Spark	v3.2.3
Tez	v0.10.1
Zeppelin	v0.10.1
Zookeeper	v3.5.9

При розгортанні екосистеми можна вибрати який сервіс необхідно інсталювати, а також можна додавати власні сервіси та формувати власні артефакти, додаючи їх у стек. Концепції «стек сервісів Hadoop» та «Ambari MPack» наведено розглянуто у пункті 3.1.3.

3.1.3. Використання стеків сервісів Hadoop та Ambari MPack. Для розгортання артефактів Ambari використовує попередньо визначені стеки сервісів. Стек Ambari – це сукупність компонентів та технологій, які використовуються самим програмним рішенням Ambari для спрощення та автоматизації розгортання, налаштування, управління та моніторингу окремих сервісів Hadoop та у загальному кластері Hadoop [4]. Кожен компонент стеку виконує певну роль при розгортанні та взаємодіє з іншими компонентами для досягнення спільної мети. Стеки мають власні визначені версії сервісів Hadoop, а також вони забезпечують власні інструменти для керування версіями сервісів. Кожна версія стеку пропонує попередньо визначений перелік сервісів певних версій для забезпечення сумісності.

Загальна структура кожного стеку Ambari наступна:

– blueprints – шаблони, що визначають конфігурацію кластеру, включаючи сервіси, хости та їхні ролі.;

– properties – налаштування стеку, які містять основні параметри конфігурації для сервісів та компонентів;

– repos – репозиторії пакетів для встановлення компонентів стеку. Визначають за якою URL буде звертатись Ambari для завантаження пакетів в залежності від операційної системи. Зазвичай для використання власних артефактів попередню URL необхідно змінювати на URL власного репозиторію;

– services – конфігурація окремих сервісів, яка включає їх опис, залежності та специфічні налаштування;

– metainfo – метаінформація про стек, така як його версія, підтримувані платформи та сумісність.;

– role_command_order.json – визначає порядок виконання команд для ролей під час інсталяції, запуску та оновлення.

Ambari MPack (Management Pack) – це стек сервісів, який інтегрується «на льоту» в Ambari, не вносячи змін при цьому безпосередньо в проєкт. Використовується для розгортання і управління компонентами екосистеми Hadoop.

Ambari MPack від Bigtop забезпечує зручний спосіб додавання нових сервісів до вже наявного кластеру, який керується Ambari. Це дозволяє користувачам

використовувати розширені можливості Bigtop для управління і тестування компонентів Hadoop через інтерфейс Ambari, що значно полегшує адміністрування кластера.

Проте, для постійного використання рекомендується інтегрувати визначений у Ambari MPack стек безпосередньо в проєкт Ambari (за шляхом ambari-server/src/main/resources/stacks). Таким чином не буде необхідності інтегрувати Ambari MPack вже після формування та встановлення пакету «Ambari Server».

3.1.4. Формування власних артефактів Ambari та їх завантаження у віддалені репозиторії. Щоб сформувати усі необхідні артефакти Ambari у корені склонованого GitHub проєкту виконано команду «mvn -B install package jdeb:jdeb "-DskipTests" "-Drat.ignoreErrors=true" -Dpython.ver="python >= 2.6" -Djdk.version="1.8" -Dviews -Pdeb "-Dodp.release.number=134"». Ідентичну команду можна виконати для формування rpm-пакетів, замінивши лише параметр «jdeb:jdeb» на «rpm:rpm».

Вимоги для формування deb-пакетів Clemlab Ambari включають наявність наступних встановлених пакетів:

– Maven;

– OpenJDK-dev 8;

– Python2-dev;

– GCC;

– Git.

В результаті отримано наступні сформовані deb-пакети:

– ambari-server – основний компонент, відповідає за встановлення вебсервера Ambari;

– ambari-agent – легковаговий агент, який встановлюється на кожному вузлі кластера. Призначений для отримання команд від вебсервера та їх виконання на вузлах;

– ambari-metrics – компонент для збору, зберігання та візуалізації метрик продуктивності кластера;

– ambari-infra-solr – компонент, який надає інфраструктуру для зберігання та пошуку журналів роботи кластера. Він заснований на Apache Solr і інтегрується з інструментами аналізу журналів, наприклад Log Search.

Збудовані deb-пакети, які можна використовувати для встановлення таких описаних вище компонентів, необхідно завантажити на окремий репозиторій. Наприклад, для цього можна використати apt-репозиторій Nexus 3. Таким чином забезпечується зберігання власних артефактів, які згодом легко завантажити та розгорнути, використовуючи Ambari.

Для дослідження створено та попередньо підписано новий apt-рпепозиторій для зберігання окремо сформованих артефактів проєкту Ambari, а також артефактів стеку Bigtop. Ці репозиторії названо “ambari-jammy” та “bigtop-3.2.1” відповідно.

Вміст apt-репозиторію “ambari-jammy” на сервері Nexus 3 зображено на рис. 3.1.

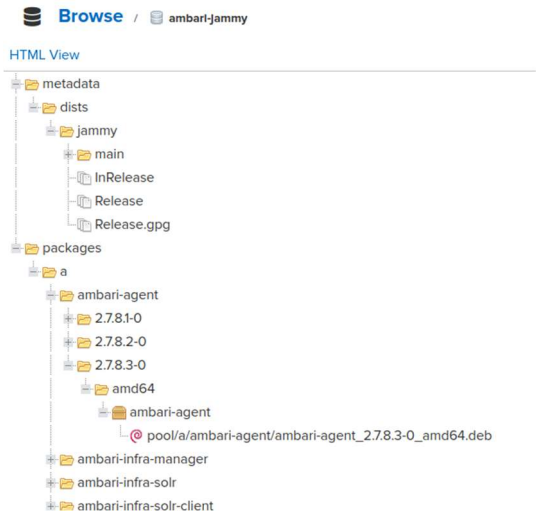

Рис. 3.1. Вміст apt-репозиторію “ambari-jammy” на сервері Nexus 3

3.1.5. Ознайомлення із структурою проєкту Apache Bigtop та формування власних артефактів сервісів стеку Bigtop. Основні компоненти структури Bigtop репозиторію наступні:

– bigtop-ci — вміщує у собі Jenkins CI реалізацію;

– bigtop-deploy — вміщує реалізацію для розгортання артефактів використовуючи juju або puppet;

– bigtop-packages/src — вміщує пакети артефактів;

– bigtop-test-framework — вміщує фреймворк itest для тестів;

– bigtop-tests — вміщує тести для артефактів;

– docker — вміщує підготовлені docker образи;

– gradle/wrapper — обгортка для Gradle;

– provisioner — вміщує реалізацію для розгортання та тестування збірок в контейнерах докер локально;

– gradlew — скрипт, з допомогою якого можна виконувати Gradle завдання, створенні розробниками Bigtop;

– bigtop.bom — у даному файлі вказано необхідну інформацію про репозиторії та версії артефактів, звідки bigtop завантажує вихідний код;

– pom.xml — у даному файлі описано проєкт та усі необхідні залежності для Maven.

Для виконання більшості основних завдань при роботі з проєктом Apache Bigtop використовується інструмент Gradle. Перелік та опис основних використовуваних команд:

– allclean — видаляє директорії, які було створено під час та після формування артефактів;

– <package>-pkg-ind, де <package> — назва артефакту — виконує формування артефакту та зберігає результати у директорію «output». Приставка «ind» вказує Bigtop, що необхідно виконувати усю збірку у docker-контейнері;

– repo-ind — створює yum-/apt-репозиторій, в який включає сформовані артефакти з директорії. Дане завдання також виконується у docker-контейнері;

– docker-provisioner — створює кластер локально з docker-контейнерів та розгортає артефакти Hadoop на них. За потреби можна запустити тести використовуючи додаткові параметри;

– docker-provisioner-destroy — видаляє створений кластер та видаляє створений файл «.provision_id».

Для формування артефактів обраного стеку сервісів, а саме сервісів Hadoop, ZooKeeper, Hive та Spark необхідно виконати у корені проєкту Bigtop команду

«allclean hadoop-pkg-ind zookeeper-pkg-ind hive-pkg-ind spark-pkg-ind -Pconfig=config_ubuntu-22.04.yaml -POS=ubuntu-22.04 -Pprefix=3.2.1». Після запуску цієї команди Apache Bigtop спочатку створює докер контейнер, в якому виконуватиметься формування артефакту, клонує необхідний GitHub репозиторій сервісу і застосовує спеціальні git-патчі для внесення необхідних змін у локальний репозиторій, не змінюючи при цьому оригінальні origin-гілки на GitHub. Після цього він запускає необхідну команду для формування проєкту і пакує двійкові файли у інсталяційні пакети (deb або rpm). Після цього директорія із сформованими артефактами копіюється з докер контейнера у корінь проєкту і артефакти можна завантажувати на віддалений репозиторій.

Bigtop також надає можливість створити власний локальний yum-/apt-репозиторій, якщо його необхідно зберегти на локальній машині. При цьому вказування в полі URL репозиторія локальний шлях до стеку не є доцільним, адже таким чином є потреба розмішувати даний репозиторій локально за певним однаковим шляхом на усіх вузлах.

3.2. Розгортання екосистеми Hadoop з використанням власних артефактів

Для попередньої підготовки до розгортання екосистеми Hadoop виконано наступні кроки:

– встановлено пакет ambari-server на edge-вузол, налаштовано та запущено;

– встановлено пакет bigtop-ambari-mpack та додано Bigtop Ambari MPack до Ambari Server командою “sudo ambari-server install-mpack --mpack=/usr/lib/bigtop-ambari-mpack/bgtp-ambari-mpack-1.0.0.0-SNAPSHOT-bgtp-ambari-mpack.tar.gz”;

– встановлено пакети ambari-agent на всіх вузлах кластера, налаштовано та запущено.

Лістинг команд для встановлення, налаштування та запуску сервера Ambari зображено на рис. 3.2.

```
sudo apt install ambari-server -y;
sudo ambari-server setup --jdbc-db = postgres --jdbc-driver =
/usr/share/java/postgresql-42.7.3.jar;
sudo ambari-server setup-security --security-option = setup-https --api-
ssl = true --api-ssl-port = 8043 --import-key-path = ambari_ssl_tls.key --
import-cert-path = ambari_ssl_tls.crt --pem-password = password;
sudo ambari-server setup;
sudo ambari-server start
```

Рис. 3.2. Лістинг команд для встанволення, налаштування та запуску сервера Ambari

Лістинг команд для встановлення, налаштування та запуску агента Ambari зображено на рис. 3.3.

```
sudo apt install ambari-agent -y;
sudo ambari-agent reset isv-s-hdp-edge-01.example.com;
sudo ambari-agent start
```

Рис. 3.3. Лістинг команд для встановлення, налаштування та запуску агента Ambari

Після запуску та налаштування Ambari буде доступний за адресою isv-s-hdp-edge-01.example.com:8043.

Перейшовши на вебінтерфейс Ambari запущено покроковий інсталятор для розгортання сервсів Hadoop.

Розгортання складається з наступних кроків:

– вибір версії стеку для розгортання (обрано стек, що надає Bigtop v3.2.1, а саме BGTP-1.0);

– налаштування параметрів реєстрації хостів. Надано список хостів, на яких розгортатиметься екосистема, а також обрано один з двох запропонованих варіантів реєстрації – використовуючи SSH або ручна реєстрація. При використанні SSH Ambari сам використовуючи наданий згенерований ssh-ключ встановить та налаштує агенти Ambari на усіх наведених хостах. При ручній реєстрації необхідно виконати встановлення та налаштування агентів вручну. Обрано другий варіант реєстрації;

– реєстрація хостів. цей етап проводить реєстрацію та попередню перевірку хостів на можливість розгортання сервісів Hadoop;

– вибір сервісів для розгортання. Обрано сервіси HDFS, YARN + MapReduce2, Tez, Hive, ZooKeeper, Ambari Metrics та Spark;

– розподіл master-компонентів на вузлах кластера;

– розподіл slave-/client-компонентів на робочих вузлах кластера;

– налаштування конфігурації сервісів;

– розгортання.

Після успішного розгортання як результат дослідження отримано тестовий кластер Hadoop, розгорнутий з використанням виключно власних збудованих артефактів Ambari та Bigtop.

Головний дашборд розгорнутого тестового кластера Hadoop в Ambari зображено на рис. 3.4.

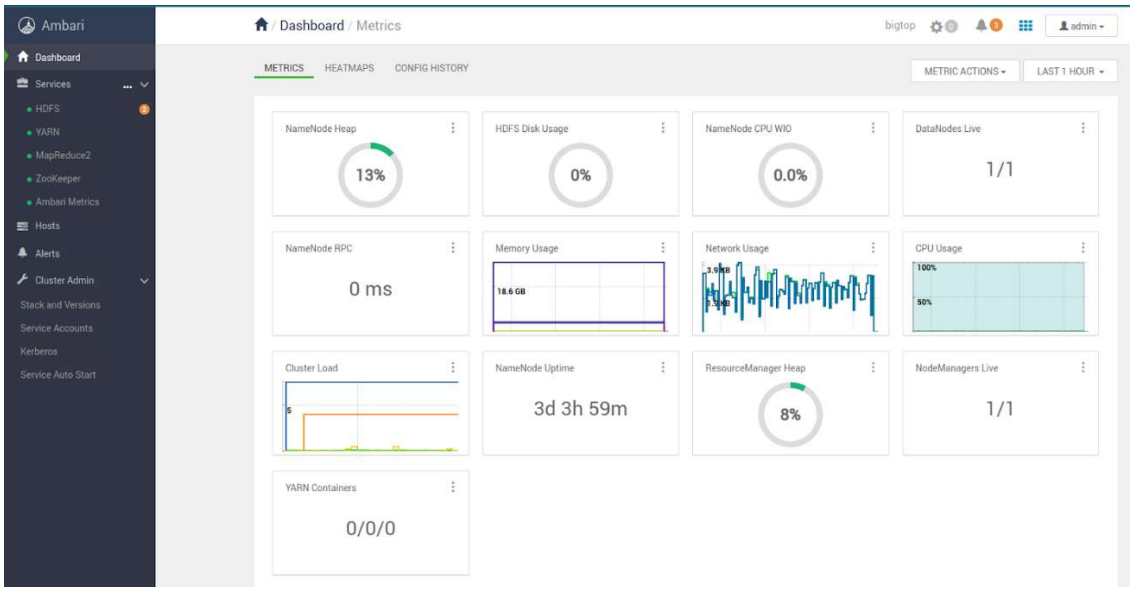

Рис. 3.4. Головний дашборд розгорнутого тестового кластера Hadoop в Ambari

3.3. Варіанти переналаштування артефактів екосистеми Hadoop з використанням Apache Bigtop

На даний момент існує три основних методи переналаштування артефактів екосистеми Hadoop з використанням Apache Bigtop:

– створення власних відгалужень або використання відгалужень інших організацій GitHub репозиторіїв оригінальних сервісів та внесення змін

безпосередньо в код програм, після чого необхідно змінити посилання на репозиторій у файлі налаштування “bigtop.bom”;

– створення власних git-патчів для внесення змін у локальний репозиторій, клонований з оригінального GitHub репозиторію. Такий механізм вже реалізований для артефактів стеку Bigtop. Вони застосовуються під час формування артефактів для виправлення помилок, покращення коду оригінальних проєктів сервісів або забезпечення сумісності між сервісами;

– реалізація та інтеграція власних сервісів у стек. Даний метод передбачає повну розробку власного сервісу (створення нового проєкту, його структури, написання програмних скриптів, цифрове підписування файлів тощо), а також написання тестів та реалізацію механізму пакування двійкових файлів [35].

Дослідивши кожен з наведених методів можна дійти висновку, що метод створення git-патчів є найбільш збалансованим рішенням для адаптації існуючих сервісів із мінімальними ризиками. Використання відгалужень може бути корисним для швидкого тестування нових ідей або тимчасових змін, однак у довгостроковій перспективі це ускладнює підтримку та оновлення, особливо при інтеграції з оригінальними репозиторіями. У випадку унікальних потреб чи інноваційних задач варто розглянути реалізацію власних сервісів, хоча це потребує більше ресурсів і часу.

Переваги та недоліки кожного з вищеописаних методів переналаштування артефактів екосистеми Hadoop з використанням Apache Bigtop наведено у таблиці 3.2.

Таблиця 3.2

Переваги та недоліки методів переналаштування артефактів екосистеми Hadoop з використанням Apache Bigtop

Метод	Переваги	Недоліки
Відгалуження GitHub-репозиторіїв і модифікація коду	Гнучкість у модифікації оригінального коду. Можливість адаптації сервісів до потреб.	Потребує глибокого розуміння структури програм. Ускладнює оновлення сервісів через підтримку власного відгалуження.

Продовж. табл. 3.2

Власні git-патчі	Модульність змін. Патчі легко інтегрувати або видаляти. Спрощене оновлення завдяки збереженню оригінального коду. Повторне використання.	Розробка патчів може бути складною для великих змін. Менша гнучкість у порівнянні з прямим редагуванням коду.
Інтеграція власних специфічних сервісів	Повний контроль над функціональністю. Можливість створення унікальних рішень. Незалежність від обмежень існуючих сервісів.	Значні витрати часу та ресурсів на розробку. Висока складність реалізації та підтримки. Потребує досвіду в розробці програмного забезпечення.

3.4. Налаштування власного CI/CD з використанням сервісу Jenkins

Розробники Apache Bigtop керують власним налаштуванням системи CI і використовують при цьому сервіс Jenkins. Окрім цього, використання контейнерів Docker забезпечують легку інтеграцію проєкту з інструментами безперервної інтеграції та доставки, що дозволяє автоматизувати процеси розробки, тестування та впровадження нових артефактів у кластер Hadoop.

Всього у розробників Bigtop 1 головний сервер + 4 підлеглих агентів. Є одне сховище 1TB, призначене головному, ще 2 * 800 GB виділяється двом підлеглим агентам (06 і 07), відповідно. Іншим двом вузлам (02 і 03) виділено 200 GB для роботи кожного вузла, тому вони зарезервовані для виконання завдань, неважливих до пам’яті, таких як ініціалізація.

Щоб налаштувати майстер-сервер Jenkins за допомогою Docker виконано відповідні команди. Лістинг команд для налаштування майстер-сервера Jenkins з використанням Docker зображено на рис. 3.5.

```
sudo adduser jenkins -u 1000
sudo apt install -y docker git
sudo usermod -aG docker jenkins
sudo su - jenkins
docker run -d --name jenkins-master -p 8080:8080 -v
/home/jenkins:/var/jenkins_home jenkins:jenkins:lts-jdk17
```

Рис. 3.5. Лістинг команд для налаштування майстер-сервера Jenkins з використанням Docker

Після запуску Jenkins необхідно завантажити додаткові плагіни: credentials, disk-usage, dynamic axis, git, gradle, junit, matrix project, ssh agent, timestamper.

Для налаштування так званого «ізольованого контролера Jenkins», необхідно додатково налаштувати мінімум ще один підлеглий-агент сервер, на якому будуть виконуватись усі CI/CD задачі. Для цього згенеровано SSH ключ мастер-сервера та додано його публічний на підлеглому агенті.

Лістинг команд для налаштування підлеглого агента Jenkins зображено на рис. 3.6.

```
sudo apt install docker docker-compose git java ruby
sudo adduser jenkins
sudo usermod -aG docker jenkins
sudo su - jenkins
mkdir .ssh
cat << EOF > .ssh/authorized_keys
> (MASTER_PUBLIC_KEY)
> EOF
chmod 700 .ssh
chmod 400 .ssh/authorized_keys
```

Рис. 3.6. Лістинг команд для налаштування підлеглого агента Jenkins

Після цього необхідно безпосередньо на Jenkins створити секрети приватного ssh-ключа мастер-сервера та додати нового агента, вказавши підключення через ssh. В налаштуваннях агента вказати необхідну кількість виконувачів (executors), а в налаштуваннях мастера вказати дану кількість 0.

Після налаштування Jenkins створено 3 CI/CD завдання — Bigtop-3.2.1-packages, Bigtop-3.2.1-deployments та Bigtop-3.2.1-smoke-tests. Для створення нового

завдання необхідно спочатку створити новий елемент, тип “Multi-configuration project”. В розділі “Source Code Management” заповнити URL git репозиторію та вказати гілку, а у розділі “Configuration Matrix” додати “user-defined axis”, вказавши ім’я та значення.

Для Bigtop-3.2.1-packages та Bigtop-3.2.1-deployments завдань:

– ім’я: BUILD_ENVIRONMENTS. Значення: ubuntu-22.04;

– ім’я: COMPONENTS. Значення: ambari hadoop hive spark zookeeper.

Для Bigtop-3.2.1-smoke-tests завдання:

– ім’я: OS. Значення: ubuntu-22.04-amd64-deploy;

– ім’я: COMPONENTS. Значення: ambari.bigtop-utils@ambari; hdfs.yarn@hdfs; hdfs.yarn.hive@hive; hdfs.yarn@mapreduce; hdfs.yarn.spark@spark; hdfs.yarn@yarn; zookeeper@zookeeper.

Компоненти до знака ‘@’ — ті, які необхідно розгорнути. Компоненти після — ті, які необхідно протестувати.

Скрипти для виконання завдань CI/CD реалізації наведено у додатку Б.

Dashboard зі створеними завданнями та результатами їх виконання в Jenkins зображено на рис. 3.7.

Рис. 3.7 — Dashboard зі створеними завданнями та результатами їх виконання в Jenkins

Отже, після успішного практичного дослідження та реалізації запропонованого методу розгортання артефактів екосистеми опрацювання великих даних Hadoop

виконано додаткове порівняння з іншими некомерційними рішеннями. Зокрема, порівняння охоплюють описаний у другому розділі фреймворк DICE та використання лише одного інструменту для формування та розгортання артефактів Apache Bigtop.

Оцінку та порівняння проведено на основі наступних критеріїв:

- зручність використання;
- портативність;
- вимоги до ресурсів;
- відтворюваність;
- гнучкість;
- масштабованість.

Уся необхідна інформація для порівняння була або безпосередньо протестована, або взята з відповідної документації проєктів та інших літературних джерел.

Оцінка зручності використання включає в себе не лише необхідний технічний рівень знань, але й легкість використання. У запропонованому методі складність полягає в підготовці кожного компонента для розгортання, проведення тестування та складанні всіх необхідних елементів для створення архітектури. Проте після того, як архітектура підготовлена, її можна розгорнути з мінімальними технічними знаннями. Користувачеві потрібно лише виконати відповідний Ansible плейбук для вузлів розгортання, а весь процес далі автоматизується.

Сам проєкт Apache Bigtop надає спеціальний shell-скрипт, який готує кожен вузол для пакетного розгортання, що забезпечується спільнотою розробників. У разі використання Docker основна увага приділяється Hadoop та його екосистемі. Керування розгортанням здійснюється в лише через Puppet-скрипти. При цьому графічного вебінтерфейсу для простоти розгортання та подальшого управління розгорнутим кластером Hadoop немає.

Фреймворк DICE здебільшого вимагає лише використання плагіна Eclipse. З його допомогою IDE проводить користувача через різні стадії розробки та розгортання дата-інтенсивних додатків (DIA), включаючи моделювання та початкову

реалізацію прототипу. Проте значний обсяг інформації, необхідний на всіх етапах, ускладнює швидкий старт або оперативне розгортання системи.

Оцінка портативності передбачає визначення зусиль, необхідних для роботи або міграції архітектури на різні ресурси. Завдяки можливості як проєкту Ambari, так і проєкту Bigtop формувати артефакти, що можуть бути призначені для розгортання на різних операційних системах, запропонований метод розгортання забезпечує високу портативність. Для перевірки цього критерію змінювались цільові машини та операційні системи на них. Проте, варто відзначити, що контейнерні технології, такі як Docker, здатні забезпечити максимальну портативність лише з мінімальними змінами в конфігурації.

Для самого рішення Bigtop розгортання артефактів не забезпечує легкої портативності, оскільки перенесення встановленої системи на інший ресурс вимагає налаштування всіх puppet-скриптів з нуля.

DICE дозволяє взаємодіяти з різними хмарними платформами, де можна розгортати DIA. Використання переналаштовуваних файлів і IDE дозволяє досягти високого рівня портативності.

Для детального дослідження відтворюваності перевірено, чи забезпечує запропонований метод розгортання однакову архітектуру розгортання при однакових конфігураціях. В результаті трьох спроб розгортання в кожному випадку архітектура отримувала однаковий вигляд.

Для DICE результати можна отримати аналогічні в тому випадку, якщо все налаштовано правильно, але ці конфігурації є в свою чергу складнішими.

Щодо критерію вимог до ресурсів, то у даному випадку при однакових конфігураціях кожен метод розгортання використовує приблизно одну і ту ж кількість ресурсів.

Щодо критерію гнучкості, то запропонований підхід дозволяє відносно легко додавати, видаляти та модернізувати сервіси екосистеми Hadoop використовуючи стек Ambari. DICE у свою чергу не пропонує додаткових технологій через високу складність інструментів, конфігурацій та припинення підтримки проєкту у 2018 році.

Щодо критерію масштабованості, то запропонований метод має в цьому плані суттєву перевагу, так як рішення Apache Ambari дозволяє легко та без проблем масштабовувати кластер Hadoop. Інші ж рішення в цьому плані мають певні обмеження, а саме потребу оновлювати конфігурацію архітектури вручну, незалежно чи це необхідно виконувати у файлах blueprint, чи у додаткових автоматизаційних інструментах Ansible, Chef та Puppet.

Підводячи підсумки порівняння та проведених досліджень можна впевнено стверджувати, що запропонований метод є робочим і достатньо конкурентноспроможнім. Він виступає кращим рішенням серед інших некомерційних рішень за низкою критеріїв, а саме має перевагу у критеріях:

- зручність використання;
- портативність;
- масштабованість;
- гнучкість.

РОЗДІЛ 4
ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1. Охорона праці

Оскільки, дослідження методів та засобів розгортання артефактів екосистеми опрацювання великих даних Hadoop, а також застосування програмних інструментів автоматизації передбачає використання комп'ютерної техніки, зокрема ПК та периферійних пристроїв, то обов'язковим є дотримання вимог з охорони праці і техніки безпеки.

Для ефективної і безпечної роботи колективу комп'ютерних та програмних інженерів, необхідно організувати безпечні умови праці. При цьому керівник організації несе безпосередню відповідальність за порушення нормативно-правових актів з охорони праці. Окрім цього, на робочих місцях працівників необхідно забезпечити дотримання вимог, затверджених Наказом Мінсоцполітики від 14.02.2018 за № 207 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями».

Приміщення, в яких розміщуються робочі місця працівників загального призначення, обладнуються системою автоматичної пожежної сигналізації та засобами пожежогасіння відповідно до вимог ДБН В.2.5-56:2014, НАПБ А.01.001-2014 і вимог нормативно-технічної та експлуатаційної документації виробника. Проходи до засобів пожежогасіння мають бути вільними.

Лінія електромережі для живлення комп'ютера та периферійних пристроїв повинні бути виконаними як окрема групова трипровідна мережа шляхом прокладання фазового, нульового робочого та нульового захисного провідників. Нульовий захисний провідник використовується для заземлення (занулення) електроприймачів. Не допускається використовувати нульовий робочий провідник як нульовий захисний провідник. Нульовий захисний провідник прокладається від стійки групового розподільчого щита, розподільчого пункту до розеток електроживлення.

Площа перерізу нульового робочого та нульового захисного провідника в груповій трипровідній мережі має бути не менше площі перерізу фазового провідника. Усі провідники мають відповідати номінальним параметрам мережі та навантаження, умовам навколишнього середовища, умовам розподілу провідників, температурному режиму та типам апаратури захисту, вимогам НПАОП 40.1-1.01-97.

У приміщенні, де одночасно експлуатуються понад п’ять комп’ютерів, на помітному, доступному місці встановлюється аварійний резервний вимикач, який може повністю вимкнути електричне живлення приміщення, крім освітлення. Комп’ютери повинні підключатися до електромережі тільки за допомогою справних штепсельних з'єднань і електророзеток заводського виготовлення.

Не допускається підключати комп’ютери до звичайної двопровідної електромережі, в тому числі – з використанням перехідних пристроїв. Електромережі штепсельних з'єднань та електророзеток для живлення комп’ютерної техніки повинні бути виконаними за магістральною схемою, по 3-6 з'єднань або електророзеток в одному колі. Штепсельні з'єднання та електророзетки для напруги 12 В та 42 В за своєю конструкцією мають відрізнятися від штепсельних з'єднань для напруги 127 В та 220 В. Штепсельні з'єднання та електророзетки, розраховані на напругу 12 В та 42 В, мають візуально (за кольором) відрізнятися від кольору штепсельних з'єднань, розрахованих на напругу 127 В та 220 В.

Важливим, з точки зору охорони праці, є забезпечення достатньої величини природного та штучного освітлення. Нормативні величини освітленості робочих місць для різних видів робіт та відповідних зорових навантажень визначаються ДБН В.2.5-28:2018 «Природне і штучне освітлення».

Організація робочого місця комп’ютерного інженера повинна забезпечувати відповідність усіх елементів робочого місця та їх розташування ергономічним вимогам ДСТУ 8604:2015 «Дизайн і ергономіка. Робоче місце для виконання робіт у положенні сидячи. Загальні ергономічні вимоги». Відстань від екрана до ока фахівців, які працюють за комп’ютером визначається згідно з вимогами ДСанПіН 3.3.2.007-98.

Розміщення принтера або іншого пристрою введення-виведення інформації на робочому місці має забезпечувати добру видимість екрана комп’ютера, зручність

ручного керування пристроєм введення-виведення інформації в зоні досяжності моторного поля згідно з вимогами ДСанПіН 3.3.2.007-98.

Таким чином, у результаті аналізу вимог щодо охорони праці користувачів комп'ютерів, визначено особливості організації робочих місць, вимог з електробезпеки, природного та штучного освітлення для ефективної і безпечної роботи фахівців на підприємтвах

4.2. Підвищення надійності захисту працівників підприємства під час роботи в надзвичайних ситуаціях

При оцінці надійності захисту виробничого персоналу необхідно враховувати, що практично будь-які наслідки надзвичайної ситуації (далі – НС) можуть призвести до ураження людей та стати причиною їхньої смерті або призвести до втрати працездатності на тривалий час [6, 36].

Надійність захисту виробничого персоналу є одним з важливих факторів, які визначають стійкість роботи підприємств у надзвичайних ситуаціях мирного та воєнного часів.

Вихідні дані для розрахунку:

Загальна чисельність найбільшої працюючої зміни – 100 осіб. При цьому лише частина з них здатна укритися своєчасно в ЗС, а саме 80 осіб. Згідно з даними, сигнал про НС своєчасно отримують 90 осіб. При цьому, лише 70 осіб успішно пройшли необхідні інструктажі та навчання. Встановлено, що фактичний час підготовки споруд становить 20 хвилин, тоді як допустимий дорівнює 30 хвилинам. Необхідно визначити показник надійності захисту робочого персоналу під час НС на підприємстві.

Коефіцієнт надійності захисту як показник надійності визначається на основі окремих показників, що характеризують підготовленість об'єкту до виконання завдань захисту робітників та службовців за основними складовими задачами.

Оцінку надійності захисту виробничого персоналу проведено в такій послідовності:

– оцінка інженерного захисту робітників та службовців об’єкта визначається за формулою:

$$К_{ІНЖ.ЗАХ.}\frac{N_{ІНЖ.ЗАХ.}}{N}, \quad (4.1)$$

де $К_{ІНЖ.ЗАХ.}$ – коефіцієнт інженерного захисту робітників;

$N_{ІНЖ.ЗАХ.}$ – кількість працівників, які можуть прийняти укриття;

N – загальна чисельність найбільшої працюючої зміни.

Таким чином, коефіцієнт інженерного захисту робітників становить 0,9 (90 / 100).

– вивчається система сповіщення та оцінюється можливість своєчасного доведення сигналу сповіщення до робітників та службовців. Коефіцієнт сповіщення визначається за формулою:

$$К_{СП}\frac{N_{СП}}{N}, \quad (4.2)$$

де $К_{СП}$ – коефіцієнт сповіщення;

$N_{СП}$ – кількість сповіщених працівників про необхідність перейти в укриття;

N – загальна чисельність найбільшої працюючої зміни.

Таким чином, коефіцієнт сповіщення робітників становить 0,8 (80 / 100).

– оцінка навченості виробничого персоналу способам захисту та діям за сигналами сповіщення визначається за формулою:

$$К_{НАВЧ}\frac{N_{НАВЧ}}{N}, \quad (4.3)$$

де $К_{НАВЧ}$ – коефіцієнт навченості;

$N_{НАВЧ}$ – кількість навчених працівників;

N – загальна чисельність найбільшої працюючої зміни.

Таким чином, коефіцієнт навченості робітників становить 0,7 (70 / 100).

– далі визначається готовність сховища до прийому людей. Порівнюючи фактичний час підготовки сховища $Т_{Г.\ ФАК.}$ з потрібним $Т_{Г.\ ПОТ.}$, визначається готовність сховища до прийому людей. Для оцінки надійності захисту враховуються лише ті сховища, для яких працює закономірність $Т_{Г.ФАК.}$ розділені на $Т_{Г.ПОТ.}$ є менше або рівним одиниці, тому відповідно до вхідних даних $К_{ГОТ.}$ дорівнює одиниці, адже $Т_{Г.ФАК.} = 20$, а $Т_{Г.ПОТ.} = 30$;

На основі отриманих показників коефіцієнт надійності захисту робітників та службовців $К_{Н.З.}$ визначається наступною формулою:

$$К_{Н.З.} = \min(К_{ІНЖ.ЗАХ.}, К_{СП.}, К_{НАВЧ.}, К_{ГОТ.}), \quad (4.3)$$

де $К_{Н.З.}$ – коефіцієнт надійності захисту.

Таким чином, коефіцієнт надійності захисту становить 0,7.

За результатом проведеного дослідження виявлено слабкі місця в організації захисту працівників, зокрема недостатню місткість укриттів і низький рівень навченості персоналу. Для підвищення надійності захисту рекомендовано збільшити місткість захисних споруд, організувати додаткові навчання для працівників і регулярно перевіряти готовність споруд до використання. Очікується, що реалізація цих заходів дозволить підвищити загальний коефіцієнт надійності захисту до рівня $К_{Н.\ З.} \geq 0{,}9$, що є оптимальним показником для ефективного функціонування підприємства в умовах надзвичайних ситуацій [6, 36].

ВИСНОВКИ

Основні висновки та наукові результати дослідження можна узагальнити наступним чином:

1. На основі аналізу технічної документації, публікацій, вихідного коду проєктів та наукових джерел, що присвячені екосистемі Hadoop та розгортанню артефактів сервісів даної екосистеми, проведено огляд та аналіз наявних на даний момент методів та засобів, сформульовано їх переваги та недоліки.

2. Відповідно до проведеного дослідження методів та засобів для розгортання артефактів, обдумано та сформовано власний комбінований метод розгортання екосистеми Hadoop, який відповідає поставленим вимогам і є оптимальною з точки зору вартості та доступності даного рішення.

3. Обрано та апробовано засоби для автоматизації розгортання артефактів, відповідно до запропонованого методу.

4. Проведено експериментальні дослідження, в ході яких вдалось успішно розгорнути стек власних артефактів екосистеми Hadoop у приватному хмарному середовищі, використовуючи обрані інструменти – Ambari та Apache Bigtop. Додатково реалізовано та інтегровано у хмарне середовище обрану систему CI/CD, з допомогою якої вдалось автоматизувати низку рутинних процесів, таких як формування артефактів та проведення попереднього їх розгортання і тестування.

5. Здійснено аналіз отриманих результатів, проведено порівняння результатів запропонованих інструментів автоматизації з іншими некомерційними рішеннями, а також обґрунтовано доцільність запропонованого методу розгортання власних артефактів екосистеми опрацювання великих даних Hadoop.

Таким чином запропоновані підходи до автоматизації розгортання артефактів екосистеми Apache Hadoop реалізовані у вигляді певних наборів конфігурацій та програмних сценаріїв й апробовані, а поставлені задачі магістерського дослідження виконано.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Луцик Н.С., Луцків А.М., Осухівська Г.М., Тиш Є.В. Програма та методичні рекомендації з проходження практики за тематикою кваліфікаційної роботи для студентів спеціальності 123 «Комп'ютерна інженерія» другого (магістерського) рівня вищої освіти усіх форм навчання. Тернопіль: ТНТУ. 2024. 45 с.

2. Луцик Н.С., Луцків А.М., Осухівська Г.М., Тиш Є.В. Методичні рекомендації до виконання кваліфікаційної роботи магістра для студентів спеціальності 123 «Комп'ютерна інженерія» другого (магістерського) рівня вищої освіти усіх форм навчання. Тернопіль: ТНТУ. 2024. 44 с.

3. Варавін А.В., Лещишин Ю.З., Чайковський А.В. Методичні вказівки до виконання курсового проєкту з дисципліни «Дослідження і проєктування комп'ютерних систем та мереж» для здобувачів другого (магістерського) рівня вищої освіти спеціальності 123 «Комп'ютерна інженерія» усіх форм навчання. Тернопіль: ТНТУ, 2024. 32 с.

4. Мельник Н. Методи та засоби розгортання артефактів екосистеми Hadoop. Матеріали VII міжнародної студентської науково-технічної конференції "Природничі та гуманітарні науки. Актуальні питання" Тернопільського національного технічного університету імені Івана Пулюя, Тернопіль: ТНТУ, 2024, 345-346 с.

5. Мельник Н. О. Розгортання власних артефактів екосистеми Hadoop. Матеріали V міжнародної науково-практичної конференції учених та студентів «Цифрова економіка як фактор інновацій та сталого розвитку суспільства» Тернопільського національного технічного університету імені Івана Пулюя, Тернопіль: ТНТУ, 2024, 182-183 с.

6. Стручок В. С. Методичний посібник для здобувачів освітнього ступеня «магістр» всіх спеціальностей денної та заочної (дистанційної) форм навчання «БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ» Тернопіль: ТНТУ, 2024. 155 с.

7. Микитишин А. Г., Митник М. М., Стухляк П. Д., Пасічник В. В. Комп’ютерні мережі. Книга 1 [навчальний посібник]. Львів: «Магнолія 2006», 2013. 256 с.

8. Микитишин А. Г., Митник М. М., Стухляк П. Д., Пасічник В. В. Комп’ютерні мережі. Книга 2. [навчальний посібник]. Львів: "Магнолія 2006", 2014. 312 с.

9. Джавад А. Ш., Мухаммад А. Х. Big Data Systems. A 360-degree Approach. Лондон: Taylor & Francis, 2023. 340 с.

10. Офіційний сайт Apache Hadoop. Apache Hadoop. URL: https://hadoop.apache.org (дата звернення: 10.09.2024).

11. Інформація про HDFS. HDFS Architecture. URL: https://hadoop.apache.org/docs/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html (дата звернення: 20.09.2024).

12. Інформація про фреймворк MapReduce. MapReduce Tutorial. URL: https://hadoop.apache.org/docs/stable/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html (дата звернення: 20.09.2024).

13. Офіційний сайт Apache Zookeeper. Apache Zookeeper. URL: https://zookeeper.apache.org (дата звернення: 21.09.2024).

14. Офіційний сайт Apache Hive. Apache Hive. URL: https://hive.apache.org (дата звернення 25.09.2024).

15. Офіційний сайт Apache Spark. Apache Spark. URL: https://spark.apache.org (дата звернення: 05.10.2024).

16. Офіційний сайт Apache Kafka. Apache Kafka. URL: , (дата звернення: 06.10.2024).

17. Apache Bigtop. Apache Software Foundation. URL: https://bigtop.apache.org/, (дата звернення: 10.10.2024).

18. Apache Ambari. Apache Software Foundation. URL: https://ambari.apache.org/, (дата звернення: 11.10.2024).

19. Defining a Custom Stack and Services. Confluence Apache Ambari. URL: https://cwiki.apache.org/confluence/display/AMBARI/Defining+a+Custom+Stack+and+Services, (дата звернення: 22.10.2024).

20. Davoudian, A. and M. Liu, "Big Data Systems", ACM Computing Surveys, 53(5), 2020, pp. 1–39.

21. Chang, W.L. and N. Grady, "NIST Big Data Interoperability Framework: Volume 1, Definitions", 2019.

22. Roy, C., S. Swarup Rautaray, and M. Pandey, "Big Data Optimization Techniques: A Survey", International Journal of Information Engineering and Electronic Business, 10(4), 2018, pp. 41–48.

23. Günther, W.A., M.H. Rezazade Mehrizi, M. Huysman, and F. Feldberg, "Debating big data: A literature review on realizing value from big data", The Journal of Strategic Information Systems, 26(3), 2017, pp. 191–209.

24. Volk, M., D. Staegemann, N. Jamous, M. Pohl, and K. Turowski, "Providing Clarity on Big Data Technologies", International Journal of Intelligent Information Technologies, 16(2), 2020, pp. 49–73.

25. Volk, M., D. Staegemann, S. Bosse, R. Häusler, and K. Turowski, "Approaching the (Big) Data Science Engineering Process", in Proceedings, 5th International Conference on Internet of Things, Big Data and Security, Prague, Czech Republic. 2020. SCITEPRESS.

26. Immonen, A., P. Paakkonen, and E. Ovaska, "Evaluating the Quality of Social Media Data in Big Data Architecture", IEEE Access, 3, 2015, pp. 2028–2043.

27. Kune, R., P.K. Konugurthi, A. Agarwal, R.R. Chillarige, and R. Buyya, "The anatomy of big data computing", Software: Practice and Experience, 46(1), 2016, pp. 79–105.

28. Casale, G. and C. Li, "Enhancing Big Data Application Design with the DICE Framework", in Advances in Service-Oriented and Cloud Computing, Z.Á. Mann and V. Stolz, Editors. Springer International Publishing : Cham, 2018.

29. Офіційна документація Cloudera. Cloudera. URL: https://docs.cloudera.com/cdsw/1.10.5/architecture-overview/topics/cdsw-cloudera-manager.html (дата звернення: 23.10.2024).

30. Офіційний сайт Clemlab Ambari та ODP. Open Source Data Platform - ODP - Clemlab. URL: https://www.opensourcedataplatform.com/ (дата звернення: 24.10.2024)

31. Офіційна документаця Puppet. Documentation Puppet by Perforce. URL: https://help.puppet.com/ (дата звернення: 25.10.2024).

32. Офіційна документаця Chef. Chef Documentation. URL: https://docs.chef.io/ (дата звернення: 25.10.2024).

33. Офіційна документаця Ansible. Ansible Documentation. URL: https://docs.ansible.com/ansible/latest/index.html (дата звернення: 25.10.2024).

34. Реалізація CI від Apache Bigtop. Bigtop CI Setup Guide. URL: https://cwiki.apache.org/confluence/display/BIGTOP/Bigtop+CI+Setup+Guide (дата звернення: 04.11.2024).

35. Формування та реалізація власного стеку Ambari. How-To Define Stacks and Services. URL: https://cwiki.apache.org/confluence/display/AMBARI/How-To+Define+Stacks+and+Services, (дата звернення: 06.11.2024).

36. В.С. Стручок. Навчальний посібник «ТЕХНОЕКОЛОГІЯ ТА ЦИВІЛЬНА БЕЗПЕКА. ЧАСТИНА «ЦИВІЛЬНА БЕЗПЕКА»». Тернопіль: ФОП Паляниця В. А., 2022. 156 с.

37. A. Lutskiv, N. Popovych, Big data-based approach to automated linguistic analysis effectiveness, Proceedings of the 2020 IEEE 3rd International Conference on Data Stream Mining and Processing, DSMP 2020, Lviv, (2020) 438-443.

38. A. Lutskiv, N. Popovych, Big data approach to developing adaptable corpus tools CEUR Workshop Proceedings, Lviv, (2020) 374-395.

Додаток А.

Тези конференцій

Міністерство освіти і науки України
Тернопільський національний технічний університет
імені Івана Пулюя
Маріборський університет (Словенія)
Технічний університет в Кошице (Словаччина)
Каунаський технологічний університет (Литва)
Львівський національний університет
імені Івана Франка,
Гірничо-металургійна академія ім. Станіслава Сташиця (Польща)
Луцький національний технічний університет,
Чернівецький національний університет
імені Юрія Федьковича,
Вроцлавський економічний університет (Польща)
Університет технологій та економіки
імені Хелени Ходковської (Польща)
Донбаська державна машинобудівна академія

Студентське наукове товариство

VII МІЖНАРОДНА

студентська науково - технічна конференція

"ПРИРОДНИЧІ ТА ГУМАНІТАРНІ НАУКИ.

АКТУАЛЬНІ ПИТАННЯ"

25-26 квітня 2024 р.

(збірник тез конференції)

Тернопіль 2024

УДК 004.45+004.62
Мельник Н. – ст. гр. СІм-52
Тернопільський національний технічний університет імені Івана Пулюя

МЕТОДИ ТА ЗАСОБИ РОЗГОРТАННЯ АРТЕФАКТІВ ЕКОСИСТЕМИ HADOOP

Науковий керівник: доцент, кандидат технічних наук Луцків А. М.

Melnyk N.
Ternopil Ivan Puluj National Technical University

METHODS AND MEANS OF DEPLOYING ARTIFACTS OF THE HADOOP ECOSYSTEM

Supervisor: Associate Professor, PhD A. Lutskiv

Ключові слова: кластер, Hadoop, артефакт, великі дані
Keywords: cluster, Hadoop, artifact, Big Data

Every day the amount of different types of data produced by humanity increases greatly. Therefore, software tools for data management and processing are of great importance. Due to its volume, velocity and variety, this data can be treated as a Big Data [1]. Nowadays, appropriate software tools mostly belong to Apache Hadoop ecosystem [2] which comprises a lot of tools and libraries developed by different software engineering teams with open source licenses. To provide reliable, resilient and efficient platform for data processing these tools should be consistent in the following aspects: library versions, configuration and deployment. The main objective of this research is to provide an approach to resolve this task.

Big Data management is cyclical in nature. It can be generally divided into five phases: capture, organize, integrate, analyze, and act (Fig. 1). Hadoop components are specially designed to perform tasks on each of the listed phases [3].

Figure 1. Cycle of Big Data Management [2]

The Hadoop ecosystem is a collection of freely distributed open-source software products designed for processing and analyzing large volumes of data. This ecosystem emerged in response to the need for efficient handling and storing large volumes of structured

and unstructured data, which is constantly growing. The main components of the Hadoop ecosystem are Hadoop Distributed File System (HDFS), MapReduce, Hadoop YARN (Yet Another Resource Negotiator), and many additional projects, that extend the capabilities of Hadoop core components for various use cases. The most popular of these additional projects are Spark, Zookeeper, Hive, HBase, and Kafka.

Hadoop artifacts are built packages of software components of the Hadoop ecosystem. Efficient deployment and testing of Hadoop artifacts are critically important for several reasons, such as system reliability, performance optimization, resource utilization, data security, and core quality assurance.

It is also worth noting separately the Apache Ambari and Apache Bigtop projects, which are related to research on the given topic.

The main goal of Ambari [4] is to provide provisioning, management, monitoring, and operating Hadoop clusters at scale. Ambari gives a central location to examine the status, configuration, and resource usage for each one of these. For provisioning a Hadoop cluster, Ambari provides a step-by-step wizard for installing Hadoop services across any number of hosts. Ambari handles the configuration of Hadoop services for the cluster. For managing the cluster, Ambari provides central management for starting, stopping, and reconfiguring Hadoop services across the entire cluster. For monitoring a Hadoop cluster, Ambari provides a dashboard for monitoring the health and status of the Hadoop cluster, leverages a metric system for collecting metrics, and also leverages an alert framework to notify the administrators when nodes go down, disk space is low, or anything requires an administrator's attention.

Apache Bigtop [5] serves as a comprehensive platform for packaging, testing, and deploying Hadoop ecosystem components. It provides a uniform build and testing environment for various Hadoop projects, ensuring compatibility and interoperability across the ecosystem. Also, this project allows customization of Hadoop distributions to suit specific requirements.

In summary, Apache Ambari and Apache Bigtop play pivotal roles in simplifying cluster management, ensuring quality assurance, fostering collaboration, and driving innovation within the Hadoop ecosystem. Their adoption is essential for organizations looking to harness the full potential of Hadoop for Big Data processing and analytics. The result of this research will be a fully configured and tested cluster with Apache Hadoop ecosystem, taking into account performance and data security issues as well.

References

1. Jawwad A. S., Muhammad A. K. Big Data Systems. A 360-degree Approach. Chapman & Hall, 2021.

2. Apache Hadoop. Apache Software Foundation. URL: https://hadoop.apache.org/, accessed 1 May 2024.

3. Kaur I. Cycle of Big Data Management. (2020) URL: https://ishmeetk10.medium.com/cycle-of-big-data-management-ac9b99899a44.

4. Apache Ambari. Apache Software Foundation. URL: https://ambari.apache.org/, accessed 1 May 2024.

5. Apache Bigtop. Apache Software Foundation. URL: https://bigtop.apache.org/, accessed 1 May 2024.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
імені ІВАНА ПУЛЮЯ
НАУКОВЕ ТОВАРИСТВО ім. ШЕВЧЕНКА

М А Т Е Р І А Л И

XII НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ

«ІНФОРМАЦІЙНІ МОДЕЛІ, СИСТЕМИ ТА ТЕХНОЛОГІЇ»

18–19 грудня 2024 року

ТЕРНОПІЛЬ
2024

УДК 004.45+004.62
Н. Мельник; А. М. Луцків, канд. техн. наук, доцент
(Тернопільський національний технічний університет імені Івана Пулюя, Україна)

РОЗГОРТАННЯ ВЛАСНИХ АРТЕФАКТІВ ЕКОСИСТЕМИ HADOOP

UDC 004.45+004.62
N. Melnyk; A. Lutskiv, Ph.D., Associate Professor

DEPLOYMENT OF OWN ARTIFACTS OF THE HADOOP ECOSYSTEM

Ключові слова: Великі Дані, Hadoop, артефакт, Bigtop, Ambari.
Key words: Big Data, Hadoop, artifact, Bigtop, Ambari.

Зростання обсягу даних у світі робить критично важливим використання програмного забезпечення для їх ефективної обробки та управління [1]. Однією з найпоширеніших платформ для роботи з Big Data є екосистема Apache Hadoop, яка забезпечує можливості для обробки великих обсягів структурованих і неструктурованих даних [2]. Одним із важливих аспектів у цьому контексті є розгортання власних артефактів Hadoop, що дозволяє адаптувати платформу під конкретні потреби організацій. Артефакти Hadoop – це сформовані пакети програмних компонентів екосистеми Hadoop.

Актуальність дослідження методу розгортання власних артефактів екосистеми опрацювання великих даних Hadoop обумовлена необхідністю забезпечення ефективної та надійної роботи з даними. Системи, побудовані на основі Hadoop, використовуються в різних сферах, таких як аналітика структурованих і неструктурованих даних, машинне навчання та управління ресурсами.

Основною метою використання методу розгортання власних артефактів Hadoop виступає забезпечення більшої незалежності від зовнішніх корпорацій та розробників, що надають свої рішення, а також забезпечення можливості повного налаштування екосистеми під власні потреби компанії методом перетворення і переконфігурування наявних рішень для власних потреб або створення власних сервісів і інтегрування їх у інфраструктуру.

Використання таких інструментів, як Apache Bigtop для побудови та тестування артефактів, та Apache Ambari для управління кластерами, дає змогу значно підвищити продуктивність шляхом переконфігурування сервісів платформи з урахуванням власних вимог та забезпечити високу якість розгорнутих систем.

Apache Bigtop виступає важливим інструментом для побудови й тестування артефактів екосистеми Hadoop. Він забезпечує можливість пакування та перевірки сумісності компонентів, що гарантує стабільність і ефективність роботи розгорнутих систем [3].

Для формування артефактів Apache Bigtop спочатку клонує необхідний GitHub репозиторій сервісу і застосовує спеціальні git-патчі для внесення необхідних змін у локальний репозиторій, не змінюючи при цьому оригінальні origin-гілки на GitHub. Після цього він запускає необхідну команду для формування проєкту і пакує двійкові файли у інсталяційні пакети (deb або rpm).

Існує три основних варіанти переналаштування екосистеми Hadoop з використанням Apache Bigtop:

- створення власних відгалужень або використання відгалужень інших організацій GitHub репозиторіїв оригінальних сервісів та внесення змін безпосередньо в код програм, після чого необхідно змінити посилання на репозиторій у файлі налаштування “bigtop.bom”;
- створення власних git-патчів для внесення змін у локальний репозиторій, клонований з оригінального GitHub репозиторію;
- реалізація та інтеграція власних сервісів у стек. Даний метод передбачає повну розробку власного сервісу (створення нового проєкту, його структури, написання програмних скриптів, цифрове підписування файлів тощо), а також написання тестів та реалізацію механізму пакування двійкових файлів.

Одним із ключових завдань при роботі з екосистемою Hadoop є розгортання та управління кластером. Apache Ambari спрощує цей процес, надаючи інструменти для налаштування, моніторингу та повного управління кластером Hadoop. Ambari забезпечує зручний інтерфейс для встановлення, конфігурації та оновлення сервісів Hadoop.

Для розгортання артефактів Ambari використовує попередньо визначені стеки. Стек Ambari – це сукупність компонентів та технологій, які використовуються самим програмним рішенням Ambari для спрощення та автоматизації розгортання, налаштування, управління та моніторингу окремих сервісів Hadoop та у загальному кластері Hadoop [4]. Кожен компонент стеку виконує певну роль при розгортанні та взаємодіє з іншими компонентами для досягнення спільної мети. Стеки можуть мати власні версії, а також вони забезпечують власні інструменти для керування версіями сервісів. Кожна версія стеку пропонує попередньо визначений перелік сервісів певних версій для забезпечення сумісності. Варто окремо відзначити, що для забезпечення сумісності необхідно використовувати вже наявні API та протоколи, реалізовувати обробку даних у форматах, визначених та підтримуваних сервісами із якими ведеться взаємодія, забезпечувати безпеку тощо. Для перевірки сумісності необхідно проводити тестування та перевіряти роботу як кожного з сервісів, так і взаємодію між ними перед використанням у кінцевому робочому середовищі (Production).

Використання таких інструментів, як Apache Bigtop та Apache Ambari, є критично важливим для побудови і розгортання артефактів екосистеми Hadoop. Завдяки цим інструментам можна досягти високого рівня продуктивності та стабільності роботи систем, що опрацьовують великі обсяги даних. Власні артефакти дають змогу адаптувати Hadoop до конкретних завдань організації, забезпечуючи при цьому ефективну обробку і зберігання даних.

Література

1. Джавад А. Ш., Мухаммад А. Х. Big Data Systems. A 360-degree Approach. Лондон: Taylor & Francis, 2023. 340 с.
3. Apache Hadoop. Apache Software Foundation. URL: https://hadoop.apache.org/, (дата звернення: 20.11.2024).
4. Apache Bigtop. Apache Software Foundation. URL: https://bigtop.apache.org/, (дата звернення: 22.11.2024).
5. Defining a Custom Stack and Services. Confluence Apache Ambari. URL: https://cwiki.apache.org/confluence/display/AMBARI/Defining+a+Custom+Stack+and+Services, (дата звернення: 04.12.2024).

Додаток Б.

Скрипти для виконання завдань CI/CD реалізації

Execute-скрипт завдання “Bigtop-3.2.1-packages”:

```
docker run --rm -v `pwd`:/ws --workdir /ws \
bigtop/slaves:3.2.1-$BUILD_ENVIRONMENTS \
bash -l -c "./gradlew allclean ${COMPONENTS}-pkg"
```

Execute-скрипт завдання “Bigtop-3.2.1-deployments”:

```
# destroy previous cluster
./gradlew docker-provisioner-destroy

# setup configuration file
CONFIG="config_${OS}_Bigtop-3.2.1-deployments.yaml"
sed "s/components.*/components: [hdfs, yarn, mapreduce, ${COMPONENTS}]/g"
provisioner/docker/config_${OS}.yaml > provisioner/docker/${CONFIG}

./gradlew docker-provisioner-destroy

# provision
# Since the puppet deployment will always return 0,
# we need to save the log and grep error to determine status
./gradlew -Pconfig=${CONFIG} -POS=${OS} -Pnum_instances=3 -Pprefix=3.2.1
docker-provisioner > tmp.log 2>&1
cat tmp.log


# destroy provisioned cluster
./gradlew docker-provisioner-destroy


# Fail the build if Errors occur in tmp.log
grep Error tmp.log && exit 1
exit 0
```

Execute-скрипт завдання “Bigtop-3.2.1-smoke-tests”:

```
# Setup configuration file
CONFIG="config_${OS}_Bigtop-smoke-tests.yaml"
DEPLOY_COMPONENTS=`echo ${COMPONENTS} | cut -d '@' -f 1 | sed "s/\\./,/g"`
TEST_COMPONENTS=`echo ${COMPONENTS} | cut -d '@' -f 2 | sed "s/\\./,/g"`
OS_FOR_URL=`echo ${OS} | awk -F"-" '{ print $1"-"$2"-"$3}' | sed "s@-@/@g"`

# The combination of DISTRO X ARCH is complicated. Resolved in the following
logic
OS_WO_ARCH=`echo ${OS} | awk -F"-" '{ print $1"-"$2}'`
ARCH=`echo ${OS} | awk -F"-" '{ print $3}'`
case "${ARCH}" in
  aarch64|arm64)
    ARCH_SUFFIX="-aarch64"
    ;;
  ppc64le|ppc64el)
```

```
    ARCH_SUFFIX="-ppc64le"
    ;;
  x86_64|amd64)
    ARCH_SUFFIX=""
    ;;
  *)
    echo "Unsupported arch [${ARCH}]"
    exit 1
    ;;
esac

# Determine distro for config.yaml
case "${OS}" in
  centos-*|fedora-*|opensuse-*)
    DISTRO=centos
    ;;
  debian-*|ubuntu-*)
     DISTRO=debian
    ;;
  *)
    echo "Unsupported distro [${OS}]"
    exit 1
    ;;
esac

cat > provisioner/docker/${CONFIG} <<-__EOT__
docker:
        memory_limit: "${MEM:-4g}"
        image: "bigtop/puppet:3.2.1-${OS_WO_ARCH}"

repo: http://repos.bigtop.apache.org/releases/3.2.1/${OS_FOR_URL}
distro: ${DISTRO}
components: [${DEPLOY_COMPONENTS}]
enable_local_repo: false
smoke_test_components: [${TEST_COMPONENTS}]
__EOT__

cat provisioner/docker/${CONFIG}

./gradlew docker-provisioner-destroy

./gradlew    -Pconfig=${CONFIG}    -POS=${OS_WO_ARCH}    -Pnum_instances=3    -
Pprefix=3.2.1 -Prun_smoke_tests docker-provisioner

./gradlew docker-provisioner-destroy
```