Приватний заклад вищої освіти
Одеський технологічний університет «ШАГ»
Кафедра інформаційних технологій та фундаментальної підготовки

випускна кваліфікаційна робота бакалавра

«ІНФОРМАЦІЙНА СИСТЕМА КЕРУВАННЯ ЕЛЕКТРОННИМИ ДОКУМЕНТАМИ ТА ПУБЛІКАЦІЯМИ (за прикладом сервісу Yakaboo.ua) (адмін)»

на здобуття бакалаврського рівня вищої освіти
зі спеціальності «122 Комп'ютерні науки»

Виконавці проєкту

№	П.І.Б.	Група
1	Амікішиєв Е. Н.	КН-А-191
2	Желіховський А. В.	КН-П-192
3	Мануйлов М. О.	КН-П-192
4	Омельницький М. М.	КН-П-192
5	Горищак К. С.	КН-Д-192

Автор звіту

_____Амікішиєв Ельмір Намікович

Одеса – 2023

АНОТАЦІЯ

УДК 004.9
А-62

Амікішиєв Е. Н. Інформаційна система керування електронними документами та публікаціями : Автореферат випускної кваліфікаційної роботи на здобуття бакалаврського рівня вищої освіти зі спеціальності «122 Комп'ютерні науки». - Одеса: ОТУШ, 2023 - 13 с.

Робота присвячена дослідженню, розробці, тестування і впровадження методології DevOps в систему керування електронними документами та публікаціями, шляхом створення власного інтернет-магазина Historium.

Мета інтернет-магазину Historium — привабити користувача до придбання електронного контенту та користування самим контентом у будь-якому місці, навіть під час мобільності. Система обґрунтовується існуючим інтернет-магазином yakaboo.com, але з власним інтерфейсом користувача і покращенні функціональні можливості системи. Основний контент у системі — електронні книги будь-якого жанру.

Функція самої системи містить в собі реєстрацію та авторизації акаунта, можливість зробити замовлення товара на сайті, отримання інформації щодо історії замовлень, а також адаптація відображення веб-сайта у стаціонарних та мобільних пристроїв. Основними задачами DevOps-інженера у цьому проєкті була підтримка і забезпечення цілодобового місця розміщення інформації в файловому сховищі хмарного сервісу, а також використання конвеєра Jenkins за допомогою налаштування та роботи з програмним забезпеченням Docker.

У якості мети дослідження є наступні завдання:

- Огляд інформаційної системи щодо її швидкості, наявності, масштабування, прогнозованості та її стійкості.
- Визначити програмні засоби (інтерфейси API), які дозволять отримувати інформацію з бази даних.
- Перевірка досягнутих результатів щодо стійкості системи, здійснюється шляхом реалізації результатів в окремому тестовому середовищі, що знаходиться поза межею потужностей і видимості самого проєкта.
- Збільшити частоту випуску оновлень (релізів) проєкта, шляхом використання основою DevOps (development & operations) в самій команді проєкта.

При складанні звіту було використано 16 посилань на електронні джерела інформації.

ЗМІСТ

ВСТУП

Цифрові технології з кожним днем пронизують майже всі сфери людського життя. Неможливо було уявити раніше 21 століття, що майже кожна людина буде мати змогу читати унікальні книги авторів усього світу.

Не всі люди мають змогу знайти необхідний матеріал у паперовому варіанті в своїх країнах. Навіть якщо знаходили, то пропонувались ціни, в залежності від процесу виготовлення, цінності матеріала та кількості тиражу на весь світ.

Завдяки технологічному прогресу, кожна людина має можливість придбати не лише книгу, а інші види писемності, завдяки наявності Інтернету, цифровізації паперових матеріалів у електронний формат, а також набагато краще сприяє торгівля.

Одним із ключових аспектів цифрової трансформації є розробка та впровадження інформаційних систем керування електронними документами та публікаціями. Система не лише забезпечує вільний доступ до отримання інформації, але й революціонізують способи зберігання та покращується поширення знань авторів книг серед людства усього світу.

Проєкт має відносини з використанням DevOps — це набір практик, що поєднує розробку програмного забезпечення з його обслуговуванням і експлуатацією.[1]

DevOps не тільки вважається лише набором практик. Це методологія розв'язань проблем взаємодії між IT-фахівцями у команді(-ах) над проєктом(-ами) впродовж життєвого циклу, а саме, підвищення швидкості і розгортання програмних забезпечень. Її також вважають інструментом, ідеологією, філософією культури і мостом між командами розробки та експлуатації.

DevOps має свої основні принципи, а саме:

- автоматизація процесів розробки;
- використання інструментів для конфігурації та розгортання ПЗ;
- забезпечення неперервної інтеграції та неперервної доставки (CI/CD) програмного коду;
- сумісну роботу команди над проєктом.

Метою використання методології DevOps є швидке впровадження функцій та змін в програмне забезпечення, зниження виникнення помилок та ризиків, покращення стабільності та надійності системи самого продукта, а також зменшення часу циклу розробки і більш ефективне використання ресурсів проєкта.

Методологія передбачає збільшення комунікації, взаєморозуміння між IT-фахівцями в самій команді.

Наступним кроком є подання детального технічного завдання до проєкту.

У рамках цього звіту розглядаються клієнтська і серверна частина, що являє собою веб-сайтом в Інтернеті, створений для користувачів, які використовують стаціонарні та мобільні пристрої.

Звіт складається з трьох розділів, в яких описується характеристики частини розробки, етапи розробки та зміни у IT-проєкті фахівцем DevOps, а також отримання результату після розгортання та тестування продукту.

Загальний висновок усіх учасників цього проєкту є сукупністю звітів щодо результату досягнення виконання власних частин проєкта.

ТЕХНІЧНЕ ЗАВДАННЯ ДО ПРОЄКТУ

Загальний опис.
Пошук інструментів та задоволення потреб для команди розробки і подальшої роботи програмного забезпечення (ПЗ) інформаційної системи керування електронними документами та публікаціями.
Під задоволенням потреб для команди розробки, мається на увазі, створення основного хмарного сервера з віртуалізацією, що дозволить використовувати необхідні пакети ПЗ у середовищі Ubuntu Server 22.04.
Для цього, треба встановити та налаштувати ПЗ nginx, git, Node.js, npm, Docker, Docker Compose, let's encrypt/certbot в самому основному віртуальному сервері.
Окремо треба створити загальнодоступне віртуальне сховище даних у хмарному сервісі, що дозволить команді розробникам отримувати інформацію стосовно стану і вмісту самого сховища.
Команда розробників надсилатимуть файли до віртуального сховища у хмарному сервісі, щоб мали змогу отримувати інформацію конкретного файла, як посилання.
Після цих дій, за допомогою інструмента Docker організувати розгортання та складання контейнерів на основному віртуальному сервері.
Далі, потрібно створити другий хмарний віртуальний сервер для встановлення та налаштування інструмента Jenkins, для організації неперервної інтеграції та неперервної доставки (CI/CD) збірки кода з репозиторій продукта у GitHub.
Щоб мати можливість моніторити усі працюючі системи проєкта та збирати метрики, потрібно створити третій хмарний віртуальний сервер, встановити та налаштувати ПЗ Prometheus, Grafana, Node Exporter, Jenkins Exporter.

Призначення:
ПЗ орієнтується на створення умови використання для налаштувань віртуальних хмарних серверів, а також віртуального сховища зберігання файлів.
Для інформаційної системи керування електронними документами та публікаціями створюється веб-сайт Historium, зі своїми унікальними особливостями та цільовим призначенням.

Вимоги до функціоналу:
1. Для створення хмарного віртуального сервера та сховища для зберігання файлів, має використовуватись хмарний сервіс AWS.
2. Встановлення і налаштування на основному хмарному віртуальному сервері з операційною системою Ubuntu Server 22.04 і з наступними компонентами: nginx, git, Node.js, npm, Docker, Docker Compose.
3. Завантаження репозиторіїв GitHub проєкта Historium на основний хмарний віртуальний сервер.
4. Створення хмарного віртуального сховища зберігання файлів. Налаштувати його загальнодоступним і надати ключи доступа до контейнера команді розробникам.
5. Створити Dockerfile для репозиторіїв проєкта Historium на GitHub.
6. Створення другого хмарного віртуального сервера з ОС Ubuntu Server 22.04, встановити і налаштувати роботу ПЗ Jenkins.
7. Створення, налаштування та тестування pipeline jobs для Docker в самій Jenkins.
8. Створення третього хмарного віртуального сервера з ОС Ubuntu Server 22.04, встановити і налаштувати роботу ПЗ Prometheus, Node Exporter, Grafana.

Архітектура
За клієнт-серверною архітектурою ПЗ має бути спроектована з наступними розподіленими вузлами:

- Mockup — набор графічних образів інтерфейсів системи за всіма видами áкторів, функцій та пристроїв.
- Frontend — загальний вигляд інтерфейсу клієнтського ПЗ.
- Backend — серверне ПЗ, яка забезпечує взаємодію з базою даних, збереження та оброблення поточних даних. Надає програмний інтерфейс взаємодії (API) з іншими вузлами проєкта.
  Модуль ПЗ повинна мати засіб тестування функціональності.
- DevOps — роль серверної архітектури, яка відповідає за впровадження ефективної розробки, автоматизації самої розробки, а також моніторингом і керуванням інфраструктурою проєкта Historium 24/7 годин.

Технології
За допомогою системи контролі версії ПЗ Git 2.34.1, повинно здійснюватись зберігання вихідних ПЗ.
Структура файлів та організація каталогів, повинні відповідати архітектурі ПЗ наступним чином:

- Mockup
- Frontend
- Backend
- DevOps

Для кожного з модулів рекомендується (з можливістю часткової зміни) наступні технології створення:

- Mockup — визначається особисто з командою та ментором.
- Frontend — JS, HTML, CSS або JS фреймворки, зокрема Vue.js, React.js, Node.js
- Backend — MongoDB, Node.js.
- DevOps — nginx, nginx proxy, git-all, Docker, Docker compose, Jenkins, Prometheus, Grafana, Node Exporter.

Звітність
Вихідний код повинен оформлятися у відповідності до загальноприйнятих стандартів, на зразок стандартів GNU за посиланнням: http://www.gnu.org/prep/standards/, у тому числі з додержанням вимог до коментування та документування коду.

РОЗДІЛ 1
ЗАГАЛЬНА ХАРАКТЕРИСТИКА СИСТЕМИ

Згідно з технічним завданням (ТЗ) проєкту, система спроектована за клієнт-серверною архітектурою розподілу.

Таким чином, архітектурні частини у системи були відокремлені, а саме:

- Mockup
- Frontend
- Backend
- DevOps

Частина Mockup є набором моделей взаємодій, і поведінки користувачів з інформаційною системою.

Моделі цієї частини, описують взаємодію унікальних користувачів з інформаційною системою, а також використання переходів між інтерфейсами системи. При створенні інтерфейсів були застосовані підходи UI/UX, колористика, а також можливість адаптувати підтримку десктопних та мобільних пристроїв.

Клієнтська частина системи Frontend, призначена для надання доступу користувачів до функцій системи проєкта, шляхом використання візуальної частини інтерфейса системи.

Частина Frontend виконана на базі програмних забезпечень Vue.js 3, JavaScript. Ресурси цієї частини, розміщені на репозиторії GitHub проєкта Historium за посиланням: https://github.com/historium-store/historium

Частина Backend є централізованим вузлом збереження та оброблення даних системи, протягом усього життєвого циклу інформаційної системи. До складу цієї частини є база даних проєкту та програмний інтерфейс (API) доступу до функцій самої системи, зокрема, керування товарами, авторизації.

Також частина Backend виконана на базі програмних забезпечень Node.js та СУБД MongoDB. Ресурси цієї частини, розміщені на репозиторії GitHub проєкта Historium за посиланням: https://github.com/historium-store/api

Частина DevOps забезпечує роботу усієї серверної частини системи. За допомогою цієї методології, є співпраця і комунікація команди розробників з системним адміністратором або DevOps-інженером для того, щоб забезпечити і виконати потреби від команди розробників.

Реалізація частини DevOps виконана на базі програмних забезпечень від AWS, а саме: три хмарних віртуальних машин AWS Elastic Cloud 2 зі встановленою операційною системою Ubuntu Server 22.04, одне хмарне віртуальне сховище файлів AWS S3 bucket; програмні забезпечення до операційної системи Ubuntu Server 22.04: nginx, npm, let’s encrypt/certbot, Docker, Docker Compose, Jenkins, Prometheus, Grafana, Node Exporter.

РОЗДІЛ 2
РЕАЛІЗАЦІЯ СЕРВЕРНОЇ ЧАСТИНИ

У реалізації серверної частини інформаційної системи використані наступні технології:

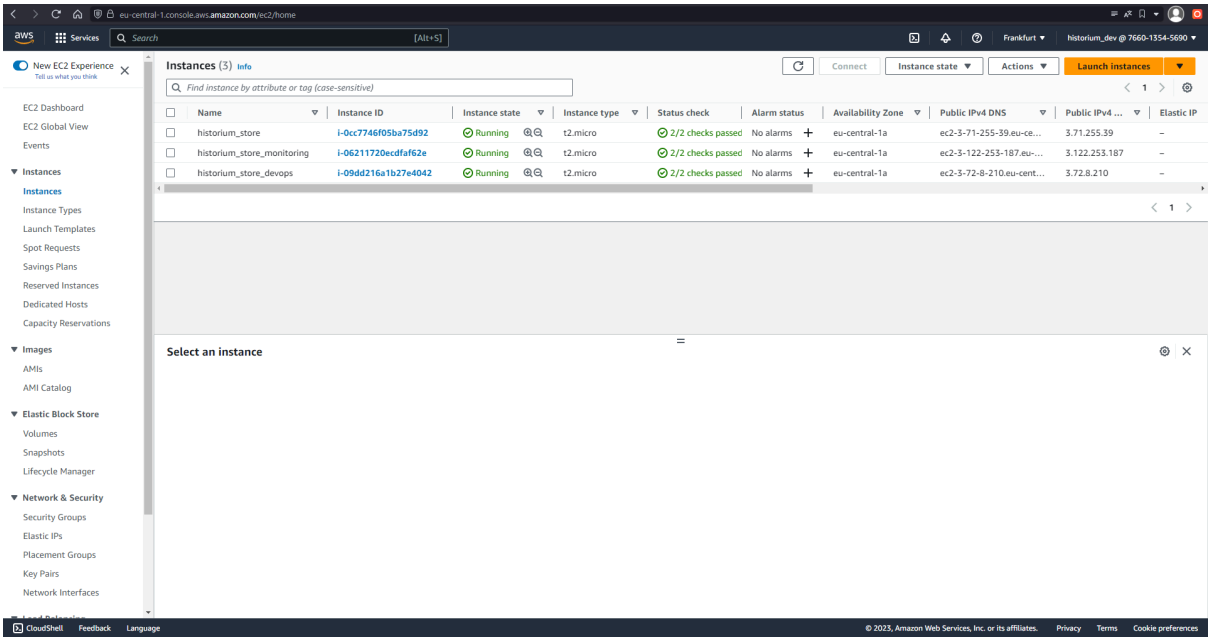
AWS (Amazon Web Services) — це платформа хмарних обчислень, яка надає широкий спектр послуг для розробки, впровадження та керування програмними додатками і інфраструктурою в хмарному середовищі.

За допомогою цієї платформи, спрощується можливість створювати віртуальну машину за допомогою веб-сервісу Amazon EC2, використання інструментами Vue.js, Node.js та інші, у хмарному середовищі, а також створення і використання контейнера зберігання файлів Amazon S3 bucket, за рахунок використання обчислювальних потужностей компанії Amazon.

AWS вважається надійним та провідним хмарним провайдером, який досі залишається актуальним на ринку. Це набагато більший проєкт, ніж інтернет-магазин Amazon. Мають свою кількість клієнтів, різноманіття технологічних продуктових рішень для використання, а також AWS є масштабованою системою. Тобто, спрощується можливість мати доступ до продуктів AWS поза межею власного фізичного пристрою, і це є основною перевагою хмарних рішень.

Початок роботи супроводжується з використанням сервісу AWS для створення хмарного сервера.

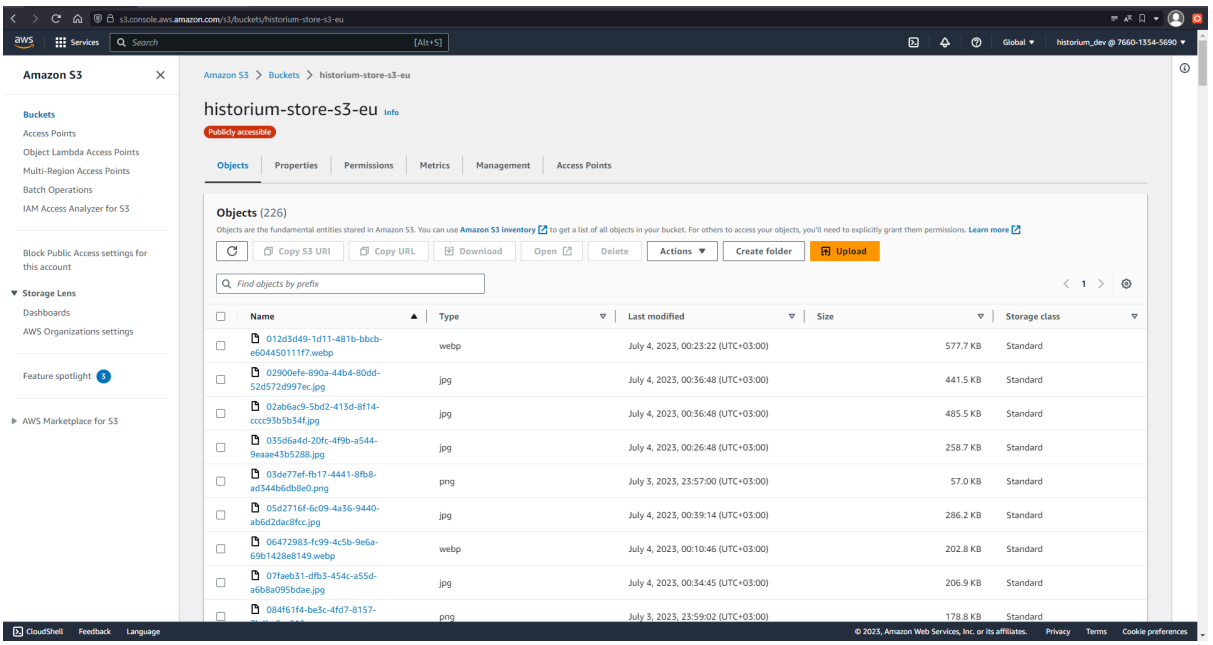
1. Для створення хмарних віртуальних серверів з ОС Ubuntu Server 22.04, а саме, віртуальних машин на базі AWS, був використаний тимчасовий безкоштовний сервіс Amazon EC2 з його розташуванням у м.Франкфурт, Німеччина. Перша хмарна віртуальна машина AWS EC2 має ім’я historium_store і вважається основною через те, що саме на цю хмарну віртуальну машину будуть встановленні та налаштовані необхідні програмні забезпечення для роботи проєкту і звернення клієнтів, за допомогою адреси historium.store, згідно з технічним завданням до проєкту.



2. Встановлення пакетів (програмних забезпечень) nginx, git-all, npm, let's encrypt/certbot, Docker, Docker Compose на основній хмарній віртуальній машині на базі Amazon EC2.

3. Копіювання 2-х репозиторій проєкта Historium з GitHub, до основної хмарної віртуальної машини Amazon EC2.

4. Створення тимчасового безкоштовного контейнера зберігання файлів Amazon S3 bucket з його розташуванням у м.Франкфурт, Німеччина.

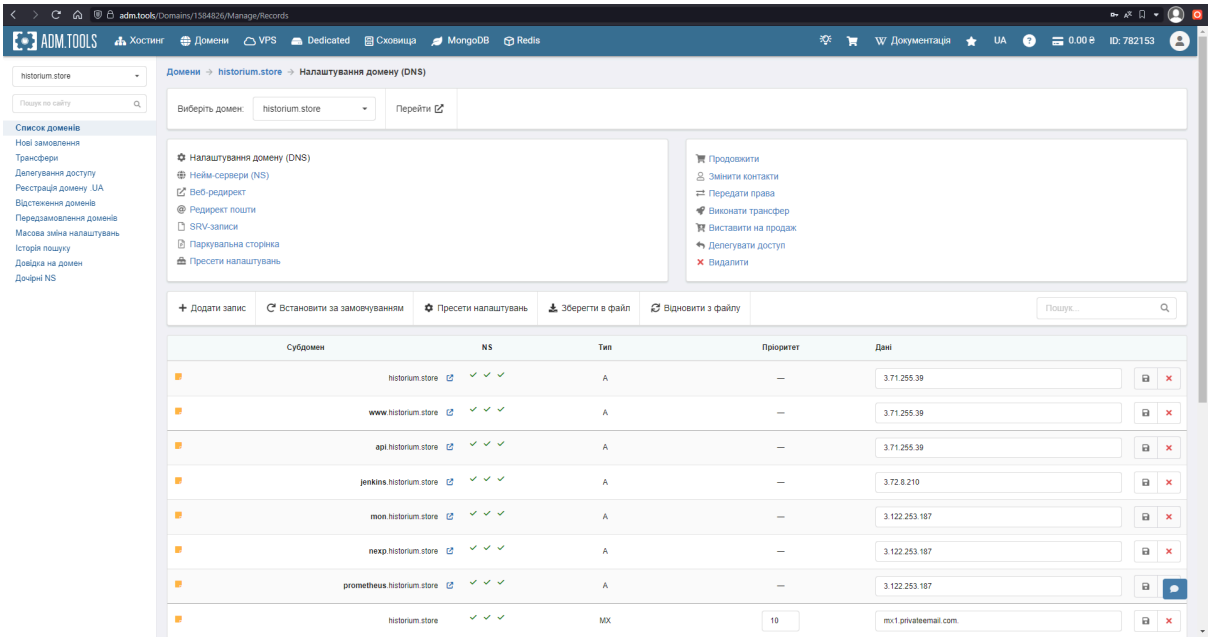


5. Зміна прав відображення контейнера зберігання файлів на загальнодоступний, за допомогою JSON коду:

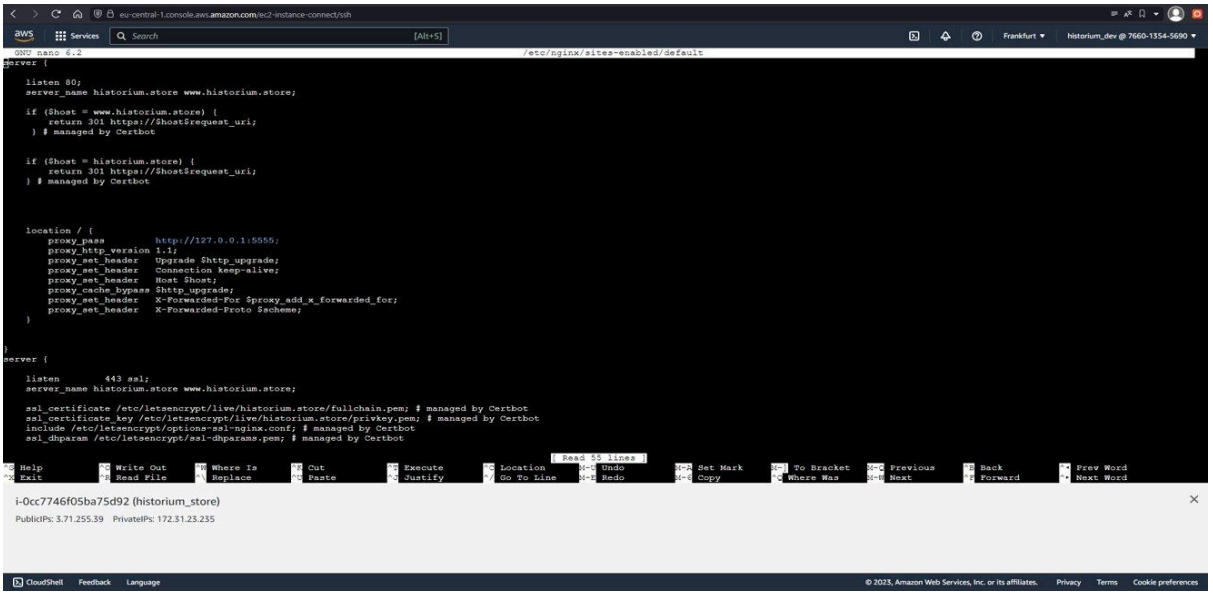
```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowPublicReadAccess",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::historium-bucket-eu/*"
    }
  ]
}
```

6. Поширення секретних ключів команді розробникам, щоб мали змогу переглядати, відправляти та отримувати файли з контейнера зберігання файлів, як посилання на сайт.

7. Придбання доменного імені historium.store і прив’язка імені до IP адреси основного хмарного віртуального серверу Amazon EC2 за допомогою використання адміністраторської панелі adm.tools від хостингу ukraine.com.ua.



8. Налаштування на основному хмарному віртуальному сервері Amazon EC2 historium_store зворотного проксі-серверу на базі nginx, а також налаштування let’s encrypt/certbot для отримання сертифікатів на доменні імена historium.store для архітектурної частини Frontend і api.historium.store для архітектурної частини Backend.



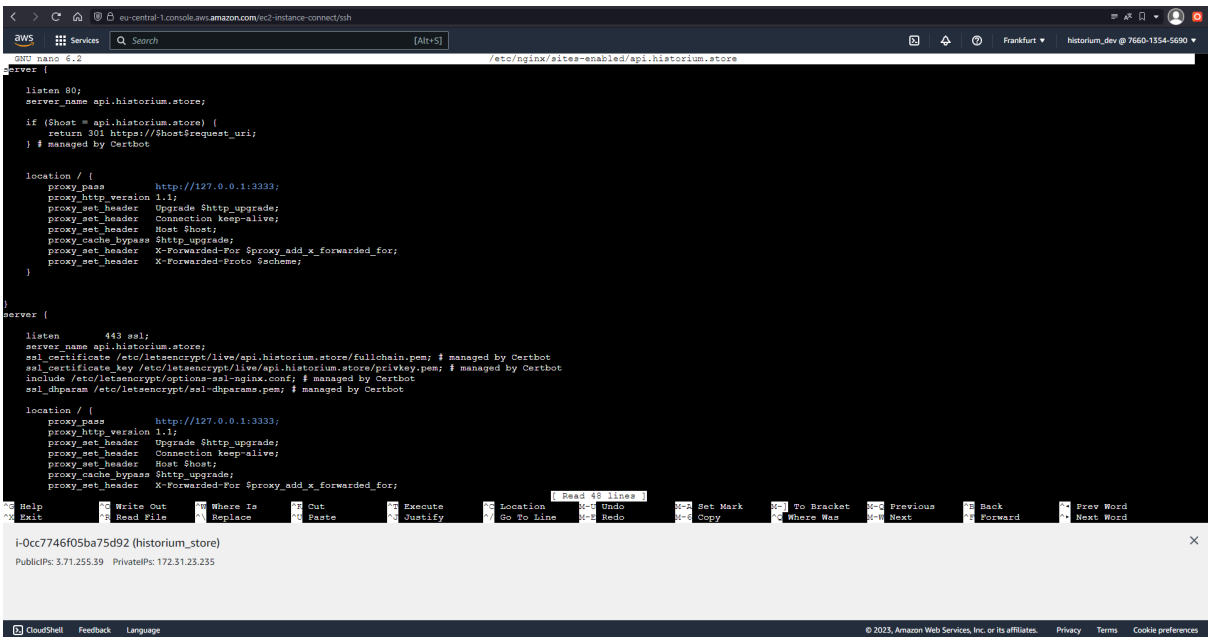
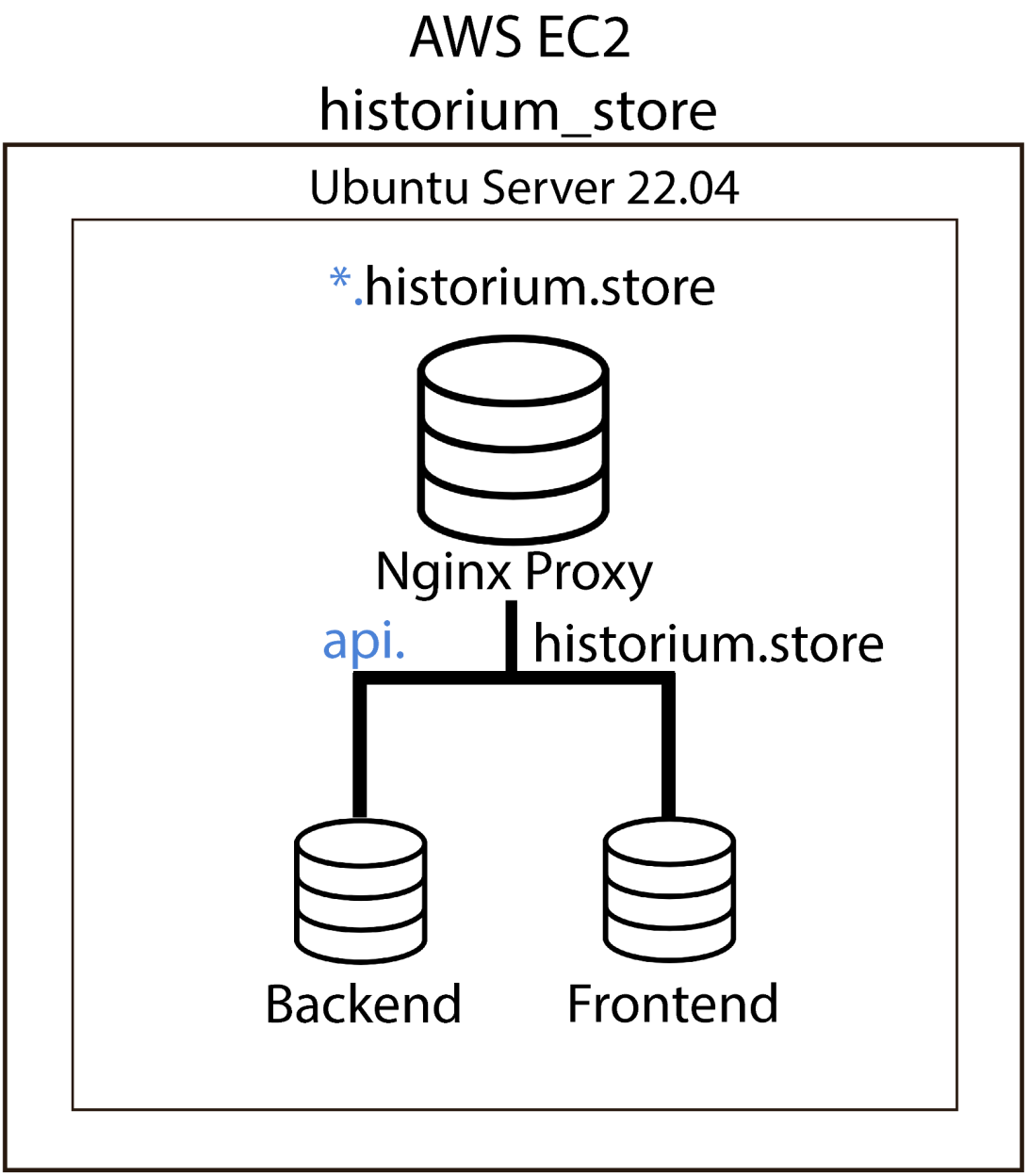


Схема роботи Nginx proxy на 1-й хмарної віртуальної машини:



Конфігураційний файл nginx для архітектурної частини Frontend AWS historium_store:

```
server {

    listen 80;
    server_name historium.store www.historium.store;

    if ($host = api.historium.store) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

    location / {
        proxy_pass         http://127.0.0.1:5555;
        proxy_http_version 1.1;
        proxy_set_header   Upgrade $http_upgrade;
        proxy_set_header   Connection keep-alive;
        proxy_set_header   Host $host;
        proxy_cache_bypass $http_upgrade;
        proxy_set_header   X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header   X-Forwarded-Proto $scheme;
    }


}

server {

    listen        443 ssl;
    server_name historium.store www.historium.store;

    ssl_certificate /etc/letsencrypt/live/historium.store/fullchain.pem; # managed by
Certbot
    ssl_certificate_key /etc/letsencrypt/live/historium.store/privkey.pem; # managed
by Certbot
    include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot
    location / {
        proxy_pass         http://127.0.0.1:5555;
        proxy_http_version 1.1;
        proxy_set_header   Upgrade $http_upgrade;
        proxy_set_header   Connection keep-alive;
        proxy_set_header   Host $host;
        proxy_cache_bypass $http_upgrade;
        proxy_set_header   X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header   X-Forwarded-Proto $scheme;
    }

}
```

Конфігураційний файл nginx для архітектурної частини Backend historium_store:

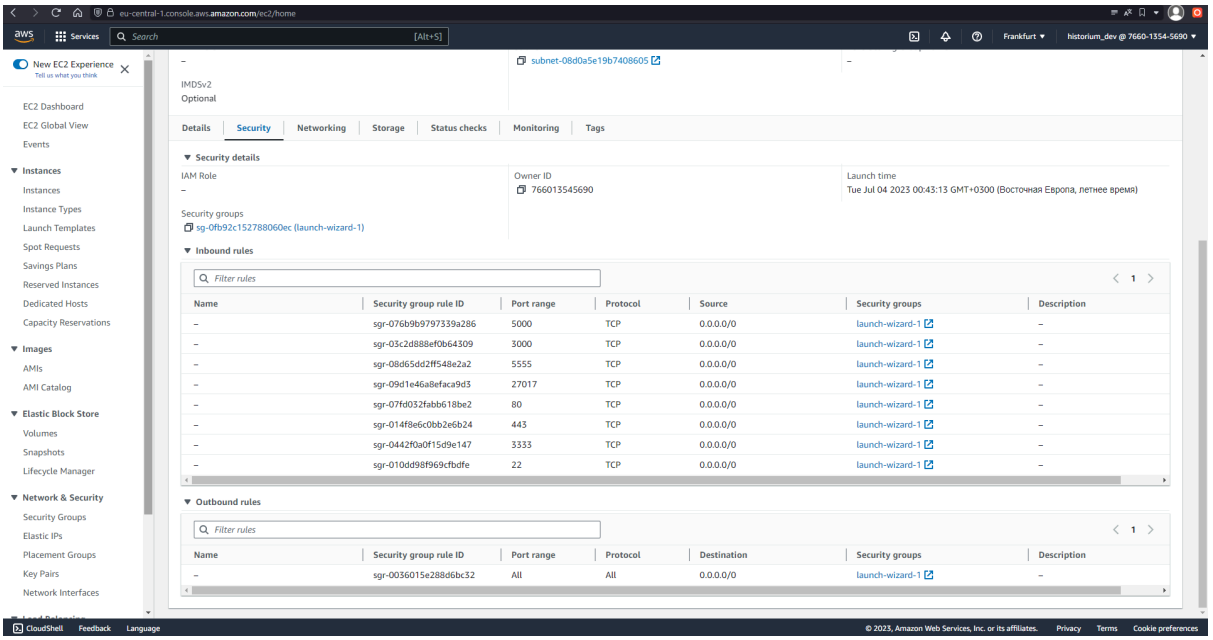
```
server {
    if ($host = api.historium.store) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

    listen 80;
    server_name api.historium.store;

    location / {
        proxy_pass         http://127.0.0.1:3333;
        proxy_http_version 1.1;
        proxy_set_header   Upgrade $http_upgrade;
        proxy_set_header   Connection keep-alive;
        proxy_set_header   Host $host;
        proxy_cache_bypass $http_upgrade;
        proxy_set_header   X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header   X-Forwarded-Proto $scheme;
    }

}
server {

    listen        443 ssl;
    server_name api.historium.store;
    ssl_certificate /etc/letsencrypt/live/api.historium.store/fullchain.pem; # managed
by Certbot
    ssl_certificate_key /etc/letsencrypt/live/api.historium.store/privkey.pem; #
managed by Certbot
    include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot

    location / {
        proxy_pass         http://127.0.0.1:3333;
        proxy_http_version 1.1;
        proxy_set_header   Upgrade $http_upgrade;
        proxy_set_header   Connection keep-alive;
        proxy_set_header   Host $host;
        proxy_cache_bypass $http_upgrade;
        proxy_set_header   X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header   X-Forwarded-Proto $scheme;
    }
}
```

9. Встановлення портів 3000, 3333, 5000, 5555 та 27017 на вхід.

Порти 3000 та 3333 встановлені на вхід у налаштуваннях AWS EC2 historium_store для того, щоб за зверненням до адреси хмарної віртуальної машини historium_store відображалися елементи архітектурної частини Backend (API).

Порти 5000 та 5555 встановлені на вхід у налаштуваннях AWS EC2 historium_store для того, щоб за зверненням до адреси хмарної віртуальної машини historium_store відображалися елементи архітектурної частини Frontend.

Порт 27017 встановлений на вхід у налаштуваннях AWS EC2 historium_store для того, щоб за зверненням до адреси хмарної віртуальної машини historium_store була можливість користуватись базою даних на базі mongoDB в рамках роботи веб-сайта проєкта Historium.

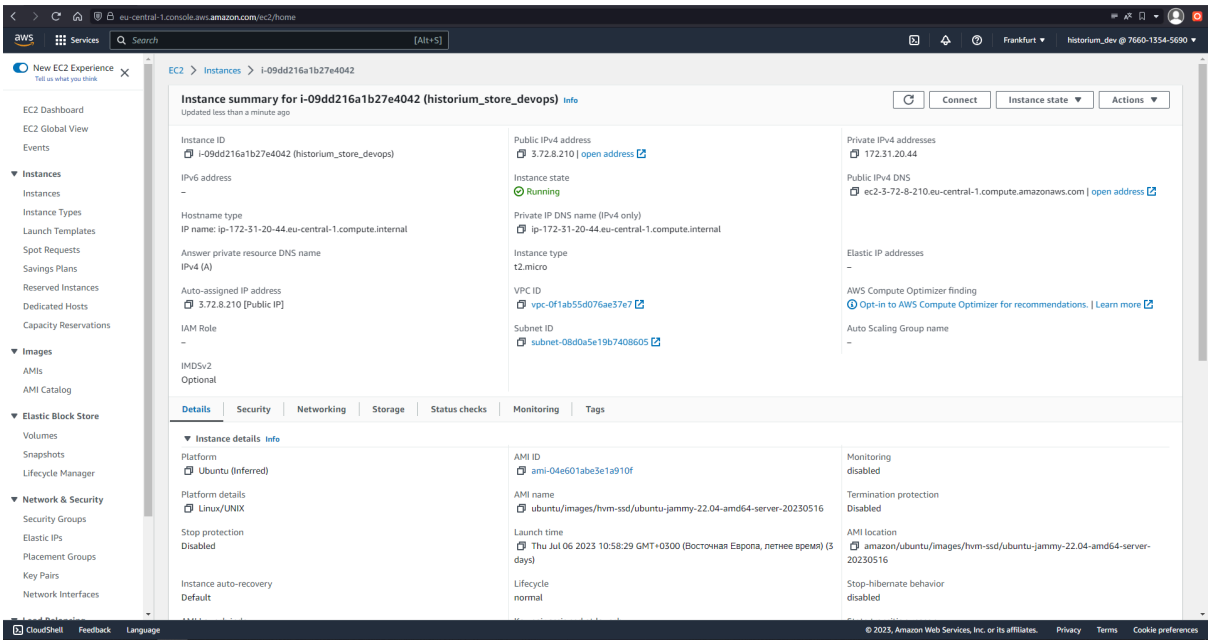


10. Створення 2 образів Dockerfile для частини Frontend і Backend на основному хмарному віртуальній машині, а також docker-compose.yml для автоматичного розгортання і активації контейнерів.

FROM node:alpine	FROM node:alpine
WORKDIR /historium	WORKDIR /api
EXPOSE 5000	EXPOSE 3000
COPY package*.json .	COPY package*.json .
RUN npm install	RUN npm install
COPY . .	COPY . .
CMD ["npm", "run", "dev"]	CMD ["npm", "run", "start"]

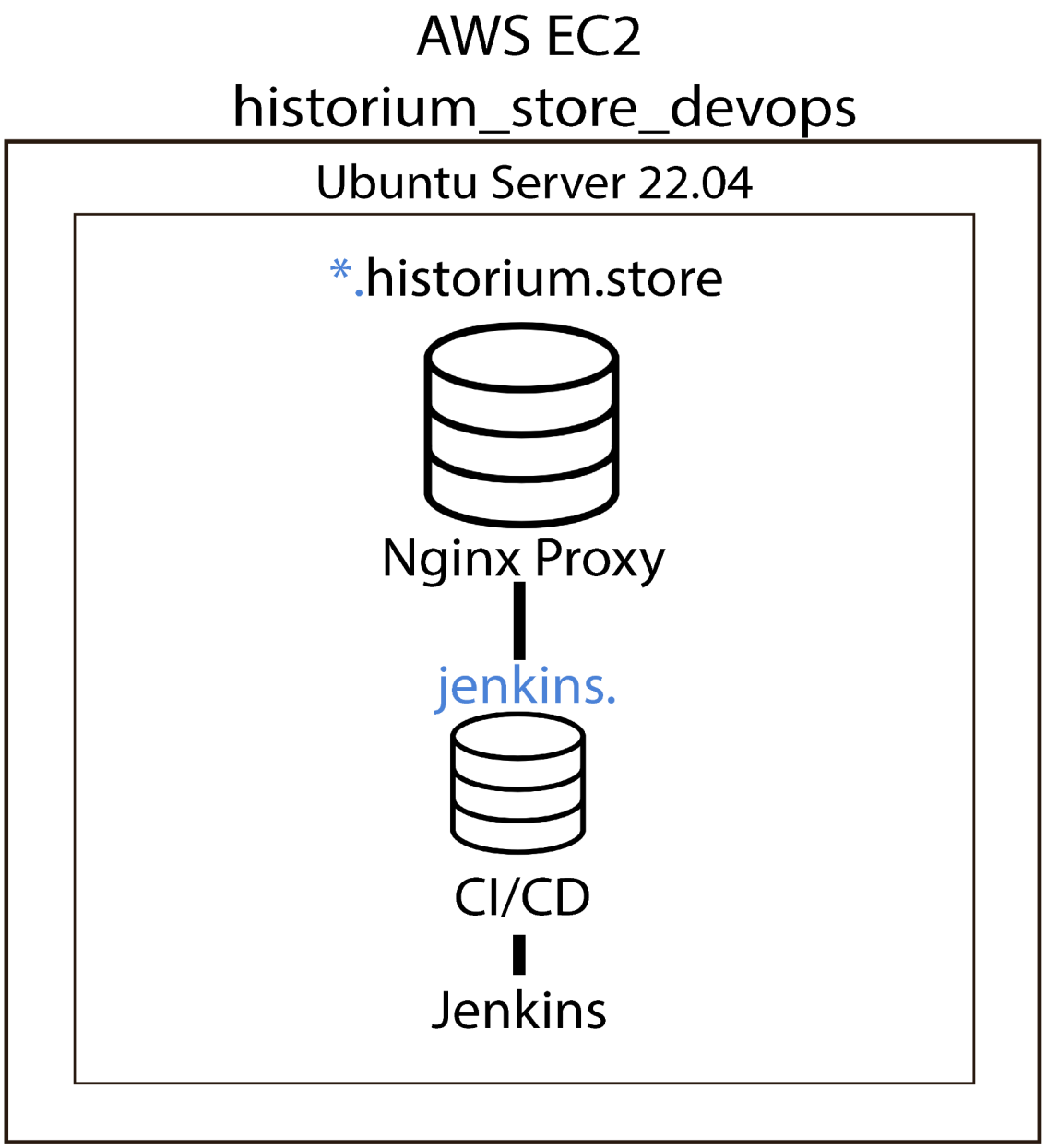
```
version: '3'

services:
  frontend:
    build: ./historium
    restart: always
    ports:
      - $FRONTEND_PORTS
    environment:
      - VITE_API_URL=$API_URL
  api:
    build: ./api
    restart: always
    ports:
      - $BACKEND_PORTS
    depends_on:
      - mongo
    environment:
      - PORT=$PORT
      - SECRET=$SECRET
      - JWT_EXPIRATION=$JWT_EXPIRATION
      -
CONNECTION_STRING=mongodb://${MONGO_USERNAME}:${MONGO_PASS
WORD}@${MONGO_HOSTNAME}/?retryWrites=true&w=majority
      - EMAIL_PASSWORD=$EMAIL_PASSWORD
      - NODE_ENV=$NODE_ENV
      - S3_ACCESS_KEY=$S3_ACCESS_KEY
      - S3_SECRET_ACCESS_KEY=$S3_SECRET_ACCESS_KEY
      - S3_BUCKET_REGION=$S3_BUCKET_REGION
      - S3_BUCKET_NAME=$S3_BUCKET_NAME
      - VONAGE_API_KEY=$VONAGE_API_KEY
      - VONAGE_API_SECRET=$VONAGE_API_SECRET
      - BACKUP_DIR_PATH=$BACKUP_DIR_PATH
  mongo:
    image: mongo
    restart: always
    volumes:
      - db:/var/lib/mongodb
    environment:
      - MONGO_INITDB_ROOT_USERNAME=$MONGO_USERNAME
      - MONGO_INITDB_ROOT_PASSWORD=$MONGO_PASSWORD

volumes:
  db:
```

10. Розгортання другої віртуальної машини Amazon EC2, яка має ім’я historium_store_devops.



Після цих дій, встановлюється і налаштовується ПЗ nginx аналогічно, як з основною віртуальною машиною historium_store. Але конфігураційний файл nginx відрізняється портом 8080 і назвою самого домену jenkins.historium.store для звернення до Jenkins.

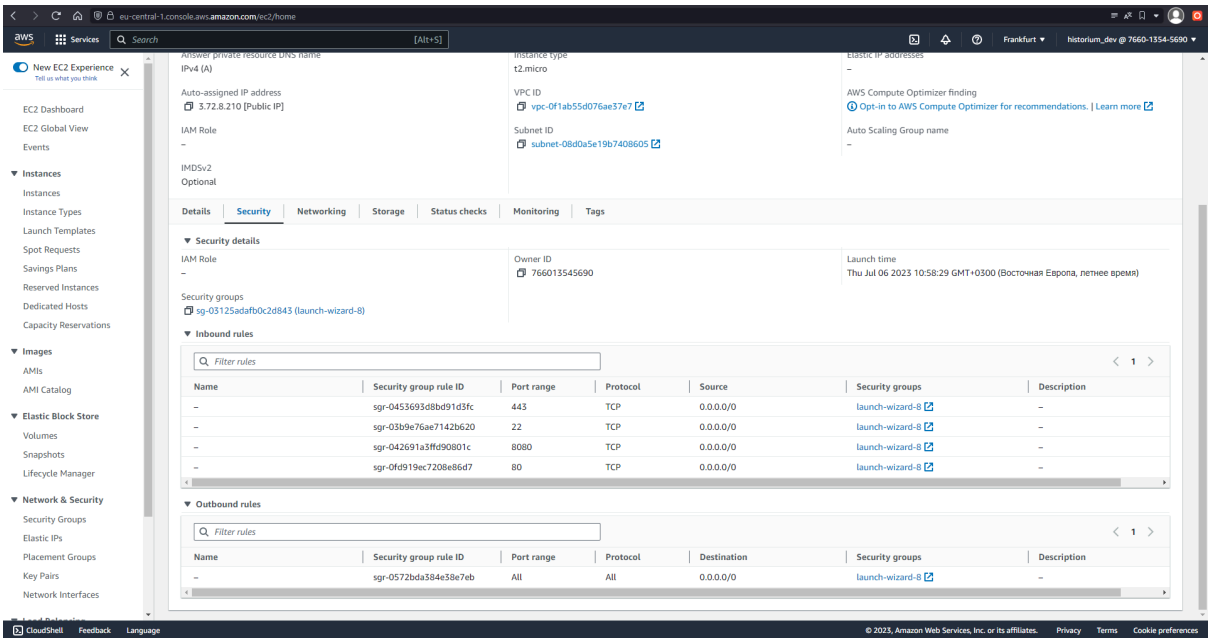
Схема роботи Nginx proxy на 2-й хмарній віртуальній машині:



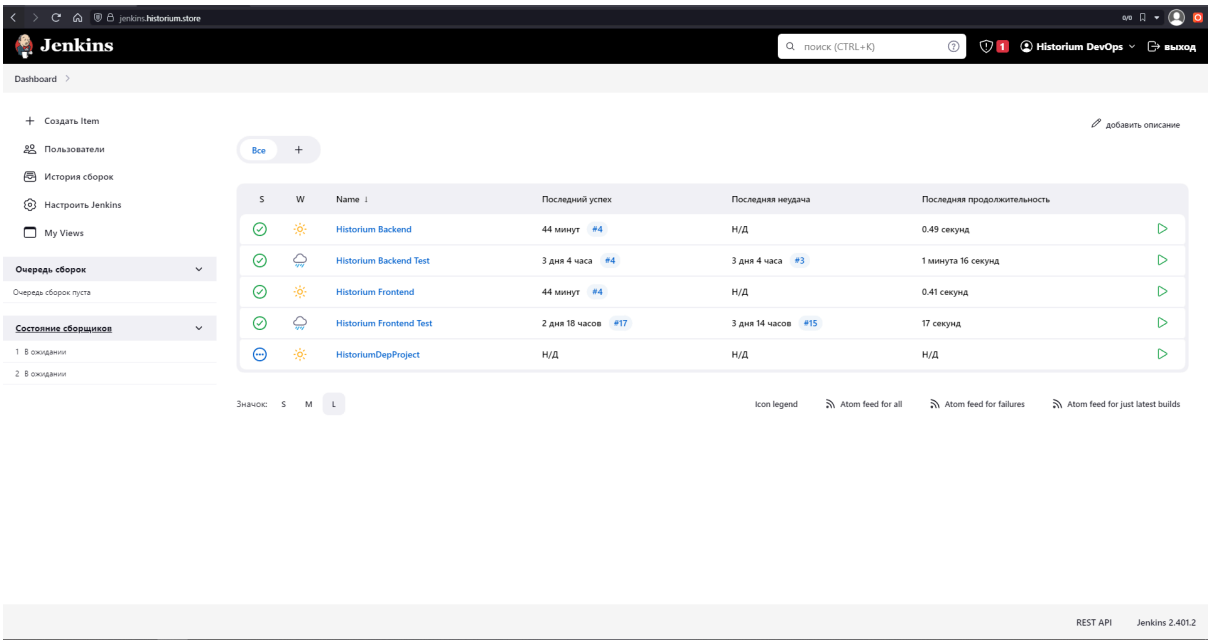
Конфігураційний файл nginx для архітектурної частини DevOps у хмарній віртуальній машині historium_store_devops:

```
server {

    listen 80;
    server_name jenkins.historium.store;

   if ($host = jenkins.historium.store) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

    location / {
        proxy_pass         http://127.0.0.1:8080;
        proxy_http_version 1.1;
        proxy_set_header   Upgrade $http_upgrade;
        proxy_set_header   Connection keep-alive;
        proxy_set_header   Host $host;
        proxy_cache_bypass $http_upgrade;
        proxy_set_header   X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header   X-Forwarded-Proto $scheme;
    }

}
server {

    listen 443 ssl; # managed by Certbot
    server_name jenkins.historium.store;

    ssl_certificate /etc/letsencrypt/live/jenkins.historium.store/fullchain.pem; #
managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/jenkins.historium.store/privkey.pem; #
managed by Certbot
    include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot

    location / {
        proxy_pass         http://127.0.0.1:8080;
        proxy_http_version 1.1;
        proxy_set_header   Upgrade $http_upgrade;
        proxy_set_header   Connection keep-alive;
        proxy_set_header   Host $host;
        proxy_cache_bypass $http_upgrade;
        proxy_set_header   X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header   X-Forwarded-Proto $scheme;
    }

}
```

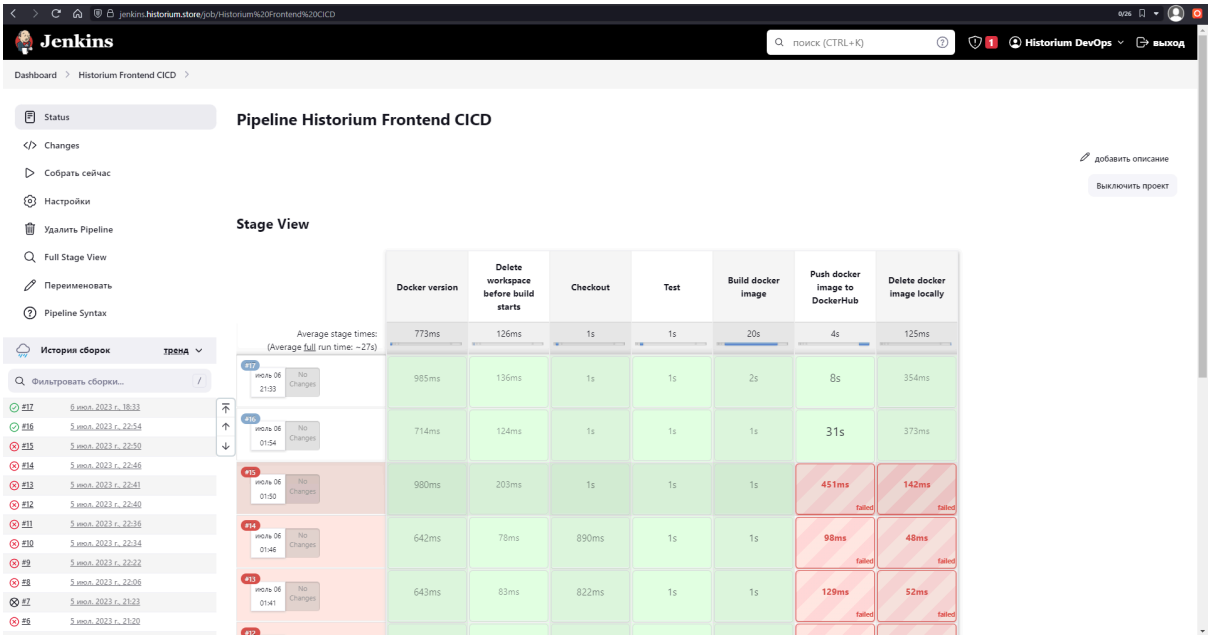
Порт 8080 встановлений на вхід у налаштуваннях AWS EC2 historium_store_devops для того, щоб мати змогу побачити сайт Jenkins за зверненням до адреси хмарної віртуальної машини historium_store_devops.

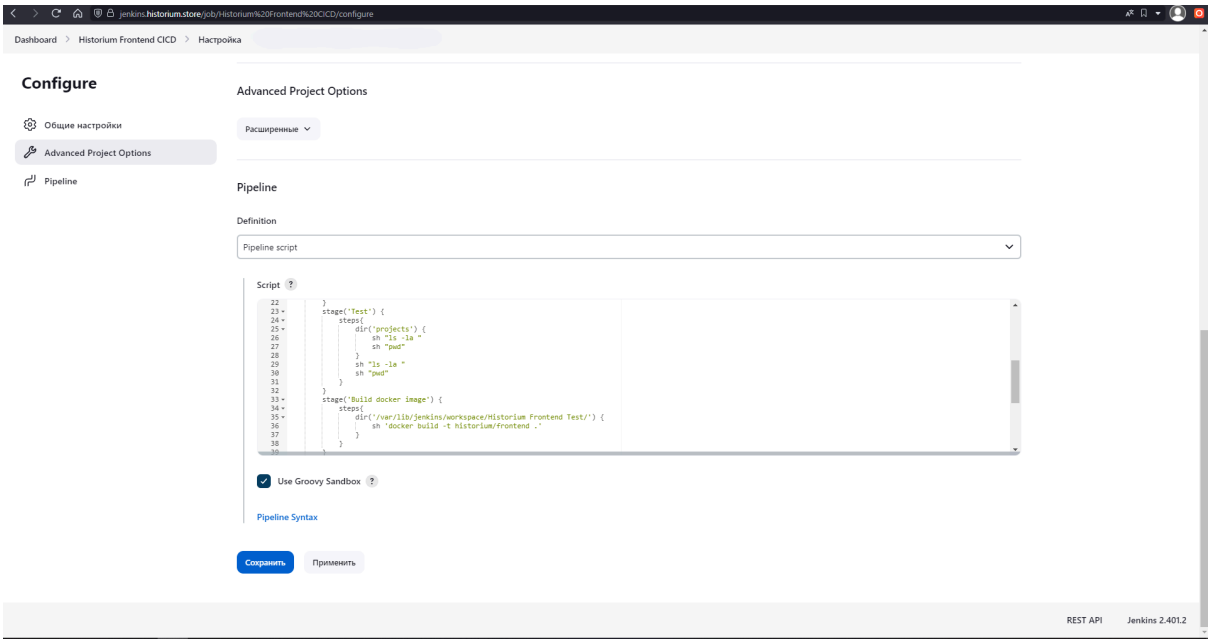


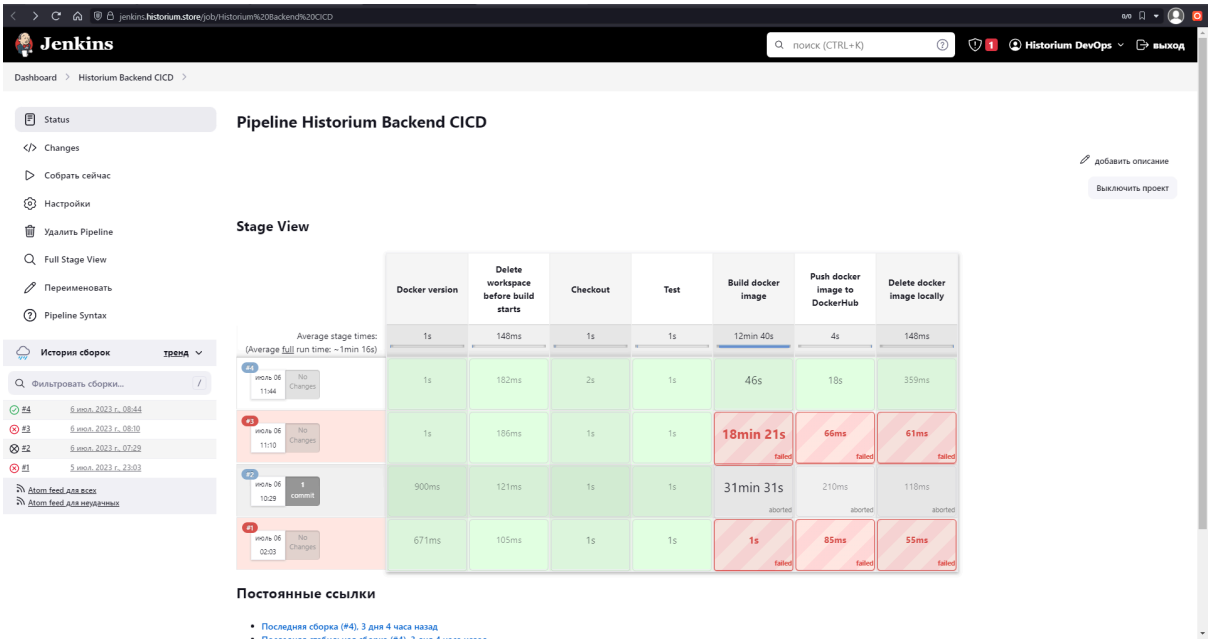
11. Встановлення та налаштування Jenkins.

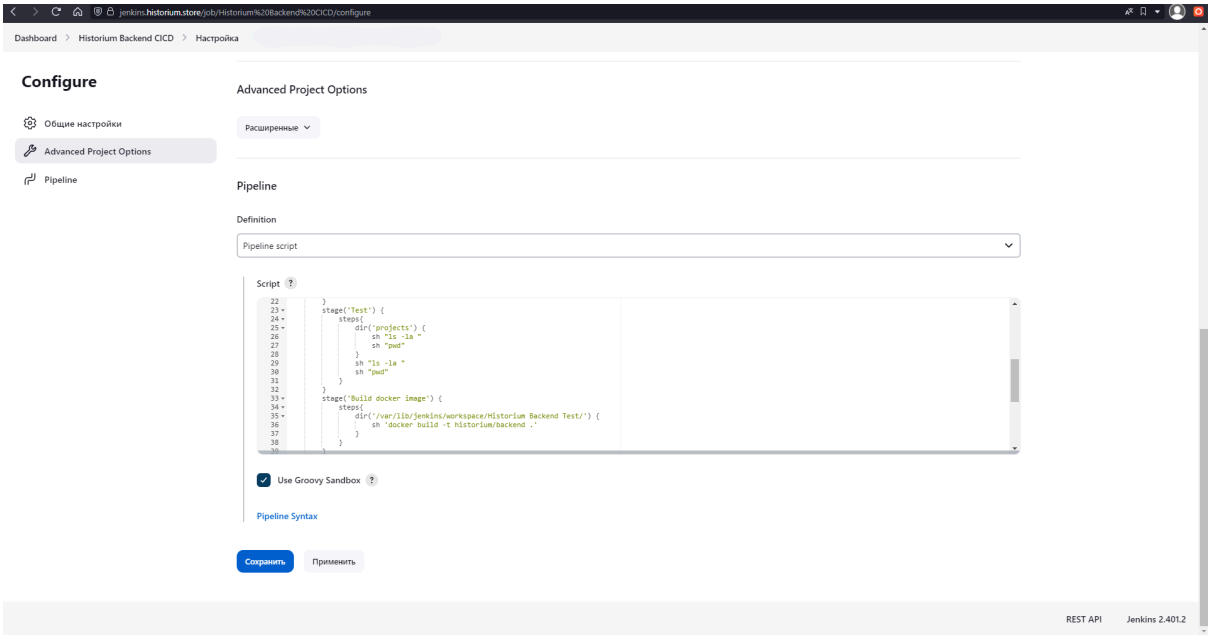


12. Створення та налаштування 2 pipeline jobs Historium Frontend CICD і Historium Backend CICD в Jenkins.

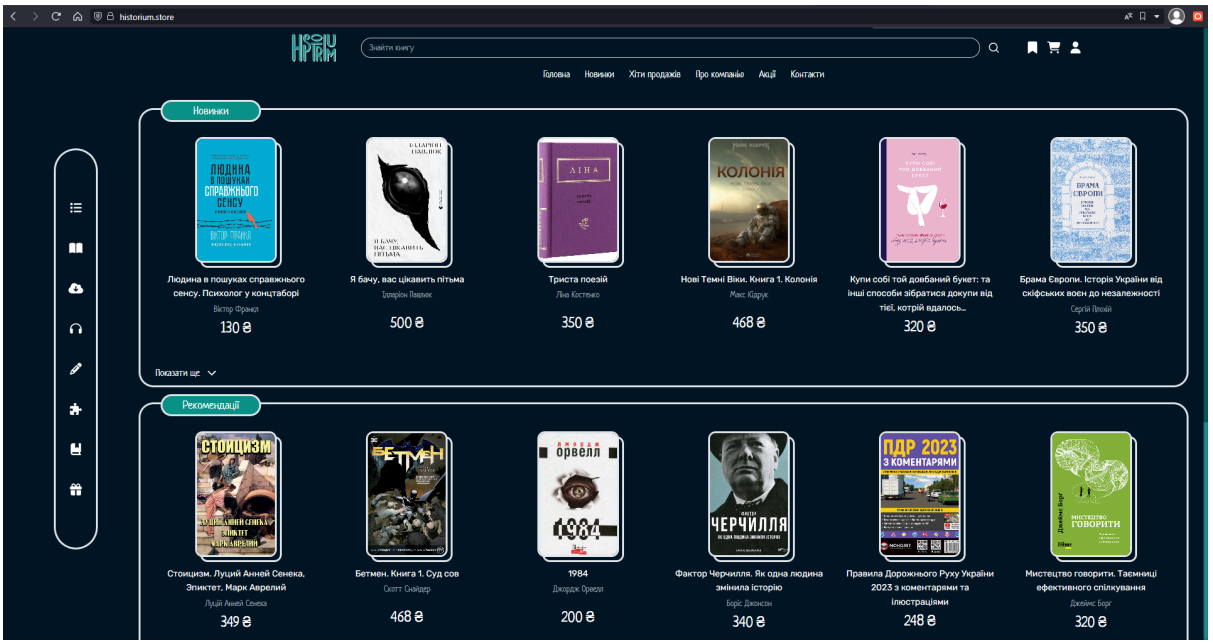


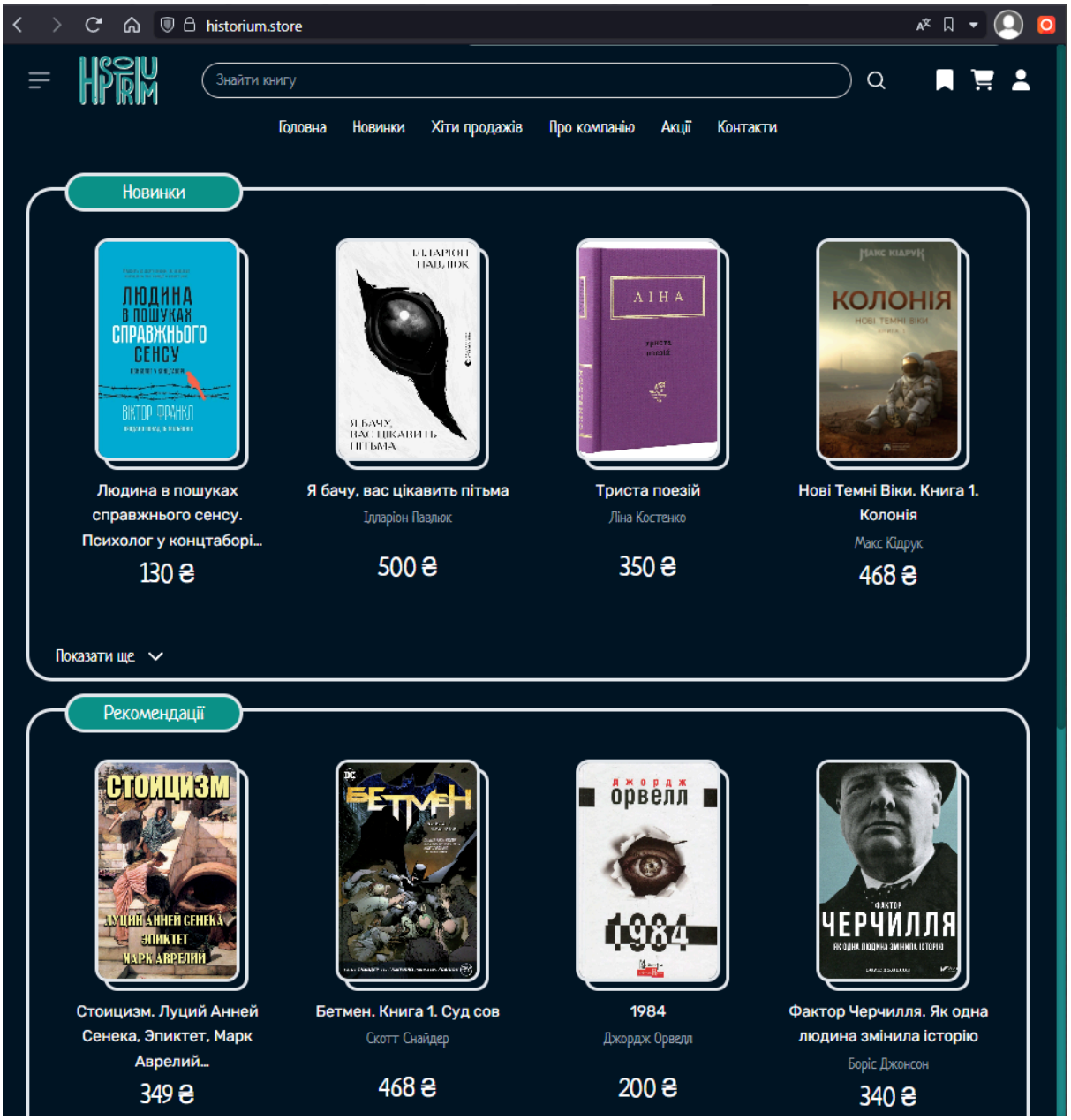


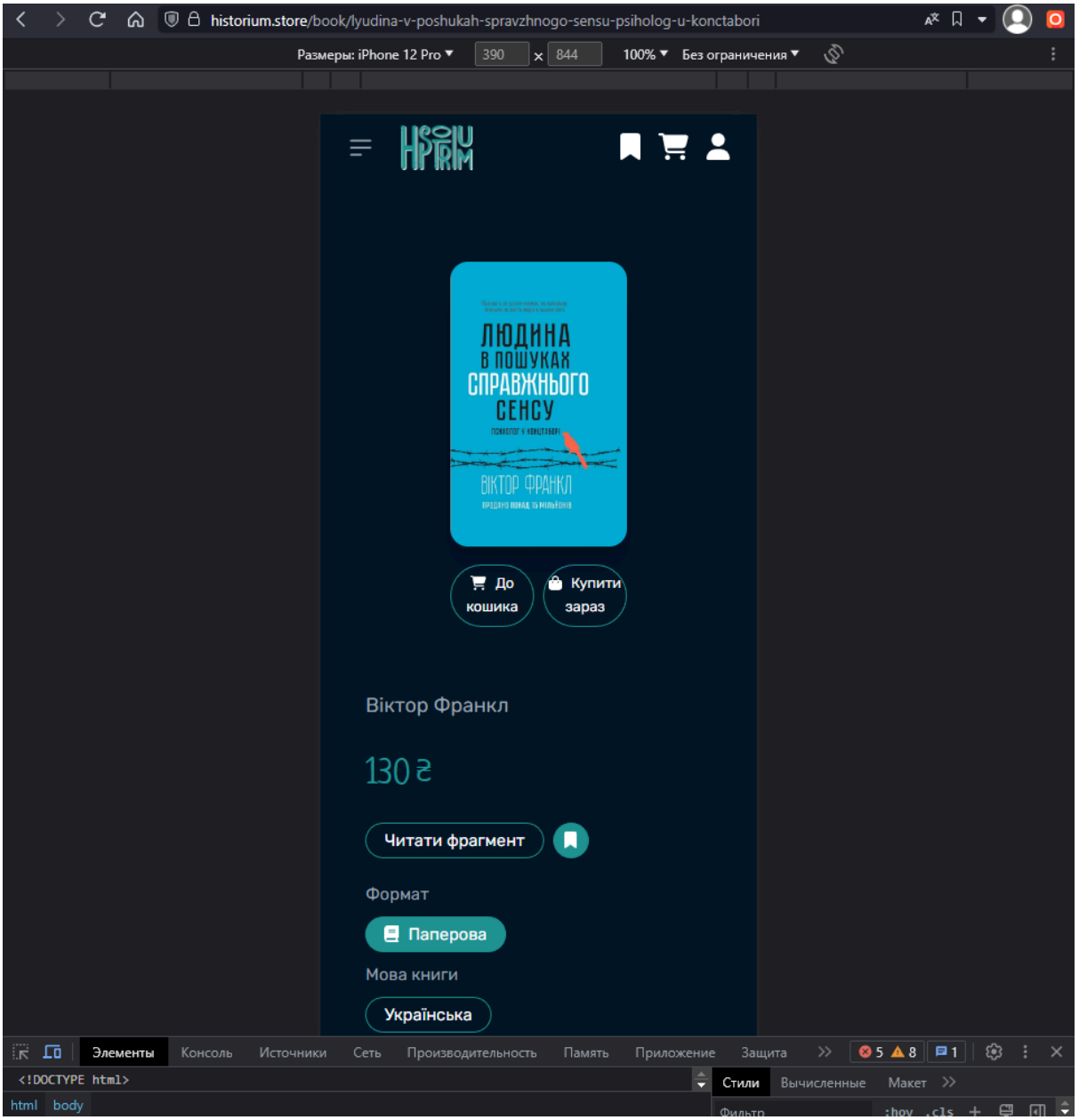

jenkins.historium.store/job/Historium%20Backend%20CICD
Jenkins
поиск (CTRL+K)
Historium DevOps
выход
Dashboard
Historium Backend CICD
Status
Changes
Собрать сейчас
Настройки
Удалить Pipeline
Full Stage View
Переименовать
Pipeline Syntax
История сборок
тренд
Фильтровать сборки...
#4 6 июл. 2023 г., 08:44
#3 6 июл. 2023 г., 08:10
#2 6 июл. 2023 г., 07:29
#1 5 июл. 2023 г., 23:03
Atom feed для всех
Atom feed для неудачных
Pipeline Historium Backend CICD
добавить описание
Выключить проект
Stage View
Docker version
Delete workspace before build starts
Checkout
Test
Build docker image
Push docker image to DockerHub
Delete docker image locally
Average stage times:
(Average full run time: ~1min 16s)
1s
148ms
1s
1s
12min 40s
4s
148ms
46s
18s
359ms
18min 21s
66ms
61ms
31min 31s
210ms
118ms
900ms
121ms
671ms
105ms
1s
85ms
55ms
Постоянные ссылки
Последняя сборка (#4), 3 дня 4 часа назад


jenkins.historium.store/job/Historium%20Backend%20CICD/configure
Dashboard
Historium Backend CICD
Настройка
Configure
Общие настройки
Advanced Project Options
Pipeline
Advanced Project Options
Расширенные
Pipeline
Definition
Pipeline script
Script
stage('Test') {
steps{
dir('projects') {
sh "ls -la "
sh "pwd"
}
sh "ls -la "
sh "pwd"
}
}
stage('Build docker image') {
steps{
dir('/var/lib/jenkins/workspace/Historium Backend Test/') {
sh 'docker build -t historium/backend .'
}
}
Use Groovy Sandbox
Pipeline Syntax
Сохранить
Применить
REST API
Jenkins 2.401.2

13. Результати розгортання та роботи проєкта Historium:




historium.store
Знайти книгу
Головна
Новинки
Хіти продажів
Про компанію
Акції
Контакти
Новинки
Людина в пошуках справжнього сенсу. Психолог у концтаборі...
130 ₴
Я бачу, вас цікавить пітьма
Ілларіон Павлюк
500 ₴
Триста поезій
Ліна Костенко
350 ₴
Нові Темні Віки. Книга 1. Колонія
Макс Кідрук
468 ₴
Показати ще
Рекомендації
Стоицизм. Луцій Анней Сенека, Епіктет, Марк Аврелій...
349 ₴
Бетмен. Книга 1. Суд сов
Скотт Снайдер
468 ₴
1984
Джордж Орвелл
200 ₴
Фактор Черчилля. Як одна людина змінила історію
Борис Джонсон
340 ₴


historium.store/book/lyudina-v-poshukah-spravzhnogo-sensu-psiholog-u-konctabori
Размеры: iPhone 12 Pro
390
844
100%
Без ограничения
ЛЮДИНА В ПОШУКАХ СПРАВЖНЬОГО СЕНСУ
ВІКТОР ФРАНКЛ
До кошика
Купити зараз
Віктор Франкл
130 ₴
Читати фрагмент
Формат
Паперова
Мова книги
Українська
Элементы
Консоль
Источники
Сеть
Производительность
Память
Приложение
Защита
<!DOCTYPE html>
html
body
Стили
Вычисленные
Макет

Тепер, так як у нас є усі необхідні умови для роботи продукту, потрібно мати можливість моніторити стан проєкту, шляхом розгортання окремих сервісів на третій хмарній віртуальній машині.

Для цього, нам потрібні сервіси Prometheus, Grafana, а також Node Exporter.

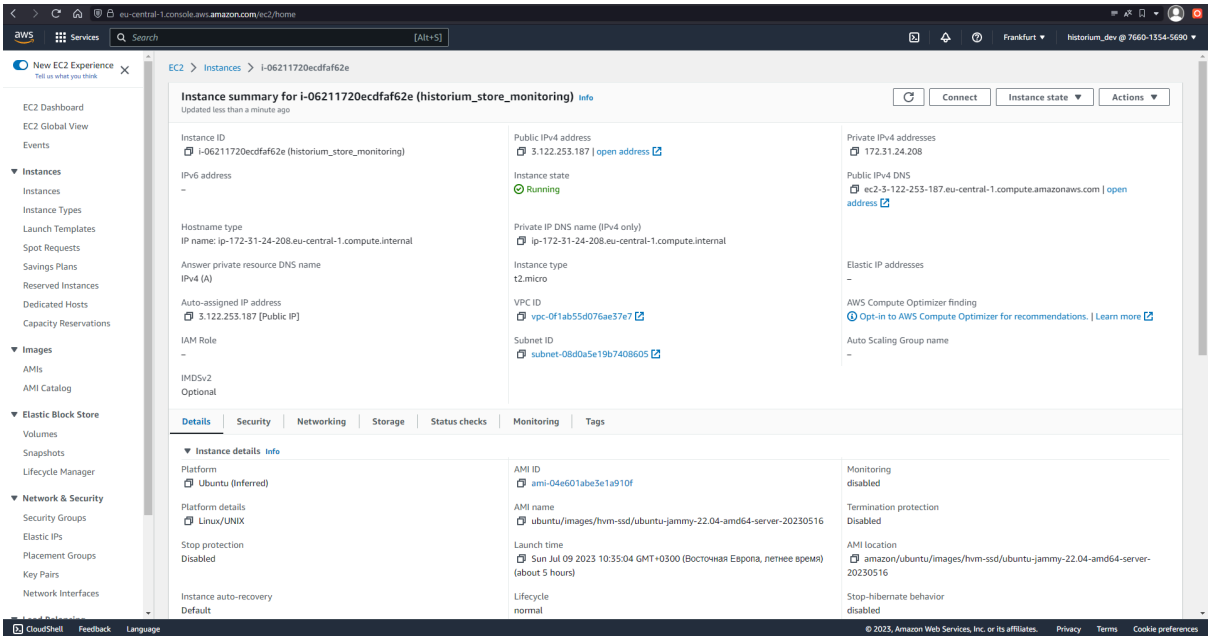
Prometheus – це популярне рішення з відкритим вихідним кодом, призначене для моніторингу та відправлення попереджень та оптимізоване для контейнерних середовищ. [2]

Grafana – безкоштовна система візуалізації даних, орієнтована на дані систем IT-моніторингу. [3]

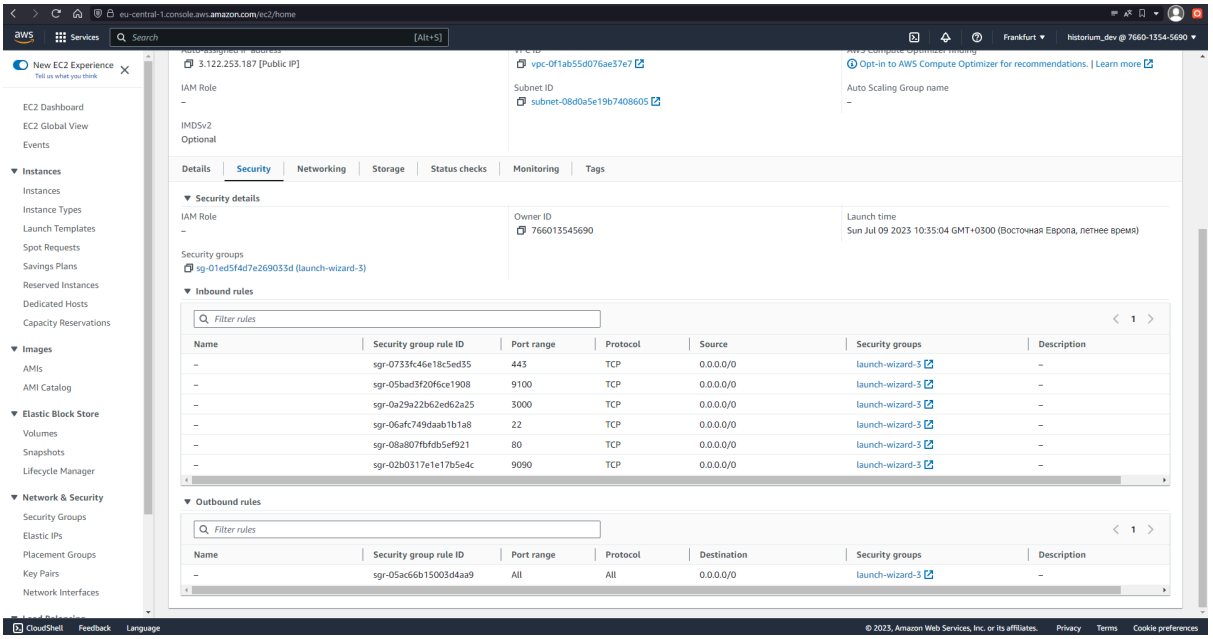
Node Exporter – агент збору метрик для операційних систем на базі ядра Linux, розроблений для використання з системами моніторингу, такими як Prometheus. Він збирає безліч метриків системи, таких як використання процесора, пам'яті, мережі, дисків та інших. [4]

Продовжується супроводження виконання робіт з використанням сервісу AWS.

14. Створення третьої окремої хмарної віртуальної машини на базі Amazon EC2, яка має ім’я historium_store_monitoring для моніторінгу систем проєкта і налаштування портів 3000, 9090, 9100 на вхід.

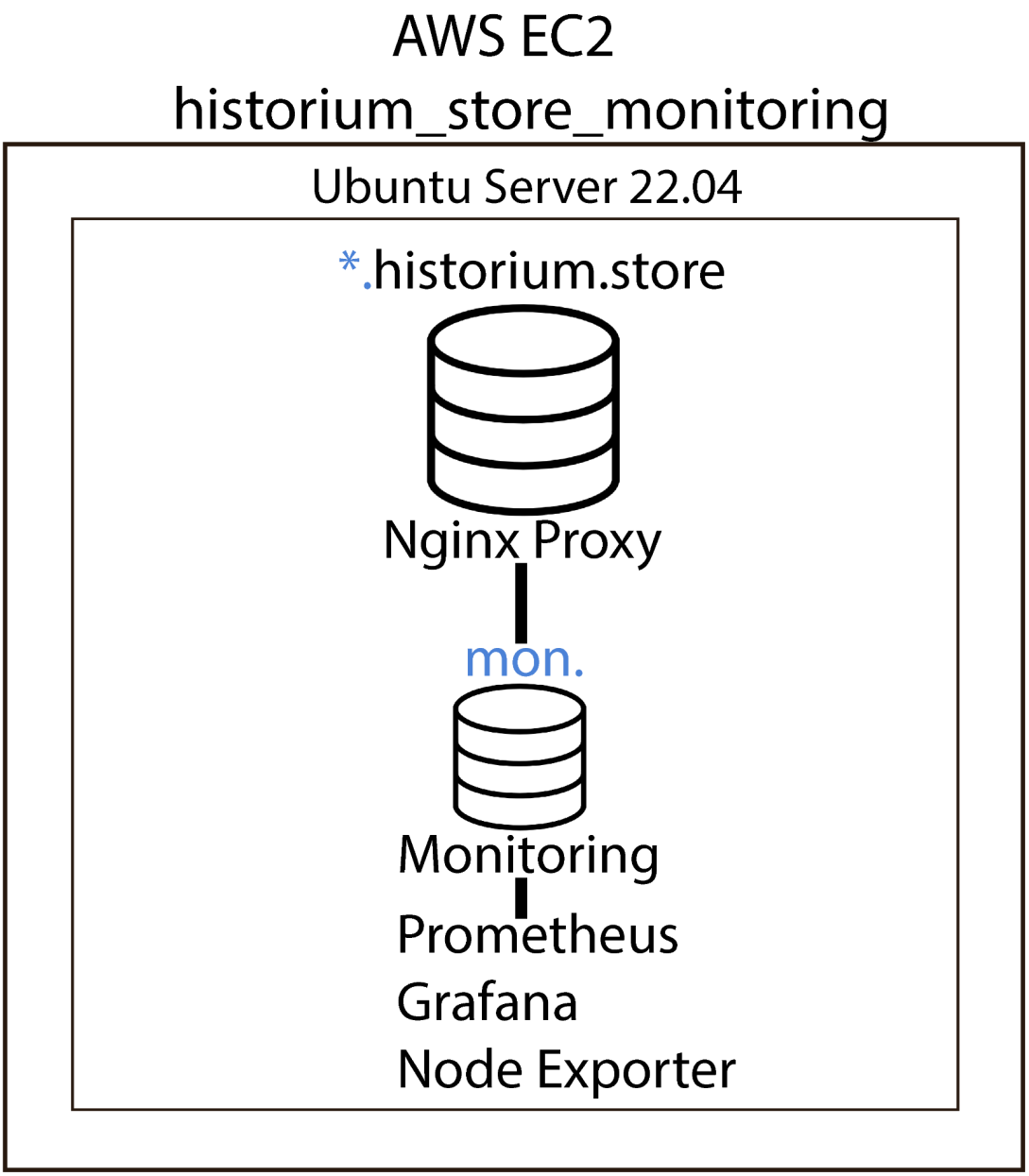


Порти 3000, 9090 та 9100 встановлені на вхід у налаштуваннях AWS EC2 historium_store_monitoring для того, щоб мати змогу побачити роботу ПЗ Prometheus, Node Exporter та Grafana за зверненням до адреси хмарної віртуальної машини historium_store_devops.



Після цих дій, встановлюється і налаштовується ПЗ nginx аналогічно, як з двома попередніми хмарними віртуальними машинами. Але конфігураційний файл nginx відрізняється портом 9100 і назвою самого домену mon.historium.store для звернення до Grafana.

Схема роботи Nginx proxy на 3-й хмарній віртуальній машині:



Конфігураційний файл nginx для архітектурної частини DevOps у хмарній віртуальній машині historium_store_monitoring для ПЗ Prometheus:

```
server {

    listen 80;
    server_name prometheus.historium.store;

    if ($host = prometheus.historium.store) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

    location / {
        proxy_pass         http://127.0.0.1:9090;
        proxy_http_version 1.1;
        proxy_set_header   Upgrade $http_upgrade;
        proxy_set_header   Connection keep-alive;
        proxy_set_header   Host $host;
        proxy_cache_bypass $http_upgrade;
        proxy_set_header   X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header   X-Forwarded-Proto $scheme;
    }

}
server {

    listen 443 ssl;
    server_name prometheus.historium.store;

    ssl_certificate /etc/letsencrypt/live/prometheus.historium.store/fullchain.pem; #
managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/prometheus.historium.store/privkey.pem;
# managed by Certbot
    include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot

    location / {
        proxy_pass         http://127.0.0.1:9090;
        proxy_http_version 1.1;
        proxy_set_header   Upgrade $http_upgrade;
        proxy_set_header   Connection keep-alive;
        proxy_set_header   Host $host;
        proxy_cache_bypass $http_upgrade;
        proxy_set_header   X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header   X-Forwarded-Proto $scheme;
    }
}
```

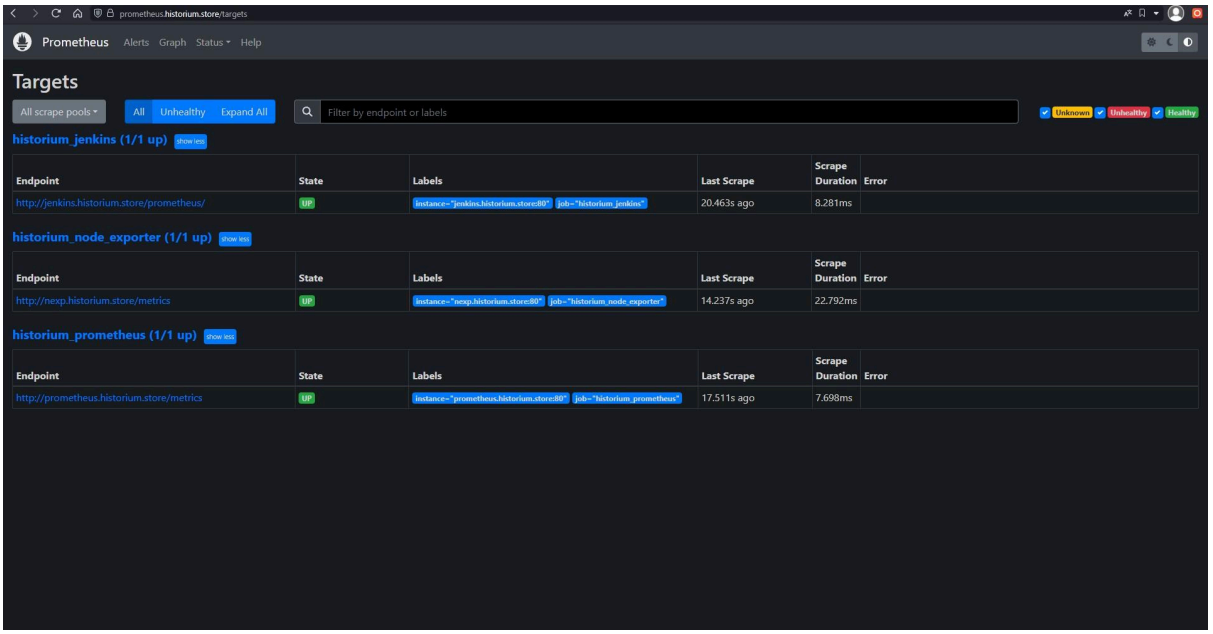
Конфігураційний файл nginx для архітектурної частини DevOps у хмарній віртуальній машині historium_store_monitoring для ПЗ Node Exporter:

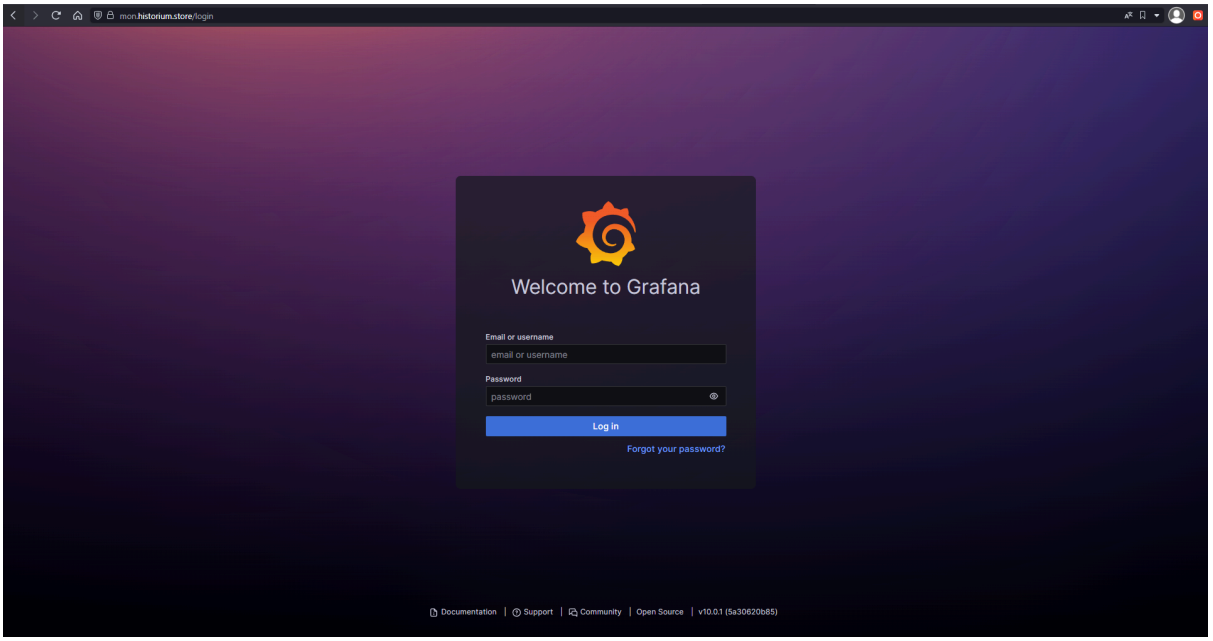
```
server {

    listen 80;
    server_name nexp.historium.store;

    location / {
        proxy_pass        http://127.0.0.1:9100;
        proxy_http_version 1.1;
        proxy_set_header   Upgrade $http_upgrade;
        proxy_set_header   Connection keep-alive;
        proxy_set_header   Host $host;
        proxy_cache_bypass $http_upgrade;
        proxy_set_header   X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header   X-Forwarded-Proto $scheme;
    }


}
server {

    listen 443 ssl;
    server_name nexp.historium.store;

    ssl_certificate /etc/letsencrypt/live/nexp.historium.store/fullchain.pem; #
managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/nexp.historium.store/privkey.pem; #
managed by Certbot
    include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot

    location / {
        proxy_pass        http://127.0.0.1:9100;
        proxy_http_version 1.1;
        proxy_set_header   Upgrade $http_upgrade;
        proxy_set_header   Connection keep-alive;
        proxy_set_header   Host $host;
        proxy_cache_bypass $http_upgrade;
        proxy_set_header   X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header   X-Forwarded-Proto $scheme;
    }

}
```

Конфігураційний файл nginx для архітектурної частини DevOps у хмарній віртуальній машині historium_store_monitoring для ПЗ Grafana:

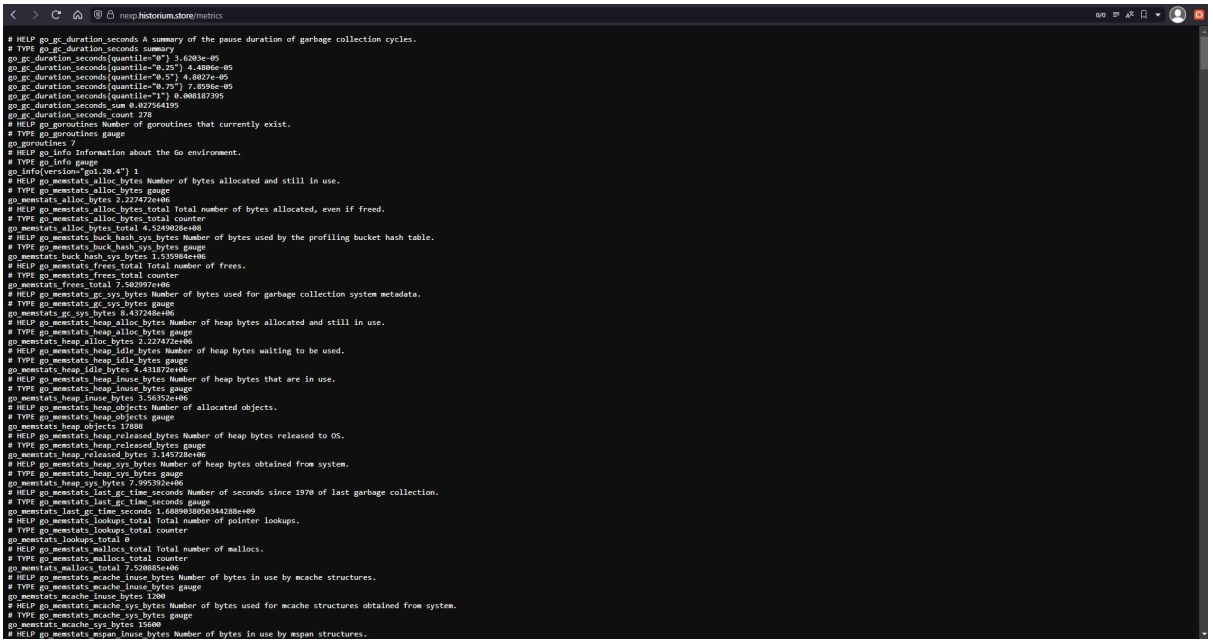
```
server {

    listen 80;
    server_name mon.historium.store;

    if ($host = mon.historium.store) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

    location / {
        proxy_pass        http://127.0.0.1:3000;
        proxy_http_version 1.1;
        proxy_set_header   Upgrade $http_upgrade;
        proxy_set_header   Connection keep-alive;
        proxy_set_header   Host $host;
        proxy_cache_bypass $http_upgrade;
        proxy_set_header   X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header   X-Forwarded-Proto $scheme;
    }


}
server {

    listen 443 ssl;
    server_name mon.historium.store;

    ssl_certificate /etc/letsencrypt/live/mon.historium.store/fullchain.pem; #
managed by Certbot
    ssl_certificate_key /etc/letsencrypt/live/mon.historium.store/privkey.pem; #
managed by Certbot
    include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot

    location / {
        proxy_pass        http://127.0.0.1:3000;
        proxy_http_version 1.1;
        proxy_set_header   Upgrade $http_upgrade;
        proxy_set_header   Connection keep-alive;
        proxy_set_header   Host $host;
        proxy_cache_bypass $http_upgrade;
        proxy_set_header   X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header   X-Forwarded-Proto $scheme;
    }

}
```

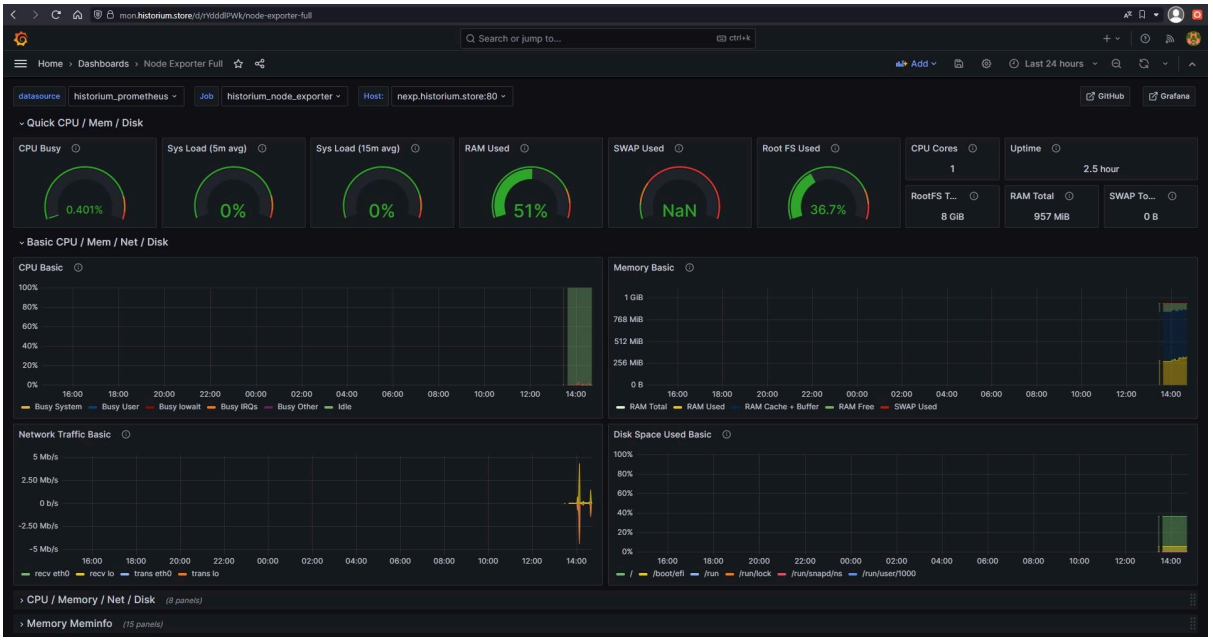
15. Встановлення і налаштування ПЗ Prometheus, Node Exporter, Grafana.



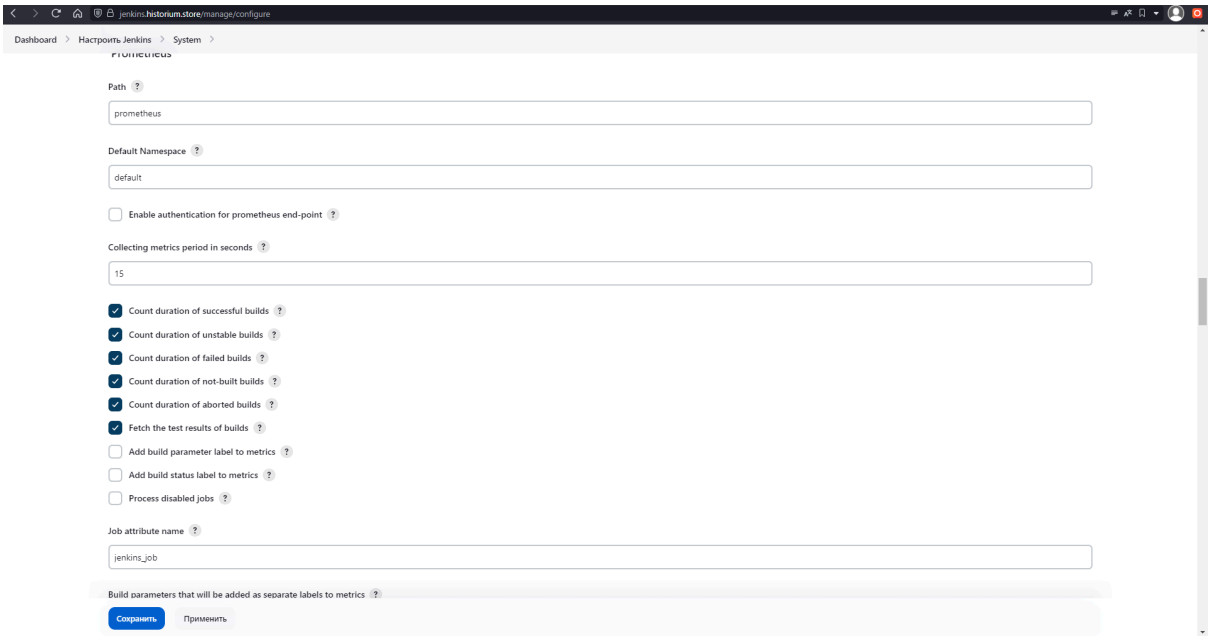


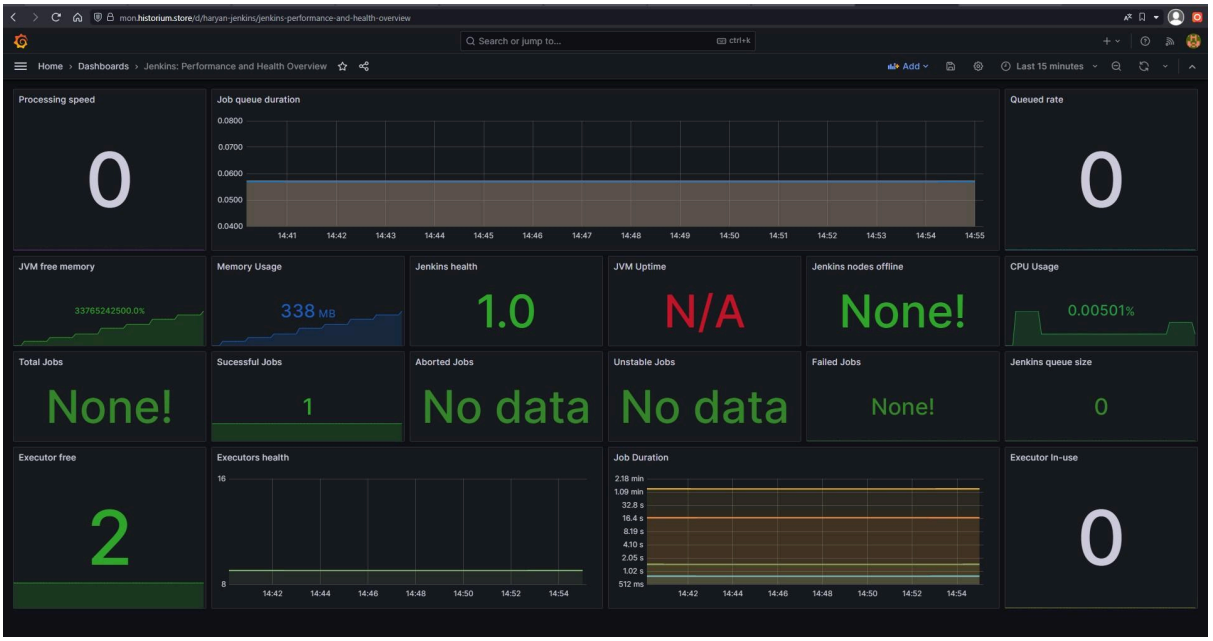
16. Отримання метрік за допомогою Node Exporter і відображення стану за допомогою плагіну Node Exporter Full ID 1860 в Grafana.





17. Отримання метрік Jenkins за допомогою плагіну Prometheus metrics і CloudBees Disk Usage Simple з відображенням стану за допомогою таблиці Jenkins: Performance and Health Overview ID 9964 в Grafana.




mon.historium.store/d/haryan-jenkins/jenkins-performance-and-health-overview
Search or jump to...
ctrl+k
Home › Dashboards › Jenkins: Performance and Health Overview
Add
Last 15 minutes
Processing speed
0
Job queue duration
0.0800
0.0700
0.0600
0.0500
0.0400
14:41 14:42 14:43 14:44 14:45 14:46 14:47 14:48 14:49 14:50 14:51 14:52 14:53 14:54 14:55
Queued rate
0
JVM free memory
33765242500.0%
Memory Usage
338 MB
Jenkins health
1.0
JVM Uptime
N/A
Jenkins nodes offline
None!
CPU Usage
0.00501%
Total Jobs
None!
Sucessful Jobs
1
Aborted Jobs
No data
Unstable Jobs
No data
Failed Jobs
None!
Jenkins queue size
0
Executor free
2
Executors health
16
8
14:42 14:44 14:46 14:48 14:50 14:52 14:54
Job Duration
2.18 min
1.09 min
32.8 s
16.4 s
8.19 s
4.10 s
2.05 s
1.02 s
512 ms
14:42 14:44 14:46 14:48 14:50 14:52 14:54
Executor In-use
0

РОЗДІЛ 3
РЕЗУЛЬТАТИ ВИПРОБУВАНЬ

Результати випробувань та реалізації серверної частини інформаційної системи керування електронними документами та публікаціями, є цікавою та успішною в реалізації.

Основний фокус був зроблений на використання і налаштування хмарного серверу, а також хмарного сховища зберігання файлів. За допомогою впровадження технології Amazon Web Services, це дозволило задовольнити потреби команді розробників щодо розміщення хмарного серверу і контейнера зберігання файлів.

Саме з цього моменту відбувся процес початку роботи, а саме, встановлення та налаштування необхідних компонентів для роботи проєкта. Для розгорнення компонентів Git, Node.js, MongoDB, Docker, була створена тимчасова безкоштовна віртуальна машина Amazon Elastic Cloud 2 від AWS, зі встановленою операційною системою Ubuntu Server 22.04.

Щоб забезпечити безперервне постачання та автоматизації процесів, на окремому хмарному віртуальному сервері використовується інструмент Jenkins. Після його встановлення, були створені та налаштовані pipeline для збірки проєкта Historium з репозиторій GitHub та розгортання у контейнері Docker. Таким чином, це дозволило зменшити кількість зайвих годин та зусилля, пов'язаних з розробкою та впровадженням серверної частини інформаційної системи.

У підсумку, результат випробувань та реалізації серверної частини інформаційної системи керування електронними документами та публікаціями, демонструють успіх у впровадженні сучасних інструментів та практичних розробок, що забезпечують ефективне використання потужностей проєкта та її масштабованістю. І головне те, що клієнти та гості веб-сайту historium.store, мають змогу користуватись самим ресурсом 24/7.

Клієнтські та серверні частини проєкта Historium, доступні в наявності репозиторій GitHub за посиланням:

- https://github.com/historium-store/historium.git
- https://github.com/historium-store/api.git
- https://historium.store

ВИСНОВКИ

Розглянули інформаційні системи, які враховують можливість збереження та взаємодії користувача з товаром.

З'ясовано, що існує тенденція поширення писемності та інших матеріалів. Створення площадки, що дозволить збирати необхідні книги та продавати їх усім користувачам, має сенс стати популярним.

У якості прикладу, для реалізації була вибрана система поширення та продажу книг, шляхом створення інтернет-магазина.

Визначили необхідні програмні засоби, а також середовище, у якому будуть перебувати ресурси проєкта. Інформаційна система керування електронними документами та публікаціями була спроектована, реалізована та випробувана серед команди проєкта.

Система має розподілену архітектуру "клієнт-сервер" та складається з 3 компонентів:

- Mockup
- Frontend
- Backend
- DevOps

Система була випробувана шляхом використання продукту через десктопну версію браузера та у мобільній версії. Усі результати загальної працездатності системи були підтверджені.

ПЕРЕЛІК ВИКОРИСТАНИХ ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ СЕРВЕРНОЇ ЧАСТИНИ

1. Tutorial: Get started with Amazon EC2 Linux instances
   URL: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EC2_GetStarted.html

2. GitHub
   URL: https://docs.github.com/ru/get-started/using-git/about-git

3. Nginx documentation
   URL:https://nginx.org/en/docs/?_ga=2.127011272.456524962.1688913328-1431179894.1688913328

4. Part 2.2 - Install Nginx and configure it as a reverse proxy server
   URL:https://learn.microsoft.com/en-us/troubleshoot/developer/webapps/aspnetcore/practice-troubleshoot-linux/2-2-install-nginx-configure-it-reverse-proxy

5. How To Install Node.js on Ubuntu 22.04
   URL: https://www.digitalocean.com/community/tutorials/how-to-install-node-js-on-ubuntu-22-04

6. Install Docker Engine on Ubuntu
   URL:https://docs.docker.com/engine/install/ubuntu/

7. Let’s Encrypt documentation
   URL: https://letsencrypt.org/docs/

8. User documentation Jenkins
   URL: https://www.jenkins.io/doc/book/installing/linux/

9. Publish Over SSH
   URL: https://plugins.jenkins.io/publish-over-ssh/

10. Prometheus Overview
    URL: https://prometheus.io/docs/introduction/overview/

11. Grafana Documentation
    URL: https://grafana.com/docs/grafana/latest/

12. Вікі ukraine.com.ua
    URL: https://www.ukraine.com.ua/uk/wiki/

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. DevOps культура: як побудувати ефективну команду?
   URL: https://www.superprof.com.ua/blog/shcho-take-devops/

2. Amazon Managed Service for Prometheus
   URL: https://aws.amazon.com/ru/prometheus/

3. Grafana что это и зачем нужно?
   URL: https://www.mivocloud.com/ru/blog/Grafana-what-is-it-and-why-is-it-needed

4. Превентивный мониторинг на базе Prometheus/VictoriaMetrics, Grafana: практика DevOps
   URL: https://gitinsky.com/preventive-monitoring