Приватний заклад вищої освіти

Одеський технологічний університет «ШАГ»
Кафедра інформаційних технологій та фундаментальної підготовки

випускна кваліфікаційна робота бакалавра

«ІНФОРМАЦІЙНА СИСТЕМА УЗГОДЖЕННЯ ВЗАЄМОДІЇ КОРИСТУВАЧІВ (адмін)»

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

Виконавці проекту

№	П.І.Б.	Група
1	Палаткін Данило Анатолійович	КН-А-191

Автор звіту

_____ Палаткін Данило Анатолійович

Одеса – 2023

АНОТАЦІЯ

УДК 004.92

Палаткін Д. А. Автореферат випускної кваліфікаційної роботи на здобуття бакалаврського рівня вищої освіти зі спеціальності «122 Комп’ютерні науки». - Одеса: ОТУШ, 2023 - 13 с.

Дипломна робота описує розробку та впровадження системи DevOps для аналогу проекта kabanchik.ua під назвою BusyBee. Kabanchik.ua це популярна фріланс-платформа, яка допомагає користувачам знайти виконавця для виконання необхідних задач, а виконавцям знайти швидкий спосіб заробити.

Ця дипломна робота ставить за мету показ та опис використання набутих практик та інструментів DevOps для автоматизації та оптимізації розробки, розгортання та тестування системи.

Основні задачі дипломної роботи включають:

- Впровадження інструментів та практик DevOps в існуючий проект BusyBee.
- Налаштування інтеграції та постійної доставки CI/CD для полегшення розробки та впровадження змін в проект.
- Оцінка інтеграції практик DevOps в проект, аналіз ефективності впровадження та користь від використання цієї системи.
- Впровадження системи моніторингу та логування для підтримки надійності та доступності платформи.

Головною метою дипломної роботи є підвищення ефективності розробки та впровадження нових функції для проекту BusyBee, шляхом використання практик та інструментів DevOps. Очікується, що за допомогою впровадження систем DevOps допоможе в швидкості розробки, підвищення автоматизації та стабільності впровадження програмного забезпечення.

Для реалізації практик та інструментів DevOps у проекті BusyBee використовуються основні сервіси та програми, такі як контейнеризація за допомогою Docker, керування версіями коду за допомогою Git, автоматизація збірки та розгортання за допомогою Nginx та Jenkins.

ЗМІСТ

АНОТАЦІЯ **2**
ВСТУП **5**
Мета проекту 5
Основна функціональність: 5
Вимоги до Backend: 5
Вимоги до DevOps: 6
Вимоги до Design: 6
Вимоги до Frontend: 6
Загальний опис проекту: 7
Опис функціональності: 7
Розміщення завдань: 7
Пошук виконавців: 7
Оцінки та відгуки: 7
Розширюваність: 7
Дизайн та користувацький інтерфейс: 8
Загальний стиль дизайну: 8
Макети та прототипи: 8
База даних та зберігання даних: 8
Вибір та опис бази даних: 8
Безпека: 8
Аутентифікація та авторизація: 8
Захист даних: 8
Інфраструктура: 9
Контейнеризація: 9
Хостинг: 9
CI/CD: 9
Моніторинг: 9
Система контролю версій: 9
Використання системи контролю версій: 9
Гілкування та злиття: 9
Робота з командою розробників: 10
Інтеграція з CI/CD: 10
Управління версіями та тегування: 10
Резервне копіювання: 10
Управління проектом: 10
Створення плану проекту: 10
Звітність: 11
РОЗДІЛ 1. Загальна характеристика системи **12**
Frontend 12

Design..... 12
Backend..... 13
Devops..... 14
РОЗДІЛ 2..... 15
Реалізація DevOps.....15
Вимоги до проекту.....15
Вибір інструментів розробки.....16
Діаграма роботи проекту.....16
Вибір хостингу в AWS та початкове налаштування серверів за допомогою Terraform.....17
CI/CD Server:..... 18
Main Server:.....21
Створення Docker-файлів та Docker Compose-файлів..... 21
Налаштування сервера Jenkins та CI/CD.....23
Послідовність дій:.....23
Створення завдання.....23
Налаштування тригера.....23
Налаштування складання проекту.....23
Налаштування деплою.....23
Перевірка та тестування..... 23
Налаштування сервера моніторингу та оповіщень..... 24
ВИСНОВКИ..... 25
ПЕРЕЛІК ВИКОРИСТАНИХ ТЕХНОЛОГІЙ..... 26

ВСТУП

Мета проекту

Створити інформаційну систему узгодження взаємодії користувачів під назвою "Busy Bee" для надання платформи, де користувачі можуть розміщувати завдання та знаходити виконавців.

Основна функціональність:

Розміщення завдань, пошук виконавців, управління користувачами, платіжна система.

Вимоги до Backend:

- Архітектура:
  - Використання монолітної архітектури.
- Технології:
  - Мова програмування: C#
  - Фреймворк: ASP.NET WebApi.
  - ORM: Entity Framework Core для роботи з базою даних PostgreSQL.
  - Мапер: AutoMapper для перетворення даних між моделями.
  - Валідація: FluentValidation для перевірки вхідних даних.
  - Аутентифікація та авторизація: Використання Azure AD B2C для управління користувачами.
  - Шаблон проектування: Clean architecture.
  - Unit of Work / Repository Pattern: Для управління доступом до даних.
  - Документація API: Використання Swagger для автоматичного генерування документації.
  - Тестування: Використання Postman для тестування API.

Вимоги до DevOps:

- Контейнеризація: Використання Docker для контейнеризації додатка.
- Хостинг: Розгортання на Amazon Web Services.
- Система контролю версій: Використання Git для управління кодовою базою.
- CI/CD інструменти: Використання Jenkins для автоматизації процесів CI/CD.
- Планування та розгортання інфраструктури проекту.
- Автоматизація процесів розгортання, моніторингу та масштабування системи.
- Встановлення та налаштування інструментів для CI/CD.
- Забезпечення безпеки та контролю якості проекту.
- Управління конфігурацією та забезпечення надійності системи.

Вимоги до Design:

- Проектування користувацького інтерфейсу та користувацького досвіду.
- Використання інструментів для дизайну: Фігма, Ілюстратор, Фотошоп.

Вимоги до Frontend:

- Фреймворк: Використання Angular для розробки фронтенд-частини сайту.
- Інтегроване середовище розробки: Використання WebStorm для розробки на Angular.
- Управління залежностями: Використання npm для управління пакетами.

Загальний опис проекту:

Опис функціональності:

Розміщення завдань:

- Створення нового завдання з вказанням категорії, опису та інших необхідних полів.
- Редагування та видалення завдання.
- Управління статусом завдання (відкрите, в роботі, завершене).

Пошук виконавців:

- Пошук виконавців за категоріями, навичками, локацією та іншими параметрами.
- Відображення списку доступних виконавців з основною інформацією та рейтингом.
- Фільтрація результатів пошуку та сортування за різними критеріями.

Оцінки та відгуки:

- Можливість оцінювати виконавців та залишати відгуки після завершення завдання.
- Відображення рейтингу та відгуків на профілях виконавців.
  Цільова аудиторія:
- Визначення цільової аудиторії відповідно до виду завдань та послуг, пропонованих на сайті.
- Аналіз потреб і очікувань користувачів щодо функціональності та користувацького досвіду.

Розширюваність:

- Розробка модульної архітектури, що дозволяє легко додавати нові функції та можливості.
- Розгляд можливості інтеграції з іншими сервісами або платформами для розширення функціональності сайту.

Дизайн та користувацький інтерфейс:

Загальний стиль дизайну:

- Визначення бажаного стилю дизайну (мінімалістичний).
- Визначення колірної палітри, шрифтів та інших візуальних елементів, що відповідають обраному стилю.

Макети та прототипи:

- Створення макетів користувацького інтерфейсу для різних сторінок та функціональних елементів сайту.
- Розробка інтерактивних прототипів для демонстрації користувацької взаємодії та навігації по сайту.

База даних та зберігання даних:

Вибір та опис бази даних:

- Розробка структури бази даних, включаючи таблиці, зв'язки між ними та ключові поля.
- Врахування вимог до продуктивності та масштабованості бази даних.
- Зберігання даних:
- Визначення даних, які повинні зберігатися в базі даних, включаючи інформацію про користувачів, завдання, коментарі тощо.
- Розробка моделей даних та визначення зв'язків між ними.

Безпека:

Аутентифікація та авторизація:

- Реалізація процесів аутентифікації користувачів з використанням Azure AD B2C.
- Встановлення прав доступу та ролей для різних типів користувачів.

Захист даних:

- Застосування заходів безпеки для захисту користувацьких даних, включаючи шифрування зберігання та передачі даних.

Інфраструктура:

Контейнеризація:

- Використання Docker для контейнеризації Backend та Frontend частини додатка.
- Створення Docker-контейнерів для компонентів додатка та їх оркестрації.

Хостинг:

- Розгортання додатка на хмарному сервісі Amazon Web Services.

CI/CD:

- Використання Jenkins для налаштування процесів CI/CD.
- Автоматизація процесів розгортання, тестування та випуску нових версій додатка.

Моніторинг:

- Налаштування моніторингу системи.
- Відстеження метрик продуктивності, завантаження та інших параметрів для забезпечення стабільності та швидкодії сайту.

Система контролю версій:

Використання системи контролю версій:

- Впровадження Git для управління версіями та переглядом історії змін.
- Створення репозиторію Git та впровадження спільної роботи розробників за його допомогою.

Гілкування та злиття:

- Визначення стратегії гілкування та злиття коду, наприклад, використання моделі Git Flow.
- Створення та керування різними гілками для розробки нових функцій, виправлення помилок та випуску стабільних версій.

Робота з командою розробників:

- Опис угод щодо найменування гілок, коментарів до комітів та інших правил для спільної роботи над проектом.
- Визначення процесу рецензування коду та схвалення змін.

Інтеграція з CI/CD:

- Використання інструментів CI/CD, таких як Jenkins, для автоматизації процесів зборки, тестування та розгортання додатка.
- Налаштування пайплайнів CI/CD, що включають етапи збірки, тестування, аналізу коду та розгортання на сервері.

Управління версіями та тегування:

- Опис правил для присвоєння версій релізам, виправленням помилок та іншим змінам.
- Використання тегів Git для маркування важливих точок в історії проекту, таких як релізи або майлстоуни.

Резервне копіювання:

- Розробка стратегії резервного копіювання репозиторію Git для забезпечення збереженості коду та історії змін.

Управління проектом:

Створення плану проекту:

- Визначення етапів розробки, термінів та завдань.
- Встановлення пріоритетів та розподіл ресурсів. Комунікація та співпраця:
- Встановлення засобів комунікації та співпраці між членами команди.
- Використання інструментів для спільного редагування та обміну документами.

Звітність:

- Вихідний код повинен оформлятися у відповідності до загальноприйнятих стандартів, на зразок стандартів GNU http://www.gnu.org/prep/standards/, у тому числі з додержанням вимог до коментування та документування коду.
- Зберігання вихідних кодів ПЗ повинно здійснюватись за допомогою системи контроля версій (вибір конкретної системи узгоджується групою проекту). Структура файлів та каталогів повинна відбивати архітектуру ПЗ, на зразок:
    - root
    - Backend
    - Web
- Підготовка звітів про стан проекту та результати роботи.
- Презентація результатів перед командою та замовником проекту.

РОЗДІЛ 1. Загальна характеристика системи

Архітектура платформи складається з таких частин:

- Frontend
- Design
- Backend
- Devops

Кожна з цих частин виконує важливі функції і має свою суттєву роль у проекті:

Frontend

Фронтенд є тим, що користувачі бачать та взаємодіють з платформою BusyBee. Його суттєва частина полягає в наступному:

- Розробка інтерфейсу користувача: Фронтенд-розробники відповідають за розробку естетичного та інтуїтивно зрозумілого дизайну, який забезпечує зручну навігацію та взаємодію з платформою для користувачів.
- Реалізація клієнтської логіки: Фронтенд-розробники виконують програмування на мові JavaScript та використовують фреймворки, такі як React або аналогічні, для створення функціональних компонентів та обробки подій користувача.
- Взаємодія з бекендом: Фронтенд комунікує з бекендом через API, взаємодіючи з базою даних та отримуючи необхідні дані для відображення на сторінках платформи.

Frontend необхідний, оскільки він створює враження користувача про платформу BusyBee і забезпечує зручну та ефективну взаємодію з нею. Візуальний дизайн, логіка frontend та взаємодія з бекендом роблять платформу привабливою та інтуїтивно зрозумілою для користувачів.

Design

Дизайн є ключовим елементом платформи BusyBee і відповідає за створення естетичного та функціонального інтерфейсу для користувачів. Його суттєва частина полягає в наступному:

- Розробка макетів: Дизайнер створює макети веб-сторінок та мобільних додатків, враховуючи потреби користувачів, принципи взаємодії та сучасні дизайн-стандарти.
- Візуальне оформлення: Дизайнер виконує вибір кольорової палітри, шрифтів, іконок та інших елементів дизайну, що сприяють створенню привабливого та зрозумілого інтерфейсу.
- Подання інформації: Дизайнер розробляє ефективні способи відображення даних та важливої інформації, забезпечуючи логічну та зручну навігацію користувачів.

Design необхідний для того, щоб платформа BusyBee була привабливою та зрозумілою для користувачів. Естетичний дизайн, правильна організація інформації та інтуїтивно зрозумілі елементи дозволяють користувачам легко орієнтуватись та взаємодіяти з платформою.

Backend

Бекенд відповідає за обробку бізнес-логіки, збереження даних та взаємодію з фронтендом та базою даних. Суттєва частина backend включає:

- Розробка серверної логіки: Бекенд-розробники створюють серверну частину, яка обробляє запити від frontend та забезпечує відповідну взаємодію з базою даних.
- Безпека та автентифікація: Бекенд-розробники впроваджують механізми автентифікації та авторизації користувачів, а також заходи безпеки для захисту даних.
- Інтеграція з базою даних: Бекенд-розробники виконують розробку та оптимізацію бази даних, забезпечуючи ефективне зберігання та отримання інформації.
- Кожна з цих частин - Frontend, Design, Backend - має свою важливу роль у реалізації платформи BusyBee. Робота цих елементів в комбінації дозволяє створити функціональну та привабливу платформу для користувачів.

Backend є важливим компонентом платформи BusyBee, оскільки він забезпечує обробку даних, безпеку та взаємодію з фронтендом. Від правильної реалізації backend залежить швидкодія платформи та безпека користувачів.

Devops

DevOps - це важлива частина команди проекту BusyBee, яка відповідає за забезпечення ефективної розробки, впровадження та управління інфраструктурою та середовищем розробки. Основні завдання DevOps включають:

- Налаштування інфраструктури: DevOps-інженери створюють та налаштовують середовища розробки, тестування та виробництва, використовуючи автоматизацію та інструменти інфраструктури як коду.
- Конфігураційне управління: DevOps-інженери використовують інструменти, такі як Ansible, для управління конфігураціями серверів та забезпечення стабільності та складності середовища.
- Автоматизація процесів: DevOps-інженери автоматизують процеси розробки, випуску та впровадження програмного забезпечення, використовуючи інструменти, такі як Jenkins або GitLab CI/CD.
- Моніторинг та логування: DevOps-інженери встановлюють механізми моніторингу та логування, щоб відстежувати продуктивність, виявляти проблеми та забезпечувати надійну роботу платформи.

DevOps-інженери необхідні для платформи BusyBee, оскільки вони забезпечують безперебійну розробку та впровадження програмного забезпечення, автоматизоване управління інфраструктурою та оптимізацію процесів. Їхні зусилля спрямовані на забезпечення швидкої, стабільної та безпечної роботи платформи для задоволення потреб користувачів.

РОЗДІЛ 2
Реалізація DevOps

У цьому розділі описується реалізація DevOps практик для проекту BusyBee. Розглянемо основні кроки реалізації DevOps інструментів, а саме вибір хостингу, налаштування серверів під необхідні сервіси для роботи з проектом, встановлення необхідних компонентів, налаштування автоматизації, налаштування контролю версій та автоматизації розгортання та складання.

1. Вимоги до проекту
2. Вибір інструментів розробки
3. Діаграма роботи проекту
4. Вибір хостингу в AWS та початкове налаштування серверу
5. Налаштування серверів за допомогою Terraform
6. Створення Docker-файлів та Docker Compose-файлів
7. Налаштування сервера Jenkins та CI/CD
8. Налаштування моніторингу

Вимоги до проекту

При розробці проекту busybee вимоги до devops-інженера включали такі аспекти:

- Хостинг: Потрібно обрати та налаштувати хостинг для розташування інстраструктури. Хостинг має забезпечувати безперервну роботу, доступність, надійність, а також бути відкритим для розташування необхідних сервісів та управління інфраструктурою.
- DNS: Налаштування DNS для зв’язку з IP-Адресами серверів та рєестрації домену та піддоменів. Він має забезпечувати надійну роздільність імен та перенаправлення трафіку.
- Безперервна інтеграція та розгортання (CI/CD): Налаштувати систему безперервної інтеграції та розгортання для тестування оновлень та ініціалізації нових оновлень без загрози для проекту.

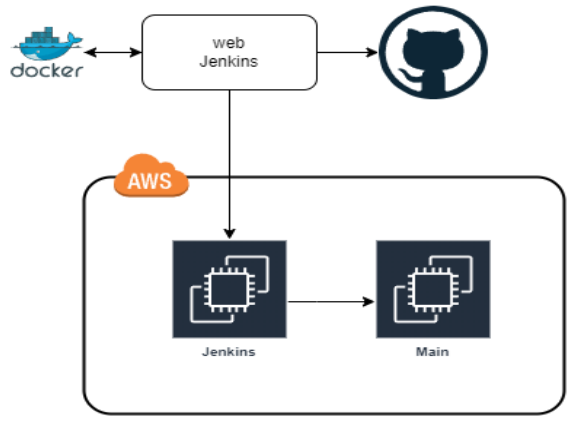
Вибір інструментів розробки

У проекті BusyBee були обрані наступні інструменти, необхідні для ефективної роботи інженера DevOps:

- Docker: Docker дозволяє створювати та керувати контейнерами, забезпечуючи ізоляцію та переносимість додатків. Він дозволяє розробникам безперешкодно працювати в однорідному середовищі, спрощуючи процес розгортання додатків
- AWS: AWS хмарний сервіс з різноманітним функціоналом який ми обрали як хостинг. Має в собі багато потрібних функції які можуть покращити проект як у безпеці так і у зручності, та багато іншого.
- Terraform: Terraform це інструмент легкого налаштування інфраструктури за допомогою коду. Він допомагає швидко орієнтуватись у налаштуваннях та швидко змінювати за допомогою однієї строки коду.
- Jenkins: Jenkins це популярний інструментом для безперервної інтеграції та розгортання (CI/CD). Він автоматизує процеси складання, тестування та розгортання додатків, забезпечуючи швидкий зворотній зв'язок розробникам та полегшуючи управління кодовою базою.

Вибір саме цих інструментів обумовлений по-перше зручністю використанні, популярністю, гнучким налаштуванням та можливістю інтеграції і взаємодії між собою. Вони допомагають у керуванні інфраструктурою, автоматизації процесів та надає можливість забезпечити надійність та безпеку проекту.

Діаграма роботи проекту.



Вибір хостингу в AWS та початкове налаштування серверів за допомогою Terraform.

Було проведено вибір хостингу AWS для цього проекту, саме з причини популярності цього сервісу, великим вибором функцій та зручністю налаштування.

Для ефективного використання цього сервісу налаштування відбувалось за допомогою Terraform. Він дозволяє швидко орієнтуватись у налаштуваннях, тому що воно відбувається завдяки коду.

```
resource "aws_instance" "web" {
  ami                         = var.ami
  instance_type               = var.instance_type
  key_name                    = var.key_name
  security_groups             = var.sg
  tags                        = var.tags
  user_data                   = var.user_data
  user_data_replace_on_change = true
}
```

Для зручності використання цього проекту іншим DevOps спеціалістом, було розділено основні налаштування ще на декілька гілок:

```
variable "ami" {
  type = string
}
variable "instance_type" {
  type = string
}
variable "user_data" {
  type = string
```

```
}
variable "tags" {
  type = map(string)
}
variable "sg" {
  type = list(string)
}
variable "key_name" {
  type = string
}
```

```
variable "ami" {
  type    = string
  default = "ami-0b0c5a84b89c4bf99"
}
variable "instance_type" {
  type    = string
  default = "t2.micro"
}
```

```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "5.6.2"
    }
  }
}
provider "aws" {
  region = "eu-central-1"
}
```

Початково було обрано t2.micro тип серверу, але у процесі впровадження DevOps систем цього виявилось мало, тому я обрав тип серверу t2.medium.

CI/CD Server(Jenkins):

Цей сервер відповідає за неперервну інтеграцію та доставку (CI/CD) за допомогою інструменту Jenkins. На ньому налаштована автоматична збірка та деплой проекту.
Спочатку запускається перший процес, який збирає докерімейдж та пушить його в докерхаб, після вдалого завершення одразу автоматично запускається другий процес який переводить докерімейдж на кінцевий основний сервер з сайтом, де його розгортає та ми бачимо робочий сайт.

Запуск першого процесу: http://3.78.166.15:8080/job/build_docker2/
Запуск другого процесу: http://3.78.166.15:8080/job/build_docker/

Конфігураційний файл першого процесу:

```
node {
    stage('Clone repository') {
        git credentialsId: 'jenkins-ssh-key', url:
'git@github.com:DLowoY/diplom.git'
    }

    stage('Build web image') {
        dir('/var/lib/jenkins/workspace/build_docker/BusyBeeWeb') {
            dockerImage = docker.build("dlowoy/my-app-web:latest")
        }
    }

    stage('Push web image') {
        withDockerRegistry([ credentialsId: "dockerhub-cred", url: "" ]) {
        dockerImage.push()
        }
    }

    stage('Build image') {
        dir('/var/lib/jenkins/workspace/build_docker/BusyBee') {
            dockerImage = docker.build("dlowoy/my-app:latest")
        }
    }
```

```
    stage('Push image') {
        withDockerRegistry([ credentialsId: "dockerhub-cred", url: "" ]) {
        dockerImage.push()
        }
    }

    stage('Delete old images') {
        sh 'docker system prune -a -f'
        sh 'docker ps -a'
        sh 'docker images -a'
    }
}
```

Конфігураційний файл другого процесу:

```
docker stop busybee.api
docker rm busybee.api
docker rmi dlowoy/my-app-web:latest

docker stop postgres
docker rm postgres
docker rmi postgres:latest

docker compose -f ~/jenkins_remote_dir/docker-compose.yml up -d

docker stop my-app
docker rm my-app
docker rmi dlowoy/my-app:latest

docker pull dlowoy/my-app:latest
docker run -dit -p 80:80 --restart=always --name my-app dlowoy/my-app:latest
```

Main Server:

Це основний сервер який приймає на себе вдало запушений докерімедж, який у кінцевому показує вже готовий до користування сайт. Він надає доступ саме до веб-інтерфейсу щоб користувачі могли взаємодіяти з проектом.
Посилання на працюючий сайт: http://18.156.149.74/

Окрім всього були налаштовані деякі правила доступу до налаштування серверів, чим саме забезпечено проект від стороннього доступу, це забезпечує доступ до налаштувань серверу лише авторизованим користувачам у яких є права доступу. Але ще можливо покращити безпеку та надійність системи.

Створення Docker-файлів та Docker Compose-файлів.

Dockerfile:

```
docker stop busybee.api
docker rm busybee.api
docker rmi dlowoy/my-app-web:latest

docker stop postgres
docker rm postgres
docker rmi postgres:latest

docker compose -f ~/jenkins_remote_dir/docker-compose.yml up -d

docker stop my-app
docker rm my-app
docker rmi dlowoy/my-app:latest

docker pull dlowoy/my-app:latest
docker run -dit -p 80:80 --restart=always --name my-app dlowoy/my-app:latest
```

Docker-compouse:

```
version: "3.9"
```

```
services:
  busybee_api:
    image: dlowoy/my-app:latest
    container_name: busybee.api
    restart: always
    volumes:
      - /var/www/images:/app/images
      - /var/www/files:/app/files
    depends_on:
      - postgres_db
    environment:
      - ASPNETCORE_ENVIRONMENT=Development
      - ASPNETCORE_URLS=http://+:80
      -
ConnectionStrings:DefaultConnection=Host=postgres;Database=busy_bee_db;Usern
ame=postgres;Password=12345678
      - AzureAdB2C:Instance=https://busybeeua.b2clogin.com
      - AzureAdB2C:Domain=busybeeua.onmicrosoft.com
      - AzureAdB2C:ClientId=44d3647b-fe43-4b84-bc50-7b99f37fd8f2
      - AzureAdB2C:SignedOutCallbackPath=/signout/B2C_1_SignupSignin
      - AzureAdB2C:SignUpSignInPolicyId=B2C_1_SignupSignin
      - AzureAdB2C:OpenApiClientId=0e271133-d0bf-493a-976d-8146db6770a4
      -
AzureAdB2C:OpenApiScopeApi=https://busybeeua.onmicrosoft.com/api/main
      - LocalStorage:ImagesUrlPrefix=/images/
      - LocalStorage:ImagesFolderPath=images
      - LocalStorage:FilesUrlPrefix=/files/
      - LocalStorage:FilesFolderPath=files
      - LocalStorage:CategoryIconFilenamePrefix=category_icon_
      - LocalStorage:UserPhotoFilenamePrefix=user_photo_
      - LocalStorage:PortfolioFilenamePrefix=portfolio_
    ports:
      - "127.0.0.1:8001:80"

  postgres_db:
    image: postgres:latest
    container_name: postgres
    restart: always
    environment:
      - POSTGRES_USER=postgres
      - POSTGRES_PASSWORD=12345678
    ports:
      - "5432:5432"
```

Налаштування сервера Jenkins та CI/CD

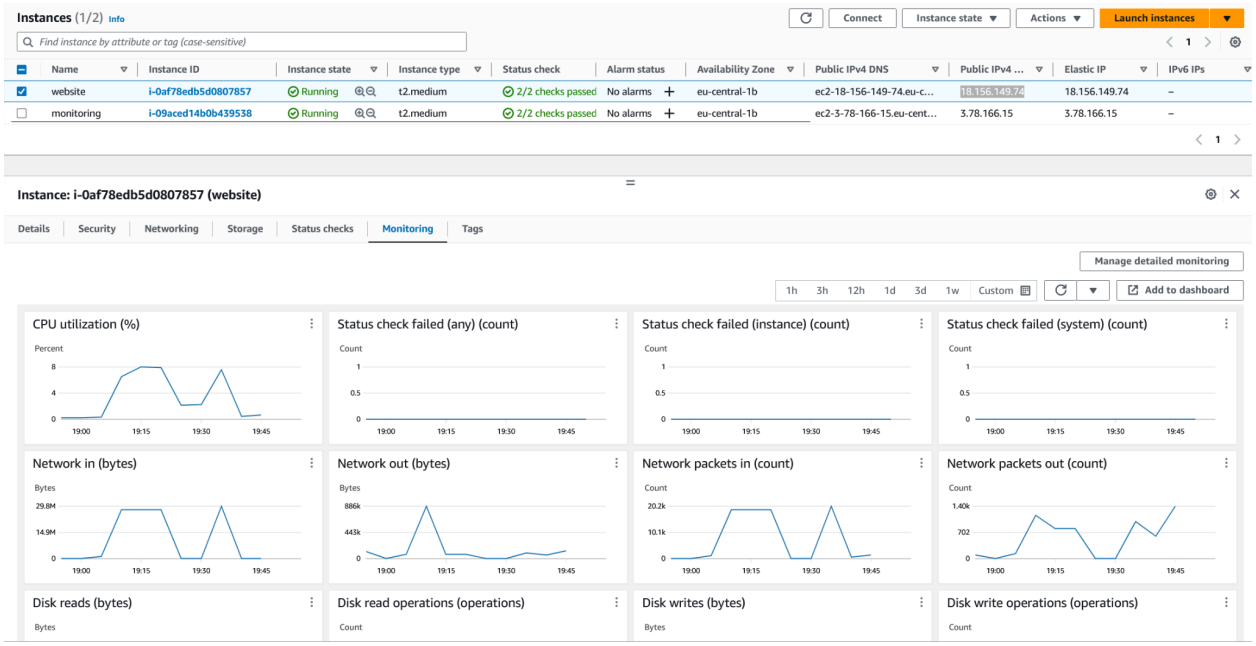
При розробці проекту створення сайту busybee налаштування Jenkins і CI/CD (Continuous Integration/Continuous Deployment) є важливою частиною процесу DevOps. Цей розділ демонструє детальну послідовність налаштування Jenkins, створення завдання, налаштування тригеру по якому після вдалого виконання першого процесу запускається автоматично другий.

Послідовність дій:

- Створення завдання:
    - Створення завдань Jenkins, які будуть виконувати CI/CD процес для конкретного проекту.
    - Створення PAT, вказуючи джерело, в нашому випадку до AWS.
- Налаштування тригера:
    - Обрати тип тригеру за розкладом або при заміні коду.
    - Встановлення та налаштування необхідних параметрів для вибраного тригеру.
- Налаштування складання проекту:
    - Визначення кроків, які мають виконуватися під час кожного запуску процесу.
    - Налаштування середовища виконання з вказанням залежності та параметри складання проекту.
- Налаштування деплою:
    - Установка плагіну "Publish Over SSH", який дозволить розгорнути збирання на потрібному сервері.
    - Налаштування доступу до SSH, та визначення кроків деплою, які мають виконуватись.
- Перевірка та тестування:
    - Перевірка процесу складання деплою.

Налаштування сервера моніторингу

Моніторинг є невід'ємною частиною проекту, але в AWS вже є влаштований моніторинг який показує навантаженість, та використання ресурсів серверу.



ВИСНОВКИ

У ході виконання даної дипломної роботи було проведено значну кількість роботи з впровадження принципів DevOps в проекту BusyBee. Цей процес покращив відмовостійкість продукти, можливість робити зміни та оновлення продукти без страху загубити попередню стабільну версію.

Одне з ключових завдань було впровадження безперервної доставки (CI/CD) автоматизації та можливості оновлення продукту без проблем з втратою стабільної версії та перешкодженням роботи проекту. Це допомогло легко тестувати оновлення, або нові функції без втрати стабільної версії.

Система моніторингу була взята вже влаштована в сервіс AWS, (AWS CloudWatch) яка допомагає передивлятись навантаження на сервер, виявляти проблема, з тих даних вже формувати в чому може бути проблема.

Незважаючи на проведену велику роботу над проектом, можно його ще покращити налаштувавши систему моніторінгу через Graphana та Prometheus. Також підключити сервери до домену для розподілення навантаження на сайт.

Ще для підвищення надійності продукти можно буде впровадити резервне копіювання серверів, та балансу навантаження між іншими серверами, для більш ефективної відказостійкості. Це дозволить запобігти перевантаженню сайта через великий трафік, та дозволить швидко відновивти втраченні данні.

Підсумок всього, те що впровадження практик та систем DevOps дуже корисна для проекту, тому це дуже необхідна частина будь-якої команди. Але ще є в чому покращити проект, та зробити його набагато краще.

ПЕРЕЛІК ВИКОРИСТАНИХ ТЕХНОЛОГІЙ

№ п.п.	**Ресурс**
1.	Docker, Docker Compose, DockerHub Розробник: Docker, Inc. Документація
2.	Nginx Розробник: F5, Igor Sysoev, Netcraft Документація
3.	Jenkins Розробник: Kohsuke Kawaguchi Документація
4.	Publish Over SSH Розробник: Gavin McDonald, Olivier Lamy Документація
5.	ChatGPT Розробник: OpenAI Документація
6.	Terraform Розробник: HashiCorp Документація
7.	AWS DevOps Розробник: Amazon Документація
8.	Let’s Encrypt Розробник: ISRG Документація
9.	Ukraine.com.ua Розробник: ТОВ "Хостинг «Україна»" Документація