**Міністерство Освіти і Науки України**

**Київський Національний Університет імені Тараса Шевченка**

**Факультет Інформаційних Технологій**

**Кафедра Інформаційних систем та технологій**

Звіт з лабораторної роботи № 2

з дисципліни **«Проєктування програмного та апаратного забезпечення»**

Виконав студент 1-го курсу магістратури

          групи ІРма-12

Гаврасієнко Є.О.

**Київ – 2024**

**Тема: “Створити архітектуру хмарного сервісу для відеоконференцій.”**

**Завдання 1: Вибір архітектурного стилю**

1. **Обрати сценарій**

**“Створити архітектуру хмарного сервісу для відеоконференцій.”**

1. Обґрунтувати вибір архітектурного стилю.

**Для розробки хмарного сервісу відеоконференцій обрано мікросервісну архітектуру.**

**Чому саме мікросервіси?**

**1. Масштабованість**

* **Відеоконференції мають змінне навантаження: під час пікових годин кількість активних користувачів може значно зростати.**
* **Мікросервісний підхід дозволяє масштабувати окремі компоненти (наприклад, сервіс відеопотоків або чат) незалежно один від одного.**

**2. Гнучкість і незалежність компонентів**

* **Відеоконференція складається з багатьох окремих функцій (аутентифікація, відеострімінг, чат, запис тощо), які можна реалізувати у вигляді окремих мікросервісів.**
* **Це спрощує підтримку та розробку нових функцій без впливу на всю систему.**

**3. Стійкість до збоїв**

* **Якщо один мікросервіс виходить з ладу (наприклад, запис відео), інші (наприклад, сам відеодзвінок) продовжують працювати.**
* **Використання механізмів балансування навантаження (наприклад, Kubernetes) покращує надійність.**

**4. Використання спеціалізованих технологій для різних завдань**

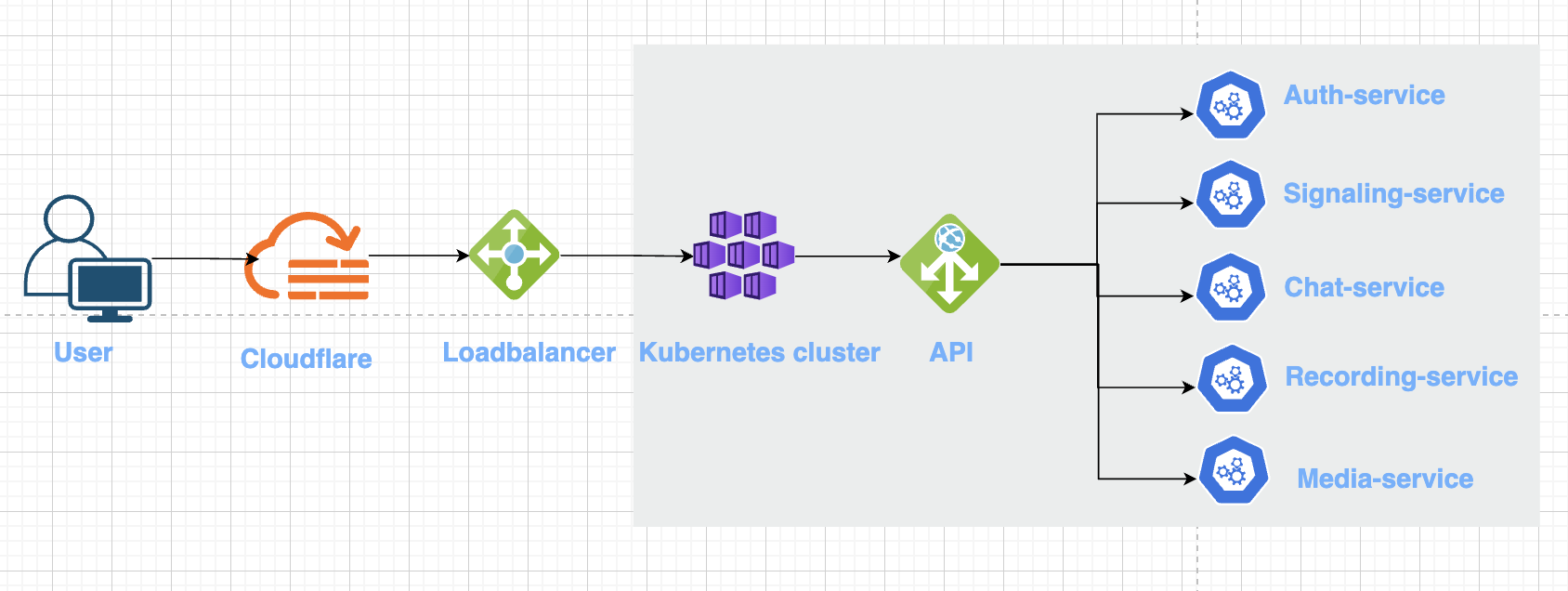
* **Сервіс відеопотоків може використовувати WebRTC і Kurento,**
* **Аутентифікація – Keycloak,**
* **Чат – WebSockets або Firebase,**
* **Запис відео – FFmpeg і AWS S3.**
* **Усе це легше реалізувати в мікросервісній архітектурі, де кожен сервіс має власну технологічну стек.**

**5. Простота оновлення та безперервної інтеграції (CI/CD)**

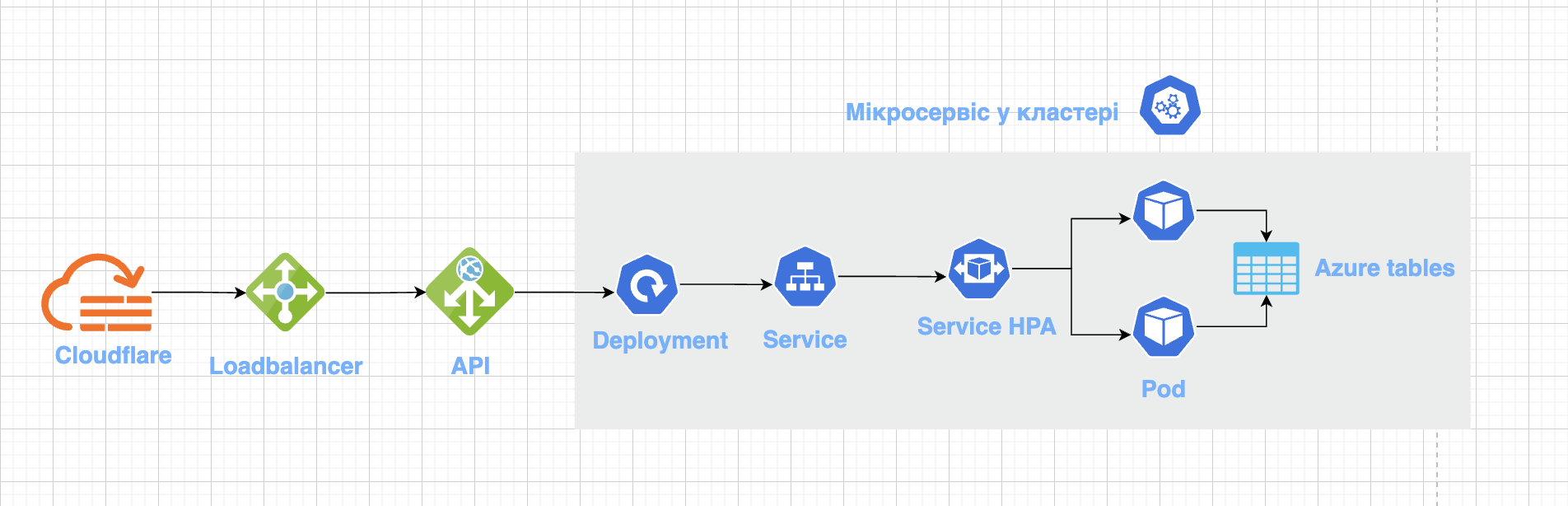
* **Кожен мікросервіс можна оновлювати незалежно від інших, що спрощує релізи.**
* **Автоматизація розгортання через Kubernetes і GitHub Actions забезпечує швидке впровадження змін.**

**Завдання 2: Побудова архітектурної схеми**

**Створити Component Diagram для обраної архітектури.**



**Додати Deployment Diagram для розгортання системи.**



Завдання 3: Аналіз переваг і недоліків

1. Сильні сторони обраного рішення:

✅ Гнучкість та масштабованість

* Використання мікросервісної архітектури дозволяє незалежно масштабувати окремі сервіси, що важливо для високонавантажених систем.
* Автоматичне масштабування (HPA) в Kubernetes забезпечує адаптацію до навантаження в реальному часі.

✅ Висока доступність та відмовостійкість

✅ Cloudflare забезпечує захист від DDoS-атак і прискорює роботу системи.

✅ Гнучке керування трафіком

* API Gateway дозволяє ефективно маршрутизувати запити між мікросервісами.

✅ Зручне оновлення

* Мікросервіси можна оновлювати незалежно один від одного, що спрощує розгортання нових функцій.

2. Можливі ризики

⚠️ Складність управління • Kubernetes вимагає досвіду адміністрування, а також ретельного налаштування CI/CD для безперебійного розгортання.

⚠️ Збільшена затримка через мережеві виклики • Оскільки мікросервіси комунікують через мережу, це може спричиняти накладні витрати та додаткові затримки.

⚠️ Безпека • Необхідно ретельно контролювати аутентифікацію між сервісами (наприклад, за допомогою JWT або mTLS).

3. Порівняння з альтернативними підходами:

|  |  |  |
| --- | --- | --- |
| Підхід | Переваги | Недоліки |
| Монолітна архітектура | Простота управління, менше мережевих затримок | Важко масштабувати, складно оновлювати окремі компоненти |
| Мікросервіси (обране рішення) | Гнучкість, масштабованість, незалежне оновлення | Вища складність, потреба в оркестрації (Kubernetes) |
| Serverless (FaaS, наприклад, AWS Lambda) | Мінімальні витрати на інфраструктуру, автоматичне масштабування | Менший контроль над середовищем, обмеження щодо тривалості виконання запитів |

Висновок

Обране рішення на основі мікросервісів у Kubernetes-кластері є найбільш підходящим для відеоконференцій через його масштабованість, надійність та гнучкість. Альтернативи, такі як моноліт чи serverless, можуть бути простішими, але не забезпечать такого рівня контролю над навантаженням і безпекою.