Додаток А

Слайди презентації

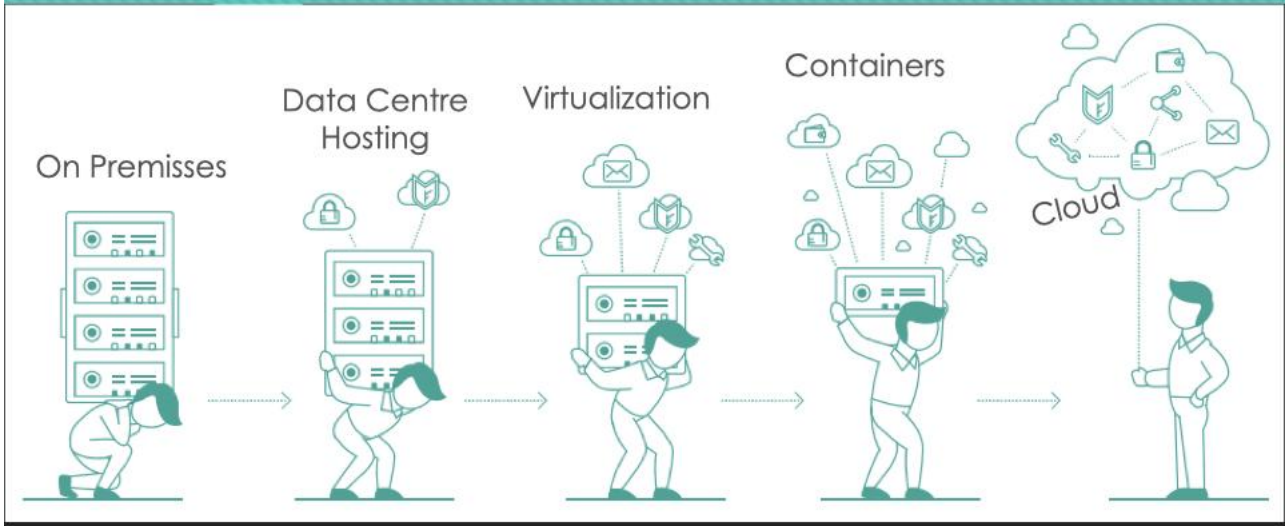
Актуальність тематики

- Обмежений бюджет розробки ПЗ
- Необхідно заздалегідь визначити навантаження системи та фінансові можливості клієнта
- Необхідно визначити організацію архітектури проекту з фінансової сторони розробки
- Ризик перенесення робочого навантаження в хмару
- Немає визначених критеріїв, за якими варто переносити обчислення у хмару

Мета роботи

- Дослідження методів оптимізації обчислень у хмарних технологіях
- Аналіз витрат на інфраструктуру
- Дослідити вірність вибору парадигми хмарних обчислень для застосування в продакшн-системах на основі критерій продуктивності і оплати послуг

Еволюція світу обчислень



Основні моделі надання послуг у хмарі

IaaS
Інфраструктура як Сервіс

PaaS
Платформа як Сервіс

SaaS
Програмне забезпечення як Сервіс

Модель багаторівневої хмарної мережі

Учасники бізнес-процесу

- провайдери інфраструктури (IaaS-provider)
- провайдери платформи (PaaS-provider)
- провайдери ПЗ і готових рішень (SaaS-provider)
- кінцеві абоненти послуг

	Критерии оптимизации	Бизнес-цели	Средства оптимизации
SaaS	1. Уровень качества обслуживания абонентов. 2. Затраты на ресурсы PaaS на одного абонента.	1. Поддержка заданного уровня качества обслуживания. 2. Снижение затрат на ресурсы PaaS.	1. Изменения топологии виртуализации PaaS 2. Настройка параметров виртуализации PaaS 3. Доработка ПО
PaaS	1. Уровень утилизации ВМ 2. Уровень качества обслуживания SaaS 3. Утилизация лицензий на БПО	1. Увеличение прибыли от платформы	1. Изменение количества ВМ 2. Изменения количества лицензий на БПО 3. Управление настройками ВМ
IaaS	1. Уровень утилизация ресурсов (процессоров, оперативной памяти, жесткого диска) 2. Уровень качества обслуживания PaaS	1. Утилизация аппаратного обеспечения	1. Распределение виртуальных машин 2. Изменение настроек ВМ 3. Изменение настроек сети

Підсумок етапу:
Підходи до розробки SaaS, що рекомендовані для успішного просування рішення

- ПЗ має підтримувати можливість надання послуг великій кількості (тисячам) абонентів
- При розробці розподіленої системи треба враховувати регіон дата-центру
- ПЗ має забезпечувати засоби поділу даних абонентів
- ПЗ має підтримувати можливості швидкого масштабування
- При розробці ПЗ має бути вирішено питання доступності до послуги відповідно до вимог SLA
- При розробці ПЗ повинна враховуватися швидкість виконання кожної транзакції
- ПЗ має надавати можливість кожному абоненту налаштовувати послугу відповідно до власних виробничих завдань, не змінюючи налаштування інших абонентів
- ПЗ має допускати глобальне використання без географічних обмежень, включаючи мовні

Основні моделі надання послуг

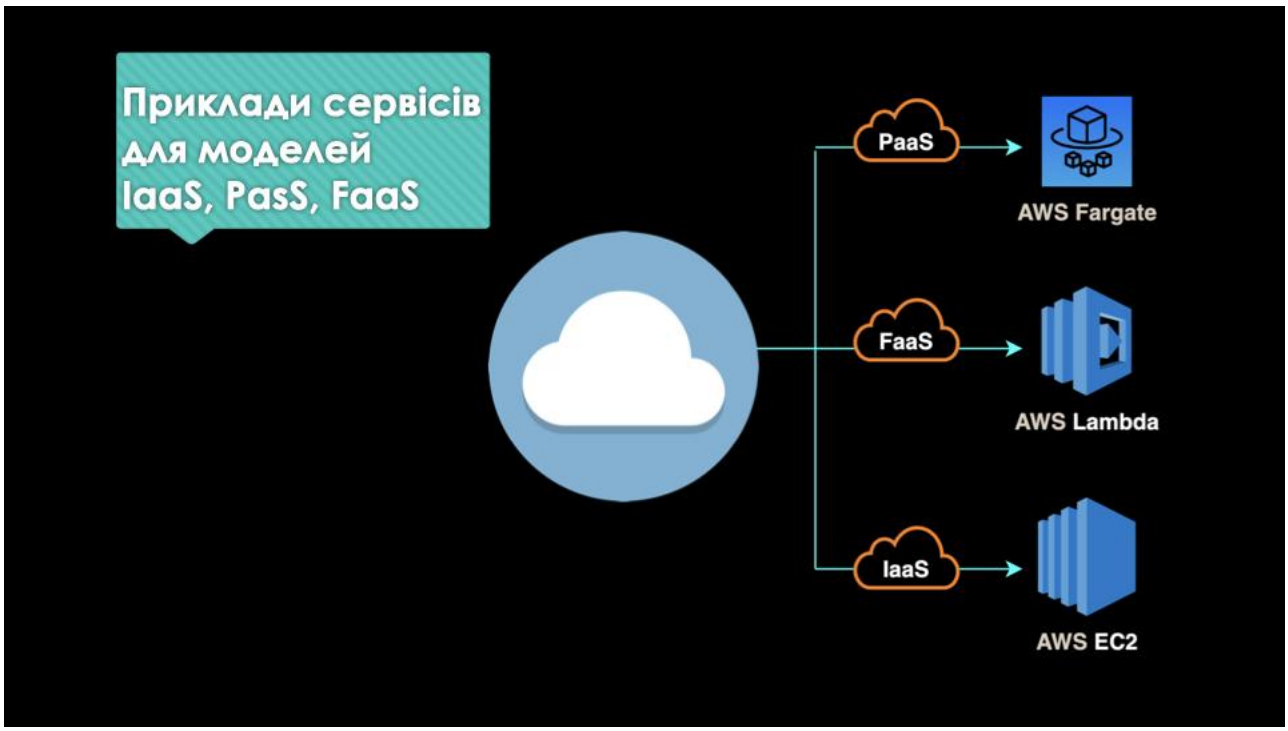
IaaS
Інфраструктура як Сервіс

PaaS
Платформа як Сервіс

FaaS
Функція як Сервіс

FaaS – Функція як сервіс

- Безсерверний запуск коду
- Розробники не турбується щодо обслуговування back-end серверів
- Обробка здійснюється на обчислювальних контейнерах, що не мають статусу
- Обробка починається лише з певною подією і завершується по закінченню обробки або по обмеженням часу



AWS Lambda

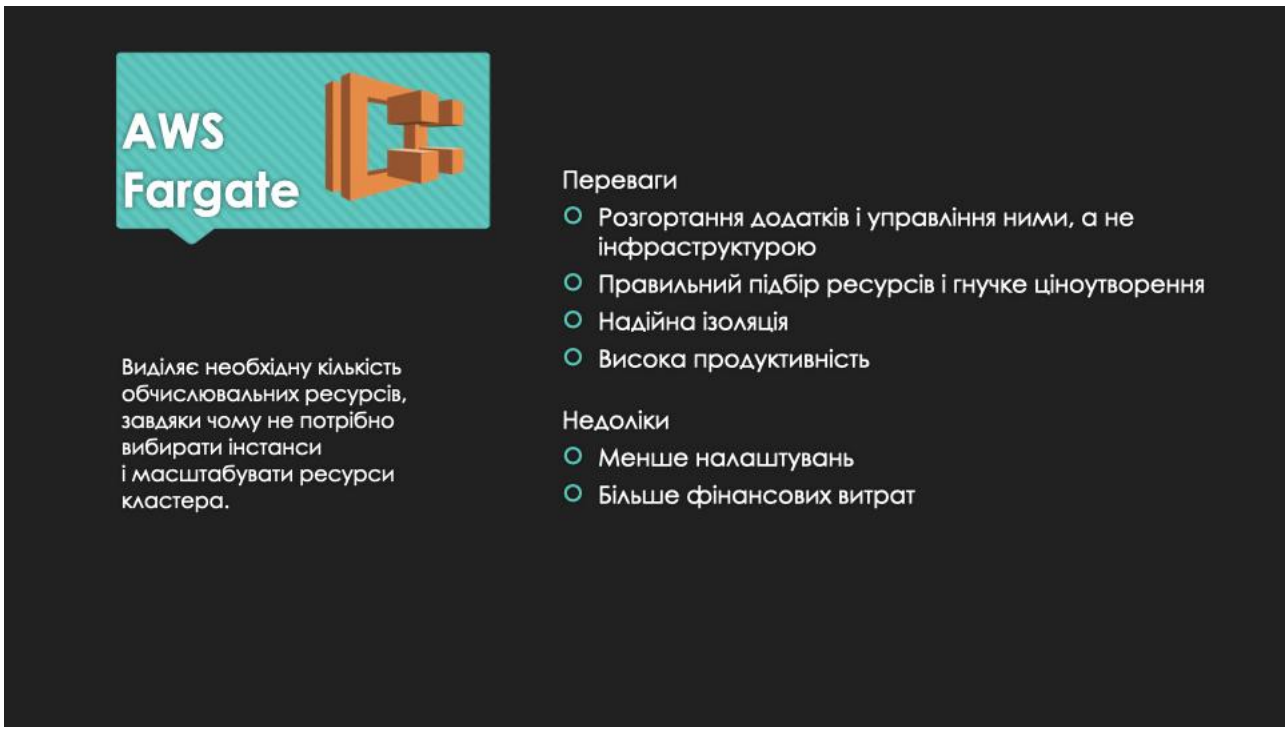
AWS Lambda дозволяє запускати програмний код без виділення серверів і керування ними. Оплаті підлягає тільки фактичний час виконання обчислень.

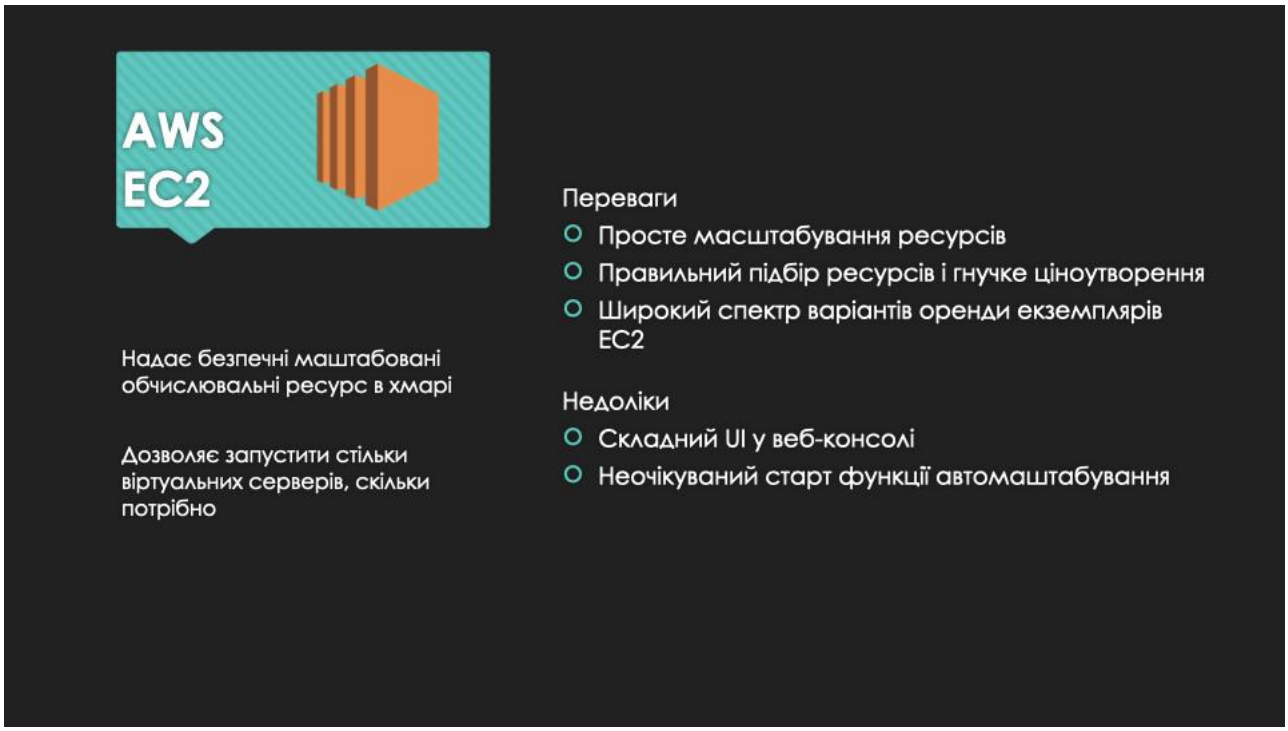
Переваги

- Відсутнє управління сервером
- Безперервне масштабування
- Висока продуктивність

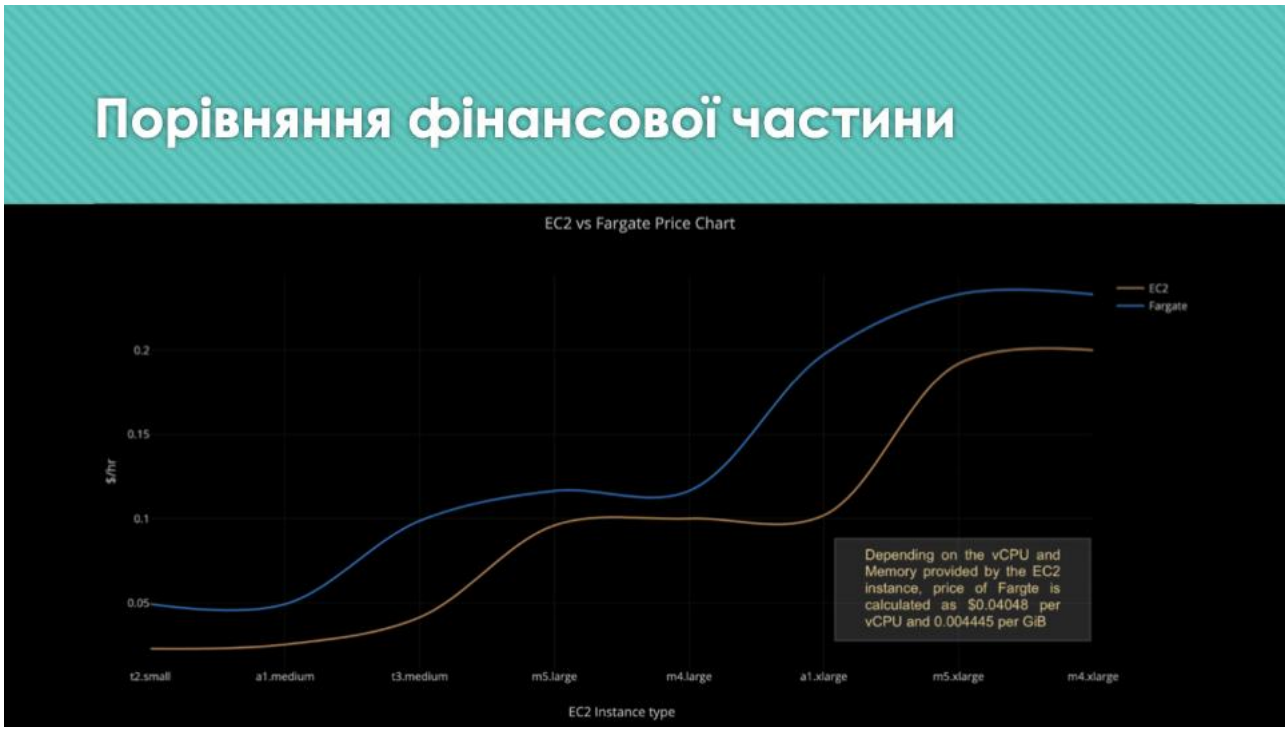
Недоліки

- Складні схеми викликів
- Відсутній контроль над навколишнім середовищем системи


AWS
Fargate
Виділяє необхідну кількість обчислювальних ресурсів, завдяки чому не потрібно вибирати інстанси і масштабувати ресурси кластера.
Переваги
Розгортання додатків і управління ними, а не інфраструктурою
Правильний підбір ресурсів і гнучке ціноутворення
Надійна ізоляція
Висока продуктивність
Недоліки
Менше налаштувань
Більше фінансових витрат


AWS
EC2
Надає безпечні маштабовані обчислювальні ресурс в хмарі
Дозволяє запустити стільки віртуальних серверів, скільки потрібно
Переваги
Просте масштабування ресурсів
Правильний підбір ресурсів і гнучке ціноутворення
Широкий спектр варіантів оренди екземплярів EC2
Недоліки
Складний UI у веб-консолі
Неочікуваний старт функції автомаштабування


Баланс між операційним навантаженням та гнучкістю сервісів AWS
Найбільше операційне навантаження
Найменше операційне навантаження
AWS EC2
AWS Fargate
AWS Lambda
Найбільша гнучкість налаштування
Найбільша гнучкість налаштування


Порівняння фінансової частини
EC2 vs Fargate Price Chart
EC2
Fargate
$/hr
0.2
0.15
0.1
0.05
Depending on the vCPU and Memory provided by the EC2 instance, price of Fargte is calculated as $0.04048 per vCPU and 0.004445 per GiB
t2.small
a1.medium
t3.medium
m5.large
m4.large
a1.xlarge
m5.xlarge
m4.xlarge
EC2 Instance type

Serverless підходи

Fargate

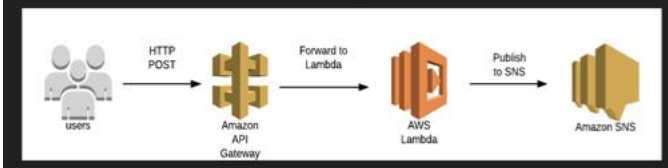
- Працює з Docker
- Оплата за використану пам'ять за годину, об'єм пам'яті обмежений процесором
- Варіант оптимізації – використання екземплярів EC2
- Автоматичний експорт метрик в CloudWatch

Lambda

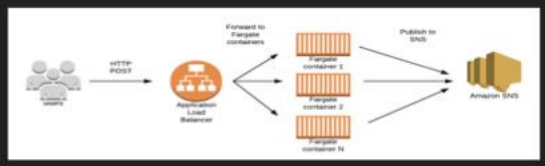
- Локальне налагодження функцій
- Оплата по кількості запитів, пам'яті та секунд виконання функції
- Варіант оптимізації – налаштування CloudWatch Billing Alarms
- Автоматичний експорт метрик в CloudWatch, Datadog, SumoLogic

Випробовування Serverless сервісів

Fargate



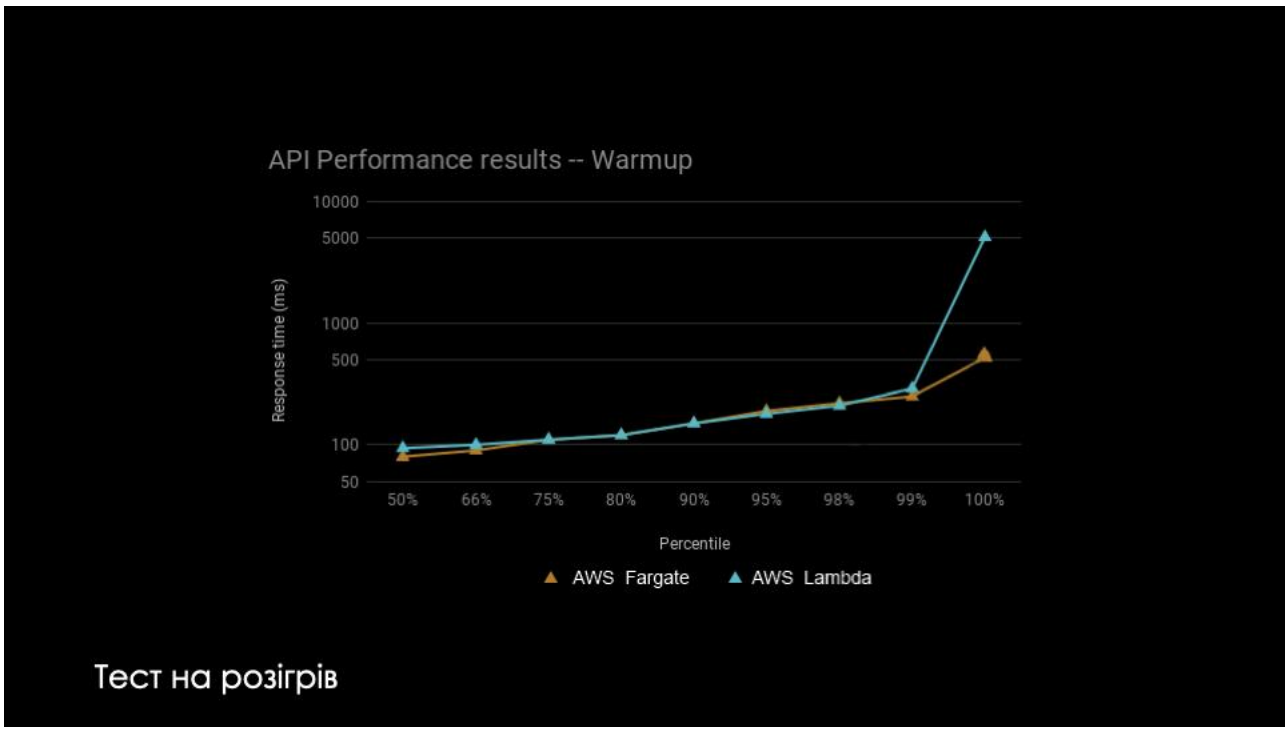
Lambda



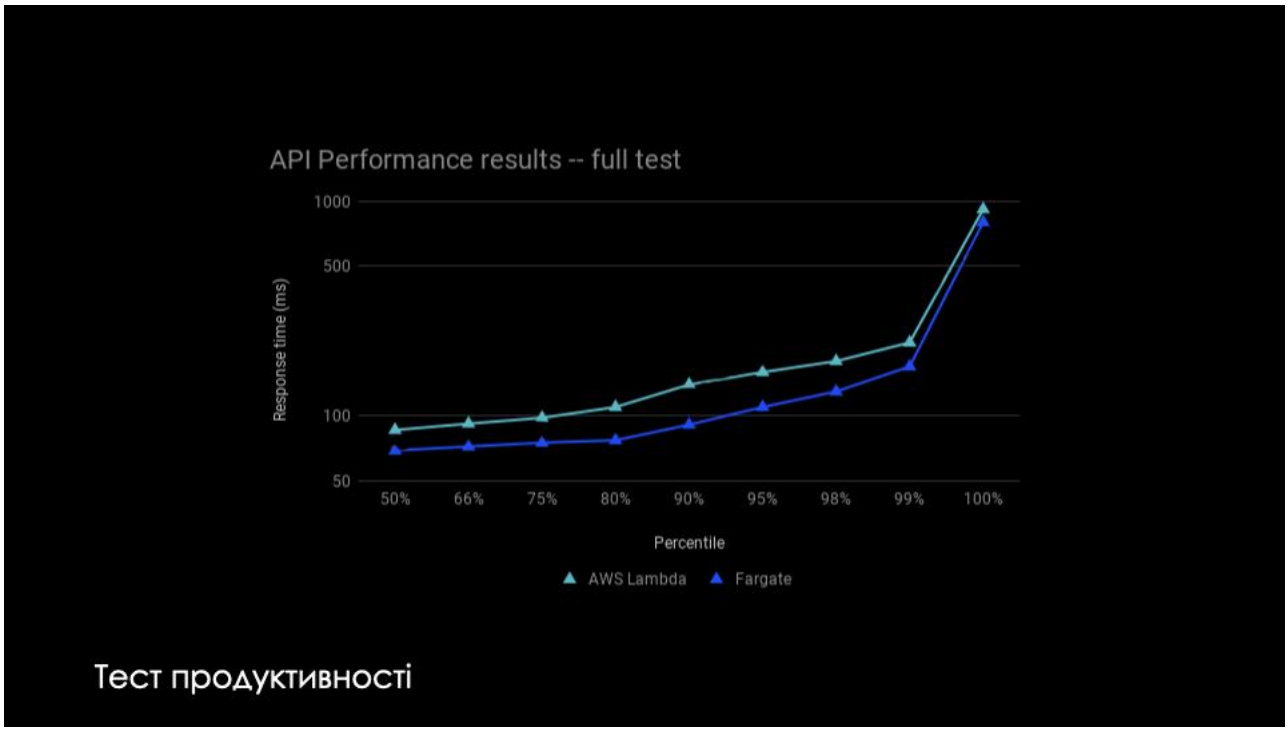
«Розігрів» інфраструктури
- Старт – 2 000 запитів, 40 запитів в секунду
- 15 000 запитів

Інтенсивність – 100 запитів в секунду

Тест продуктивності
- 15 000 запитів



Тест на розігрів



Тест продуктивності

Висновки

Висока продуктивність → **Fargate**
Більший комплекс налаштувань та управління інфраструктурою

Інші випадки → **Lambda**
Просте розгортання, надійний та легкий для маштабування

Задача роботи виконана
Було досліджено технології хмарних обчислень, шляхи оптимізації та визначено їх доцільність
Проведено комплексне дослідження, опис та аналіз для AWS Fargate та AWS Lambda
Для комерційного використання має враховуватись похибка у результатах дослідження

Дякую за увагу!

Доповідь завершено

Додаток Б

Лістинг

```
import json
import os

import boto3

TOPIC_ARN = os.environ['TOPIC_ARN']
sns = boto3.client('sns')


def ingest(event, context):
    payload = event['body']

    sns.publish(
      TopicArn=TOPIC_ARN,
      Message=payload
    )

    response = {
        "statusCode": 200,
        "body": json.dumps({ "message": "ok" })
    }

    return response
```