Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Інформаційні системи

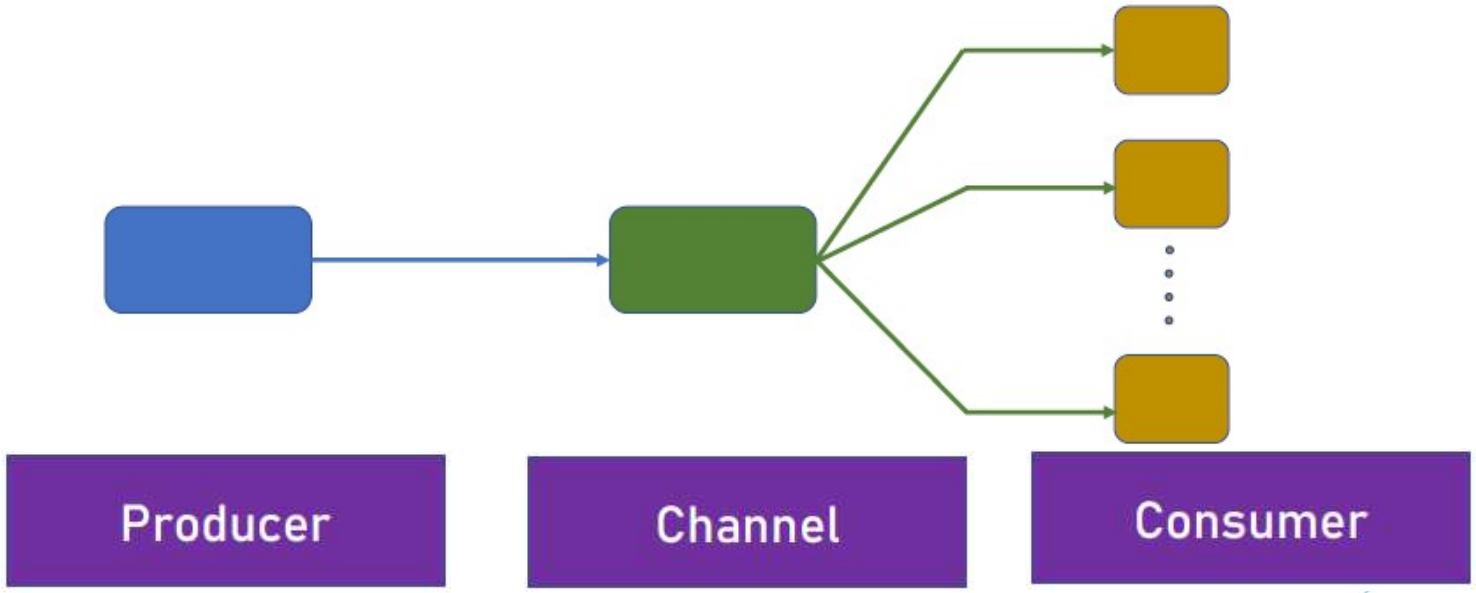
Викладач: к.т.н., доц. Саяпіна Інна Олександрівна

План заняття:

- Архітектура, керована подіями (Event Driven Architecture):
  - Архітектура, керована подіями (Event Driven Architecture)
  - Event Sourcing (постачання подій) та CQRS
  - Область використання EDA
  - Streaming & EDA
  - Логування та моніторинг
  - Впровадження EDA
- Розгортання мікросервісів

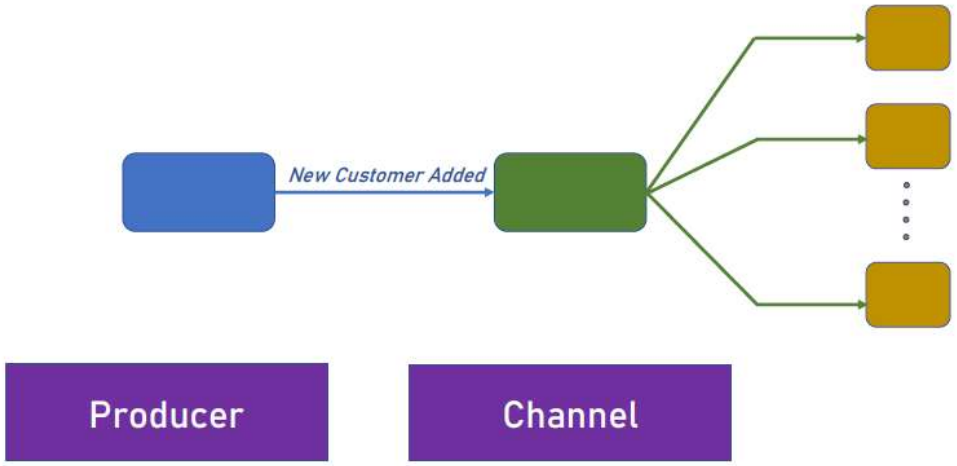
Event Driven Architecture (EDA)

- Парадигма архітектури програмного забезпечення, яка використовує події як засіб зв’язку між службами
- Має три основні компоненти:



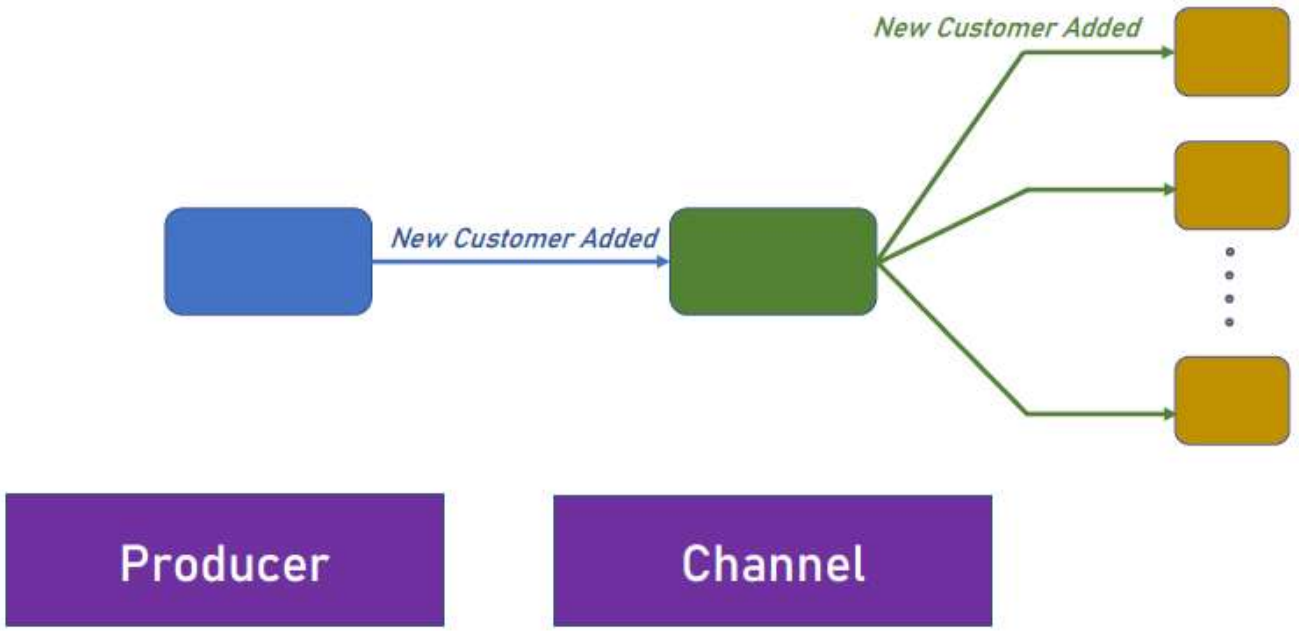
Producer

- Компонент/служба, що надсилає подію
- Часто називають Publisher
- Зазвичай подія повідомляє про щось, що зробив компонент
- Приклади:
  - Обслуговування клієнтів: --> Подія додавання нового клієнта
  - Служба інвентаризації: --> Товар розпродано



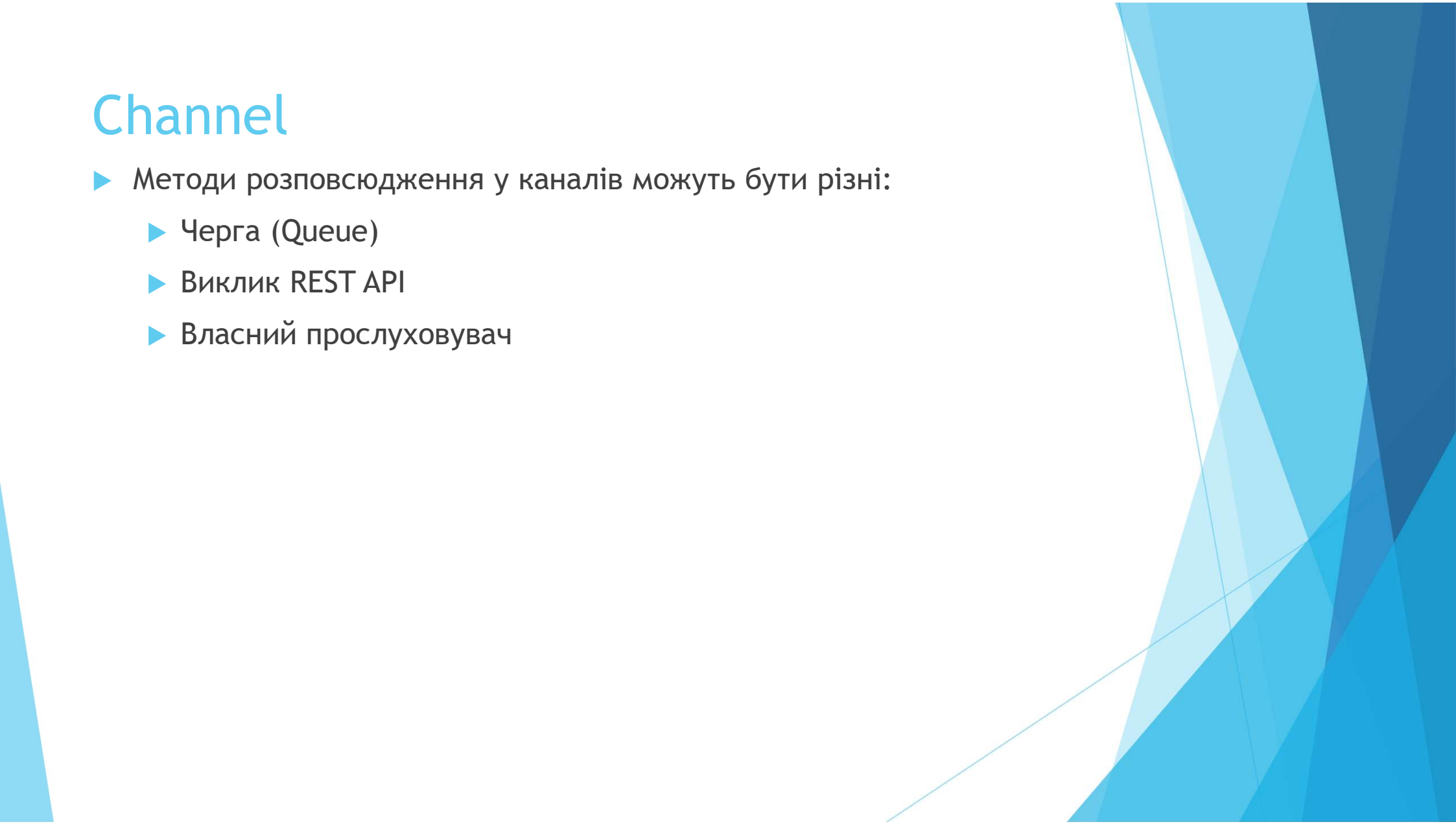
Channel

- Канал розповсюджує події для споживачів (Consumers)



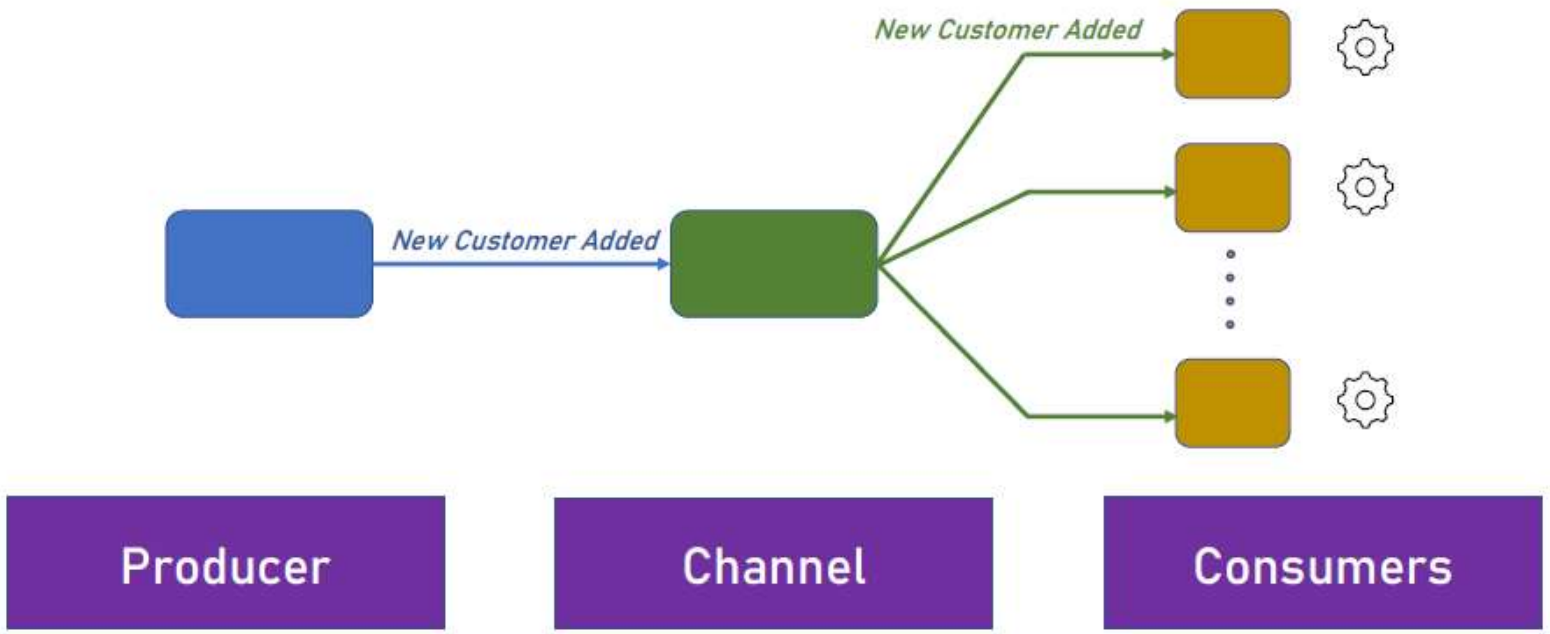
Channel

- Методи розповсюдження у каналів можуть бути різні:
  - Черга (Queue)
  - Виклик REST API
  - Власний прослуховувач



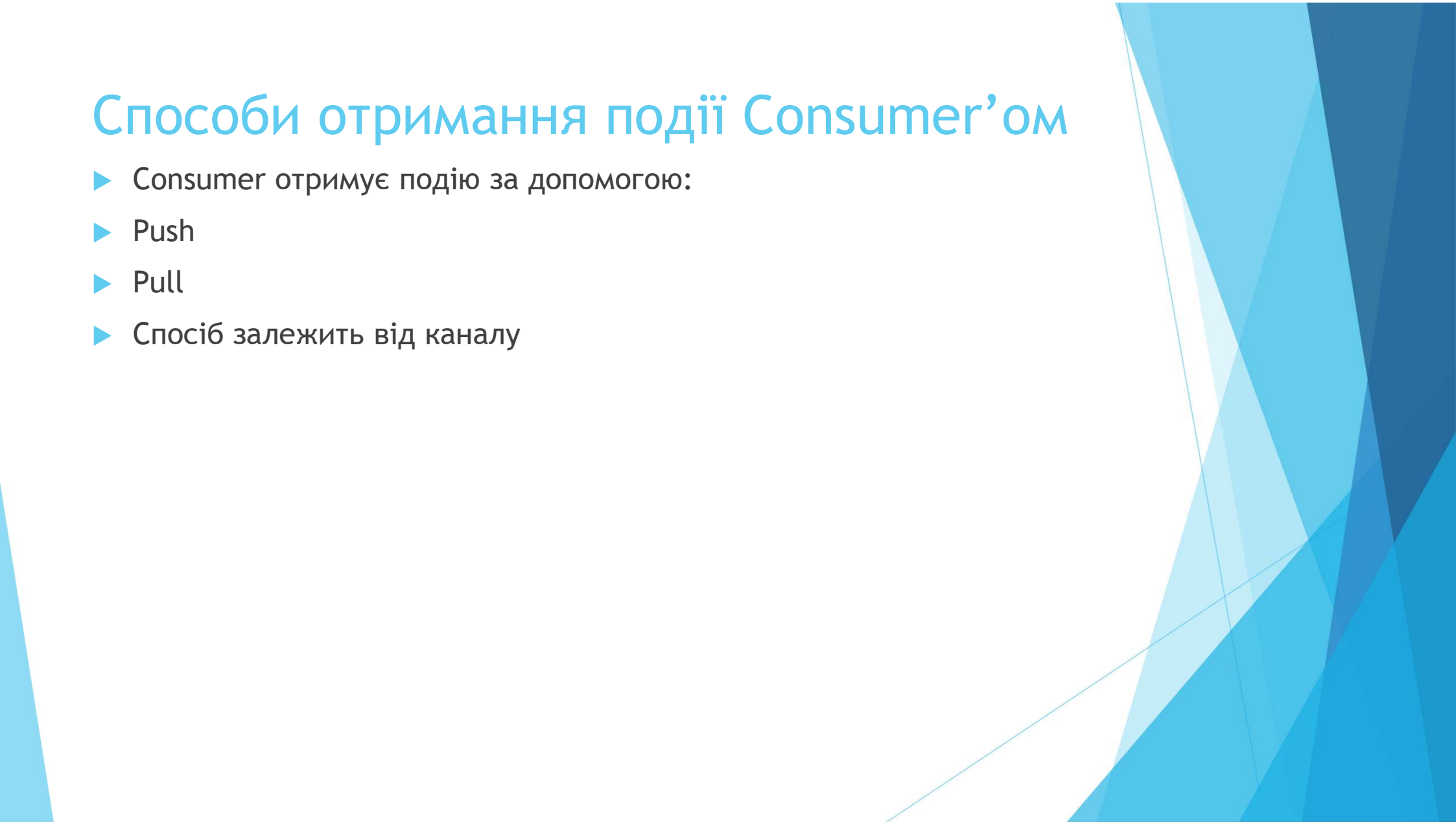
Consumer

- Consumer отримує та обробляє події



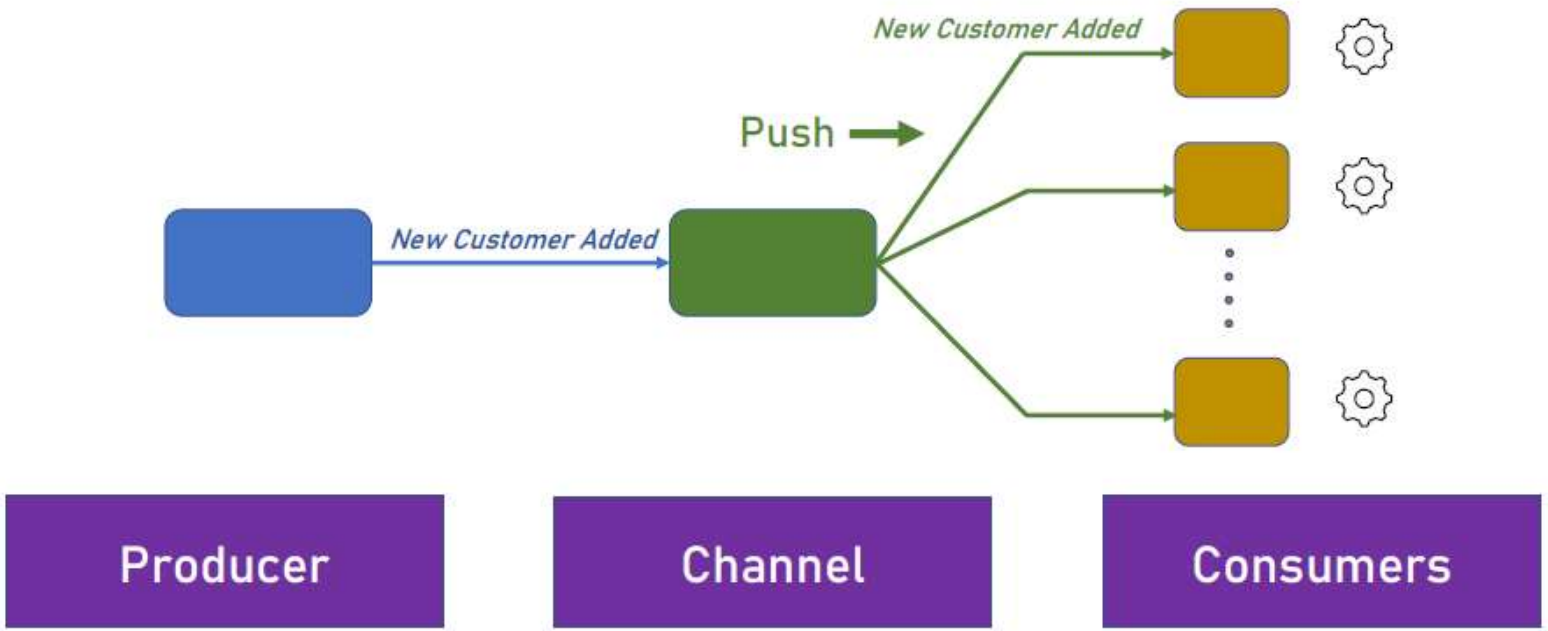
Способи отримання події Consumer’ом

- Consumer отримує подію за допомогою:
- Push
- Pull
- Спосіб залежить від каналу



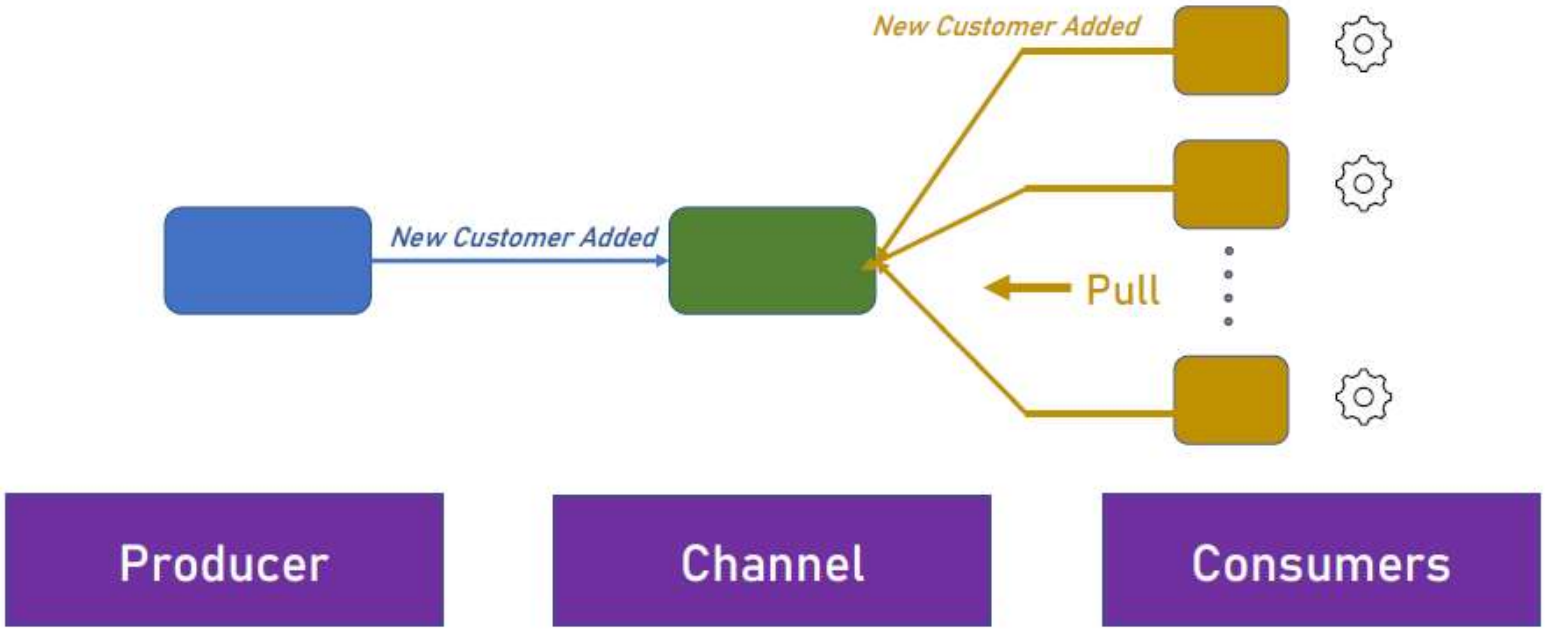
Push метод

- Consumer отримує подію за допомогою push - канал надсилає подію Consumer'ам



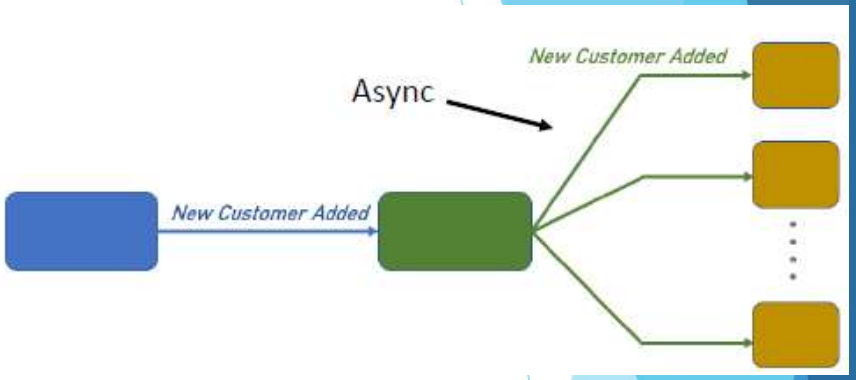
Pull метод

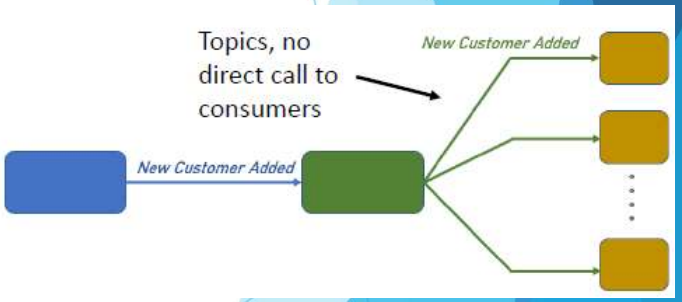
- Consumer отримує подію за допомогою pull - час від часу Consumer'и завертаються до каналу і питають чи є нові події для обробки



Переваги EDA

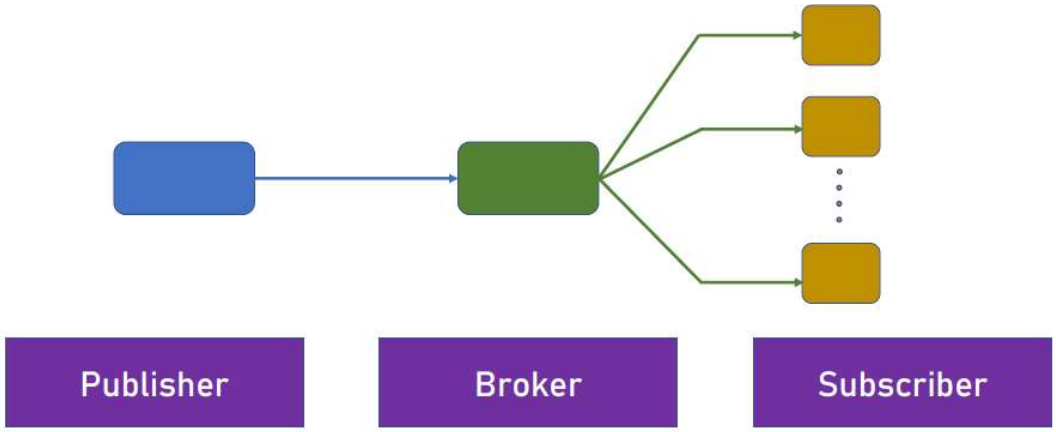
- Вирішені проблеми мікросервісів з
  - Продуктивністю
    - EDA — це асинхронна архітектура
    - Канал не чекає відповіді від споживача
    - Відсутність вузьких місць продуктивності
  - Зчепленням
    - Producer надсилає події на канал,
    - канал розподіляє події по темах (Topic) / чергах (Queue),
    - обидва не знають, хто слухає подію(за винятком WebHooks)
    - Немає зчеплення
  - Масштабованістю
  - Багато споживачів можуть слухати події з каналу, за потреби можна додати більше, Канал та Producer не зачіпаються при масштабуванні, Повна масштабованість





EDA та Pub/Sub

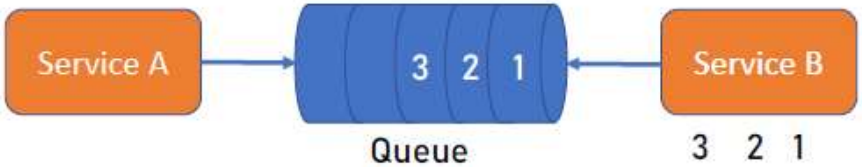
- Архітектура, керована подіями, часто згадується разом із Pub/Sub
- Pub/Sub = Publish/Subscribe Публікація та Підписка
- Шаблон обміну повідомленнями, який використовується архітектурою, керованою подіями
- Pub Sub компоненти: publisher, broker та subscriber.



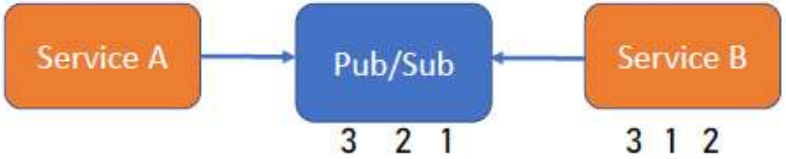
- Різниця: EDA описує всю архітектуру системи, а Pub/Sub — шаблон обміну повідомленнями, який використовує система

Порядок повідомлень в EDA та Pub/Sub

- Системи обміну повідомленнями часто гарантують порядок повідомлень



- З архітектурою, керованою подіями (особливо з Pub/Sub), порядок не завжди гарантований



- На порядок може вплинути затримка користувача, продуктивність коду тощо
- Якщо порядок надходження повідомлень важливий, переконайтеся, що вибрали канал, який це підтримує (Apache Kafka, RabbitMQ, Azure Service Bus)

Orchestration та Choreography.

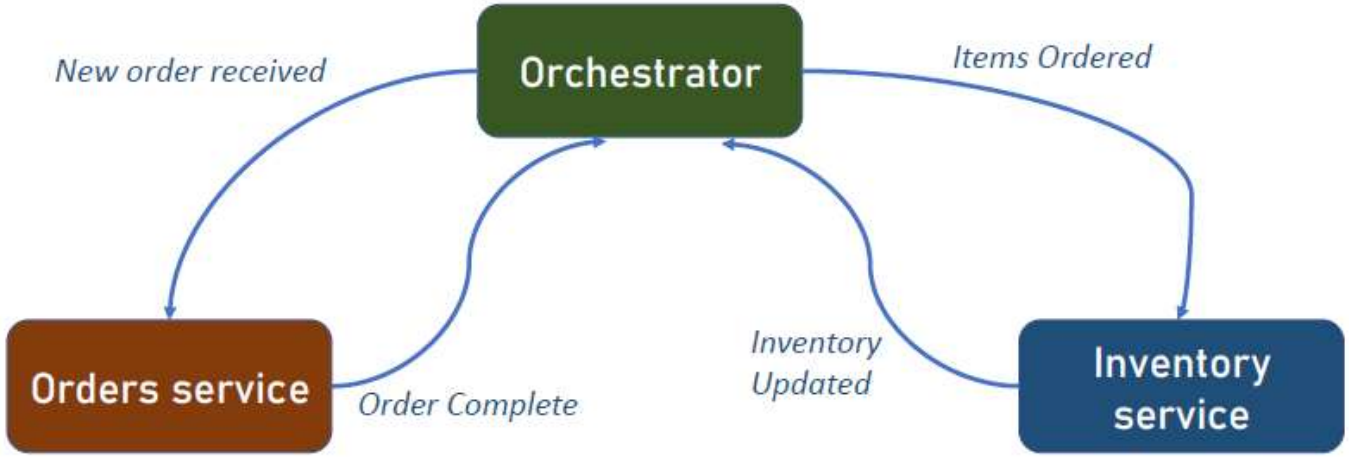
- Архітектура, керована подіями, зазвичай використовує один із двох архітектурних стилів
- Orchestration - оркестрація
- Choreography - хореографія.

Orchestration

- Оркестрація – це централізований підхід, коли існує єдиний компонент, який відповідає за контроль і координацію взаємодії між різними службами. Цей компонент називається оркестратором, і він виконує роль «диригента» загального процесу. Оркестратор контролює потік повідомлень між різними службами та відповідає за те, щоб процес виконувався правильно та в правильному порядку. Самі служби, як правило, не знають про більший процес і просто відповідають на повідомлення, які вони отримують від оркестратора.

Orchestration

- Потік подій у системі визначається центральним оркестратором
- Оркестратор отримує вихідні дані від компонентів і викликає наступний компонент у потоці
- Наступний компонент надсилає результат назад до оркестратора тощо.

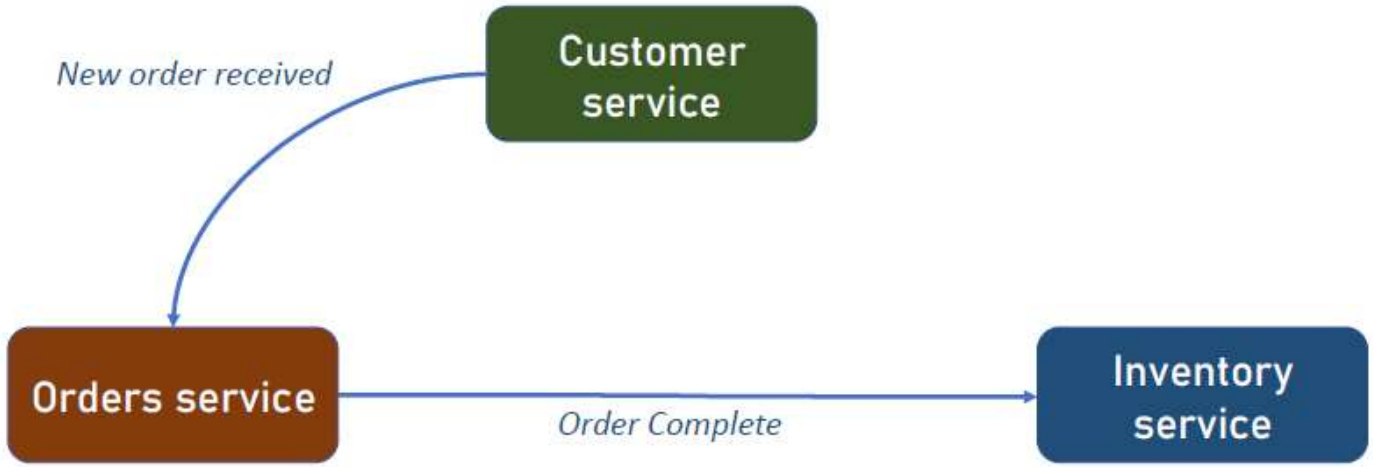


Choreography

- хореографія — це децентралізований підхід, коли кожна служба відповідає за координацію власної взаємодії з іншими службами. Немає центрального оркестратора, і служби спілкуються один з одним напряму. Кожна служба «знає», що їй потрібно робити на основі повідомлень, які вона отримує від інших служб, і вона відповідає за координацію власної поведінки. Цей підхід часто називають «танцем», оскільки кожна служба виконує власний набір кроків, а загальний процес виникає на основі взаємодії між службами.
- Обидва підходи мають свої сильні та слабкі сторони, і вибір між ними залежить від конкретних вимог системи.

Choreography

- Немає центрального компонента, який знає все про процес
- Кожен компонент повідомляє про статус подій
- Інші компоненти прислухаються до подій і діють відповідно



Orchestration та Choreography

Orchestration

- Логіка зосереджена в одному місці, її легше підтримувати
- Центральний шлюз трафіку полегшує моніторинг і логування
- Більше підходить для складних процесів, які вимагають високого ступеня координації та контролю

Choreography

- Немає впливу посередників на продуктивність
- Надійність (якщо один компонент виходить з ладу, решта все одно працюють)
- Більше підходить для систем, які є більш децентралізованими та гнучкими

Приклади систем на основі EDA

- Система, яка обробляє клацання користувачами на веб-сайті. У цій системі кожна подія клацання обробляється як подія та обробляється набором мікросервісів, які обробляють подію та виконують відповідні дії. Якщо користувач натискає кнопку, щоб купити продукт, подія кліку обробляється мікросервісом, який обробляє замовлення, мікросервісом, який обробляє платіж, і мікросервісом, який оновлює кількість товару на складі. Це дозволяє системі бути високомасштабованою та стійкою, оскільки вона може легко обробляти велику кількість подій паралельно та відновлюватися після збоїв.
- Платформа для торгівлі акціями. У цій системі такі події, як зміни цін на акції, угоди, здійснені користувачами, і новини, що впливають на ринок, можуть розглядатися як події. Ці події можуть ініціювати відповідні дії, такі як надсилання сповіщень користувачам, оновлення торгових позицій і виконання угод.

Загальна оцінка

Architecture characteristic	Star rating
Partitioning type	Technical
Number of quanta	1 to many
Deployability	★★★
Elasticity	★★★
Evolutionary	★★★★★
Fault tolerance	★★★★★
Modularity	★★★★
Overall cost	★★★
Performance	★★★★★
Reliability	★★★
Scalability	★★★★★
Simplicity	★
Testability	★★

Особливості використання EDA

- Масштабованість не є проблемою
- За потреби можна додавати нових споживачів без змін в архітектурі
- Чудово підходить для змінного навантаження
- Розглядайте EDA, якщо зв’язок між службами може бути асинхронним
- Архітектуру, керовану подіями, непросто реалізувати
- Вимагає встановлення та налаштування каналів
- Особливості логування та моніторингу
- EDA використовує багато трафіку
- Мережа має бути надійною, інакше продуктивність буде низькою
- EDA не підходить для: невеликих систем з кількома сервісами, Синхронно орієнтованих систем (тобто систем, які обслуговують в основному запити кінцевих користувачів)

Event Sourcing and CQRS

- Події також можна використовувати як основні будівельні блоки даних
- Event Sourcing і CQRS пропонують шаблон для зберігання даних як подій
- Традиційні бази даних містять дані про поточний стан сутності. Наприклад, сутність співробітники:

emp_id	first_name	last_name	address	role	date_join
1	John	Smith	Beverly Hills 90210	Development Manager	2009-04-23
2	Sarah	Jones	42nd st. NYC	Sales	2019-01-30
3	Britney	Flyn	Marigold Lane, Boca Raton	HR	2022-05-19

- Немає інформації про попередні події та зміни, що відбувались

Event Sourcing

- Шаблон сховища даних, у якому кожна зміна даних фіксується та зберігається
- База даних зберігає список змін для сутності, а не саму сутність
- Жодних оновлень чи видалень, лише вставки
- Кожен рядок документує зміну властивостей сутності
- У цьому шаблоні база даних називається Event Store

event_id	timestamp	event
1	2009-04-23	Employee John Smith joined
2	2009-04-23	Address of John Smith updated to Hott Street, Clinton
3	2009-04-23	Role of John Smith updated to Junior Developer
4	2013-05-22	Address of John Smith updated to Beverly Hills 90210
5	2017-09-12	Role of John Smith updated to Development Manager
6	2019-01-30	Employee Sarah Jones joined
7	2019-01-30	Role of Sarah Jones updated to Sales

Event Sourcing

Переваги

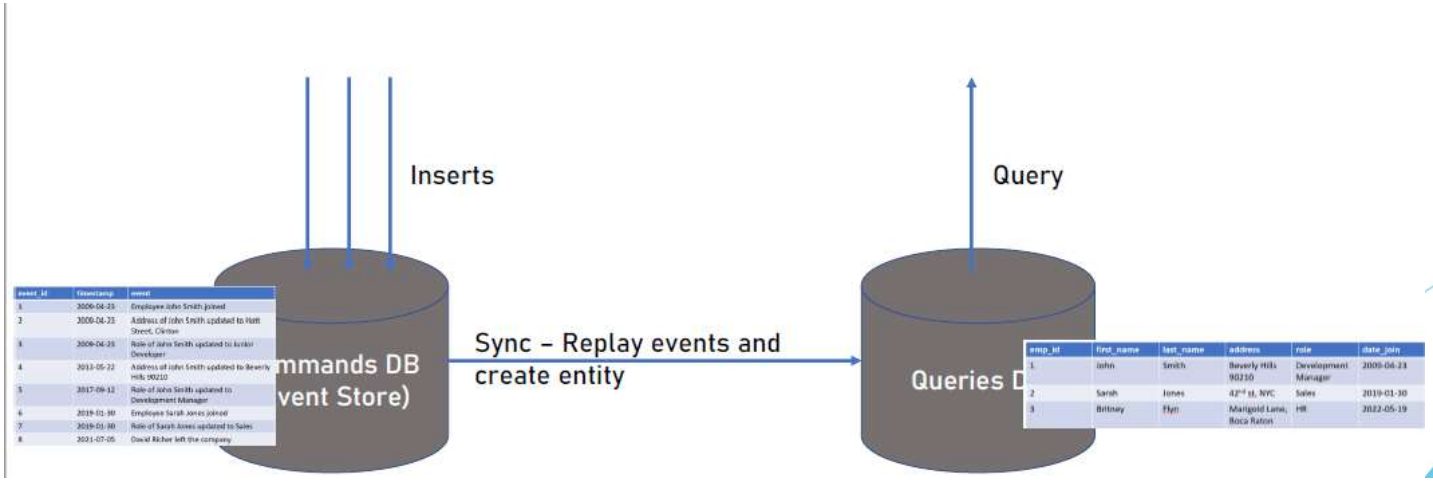
- Надзвичайно простий перегляд історичних даних
- Проста структура бази даних
- Прості операції з базою даних (без оновлень, без паралелізму)
- Дуже швидкі вставки

Недоліки

- Перегляд поточного стану об’єкта громіздкий і повільний
- Велика ємність бази даних (багато записів на сутність)

CQRS - Command and Query Responsibility Segregation

- Відокремлення команд (оновлення / вставка / видалення) від запитів
- Окремі бази даних для команд та запитів
- База даних команд реалізована як сховище подій для підвищення продуктивності та простоти
- База даних запитів зберігає сутності
- База даних синхронізується за допомогою центрального механізму синхронізації

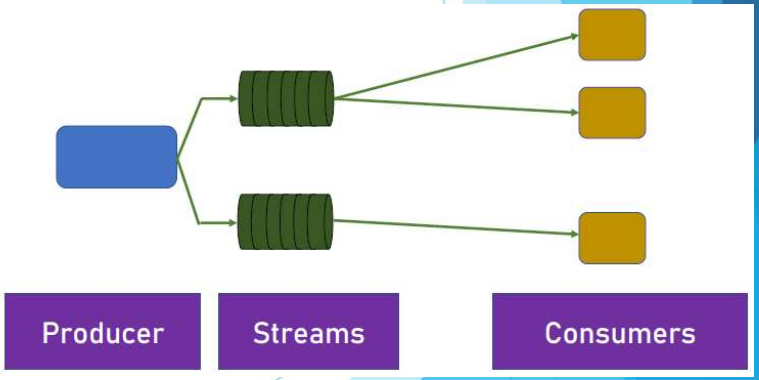


Коли використовувати Event Sourcing і CQRS

- Коли доступ до історичних даних надзвичайно важливий
- Регулювання, фінанси, охорона здоров'я тощо.
- Коли дані великі і повторне відтворення подій неможливе
- Коли продуктивність критична (вставки або запити)

Event Streaming

- **Event Streaming (**Потокова передача подій) — ще один шаблон, Орієнтований на події
- Механізми **Event Streaming** публікують події у потік подій “stream”
- Наприклад:
- Телеметрія з датчиків, системні логи тощо.
- Події публікуються в “stream”
- Споживачі підписуються на певний stream
- Події зберігаються в stream’і протягом певного часу



Event Streaming

- Споживачі можуть отримати події, які були надіслані в минулому (зазвичай до кількох днів)
- Streaming Engine можна використовувати як центральну базу даних (єдине «джерело істини»)
- Не всі події обов’язково обробляються
- Деякі з них можуть бути неактуальними
- Найрозповсюдженійший Event Streaming tool

Event Streaming vs EDA

Event Streaming

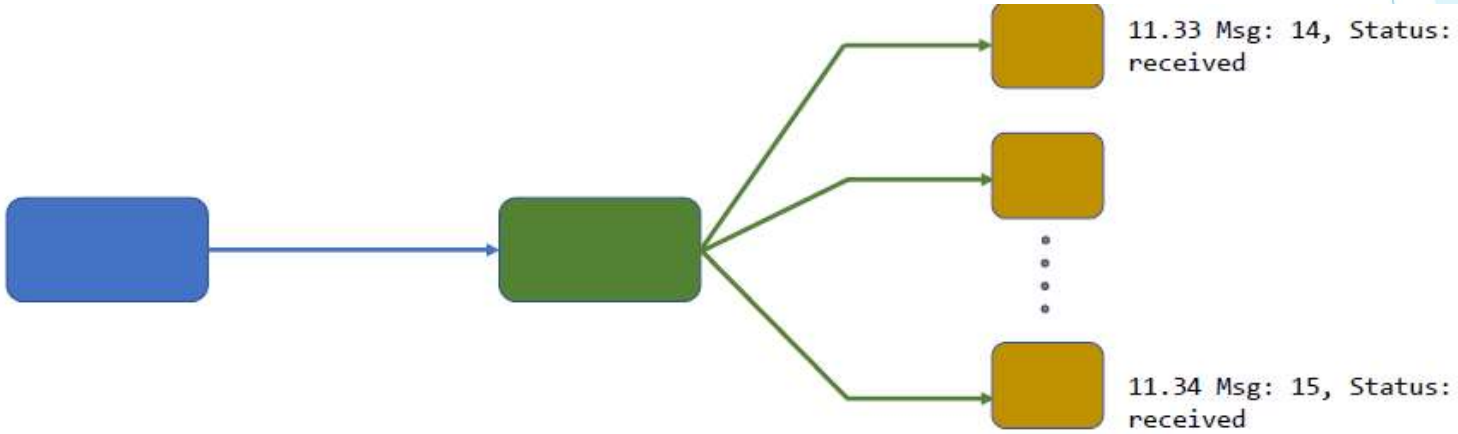
- Зазвичай використовується для подій, створених ззовні системи (Дані датчиків, логи)
- Події зберігаються
- Не всі події обробляються
- Передбачене високе навантаження

EDA

- Зазвичай використовується для подій, що відбуваються всередині системи
- Події не зберігаються
- Всі події обробляються
- Немає високого навантаження

Проблеми логування в EDA

- В EDA є розподілені, незв’язані компоненти
- Отримати уніфікований, хронологічно впорядкований лог важко
- Формати логів різні
- Мітки часу не синхронізовані



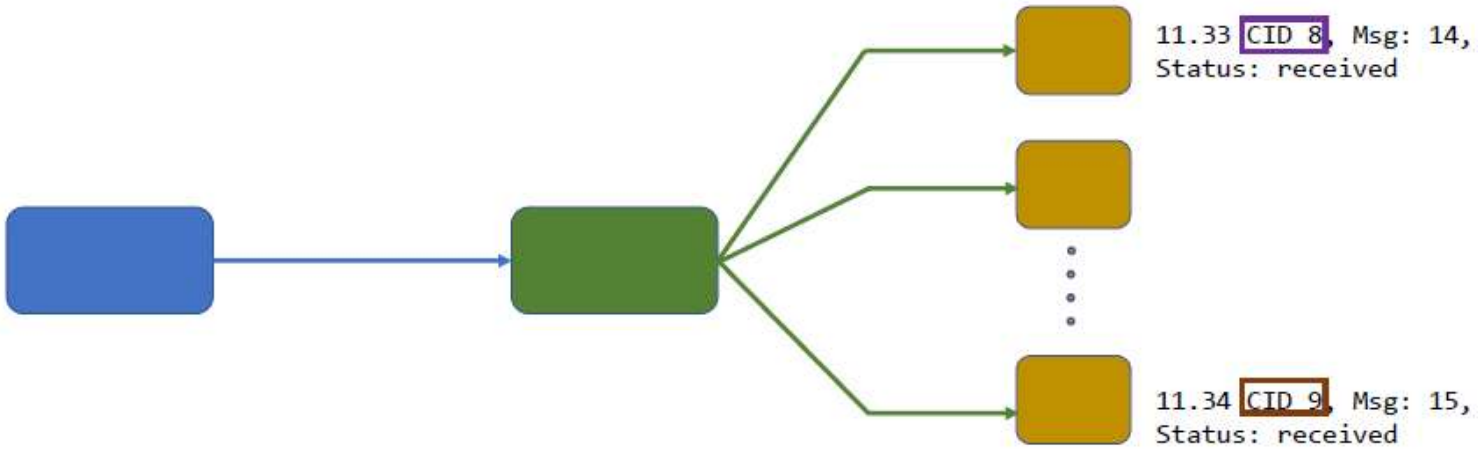

```
11.28 Msg 14 was sent
11.30 Msg 15 was sent
```

```
11.26 Receiving msg 14...
11.26 Handling msg 14...
11.28 Sending msg 14
11.29 Receiving msg 15...
```

Different formats | Timestamps not synced | Different destinations | Many records, hard to trace

Correlation Id

- На початку кожної транзакції до повідомлення додається унікальний ідентифікатор
- Цей ідентифікатор реєструється як частина запису логу
- Дозволяє відстежувати лог між компонентами




```
11.28 CID 8, Msg 14 was sent
11.30 CID 9, Msg 15 was sent
```

```
11.26 CID 8, Receiving msg 14...
11.26 CID 8, Handling msg 14...
11.28 CID 8, Sending msg 14
11.29 CID 9, Receiving msg 15...
```

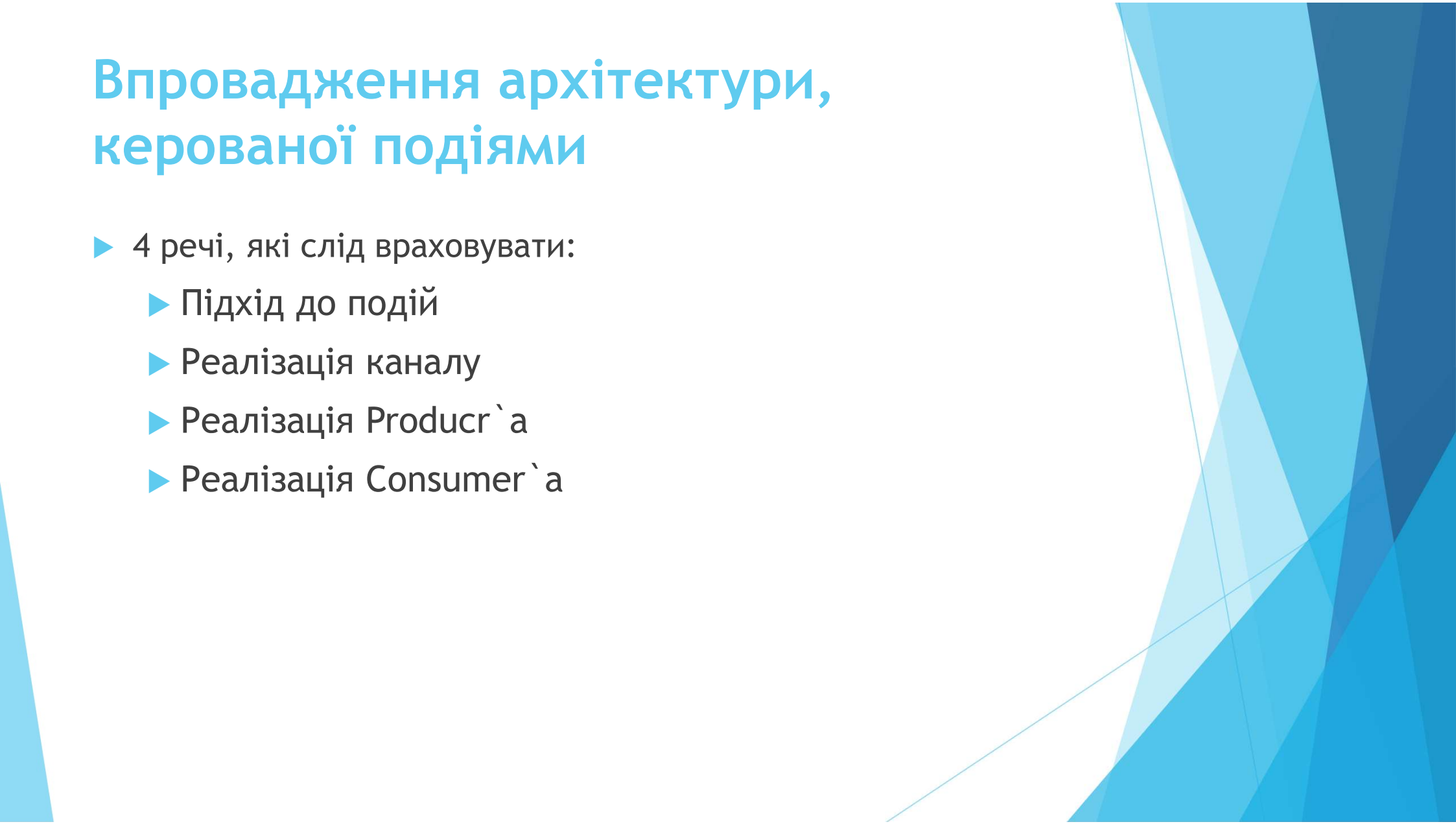
Централізована система логування (Central Logging Engine)

- Централізована система, яка використовується для агрегування всіх записів логу
- Має конвертери, які перетворюють формати логів
- Чудові можливості аналітики та візуалізації
- Можна використовувати існуючий канал для транспортування логів
- Багато існуючих розроблених систем. Наприклад, Elastic Stack.

- Слід врахувати:
  - Логи повинні відображати поведінку системи, а не лише помилки!
  - Має бути можливість відтворити транзакцію за допомогою логів

Впровадження архітектури, керованої подіями

- 4 речі, які слід враховувати:
  - Підхід до подій
  - Реалізація каналу
  - Реалізація Producr`a
  - Реалізація Consumer`a



Підхід до подій

- **Зберігання подій**
- Канал зберігає подію для подальшої обробки
- Визначається період зберігання, після закінчення якого подія видаляється
- Чудово підходить для трансляції подій і коли канал є джерелом істини

- **Події не зберігаються**
- Канал публікує події, а не зберігає їх
- Якщо споживач пропустив подію, її неможливо відтворити
- Використовується в основному для системних подій

Реалізація каналу

- **Події зберігаються**
- Використовується механізм обміну повідомленнями/чергами
- Поширені двигуни:-RabbitMQ-Apache Kafka

- **Події не зберігаються**
- Використовується Event Publisher
- Конкретний механізм залежить від використовуваної платформи, типів інтерфейсів тощо

Реалізація каналу

- **Події не зберігаються**
- Використовується Event Publisher
- Конкретний механізм залежить від використовуваної платформи, типів інтерфейсів тощо

- Вибір Event Publisher

Use…	When…
Azure Event Grid or similar	- Hosted in the cloud - Need strong integration between backend services
WebHooks	- Receivers expose REST API - Need something simple and quick
HTTP Push Notification	- Need to notify the end user

SignalR, Socket.IO, gRPC

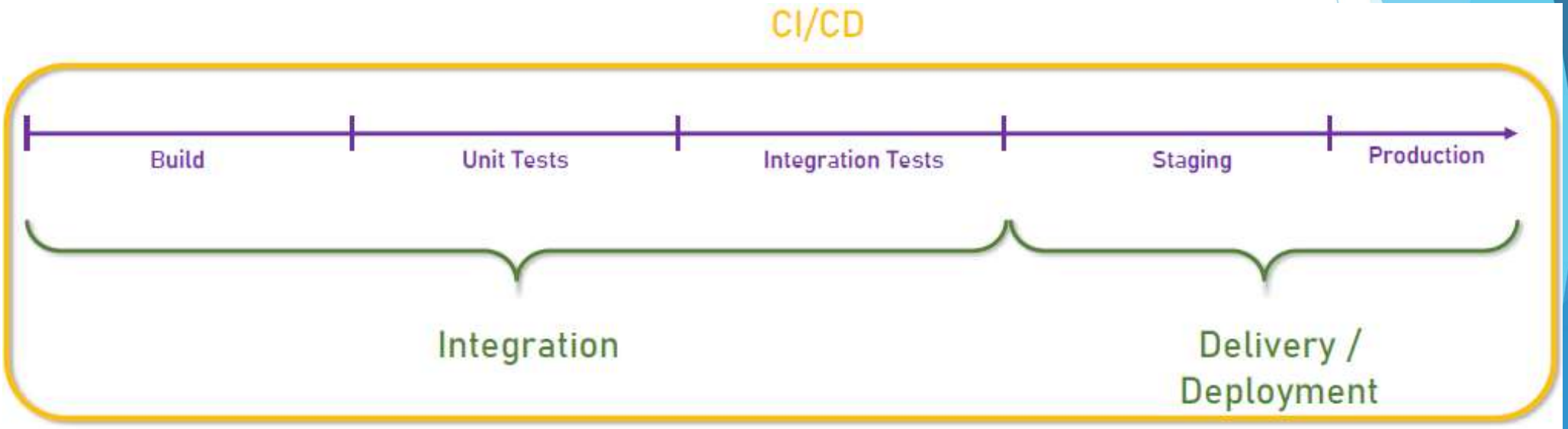
Розгортання мікросервісів

- Розгортання мікросервісів є надзвичайно важливим
- Важливість автоматизації інфраструктури
- Повільне та складне розгортання зробить всю систему неефективною та марною
- Архітектор повинен розуміти розгортання, але він не є відповідальним за нього

CI/CD (continuous integration and continuous delivery/continuous deployment)

- Забезпечує:
- Безперервну інтеграцію / Безперервну доставку
- Повну автоматизацію етапів інтеграції та доставки

Інтеграція та доставка



Що надає використання CI/CD?

- Швидший цикл випуску
- Надійність
- Звітність

CI/CD Pipelines (CI/CD конвейери)

- Основа процесу CI/CD
- Визначають набір дій для виконання в рамках CI/CD
- Зазвичай визначаються за допомогою YAML із представленням інтерфейсу користувача