

텍스토리 MSA 발표

별소프트 권영택

목표

- 학습 목적으로 MSA 시스템 구성 경험
- Kubernetes 적극 활용, 서비스/데이터/모니터링 전체 흐름 체험
- React 실습
- 서비스별 DB 분리 경험
- Redis를 활용하여 세션 방식 구성 (세션 중앙화)
- Kafka를 활용한 이벤트 처리 경험

주요 기능

User service	로그인/회원가입	친구 찾기/등록
File service	파일 업로드	파일 뷰 (src URL)
Posting service	포스트 글 올리기	포스트 목록 조회
Notification service	알림 목록	알림 기능 (웹소켓)

주요 페이지

/login

홈으로

로그인

계정에 로그인하여 서비스를 이용하세요

이메일

test1@gmail.com

비밀번호

로그인

계정이 없으신가요? 회원가입

/signup

홈으로

회원가입

새로운 계정을 만들어 서비스를 이용하세요

이메일

example@email.com

비밀번호

비밀번호를 입력하세요

비밀번호 확인

비밀번호를 다시 입력하세요

닉네임

닉네임을 입력하세요

생년월일

년-월-일

핸드폰번호

010-1234-5678

회원가입

이미 계정이 있으신가요? 로그인

/

신짱구

What's on your mind?


Post

Video Photo Feeling

철수 2일 전

학원가는중

신짱구 1일 전



오도방구~

신짱구 1일 전

오옷?

신짱구 2일 전

물루랄라
떡잎방범대

유리

/friends

신짱구

친구 찾기

test

검색

검색 결과

유리 (test04@gmail.com)

친구

철수 (test03@gmail.com)

친구

봉미선 (test01@gmail.com)

친구

신용만 (test02@gmail.com)

친구

신짱구 (test1@gmail.com)

(나)

/notification

신짱구

알림

전체 읽지 않음

이전 주

유리 님이 게시글을 올렸습니다.

읽음

봉미선 님이 게시글을 올렸습니다.

읽음

신용만 님이 게시글을 올렸습니다.

읽음

철수 님이 게시글을 올렸습니다.

읽음

유리 님이 회원님과 친구가 되었습니다.

읽음

/profile

신짱구

현재 프로필 이미지



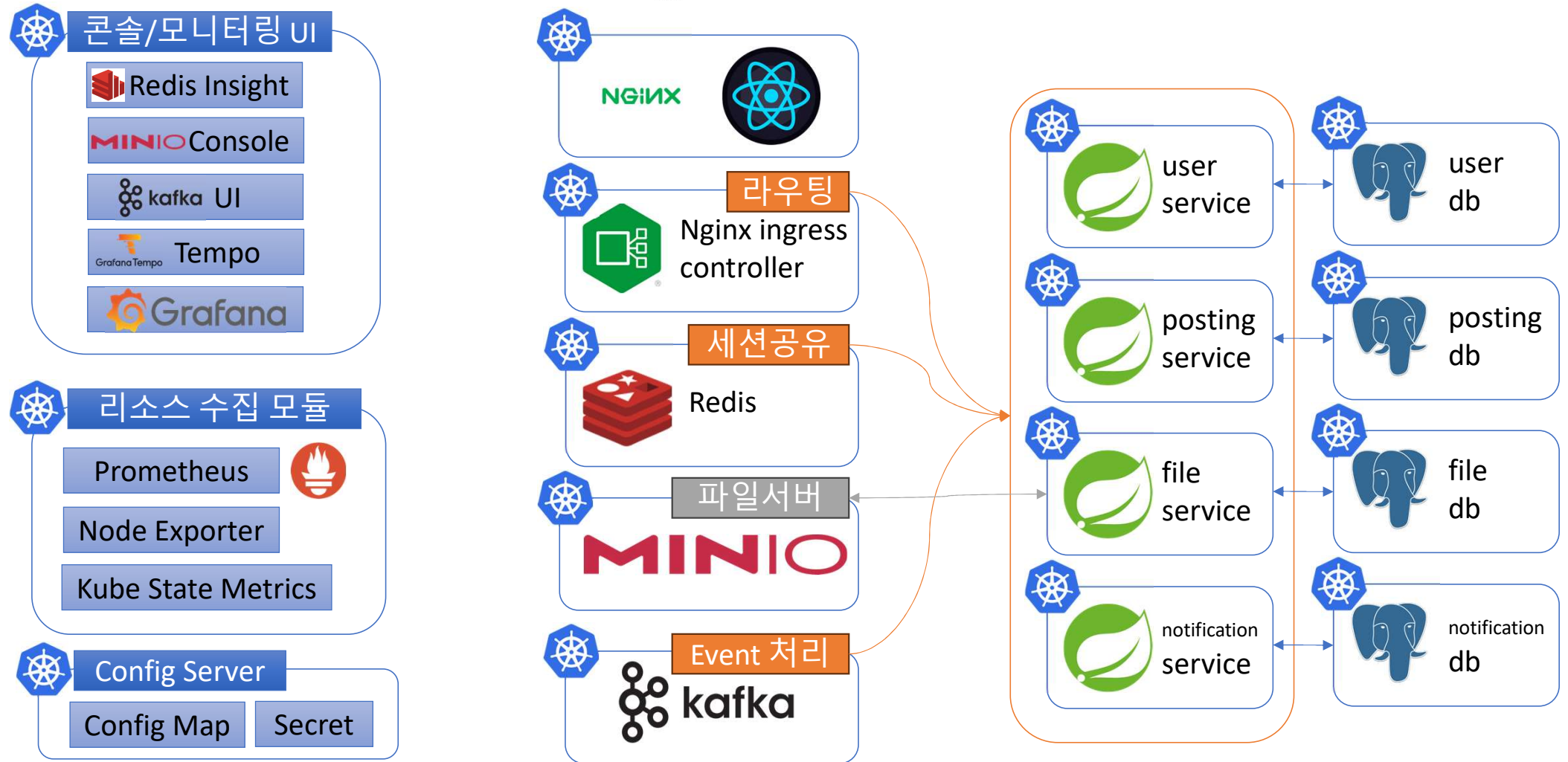
프로필 이미지 업로드

파일 선택

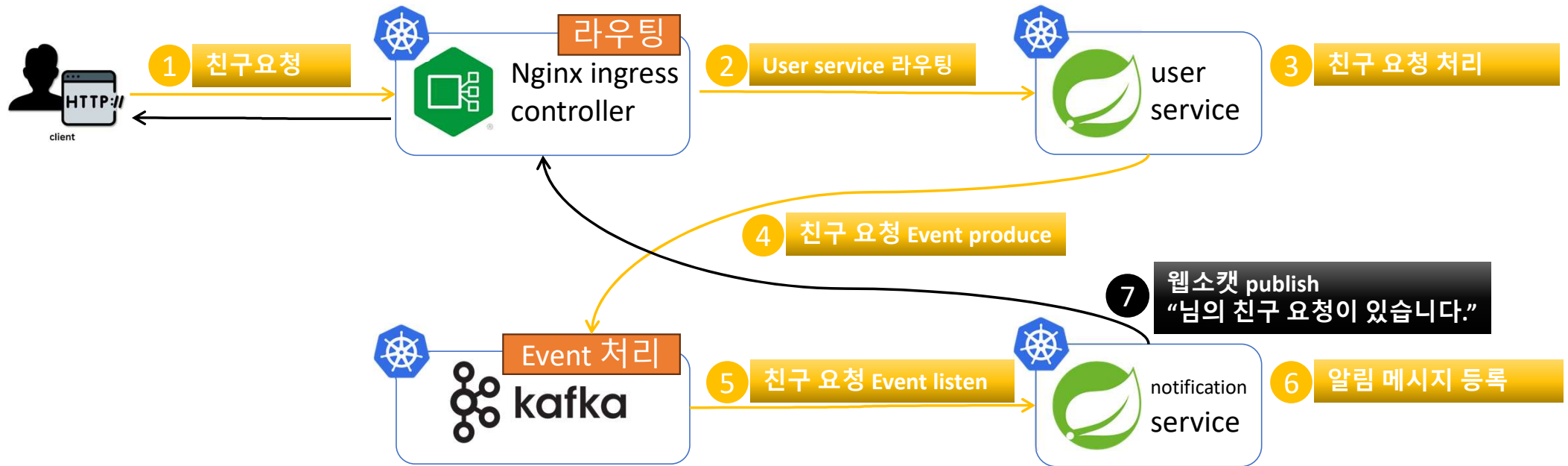
선택된 파일 없음

업로드 취소

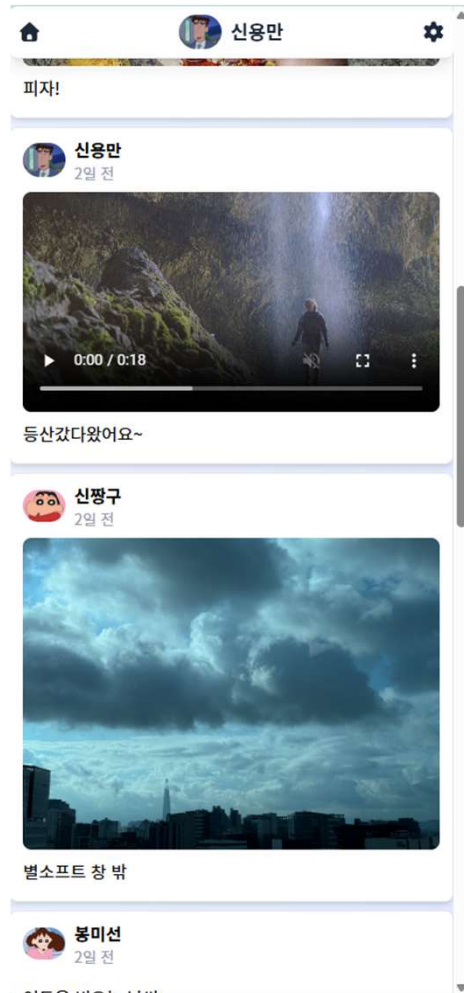
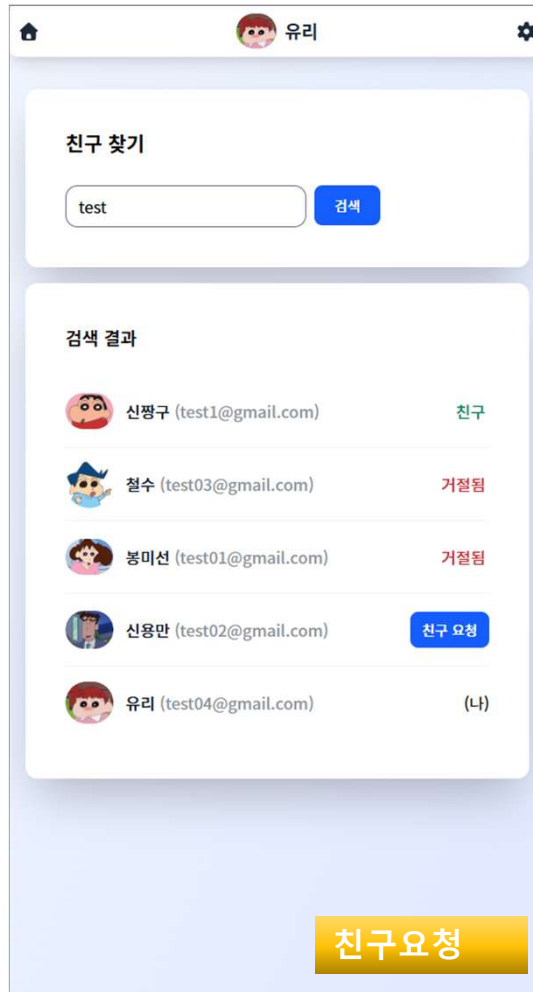
전체 아키텍처



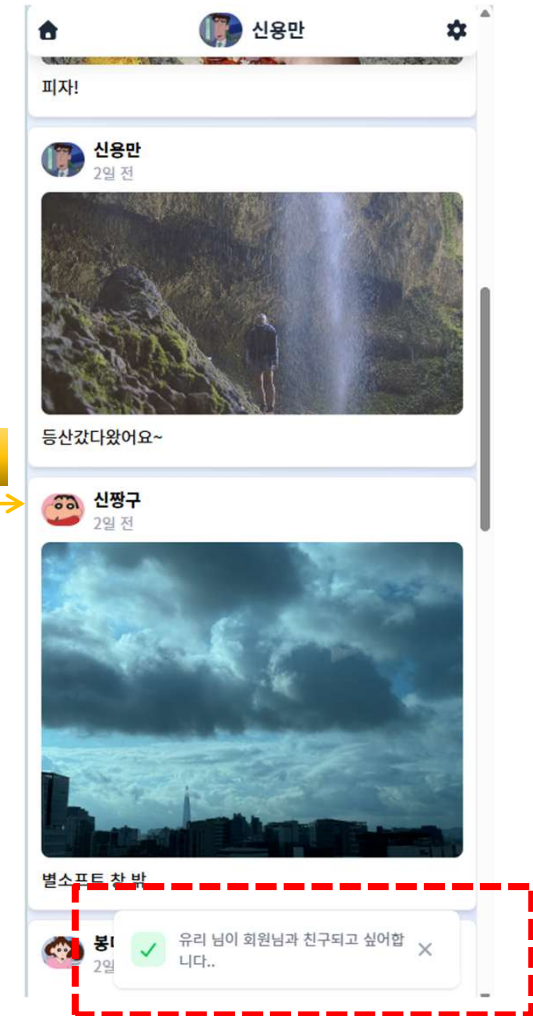
Kafka 이용 사례 1 - 친구요청



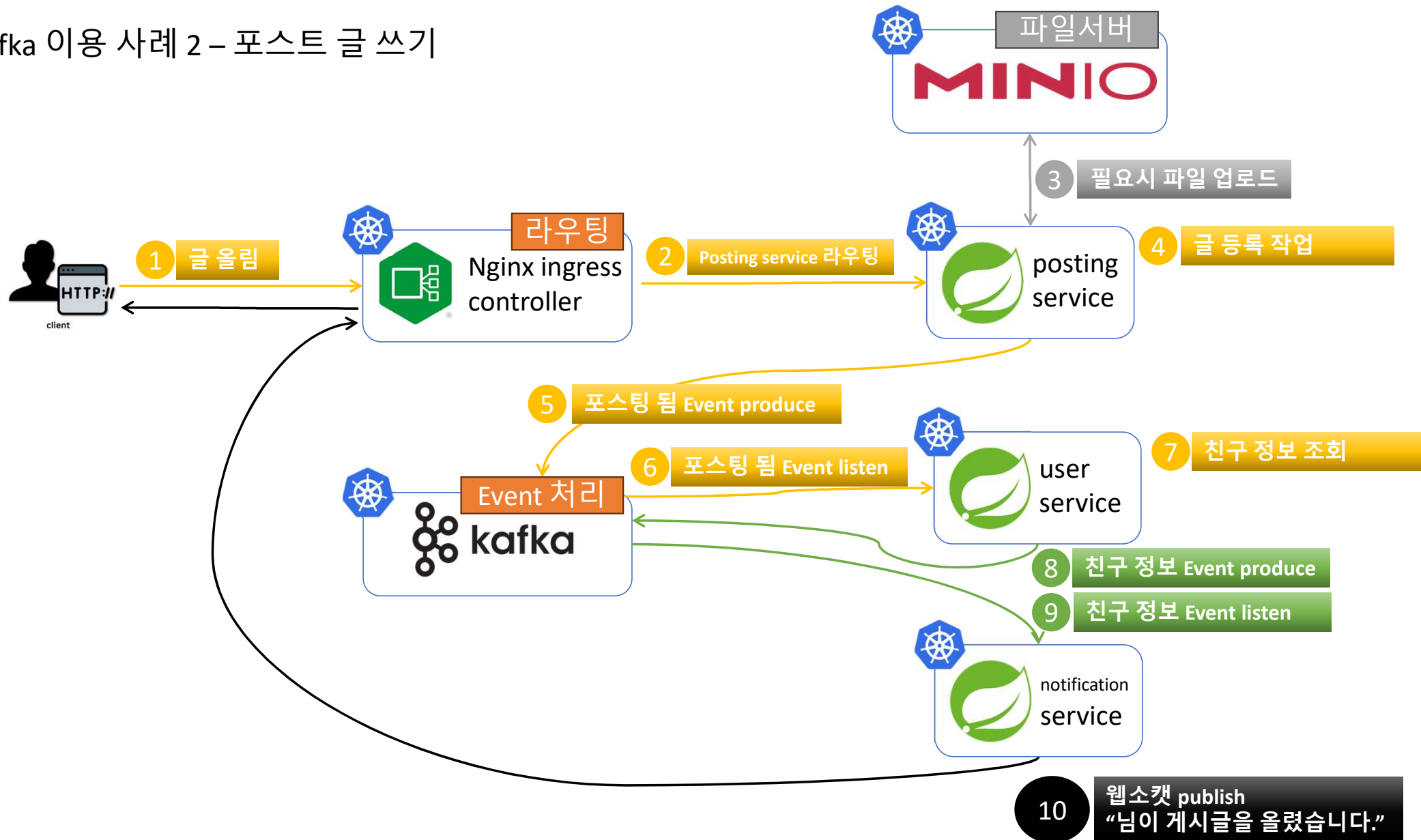
Kafka 이용 사례 1 - 친구요청



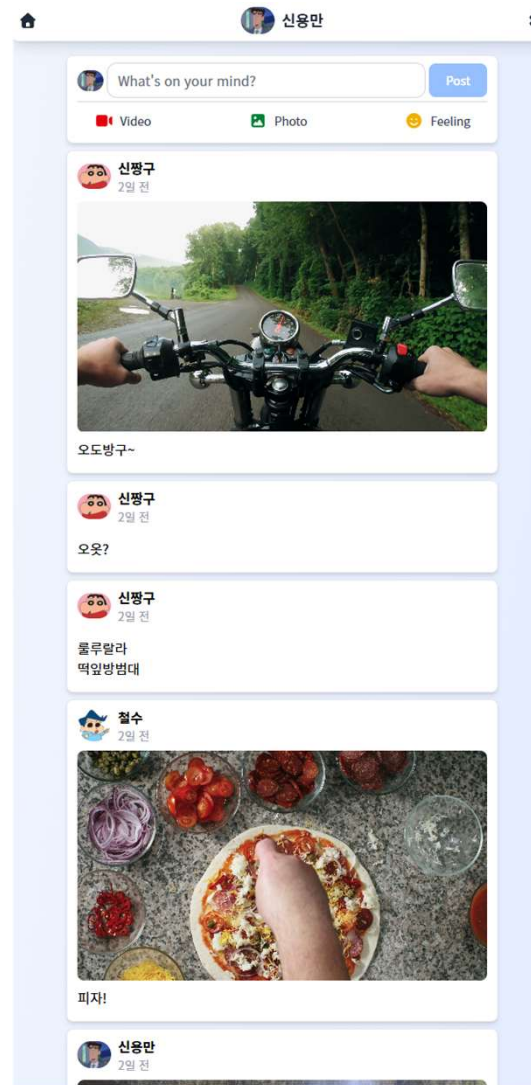
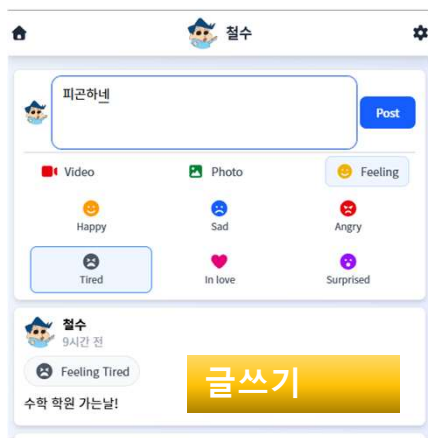
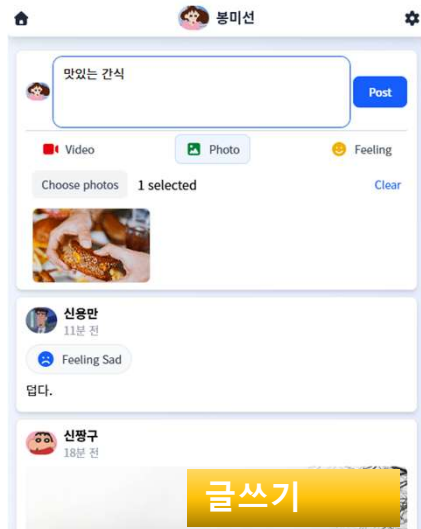
알림 받음



Kafka 이용 사례 2 – 포스트 글 쓰기



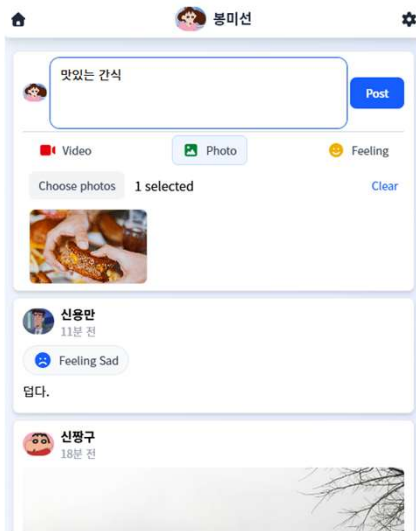
Kafka 이용 사례 2 – 포스트 글 쓰기



알림 받음



포스트 글 쓰기 사례 trace 하기



```
[abd98949d1f805142f6c50dbbbd0ac12-14cdc4b95693772d] c.e.p.controller.PostingController : [save] session ID : a66dca6f-1eef-454d-  
[abd98949d1f805142f6c50dbbbd0ac12-14cdc4b95693772d] com.example.postingservice.model.Users : [parseLoggedInInfo] session value : {seq=  
0:52:08.441808, crtIp=10.42.0.1, crtSeq=null, udtDt=2025-08-19T00:52:08.441808, udtIp=null, udtSeq=null}  
[abd98949d1f805142f6c50dbbbd0ac12-14cdc4b95693772d] c.e.p.controller.PostingController : [save] loggedIn : Users(seq=2, email=te  
52:08.441808, crtIp=10.42.0.1, crtSeq=null, udtDt=2025-08-19T00:52:08.441808, udtIp=null, udtSeq=null)  
[abd98949d1f805142f6c50dbbbd0ac12-14cdc4b95693772d] c.e.p.controller.PostingController : [save] ip : 10.42.0.1  
[abd98949d1f805142f6c50dbbbd0ac12-14cdc4b95693772d] c.e.p.controller.PostingController : [save] file : org.springframework.web.m  
  
[abd98949d1f805142f6c50dbbbd0ac12-14cdc4b95693772d] c.e.p.controller.PostingController : [save] activeAction : image  
[abd98949d1f805142f6c50dbbbd0ac12-14cdc4b95693772d] c.e.p.controller.PostingController : [save] selectedFeeling : null  
[abd98949d1f805142f6c50dbbbd0ac12-14cdc4b95693772d] c.e.p.controller.PostingController : [save] contents : 맛있는 간식  
[abd98949d1f805142f6c50dbbbd0ac12-14cdc4b95693772d] c.e.p.service.PostingServiceImpl : [savePosting] file : org.springframework  
  
[abd98949d1f805142f6c50dbbbd0ac12-14cdc4b95693772d] c.e.p.service.PostingServiceImpl : [savePosting] 파일 업로드 !
```

로그 확인시 Trace ID는 **abd98949d1f805142f6c50dbbbd0ac12**

포스트 글 쓰기 사례 trace 하기 – Grafana Tempo로 해당 Trace ID로 조회

The screenshot displays the Grafana Tempo web interface. The left sidebar contains navigation links: Home, Bookmarks, Starred, Dashboards, Playlists, Snapshots, Library panels, Shared dashboards, Explore (selected), Drilldown, Metrics, Logs, Traces, Profiles, Alerting, Connections, Add new connection, Data sources, Administration, Provisioning, General, Plugins and data, Users and access, and Authentication. The main area shows a trace for the query `abd98949d1f805142fc50dbbd0ac12`. The trace is titled `posting-service: http post /posting/save` with a duration of 742.57ms. It shows a service graph with three spans: `posting-service` (742.57ms), `HTTP POST` (53.67ms), and `file-service` (50.2ms). The `file-service` span is expanded, showing its details: `http post /file/upload/posting`, duration 50.2ms, start time 108.16ms, and status `unset`. The interface also includes a search bar, a 'Run query' button, and a 'Query inspector' button.

Home > Explore > tempo

Q Search... ctrl+k +

Query history Share

Split Add < Last 1 hour > Run query

Outline tempo

Queries Traces

Query type Search TraceQL Service Graph

Build complex queries using TraceQL to select a list of traces.

Documentation

abd98949d1f805142fc50dbbd0ac12

Search Options Limit: 20 Spans Limit: 3 Table Format: Traces Streaming: Disabled Metrics Options Step: auto Type: Range Streaming: Disabled

+ Add query Query inspector

Trace

posting-service: http post /posting/save 742.57ms

2025-09-03 14:41:46.705 /posting/save

Span Filters 3 spans Prev Next

Service & Operation

posting-service http post /posting/save (742.57ms)

http post /posting/save

Service: posting-service Duration: 742.57ms Start Time: 0µs (14:41:46.705)

Child Count: 1 Kind: server Status: unset

Library Name: org.springframework.boot Library Version: 3.5.4

Span attributes: exception = none http.url = /posting/save method = POST outcome = SUCCESS status = 200 uri = /posting/save

Resource attributes: service.name = posting-service telemetry.sdk.language = java telemetry.sdk.name = opentelemetry telemetry.sdk.version = 1.49.0

SpanID: 14cdc4b95693772d

HTTP POST (53.67ms)

file-service http post /file/upload/posting (50.2ms)

http post /file/upload/posting

Service: file-service Duration: 50.2ms Start Time: 108.16ms (14:41:46.813)

Kind: server Status: unset Library Name: org.springframework.boot Library Version: 3.5.4

Span attributes: exception = none http.url = /file/upload/posting method = POST outcome = SUCCESS status = 200 uri = /file/upload/posting

Resource attributes: service.name = file-service telemetry.sdk.language = java telemetry.sdk.name = opentelemetry telemetry.sdk.version = 1.49.0

SpanID: df33bc843a7348b3

포스트 글 쓰기 사례 trace 하기 – Kafka 포스팅 됨

```
/** insertPosts */
    insert into posts(user_seq, contents, contents_type, contents_file_seq, selected_feeling, crt_dt, crt_ip, crt_seq)
    values(2, '맛있는 간식', 'image', 25, NULL, CURRENT_TIMESTAMP, '10.42.0.1', 2);
<== Updates: 1
Closing non transactional SqlSession [org.apache.ibatis.session.defaults.DefaultSqlSession@7bd5632f]
2025-09-03T05:41:46.912Z INFO 1 --- [posting-service] [p-nio-90-exec-1] [abd98949d1f805142f6c50dbbbd0ac12-14cdc4b95693772d] c.e.p.kafka.PostingEventProducer
[Kafka] sending to posting.user.friends payload={"key":"52494c8f-bb6b-40a8-a435-ad8bf3104a75","type":"posted","userSeq":2,"friendSeqs":[],"ip":"10.42.0.1"}
2025-09-03T05:41:46.943Z INFO 1 --- [posting-service] [p-nio-90-exec-1] [abd98949d1f805142f6c50dbbbd0ac12-14cdc4b95693772d] o.a.k.clients.producer.ProducerConfig
ProducerConfig values:
    acks = -1
    auto.include.jmx.reporter = true
    batch.size = 16384
    bootstrap.servers = [kafka:9990]
    buffer.memory = 33554432
    client.dns.lookup = use_all_dns_ips
    client.id = posting-service-producer-1
    compression.gzip.level = -1
    compression.lz4.level = 9
    compression.type = none
    compression.zstd.level = 3
    connections.max.idle.ms = 540000
    delivery.timeout.ms = 120000
```

로그로 Event Key 확인

포스트 글 쓰기 사례 trace 하기 – Kafka 포스팅 됨 – Kafka UI 에서 확인

The screenshot shows the Kafka UI for the topic `posting.user.friends`. The interface includes a sidebar with navigation links (Dashboard, local, Brokers, Topics, Consumers) and a main content area with tabs for Overview, Messages, Consumers, Settings, and Statistics. The Messages tab is active, displaying a table of messages. The selected message has an Offset of 4, Partition of 0, and a Timestamp of 2025. 9. 3. 14시 41분 47초. The Key is `52494c8f-bb6b-40a8-a435-ad8bf3104a75` and the Value is `{"key":"52494c8f-bb6b-40a8-a435-ad8bf3104a75", "type": "posted", "userSeq": 2, "friendSeq": [], "ip": "10.42.0.1"}`. The Value Serde is String (Size: 107 Bytes).

Offset	Partition	Timestamp	Key	Value
4	0	2025. 9. 3. 14시 41분 47초	52494c8f-bb6b-40a8-a435-ad8bf3104a75	<code>{"key":"52494c8f-bb6b-40a8-a435-ad8bf3104a75", "type": "posted", "userSeq": 2, "friendSeq": [], "ip": "10.42.0.1"}</code>

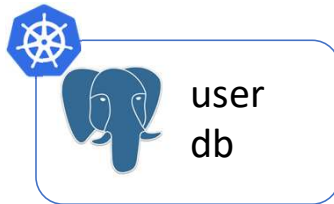
The screenshot shows the Kafka UI for the topic `user.notifications.posting`. The interface is similar to the previous one, with the Messages tab active. The selected message has an Offset of 4, Partition of 0, and a Timestamp of 2025. 9. 3. 14시 41분 47초. The Key is `52494c8f-bb6b-40a8-a435-ad8bf3104a75` and the Value is `{"key":"52494c8f-bb6b-40a8-a435-ad8bf3104a75", "type": "posted", "userSeq": 2, "friendSeq": [1, 3, 4], "ip": "10.42.0.1"}`. The Value Serde is String (Size: 112 Bytes).

Offset	Partition	Timestamp	Key	Value
4	0	2025. 9. 3. 14시 41분 47초	52494c8f-bb6b-40a8-a435-ad8bf3104a75	<code>{"key":"52494c8f-bb6b-40a8-a435-ad8bf3104a75", "type": "posted", "userSeq": 2, "friendSeq": [1, 3, 4], "ip": "10.42.0.1"}</code>

느낀점 1/3

DB를 무작정 많이 분리 함으로서 많이 불편하다.

- 안 좋은 예시를 경험한 것일 수도 있으나, 여러 데이터 처리가 필요하면 feign(API) 및 kafka 처리를 많이 활용하게 된다.
- 데이터량 부담이 많은 경우가 아니라면 DB를 분리할 이유가 없다.
- 공통코드 처리가 어렵다. 공통코드 관리를 중앙화 하거나 각 DB별 공통코드 관리해야함.



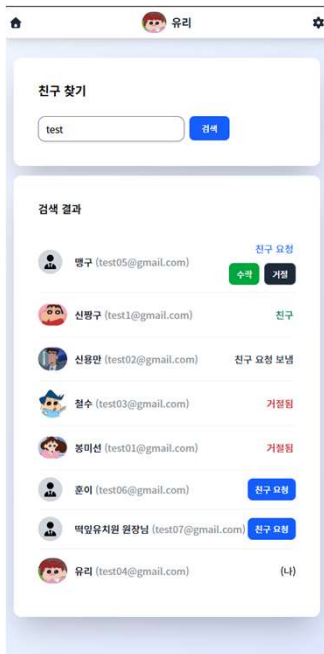
느낀점 2/3

Feign 및 Kafka 처리시 model DTO 및 payload DTO 동기화 방안 고민이 필요하다.



느낀점 3/3

React.js 문법이 Vue.js에 비해서 불편하고 어렵다.



```
<div className="friends-actions">
  { /* 친구 상태별 버튼 */ }
  <div className="friends-action-group">
    {user.friendStatus === "friend" && (
      <span className="status-friend">친구</span>
    )}
    {user.friendStatus === "received" && (
      <span className="status-request">친구 요청</span>
    )}
    {user.friendStatus === "requested" && (
      <span className="status-sent">친구 요청 보냄</span>
    )}
    {user.friendStatus === "rejected" && (
      <span className="status-declined">거절됨</span>
    )}
  </div>
  { [
    "friend",
    "received",
    "requested",
    "rejected",
  ].includes(user.friendStatus) && (
    <Button
      color="blue"
      size="sm"
      onClick={() : Promise<void> => sendFriendRequest(user.seq)}
    >
      친구 요청
    </Button>
  ) }
</div>
```

- Vue.js에서는 각 컴포넌트에서 style 태그에 scoped 속성이 있지만, React.js에서는 그런 유사한 속성이나 기능 제공이 없다.

- Vue.js에서는 v-if, v-else-if, v-else directive가 있지만, React.js에서는 유사한 기능이 없다.

- 전역 state 관리하는 모듈로 Vue.js에서는 Pinia를 사용하지만, React.js에서는 Redux를 사용한다. Pinia state의 경우 어디서든 사용이 가능하지만, Redux state는 “컴포넌트” 내에서만 사용이 가능하다.

끝. 감사합니다.