Allezon Analytics Platform

an alternative name was
Allebaba / Allechlop

The Principles

- We want to build something practical.
- We want to face real-life challenges.
- It should be a nice point in CV / project portfolio.

The Principles

- We want to build something practical.
- We want to face real-life challenges.
- It should be a nice point in CV / project portfolio.
- We want to **have fun**.

The Principles

- We want to build something practical.
- We want to face real-life challenges.
- It should be a nice point in CV / project portfolio.
- We want to **have fun**.

We == You together with us

Our Client

- Allezon - one of the biggest online shopping platform.
- They want to build a data-collection and analytics platform.
- Events they want to process are users' actions on their website.

Our Client

- Allezon - one of the biggest online shopping platform.
- They want to build a data-collection and analytics platform.
- Events they want to process are users' actions on their website.
- Ambitious plans:
  - recommandations
  - real time bidding
  - on-line analytics
  - ad-hoc queries
  - anomaly detection
  - platform monitoring

Our Client

- Allezon - one of the biggest online shopping platform.
- They want to build a data-collection and analytics platform.
- Events they want to process are users' actions on their website.
- Ambitious plans:
    - recommandations
    - real time bidding
    - on-line analytics
    - ad-hoc queries
    - anomaly detection
    - platform monitoring
    - everything **powered by AI**, of course ;)

Our Client

- Allezon - one of the biggest online shopping platform.
- They want to build a data-collection and analytics platform.
- Events they want to process are users' actions on their website.
- Ambitious plans:
  - recommandations
  - real time bidding
  - on-line analytics
  - ad-hoc queries
  - anomaly detection
  - platform monitoring
  - everything **powered by AI**, of course ;)

- But, let's start small.

Data - Events - User Tags

- Simplified model
  - users identified by cookies
  - actions: only VIEWs and BUYs

```
{
  "time": int64,
  "cookie": string,
  "country": string,
  "device": PC | MOBILE | TV,
  "action": VIEW | BUY,
  "origin": string,
  "product_info": {
    "product_id": string,
    "brand_id": string,
    "category_id": string,
    "price": int32
  }
}
```

Use Case 1: Collecting Events

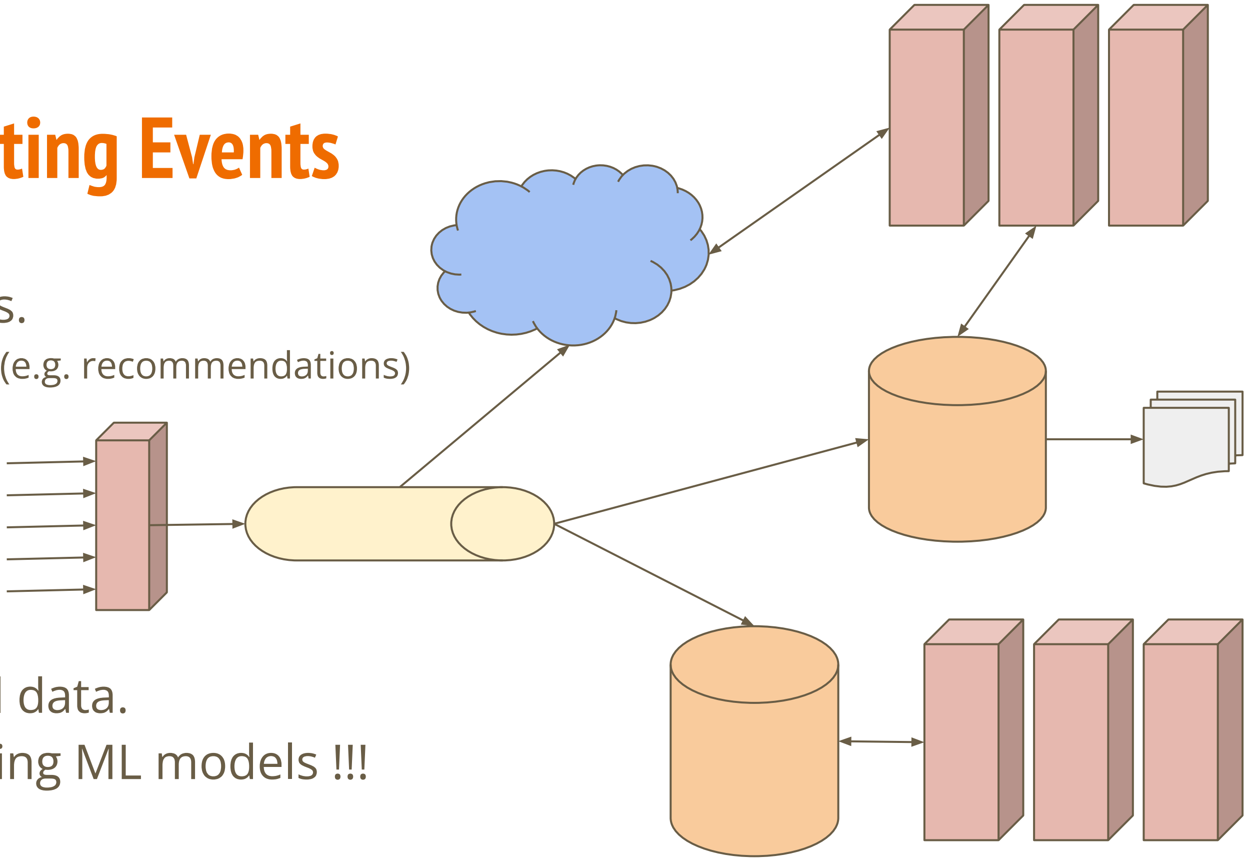
- Building user profiles.
  - input for ML models (e.g. recommendations)
- On-line analytics.
  - trends, patterns
  - anomalies
  - monitoring
  - KPIs
- Queries on historical data.

Use Case 1: Collecting Events

- Building user profiles.
  - input for ML models (e.g. recommendations)
- On-line analytics.
  - trends, patterns
  - anomalies
  - monitoring
  - KPIs
- Queries on historical data.
- Data points for training ML models !!!

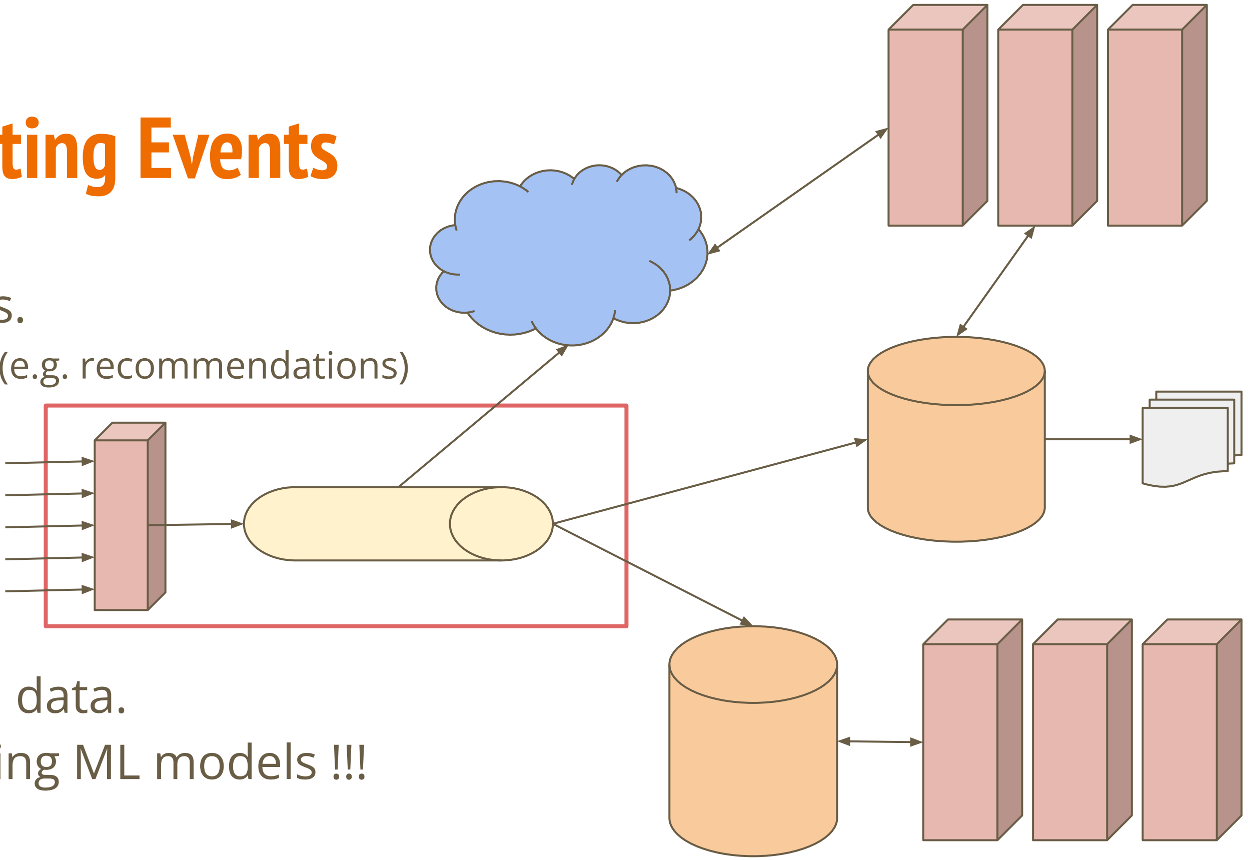
Use Case 1: Collecting Events

- Building user profiles.
  - input for ML models (e.g. recommendations)
- On-line analytics.
  - trends, patterns
  - anomalies
  - monitoring
  - KPIs
- Queries on historical data.
- Data points for training ML models !!!



Use Case 1: Collecting Events

- Building user profiles.
  - input for ML models (e.g. recommendations)
- On-line analytics.
  - trends, patterns
  - anomalies
  - monitoring
  - KPIs
- Queries on historical data.
- Data points for training ML models !!!



Use Case 2: User Profiles

- User Profile
  - cookie
  - views (last 200)
  - buys (last 200)

Use Case 2: User Profiles

- User Profile
  - cookie
  - views (last 200)
  - buys (last 200)
- Requirements
  - max throughput: 1000 req/s
  - request timeout: 200 ms
  - data latency: 10s

Use Case 2: User Profiles

- User Profile
  - cookie
  - views (last 200)
  - buys (last 200)
- Requirements
  - max throughput: 1000 req/s
  - request timeout: 200 ms
  - data latency: 10s
- Applications
  - recommender systems
    - real-time-bidding ad auctions
  - security
    - fraud/bot detection

Use Case 3: Aggregates

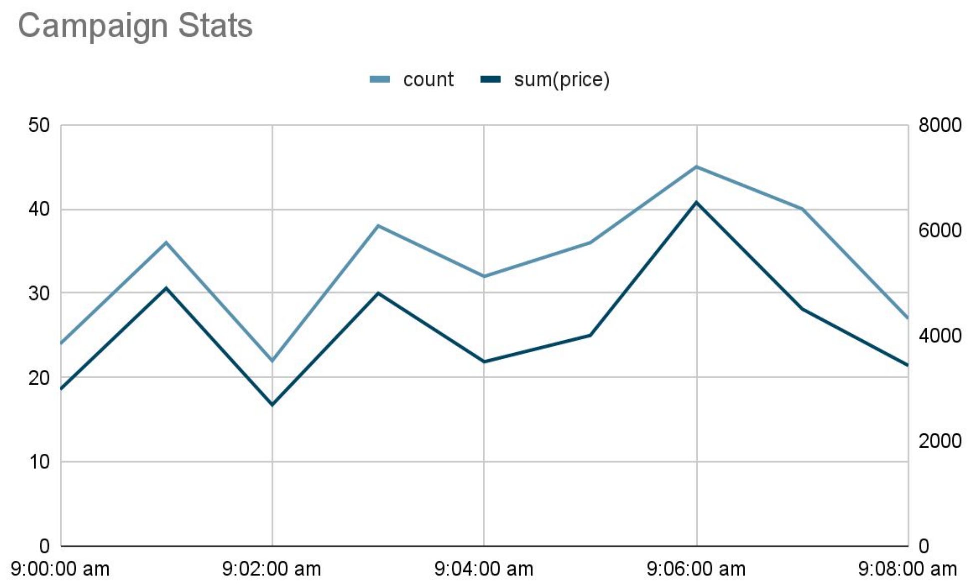
1m_bucket	action	origin	count	sum(price)
9:00:00	BUY	NIKE_SHOES_CAMPAIGN	24	2976
9:01:00	BUY	NIKE_SHOES_CAMPAIGN	36	4896
...	...	...	...	...
9:08:00	BUY	NIKE_SHOES_CAMPAIGN	27	3429

```
{
  "time": int64,
  "cookie": string,
  "country": string,
  "device": PC | MOBILE | TV,
  "action": VIEW | BUY,
  "origin": string,
  "product_info": {
    "product_id": string,
    "brand_id": string,
    "category_id": string,
    "price": int32
  }
}
```

Use Case 3: Aggregates

```
{
  "time": int64,
  "cookie": string,
  "country": string,
  "device": PC | MOBILE | TV,
  "action": VIEW | BUY,
  "origin": string,
  "product_info": {
    "product_id": string,
    "brand_id": string,
    "category_id": string,
    "price": int32
  }
}
```

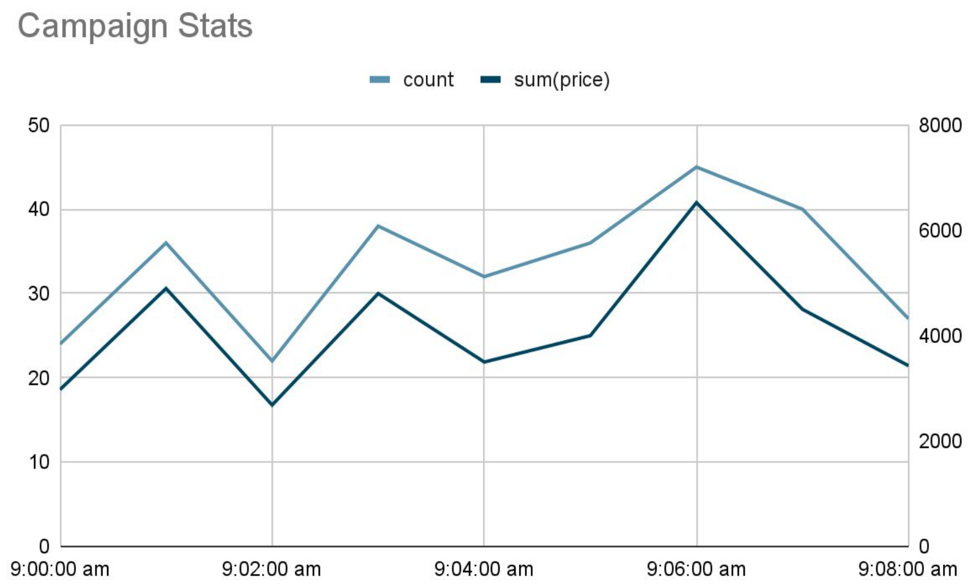
1m_bucket	action	origin	count	sum(price)
9:00:00	BUY	NIKE_SHOES_CAMPAIGN	24	2976
9:01:00	BUY	NIKE_SHOES_CAMPAIGN	36	4896
...	...	...	...	...
9:08:00	BUY	NIKE_SHOES_CAMPAIGN	27	3429



Use Case 3: Aggregates

```
SELECT 1m_bucket(time), action, [origin, brand_id, category_id], count(*), sum(price)
    FROM events
    WHERE time >= ${time_range.begin} and time < ${time_range.end}
        AND action = ${action}
        [AND origin = ${origin}]
        [AND brand_id = ${brand_id}]
        [AND category_id = ${category_id}]
    GROUP BY 1m_bucket(time), action, [origin, brand_id, category_id]
    ORDER BY 1m_bucket(time)
```

```
{
  "time": int64,
  "cookie": string,
  "country": string,
  "device": PC | MOBILE | TV,
  "action": VIEW | BUY,
  "origin": string,
  "product_info": {
    "product_id": string,
    "brand_id": string,
    "category_id": string,
    "price": int32
  }
}
```



Testing Platform

- You can subscribe to a stream of events and queries
    - events == user tags (Use Case 1)
    - queries
        - UserProfileQuery (Use Case 2)
        - AggregatesQuery (Use Case 3)
-

Testing Platform

- You can subscribe to a stream of events and queries
  - events == user tags (Use Case 1)
  - queries
    - UserProfileQuery (Use Case 2)
    - AggregatesQuery (Use Case 3)
- Subscription parameters
  - host
  - port
  - throughput
  - seed

Testing Platform

- You can subscribe to a stream of events and queries
  - events == user tags (Use Case 1)
  - queries
    - UserProfileQuery (Use Case 2)
    - AggregatesQuery (Use Case 3)
- Subscription parameters
  - host
  - port
  - throughput
  - seed
- Actions on subscriptions
  - START / CLOSE
  - PAUSE / RESUME
- Debug mode
  - initially ENABLED