

Course Structure & Coverage



Parts in This Course

Kafka core usage

1

Kafka in microservice
Kafka stream

2

Kafka connect
Kafka user interface

3

Kafka schema registry
Avro on Kafka
Kafka REST proxy

4

ksqldb

5

Source code
Resource & references
Discount codes

6

*Take the parts in sequence
Future updates are free
Updates might come part-by-part
Learn by console (terminal) first
Skip to Part 3 for UI reference*

Take What You Need

- × Take parts that match your learning requirements
- × Examples
- × Can learn Kafka basic concept on early part of the course
- × Only need this one Kafka course to learn many Kafka aspects



Kafka & Spring

- × Introduction
- × Technology In This Course
- × Installation
- × Kafka Basic Concepts
- × Writing Java Spring Codes for Kafka
- × JSON Processing
- × Error handling
- × Retry mechanism



Kafka in Microservice & Kafka Stream

- × Kafka in microservice architecture
- × Simulate real world implementation
- × Kafka Stream with Spring
- × Don't jump to Kafka Stream if you are new to Kafka



Kafka Connect & User Interface

- × Fetch data from other system to kafka
- × Fetch data from kafka to other system
- × Use case for real-world sample
- × Sample user interface



Schema Registry & REST Proxy

- ✗ Avro
- ✗ Use avro in kafka & kafka stream
- ✗ Kafka schema registry
- ✗ Kafka user interface (free for development)
- ✗ Kafka REST API



KsqlDB

- × SQL like for event-streaming
- × Built on top of kafka stream
- × Speed up kafka usage for team familiar with SQL



Course Coverage

- × Complex and broad topic
- × Covered in this course
 - × Use Kafka with Java Spring Boot
 - × Basic Kafka installation
- × NOT covered in this course
 - × Kafka administration / monitoring for production usage



Download Source Code



Download Source Code

- × Available on last section of the course (**Resources & References**)
- × Source code, Postman collection, Kafka scripts



Different Source Code

- ✗ Video vs source file (build.gradle, .java, etc)
- ✗ Due to course update
- ✗ Difference is minor & not breaking the app



AI Assistant In This Course



AI Assistant

- × The rise of AI in software engineering
- × This course uses GitHub Copilot as AI assistant
- × ***Using GitHub Copilot is NOT mandatory for the course***
 - × *Can fully participate and gain comprehensive understanding, even without using GitHub Copilot*



AI Assistant

- × This course offers learning path with AI assistant
- × Using GitHub Copilot as the AI assistant
- × Dedicated section to explain about AI Assistant
- × The course uses AI assistant when writing codes
- × Course content is the same value & usability whether using AI assistant or not



ChatGPT?

- × ChatGPT is famous when this video is recorded
- × GitHub Copilot for source code / software engineering
- × Lesson explaining ChatGPT and GitHub Copilot
- × ***The course mainly uses GitHub Copilot***
- × ChatGPT is additional & optional knowledge



About This Section



About This Section

- × This section is about AI & LLM
- × Optional
- × Can skip if not interested in AI
- × Following this section is recommended
 - × AI & LLM using ChatGPT & GitHub Copilot as examples
 - × Prompt engineering



AI & Large Language Models (LLM)



Artificial Intelligence (AI)

- × Smart computer friend
 - × Learns from experiences
 - × Recognizes patterns.
- × Called as *Artificial Intelligence (AI)*
- × AI do specific tasks based on learning
- × AI already part of our lives



Large Language Models (LLM)

- × AI trained based on huge amount of text data
- × Understands and generates human-like language
- × Handles natural language (e.g. English) processing tasks
- × "Large" refers to extensive neural network size
- × "Neural networks" brief description



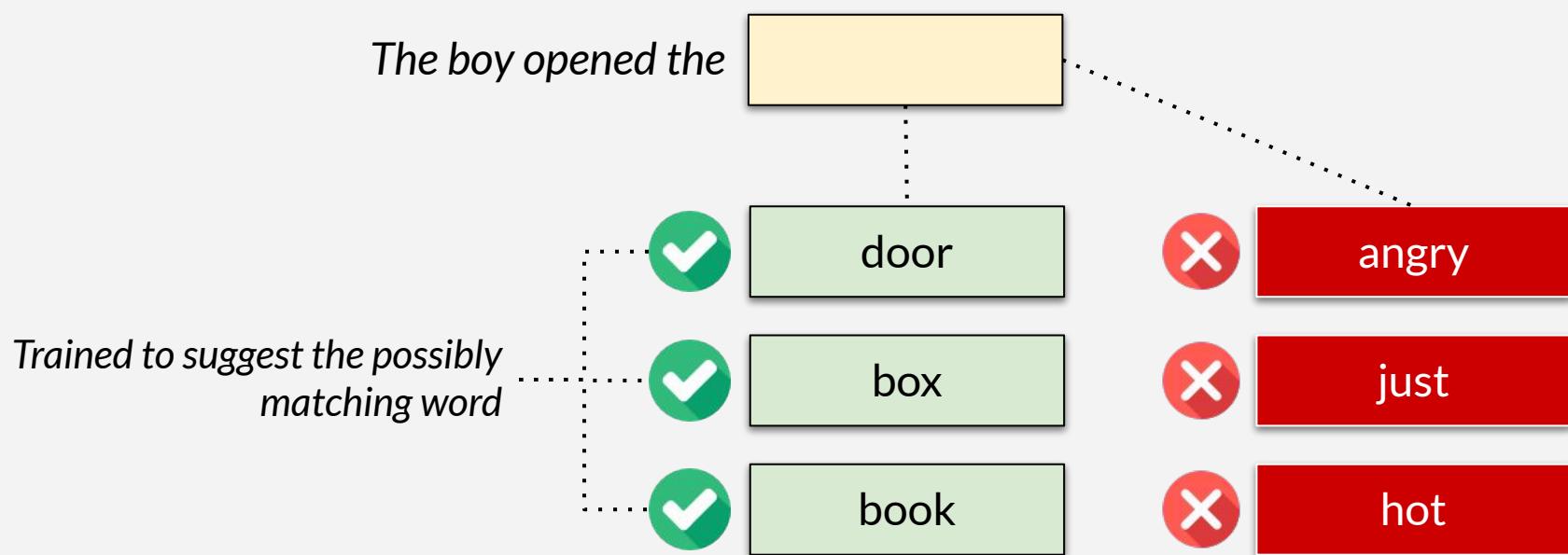
Large Language Models (LLM)

- × Famous LLM: *ChatGPT*
- × LLM with specific task: *Github Copilot*
- × Glossary
 - × LLM / Large Language Models / models
 - × Github Copilot / Copilot



How (Basic) LLM Works

Trained to predict the next word



How (Fine-Tuned) LLM Works

Trained to follow instruction / solve specific task

Continuously, the LLM uses word context to generate the next word, until completing the specified task.

Generate a description for a golden retriever dog

The golden retriever is a friendly, intelligent, and loyal dog breed with a distinctive golden coat. Known for their affectionate nature and playful demeanor, ...

Will be used in this course (e.g. ChatGPT, Github Copilot)

Prompt

Input (*prompt*)

Make a Python code that receives a number (x) as input and generates the first x Fibonacci sequence. Display the Fibonacci sequence in the console.

Create the menu for the entire week, and make sure no duplicate menus within the week.

In English (or any natural language that supported by the model)



LLM

Output (*response*)

```
def generate_fibonacci_sequence(x):  
    fibonacci_sequence = []  
    a, b = 0, 1  
  
    for _ in range(x):  
        fibonacci_sequence.append(a)  
        a, b = b, a + b  
  
    return fibonacci_sequence  
  
def main():  
    ...  
    ...
```

Fruits: Apple Slices with Peanut Butter
Dairy: Yogurt

...

Prompt

- ✗ Context
- ✗ Constraints
- ✗ Output format
- ✗ Prompt contains context at minimum
- ✗ Includes the necessary information in the prompt to produce desired output



Prompt - Context

What kind of output we want from the models

Write a Python application that asks the user to input a number and then displays the number from 0 until the user inputs it to the terminal.

Write a product description about an apple. Focus on the nutrition facts and health benefits.

Prompt - Constraints

Limitations / requirements to guide the generated output

Write a Python application that asks the user to input a number and then displays the number from 0 until the user inputs it to the terminal.

The user input must be an even number, at most 10 thousand.

Write a product description about an apple. Focus on the nutrition facts and health benefits.

The output must be a maximum of 200 words.

Prompt - Output Format

Structure or layout of the generated output

Write a Python application that asks the user to input a number and then displays the number from 0 until the user inputs it to the terminal.

The user input must be an even number, at most 10 thousand.

Make the output in JSON format where the numbers are in an array named "user_numbers".

Write a product description about an apple. Focus on the nutrition facts and health benefits.

The output must be a maximum of 200 words.

Display the facts as bullet points.

Important Points on Course With AI Assistant



LLM Characteristics In The Course

- × *Rapidly evolving technology*
- × *Discrepancies between LLM software and course content is expected*
- × *Non-deterministic nature: different responses for the same task*
- × *AI assistant has limitations, requires validation and adjustment skills*
- × *Emphasize the role of AI as an **assistant**, not the final decision maker*
- × *Learning method may differ from traditional courses*



About LLM Response (Output)

- × Verification is crucial to ensure the generated output aligns with the intended purpose
- × Exercise healthy skepticism and avoid blind trust in the accuracy of the output
- × Re-check for errors or unintended meanings in the generated content



LLM Used In The Course

- ✗ **ChatGPT**
 - ✗ Free version
 - ✗ The course using paid version
 - ✗ Can use the free version (less quality vs paid)
 - ✗ chatgpt.com
- ✗ **Github Copilot**
 - ✗ 30-day free trial, then need paid license
 - ✗ Main AI assistant in this course
 - ✗ The course using paid version
 - ✗ More detail later
 - ✗ github.com/features/copilot



LLM Used In The Course

- × Always check official website for latest pricing information
- × **AI assistant = ChatGPT / Github Copilot**



Prompt & Response In The Course

- × Some are shown as samples only (on slides)
- × Some used as actual prompt to **AI assistant**
- × Use same prompt with course or your own prompt



Prompt & Response In The Course

- ✗ LLM is **non-deterministic!**
- ✗ Source code is provided as course **reference**
- ✗ Source code from AI assistant response can be **different** from course reference
- ✗ Different AI assistant can produce **different** response or same prompt
- ✗ Re-enter same prompt can produce **different** response
- ✗ Source code from AI assistant **might not always work** as expected
- ✗ **Differs** from traditional (non AI assistant) course



Download AI Prompts



Prompts

- × Many prompts available for download
 - × Simpler / less important prompts are not provided for download
- × Source code / technical scripts available for download
 - × Only for reference
- × Download from last section of the course:
Resources & References



Prompts & Source Code

- ✗ Minor difference is fine
- ✗ Might be caused by course update
- ✗ Prompts / source code / scripts should still usable



Prompt Engineering



Prompt Engineering

- × Designing effective prompts for AI assistant input
- × Instruction to produce the expected output
- × Prompt engineering goal



Write Clear & Specific Instructions



Write Clear & Specific Instructions

- × As clear & specific as possible
- × Guide AI assistant, reduce chance of irrelevant response
- × Can be long instructions that provides more clarity



Use Delimiters

- × Use delimiters to indicate distinct parts
- × Common delimiters
 - × **### our-text ###**
 - × **--- our-text ---**
 - × **""" our-text """**
 - × **''' our-text '''**
 - × **<our-text>**
 - × **<tag> our-text </tag>**



Use Delimiters

Summarize this story

###

The story

###

Write code that asks a string as input, and calculate how many times each alphabet appears in the string.

Write the code in Python, Java, and PHP

<task>

Generate three lists, where each list represents a continent on the earth.

<task>

<task>

Each list must have four items, where each item in the list represents a country in the respective continent. State the name, capital, currency, and language of each country. If the country has more than one currency or language, state them all.

<task>

Ask Specific Output Format

- × Might needed for further processing (e.g. CSV, JSON)
- × Can ask for non-technical output format



Ask Specific Output Format

Give me the five first multiples of 15 and their factors **in JSON format**. For each multiple, show how many counts of even factor and odd factor. **Use snake case for the JSON attribute names.**

List all Southeast Asian countries, along with their capital and main currency. **Make your output in a sentence like this:**
"Country X, which is located in Southeast Asia, has capital Y and currency Z."

###

You will be given a paragraph containing the car manufacturer, model, fuel type, and manufacturer year.

###

###

For each car in the list, convert them as SQL insert statements to PostgreSQL database.

The table name is "cars".

The table columns are

- car_id (data type UUID, randomly generated)
- manufacturer
- model
- fuel_type (FOSSIL / ELECTRIC / HYBRID)
- manufacture_year (integer)
- created_at (timestamp with timezone), give the current timestamp for this column

###

Here is the paragraph.

###

The paragraph

###

Check Conditions

- ✗ AI assistant can guess response, but not always accurate
- ✗ Instruct AI assistant to bail out if cannot execute request



Check Conditions

###

You will be provided with text delimited by triple hash.

If it contains a sequence of instructions, rewrite those instructions in the following format:

Step 1: ...

Step 2: ...

...

Step N: ...

###

###

If the text does not contain instructions, write "No steps provided".

###

Here is the text

###

The text

###

Give Examples

- × Give examples to clarify expected output
- × Simple prompt
 - × *Give me five random countries, their languages, Independence Day, and currencies. Give me the output format in JSON array.*
- × More structured JSON?



Give Examples

```
###
```

Give me five random countries, their languages, independence day, and currencies. Give me the output format in JSON array.

```
###
```

For output format, see the following JSON example.

```
###
```

```
[
```

```
{
```

```
  "name": "Indonesia",
```

```
  "independenceDay": "1945-08-17",
```

```
  "languages" : ["Bahasa Indonesia"],
```

```
  "currency": [
```

```
    {
```

```
      "currencyName": "Indonesian Rupiah",
```

```
      "currencyCode": "IDR"
```

```
    }
```

```
  ]
```

```
},
```

```
{
```

```
...
```

```
}
```

```
]
```

```
###
```

Explain The Steps

- × A too-complex prompt may trigger incorrect guess
- × Human analogy
- × Provide step-by-step to work more thoroughly
- × More detailed effort for the prompt



Explain The Steps

Step 1

You will be given a text between tag <text>...</text>

###

Step 2

Summarize the given text in a maximum of 100 words

###

Step 3

Extract all nouns, verbs, and adjectives from given <text>...</text>, and put each in an array

###

Step ...

More specific step...

###

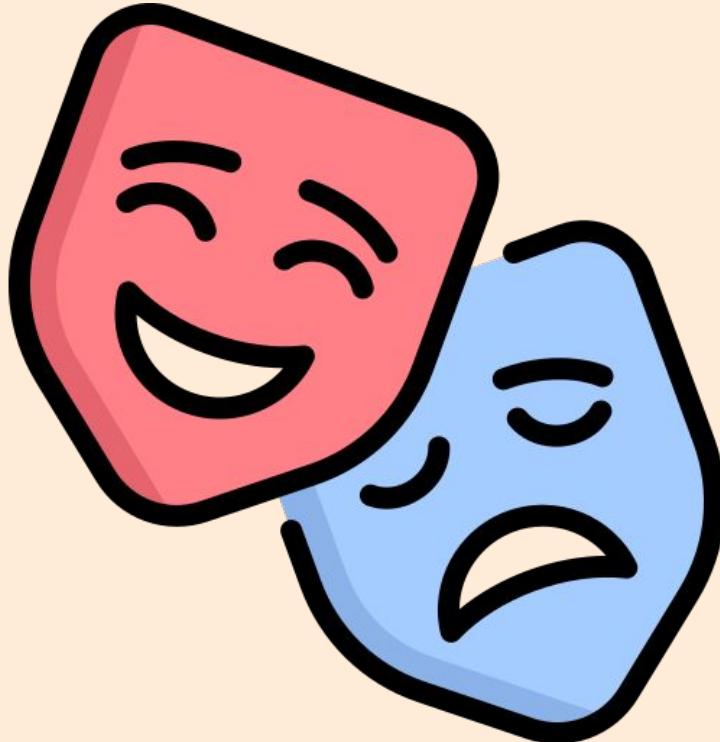
Here is the text

<text>

The text

</text>

Role Prompting



Role Prompting

- × Assigning specific role to the AI assistant
 - × As engineer, dentist, comedian, ...
- × Inform the AI assistant about more context in a role



Role Prompting

You are a high school teacher trying to explain a financial plan to your students.

Explain in detail why financial planning and investing are important for the long term.

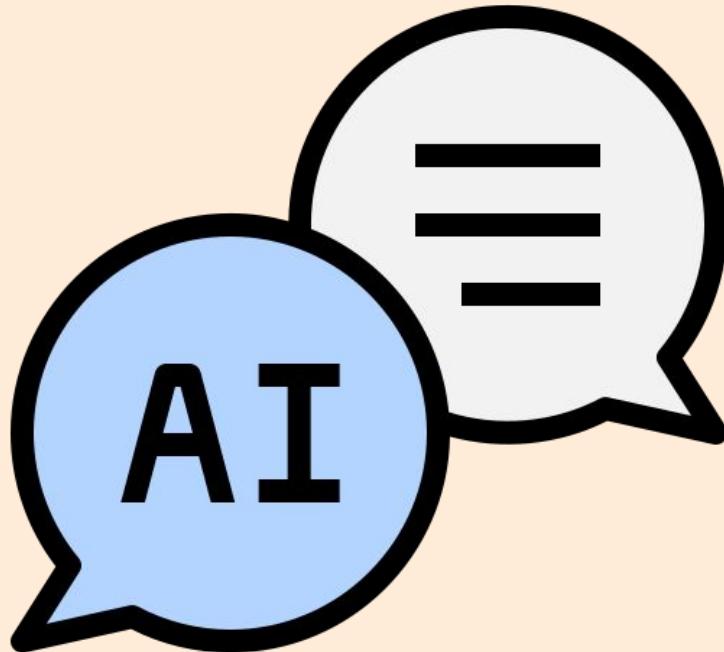
You are a professional financial worker trying to explain a financial plan to a wealthy potential client.

Explain in detail why financial planning and investing are important for the long term.

You are a mother trying to explain a financial plan to your 10-year-old daughter, who has never known about financial terms.

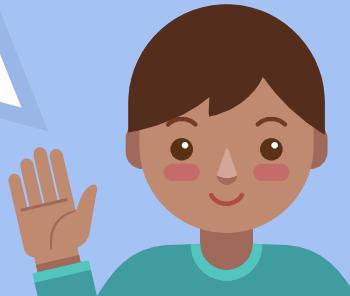
Explain in detail why financial planning and investing are important for the long term.

Single Prompt vs Conversation



Conversation Style Prompts

- × "Conversation" with LLM
- × Is multiple prompts (conversation) easier?
- × Can be effective
- × Risk of output bias in long conversation
- × Can less clear, less precise
- × Conversation work fine in many cases
- × Adopt strategy explained before for either single prompt or conversation



AI Assistant in Software Engineering



How AI Assistant Change The Way We Learn

Before LLM: Google, browse coding/tutorial sites

The image shows a screenshot of a web browser with two tabs open. The left tab is titled "Python" and contains the following Python code:

```
for i in range(1, 11):
    print(i)
```

The right tab is titled "Java" and contains the following Java code:

```
public class PrintNumbers {
    public static void main(String[] args) {
        for (int i = 1; i < 11; i++) {
            System.out.println(i);
        }
    }
}
```

Both tabs show search results for "python program to print numbers 1 to 10" and "Java print numbers", respectively, with the Stack Overflow logo visible at the top of each tab.

How AI Assistant Change The Way We Learn

With AI assistant: Ask what we want in English (or other supported language)

You

write a python application that shows numbers 1 to 10 to terminal

ChatGPT

Certainly! Here's a simple Python script that prints numbers 1 to 10 to the terminal:

```
python                                ⌂ Copy code

# Python application to print numbers 1 to 10

def print_numbers():
    for i in range(1, 11):
        print(i)

if __name__ == "__main__":
    print_numbers()
```

Copy and paste this code into a Python file (e.g., `print_numbers.py`) and run it using the command:

```
bash                                ⌂ Copy code

python print_numbers.py
```

It will display the numbers 1 to 10 in your terminal.

You

write a java application that shows numbers 1 to 10 to terminal

ChatGPT

Certainly! Here's a simple Java program that prints numbers 1 to 10 to the terminal:

```
java                                ⌂ Copy code

public class PrintNumbers {
    public static void main(String[] args) {
        printNumbers();
    }

    private static void printNumbers() {
        for (int i = 1; i <= 10; i++) {
            System.out.println(i);
        }
    }
}
```

Copy and paste this code into a Java file (e.g., `PrintNumbers.java`) and compile and run it using the following commands:

```
bash                                ⌂ Copy code

javac PrintNumbers.java
java PrintNumbers
```

This will compile and execute the Java program, and you'll see the numbers 1 to 10 printed in the terminal.

How LLM Change The Way We Learn

With LLM: Ask what we want in English (or other supported language)

A screenshot of a code editor interface. At the top, there is a search bar containing the text "write a python application that shows numbers 1 to 10 to terminal". Below the search bar are three buttons: "Accept", "Discard", and a refresh icon. To the right of these buttons is the text "Changed 3 lines" followed by two small icons. The main area of the editor displays the following Python code:

```
for i in range(1, 11):
    print(i)
```

A screenshot of a code editor interface. At the top, there is a search bar containing the text "write a java application that shows numbers 1 to 10 to terminal". Below the search bar are three buttons: "Accept", "Discard", and a refresh icon. To the right of these buttons is the text "Changed 8 lines" followed by two small icons. The main area of the editor displays the following Java code:

```
public class NumberGenerator {
    public static void main(String[] args) {
        for (int i = 1; i <= 10; i++) {
            System.out.println(i);
        }
    }
}
```

Syntax

- × Rules to form correct & meaningful sentences in a language
- × Rules that dictate how softwares are written using particular programming language
- × Each programming language has its syntax
- × Following syntax is essential
 - × Java relies heavily on bracket { }
 - × Python relies heavily on indentation



LLM & Syntax

- × AI assistant takes requirement in English & returns the source code syntax
- × Learning programming syntax historically has been difficult for new learners
- × Need to know syntax details to implement functionality
- × AI assistant helps ease the syntax complexity



LLM & Syntax

- × Example: math multiplication table (involves nested loop)
- × New learner might put wrong syntax
- × AI assistant generate code with correct syntax
- × Example follows



LLM Explains Code

- × Need to know syntax & algorithm to read/write code
- × AI assistant able to explain piece of code
- × Mistake might happen
- × Use AI assistant explanation as starting point



Re-write (Simplify/Refactor) Code

- × Different ways to accomplish a same task
- × AI assistant produce code that is easier to read
 - × Might also easier to enhance / fix
- × Valuable function of AI assistant
- × Don't blindly trust the modified code



Identifying Bug

- × Programming logic mistake that can cause a software behaves incorrectly
- × Failure in a specific circumstance
- × Can be quite annoying
- × Ask AI assistant to detect possible bug
- × Might not always found



AI Assistant in Software Testing



Introduction to Software Testing

- × Critical component of software development
- × Evaluates software application for expected behavior
- × Assures software quality, reliability, and performance
- × Objectives
 - × Written requirements (user stories) met
 - × Critical paths are covered
 - × Software is bug-free



Functional vs Non-Functional

- × **Functional:** specific behaviors or functions
 - × Example: correct math results in a calculator
- × **Non-functional:** overall qualities like performance, accessibility, scalability
 - × Example: process time (performance)
 - × Example: resource usage



The Nature of Software Testing

- × *Expected* condition
- × *Actual* condition
- × Compares expected vs actual software conditions
- × Identifies discrepancies (bugs)



Expected vs Actual

Functional	Input	Expected	Actual	Result
Addition	$6 + 8$	14	14	OK
Addition	$92 + 11$	103	100	BUG 
Subtraction	$95 - 10$	85	85	OK
Subtraction	$2 - 19$	-17	0	BUG 
Multiplication	$4 * 7$	28	28	OK
Multiplication	$10 * 19$	190	100	BUG 
Division	$28 / 4$	7	7	OK
Division	$15 / 0$	N/A	0	BUG 

Test Case

- × Predefined sequence for test & verify requirement
- × Ensure consistent and repeatable testing procedures
- × (Usually) refers to manual



Test Case Examples

Testing Addition Functionality

Objective:

Verify that the calculator correctly adds two numbers.

Steps:

1. Launch the calculator application
2. Enter the first number (e.g., 5)
3. Press the addition (+) button
4. Enter the second number (e.g., 3)
5. Press the equals (=) button

Expected Result:

The calculator should display the result 8, confirming that $5 + 3 = 8$

Testing Subtraction Functionality

Objective:

Verify that the calculator correctly subtracts one number from another.

Steps:

1. Open the calculator application
2. Input the first number (e.g., 10)
3. Select the subtraction (-) button
4. Input the second number (e.g., 4)
5. Press the equals (=) button

Expected Result:

The calculator should show the result 6, indicating that $10 - 4 = 6$.

Test Script

- × Code or sequence of commands
- × Automate test case execution
- × Run by tool
- × Simulate manual process / system behavior



Test Script Examples

```
describe("Testing Addition Functionality", () => {
  it("Verifies that the calculator correctly adds two numbers",
() => {
  cy.visit("calculator-url");

  cy.get("#number-input").type("5");
  cy.get("#add-button").click();
  cy.get("#number-input").type("3");
  cy.get("#equals-button").click();

  cy.get("#result-display").should("have.text", "8");
});
});
```

```
import { check } from "k6";
import http from "k6/http";

export default function () {
  const res = http.get("calculator-url");

  check(res, {
    "Status is 200": (r) => r.status === 200,
  });

  const subtractResponse = http.post("calculator-url/subtract", {
    firstNumber: "10",
    operation: "subtract",
    secondNumber: "4",
  });

  check(subtractResponse, {
    "Subtraction result": (r) => res.json().result === 6,
  });
}
```

Tools in Software Testing

- × Challenges in software testing
- × Use tools to address challenges
- × Tools types:
 - × Automation
 - × Performance
 - × Bug tracking
- × Benefits of using tools



Tools in Software Testing

Type	Example
Automation	Cypress Selenium TestComplete
Performance	Apache JMeter Gatling Grafana K6
Bug tracking	Asana Bugzilla JIRA

Automation Bias

Misconception: tool can test as comprehensively as a human



January

$x + y = \dots$
Result displayed on textbox
Mathematically correct

A new textbox added on UI
Result displayed on new textbox
New textbox is hidden

Automated test not fail because it checks certain condition only

Expected = actual (automated test)

Expected <> actual (manual test)

January

Develop automation script
Automated tests works fine

Expected = actual



February

$2 + 3 = \text{no result displayed}$
Bug!

Expected <> actual

Automation Bias

- ✗ Assuming tool can replicate human
- ✗ Difference between manual and automated testing
- ✗ Approach toward tools
- ✗ Tool is complementary to manual testing, not replacement
- ✗ Maintain quality & mitigating risk



AI Assistant Role

- × Good software testing
- × Enhances and speeds up testing processes
- × AI assistant capabilities:
 - × Generative
 - × Transformation



Generative - Identity & Design

Test

- × Suggest/expand test scenario ideas
- × Completing, not replacing human test design abilities
- × Examine & adjust when needed



Generative - Identify & Design Test

Create test cases for the following user story and the acceptance criteria. The test must include positive test case (happy path) and negative test cases (not happy path, e.g. error)

```
<story>
```

```
...
```

```
...
```

```
</story>
```

Generative - Generate Test Data

- × Generating high-quality test data is hard
- × AI Assistant speed up test data creation
- × Convert / create data based on existing sample data



Generative - Create Test Script

- × AI assistant to create test script for testing tools
- × Don't blindly trust & use the generated script
- × As starting point
- × Use our knowledge & skills to modify the generated script



Transformative - Change Structure

- × Transform data from one structure to another
- × Examples:
 - × Plain text data into SQL statement
 - × Tabular data to JSON
 - × Other formats



Transformative - Translate Code

- × Convert code from one language to another
- × Keep the original code logic and flow
- × Always test the AI assistant output



ChatGPT & Github Copilot Introduction



AI Assistant Role

- × Change how software engineers work
- × AI assistant capabilities
- × This course uses *ChatGPT* and *Github Copilot*
- × About ChatGPT
- × About Github Copilot



ChatGPT vs. Github Copilot

- × Distinct purposes
- × **ChatGPT**
 - × Understanding & generating natural language text (e.g. English)
 - × Common tasks
- × **Github Copilot**
 - × Focus on code
 - × Specialized capabilities for code tasks



Best Practices

- × ChatGPT proficiency is on natural language processing
- × Github Copilot proficiency is on code
- × Best practices:
 - × Complementary usage
 - × Continuous feedback
 - × Not an absolute solution
 - × Ethical considerations



Github Copilot Considerations

- × Copilot learns from Github codebases
 - × Learn from other people codebase
- × Be aware of license & restrictions
- × Ensure generated code does not compromise privacy

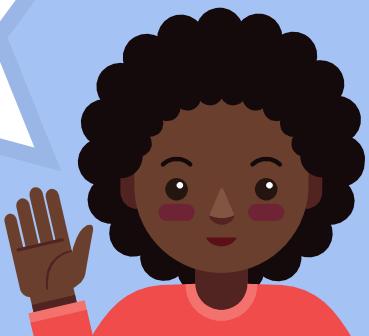


ChatGPT & Github Copilot Installation



ChatGPT

- × Access on chatgpt.com
- × Register and log in
- × Registration is optional
- × Free & paid version available
- × This course recorded using paid version
- × You can use either free or paid version



GitHub Copilot

- ✗ The course will not use GitHub for source code management
- ✗ GitHub Copilot requires GitHub account
- ✗ GitHub account is free
- ✗ GitHub Copilot: 30-day free trial, then need to pay
- ✗ Always check latest pricing update



Visual Studio Code

- × My laptop : Windows 11
- × Mac / Linux can be used
- × Use **Visual Studio Code**
- × Free from code.visualstudio.com
- × Separated lesson on Visual Studio Code + GitHub Copilot installation



Messaging System



Messaging System

- × Application-to-application (A2A) communication
- × Handle messages on organization
- × Integration



Example Use Case



Human Resource System

New employee, employee resign



Marketing System

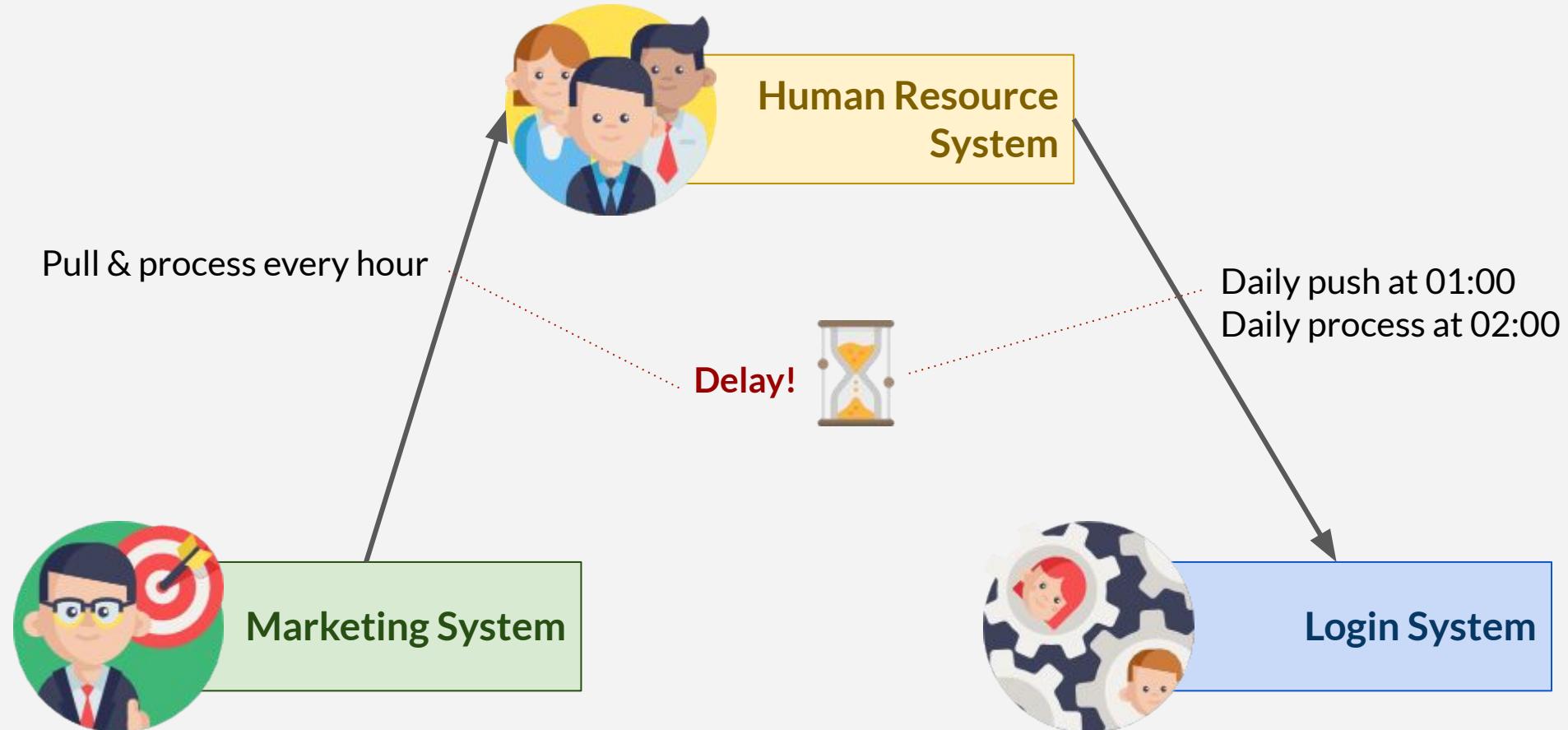
Has its own user management module



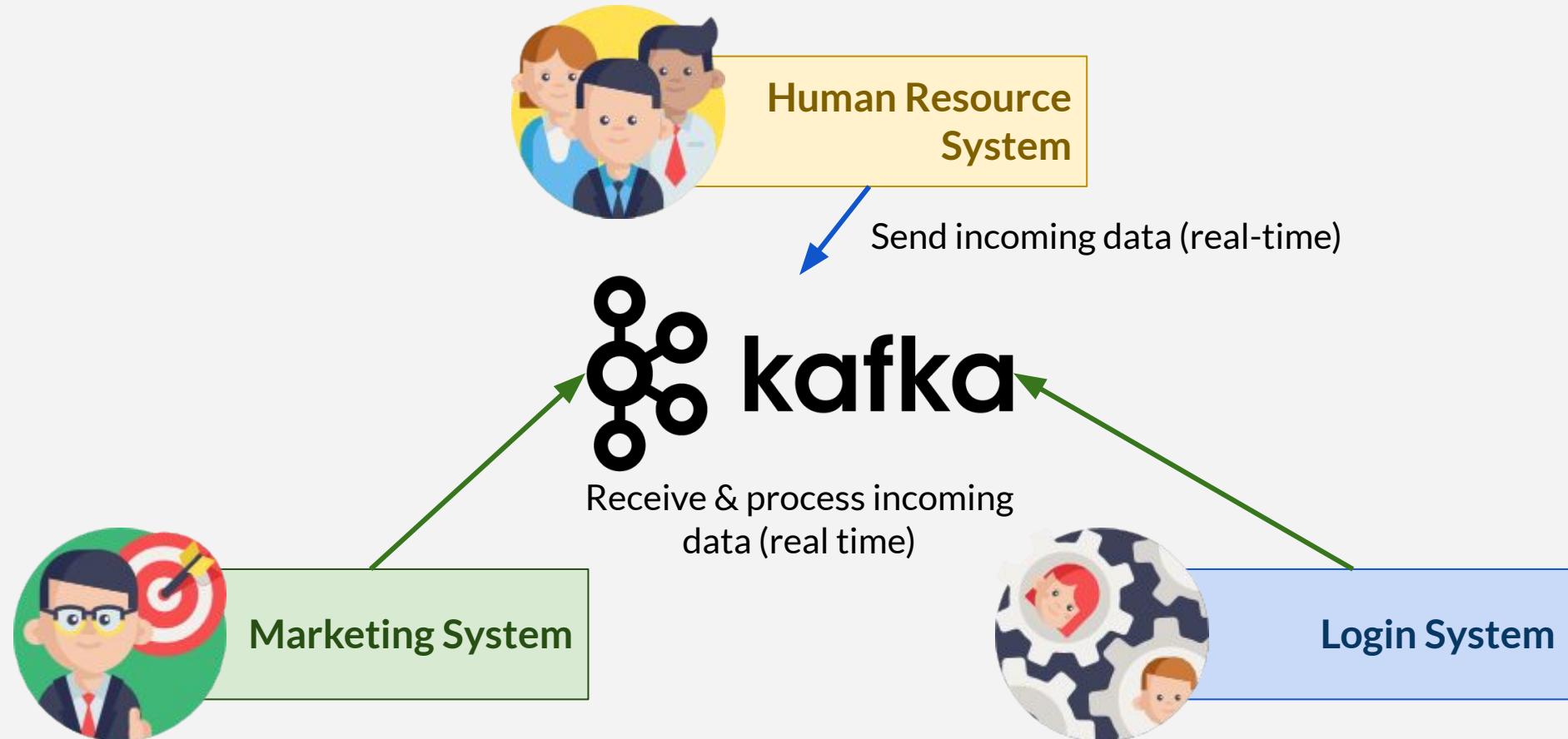
Login System

Used by other systems for store username/password

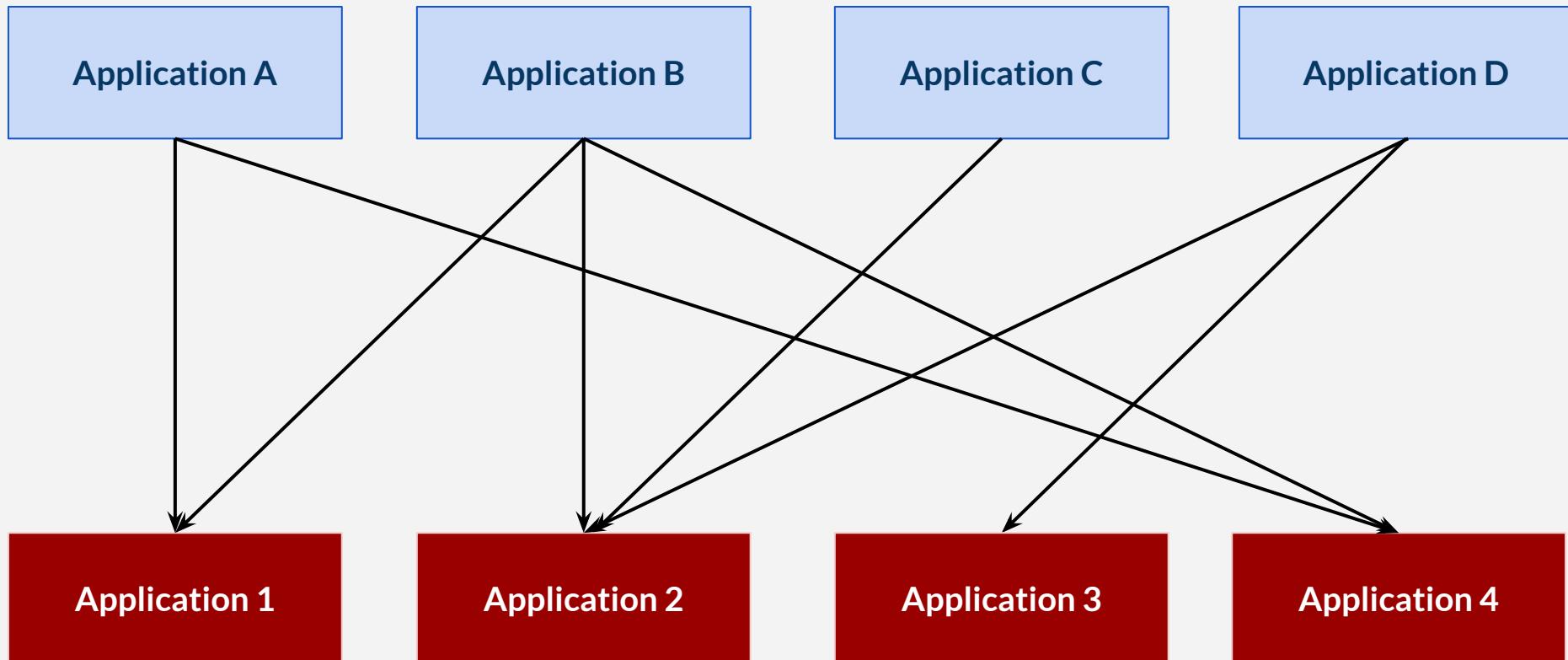
Without Messaging System



With Messaging System



Without Messaging System

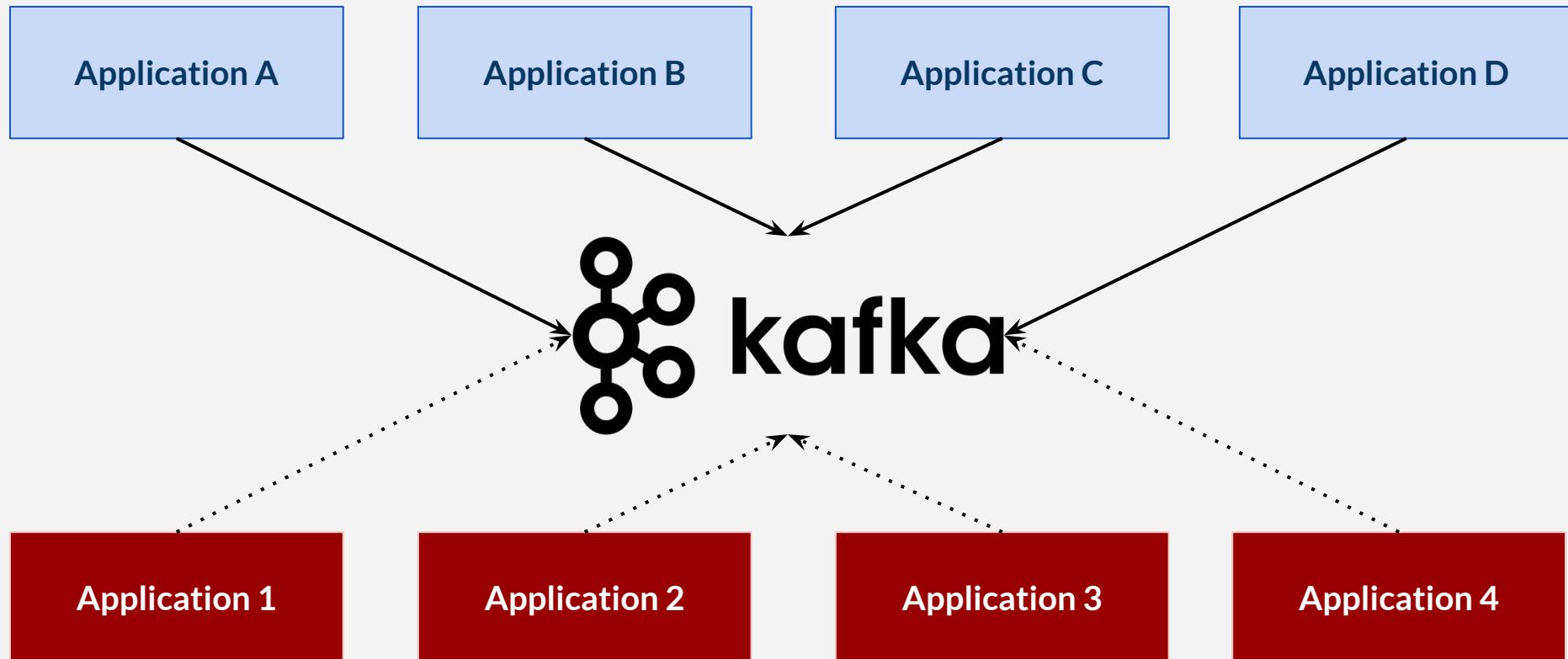


Problems?

- × Many integrations point
- × Different implementation
 - × Data protocol : database link, API, file
 - × Data format: binary? JSON? CSV? XML
- × Maintain connection



Kafka as Messaging System



Other Possible Use Cases

- × Every received payment data : create accounting journal, send notification to customer, inform logistic to send item
- × Every possible fraud data : temporarily suspend user account, send email to auditor, hold transaction
- × Avoid resource locking for time-consuming process



Kafka = Messaging System?

- ✗ Kafka is not only messaging system
- ✗ Detail in next lesson



Kafka - Introduction

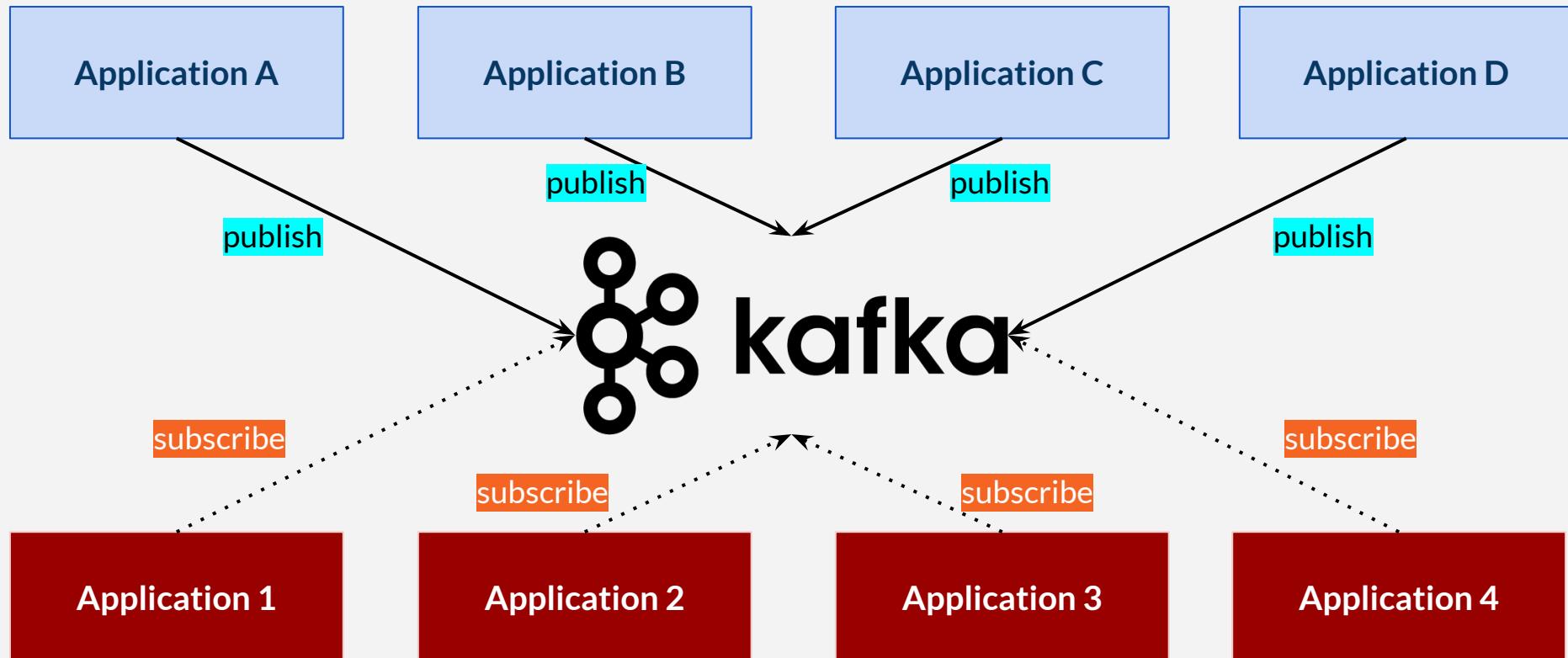


Event Streaming Platform

- × Kafka is event streaming platform
- × Capabilities:
 - × *To publish (or write) and subscribe to (or read) streams of events, including continuous import/export of your data from another system.*
 - × *To store streams of events durably and reliably as long as you want*
 - × *To process streams of events as they occur or retrospectively*
- × This lesson provides basic introduction



Publish and subscribe to streams of events

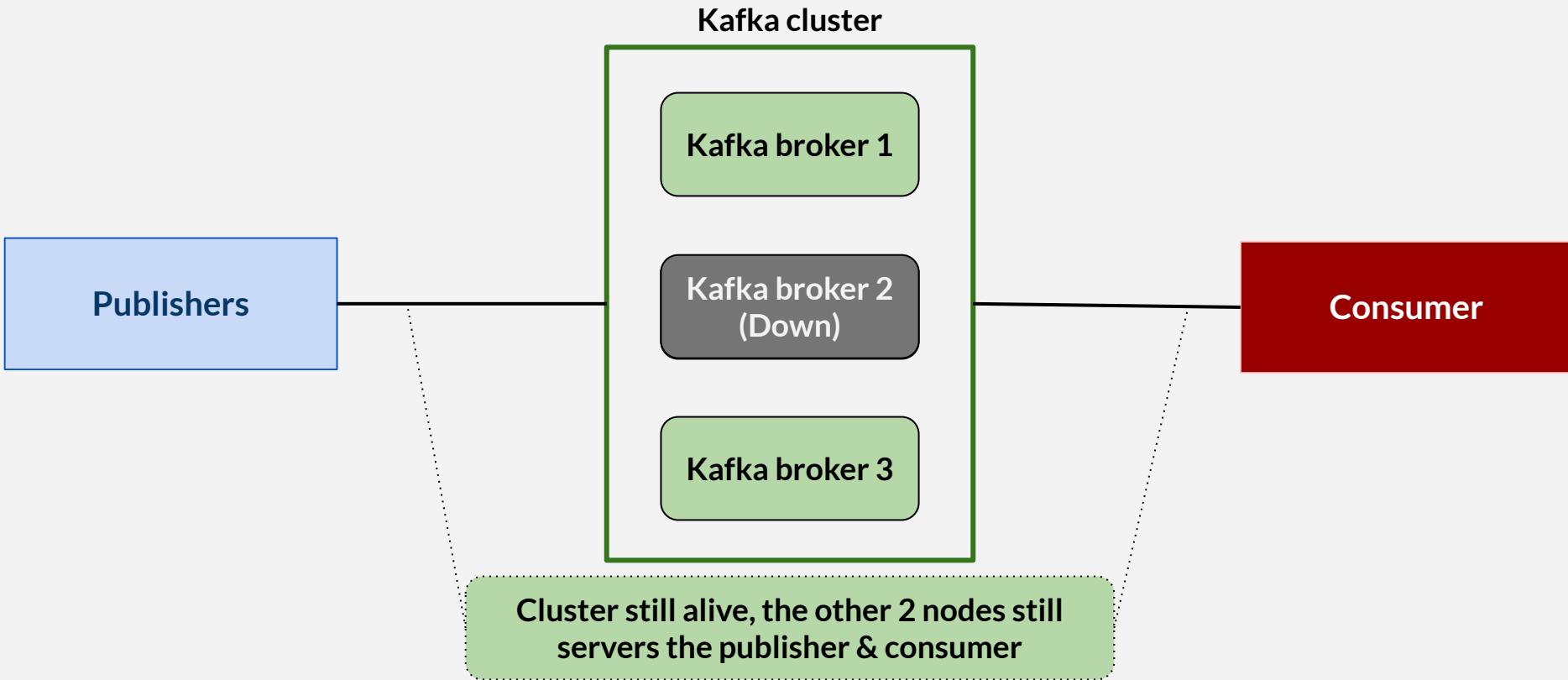


Kafka Cluster

- × Kafka machine also known as kafka **server** or **broker**
- × Single kafka broker is possible
- × Cluster (several brokers as unity) for high performance and reliability



Store events durably and reliably



Process streams of events as they occur



Kafka Use Cases

- × Central messaging system
- × Activity / application log
- × Storing IoT data
- × System decoupling
- × Asynchronous processing
- × Part of big data ecosystem

Kafka as data streaming mechanism
We need to write publisher & consumer



Kafka Distribution

- × Apache Kafka
- × Confluent (www.confluent.io)
- × Start with basic Kafka
- × Continue with Confluent platform



Java & Spring



Java Programming Language

- × One of most-used programming language
- × Regular update
- × Runs on multiple platforms: Windows, Mac, Linux, Android
- × Oracle & Open source implementation (OpenJDK, Amazon Coretto)



Spring Framework

- × Spring framework helps a lot
- × Testable
- × Free & open source
- × Has almost everything required for enterprise programming
- × Complex configuration on early days
- × Spring boot simplify configurations a lot
- × Start fast with spring boot
- × <https://spring.io>



Spring Technology Stack

- × Microservice & reactive support
- × SQL & NoSQL database
- × Spring cloud for microservice
- × Messaging system (Kafka, RabbitMQ)
- × And many more (<https://spring.io/projects>)



Gradle

- ✗ Build tools & dependency management
- ✗ Maintaining library manually is nightmare
- ✗ Gradle manages library dependencies
- ✗ Build distributable package
- ✗ Spring Boot generates initial script



Visual Studio Code

- × Integrated Development Environment
- × Code completion
- × Automate processes
- × Easy debugging
- × Build tools & dependency management using
Gradle
- × Extensions



Install Java



Java

- ✗ Oracle (need pay license for production)
- ✗ OpenJDK, Amazon Coretto (free for production)
- ✗ This course uses Amazon Coretto
- ✗ This course requires minimum Java 17
- ✗ Course recorded using Java 21



Kafka Installation

For Development Environment



How to Install Kafka

- ✗ Binaries / native installation
- ✗ Docker **This course**
- ✗ My machine : Windows 11
- ✗ Suitable for development, not for production



What & Why Docker

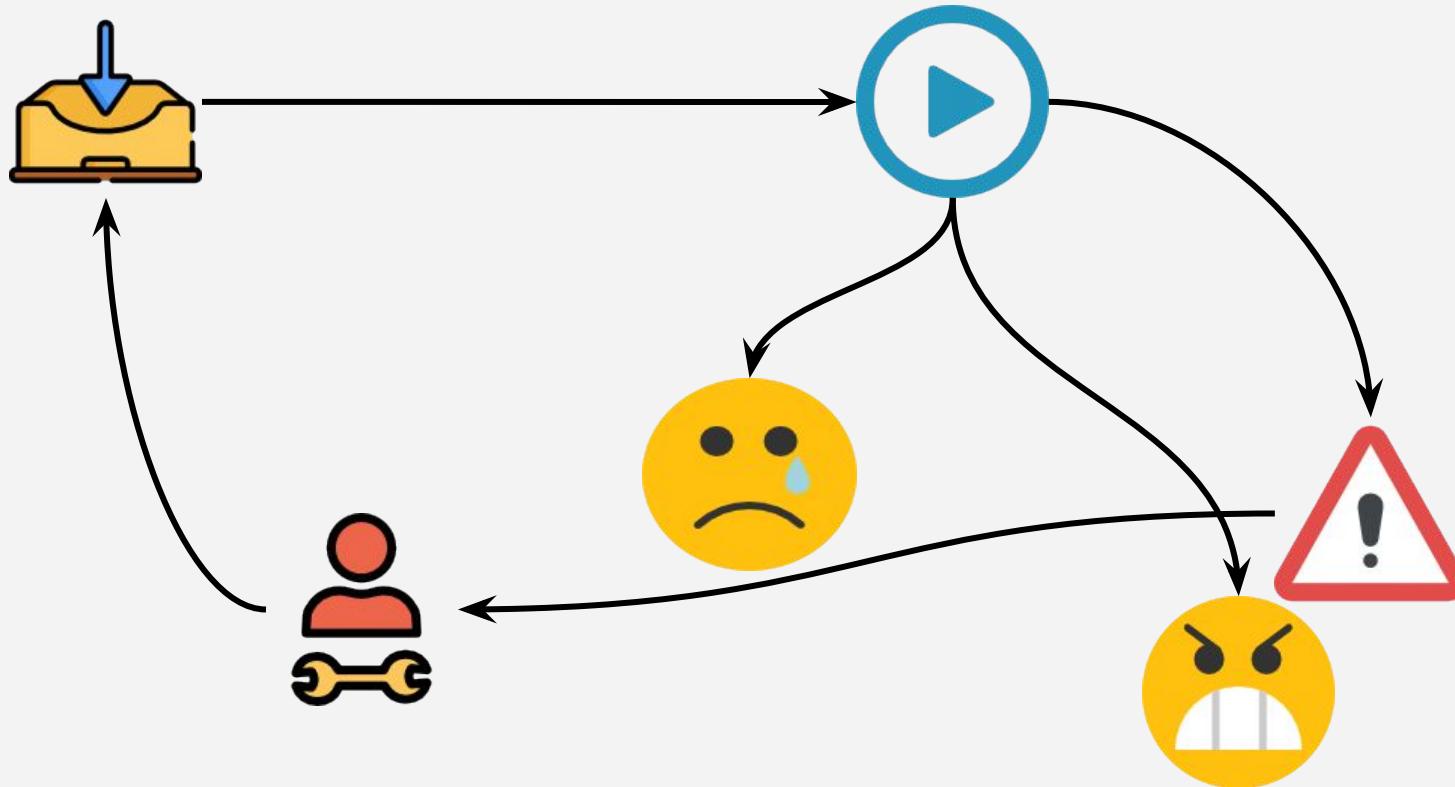




Installation Process (Happy Path)



Installation Process (NOT Happy Path)



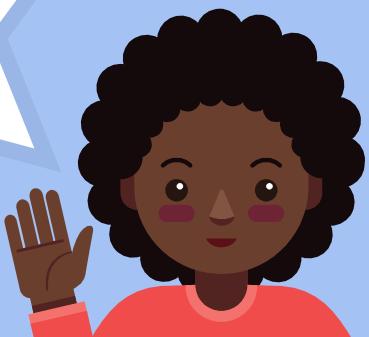
Docker as Alternative

- ✗ Docker handle dependencies
- ✗ Run anywhere (as long as it has docker runtime)
- ✗ Many images on docker hub



Docker in This Course

- ✗ Not a Docker course
- ✗ For easy installation
- ✗ Several basic Docker concepts
- ✗ Image : contains everything to run the application
- ✗ Container : instance of image, runs an application



Install & Run Kafka Using Docker



Docker Compose

- × More than 1 container to run application
- × Example
- × This course uses several containers
- × Script to run several containers at once
- × Using docker compose



Kafka on Docker

- × Kafka using Docker compose
- × Docker compose script available on lesson **Resource & Reference** (last section)
 - × One or more docker containers
 - × Configure & run using docker-compose yml script



Kafka & Zookeeper

- × Kafka older than 3.4 requires Zookeeper
 - × Coordinates Kafka brokers
 - × Manage metadata, ...
- × Kafka 3.4 or newer does not need Zookeeper
- × This course does not use Zookeeper



```
#> docker-compose up
```

```
#> docker-compose -f [script-file] -p [project] up -d
```

File, to replace [script-file]	Replace [project]	Complete command
docker-compose-core.yml	core	#> docker-compose -f docker-compose-core.yml -p core up -d
docker-compose-connect.yml	connect	#> docker-compose -f docker-compose-connect.yml -p connect up -d
docker-compose-connect-sample.yml	connect-sample	#> docker-compose -f docker-compose-connect-sample.yml -p connect-sample up -d
docker-compose-ui.yml	kafka-ui	#> docker-compose -f docker-compose-ui.yml -p kafka-ui up -d
docker-compose-full.yml	full	#> docker-compose -f docker-compose-full.yml -p full up -d
docker-compose-full-sample.yml	full-sample	#> docker-compose -f docker-compose-full-sample.yml -p full-sample up -d

```
#> docker-compose -f [script-file] -p [project] down
```

Part	What to run (in sequence)
1 - core kafka	<pre>#> docker-compose -f docker-compose-core.yml -p core up -d</pre>
2 - kafka connect	<pre>#> docker-compose -f docker-compose-core.yml -p core down #> docker-compose -f docker-compose-connect.yml -p connect up -d #> docker-compose -f docker-compose-connect-sample.yml -p connect-sample up -d</pre>
3 - kafka user interface	<pre>#> docker-compose -f docker-compose-connect.yml -p connect down #> docker-compose -f docker-compose-connect-sample.yml -p connect-sample down #> docker-compose -f docker-compose-ui.yml -p kafka-ui up -d</pre>
4 - kafka full (schema registry, REST proxy, ksqlDB, kafka connect, UI)	<pre>#> docker-compose -f docker-compose-ui.yml -p kafka-ui down #> docker-compose -f docker-compose-full.yml -p full up -d #> docker-compose -f docker-compose-full-sample.yml -p full-sample up -d</pre>

Install Visual Studio Code



Visual Studio Code

- × My laptop : Windows 11
- × Mac / Linux can be used
- × Use **Visual Studio Code**
- × Free from code.visualstudio.com



Visual Studio Code

With GitHub Copilot



Visual Studio Code + GitHub Copilot

- × Lesson shows VS Code + GitHub Copilot for Java projects
- × The course uses Visual Studio Code with GitHub Copilot enabled
- × Differences if not using GitHub Copilot
 - × With GitHub Copilot: longer, more complex autocomplete suggestions
 - × Without GitHub Copilot: standard autocomplete
 - × Other GitHub Copilot related feature (e.g. Copilot Chat)



Visual Studio Code + GitHub Copilot

- ✗ This lesson can be skipped if you are not using GitHub Copilot
- ✗ Using GitHub Copilot in the course is optional
- ✗ Source code & explanation is usable with/without GitHub Copilot
- ✗ Additional knowledge of using GitHub Copilot
- ✗ 30-day free trial is available when the video recorded (can change anytime)



Kafka Basic Concepts



In This Section

- × Kafka theory
- × **Recommended** : follow all lessons in this section to avoid confusion

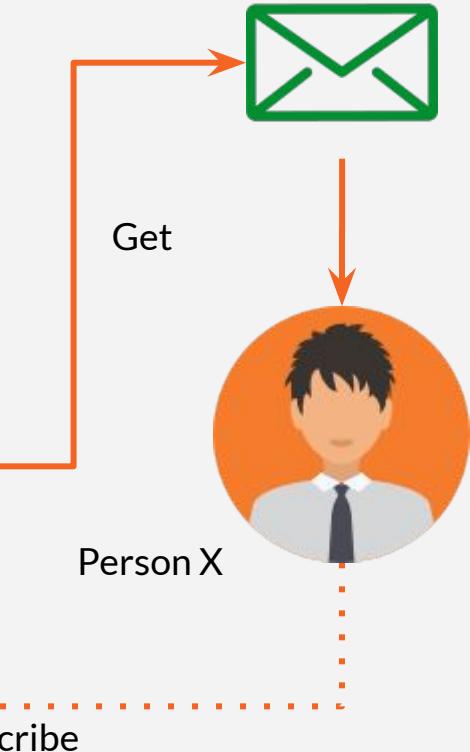
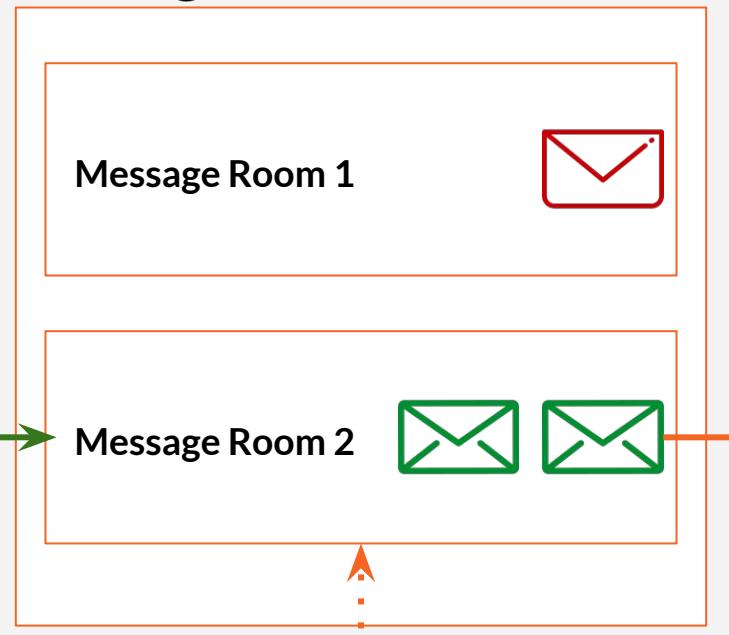


Kafka Analogy

A Simplified Explanation



"Unique" Warehouse



Publisher /
producer



↓
Publish /
produce



"Unique" Warehouse



Topic / partition /
offset



Topic / partition /
offset



Get



Subscriber /
consumer /
listener

Subscribe / consume / listen

Synonyms

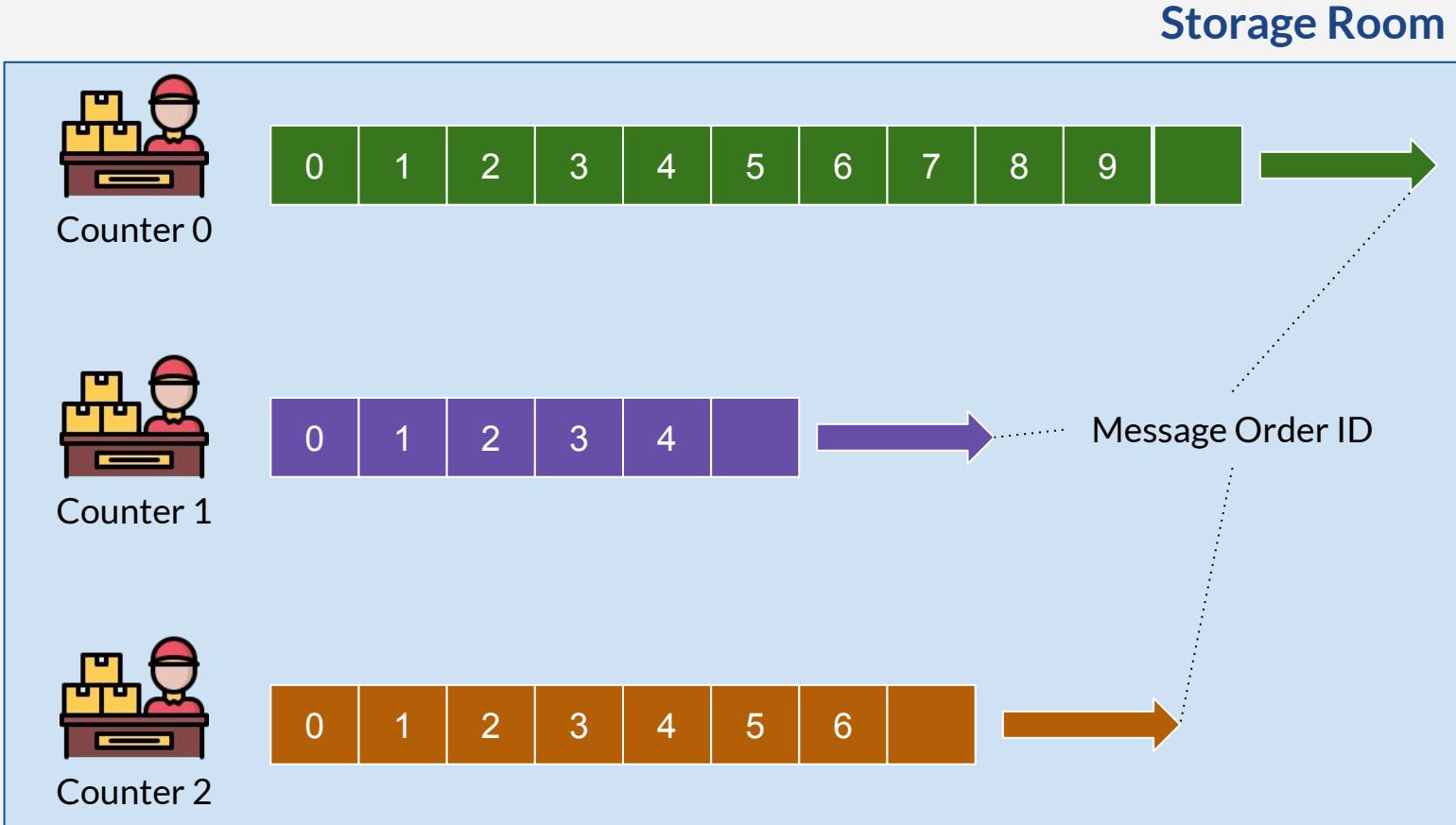
- ✗ Producer / publisher
- ✗ Send / publish / produce message
- ✗ Subscriber / consumer / listener
- ✗ Subscribe / consume / listen
- ✗ Don't get confused with terminology synonyms



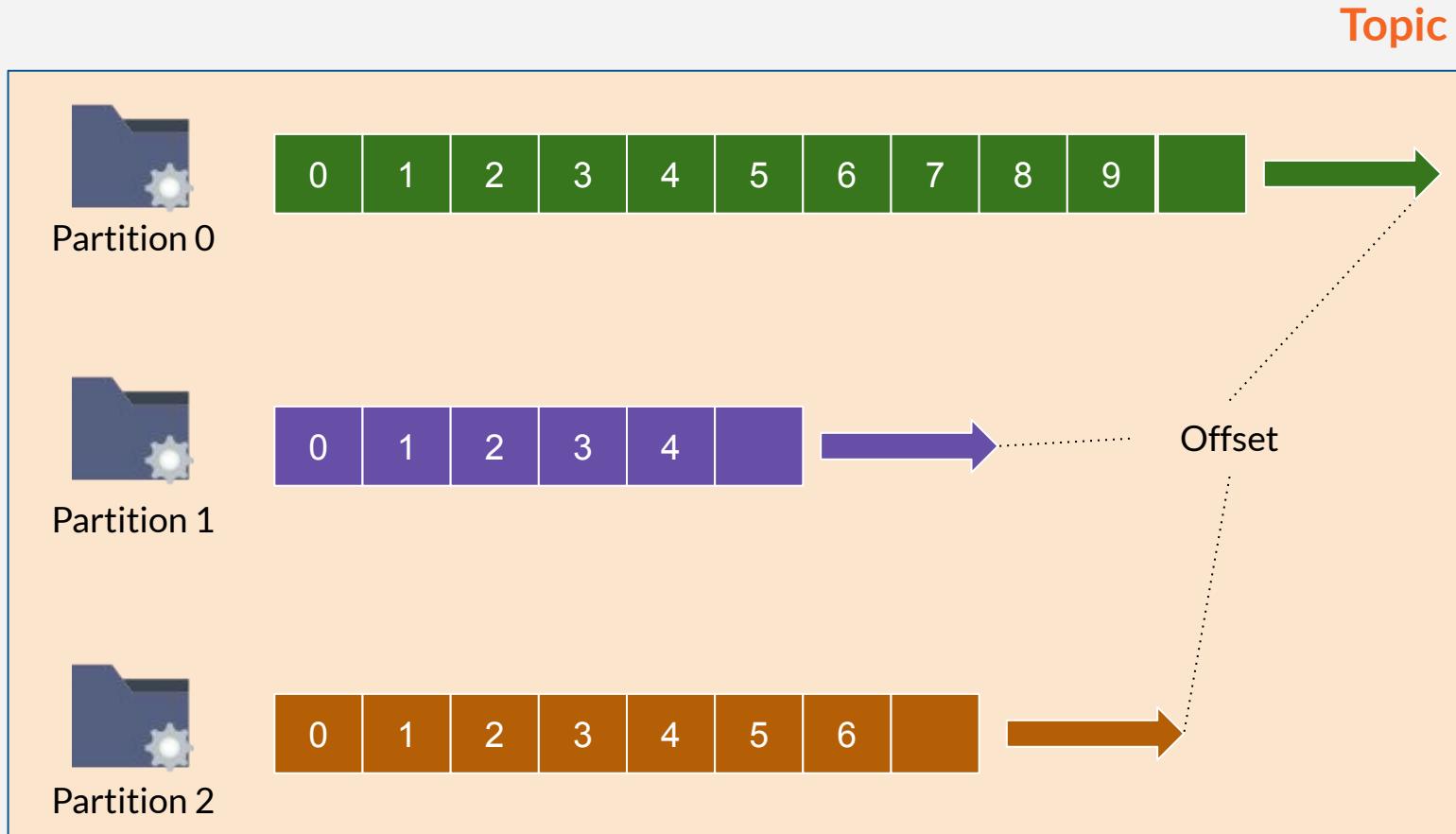
Topic, Partition & Offset



Storage Room Structure



Storage Room Structure (in Kafka)



Topic

- × You can have many topic as you need
- × Messages stored for certain retention period
 - × Default is 7 days
 - × Able to set value
- × Each topic has name
- × Message is *immutable*



Partition & Offset

- × 1 topic : 1 or more partition(s)
- × Partition is a way to achieve parallelism
- × Messages stored in order for each partition
(guaranteed per partition)
- × Order across partition not guaranteed
- × Each message has offset started from zero
- × Offset per partition
 - × Partition 0, offset 0 | 1 | 2 | 3 |
 - × Partition 1, offset 0 | 1 | 2 | 3 |
 - × Partition 2, offset 0 | 1 | 2 | 3 |
- × Each partition is independent



Topic & Partition

- ✗ Define partition when creates topic
- ✗ Can add partition later
- ✗ Can't delete partition later
- ✗ Because delete partition = delete data (cause data loss)



Example



- Fetch data every minute
- Topic : **t-commodity-price**
- Message structure: timestamp, commodity type (iron / gold /), price
- Initial three partitions (can be add more later)
- One consumer per partition
- More detail later

Producer

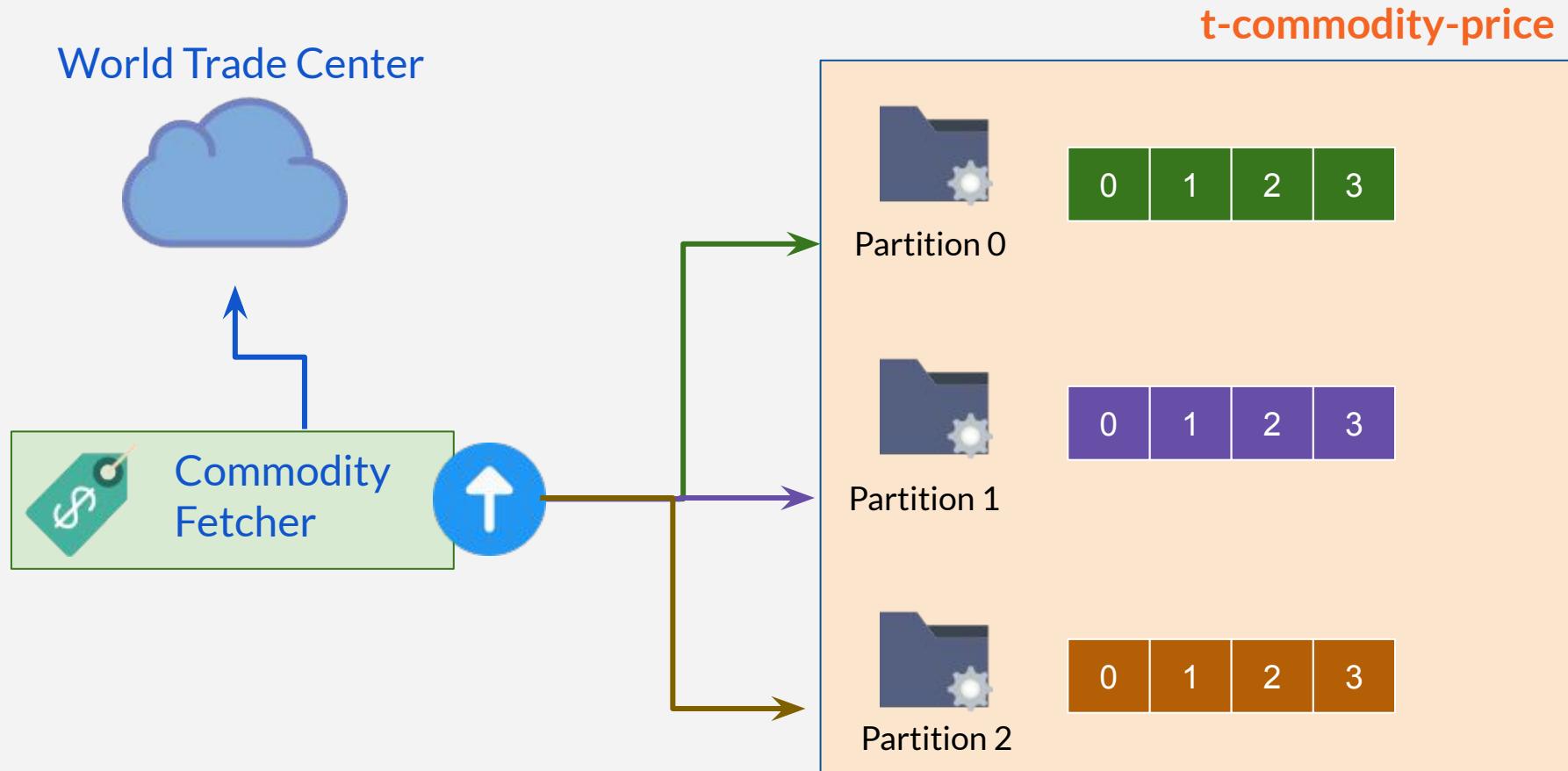


Kafka Producer

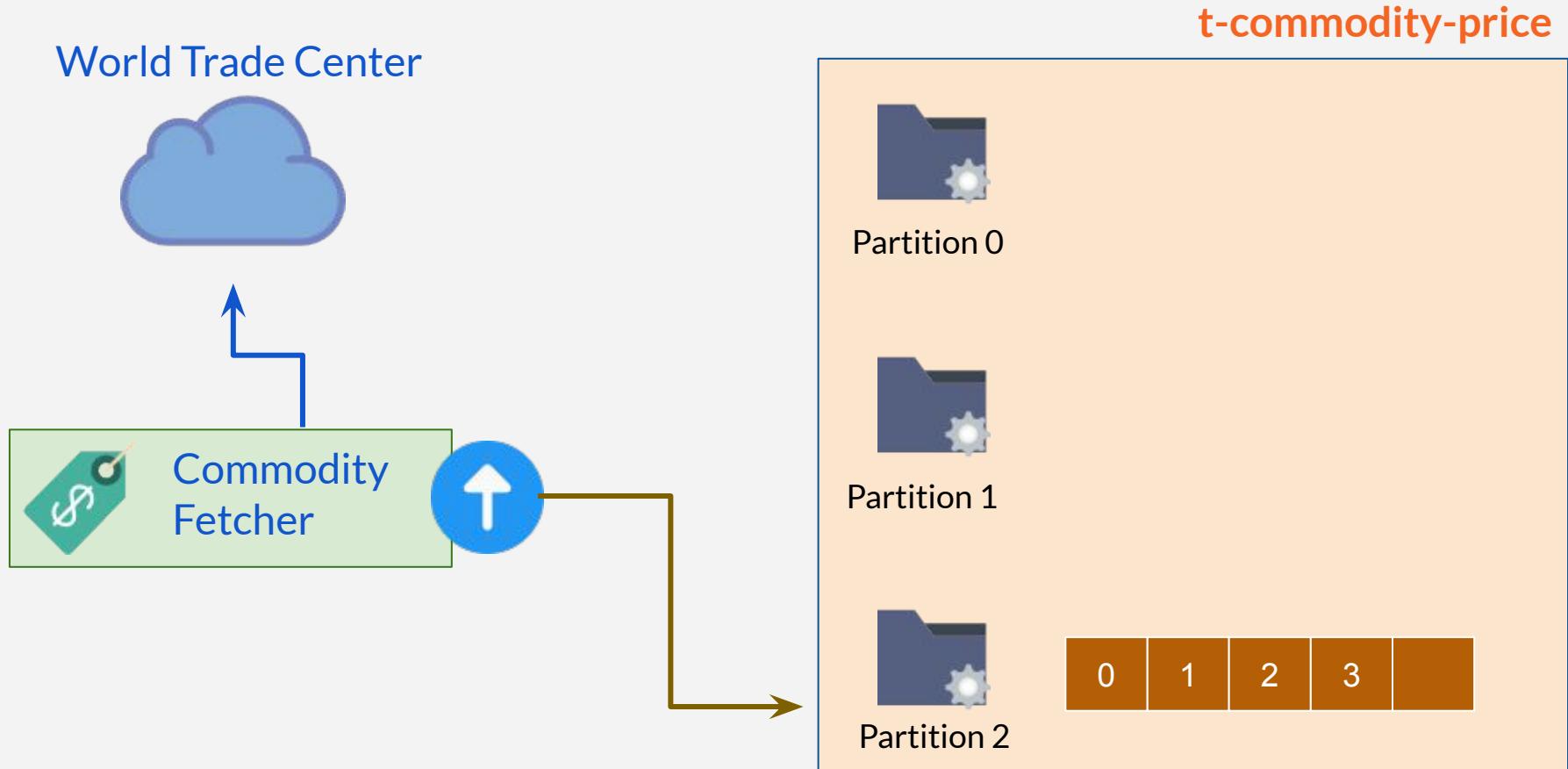
- × Producer send message to Kafka
- × Sends to which topic + message content
- × Kafka will automatically select partition (we can override this behaviour)



Kafka - How Producer Works

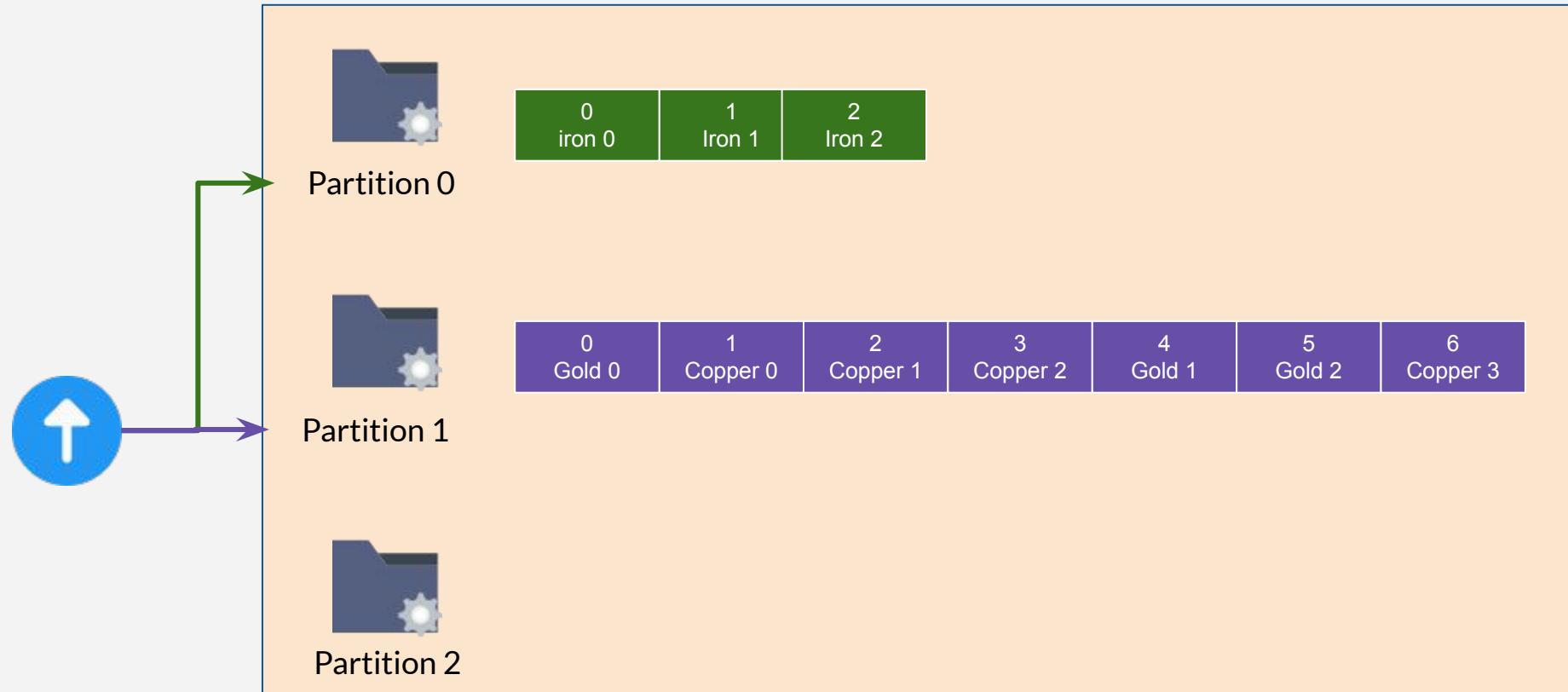


Kafka - Define Partition for Producer



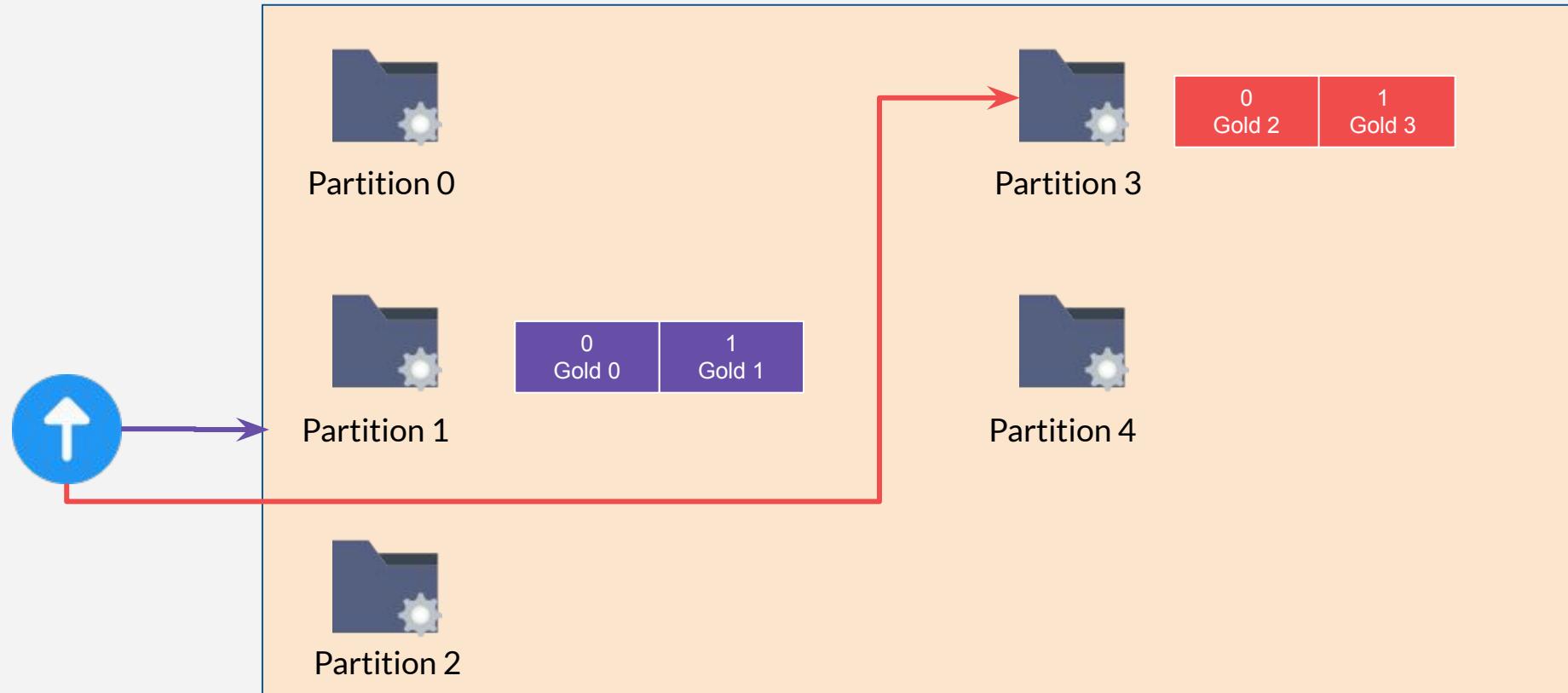
Kafka - Producer Key

t-commodity-price



Kafka - Producer Key When Add Partition

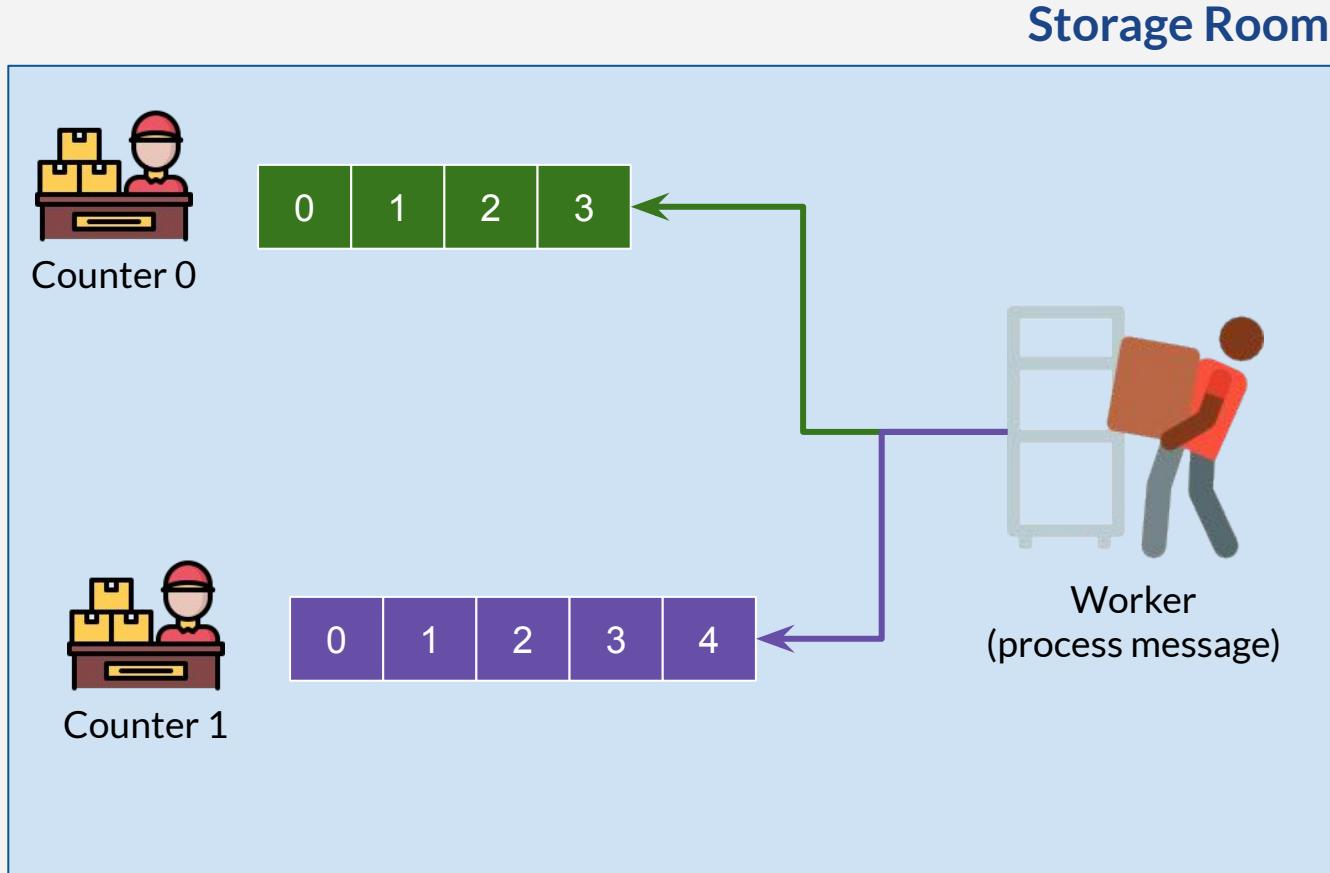
t-commodity-price



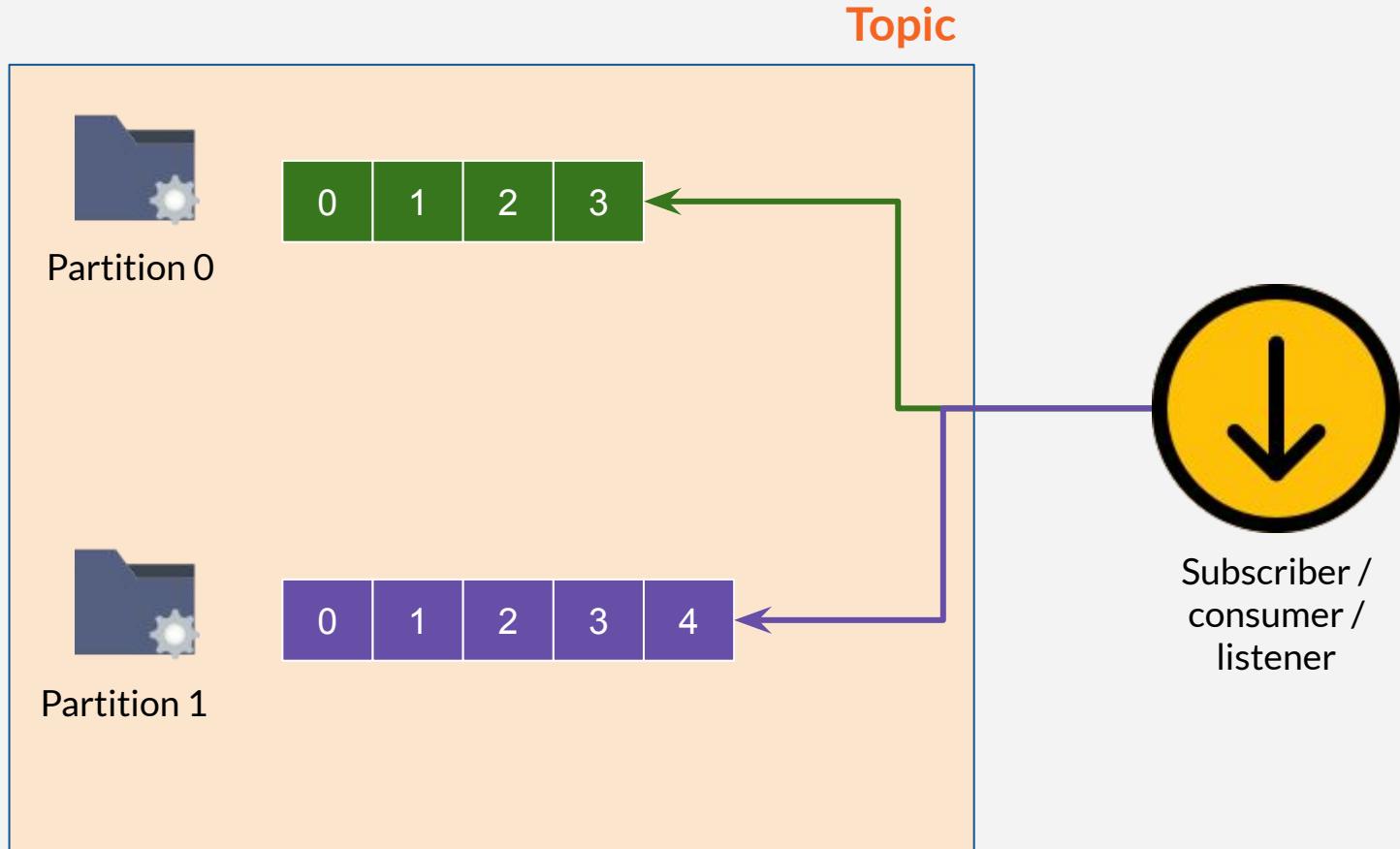
Consumer & Consumer Group



Storage Room - Processing Message



Kafka - Processing Message

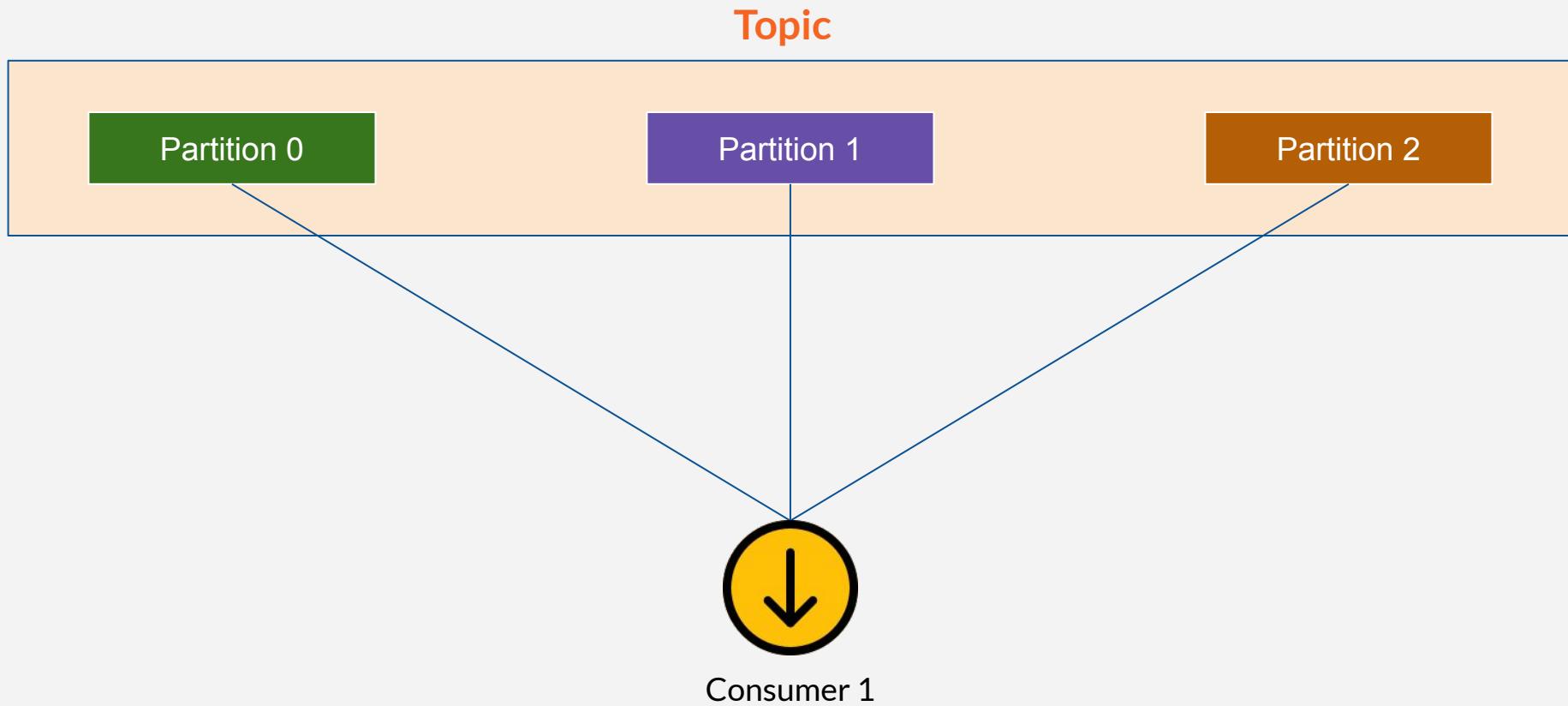


Kafka Consumer

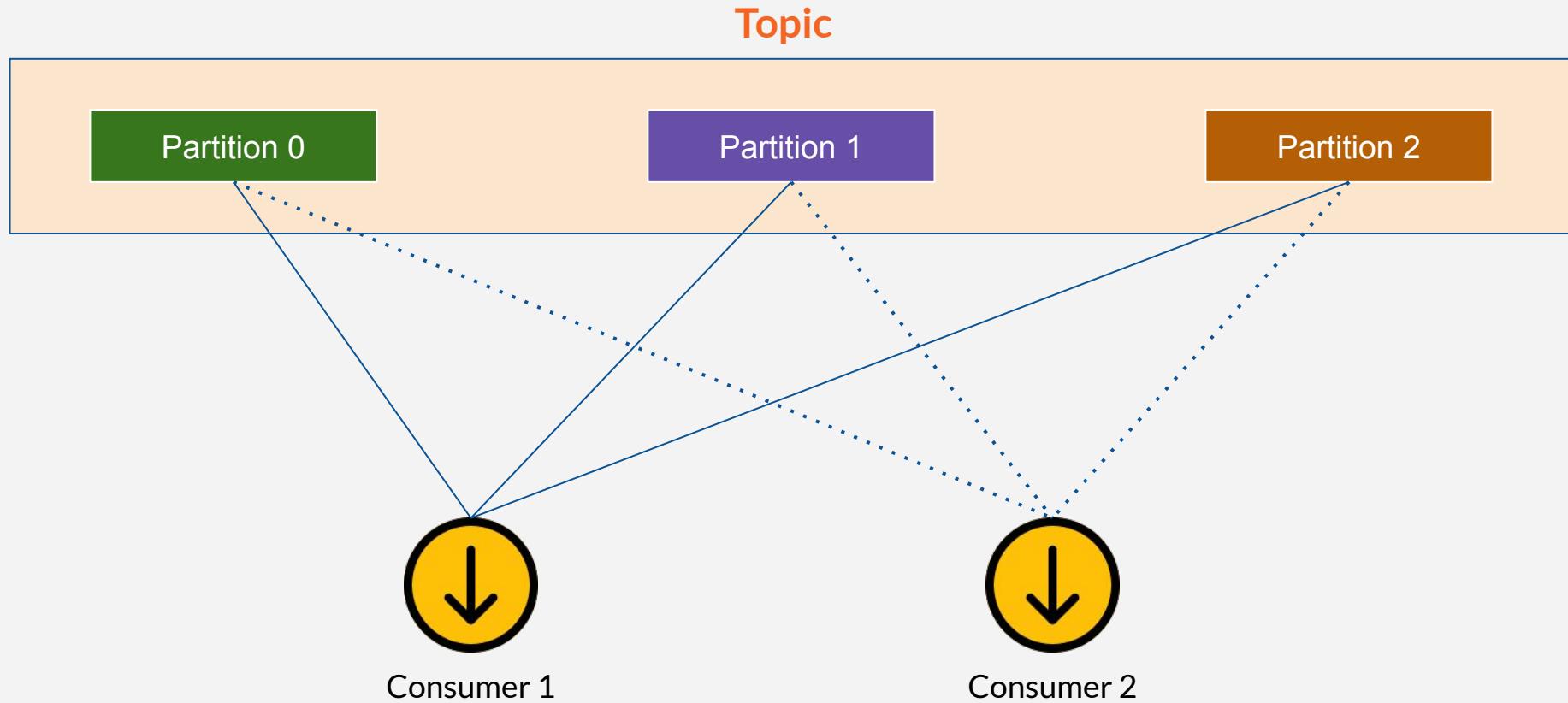
- × Consumer guaranteed to read data in order for each partition
- × Order is incrementing by offset (low offset to high), cannot reverse
- × Each partition maximum one consumer per consumer group
- × One consumer can read from more than one partition



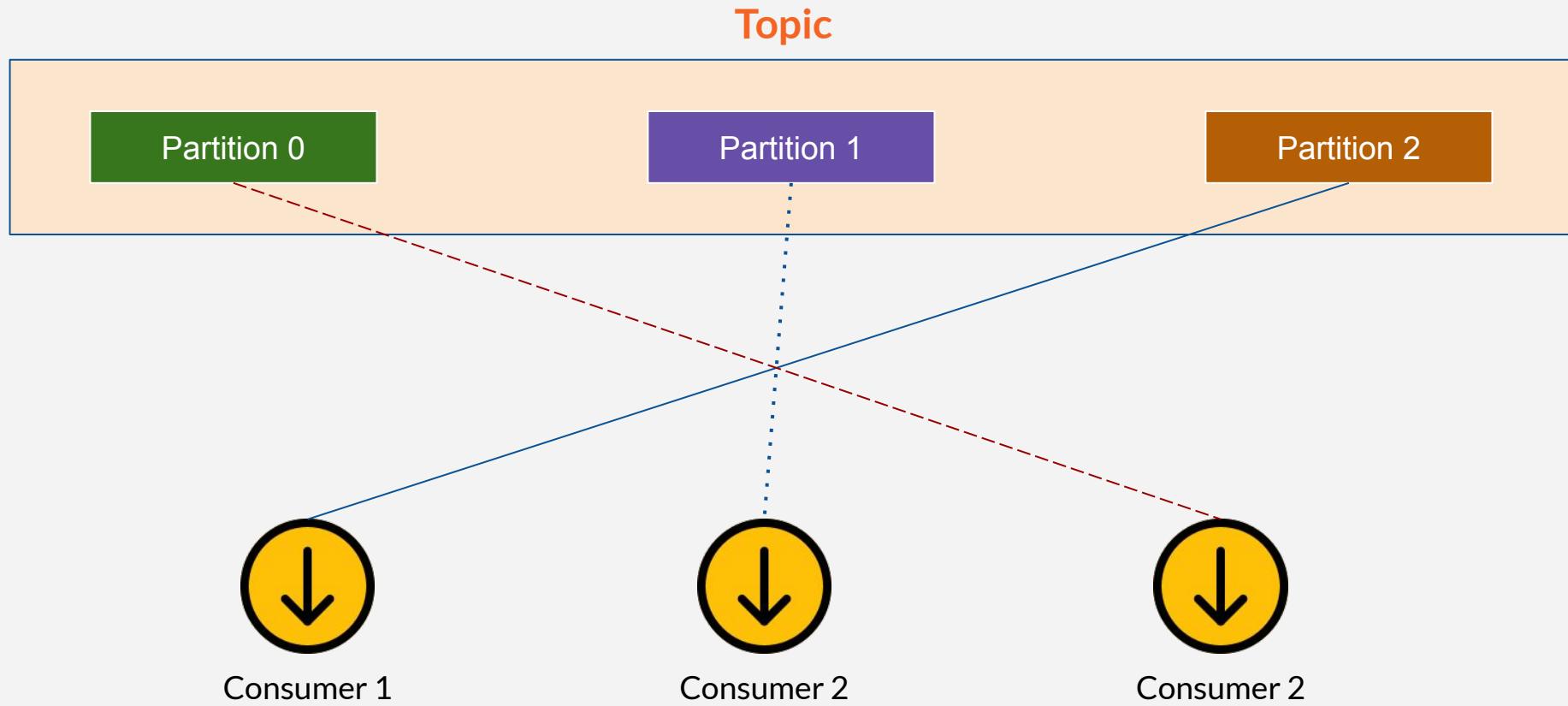
Single Consumer for All Partitions



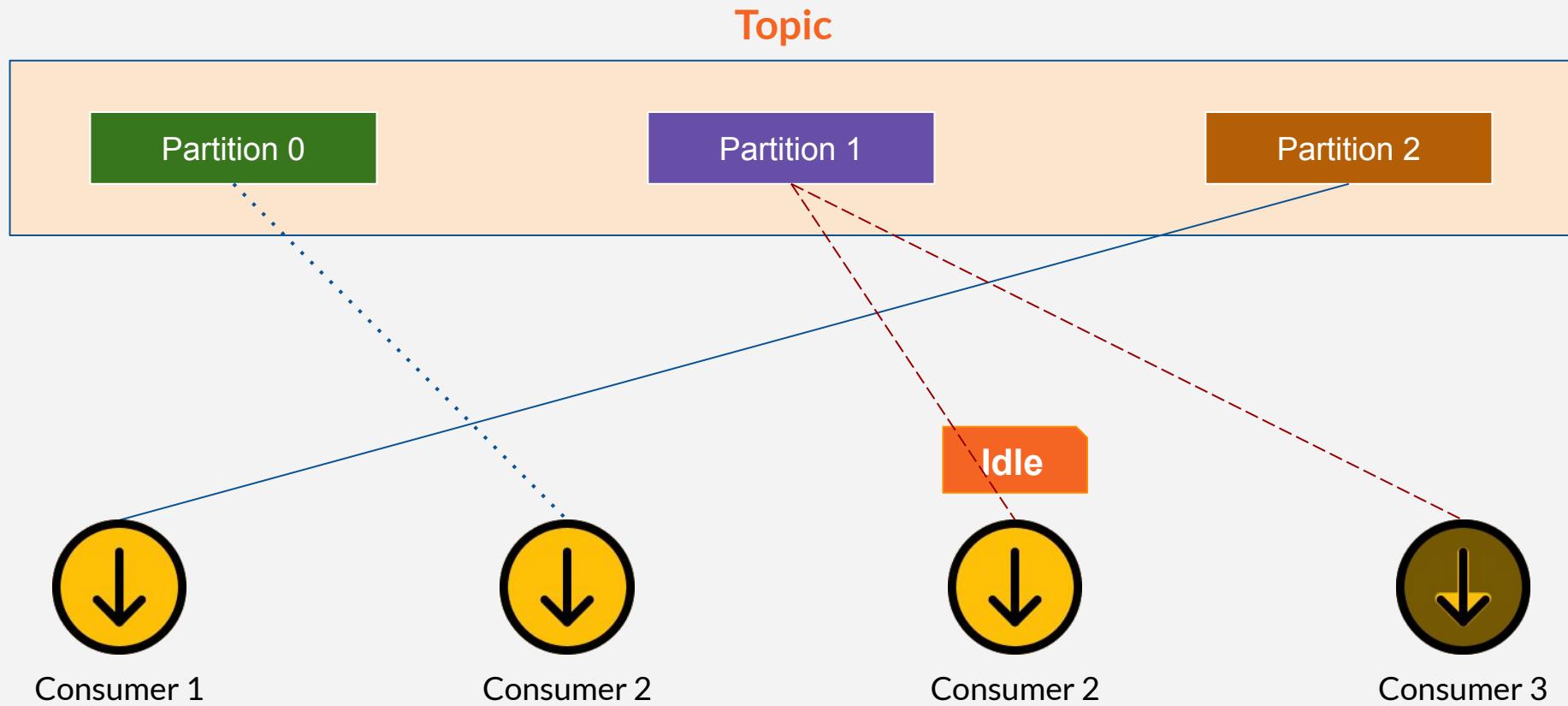
Consumer < Partition



Consumer = Partition



Consumer > Partition

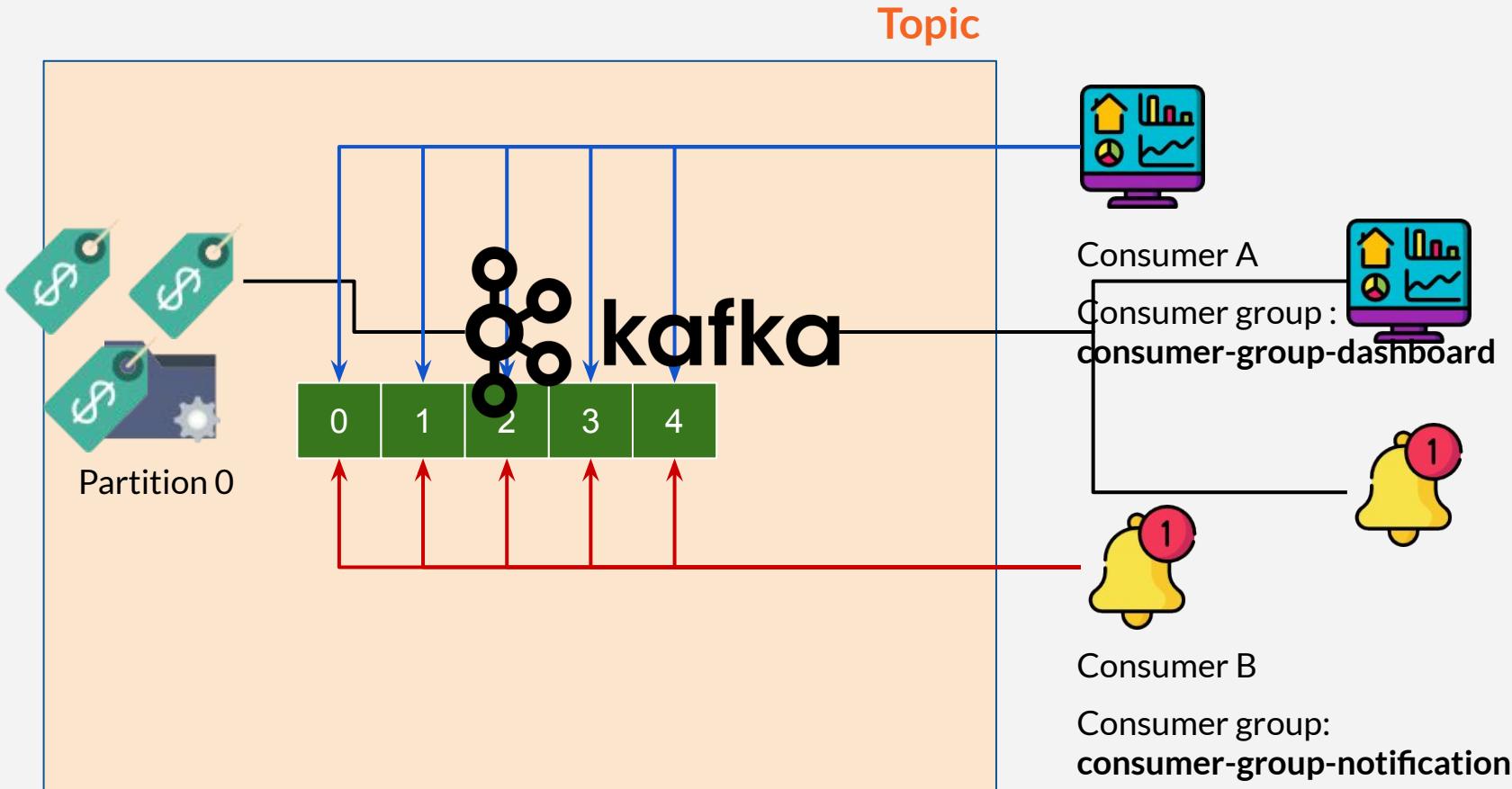


Kafka Consumer Group

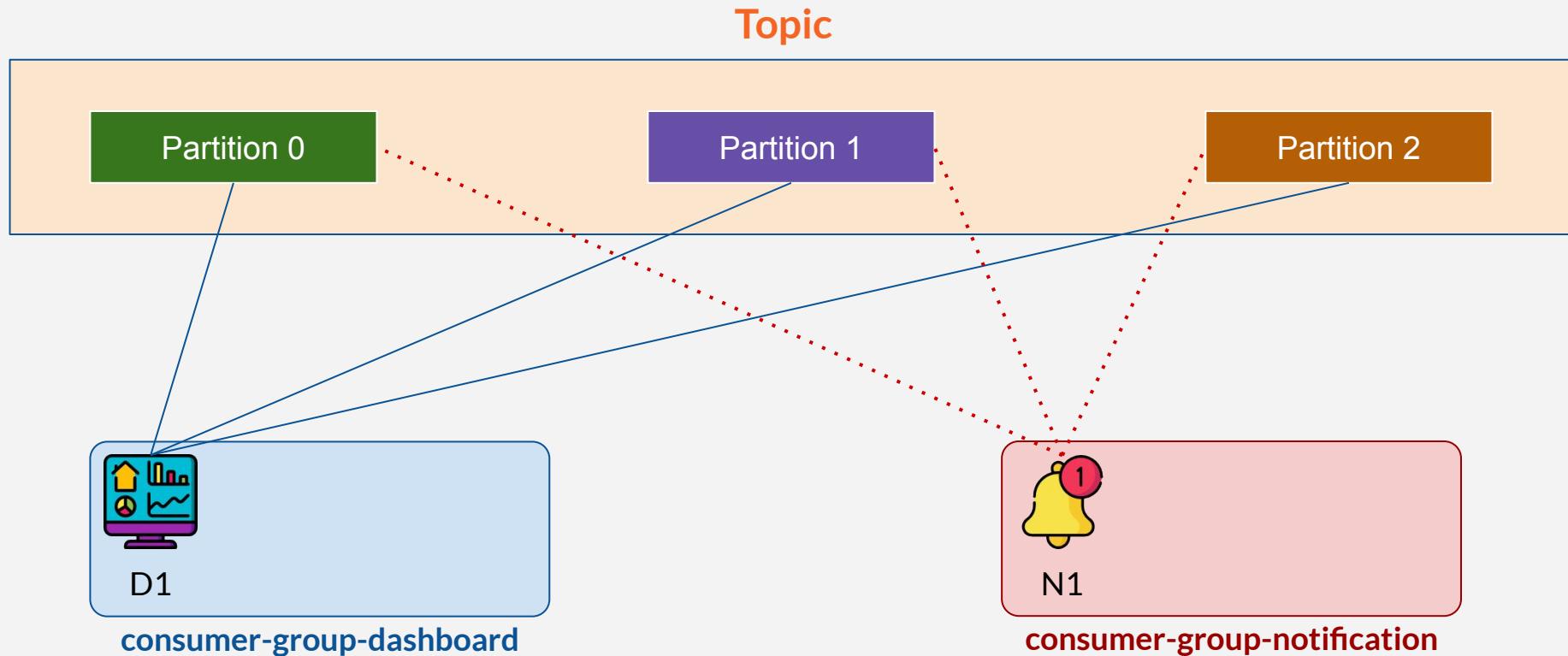
- × Group consumers based on functionalities
- × Partition & consumer group for parallel consume



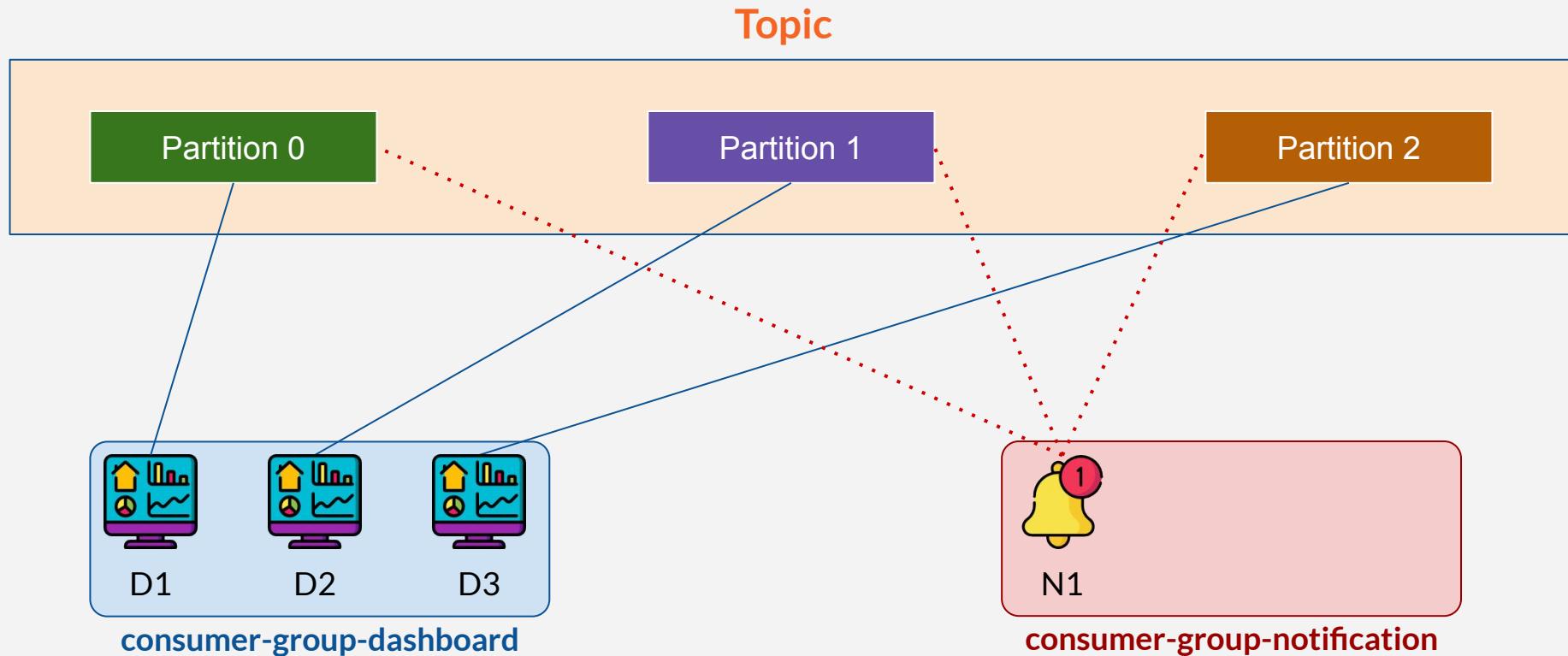
Kafka - Consumer Group



Consumer Group is Independent



Consumer Group is Independent



Consumer Offset & Delivery Semantic

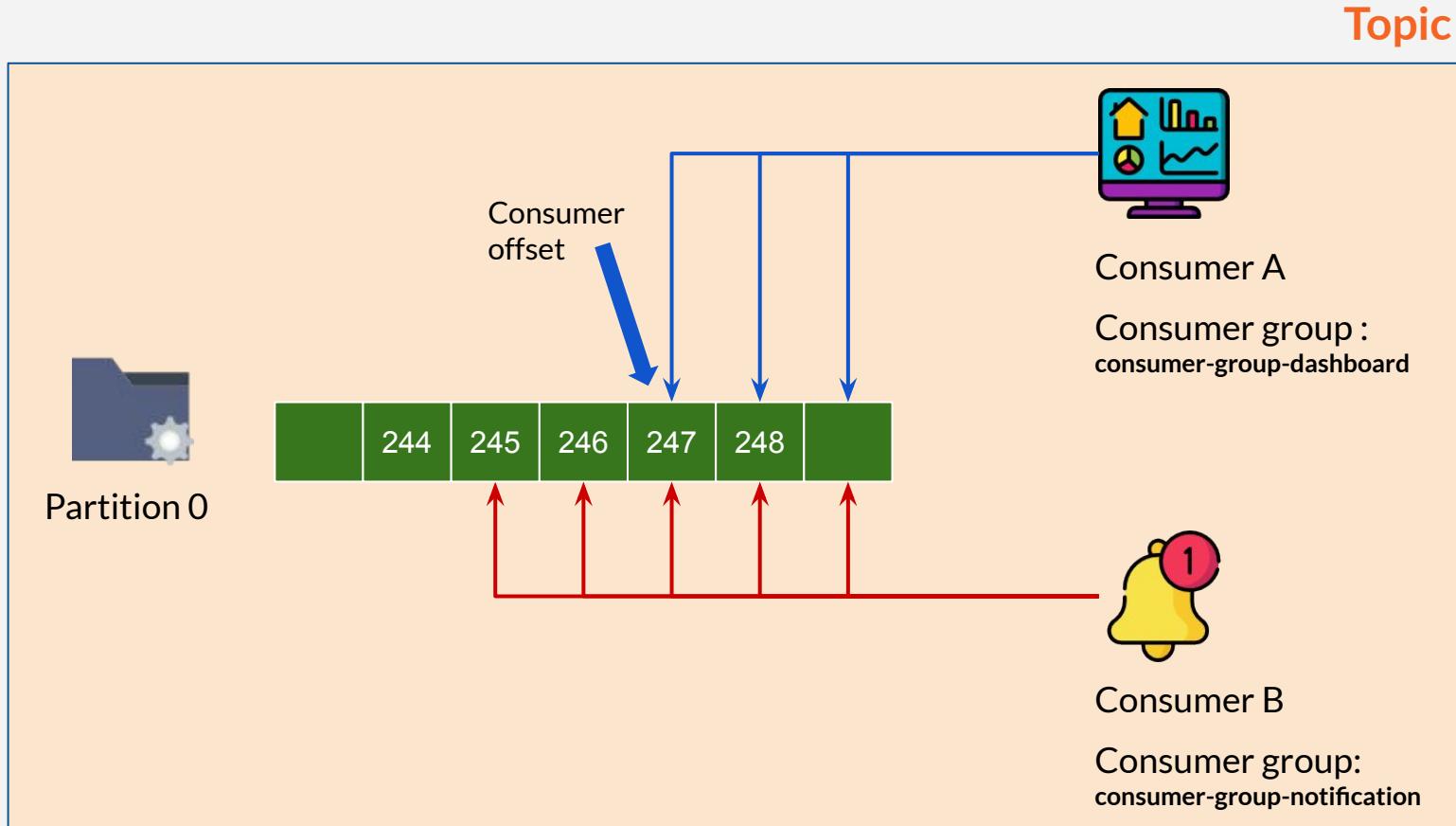


Consumer Offset

- × Remember the offset that has been read
- × Unique for each partition
- × Kafka save consumer offset
- × Checkpoint that indicates last read



Kafka - Consumer Offset



Delivery Semantic

- × Consumer choose when to saving (commit) offset
- × At-most-once
 - × Zero (unprocessed) or once (processed)
 - × Possible message loss (on error process)
- × At-least-once
 - × One or more
 - × Something wrong, able to re-process
 - × Create idempotent consumer
- × Exactly-once
 - × Once only
 - × Hard to implement



Zookeeper



Zookeeper

- × Kafka designed for highly available
- × Kafka cluster : group of kafka brokers (servers)
- × Zookeeper manage those brokers
 - × Add broker
 - × Broker dies
 - × Synchronize data
- × This course will not focus on zookeeper
- × Zookeeper server must runs for kafka to start
(older Kafka)
- × No need zookeeper to start Kafka (version 3.4+)
- × This course does not use Zookeeper



Spring Boot



What is Spring Boot?

- × Accelerate coding process
- × Handles most of configurations
- × Predefined configurations on
application.properties / application.yml
- × Automatic dependency management
- × Online project generator



Spring Initializr

- × Spring Boot project generator
- × Go to : <https://start.spring.io/>
- × Generate two projects : producer and consumer



Spring Boot Configuration

- Under *src/main/resources*
- Filename: *application.properties* or *application.yml*
- Prefer yml format for readability (optional)

application.properties

```
spring.kafka.producer.batch-size=10  
spring.kafka.producer.client-id=my-id
```

application.yml

```
spring:  
  kafka:  
    producer:  
      batch-size: 10  
      client-id: my-id
```

Hello Kafka

Topic & Partition



Executing Kafka Commands

- × Kafka server is console-based, no UI
- × Execute kafka commands is different for native install / docker
- × I use Docker installation on top of Windows
- × Installation
 - × Docker : follow this lesson
 - × Linux : [kafka-home]/bin/[command].sh
 - × Windows: [kafka-home]/bin/windows/[command].bat



Hello Kafka

Java Spring Code



Hello Kafka

With AI Assistant



Working With AI Assistant

- × Send prompt using dedicated chat or inline chat
- × Ensure code structure remains valid
- × Don't trust the AI assistant blindly
- × Human is still the decision maker
- × ***AI assistant's output is non-deterministic!***



Consumer is Real Time Indeed



Consumer is Real Time Indeed With AI Assistant



IMPORTANT

Coding With AI Assistant



Working With AI Assistant

- × Visual Studio Code + GitHub Copilot:
 - × Autocomplete (press TAB to accept)
 - × Chat window
 - × Inline chat (press CTRL+I or Command+I)
 - × Source code comment



Working With Github Copilot

- × Reality: work together with AI assistant
- × Using Github Copilot:
 - a. *Define our needs*
 - b. *Chat window / inline chat / source code comments / autocomplete*
 - c. *Copilot give suggestions*
 - d. *Decide whether suggestion is good*
 - e. *Repeat*
- × **Source code still can be used even if not using Copilot (see Resources & References)**



ChatGPT?

- ✗ Github Copilot is more suitable
- ✗ Can still use ChatGPT (using correct prompts)
- ✗ Demonstration videos on some lessons
- ✗ Copilot integrated directly to Visual Studio Code
- ✗ Copilot generally generates more accurate code
- ✗ Can use ChatGPT without Github Copilot
 - ✗ Jump back and forth between Visual Studio Code & web browser



Working With AI Assistant

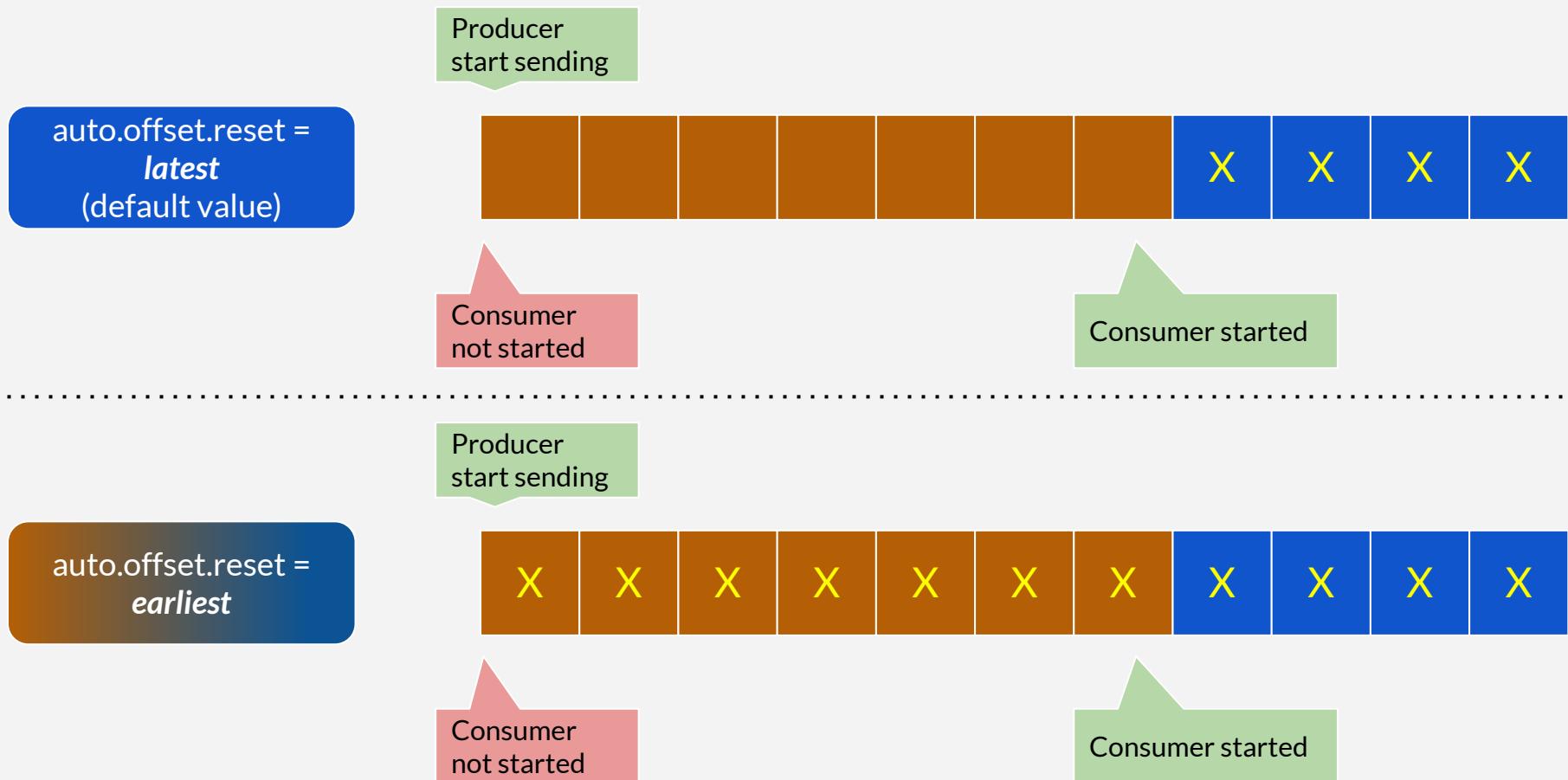
- × *Different output between AI assistant and course video is expected*
- × *Non-deterministic nature: different responses for the same task*
- × *Code suggested by AI assistant needs to be validated and (maybe) adjusted*
 - × *Does not match the requirement*
 - × *Library version is not the newest*



"Fixing" Consumer



Consumer Offset on First Run



Producing Message With Key



Message with Key

- × Same key goes to same partition
- × As long as we dont change the partition number



Multiple Consumers for Each Topic

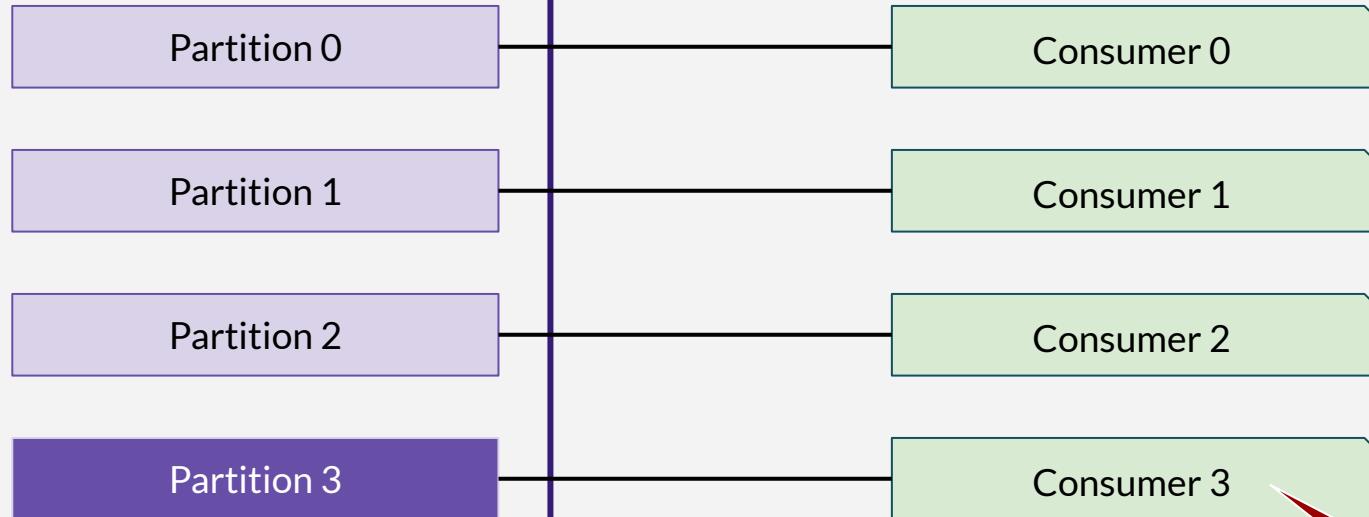


Do We Need It?

- ✗ Publisher (producer) works faster than consumer
- ✗ Subscriber (consumer) bottleneck
- ✗ Solution? Use multiple consumers
- ✗ Multiple consumers depends on partition number
- ✗ See previous lessons about topic, partition, and consumers



t-multi-partitions



Idle

What About Deleting Partition?

- ✗ Delete topic is OK (all partitions deleted)
- ✗ Can't delete partition
- ✗ Can cause data loss
- ✗ Wrong key distribution
- ✗ Decrease partition > delete & recreate topic

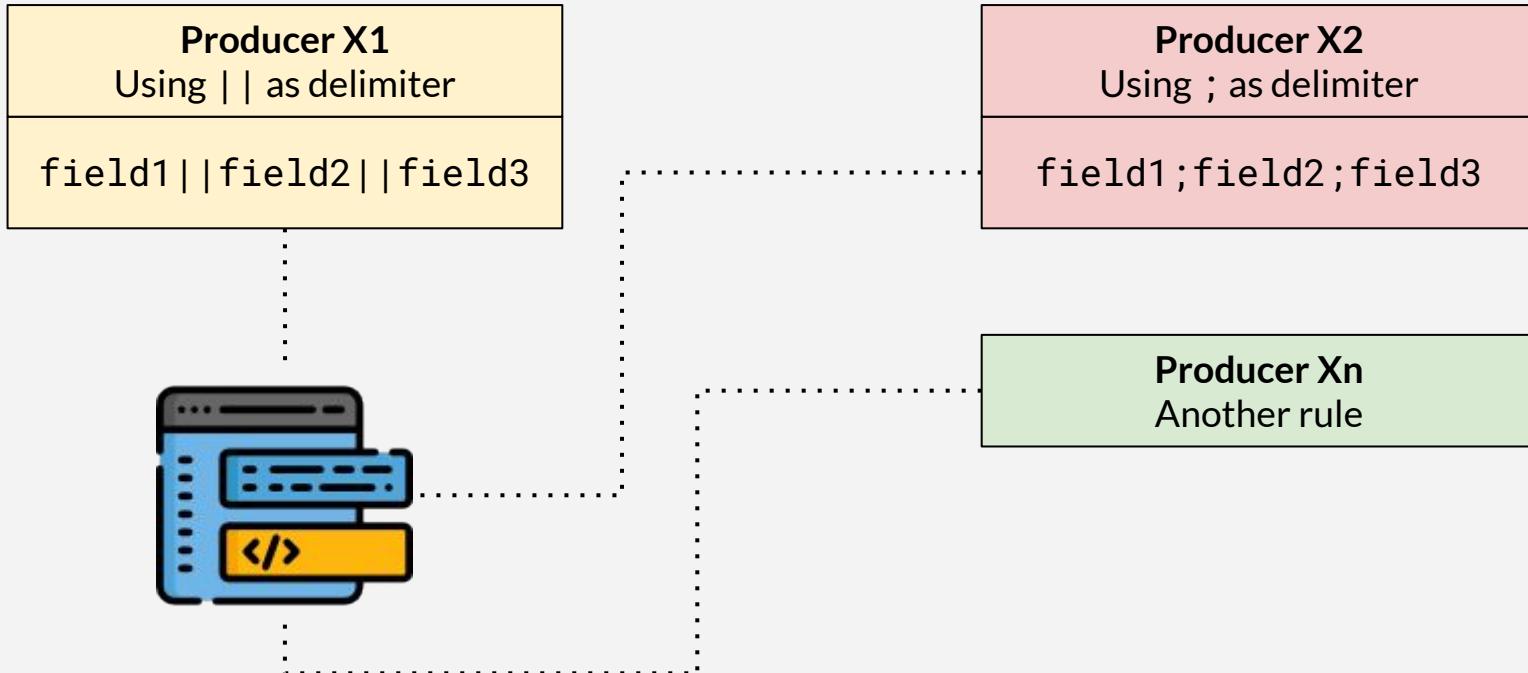


Why JSON?



String Format

Kafka can accept any string as data
What about proprietary (custom) format?



JSON

- × Use JSON as standard format
- × Java etc has library to create & parse JSON
- × Application focus on business logic instead of parsing data
- × Java uses Jackson or GSON library
- × This course will use Jackson



Producing JSON Message



Customize JSON Format



JSON Customization

- × Default :
 - × someAttribute
 - × Local date : verbose format
- × Customize by Jackson annotations
- × Different teams, different standards (some using snake_case, some using kebab-case)
- × Avoid miscommunication
- × Useful in data exchanges



Consuming JSON Message



Consuming with Consumer Groups

Create Producer

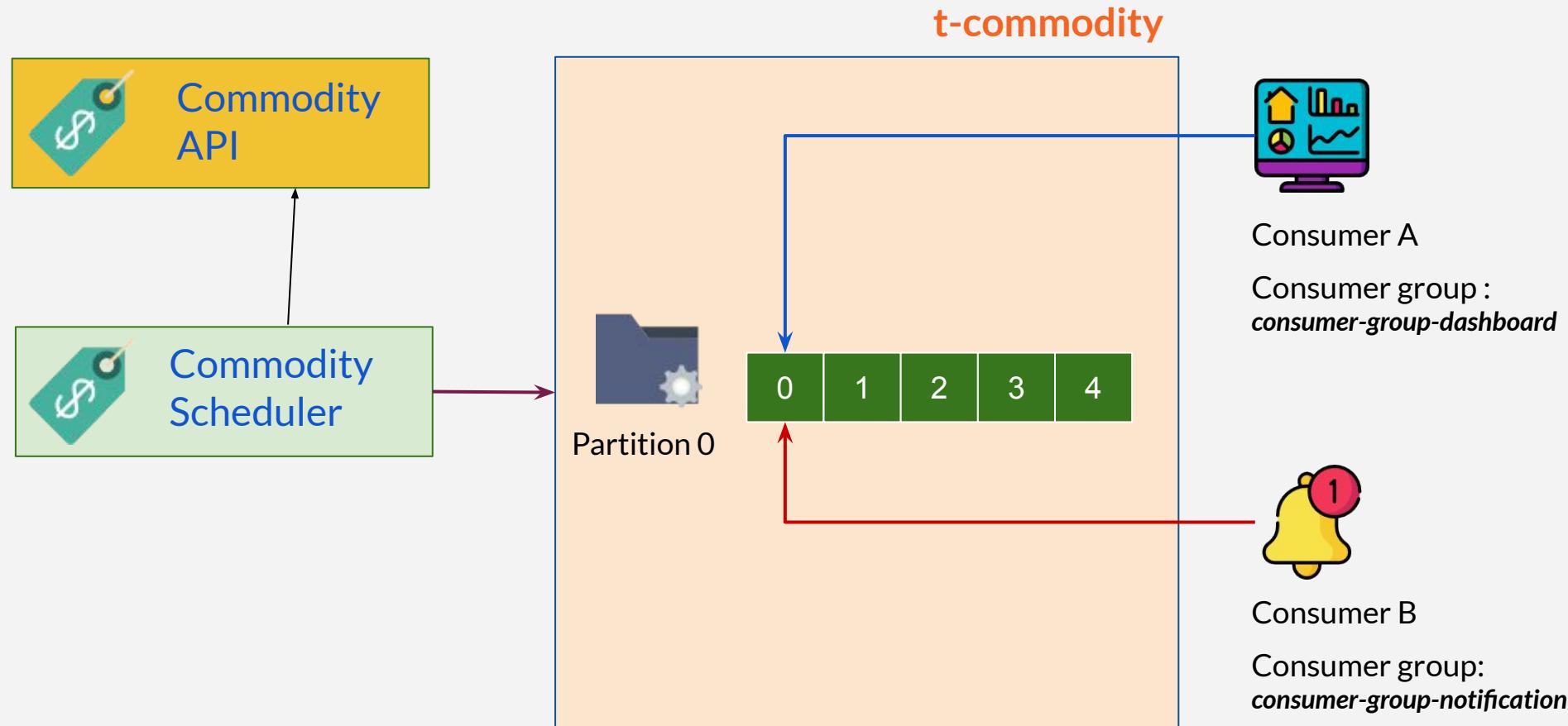


What We Will Create

- × Consumer with consumer group
- × Producer : publish commodity data
- × Functionalities
 - × 1: Update dashboard
 - × 2: Send notification
- × Read same message, different functionality



Kafka Schema



Commodity.java

name : String Example : oil, gold, coffee

price : double Example : 185.89

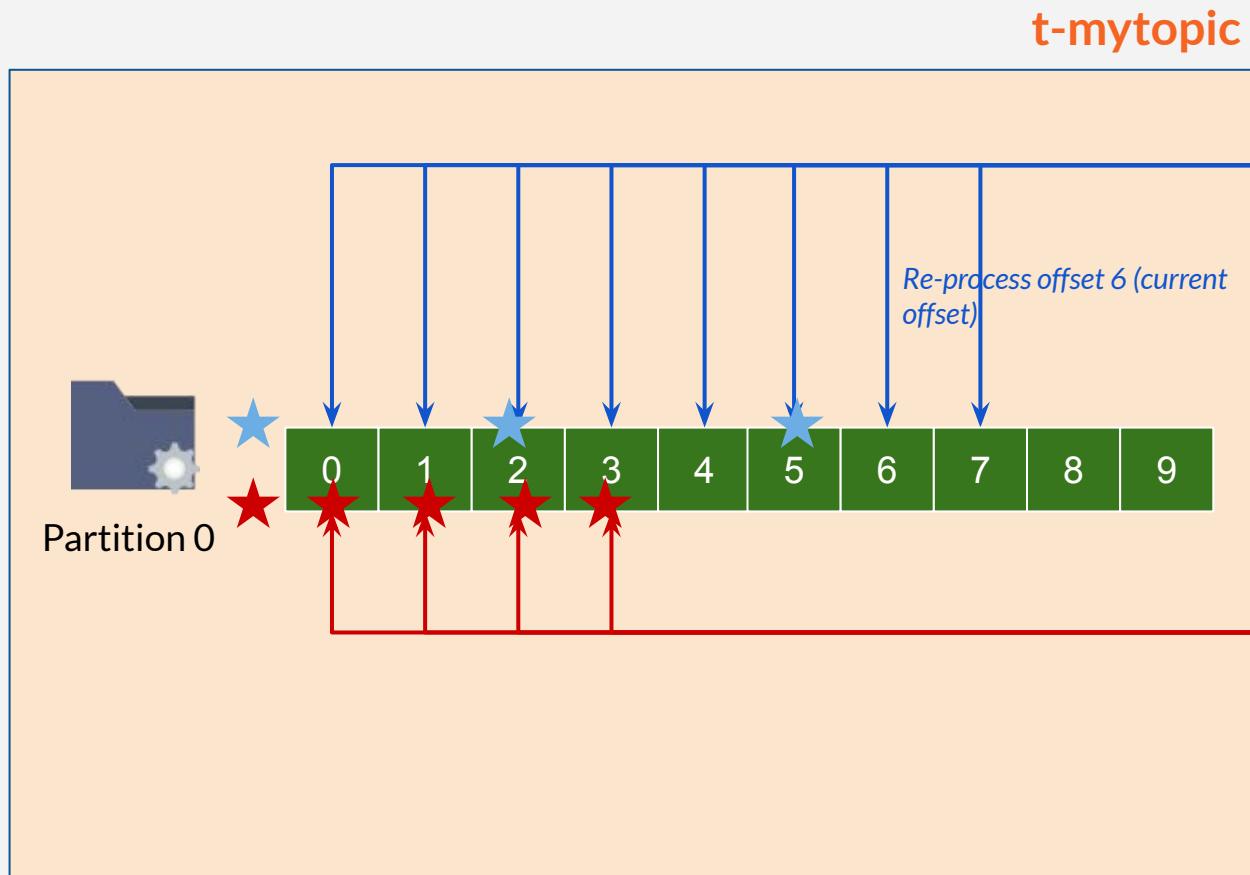
measurement : String Example : barrel, ounce, tonne

timestamp : long UNIX timestamp

Handling Consumer Offset



Consumer Group & Offset



Crash & restarted when
processing offset 7



Consumer X

Consumer group :
consumer-group-alpha



Consumer Y

Consumer group:
consumer-group-beta

Retention Period

- ✗ Consumer expected to read sequentially
- ✗ Possible bug that causes consumer down
- ✗ Kafka default retention period is 7 days
 - ✗ Offset is invalid since data no longer exists after 7 days
- ✗ Set retention period using broker config:
offset.retention.minutes



Auto Offset Reset

- × New consumer group ID behavior is determined from Spring Kafka
spring.kafka.consumer.auto-offset-reset
- × **earliest**
- × **latest** (default value)
- × **none**

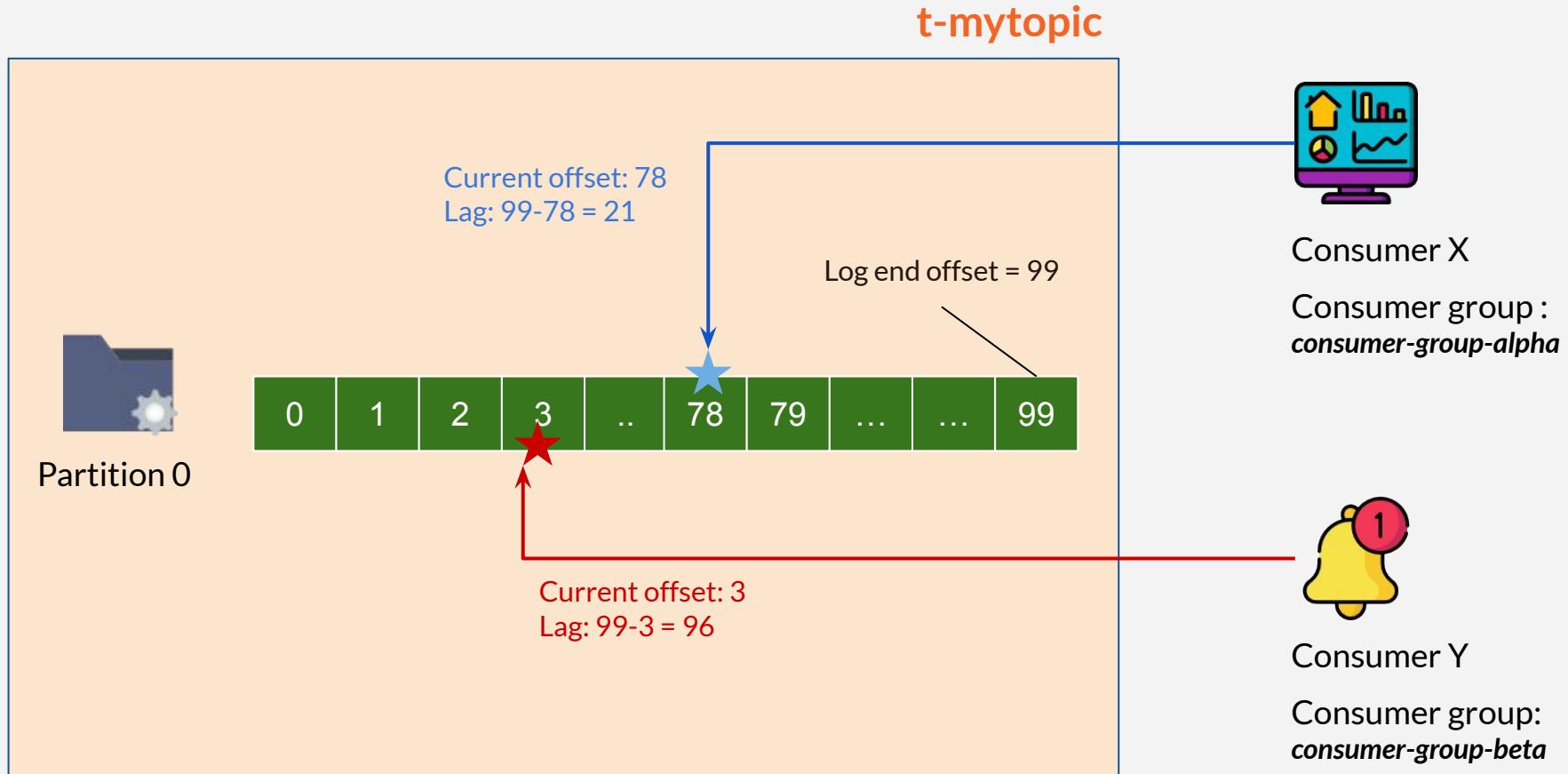


Lag & Manual Offset Reset

- ✗ Difference between latest message vs message consumer group has processed
- ✗ How to calculate
- ✗ Monitoring & identify potential bottlenecks
- ✗ Bigger lag meaning



Consumer Lag



Replay data

- × Replay data: reprocessing previously consumed messages
- × Use cases:
 - × Correcting errors
 - × Data recovery
 - × Historical analysis
 - × Testing
- × Reset consumer offset to earlier position
- × Consume from the resetted offset onwards



Committing Offsets

- × Set on application.yml, key
`spring.kafka.enable.auto.commit`
 - × `true`: auto commit according to broker configuration
 - × `false`: set mode manually
- × Default value for Spring Boot 3 is false



Committing Offset

- × Set on application.yml, key
spring.kafka.listener.ack-mode
 - × record
 - × batch
 - × time
 - × count
 - × count_time
 - × manual
 - × manual_immediate
- × See Spring Kafka official documentation



Configuring Kafka From Spring



Configuring Kafka From Spring

- × Kafka provides many customizable configurations
- × For usage, see official Kafka documentation
- × Link in last section of the course (**Resources & References**)
- × Configuration usage out of the course scope
- × Learn how to set the value from Spring



Topic Replication



Kafka Cluster

- × Kafka for data communication backbone
- × For learning / development : single Kafka broker
- × Kafka cluster : multiple brokers work as unity
- × At least 3 brokers for production



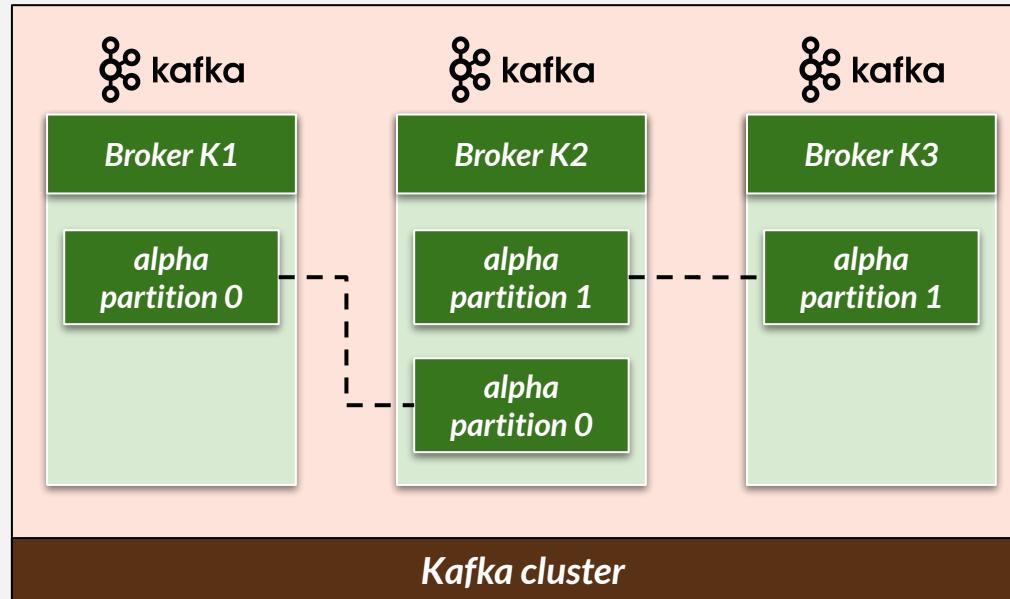
Replication

- × Copy data from one broker to another
- × Data redundancy benefit
- × Increase replication factor (2, 3, ...)
- × Data will be copied as many as replication factor
- × If a broker is down, other broker will still have the copy of data

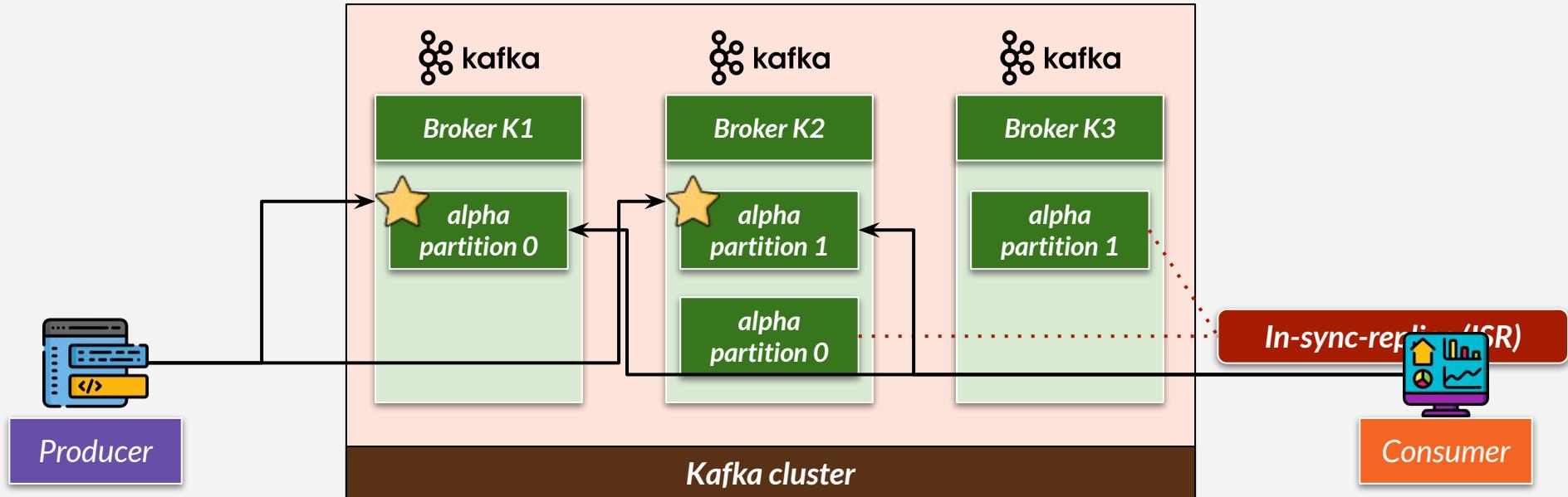


Replication

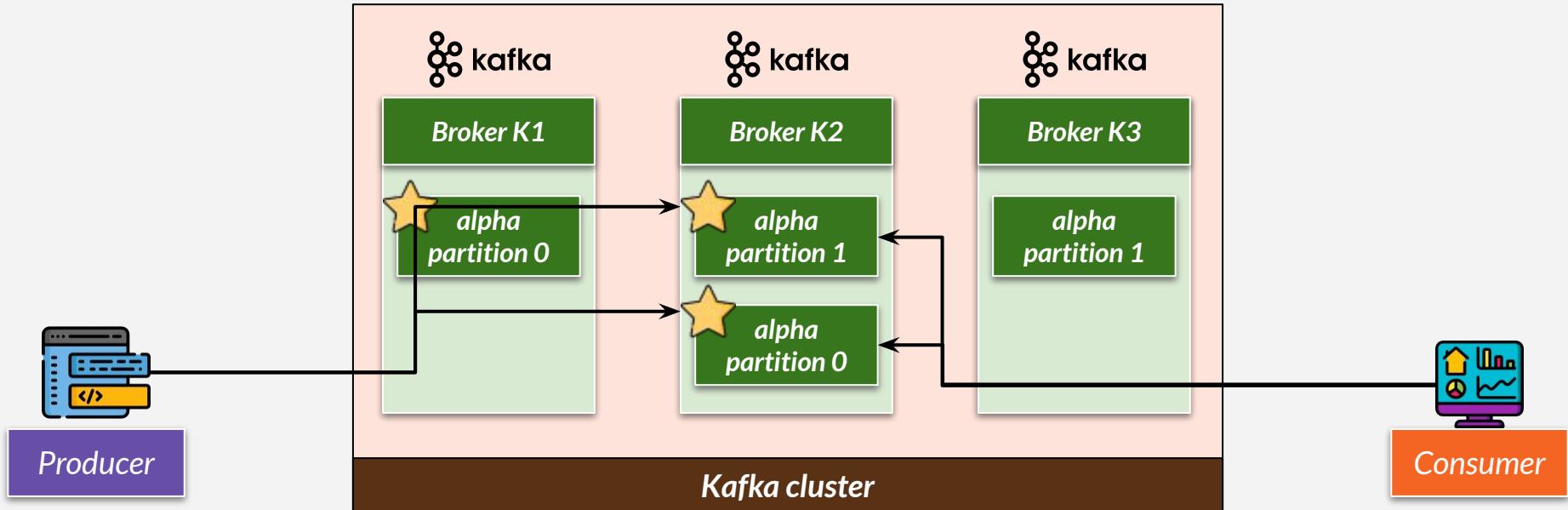
Replication factor = 2



Replication



Replication

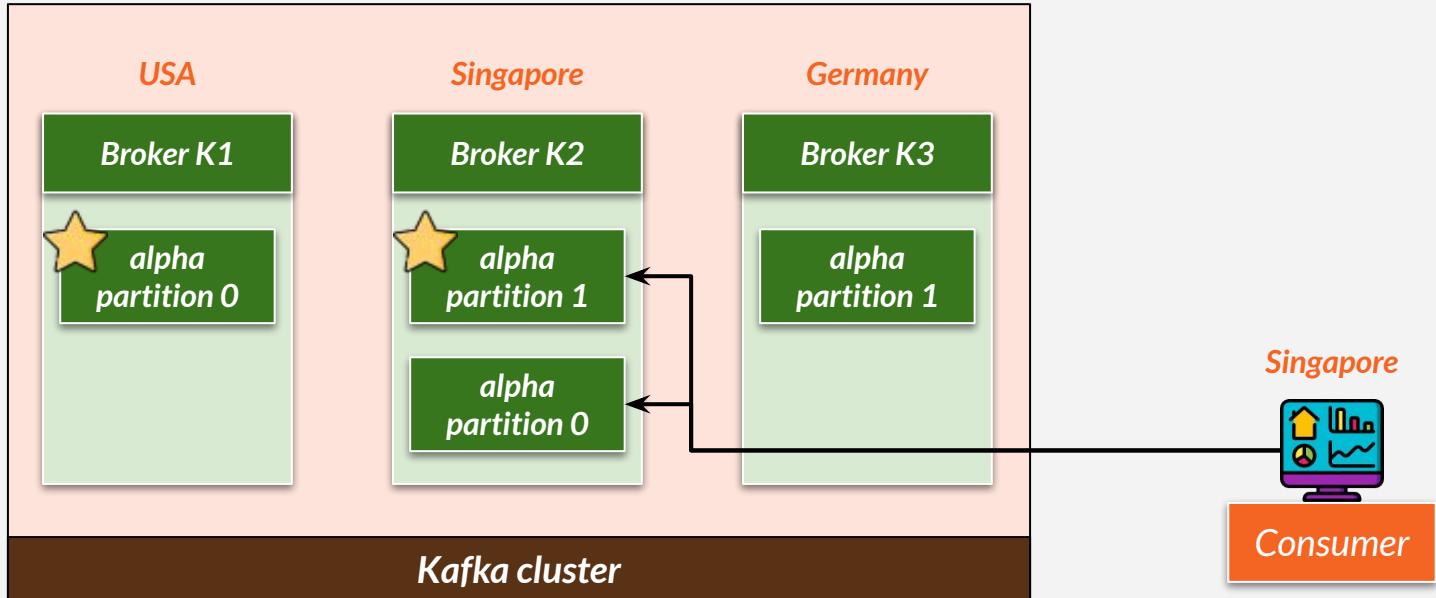


Set Replication Factor On Topic

```
#> kafka-topics.sh --bootstrap-server xxx --create --topic t-alpha --partitions 4 --replication-factor 2
```

```
#> kafka-topics.sh --bootstrap-server xxx --alter --topic t-beta --replication-factor 3
```

Consumer Replica Fetching



Consumer Replicate Fetching

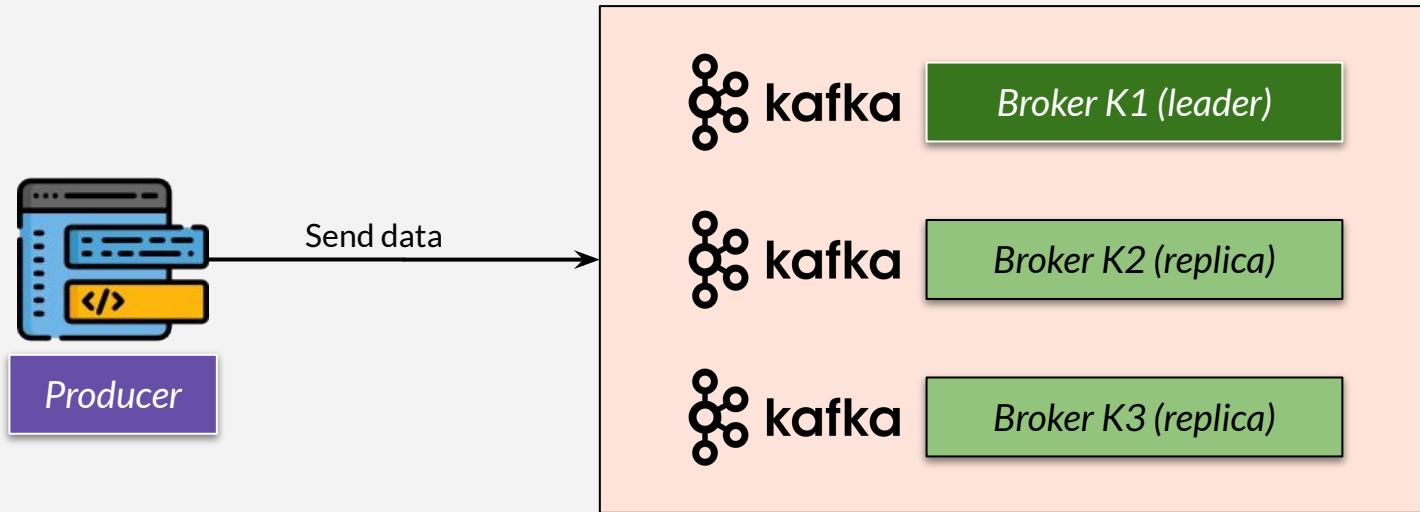
- × Kafka 2.4 or newer
- × Broker setting
 - × `rack.id = XYZ`
 - × `replica.selector.class`
- × Consumer setting
 - × `client.rack = XYZ`



Producer Acknowledgment



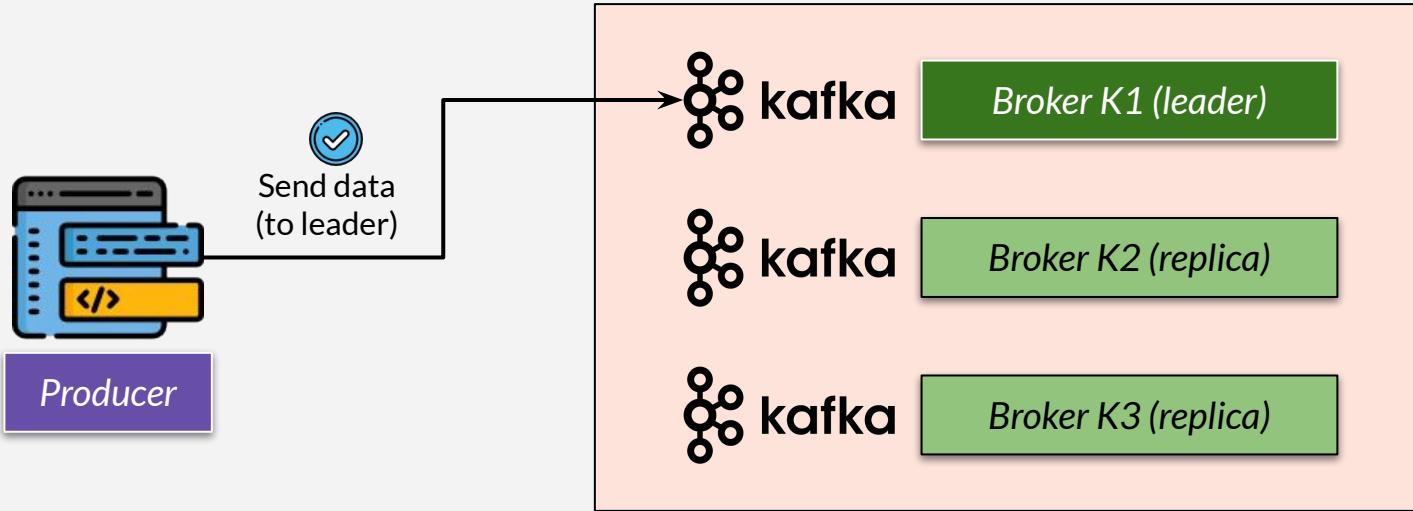
Producer Acknowledgment (Ack)



Producer can choose to receive **acknowledgement (ack)** of data writes:

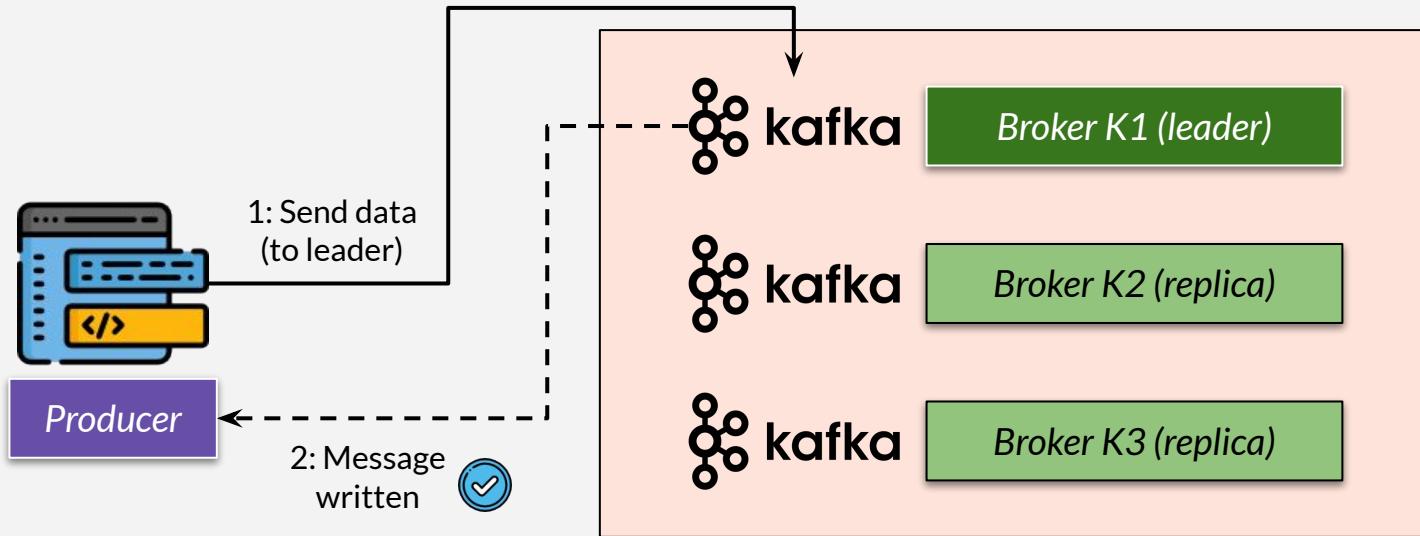
- **ack 0** : producer will not wait for acknowledgement (possible data loss)
- **ack 1** : producer will wait for leader acknowledgement (limited data loss)
- **ack -1 / all** : producer will wait for (leader + replicas) acknowledgement (no data loss)

Ack = 0



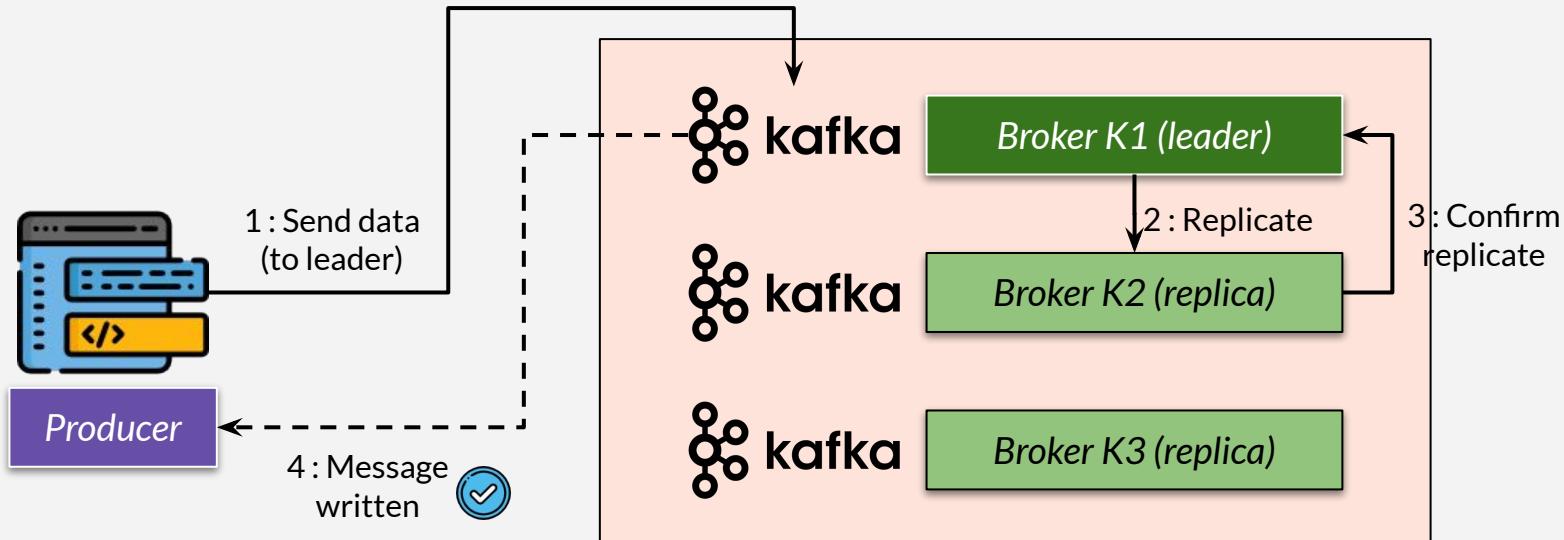
- Producer considers message successfully written when the message was sent
- If broker goes offline (or some exception happens), producer don't know and **will lose data**
- Highest throughput
- Use only on specific use cases where it's OK to lose data

Ack = 1



- Leader has written the message, while replication runs in background
- If leader encounter error during replication, producer experience **limited data loss** (message written on leader, but possibly not written on replicas)
- Default value for Kafka 1.0 to 2.8

Ack = all / -1 (sample with replication factor 2)



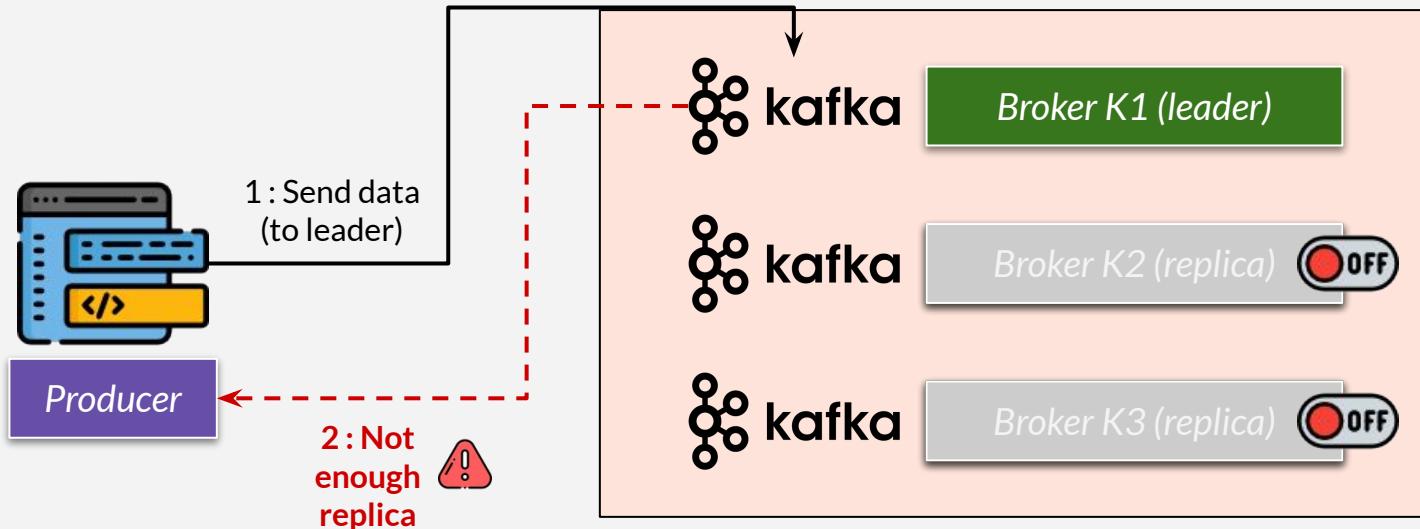
`min.insync.replicas`

- × Use acks = all in conjunction with `min.insync.replicas`
 - × In broker or per topic
- × Is there enough in-sync replica to write safely?
- × Example



Topic Durability

`min.insync.replicas = 2`
Replication factor = 3



Topic Durability

- × Recommendation: at least 3 replication factor and
2 min. insync. replicas
- × At least one replica get the data
- × Reject rather than risk losing data
- × Producer wait for replicas to be available



Set Producer Ack

application.yml

```
spring:  
  kafka:  
    producer:  
      acks: 0 / 1 / -1 / all
```

Set min.insync.replicas On Topic

```
#> kafka-topics.sh --create --bootstrap-server xxx:9092 --topic t-beta --replication-factor 3  
--partitions 3 --config min.insync.replicas=2
```

```
#> kafka-configs.sh --bootstrap-server localhost:9092 --alter --entity-type topics --entity-name t-beta  
--add-config min.insync.replicas=3
```

Rebalancing

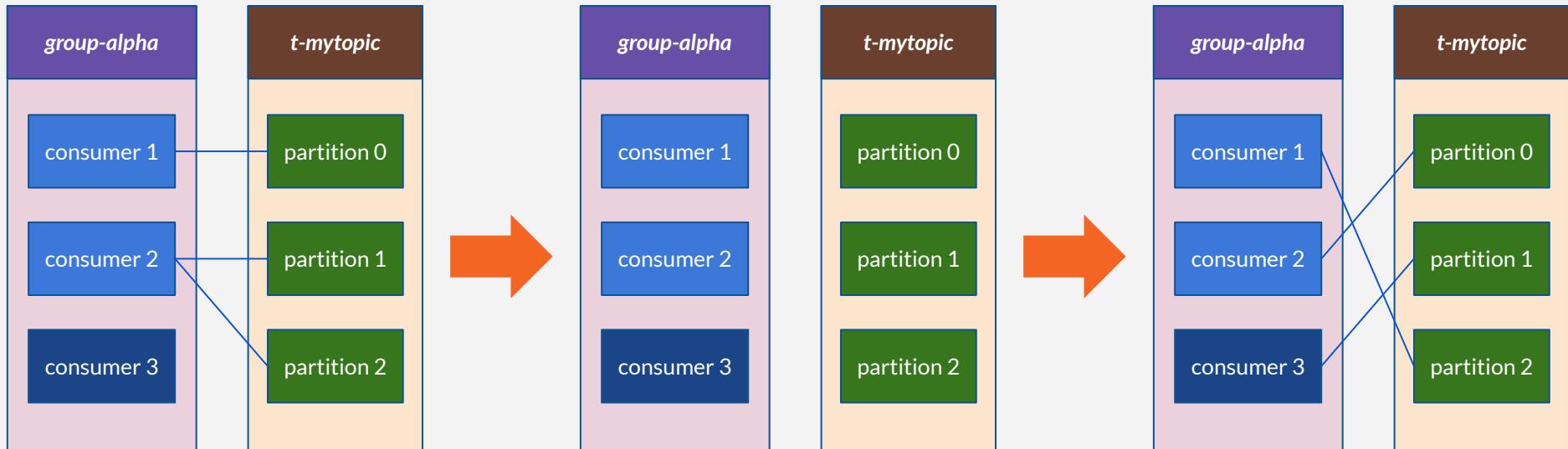


Rebalancing

- × Rebalancing: redistributing partitions across consumers
- × Triggers:
 - × Consumer joining or leaving particular consumer group
 - × Partition count on topic changes
- × Impact:
 - × Temporary unavailability
 - × Resource overhead



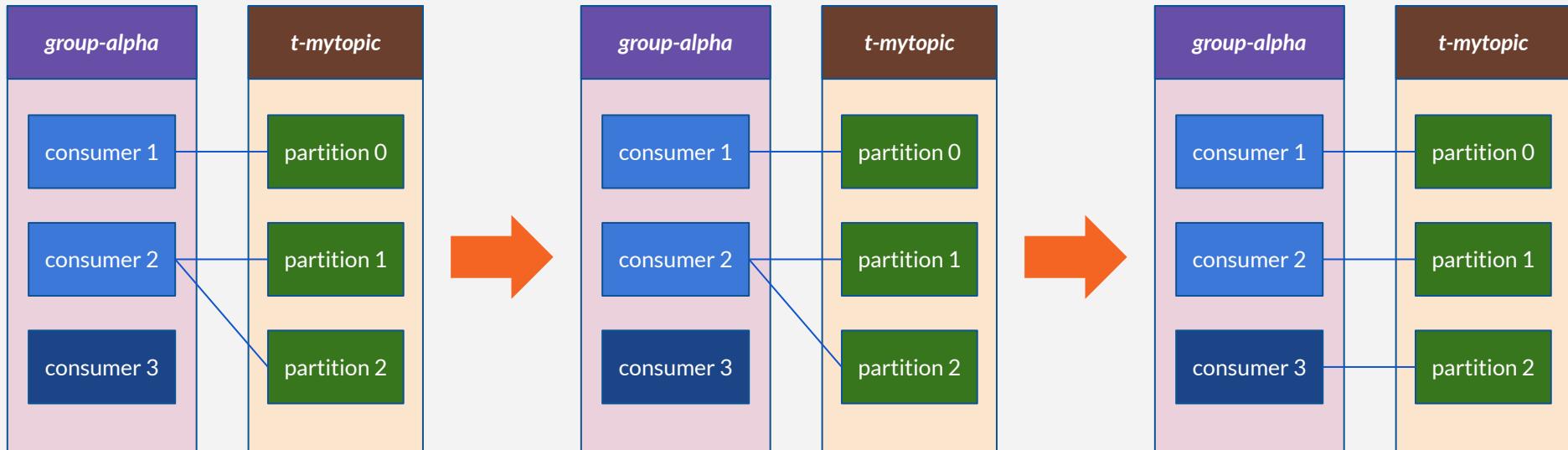
Eager Rebalance



All assignments revoked
(no consume process during
this period)

Consumers rebalanced,
partitions reassigned

Cooperative (Incremental) Rebalance



*Revoke partition two.
Consumers still consuming*

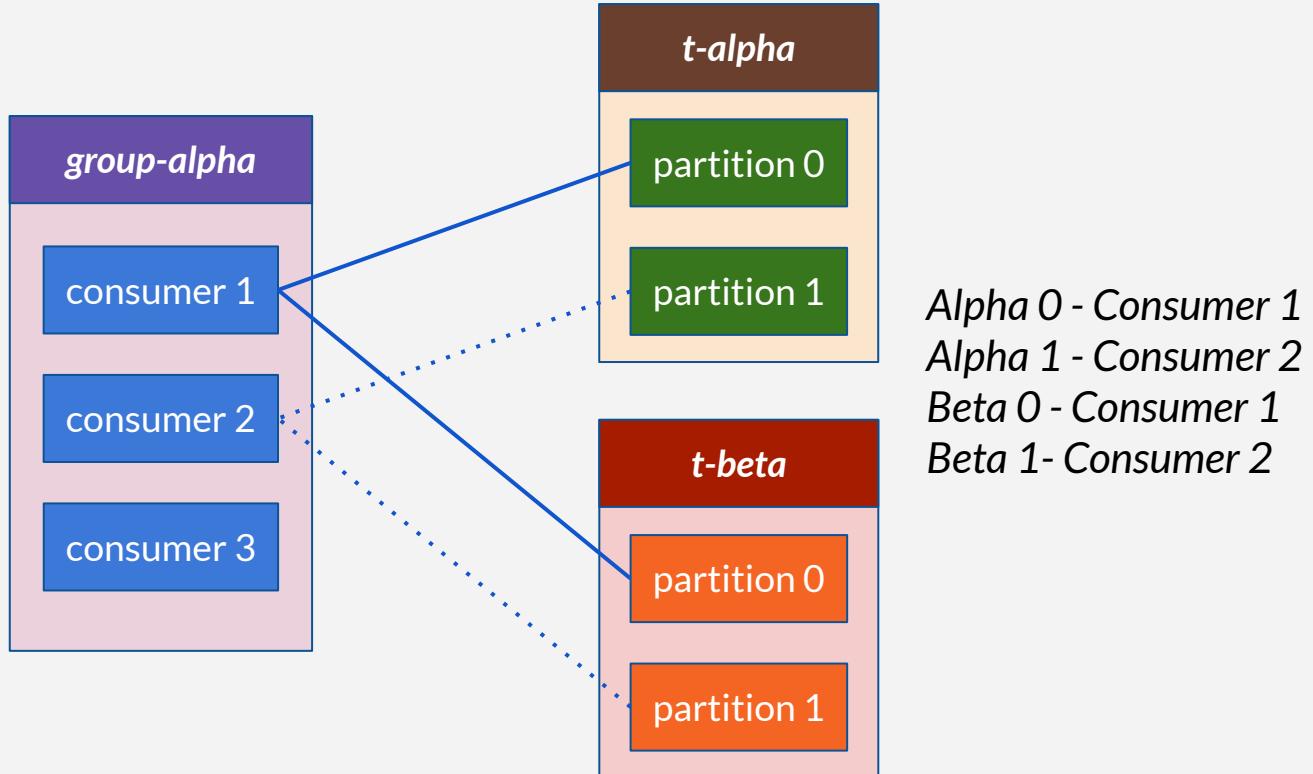
Consumers rebalanced

Illustration

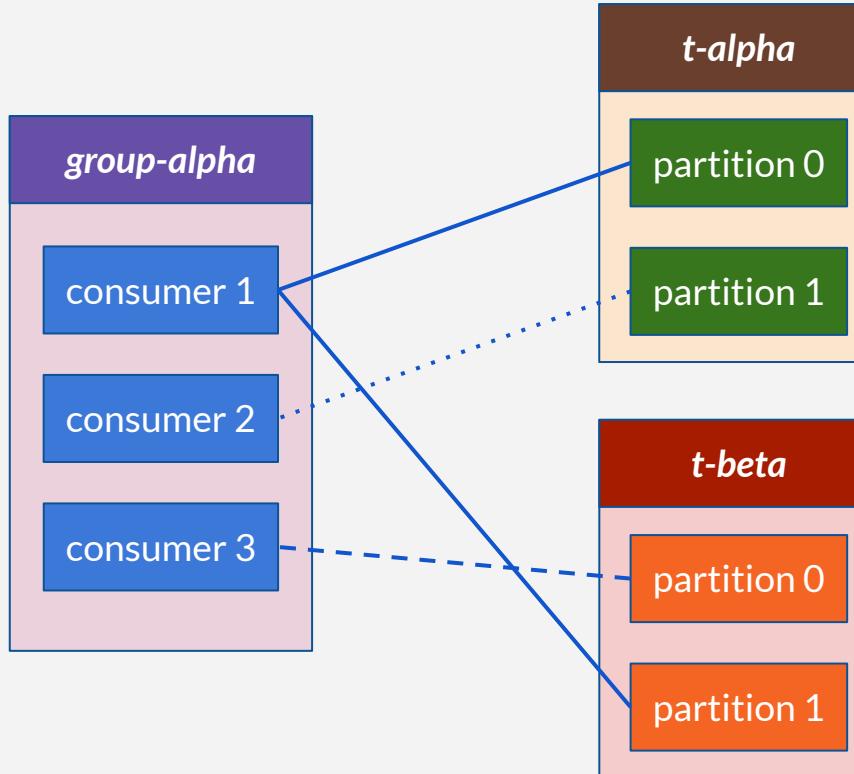
- × Set partition assignment strategy
- × Next examples:
 - × 2 topics
 - × 2 partitions each topic
 - × 3 consumers, each listen to the 2 topics



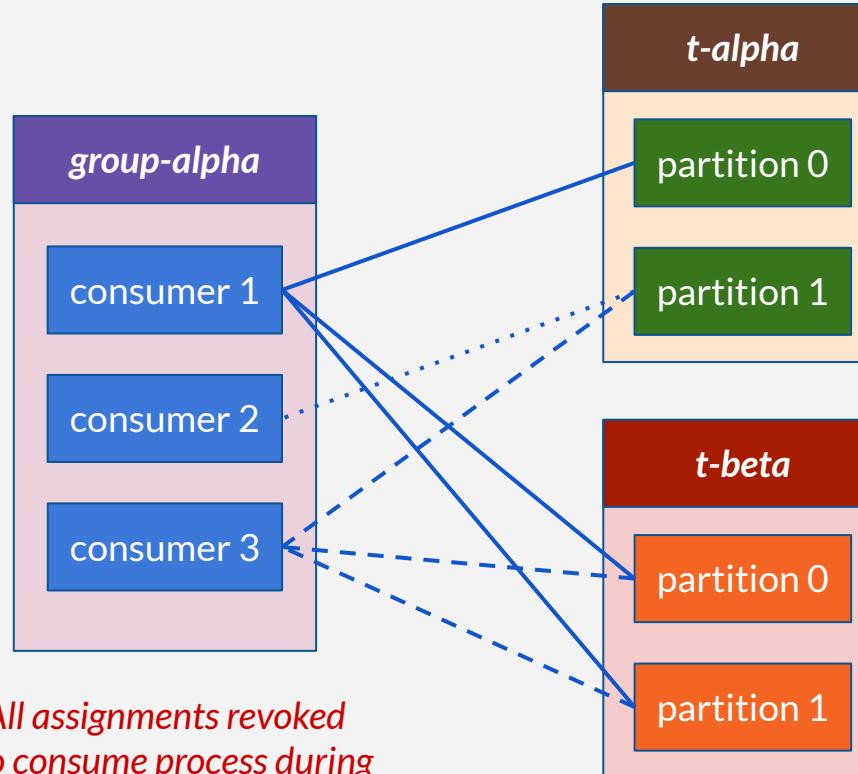
Range Assignor



Round Robin Assignor

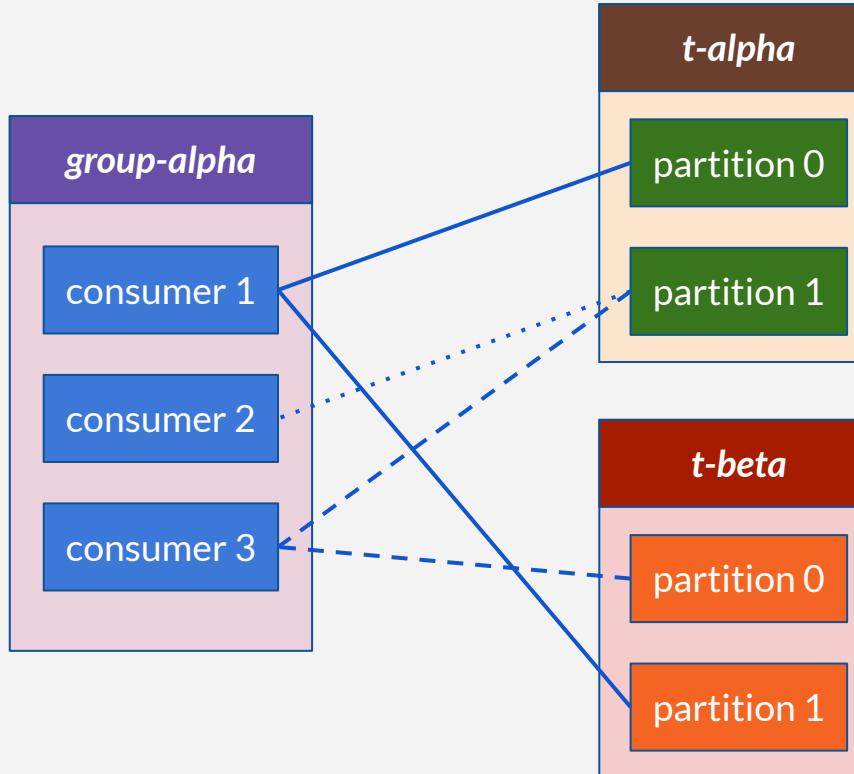


Round Robin Assignor (Remove Consumer)



Alpha 0 - Consumer 1
Alpha 1 - Consumer 3
Beta 0 - Consumer 1
Beta 1 - Consumer 3

Sticky Assignor



Alpha 0 - Consumer 1

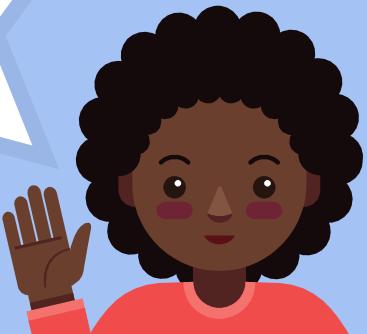
Alpha 1 - Consumer 3

Beta 0 - Consumer 3

Beta 1 - Consumer 1

Rebalance

- × Eager: range, round-robin, sticky
 - × A period where consumers stop
 - × More consumers means more time to rebalancing
- × Cooperative: cooperative-sticky
 - × Like sticky assignor
 - × Consumers can keep consuming during rebalance
- × Default strategy in Kafka 3: [**range, cooperative-sticky**]



Static Group Membership

- ✗ Influence rebalance behavior
- ✗ By default, consumer receive new member ID (consumer ID) when leave & rejoin the group
- ✗ Specify group instance ID
- ✗ Example static ID scenario
- ✗ Useful when stability of consumer group memberships is important
- ✗ Facilitate local cache
- ✗ When to use static group membership?



Kafka Metadata Refresh

- × Delay time before rebalancing occurs
- × Changes in Kafka metadata are cached
- × Configurable on parameter ***metadata.max.age.ms***
 - × Default is 300000 (5 minutes)
- × Adjusting parameter value can be useful
 - × Shorter value: faster cache update, but has resource overhead
 - × Longer value: slower cache update, but reduce resource overhead



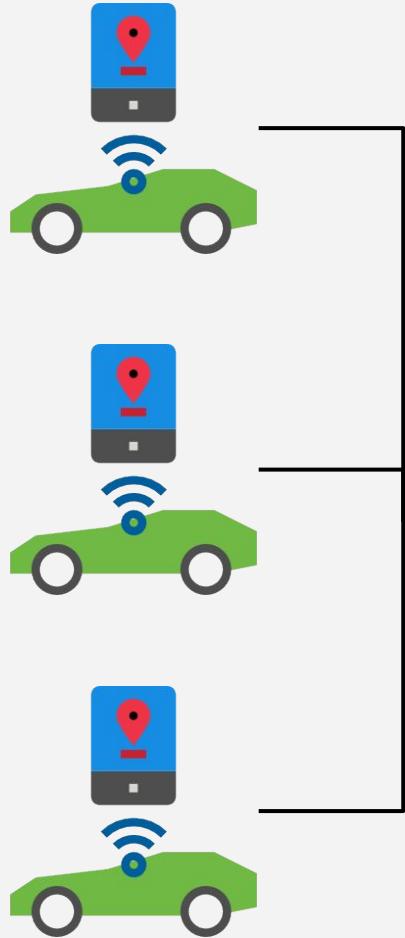
Message Filter



Filtering Message

- × Match criteria: processed
- × Not match criteria: not processed but still on topic
(not deleted)
- × Filter for each listener





{
 "carId" : "CX1580",
 "timestamp" : 1577750400,
 "distance" : 74
}

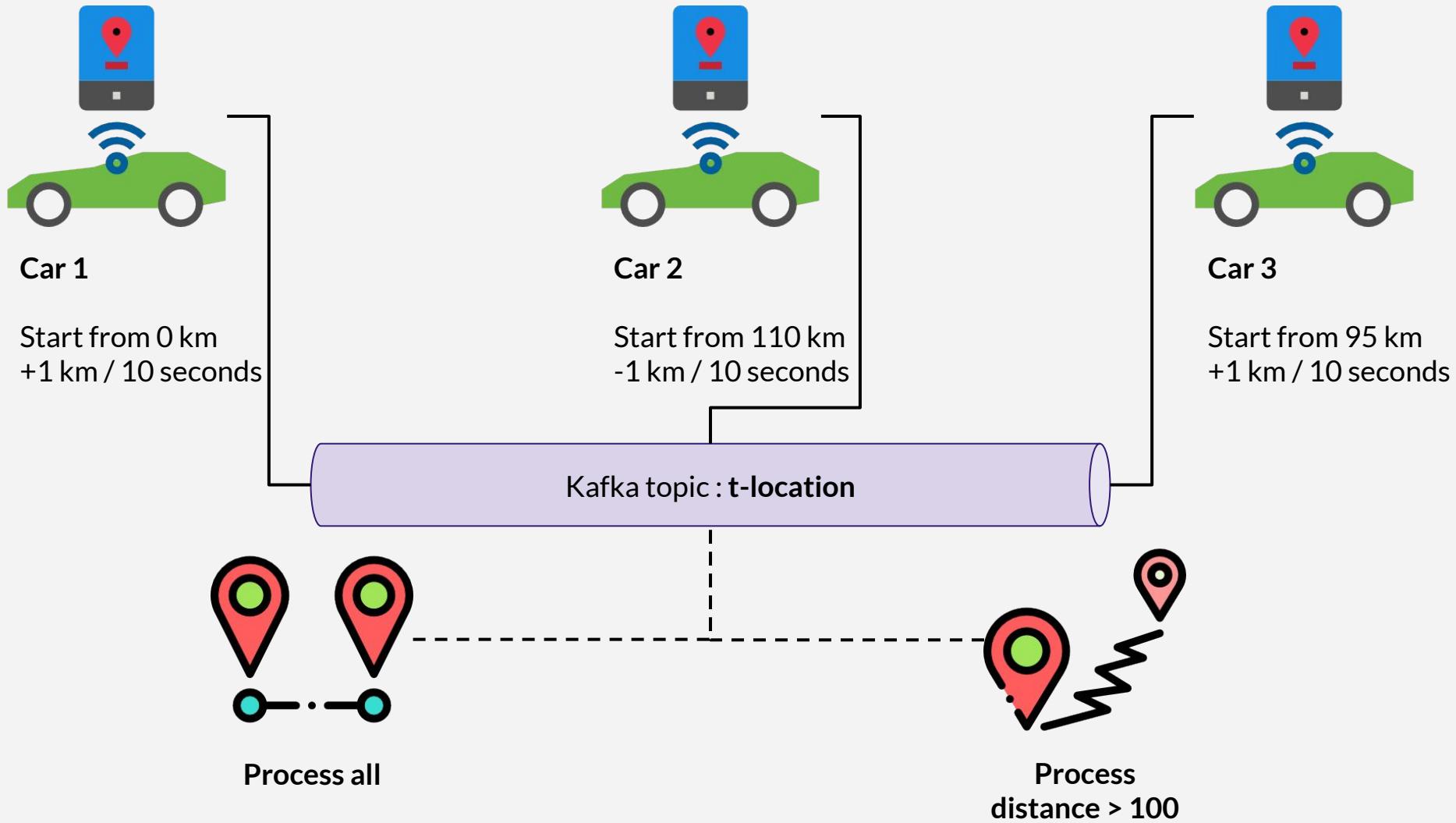
t-location



Process all



Process
distance > 100



Idempotency

Handle Duplicate Message



At-Least-Once Delivery Semantic

- × Message guaranteed to be published
- × Message might be published more than once
- × Consumer process all messages (including duplicate ones)
- × Possibilities for duplicate message



Idempotent Consumer

- × Duplicate message is OK:
 - × Outcome of processing message always same even for duplicate messages
 - × Example : update search engine index
- × Duplicate message is dangerous:
 - × Duplicate transaction
 - × Example : create (duplicate) payment
 - × Filter out duplicate message(s)



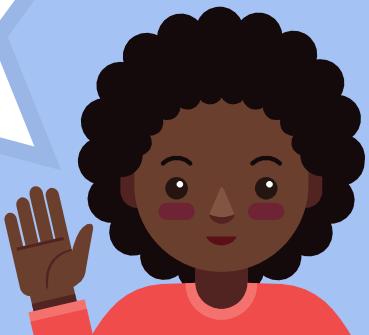
Gray Area

- ✗ It might be dangerous, or not
- ✗ Example : send promotion email to user
- ✗ Can either OK, or bad user experience
- ✗ Technically should filter out duplicate messages
- ✗ Double output is not idempotent



How to Deduplicate?

- × Unique value attached to each message
- × Consumer check unique value when receive record
 - × Never processed > store unique value, then process message
 - × Has been processed > skip message



How to Deduplicate?

- × Use database for permanent unique value
- × Use cache for temporary unique value
- × Cache
 - × Better performance
 - × Automatically remove data after certain time
 - × Example : Redis
- × Database
 - × Might publish duplicate after longer period
 - × Virtually unlimited storage
- × This lesson uses cache





id : primary key database
(surrogate)

Purchase request number :
natural key (business)

Kafka record key using ID



Partition 0

Offset 0 :
5551

Offset 1 :
5552



Partition 1

Offset 0 :
5553



5551, 5553, 5552
5553, 5551, 5552

Event	PR Event ID	PR number
Budget reserve	5551	PR-One
Approval workflow	5552	PR-One
Push notification	5553	PR-One

Kafka record key using PR Number



Partition 0

Offset 0 :
PR-One
5551

Offset 1 :
PR-One
5552

Offset 2:
PR-One
5553



Partition 1



Idempotency

Alternative Approach



No Unique Value

- × No unique value on message
- × Alternative : use object as unique value
 - × Bad idea, if the object contains large data
 - × Eat up cache memory / slow database
- × Derive key from combination of fields
 - × Combination must be unique
- × Don't use java hashCode()
 - × Not guaranteed to be unique for different object

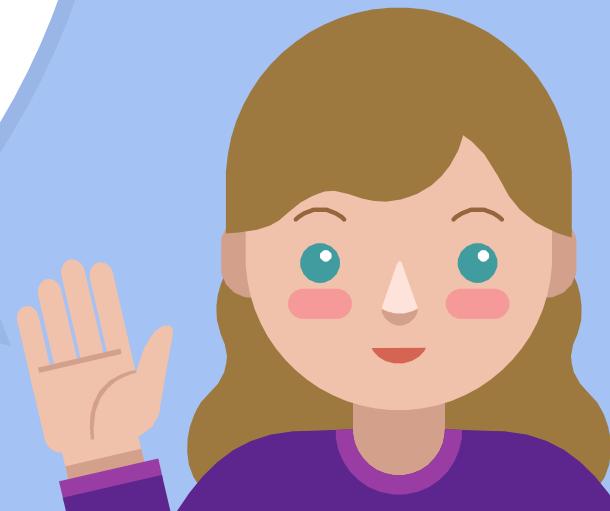


Idempotency & Message Filter



Handling Exception

@KafkaListener Error Handler



ProptionException

```
public void send(MyMessage myMessage) {  
    try {  
        // some logic  
        //  
  
        kafkaTemplate.send("topic", "messageKey", messageBody);  
    } catch (Exception e) {  
        // handle exception  
    }  
}
```

What We Will Do

- × Spring default : log exception
- × Able to implement our own error handler
- × Scenario
 - × Publish food order
 - × Exception : consume invalid food amount



Error Handler

- ✗ Implement custom logic
- ✗ Need error handler
- ✗ Use the error handler on **@KafkaListener**



Handling Exception

Common Error Handler

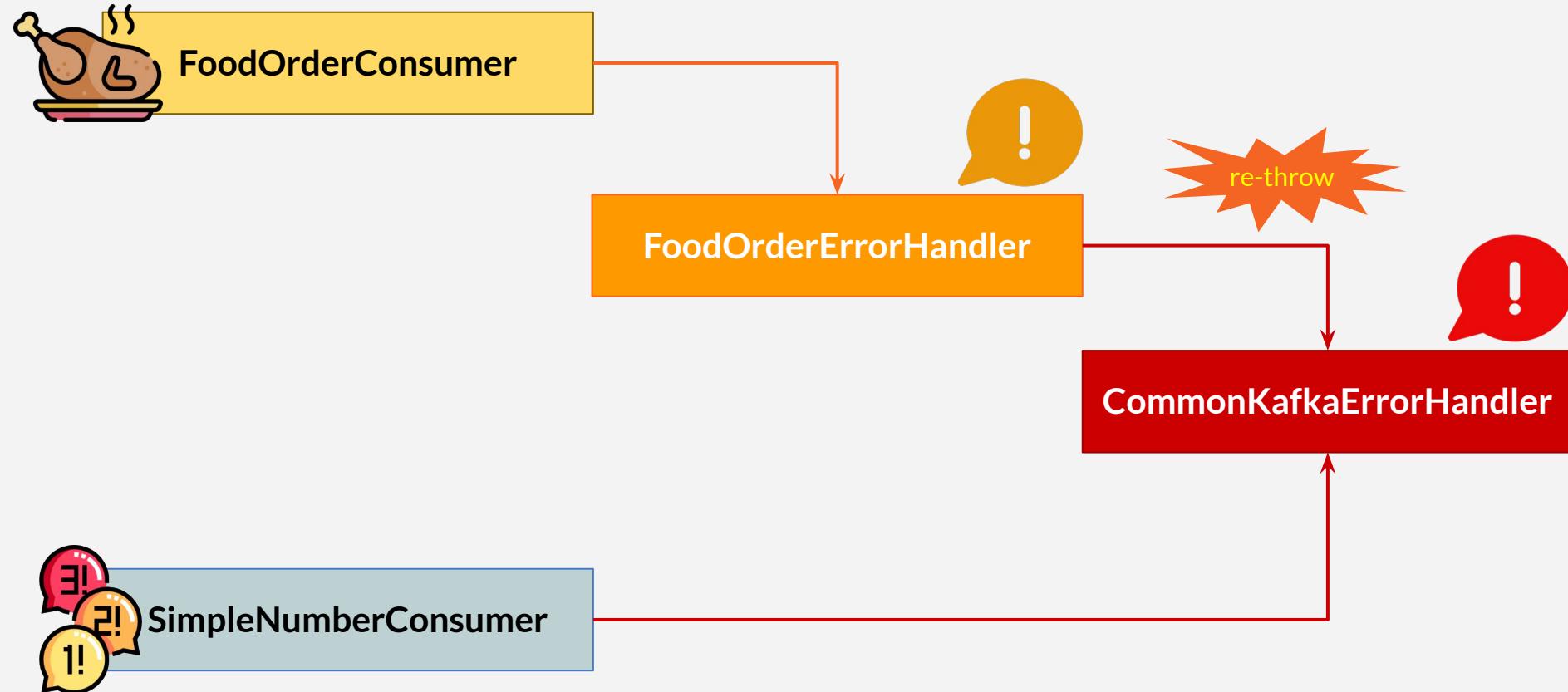


What We Will Do

- × Global error handler : works for all Kafka consumers
- × Error handler on Spring container
- × Scenario
 - × Publish random number
 - × Consuming odd number throws exception
 - × Handle using global error handler



Error Handler Flow



Consumer With Retry



Why Retry Mechanism?

- × Spring Kafka will log failed messages
- × Our own error handler
- × Case
 - × Service temporarily unavailable
 - × Retry hit service without trigger from consumer
 - × Retry for N times only



Scenario

- × Topic: t-image, 2 partitions
- × Publish message represents image
- × Consumer
 - × Simulate API call to convert image
 - × Simulate failed API call
 - × Retry after 10 seconds
 - × Retry 3 times



Blocking Retry

Message 1	Message 2	Message 3 (blocked)
-----------	-----------	---------------------



Encounter error



Retry



or



Blocking Retry

- × Good when message must be processed in sequence
- × Drawback : process halted (might bottleneck)
- × Mitigation : retry "just enough"
- × Block only on partition which has error
- × Other partition (no error) keep consuming



Handling Exception Dead Letter Topic

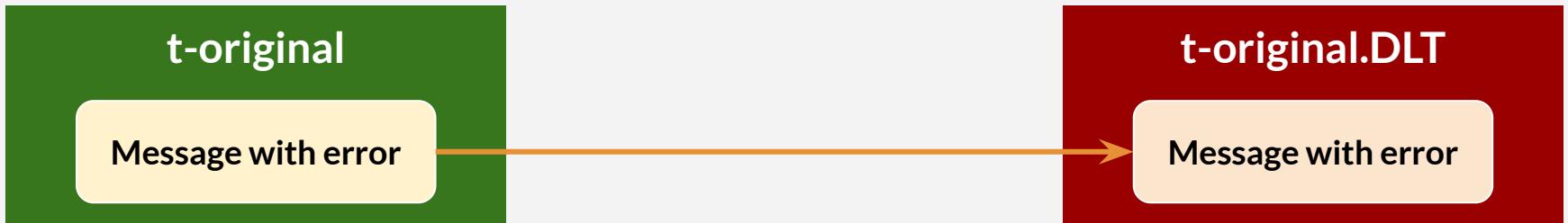


Dead Letter Topic

- × Message process keep failing
 - × Non technical issue
 - × Permanent technical issue
 - × Process message differently
- × Send such message to dead letter topic
- × Dead letter record



DeadLetterPublishingRecoverer



In this lesson:

- Send to custom topic
- Sendt to custom partition

Scenario

- × Publish to t-invoice
- × If amount is less than one, throw exception
- × Retry 5 times
- × After 5 failed retry attempts, publish to t-invoice-dead
- × Another consumer will consume from t-invoice-dead

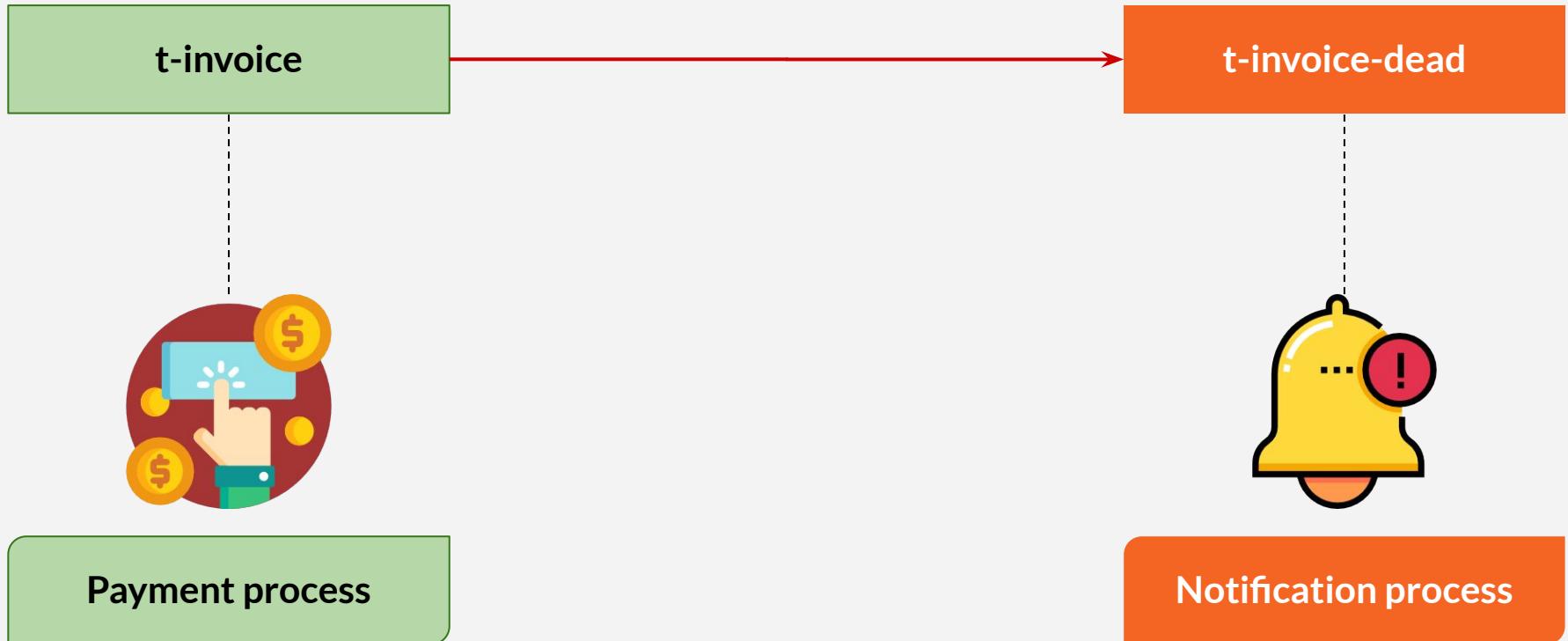


Dead Letter Records

- × Dead letter record : messages sent to DLT
- × Just regular topic / message
- × Create consumer to listen from DLT



Process Dead Letter Record



Handling Exception

Non Blocking Retry



Non Blocking Retry

- × Message can be processed independently
- × Message retry still needed
- × Spring non blocking retry



Non Blocking Retry

Run while retrying message 2 (not blocked)

Message 1	Message 2	Message 3	Message 4	Message 5
-----------	-----------	-----------	-----------	-----------



Encounter error



Retry

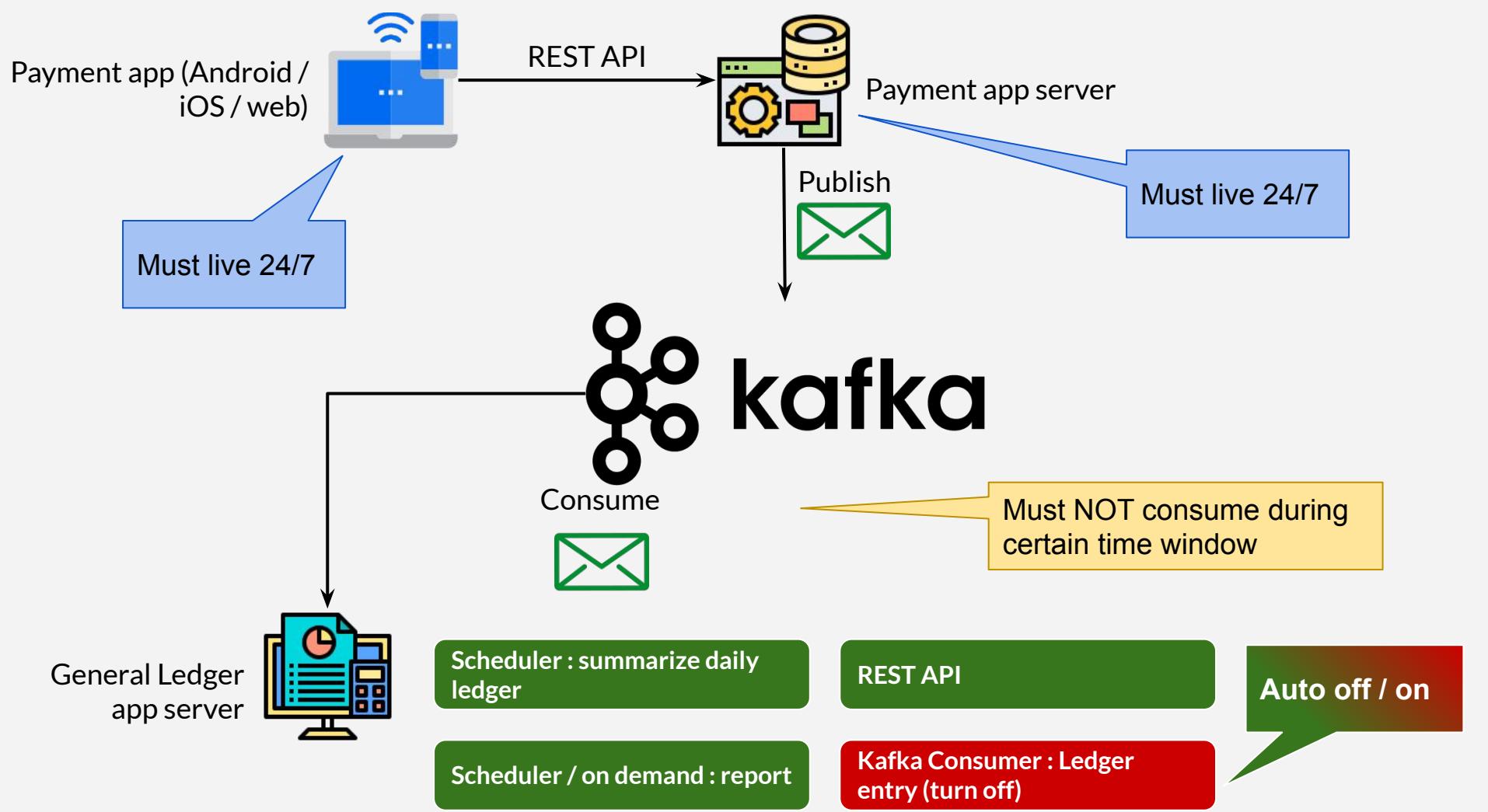


or



Scheduling Consumer





Kafka vs RabbitMQ



Popular Messaging System

- × Apache Kafka
- × Rabbitmq



Message Retention

- ✗ Kafka : by policy
- ✗ Rabbitmq : by acknowledge



Kafka: Exists on disk / memory until successful process

RabbitMQ: exists on disk / memory until successful process



Message Routing

- ✗ Kafka : no routing mechanism
- ✗ Rabbitmq : routing mechanism using exchange



RabbitMQ Queue 2

RabbitMQ Queue 1

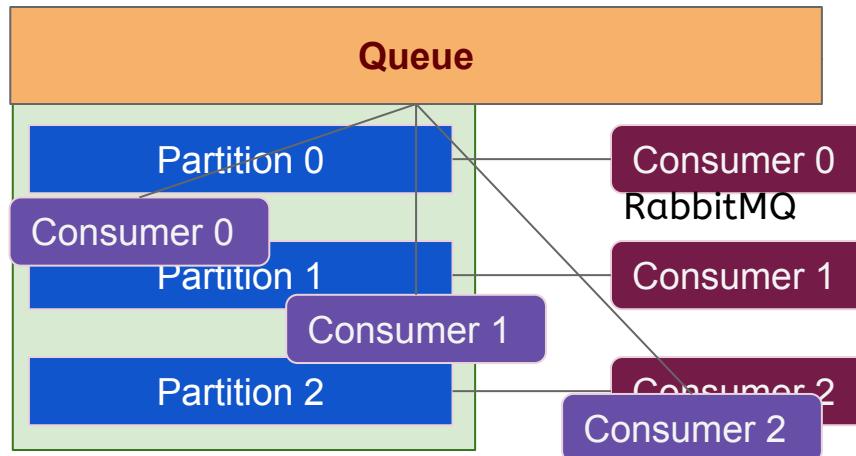
Publisher

RabbitMQ exchange

(partition)

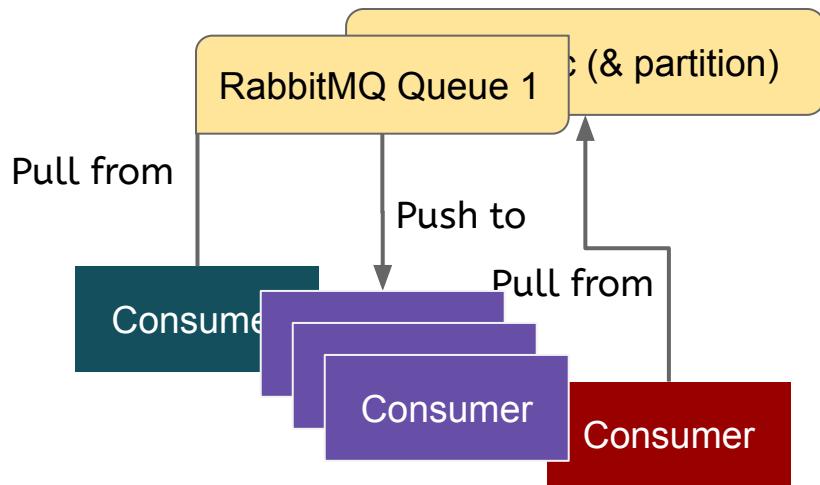
Multiple Consumers

- ✗ Kafka : topic - partition - one consumer per partition,
guaranteed order
- ✗ Rabbitmq : multiple consumer per queue, not
guaranteed order



Consumer Push / Pull Model

- × Kafka : consumer pull from Kafka topic
- × Rabbitmq : push message to consumer



RabbitMQ "Resembles" Kafka

- × Start from RabbitMQ 3.11
 - × Stream
 - × Single active consumer on stream
- × Resembles Kafka



Which One To Choose

- × It really depends
- × Some samples from my experience
- × My personal opinion:
 - × RabbitMQ is easier to manage
 - × Kafka is more scalable & good for big data
 - × Popular programming languages library is supported
 - × RabbitMQ learning curve is easier
- × My suggestion : pick only one for our organization



What We Will Build



Application Overview

- ✗ Oversimplified application
- ✗ Introducing possible kafka usage on real life
- ✗ Microservice architecture & pattern
- ✗ Microservice reference on last lesson

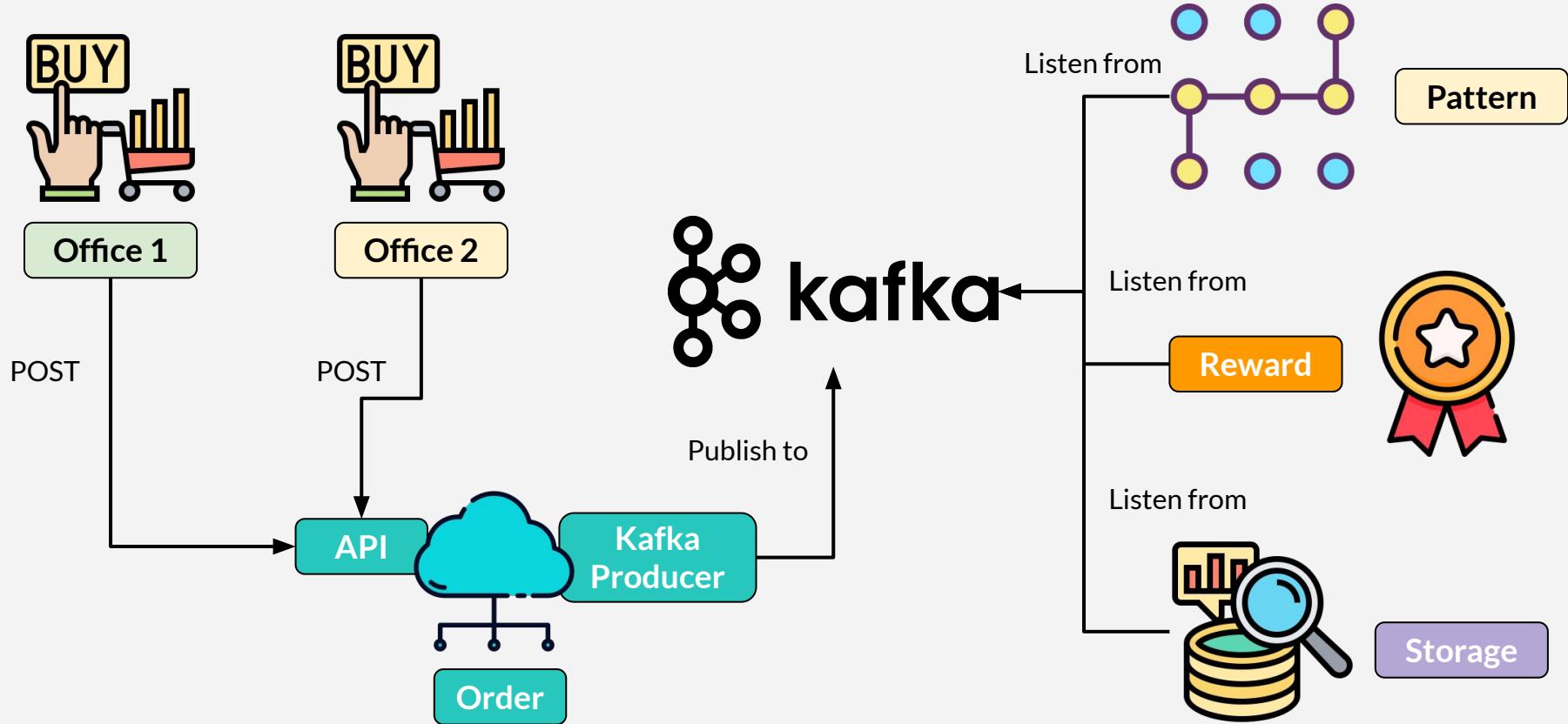


Use Case

- × Commodity trading company with multiple branches
- × Branch submit purchase order to head office
- × After process :
 - × Pattern analysis
 - × Reward
 - × Big data storage
- × Speed is the key
- × Branch office submit each order once
- × Head office process using kafka



Application Architecture



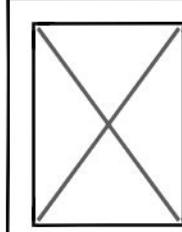


Order location

Credit Card Number

1958 2850 6094 3758

Items

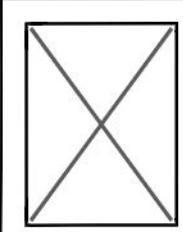


A beautiful book

\$14



Quantity

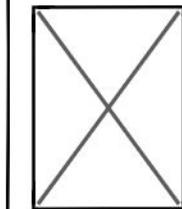


An exotic fruit

\$3



Quantity

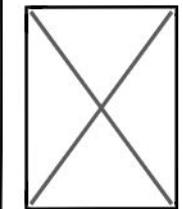


A mysterious box

\$14



Quantity



A luxury dress

\$26



Quantity

App In The Course

- × Use postman to simulate API call
- × In memory H2 database
 - × Data gone every restart
 - × Database is not focus
 - × Algorithm & source code structure



Microservices & Client

- × Spring Boot
 - × Kafka Order : port 9001
 - × Kafka Pattern
 - × Kafka Reward
 - × Kafka Storage
- × Client
 - × Postman : download from **postman.com**



Organizing Source Code



About This Lesson

- × From other course regarding Microservice Architecture & Patterns
- × Why & how to organize Spring code for microservice
- × Refer to **Resources & References** (last section of the course) for discount code

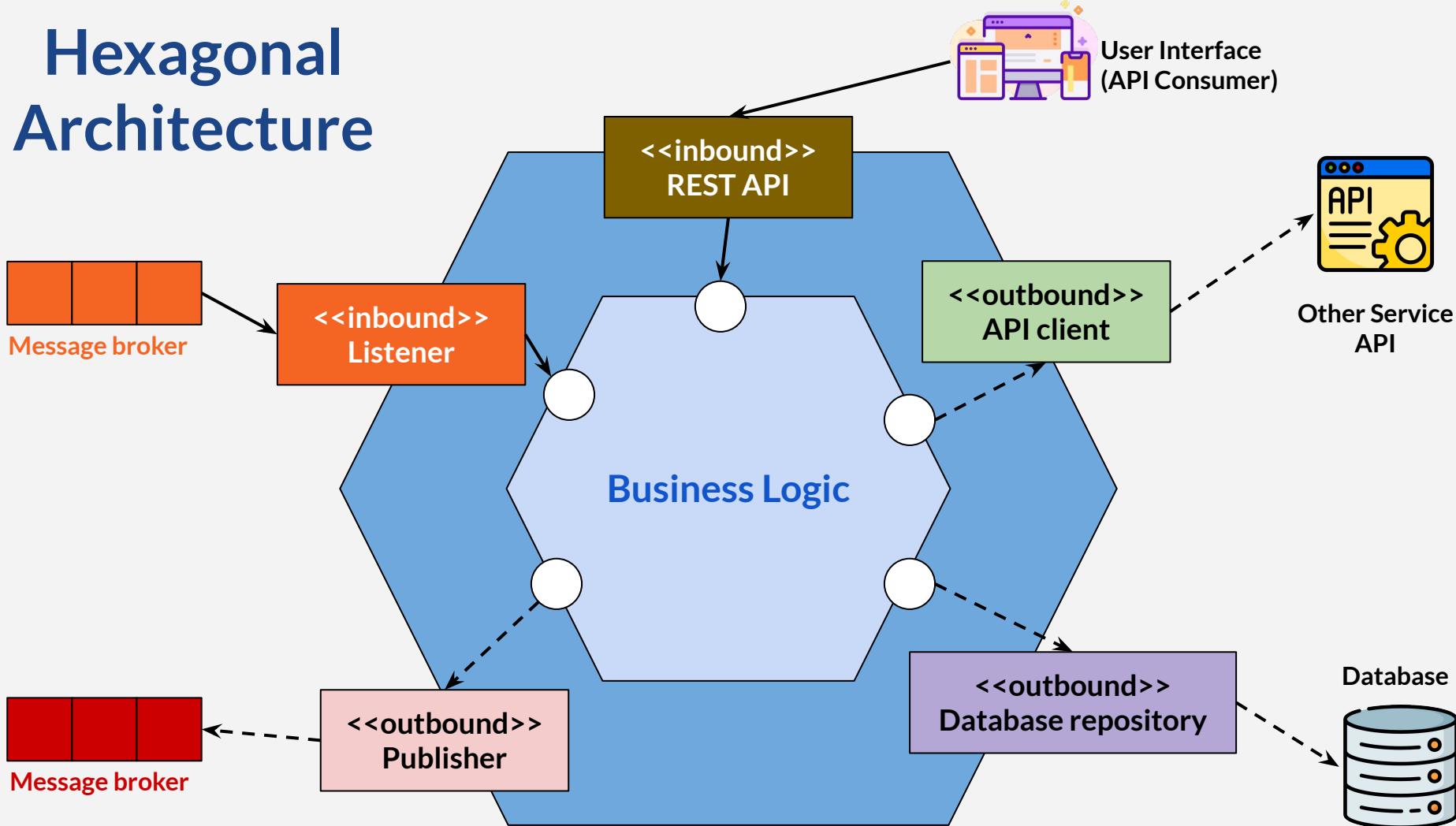


The Need of Organizing

- × Many codebase
- × Applications
 - × Easy to work with
 - × Easy future changes
- × Easy transfer knowledge & employee onboarding
- × Multiple coding & source code organizations
- × Patterns for code structure & organization



Hexagonal Architecture

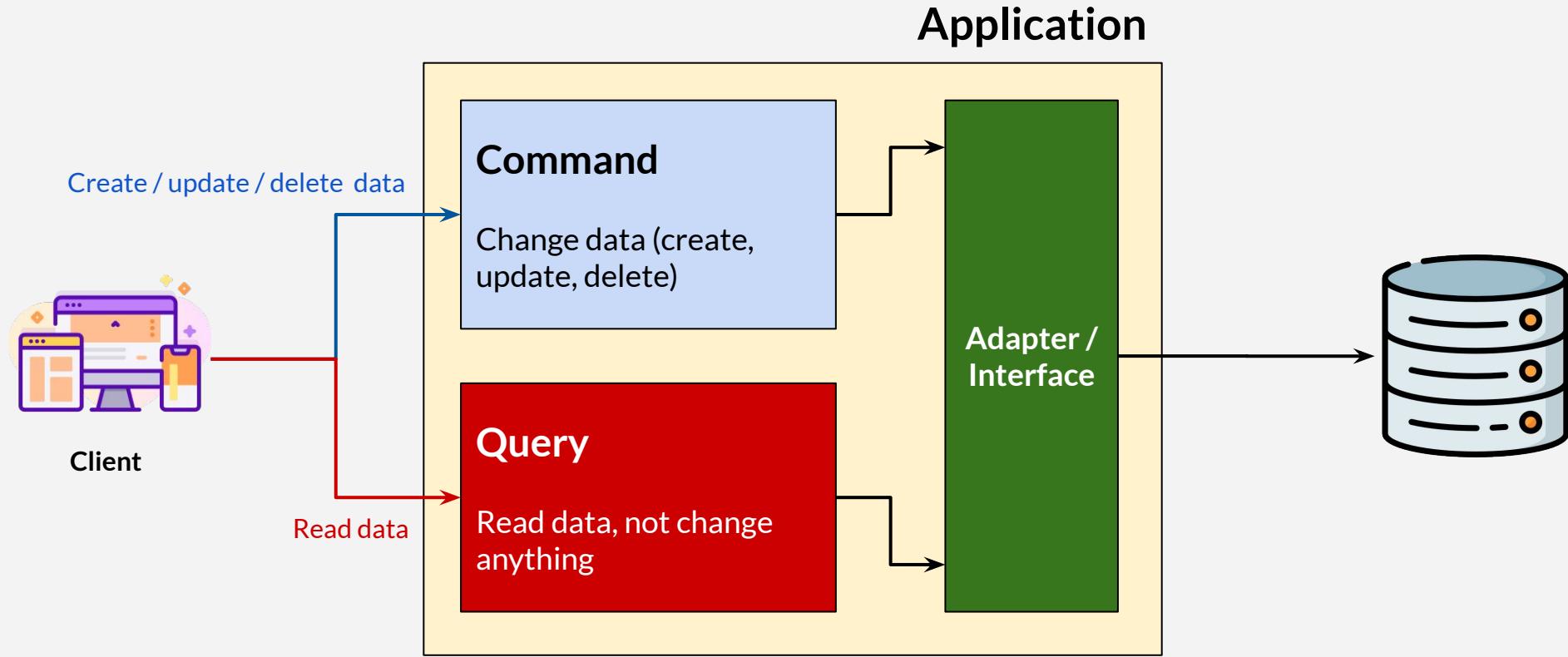


Hexagonal Architecture

- × Benefit : decouple business logic with data access
- × Easier to test or change
- × Communication using multiple adapters



Command Query Separation (CQS)

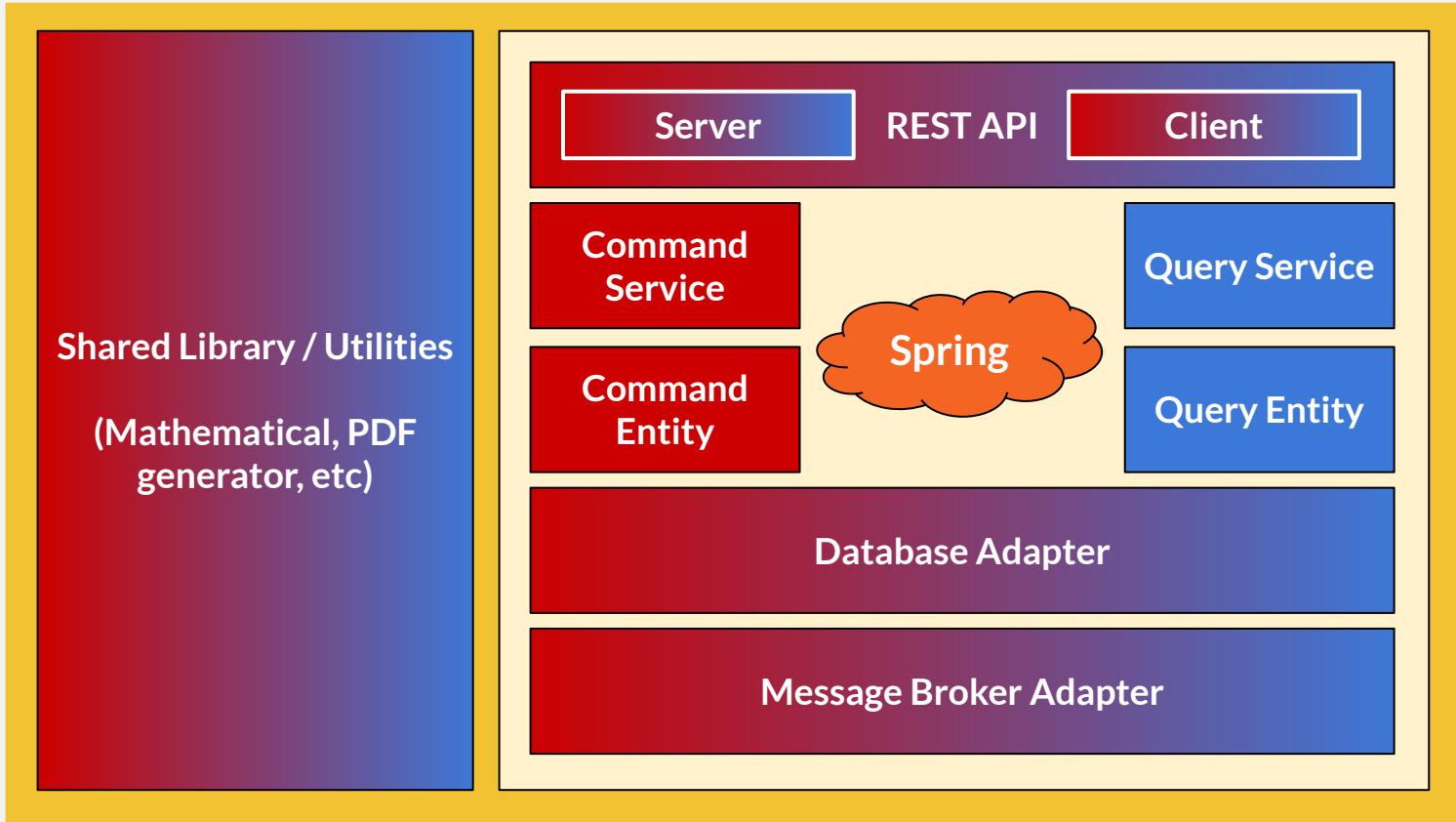


Command Query Separation

- ✗ Command / transaction / modifier / mutator
- ✗ Query / view
- ✗ Separate, don't mix
- ✗ Easier maintenance and change



Application source code



Setting Up The Projects



Spring Initializr

- × Generate 4 java / gradle projects from start.spring.io
 - × Group: **com.course.kafka**
 - × Artifact: **kafka-ms-order**, **kafka-ms-pattern**,
kafka-ms-reward, **kafka-ms-storage**
 - × Package name: **com.course.kafka** (remove any suffix)
- × Spring boot 3.x + Java 21



Dependencies

- ✗ **kafka-ms-order**
 - ✗ Web
 - ✗ Spring kafka
 - ✗ JPA
 - ✗ H2 database
- ✗ **Other 3 projects**
 - ✗ Spring kafka



Spring Application Config - Listener

```
spring:  
  kafka:  
    listener:  
      missing-topics-fatal: false
```



Topic exists:
- t.one
- t.two

```
public class SomeListener {  
  
  @KafkaListener(topics = "t-nine")  
  public void listen() {  
    ..  
  }  
}
```

Good to go!



Message Converter

```
public void send(MyMessage myMessage) {  
    var myJSONString = objectMapper.writeValueAsString(myMessage);  
    kafkaTemplate.send("topic", "key", myJSONString)  
}
```



Java > Kafka : **Serialize**
Kafka > Java : **Deserialize**

Spring Message Converter

Java
object

Kafka (with
JSON as body)

Spring Application Config - Producer

```
spring:  
  kafka:  
    producer:  
      bootstrap-servers:  
        - my.kafka.com:9092  
        - another.kafka.com:9915  
        - 192.168.5.100:29092
```

publish to



```
key-serializer: org.apache.kafka.common.serialization.StringSerializer  
value-serializer: org.springframework.kafka.support.serializer.JsonSerializer
```

KafkaTemplate<String, Object>

publish message as

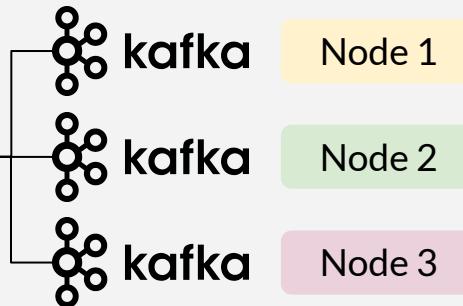
Key : String

Value : Object's JSON Representation

Spring Application Config - Consumer

```
spring:  
  kafka:  
    consumer:  
      bootstrap-servers:  
        - my.kafka.org:9092  
        - different.kafka.com:9092  
        - 192.168.18.25:29092
```

publish to



```
key-deserializer: org.apache.kafka.common.serialization.StringDeserializer  
value-deserializer: org.springframework.kafka.support.serializer.JsonDeserializer
```

listen message as

Key : String

Value : Object's JSON Representation

Spring Application Config - Consumer

```
spring:  
  kafka:  
    consumer:  
      properties:  
        spring:  
          json:  
            trusted:  
              packages: com.course.kafka.broker.message
```

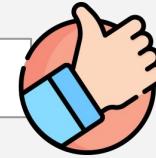
trusted
packages: "*"

I will only
deserialize these
packages



Variable Declaration

```
List<Item> items = service.getItems();
```



```
var items = service.getItems();
```



Auto Create Topics From Code



Create Topic Programmatically

- ✗ Part of Spring Kafka
- ✗ Careful when do it (especially on production)

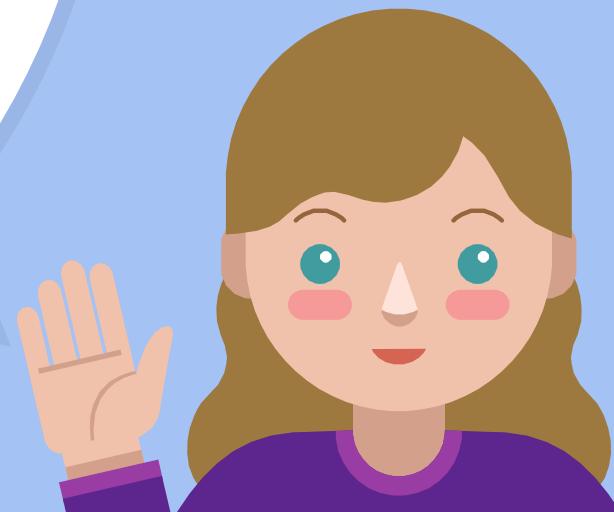


Order App Database



Order App

Kafka Producer



Order - Kafka Message



- ★ order number
- ★ location
- ★ order date & time
- ★ credit card number

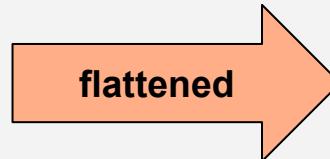


Kafka Message



Each item has

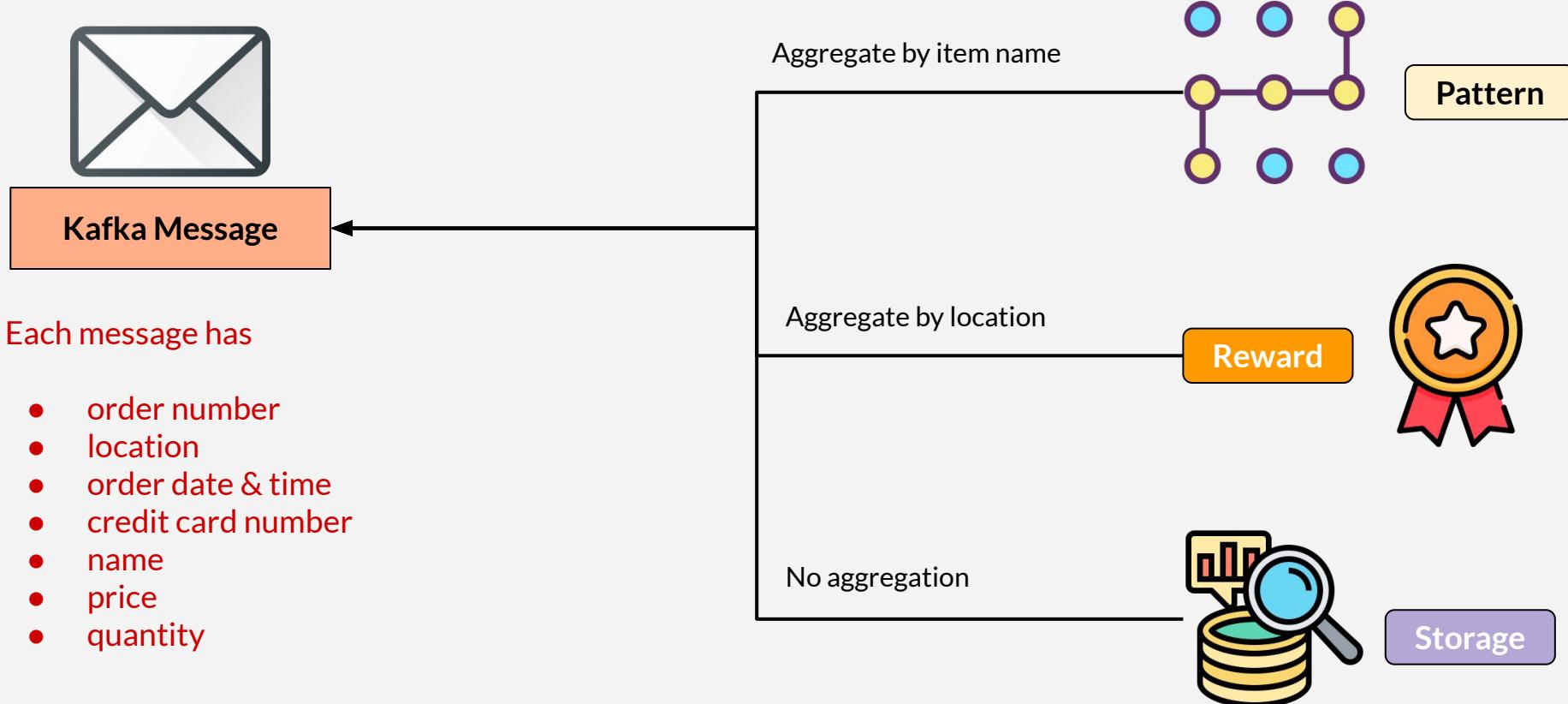
- name
- price
- quantity



Each message has

- order number
- location
- order date & time
- credit card number
- name
- price
- quantity

Order - Kafka Message



Handle Kafka Publish Result Producer Callback



KafkaTemplate Javadoc

send

```
public CompletableFuture<SendResult<K,V>> send(String topic,  
                                              K key,  
                                              @Nullable  
                                              V data)
```

Description copied from interface: KafkaOperations

Send the data to the provided topic with the provided key and no partition.

Specified by:

send in interface KafkaOperations<K,V>

Parameters:

topic - the topic.

key - the key.

data - The data.

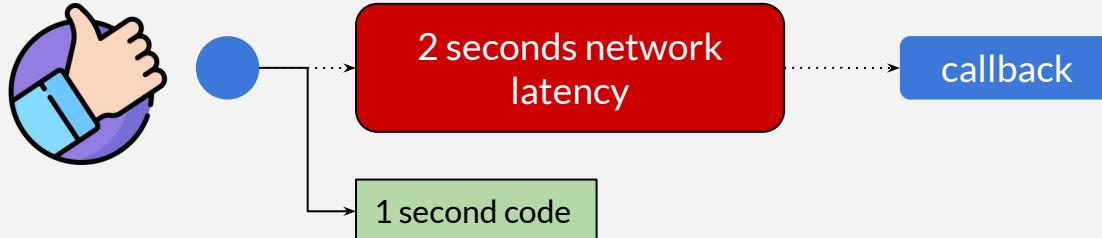
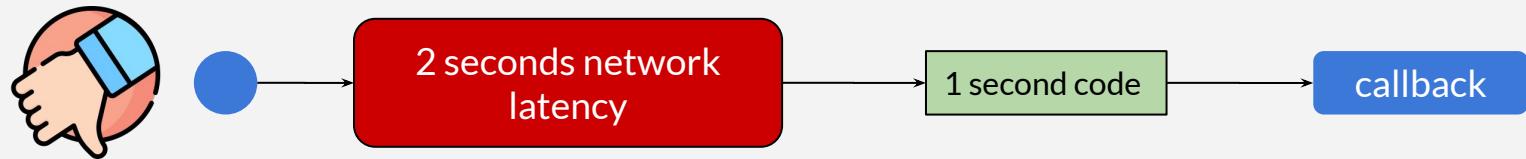
Returns:

a Future for the SendResult.

Publisher Callback

```
KafkaTemplate.send(topic, key, data)
```

```
// some other code that needs 1 second to execute
```



Order App

API & Finalize App



What We Will Build

- × REST API for submit order
- × Service & action for business logic
- × API is the entry point for business logic service & action



Service? Action?

And some things that should not have been forgotten
were lost.

History became legend.

Legend became myth.

And for two and a half thousand years, the ring
passed out of all knowledge.

Until, when chance came, it ensnared another
bearer.

`validateEmployee()`

`insertEmployeeToDatabase()`

`enablePayroll()`

`prepareOnboarding()`

`sendNotification()`

Data Change

- × Database change? Then publish message
 - × Communicate with other system
 - × Data analytics
- × We called "Change Data Capture (CDC)"
- × Database operation + message publishing
- × Manually writing code
 - × Technical error
 - × Human error



Consistent Publishing

- × Make sure database transaction & kafka publish is atomic
- × Database log tailing
- × Transactional outbox
- × See last lesson for **Microservice Architecture & Pattern** reference



Order Structure

- × Different class for API request & entity?
 - × **entity**: represents database entity
 - × **api.request**: represents API request body
- × Structure similar, but fields can differs
- × Avoid mixing class responsibility (database & API request body)
- × Cleaner code
- × Feel free to use different approach



Order App

Test the Application



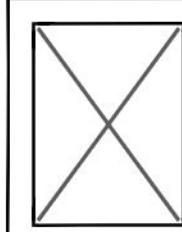


Order location

Credit Card Number

1958 2850 6094 3758

Items

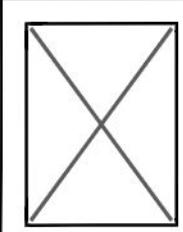


A beautiful book

\$14



Quantity

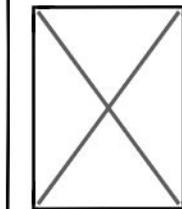


An exotic fruit

\$3



Quantity

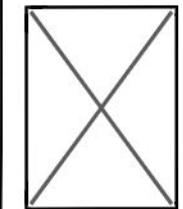


A mysterious box

\$14



Quantity

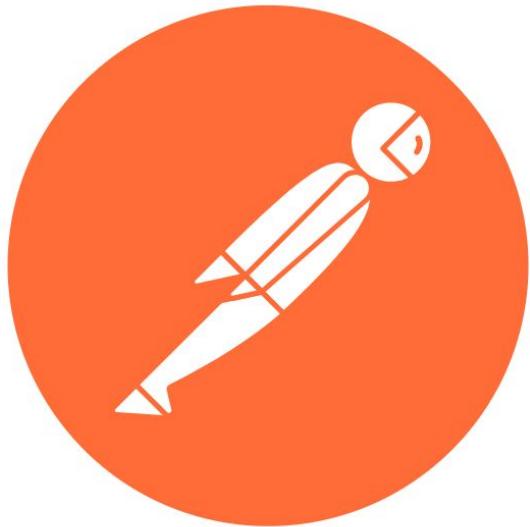


A luxury dress

\$26



Quantity



POSTMAN

Download from : postman.com

Producer Retry



Producer Retry

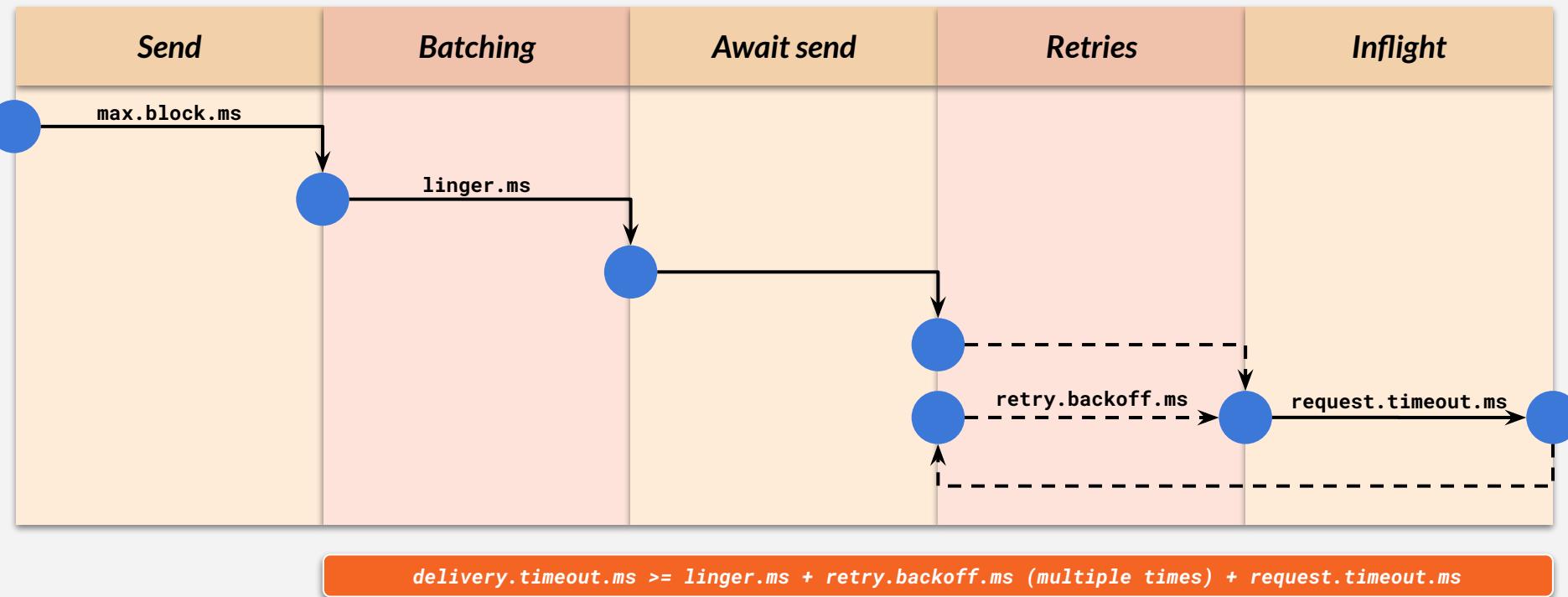
- × Encounter transient (temporary) errors
- × Handled by setting producer retry
- × Retry for transient errors does not involve programming
- × Non-transient, e.g. invalid message format, unavailable broker, generally handled on application code



Setting Producer Retry

- × Configure **spring.kafka.producer.retries** to be more than 0
- × Default value is more than zero (producer retry enabled by default)
- × Configure **retry.backoff.ms** to set backoff period (default is 100 ms)





max.block.ms : time to wait for metadata fetch and buffer allocation

linger.ms : time to wait to allow other records to be sent

retry.backoff.ms : time to wait before retrying the failed request

request.timeout.ms : time to wait for the response from Kafka broker

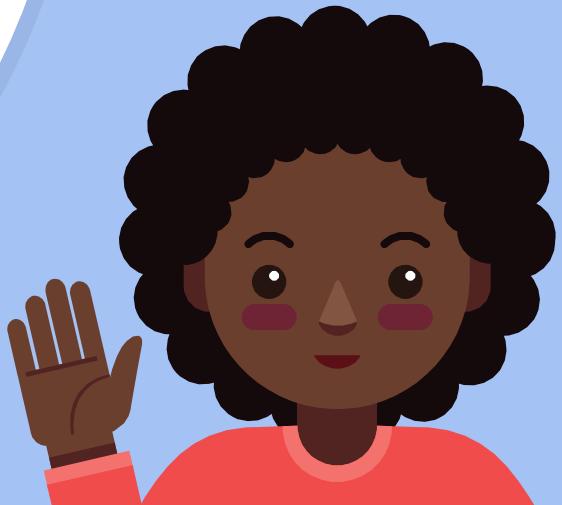
delivery.timeout.ms : time to wait for Kafka to deliver a message to a broker, before considering it a failed

Out-of-Order on Retry

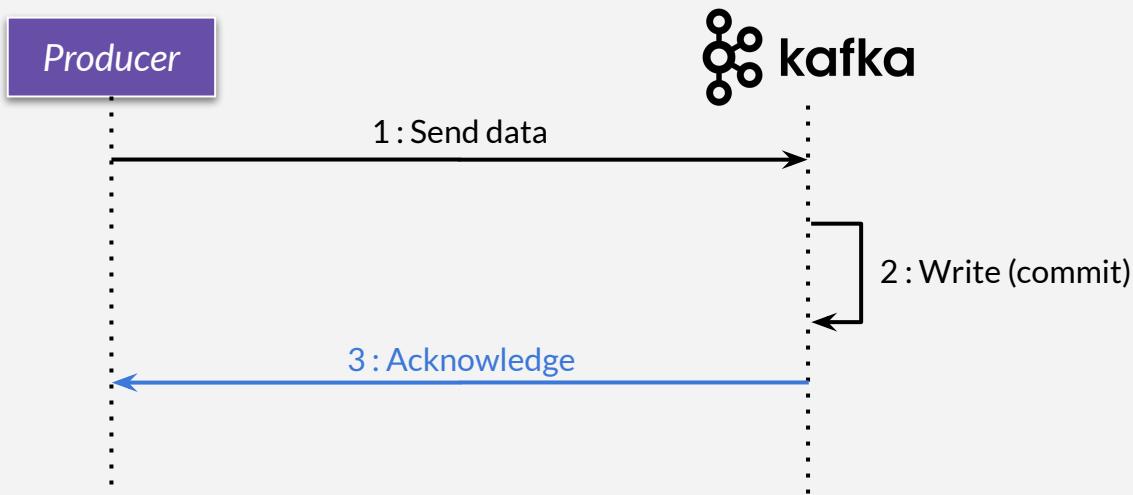
- ✗ Messages sent during retries may out of order
- ✗ Possible causes
- ✗ Could be an issue
- ✗ Use idempotent producer to handle



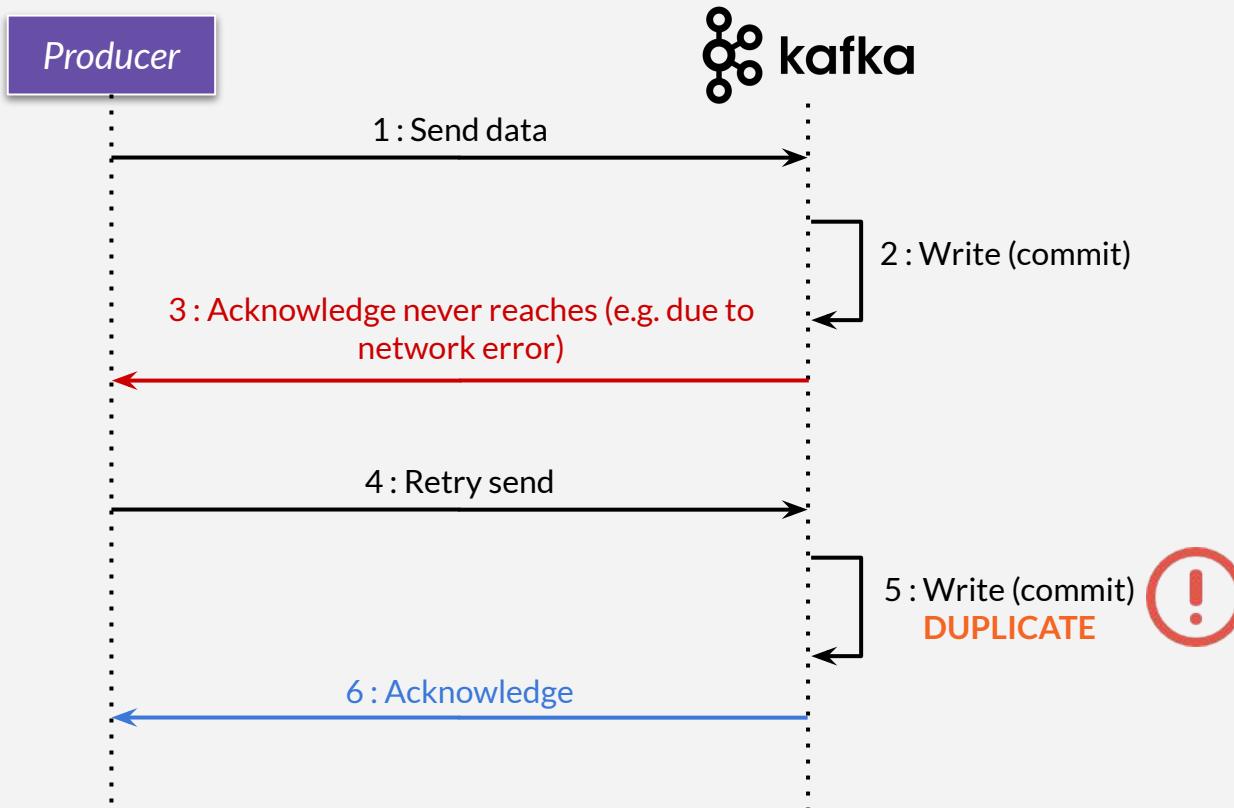
Idempotent Producer



Good Produce



Duplicate Message

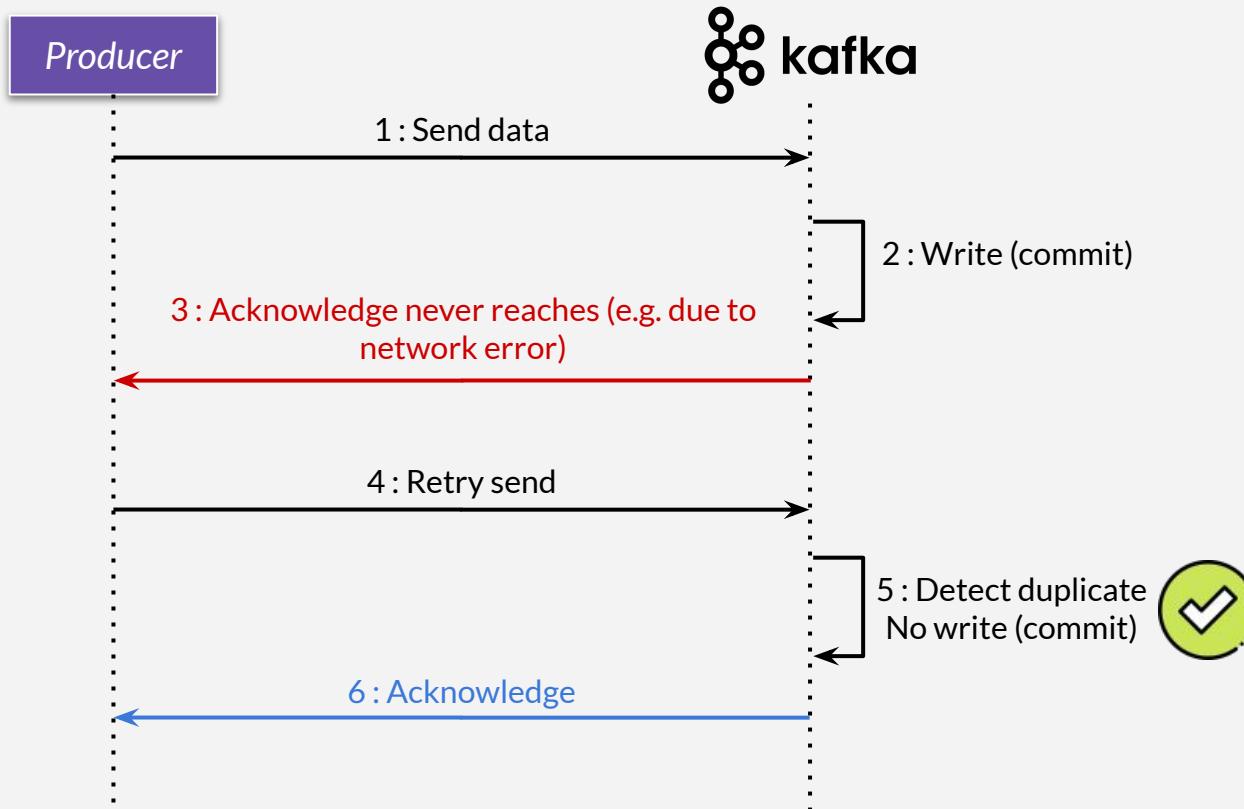


Idempotent Producer

- × Use idempotent producer
- × Guarantee deduplicates (eliminate duplicate)
- × Important where message duplication can cause issue



Idempotent Producer



Idempotent Producer

- × Set `enable.idempotence = true`
- × Kafka will use this configuration
 - × `retries = 2147483647`
 - × `max.in.flight.requests.per.connection = 5`
 - × `acks = all`
- × Simplify producer logic



Pattern App

Kafka Consumer



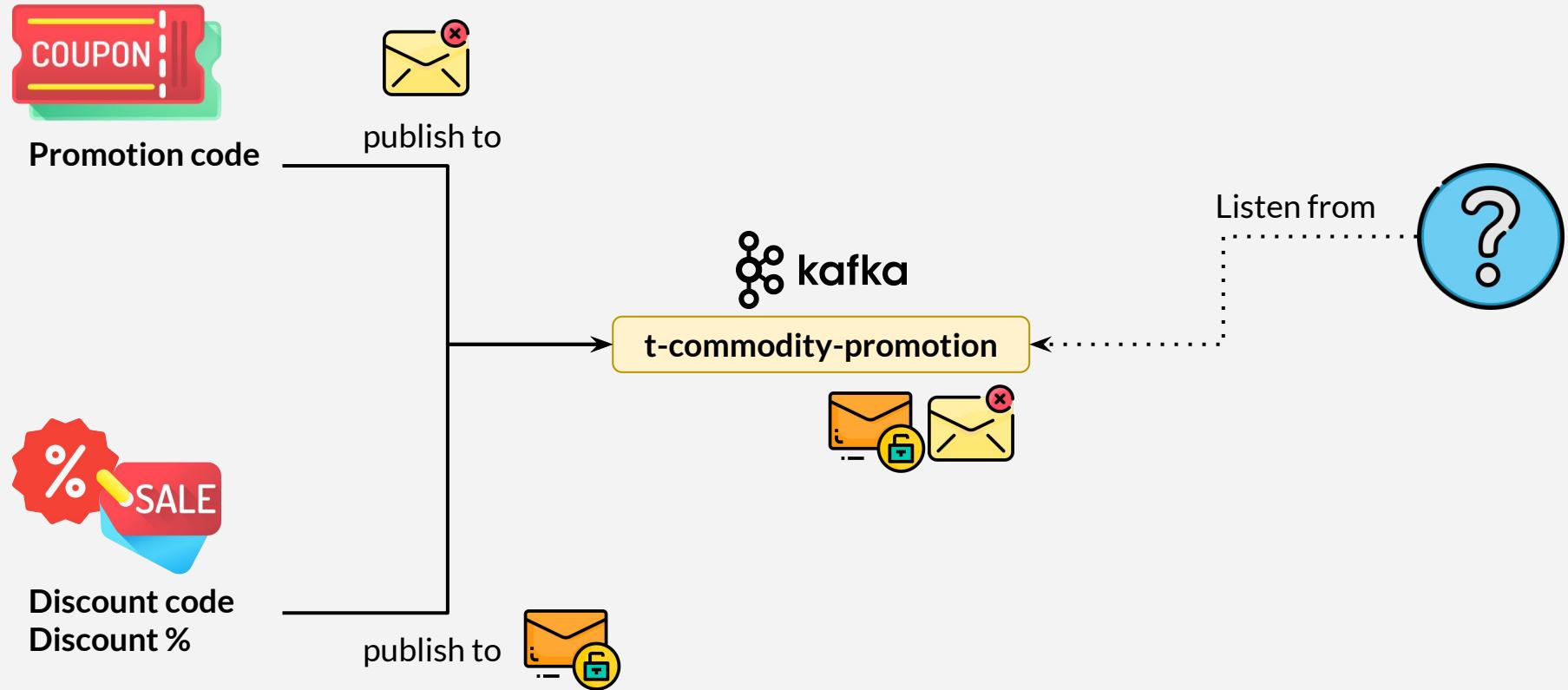
Order App Promotion Producer



Order App Discount Producer



Promotion & Discount



Source Code for Discount

- ✗ `Discount*.java`
- ✗ Package `com.course.kafka`
 - ✗ `api.request`
 - ✗ `api.server`
 - ✗ `broker.message`
 - ✗ `broker.producer`
 - ✗ `command.action`
 - ✗ `command.service`

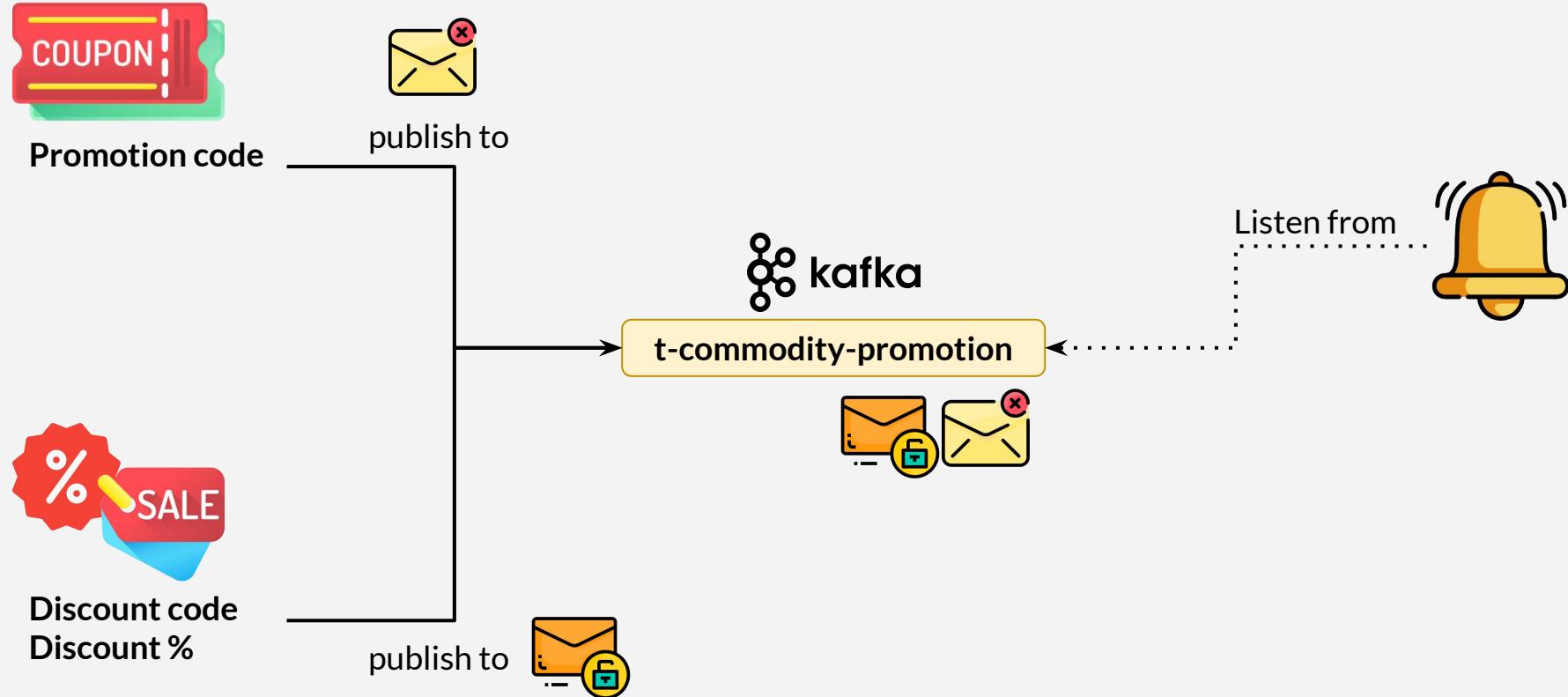


Storage App

Kafka Consumer



Promotion & Discount



Order App

Add Header To Message



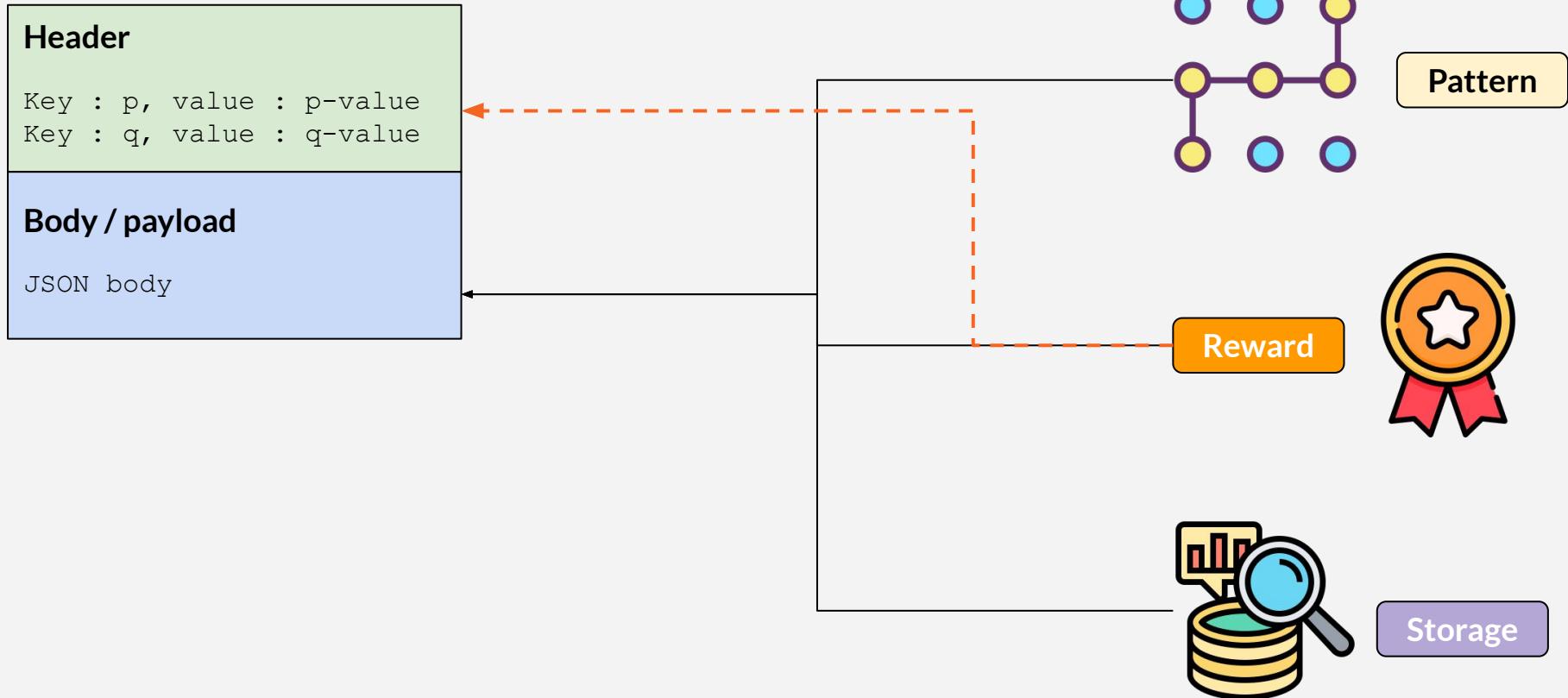
Change Requirement



Additional Bonus!

- Branch A, B, C : x% bonus
- Other branch : y% bonus
- Only for one week

Solution?



Reward App

Kafka Consumer



Request / Reply

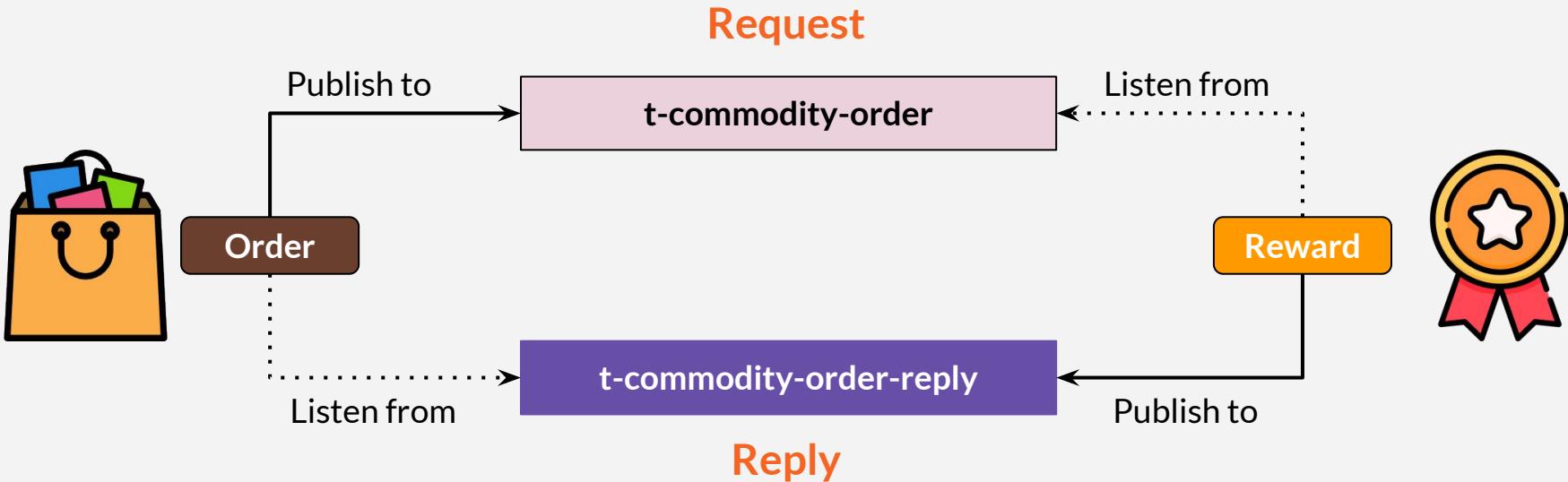


Order Team Says



- Reward team, please confirm order received
- We provide kafka topic for confirmation

Asynchronous Request / Reply



Asynchronous Request / Reply

```
public class ConsumerOnReward {  
  
    @KafkaListener(topics = "t-commodity-order")  
    @SendTo ("t-commodity-order-reply")  
    public OrderReplyMessage listen(OrderMessage requestMessage) {  
        // do some process for request  
        //  
  
        OrderReplyMessage replyMessage = new OrderReplyMessage();  
        return replyMessage;  
    }  
  
}
```

Message Compression

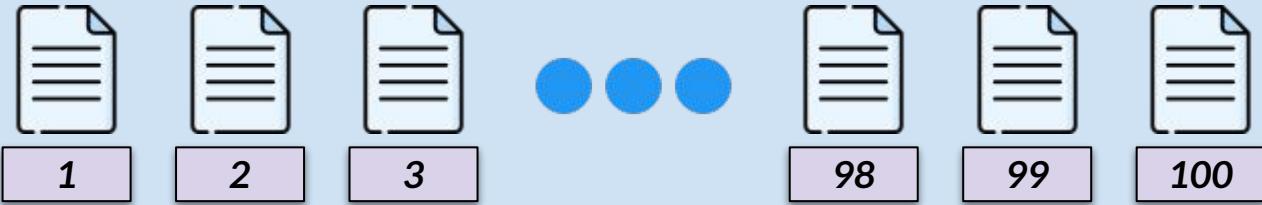


Message Compression

- × Reduce size of messages before transmitted
- × Improves overall system performance
- × Configurable in producer or broker
- × Available compression types: `none`, `gzip`, `lz4`,
`snappy`, `zstd`



*Producer
batch*



Compress batch



*Compressed messages
(smaller size)*

Send the compressed messages



Message Compression

- × Advantages:
 - × Smaller request (up to 4 times smaller)
 - × Faster transmission, better disk utilization
- × Disadvantages
 - × Additional CPU cycles on producer
 - × Additional CPU cycles on consumer
- × Small disadvantages compared to advantages



Message Compression

- × On Spring, set key
spring.kafka.producer.compression-type
- × Broker's default value is **producer**
 - × Take compressed batch as is from producer
 - × Producer need to configure the compression
(easily done on Spring)



Kafka Batch

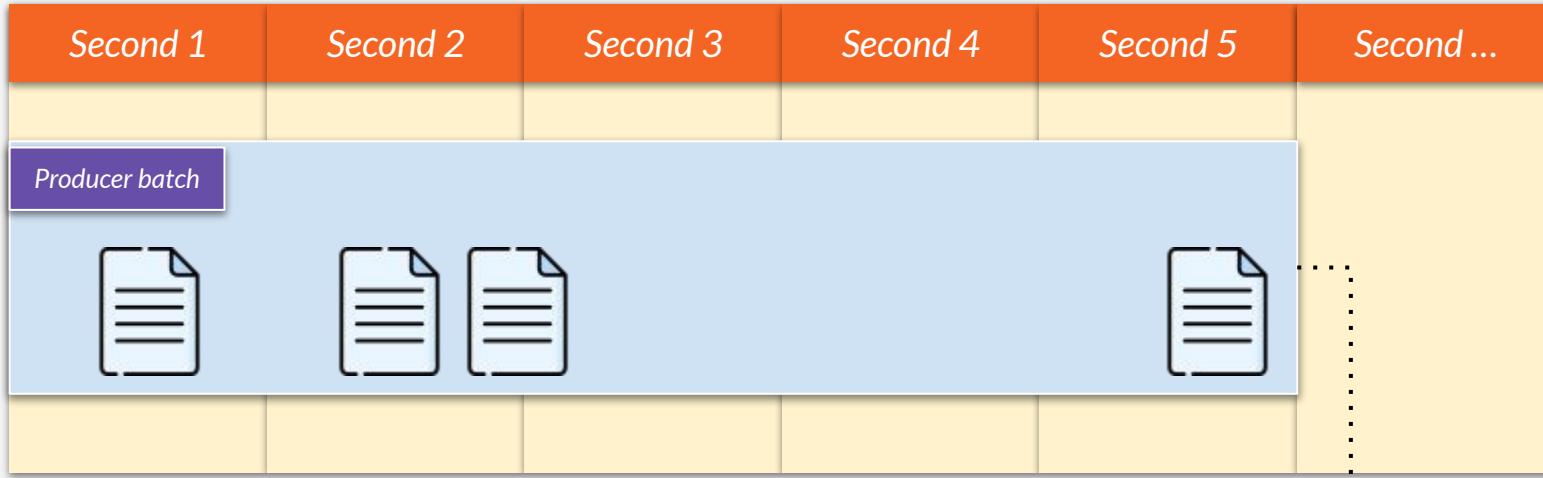
- × Further tweak : linger millisecond & batch size
- × Default behavior : send message as soon as possible
- × Batching increase throughput with low latency
- × Further benefit with compression (higher compression ratio)



Linger ms & Batch Size

- × **linger.ms** : how long to send a batch
 - × Default value is 0 (immediately send message)
 - × Example
- × **batch.size** : maximum batch size (bytes) before start send (if linger has not been elapsed)
 - × Increase batch size for larger byte size per batch





linger.ms set to 5000 (5 second)

Compressed messages
(smaller size)

Compress batch

Send the compressed messages



Batch Size

- × Maximum byte sizes for each byte
 - × Default: 16384
- × Increasing batch size leads to efficiency
- × Message bigger than batch size will not be batched
- × Set batch size to 0 to turn off batching
- × Set on Spring property
spring.kafka.producer.batch-size



Summary

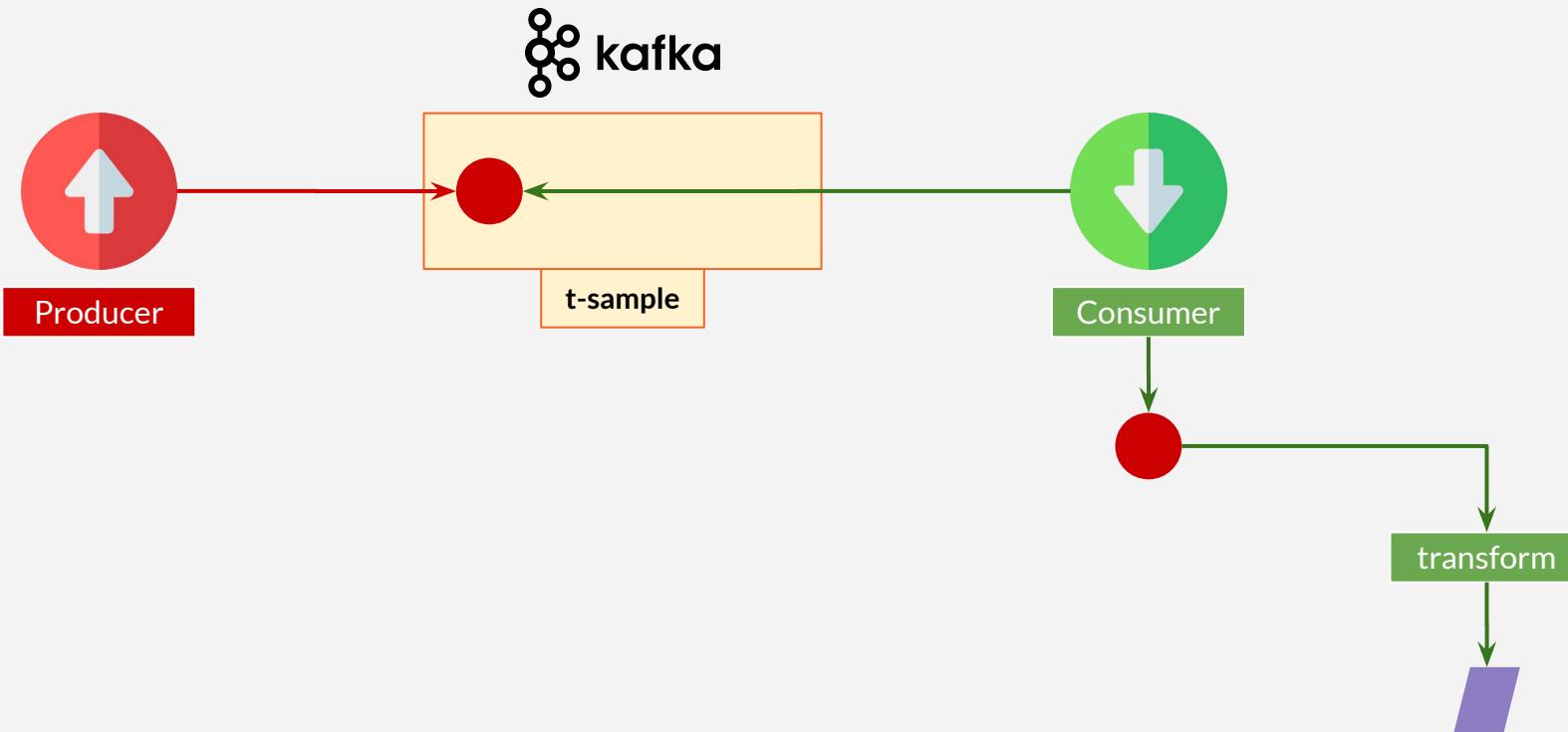
- × Increase linger millisecond to make producer wait for the batches to fill up
 - × Delay sending, use with careful
- × Increase batch size to send larger batch
- × Enable producer compression to be more efficient



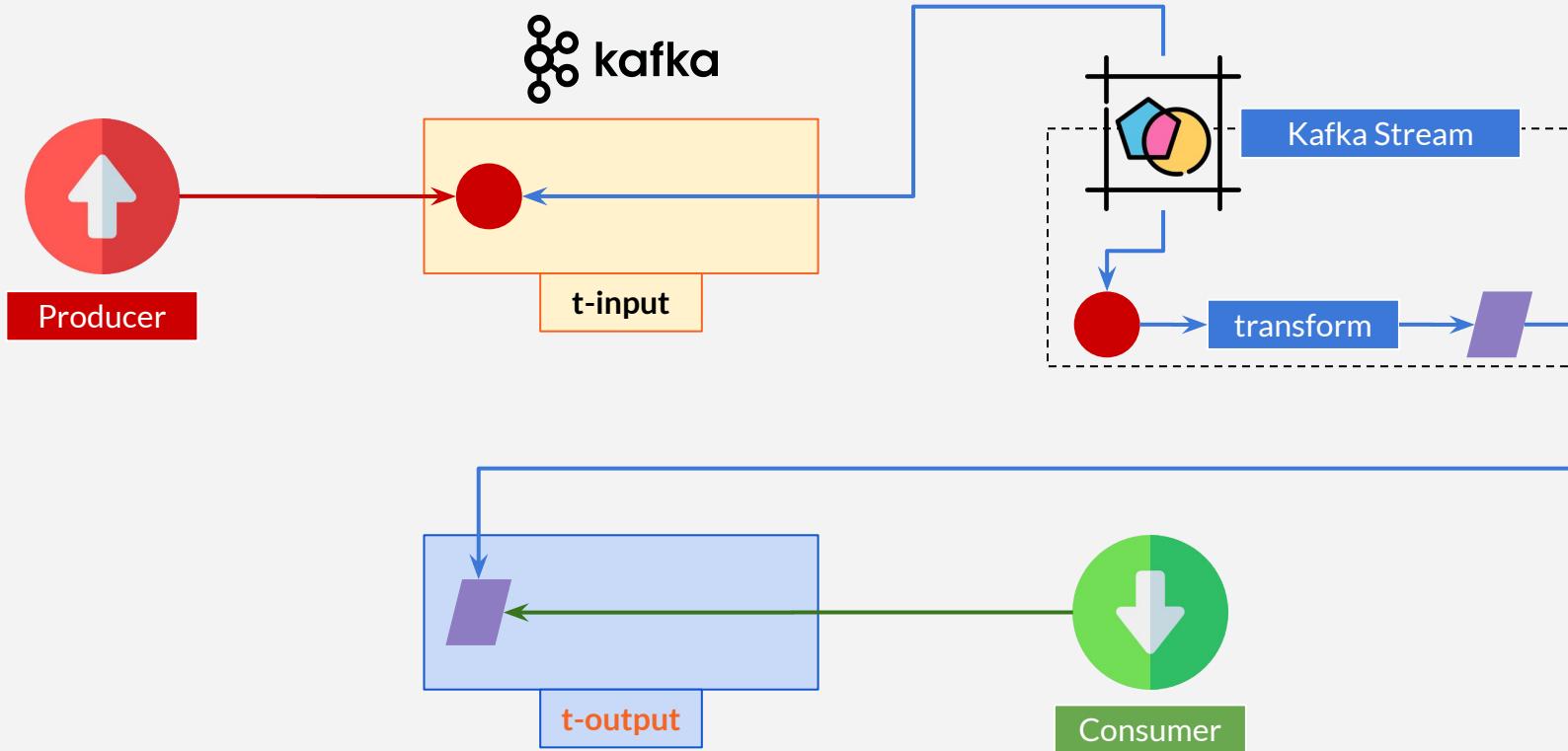
Introducing Kafka Stream

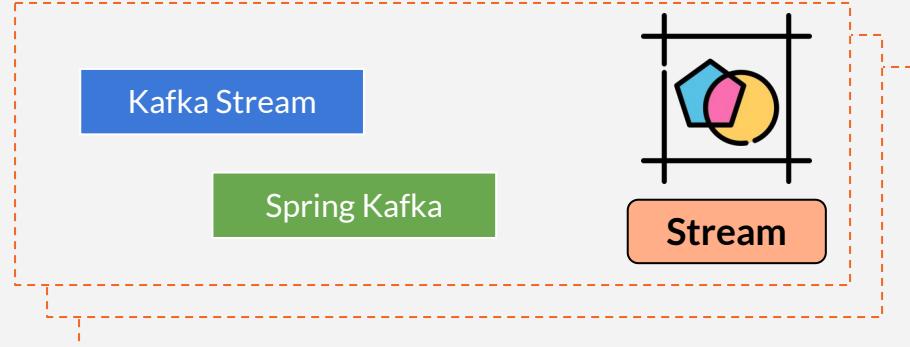
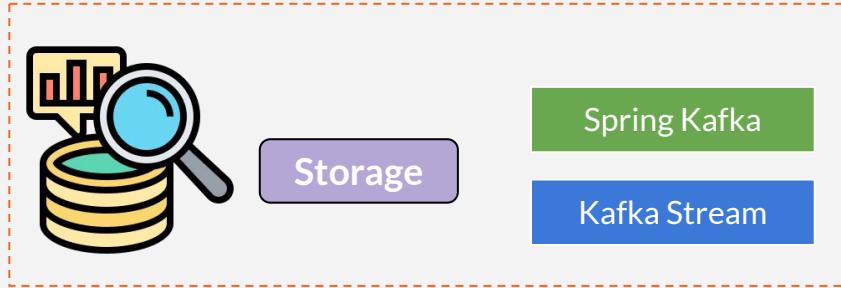
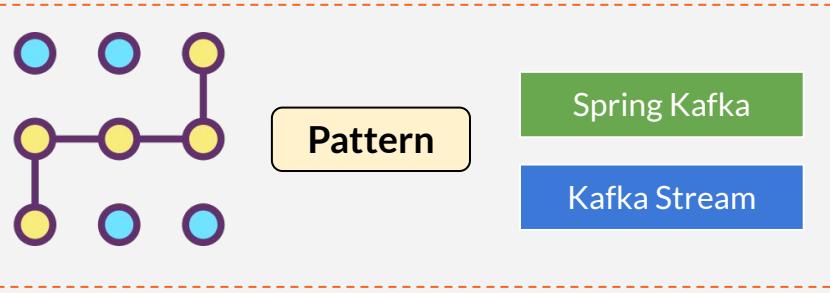


Data Transformation



Kafka Stream





Kafka Stream

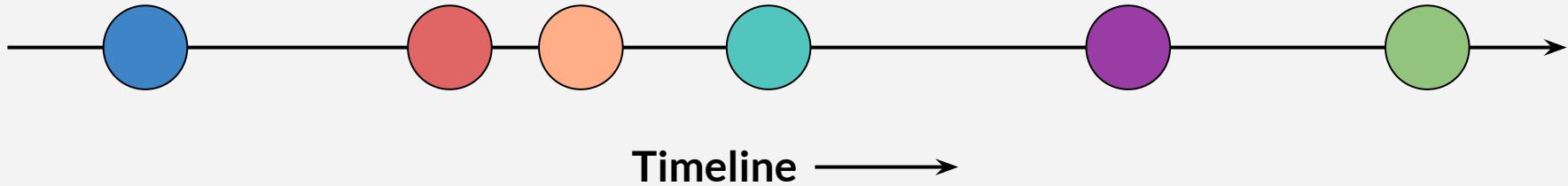
- × Stream processing framework
- × Released on 2017
- × Alternative for Apache Spark, Nifi, or Flink
- × Stream & stream processing?



Stream Processing

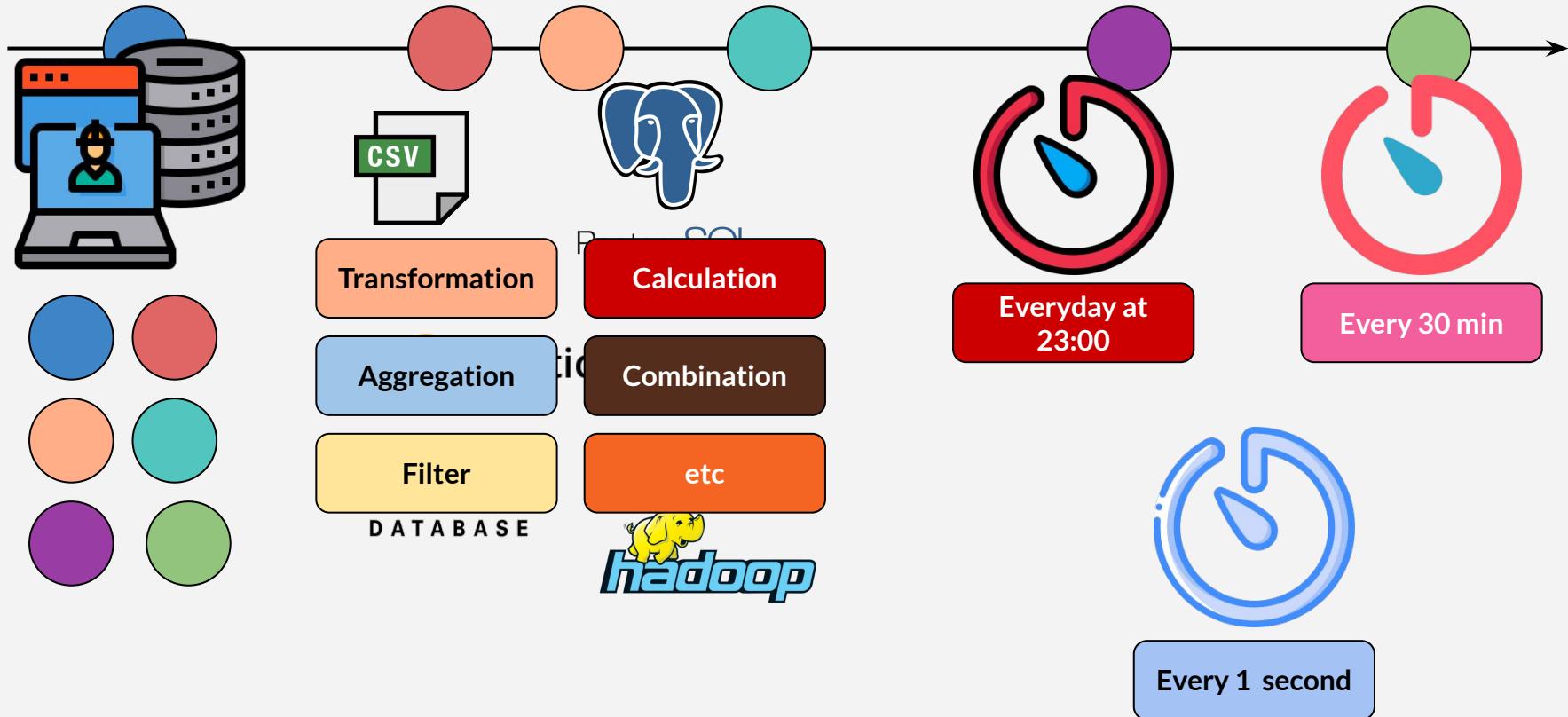


Data Stream

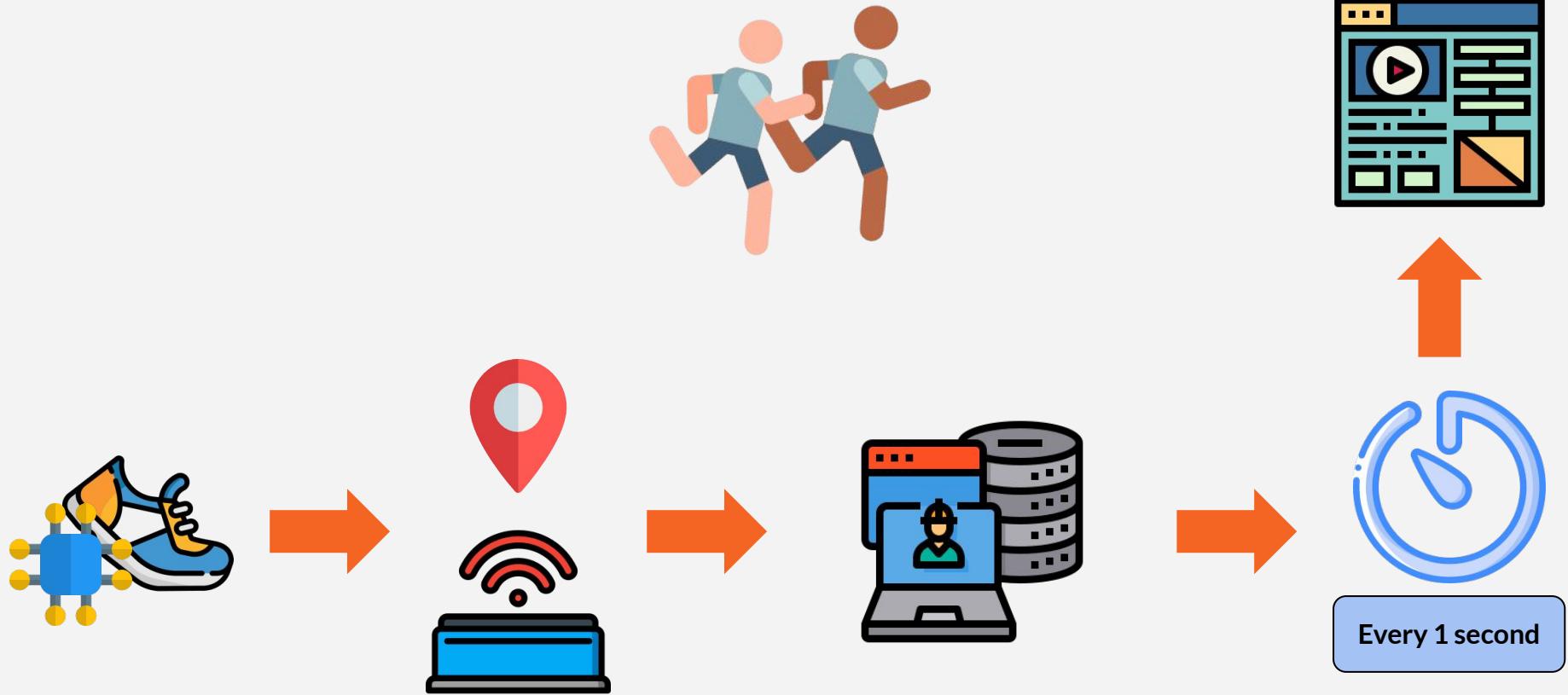


Each circle represents a data
Endless
Data (event) is immutable
Can be replayed
Think of Kafka topic as stream

Data Processing



Micro Batching



Micro Batching

Sunday
23:00

Monday
23:00

Tuesday
23:00



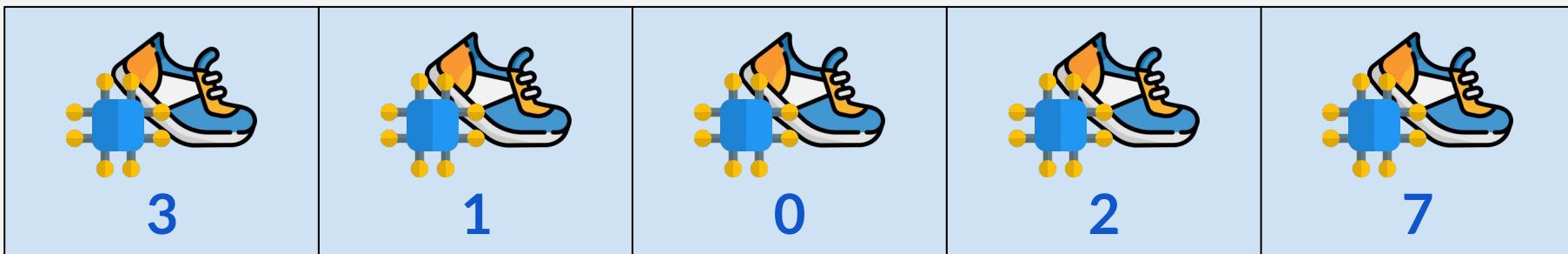
00:01

00:02

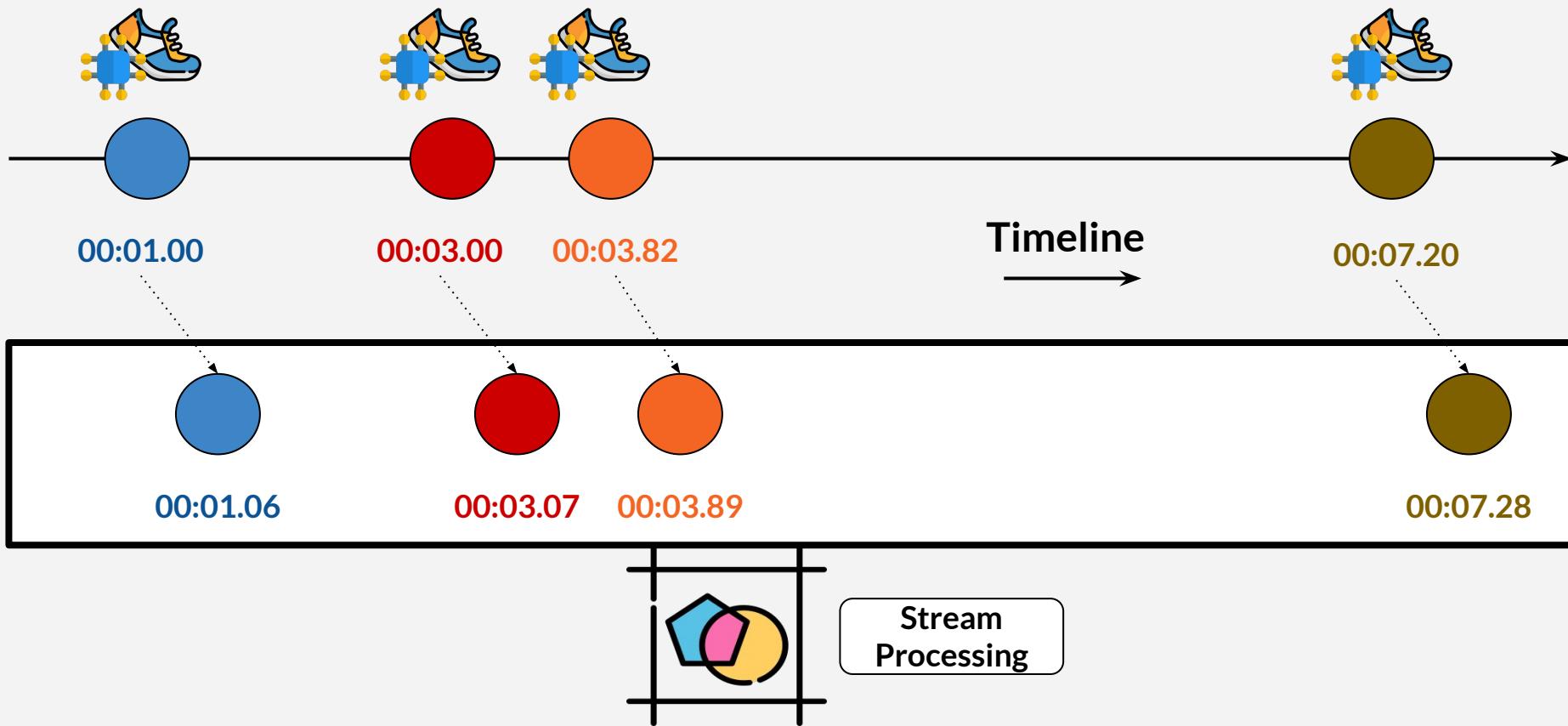
00:03

00:04

00:05



Stream Processing



Yes To Stream Processing

- × Yes, when:
 - × (relatively) fast data flow
 - × Application need to response quick to most recent data
- × Example
 - × Marathon
 - × Credit card fraud
 - × Stock trading
 - × Log analysis



No To Stream Processing

- ✗ Example
 - ✗ Daily interest
 - ✗ Forecasting



To Stream or Not To Stream?

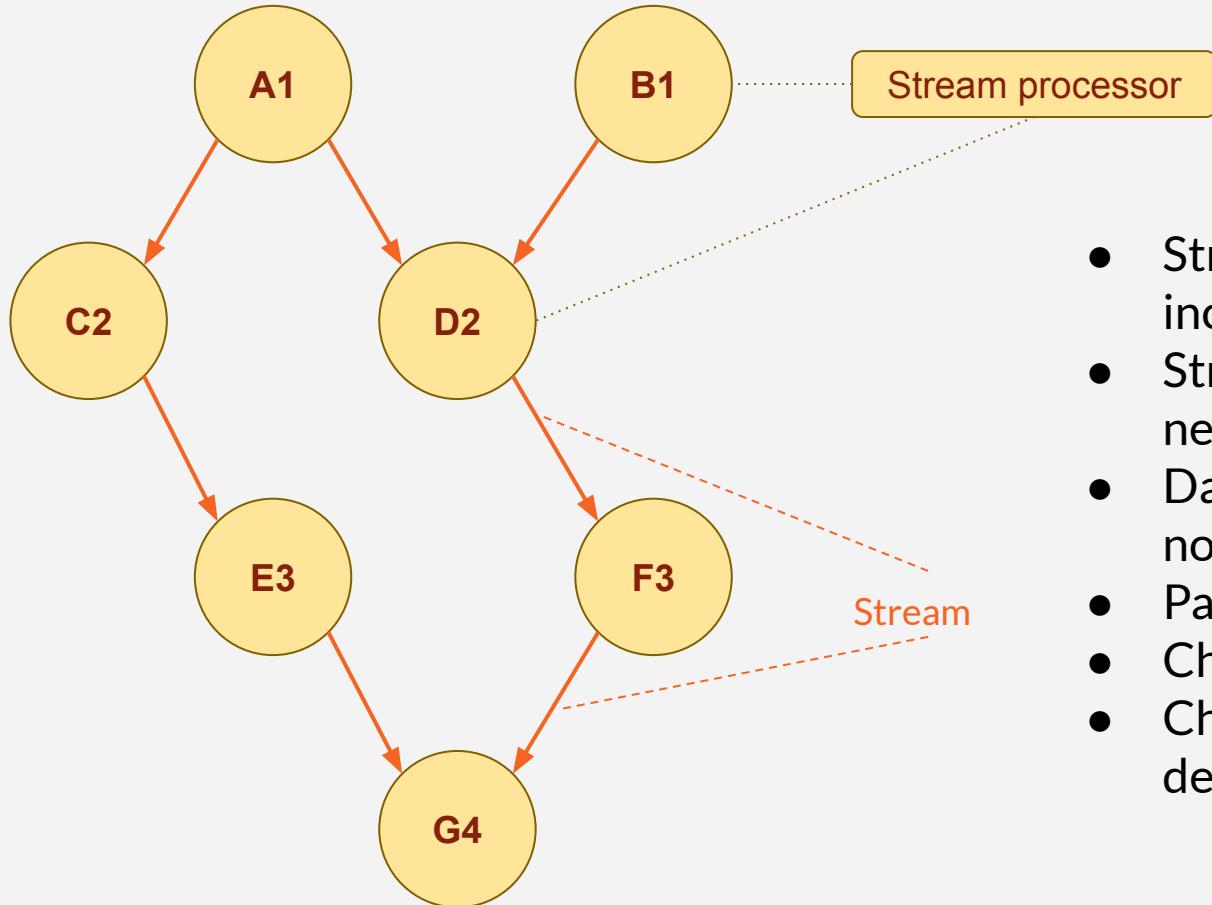
- ✗ Sample : check supply must be above threshold
- ✗ Depends on Service Level Agreement
- ✗ 30 minutes is good : batch
- ✗ Near real time : stream processing



Kafka Stream Concept

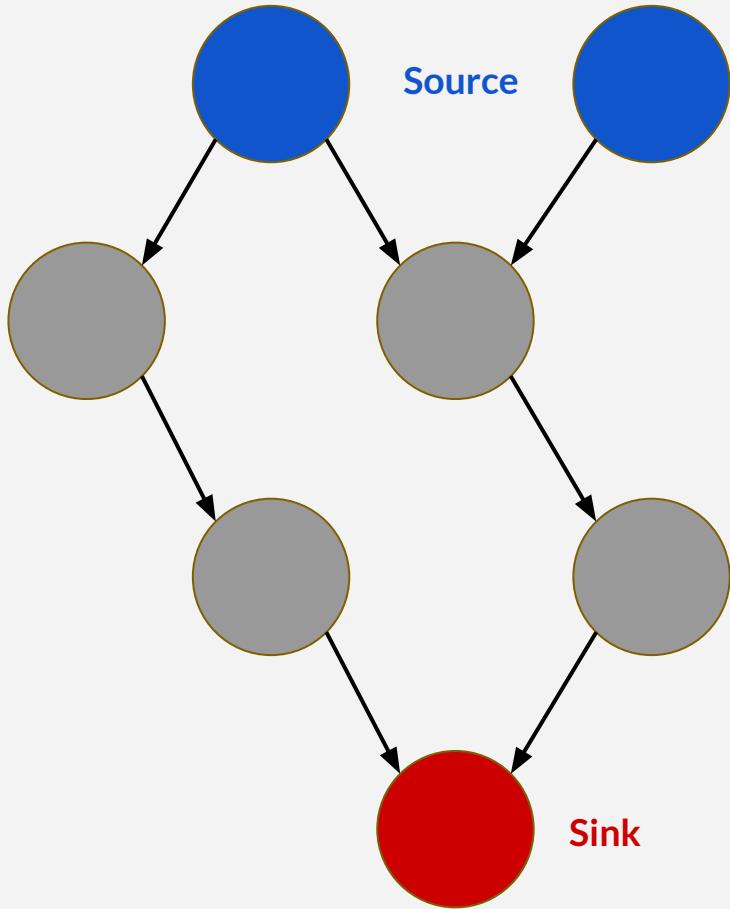


Topology / DAG



- Stream processor process incoming data stream
- Stream processor can create new output stream
- Data flows from parent to child, not vice versa
- Parent = upstream
- Child = downstream
- Child stream processor can define another child(ren)

Kafka Stream Topology



Source Processor

- Does not have upstream
- Consumes from one or more kafka topics
- Forwarding data to downstream

Sink Processor

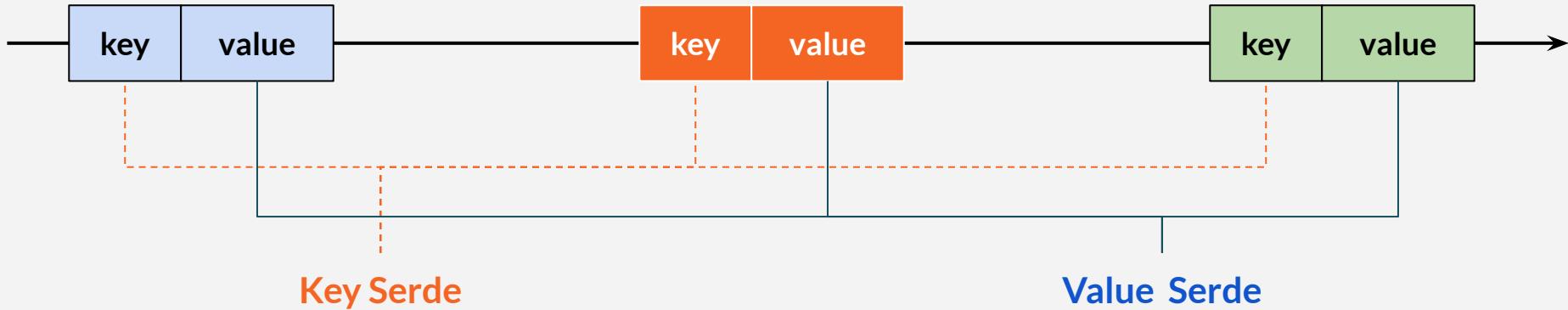
- Does not have downstream
- Receive data from upstream
- Send data to specified kafka topic

What We Will Learn

- × Low Level Processor API
- × **This Course : Kafka Stream DSL (Domain Specific Language)**
 - × Simple
 - × Provides common data transformation



Serde (Serializer / Deserializer)



```
Serdes.String()
```

```
Serdes.Long()
```

```
Serdes.ByteArray()
```

```
...
```

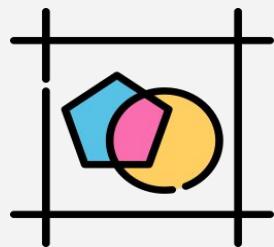
```
new JSONSerde<T>()
```

```
class CsvSerde<T> implements  
Serde<T>
```

Kafka Stream Preparation



What We Will Have



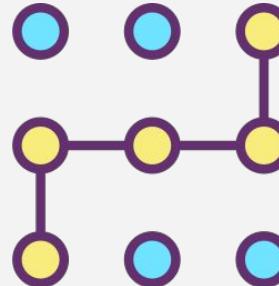
Stream

Order



Storage

Reward



Pattern

Preparation

- × Delete Kafka docker
- × Re-create kafka containers
- × Download & execute kafka stream scripts on last section (**Resources & References**)
- × Less detail on kafka console
- × Code style with Java lambda & functional (not in course scope)



Download Source Code

- × Available on last section of the course
- × Source code, postman, kafka script



Spring Initializr

- × Generate 1 java / gradle project from start.spring.io
 - × Group: **com.course.kafka**
 - × Artifact: **kafka-stream-sample**
 - × Package name: **com.course.kafka** (remove any suffix)
 - × Dependency: **Spring for Apache Kafka, Spring for Apache Kafka Streams**
- × Import the project



Additional Projects

- × Use projects from lesson kafka & microservice
- × Names will be **kafka-stream-xxxxx**
- × Same codebase with previous
- × Might have additional dummy transactions
- × Copy all java files under broker.message from kafka-stream-order to kafka-stream-sample



AI Assistant

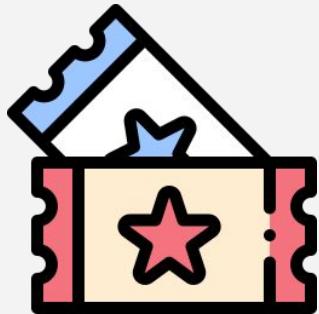
- × Use Github Copilot as AI assistant
- × Non-deterministic
- × Less reliable due to increased complexity
- × Use from chat panel / inline chat / source code comment / semi-transparent autocomplete



Hello Kafka Stream



Promotion



PrOmoTlOn7



PROMOTION7

Kafka Stream Basic Usage

build.gradle

```
implementation 'org.apache.kafka:kafka-streams:{$version}'
```

```
// define configuration
Properties props = new Properties();
props.put(...)
```

```
// create topology
KStreamBuilder builder = ...;
KStream stream = ...;
...
KafkaStreams stream = new KafkaStreams(builder, props);
```

```
// start stream
stream.start();
```

```
// stop stream
Runtime.getRuntime().addShutdownHook(new Thread(stream::close));
```

SpringKafkaStreamApp.java

Kafka Stream With Spring

build.gradle

```
implementation 'org.springframework.kafka:spring-kafka'  
implementation 'org.apache.kafka:kafka-streams'
```

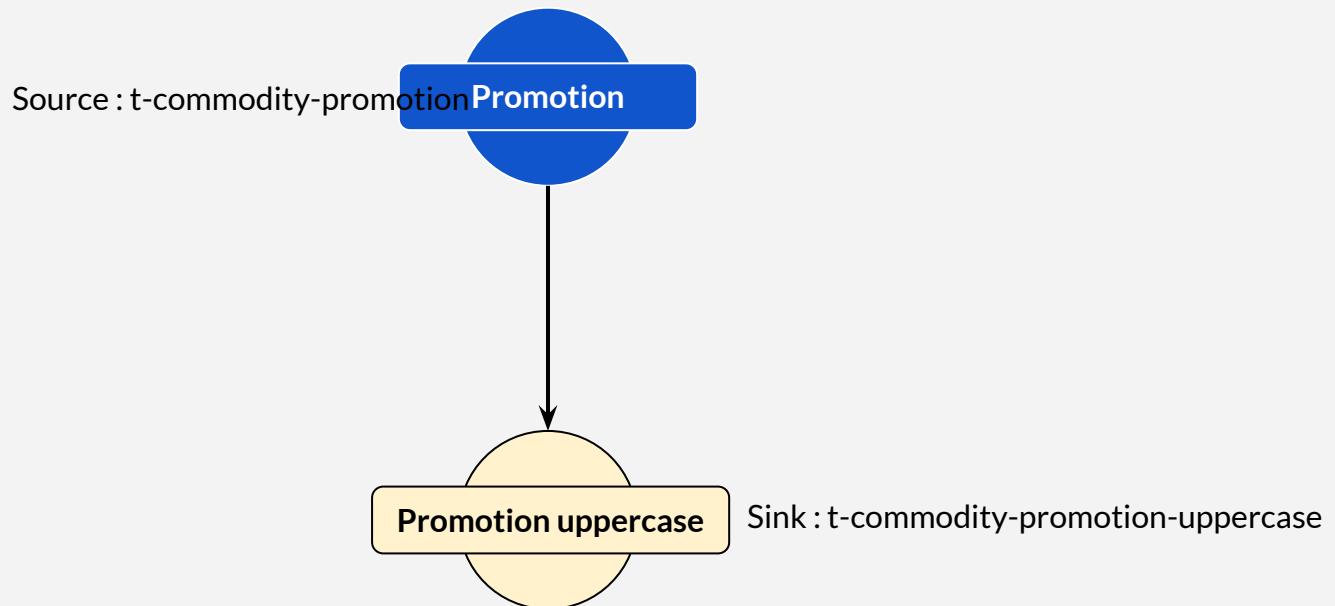
```
@Configuration  
@EnableKafkaStreams  
public class KafkaStreamConfig {  
    // define configuration  
}
```

KafkaStreamConfig.java

```
@Configuration  
public class MyStream {  
  
    @Bean  
    public KStream<String, String> kStream(StreamsBuilder builder) {  
        // define topology  
    }  
}
```

MyStream.java

Topology



Uppercase?

null	{"promotionCode":"rabbit35"}
------	------------------------------

null	{"PROMOTIONCODE":"RABBIT35"}
------	------------------------------



null	{"promotionCode":"RABBIT35"}
------	------------------------------



Hello Kafka Stream

Alternative Coding Style



```
@Configuration
public class MyStreamStyleOne {

    @Bean
    KStream<?, ?> streamMethod(StreamsBuilder builder) {
        var sourceStream = builder.stream(...);

        // ... process the stream

        return sourceStream;
    }
}
```

```
@Component
public class MyStreamStyleTwo {

    @Autowired
    void streamMethod(StreamsBuilder builder) {
        var sourceStream = builder.stream(...);

        // ... process the stream
    }
}
```

Coding Style

- × The course will use both styles
- × AI assistant can suggest using either style
- × Either style will work



Promotion Uppercase Stream Using String Serde



Promotion Uppercase Stream Using JSON Serde



Promotion Uppercase Stream With AI Assistant



Custom JSON Serde



Kafka Message Format

- × Custom data format : comma separated, tab separated, ||| separated, etc
- × Take message as string & convert
- × Custom serde
- × Serializer + Deserializer + implements Serde<T>
- × Example : Custom JSON serde



Segment



Topic

Partition 0

Segment 0
Offset 0-498

Segment 1
Offset 499-1026

Segment 2
Offset 1026-1792

Segment 3 (active)
Offset 1793...

Partition n

Segment 0
Offset 0-510

Segment 1
Offset 511-992

Segment 2 (active)
Offset 993...

- Data writes happen on the active segment
- Each partition only has one active segment

Segment Configuration

- ✗ **log.segment.bytes**
 - ✗ Maximum size of single segment (in bytes)
 - ✗ Default 1 GB
 - ✗ Closed when exceed 1 GB, new segment created
- ✗ **log.segment.ms**
 - ✗ How long before a segment is closed
 - ✗ Default 1 week
 - ✗ Closed when exceed one week (albeit size is less than configured segment bytes), new segment created



Segment

- × Kafka log is series of segments
- × Why care?
- × Material for the next lesson about Log Cleanup

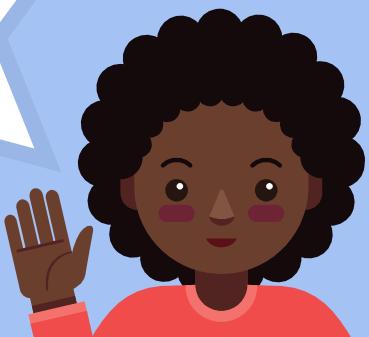


Log Cleanup



Log

- × Append-only data sequence stored as segments
- × All published data stored in Kafka log
- × Retain messages (records) for certain period
- × Stored at disk, size can grow
- × Immutable
- × High throughput & fault tolerance



Log Cleanup

- × Managing data retention & size
- × Removing outdated / unnecessary data on predefined policies
- × Policies
 - × *delete* (removes data)
 - × *compact* (retain only latest value per key)
- × The importance of log cleanup
- × Maintain health & performance of Kafka clusters



Log Cleanup Policy : *delete*



Policy : *delete*

- × Default policy for user topics
- × Data will be deleted based on log retention period
 - × Default retention is 168 hours (1 week)
- × Data can also be deleted by the maximum segment size
 - × Default maximum size is -1 (infinite)



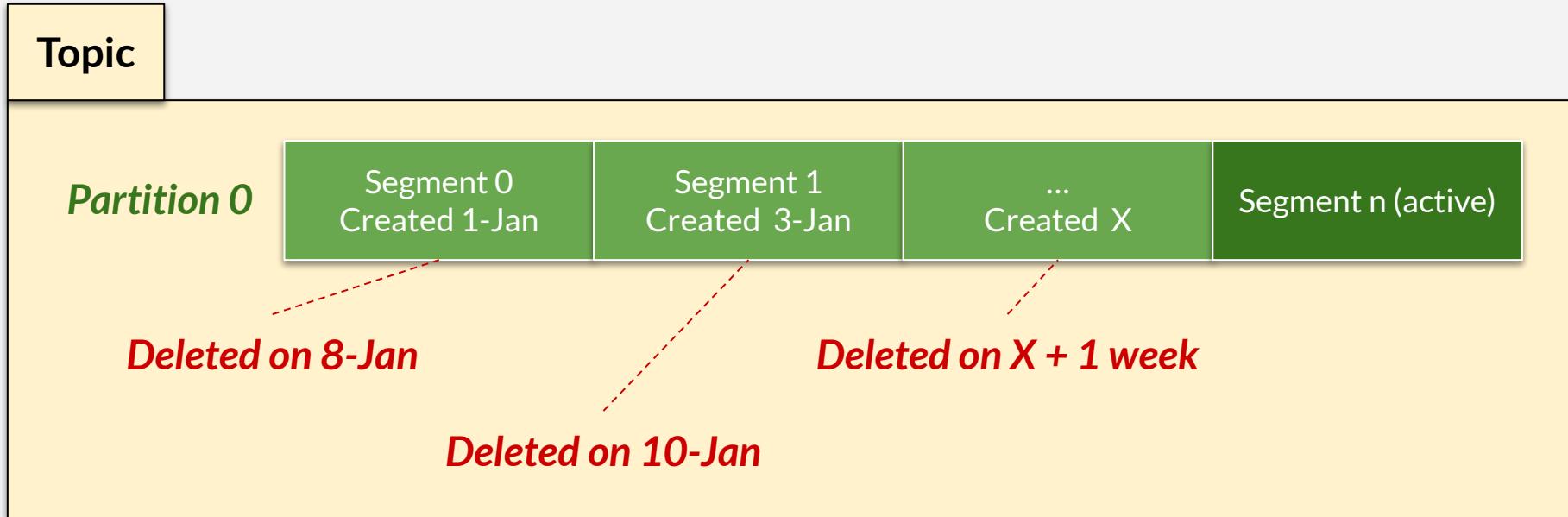
Policy : *delete*

- × Configurable
- × **log.retention.hours**
 - × How long to keep the data
 - × Default is 168 hours (1 week)
 - × Impact on adjusting the value
 - × **log.retention.minutes** or
log.retention.ms also available
- × **log.retention.bytes**
 - × Maximum size (in bytes) for each topic
 - × Default is -1 (infinite)
 - × Adjust to limit topic log's size

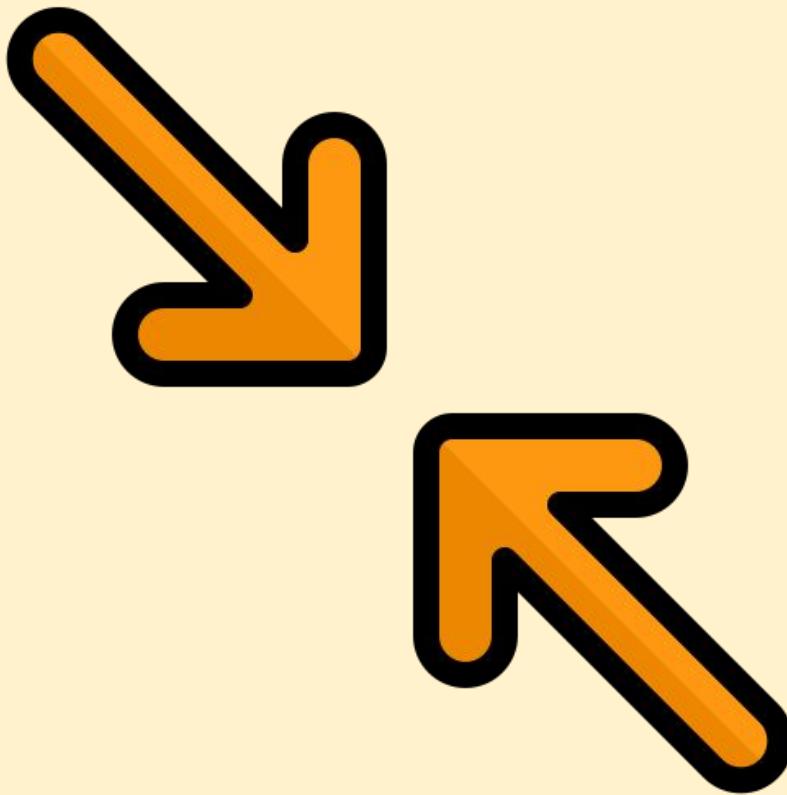


Log Deletion

168 hours retention period & infinite retention size



Log Cleanup Policy : *compact*



Policy : *compact*

- × Retain latest value of unique key within partition
- × Default policy for Kafka internal topic
- × When to use
- × Basic working process
- × Discard redundant records and maintain compact data representation



Log Compaction

Topic : T, partition 0

Offset	0	1	2	3	4	5	6	7	8	9	10
Key	Alpha	Sigma	Beta	Alpha	Omega	Alpha	Delta	Delta	Beta	Omega	Omega
Value	10	180	20	11	240	12	40	40	21	241	242

Log compaction

Topic : T, partition 0

Offset	1	5	7	8	10
Key	Sigma	Alpha	Delta	Beta	Omega
Value	180	12	40	21	242

Policy : *compact*

- × Keep the order
- × Not change offset
- × Not duplication validator
- × Can fail



Policy : *compact*

- × Configurable
- × **segment.ms**
 - × Maximum time to wait to close active segment
 - × Default is 1 week
- × **segment.bytes**
 - × Maximum time to wait to close active segment
 - × Default is 1 Gigabyte
- × **min.compaction.lag.ms**
 - × Default is 0
 - × How long to wait before a message can be compacted



Policy : *compact*

- ✗ **delete.retention.ms**
 - ✗ Default is 24 hours
 - ✗ How long to wait before deleting data marked for compaction
- ✗ **min.cleanable.dirty.ratio**
 - ✗ Default is 0.5
 - ✗ Higher value means less frequent cleaning
 - ✗ Lower value means more frequent cleaning



Policy : *compact*

- × Compaction frequency affected by traffic, **segment.ms**, and **segment.bytes**
- × Log compaction background process will open segment (file)
- × Can cause error "too many open files" if too many segments to be compacted



Settings

segment.ms = 3600000 (one hour)

segment.bytes = 1000000000 (roughly one Gigabyte)

	<i>Case 1 (high throughput)</i>	<i>Case 2 (low throughput)</i>
[A] Throughput	1 million messages / second	10k messages / second
[B] Message size	100 bytes / message	100 bytes / message
[C] Total byte per second = [A] * [B]	100 megabytes (0.1 GB) / second	1 megabyte (0.001 GB) / second
[D] Time to produce 1 GB data = 1 GB / [C]	10 seconds	1000 seconds
Segment per hour = 3600 / [D]	360 segments / hour	3.6 segments / hour

Policy : *compact*

- × Compaction in case 1 (high throughput) happen more frequently compared to case 2 (low throughput)
- × Affect log compaction frequency
- × Might need configuration adjustment



Stream & Table



Stream & Table

KStream

Ordered sequence of messages

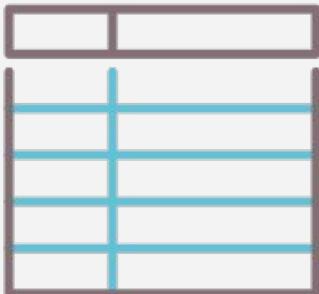
Unbounded

Inserts data



Stream

Table



KTable

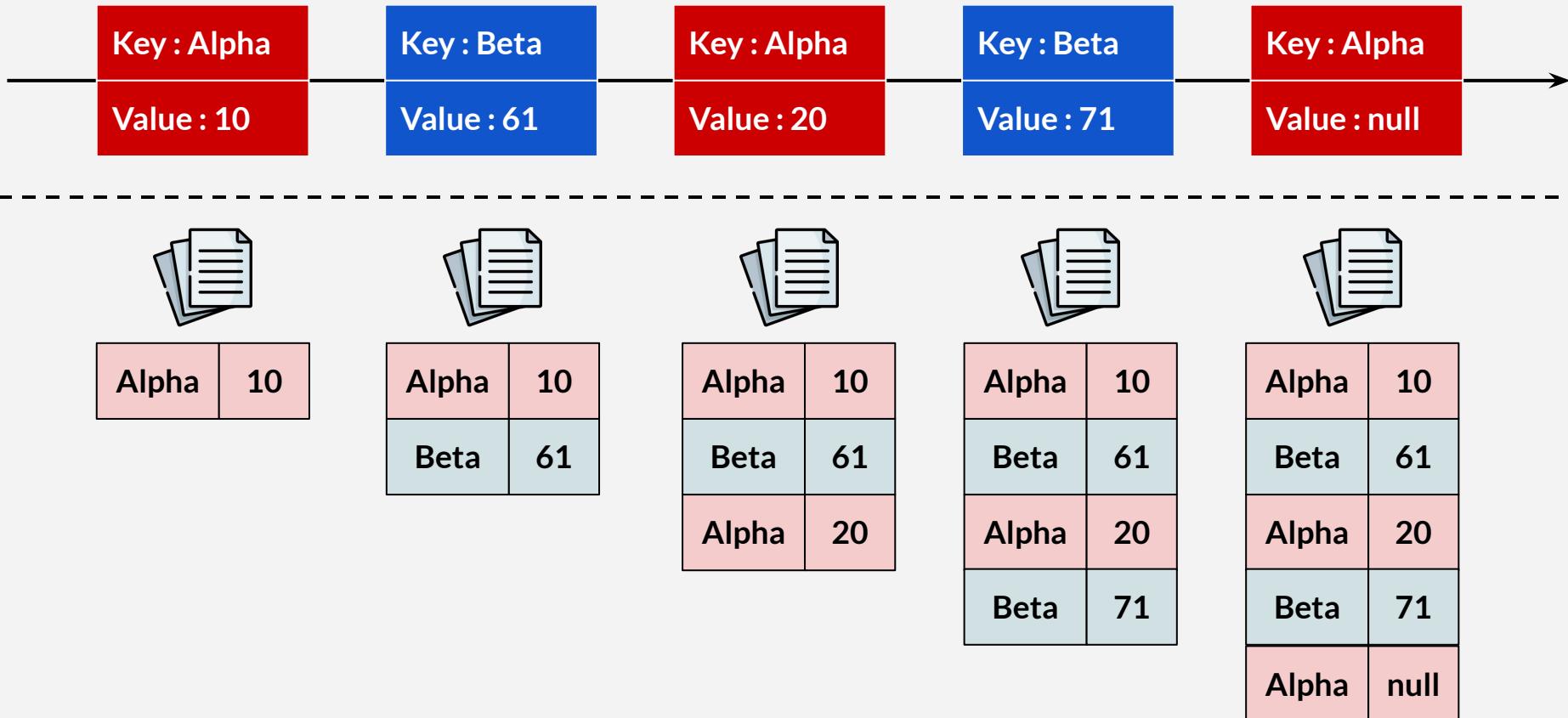
Unbounded

Upserts data : insert or update based on key

Delete on null value

Analogy : database table

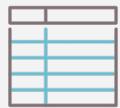
KStream : Inserts Data



KTable : Upserts / Delete Data



Insert Alpha



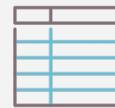
Alpha	10
-------	----

Insert Beta



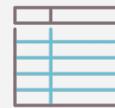
Alpha	10
Beta	61

Update Alpha



Alpha	20
Beta	61

Update Beta



Alpha	20
Beta	71

Delete Alpha



Beta	71
------	----

When to Use KStream / KTable

- × KStream
 - × Topic not log-compacted
 - × Data is partial information
- × KTable
 - × Topic is log-compacted
 - × Data is self sufficient



Available Operations Theory

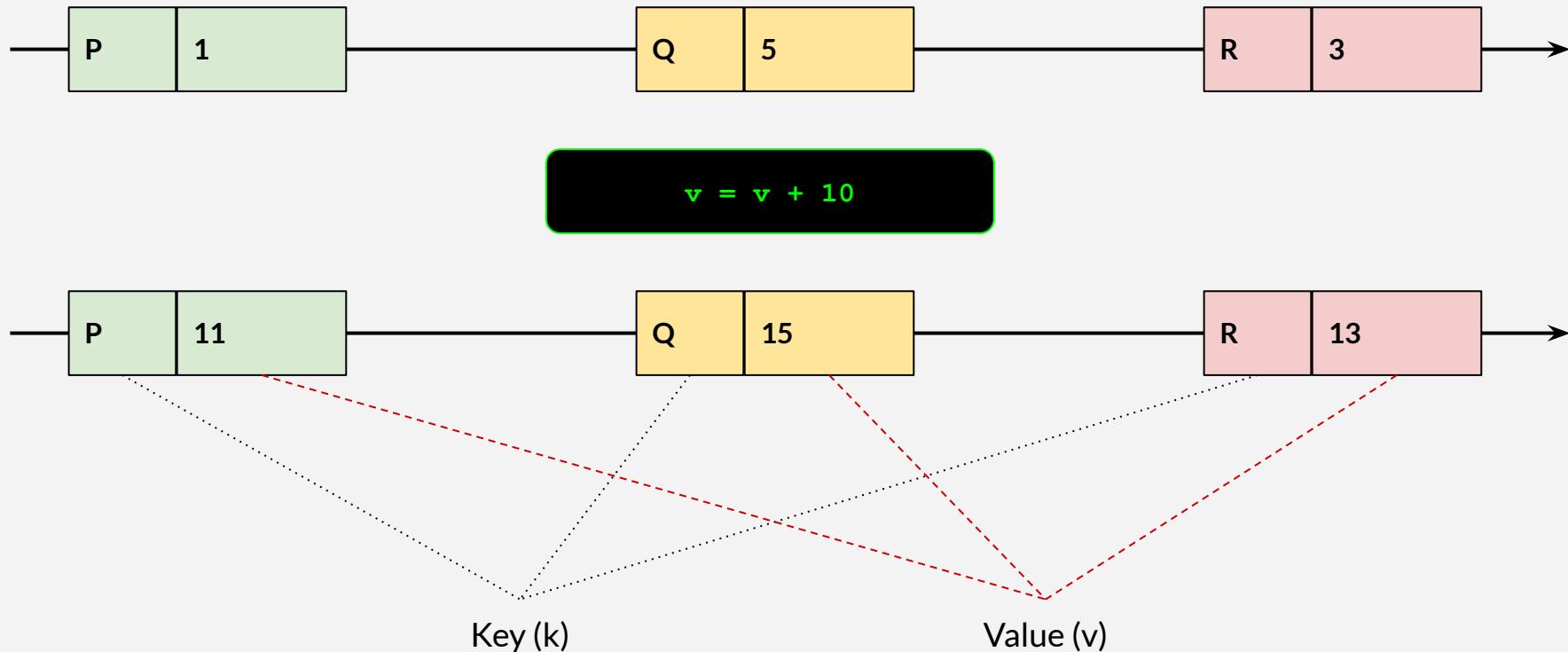


About This Lesson

- ✗ Kafka streams operations
- ✗ This lesson, then code
- ✗ Code, then this lesson



Diagram



Intermediate & Terminal Operation

- × Intermediate
 - × KStream -> KStream
 - × KTable -> Ktable
- × Terminal
 - × KStream -> void
 - × KTable -> void
 - × “Final” operation



Stateless & Stateful

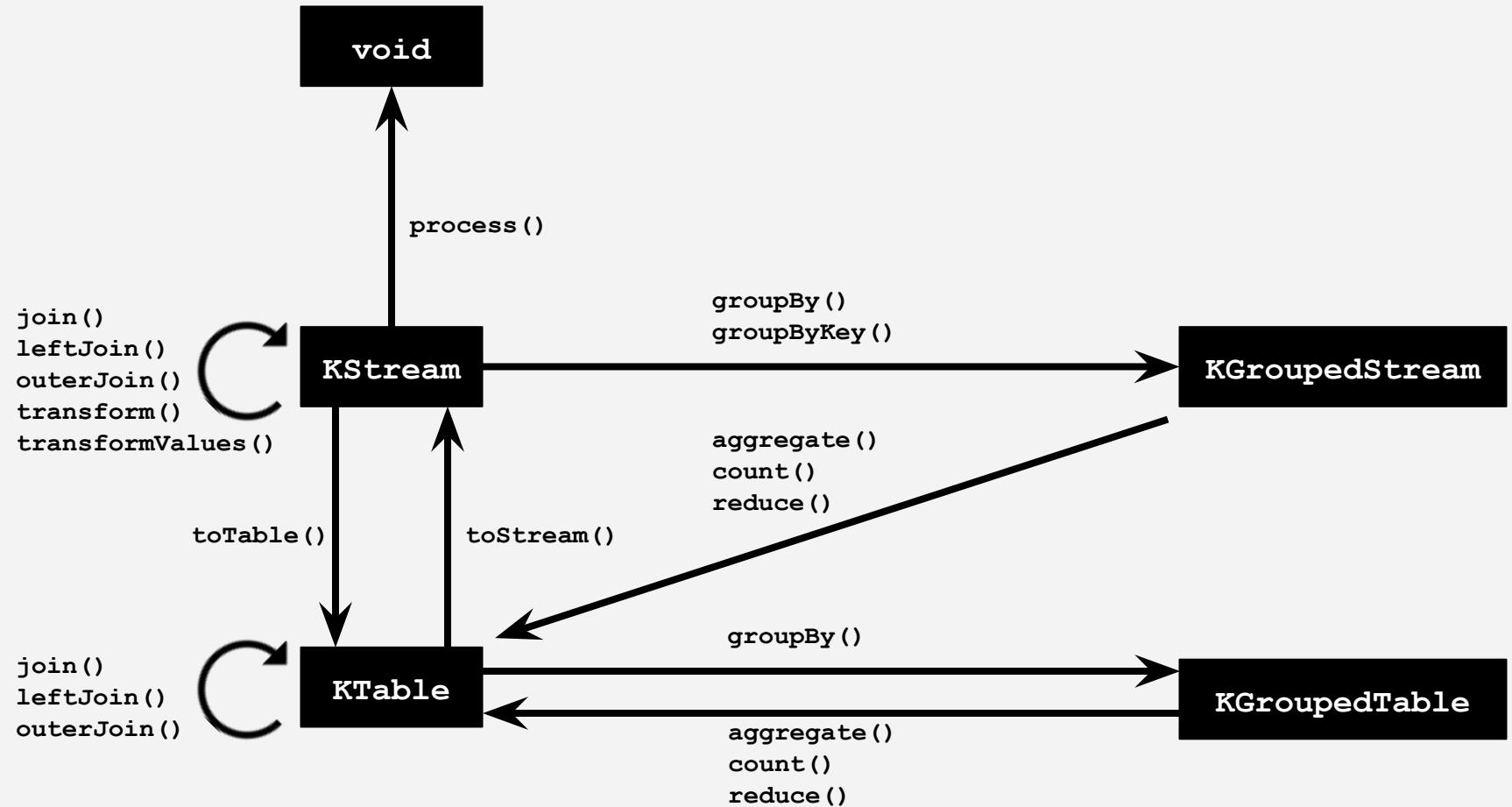
- × Stateless : process each record independently
 - × Ideal use cases?
- × Stateful : maintains state across multiple records
 - × Ideal use cases?
- × Summary



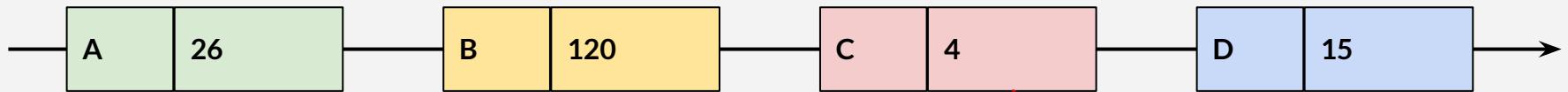
Reminder : Key & Partition

- × Key & partition is related
- × Partition according to key
- × Repartition
 - × From partition A to partition X



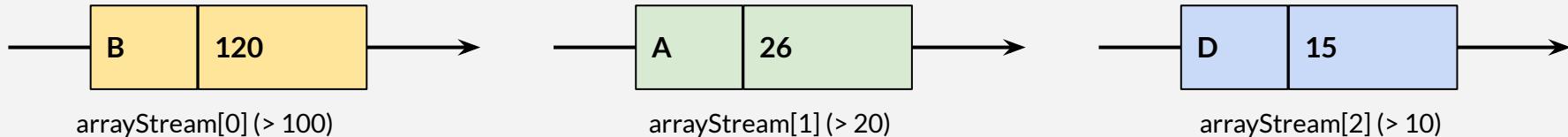


branch



```
var arrayStream = stream.branch(  
    (k, v) -> v > 100,  
    (k, v) -> v > 20,  
    (k, v) -> v > 10  
)
```

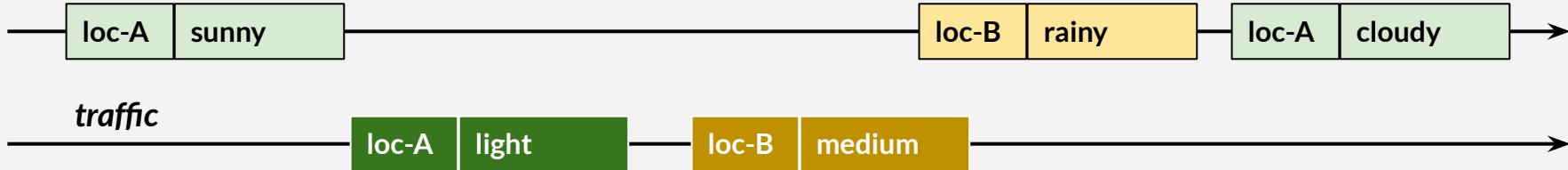
Dropped, no match



- Split stream based on predicates
- Evaluate predicate in order
- Record only placed once on first match, drop unmatched record
- Returns array of stream
- Intermediate operation
- KStream

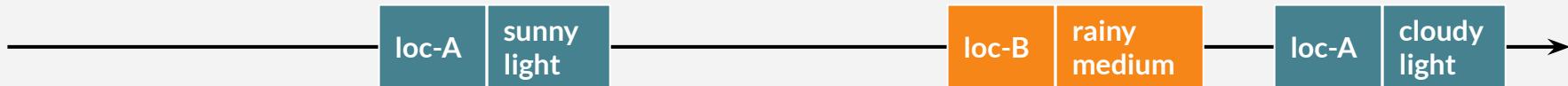
weather

cogroup



```
var groupedWeather = weatherStream.groupByKey();
var groupedTraffic = trafficStream.groupByKey();

var locationsCogroup = groupedWeather.cogroup(WEATHER_AGGREGATOR)
    .cogroup(groupedTraffic, TRAFFIC_AGGREGATOR)
    .aggregate(() -> new Location(), Materialized.with(stringSerde, jsonSerde));
```



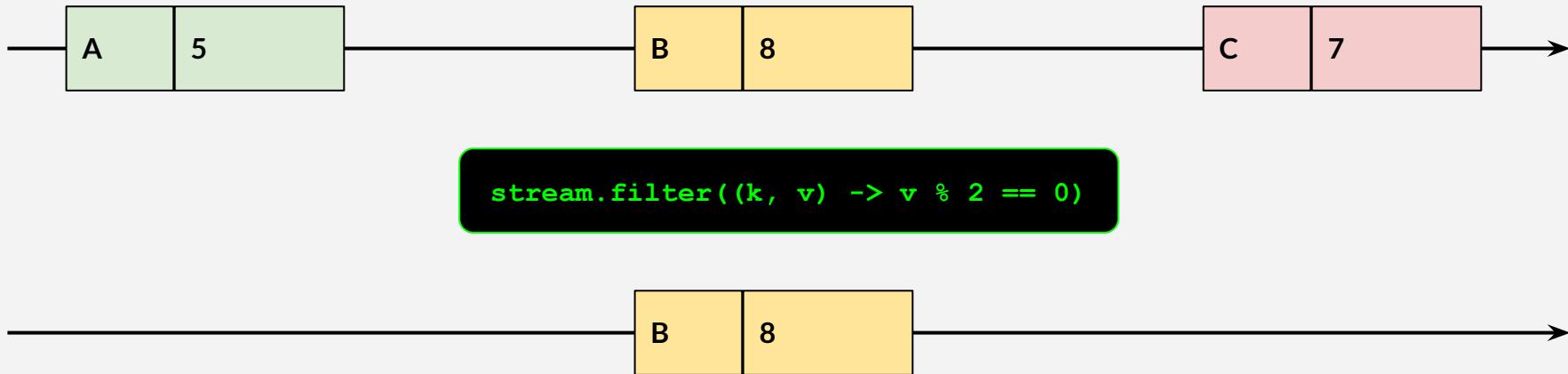
- Intermediate operation
- KStream
- Need aggregator for each cogroup
- This sample : String key, JSON value
- Difference with merge()

cogroup - aggregator

```
Aggregator<String, String, Location> WEATHER_AGGREGATOR = new Aggregator<String, String, Location>() {  
  
    @Override  
    public Location apply(String key, String value, Location aggregate) {  
        aggregate.setWeather(value);  
        return aggregate;  
    }  
};
```

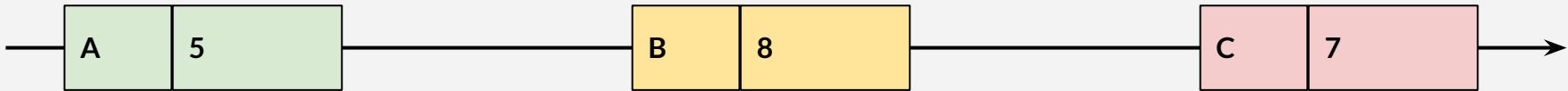
```
Aggregator<String, String, Location> TRAFFIC_AGGREGATOR = new Aggregator<String, String, Location>() {  
  
    @Override  
    public Location apply(String key, String value, Location aggregate) {  
        aggregate.setTraffic(value);  
        return aggregate;  
    }  
};
```

filter

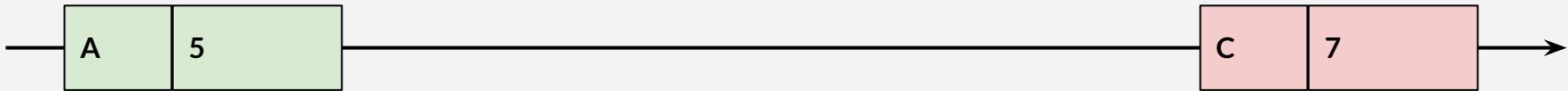


- Takes one record, produces one or zero record
- Produce record that match condition
- Does not change key or value
- Not trigger repartition
- Intermediate operation
- KStream & Ktable

filterNot



```
stream.filterNot((k, v) -> v % 2 == 0)
```



- Takes one record, produces one or zero record
- Produce record that NOT match condition
- Does not change key or value
- Not trigger repartition
- Intermediate operation
- KStream & KTable

flatMap

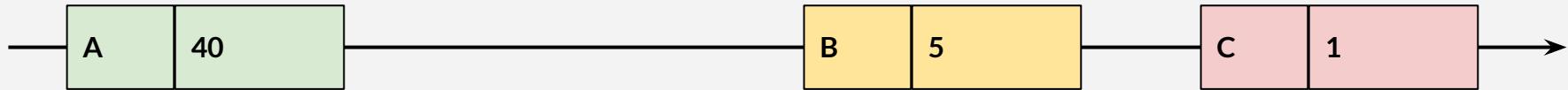


```
stream.flatMap(listPrimeFactorsAndAppendKey())
```

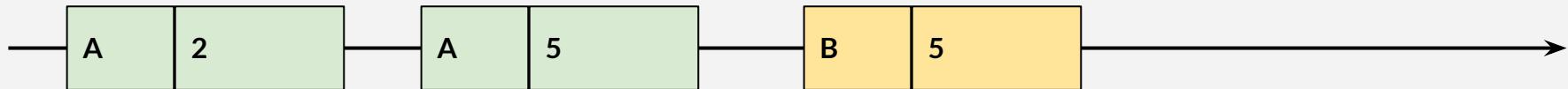


- Takes one record, produces zero or more record
- Change key
- Change value
- Trigger repartition
- Intermediate operation
- KStream

flatMapValues

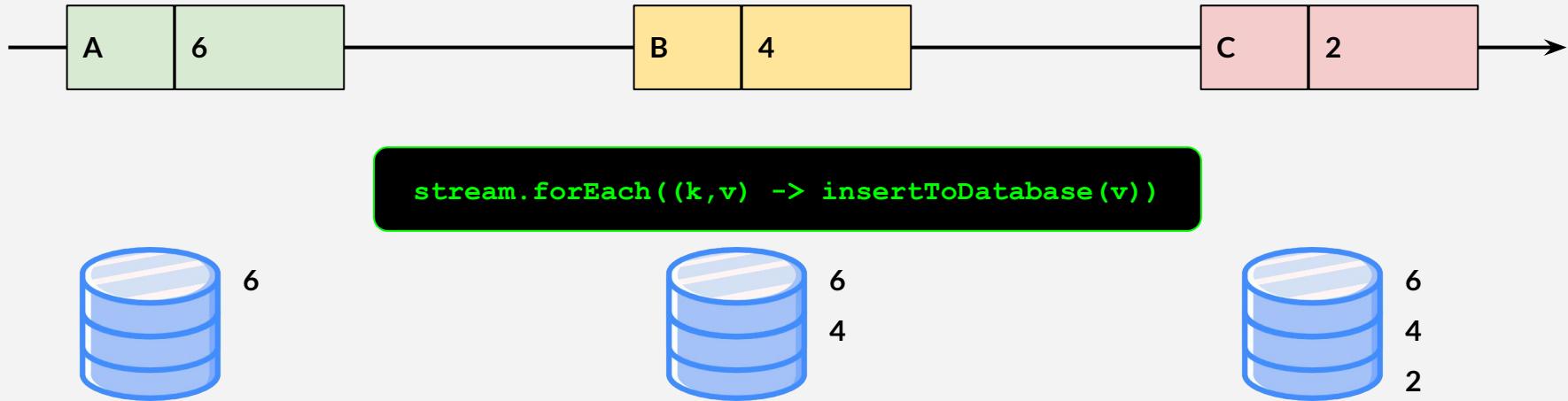


```
stream.flatMapValues(listPrimeFactors())
```



- Takes one record, produces zero or more record
- Does not change key
- Affect only value
- Not trigger repartition
- Intermediate operation
- KStream

foreach

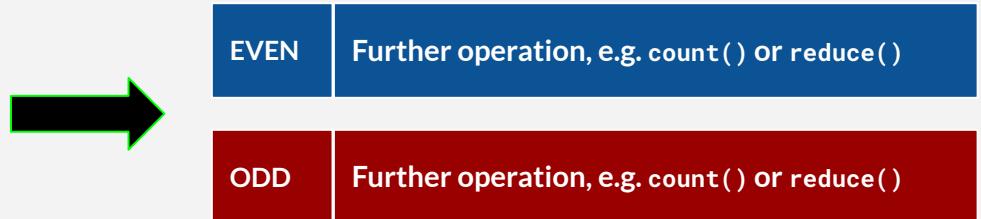


- Terminal operation
- Takes one record, produces none
- Produces side effect
- Side effect not tracked by kafka
- KStream & KTable

groupBy

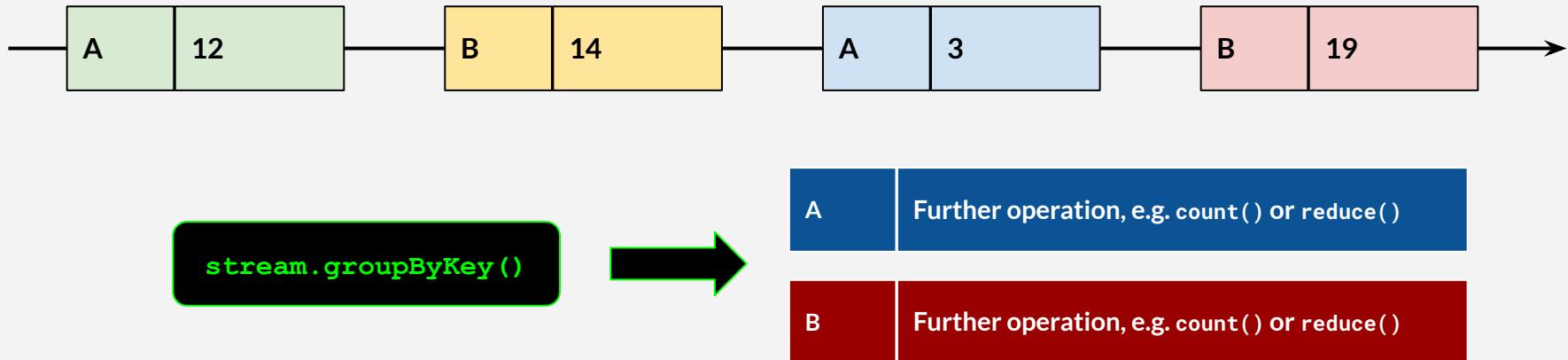


```
stream.groupBy((k, v) -> v % 2 == 0?  
    "EVEN" : "ODD")
```



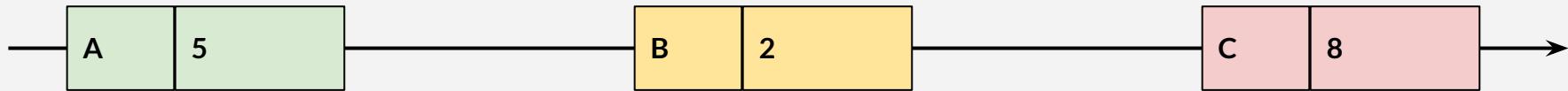
- Intermediate operation
- Group records by new key
- Process further
- KStream & KTable

groupByKey

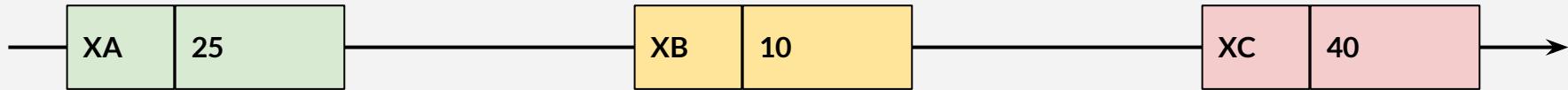


- Intermediate operation
- Group records by existing key
- KStream

map

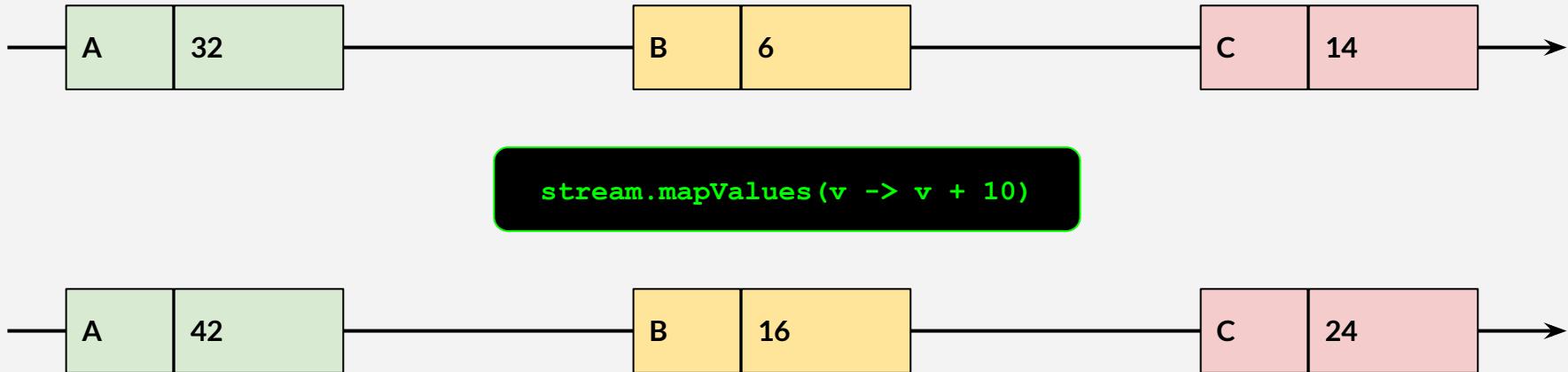


```
stream.map( (k, v) -> KeyValue.pair("X" + k, v * 5) )
```



- Takes one record, produces one record
- Change key
- Change value
- Trigger repartition
- Intermediate operation
- KStream

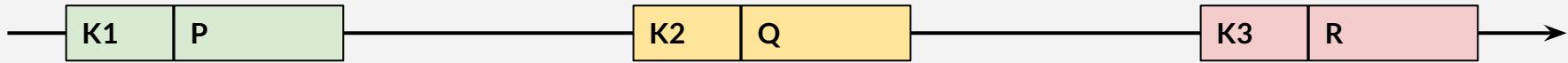
mapValues



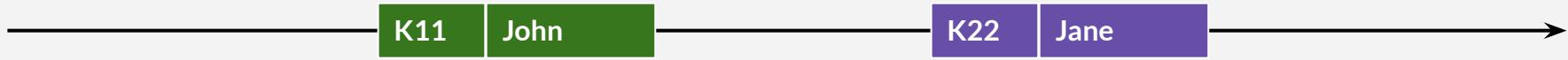
- Takes one record, produces one record
- Does not change key
- Affect only value
- Not trigger repartition
- Intermediate operation
- KStream & Ktable

merge

alphabetStream



nameStream

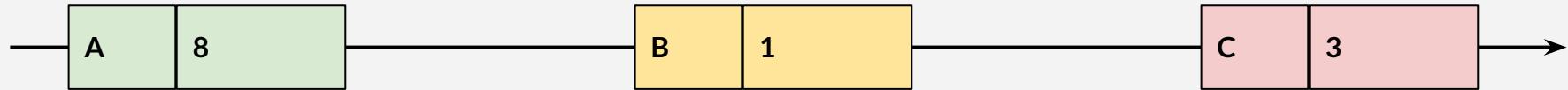


```
var mergedStream = alphabetStream.merge(nameStream)
```



- Merge two streams into one new stream
- Both stream data types must be the same
- No ordering guarantee on resulting stream
- Intermediate operation
- KStream

peek



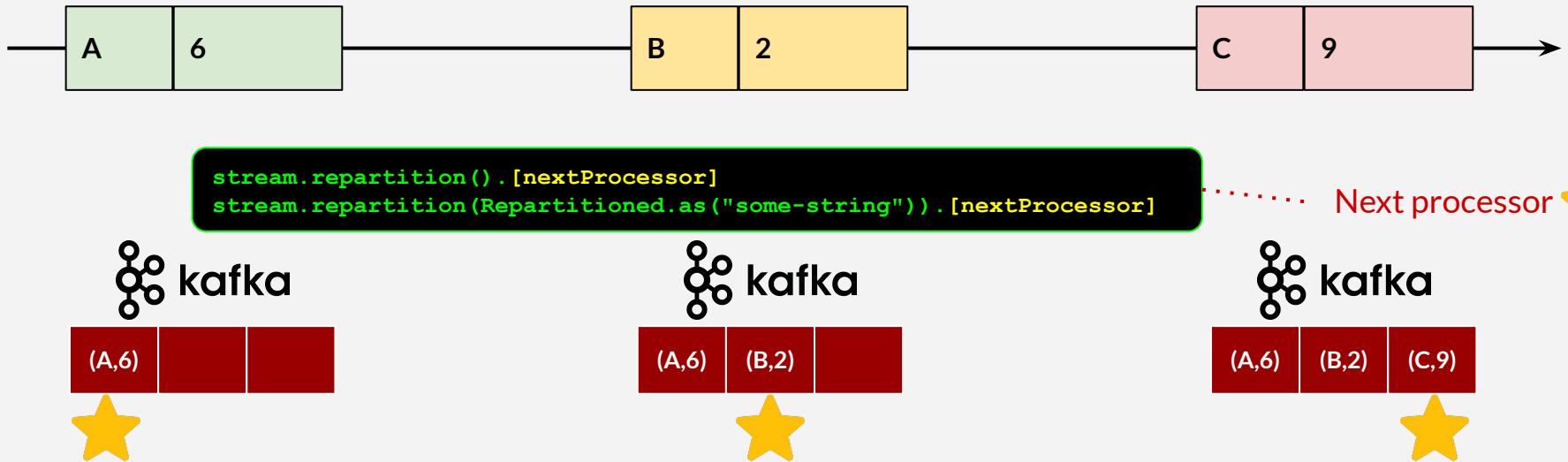
```
stream.peek( (k, v) -> insertToDatabase(v) ) .[nextProcessor]
```

..... Next processor



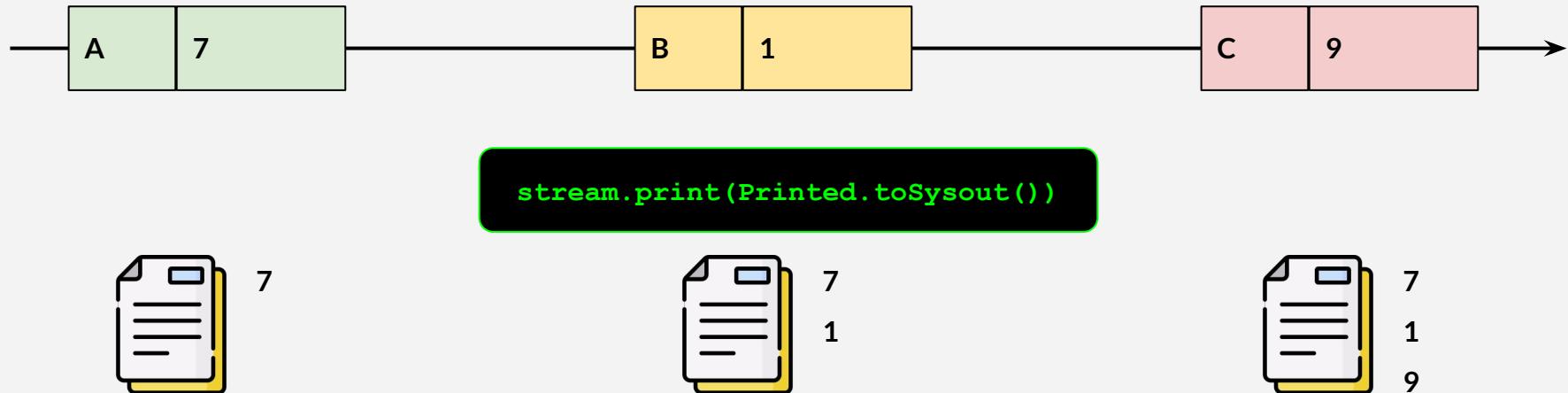
- Produces unchanged stream
- Produces side effect
- Side effect not tracked by kafka
- Result stream can be processed further
- Intermediate operation
- KStream

repartition



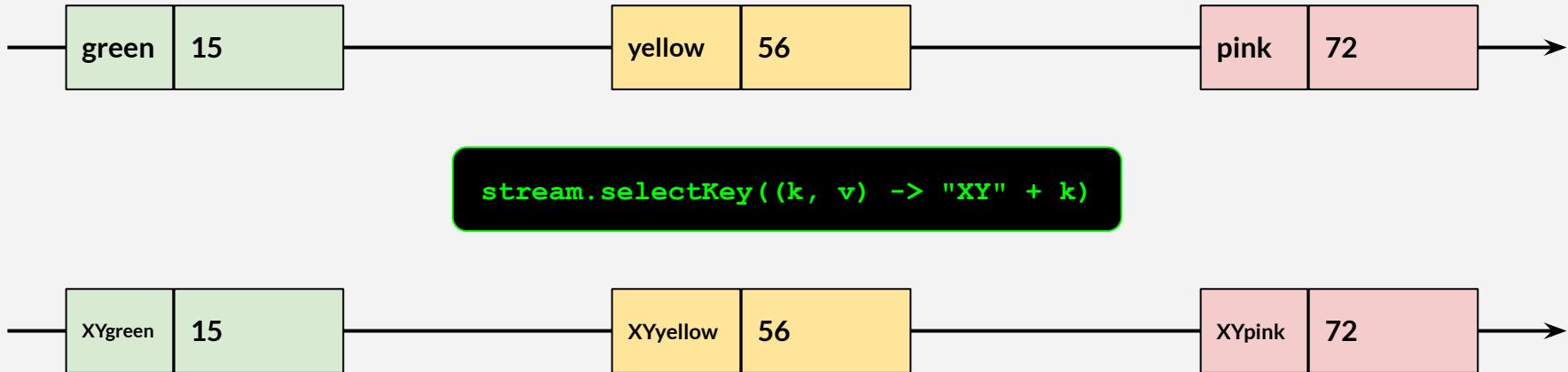
- Intermediate operation
- Write stream to destination topic
- Continue record processing
- repartition() output-topic name is fixed
- Topic only for kafka internal use
- Use through for more control on topic name (risk on using deprecated)

print



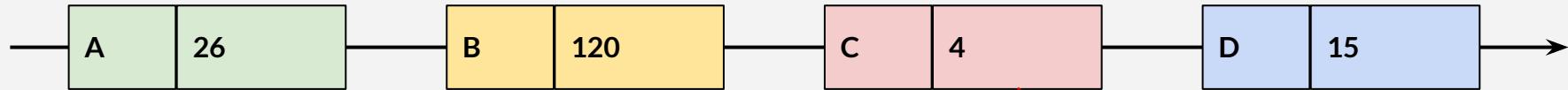
- Terminal operation
- Print each record
- Something like kafka console consumer
- Print to file or console
- KStream

selectKey



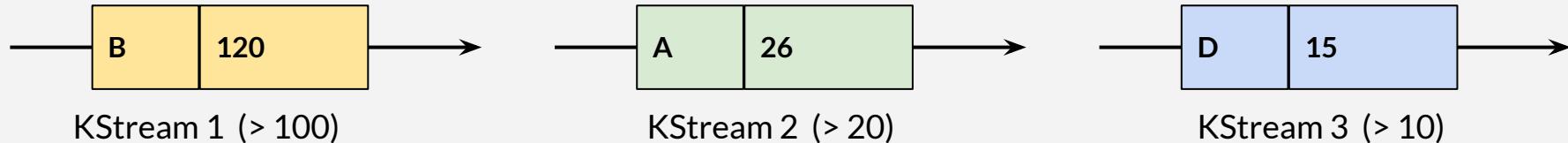
- Takes one record, produces one record
- Set / replace record key
- Possible to change key data type
- Trigger repartitioning
- Value not change
- Intermediate operation
- KStream

split



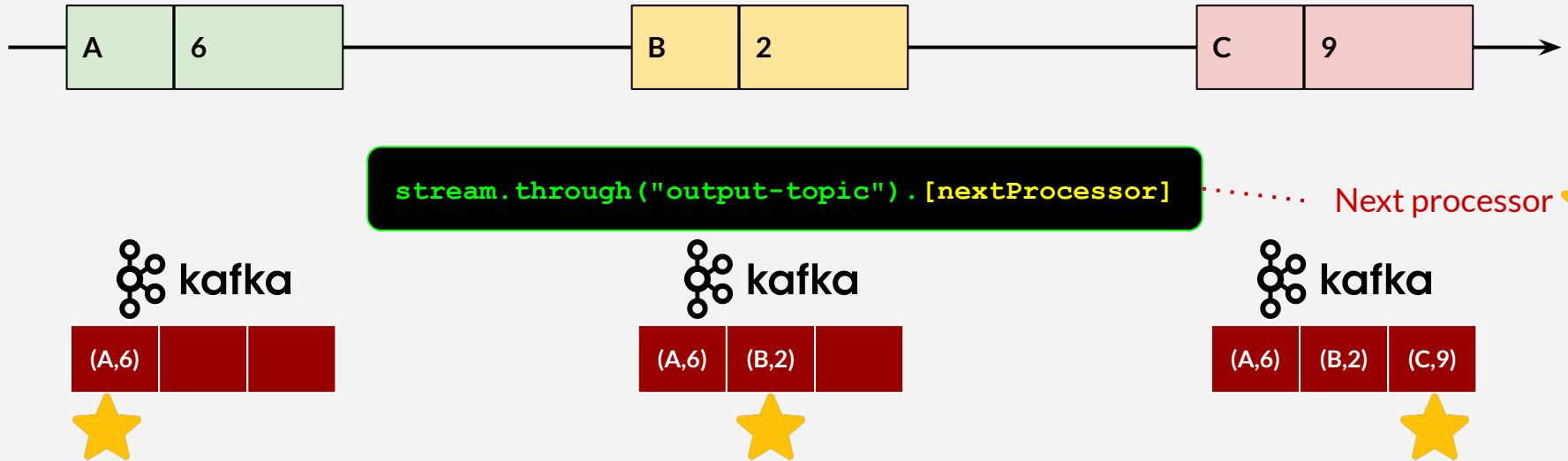
```
stream.split().  
    branch((k, v) -> v > 100, Branched.withConsumer(ks -> ks.to("t-x"))  
    .branch((k, v) -> v > 20, Branched.withConsumer(ks -> ks.to("t-y"))  
    .branch((k, v) -> v > 10, Branched.withConsumer(ks -> ks.to("t-z"))  
)
```

Dropped, no match



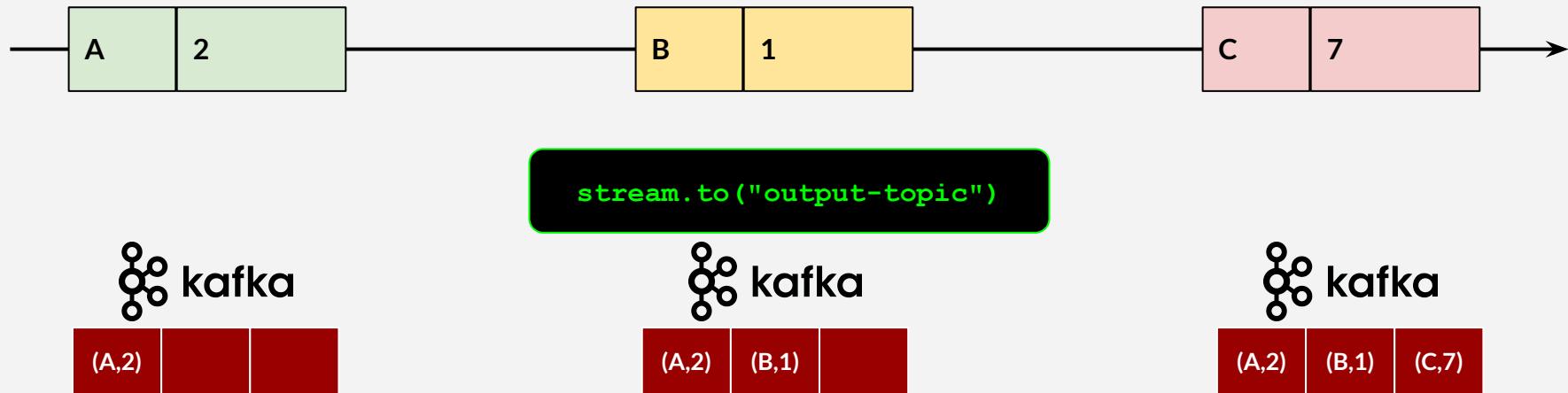
- Split stream based on predicates
- Evaluate predicate in order
- Record only placed once on first match, drop unmatched record
- Get KStream for each branch
- `split()` returns final BranchedKStream
- Each branch returns KStream to be processed further

through



- Intermediate operation
- Write stream to destination topic
- Continue record processing
- KStream

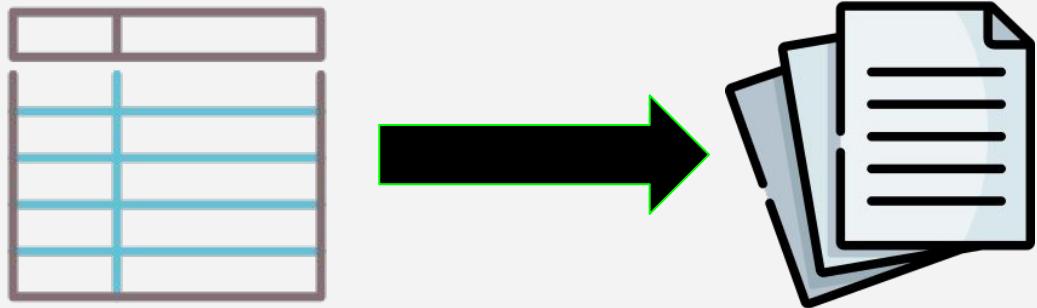
to



- Terminal operation
- Write stream to destination topic
- KStream

toStream

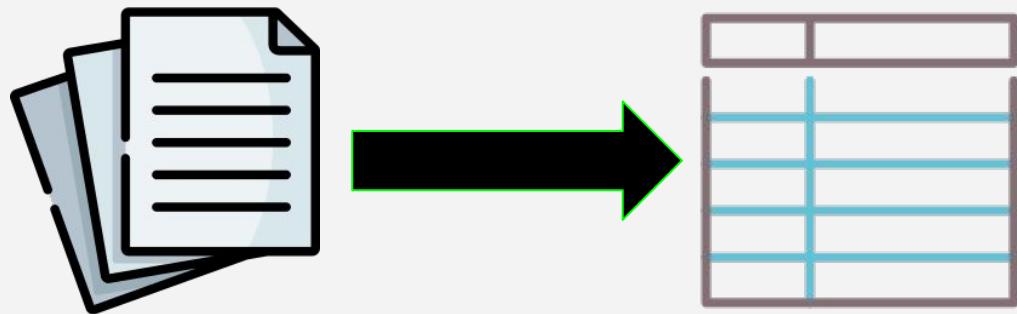
```
table.toStream()
```



- KTable
- Intermediate operation
- Convert KTable to KStream

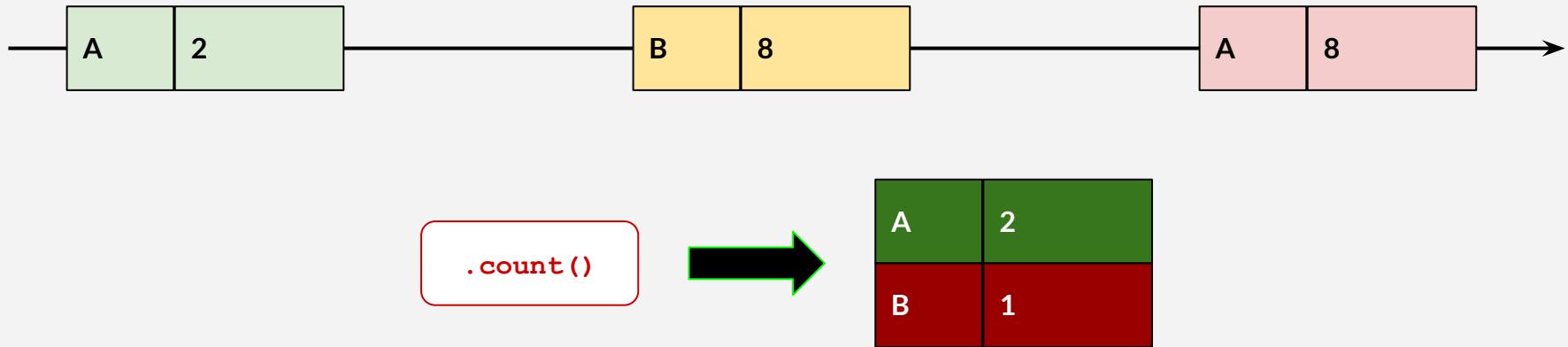
toTable

```
stream.toTable()
```



- KStream
- Intermediate operation
- Convert KStream to KTable

count



- Counts number of record based on key
- Ignore null keys or values
- KGroupedStream

aggregate

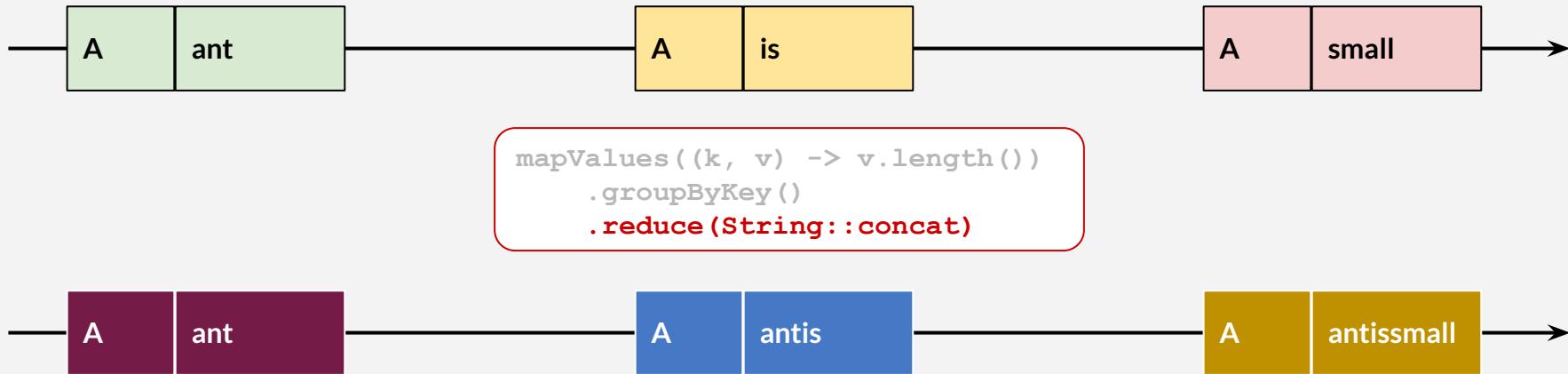


```
mapValues((k, v) -> v.length())
    .groupByKey()
    .aggregate(() -> 0, (aggKey, newValue, aggValue) -> aggValue + newValue)
```



- Aggregate record based on key
- Need initializer and adder
 - Initializer on example : () -> 0
 - Adder on example : (aggKey, newValue, aggValue) -> aggValue + newValue
- Aggregation result can be different type with input
- Ignore null keys
- On first each non-null key : call initializer, then call adder
- On non-null value : call adder
- KGroupedStream

reduce



- Specialized form / shorter syntax for aggregate
- Result and input type not change
- Need reducer
 - In example: `String.concat()`
- KGroupedStream

Available Operations Sample Code



Example Code

- × Example source code for available operations on
Resources & References
- × For simple operations (more real use case later)
 - × *broker.stream.operation.OperationDemoStream*
- × How to use the demo



Example Code

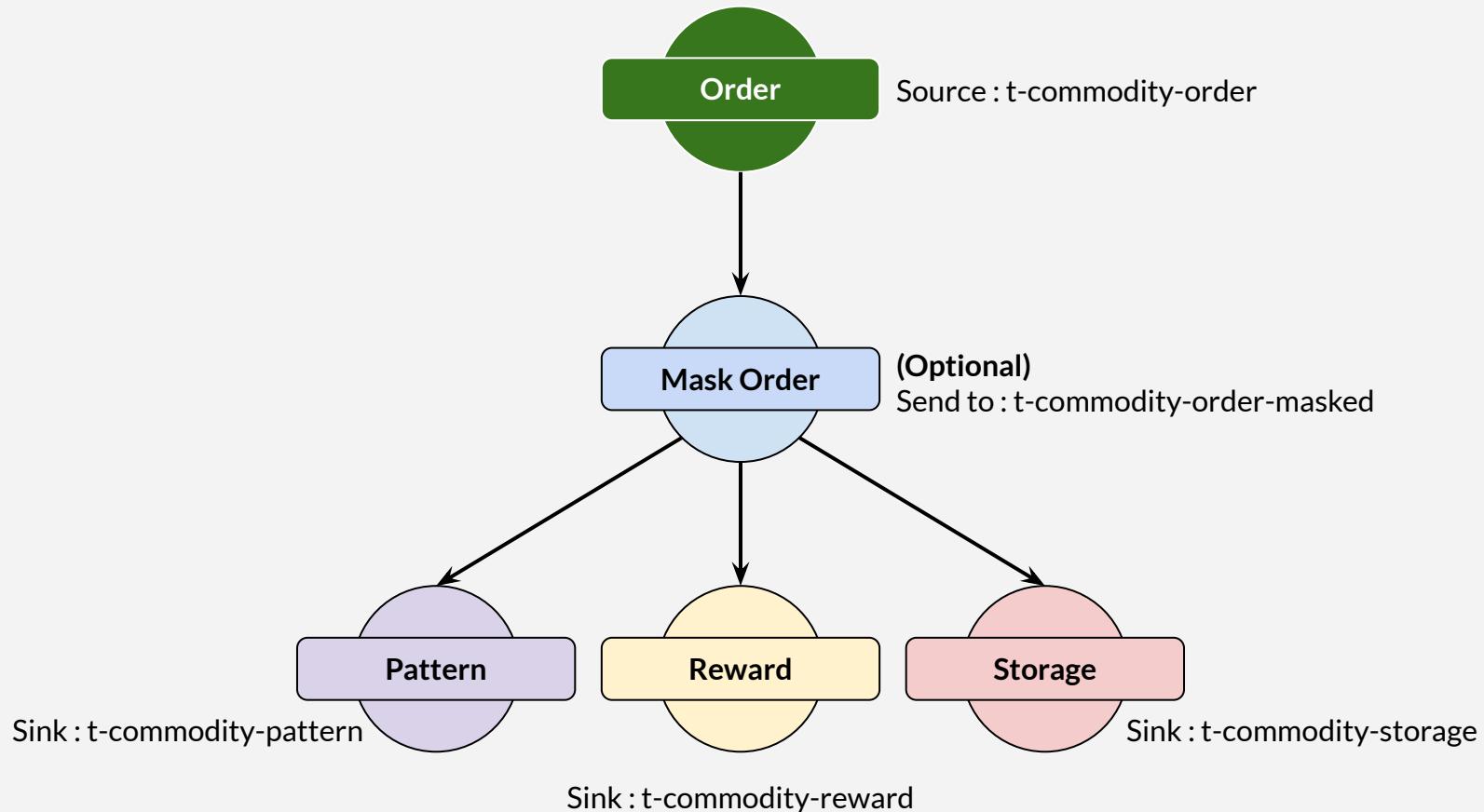
- × Examples for **KStream**
- × Concept is the same for **KTable**
- × **KTable** operations usually involve join
 - × More about this later



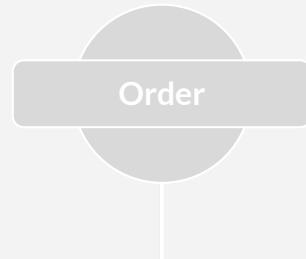
Commodity Stream Topology



Topology



Topology



Key : A123XY
Value : { JSON }

```
$> kafka-console-consumer.sh ... --property print.key=true
```

A123XY { JSON }

Key : A123XY
Value : { JSON }



Key : A123XY
Value : { JSON }

Key : A123XY	Value : { JSON }
--------------	------------------

Commodity Stream

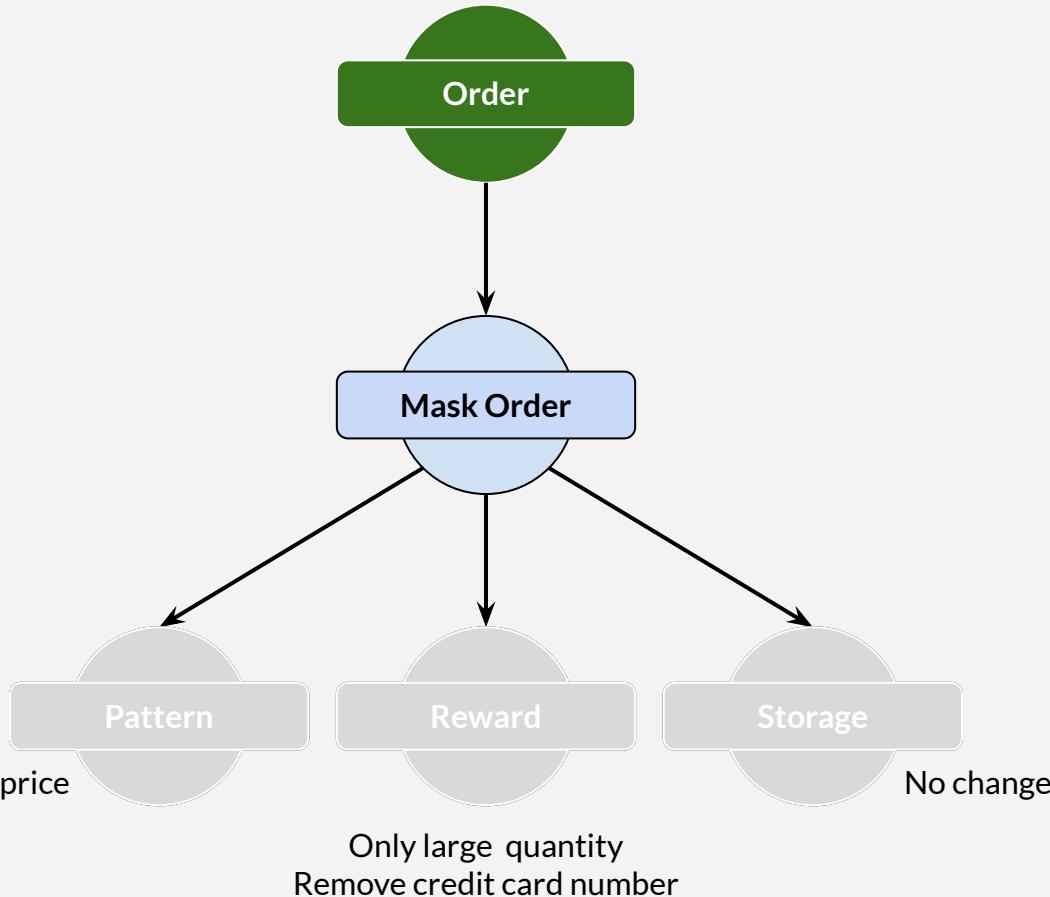
First Step



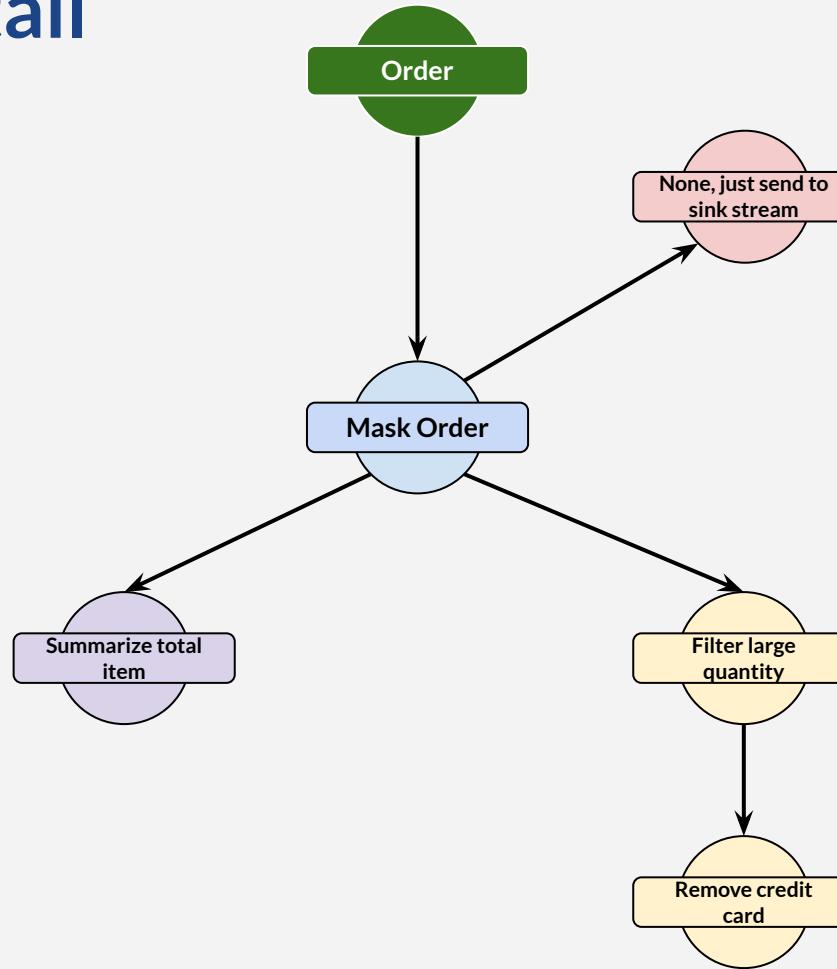
Commodity Stream Sink Processors



High Level Topology



Topology Detail

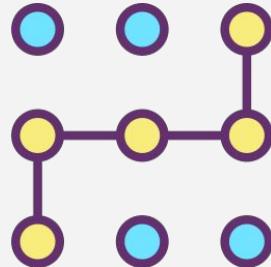


Commodity Stream

Additional Requirements



Additional Requirement



Pattern

Current : summarize item price * quantity

+ ADDITIONAL : split plastic & non plastic items



Reward

Current : give reward only for item with quantity > xxx

+ ADDITIONAL : give reward only for item that is not cheap

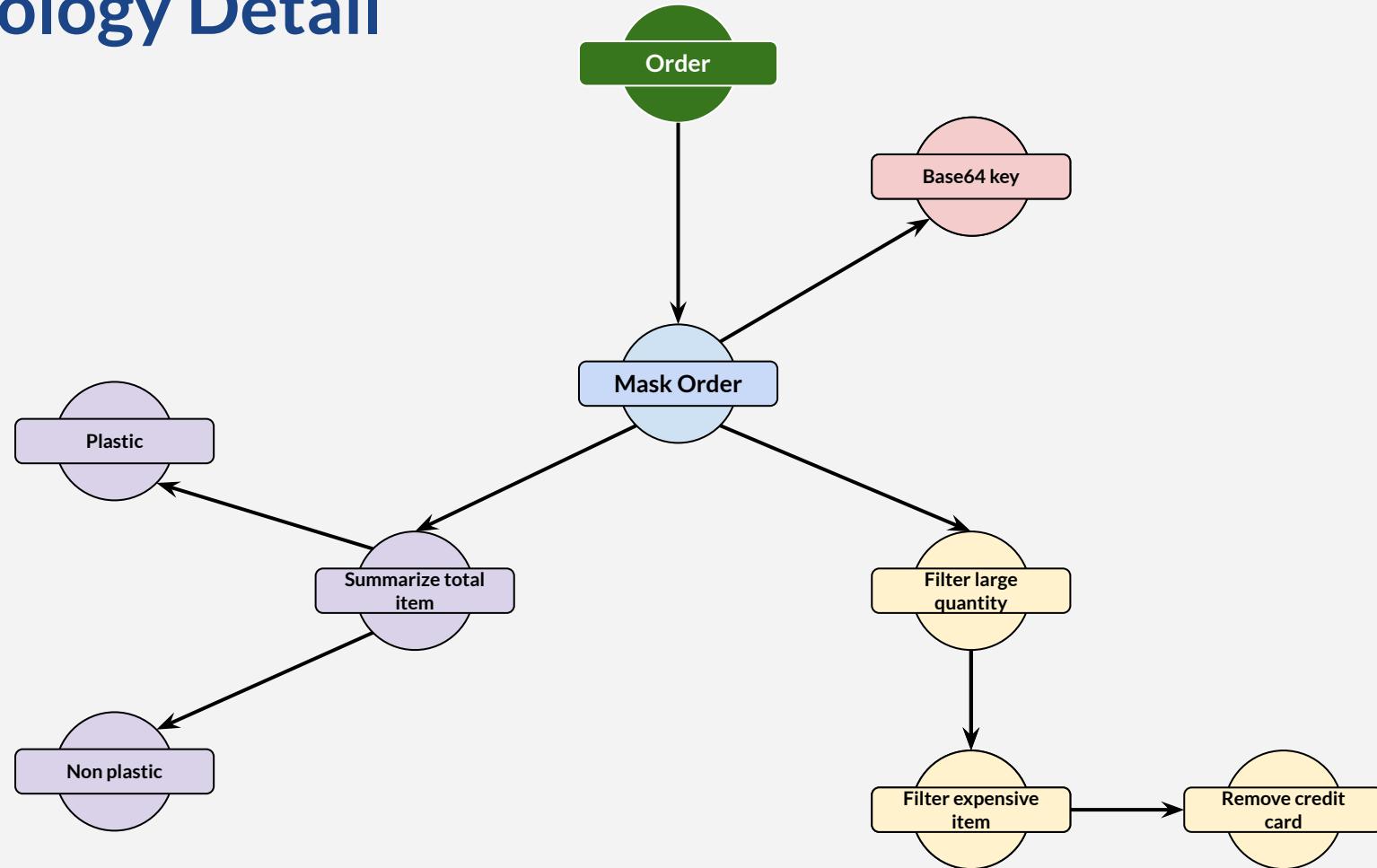


Storage

Current :-

+ ADDITIONAL : key is base64(order number)

Topology Detail



Sample Data

Cotton Dog

Price : 80
Qty : 250

Plastic Cat

Price : 400
Qty : 500

Wooden Horse

Price : 700
Qty : 90

Steel Pig

Price : 350
Qty : 270

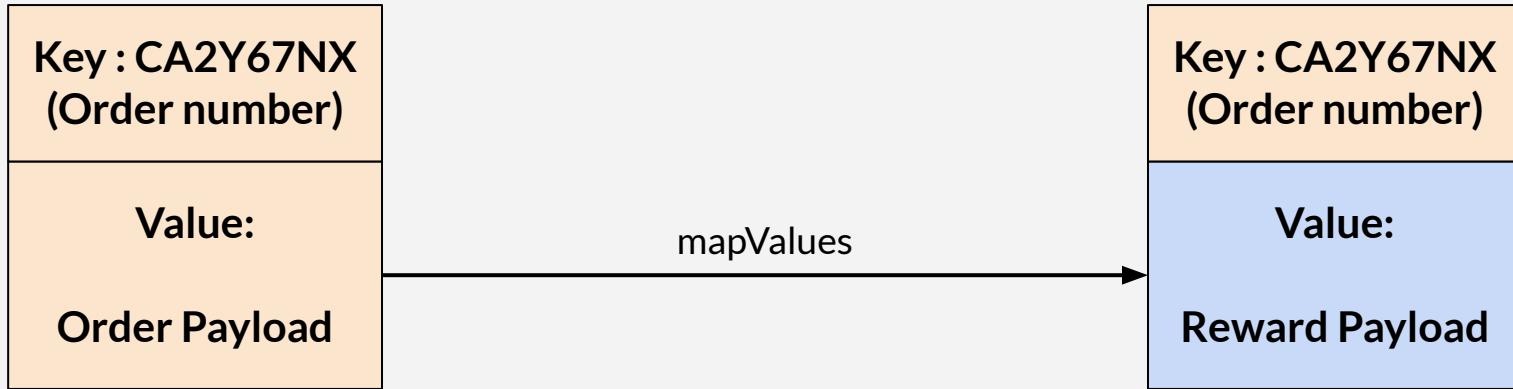
Stream (Kafka Sink Topic)	Data
Pattern - plastic	Plastic Cat
Pattern - not plastic	Cotton Dog, Wooden Horse, Steel Pig
Reward	Plastic Cat, Steel Pig
Storage	Plastic Cat, Cotton Dog, Wooden Horse, Steel Pig

Commodity Stream

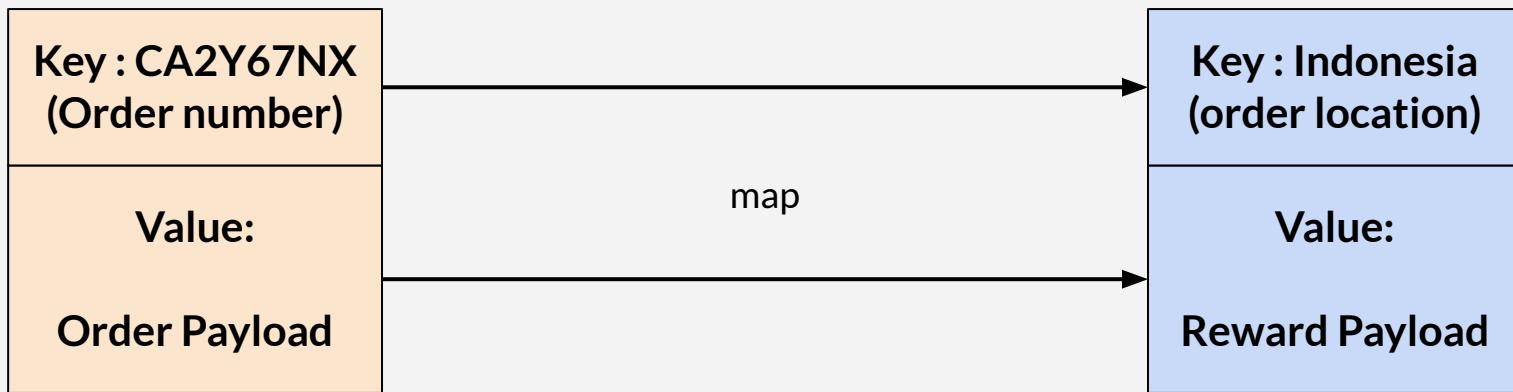
Reward Each Location



Reward Message



Reward Message

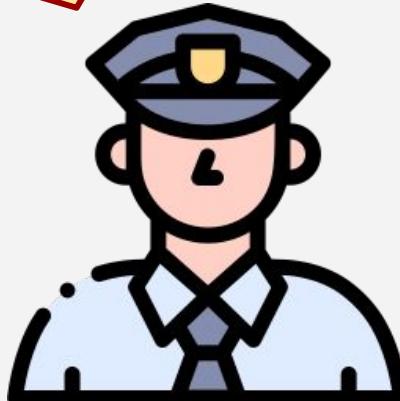


Commodity Stream

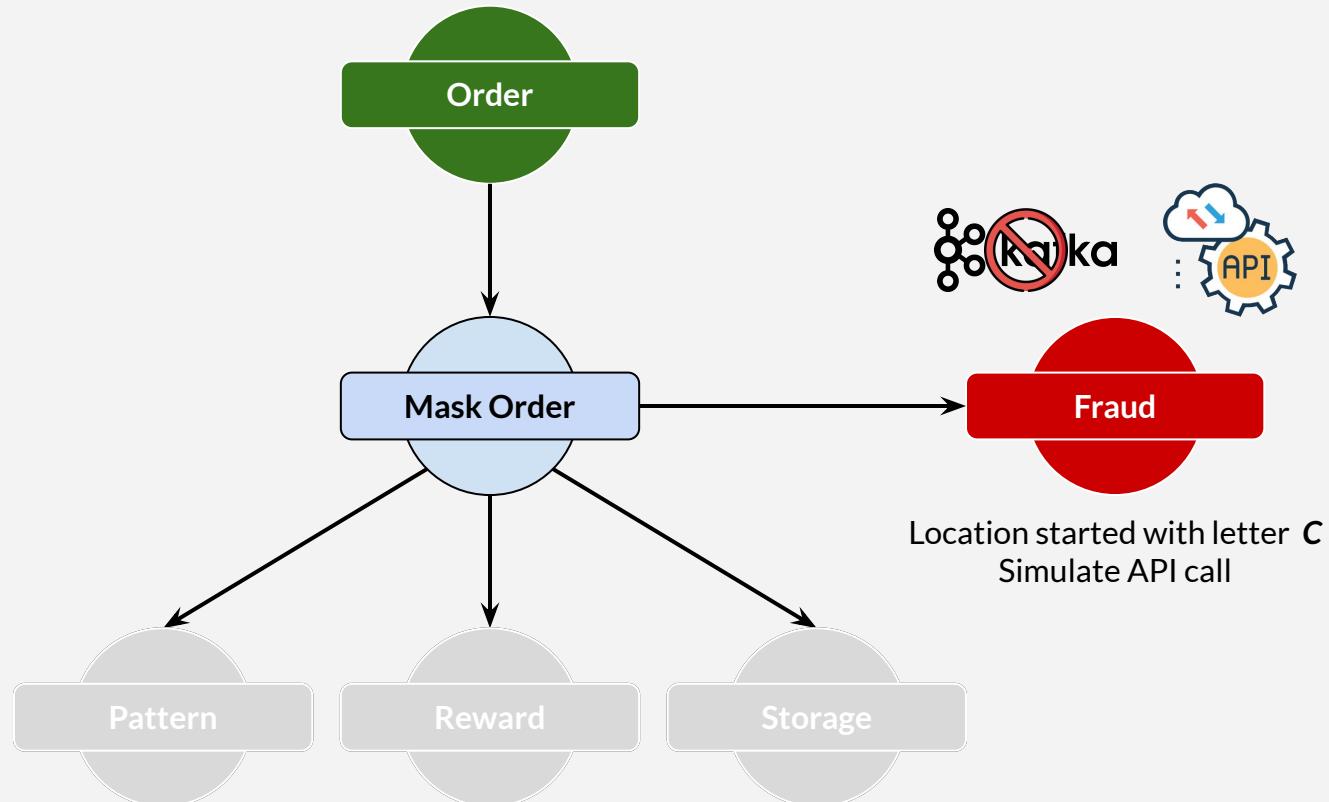
Calling API or Other Process



Something Suspicious



High Level Topology

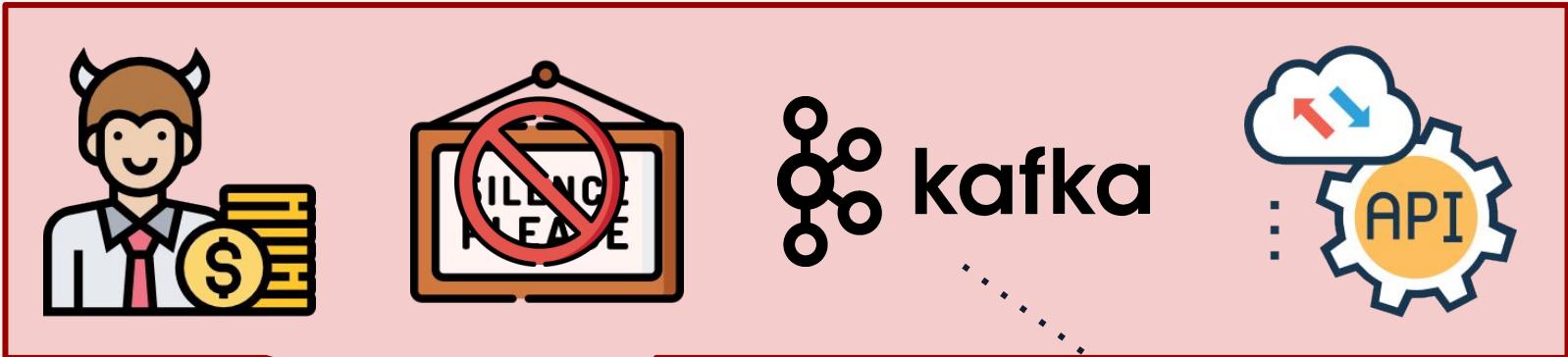


Commodity Stream

Further Fraud Processing



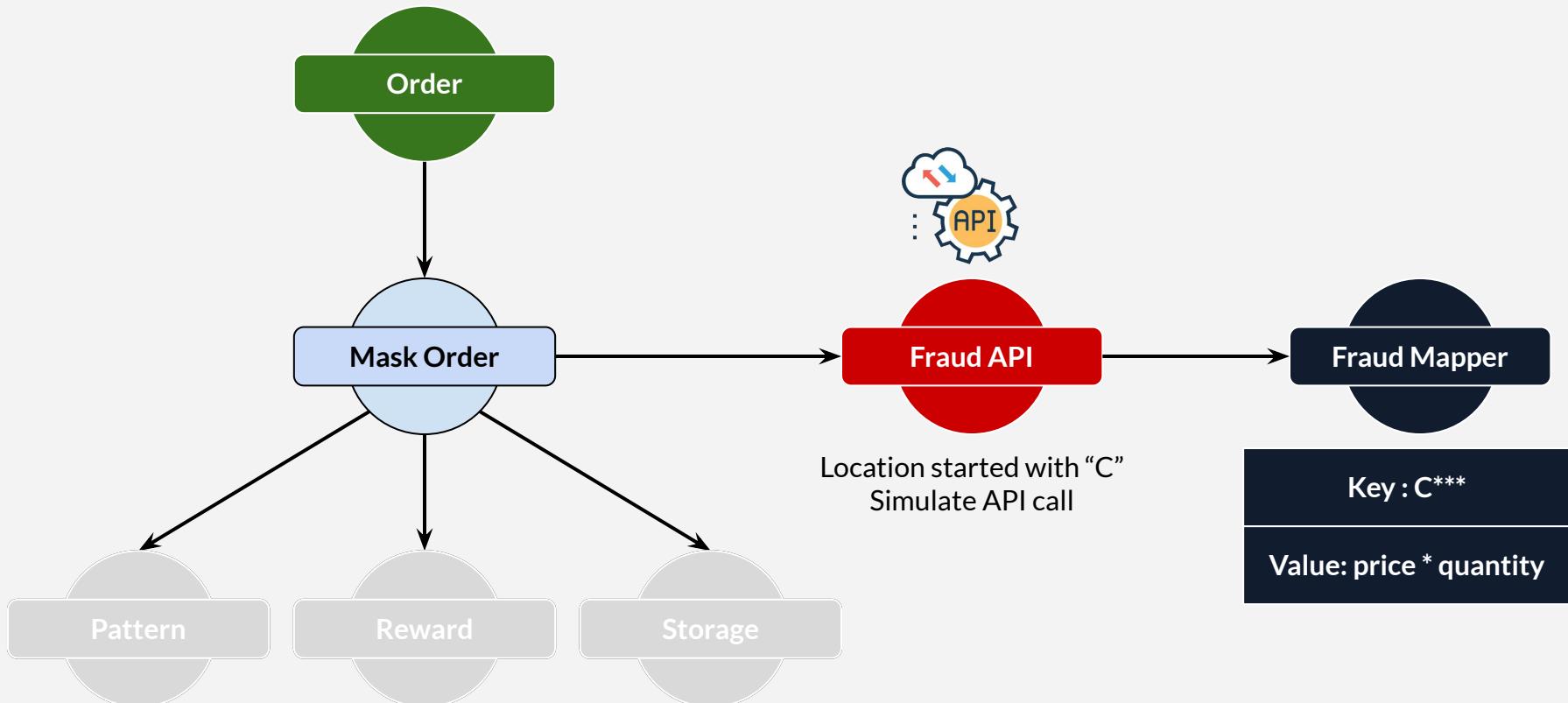
Something Suspicious



Key : C***

Value: price * quantity

High Level Topology



Where Is The AI Assistant?



AI Assistant

- × Why rarely uses chat feature?
- × Uses autocomplete extensively
- × Functionality is not very complex
 - × Detailed prompt (maybe longer than the code)
 - × Autocomplete is more efficient
- × AI assistant is **non-deterministic**
 - × Different code might cause concept not clear
 - × Different code style can lead to confusion



AI Assistant

- ✗ Not always the case
- ✗ Example: API performance testing with AI
Assistants (see Bonus on last section of the course for discount)
- ✗ AI assistant is to aid
- ✗ AI assistant is not decision maker

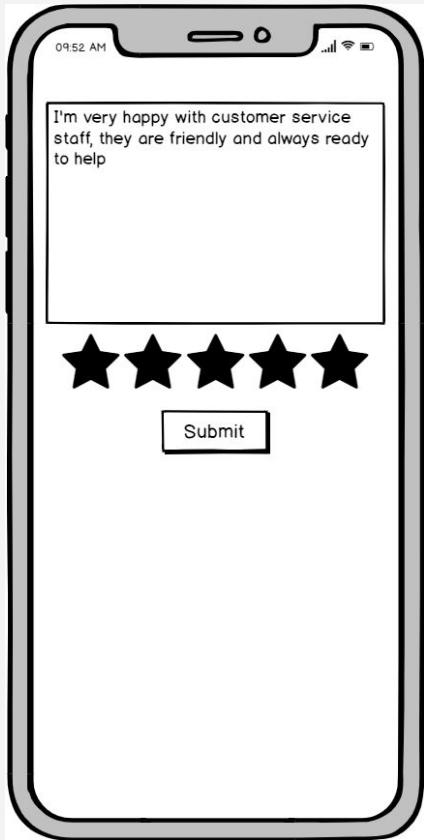


Feedback Stream

Are We Good Enough?

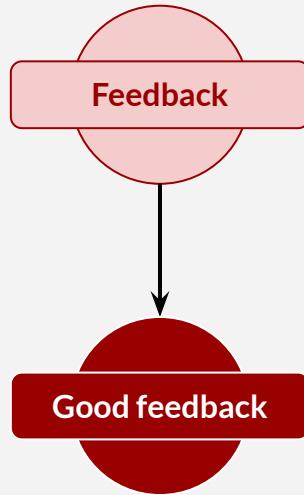


Customer Feedback



happy, good, helpful, etc

High Level Topology



Source Code for Feedback

- ✗ **Feedback*.java**
- ✗ **Package com.course.kafka**
 - ✗ **api.request**
 - ✗ **api.response**
 - ✗ **api.server**
 - ✗ **broker.message**
 - ✗ **broker.producer**
 - ✗ **command.action**
 - ✗ **command.service**

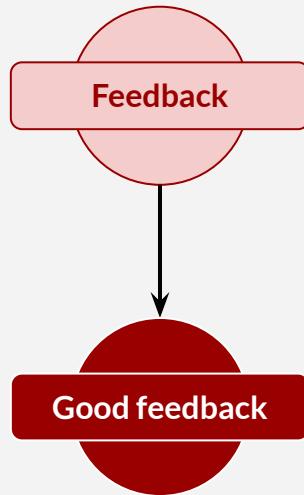


Feedback Stream

Who Owns This Feedback?



High Level Topology



Key : [branch location]

Value: [good word]

Java Stream API

- × Java stream API is not kafka stream
- × Same method names:
 - × **filter**
 - × **flatMap**
 - × **forEach**
 - × **map**
 - × **peek**



Feedback Stream

Good Feedback or Bad?

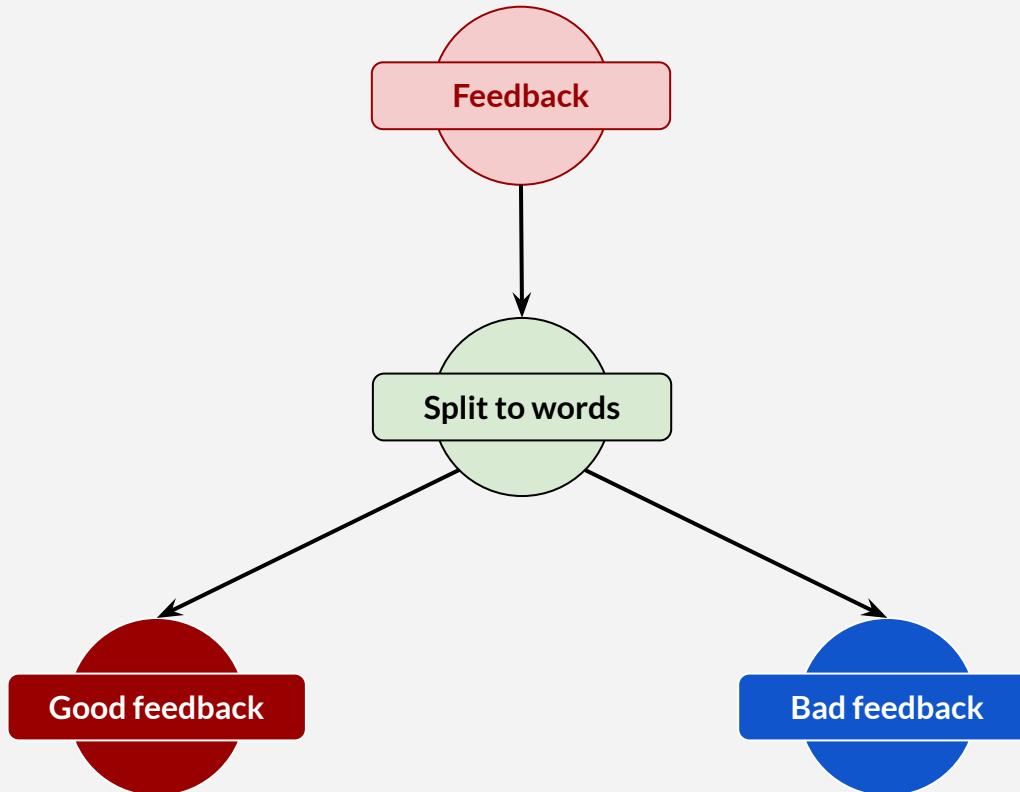


Bad Feedback?

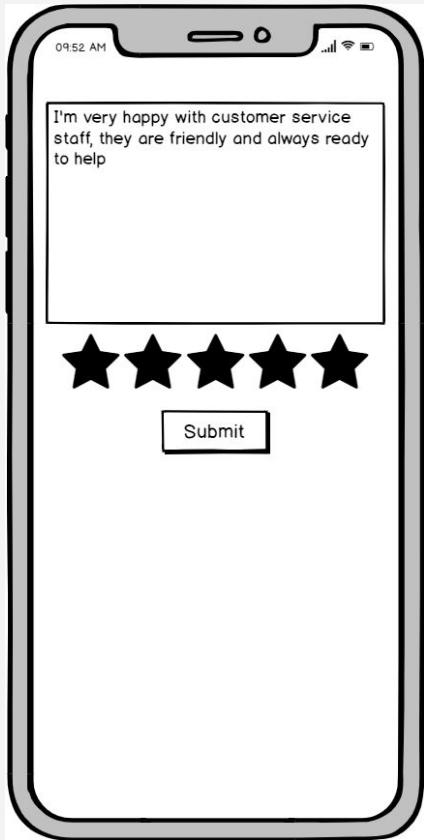
- ✗ Analyze bad feedback
- ✗ Stream to analyze feedback : good or bad?
- ✗ Pause the video & practice
- ✗ Topology
- ✗ Resume the video for answer



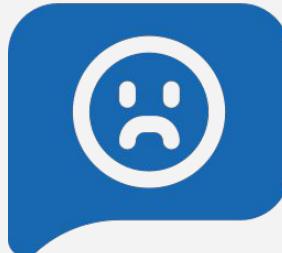
High Level Topology



Customer Feedback



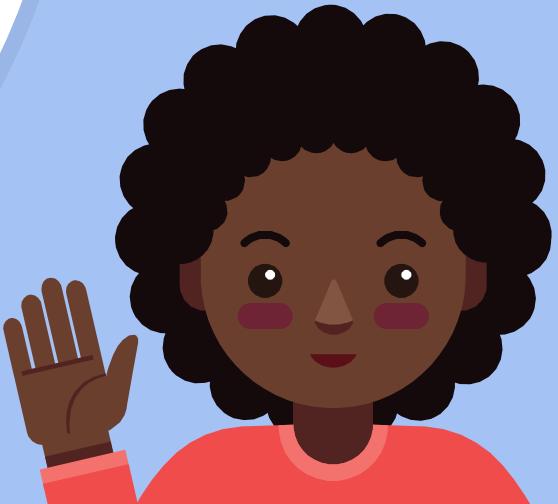
happy, good, helpful, etc



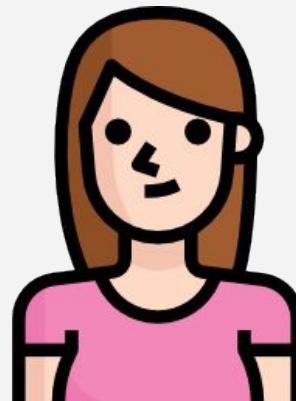
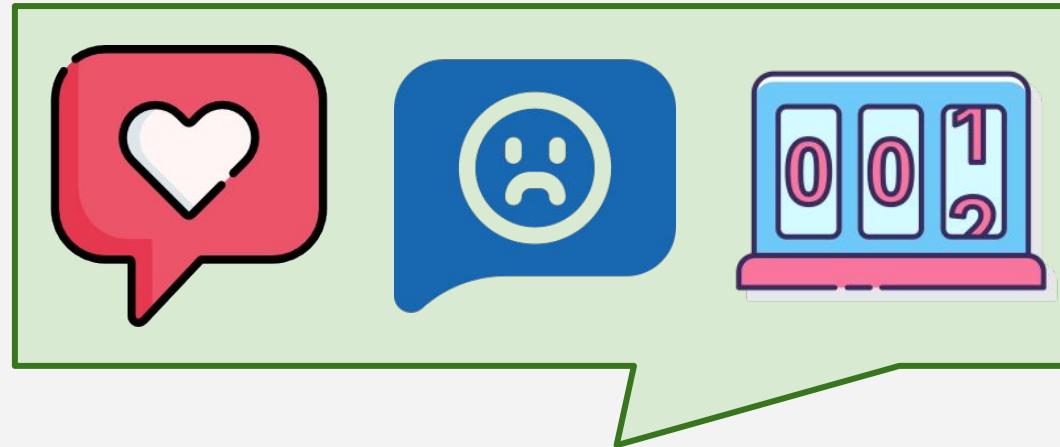
angry, sad, bad, etc

Feedback Stream

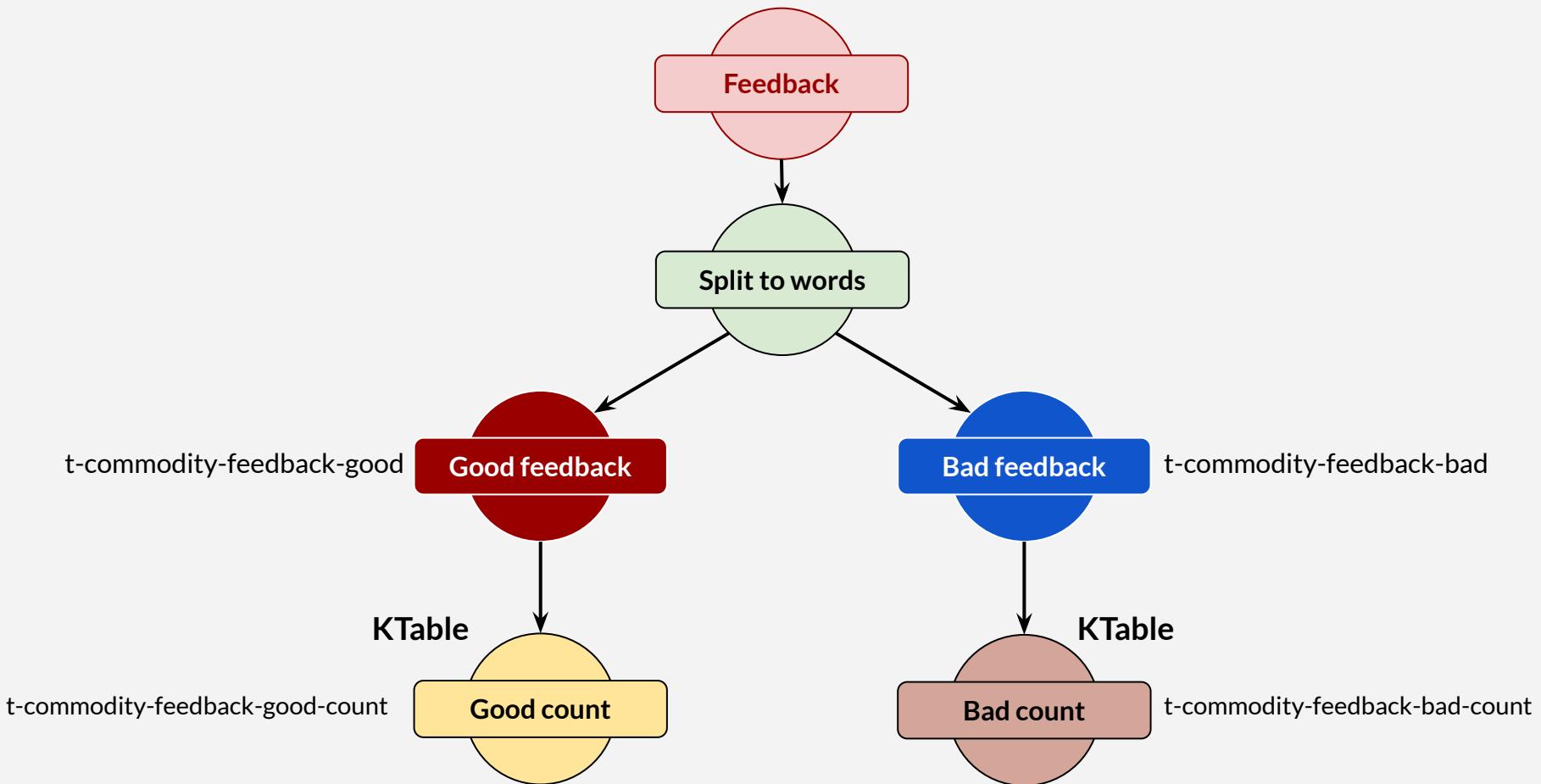
Group Using Table



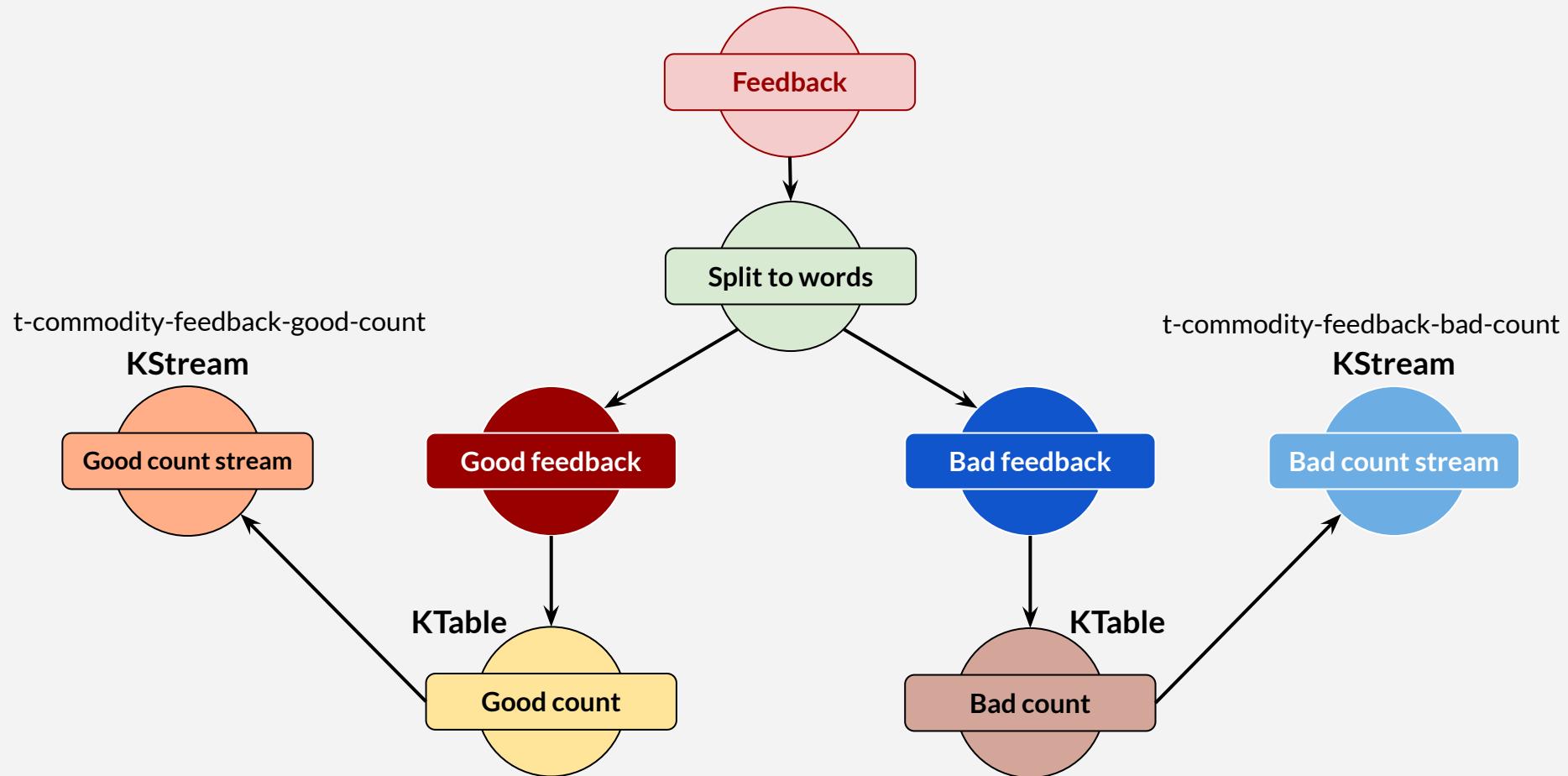
Count The Word



High Level Topology



High Level Topology



Feedback Stream

Delay on Table



Kafka Stream Configuration

- × Default configuration : 30 seconds
- × Cache and send
- × Kafka property : `commit.interval.ms`
- × Adjust configuration if needed



Feedback Stream

Send and Continue



Send and Continue

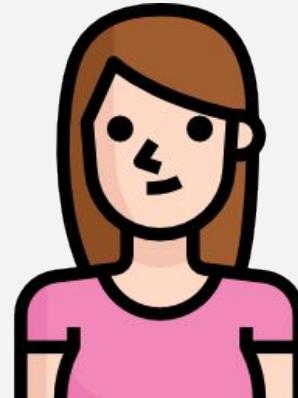
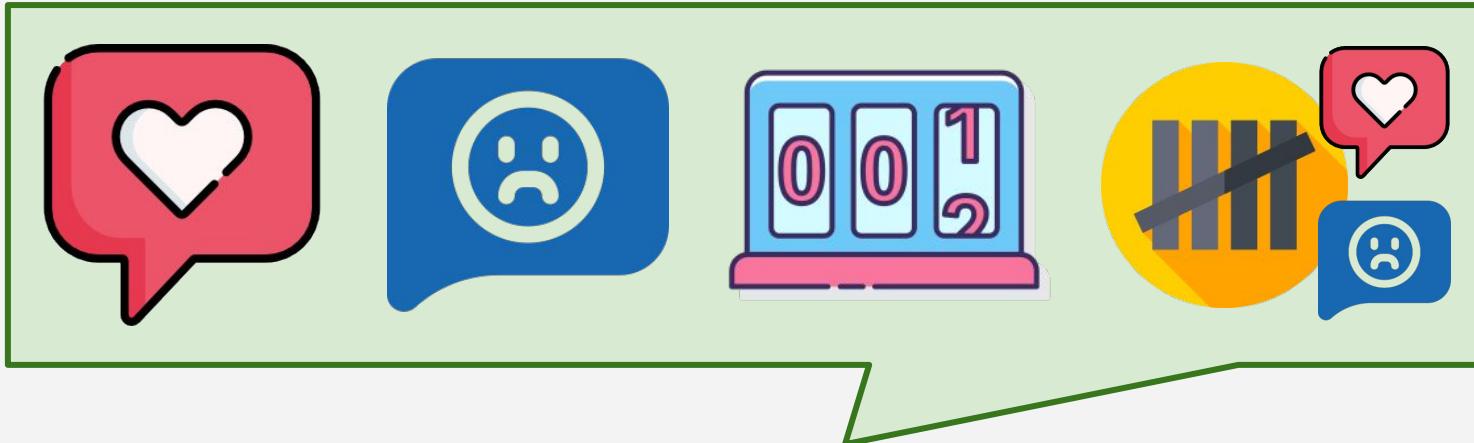
```
streams.to("first-output-topic");
streams.groupByKey().....to("second-output-topic");
```

Feedback Stream

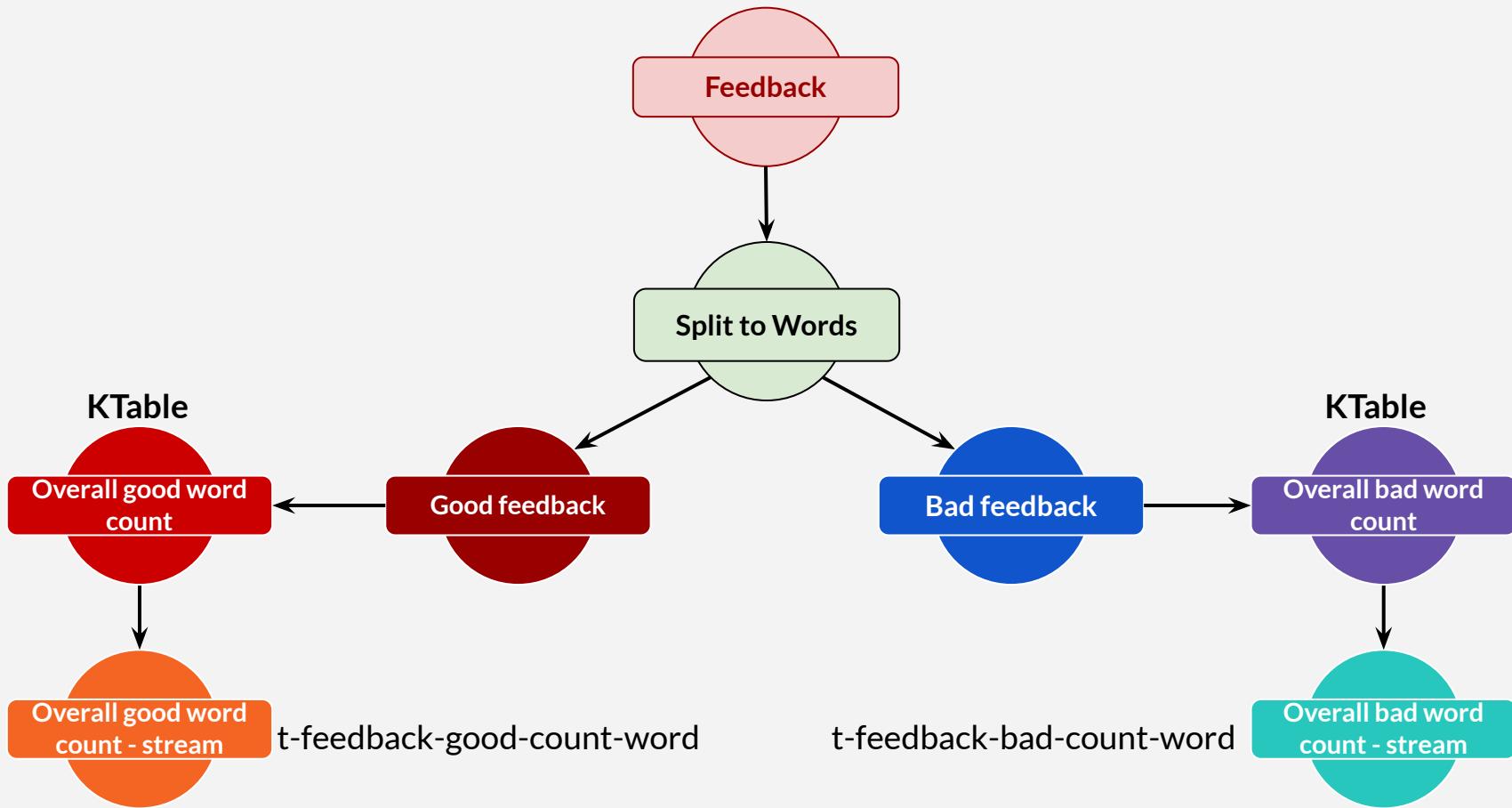
Overall Good (or Bad)



Count The Word



High Level Topology

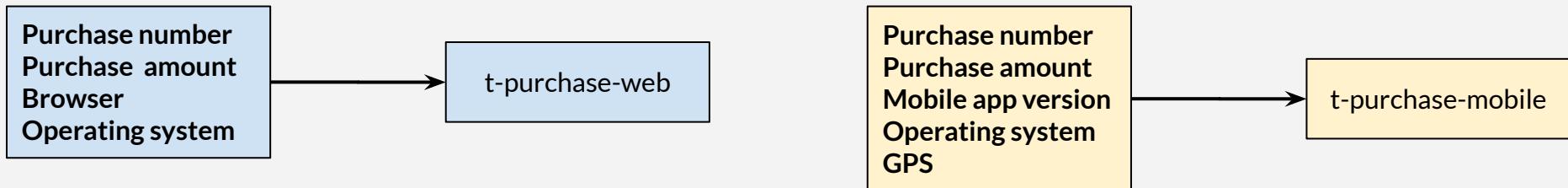


Customer Stream

Web & Mobile



Different Device, Different Data



t-purchase-all

Purchase Web 1

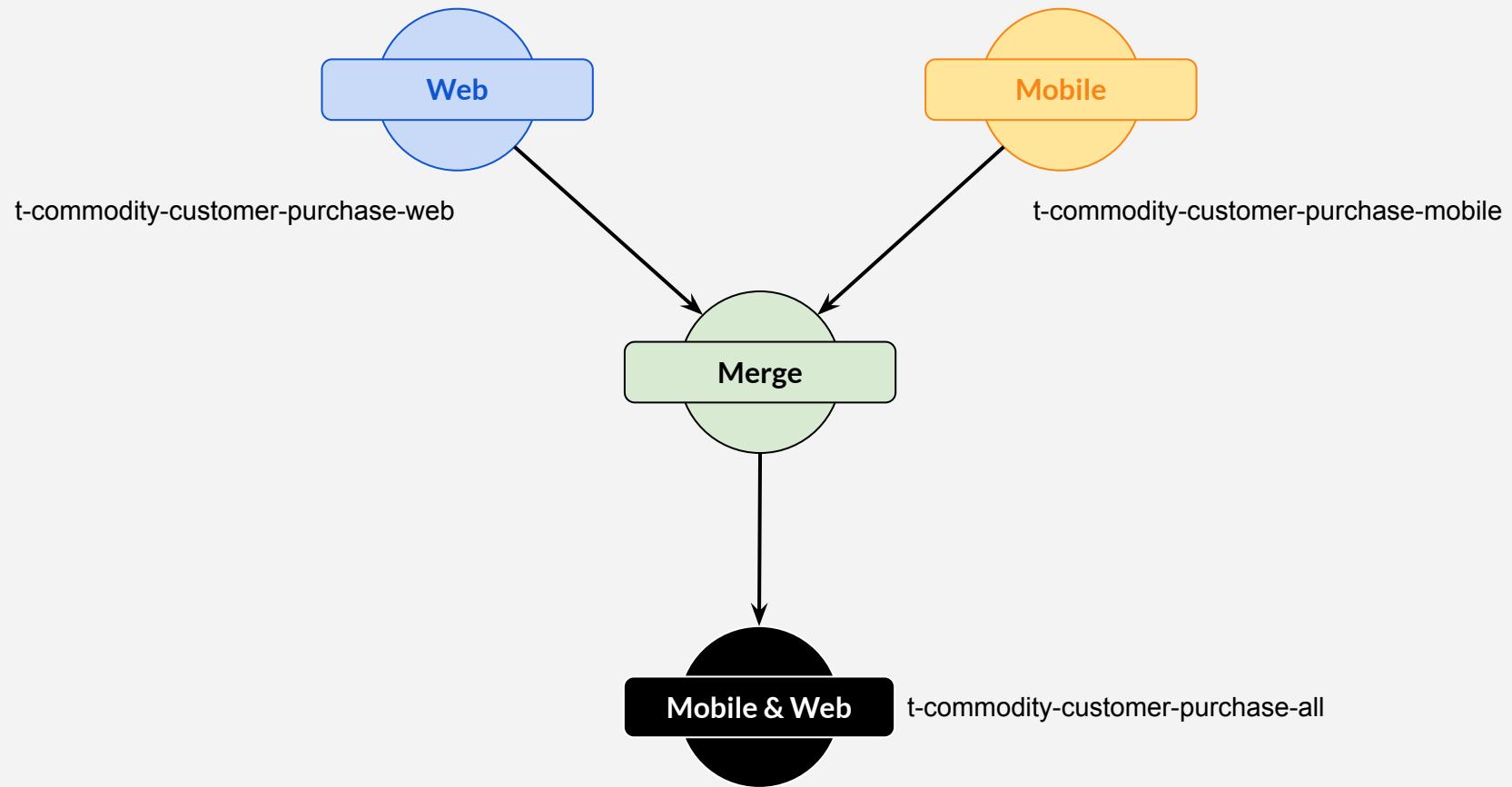
Purchase Mobile
1

Purchase Mobile
2

Purchase Web 2

Purchase Mobile
3

High Level Topology



Source Code for Customer Purchase

- ✗ `CustomerPurchase*.java`
- ✗ Package `com.course.kafka`
 - ✗ `api.request`
 - ✗ `api.server`
 - ✗ `broker.message`
 - ✗ `broker.producer`
 - ✗ `command.action`
 - ✗ `command.service`

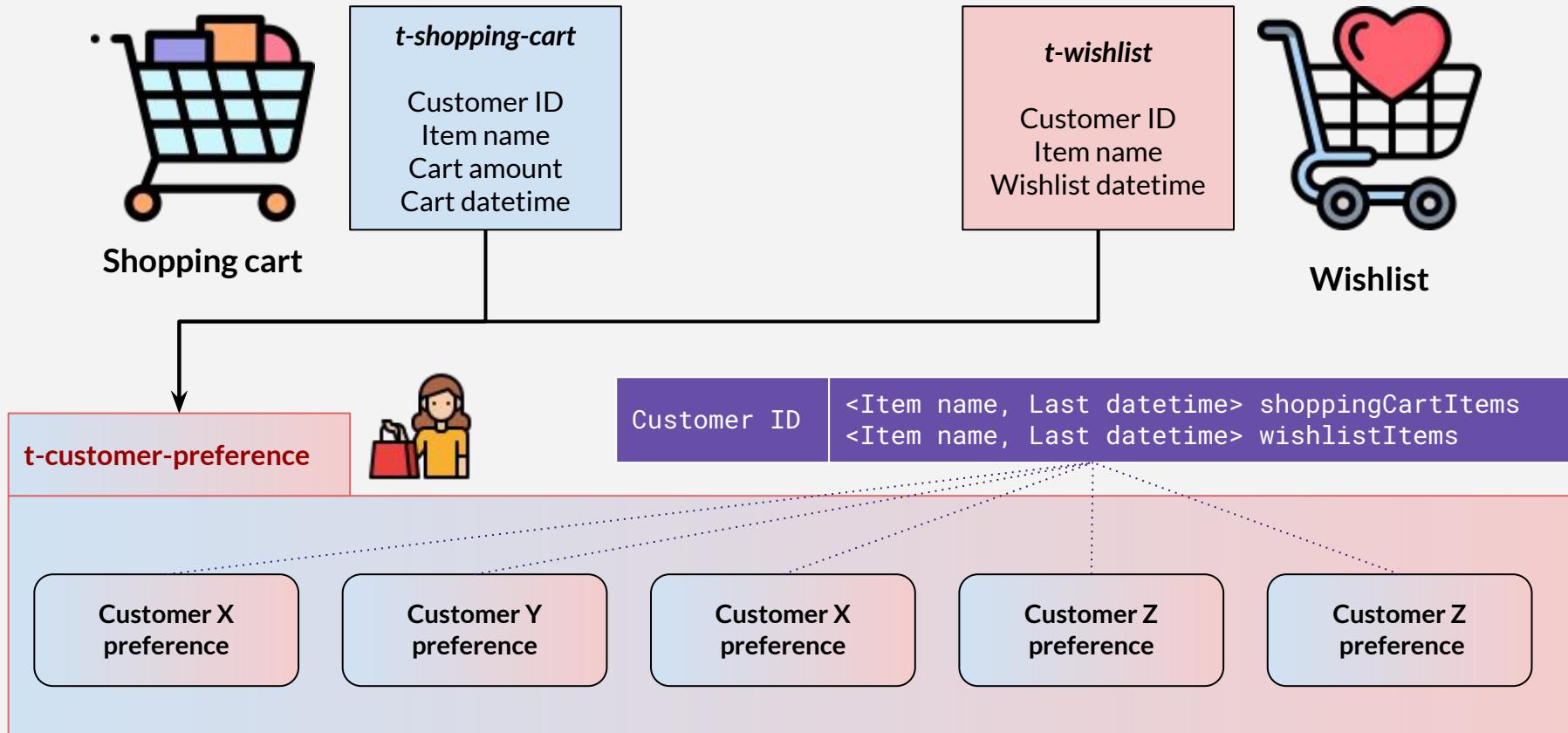


Customer Stream

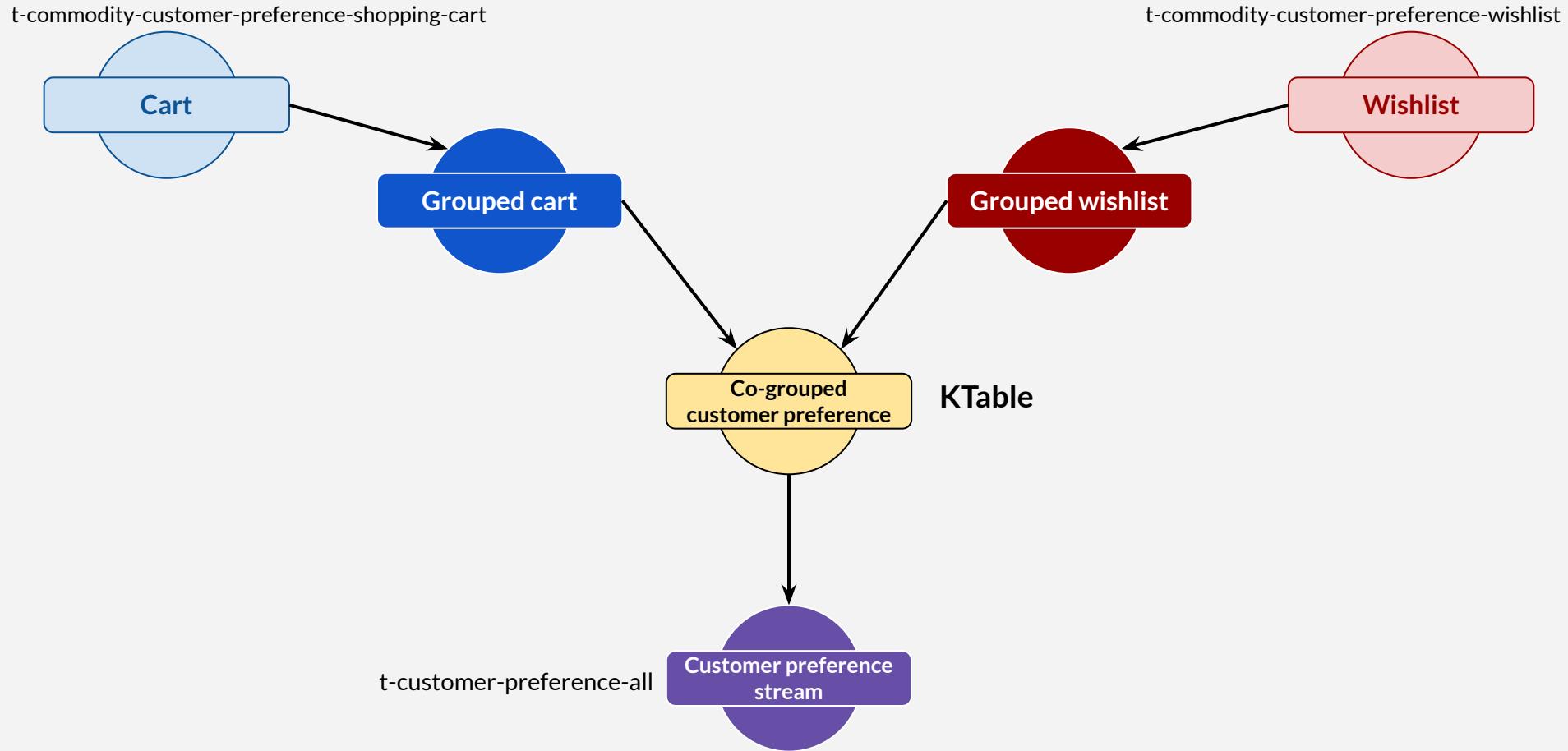
Cart & Wishlist



Two Things, One Customer Preference



High Level Topology



Source Code for Customer Purchase

- ✗ `CustomerPreference*.java`
- ✗ Package `com.course.kafka`
 - ✗ `api.request`
 - ✗ `api.server`
 - ✗ `broker.message`
 - ✗ `broker.producer`
 - ✗ `command.action`
 - ✗ `command.service`



data

Eve
Shopping cart
Apple

Eve
Shopping cart
Banana

Adam
Wishlist
Tomato

Eve
Wishlist
Cherry

Adam
Shopping cart
Garlic

Eve
Shopping cart
Apple

aggregate

Eve
Shopping cart
- Apple, T1

Wishlist

Eve
Shopping cart
- Apple, T1
- Banana, T2

Wishlist

Adam
Shopping cart

Wishlist
- Tomato, T3

Eve
Shopping cart
- Apple, T1
- Banana, T2

Wishlist
- Cherry, T4

Adam
Shopping cart
- Garlic, T5

Wishlist
- Tomato, T3

Eve
Shopping cart
- Apple, T6
- Banana, T2

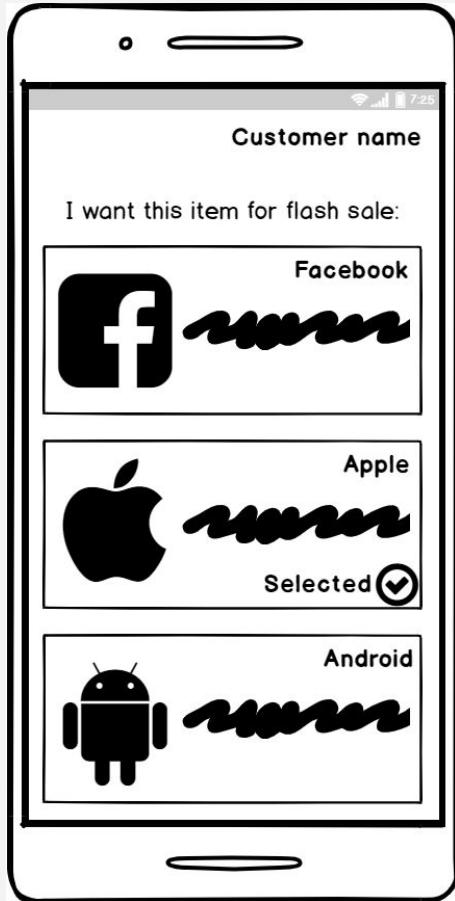

Wishlist
- Cherry, T4

Flash Sale Stream

Most Recent Data Feed



Flash Sale by Customer



- Customer can vote flash sale candidate
- Next flash sale will be most voted items
- One customer, one candidate
- Can change selection

Stream or Table?

- × Track latest selected item per customer
- × Kafka Stream table
- × Record
 - × Key : Customer ID
 - × Value : Flash sale candidate
- × Upsert (update if exist / insert if not exist)

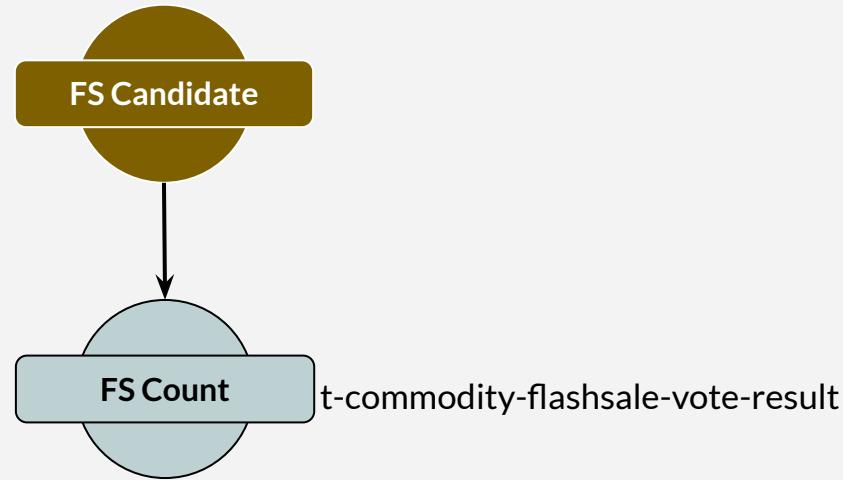


Source Code for Flash Sale Vote

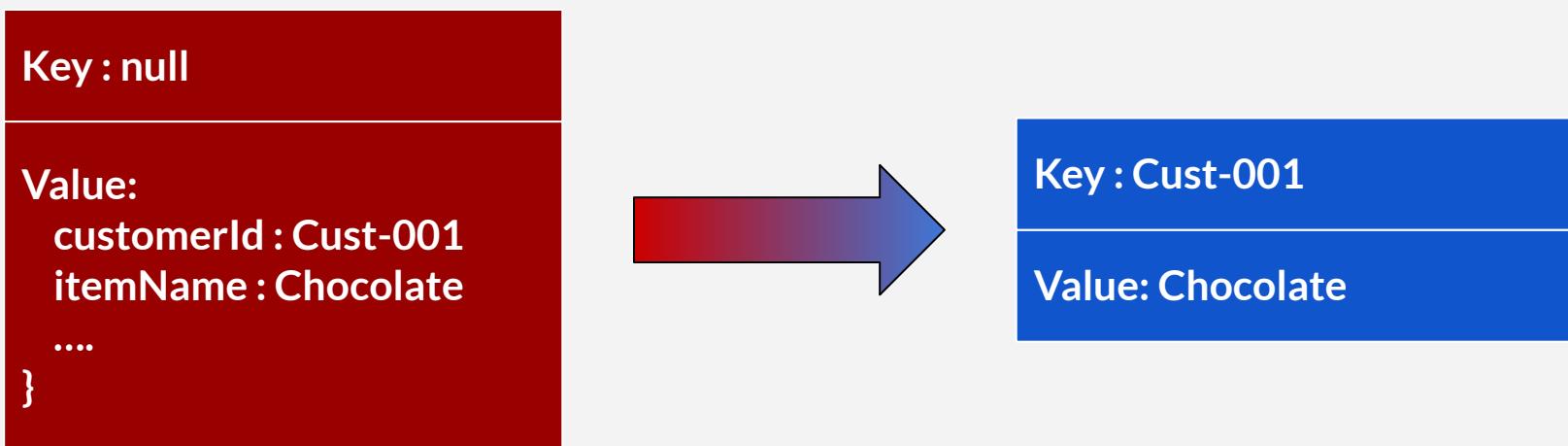
- ✗ `FlashSaleVote*.java`
- ✗ Package `com.course.kafka`
 - ✗ `api.request`
 - ✗ `api.server`
 - ✗ `broker.message`
 - ✗ `broker.producer`
 - ✗ `command.action`
 - ✗ `command.service`



High Level Topology



Processing The Message



How Vote Works?

Anna	Cookies	
------	---------	---

Olaf	Cookies	
------	---------	---

Olaf	Cake	
------	------	---

Anna	Cake	
------	------	---

Elsa	Cookies	
------	---------	---

	Cookies	1
---	---------	---

	Cookies	2
---	---------	---

	Cookies	1		Cake	1
---	---------	---	---	------	---

	Cookies	0		Cake	2
---	---------	---	---	------	---

	Cookies	1		Cake	2
---	---------	---	---	------	---

timeline



Flash Sale Stream

Stream & State



State in Kafka Stream

- × Care about current data
- × No need to know previous data
- × Flash sale example : need to know current / previous choice
- × "User choice" is kafka stream **state**
- × Stateless operations (commodity & feedback)
- × Stateful operations (flash sale vote)

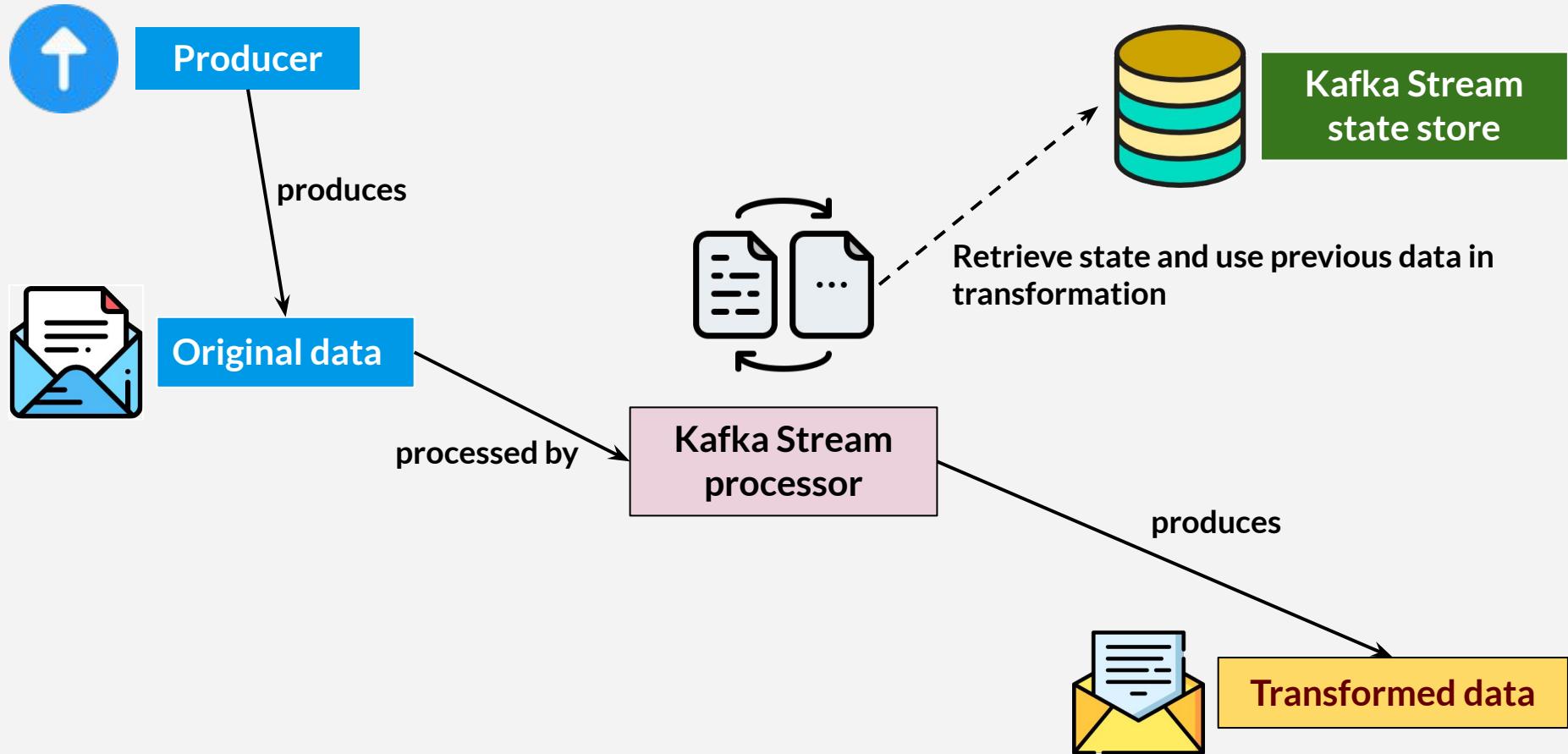


State Store

- × See existing information & connect it
- × Kept in state store
- × Key-value data storage
- × Accessed from processor
- × Kafka stream state stores:
 - × In-memory
 - × Persistent (disk based)



State Store



Important Aspects

- × Data locality
 - × Same machine with processing node
 - × No network overhead
 - × No sharing store
- × Fault tolerance
 - × Recover quickly in case application failure
 - × Use changelog topic

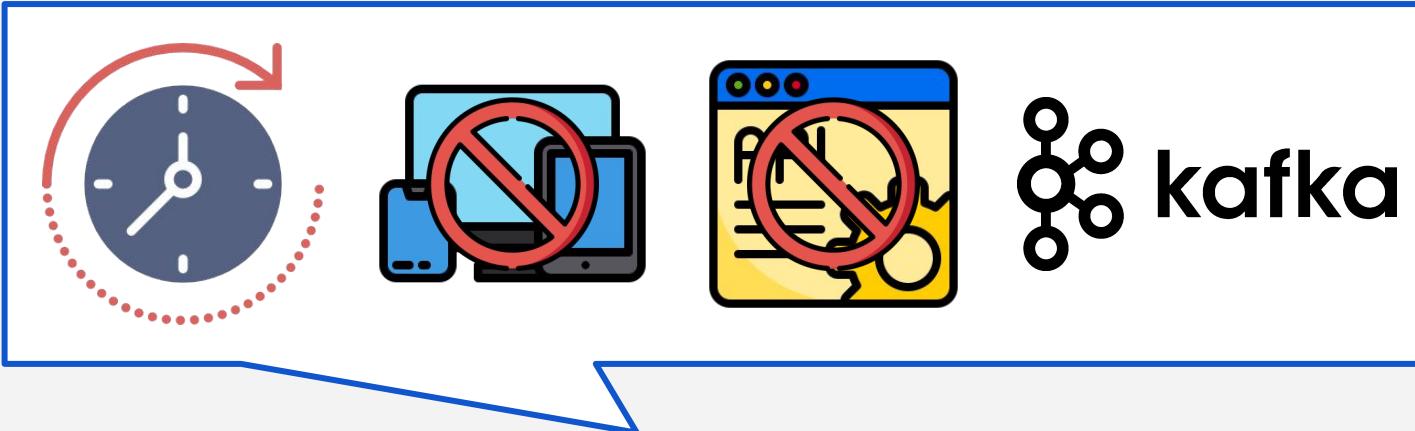


Flash Sale Stream

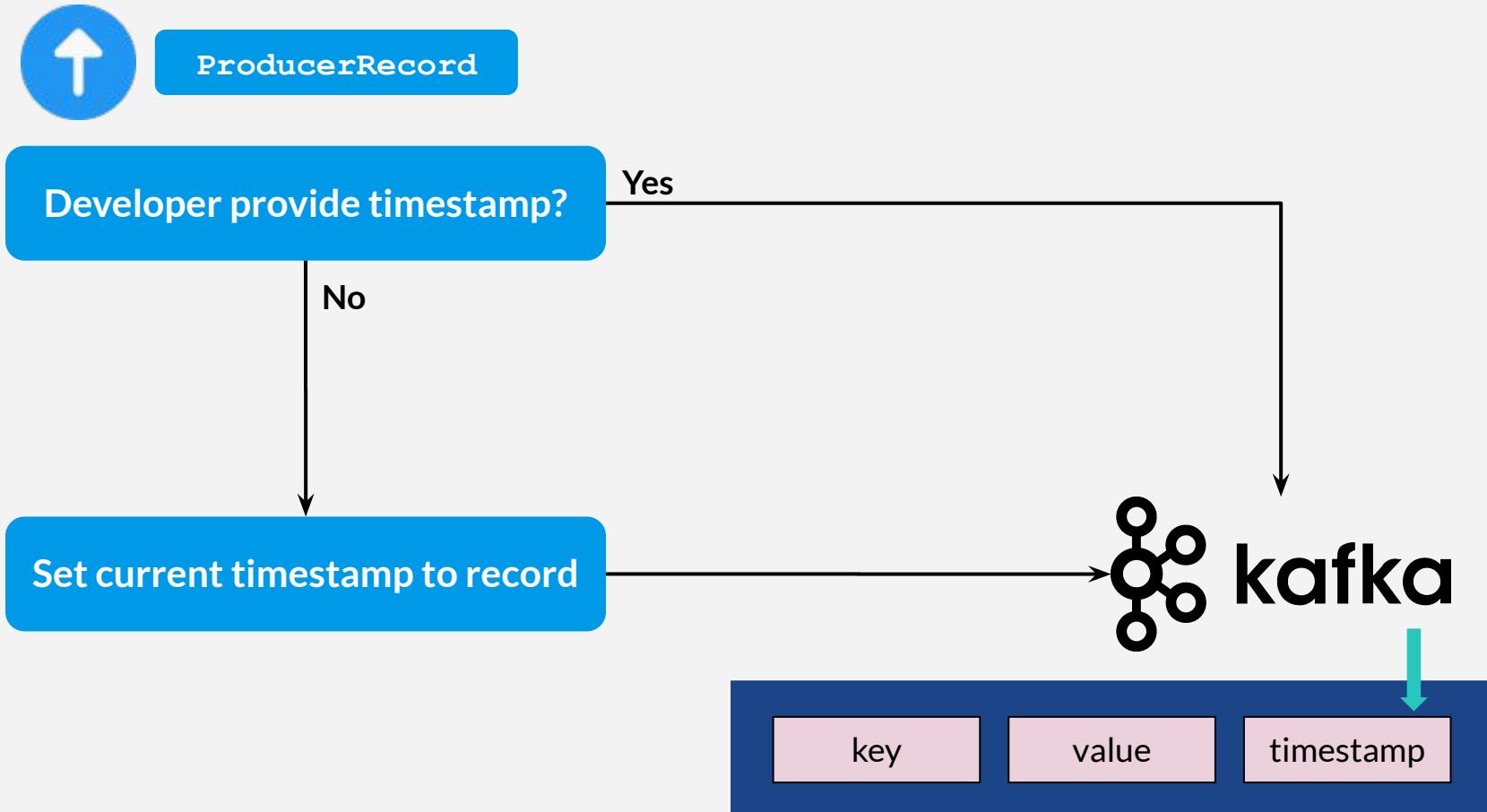
Timestamp



Flash Sale by Customer



Kafka Timestamp



Processor API

- × Use processor API
- × Extract timestamp
- × Useful not only for timestamp
- × Learn basic processor API



Value Transformer

```
stream.mapValues()
```



```
stream.transformValues()
```



```
class X implements ValueTransformer<Value, TransformedValue>  
  
.transformValues(() -> new X())
```

```
class Y implements ValueTransformerSupplier<Value, TransformedValue> {  
  
    public ValueTransformer<Value, TransformedValue> get() {...}  
  
}  
  
.transformValues(new Y())
```

OffsetDateTime to Timestamp

- ✗ Epoch time millisecond vs Java OffsetDateTime
- ✗ `OffsetDateTime.toInstant().toEpochMilli()`



processValues

- ✗ `transformValues` is deprecated
- ✗ Use `processValues`
- ✗ Requires `FixedKeyProcessor<keyInput, valueInput, valueOutput>`
- ✗ `valueInput` can be different from `valueOutput`



Feedback Rating

Average Rating



Feedback Dashboard

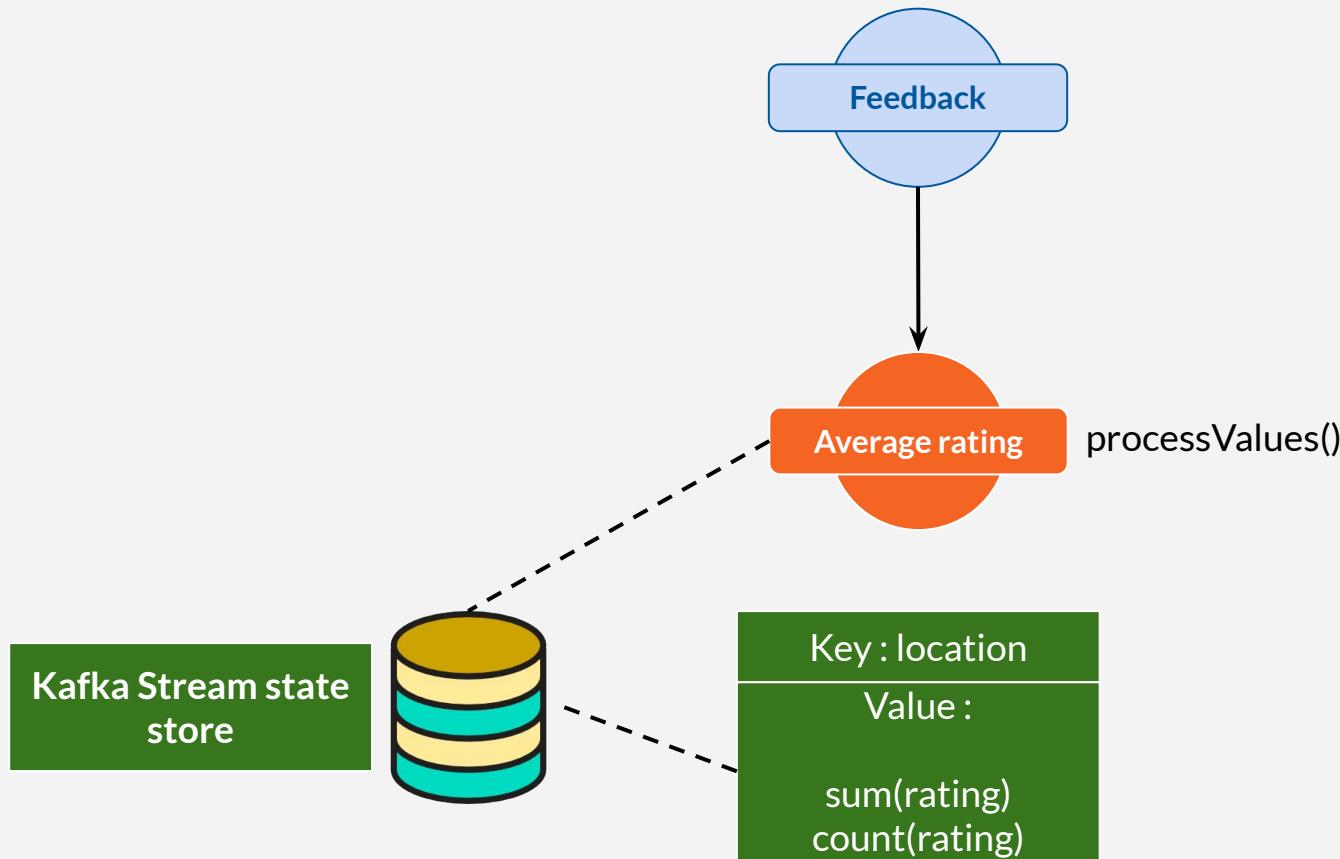


Average Rating

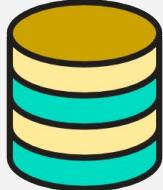
- × average = sum(ratings) / count(ratings)
- × Need to know all ratings
- × Use processor API & state store
- × State store?



High Level Topology



State Store



Indonesia
Sum : 12
Count : 4

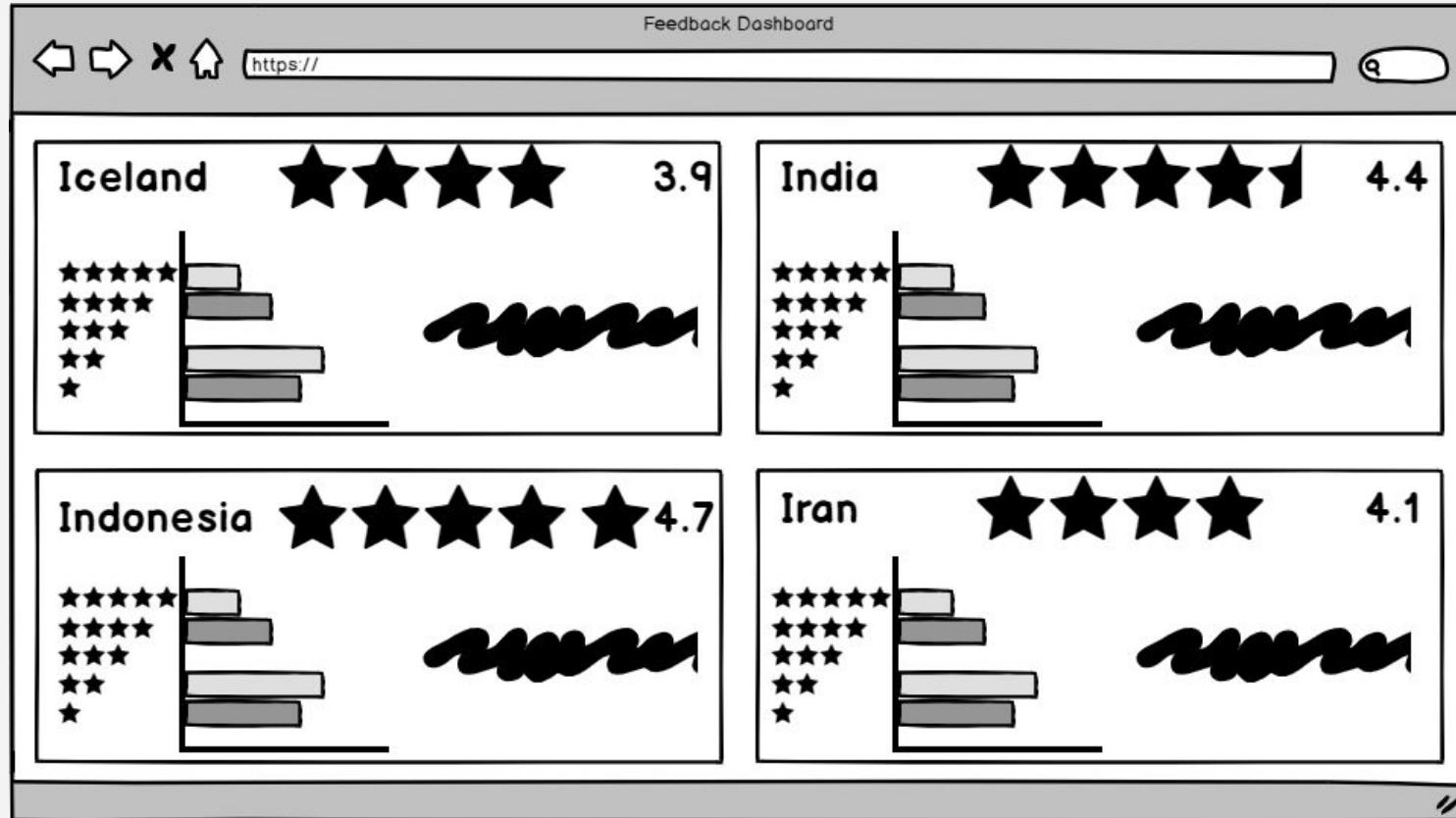
Singapore
Sum : 8
Count : 2

Feedback Rating

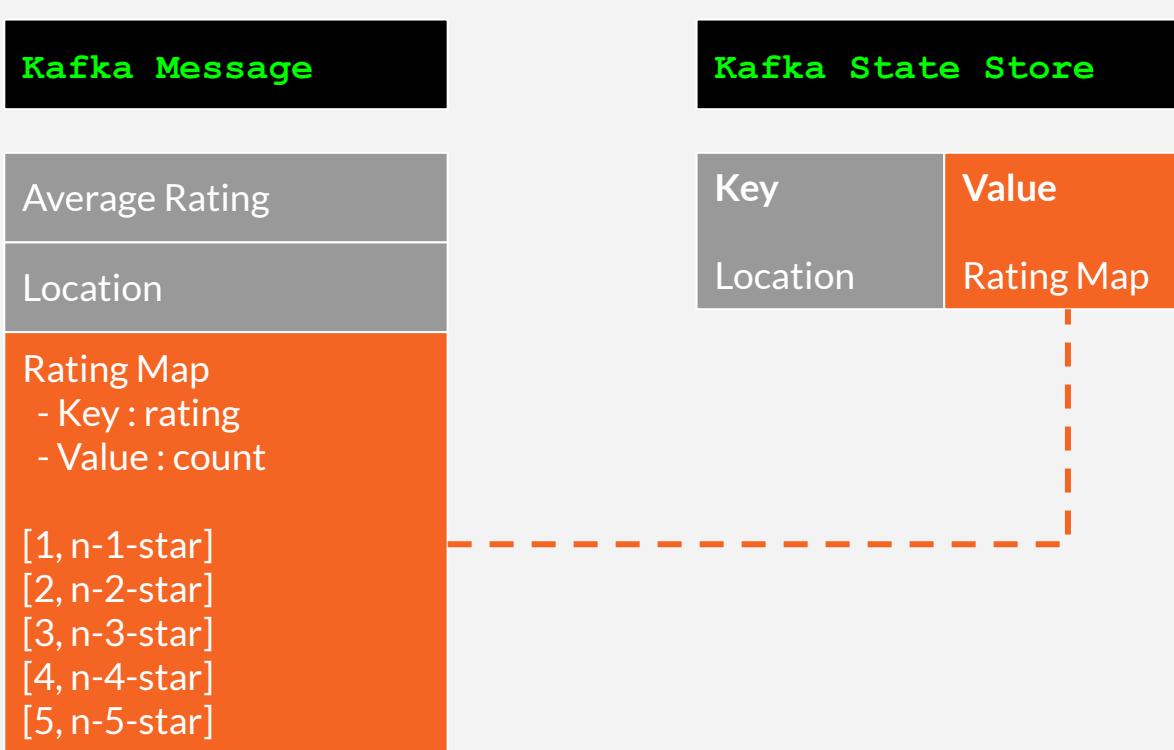
Detailed Rating



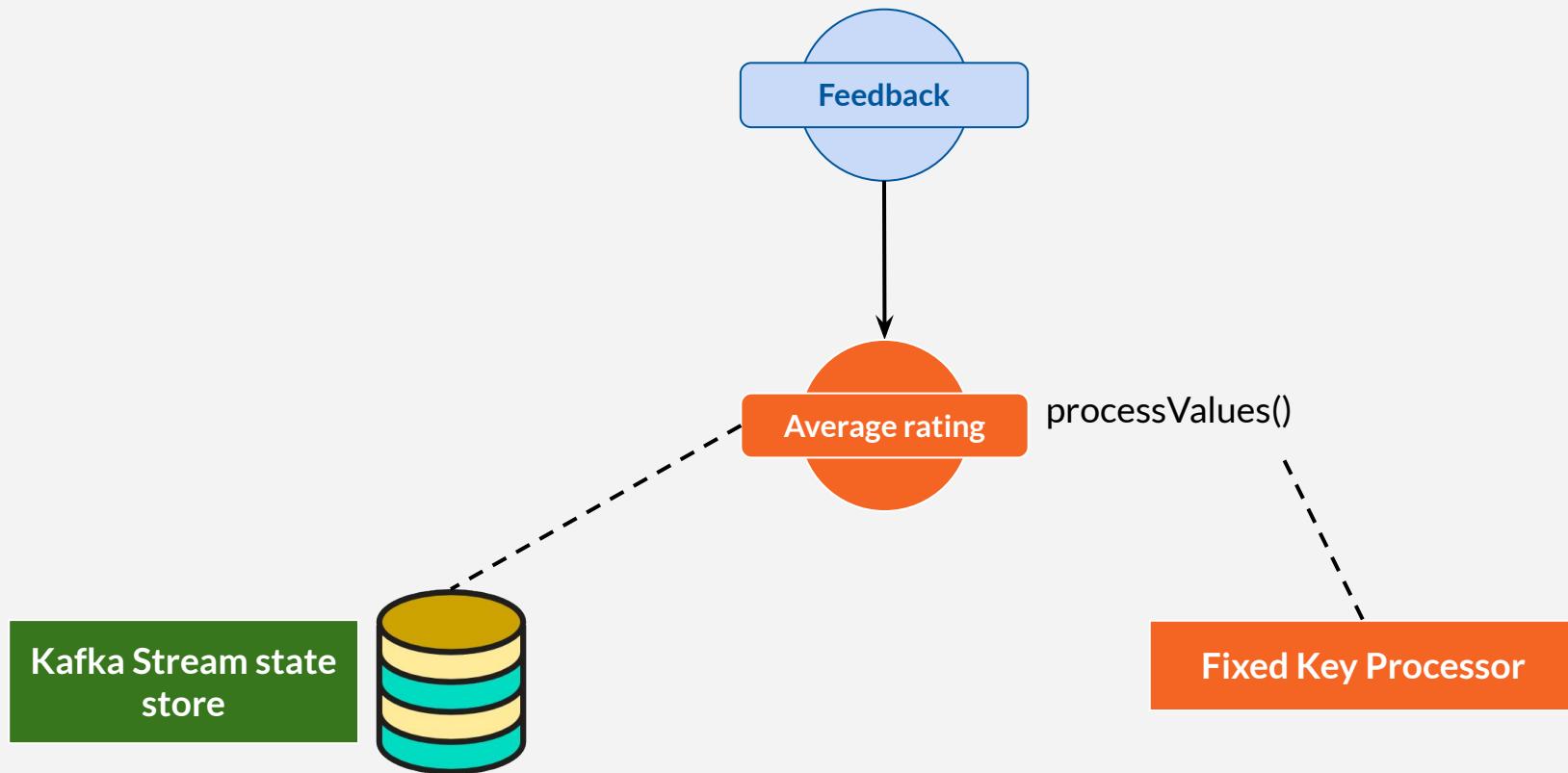
Feedback Dashboard



Data Structure



High Level Topology

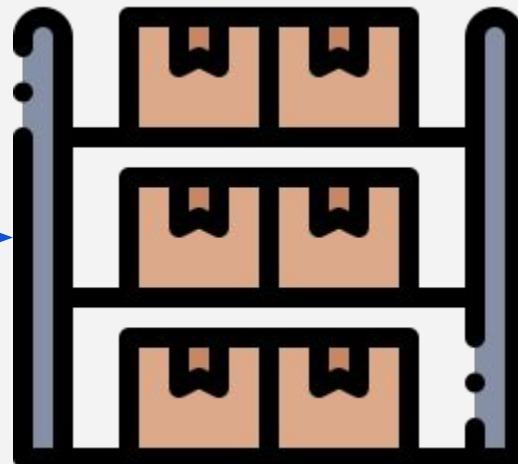


Inventory Summing Records

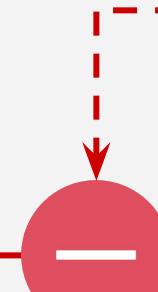


Inventory

Supplier delivery



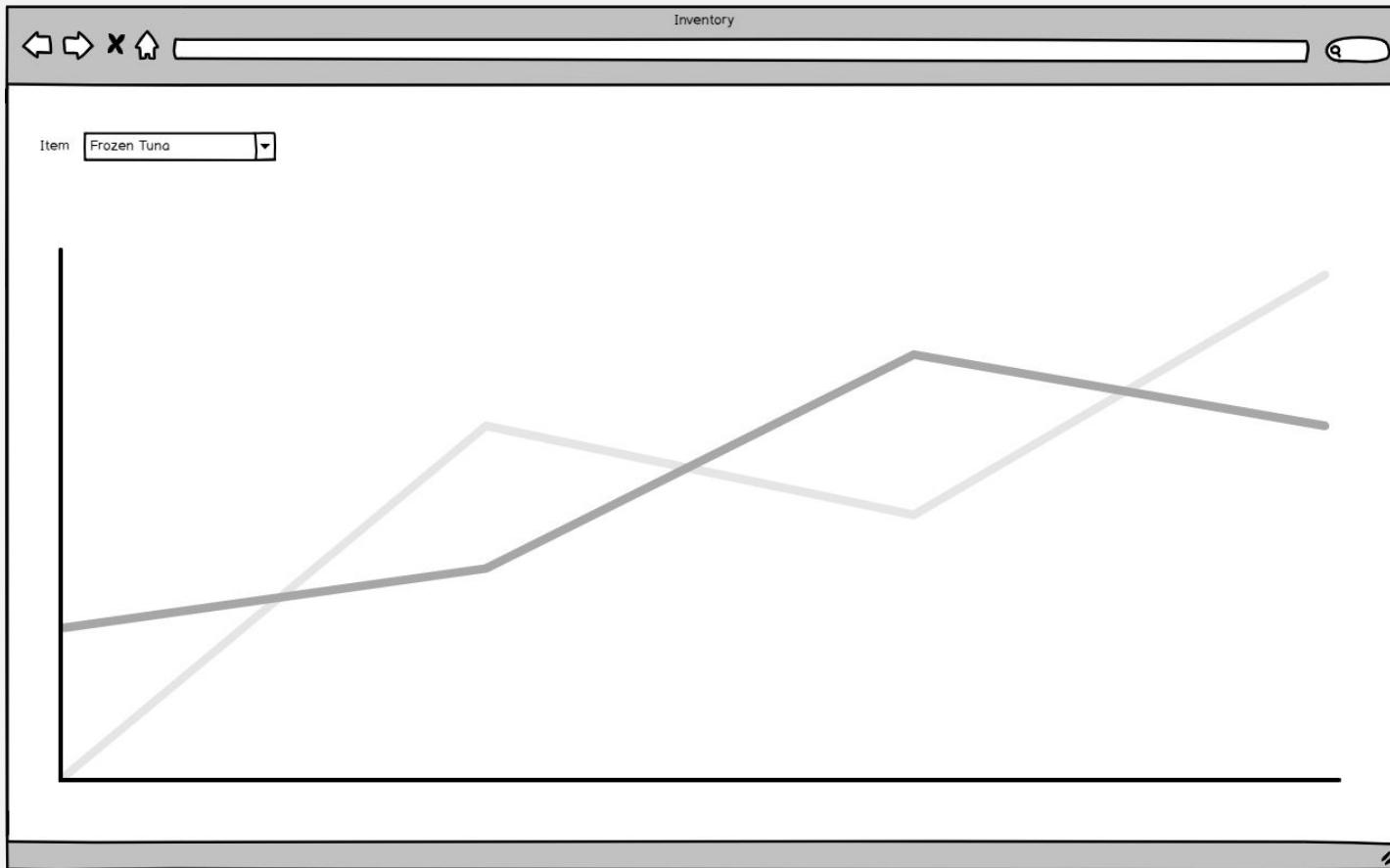
Item sold



Item refund, etc

Item moved to store display, etc

Inventory Chart



Source Code for Inventory

- ✗ **Inventory*.java**
- ✗ **Package com.course.kafka**
 - ✗ **api.request**
 - ✗ **api.server**
 - ✗ **broker.message**
 - ✗ **broker.producer**
 - ✗ **command.action**
 - ✗ **command.service**

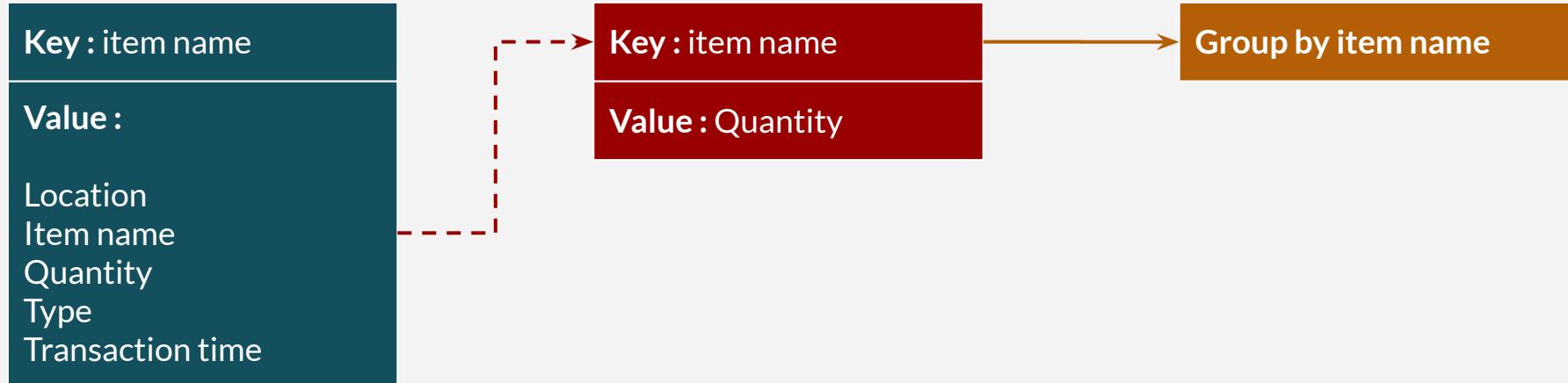


In This Section

- × Kafka stream aggregation (sum)
- × Inventory message sent
 - × Key : item name
 - × Value : item name, location, quantity, type
 - × Value (optional) : transaction time
- × No validation, e.g :
 - × Negative amount
 - × Inventory quantity still enough for removal
 - × etc



Grouping & Aggregate



Grouping & Aggregate

Record #	Record after map values	Group by key	KGroupedStream aggregate		Aggregated
			Initializer	Adder	
1	(candy, 4)	(candy, 4)	0 (for candy)	(candy, 0 + 4)	(candy, 4)
2	(apple, 2)	(apple, 2)	0 (for apple)	(apple, 0 + 2)	(apple, 2)
3	(candy, 3)	(candy, 3)		(candy, 4 + 3)	(candy, 7)
4	(candy, 1)	(candy, 1)		(candy, 7 + 1)	(candy, 8)
5	(apple, 4)	(apple, 4)		(apple, 2 + 4)	(apple, 6)
6	(apple, 5)	(apple, 5)		(apple, 6 + 5)	(apple, 11)

Inventory Subtracting Value



Grouping & Aggregate



Inventory

Using Reduce



Inventory Timestamp Extractor



Timestamp

Key : item name

Value :

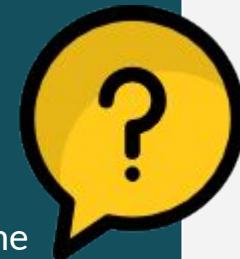
Location

Item name

Quantity

Type

Transaction time



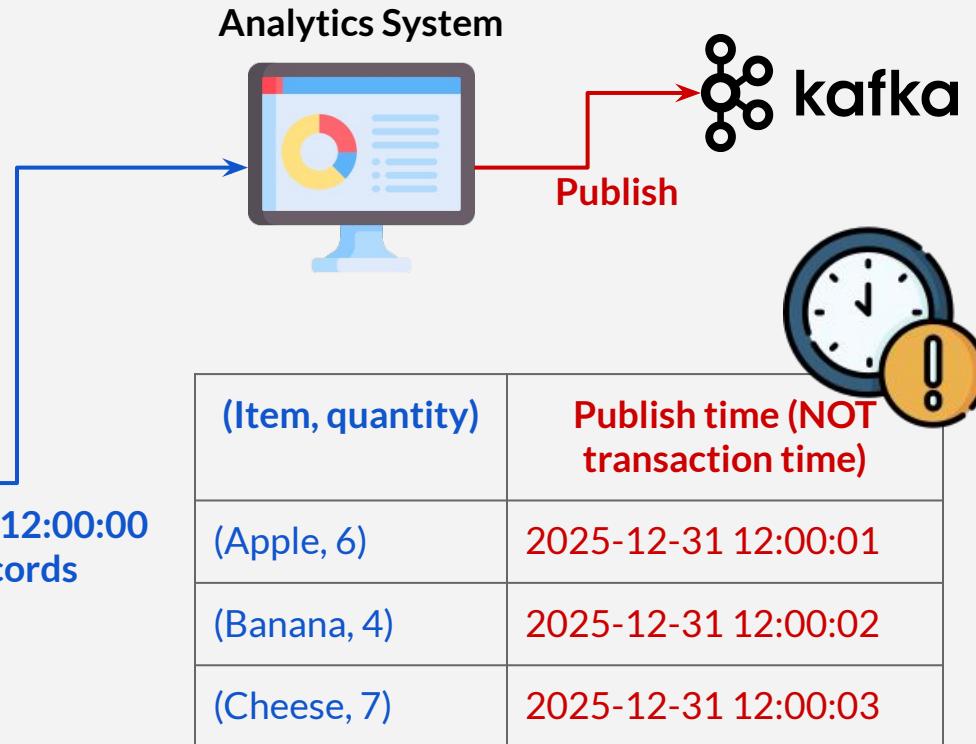
Timestamp

(Item, quantity)	Transaction time
(Apple, 6)	2025-12-31 08:56:00
(Banana, 4)	2025-12-31 10:24:00
(Cheese, 7)	2025-12-31 11:37:00



Warehouse Inventory
System

Hit API start at 12:00:00
For all 3 records



Timestamp Extractor

- × Extract transaction time from payload
- × Use it as record timestamp
- × Built-in timestamp extractor (FYI)
 - × FailOnInvalidTimestamp
 - × LogAndSkipOnInvalidTimestamp
 - × UsePreviousTimeOnInvalidTimestamp
 - × WallclockTimestampExtractor



Windowing



Windowing

- × Group records for same key
- × Time-based & session based
- × Example : How many click advertisement X in 15 minutes?
- × Need to know the accumulative result based on 15 minutes window
 - × The 1st 15 minutes : 2000
 - × The 2nd 15 minutes : 3000
 - × Same advertisement, different grouping



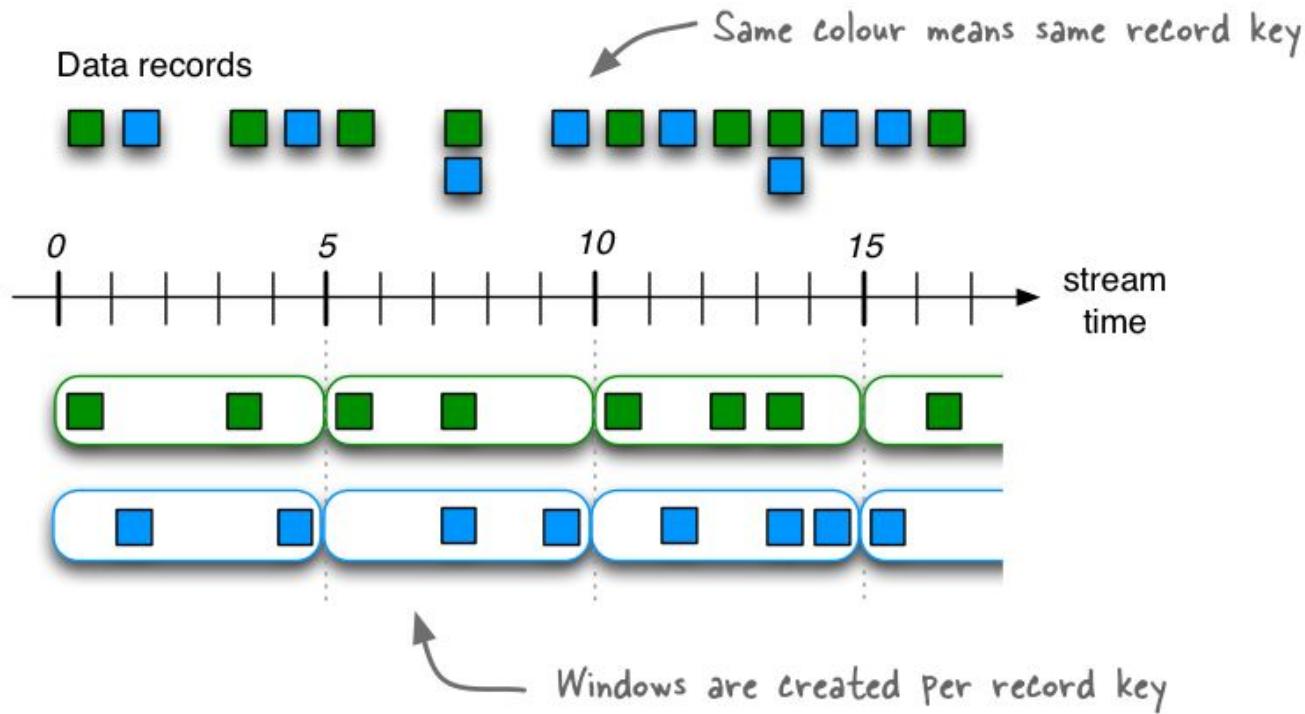
Windowing

- ✗ Tumbling Time Window
- ✗ Hopping Time Window
- ✗ Session Based Window



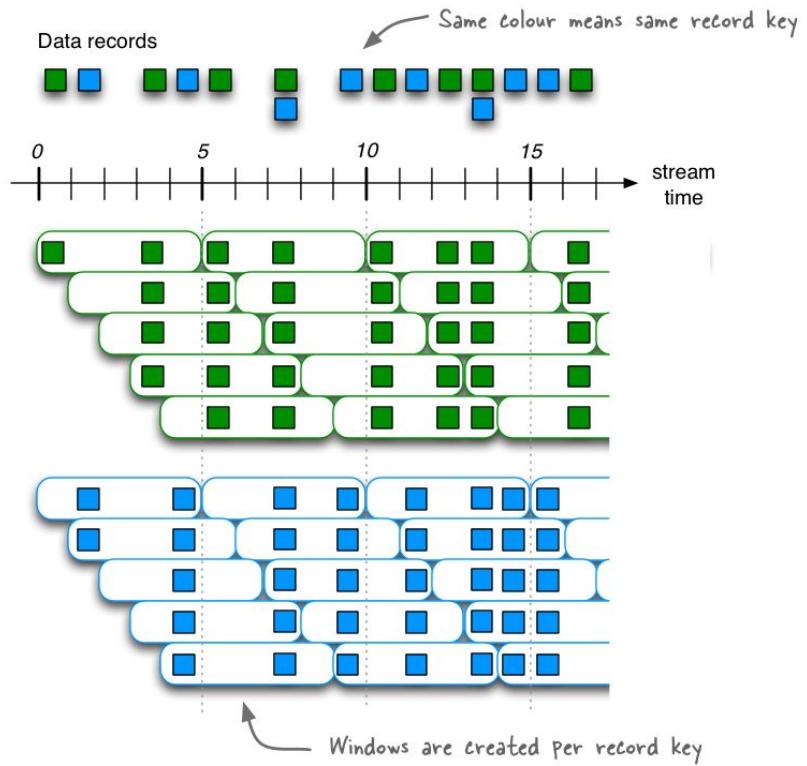
Tumbling Time Window

A 5-min Tumbling Window



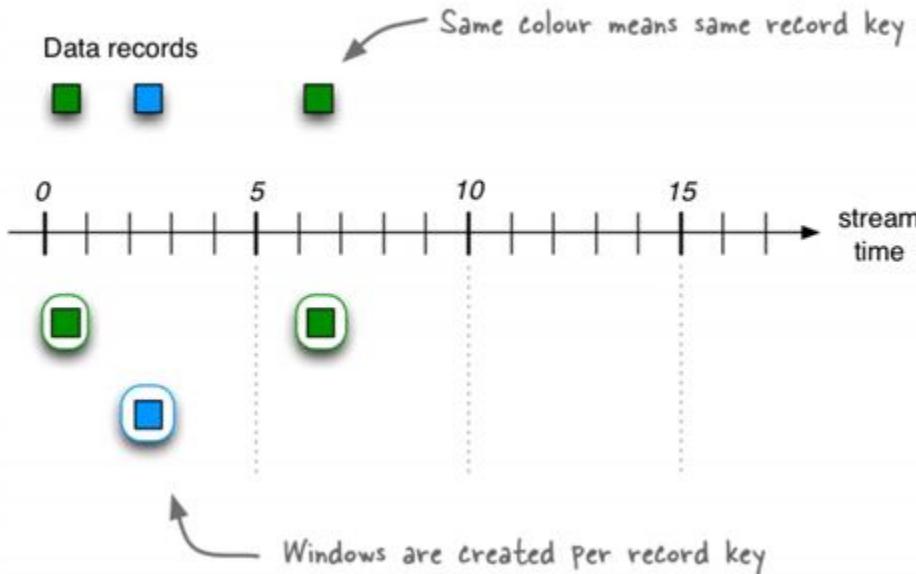
Hopping Time Window

A 5-min Hopping Window with a 1-min "hop"



Session Window

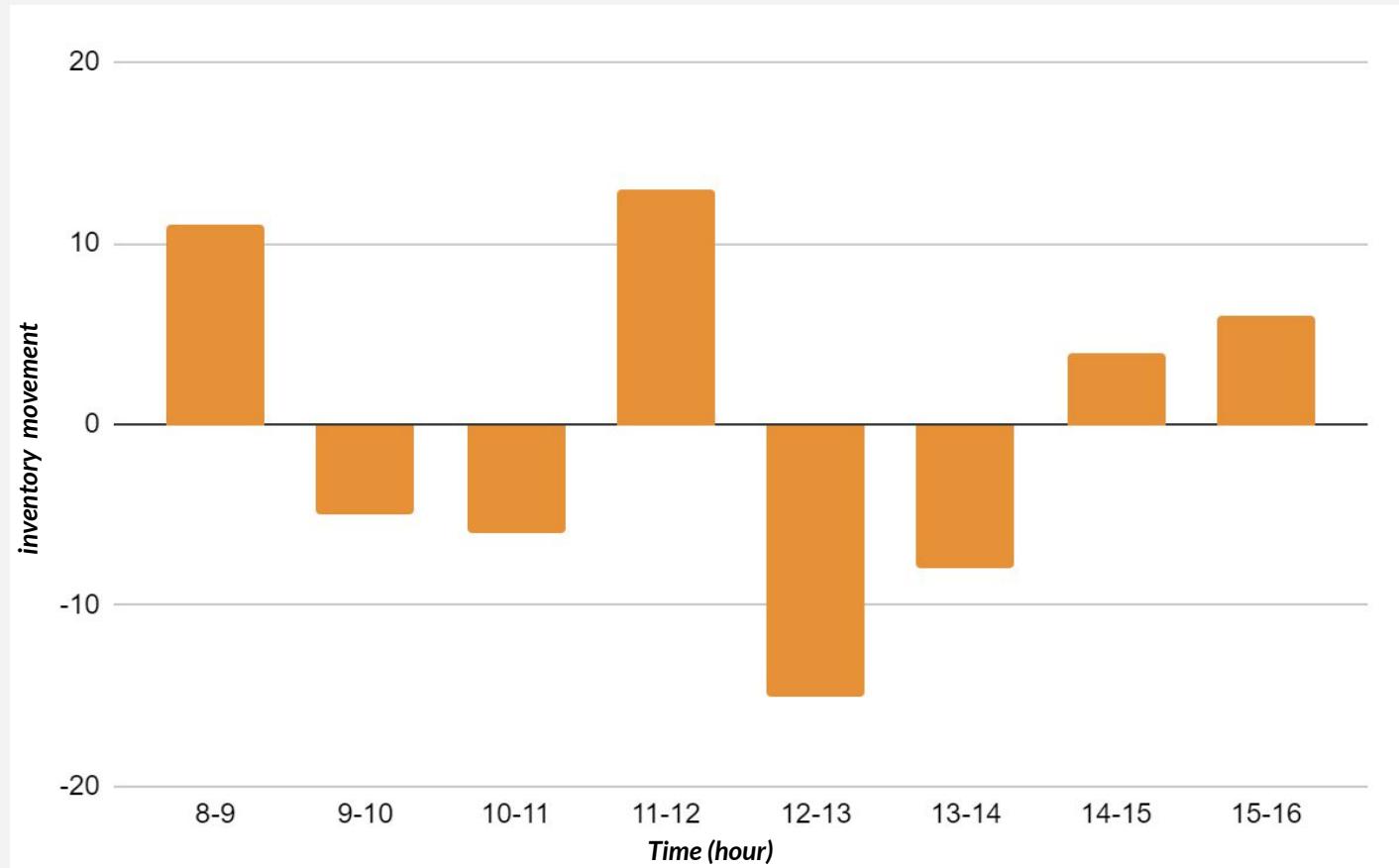
A Session Window with a 5-min inactivity gap



Inventory Tumbling Time Window



Inventory Movement



Inventory Transaction for Item X

Time Range : 08:00-09:00	
Time	Transaction
8:00:00	50
8:10:00	15
8:20:00	-8
8:30:00	7
8:40:00	-12
8:50:00	11

Aggregated Sum : 63

Time Range : 09:00-10:00	
Time	Transaction
9:00:00	9
9:10:00	4
9:20:00	-6
9:30:00	-3
9:40:00	10
9:50:00	-6

Aggregated Sum : 8

Time Range : 10:00-11:00	
Time	Transaction
10:00:00	3
10:10:00	-4
10:20:00	-17

Aggregated Sum : -18

How We Do It

- × Extract transaction time
- × Create tumbling time window



Log Output

Group 1:
08:00-09:00

[Apple@2024-12-31T08:00Z/2024-12-31T09:00Z], 64
[Apple@2024-12-31T08:00Z/2024-12-31T09:00Z], 63

Group 3:
10:00-11:00

[Apple@2024-12-31T09:00Z/2024-12-31T10:00Z], 7
[Apple@2024-12-31T09:00Z/2024-12-31T10:00Z], 8
[Apple@2024-12-31T10:00Z/2024-12-31T11:00Z], -1
[Apple@2024-12-31T10:00Z/2024-12-31T11:00Z], -18

Group 2:
09:00-10:00

[Apple@2024-12-31T08:00Z/2024-12-31T09:00Z], 63

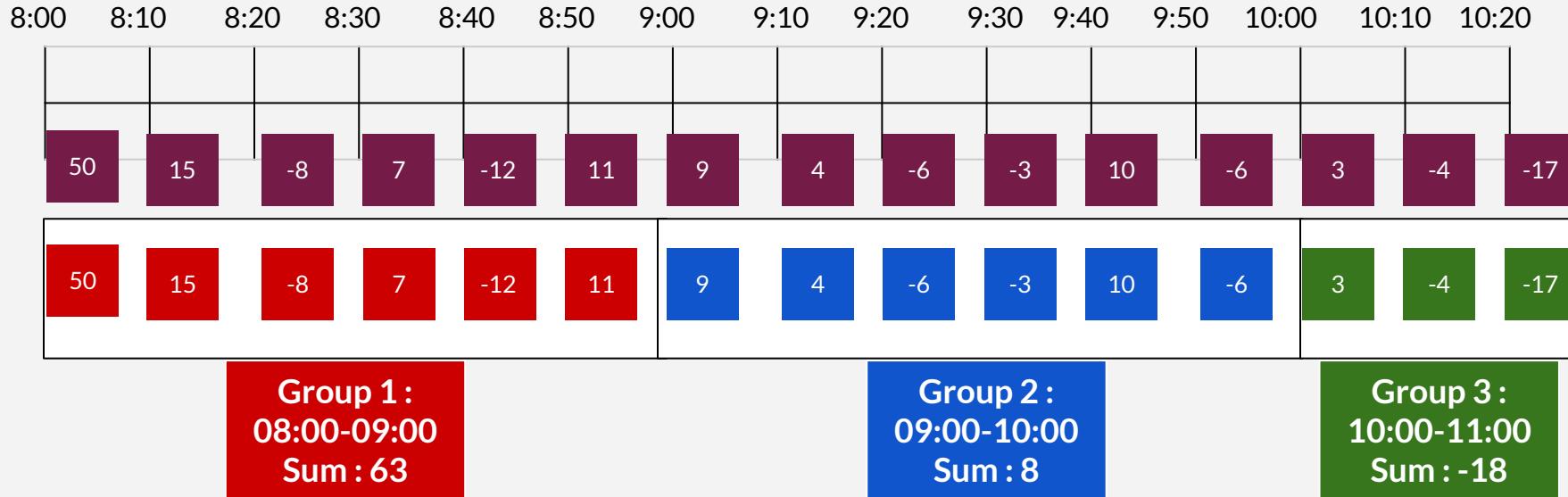
Original key

Window start

Window end

Value

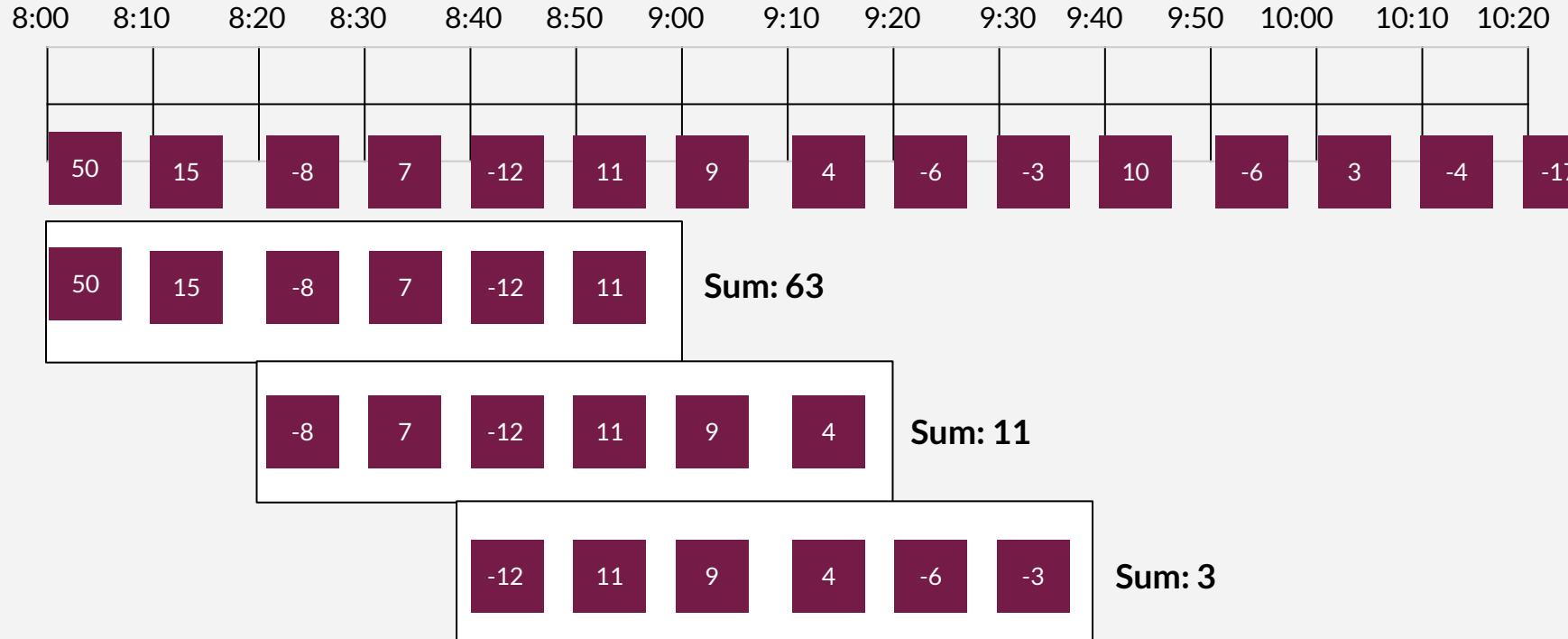
Tumbling Time Window



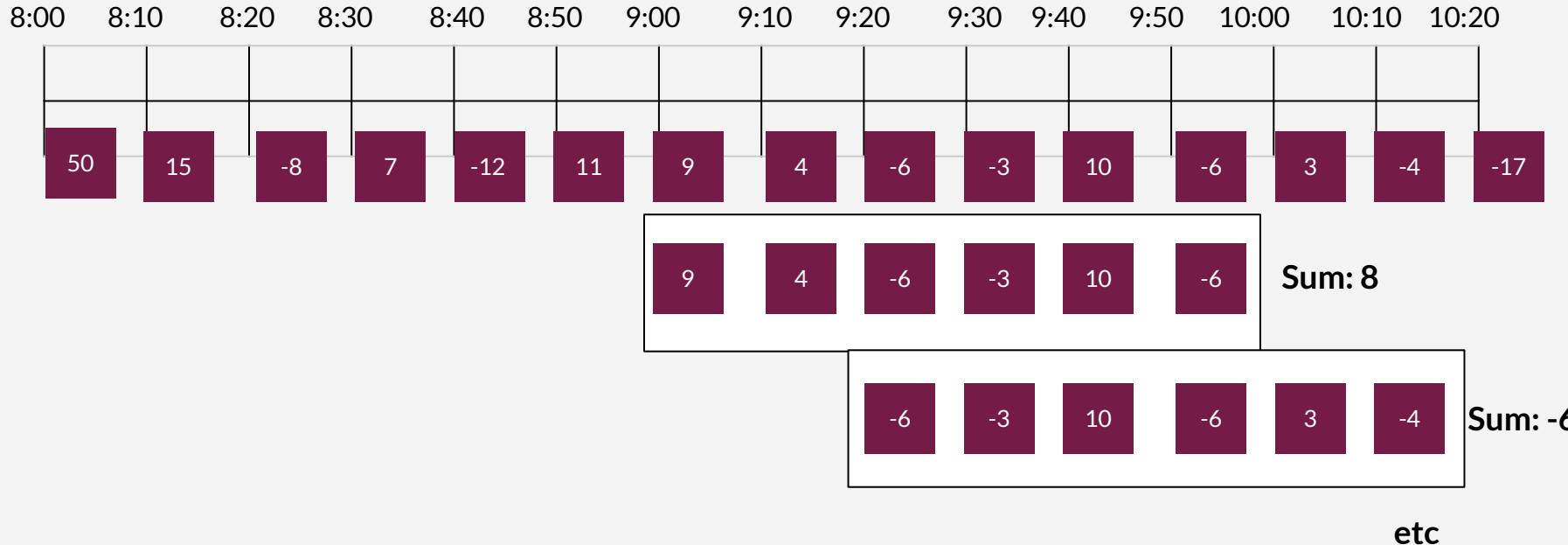
Inventory Hopping Time Window



Hopping Time Window



Hopping Time Window - continued

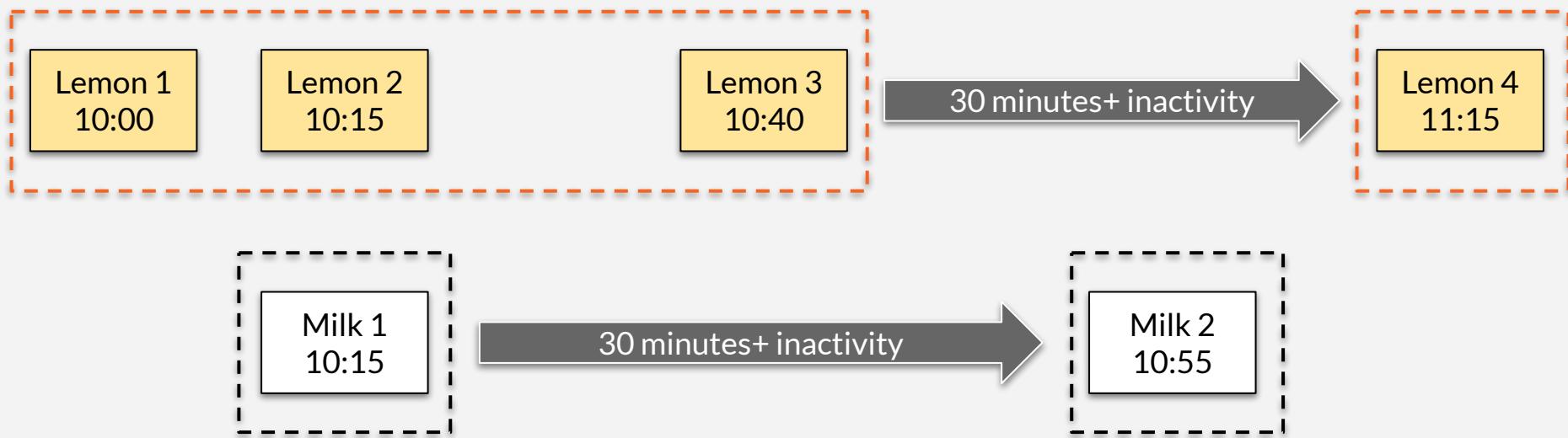


Inventory Session Window



Event

Inactivity gap : 30 minutes



Join Theory



Kafka Stream Join

- × Taking two streams / tables and make new stream / table out of them
- × Join based on record key
- × Join on:
 - × KStream & KStream
 - × KTable & KTable
 - × KStream & KTable
 - × KStream & GlobalKTable
- × Join types : inner, left, outer



Join Combinations

Left Type	Right Type	Co-Partition	Windowed	Join Types		
				Inner Join	Left Join	Outer Join
KStream	KStream	Required	Yes	Supported	Supported	Supported
KTable	KTable	Required	No	Supported	Supported	Supported
KStream	KTable	Required	No	Supported	Supported	Not Available
KStream	GlobalKTable	Not required	No	Supported	Supported	Not Available

Co-Partition

- × Combinations:
 - × KStream & KStream
 - × KTable & KTable
 - × KStream & KTable
- × Must be co-partitioned
- × Otherwise, **runtime error**



Co-partition

- × Left type must have same number of partition with right type
 - × Left : x & Right : x = co-partitioned
 - × Left : x & Right : y = Not co-partitioned
- × Same partitioning strategy
 - × Key with same value must go same partition
 - × If use custom logic to send to partition n , both must be the same



GlobalKTable

- × No need co-partition
- × GlobalKTable:
 - × Fetch / copy all data from all partitions of certain topic
 - × Live in kafka stream instance
 - × Consumes memory / disk on kafka stream instance
 - × Relatively small data & almost static
- × Can join stream to GlobalKTable, although different partition number

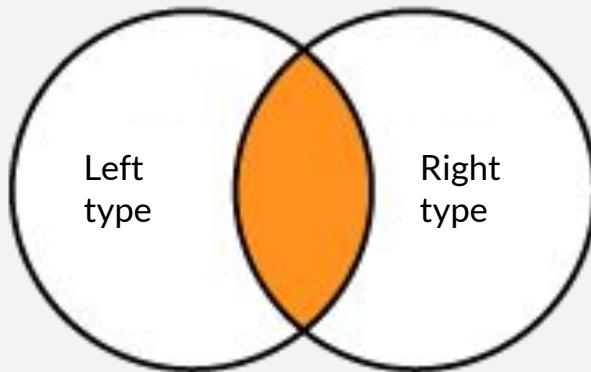


Windowed KStream-KStream

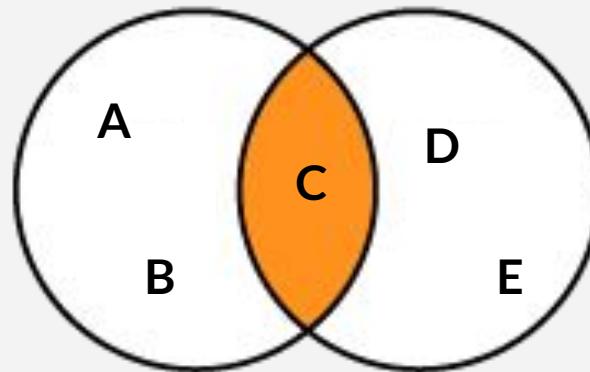
- ✗ KStream : unbounded, unlimited data
- ✗ KStream / KStream join need constraints
- ✗ Otherwise both KStream will be scanned when new data arrives -> **huge performance cost**
- ✗ Use window for constraint



Inner Join



example →

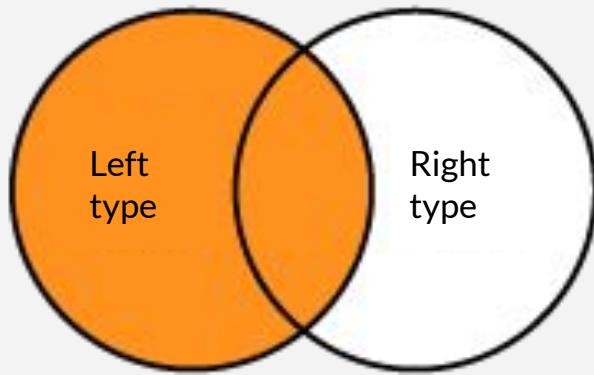


(left value, right value)

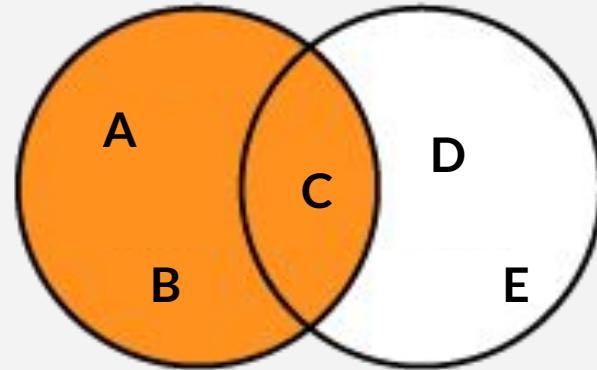
Result is:
(C, C)

Left (Primary)	Right (Secondary)
KStream	KStream
KTable	KTable
KStream	KStream
KStream	GlobalKTable

Left Join



example →

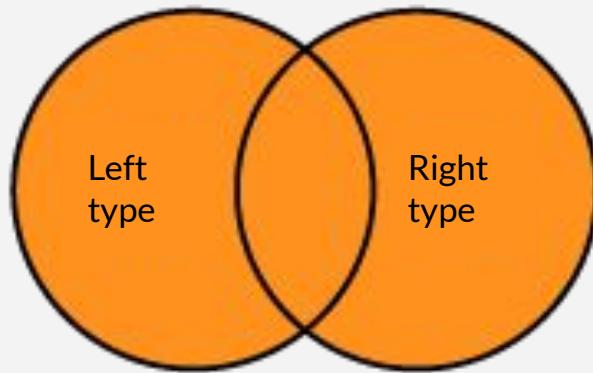


Left (Primary)	Right (Secondary)
KStream	KStream
KTable	KTable
KStream	KStream
KStream	GlobalKTable

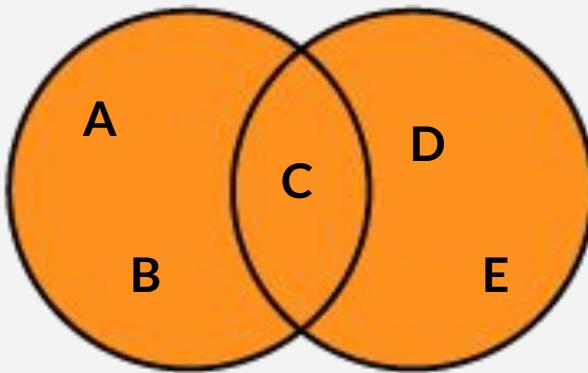
(left value, right value)

Result are:
(A, null)
(B, null)
(C, C)

Outer Join



example →



Left (Primary)	Right (Secondary)
KStream	KStream
KTable	KTable

(right value, left value)

Result are:
(A, null)
(B, null)
(C, C)
(null , D)
(null, E)

Join Stream / Stream

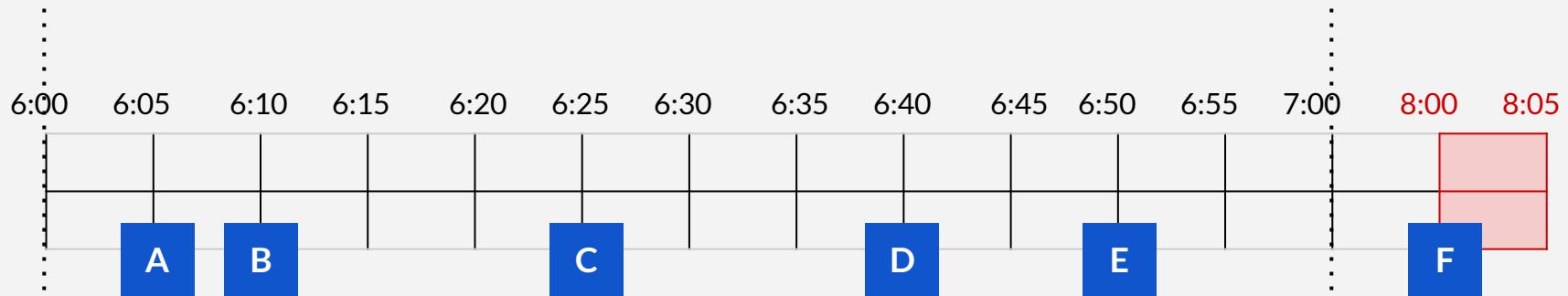


Join Window

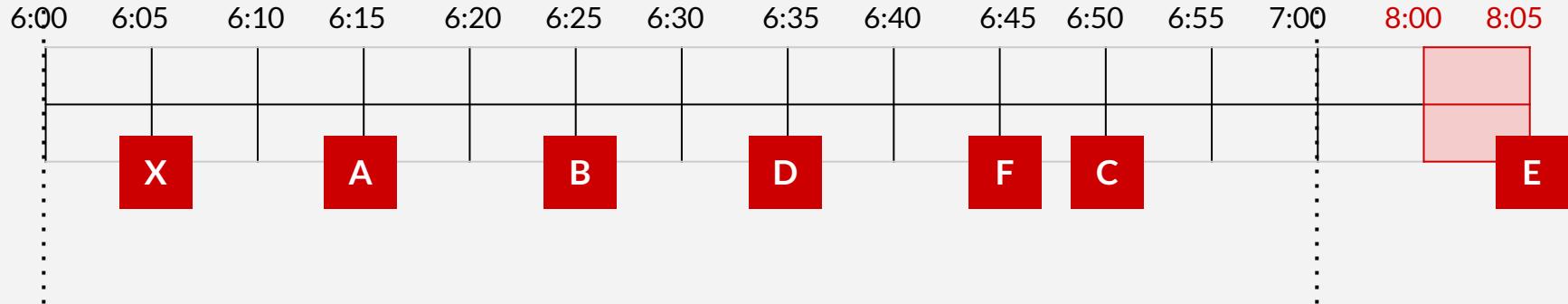
- × Class `JoinWindows`
- × Define maximum time difference
- × Record key X arrives on left stream at 7:15
 - × Matching record on right stream between 7:15 - 8:15
 - × 7:16 : match
 - × 7:50 : match
 - × 8:12 : match
 - × 8:17 : not match



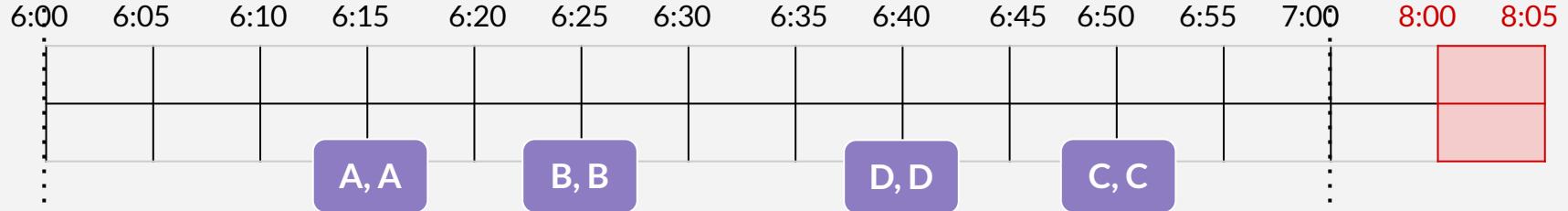
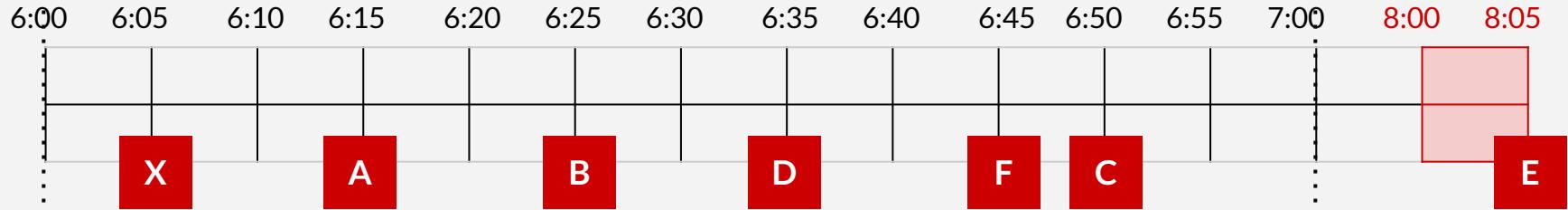
Input Streams



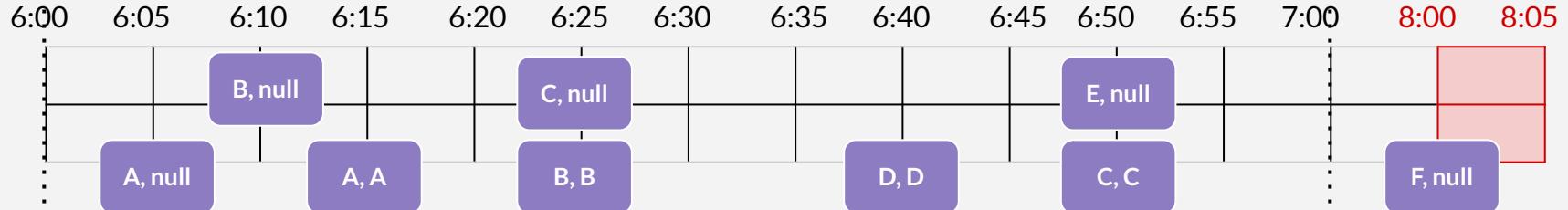
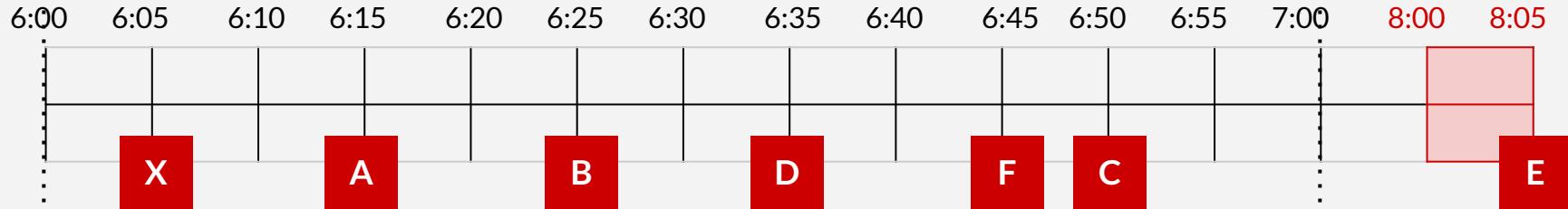
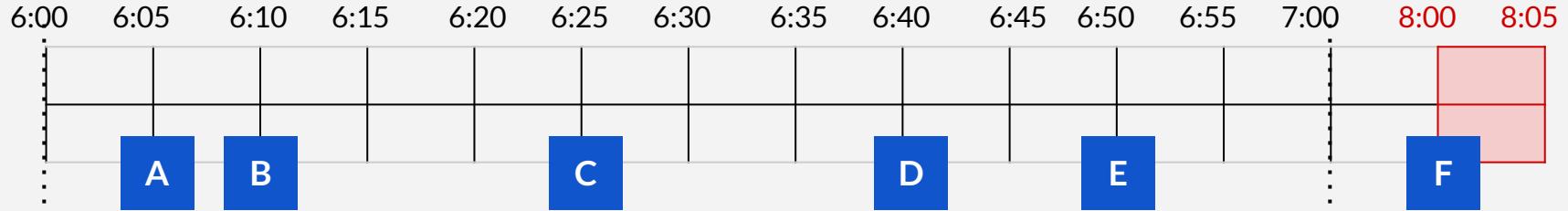
Time window : 1 hour



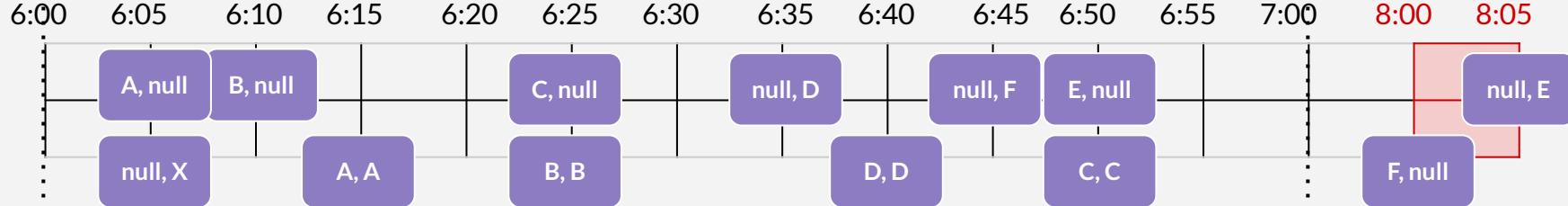
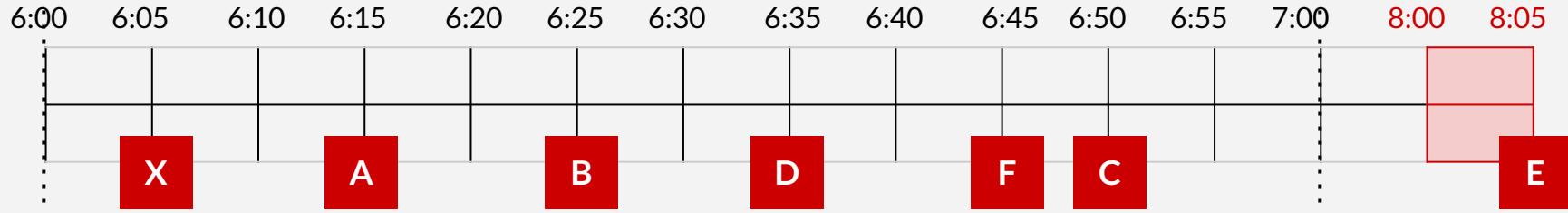
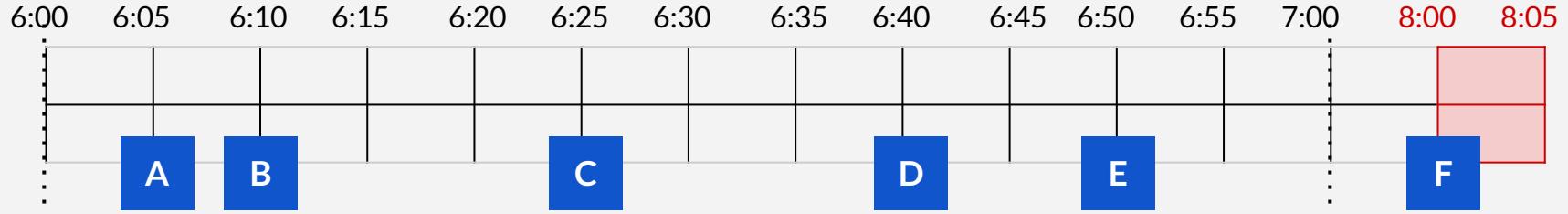
Inner Join Stream-Stream



Left Join Stream-Stream



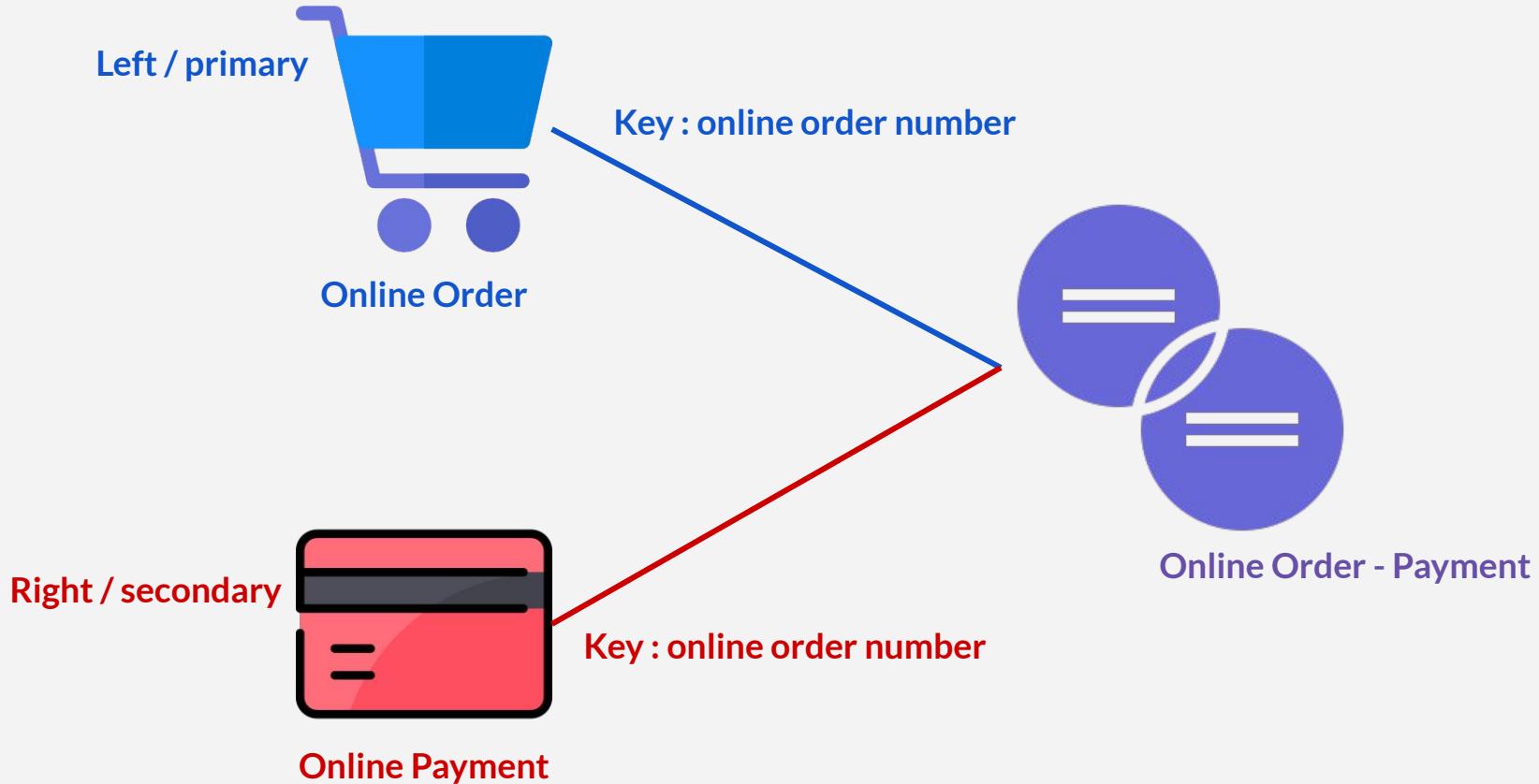
Outer Join Stream-Stream



Join Inner Join Stream / Stream



Online Order & Payment



Source Code for Online Order / Payment

- ✗ `OnlineOrder*.java`
- ✗ `OnlinePayment*.java`
- ✗ Package `com.course.kafka`
 - ✗ `api.request`
 - ✗ `api.server`
 - ✗ `broker.message`
 - ✗ `broker.producer`
 - ✗ `command.action`
 - ✗ `command.service`



Join Class

`OnlineOrderMessage.java`

1st class : left source stream

```
int a  
Int b
```



`OnlinePaymentMessage.java`

2nd class : right source stream

```
double x
```

`OnlineOrderPaymentMessage.java`

3rd class : join class

```
int a  
int b  
double x
```

join class example 2

```
int a  
double x
```

join class example 3

```
int a  
String someField  
double aPlusX()
```

Join Syntax

```
leftStream.join(rightStream, joiner, windows, joined)
leftStream.leftJoin(rightStream, joiner, windows, joined)
leftStream.outerJoin(rightStream, joiner, windows, joined)
```

```
// joiner

private JoinClass joiner(LeftClass left, RightClass right) { ... }
```

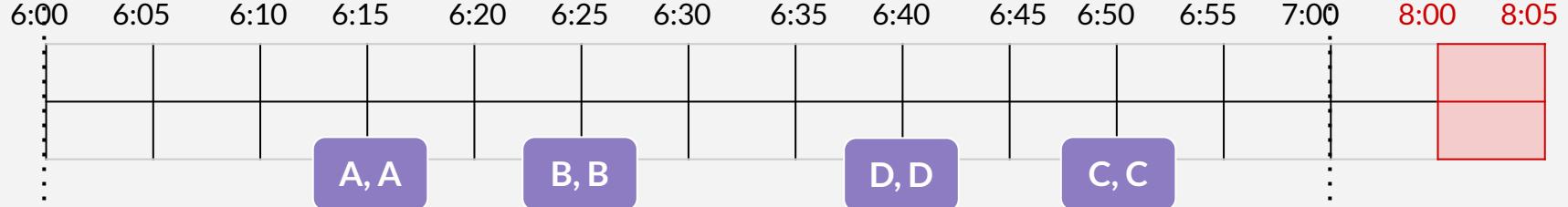
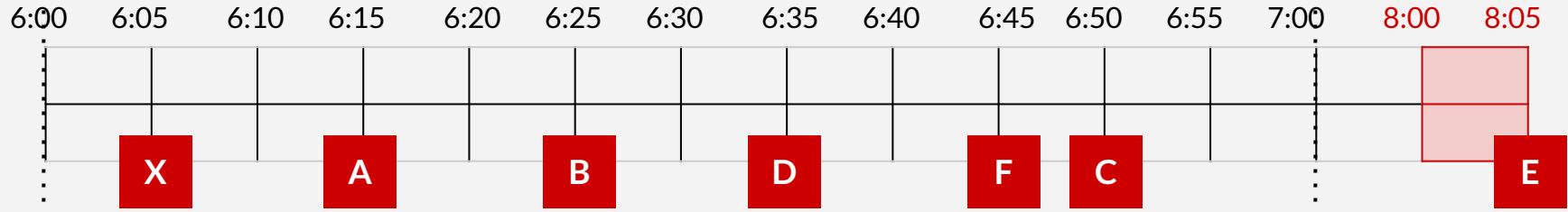
```
// windows

JoinWindows.of(...)
```

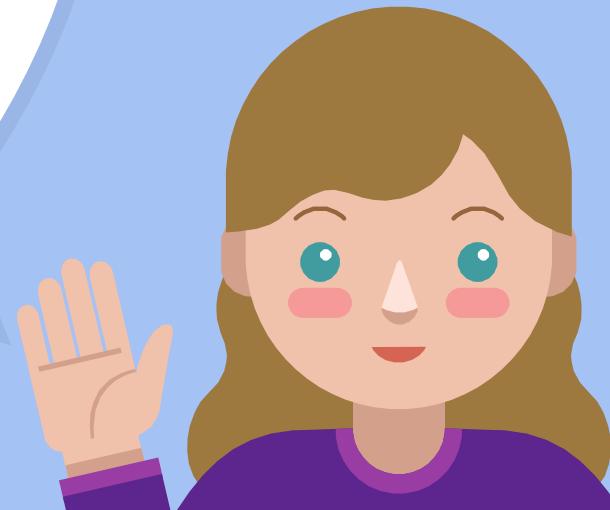
```
// joined

Joined.with(keySerde, leftSerde, rightSerde)
```

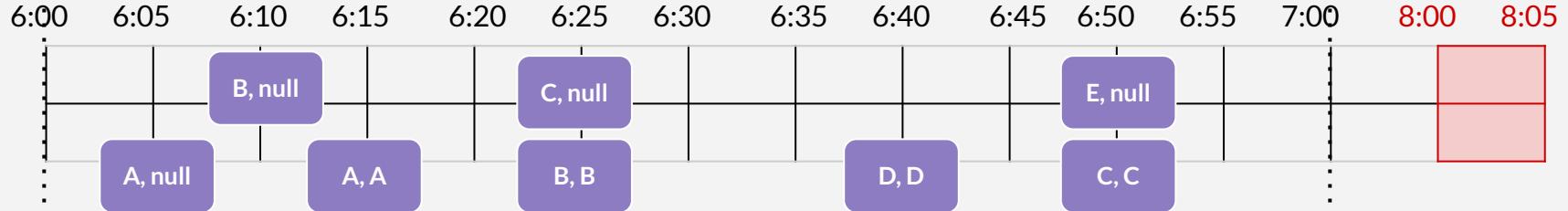
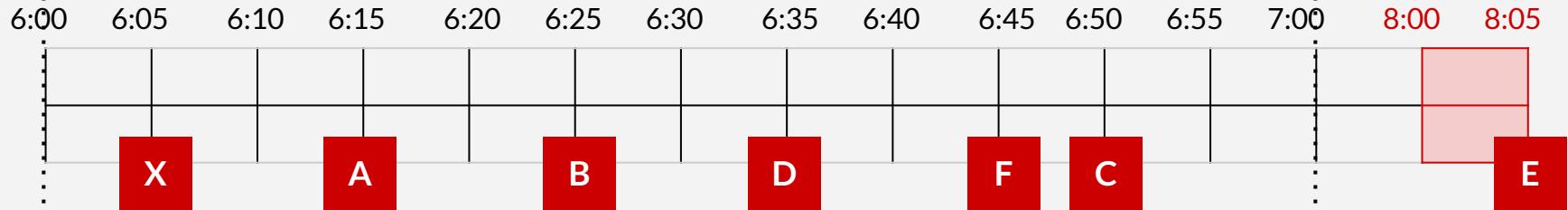
Inner Join Stream-Stream



Join Left Join Stream / Stream



Left Join Stream-Stream

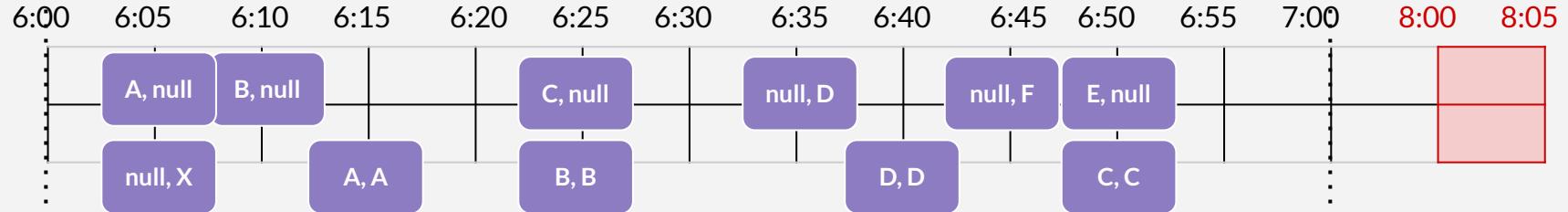
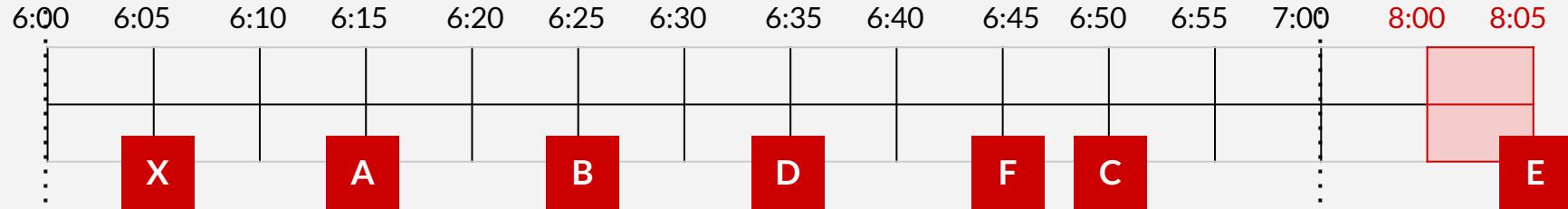


Join

Outer Join Stream / Stream



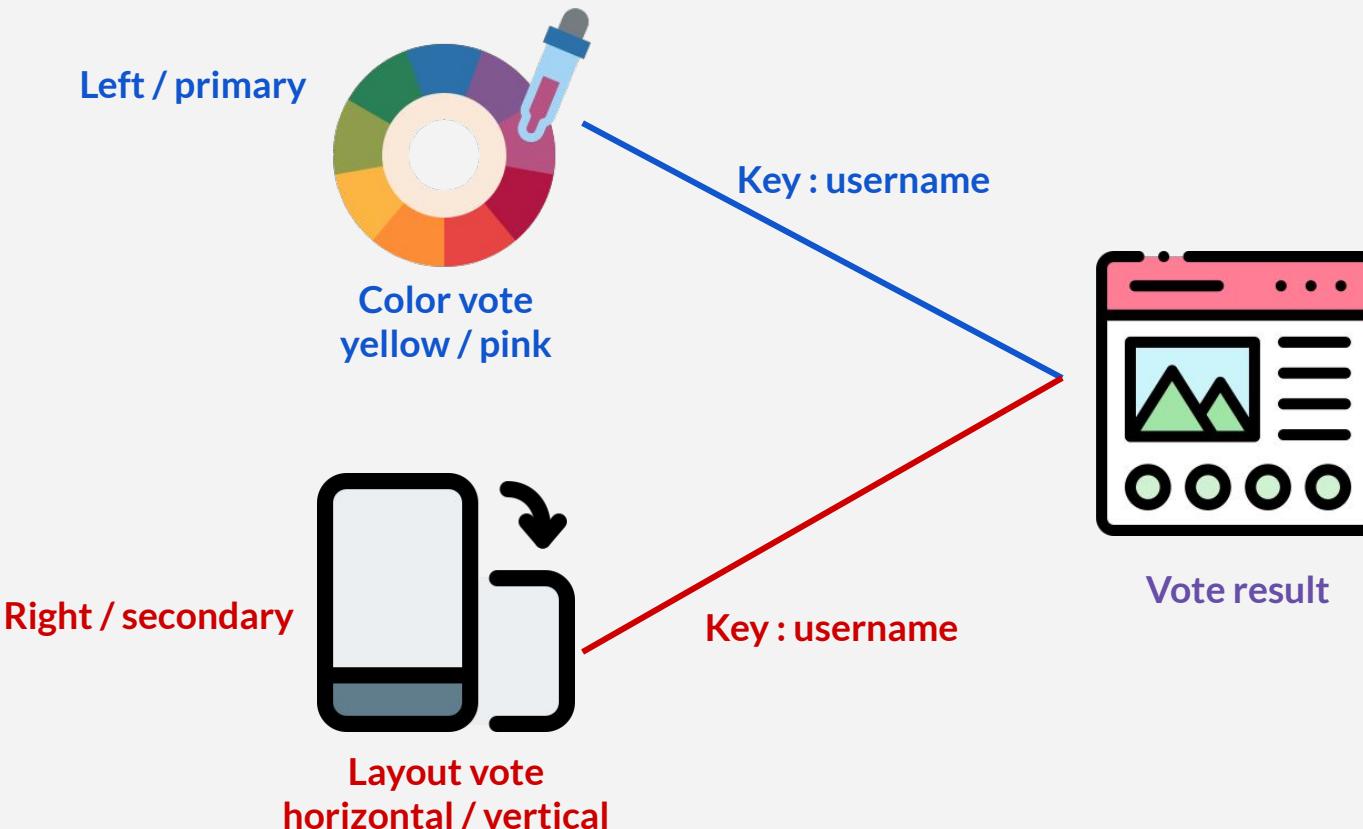
Outer Join Stream-Stream



Join Inner Join Table / Table



Web Color & Layout Vote



Source Code for Color / Layout Vote

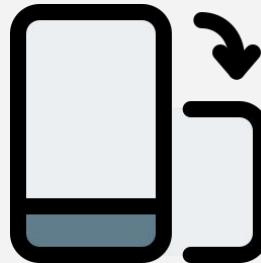
- ✗ `WebColorVote*.java`
- ✗ `WebLayoutVote*.java`
- ✗ Package `com.course.kafka`
 - ✗ `api.request`
 - ✗ `api.server`
 - ✗ `broker.message`
 - ✗ `broker.producer`
 - ✗ `command.action`
 - ✗ `command.service`



Inner Join



Color vote
yellow / pink



Layout vote
horizontal / vertical



Mandatory!



Mandatory!

Left Join



Color vote
yellow / pink



Layout vote
horizontal / vertical



Mandatory!

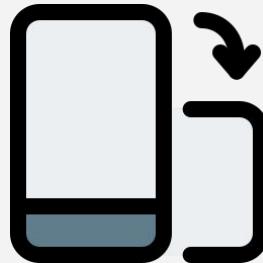


Optional

Outer Join



Color vote
yellow / pink



Layout vote
horizontal / vertical



Optional



Optional



8:05	piglet	pink	
8:25	eyore	yellow	
8:40	piglet		vertical
9:15	pooh	yellow	
9:30	tigger		vertical
9:50	piglet		horizontal
10:00	tigger	pink	
10:05	pooh		vertical
10:40	rabbit		horizontal
11:20	tigger	yellow	

timeline

piglet	pink	vertical	pink : 1, yellow : 0	hor : 0, ver : 1
tigger	pink	horizontal	pink : 1, yellow : 0	hor : 1, ver : 0
tigger	yellow	vertical	pink : 2, yellow : 0	hor : 1, ver : 1
pooh	yellow	vertical	pink : 2, yellow : 1	hor : 1, ver : 2
tigger	yellow	vertical	pink : 1, yellow : 2	hor : 1, ver : 2

Join Left Join Table / Table





8:05	piglet	pink	
8:25	eyore	yellow	
8:40	piglet		vertical
9:15	pooh	yellow	
9:30	tigger		vertical
9:50	piglet		horizontal
10:00	tigger	pink	
10:05	pooh		vertical
10:40	rabbit		horizontal
11:20	tigger	yellow	

timeline

piglet	pink		pink : 1, yellow : 0	hor : 0, ver : 0
eyore	yellow		pink : 1, yellow : 1	hor : 0, ver : 0
piglet	pink	vertical	pink : 1, yellow : 1	hor : 0, ver : 1
pooh	yellow		pink : 1, yellow : 2	hor : 0, ver : 1
piglet	pink	horizontal	pink : 1, yellow : 2	hor : 1, ver : 0
tigger	pink	vertical	pink : 2, yellow : 2	hor : 1, ver : 1
pooh	yellow	vertical	pink : 2, yellow : 2	hor : 1, ver : 2
tigger	yellow	vertical	pink : 1, yellow : 3	hor : 1, ver : 2

Join

Outer Join Table / Table





8:05	piglet	pink	
8:25	eyore	yellow	
8:40	piglet		vertical
9:15	pooh	yellow	
9:30	tigger		vertical
9:50	piglet		horizontal
10:00	tigger	pink	
10:05	pooh		vertical
10:40	rabbit		horizontal
11:20	tigger	yellow	

timeline



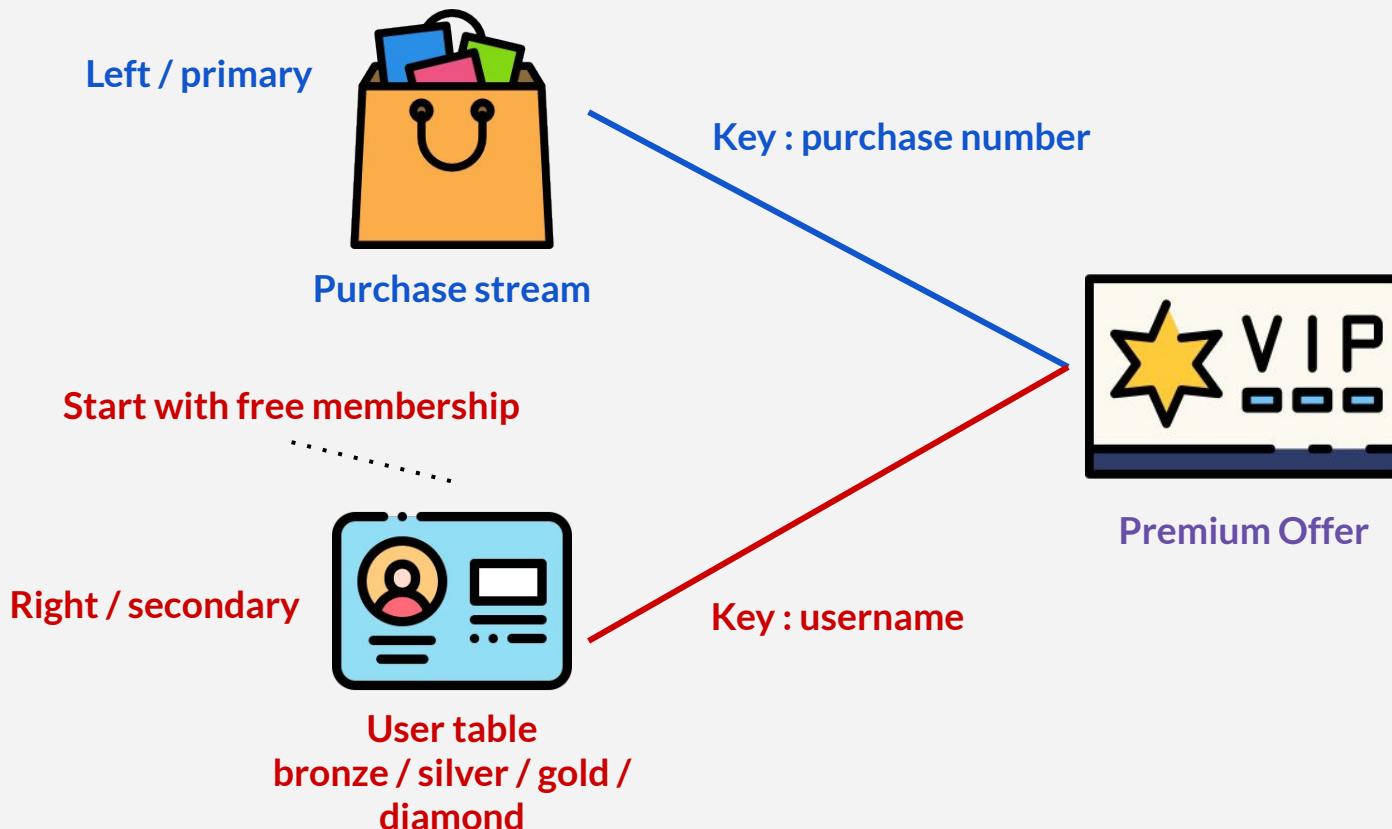
piglet	pink		pink : 1, yellow : 0	hor : 0, ver : 0
eyore	yellow		pink : 1, yellow : 1	hor : 0, ver : 0
piglet	pink	vertical	pink : 1, yellow : 1	hor : 0, ver : 1
pooh	yellow		pink : 1, yellow : 2	hor : 0, ver : 1
tigger		vertical	pink : 1, yellow : 2	hor : 0, ver : 2
piglet	pink	horizontal	pink : 1, yellow : 2	hor : 1, ver : 1
tigger	pink	vertical	pink : 2, yellow : 2	hor : 1, ver : 1
pooh	yellow	vertical	pink : 2, yellow : 2	hor : 1, ver : 2
rabbit		horizontal	pink : 1, yellow : 2	hor : 2, ver : 2
tigger	yellow	vertical	pink : 1, yellow : 3	hor : 2, ver : 2

Join

Inner Join Stream / Table



Premium Purchase & User



Source Code for Premium

- ✗ **Premium*.java**
- ✗ **Package com.course.kafka**
 - ✗ **api.request**
 - ✗ **api.server**
 - ✗ **broker.message**
 - ✗ **broker.producer**
 - ✗ **command.action**
 - ✗ **command.service**



Stream-Table Join Characteristic

- × Stream-Stream / Table-Table is **symmetric**
- × Stream-Table is **asymmetric**
 - × Only left input trigger join
 - × Right input only update table value



1	2	3	4	5	6	7	8	9	10	11	12	13	14
B 21	C 31			B 22	A 11			C 32	B 23		B 24	C 33	C 34

1	2	3	4	5	6	7	8	9	10	11	12	13	14
B, bronze C, gold		B, gold			C, silver			B, silver			A, gold C, diamond		
A, B, bronze C, gold		A, B, gold C, gold			A, B, gold C, silver			A, B, silver C, silver			A, gold B, silver C, diamond		

1	2	3	4	5	6	7	8	9	10	11	12	13	14
		C 31 gold		B 22 gold				B 23 gold				C 34 diamond	

Join

Left Join Stream / Table



1	2	3	4	5	6	7	8	9	10	11	12	13	14
	B 21	C 31		B 22	A 11		C 32	B 23		B 24	C 33		C 34

1	2	3	4	5	6	7	8	9	10	11	12	13	14

1	2	3	4	5	6	7	8	9	10	11	12	13	14
	B 21	C 31 gold		B 22 gold	A 11		C 32 silver	B 23 gold		B 24 silver	C 33 silver		C 34 diamond

Join Stream / GlobalKTable



Join Combinations

Left Type	Right Type	Co-Partition	Windowed	Join Types		
				Inner Join	Left Join	Outer Join
KStream	KStream	Required	Yes	Supported	Supported	Supported
KTable	KTable	Required	No	Supported	Supported	Supported
KStream	KTable	Required	No	Supported	Supported	Not Available
KStream	GlobalKTable	Not required	No	Supported	Supported	Not Available

KTable vs GlobalKTable

KTable

```
var table = builder.table("topic-input").filter(...)
```

GlobalKTable

```
// create intermediary stream
builder.stream("topic-input").filter(...).to("topic-intermediary");

// build global table
var globalTable = builder.globalTable("topic-intermediary");
```



KTable vs GlobalKTable

KTable

```
stream.join(table, joiner)
```

GlobalKTable

```
stream.join(globalTable, keySelector, joiner)
```

1	2	3	4	5	6	7	8	9	10	11	12	13	14
B 21	C 31			B 22	A 11			C 32	B 23	B 24	C 33		C 34

1	2	3	4	5	6	7	8	9	10	11	12	13	14
B, bronze C, gold		B, gold			C, silver			B, silver			A, gold C, diamond		
A, B, bronze C, gold		A, B, gold C, gold			A, B, gold C, silver			A, B, silver C, silver			A, gold B, silver C, diamond		

1	2	3	4	5	6	7	8	9	10	11	12	13	14
		C 31 gold		B 22 gold			C 32 gold	B 23 gold		B 24 gold	C 33 gold		C 34 diamond

1	2	3	4	5	6	7	8	9	10	11	12	13	14
B 21	C 31			B 22	A 11			C 32	B 23		B 24	C 33	C 34

1	2	3	4	5	6	7	8	9	10	11	12	13	14
B, bronze C, gold		B, gold			C, silver			B, silver			A, gold C, diamond		
A, B, C, gold		A, B, gold C, gold			A, B, gold C, gold			A, B, gold C, gold			A, gold B, gold C, diamond		

1	2	3	4	5	6	7	8	9	10	11	12	13	14
C 31 gold		B 22 gold				C 32 gold	B 23 gold		B 24 gold	C 33 gold		C 34 diamond	

GlobalKTable

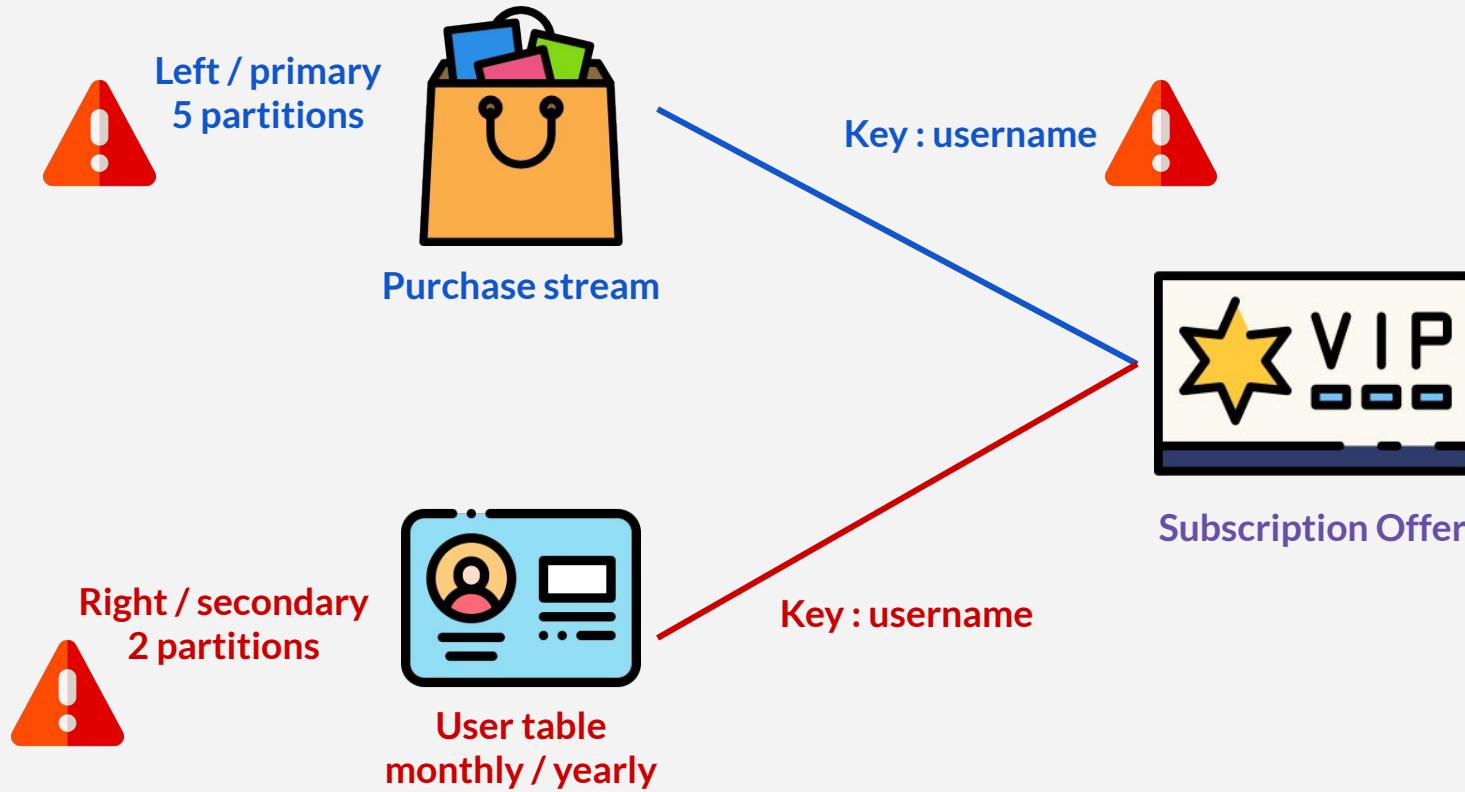
- ✗ Kafka global table is NOT database table
- ✗ Can't filter like SQL:
 - ✗ `SELECT key, value FROM kafka-topic WHERE ...`
- ✗ Latency compared to table
- ✗ Not suitable for frequent updates & a lot of data
- ✗ Frequent? A lot?
 - ✗ Many factors
 - ✗ Configure kafka correctly



Join Stream / Table Co-Partition



Subscription Purchase & User



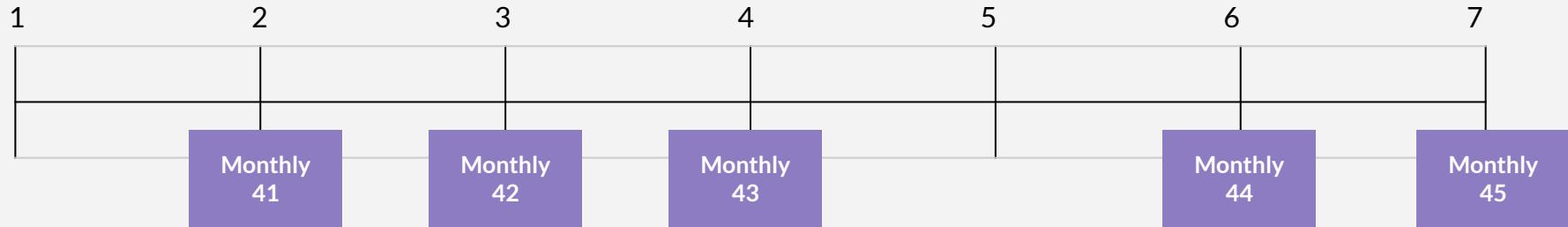
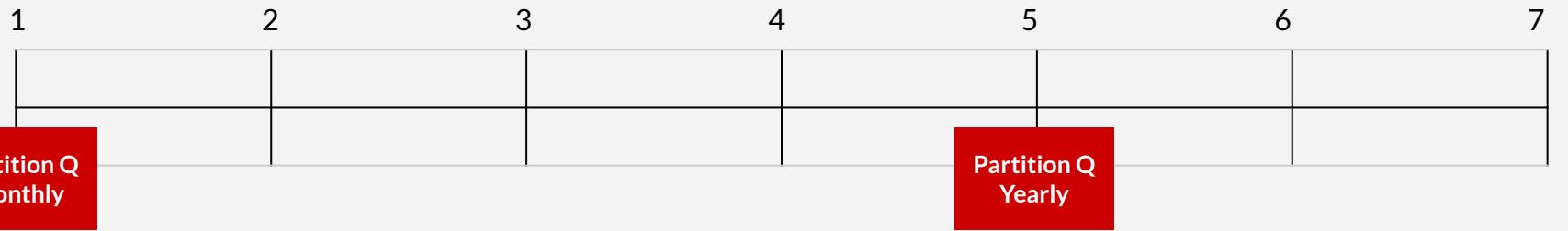
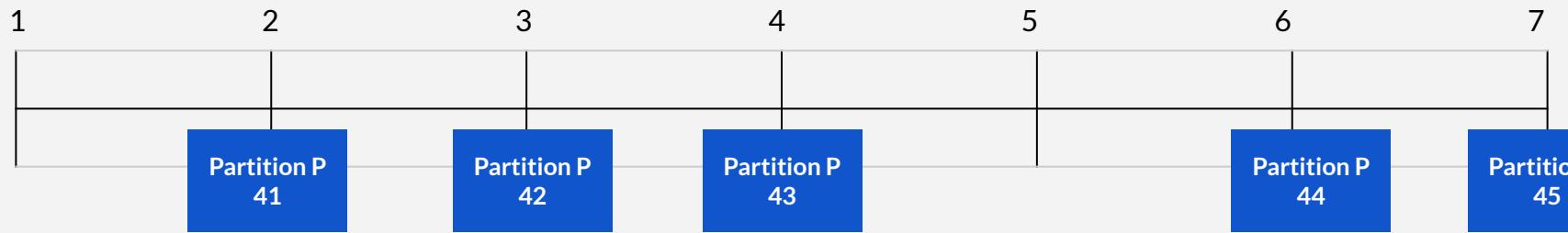
Source Code for Subscription

- ✗ `Subscription*.java`
- ✗ Package `com.course.kafka`
 - ✗ `api.request`
 - ✗ `api.server`
 - ✗ `broker.message`
 - ✗ `broker.producer`
 - ✗ `command.action`
 - ✗ `command.service`



Join Stream / GlobalKTable 2





Exactly Once Semantic



Delivery Semantic

- × At-most-once, at-least-once, exactly-once
- × Kafka producer or consumer
 - × Configure producer / consumer properties
- × Currently not in course
- × Overview on exactly-once

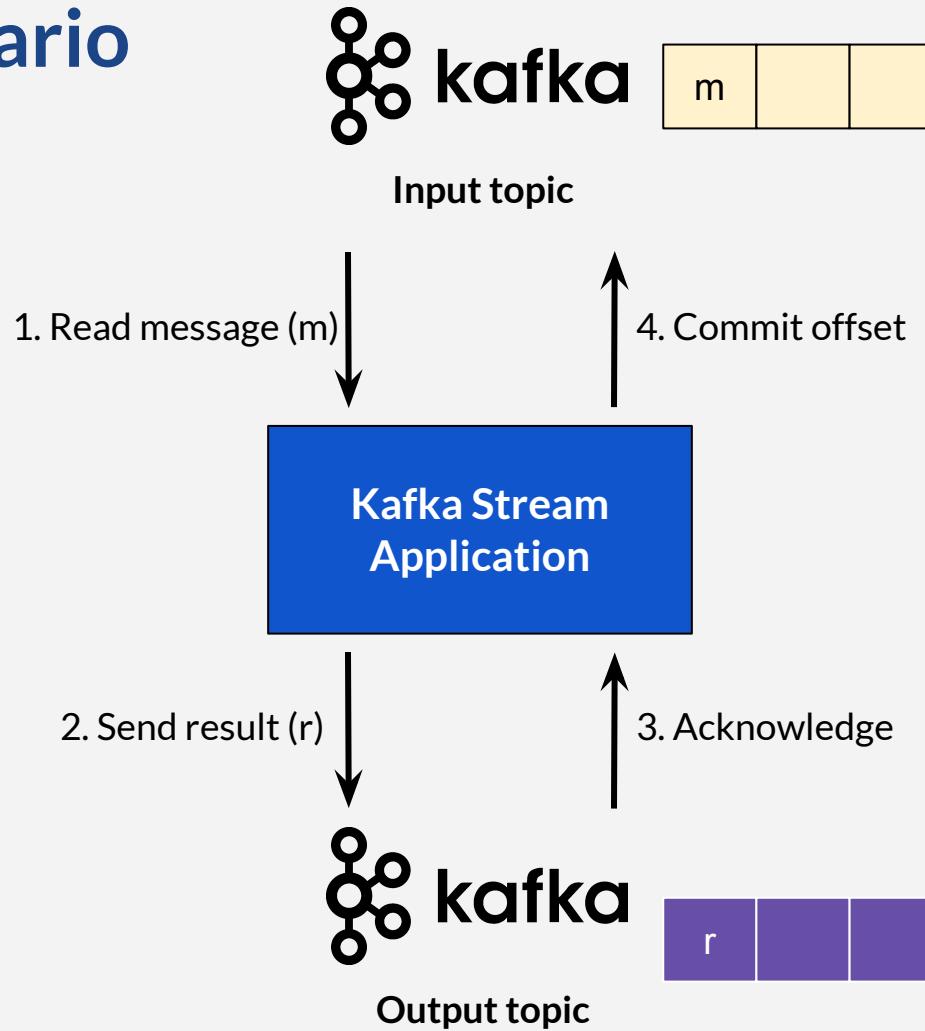


Exactly Once Semantic

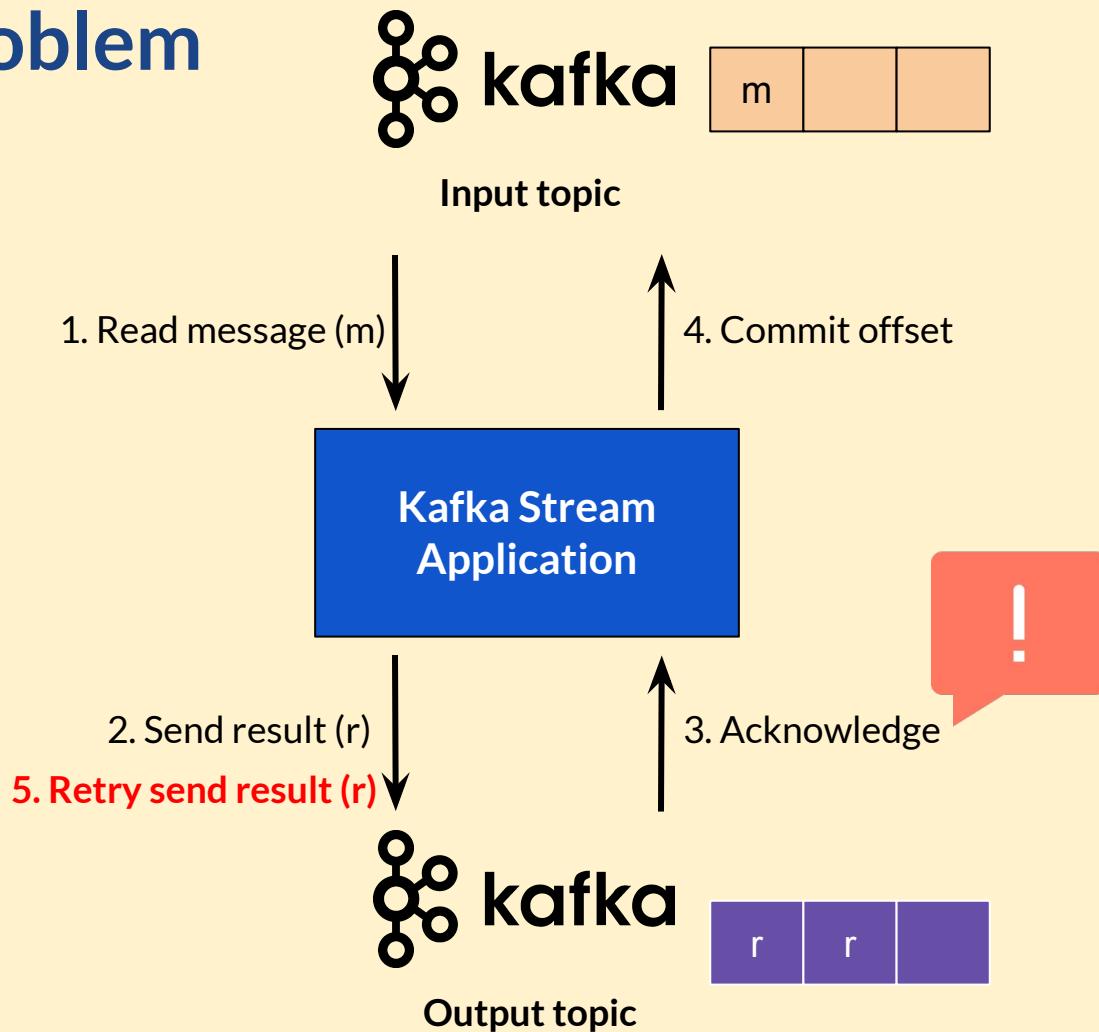
- × Produced only one copy, and consume only once
- × Retry on producer (send multiple times) can still be happened
- × Kafka will deduplicate messages
- × Kafka as input and output, without involving external system (database, text file, etc)
- × Match for kafka stream



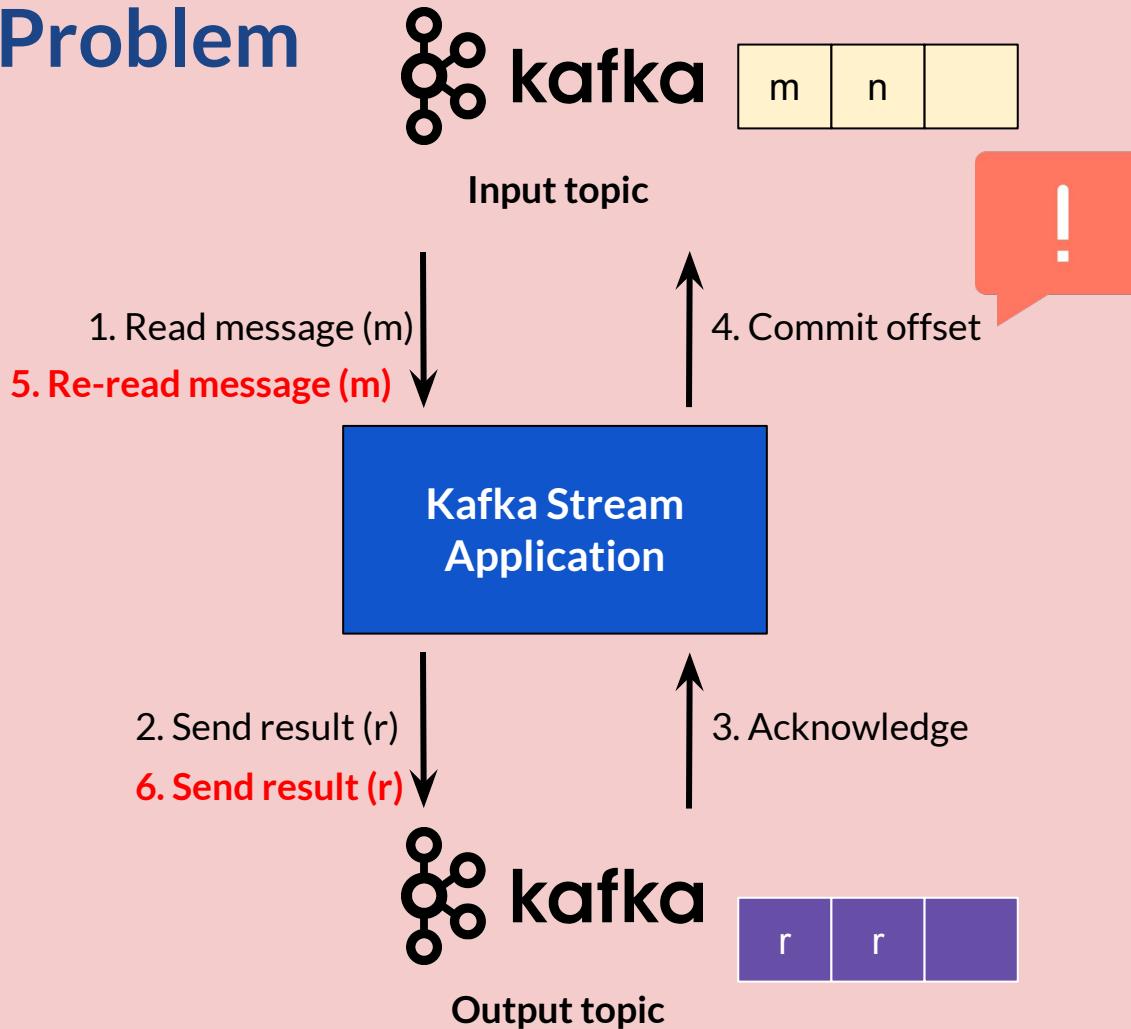
Happy Scenario



Publish Problem



Consume Problem



Exactly Once in Kafka

- × Idempotent producer
 - × Idempotent?
 - × Kafka producer will make sure only one copy of message exists on topic
- × Atomic transaction
 - × Several operations on kafka stream cycle
 - × Happened all-or-none
 - × Like commit-rollback on database



Enabling Exactly Once



Enabling Exactly Once

- × Kafka version ≥ 0.11
- × Use library that support exactly once (**Kafka Stream**)
- × Add configuration on properties
- × No additional code

