Real Time Investment Alerts using Apache Kafka at ING Bank

GOTO AMSTERDAM – JUNE 2019

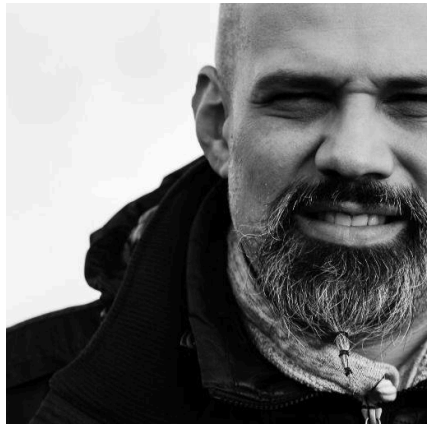
Marcos Maia & Tim van Baarsen

ING

Introduction

Tim van Baarsen
Software Engineer @ ING

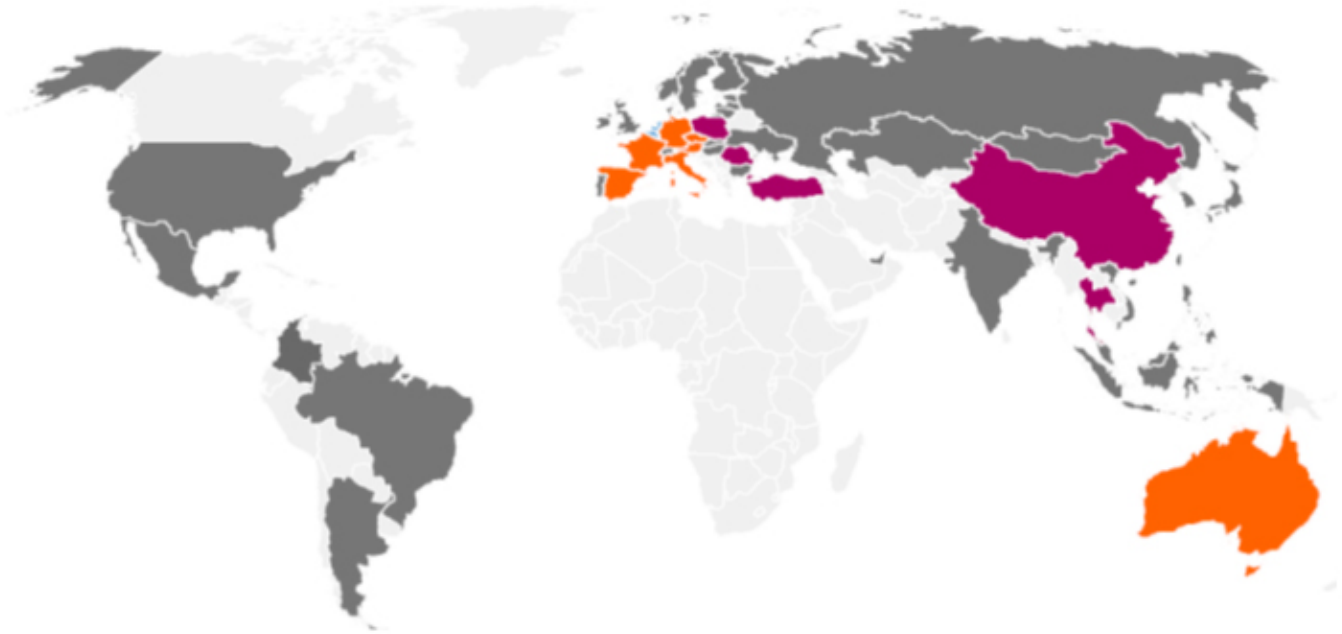
@TimvanBaarsen



Marcos Maia
Software Engineer @ ING

@thegroo

ING is active in more than 40 counties



Disclaimer: Please note that ING Bank does not have a banking license in the US and is therefore not permitted to conduct banking activities in the US.

https://www.ing.jobs

Agenda

- Use case @ ING
  - Mifid II
  - Mobile Investments App
- Overall Architecture
- Demo
- Metrics / Monitoring
- Kafka streams
- Local development / Code walk through
- Lessons learned
- Questions

Disclaimer

Both the **code** and high level **architecture** we show here today is

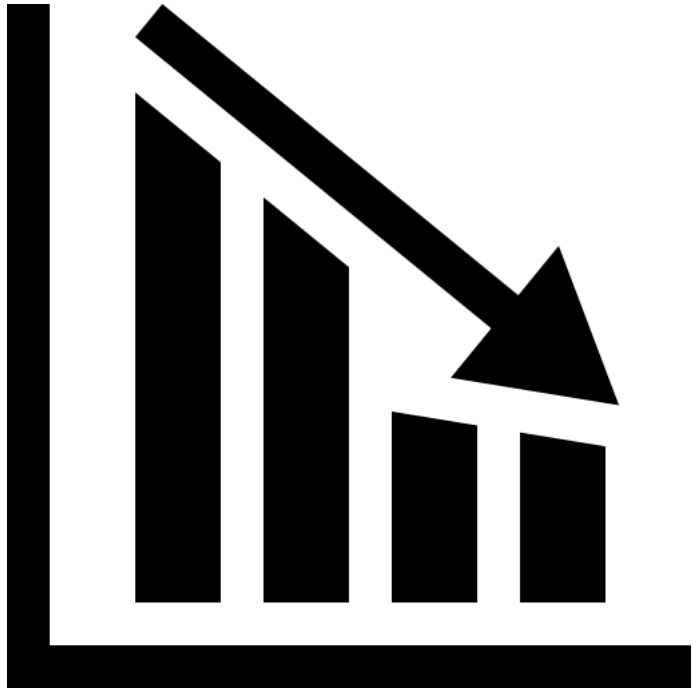
not the actual product we developed within ING!

This example is inspired by our work

Domain Lingo

Tick = Price update on stock

Mifid II: “Inform customers when a stock drops -10% in price at the end of the day!”

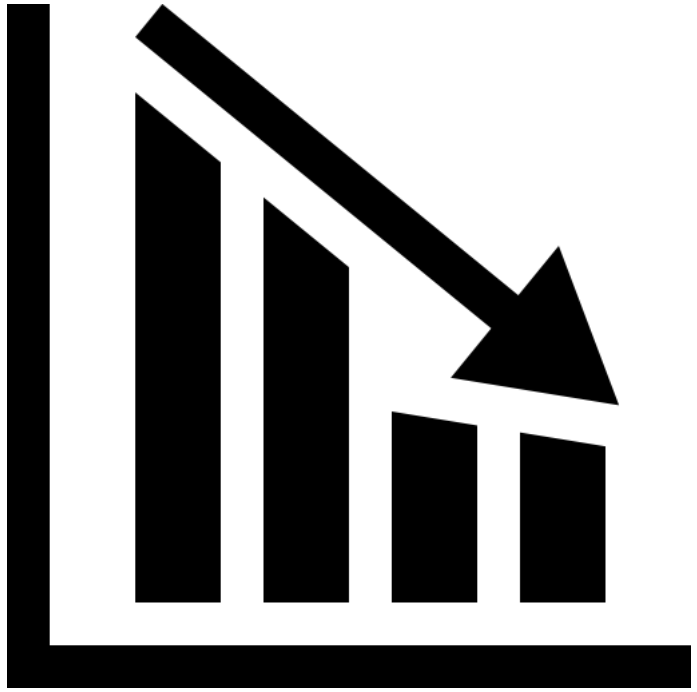


At the end of the day?

Markets **i**n **F**inancial **I**nstruments **D**irective **II** =
Regulatory reform financial markets European Union came into effect on January 3, 2018

ING

Mifid II: “Inform customers when a stock drops -10% in price at the end of the day!”

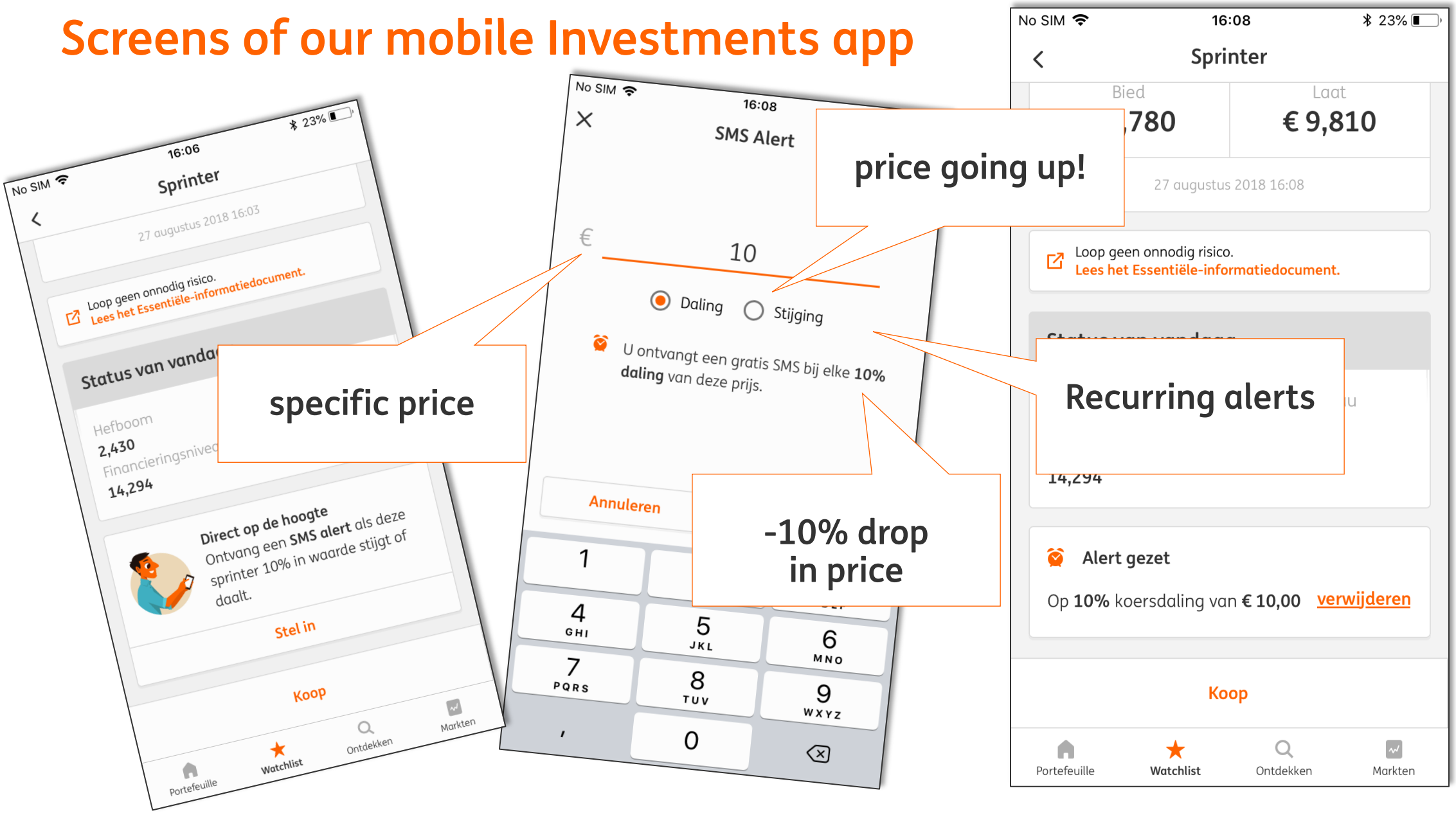


No, in real time!

Markets **i**n **F**inancial **I**nstruments **D**irective **II** =
Regulatory reform financial markets European Union came into effect on January 3, 2018

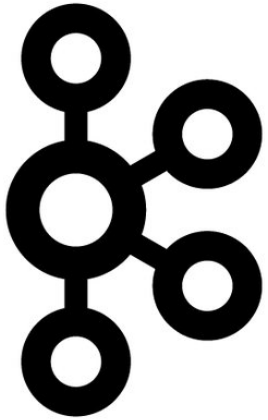
ING

Screens of our mobile Investments app

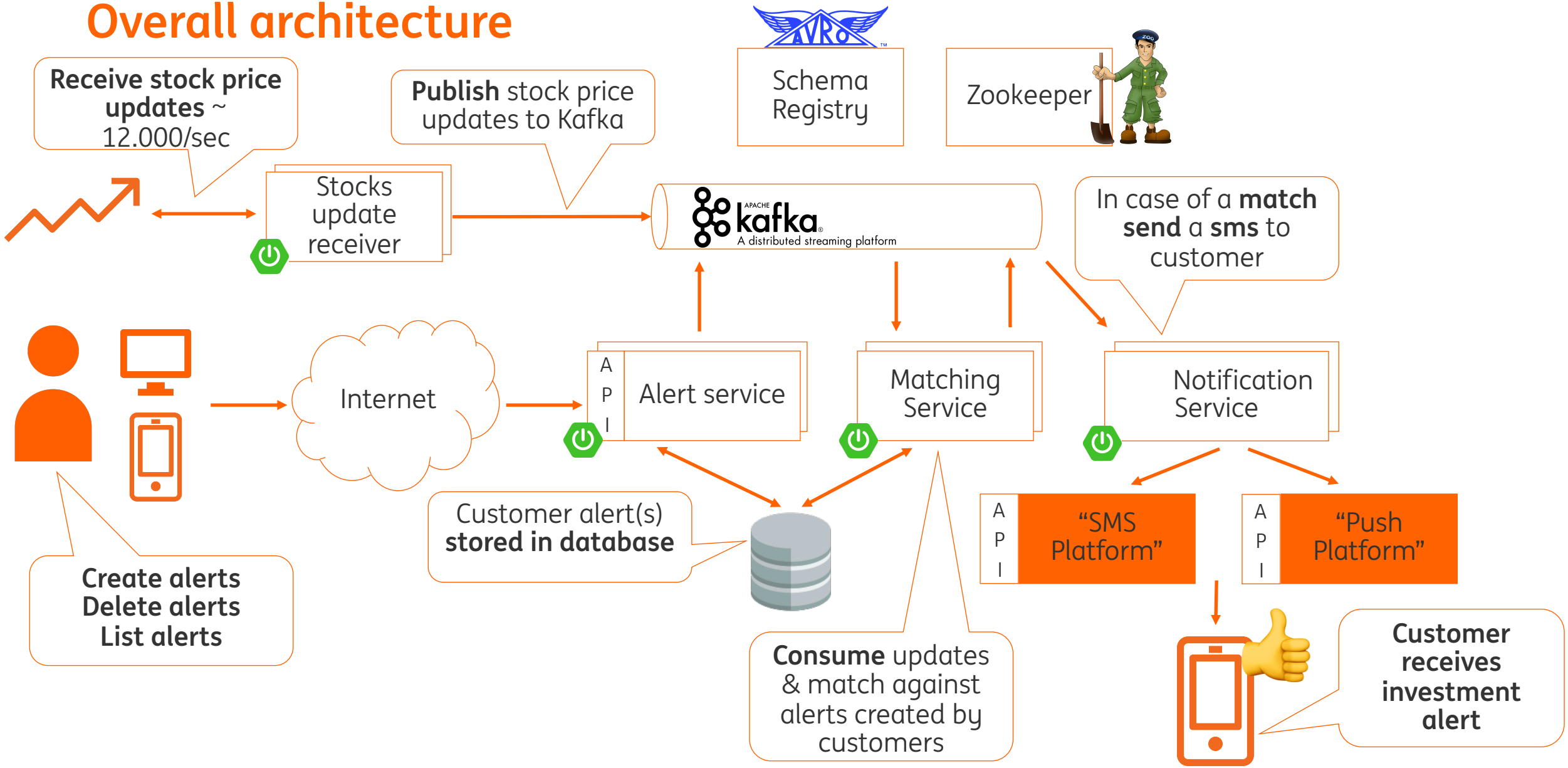


2.000 – 12.000 stock updates a second!

The perfect “match”



Overall architecture



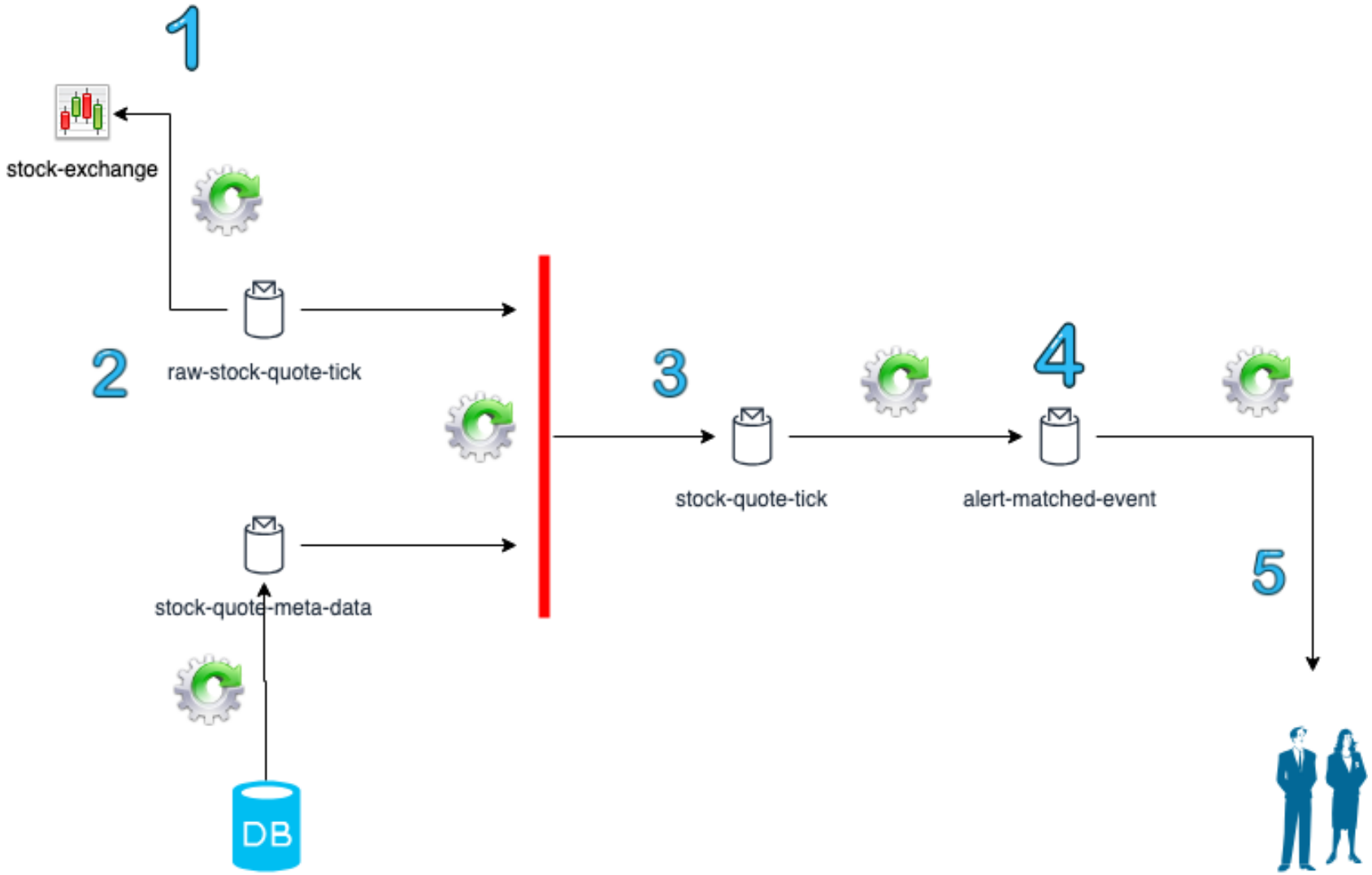
Quick Demo

ING

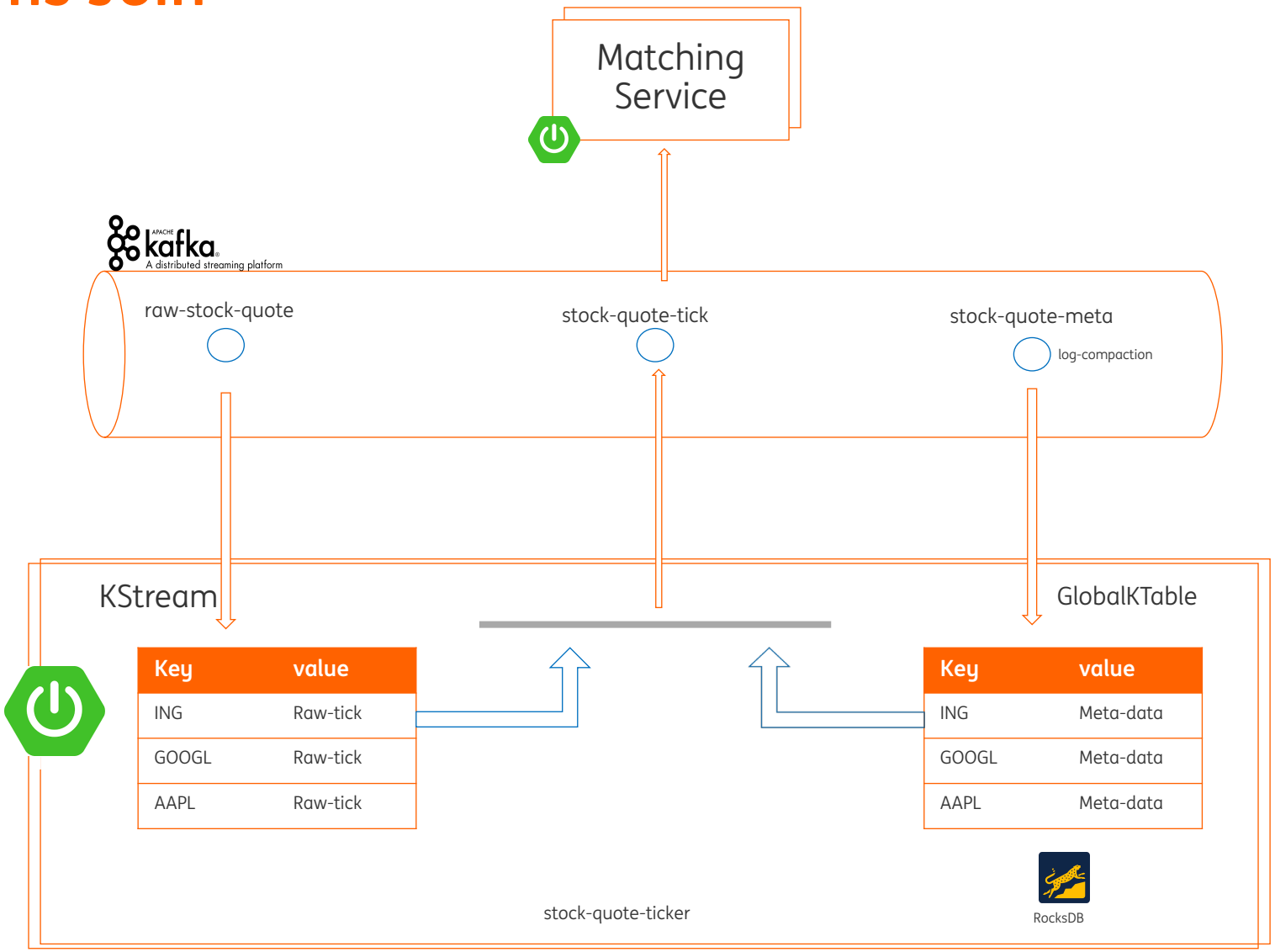
Monitoring - Metrics

- **Micrometer**
  - Think SLF4J, but for metrics (Jon Schneider)
  - Application metrics
  - Out of the box metrics
  - Business metrics
  - Vendor Neutral

Monitoring – Real-time?



Kafka Streams Join



ING

Local development

- **Docker**
  - Kafka
  - Zookeeper
  - Kafka manager
  - All our Spring Boot applications
- **Producers for development**
  - Custom Kafka command line producer
- **Lab project**

Spring Kafka

- **Producing**:
  - ProducerFactory
  - KafkaTemplate

  - *"The producer is thread safe and sharing a single producer instance across threads will generally be faster than having multiple instances."*

- **Consuming**
  - Kafka consumers are not thread safe
  - @KafkaListener
  - MessageListener (many different interfaces)
  - Access to ConsumerRecord
  - Access to manual control to commit the offset
  - Scale up multiple consumer (threads) within the application

Spring Kafka

- **Integration tests**
  - Junit rule: EmbeddedKafkaRule
  - @EmbeddedKafka
- **Metrics**
  - Kafka consumer metrics out of the box (**Micrometer**)

Lessons learned

- Use **Spring Kafka** after you understand the Kafka APIs

- Error handling
  - (De)serialization -> **Apache Avro**

- Think about # of **partitions** per topic upfront if possible (routing)
  - Can your consumers keep up? Can you scale up?

- Monitor your topics / streams in production
  - Based on metrics -> **Micrometer**

Lessons learned

- Kafka clients are backward compatible with Kafka brokers

- Kafka Streams
  - KTables
  - GlobalKTables
  - Interactive Queries
  - RocksDB build for ssd
  - Describe
  - processing.guarantee=exactly_once

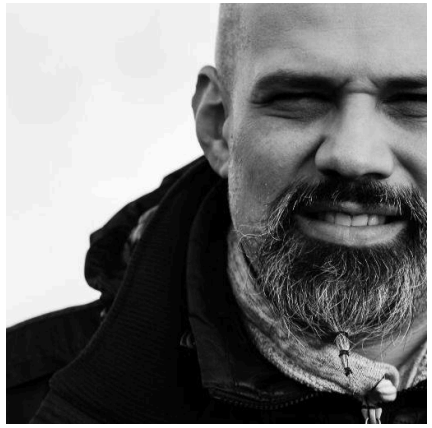
- Kafka topology
  - Plan ahead
  - small & multiple topologies

Questions

Contact

Tim van Baarsen
Software Engineer @ ING

@TimvanBaarsen



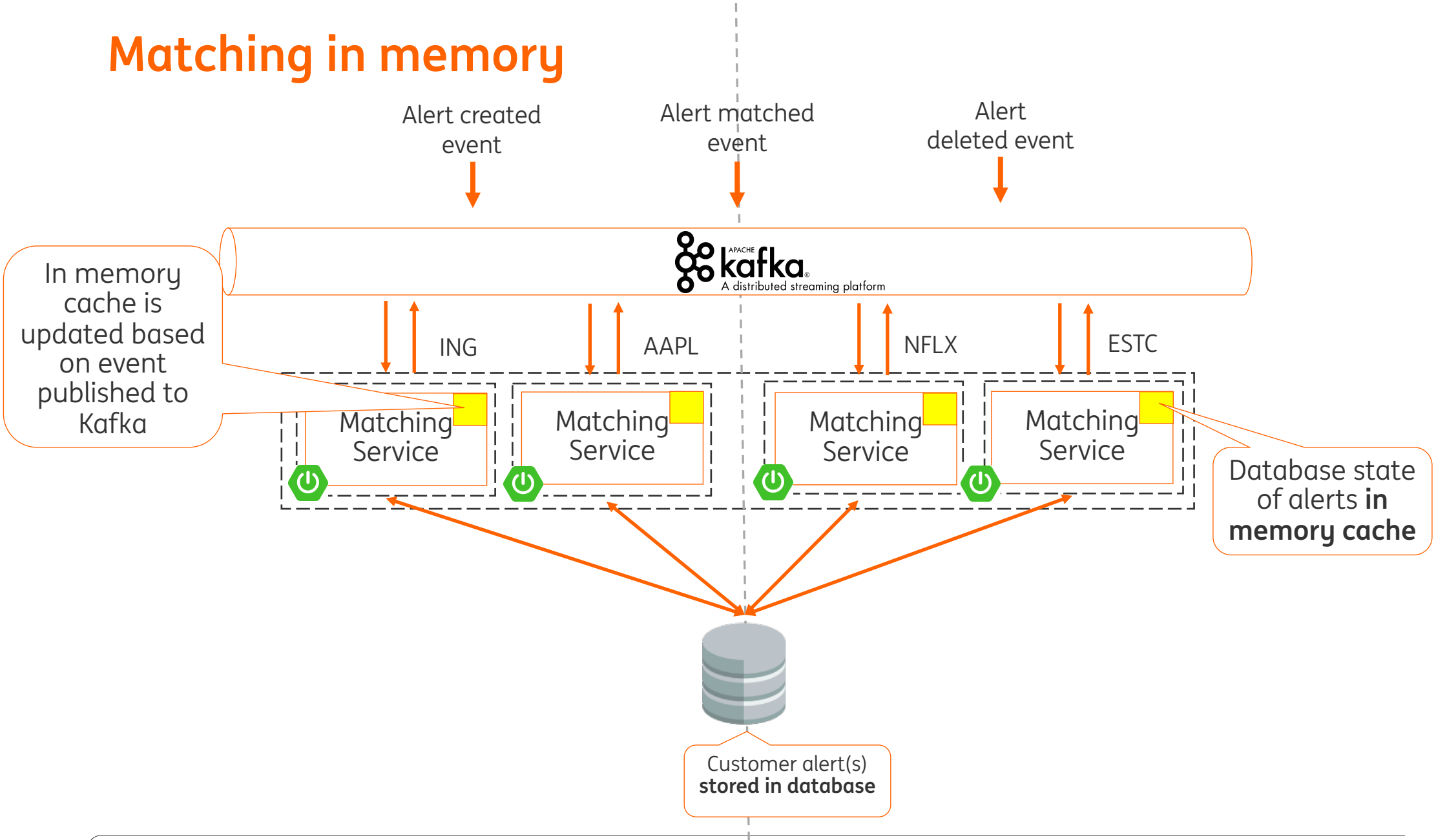
Marcos Maia
Software Engineer @ ING

@thegroo


Thank you!

Backup slides

Matching in memory



Kafka Topology