Kafka

Jiří Šebek *b6b36nss*

Co je Kafka?

- distributed streaming platform
  - From distributed message broker to a platform for processing data streams.
- Apache distribuce
- umožňuje posílat velké množství malých zpráv napříč servery
- přičemž umožňuje horizontální škálování a zároveň všechny zprávy replikuje na více serverů – **vypadne-li jeden z nich, jiný ho nahradí**

K čemu se používá ?

- logování všech zpráv, které v systému vznikají a které chceme nějak zpracovávat

Co potřebujeme ?

- Javu :) (například jdk 8)
- Kafku :
  - https://www.apache.org/dyn/closer.cgi?path=/kafka/1.0.0/kafka_2.11-1.0.0.tgz

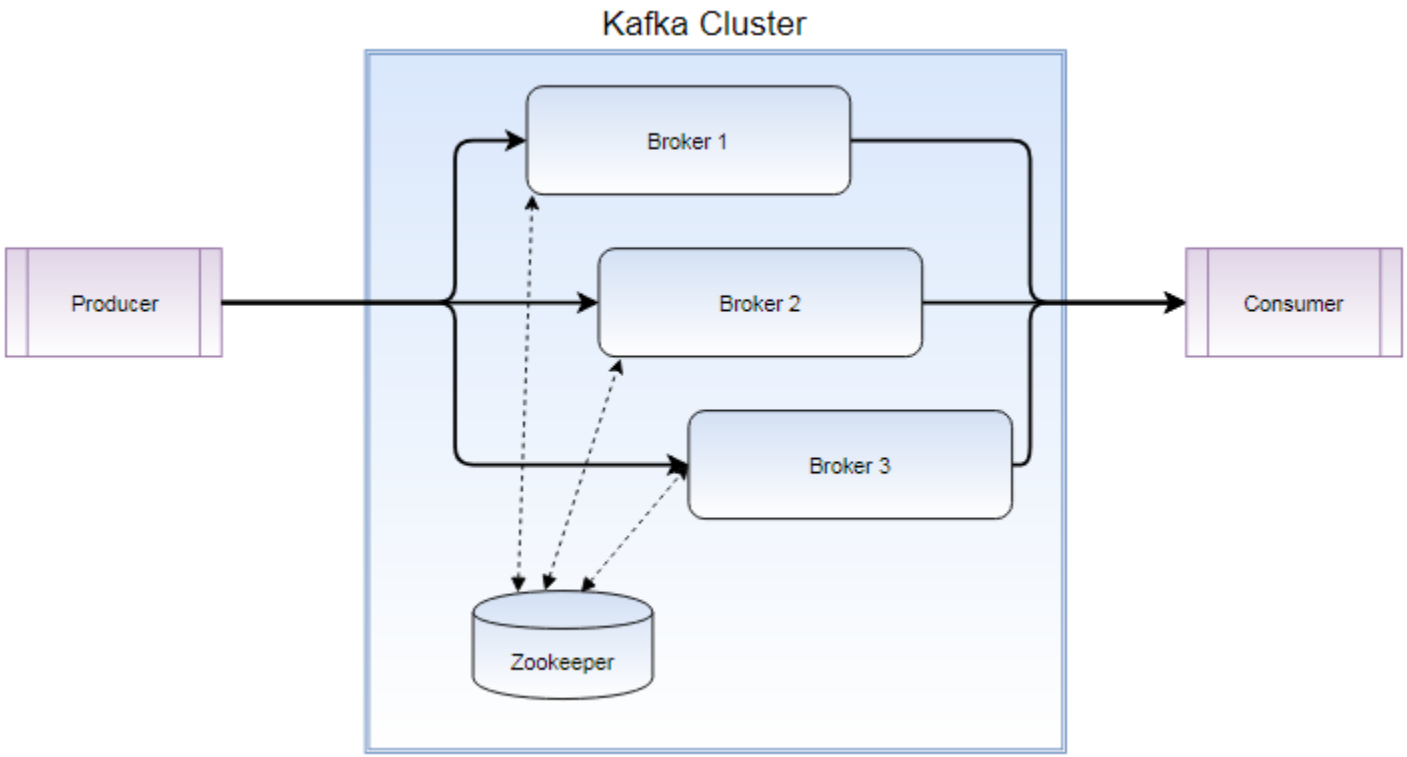
Instalace

- Otevřete kdekoli na počítači:)
  - Pozor jestli bude v ceste ke složce mezera tak budete mít problemy (pro ukázku rozbalte na ploše)
  - Linux :
    - tar -xvzf kafka_2.11-1.0.0.tgz
  - Windows :
    - Pomocí winzip/winrar apod.

Architektura

- Zookeeper : Which is used by Kafka to maintain state between the nodes of the cluster
- Kafka brokers : The “pipes” in our pipeline, which store and emit data
- Producers : That insert data into the cluster
- Consumers : That read data from the cluster

Architektura



Zookeeper

- Otevřeme si složku *kafka_2.11-1.0.0*
- *bin/zookeeper-server-start.sh config/zookeeper.properties*

Starting our brokers

- *Najděte si soubor : config/server.properties*
  - *Najděte si tyto properties :*
    - *broker.id=0*
    - *listeners=PLAINTEXT://:9092*
    - *log.dirs=/tmp/kafka-logs*
- *Vytvořte z originálního souboru 3 soubory :*
  - *config/server.1.properties*
  - *config/server.2.properties*
  - *config/server.3.properties*

Starting our brokers

- *Nahraďte properties v těchto souborech :*

server.1.properties
broker.id=1
listeners=PLAINTEXT://:9093
log.dirs=/tmp/kafka-logs1

server.2.properties
broker.id=2
listeners=PLAINTEXT://:9094
log.dirs=/tmp/kafka-logs2

server.3.properties
broker.id=3
listeners=PLAINTEXT://:9095
log.dirs=/tmp/kafka-logs3

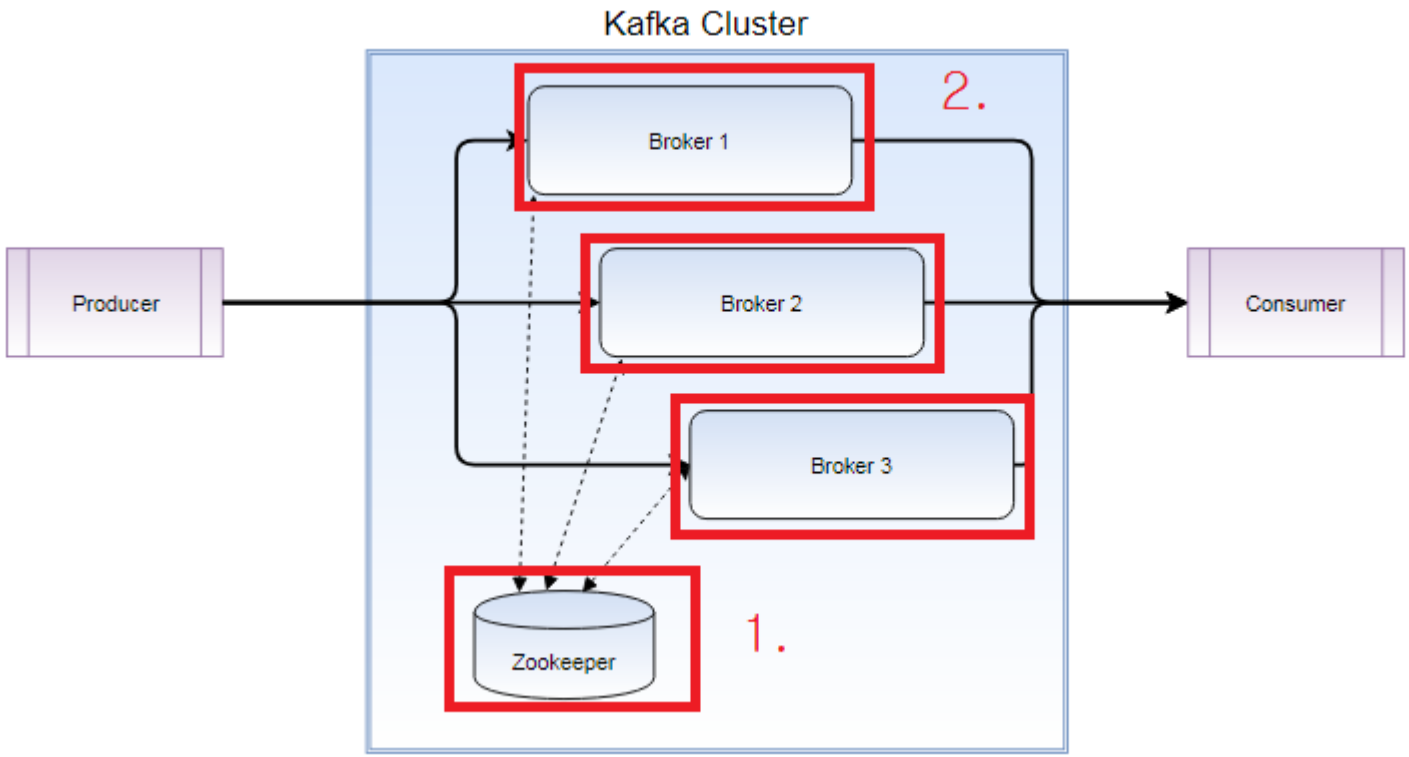
Starting our brokers

- *Vytvořte složky pro logy :*
  - */tmp/kafka-logs1*
  - */tmp/kafka-logs2*
  - */tmp/kafka-logs3*

Starting our brokers

- *Nastartujte brokery :*
  - *bin/kafka-server-start.sh config/server.1.properties*
  - *bin/kafka-server-start.sh config/server.2.properties*
  - *bin/kafka-server-start.sh config/server.3.properties*

Co máme hotovo ?



Topic

- *Vytvoříme si topic: (sdílená data)*
  - *bin/kafka-topics.sh --create --topic my-kafka-topic --zookeeper localhost:2181 --partitions 3 --replication-factor 2*
- *Důležité parametry:*
  - *Paritions – mezi kolik brokers se rozdělí data*
  - *Replication-factor – kolik kopií dat chceme*

Producer

- *Vytvoříme si producenta:*
  - *bin/kafka-console-producer.sh --broker-list localhost:9093,localhost:9094,localhost:9095 --topic my-kafka-topic*
- *Uvidíme command line do které můžeme zadat nějaké zprávy*

Consumer

- *Vytvoříme si consumer:*
  - *bin/kafka-console-consumer.sh --bootstrap-server localhost:9093 --topic my-kafka-topic --from-beginning*

Use case 1

- *Zabijte jednoho brokera pomocí ctrl+c v console*
- *Cluster je pořád funkční*
  - *Kafka toleruje : « node failes »*
- *Otevřete dalšího cosumera*
  - *bin/kafka-console-consumer.sh --bootstrap-server localhost:9093 --topic my-kafka-topic --from-beginning --group group2*
  - *Vidíme všechny zprávy*

Homework

- *Se svým nastavením si můžete zahrát mnohem víc.*
  - *Co se stane, když spadne jiný broker?*
  - *Co kdybyste měli 5 brokerů a 2 spadnou?*
  - *Co když jste změnili replikační faktor pro váš topic?*

- *Nejlepší způsob, jak vědět, jak pružná Kafka je, je experimentovat :)*

Integrace springboot

- *Dependency v pom.xml :*

```
<dependency>
    <groupId>org.springframework.kafka</groupId>
    <artifactId>spring-kafka</artifactId>
    <version>2.2.3.RELEASE</version>
</dependency>
```

Integrace springboot

- *Minimálně musíte mít zapnuté :*
  - *Zookeeper :*

    zookeeper-server-start.bat .\config\zookeeper.properties
- *Kafka cluster*
  - *kafka-server-start.bat .\config\server.properties*
    - *default, localhost:9092*

Integrace springboot

- *Vytvořte controller s 1 endpointem :*

```
@RestController
@RequestMapping(value = "/kafka")
public class KafkaController {
private final Producer producer;
@Autowired
public KafkaController(Producer producer) {
this.producer = producer;
}
@PostMapping(value = "/publish")
public void sendMessageToKafkaTopic(@RequestParam("message") String message){
this.producer.sendMessage(message);
}
}
```

Integrace springboot

- *Vytvořte producenta :*

```
@Service
public class Producer {
private static final Logger logger = LoggerFactory.getLogger(Producer.class);
private static final String TOPIC = "users";
@Autowired
private KafkaTemplate<String,String> kafkaTemplate;

public void sendMessage(String message){
logger.info(String.format("$$ -> Producing message --> %s",message));
this.kafkaTemplate.send(TOPIC,message);
}
}
```

Integrace springboot

- *Vytvořte consumer:*

```
@Service
public class Consumer {
private final Logger logger = LoggerFactory.getLogger(Consumer.class);

@KafkaListener(topics = "users", groupId = "group_id")
public void consume(String message){
logger.info(String.format("$$ -> Consumed Message -> %s",message));
}
}
```

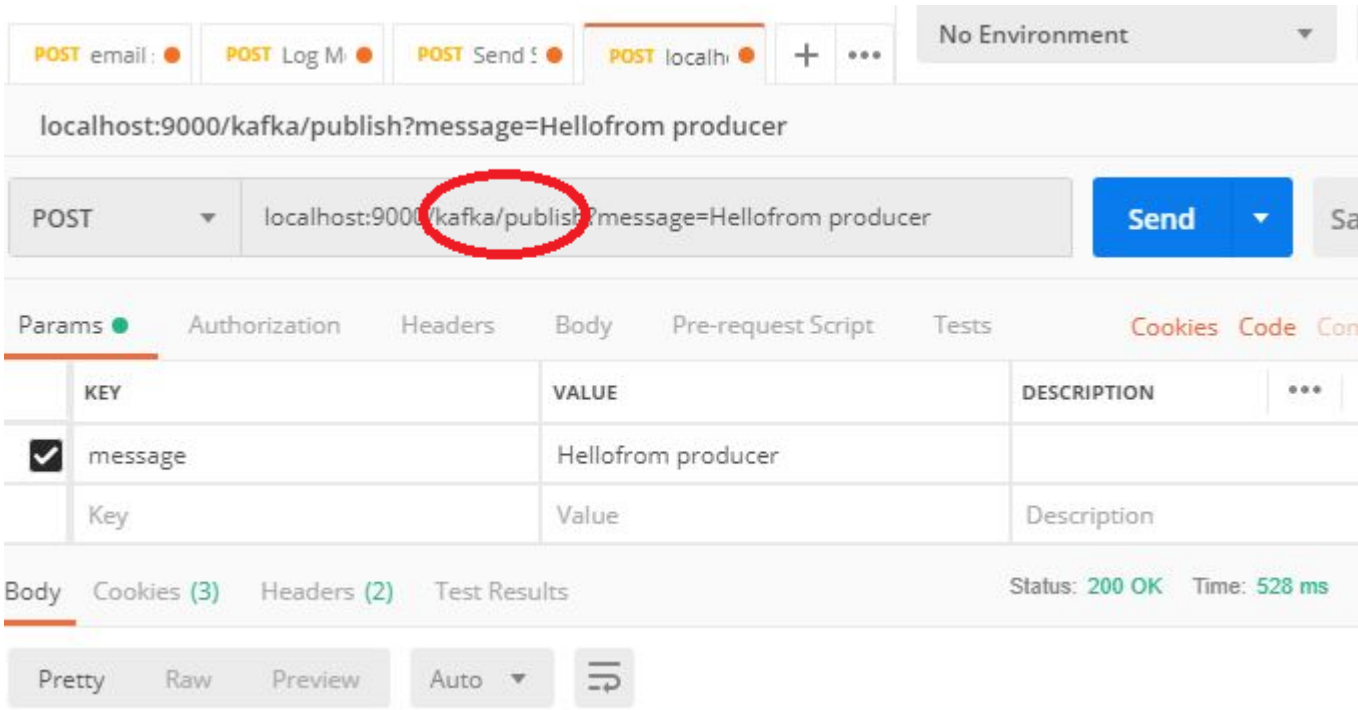
Integrace springboot

- *application.yaml*

```
server:
  port: 9000
spring:
  kafka:
    consumer:
      bootstrap-servers: localhost:9092
      group-id: group-id
      auto-offset-reset: earliest
      key-deserializer: org.apache.kafka.common.serialization.StringDeserializer
      value-deserializer: org.apache.kafka.common.serialization.StringDeserializer
    producer:
      bootstrap-servers: localhost:9092
      key-deserializer: org.apache.kafka.common.serialization.StringDeserializer
      value-deserializer: org.apache.kafka.common.serialization.StringDeserializer
```

Integrace springboot

- *Request :*



- *Výsledek :*