GOING BIG AND FAST {KAFKAESQUE} FOR KAFKA ACCESS

USER! 2021 - THE R CONFERENCE

5 - 9 JULY, 2021
Hosted by ETH Zürich / KOF

PETER MEISSNER

Consultant @virtual7 / virtual7.de
Tweet with me @peterlovesdata / petermeissner.de

INTRO

INTRO

{kafkaesque}

- R package to **connect** to Kafka, **read** from Kafka and **write** to Kafka topics.
- Kafka is an industry standard BigData technology
- available at: https://github.com/petermeissner/kafkaesque
- available via: `remotes::install_github('petermeissner/kafkaesque')`

INTRO

- **{rkafka}**
  - https://cran.r-project.org/web/packages/rkafka/index.html)
  - Java based
  - served as a blueprint to get started
  - on CRAN
  - not maintained, does not work with recent Kafka
- **{fRanz}**
  - https://github.com/uptake/fRanz
  - C++ based
  - never got finished

WHAT IS THAT KAFKA?
WHY SHOULD I CARE?

KAFKA

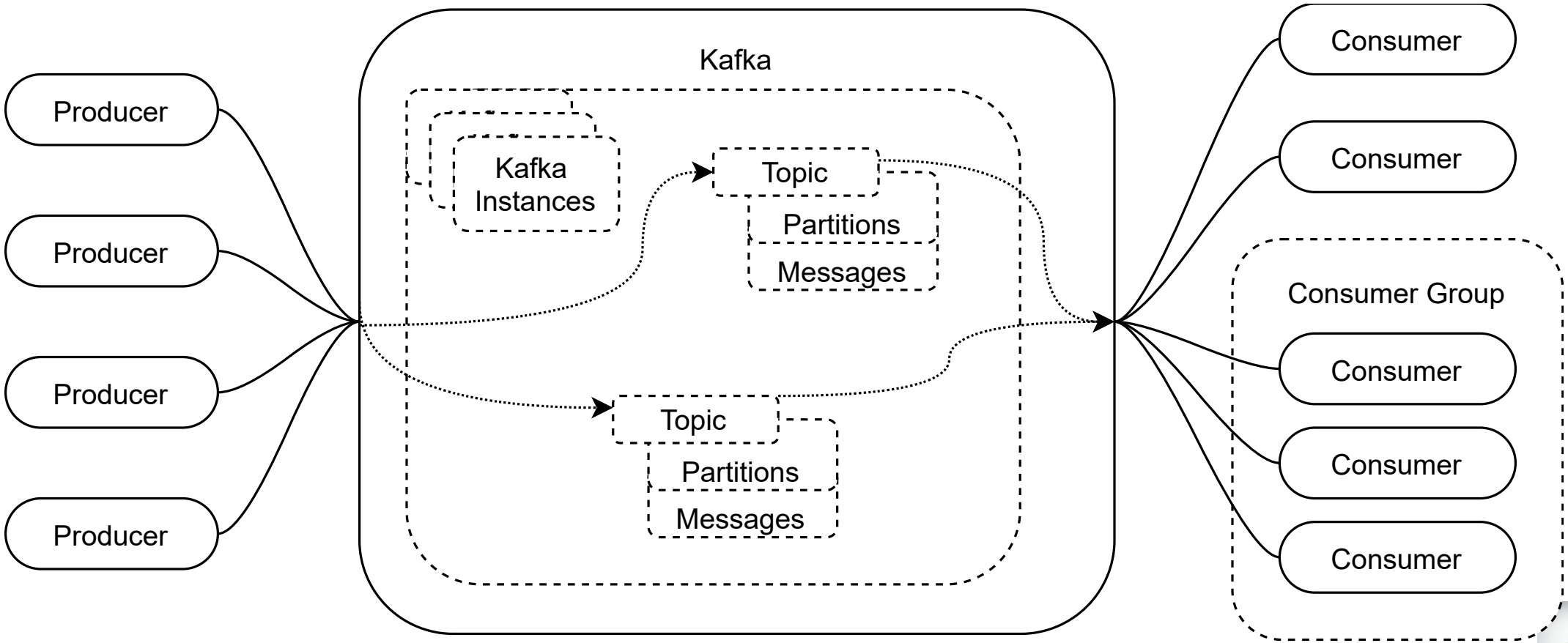
Kafka is a message queue *- put data in, get data out.*
Kafka is a log *- data is ordered, data is persistent.*

Kafka is distributed *- over a network of servers.*
Kafka is fault tolerant *- data is save, system still works if parts can fail.*
Kafka is scalable *- you can always add more servers.*
Kafka is asynchronous *- data can be consumed at any time.*

Kafka is a BigData technology *- low latency, high throughput*.
Kafka is a infrastructure technology *- think database not package*.

(see e.g.: https://kafka.apache.org/uses)

KAFKA

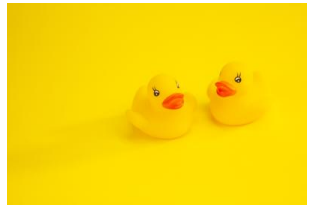

Producer
Producer
Producer
Producer
Kafka
Kafka Instances
Topic
Partitions
Messages
Topic
Partitions
Messages
Consumer
Consumer
Consumer Group
Consumer
Consumer
Consumer

OK.
BUT WHY SHOULD I CARE, AS A DATA SCIENTIST?

DATA ANALYTICS USE CASES?

Use **open and language agnostic** nature of Kafka to:

- Access data already ingested in Kafka.
- Make it easy for producers of data to share data with you.
- Share data with other users, languages, software, systems.



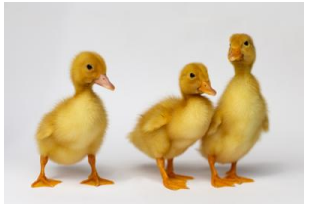
DATA ANALYTICS USE CASES?

Use **high throughput, low latency** nature of Kafka for:

- Any kind of stream processing.
- Real time.
- Near time processing.

DATA ANALYTICS USE CASES?

- Use **distributed** nature to:
  - Spread data throughout you servers.
  - Serve out and share tasks among workers.
  - Decouple time and location of data production and data consumption.



DATA ANALYTICS USE CASES?

Use **fault tolerance** and data **persistance** for

- Auditing
- Replay data
- Debug and explore data, after the fact

THERE IS A PACKAGE FOR THAT!

{KAFKAESQUE}

- Binds to **Kafka's native Java libraries**.
- Using **{rJava}** for R <> Java communication.
- Only depends on **R packages**:
  - {jsonlite}, {data.table}, {magrittr}, and {R6}
- Only depends on **Java libraries**:
  - {kafka} (+dependencies), {log4j}, {gson}

LIVE DEMO

Spin up Kafka Test Cluster

```
docker run -p 127.0.0.1:2181:2181 -p 127.0.0.1:9092:9092 petermeissner/kafkatest
```

demo_consumer.R

```
library(kafkaesque)

consumer <- kafka_consumer()
consumer$start()

consumer$topics_list()
consumer$topics_subscribe("test")

consumer$consume_next()
consumer$consume_next()
consumer$consume_next()
consumer$consume_next()
```

demo_consumer_loop.R

```
options(scipen=9000)
library(kafkaesque)

consumer <- kafka_consumer()
consumer$start()
consumer$topics_list()
consumer$topics_subscribe("test500000")

i <- 0
v <- 0

res <-
  consumer$consume_loop(
    f =
      function(loop_env){
        i <<- i + 1
        v <<- v + as.integer(loop_env$messages$value)
      },
    check =
      function(loop_env) {
        loop_env$meta$loop_counter < 10 * 1000
      }
  )

print(paste0("i = ", i, "; v = ", v))

res$meta$end_time - res$meta$start_time
```

demo_producer.R

```
library(kafkaesque)

producer <- kafka_producer()
producer$start()
producer$props()

producer$send(topic = "user2021", msg = "Ducks are the new cats.")
producer$send(topic = "user2021", msg = "Ducks are the new cats.")
producer$send(topic = "user2021", msg = "In 2021 ducks are the new cats.")
producer$send(topic = "user2021", msg = "In 2021 ducks are the new cats.")
producer$send(topic = "user2021", msg = "Don't be smug, get a duck.")
```

commandline commands

```
cd Dropbox\projects_r\kafkaesque\demo_user2021\

"C:\Program Files\R\R-4.0.5\bin\Rscript.exe" demo_consumer_commandline.R

"C:\Program Files\R\R-4.0.5\bin\Rscript.exe" demo_producer_commandline.R
```

demo_consumer_loop_commandline.R

```
options(scipen=9000)

library(kafkaesque)

consumer <- kafka_consumer()
consumer$start()
consumer$topics_list()
consumer$topics_subscribe("user2021")


res <-
  consumer$consume_loop(
    f =
      function(loop_env){
        cat(loop_env$messages$value, "\n")
      },
    check = function(loop_env){ TRUE }
  )
```

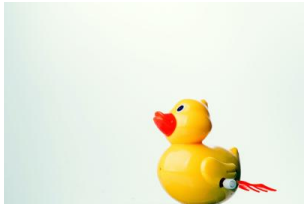
demo_consumer_loop_commandline.R

```
library(kafkaesque)
library(fortunes)

producer <- kafka_producer()
producer$start()

i <- 0
while ( TRUE ) {
  Sys.sleep(0.001)
  i <- i + 1
  producer$send(
    topic = "user2021",
    msg   = paste(i, "-", paste(fortunes::fortune(), collapse = "\n"))
  )
}
```

RETROSPECTIVE

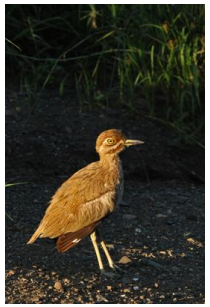
R <> JAVA

- R <> Java bindings have (unnecessarily) a **bad reputation**.
- R <> Java is more **low code** than e.g. R <> C++
- Accept that: **everything is typed**, restricted to method calls, restricted to scalars or vectors (more or less)
- R <> **Java is ok**.
- Use a **project as blueprint** (we used: https://github.com/hrbrmstr/htmlunit, or {kafkaesque} 😄)
- Just use **VSCode** for Java development.



CRAN

- If you plan on publishing on **CRAN**, things might get difficult.
  - Java wrappers are not that common.
  - CRAN has ('harsh') **package size restrictions** (5MB vs. e.g. 100MB on pypi).
  - There are **no common Java dependencies** on CRAN so everything has to be packaged by yourself ➜ size increases.
  - There is **no common way of downloading Java dependencies** as part of the installation procedure - maybe we can and should create one.



CONCLUSION

CONCLUSION

- **Kafka is a solid** technology with various use cases - also in the realm of **data analytics**
- At the moment **{kafkaesque}** does only work with text data, but can be extended,
  since the package is actually only a thin layer of R and Java.
- R <> Java is not perfect but far better than its reputation and actually quite easy.
- **{kafkaesque} is a working R binding for Kafka**,
  allowing to access and control all major Kafka API endpoints.