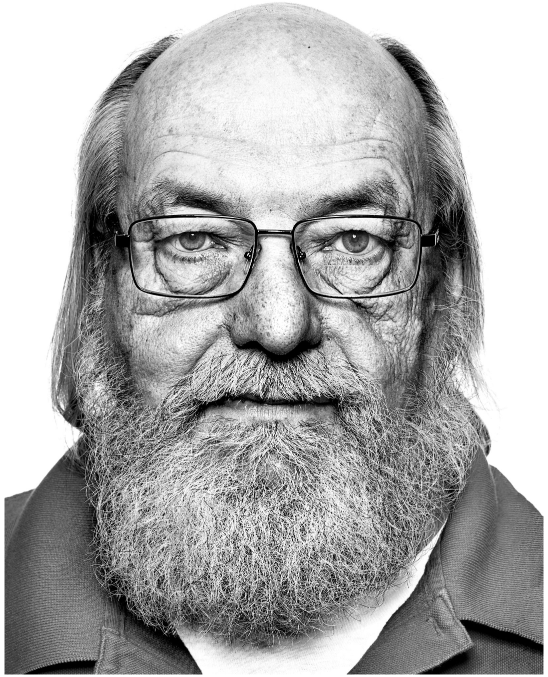
Composable microservices for streaming analytics

Rui Vieira, Michael McCune

Overview

- Microservices
- Kappa architecture
- Streams
- Microservices as primitives
- Streaming microservices in action

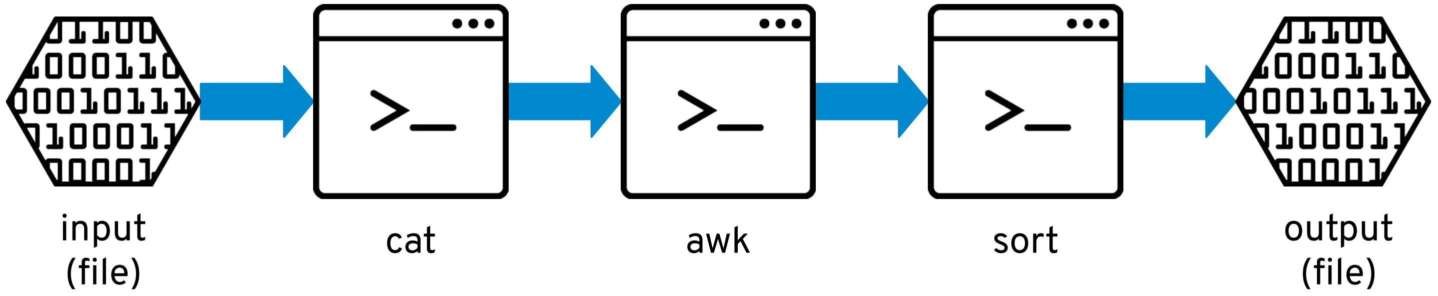
UNIX philosophy


Ken Thompson

- Write programs that do one thing and do it well.
- Write programs to work together.
- Write programs to handle text streams, because that is a universal interface.

Peter H. Salus in *A Quarter-Century of Unix* (1994)

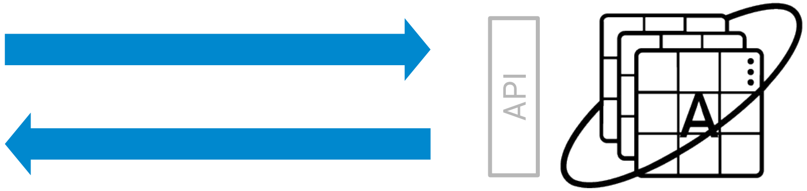
UNIX philosophy: pipelines



```
cat input.txt | awk '{print $2}' | sort > output.txt
```

What are microservices?

A microservice is a (lightweight) process with a simple and well-defined protocol, specialising in a specific task.



- Write programs that do one thing and do it well.
- Write programs to work together.
- Write programs to handle ~~text streams~~ *well defined APIs over the network*, because that is a universal interface.

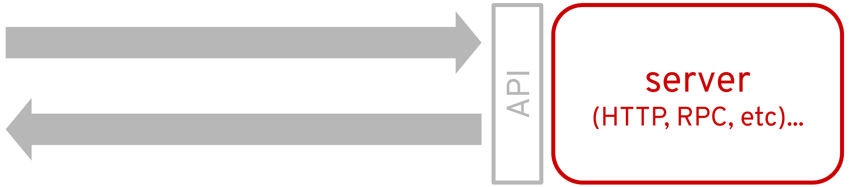
Microservices and UNIX philosophy

	UNIX	**microservices**
process	Binary executable	HTTP or RPC server, stream processor
configuration	CLI	JSON/YAML/...
communication	Pipes, temporary files,...	HTTP requests, streams, message queues,...

Microservices and UNIX philosophy

process

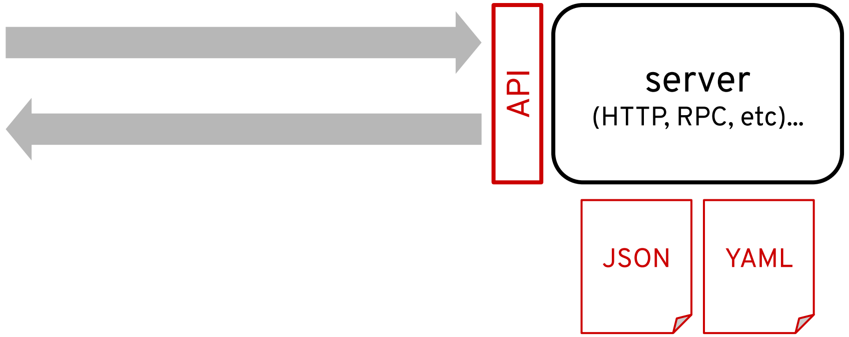
```
ls -l | awk '{print $2}' | sort > output.txt
```



Microservices and UNIX philosophy

configuration

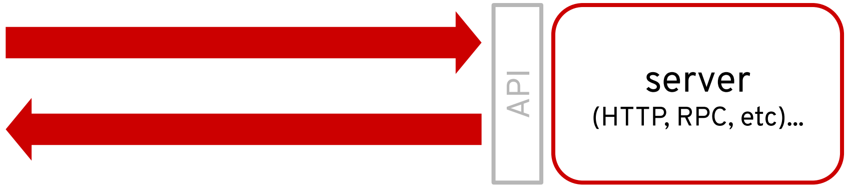
```
ls -l | awk '{print $2}' | sort > output.txt
```



Microservices and UNIX philosophy

communication

```
ls -l | awk '{print $2}' | sort > output.txt
```



Why microservices?

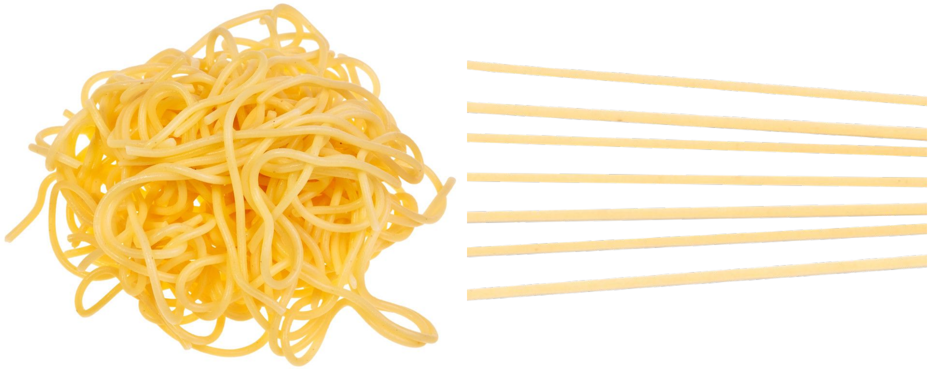
- Simplify code
- Separation of concerns
- Decouple unit testing
- Parallel development
- Simplify refactoring
- Polyglot development

Why microservices?

- Simplify code
- Separation of concerns
- Decouple unit testing
- Parallel development
- Simplify refactoring
- Polyglot development

Why microservices?

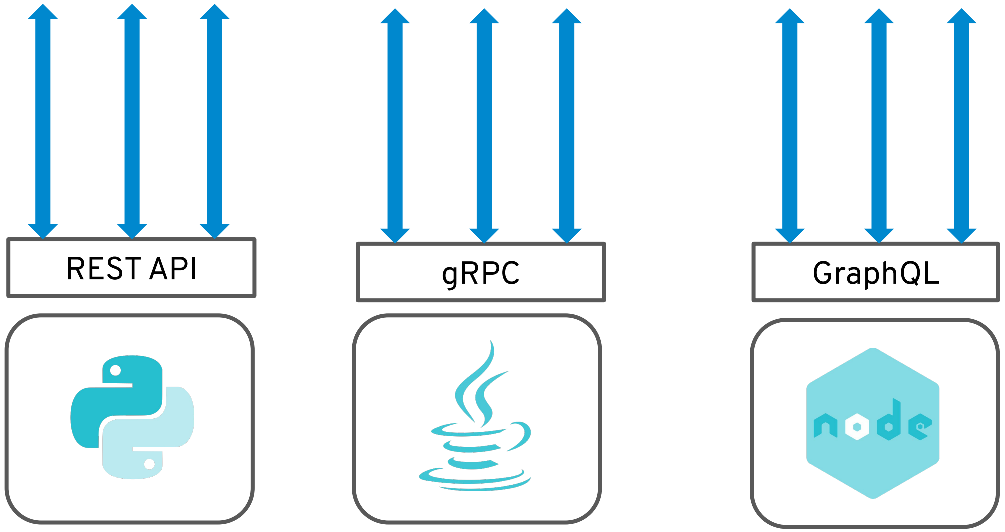
- Simplify code
- Separation of concerns
- Decouple unit testing
- Parallel development
- Simplify refactoring
- Polyglot development



Spaghetti *vs.* Spaghetto

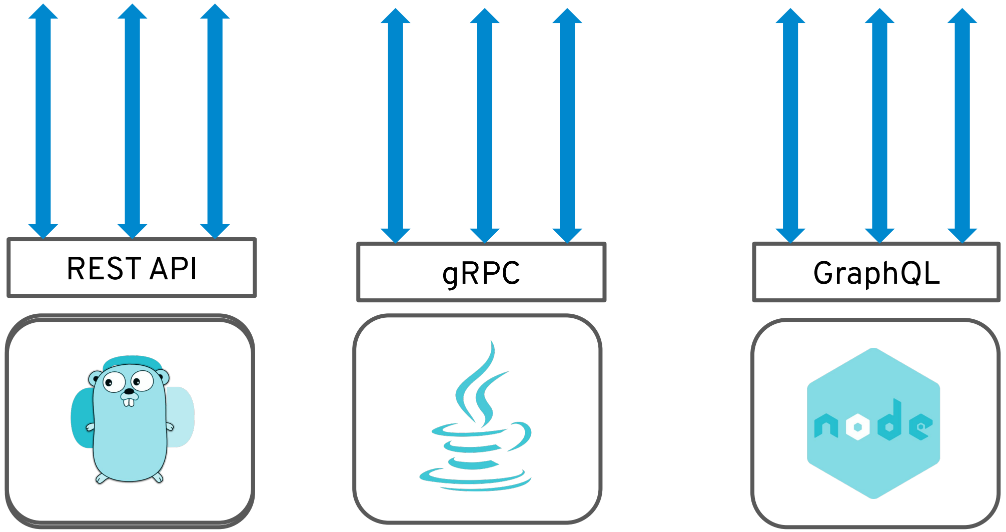
Why microservices?

- Simplify code
- Separation of concerns
- Decouple unit testing
- Parallel development
- Simplify refactoring
- Polyglot development



Why microservices?

- Simplify code
- Separation of concerns
- Decouple unit testing
- Parallel development
- Simplify refactoring
- Polyglot development



Challenges with microservices

orchestration

versioning

security

discovery

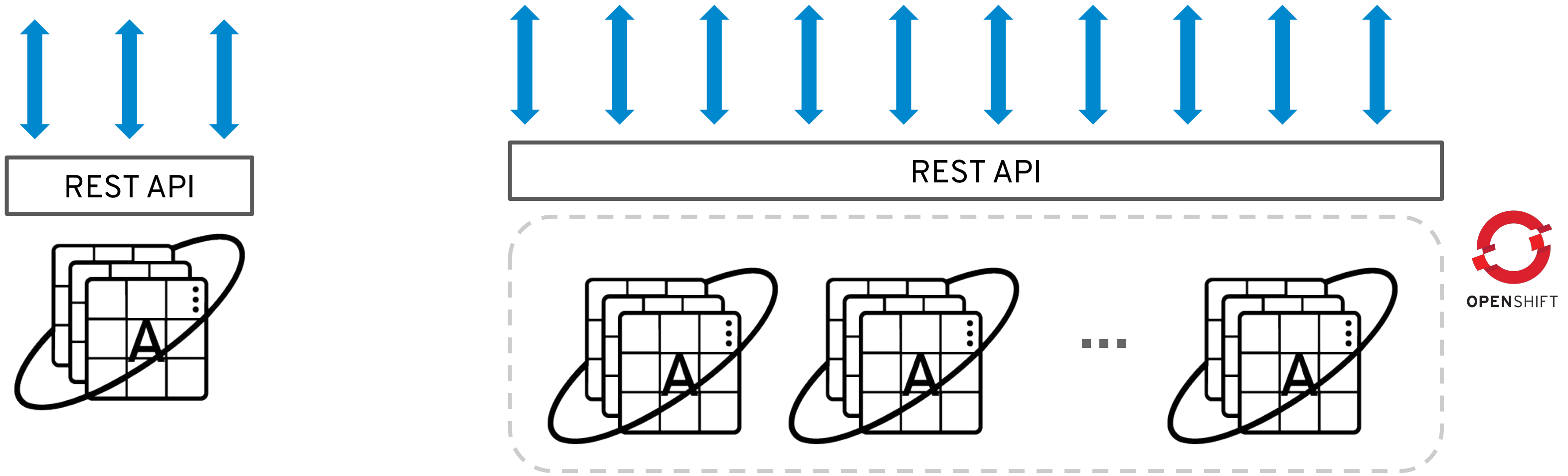
Challenges with microservices

- Orchestration
- Versioning
- Security
- Discovery

Challenges with microservices

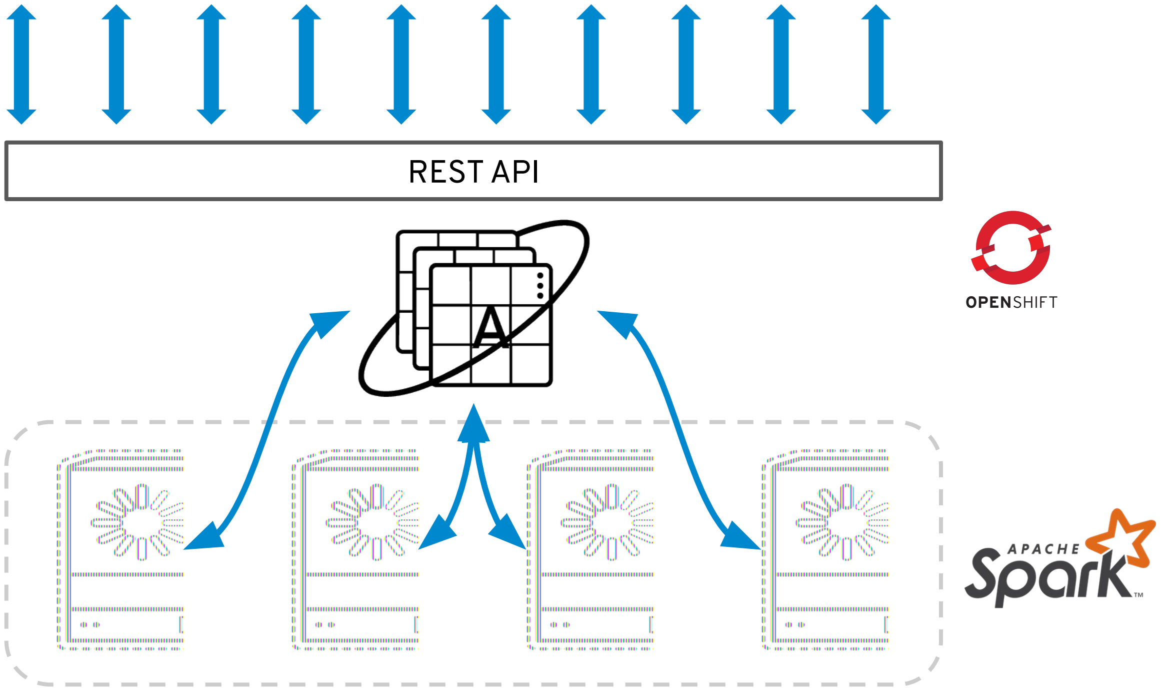
A Microservice-oriented architecture alone does not solve scalability problems

However, the stateless nature of microservices is a natural fit for containerisation. A powerful solution for scalability problems.



Challenges with microservices

A Microservice-oriented architecture alone does not solve scalability problems

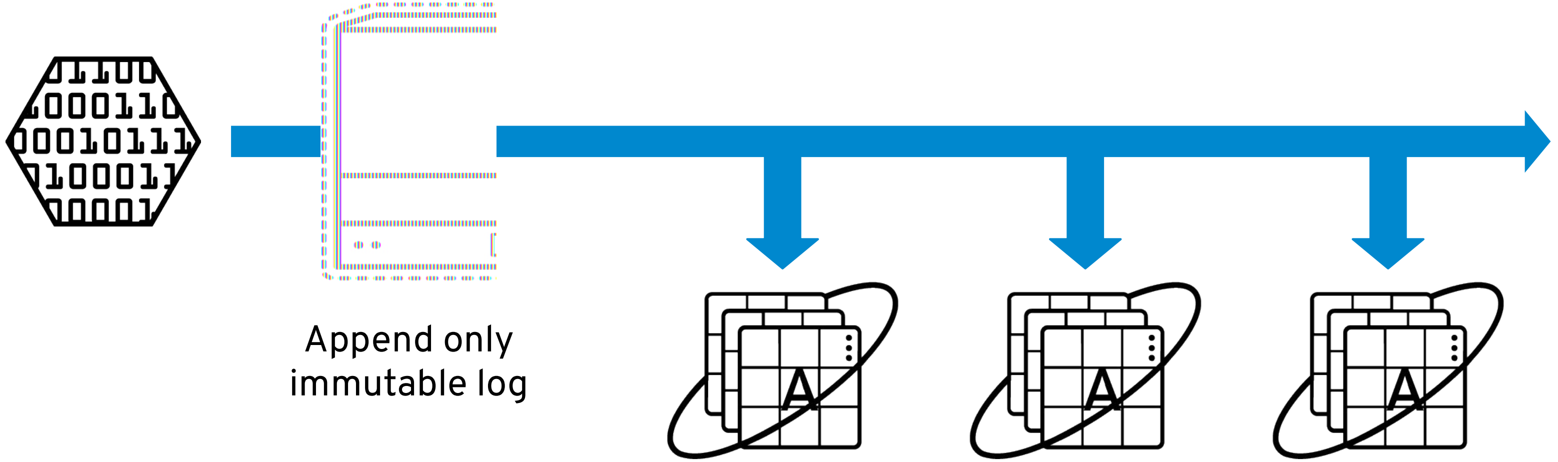


Challenges with microservices

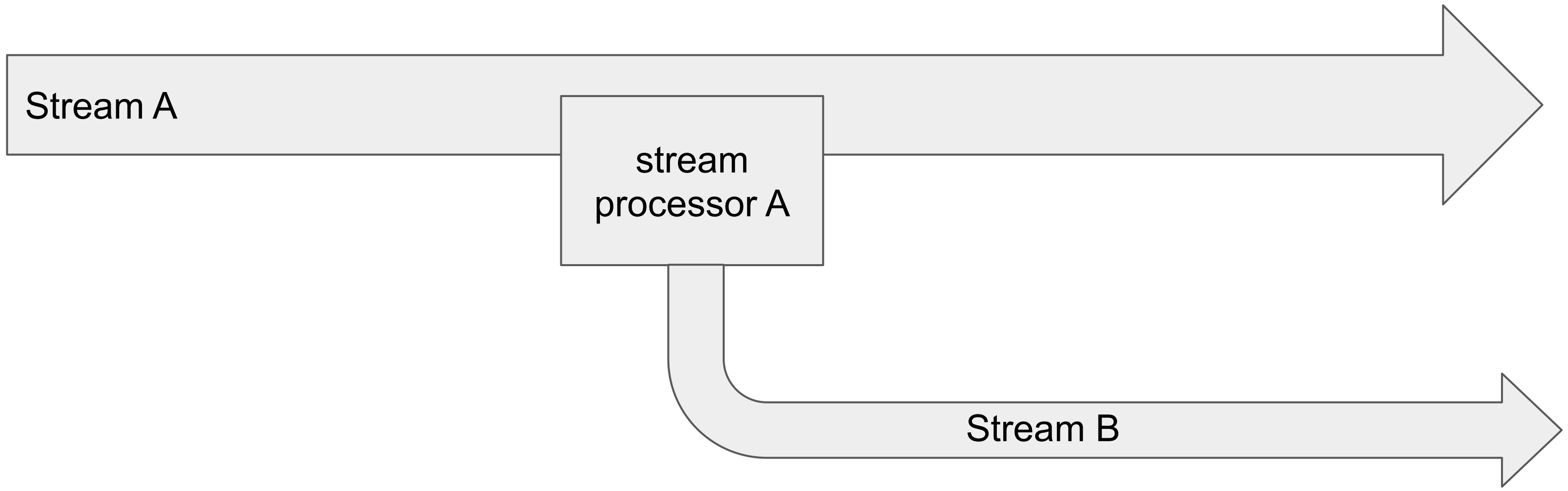
What are the boundaries of a microservice in Big Data analytics?

e.g. More than one microservice use Apache Spark, should the cluster be shared or should we have a cluster per microservice?

Kappa architecture



Stream composability



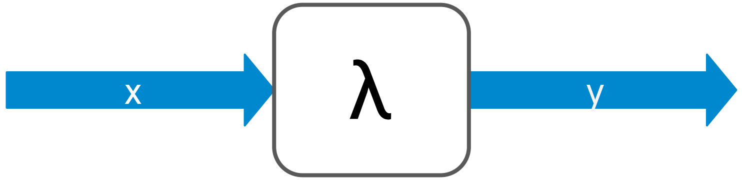
Challenges with streams

- Latency
- Stateful transformations
- Security
- Reconciliation

Modular analytics

Functional programming

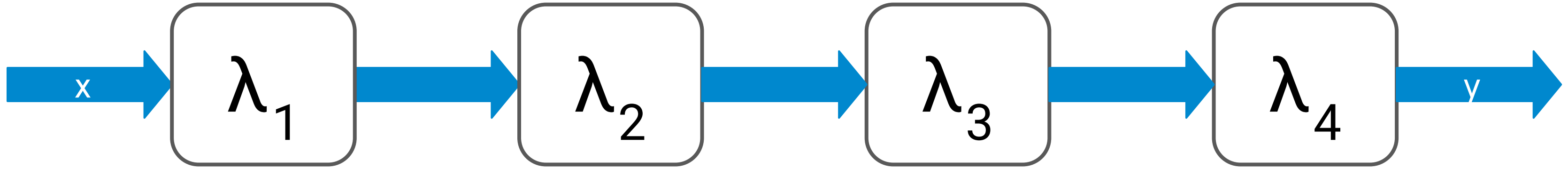
$$y = \lambda(x)$$



Modular analytics

Functional programming

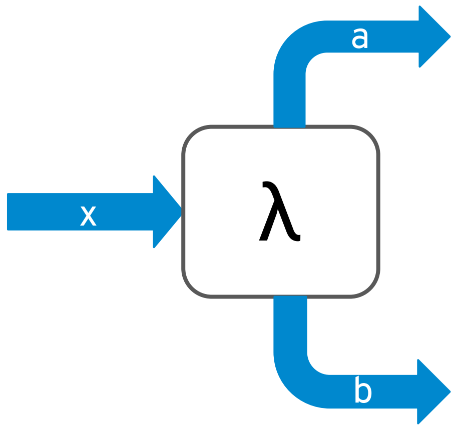
$$y = \lambda_4(\lambda_3(\lambda_2(\lambda_1(x))))$$



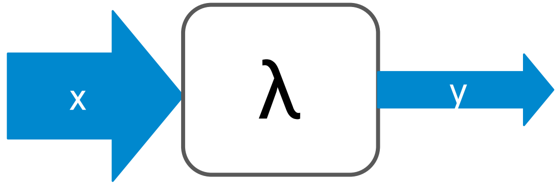
Modular analytics

Functional programming

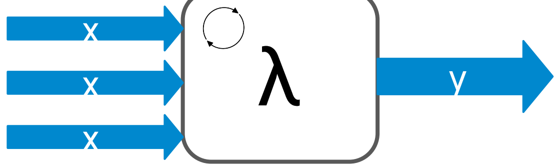
`a if x<0 else b`



`x.filter`

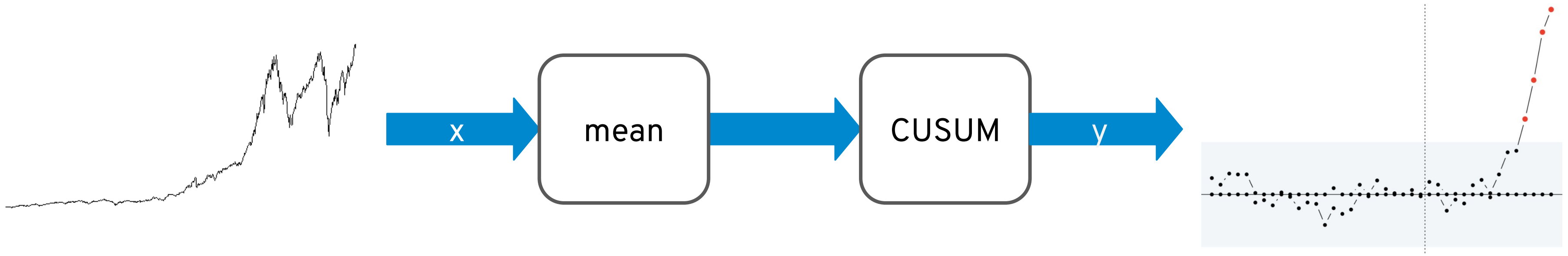


`x.reduce`



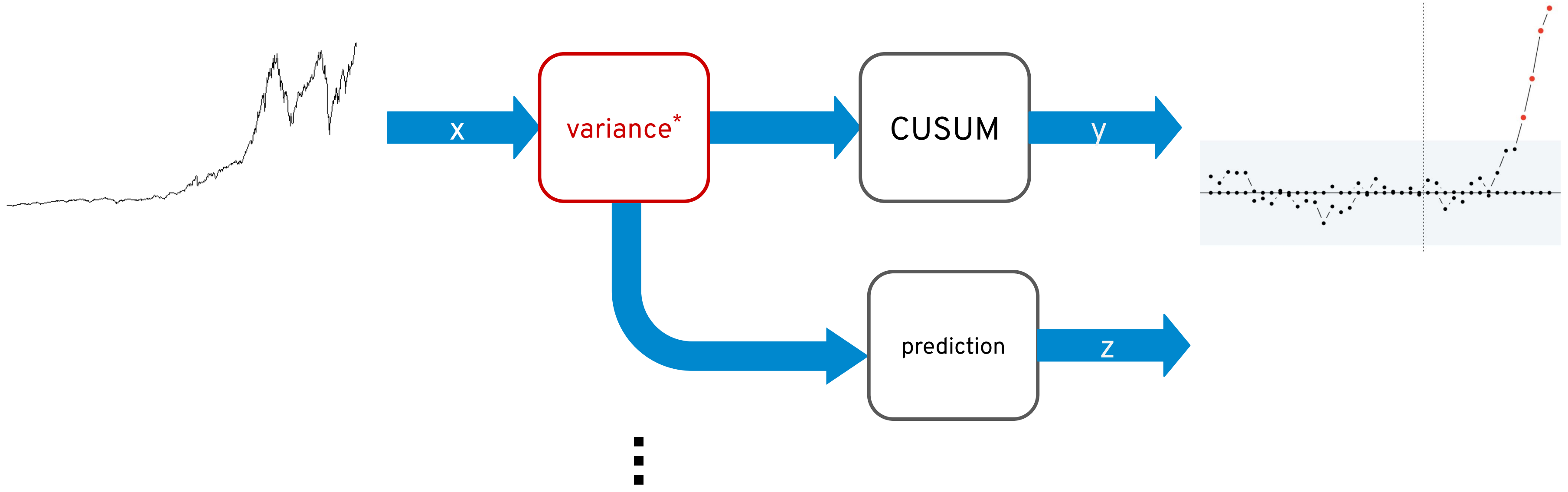
Modular analytics

Example: detect abrupt changes in a process mean



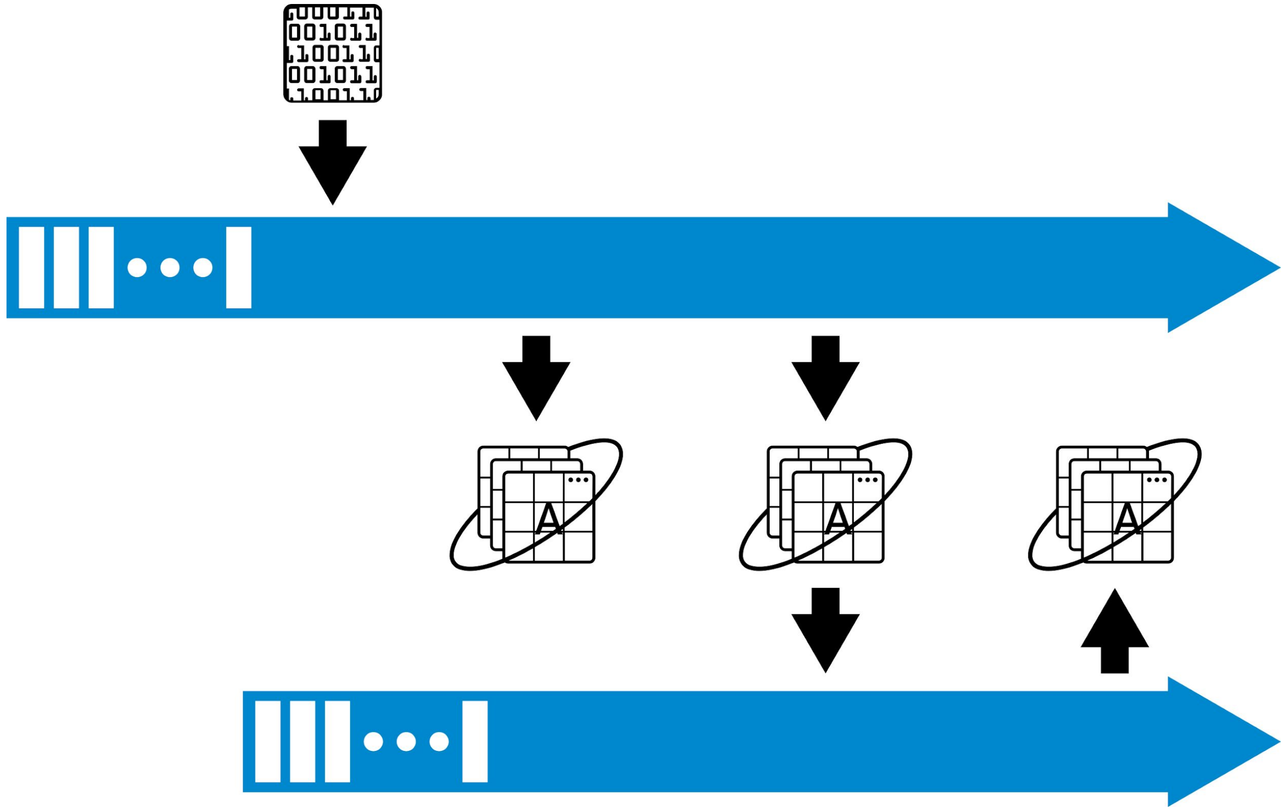
Modular analytics

Example: detect abrupt changes in a process mean



Streaming microservices in action

Generalized architecture

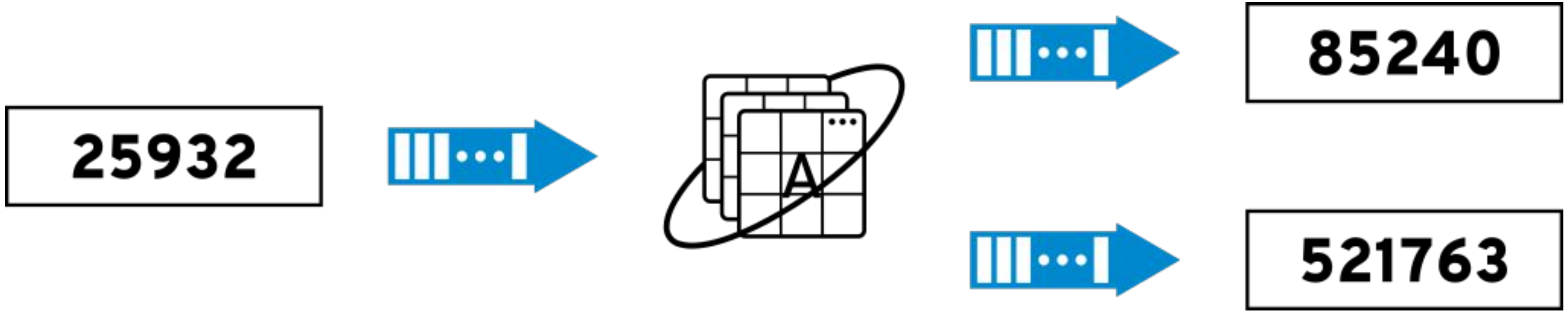


Technology choices

Things that I will use in these examples

- OpenShift
- Apache Kafka
- Apache Spark
- Python

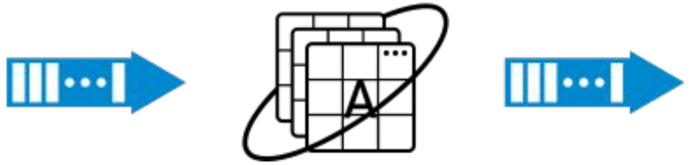
Example: Data filtering



Example: Data transformation

```
{
  "update_id": "00000000000000000479",
  "user_id": "1407702551",
  "text": "I don't care to take another bite.
           #Thursday #Halloween"
}
```

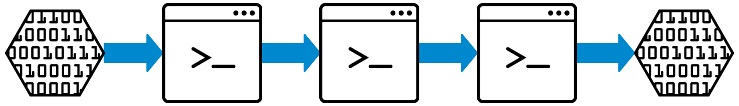
```
{
  "update_id": "00000000000000000479",
  "user_id": "1407702551",
  "text": "I don't care to take another bite.
           #Thursday #Halloween"
  "sentiments": [
    {"neg": 0.0,
     "neu": 0.5479999780654907,
     "pos": 0.4519999921321869,
     "compound": 0.5095000267028809},
    {"neg": 0.0,
     "neu": 1.0,
     "pos": 0.0,
     "compound": 0.0}
  ]
}
```

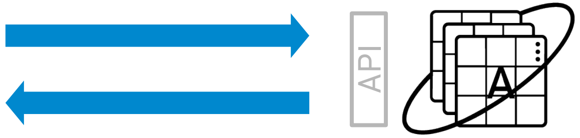


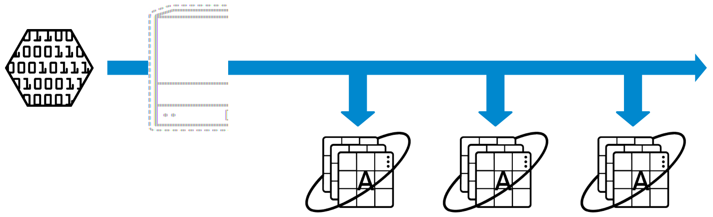
Practical concerns

Architects and developers will run into these issues

- Message formats
- Brokers, topics, and general configurations
- Data provenance
- Testing
- Debugging



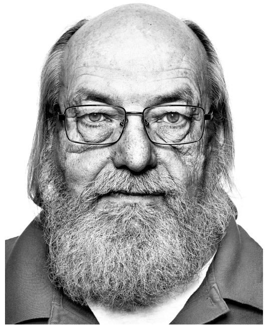


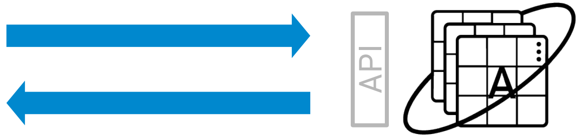


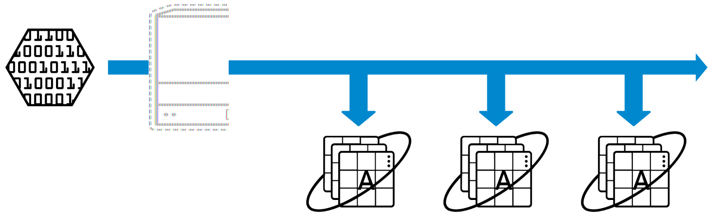
radanalytics.io

Rui Vieira
rui@redhat.com
@ruivieira@mastodon.technology

Michael McCune
msm@redhat.com
@elmiko@mastodon.technology







radanalytics.io

Rui Vieira
rui@redhat.com
@ruivieira@mastodon.technology

Michael McCune
msm@redhat.com
@elmiko@mastodon.technology