```
mgray@pivotal.io ~ ❯❯❯ $ cat /etc/bootiful_reactive.m4v
Bootiful Reactive Testing & CDC w/ Spring Boot 2

mgray@pivotal.io ~ ❯❯❯ $ whoami
Name: Mario Gray
Role: Developer Advocate / Principal Technologist
Twitter: @mariogray
Code/Github: https://www.github.com/marios-code-path/bootiful-testing

mgray@pivotal.io ~ ❯❯❯ $ history | xargs jobs
… more history ...
2006  cat < /var/spool/jobs/EdwardJones > /dev/analyst
2011  write `cat ~/co-authors` “Pro Spring Integration”
2013  mv ~ la ; cat < /var/spool/jobs/NFL_Media > /dev/integrator
2015  cat  /var/spool/jobs/MobCrush > /dev/microservices/video/stream
2017  cat  /var/spool/jobs/pivotal > /dev/advocate
```

```
mgray@pivotal.io ❯❯❯ $ cat /etc/issue
```

Whats Next: TDD, Spring CDC, Reactive

```
mgray@pivotal.io ❯❯❯ $ cat /etc/tdd/qotd
```

Don’t test trivial code getter/setter(s)

- *Kent Beck*

```
mgray@pivotal.io ❯❯❯ $ cat /etc/tdd/qotd
```

Nothing travels faster than the Speed of Light ; with the exception of bad news which obeys its own special laws.

- *Douglas Adams* (Mostly Harmless ‘92)

```
mgray@pivotal.io ❯❯❯ $ cat /etc/t
```

CDC’s are invaluable for being able to move fast without breaking other services and cause a lot of frustration with other teams.

- *Ham Vocke*

```
mgray@pivotal.io ❯❯❯ $ cat /etc/tdd/bobmartin/laws
```

1. *You are not allowed to write any production code unless it is to make a failing unit test pass.*
2. *You are not allowed to write any more of a unit test than is sufficient to fail; and compilation failures are test failures.*
3. *You are not allowed to write any more production code than is sufficient to pass the one failing unit test.*

mgray@pivotal.io ❯❯❯ $ xargs --show-limits --cdc.verifier

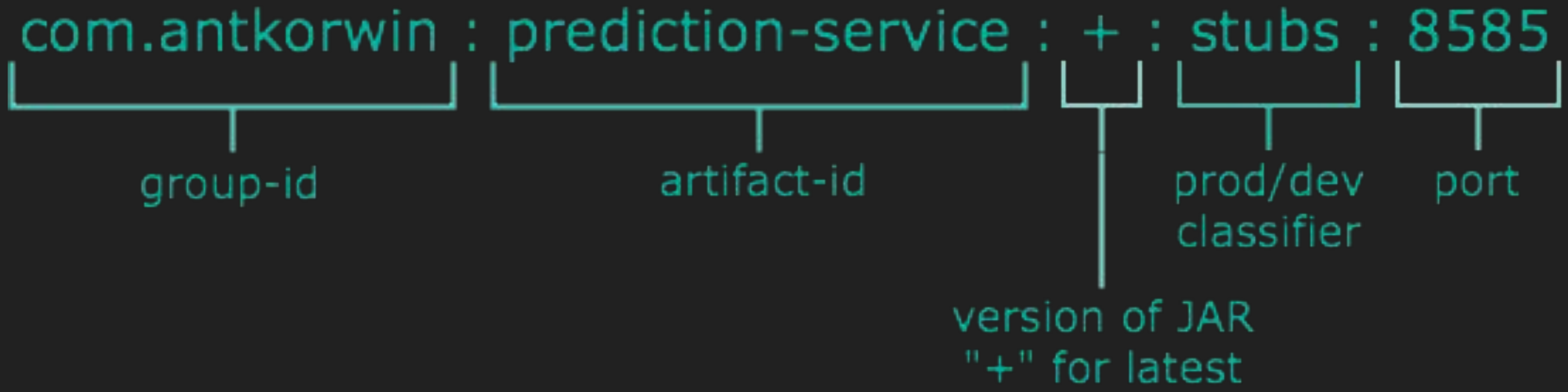
- Spring Cloud Contract Verifier enables Consumer Driven Contract (CDC) development of JVM-based applications.
- *Contract Definition Language* (CDL) to produce/generate:
  - JSON stubs used by WireMock when doing client integration.
  - Messaging routes; supports Spring Integration, Spring Cloud Stream, Spring AMQP, and Apache Camel. Custom integrations.
  - Acceptance tests (in JUnit or Spock) implementation of the API is compliant with the contract (*server tests*). Uses RestAssured.

mgray@pivotal.io ❯❯❯ $ xargs --show-limits --cdc.runner

- Stub Runner can automatically download and execute.
- Stub Jars are manageable via centralized repository like Maven.
- Service Discovery is Stubbable.
  - Takes place for Consul, Eureka, or other Directory
  - Wiremock to back stubbed service discovery when using Feign, load-balanced Rest-template or DiscoveryClient.

```
mgray@pivotal.io ~ ❯❯❯ $ xview /etc/cdc/desc.png
```

Spring Cloud Contract Locator Format:


com.antkorwin : prediction-service : + : stubs : 8585
group-id
artifact-id
version of JAR "+" for latest
prod/dev classifier
port

```
mgray@pivotal.io ❯❯❯ $ cat ~/.reactive
```

Let’s get (re)Active

```
mgray@pivotal.io ~ ❯❯❯ $ xargs --show-limits --reactive
```

Core: Fully Non-blocking foundation with efficient demand management. It directly interacts with Java 8 functional API, Completable Future, Stream and Duration.

Types: Reactor offers [0|1|N] sequences in 2 reactive composable components: Flux[N] and Mono[0|1] extensively implementing Reactive Extensions.

Comms: Non-Blocking IPC that is suited for Microservices. Reactor IPC offers backpressure-ready network engines for TCP, UDP, HTTP and WebSockets. Encoder/Decoder (encoded to POJO) functionality is fully supported.

```
mgray@pivotal.io ~ ❯❯❯ $ cat /etc/reactive/interfaces
```

Reactive Streams: 4 interfaces

- Publisher<T> // subscribe(sub)
- Subscriber<T> // onSubscribe(sub)
- Subscription // request (n)
- Processor<T, R> // subscribe (s) && onSubscribe(s)

```
mgray@pivotal.io ~ ❯❯❯ $ cat /etc/reactive/operators
```

filter(x)

map (x)

flatMap (x[][])

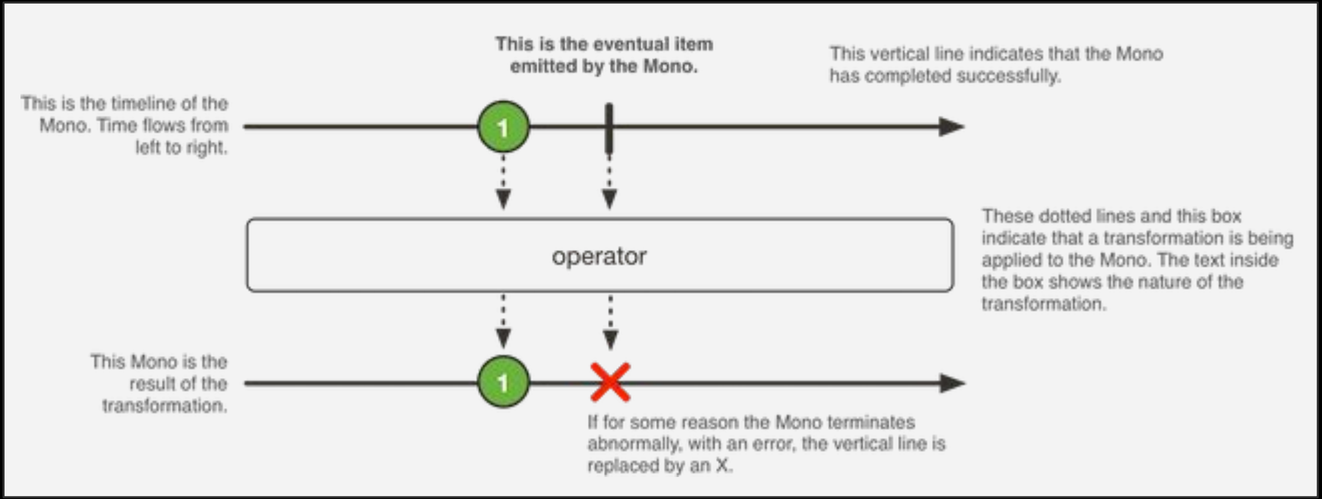
zipWith(x[])

reduce(i, x)

repeat( )

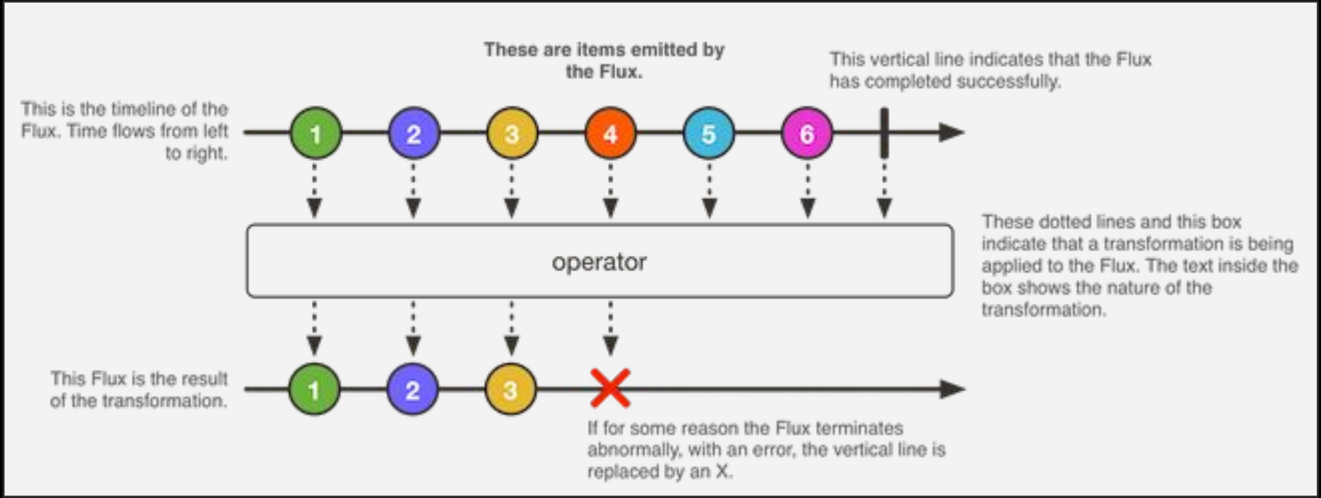
… And More !

mgray@pivotal.io ~ ❯❯❯ $ cat /etc/reactive/publisher/mono


This is the eventual item emitted by the Mono.
This vertical line indicates that the Mono has completed successfully.
This is the timeline of the Mono. Time flows from left to right.
1
operator
These dotted lines and this box indicate that a transformation is being applied to the Mono. The text inside the box shows the nature of the transformation.
This Mono is the result of the transformation.
1
If for some reason the Mono terminates abnormally, with an error, the vertical line is replaced by an X.

Source: https://projectreactor.io/docs/core/snapshot/api/reactor/core/publisher/Mono.html

mgray@pivotal.io ~ ❯❯❯ $ cat /etc/reactive/publisher/flux


These are items emitted by the Flux.
This vertical line indicates that the Flux has completed successfully.
This is the timeline of the Flux. Time flows from left to right.
1
2
3
4
5
6
operator
These dotted lines and this box indicate that a transformation is being applied to the Flux. The text inside the box shows the nature of the transformation.
This Flux is the result of the transformation.
1
2
3
If for some reason the Flux terminates abnormally, with an error, the vertical line is replaced by an X.

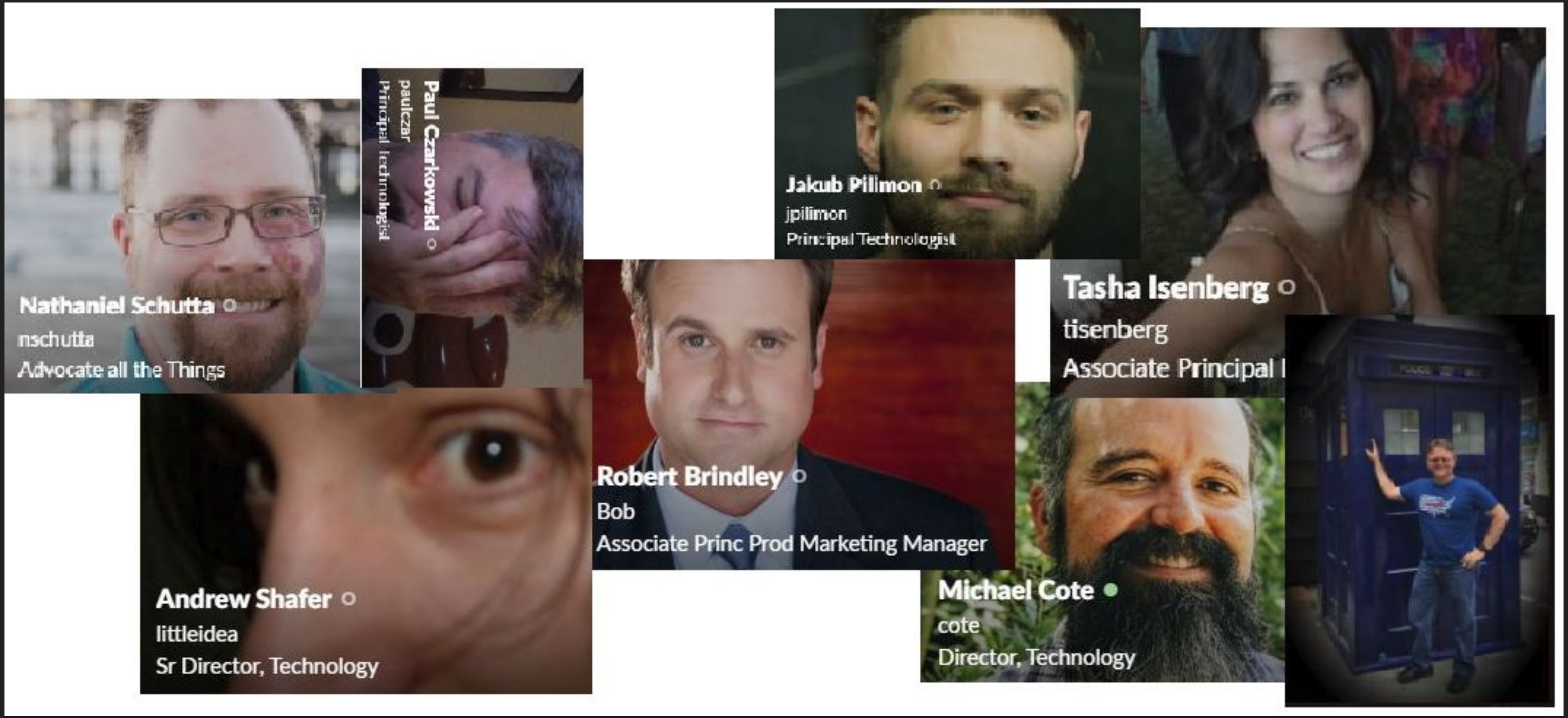
Source: https://projectreactor.io/docs/core/snapshot/api/reactor/core/publisher/Flux.html

Demo Time!

(Reactive Data, Web, then CDC)

mgray@pivotal.io ~ ❯❯❯ $ cat /etc/issue

Say Hello to \/ \/ \/   (Plus the other "usual suspects")



```
mgray@pivotal.io ~ ❯❯❯ $ exit
```

This talk is shutting down in 60 seconds.

Link to this talk: https://www.github.com/marios-code-path/bootiful-testing

BLOG: http://www.sudoinit5.com

Thanks!

Questions?

Here are some helpful learning resources:

https://github.com/reactor/reactor-ipc/blob/master/src/docs/asciidoc/net.adoc

https://github.com/reactive-ipc/reactive-ipc-jvm

http://www.reactive-streams.org
https://cloud.spring.io/spring-cloud-contract/multi/multi__spring_cloud_contract_verifier_introduction.html
https://projectreactor.io/docs/core
https://martinfowler.com/articles/201701-event-driven.html
http://jonasboner.com/foreword-reactive-design-patterns
https://projectreactor.io/docs/core/snapshot/api

https://github.com/mkheck/FSRx

(R2DBC!: https://github.com/mkheck/coffee-service-r2dbc,
https://github.com/mkheck/getting-started-r2dbc)