Drinking from the Stream

How to use messaging platforms for scalability & performance

Mark Heckler
Professional Problem Solver, Spring Developer & Advocate
www.thehecklers.com
mark@thehecklers.com
mheckler@pivotal.io
@mkheck

Pivotal

Distributed systems are easy…right?

Who am I?

- Author
- Architect & Developer
- Java Champion, Rockstar
- Professional Problem Solver
- Spring Developer & Advocate
- Creador y curador de

SPRING NOTICIAS EN ESPAÑOL

Takeaways

- **Why use messaging platforms/where do they fit in a distributed architecture?**
- Examples of leading messaging platforms
- What is Spring Cloud Stream?
- Why use it?

Takeaways

- **Why use messaging platforms/where do they fit in a distributed architecture?**
- Examples of leading messaging platforms
- What is Spring Cloud Stream?
- Why use it?

The Bezos mandate

- All teams will henceforth expose their data and functionality through service interfaces.
- Teams must communicate with each other through these interfaces.
- There will be no other form of interprocess communication allowed: no direct linking, no direct reads of another team's data store, no shared-memory model, no back-doors whatsoever. The only communication allowed is via service interface calls over the network.
- It doesn't matter what technology is used: HTTP, CORBA, pub/sub, etc.
- All service interfaces, without exception, must be designed from the ground up to be externalizable. That is to say, the team must plan and design to be able to expose the interface to developers in the outside world. No exceptions.
- Anyone who doesn't do this will be fired.

The 12 Factor Manifesto

- One codebase in revision control, many deploys
- Explicitly declare & isolate dependencies
- Store config in environment
- Treat backing services as attached resources
- Build, release, run
- Stateless processes
- Export services via port binding
- Scale out via process model
- Fast startup, graceful shutdown
- Dev/prod parity
- Treat logs as event streams
- Admin (one off) processes

Takeaways

- **Why use messaging platforms/where do they fit in a distributed architecture?**
- Examples of leading messaging platforms
- What is Spring Cloud Stream?
- Why use it?

Takeaways

- Why use messaging platforms/where do they fit in a distributed architecture?
- **Examples of leading messaging platforms**
- What is Spring Cloud Stream?
- Why use it?

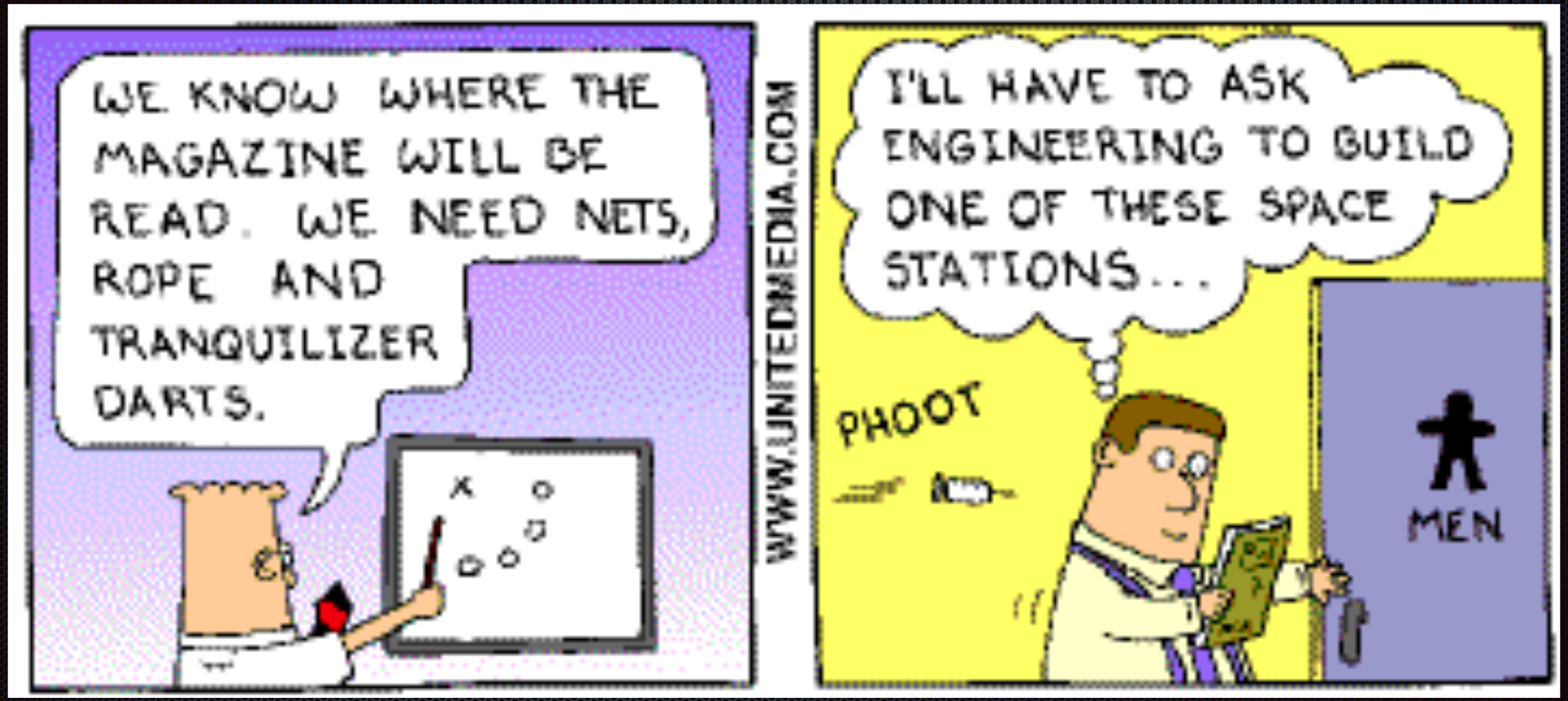
Takeaways

- Why use messaging platforms/where do they fit in a distributed architecture?
- Examples of leading messaging platforms
- **What is Spring Cloud Stream?**
- Why use it?

Takeaways

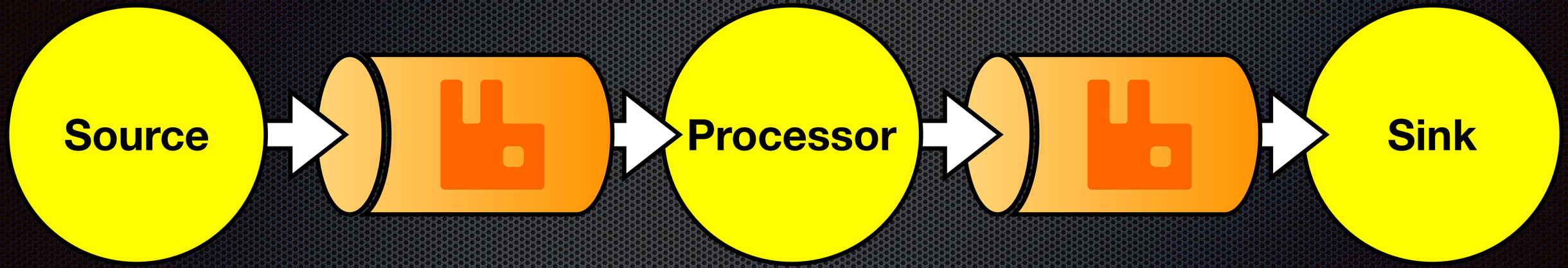
- Why use messaging platforms/where do they fit in a distributed architecture?
- Examples of leading messaging platforms
- What is Spring Cloud Stream?
- **Why use it?**

Why use it?

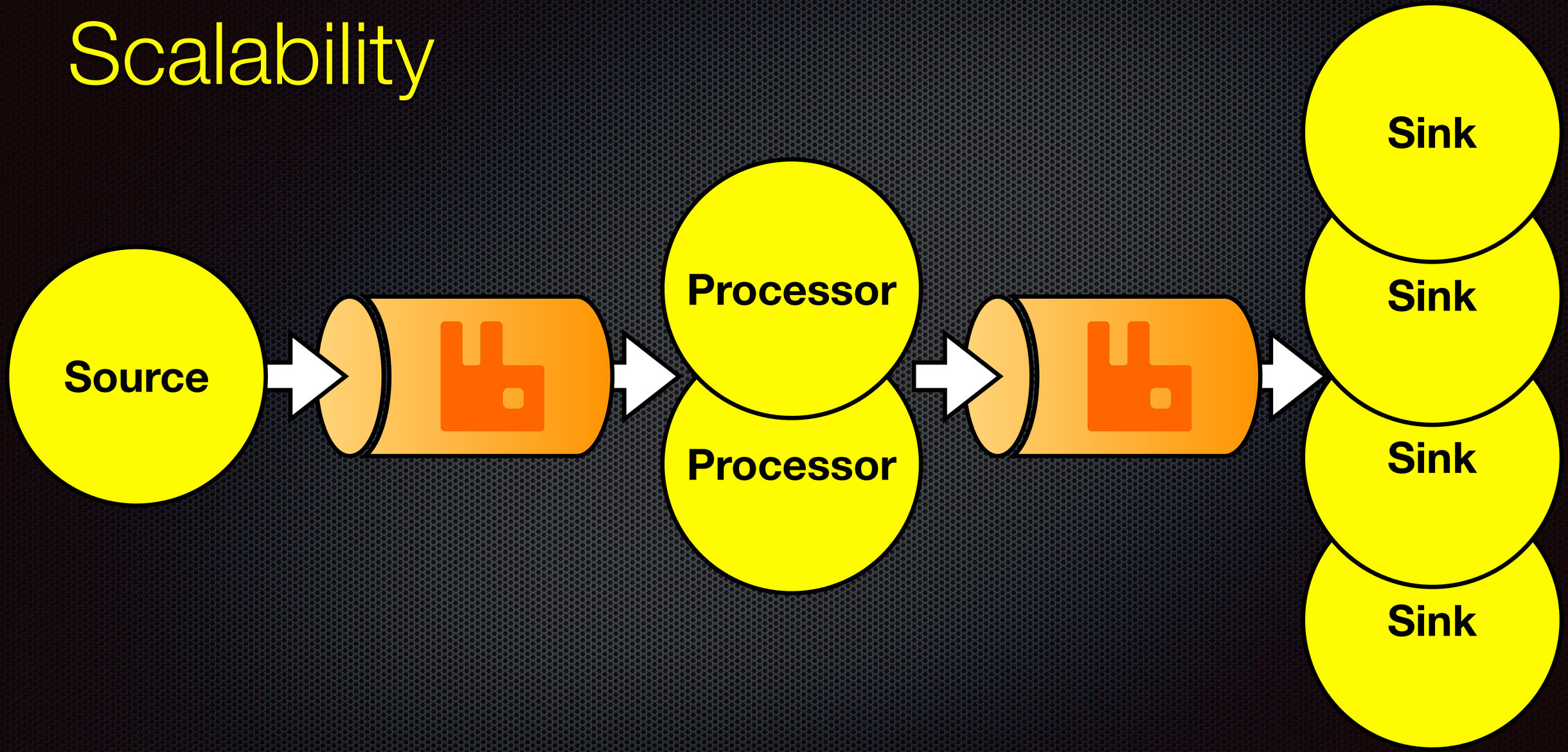

WE KNOW WHERE THE MAGAZINE WILL BE READ. WE NEED NETS, ROPE AND TRANQUILIZER DARTS.
WWW.UNITEDMEDIA.COM
I'LL HAVE TO ASK ENGINEERING TO BUILD ONE OF THESE SPACE STATIONS...
PHOOT
MEN

SCSt + ${messaging.platform} =

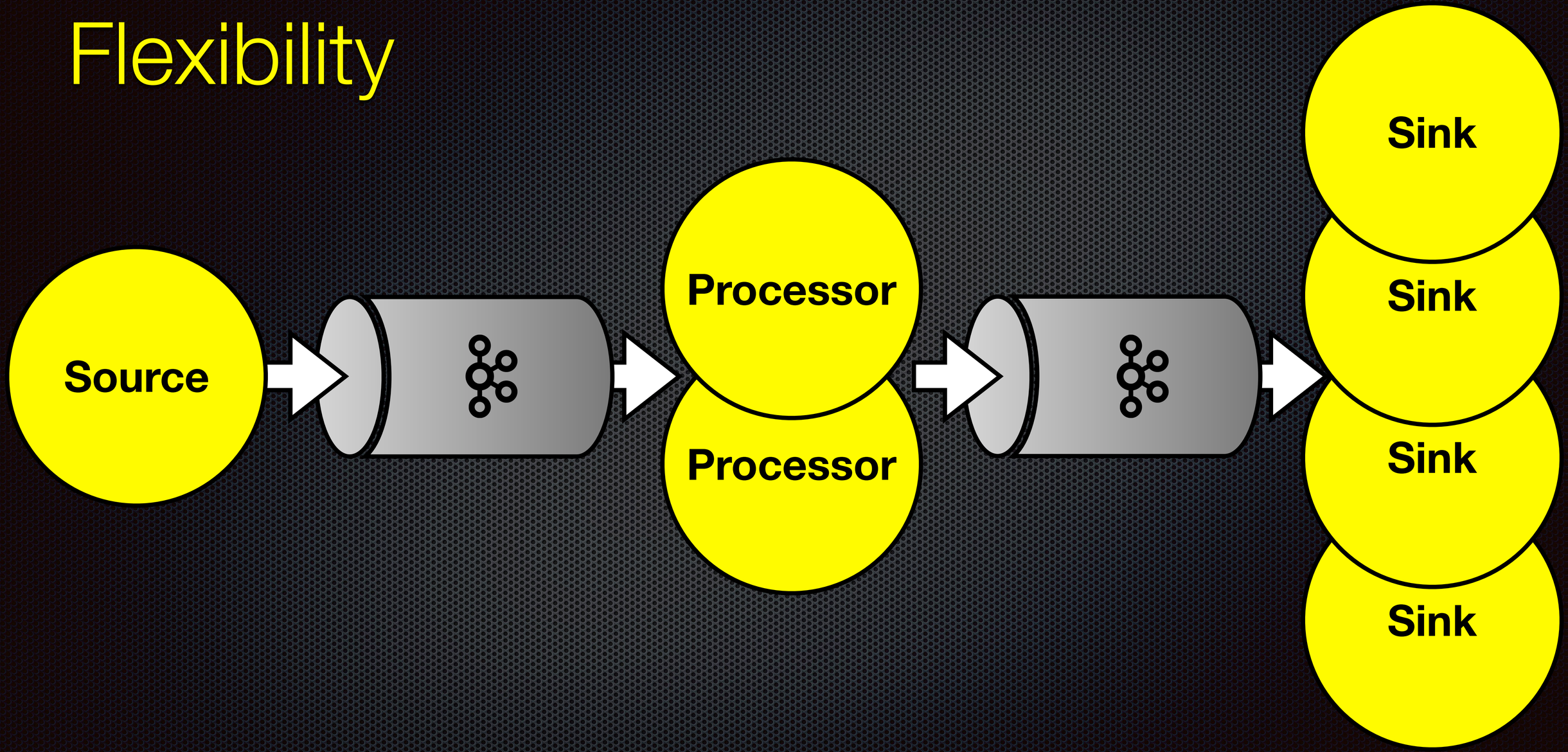
Power



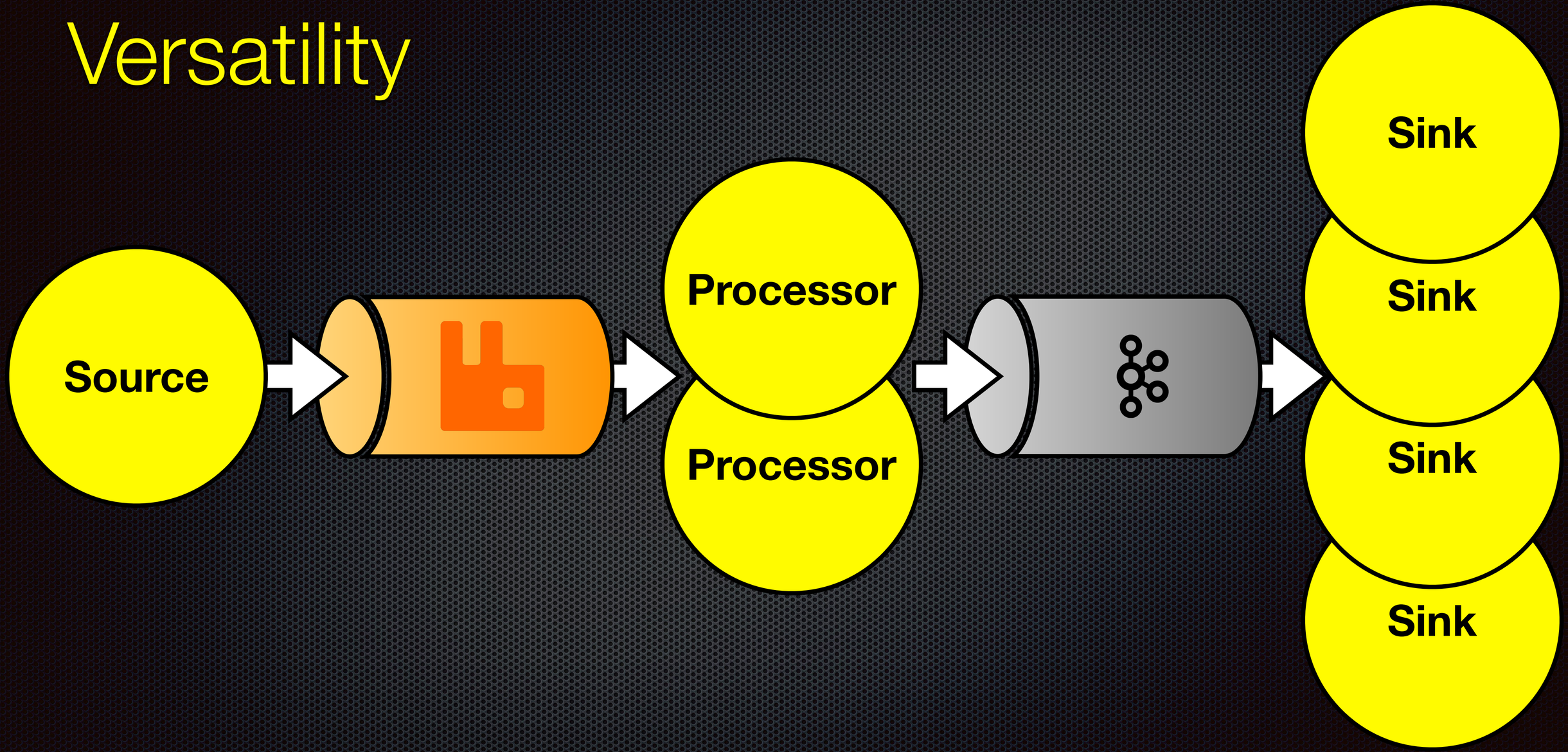
Scalability



Flexibility



Versatility



Let’s code!

@mkheck

www.thehecklers.com

Pivotal

Resources

- https://github.com/mkheck/drinking-from-the-stream
- https://cloud.spring.io/spring-cloud-stream/
- mark@thehecklers.com, mheckler@pivotal.io
- @mkheck on Twitter

Kotlin fan? https://github.com/mkheck/drinking-from-the-stream-kotlin