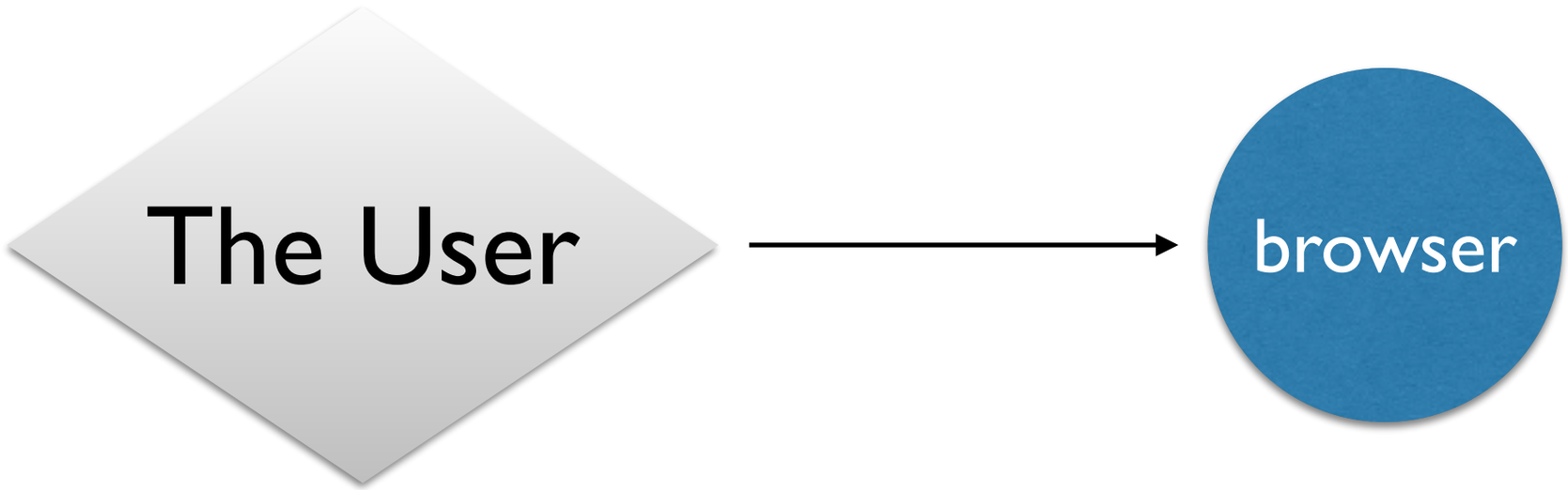
Go Reactive

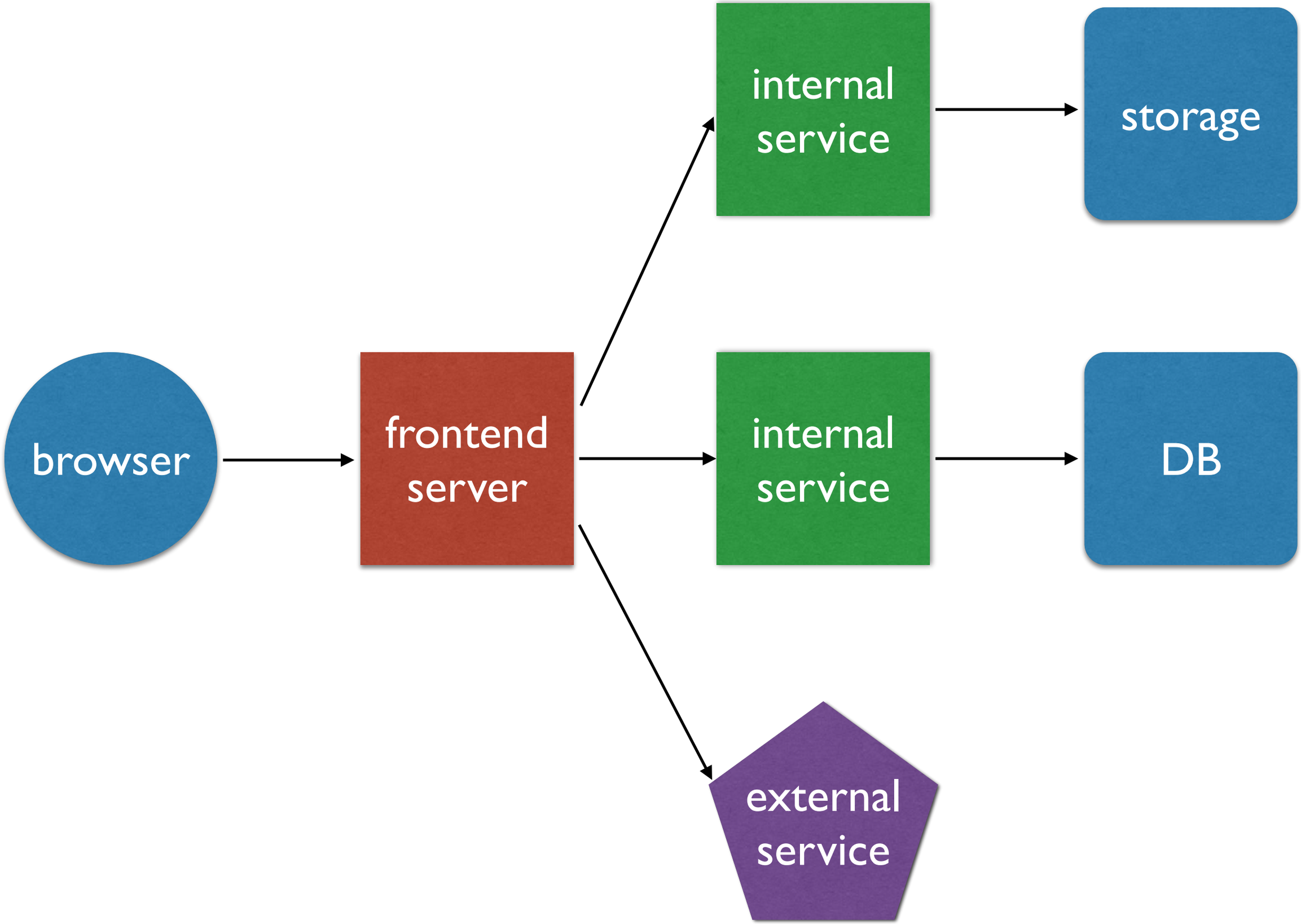
Blueprint for Future Applications

Dr. Roland Kuhn — Akka Tech Lead, Typesafe Inc.

Starting Point:

The User


The User
browser


internal service
storage
browser
frontend server
internal service
DB
external service

Responsiveness

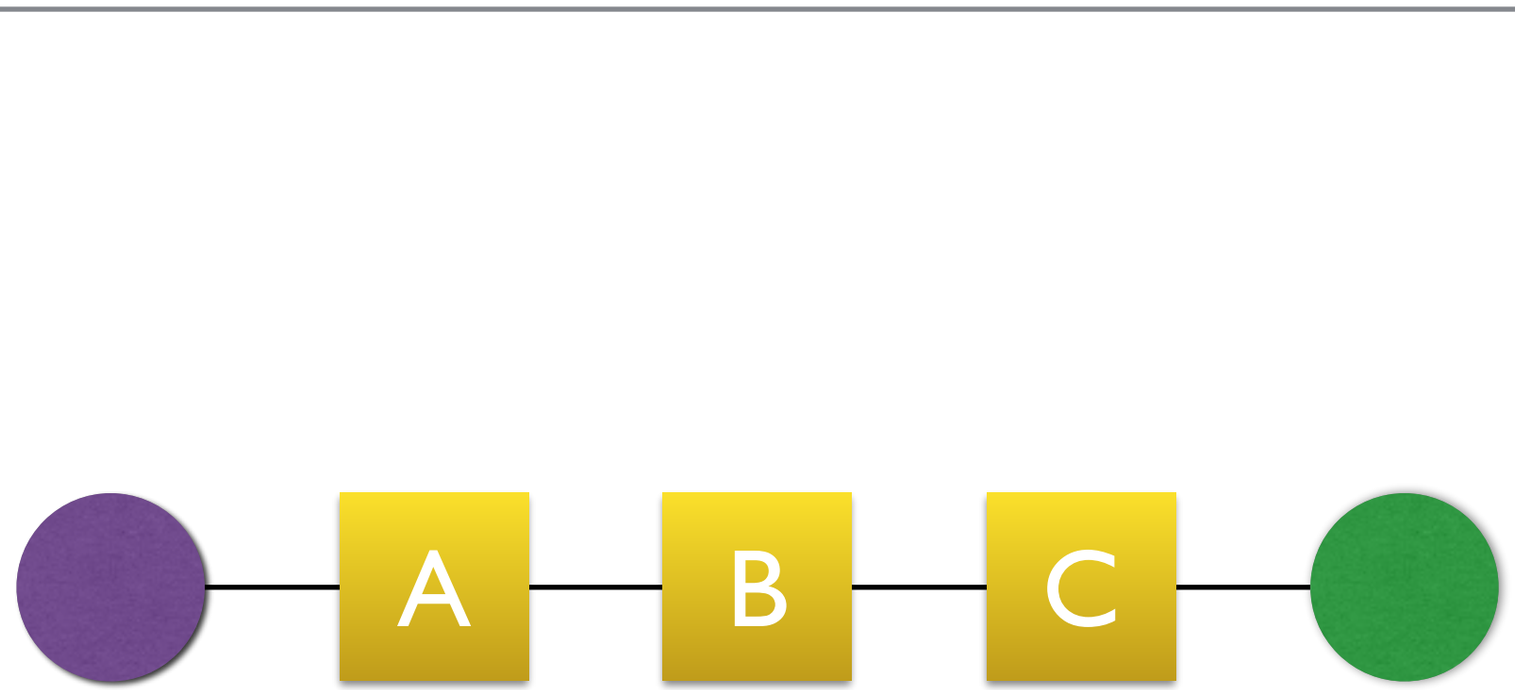
always available
interactive
(near) real-time

Bounded Latency

- fan-out in parallel and aggregate
- use circuit breakers for graceful degradation
- use bounded queues, measure flow rates

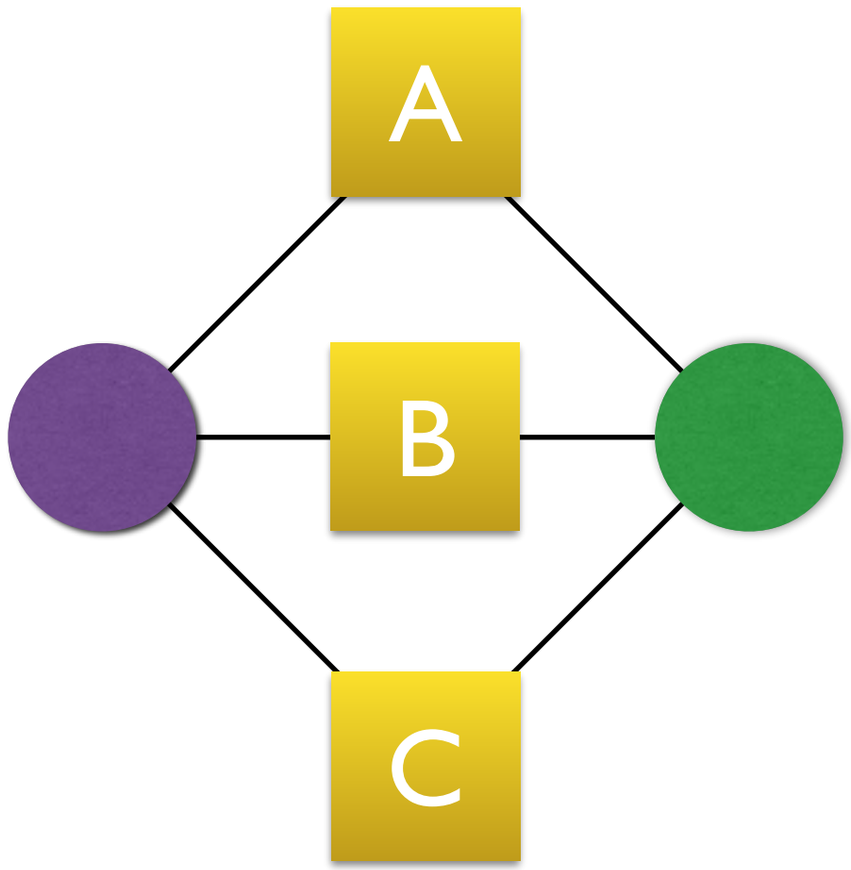
Bounded Latency

- fan
- use
- use



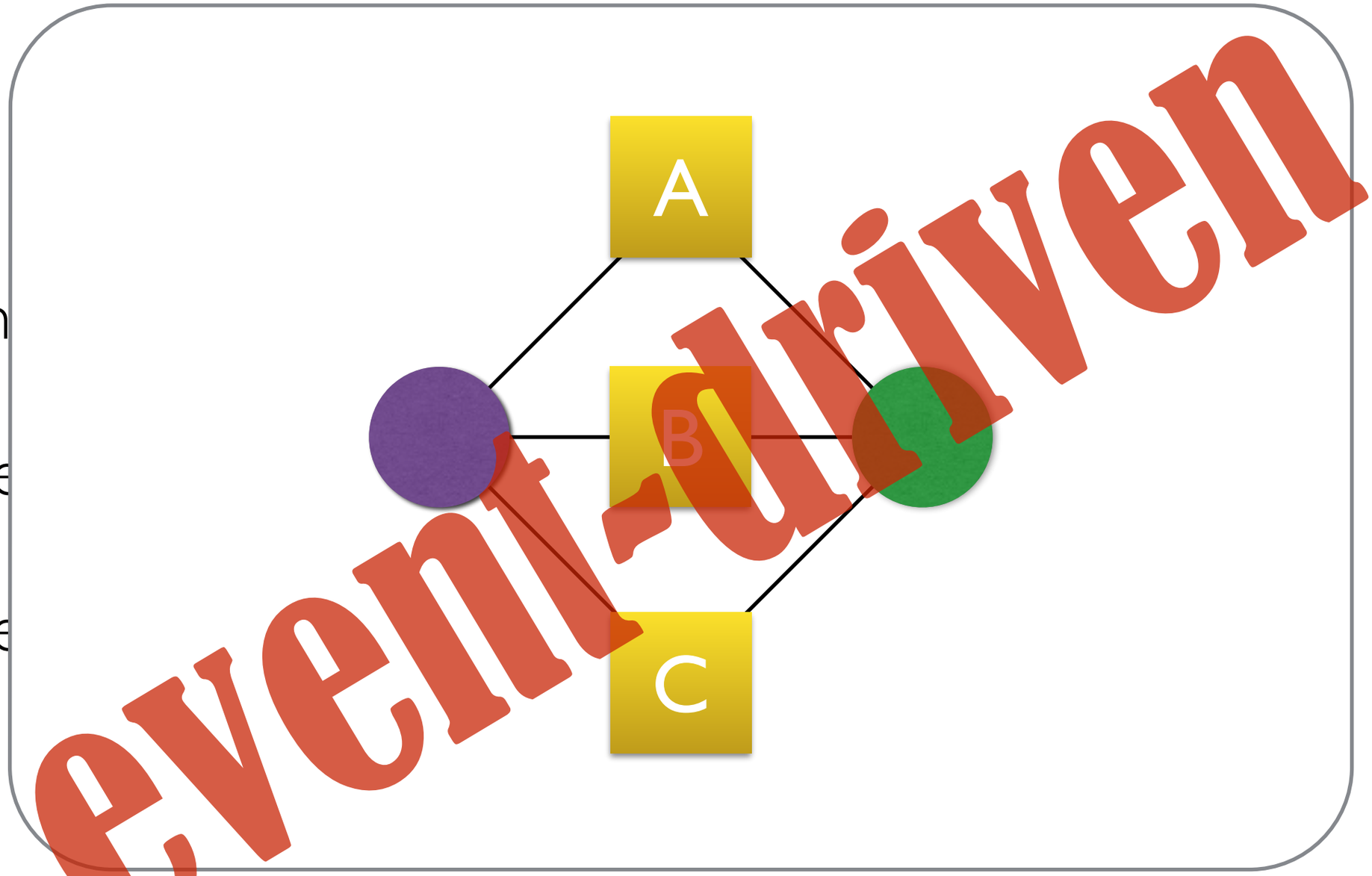
Bounded Latency

- fan
- use
- use



Bounded Latency

- fan
- use
- use

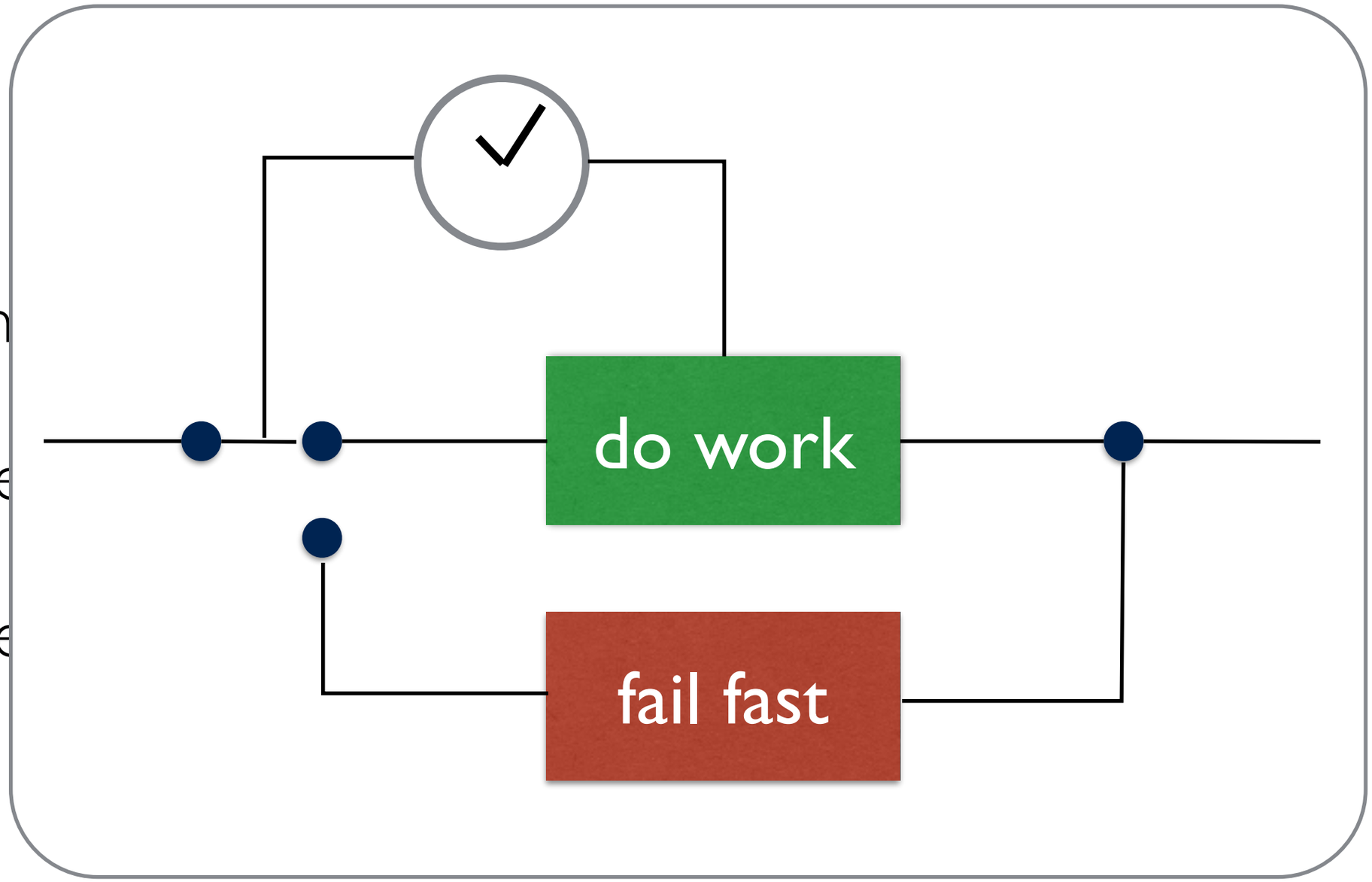


Bounded Latency

- fan-out in parallel and aggregate
- use circuit breakers for graceful degradation
- use bounded queues, measure flow rates

Bounded Latency

- fan
- use
- use



Bounded Latency

- fan-out in parallel and aggregate
- use circuit breakers for graceful degradation
- use bounded queues, measure flow rates

Bounded Latency

- fan
- use
- use

Little's Law

QueueLength = Rate • ProcessingTime

Latency = QueueLength • ProcessingTime

(averaging implied for all values)

Bounded Latency

- fan
- use
- use

Little's Law

QueueLength = Rate • ProcessingTime

Latency = QueueLength • ProcessingTime

(averaging implied for all values)

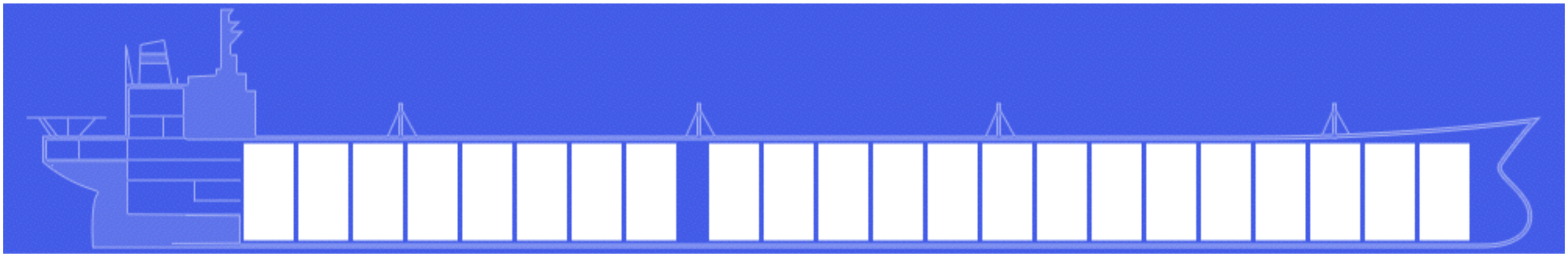
Handle Failure

- software will fail
- hardware will fail
- humans will fail
- system still needs to respond ➟ **resilience**


DELL
DELL
DELL

Asynchronous Failure

- parallel fan-out & distribution ➟ asynchronous execution
- compartmentalization & isolation

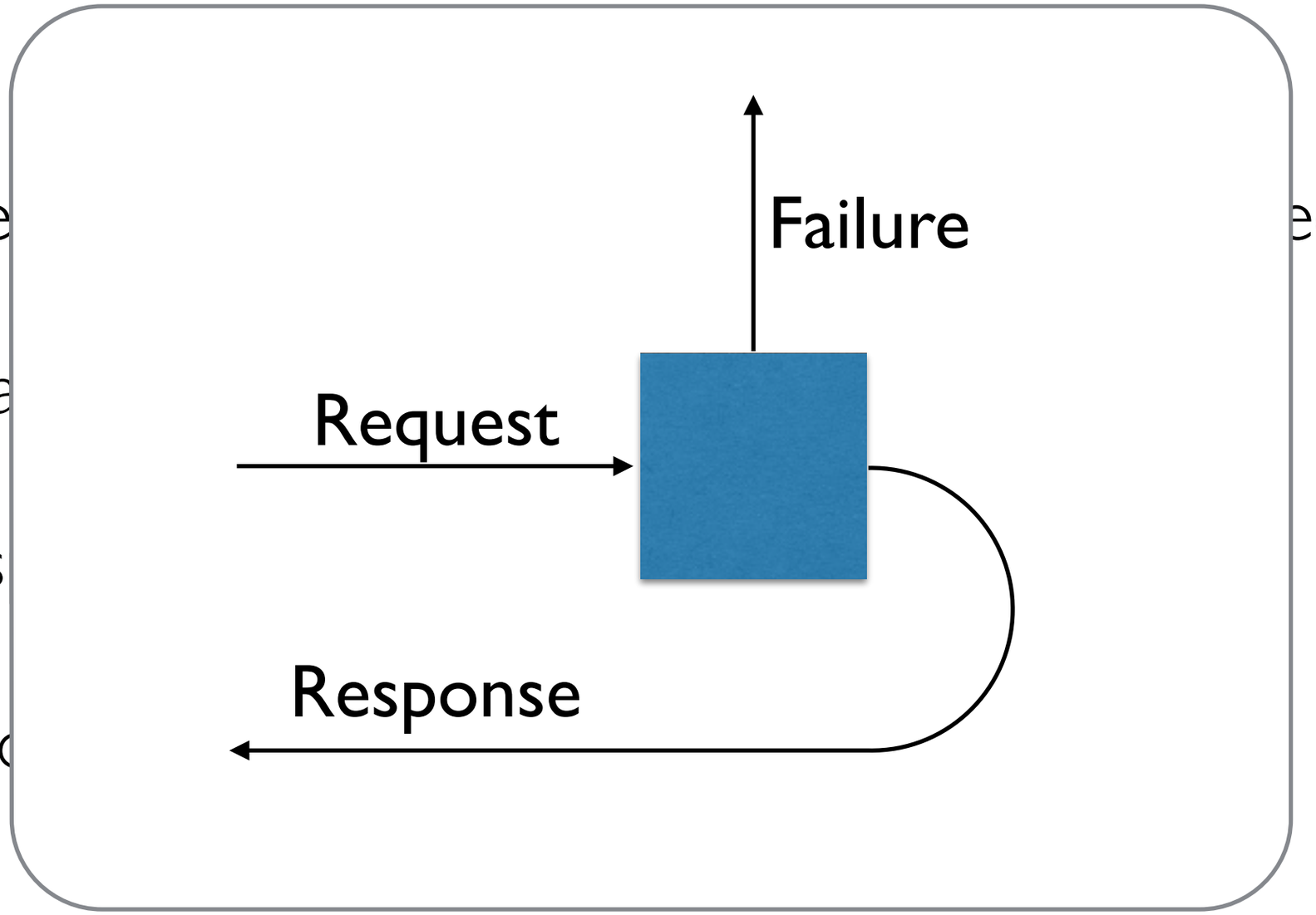


Asynchronous Failure

- parallel fan-out & distribution ➟ asynchronous execution
- compartmentalization & isolation
- no response? ➟ timeout events
- **s**omeone **e**lse's exce**p**tion? ➟ **supervision**

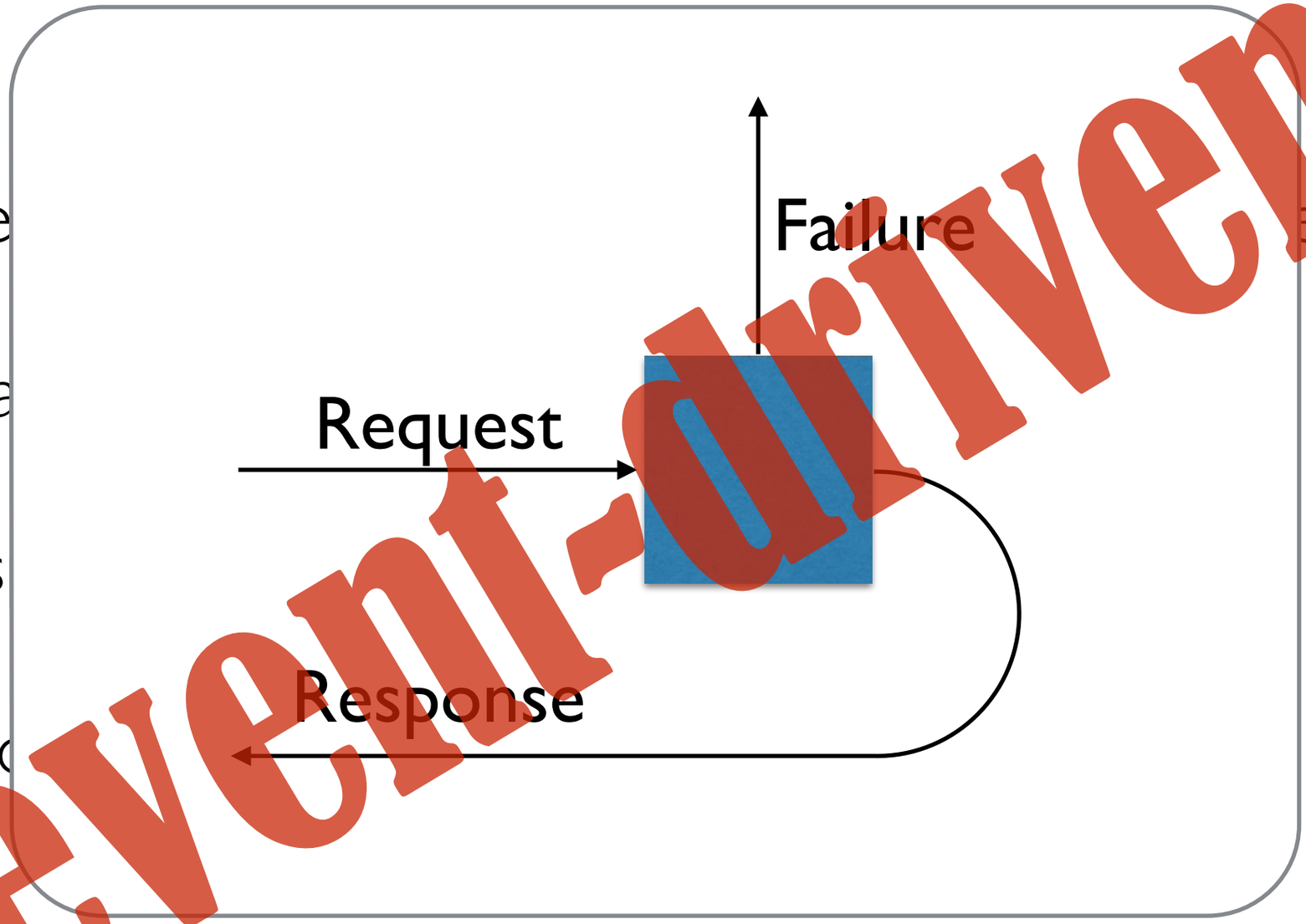
Asynchronous Failure

- paralle ecution
- compa
- no res
- **S**omec



Asynchronous Failure

- paralle
- compa
- no res
- **S**omec

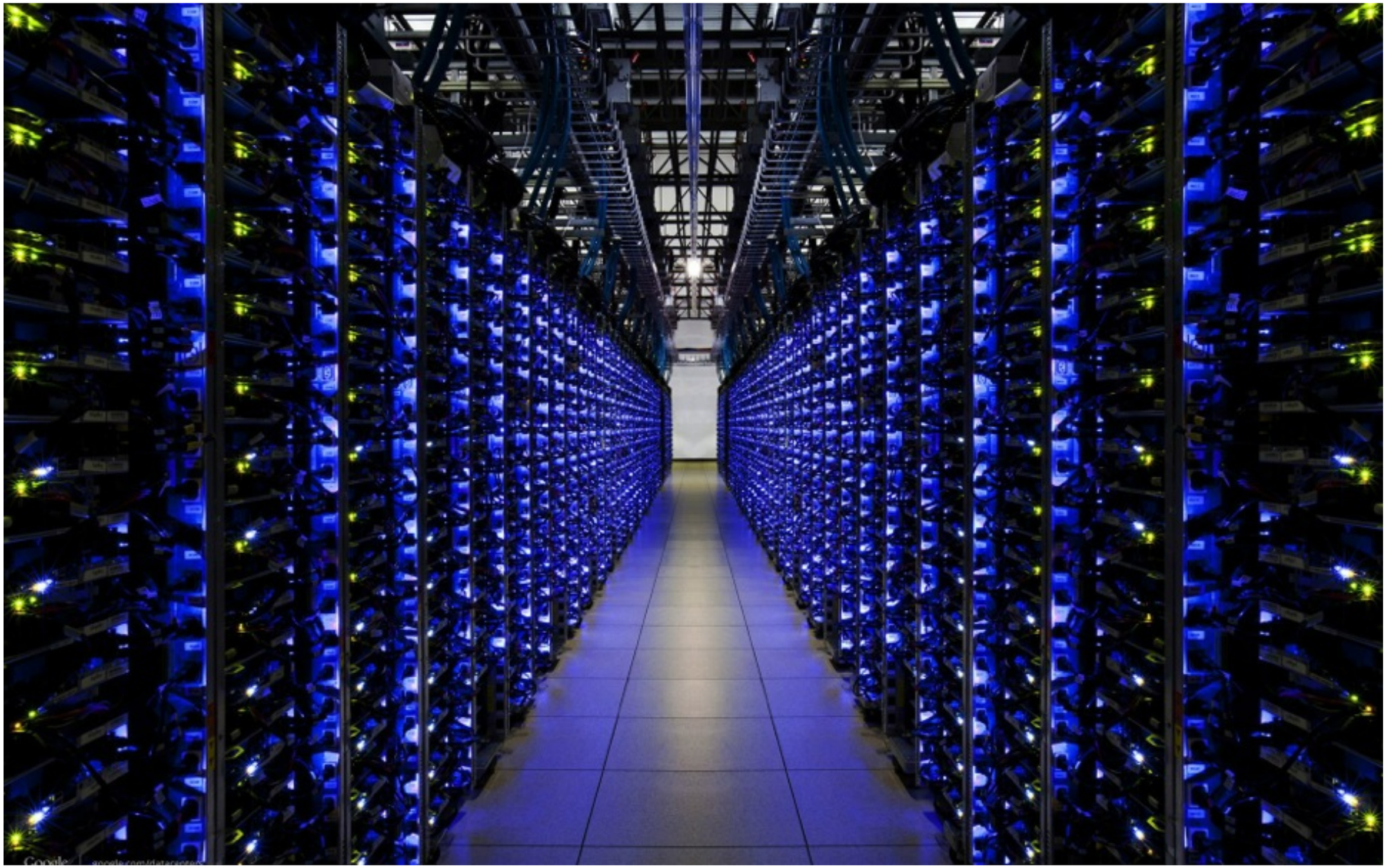


Asynchronous Failure

- parallel fan-out & distribution ➟ asynchronous execution
- compartmentalization & isolation
- no response? ➟ timeout events
- **s**omeone **e**lse's exce**p**tion? ➟ **supervision**
- **location transparency** ➟ seamless resilience

Handle Load

Handle Load



Handle Load

- partition incoming work for distribution
- scale capacity up and down on demand
- supervise and adapt
- **location transparency** ➟ seamless scalability

Handle Load

- partition incoming work for distribution
- scale capacity up and down on demand
- supervise and adapt
- **location transparency** ➟ seamless scalability

Consequences

- distribution & scalability ➟ loss of strong consistency

- CAP theorem? — not as relevant as you think

- eventual consistency ➟ gossip, heartbeats, dissemination

(http://pbs.cs.berkeley.edu)

Consequences

- distribution & scalability ➟ loss of strong consistency
- CAP theorem? — not as relevant as you think
- eventual consistency ➟ gossip, heartbeats, dissemination

(http://pbs.cs.berkeley.edu)

Corollary

- Reactive needs to be applied all the way down
- Polyglot deployments demand collaboration

But what about us,
the developers?

Trust the Machine

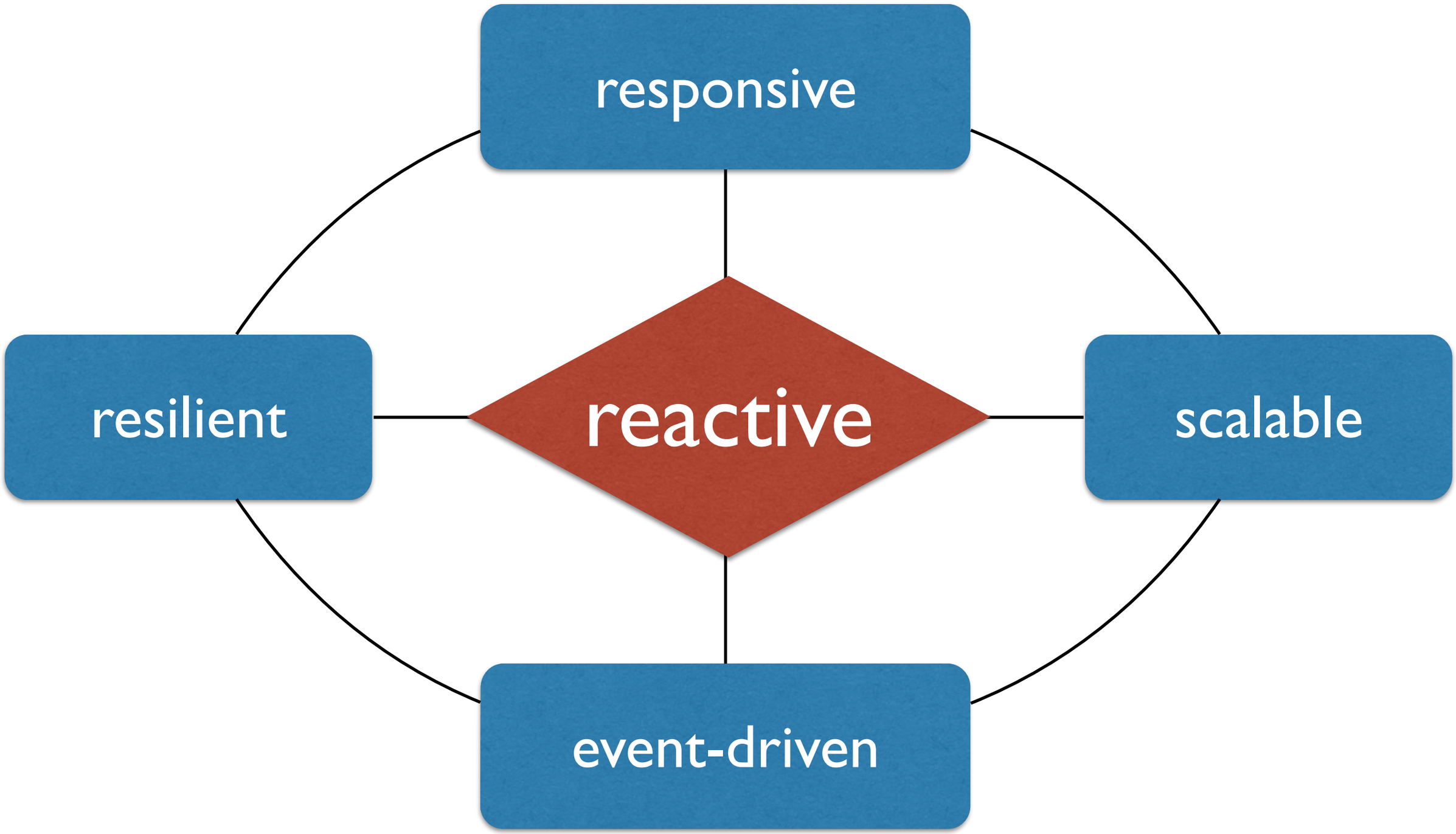
- rethink the architecture
- break out of the synchronous blocking prison
- asynchronous program flow ➠ no step-through debugging
- **loose coupling**

Simple Building Blocks

- clean business logic separate from failure handling
- distributable units of work
- effortless parallelization
- less assumptions ➟ lower maintenance cost

Simple Building Blocks

- clean business logic separate from failure handling
- distributable units of work
- effortless parallelization
- less assumptions ➟ lower maintenance cost
- independent agents ➟ **fun to work with!**



http://reactivemanifesto.org/