SWIMming

In the microservices Ocean

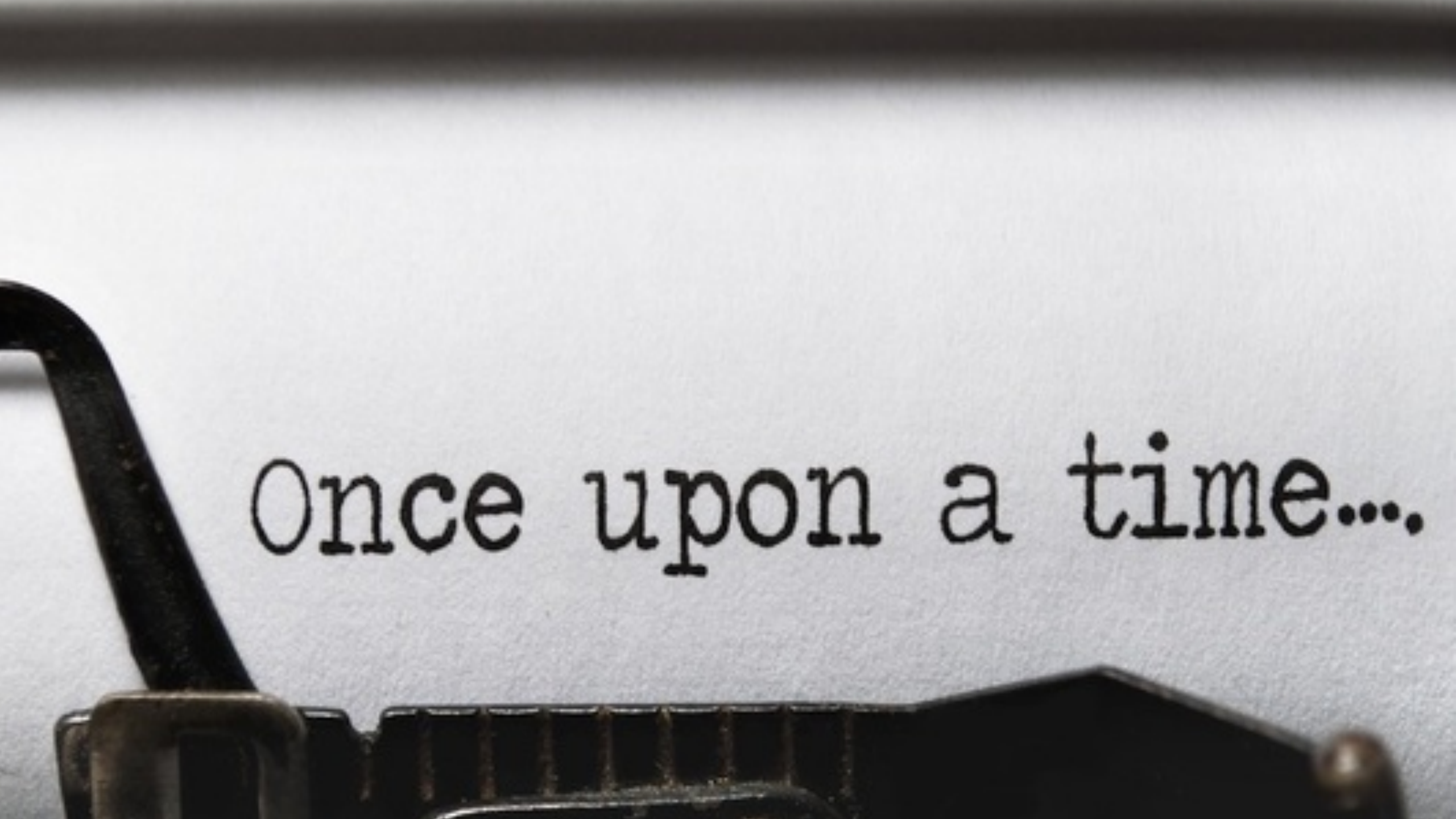

nearForm

@lucamaraschi

node/post-mortem

Resilient and scalable

With NO maintenance


Once upon a time....


GEBERIT

Divide et Impera

(Philip of Macedonia)

Microservices

Microservices

===

Stateless

Stateless

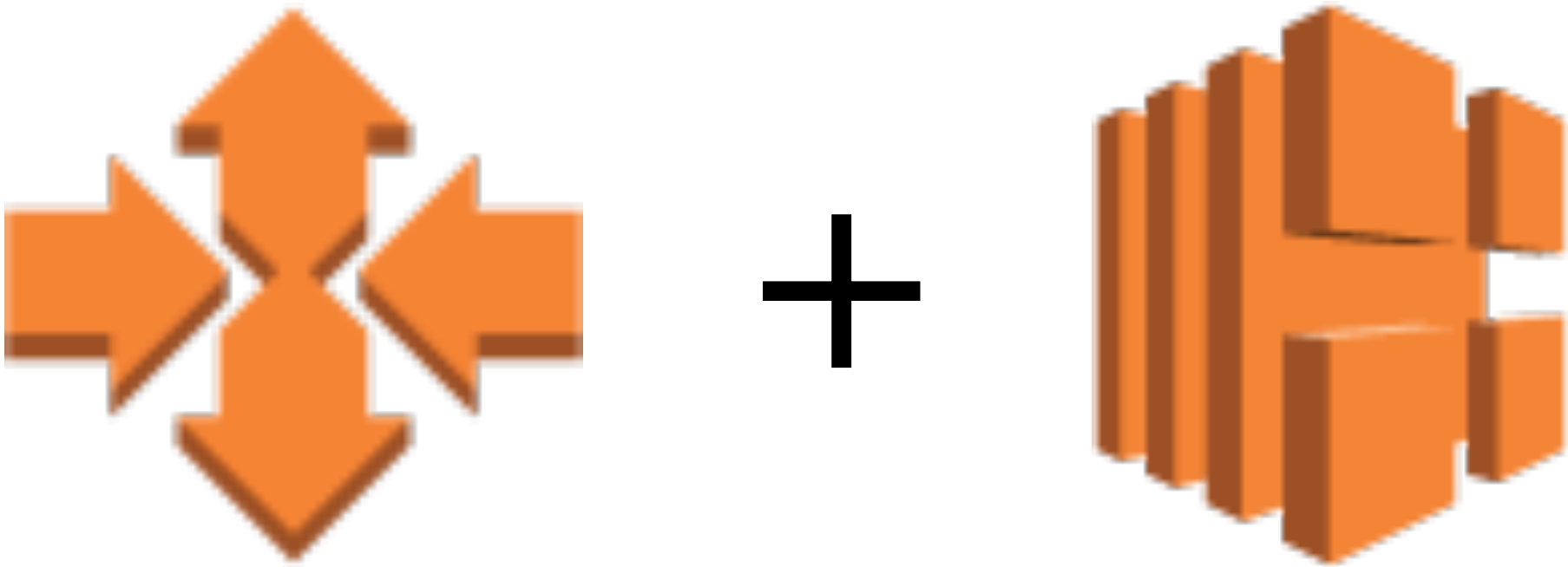
!==

Resilient

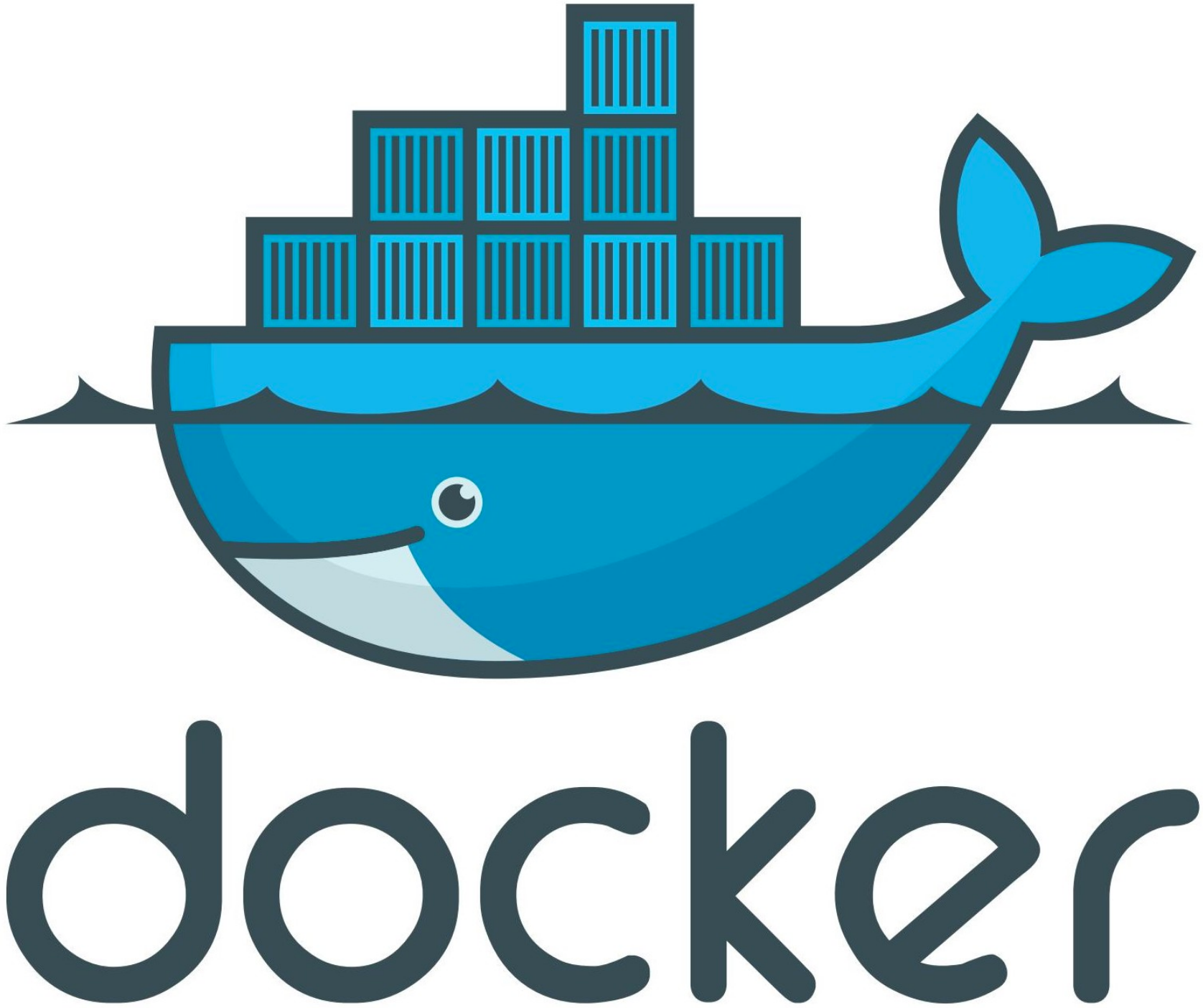

Success


WAL-MART
WAL-MART
WAL-MART


AWS



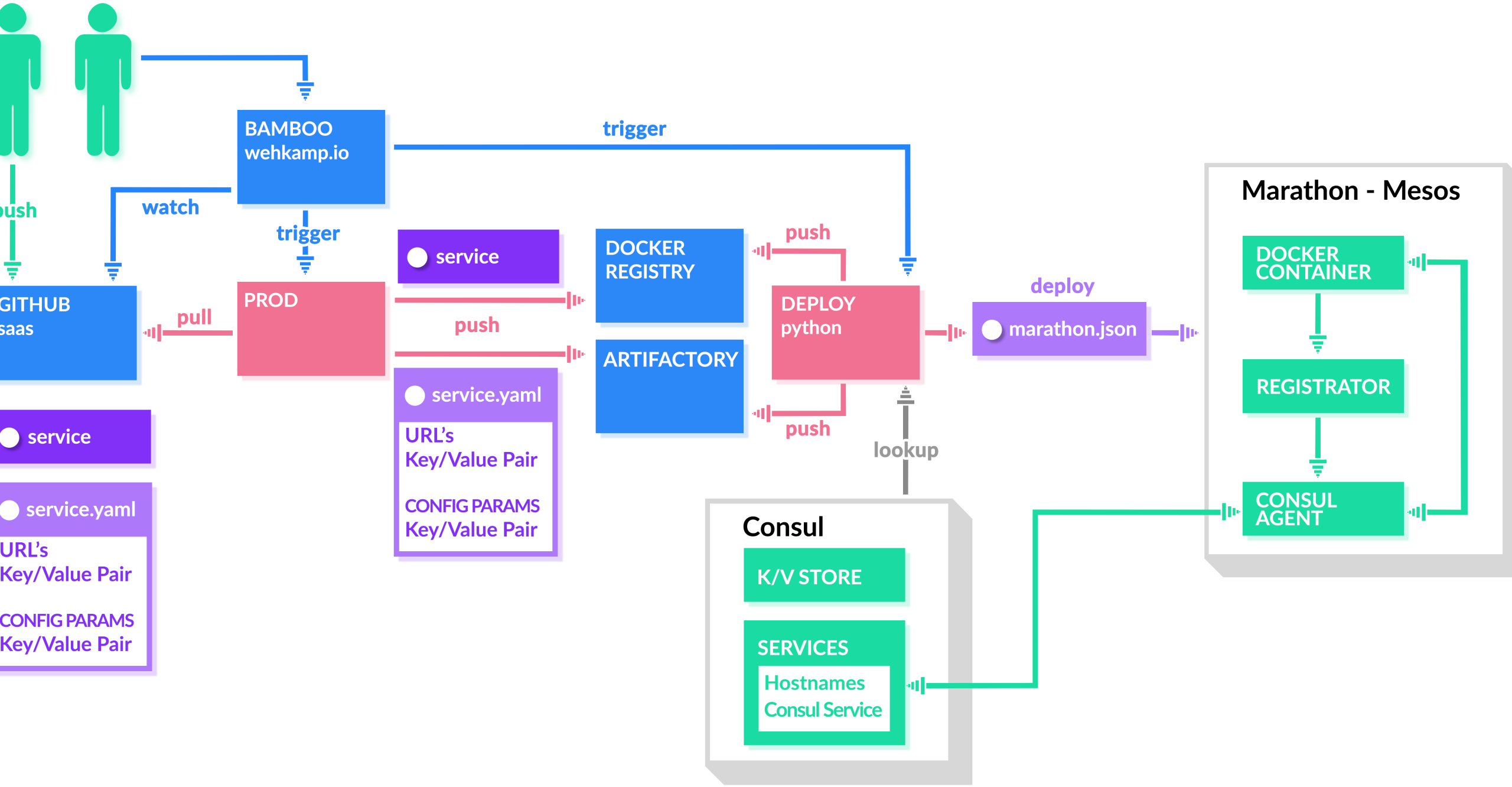

TT 213 PSA
TT 206 PSA
PSA
TT 191 PSA
TT 124
TT 126
TT 208 PSA
TT 208
TT 198 PSA
TT 172 PSA
TT 80
PSA 80
TT 166
MGM
TT 095
PSA 95


docker

Service Discovery

No RESILIENCY

No Fault Tolerance


BAMBOO
wehkamp.io
trigger
watch
trigger
push
GITHUB
saas
pull
PROD
service
push
service.yaml
URL's
Key/Value Pair
CONFIG PARAMS
Key/Value Pair
DOCKER
REGISTRY
ARTIFACTORY
push
push
DEPLOY
python
deploy
marathon.json
lookup
service
service.yaml
URL's
Key/Value Pair
CONFIG PARAMS
Key/Value Pair
Consul
K/V STORE
SERVICES
Hostnames
Consul Service
Marathon - Mesos
DOCKER
CONTAINER
REGISTRATOR
CONSUL
AGENT

Failure Detection

by heartbeatting

PROBLEM

State of the cluster

Might change all the time

SWIM: *S*calable *W*eakly-consistent *I*nfection-style Process Group *M*embership Protocol

Abhinandan Das, Indranil Gupta, Ashish Motivala*
Dept. of Computer Science, Cornell University
Ithaca NY 14853 USA
{asdas,gupta,ashish}@cs.cornell.edu

Abstract

1. Introduction

As you swim lazily through the milieu,

Scalable

Weakly consistent

Infection style

Membership protocol

Weakly consistent

vs. strongly consistent

Failure detection

Over the cluster

Information dissemination

Equal distribution

of the workload

Born for Distributed Systems

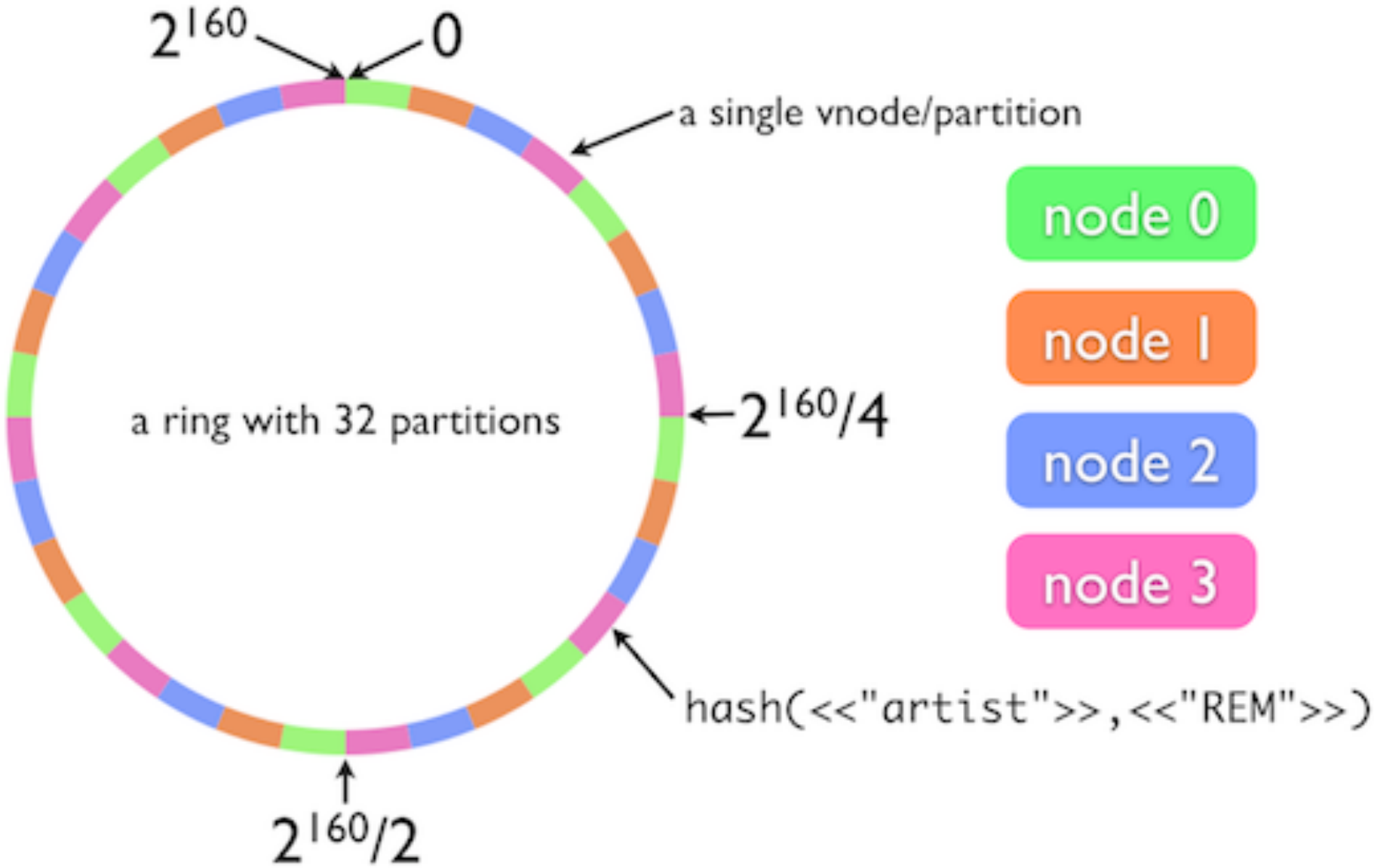

ReactionGIFS.me

But...

Adding a node to the cluster

How to rebalance the topology?

Consistent Hashring


2^{160}
0
a single vnode/partition
node 0
node 1
node 2
node 3
a ring with 32 partitions
$2^{160}/4$
hash(<<"artist">>,<<"REM">>)
$2^{160}/2$

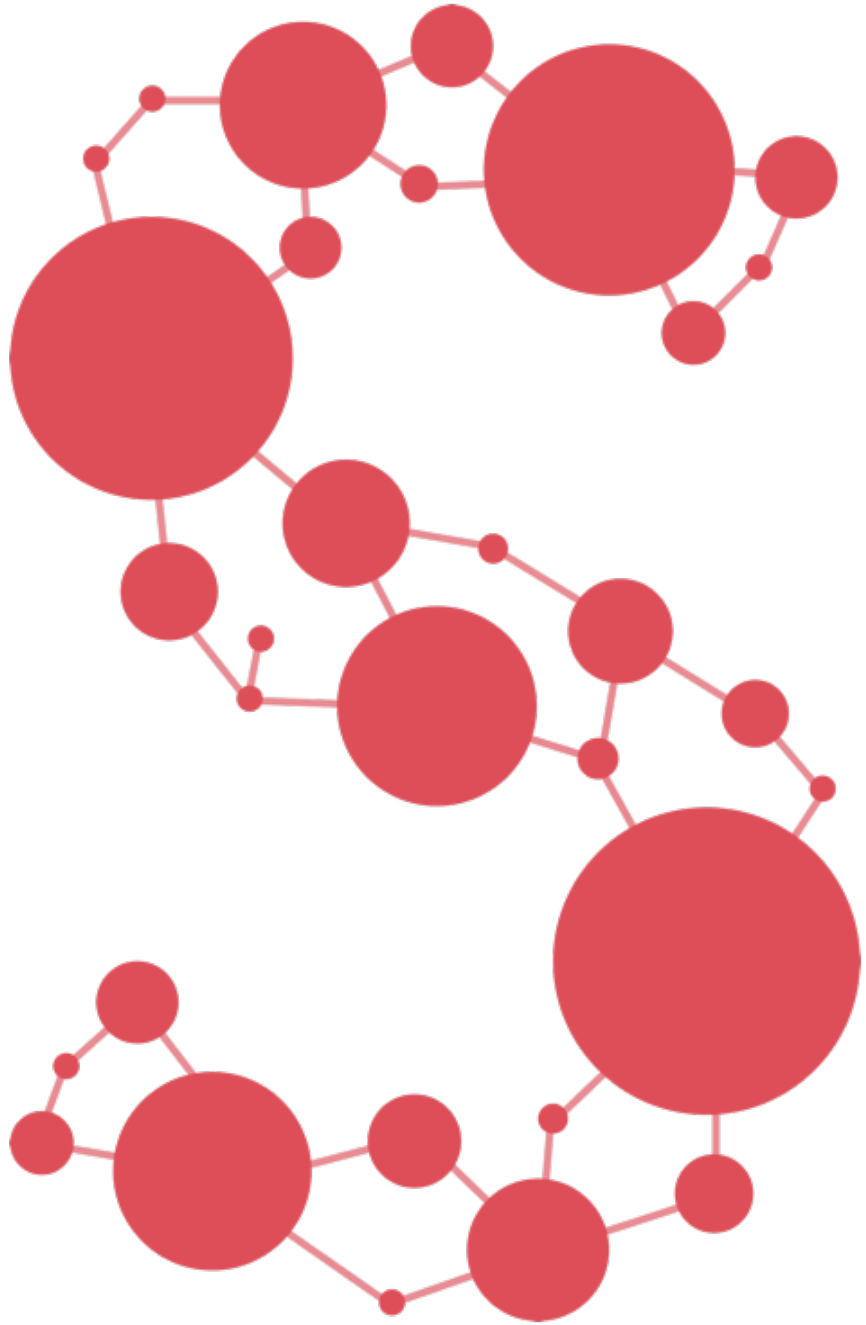
Partitioning

Basically sharding

Predictable

Elasticity

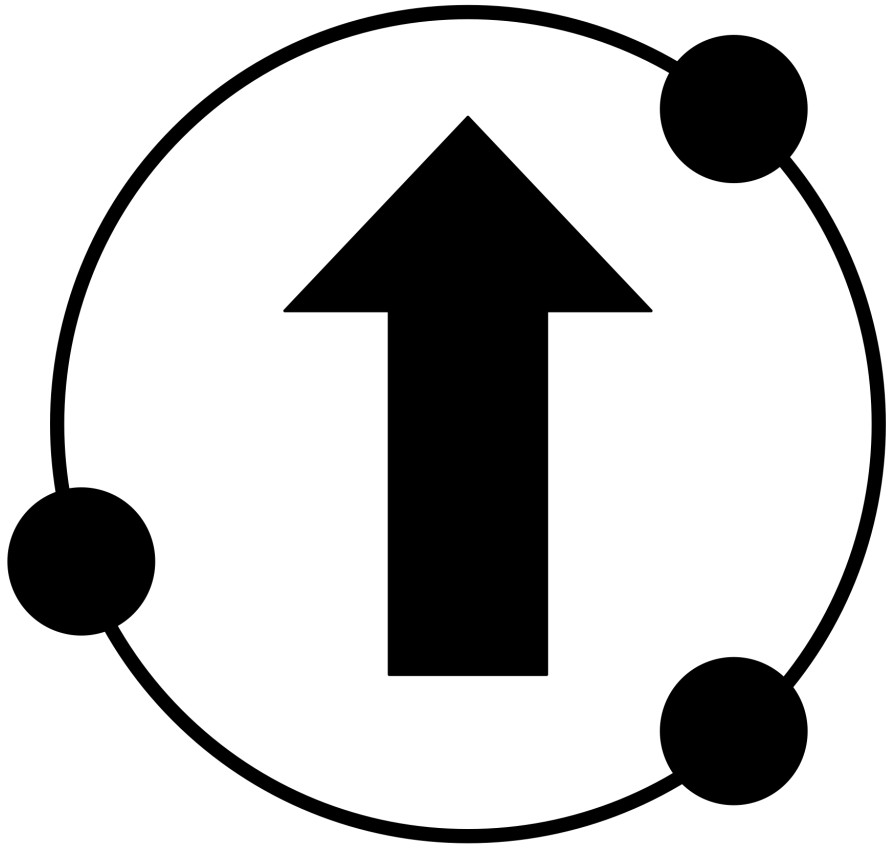
Distribution of the application state



ringpop

Scalable, fault-tolerant application-layer sharding

FORK ME ON GITHUB



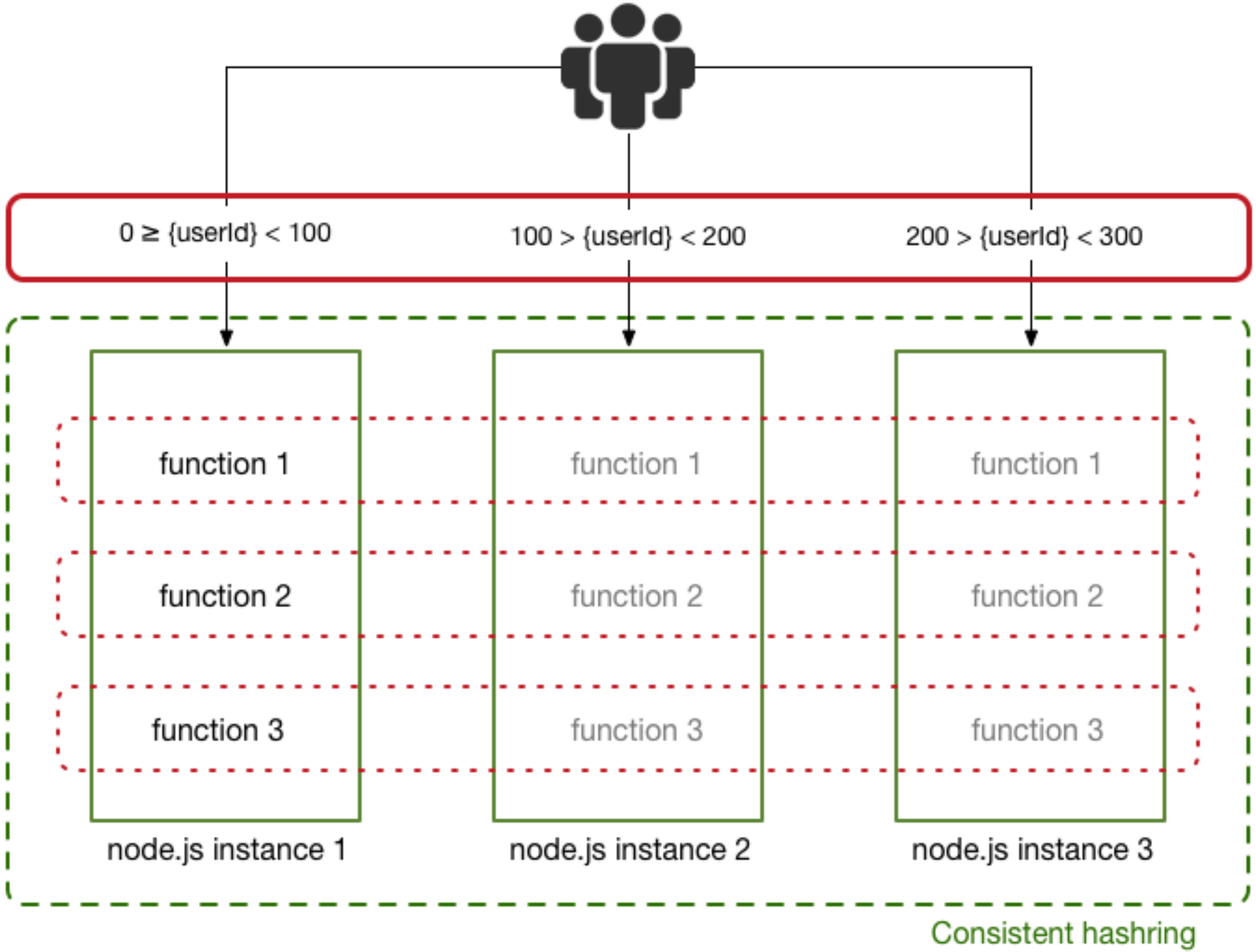
Upring

application-level
sharding for Node.js

Appplication Level SHARDING

Distribution

In an eventually consistent world


0 ≥ {userId} < 100
100 > {userId} < 200
200 > {userId} < 300
function 1
function 2
function 3
function 1
function 2
function 3
function 1
function 2
function 3
node.js instance 1
node.js instance 2
node.js instance 3
Consistent hashring

hyperbahn

Service discovery and routing for large-scale microservice operations

FORK ME ON GITHUB

Hyperbahn enables service discovery and routing for large-scale systems comprised of many microservices. Distributed, fault tolerant, and highly available, it lets one service find and communicate with others simply and reliably without having to know where those services run.


CONSUL

tchannel

Network multiplexing and framing protocol for RPC

FORK ME ON GITHUB

TChannel is a networking framing protocol used for general RPC, supporting out-of-order responses at extremely high performance where intermediaries can make a forwarding decision quickly

- Matt Sweeney -

WHY RPC?

Are FUNCTIONS

running in a cluster!

AWS Lambda

Serverless

It's the new HIPSTER

Function distribution

it's the future!

Book a clinic at

nearForm 's booth

GRAZIE!