ThoughtWorks®

PRINCIPLES OF MICROSERVICES

Sam Newman

Microxchg, Berlin 2015

ThoughtWorks®

@samnewman

There is no hyphen in “microservice”

@samnewman

micro-service

@samnewman

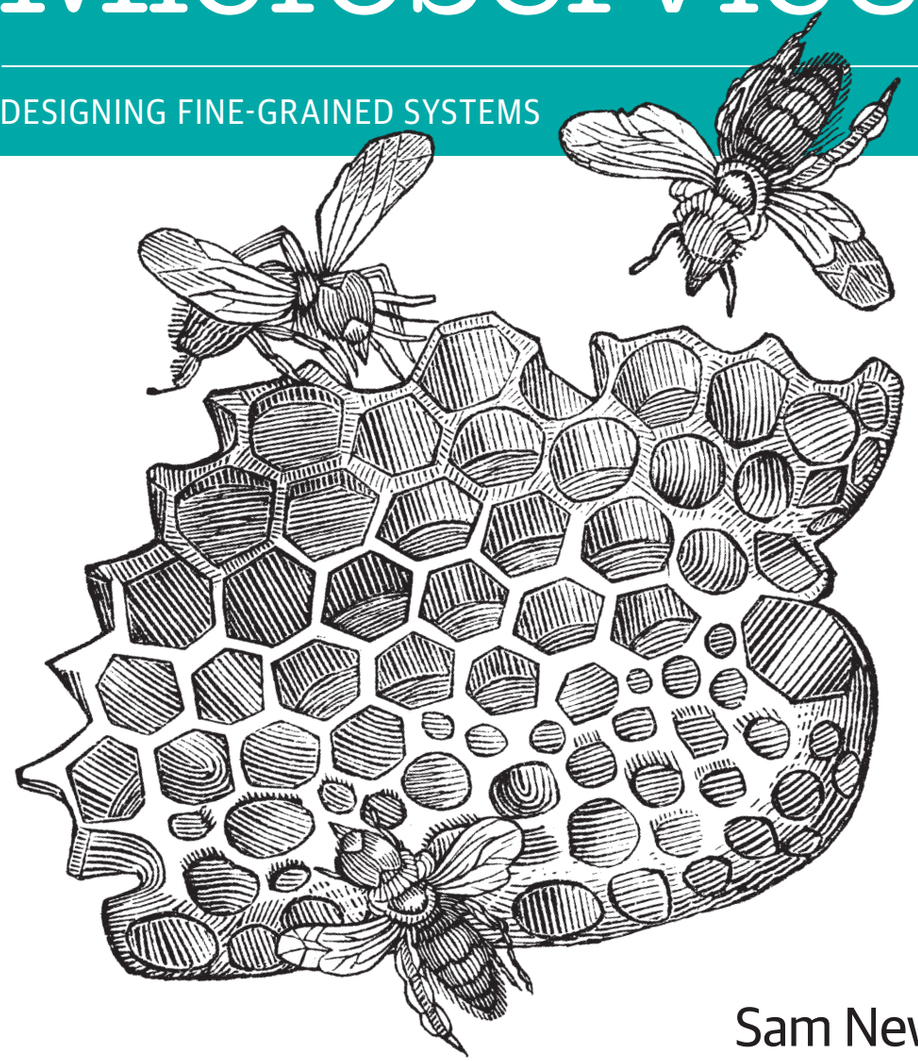
micro-service

@samnewman

O'REILLY®

Building Microservices

DESIGNING FINE-GRAINED SYSTEMS



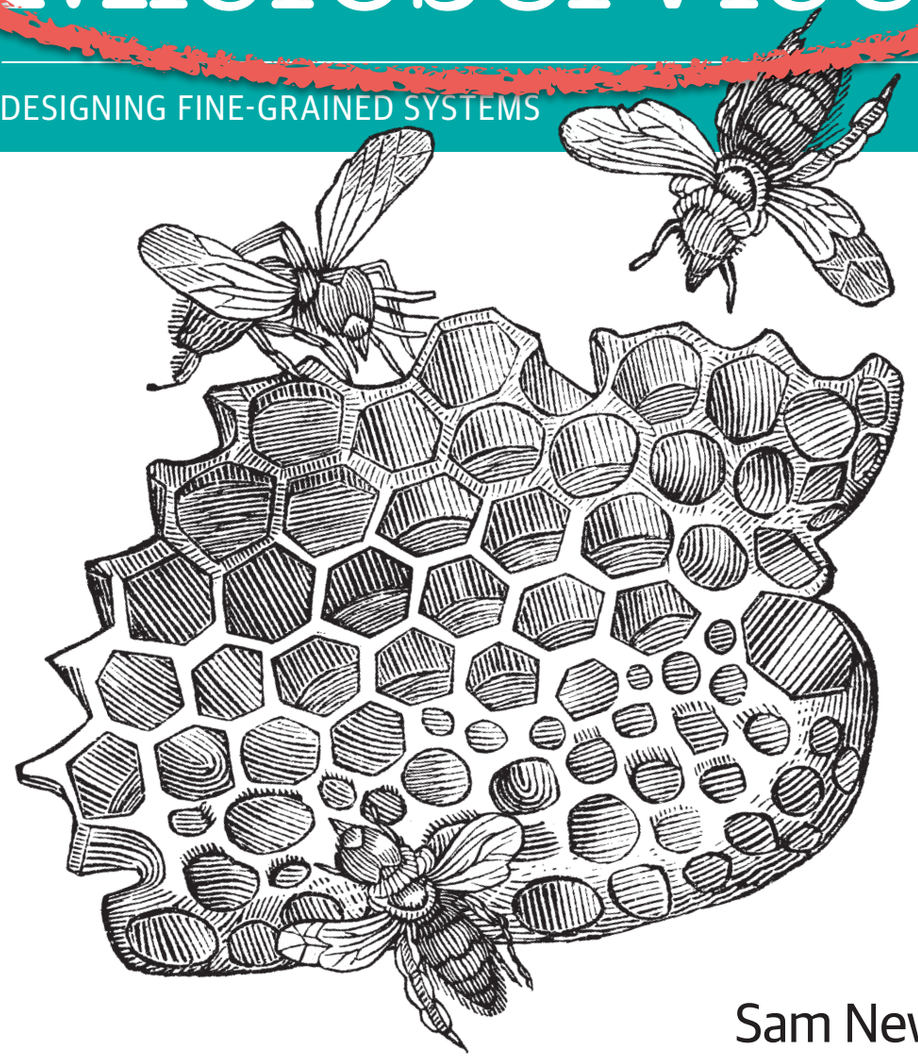
Sam Newman

@samnewman

O'REILLY®

Building Microservices

DESIGNING FINE-GRAINED SYSTEMS



Sam Newman

@samnewman

Article Talk

Microservices

From Wikipedia, the free encyclopedia

In computing, **microservices** is a software architecture design pattern, in which complex applications are composed o
language-agnostic APIs.[1] These services are small, highly decoupled and focus on doing a small task.[2]

Contents [hide]

Details [edit]

Properties of the Microservices architecture:

- The services are easy to replace
- Services are organized around capabilities, e.g. user interface frontend, recommendation, logistics, billing, etc.
- A microservices-based architecture
  - lends itself to a continuous delivery software development process.
  - is distinct from a Service-oriented architecture (SOA) in that the latter aims at integrating various (business) ap

Article Talk

Microservices

From Wikipedia, the free encyclopedia

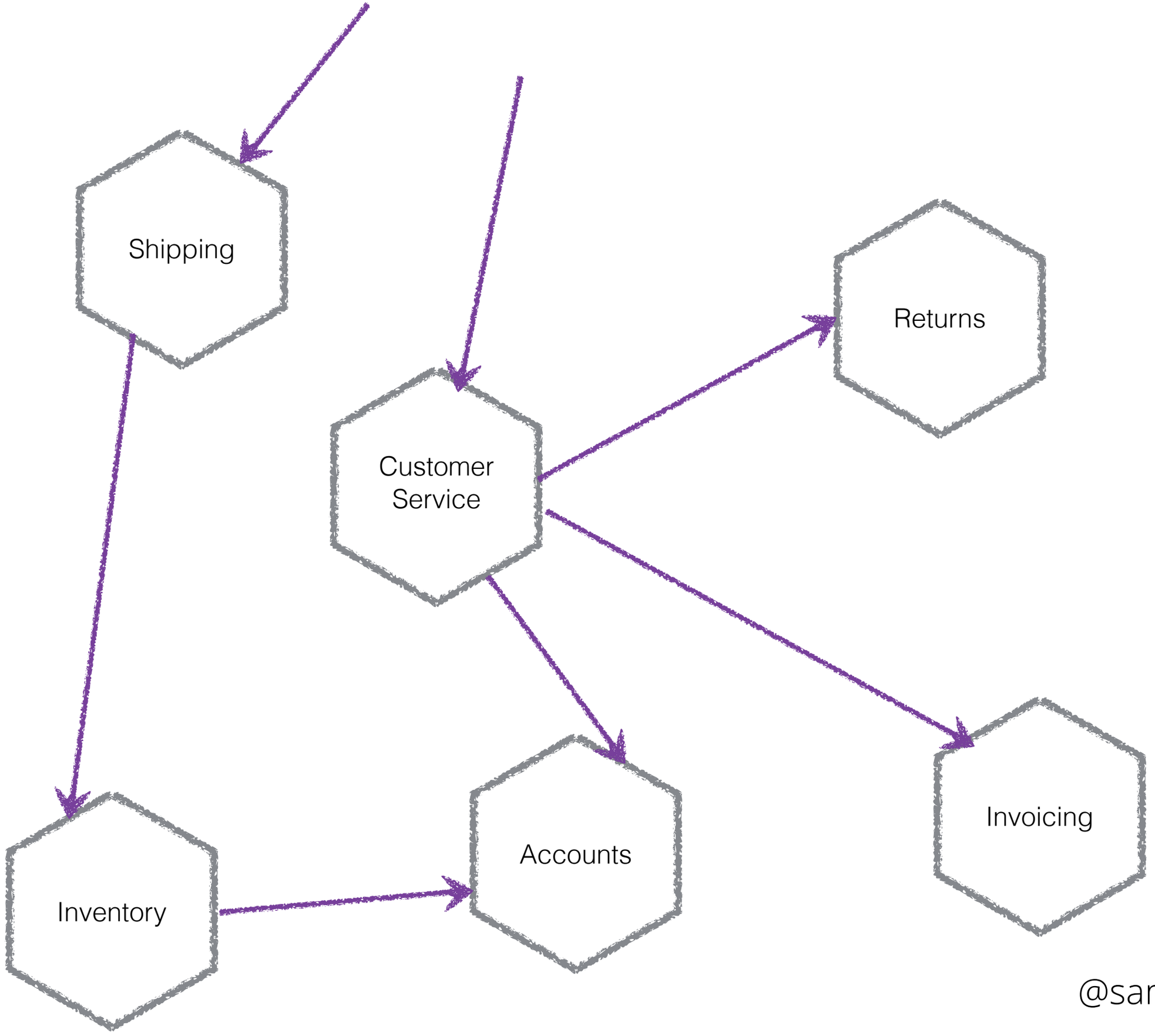
In computing, **microservices** is a software architecture design pattern, in which complex applications are composed o language-agnostic APIs.[1] These services are small, highly decoupled and focus on doing a small task.[2]

Contents [hide]

Details [edit]

Properties of the Microservices architecture:

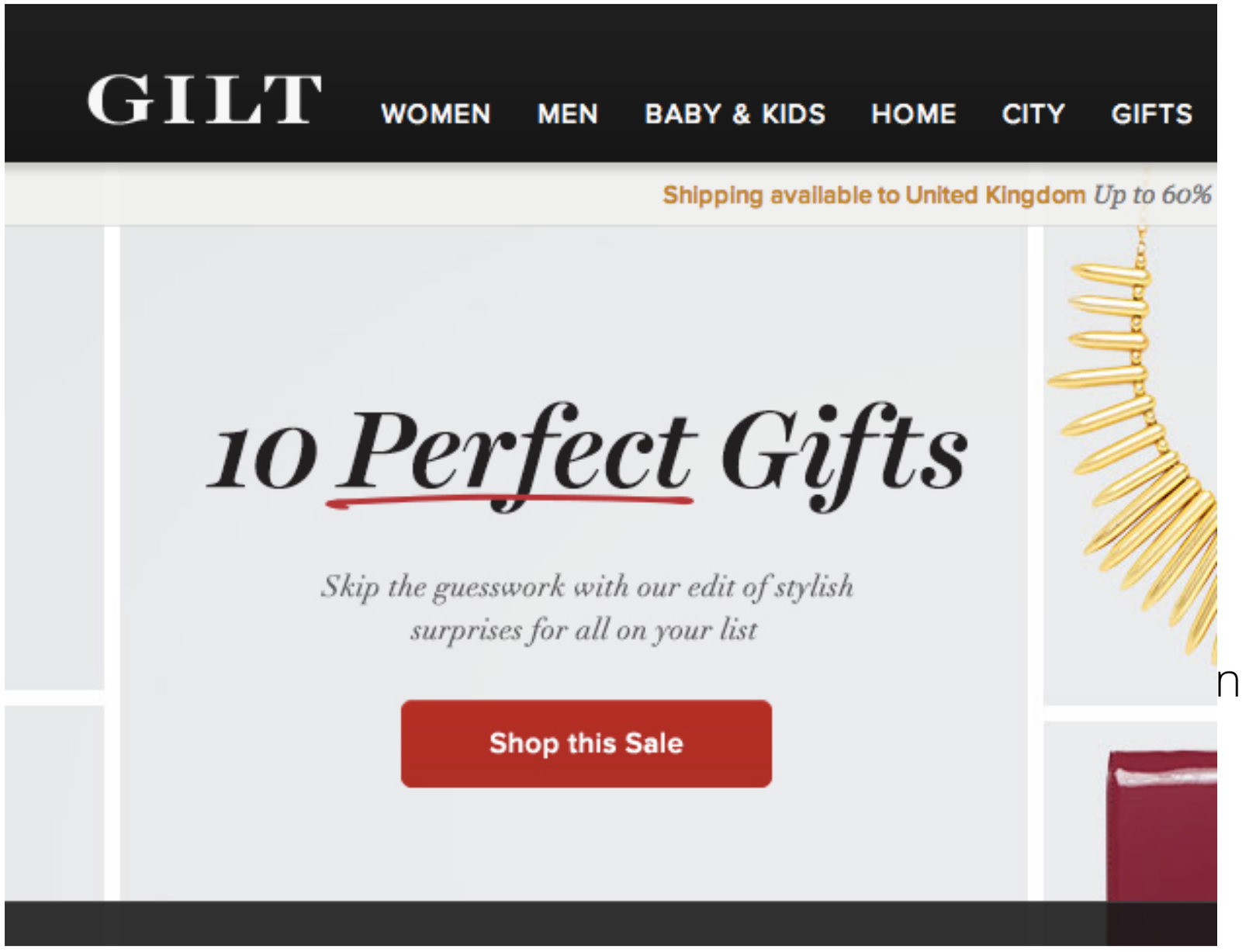
- The services are easy to replace
- Services are organized around capabilities, e.g. user interface frontend, recommendation, logistics, billing, etc.
- A microservices-based architecture
  - lends itself to a continuous delivery software development process.
  - is distinct from a Service-oriented architecture (SOA) in that the latter aims at integrating various (business) ap


Shipping
Customer Service
Returns
Inventory
Accounts
Invoicing

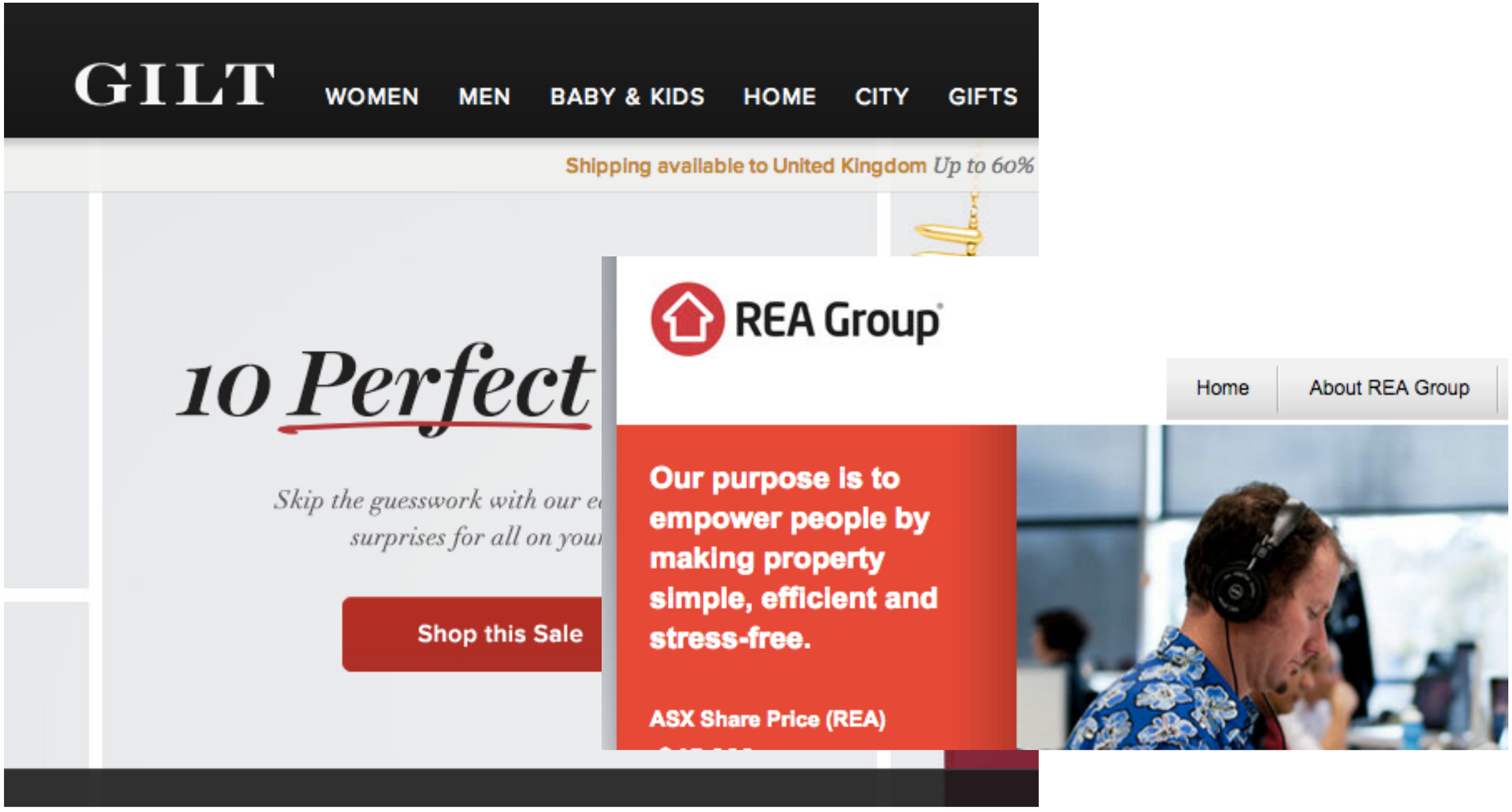
@samnewman

Small ***Autonomous*** services that ***work together***

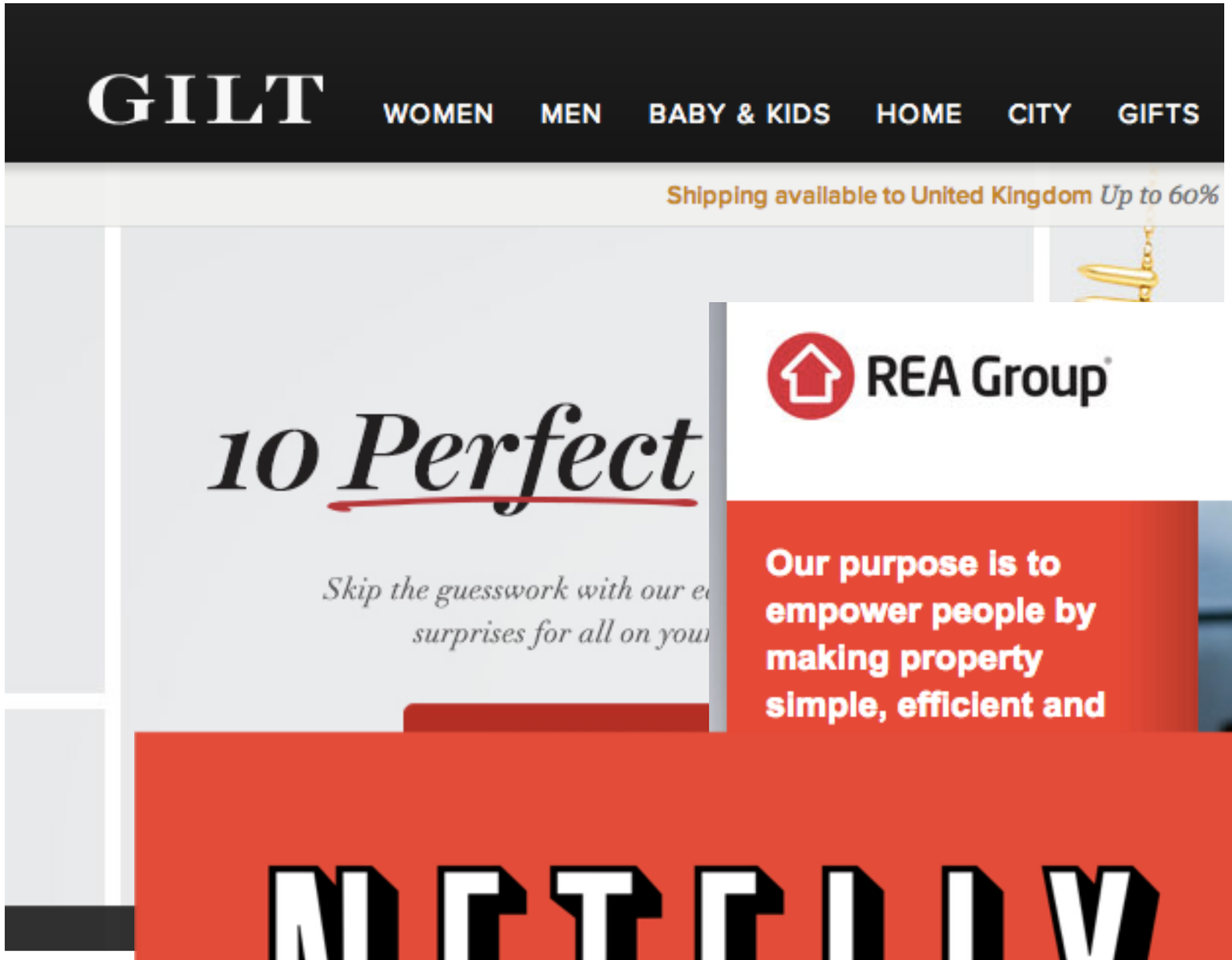
@samnewman

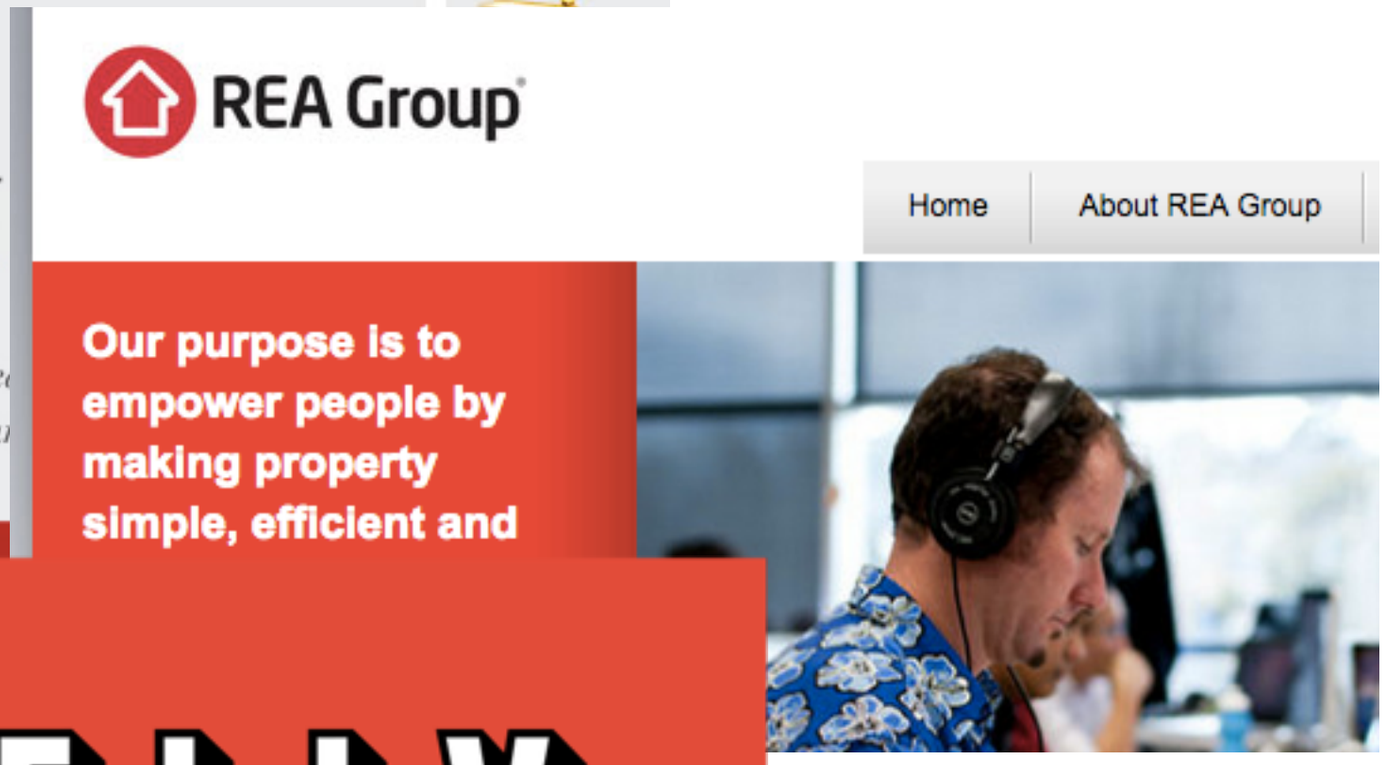

GILT
WOMEN
MEN
BABY & KIDS
HOME
CITY
GIFTS
Shipping available to United Kingdom Up to 60%
10 Perfect Gifts
Skip the guesswork with our edit of stylish surprises for all on your list
Shop this Sale

n


GILT
WOMEN
MEN
BABY & KIDS
HOME
CITY
GIFTS
Shipping available to United Kingdom Up to 60%
10 Perfect
Skip the guesswork with our e
surprises for all on you
Shop this Sale
REA Group
Home
About REA Group
Our purpose is to empower people by making property simple, efficient and stress-free.
ASX Share Price (REA)

@samnewman


GILT
WOMEN
MEN
BABY & KIDS
HOME
CITY
GIFTS
Shipping available to United Kingdom Up to 60%
10 Perfect
Skip the guesswork with our e
surprises for all on you


REA Group
Home
About REA Group
Our purpose is to empower people by making property simple, efficient and


NETFLIX

@samnewman

The Twelve Factors

I. Codebase
One codebase tracked in revision control, many deploys

II. Dependencies
Explicitly declare and isolate dependencies

III. Config
Store config in the environment

IV. Backing Services
Treat backing services as attached resources

V. Build, release, run
Strictly separate build and run stages

VI. Processes
Execute the app as one or more stateless processes

VII. Port binding
Export services via port binding

VIII. Concurrency
Scale out via the process model

IX. Disposability
Maximize robustness with fast startup and graceful shutdown

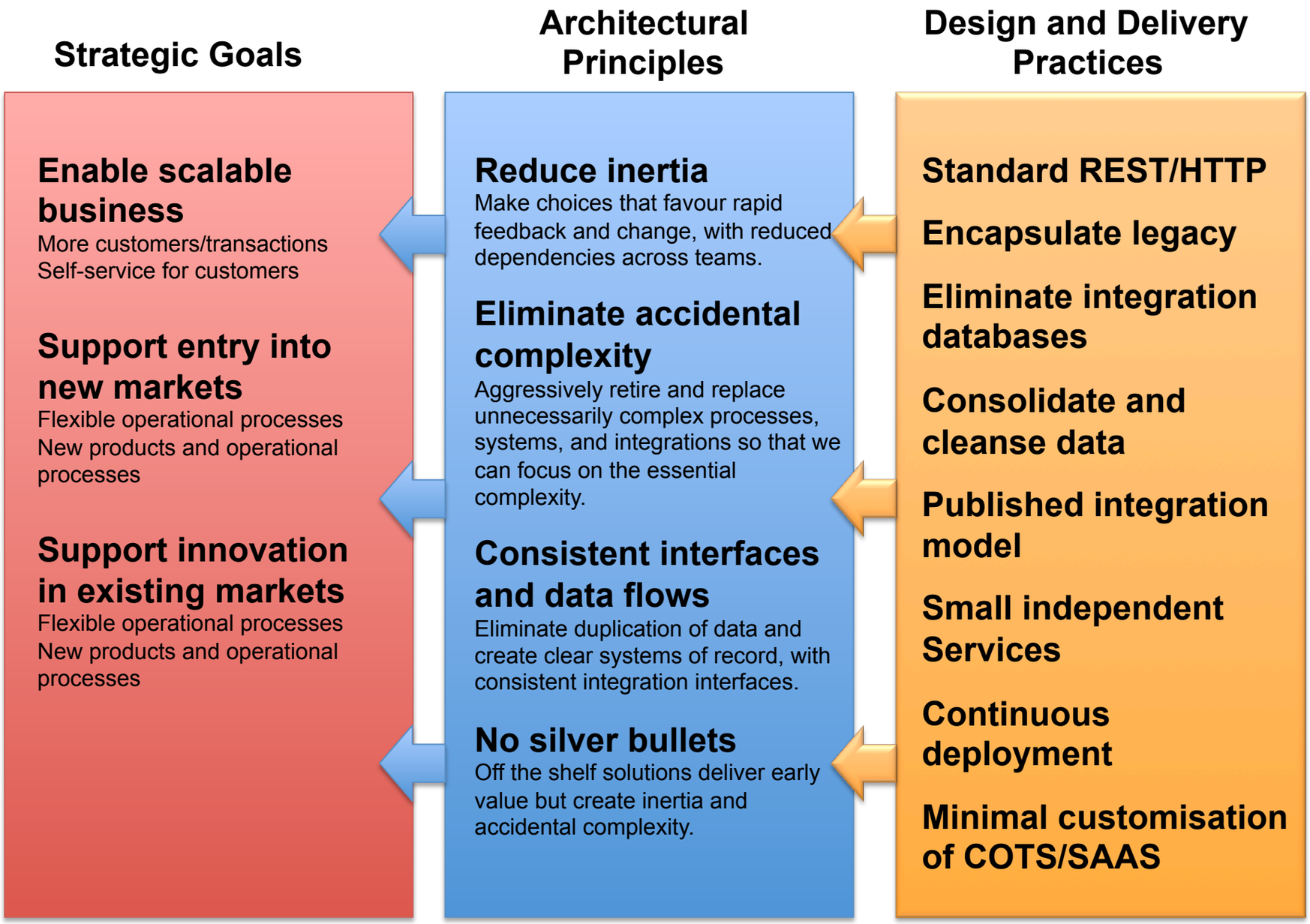
X. Dev/prod parity
Keep development, staging, and production as similar as possible

XI. Logs
Treat logs as event streams

XII. Admin processes
Run admin/management tasks as one-off processes

http://www.12factor.net/

@samnewman


Strategic Goals
Enable scalable business
More customers/transactions
Self-service for customers
Support entry into new markets
Flexible operational processes
New products and operational processes
Support innovation in existing markets
Flexible operational processes
New products and operational processes
Architectural Principles
Reduce inertia
Make choices that favour rapid feedback and change, with reduced dependencies across teams.
Eliminate accidental complexity
Aggressively retire and replace unnecessarily complex processes, systems, and integrations so that we can focus on the essential complexity.
Consistent interfaces and data flows
Eliminate duplication of data and create clear systems of record, with consistent integration interfaces.
No silver bullets
Off the shelf solutions deliver early value but create inertia and accidental complexity.
Design and Delivery Practices
Standard REST/HTTP
Encapsulate legacy
Eliminate integration databases
Consolidate and cleanse data
Published integration model
Small independent Services
Continuous deployment
Minimal customisation of COTS/SAAS

@samnewman

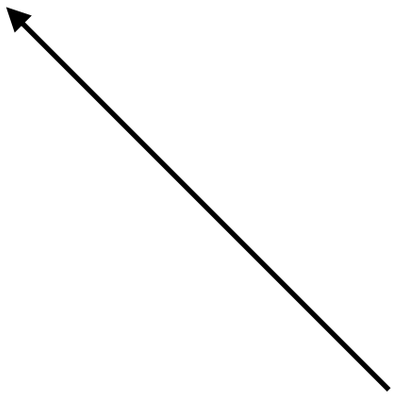
Small ***Autonomous*** services

that ***work together***

@samnewman

Principles Of Microservices

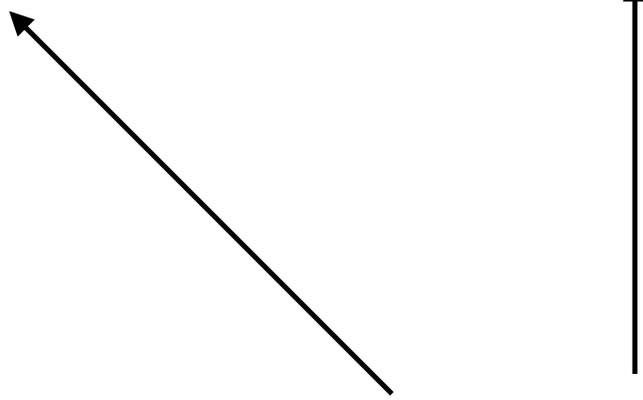
Modelled Around
Business Domain



Principles Of Microservices

Modelled Around Business Domain

Culture Of Automation

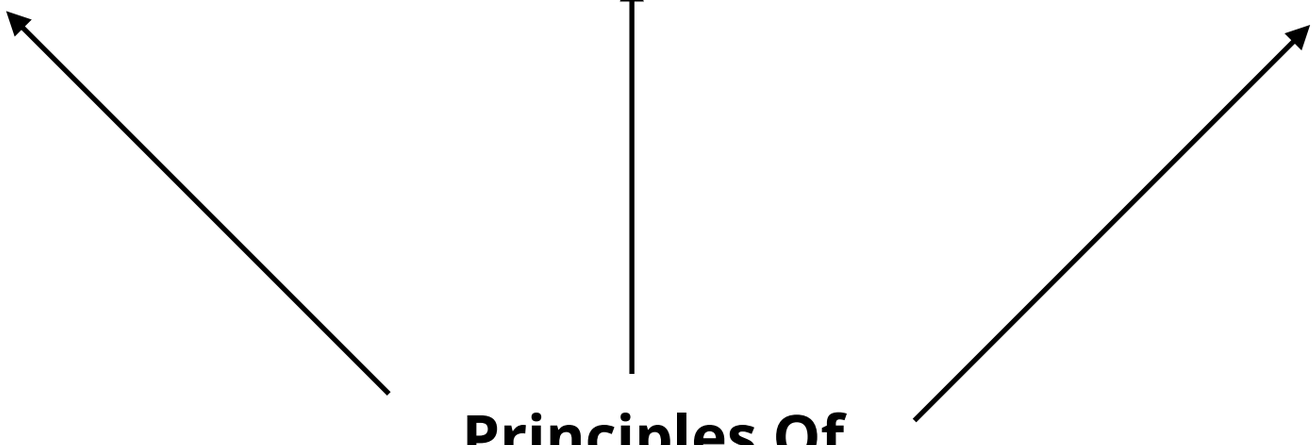


Principles Of Microservices

Modelled Around Business Domain

Culture Of Automation

Hide Implementation Details



Principles Of Microservices

Modelled Around Business Domain

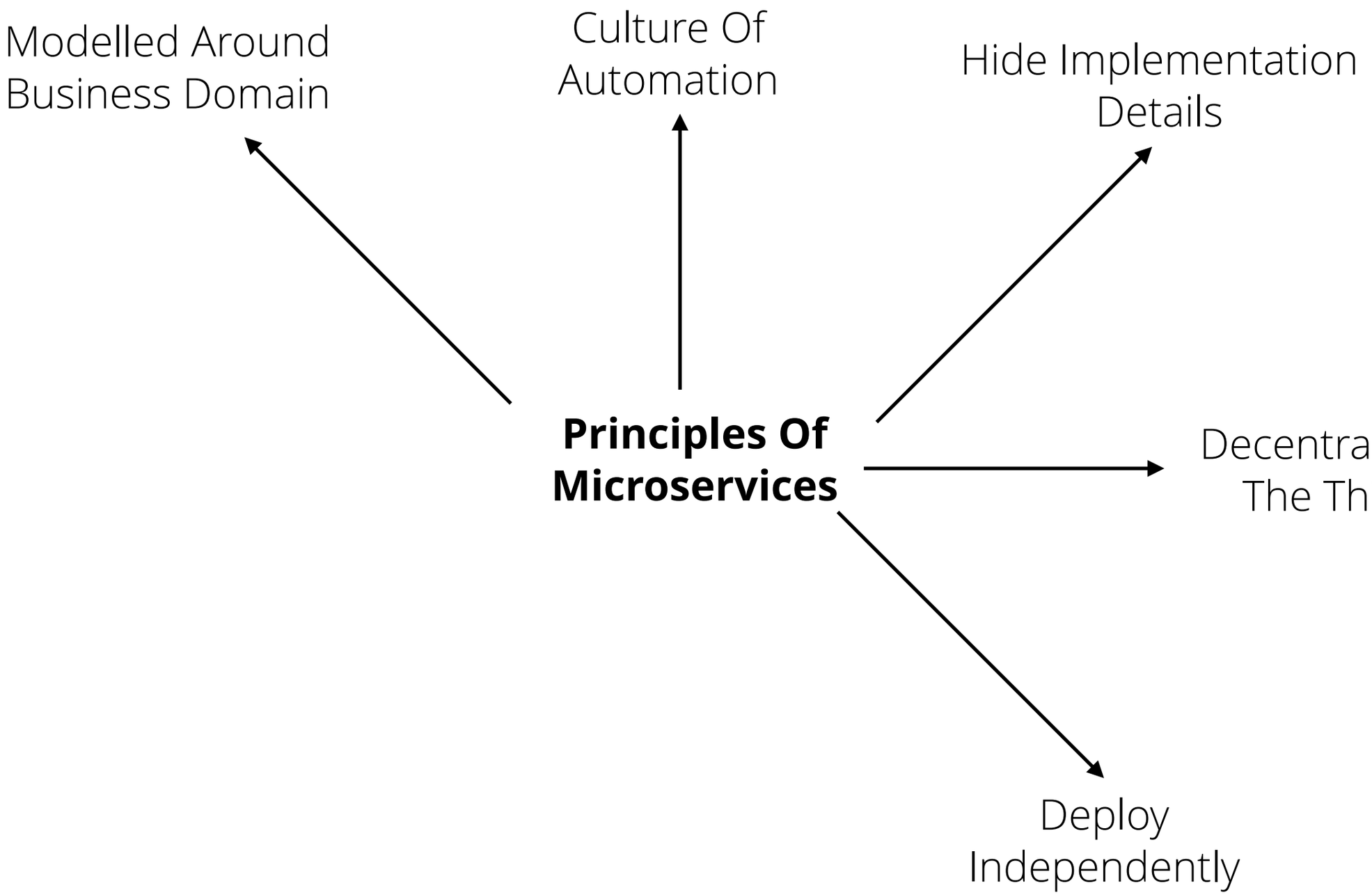
Culture Of Automation

Hide Implementation Details

Principles Of Microservices

Decentralise All The Things

@samnewman


Modelled Around Business Domain
Culture Of Automation
Hide Implementation Details
Principles Of Microservices
Decentralise All The Things
Deploy Independently

@samnewman

Modelled Around Business Domain

Culture Of Automation

Hide Implementation Details

Principles Of Microservices

Decentralise All The Things

Isolate Failure

Deploy Independently

@samnewman

Modelled Around Business Domain

Culture Of Automation

Hide Implementation Details

Highly Observable

Principles Of Microservices

Decentralise All The Things

Isolate Failure

Deploy Independently

@samnewman

Modelled Around Business Domain

Culture Of Automation

Hide Implementation Details

Highly Observable

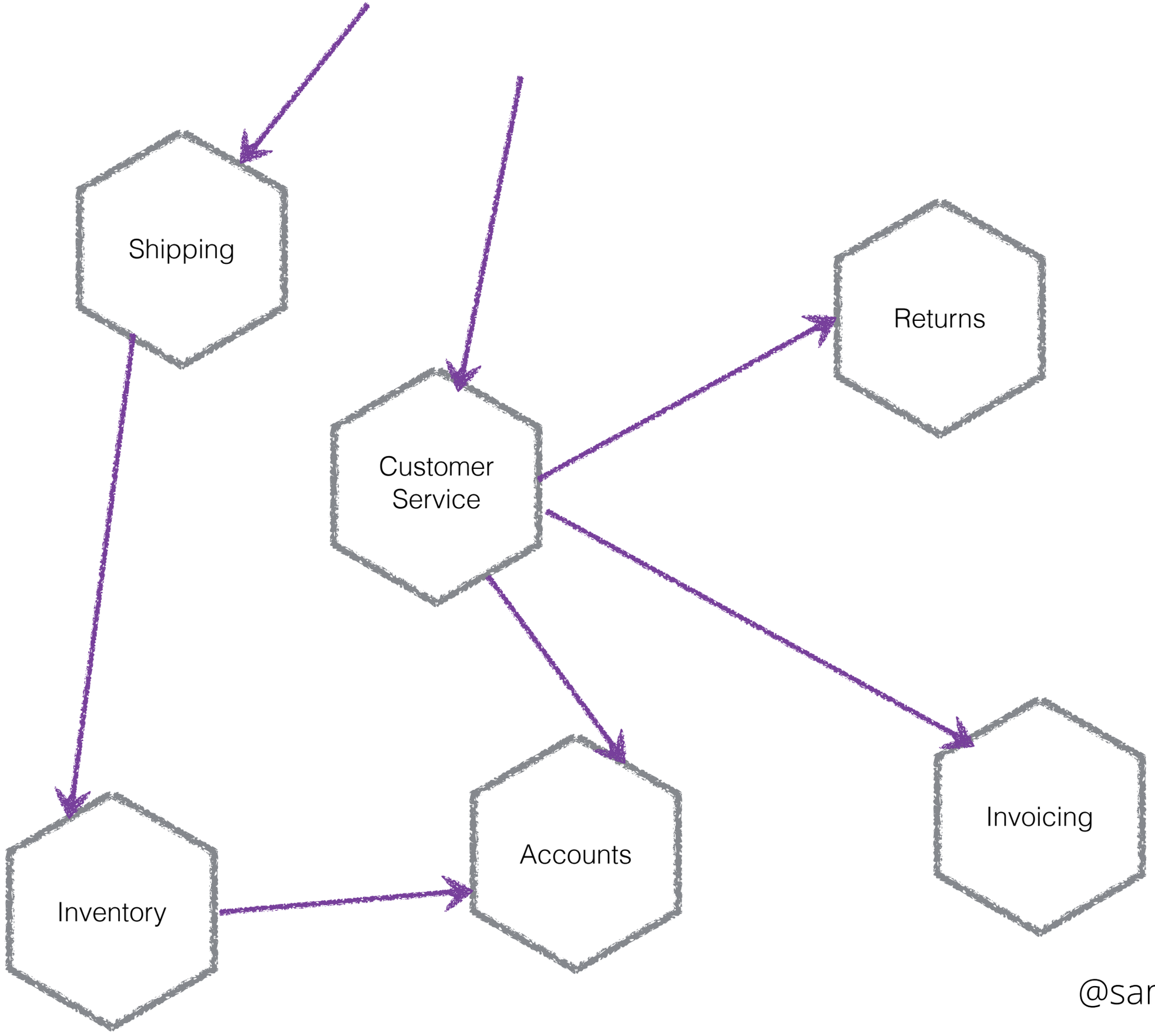
Principles Of Microservices

Decentralise All The Things

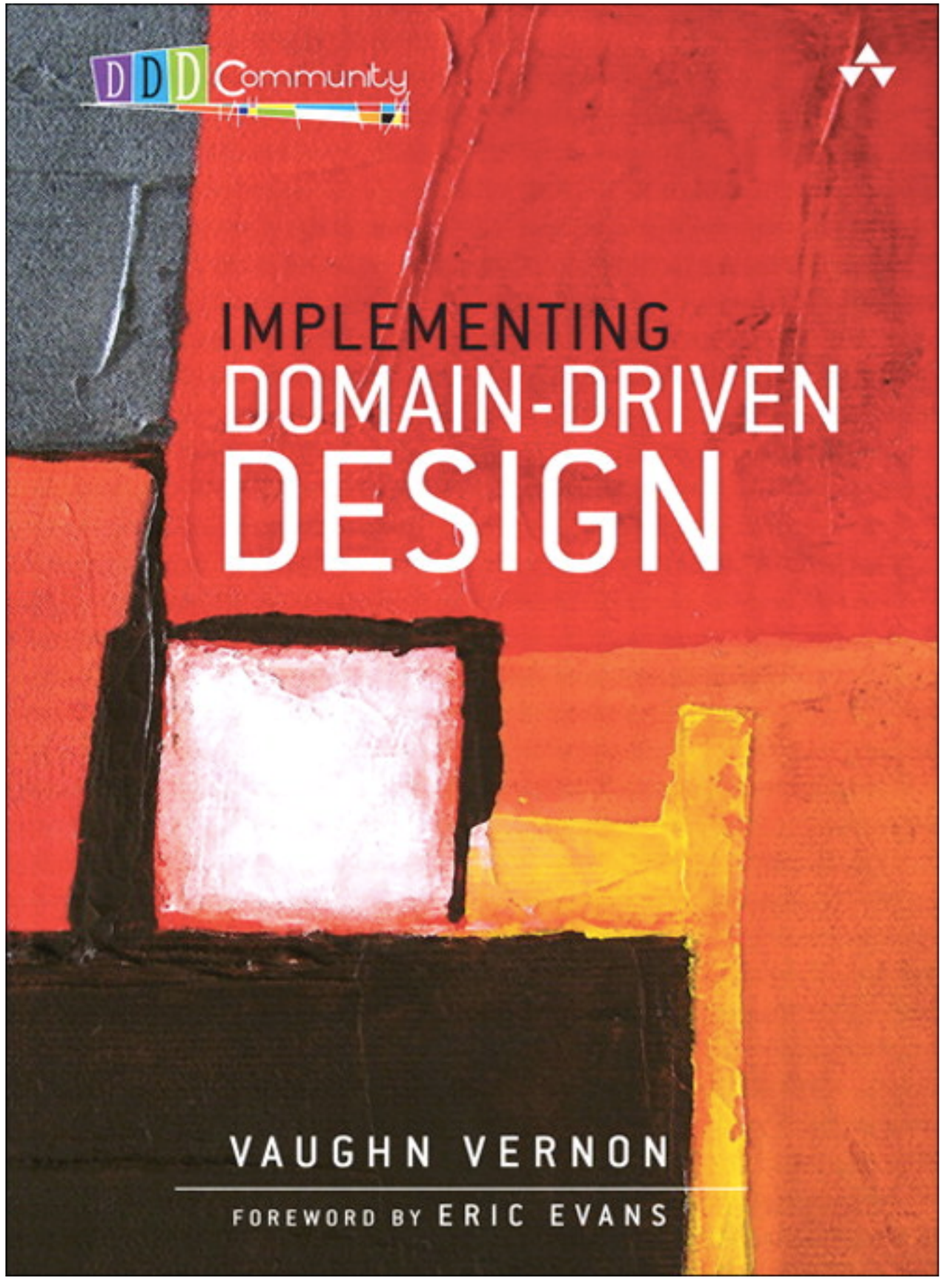
Isolate Failure

Deploy Independently

@samnewman


Shipping
Customer Service
Returns
Inventory
Accounts
Invoicing

@samnewman


DDD Community
IMPLEMENTING
DOMAIN-DRIVEN
DESIGN
VAUGHN VERNON
FOREWORD BY ERIC EVANS

@samnewman

✓ Modelled Around Business Domain

Culture Of Automation

Hide Implementation Details

Highly Observable

Principles Of Microservices

Decentralise All The Things

Isolate Failure

Deploy Independently

@samnewman

Culture Of Automation

✓ Modelled Around Business Domain

Hide Implementation Details

Highly Observable

Principles Of Microservices

Decentralise All The Things

Isolate Failure

Deploy Independently

@samnewman

@samnewman

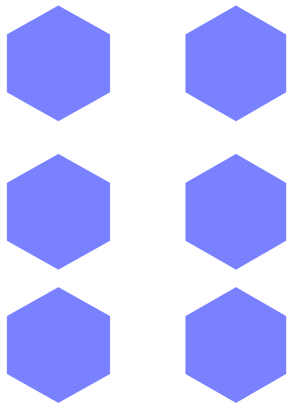
2 Microservices

3 Months

10 Microservices

2 Microservices

3 Months

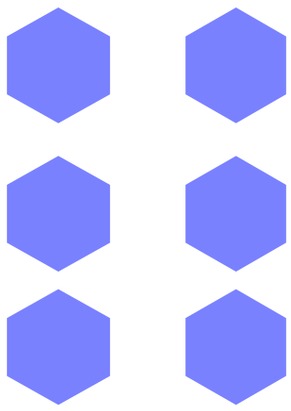


12 Months

2 Microservices

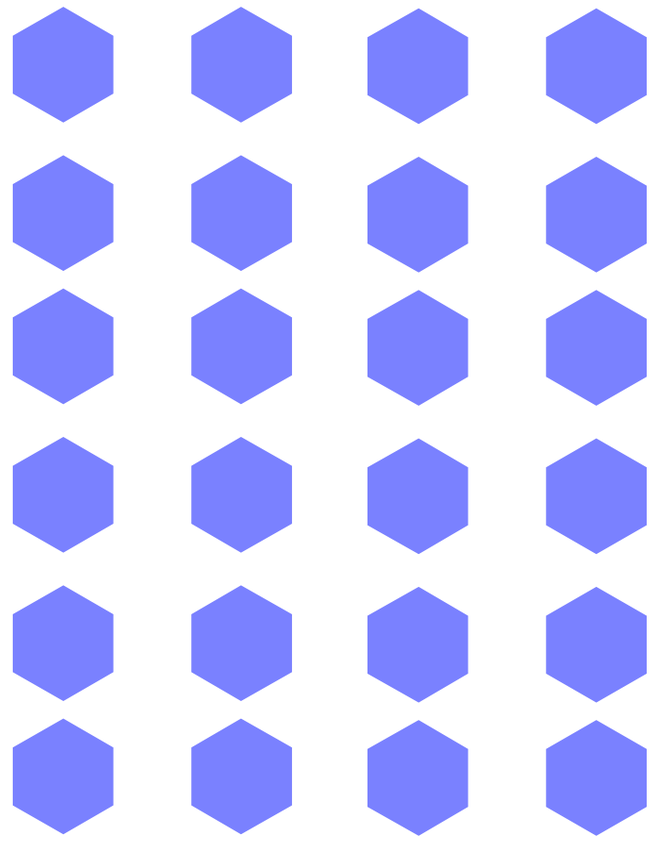
3 Months

10 Microservices



12 Months

60 Microservices



18 Months

@samnewman

Infrastructure Automation

@samnewman

Infrastructure Automation

Automated Testing

@samnewman

Infrastructure Automation

Automated Testing

Continuous Delivery

@samnewman

✓ Modelled Around Business Domain

✓ Culture Of Automation

Hide Implementation Details

Highly Observable

Principles Of Microservices

Decentralise All The Things

Isolate Failure

Deploy Independently

@samnewman

✓ Modelled Around Business Domain

✓ Culture Of Automation

Hide Implementation Details

Highly Observable

Principles Of Microservices

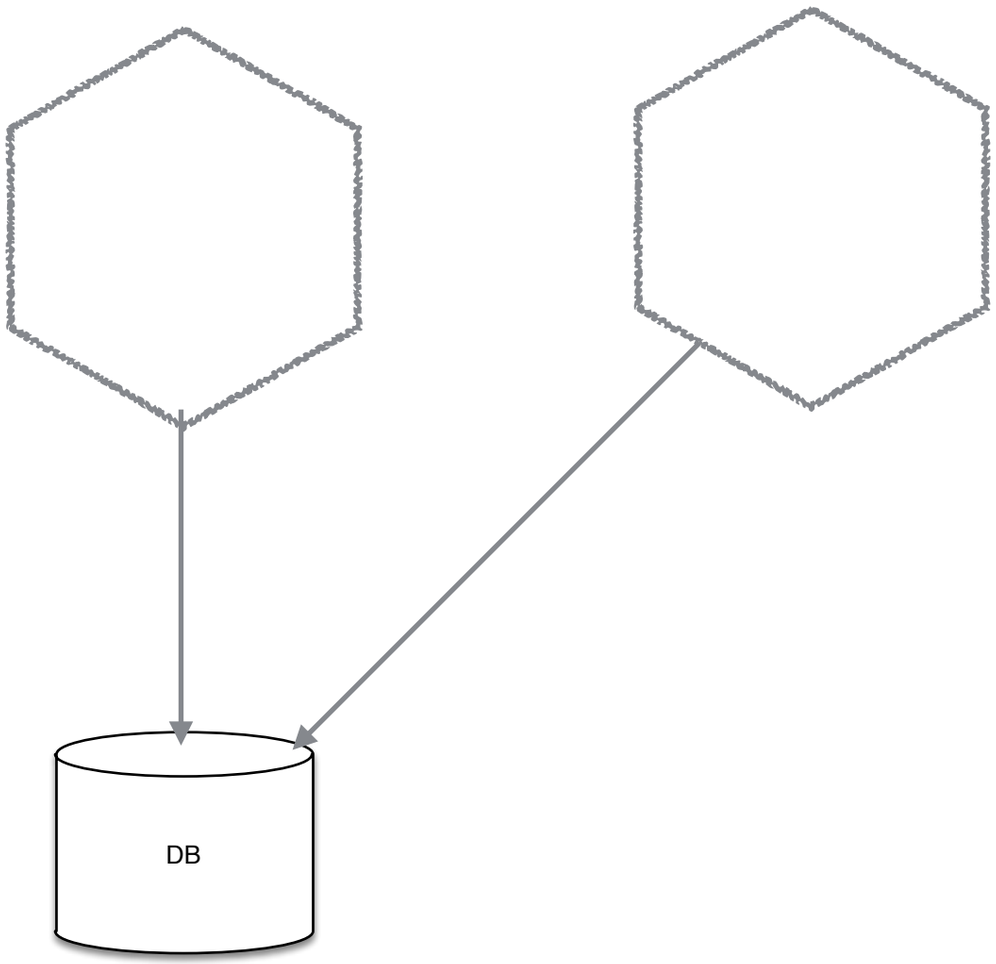
Decentralise All The Things

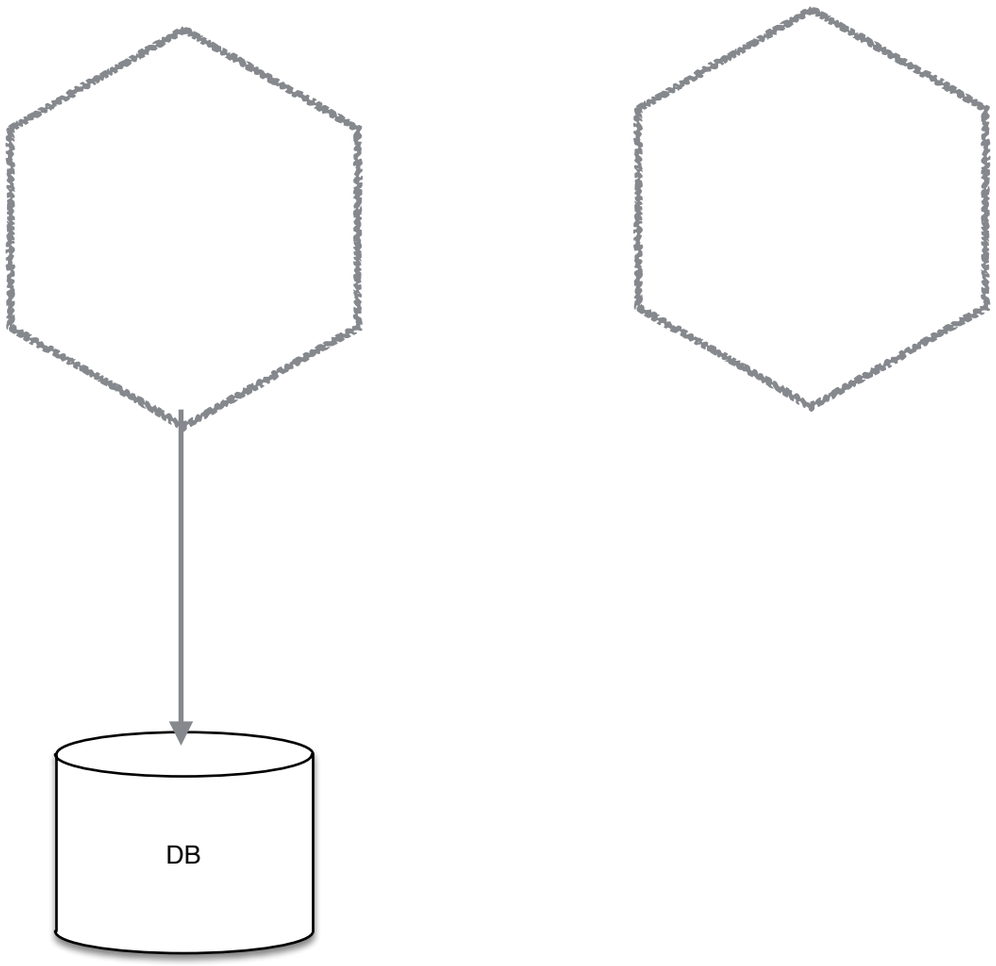
Isolate Failure

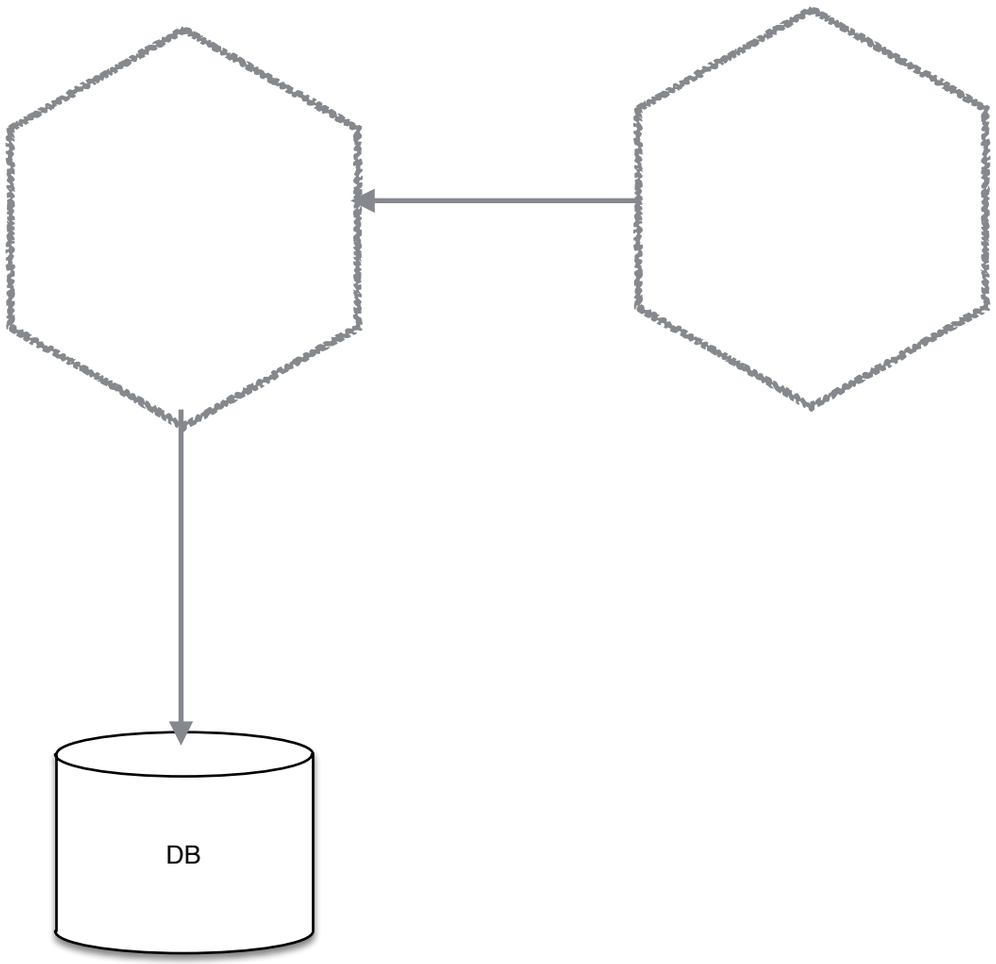
Deploy Independently

@samnewman

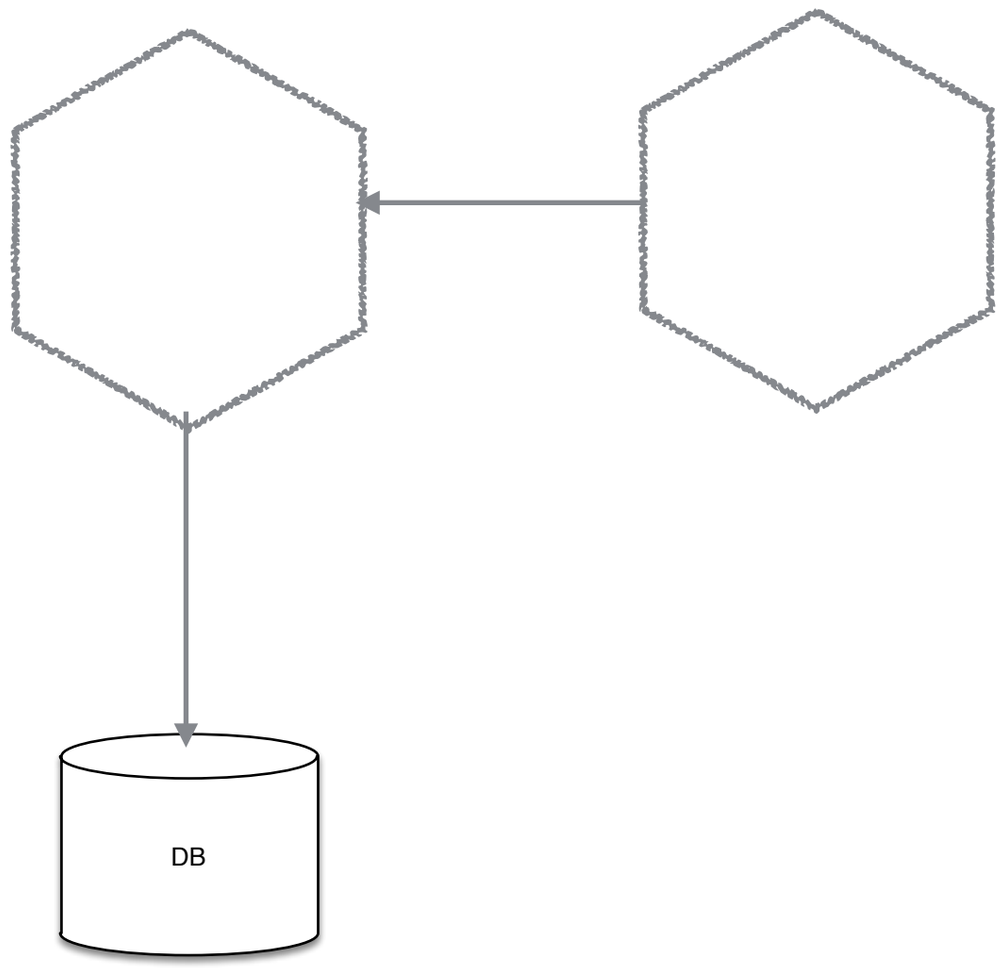

DB

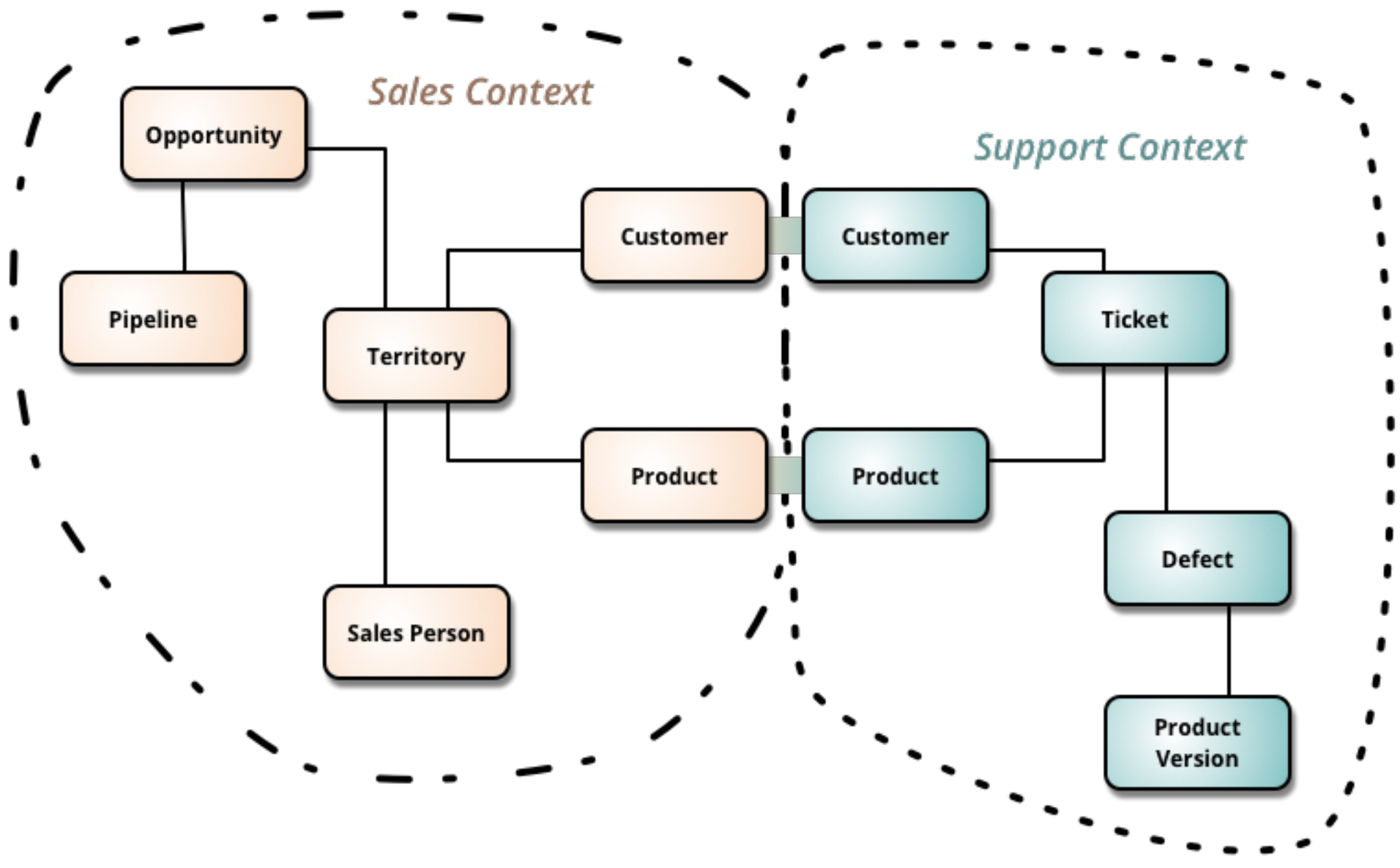

DB


DB


DB

HIDE YOUR DATABASE





@samnewman

http://martinfowler.com/bliki/images/boundedContext/sketch.png

✓ Modelled Around Business Domain

✓ Culture Of Automation

✓ Hide Implementation Details

Principles Of Microservices

Highly Observable

Isolate Failure

Decentralise All The Things

Deploy Independently

@samnewman

✓ Modelled Around Business Domain

✓ Culture Of Automation

✓ Hide Implementation Details

Highly Observable

Principles Of Microservices

Decentralise All The Things

Isolate Failure

Deploy Independently

@samnewman

@samnewman

What is autonomy?

@samnewman

What is autonomy?

Giving people as much freedom as possible to do the job at hand

@samnewman

What is autonomy?

Giving people as much freedom as possible to do the job at hand

@samnewman


DEVOLUTION?

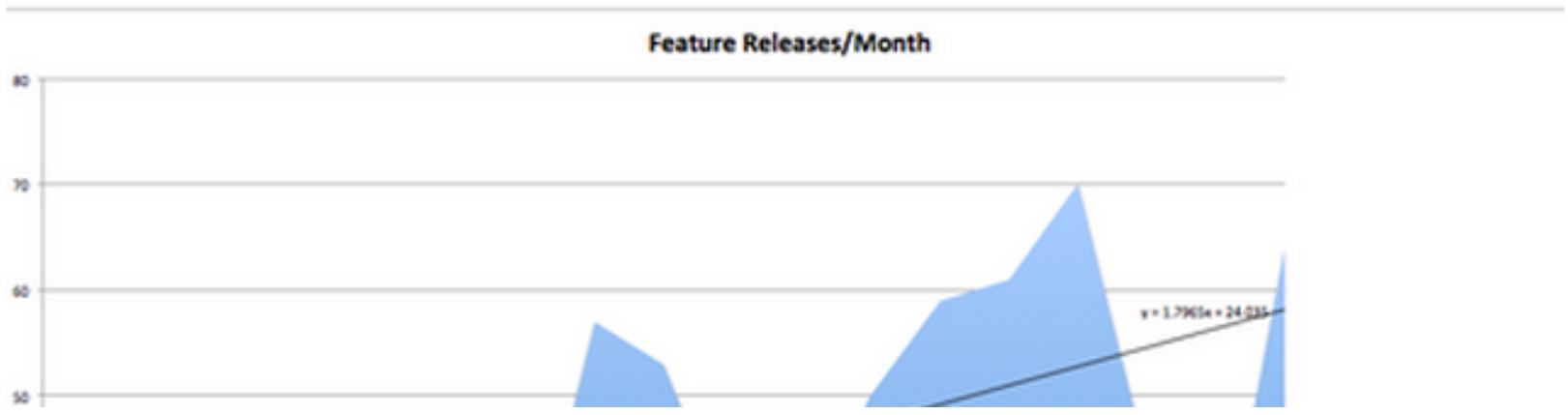
SELF-SERVICE

https://www.flickr.com/photos/katsrcool/15184711908/

GILT TECH

Search the blog

Making Architecture Work in Microservice Organizations



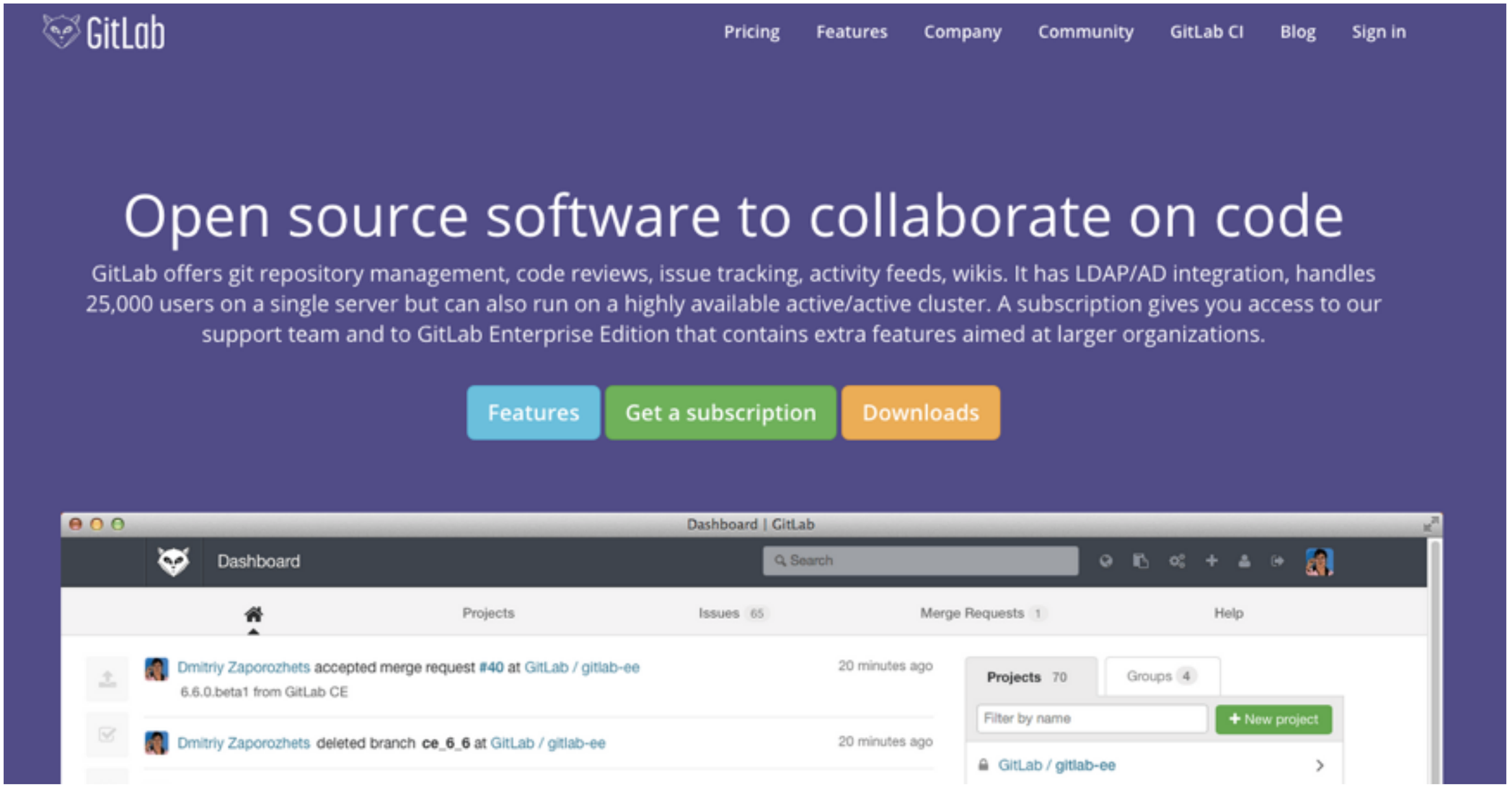
http://tech.gilt.com/post/102628539834/making-architecture-work-in-microservice

@samnewman

OWNER-OPERATOR

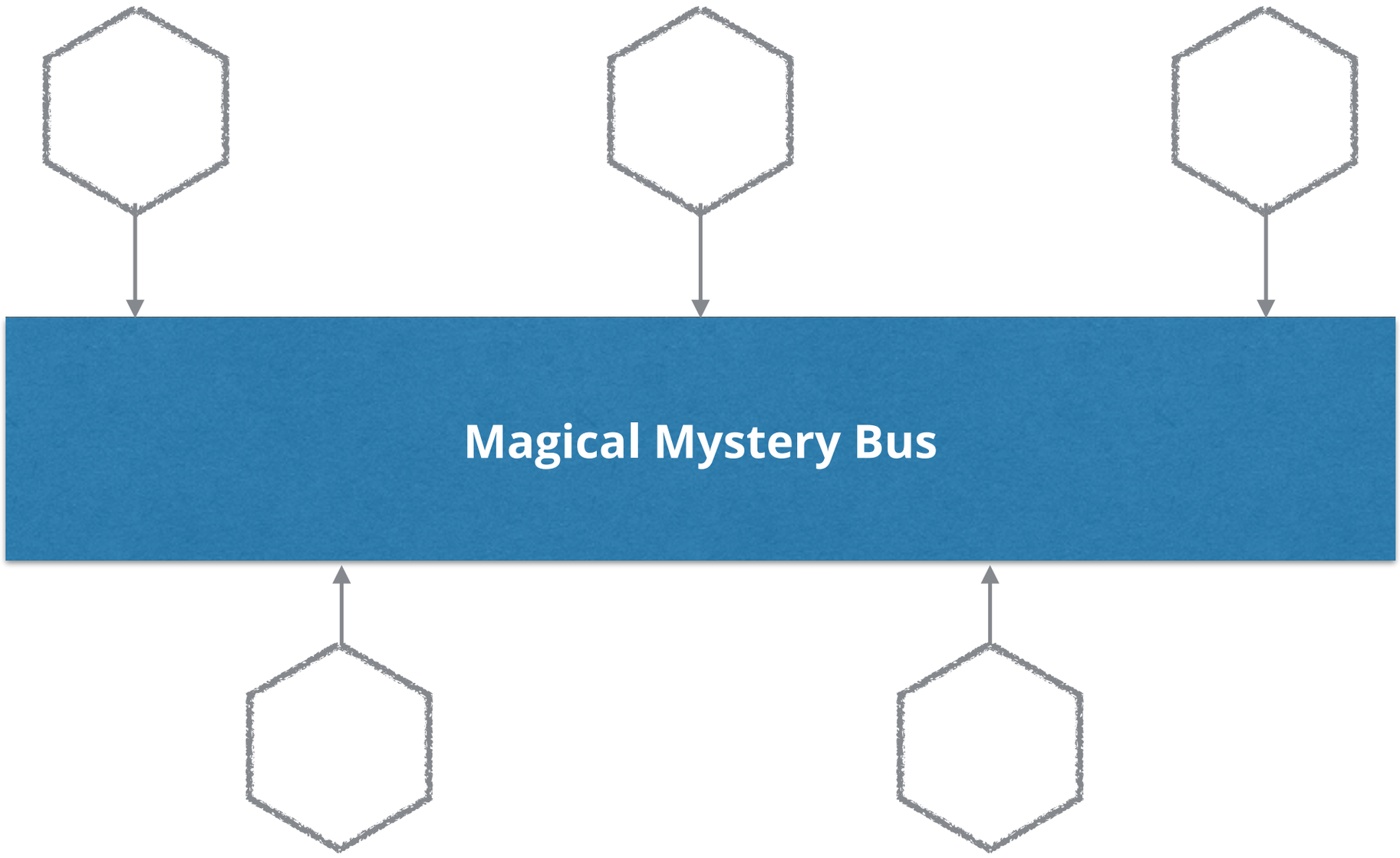
https://www.flickr.com/photos/stevendepolo/5939055612

INTERNAL OPEN SOURCE

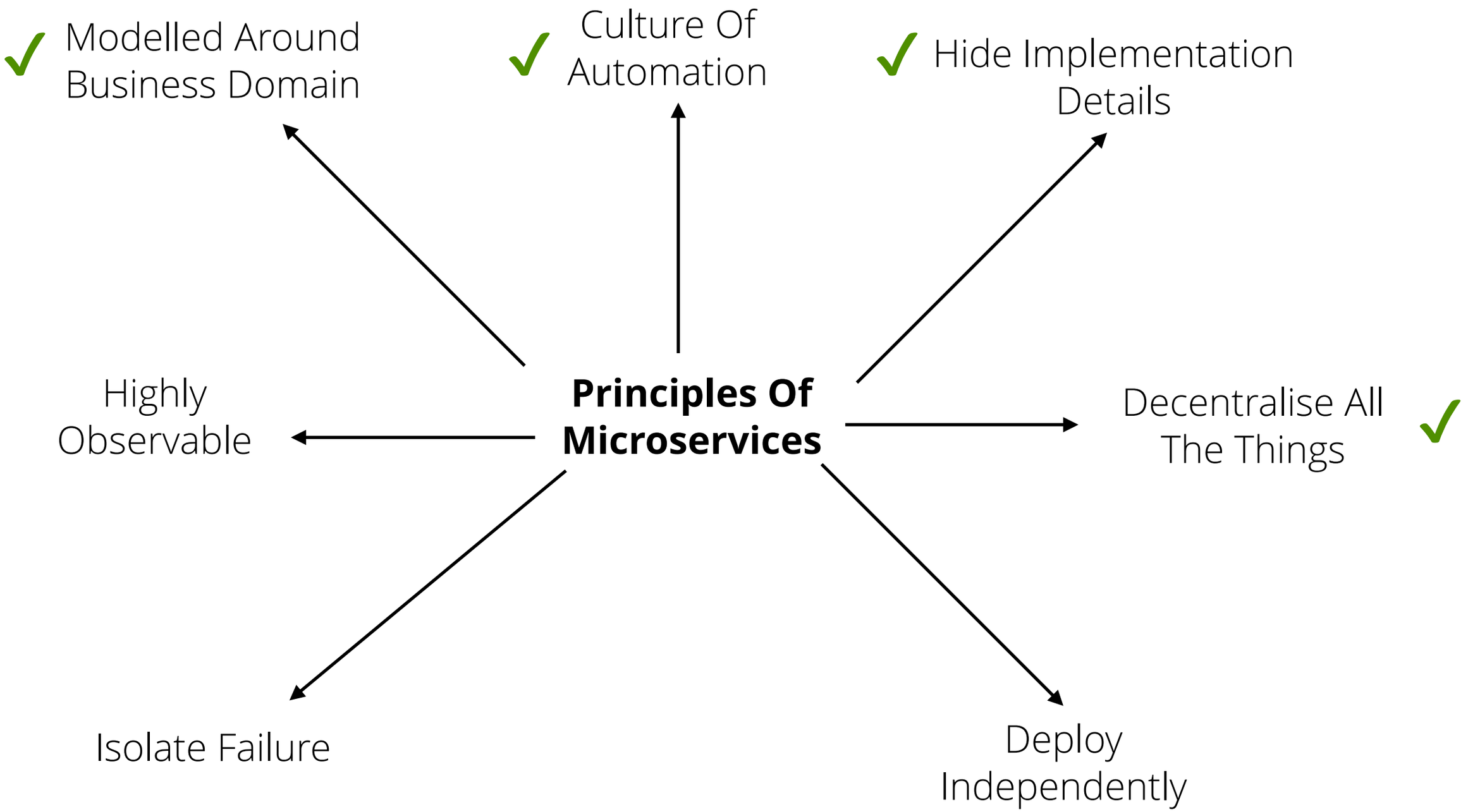


@samnewman

DUMB-PIPES, SMART ENDPOINTS



Magical Mystery Bus


✓ Modelled Around Business Domain
✓ Culture Of Automation
✓ Hide Implementation Details
Highly Observable
Principles Of Microservices
Decentralise All The Things ✓
Isolate Failure
Deploy Independently

@samnewman

✓ Modelled Around Business Domain

✓ Culture Of Automation

✓ Hide Implementation Details

Highly Observable

Principles Of Microservices

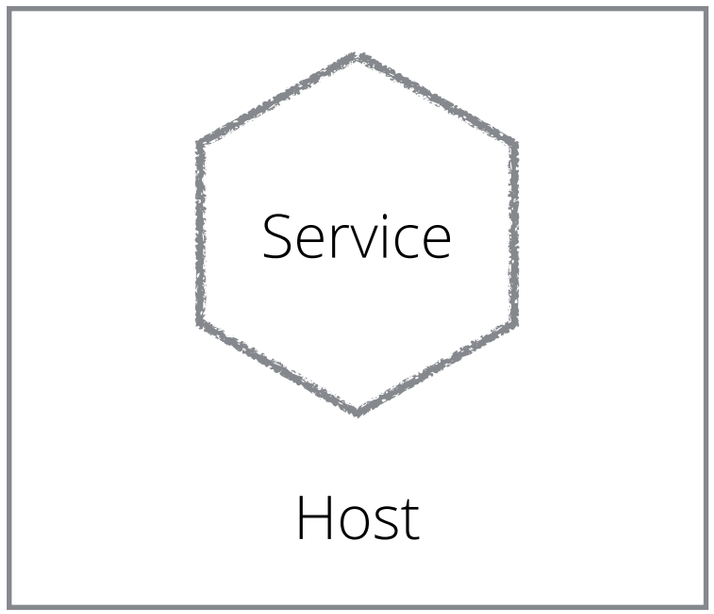
Decentralise All The Things ✓

Isolate Failure

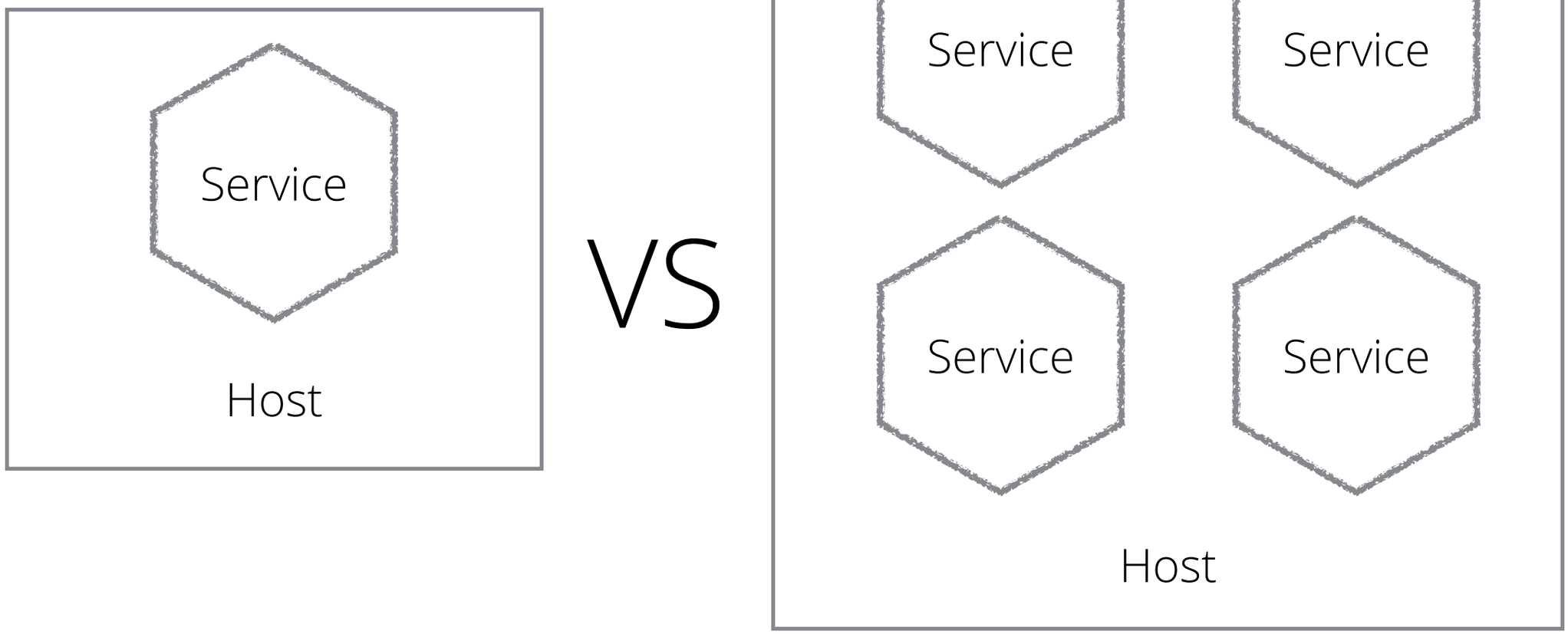
Deploy Independently

@samnewman

ONE SERVICE PER-HOST



ONE SERVICE PER-HOST



@samnewman

CONSUMER-DRIVEN CONTRACTS

@samnewman

CONSUMER-DRIVEN CONTRACTS

Expectations

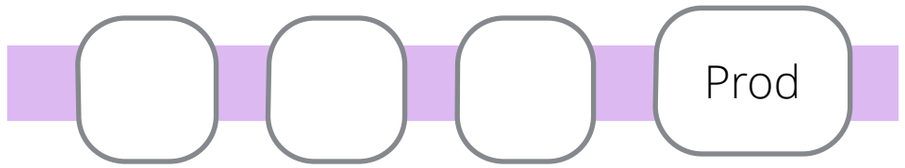
CONSUMER-DRIVEN CONTRACTS

Expectations

@samnewman

CONSUMER-DRIVEN CONTRACTS

Expectations




```
def asHtml(): Node = {
  <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
    <head>
        <link rel="stylesheet" href="/static/common.css" type="text/css" />
        <link rel="stylesheet" href={ cssForDisplayType } type="text/css" />
        <meta http-equiv="refresh" content="30" />
    </head>
    <body>
      { content(builds) }
    </body>
  </html>
}

private def content(builds: List[Build]): Elem = {
  displayType match {
    case "single" => <div> { builds.map(build => asTable(build)) } </div>
    case "smart" => {
      if (builds.length == 1) {
        <div> { builds.map(build => asTable(build)) } </div>
      } else {
        <ul class="builds">{ builds.map(build => buildToHtml(build)) }</ul>
      }
    }
    case _ => <ul class="builds">{ builds.map(build => buildToHtml(build)) }</ul>
  }
}

private def asTable(build: Build): Elem = {
  <table class={ "build " + build.getStatus.name.toLowerCase }>
          <tr valign="middle" align="center">
            <td>{ linkToBuild(build) }</td>
          </tr>
        </table>
}
```

@samnewman

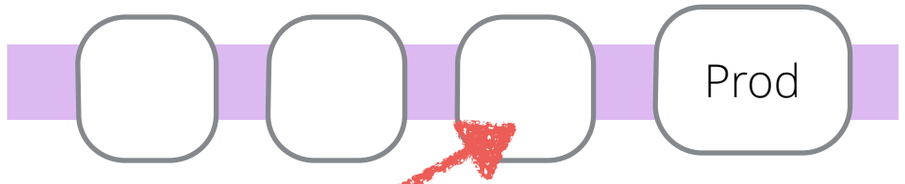
CONSUMER-DRIVEN CONTRACTS

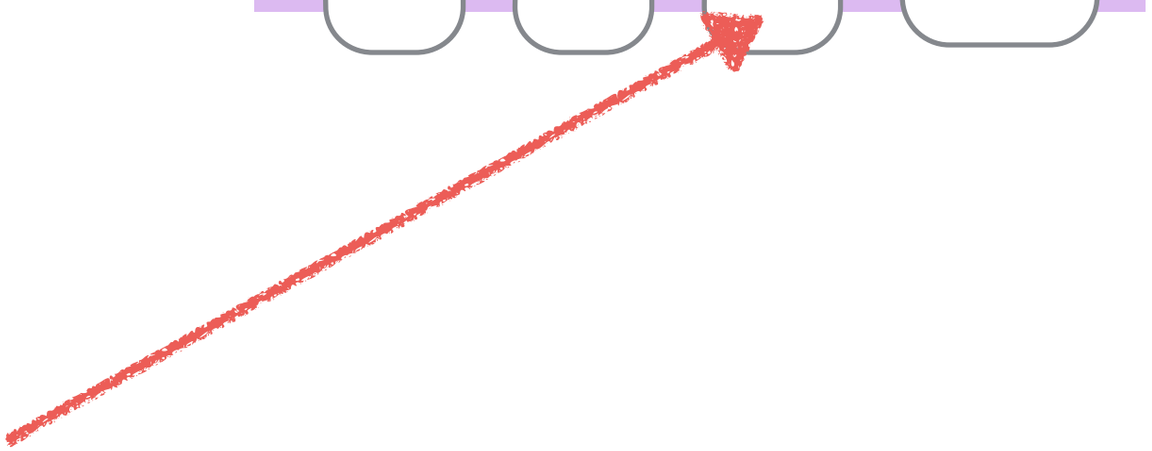
Expectations

```
def asHtml(): Node = {
  <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
    <head>
        <link rel="stylesheet" href="/static/common.css" type="text/css" />
        <link rel="stylesheet" href={ cssForDisplayType } type="text/css" />
        <meta http-equiv="refresh" content="30" />
    </head>
    <body>
      { content(builds) }
    </body>
  </html>
}

private def content(builds: List[Build]): Elem = {
  displayType match {
    case "single" => <div> { builds.map(build => asTable(build)) } </div>
    case "smart" => {
      if (builds.length == 1) {
        <div> { builds.map(build => asTable(build)) } </div>
      } else {
        <ul class="builds">{ builds.map(build => buildToHtml(build)) }</ul>
      }
    }
    case _ => <ul class="builds">{ builds.map(build => buildToHtml(build)) }</ul>
  }
}

private def asTable(build: Build): Elem = {
  <table class={ "build " + build.getStatus.name.toLowerCase }>
            <tr valign="middle" align="center">
              <td>{ linkToBuild(build) }</td>
            </tr>
          </table>
}
```





@samnewman

README.md

Pact

Define a pact between service consumers and providers, enabling "consumer driven contract" testing.

Pact provides an RSpec DSL for service consumers to define the HTTP requests they will make to a service provider and the HTTP responses they expect back. These expectations are used in the consumers specs to provide a mock service provider. The interactions are recorded, and played back in the service provider specs to ensure the service provider actually does provide the response the consumer expects.

This allows testing of both sides of an integration point using fast unit tests.

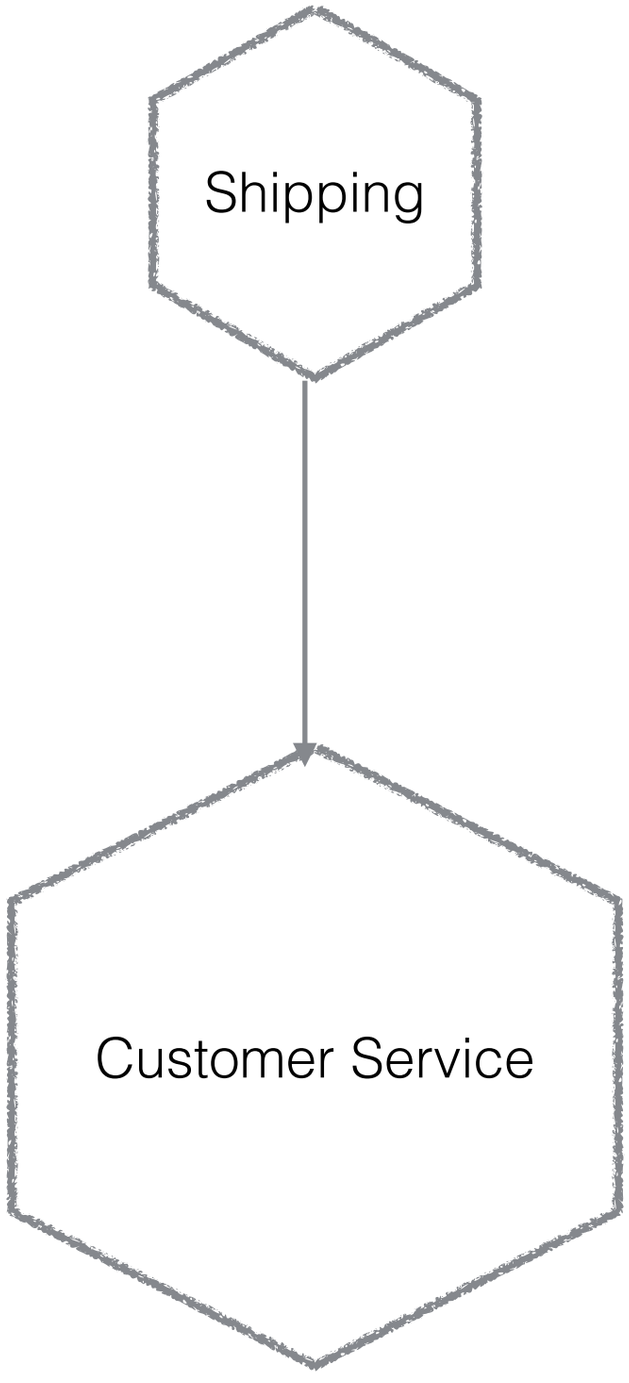
This gem is inspired by the concept of "Consumer driven contracts". See http://martinfowler.com/articles/consumerDrivenContracts.html for more information.

Travis CI Status: build passing

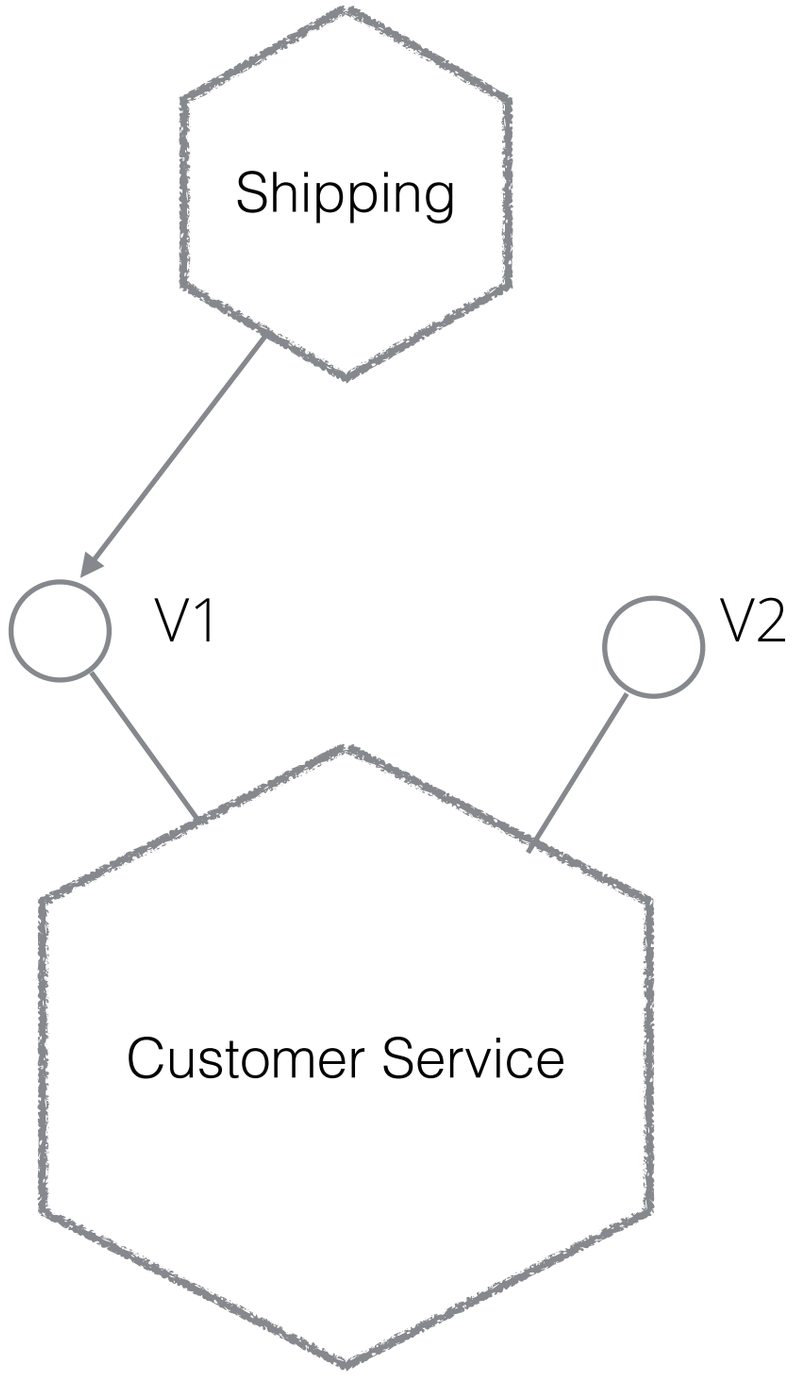
https://github.com/realestate-com-au/pact

@samnewman

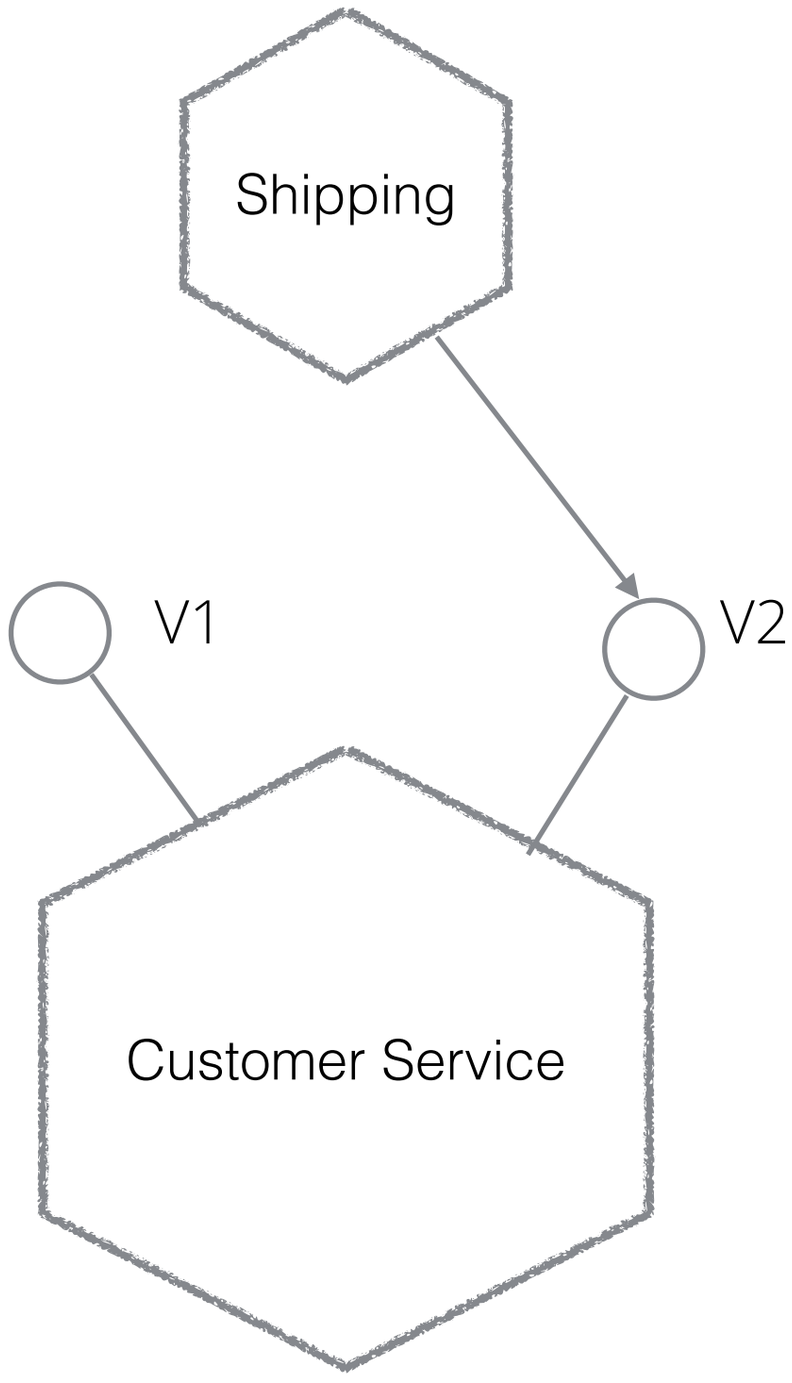
CO-EXIST ENDPOINTS



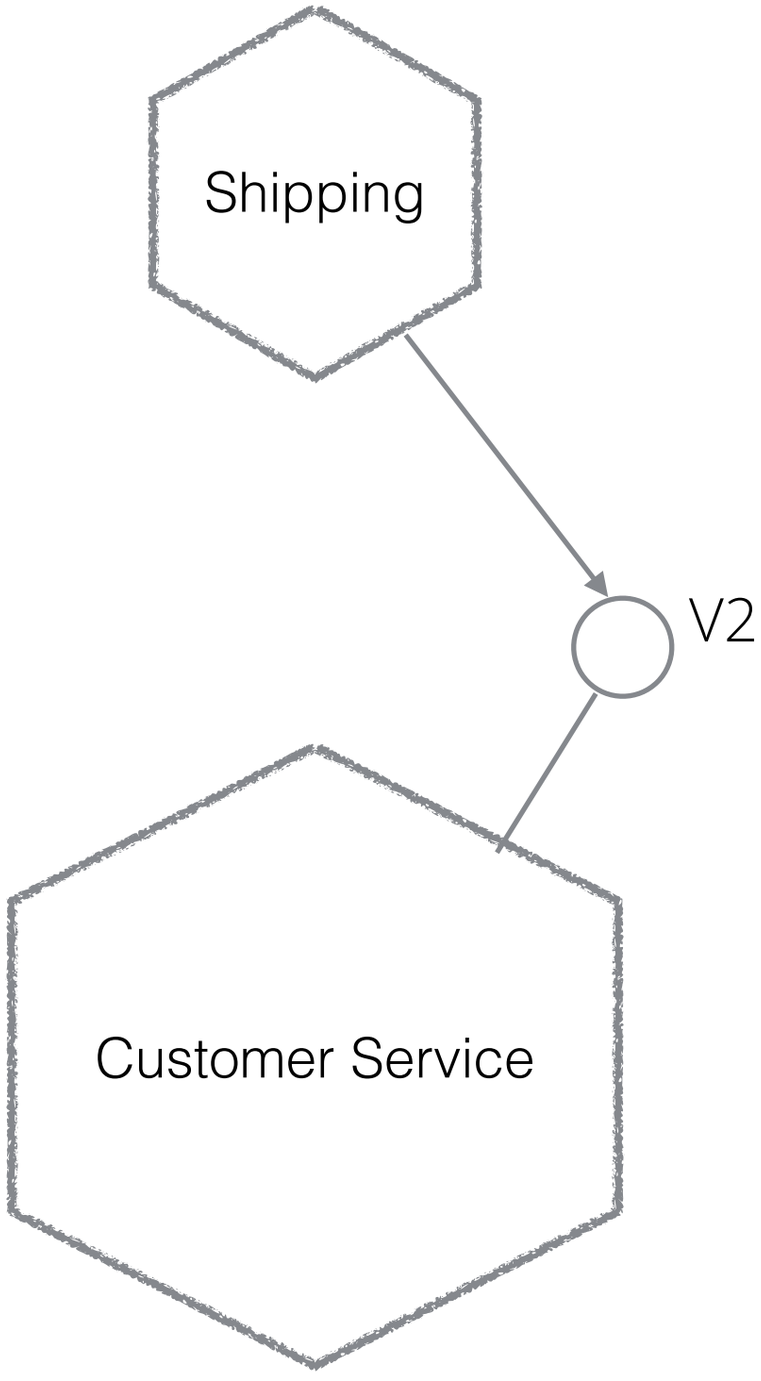
CO-EXIST ENDPOINTS

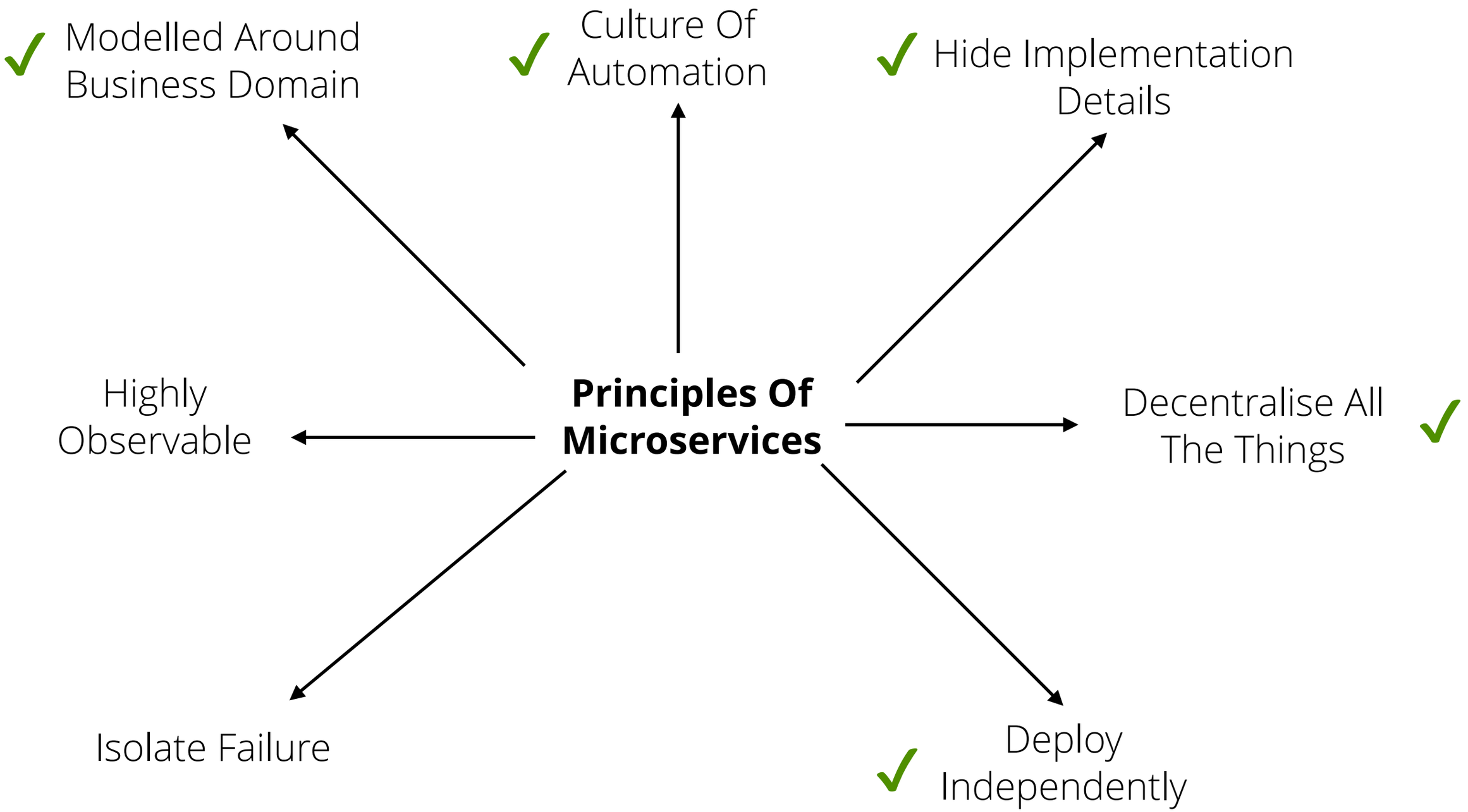


CO-EXIST ENDPOINTS



CO-EXIST ENDPOINTS




✓ Modelled Around Business Domain
✓ Culture Of Automation
✓ Hide Implementation Details
Highly Observable
Principles Of Microservices
Decentralise All The Things ✓
Isolate Failure
✓ Deploy Independently

@samnewman

✓ Modelled Around Business Domain

✓ Culture Of Automation

✓ Hide Implementation Details

Highly Observable

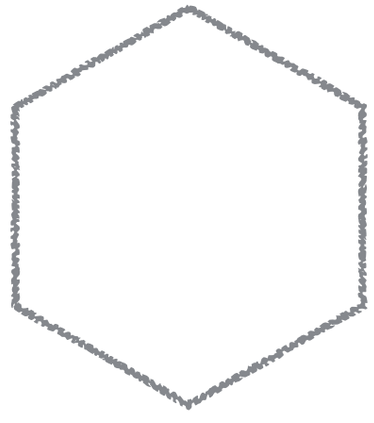
Principles Of Microservices

Decentralise All The Things ✓

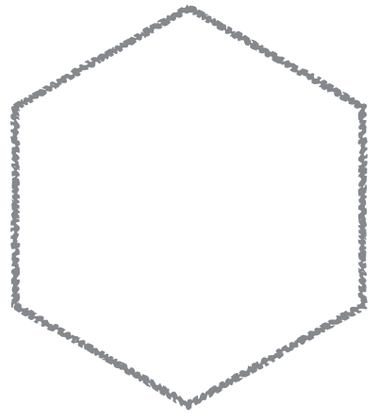
Isolate Failure

✓ Deploy Independently

@samnewman

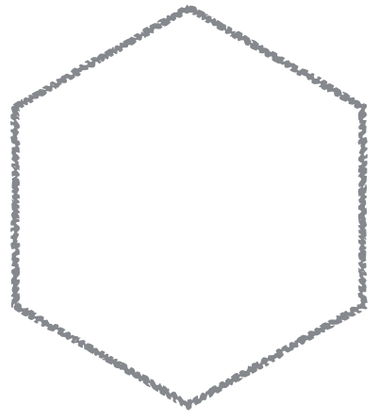


@samnewman

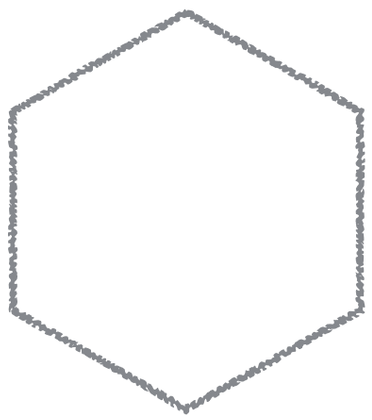


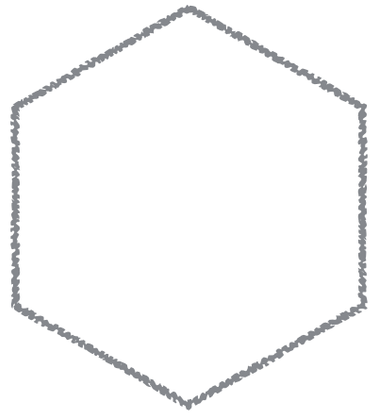
1 in 100

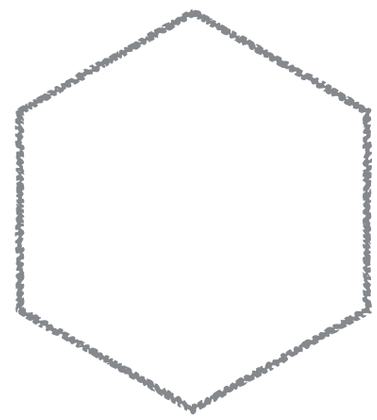
@samnewman



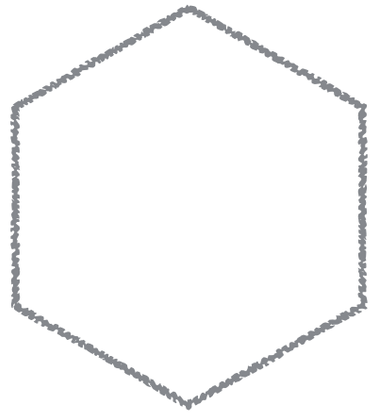
1 in 100



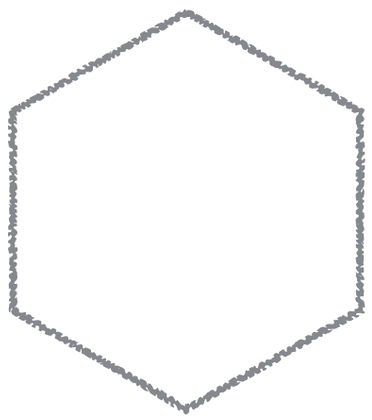


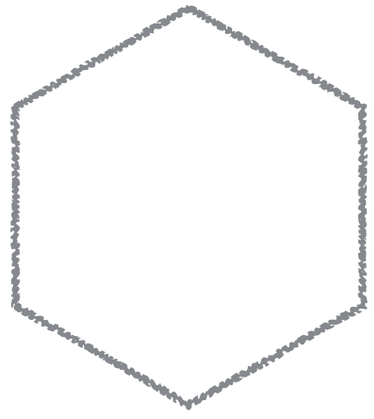


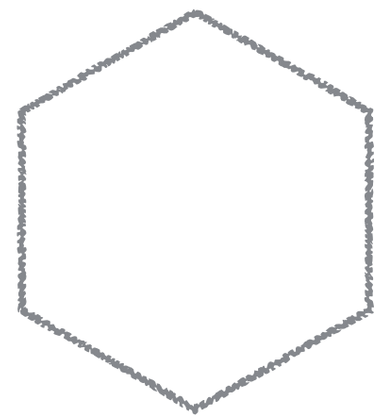
@samnewman



4 in 100







@samnewman

Strangler App

@samnewman

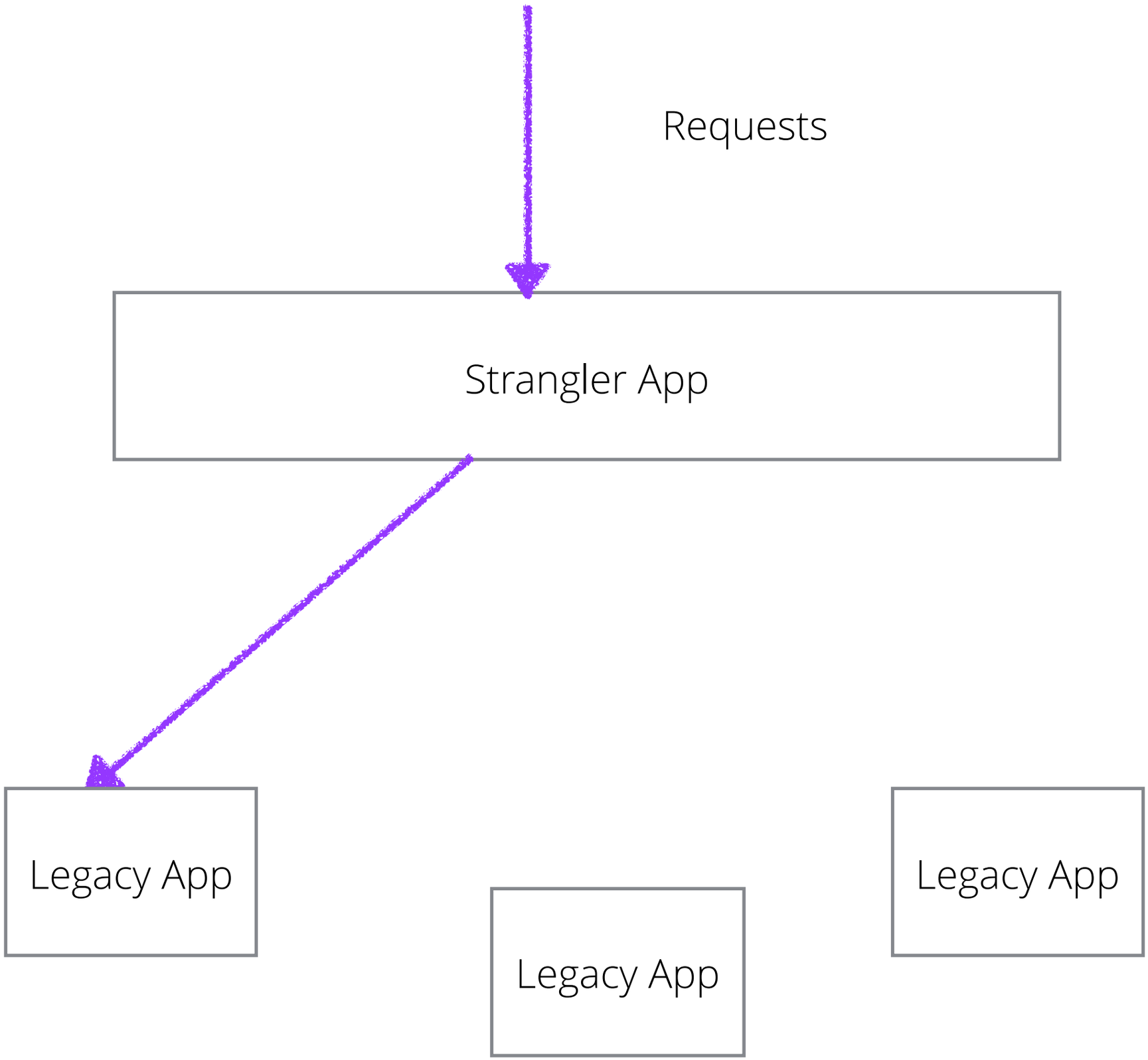
Strangler App

Legacy App

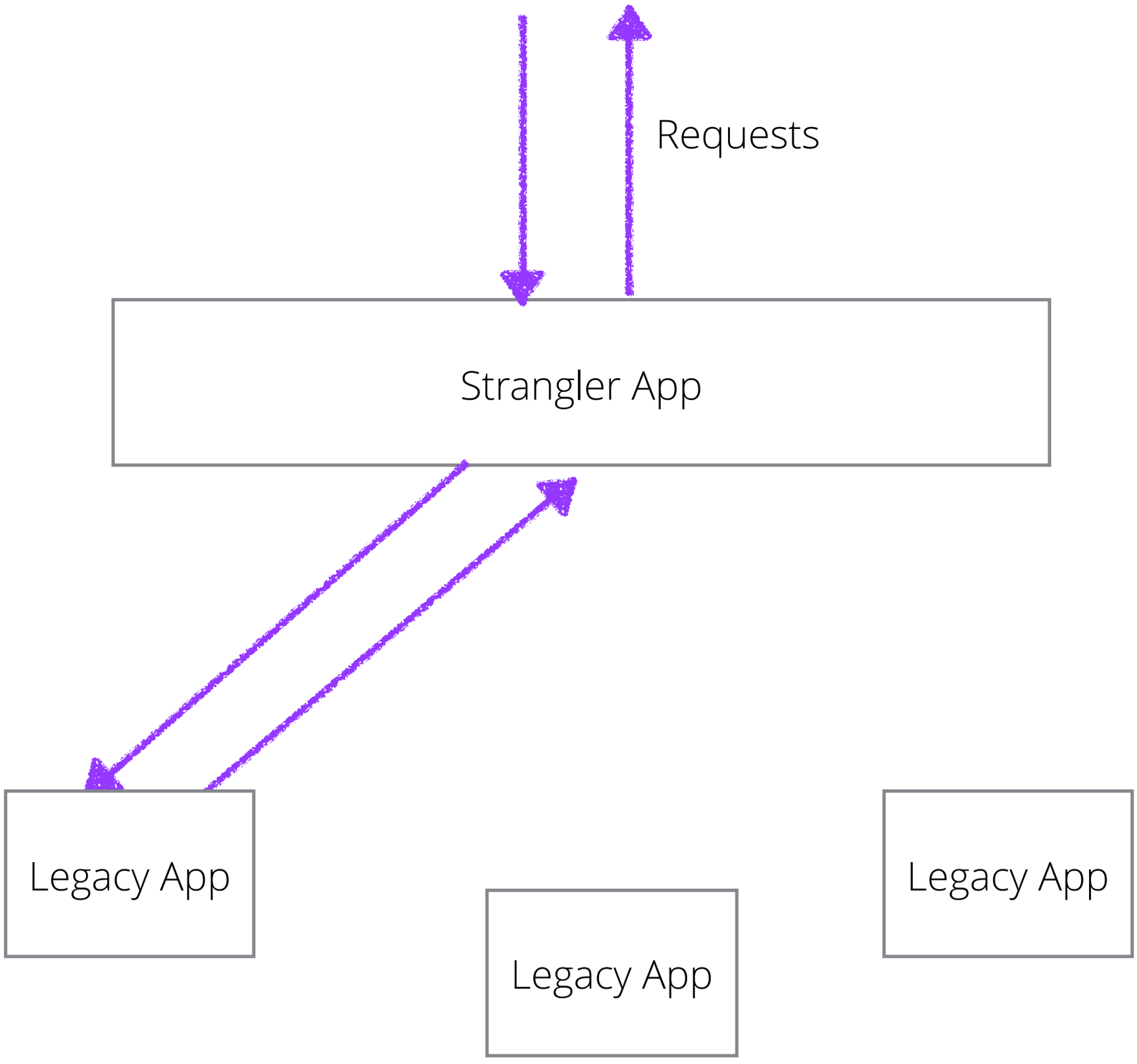
Legacy App

Legacy App

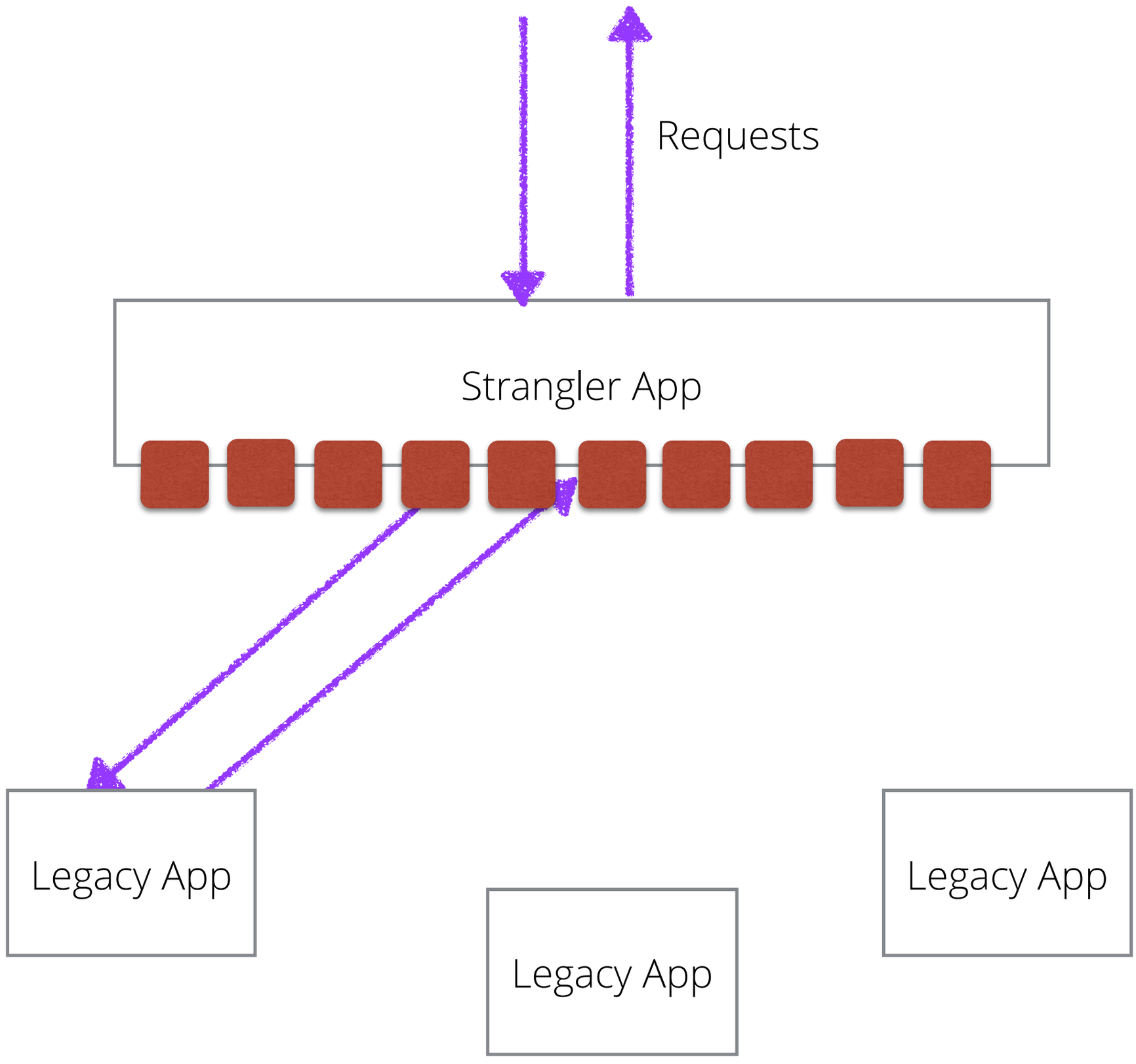
@samnewman



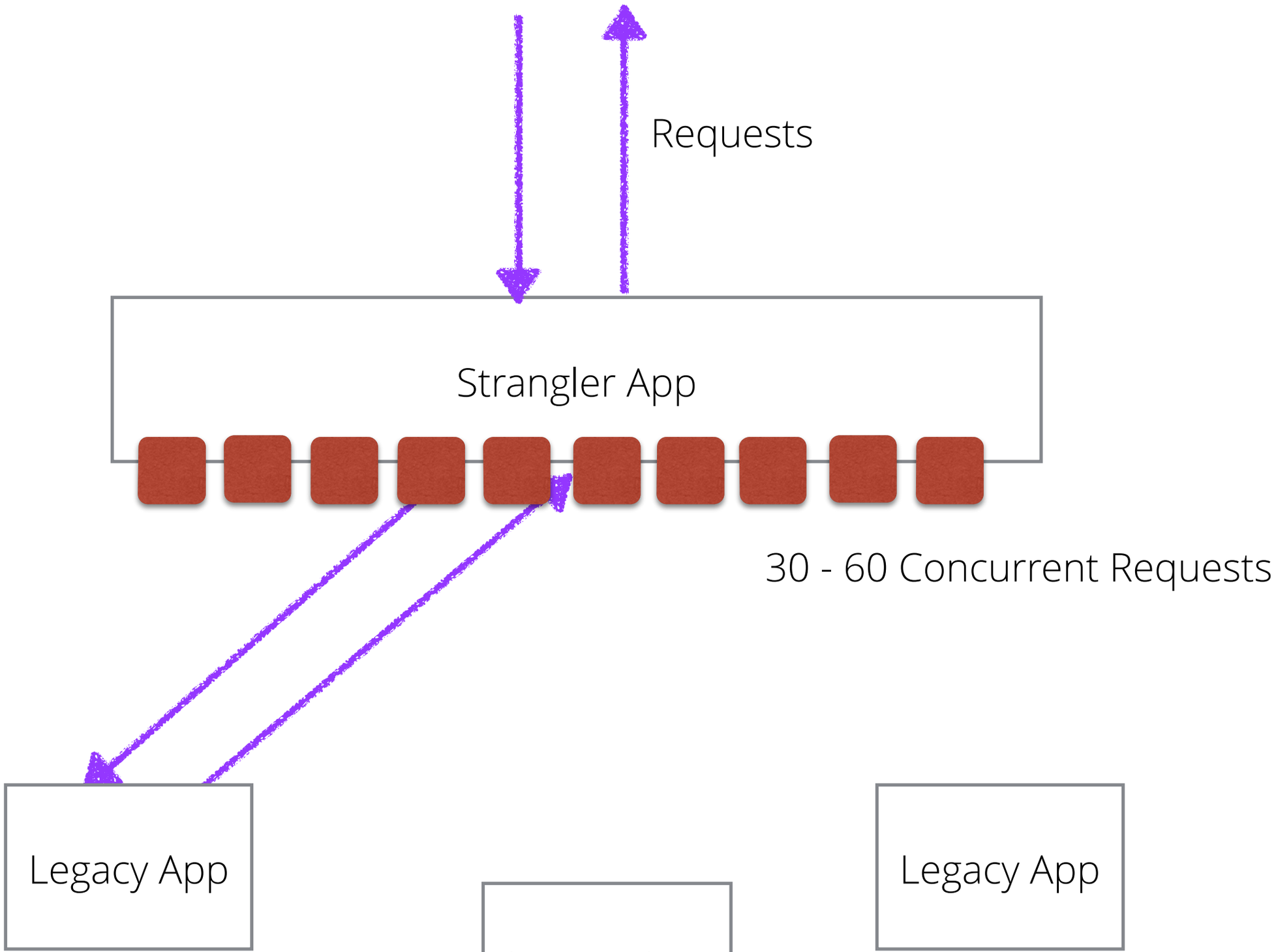
@samnewman



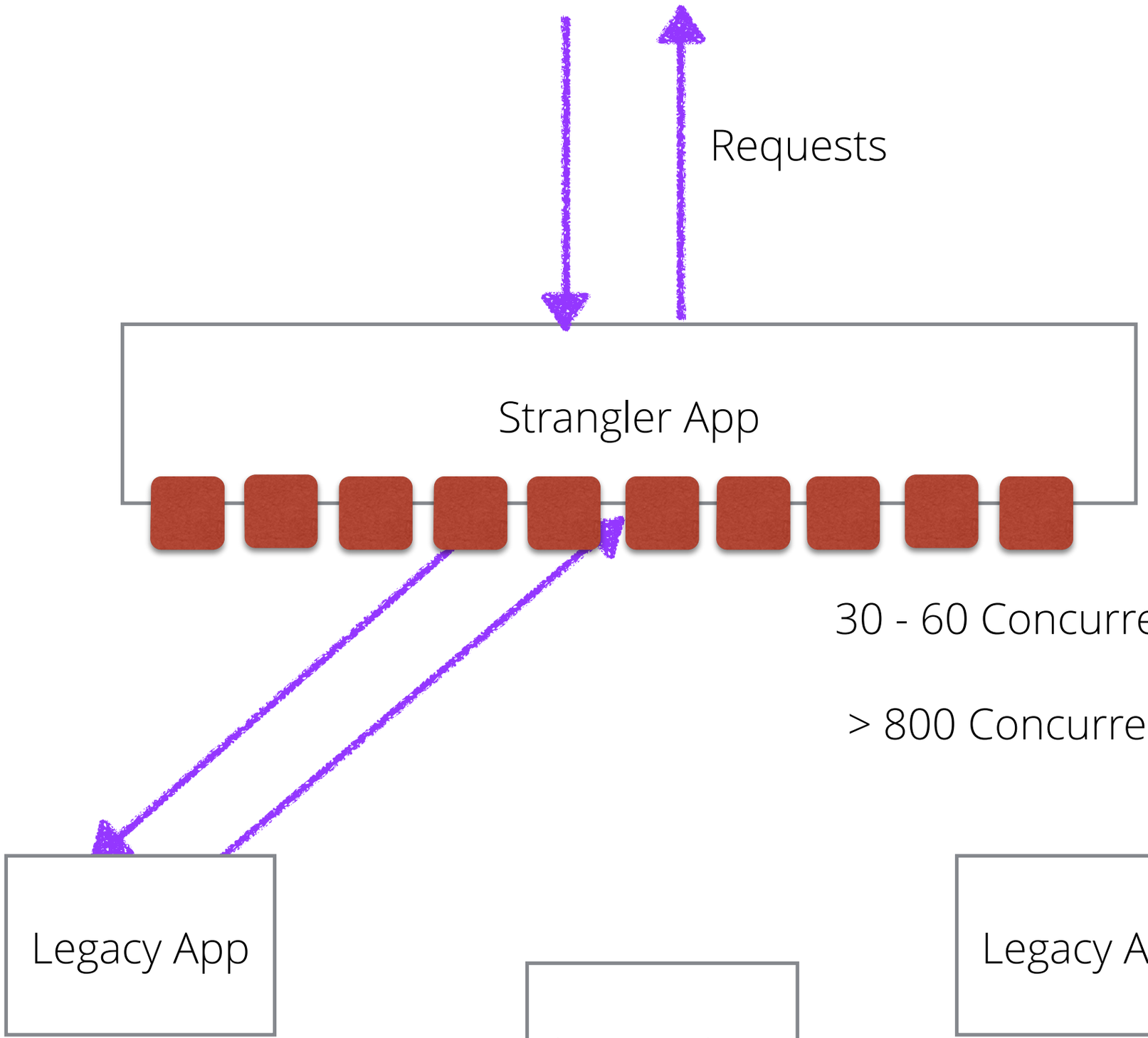
@samnewman



@samnewman

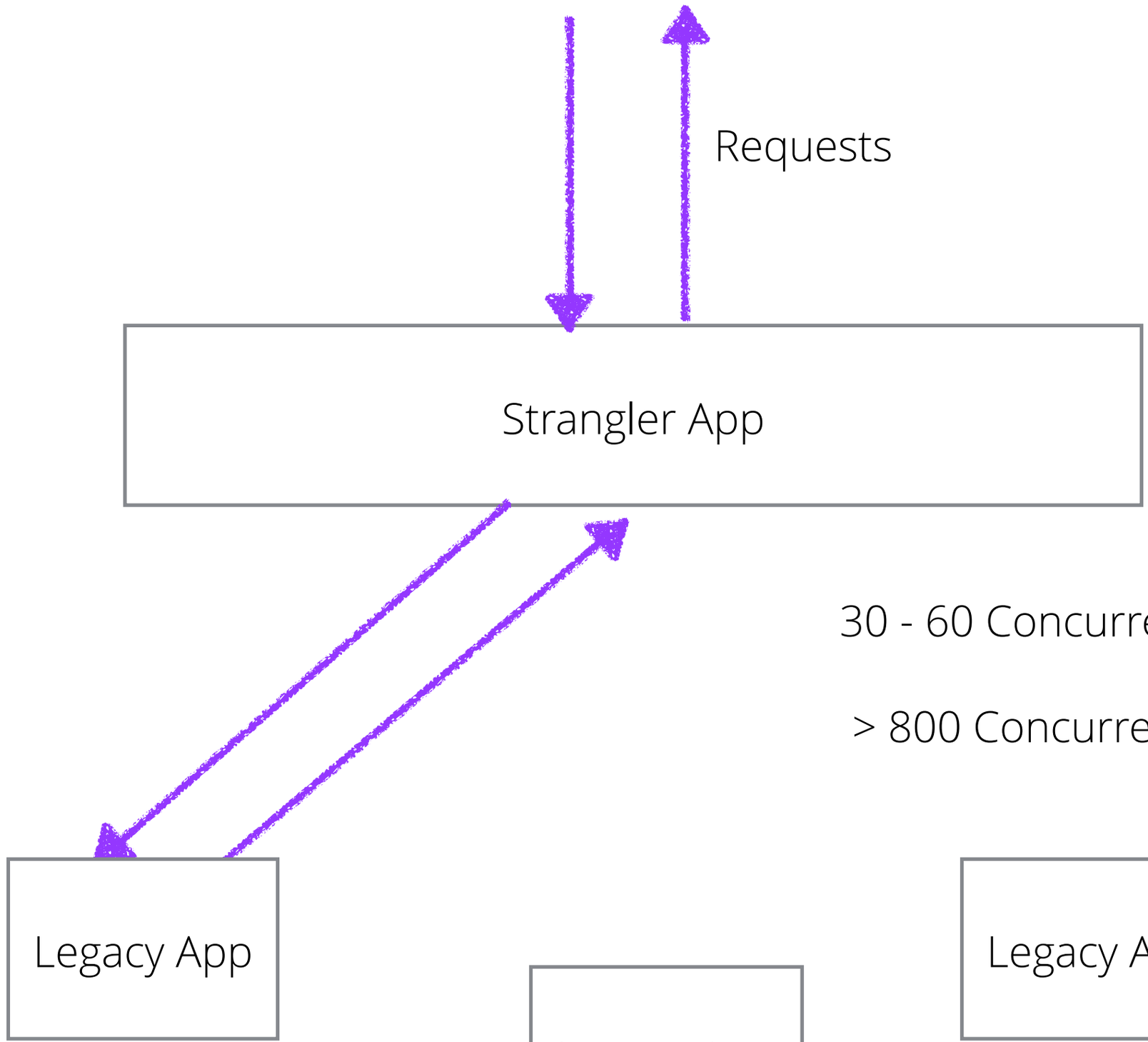

Requests
Strangler App
30 - 60 Concurrent Requests
Legacy App
Legacy App
Legacy App

@samnewman



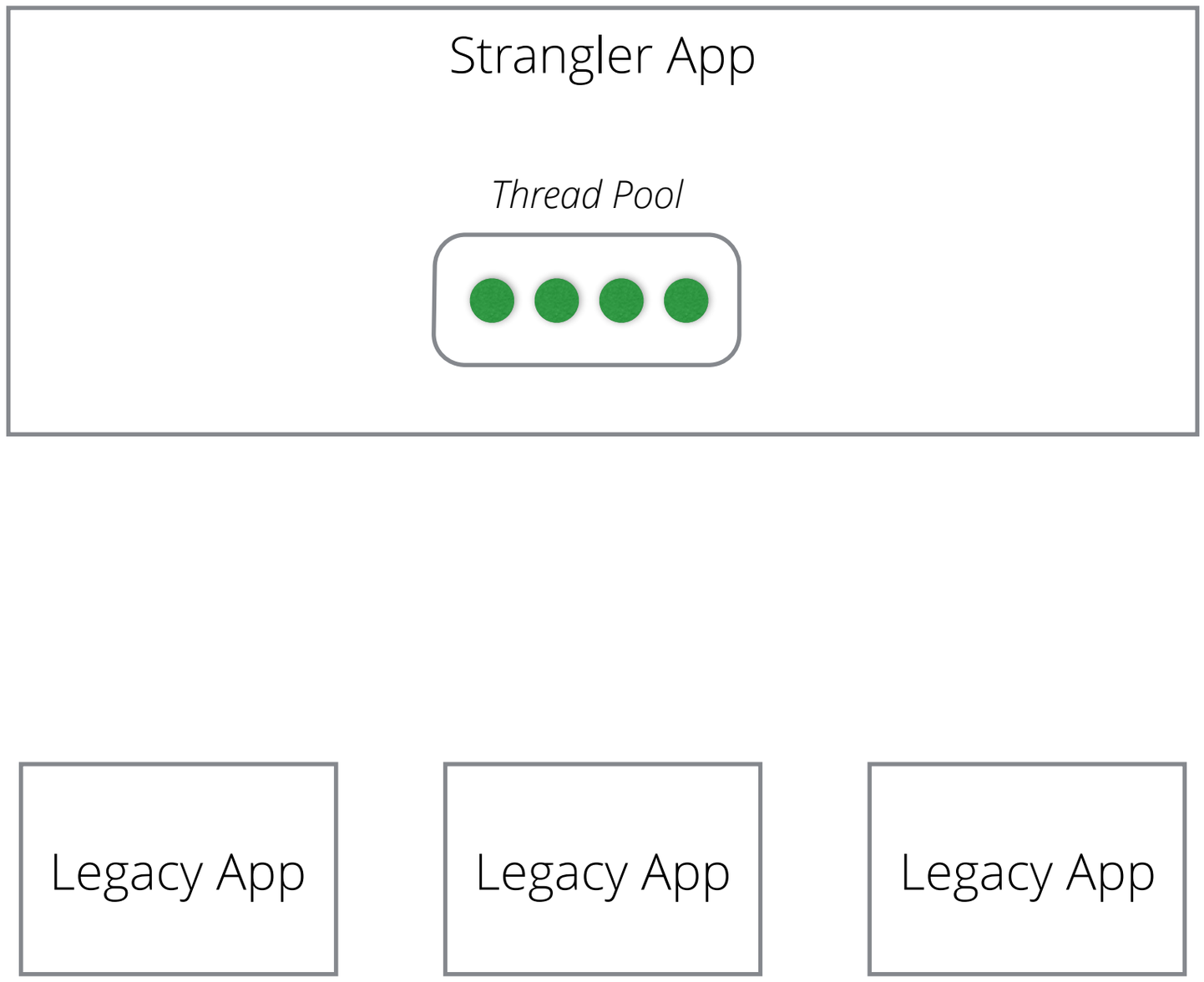
30 - 60 Concurrent Requests

> 800 Concurrent Requests



30 - 60 Concurrent Requests

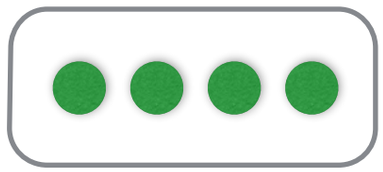
> 800 Concurrent Requests



@samnewman

Strangler App

Thread Pool



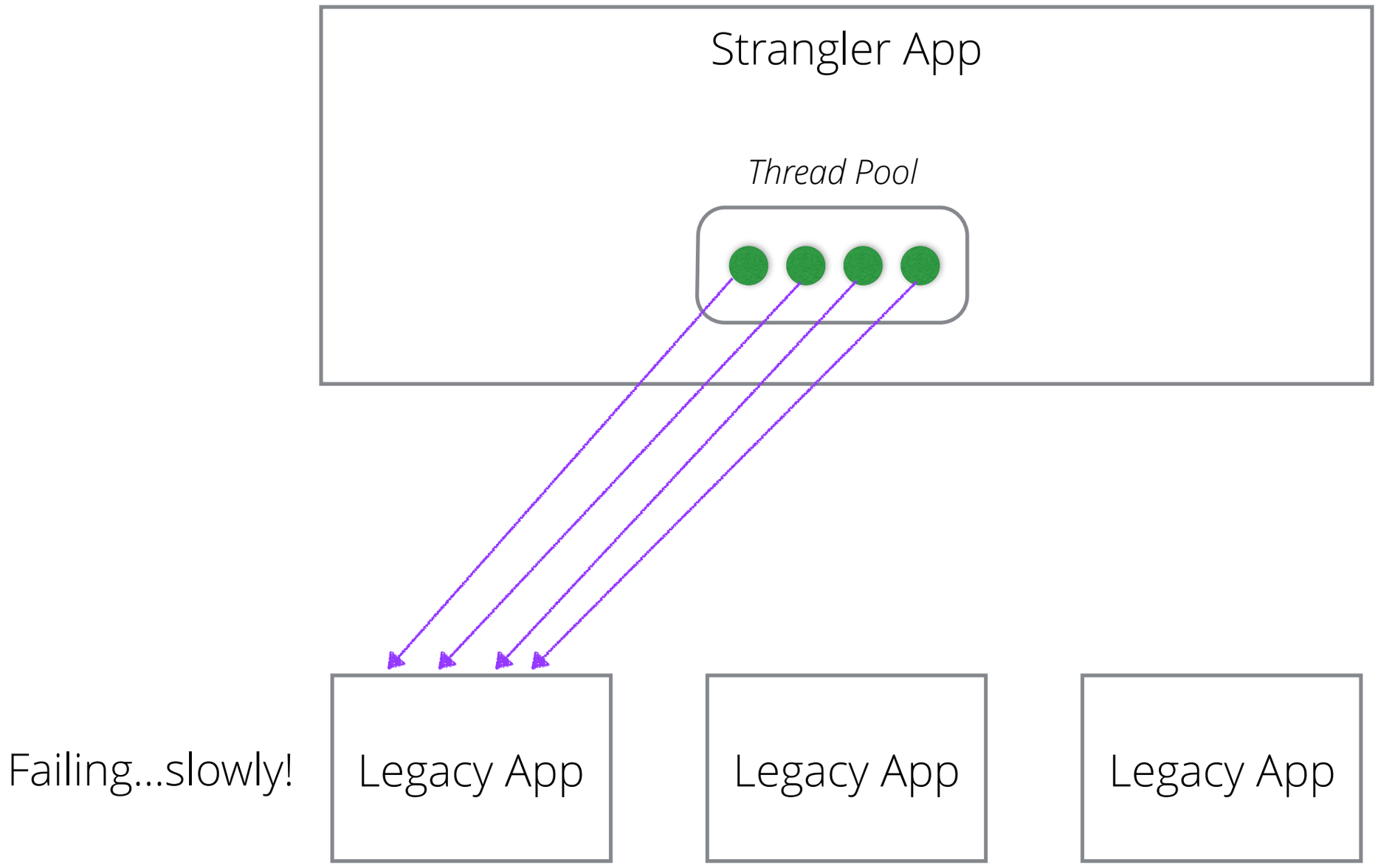
Failing...slowly!

Legacy App

Legacy App

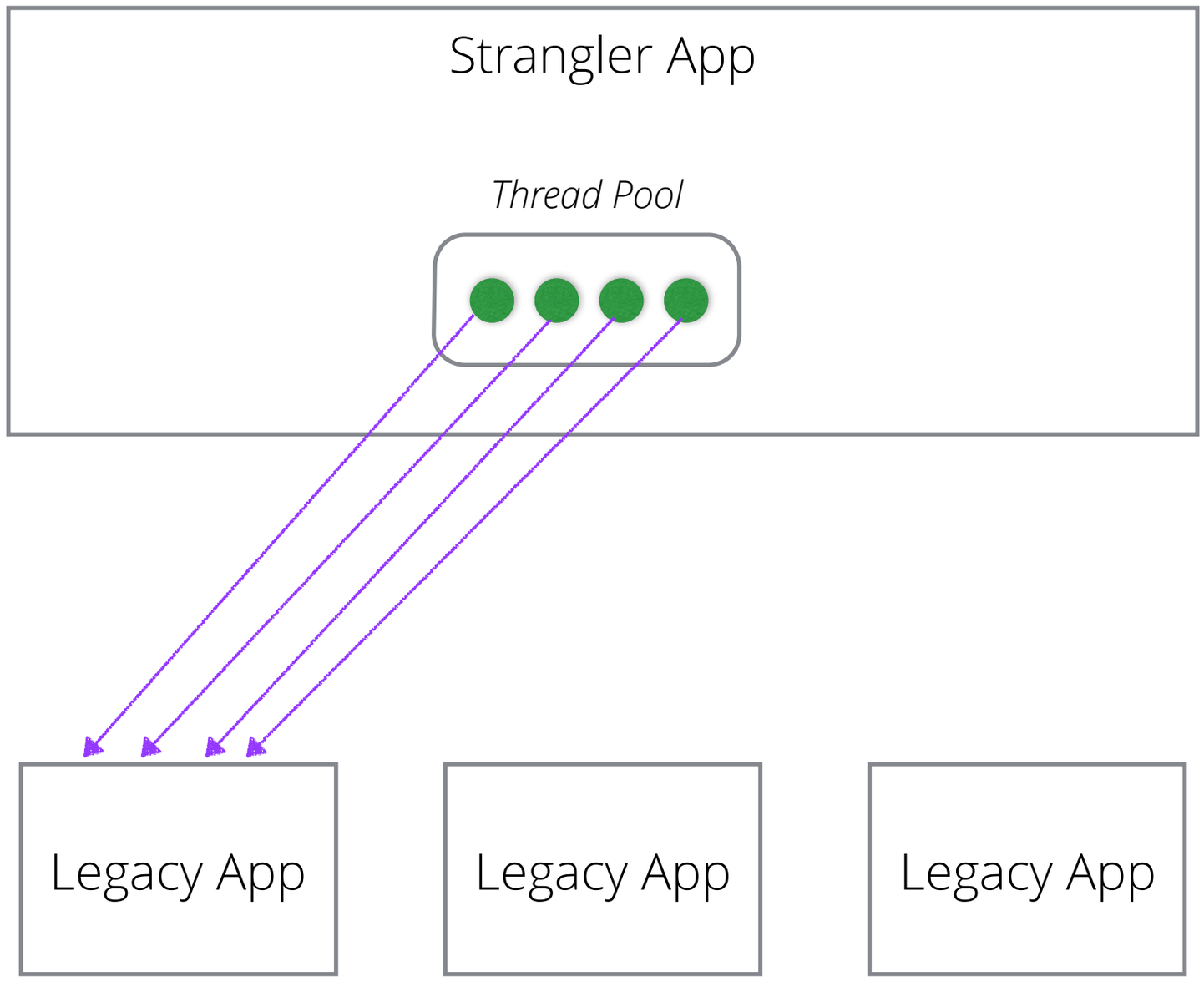
Legacy App

@samnewman


Strangler App
Thread Pool
Failing...slowly!
Legacy App
Legacy App
Legacy App

@samnewman

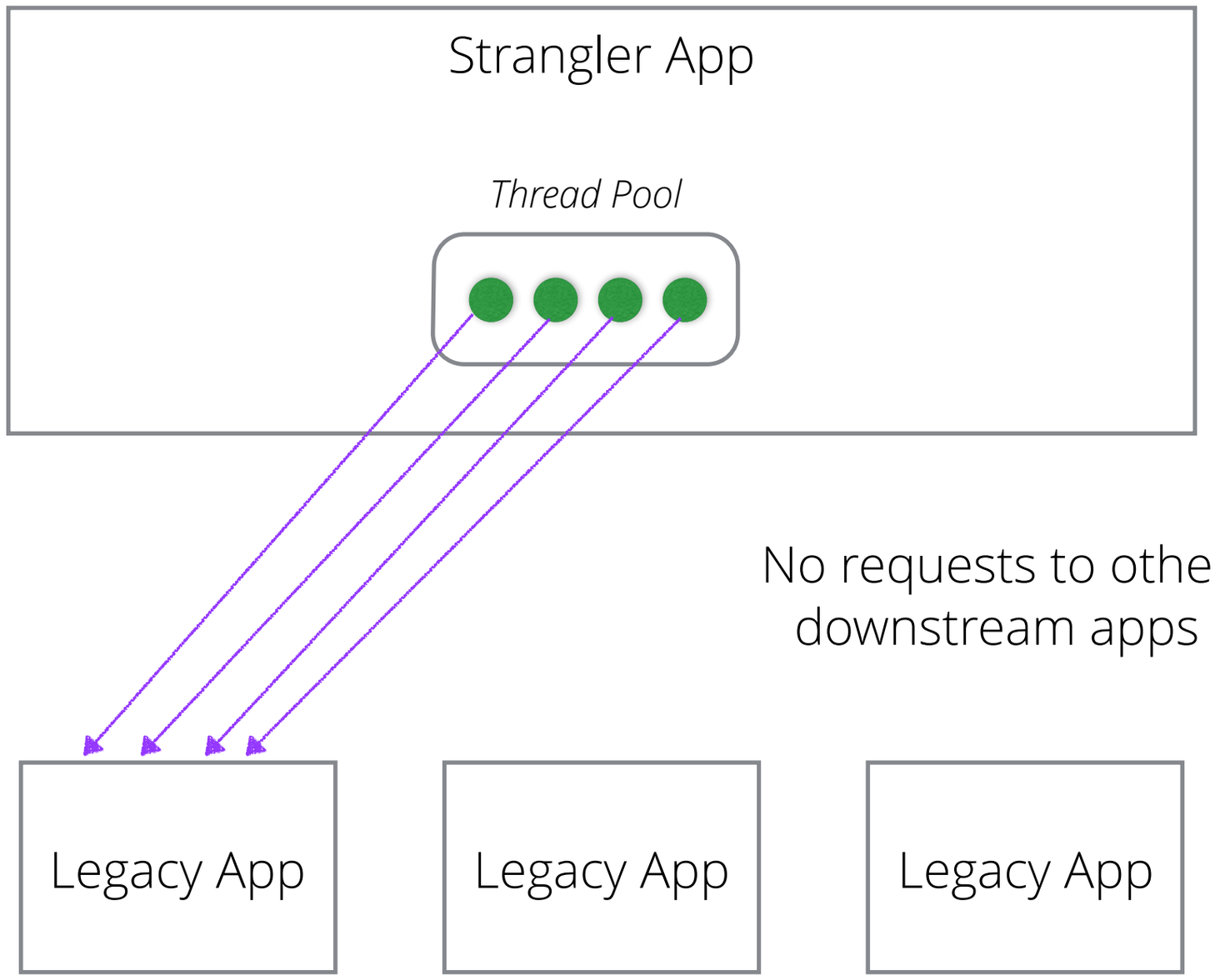
Thread-pool exhausted



Failing...slowly!

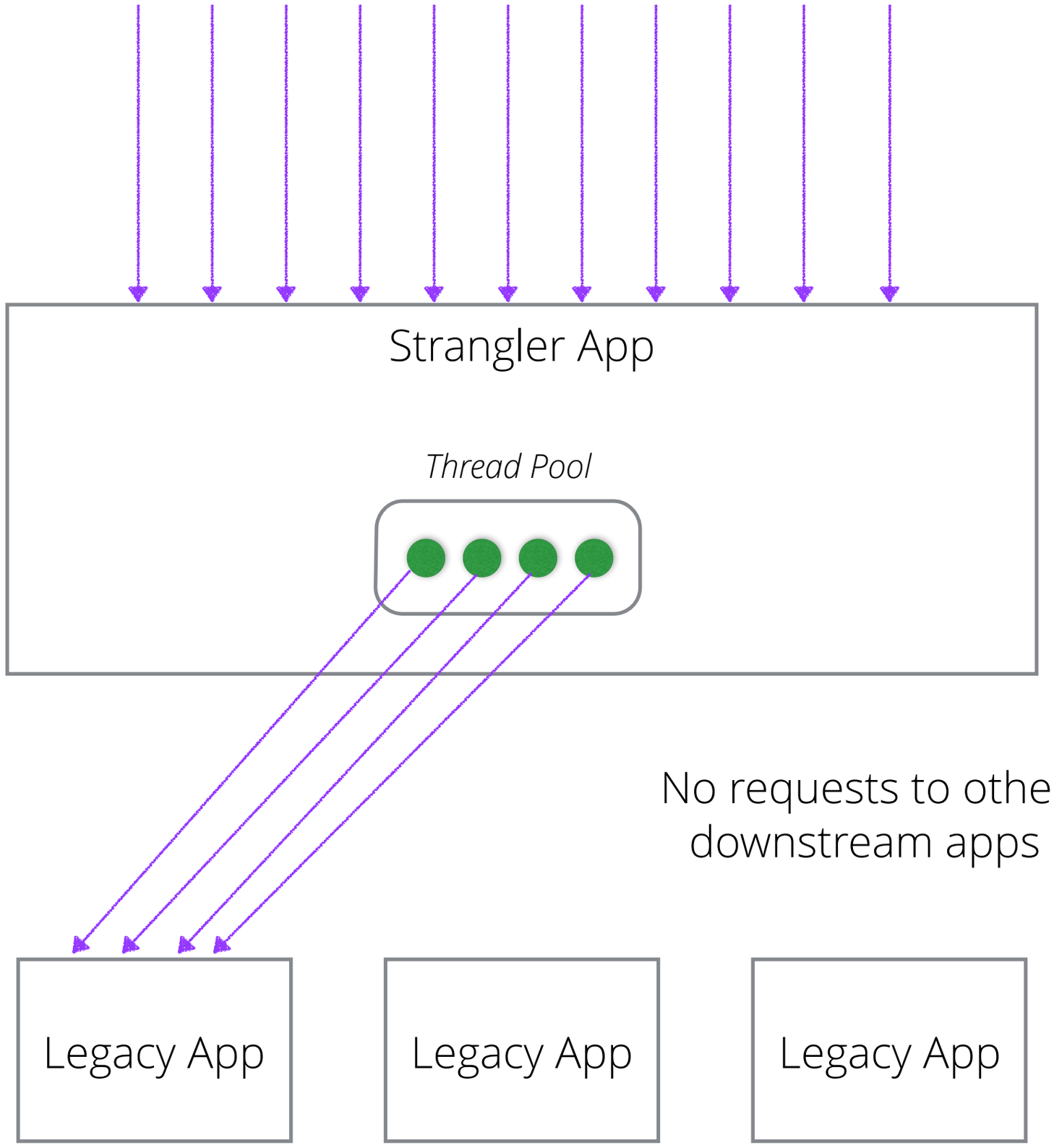
@samnewman

Thread-pool exhausted



@samnewman

Thread-pool exhausted

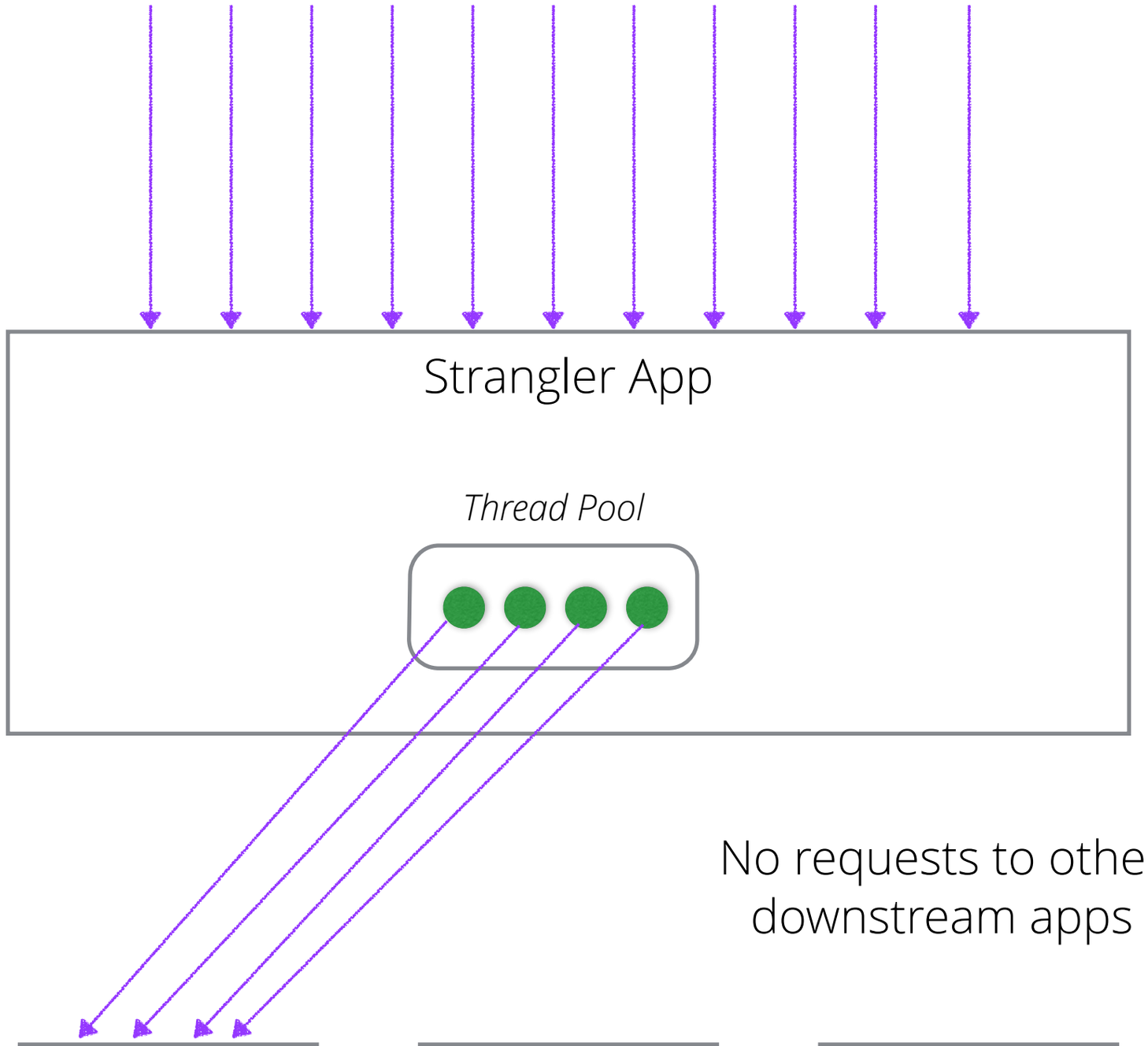


No requests to other downstream apps

Failing...slowly!

@samnewman

Requests Building Up

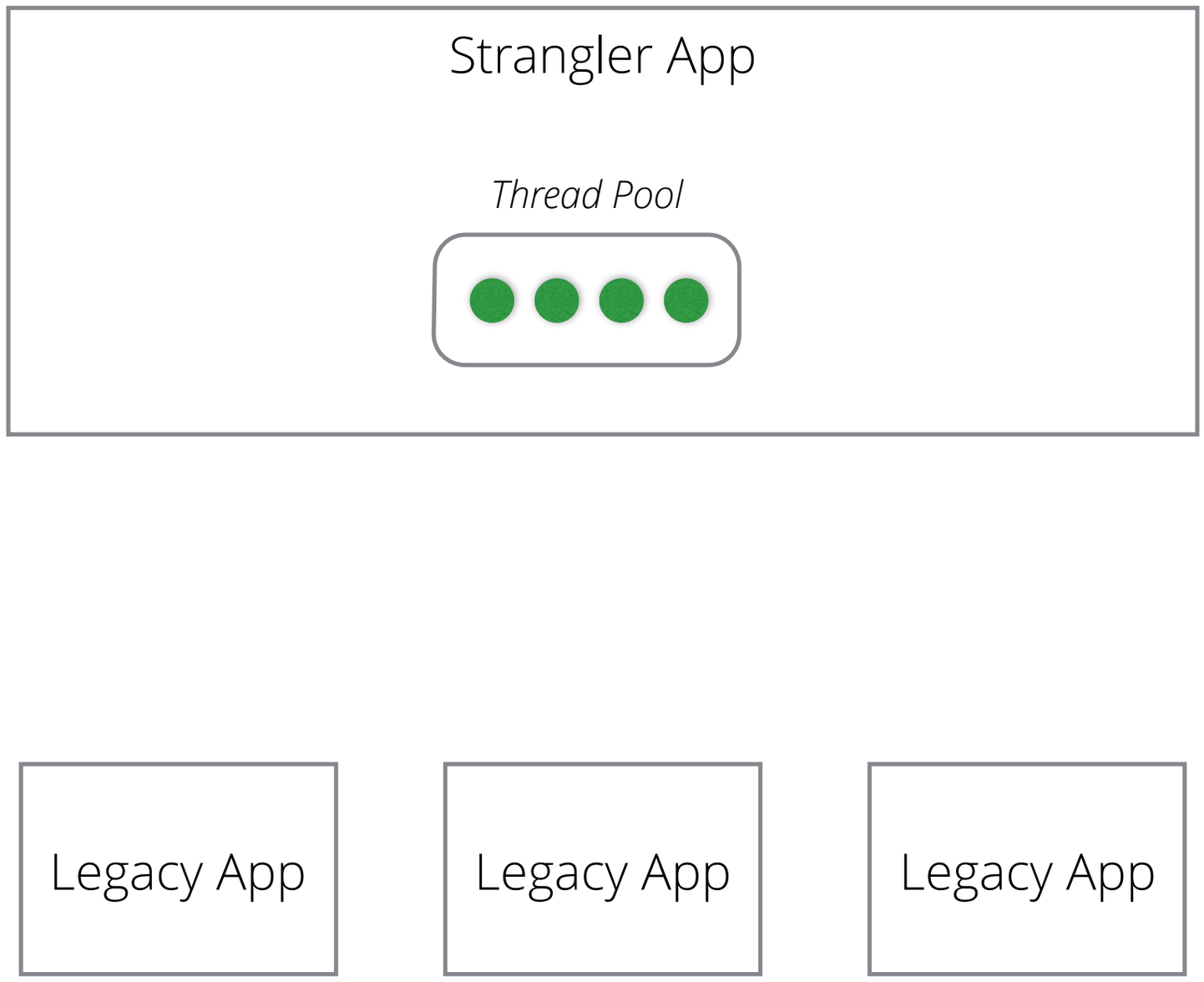


Thread-pool exhausted

No requests to other downstream apps

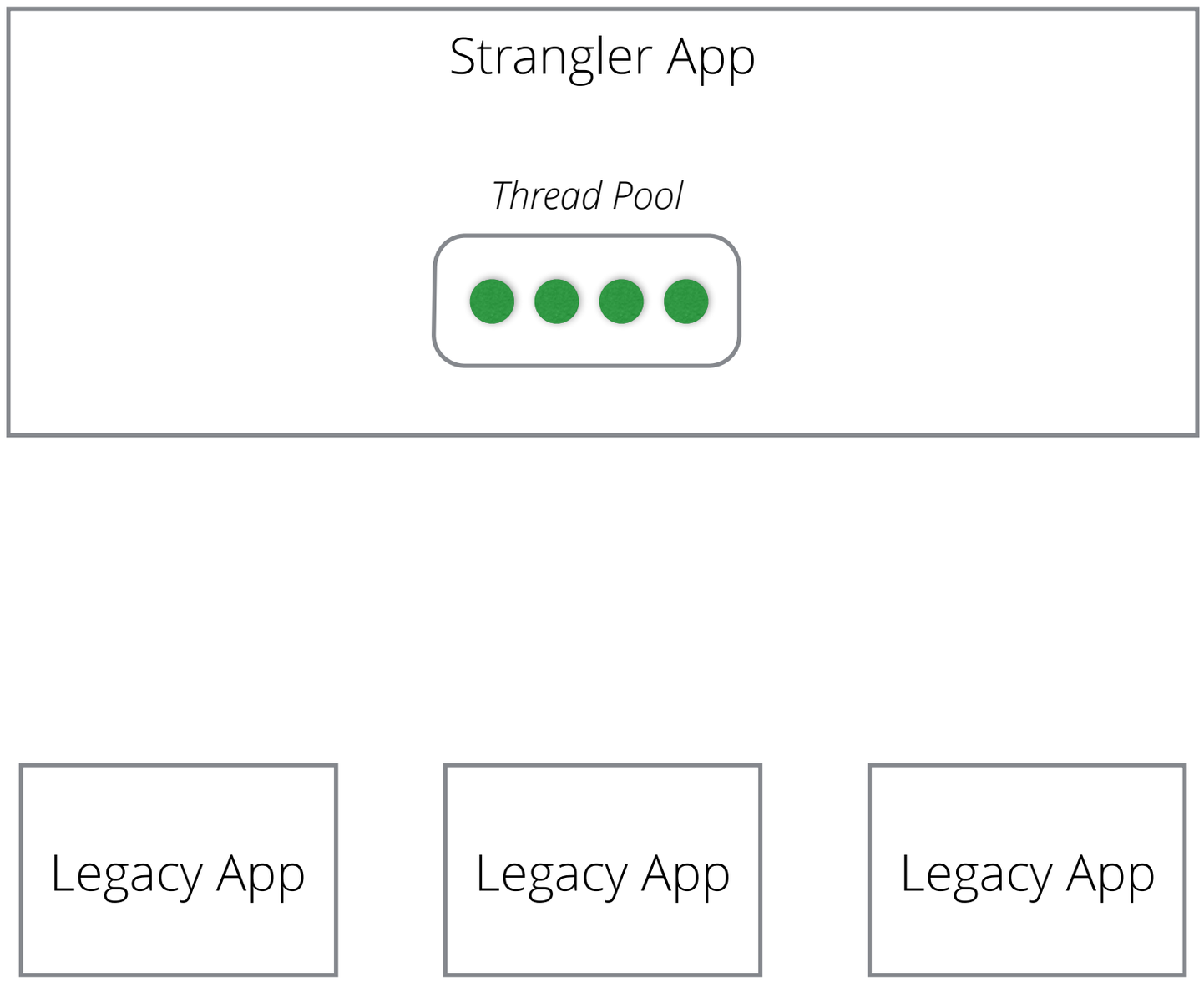
Failing...slowly!

@samnewman



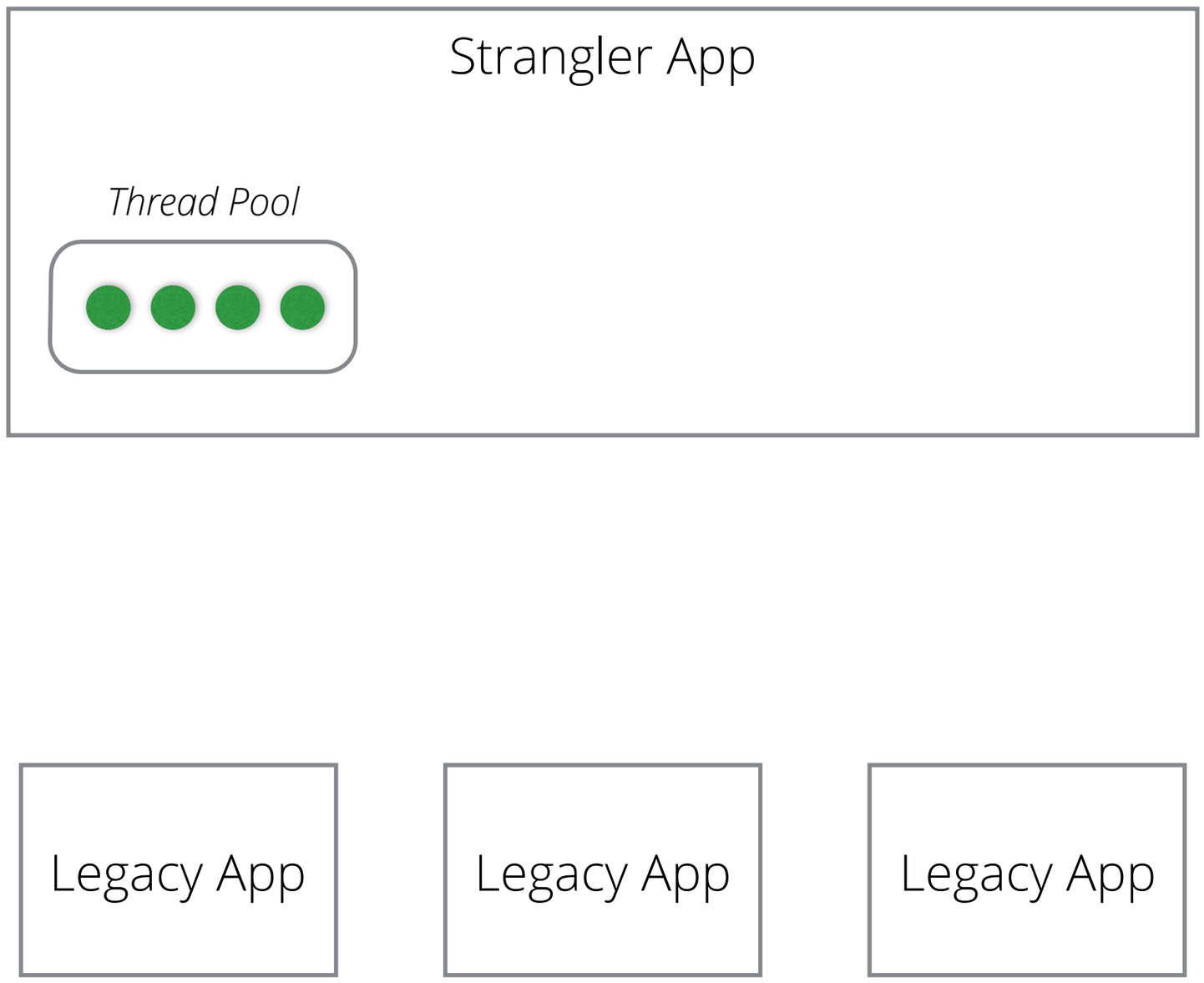
@samnewman

Fix **Timeouts**



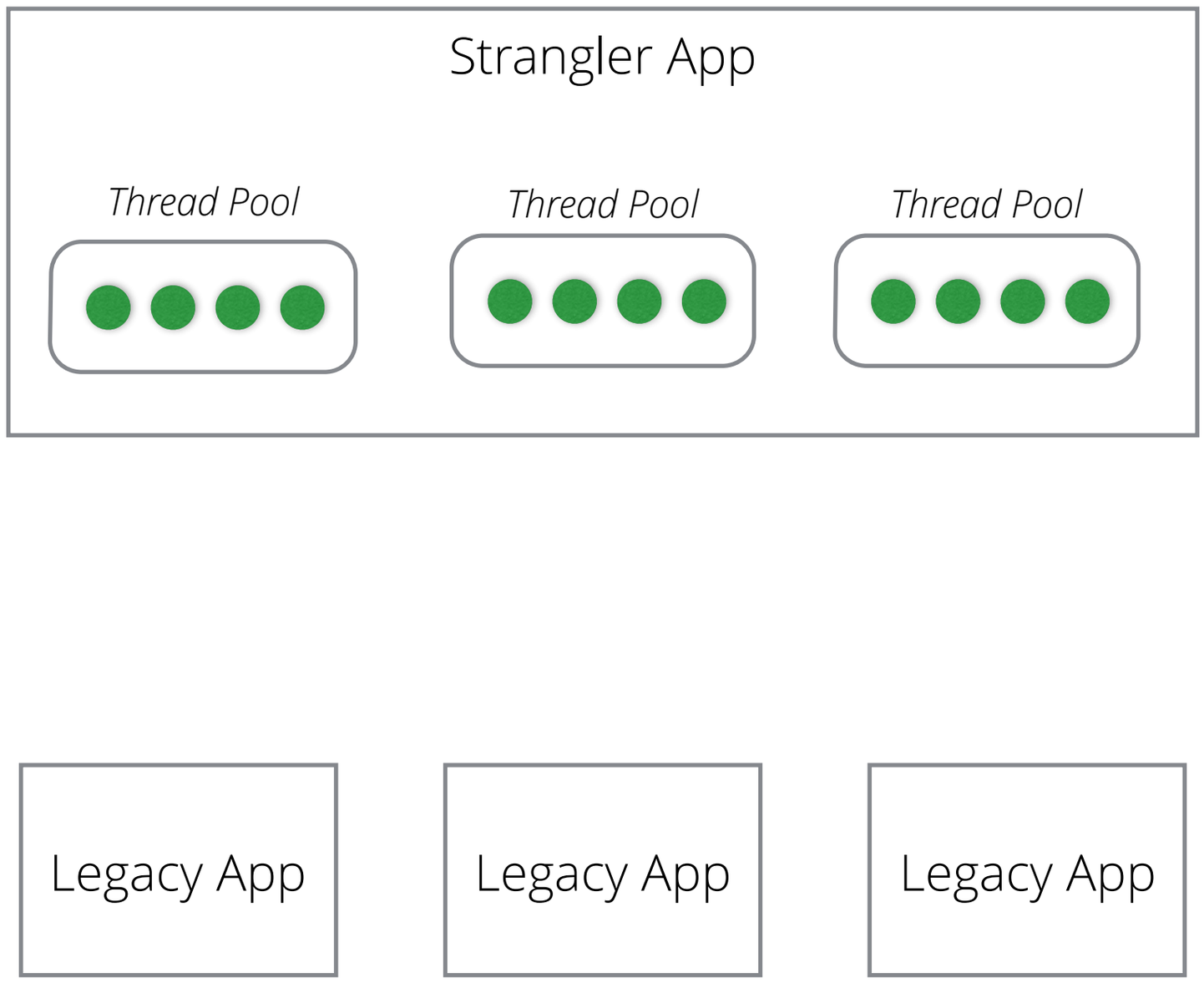
@samnewman

Fix **Timeouts**



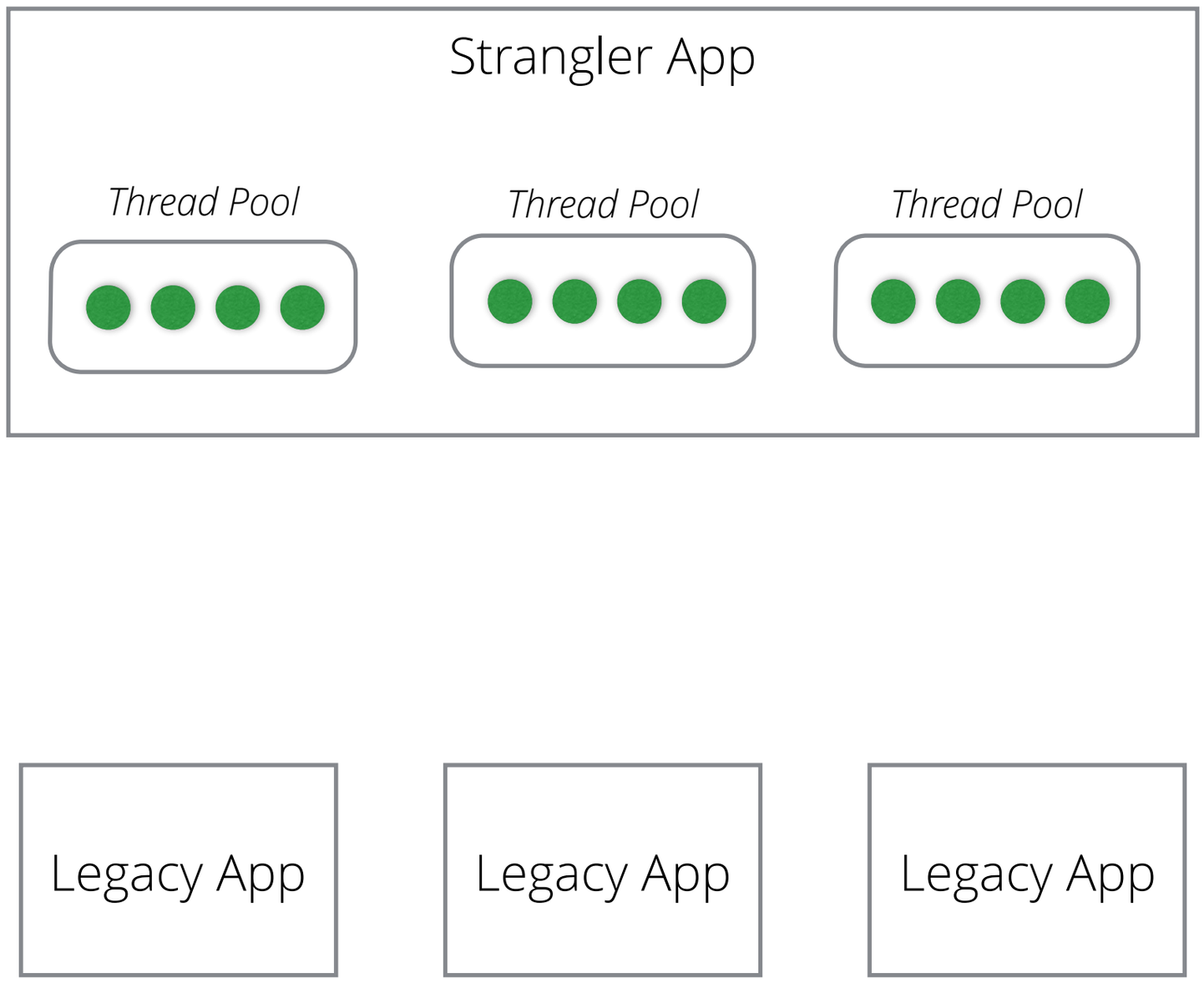
@samnewman

Fix **Timeouts**



@samnewman

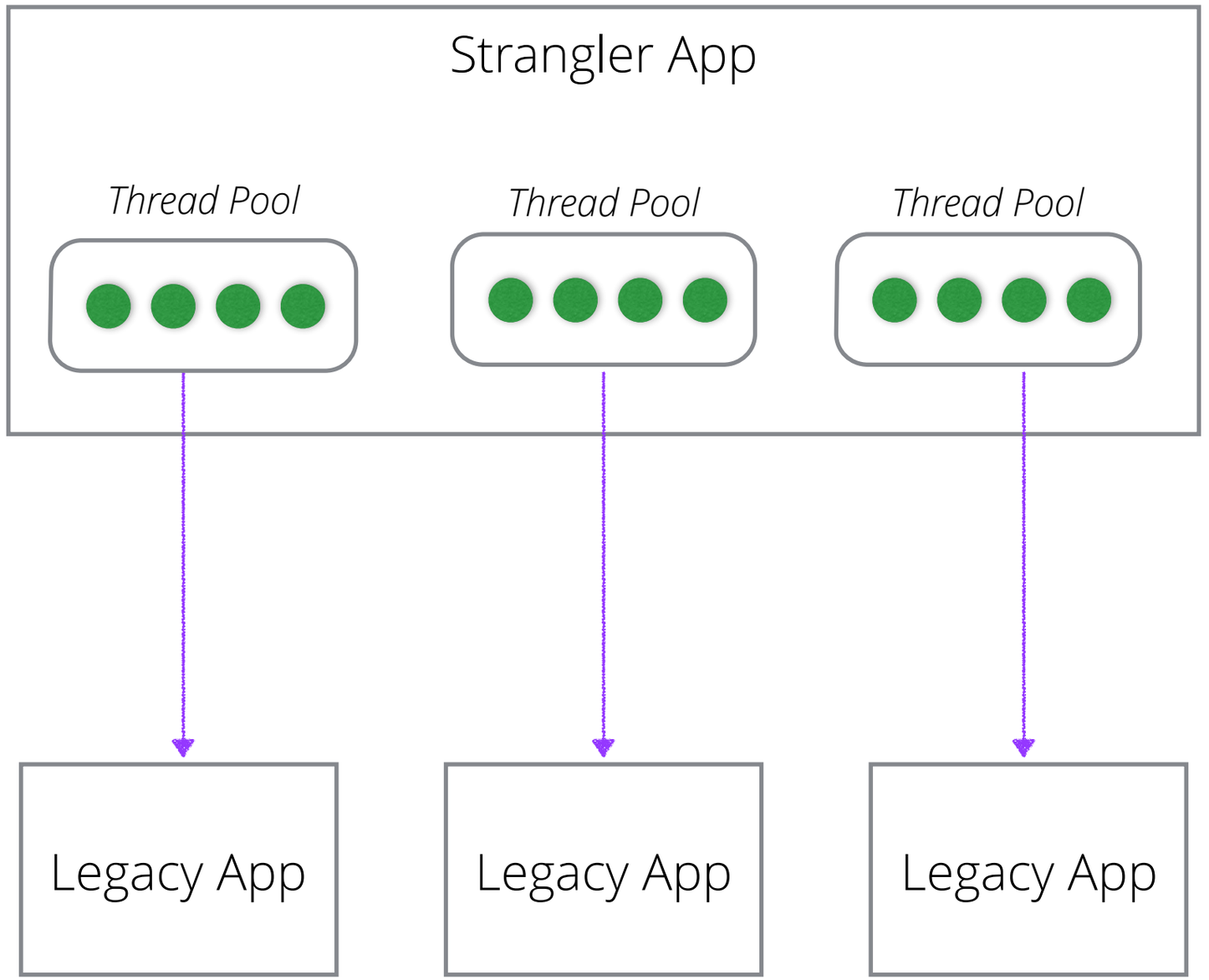
Fix **Timeouts**



Bulkhead
Downstream
Connections

@samnewman

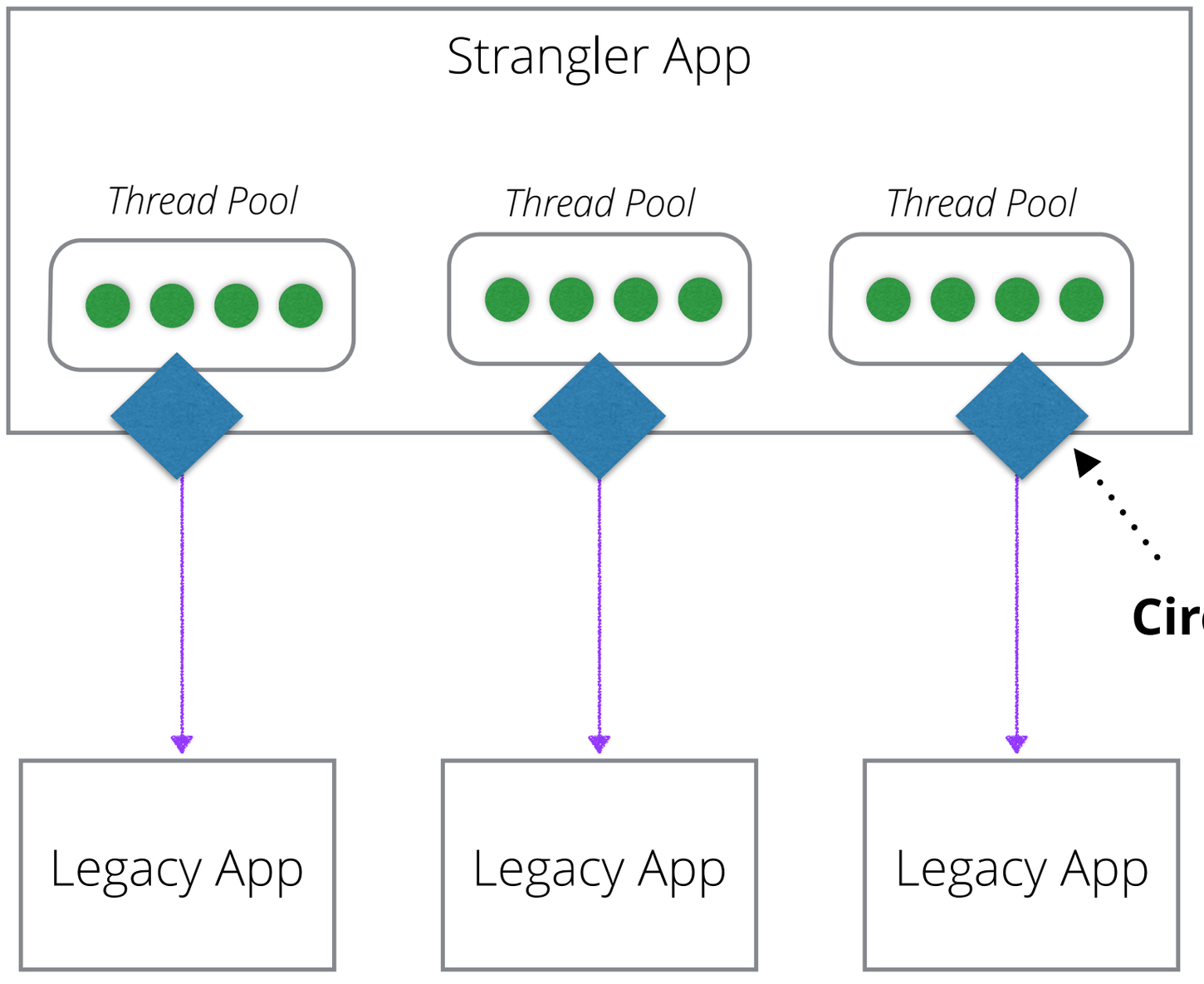
Fix **Timeouts**



Bulkhead
Downstream
Connections

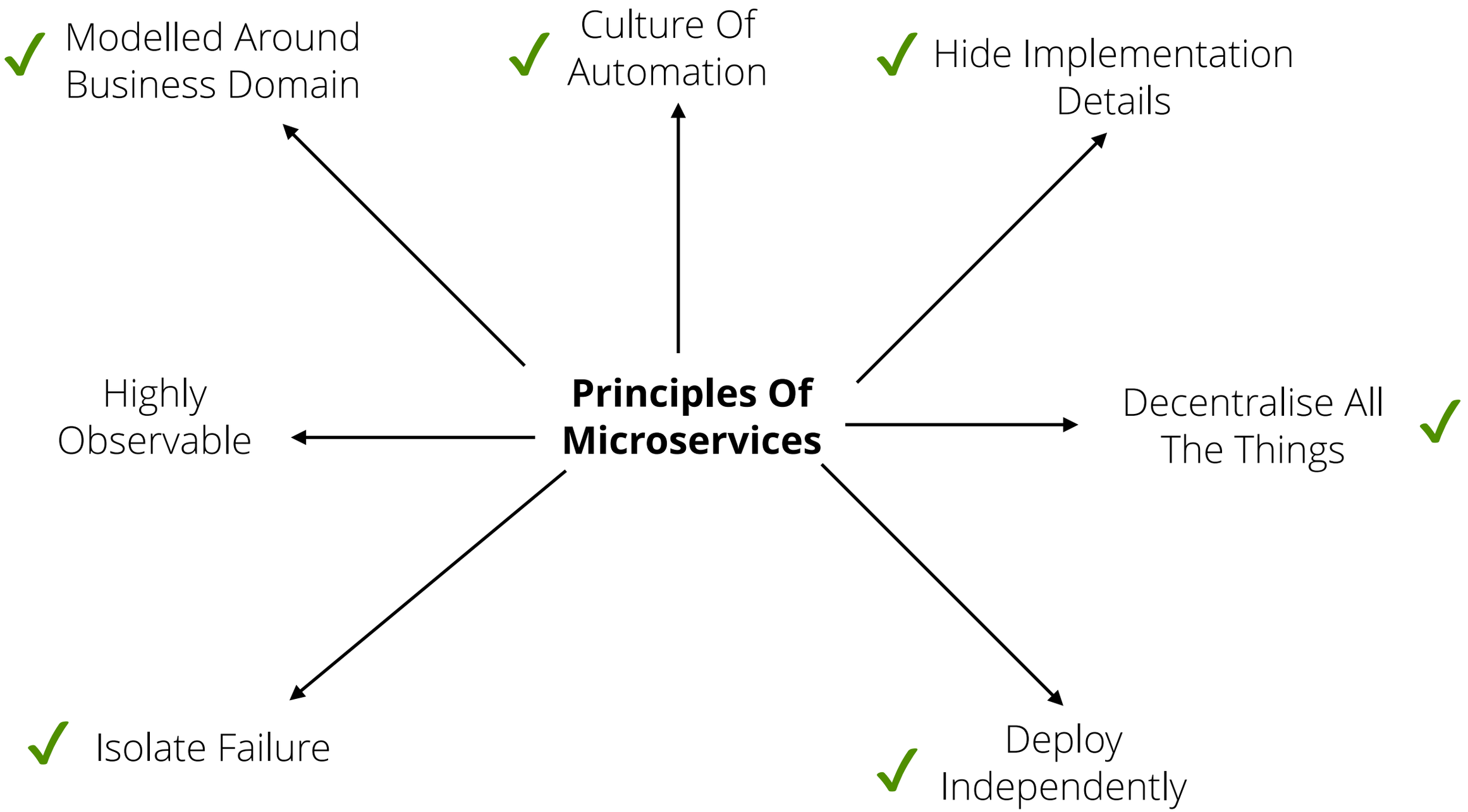
@samnewman

Fix **Timeouts**



Bulkhead
Downstream
Connections

@samnewman


✓ Modelled Around Business Domain
✓ Culture Of Automation
✓ Hide Implementation Details
Highly Observable
Principles Of Microservices
Decentralise All The Things ✓
✓ Isolate Failure
✓ Deploy Independently

@samnewman

✓ Modelled Around Business Domain

✓ Culture Of Automation

✓ Hide Implementation Details

Highly Observable

Principles Of Microservices

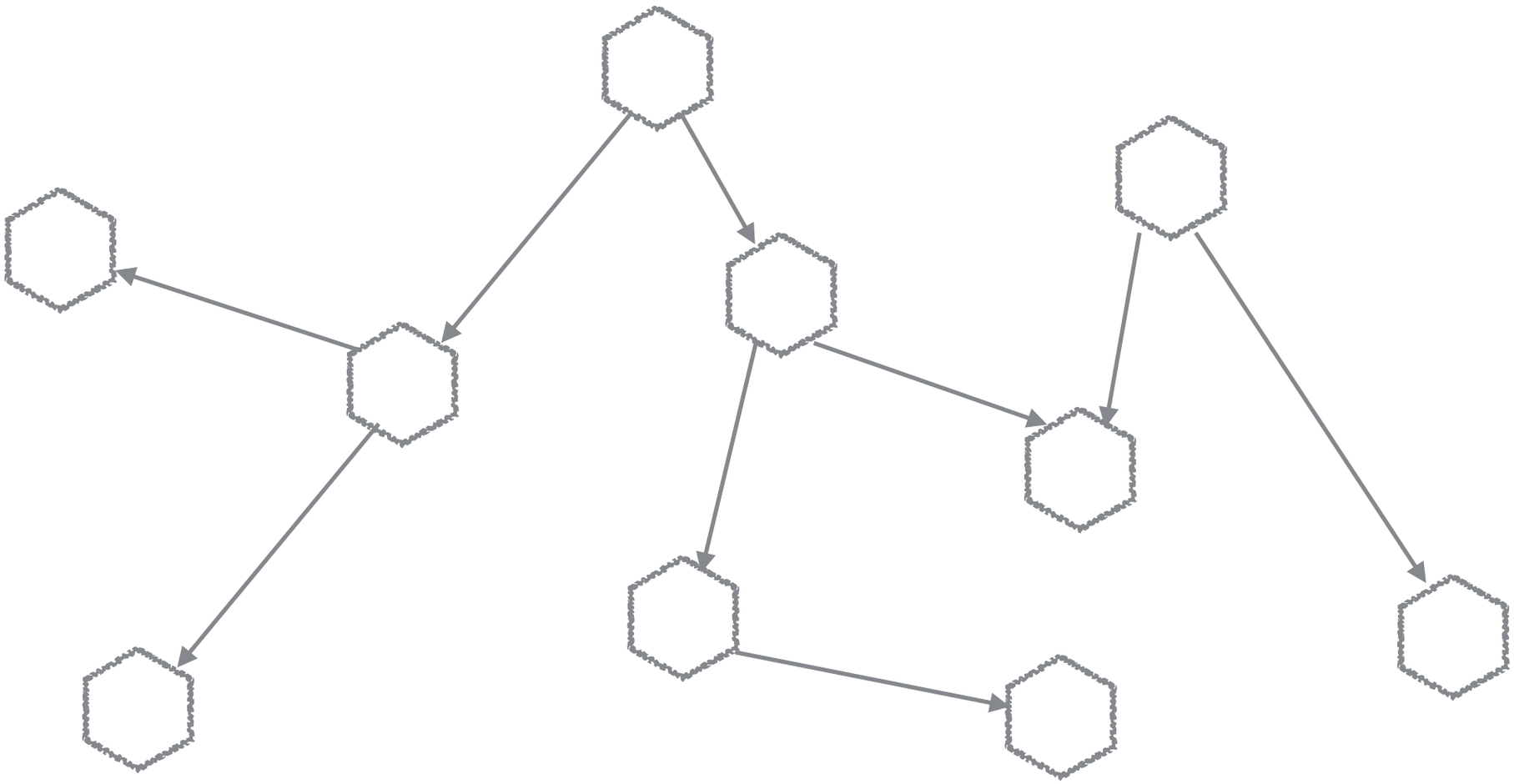
Decentralise All The Things ✓

✓ Isolate Failure

✓ Deploy Independently

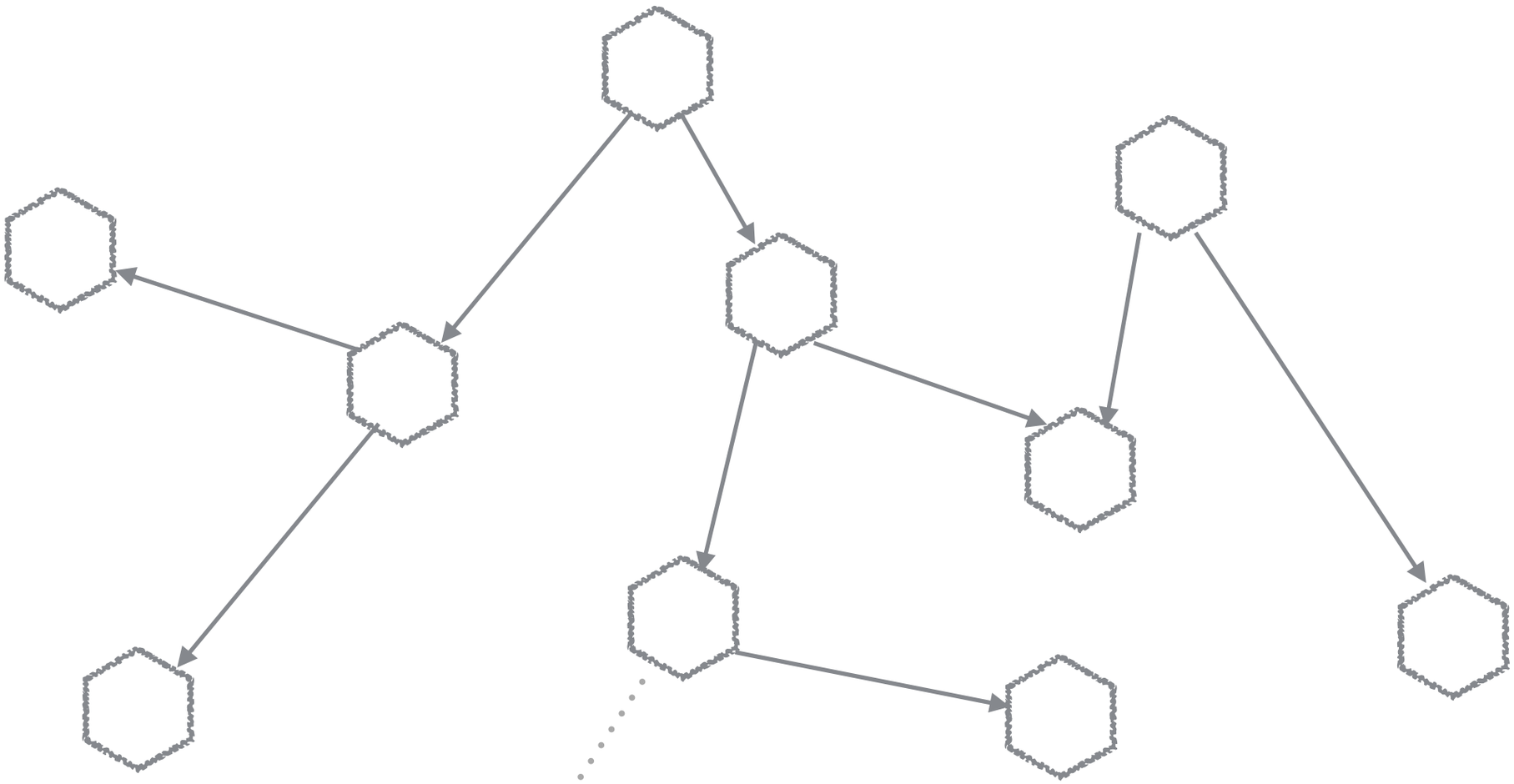
@samnewman

AGGREGATION



@samnewman

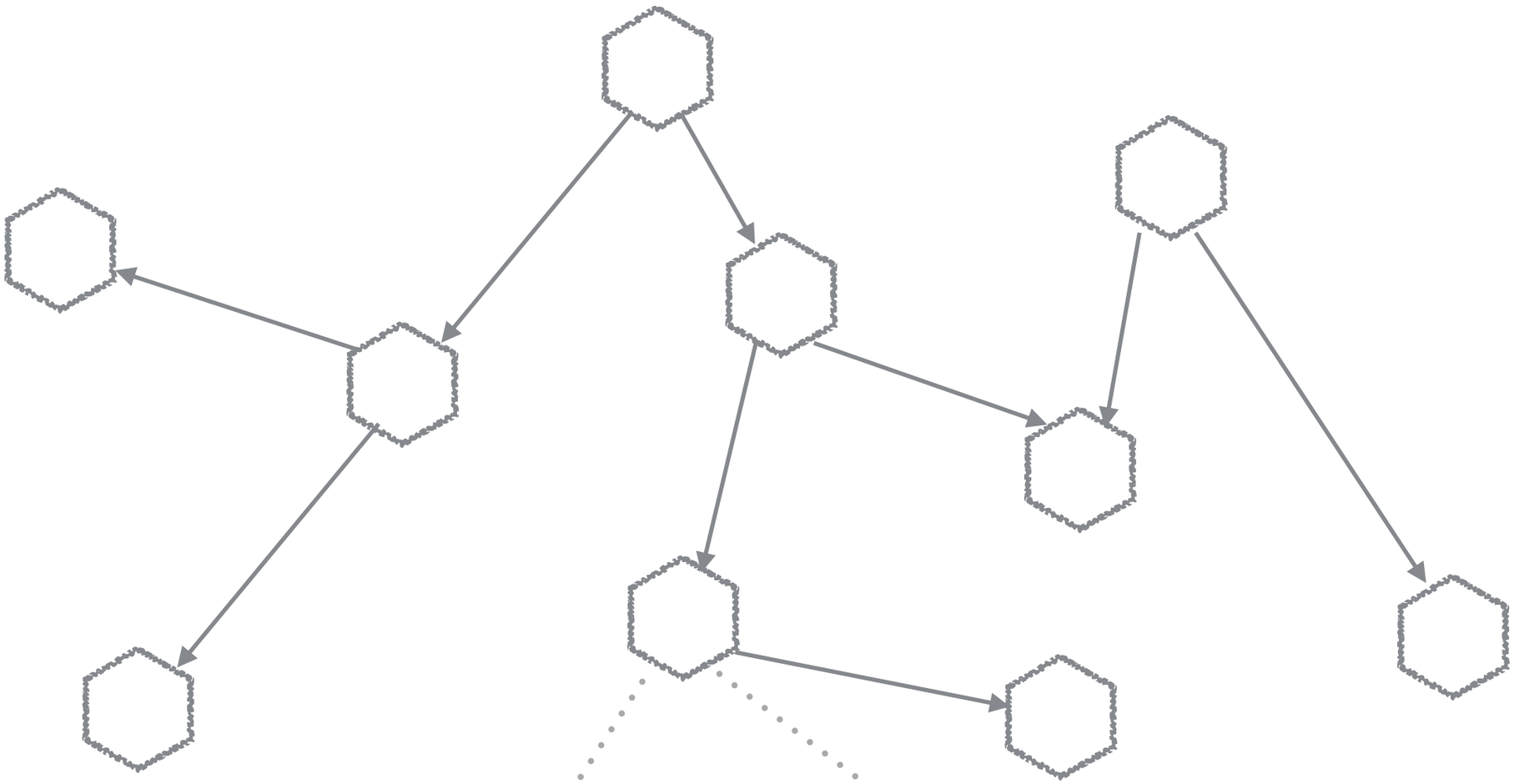
AGGREGATION



LOGS

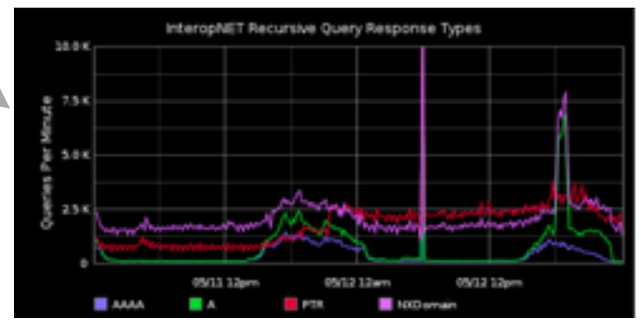
@samnewman

AGGREGATION



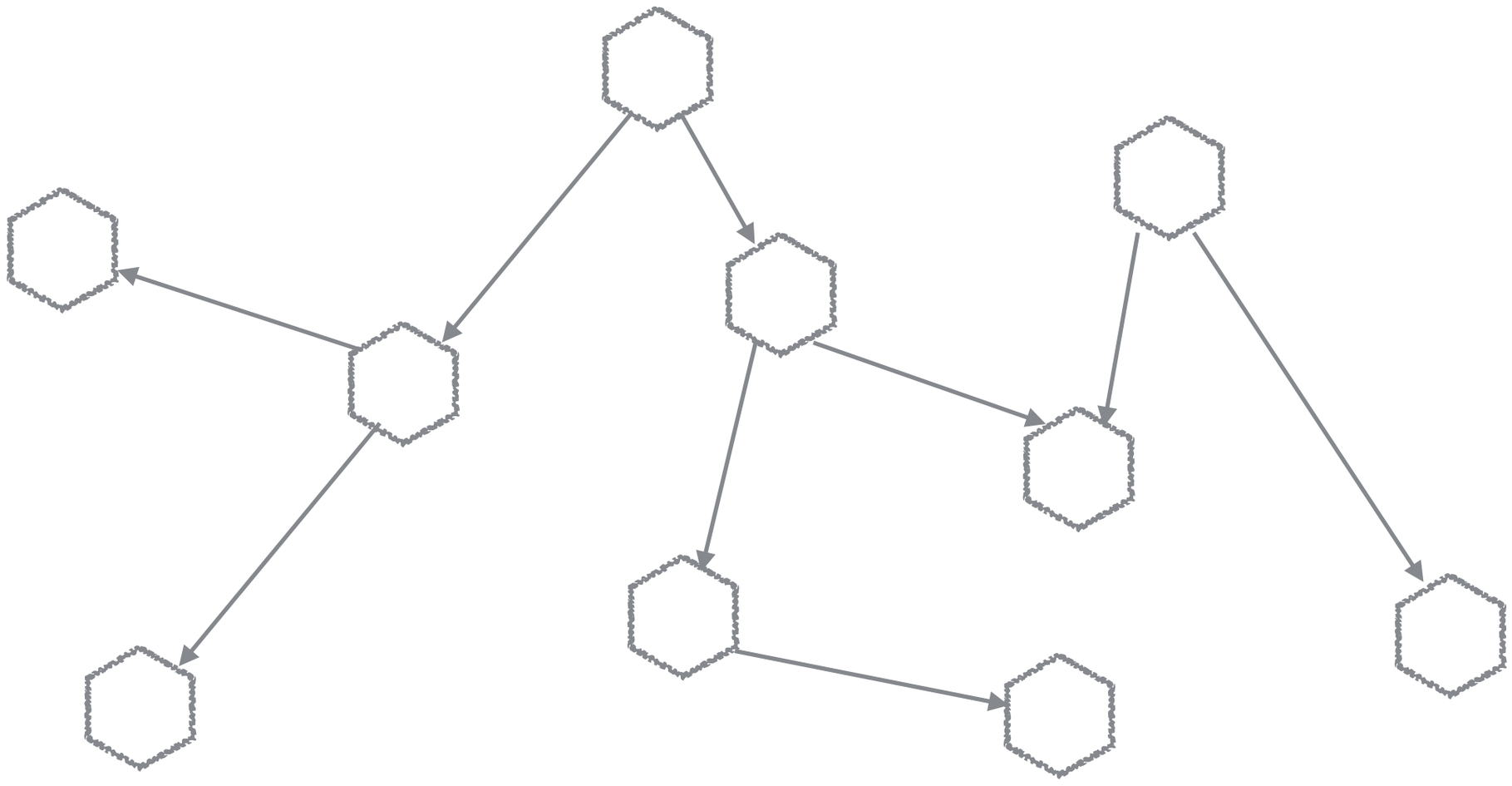
LOGS

STATS



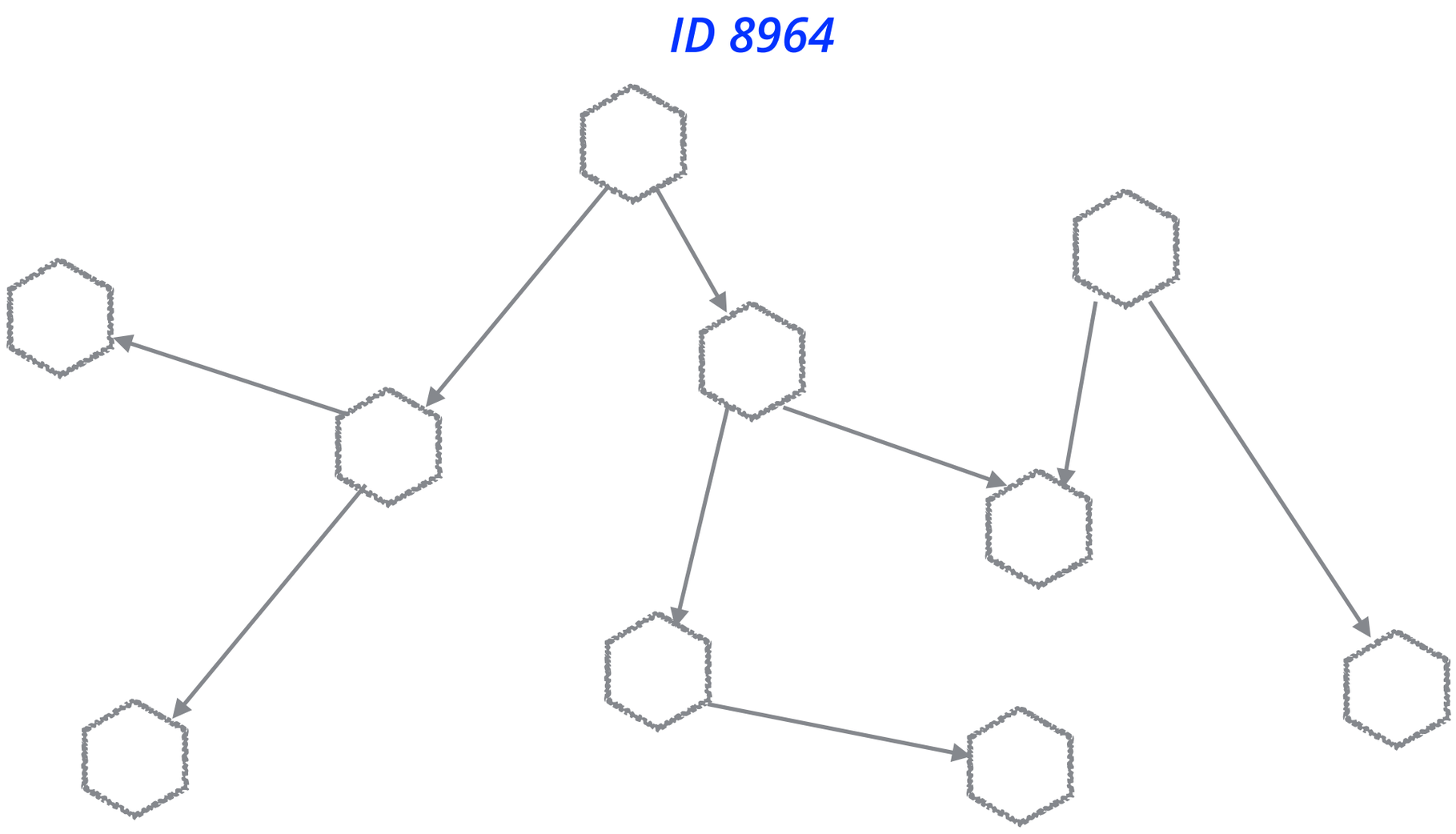
@samnewman

CORRELATION IDS



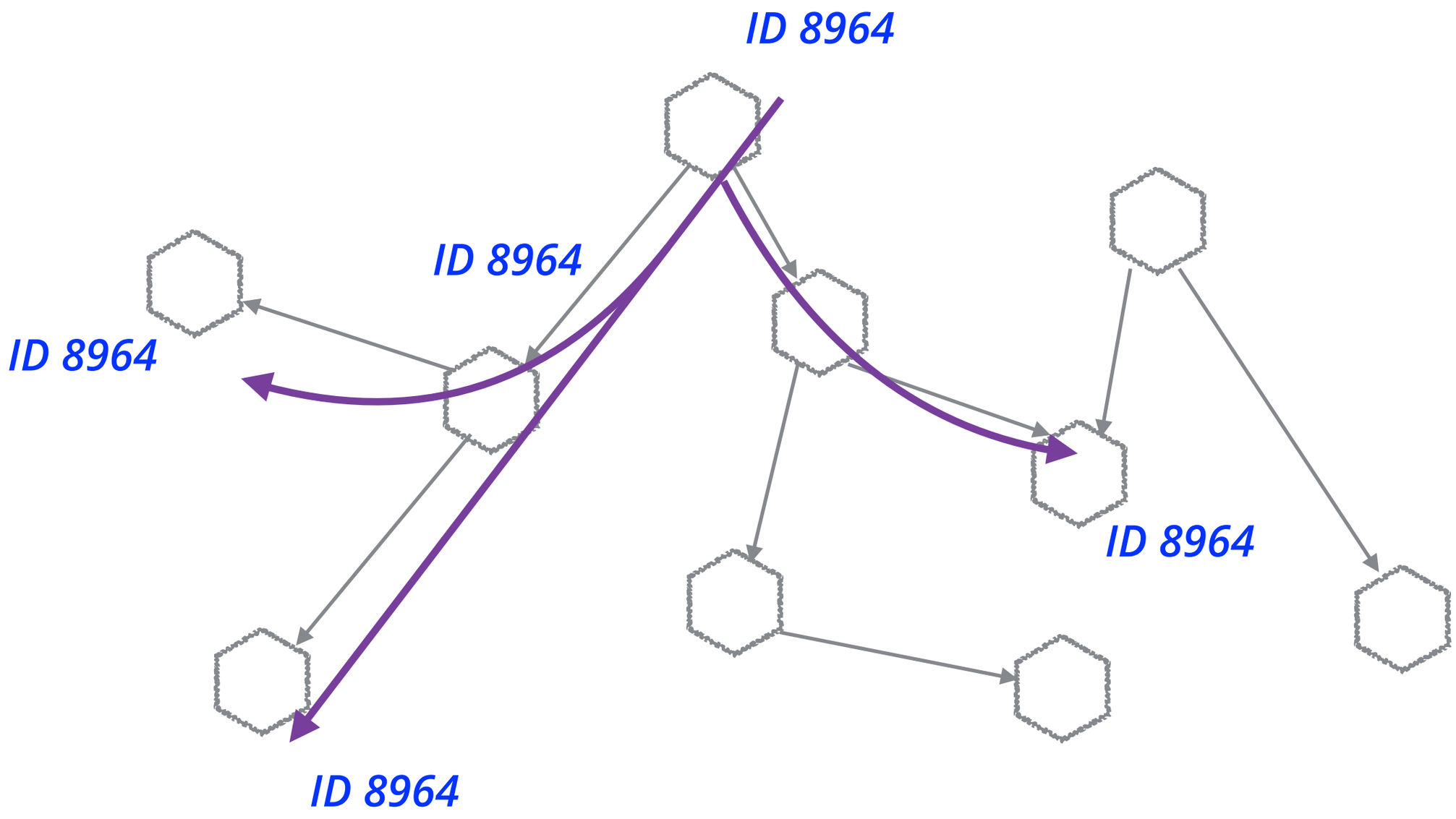
@samnewman

CORRELATION IDS



@samnewman

CORRELATION IDS

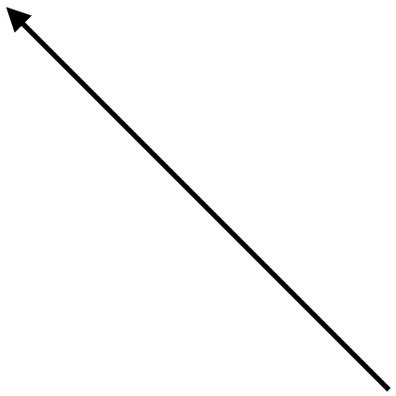


@samnewman

Principles Of Microservices

@samnewman

Modelled Around
Business Domain

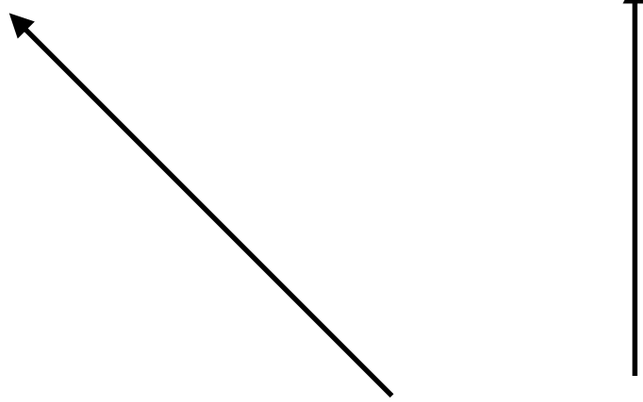


Principles Of Microservices

@samnewman

Modelled Around
Business Domain

Culture Of
Automation

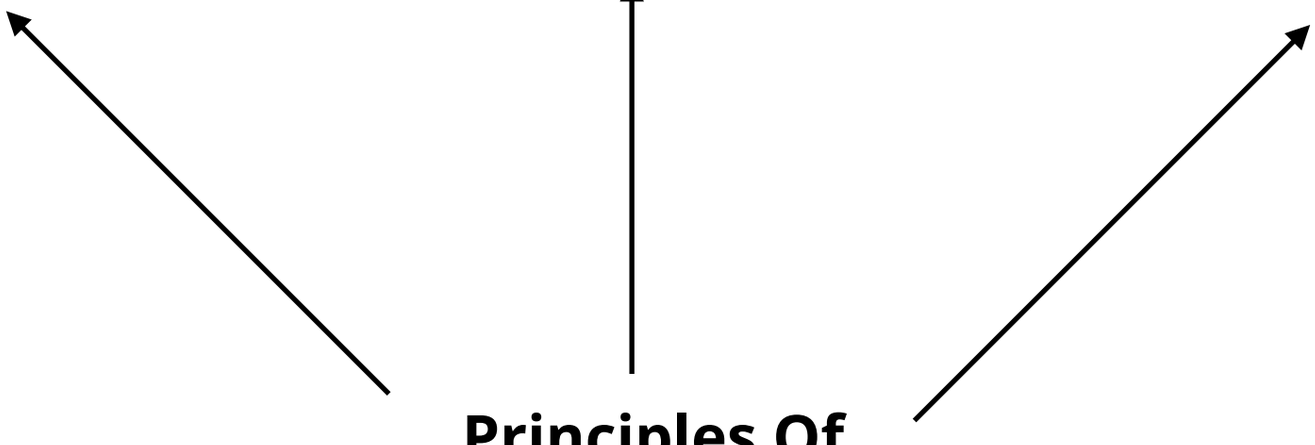


Principles Of Microservices

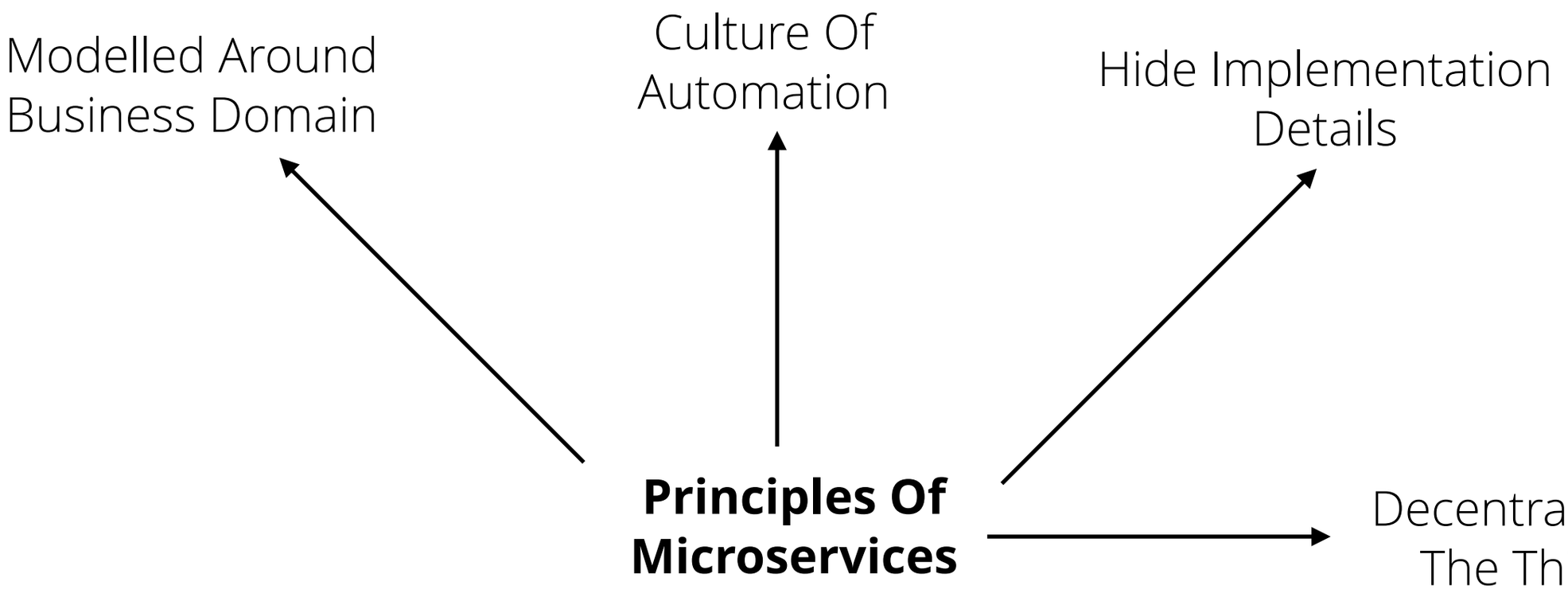
Modelled Around Business Domain

Culture Of Automation

Hide Implementation Details



Principles Of Microservices


Modelled Around
Business Domain
Culture Of
Automation
Hide Implementation
Details
Principles Of
Microservices
Decentralise All
The Things

@samnewman

Modelled Around Business Domain

Culture Of Automation

Hide Implementation Details

Principles Of Microservices

Decentralise All The Things

Deploy Independently

@samnewman

Modelled Around Business Domain

Culture Of Automation

Hide Implementation Details

Principles Of Microservices

Decentralise All The Things

Isolate Failure

Deploy Independently

@samnewman

Modelled Around Business Domain

Culture Of Automation

Hide Implementation Details

Highly Observable

Principles Of Microservices

Decentralise All The Things

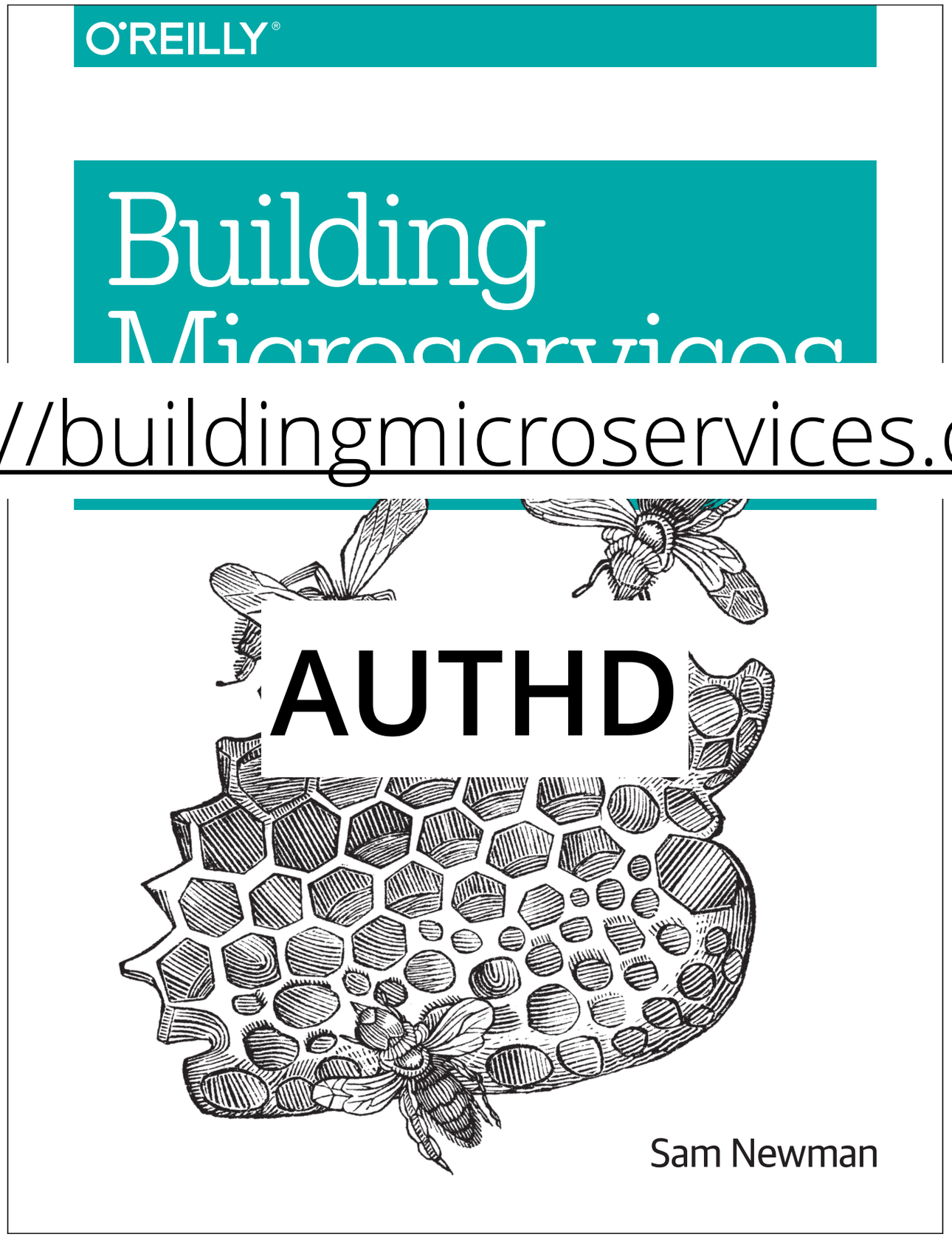
Isolate Failure

Deploy Independently

@samnewman

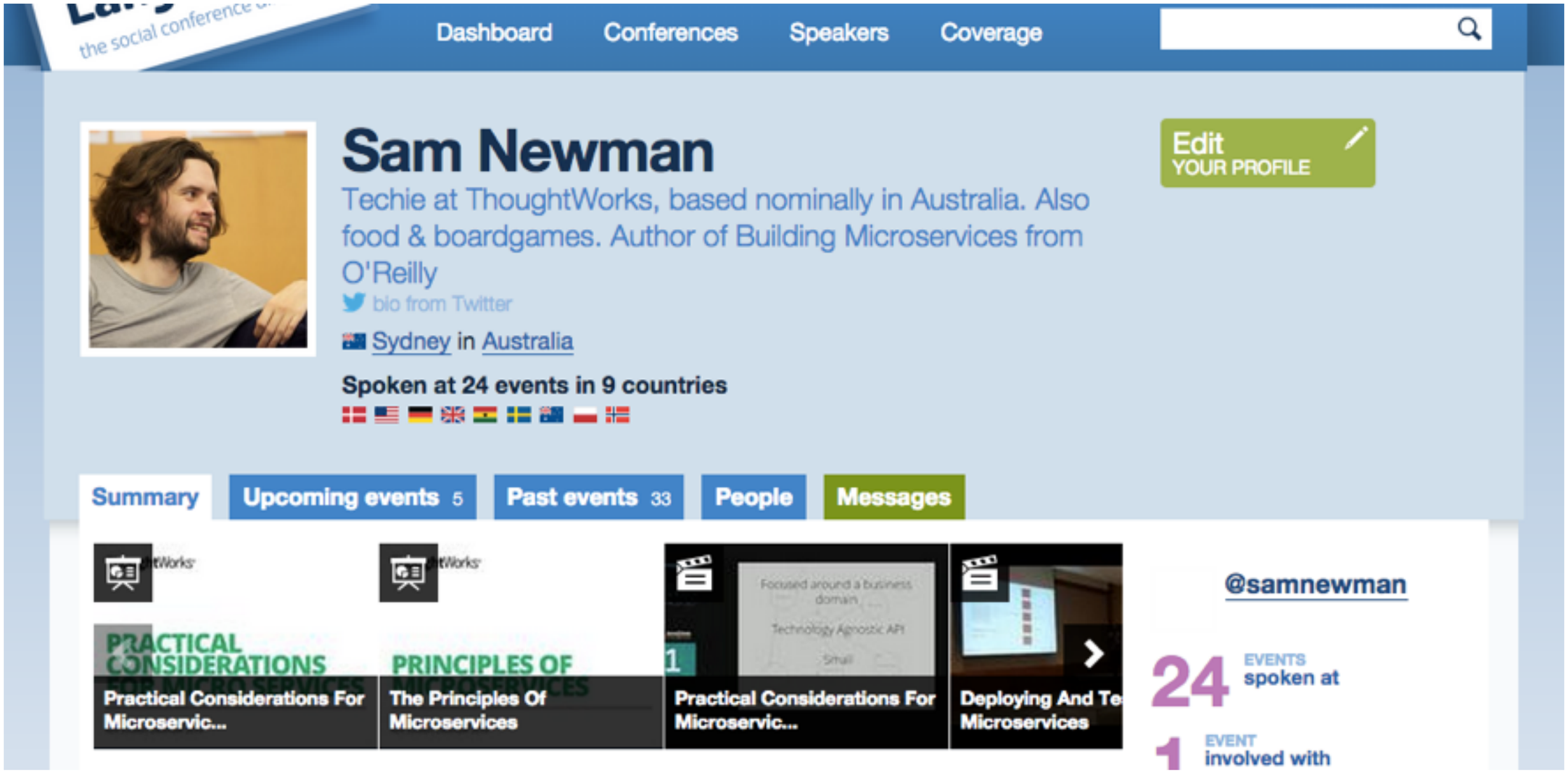

Natural
FEEDBACK MACHINE
1 SELECT TALK
2 VALUE TALK
RELEVANCE OF CONTENT
not relevant
very relevant
PRESENTATION
poor
great
OVERALL IMPRESSION
no way
definitely yes
3 SUBMIT FEEDBACK

@samnewman



http://buildingmicroservices.com/

@samnewman



http://lanyrd.com/profile/samnewman/

@samnewman

THANKS!

Sam Newman
@samnewman

ThoughtWorks®