



PRINCIPLES OF MICROSERVICES

*Sam Newman
Microxchg, Berlin 2015*

ThoughtWorks®

@samnewman

There is no hyphen
in “microservice”

@samnewman

micro-service

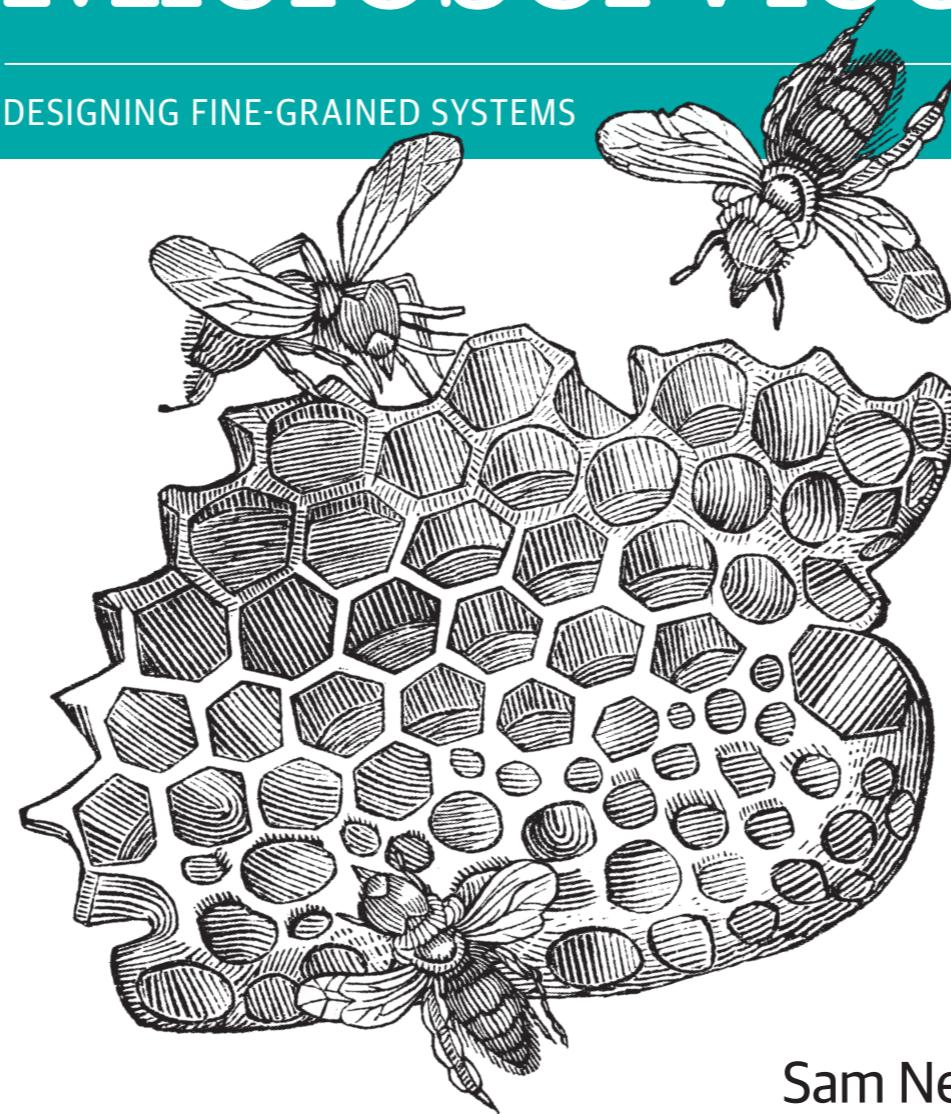
@samnewman

~~micro-service~~

O'REILLY®

Building Microservices

DESIGNING FINE-GRAINED SYSTEMS



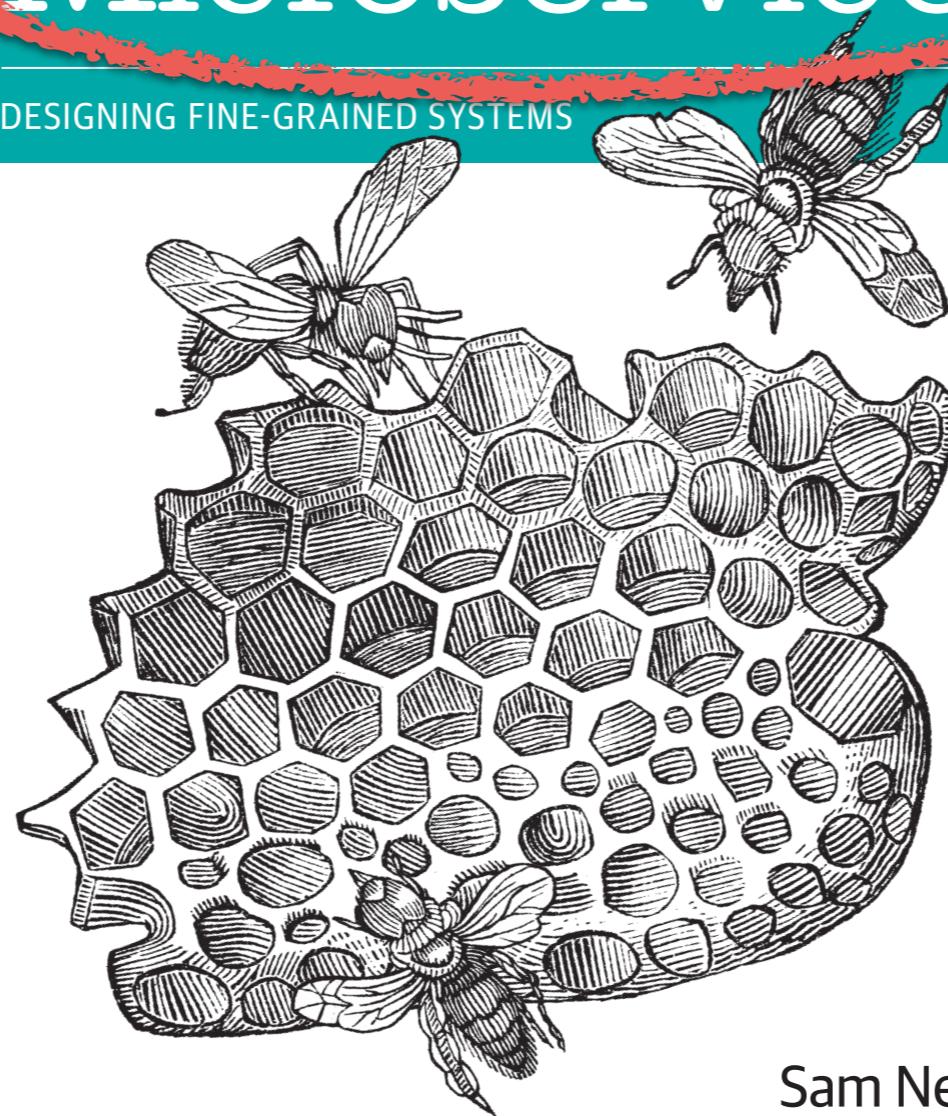
Sam Newman

@samnewman

O'REILLY®

Building Microservices

DESIGNING FINE-GRAINED SYSTEMS



Sam Newman

@samnewman

Microservices

From Wikipedia, the free encyclopedia

In computing, **microservices** is a [software architecture design pattern](#), in which complex [applications](#) are composed of small, independent services that communicate through language-agnostic [APIs](#).^[1] These [services](#) are small, highly [decoupled](#) and focus on doing a small task.^[2]

Contents [\[hide\]](#)

- [1 Details](#)
- [2 History](#)
- [3 Criticism](#)
- [4 Languages](#)
- [5 Users](#)
- [6 See other](#)
- [7 References](#)

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

Microservices

From Wikipedia, the free encyclopedia

In computing, **microservices** is a [software architecture design pattern](#), in which complex [applications](#) are composed of small, independent services that communicate through [language-agnostic APIs](#).^[1] These [services](#) are small, highly [decoupled](#) and focus on doing a small task.^[2]

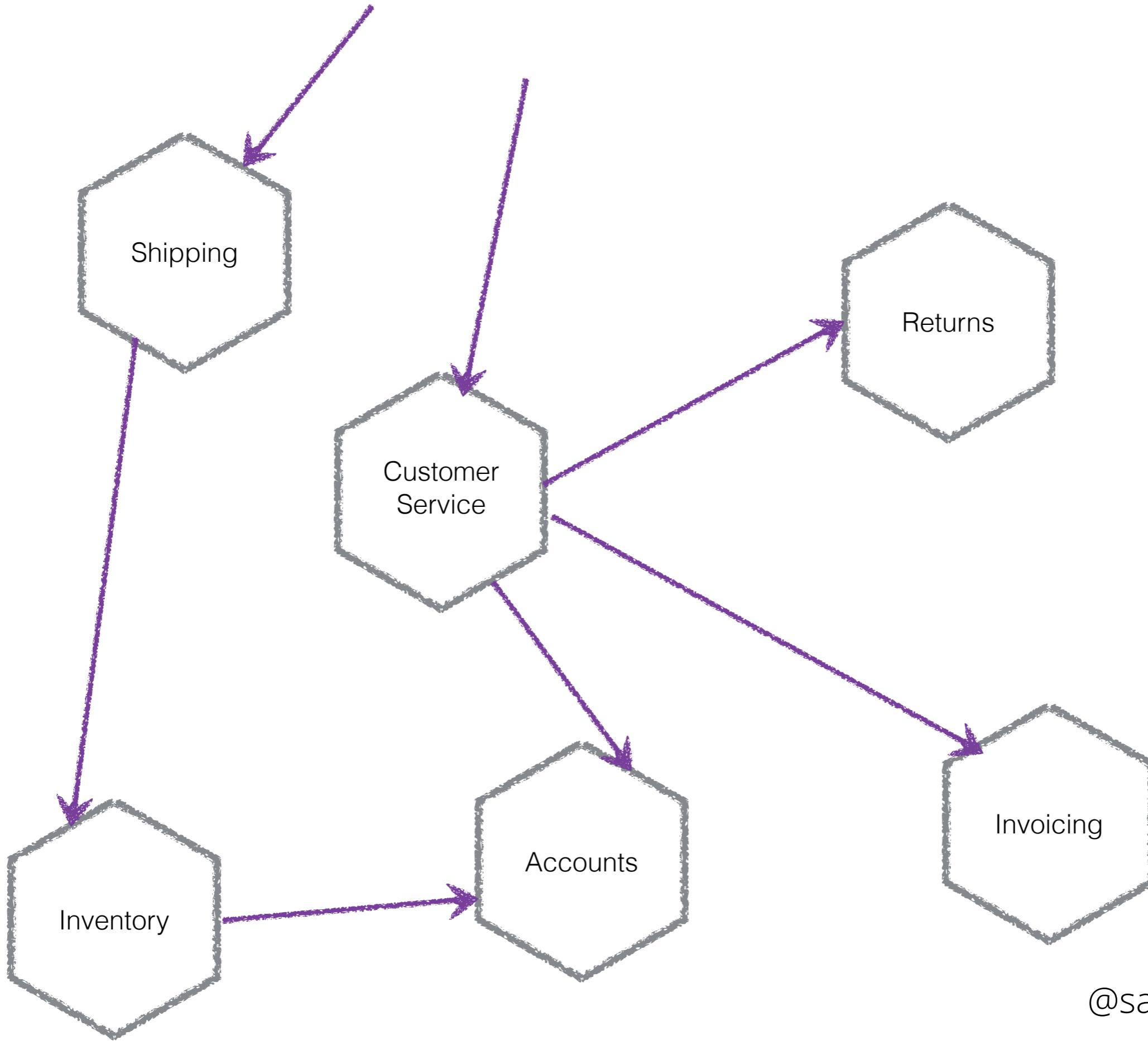
Contents [\[hide\]](#)

- [1 Details](#)
- [2 History](#)
- [3 Criticism](#)
- [4 Languages](#)
- [5 Users](#)
- [6 See other](#)
- [7 References](#)

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



@samnewman

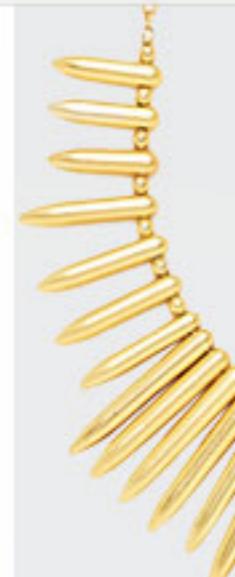
Small ***Autonomous*** services
that ***work together***

The diagram illustrates a network of six small, autonomous services represented as hexagons. The services are: Shipping, Returns, Invoicing, Accounts, Inventory, and a central Customer Service. Arrows show the flow of data between them. Shipping has a bidirectional arrow with Customer Service. Returns has a unidirectional arrow pointing to Customer Service. Invoicing has a bidirectional arrow with Customer Service. Accounts has a bidirectional arrow with Customer Service. Inventory has a unidirectional arrow pointing to Accounts. There is also a unidirectional arrow from Accounts to Invoicing.

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



Shipping available to United Kingdom Up to 60%



10 Perfect

Skip the guesswork with our exciting surprises for all on your list.

[Shop this Sale](#)

Our purpose is to empower people by making property simple, efficient and stress-free.

ASX Share Price (REA)

[Home](#)[About REA Group](#)

Shipping available to United Kingdom Up to 60%



10 Perfect

Skip the guesswork with our easy surprises for all on your list.



Our purpose is to empower people by making property simple, efficient and

Home

About REA Group

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

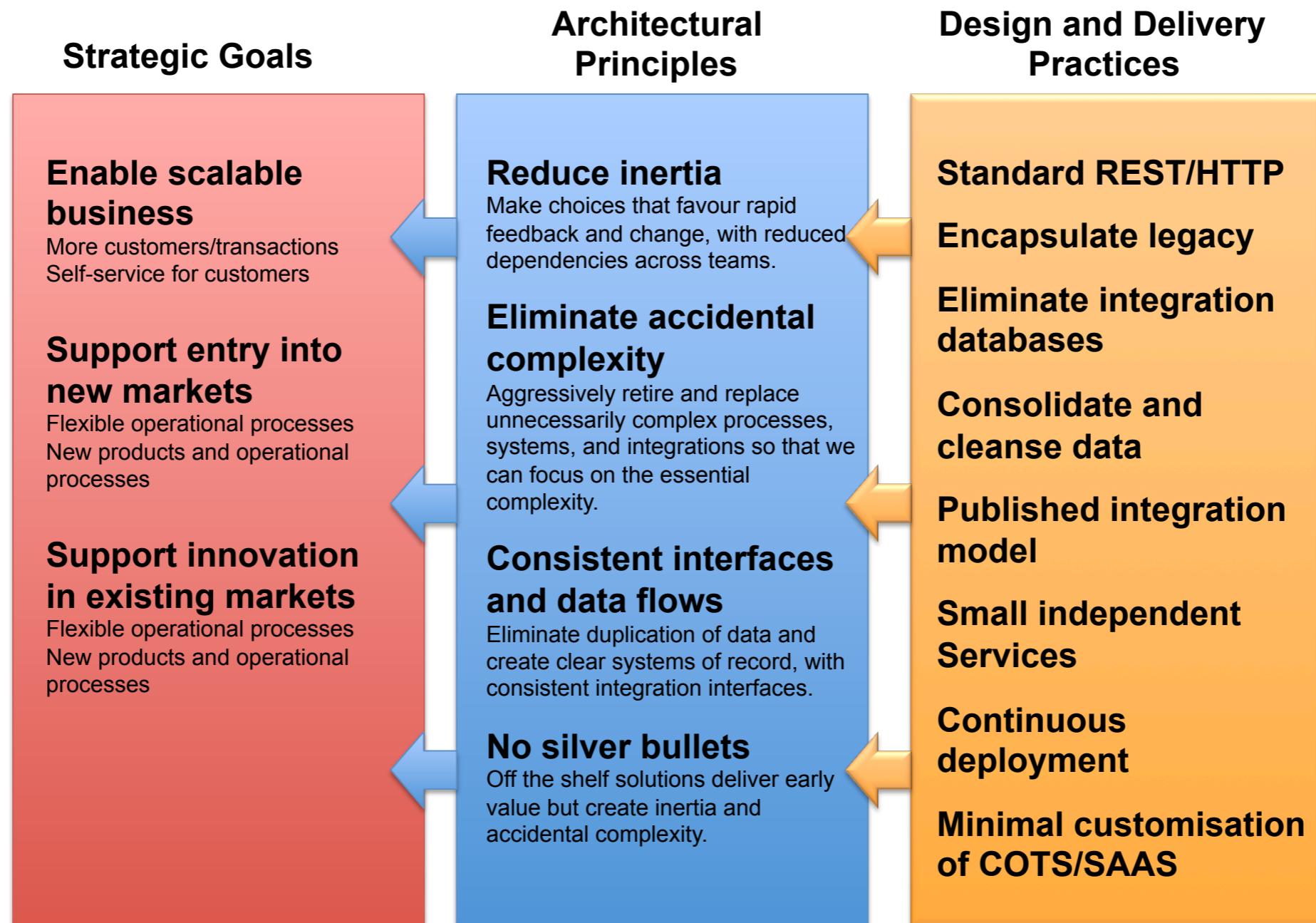
Keep development, staging, and production as similar as possible

XI. Logs

Treat logs as event streams

XII. Admin processes

Run admin/management tasks as one-off processes

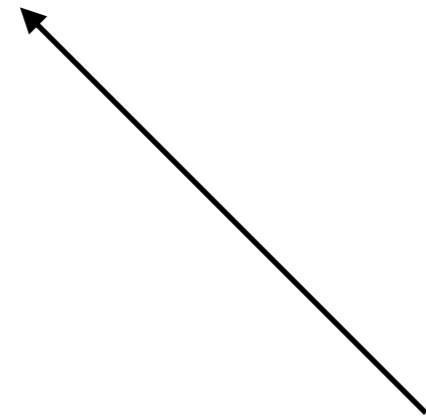


Small ***Autonomous*** services
that ***work together***

Principles Of Microservices

@samnewman

Modelled Around
Business Domain

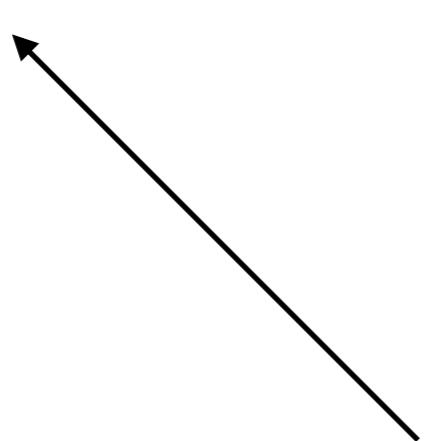


Principles Of Microservices

Modelled Around
Business Domain

Culture Of
Automation

**Principles Of
Microservices**

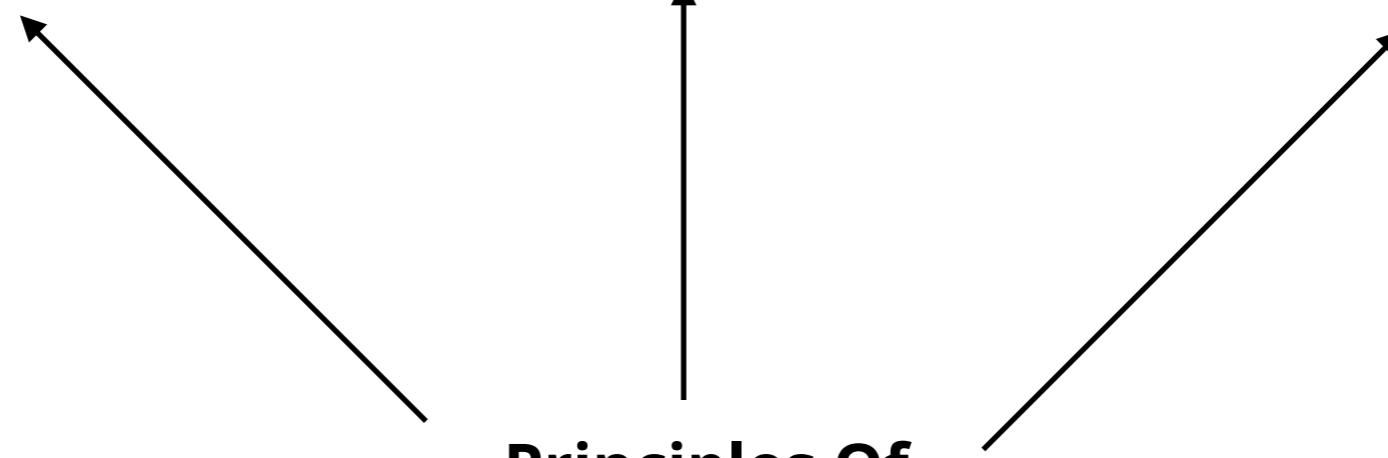


Modelled Around
Business Domain

Culture Of
Automation

Hide Implementation
Details

Principles Of Microservices



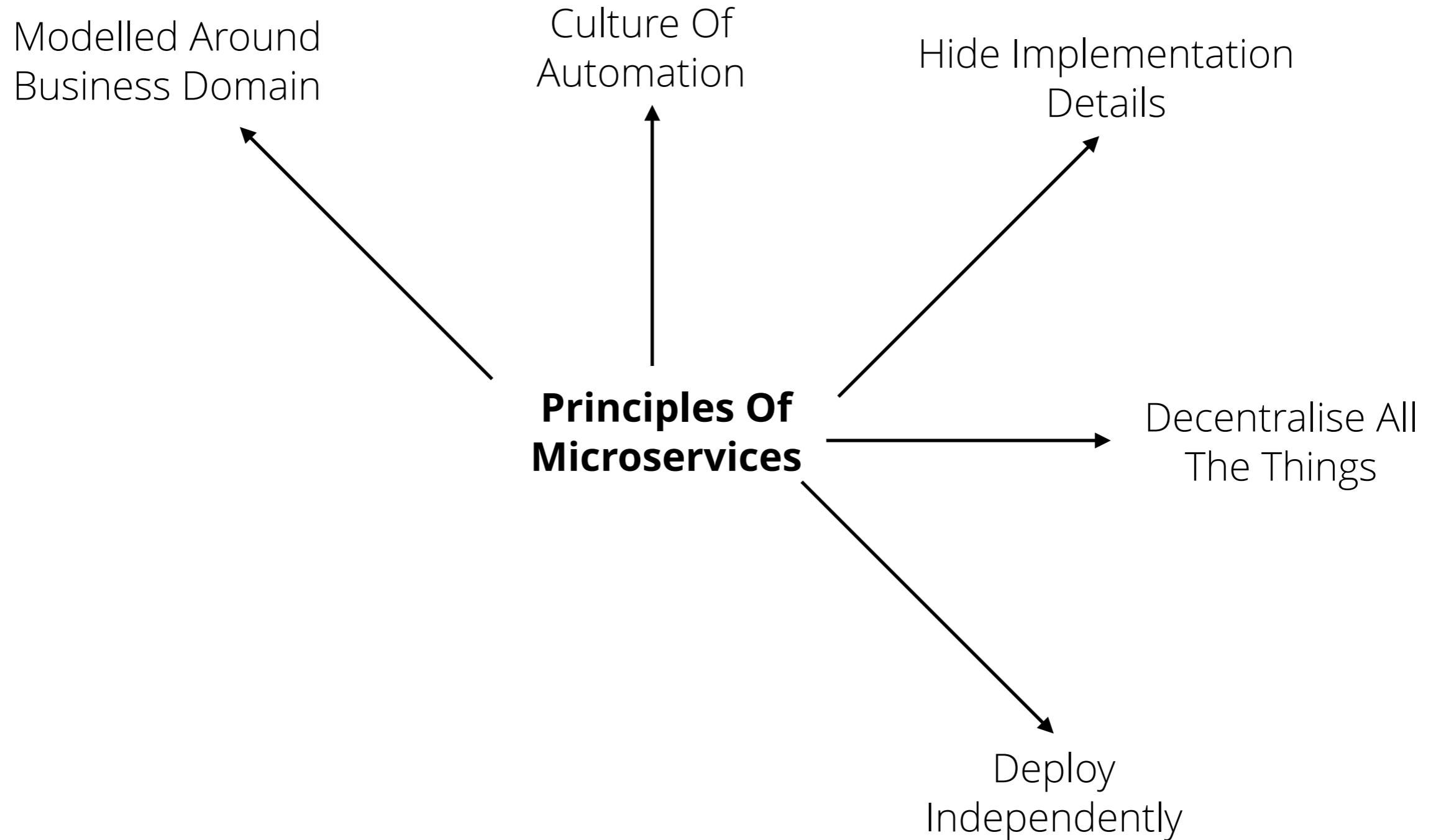
Modelled Around
Business Domain

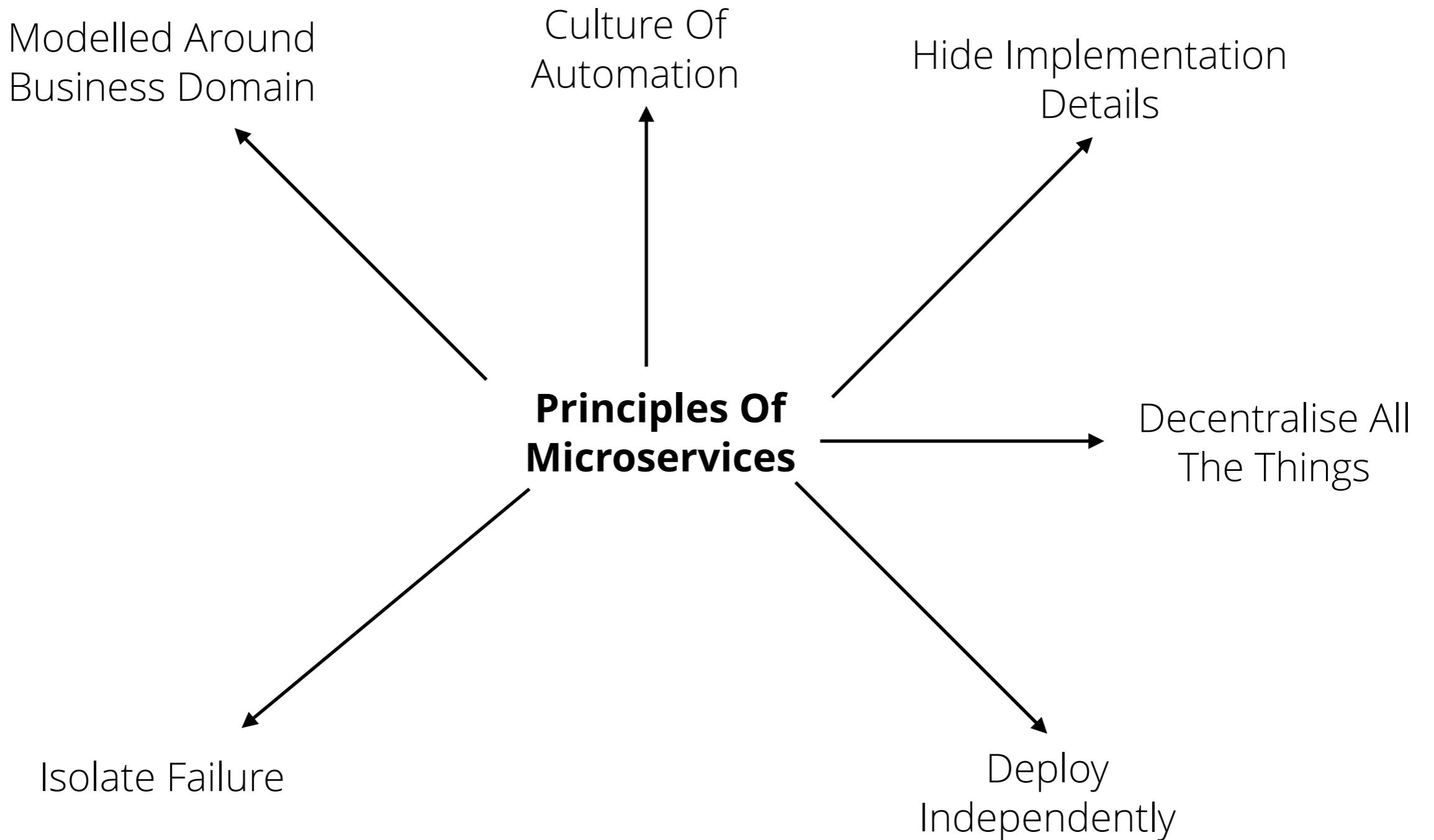
Culture Of
Automation

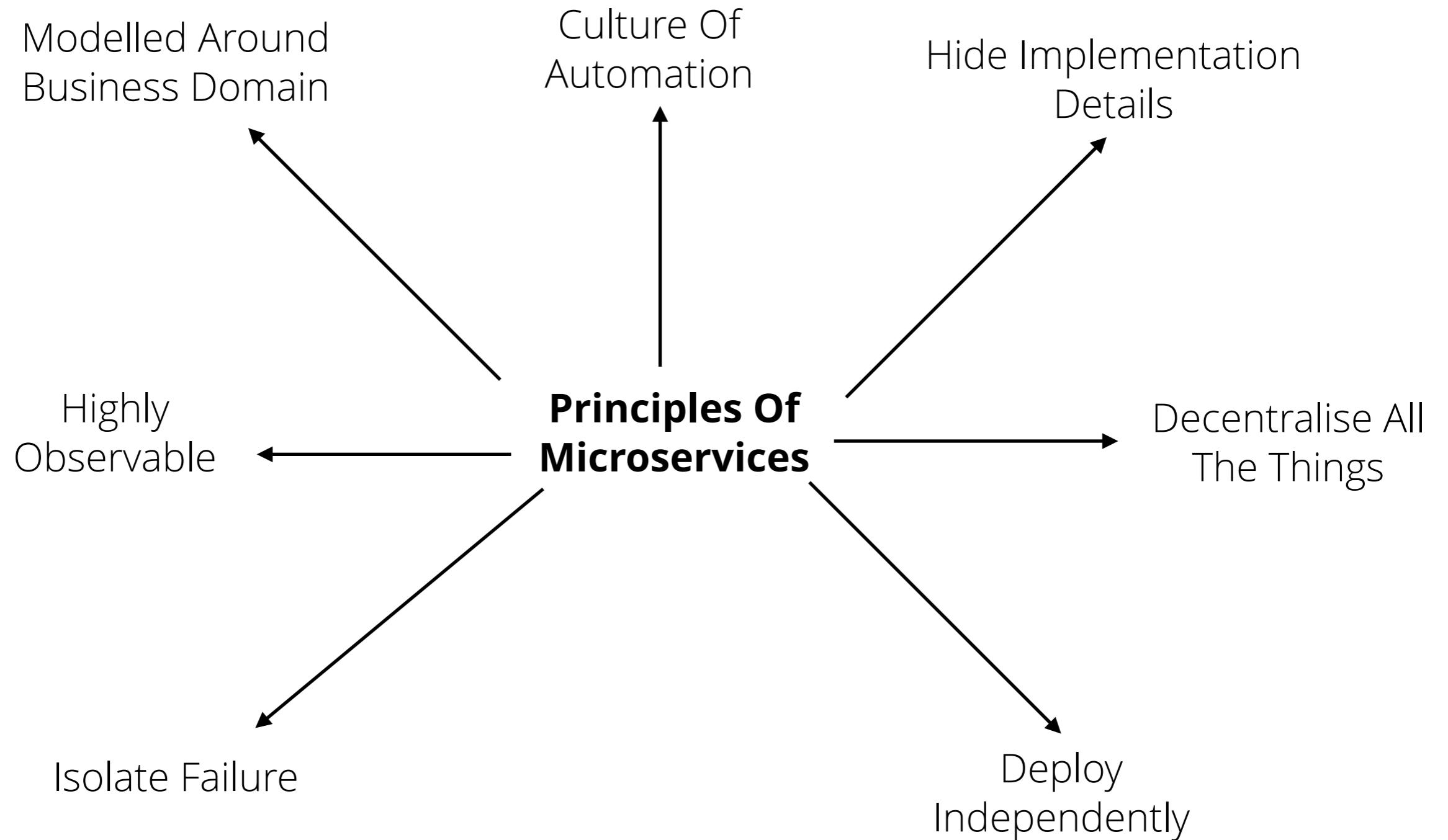
Hide Implementation
Details

Principles Of Microservices

Decentralise All
The Things







Modelled Around
Business Domain

Culture Of
Automation

Hide Implementation
Details

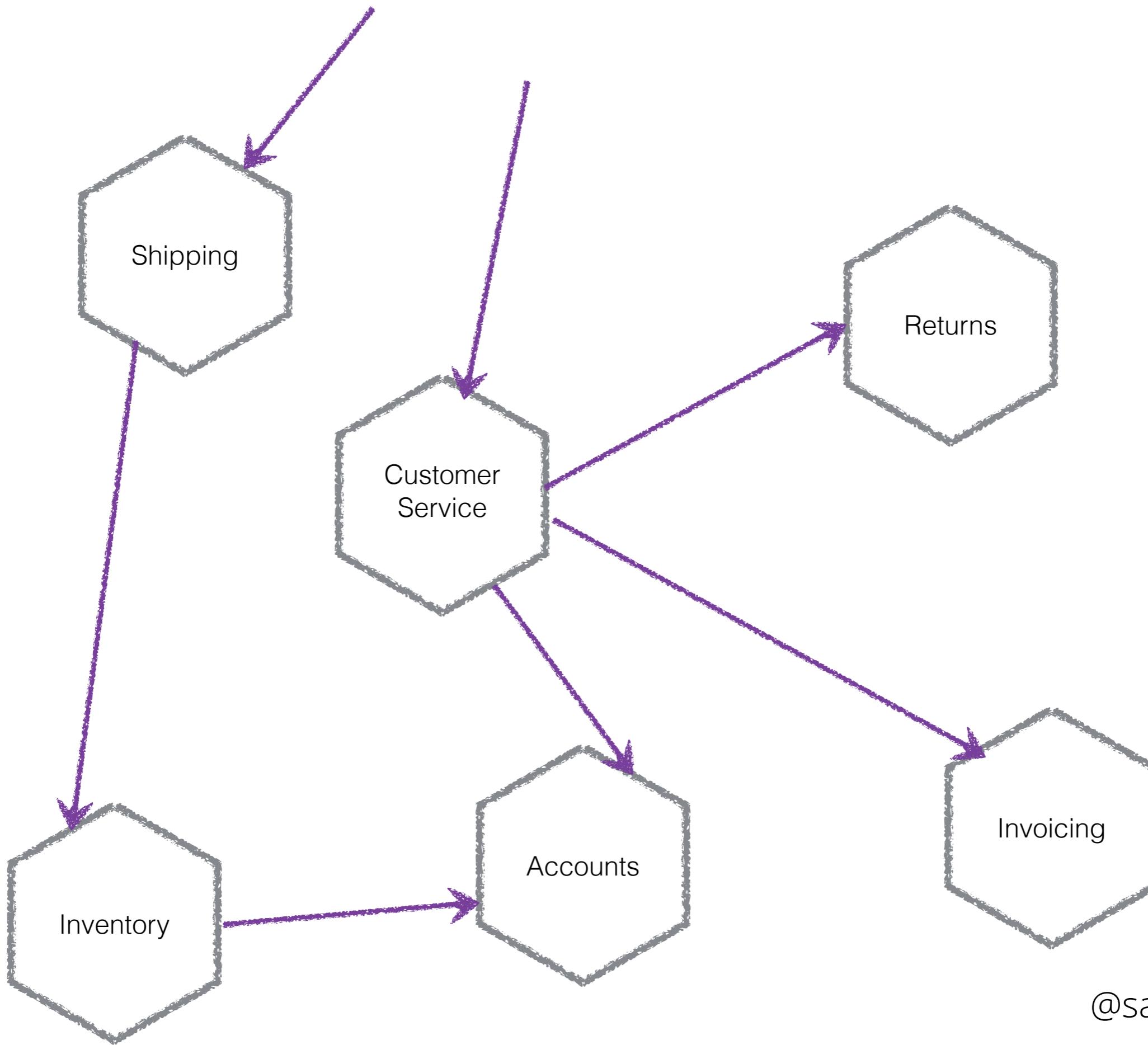
Highly
Observable

Principles Of Microservices

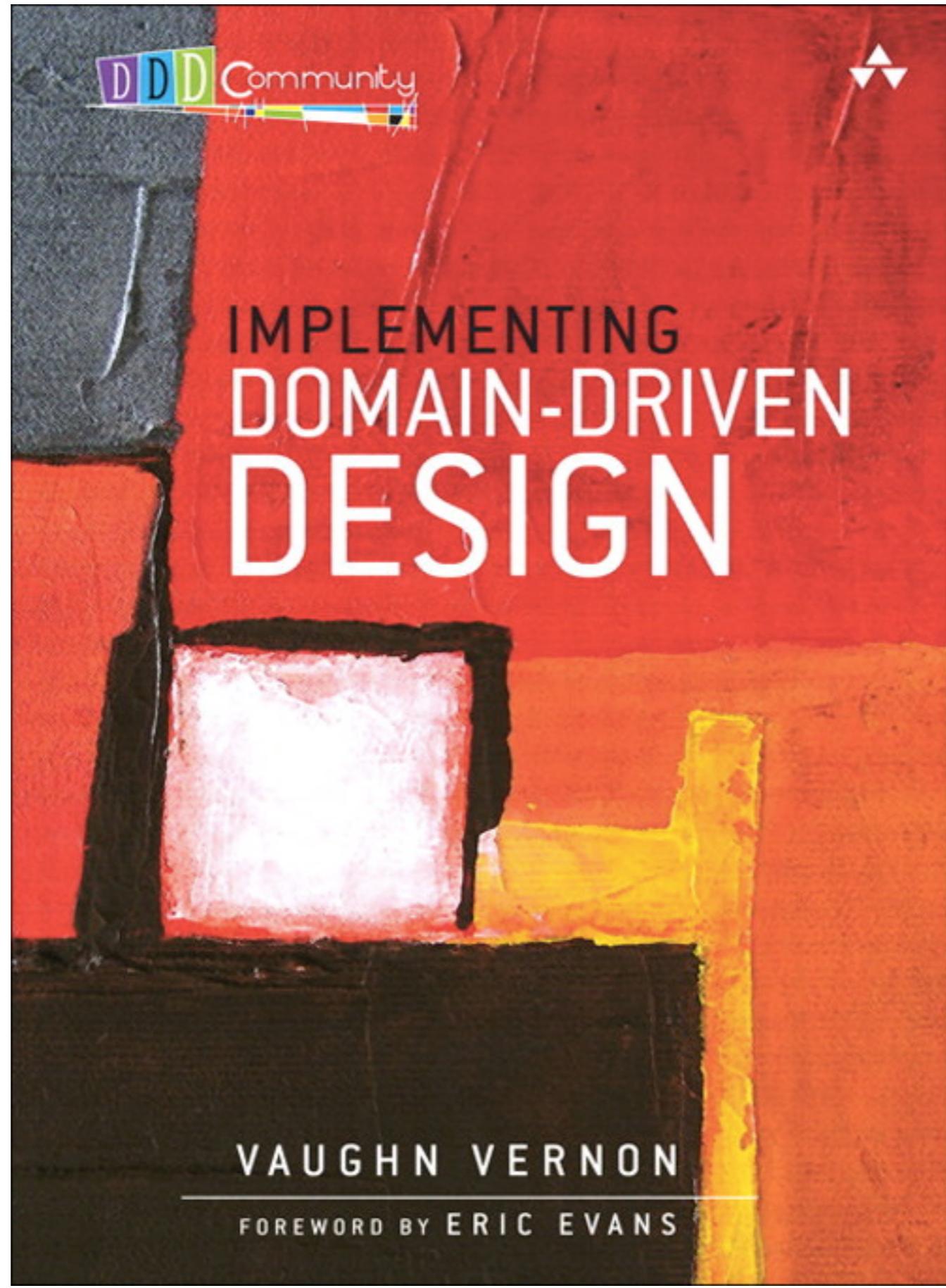
Decentralise All
The Things

Isolate Failure

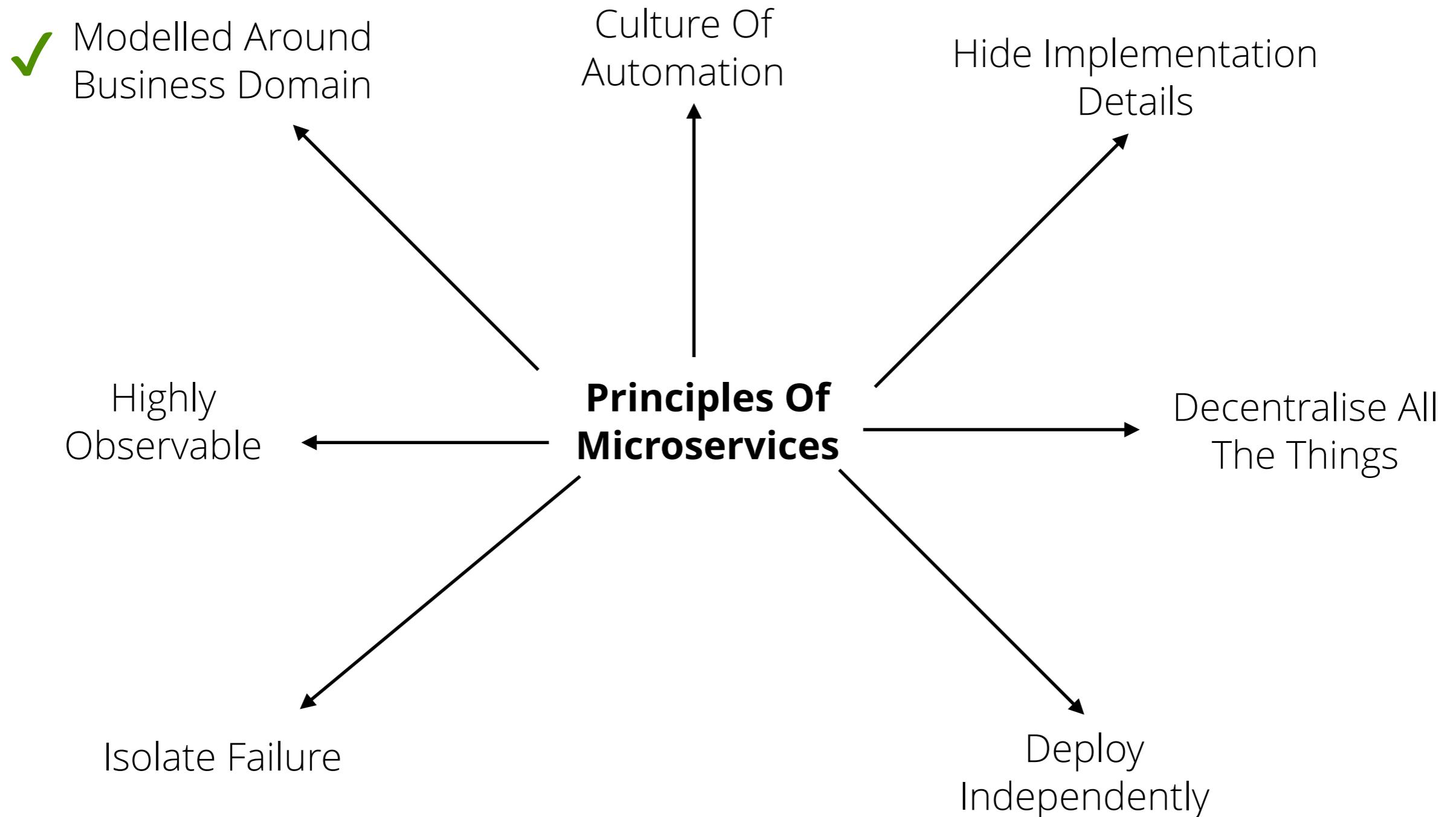
Deploy
Independently



@samnewman



@samnewman





Modelled Around
Business Domain

Culture Of Automation

Principles Of Microservices

Highly
Observable

Isolate Failure

Hide Implementation
Details

Decentralise All
The Things

Deploy
Independently

@samnewman

2 Microservices



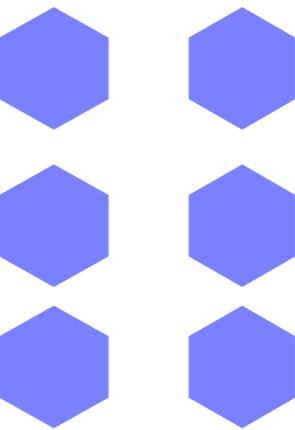
3 Months

2 Microservices

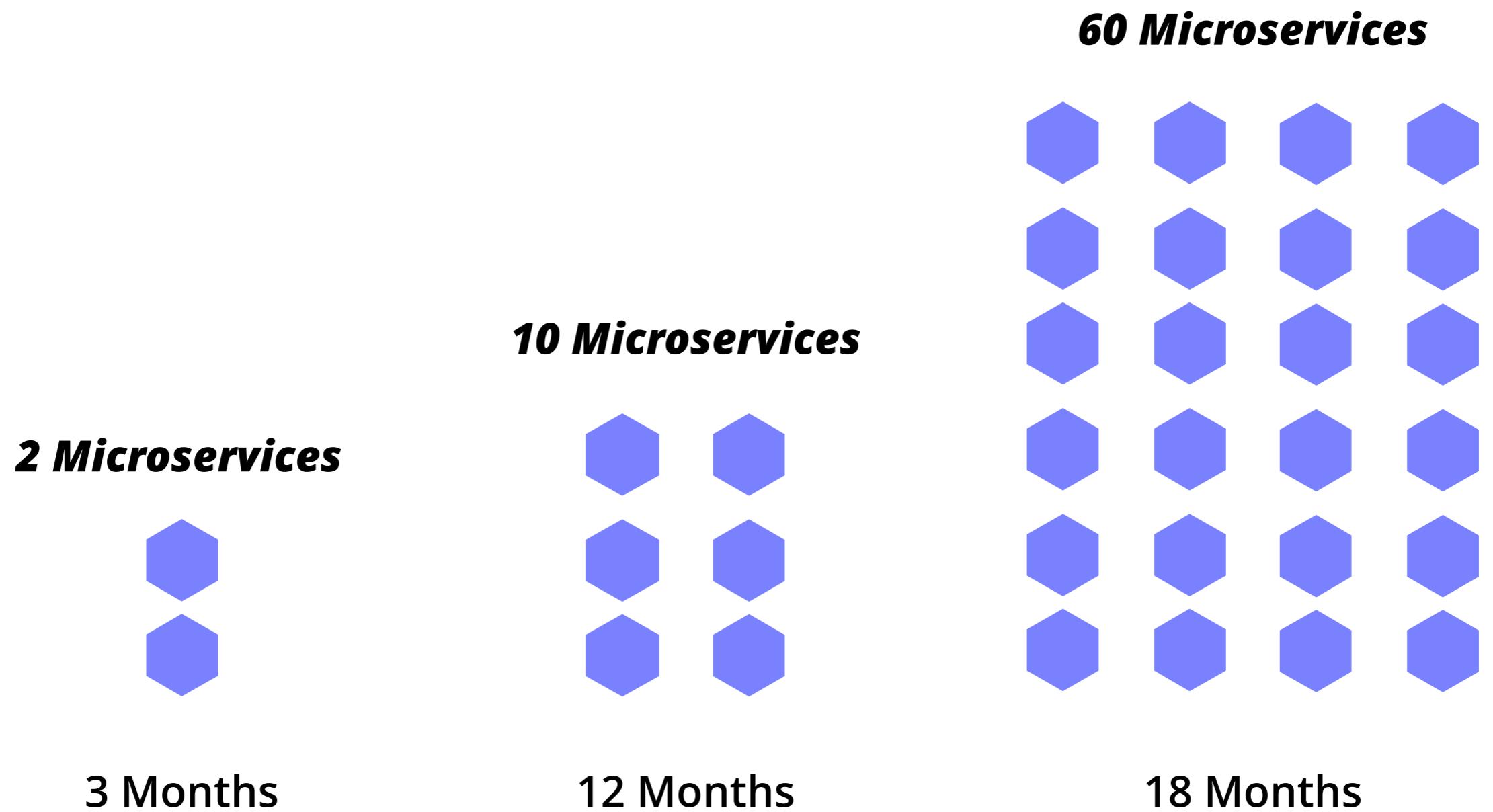


3 Months

10 Microservices



12 Months



Infrastructure Automation

@samnewman

Infrastructure Automation

Automated Testing

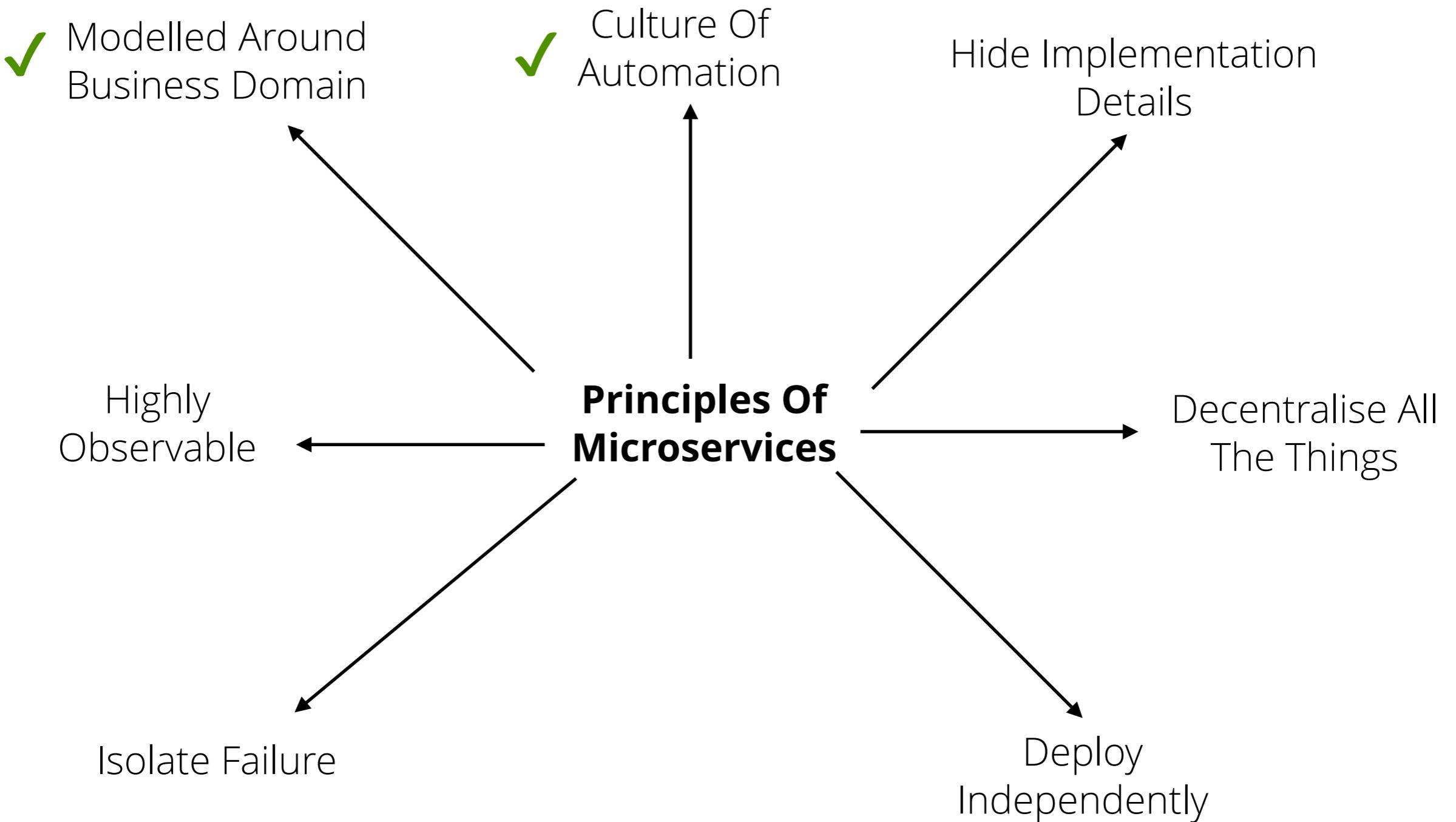
@samnewman

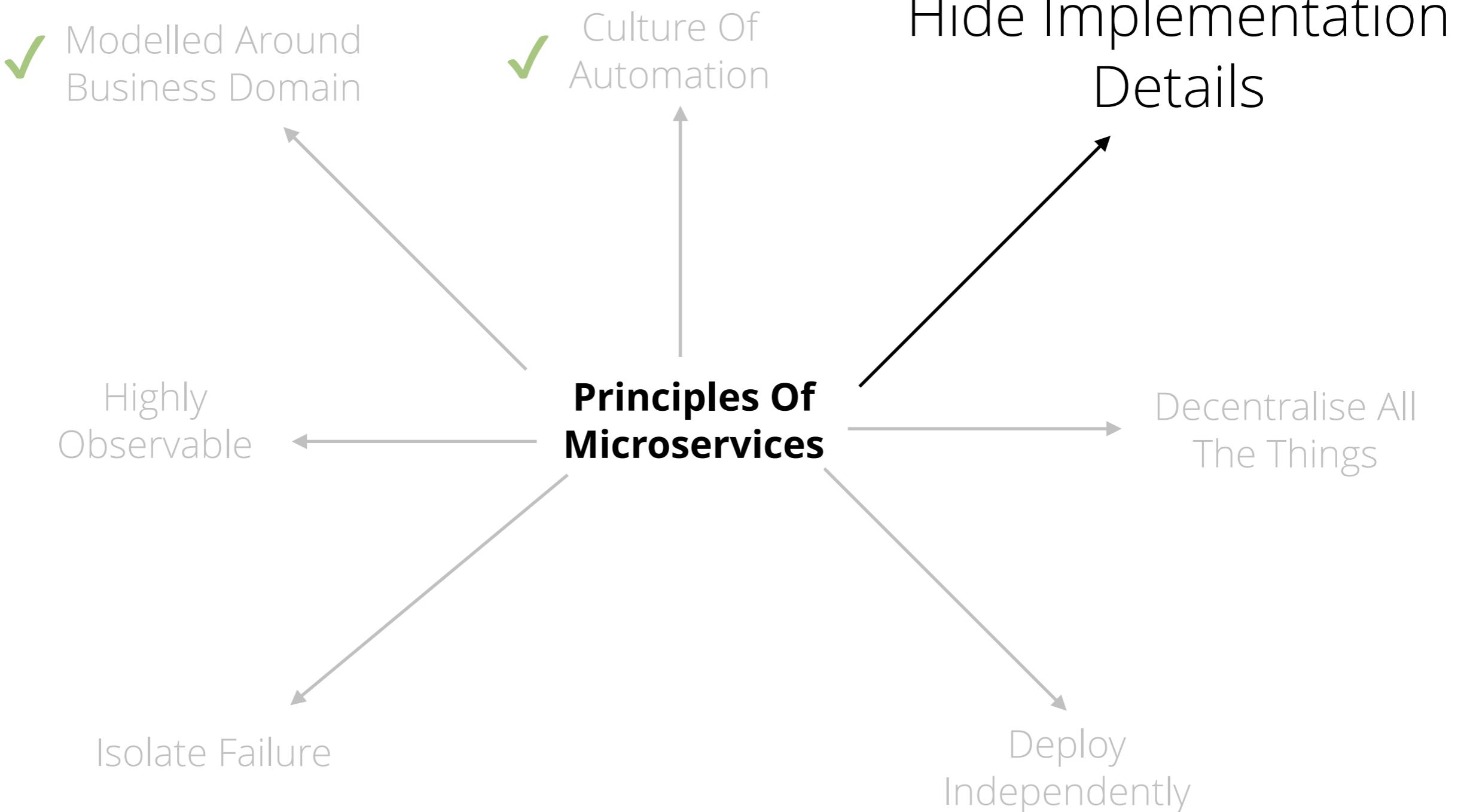
Infrastructure Automation

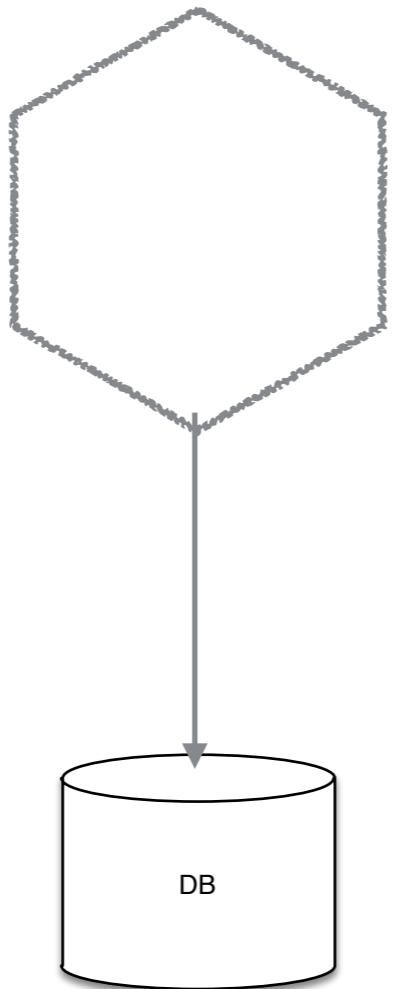
Automated Testing

Continuous Delivery

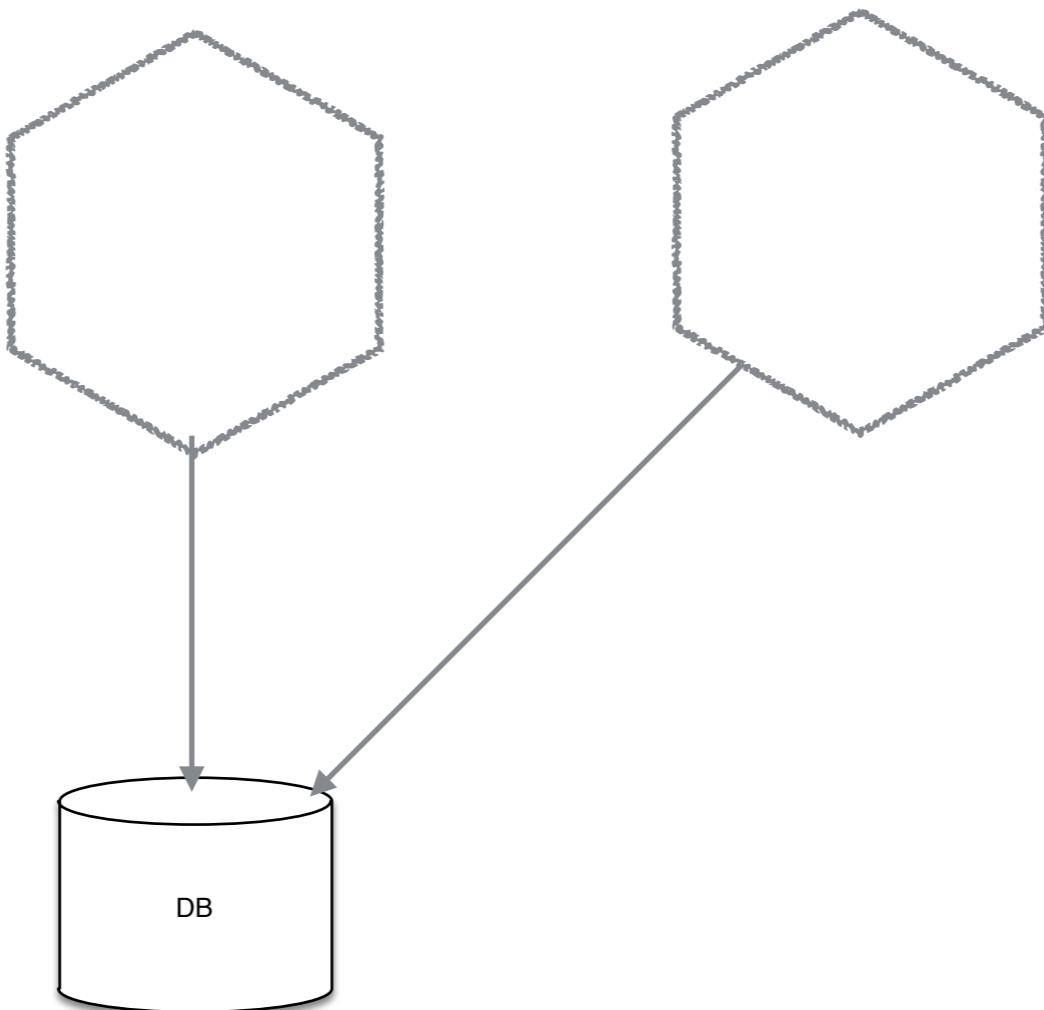
@samnewman



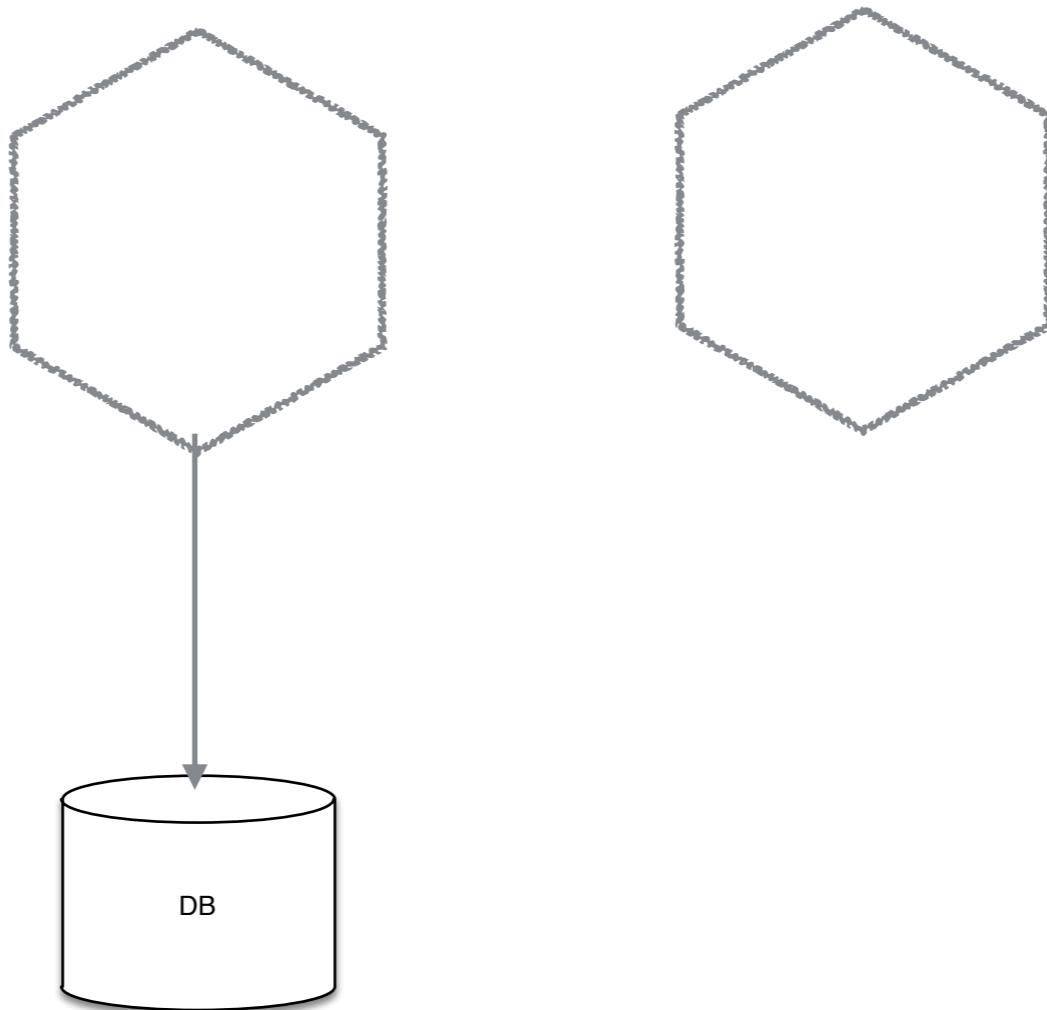




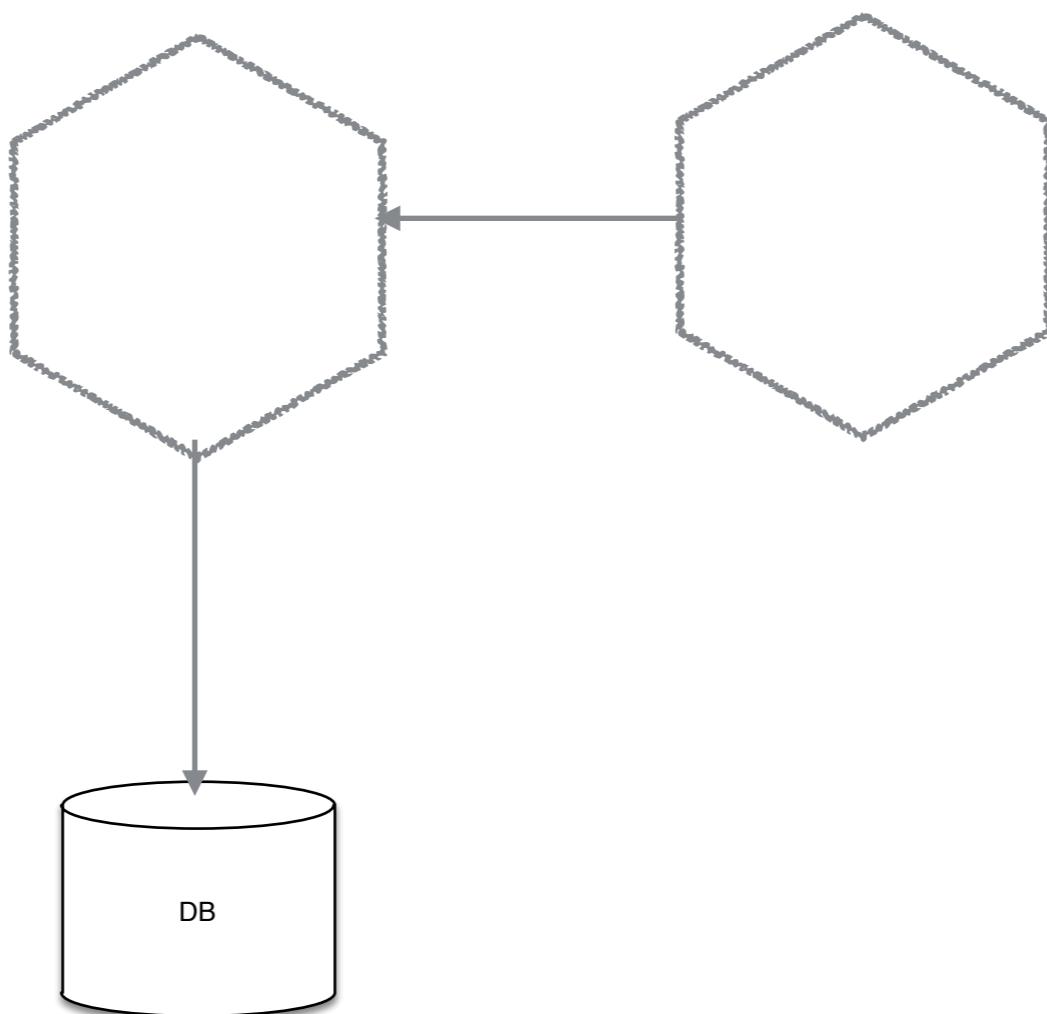
@samnewman



@samnewman

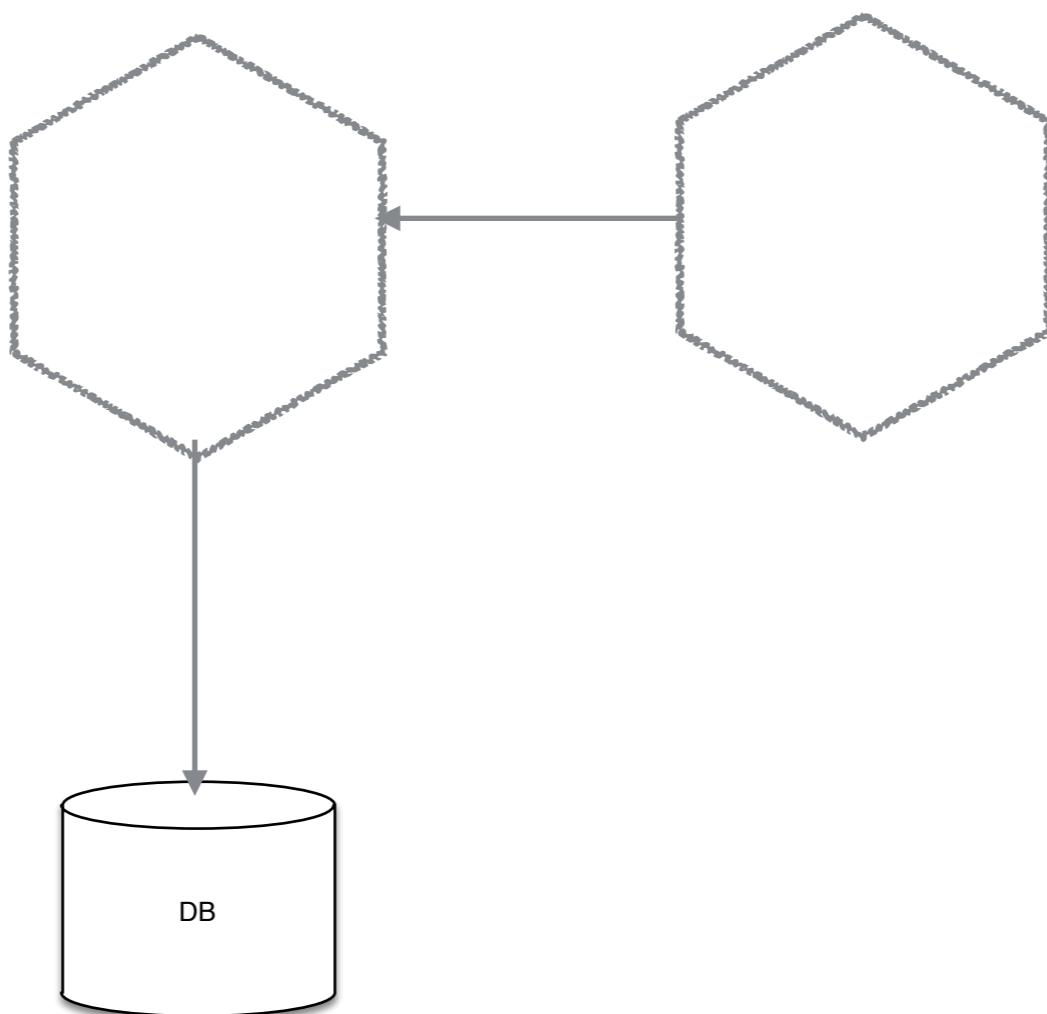


@samnewman

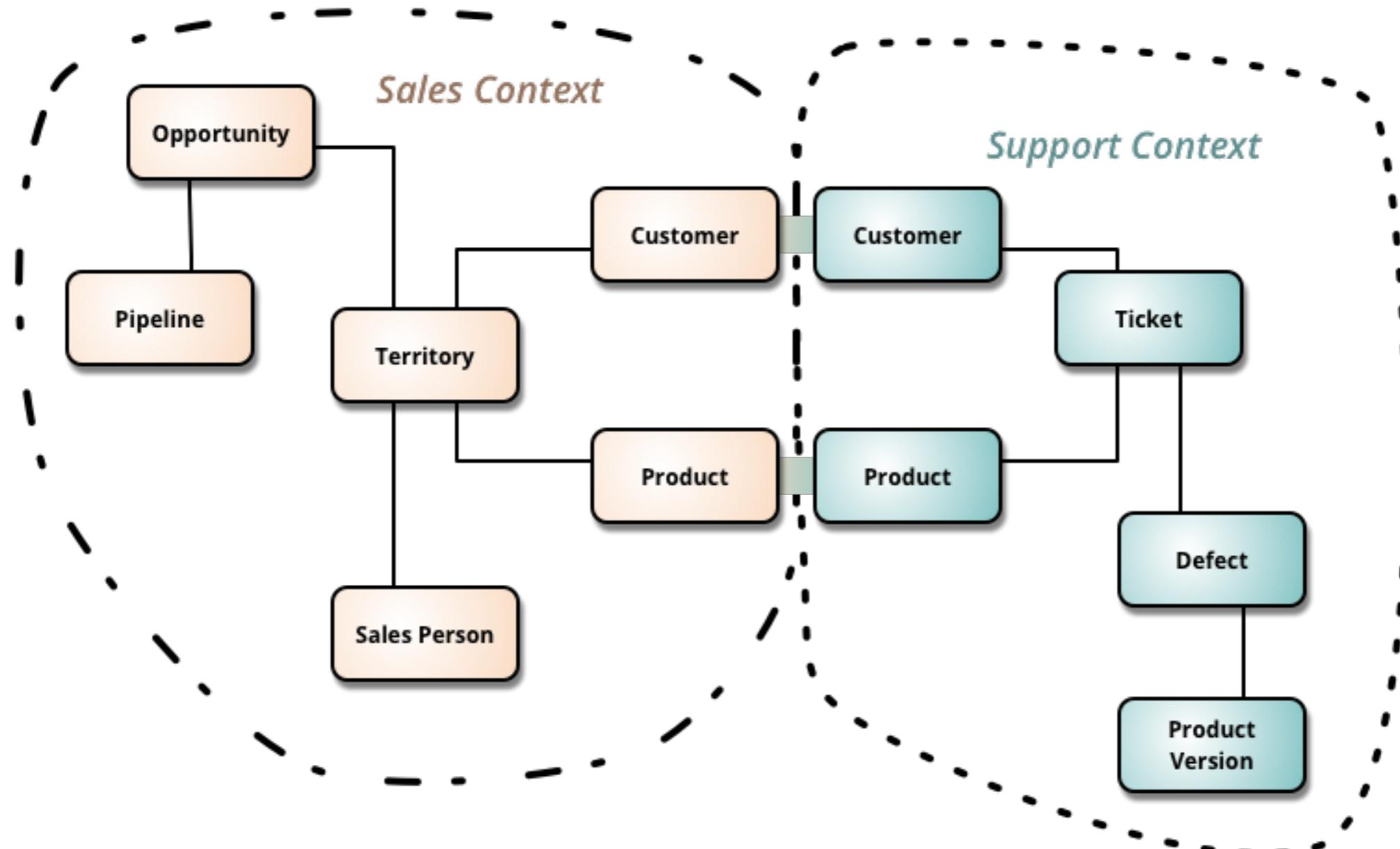


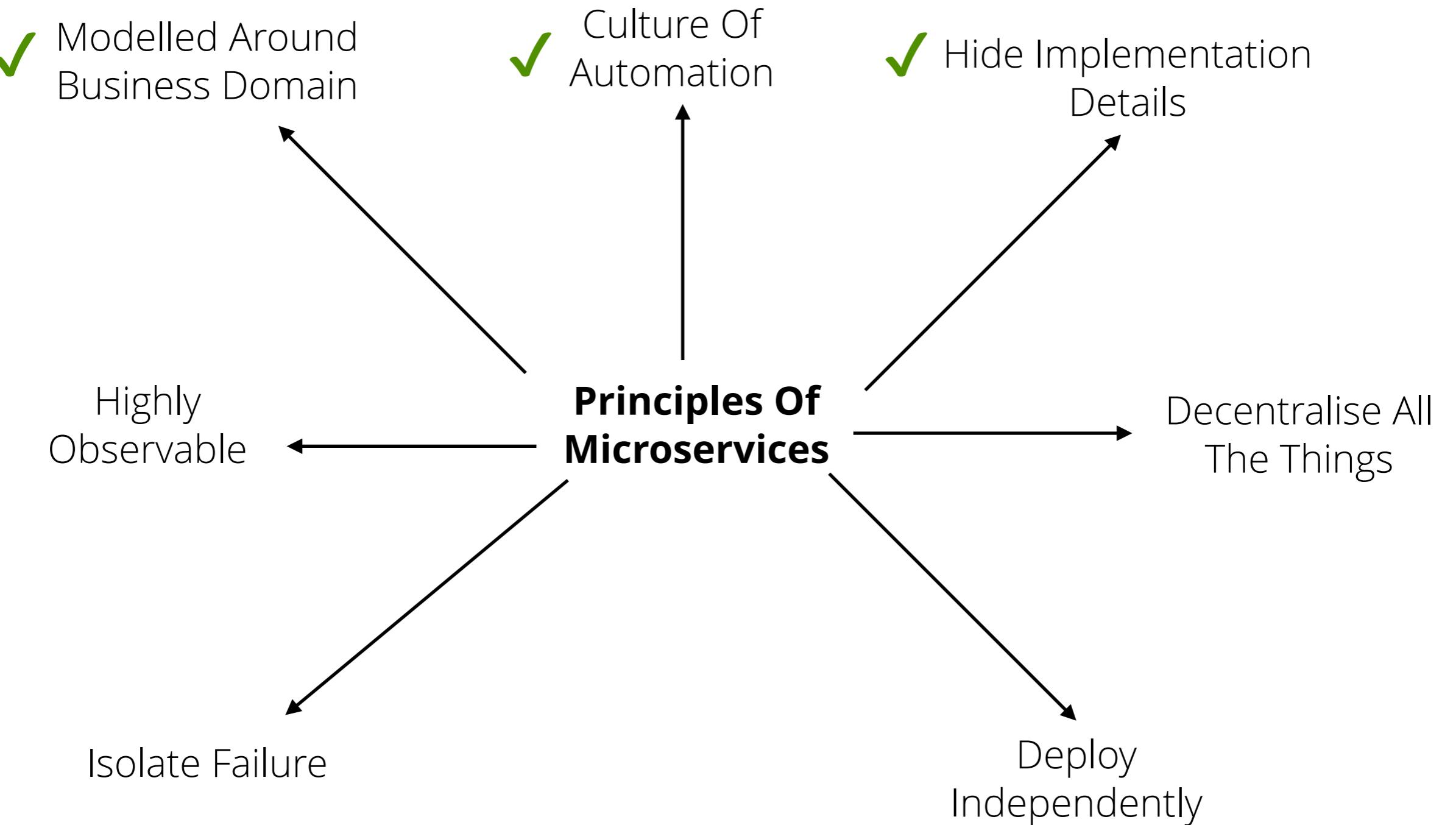
@samnewman

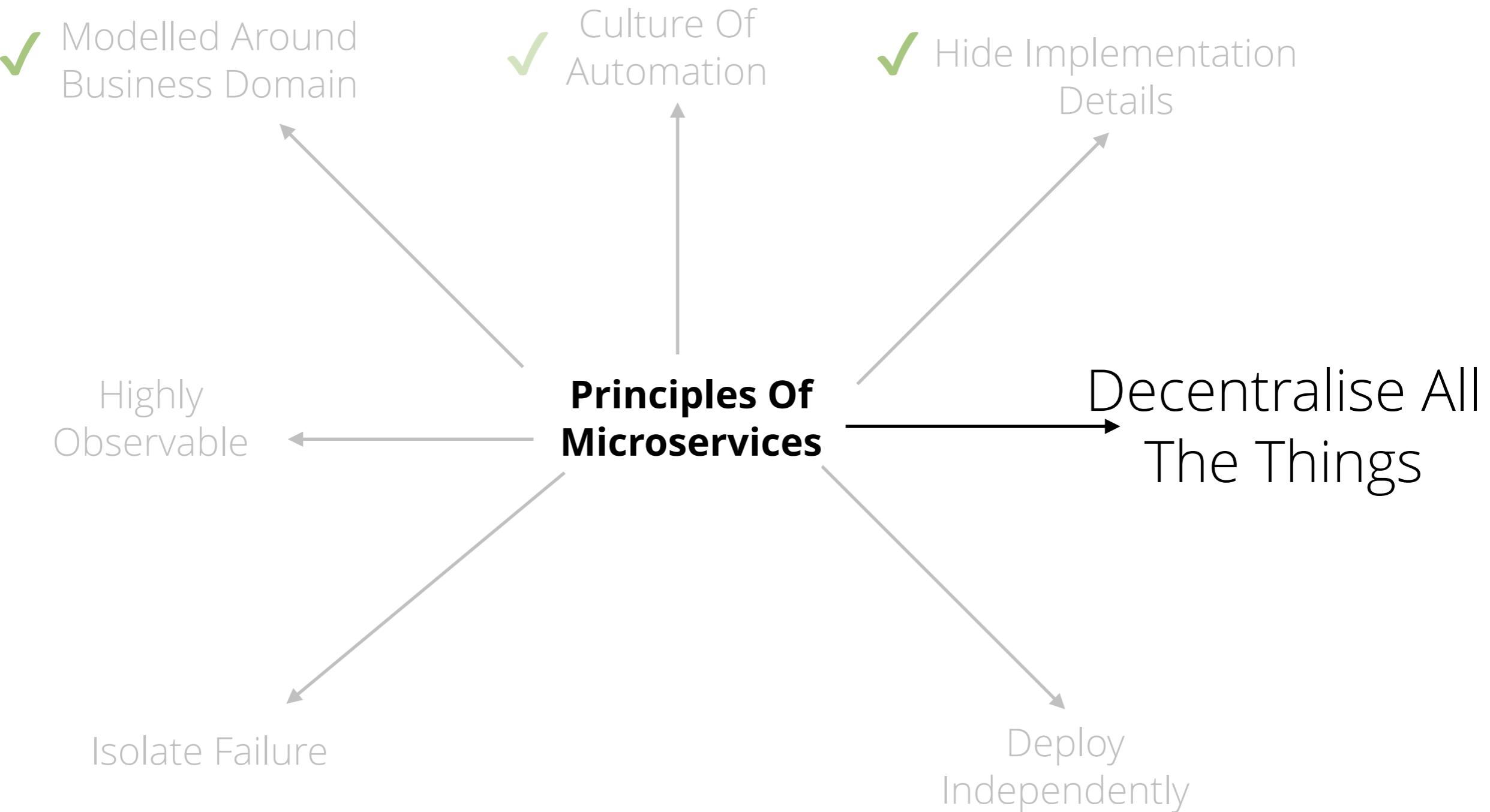
HIDE YOUR DATABASE



@samnewman







DECENTRALIZE



ALL THE THINGS

memegenerator.net

@samnewman

What is autonomy?

@samnewman

What is autonomy?

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

What is autonomy?

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

A close-up portrait of Jake Sully from the movie Avatar. He has blue alien skin and brown hair. His face is marked with blue paint in a stylized pattern. He is looking slightly upwards and to his right with a neutral expression.

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

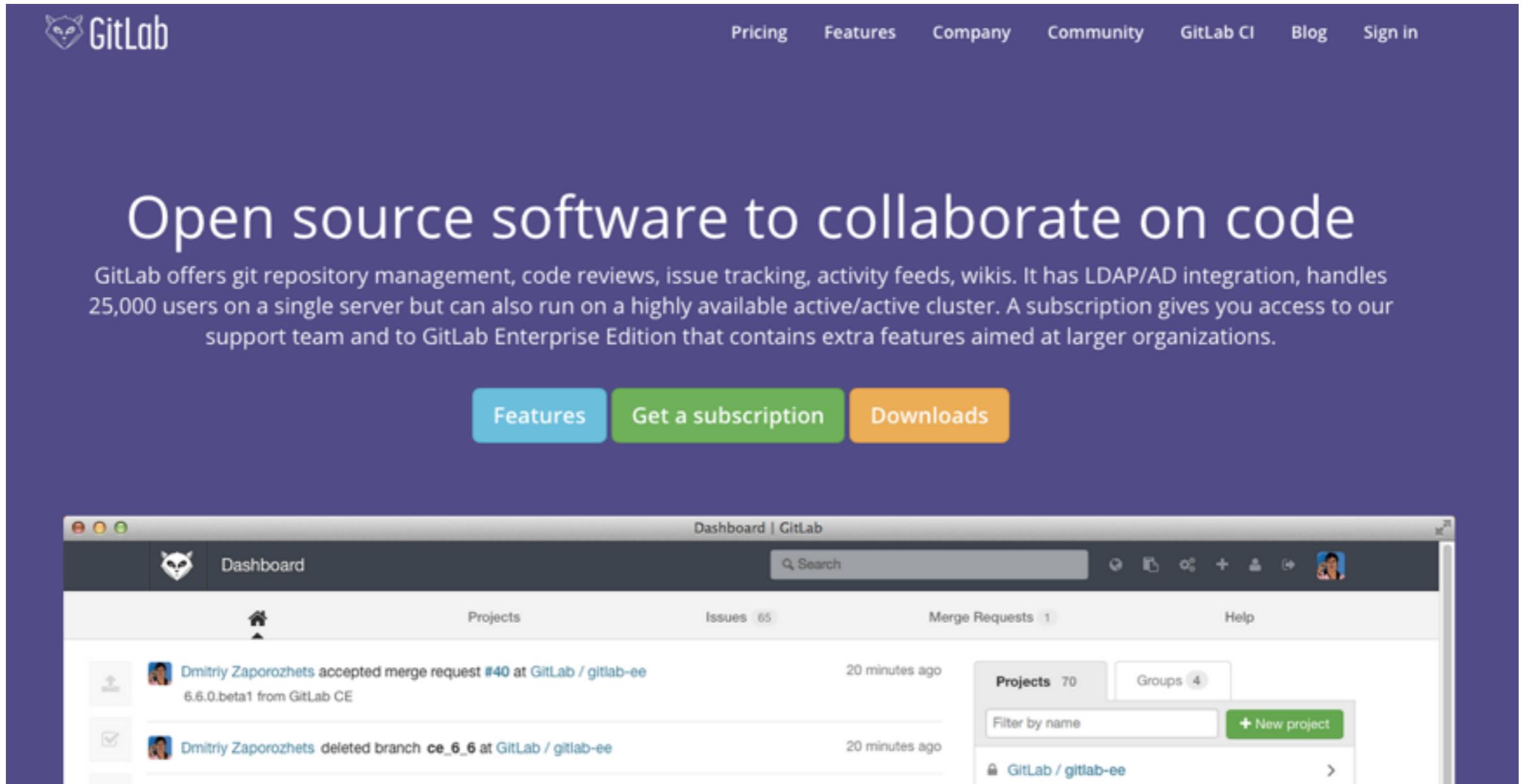
Lemonade
50¢ each



Lemonade
\$50 each

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

INTERNAL OPEN SOURCE

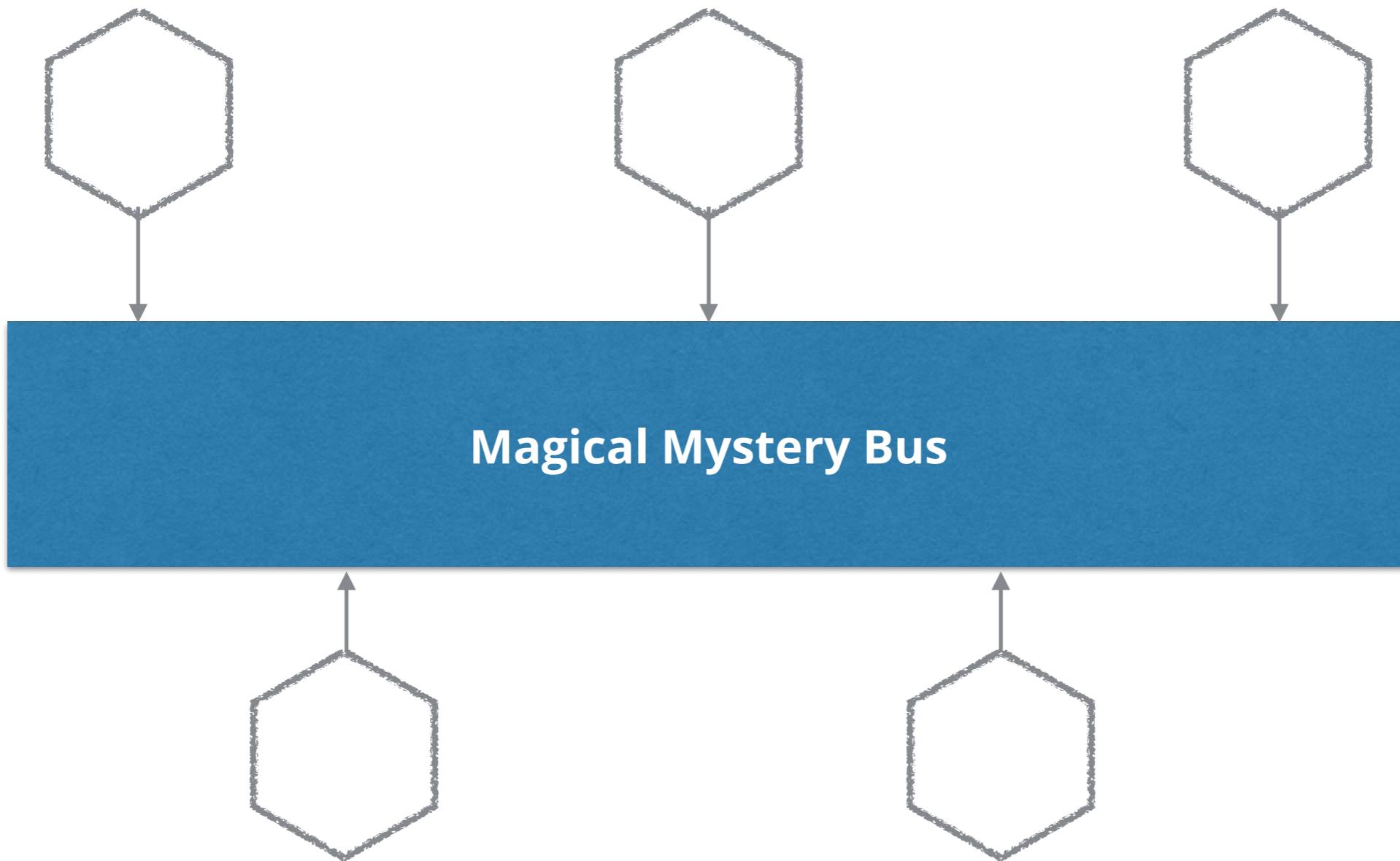


The screenshot shows the GitLab homepage with a dark purple header. The header features the GitLab logo (a cat icon) and the word "GitLab". To the right are navigation links: Pricing, Features, Company, Community, GitLab CI, Blog, and Sign in. Below the header, a large white section contains the heading "Open source software to collaborate on code" in a large, bold, sans-serif font. A descriptive paragraph follows, stating: "GitLab offers git repository management, code reviews, issue tracking, activity feeds, wikis. It has LDAP/AD integration, handles 25,000 users on a single server but can also run on a highly available active/active cluster. A subscription gives you access to our support team and to GitLab Enterprise Edition that contains extra features aimed at larger organizations." At the bottom of this section are three buttons: "Features" (blue), "Get a subscription" (green), and "Downloads" (orange). The main content area below this is a screenshot of the GitLab dashboard. The dashboard has a dark theme with a light gray header bar. The header bar includes the GitLab logo, the word "Dashboard", a search bar with placeholder text "Search", and user profile icons. Below the header, there are four main status indicators: "Projects" (70), "Groups" (4), "Filter by name", and a green button "+ New project". The main content area shows a list of recent activity items:

- Dmitriy Zaporozhets accepted merge request #40 at GitLab / gitlab-ee 6.6.0.beta1 from GitLab CE 20 minutes ago
- Dmitriy Zaporozhets deleted branch ce_6_6 at GitLab / gitlab-ee 20 minutes ago

@samnewman

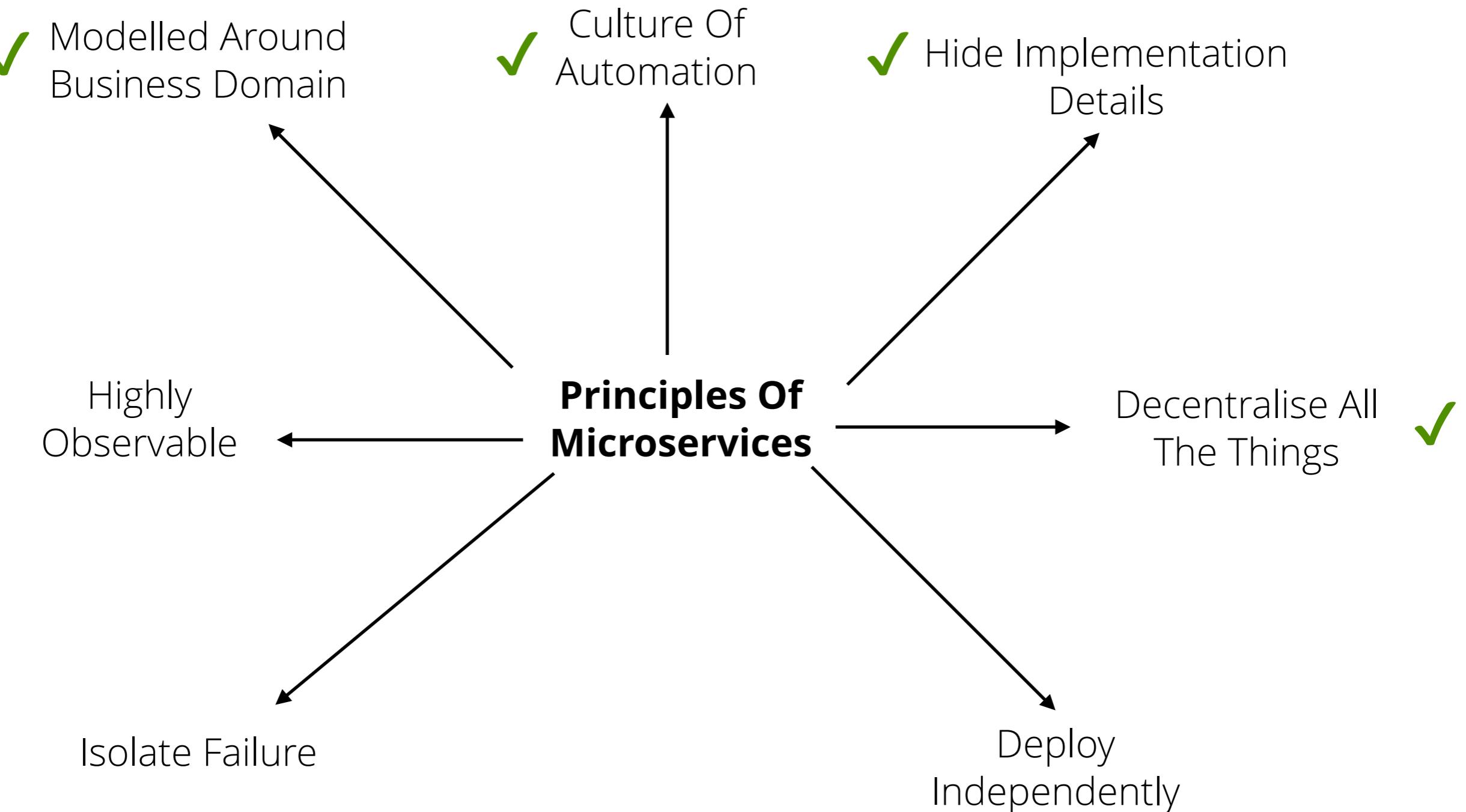
DUMB-PIPES, SMART ENDPOINTS



@samnewman

Magical Mystery Bus





✓ Modelled Around Business Domain

✓ Culture Of Automation

✓ Hide Implementation Details

Highly Observable

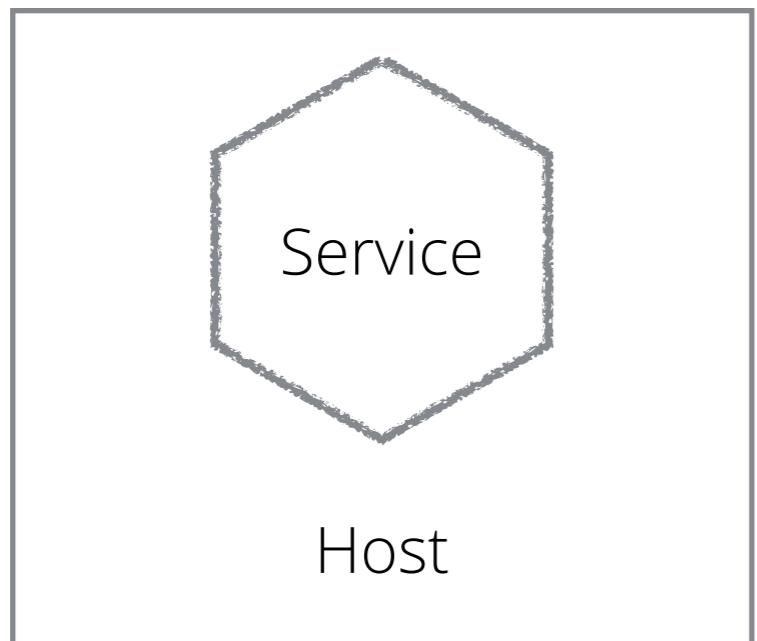
Principles Of Microservices

Decentralise All The Things ✓

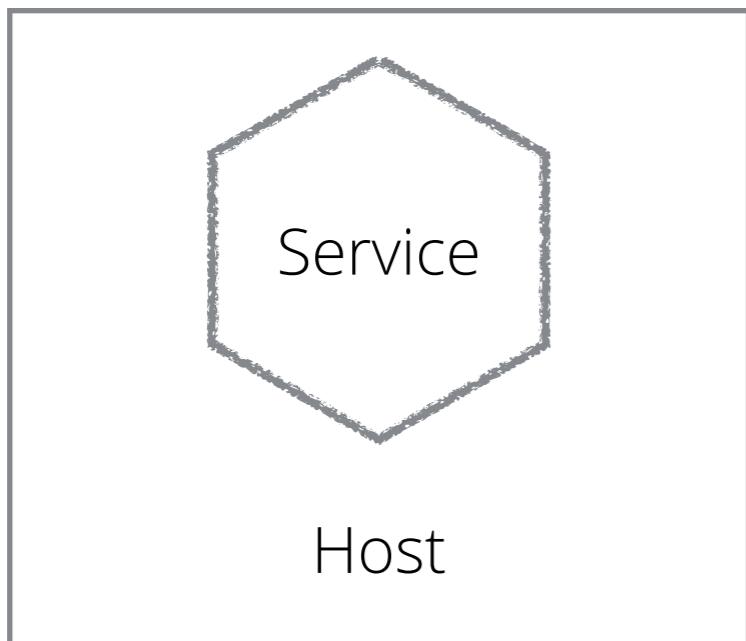
Isolate Failure

Deploy Independently

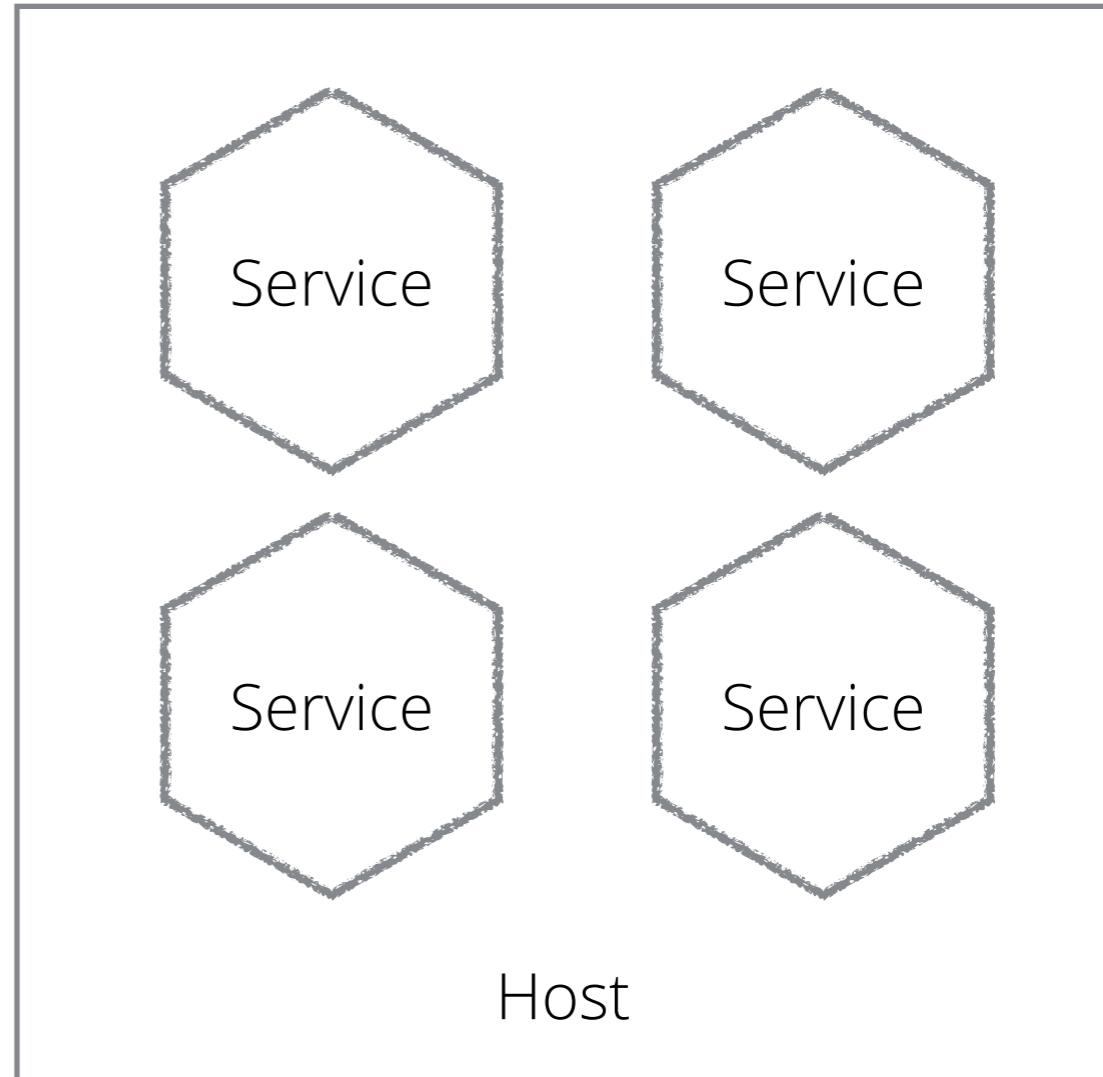
ONE SERVICE PER-HOST



ONE SERVICE PER-HOST

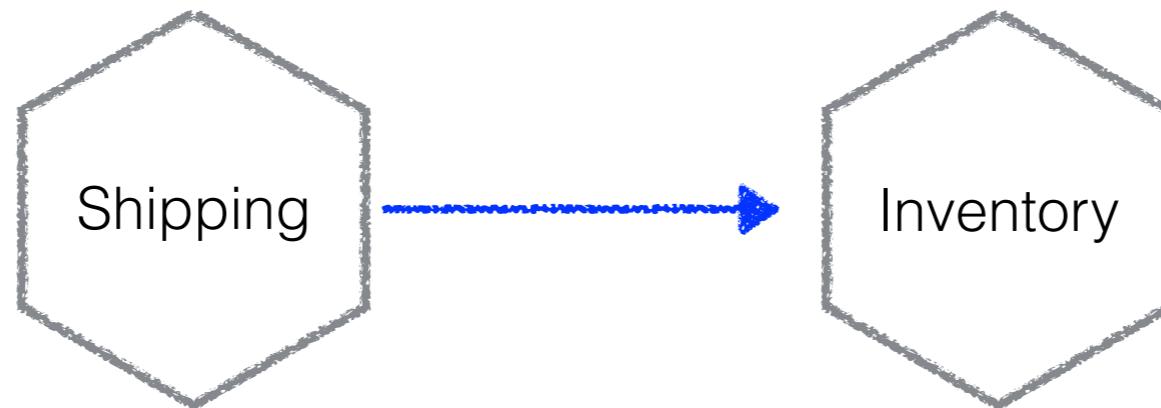


VS



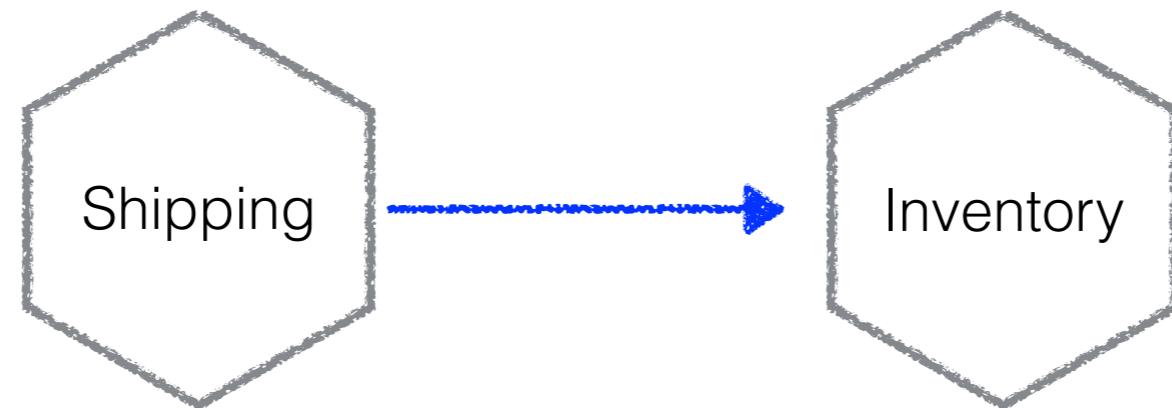
@samnewman

CONSUMER-DRIVEN CONTRACTS



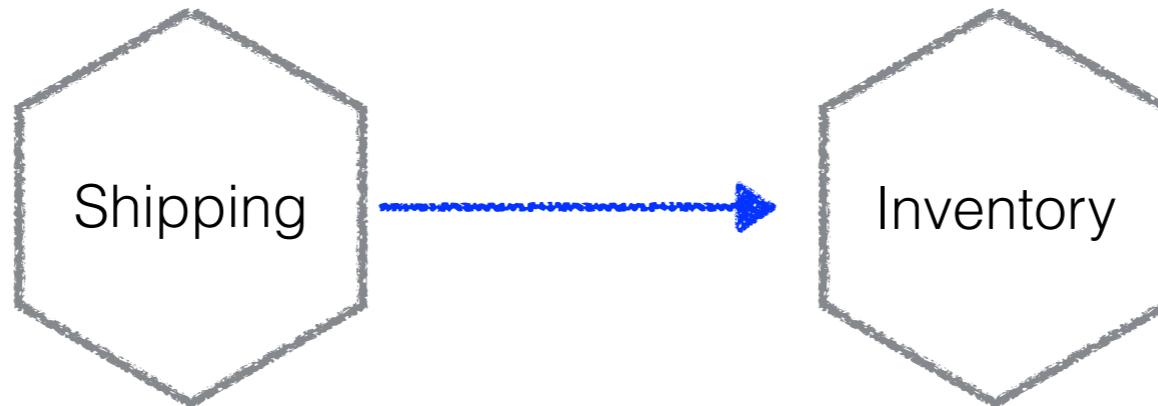
@samnewman

CONSUMER-DRIVEN CONTRACTS



Expectations

CONSUMER-DRIVEN CONTRACTS

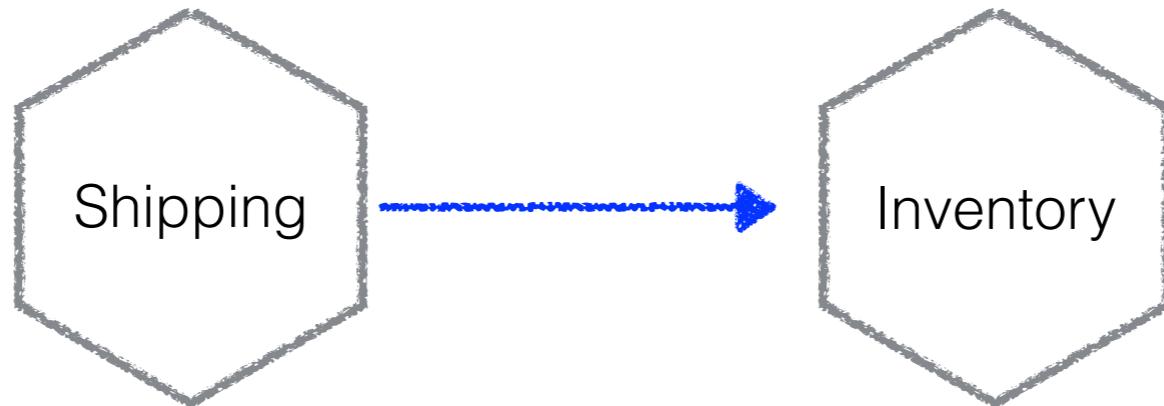


Expectations

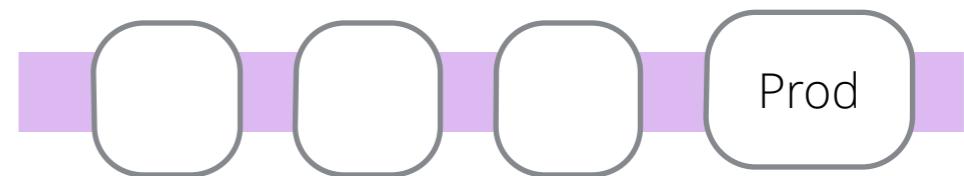


```
17 def asNode(): Node = {
18     attr: attr + "http://www.w3.org/2000/svg#id" + idLang + "en" + lang + "en";
19     children: []
20     link: link + "rel='stylesheet' href='resources/common.css' type='text/css' />
21         <link rel='stylesheet' href='resources/default.css' type='text/css' />
22         <meta http-equiv='refresh' content='30' />
23     </body>
24     <body>
25         &lt;content>${build}&lt;/content&gt;
26     </body>
27 }
28
29 private def content(build: List[Build]): Element = {
30     displayType match {
31         case "single" => ul := build.map(build => asElement(build)) :> ul
32         case "short" => {
33             if (build.length == 2) {
34                 ul := build.map(build => asElement(build)) :> ul
35             } else {
36                 ul := class("bullets")> build.map(build => bulletFormat(build)) :> ul
37             }
38         }
39         case _ => ul := class("bullet")> build.map(build => bulletFormat(build)) :> ul
40     }
41 }
42
43 private def asElement(build: Build): Element = {
44     val id := "bulid-" + build.getUniqueId.name.substring(0, 5);
45     <ul id=id>${bulletFormat(build)}</ul>
46 }
```

CONSUMER-DRIVEN CONTRACTS

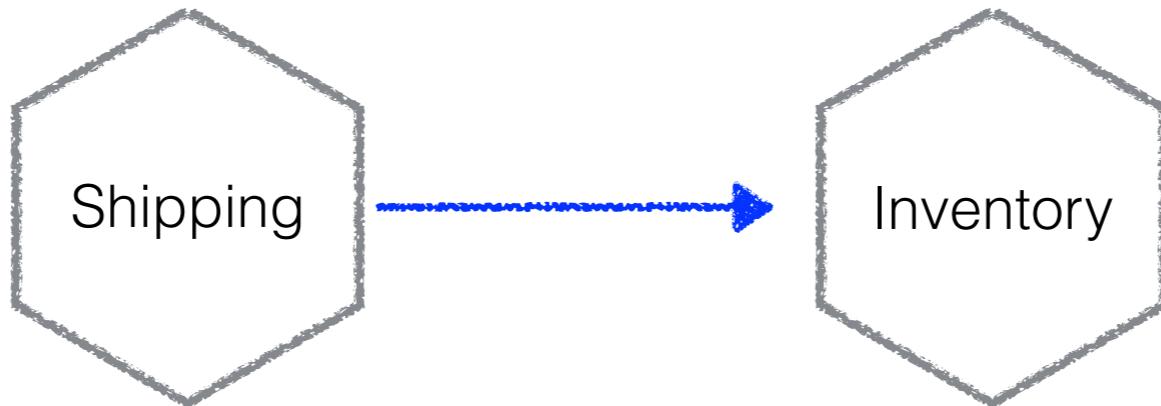


Expectations



```
17 def asNode(): Node = {
18     ->@rel="http://www.w3.org/2000/svg#node" ->@lang="en" lang="en">
19         ->head>
20             ->link rel="stylesheet" href="/static/common.css" type="text/css"/>
21             ->link rel="stylesheet" href="/common/style.css" type="text/css"/>
22             meta http-equiv="refresh" content="30"/>
23         </head>
24         <body>
25             &lt;content(build)&gt;
26         </body>
27     </html>
28 }
29
30 private def content(build: UnitBuild): Element = {
31     displayType match {
32         case "single" => s<div>{ build.map(build => asTable(build)) }</div>
33         case "short" => {
34             if (build.length == 2) {
35                 s<div>{ build.map(build => asTable(build)) }</div>
36             } else {
37                 val class="bulldot"<| build.map(build => buildFormat(build))></div>
38             }
39         }
40         case _ => s<ul class="bulldot">{ build.map(build => buildFormat(build)) }</ul>
41     }
42 }
43
44 private def asTable(build: Build): Element = {
45     table class="bulldot" = build.getOrNone.name.value.asTable>
46         <tr cell="odd"><td>{ buildFormat(build) }</td>
47             <td>{ buildFormat(build) }</td>
48         </tr>
49     </table>
50 }
```

CONSUMER-DRIVEN CONTRACTS



Expectations



```
17 def asNode(): Node = {
18     attr:attr=<a href="http://www.w3.org/2000/svg#node"> and lang='en' lang='en'>
19     child:
20     ->link rel="stylesheet" href="/static/common.css" type="text/css"/>
21     ->link rel="stylesheet" href="/common/style/> type="text/css"/>
22     meta http-equiv="refresh" content="30" />
23     </body>
24     <body>
25         & content(builds) &
26     </body>
27 }
28
29 private def content(builds: List[Build]): Element = {
30     displayType match {
31         case "single" => ul(id = build.map(build => asNode(build))) & ul
32         case "short" => {
33             if (builds.length == 2) {
34                 ul(id = build.map(build => asNode(build))) & ul
35             } else {
36                 ul(class="bullets") & build.map(build => bulletItem(build)) & ul
37             }
38         }
39         case _ => ul(class="bullet") & build.map(build => bulletItem(build)) & ul
40     }
41 }
42
43 private def bulletItem(build: Build): Element = {
44     val id = build.id + build.previousName.substring(0, 1)
45     <li id=id>bullet{id} type="button">
46         &nbsp; [ LinkToBuild(build) ] &nbsp;
47         &nbsp; &nbsp;
48     </li>
49 }
```



@samnewman

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.

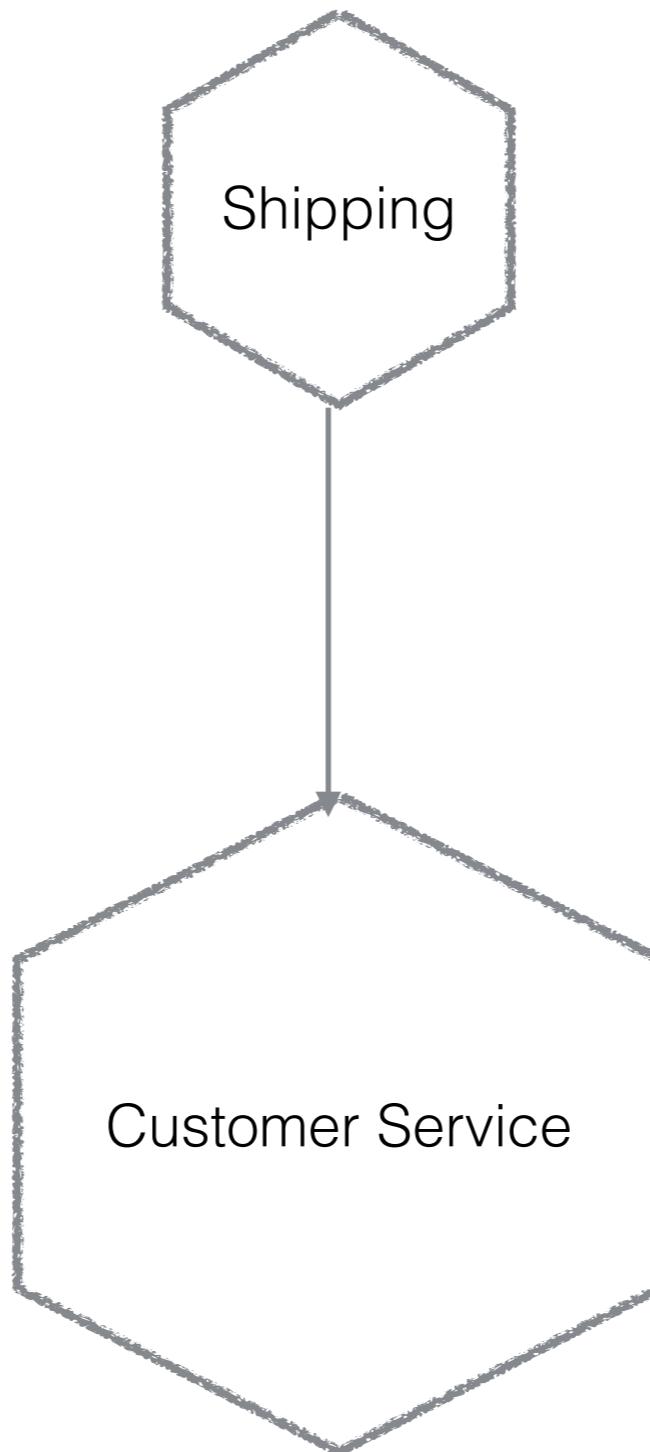
This allows testing of both sides of an integration point using fast unit tests.

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

Travis CI Status: 

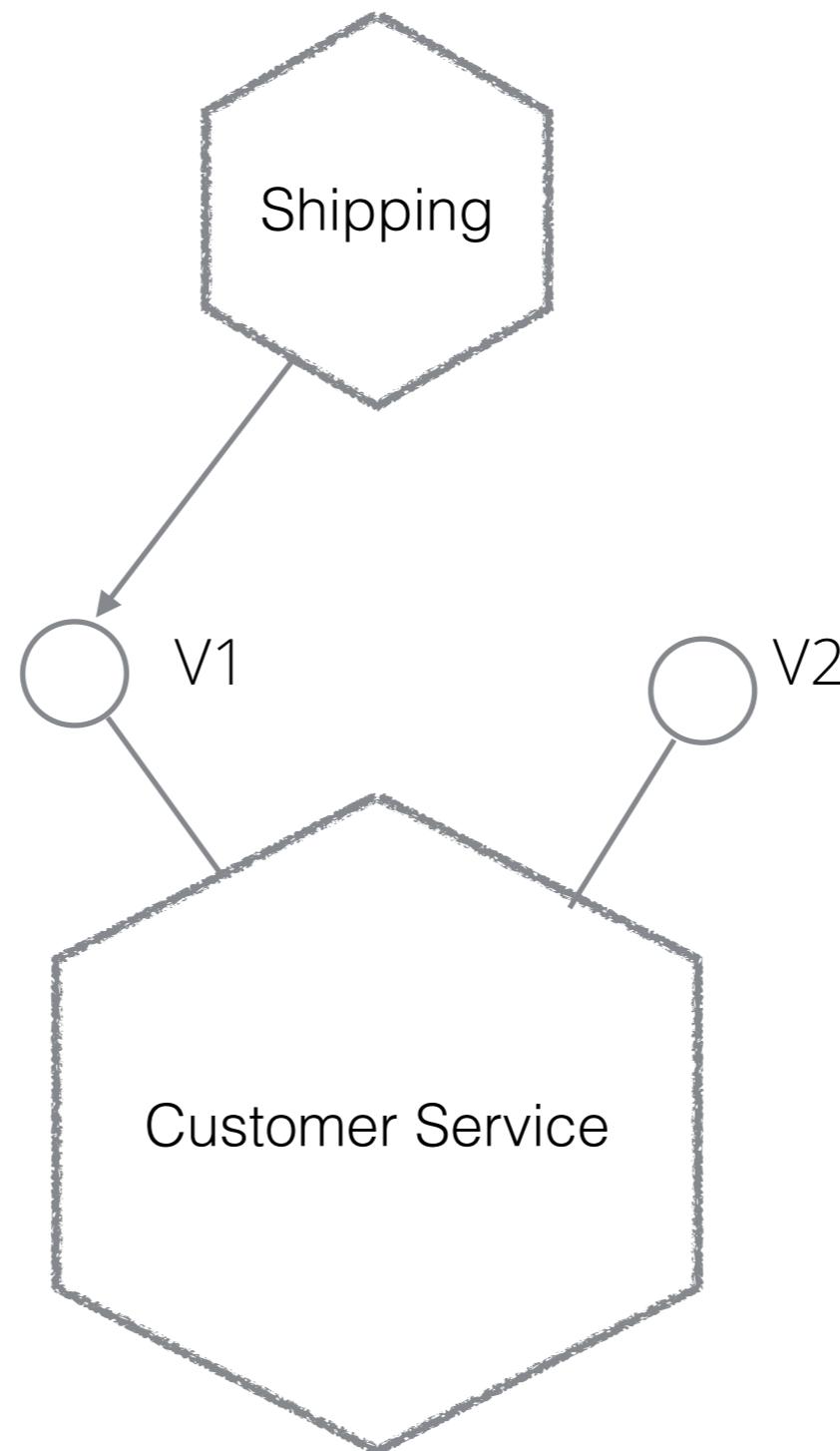
<https://github.com/realestate-com-au/pact>

CO-EXIST ENDPOINTS

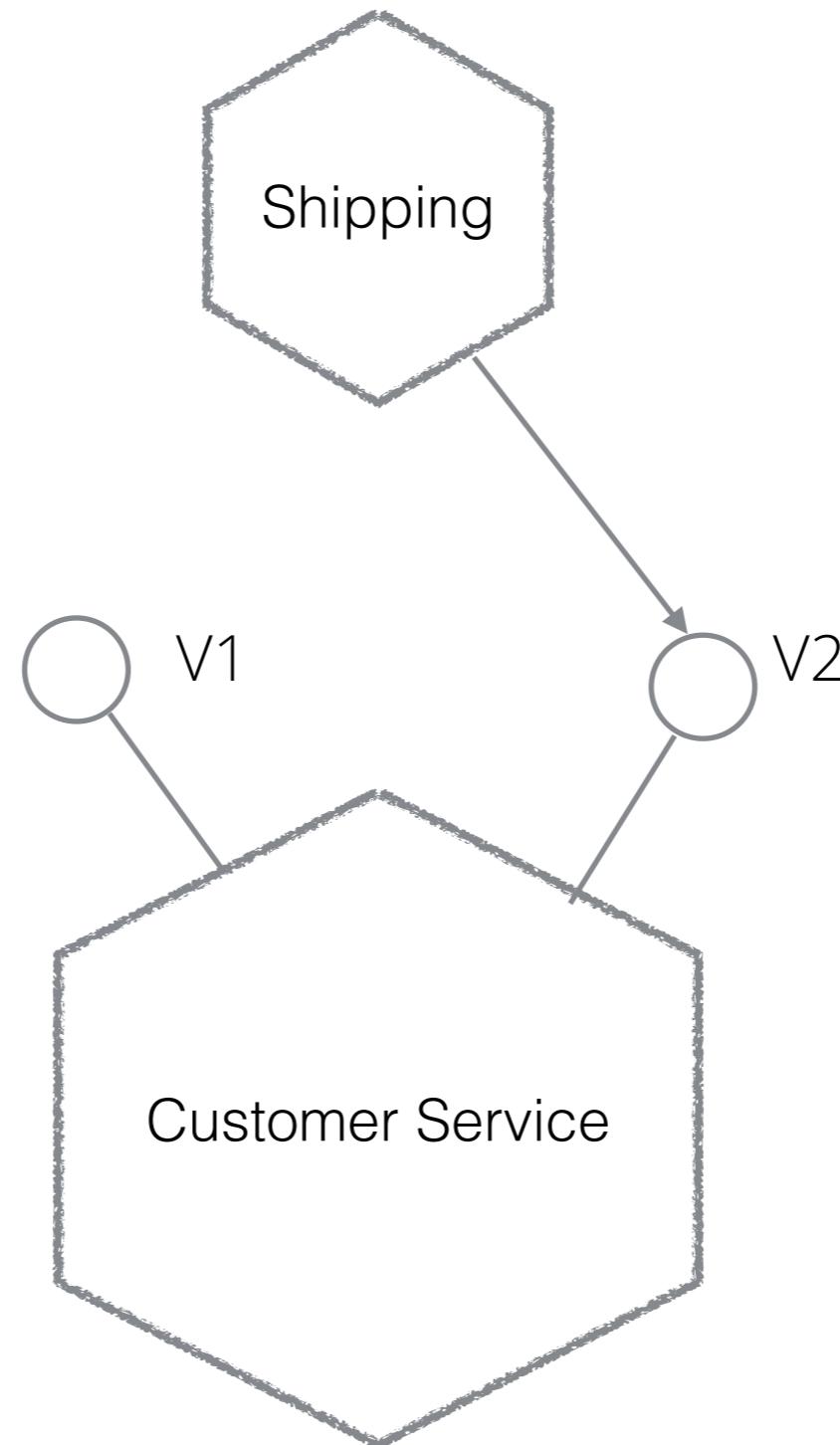


@samnewman

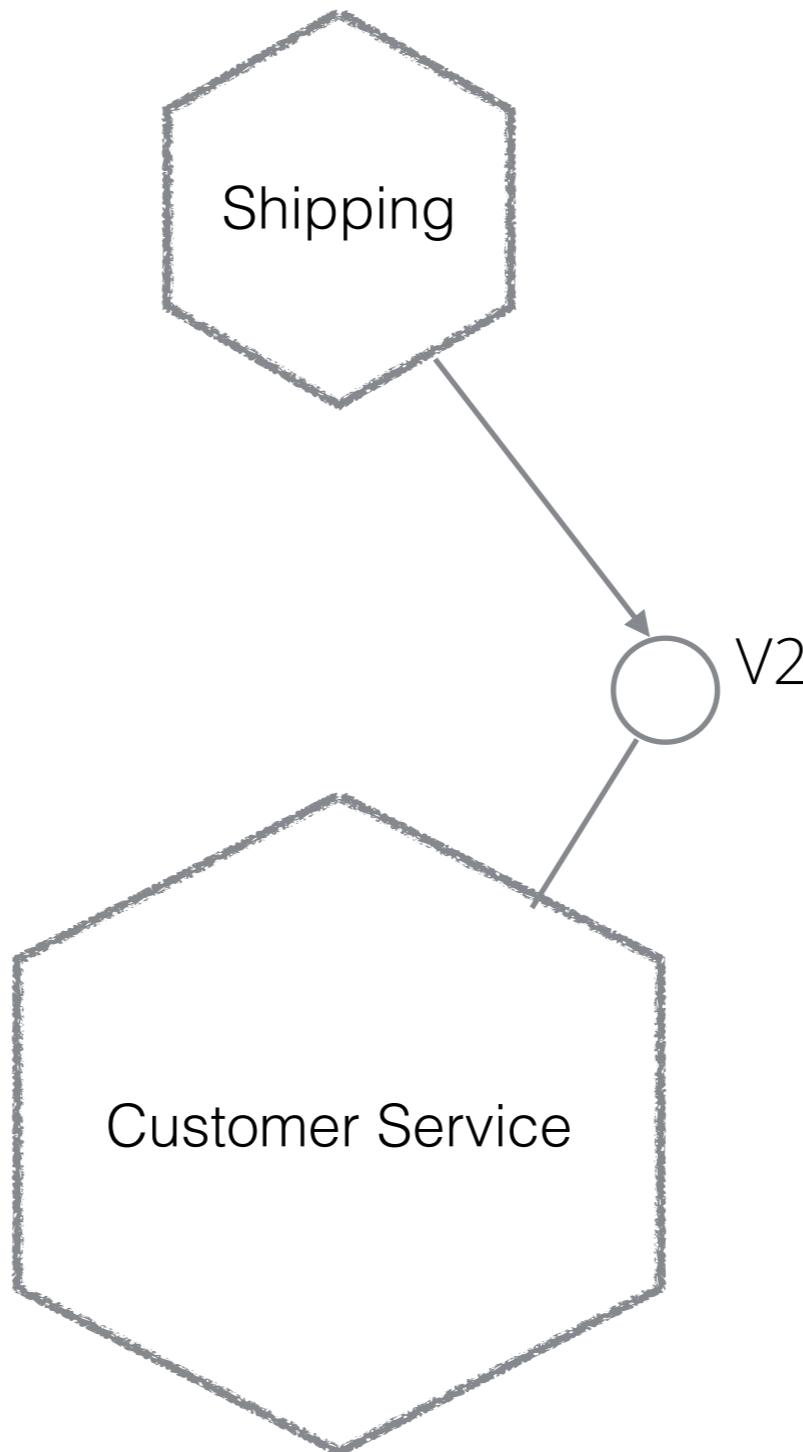
CO-EXIST ENDPOINTS



CO-EXIST ENDPOINTS



CO-EXIST ENDPOINTS



@samnewman

✓ Modelled Around Business Domain

✓ Culture Of Automation

✓ Hide Implementation Details

Highly Observable

Principles Of Microservices

Decentralise All The Things ✓

Isolate Failure

✓ Deploy Independently

✓ Modelled Around Business Domain

✓ Culture Of Automation

✓ Hide Implementation Details

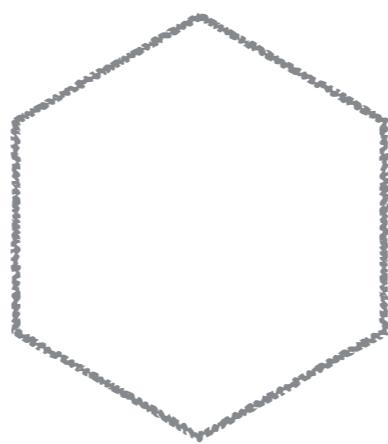
Highly Observable

Principles Of Microservices

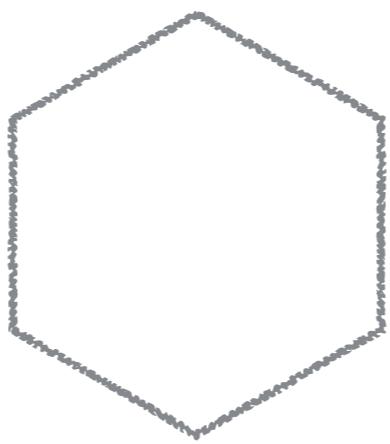
Decentralise All The Things ✓

Isolate Failure

✓ Deploy Independently

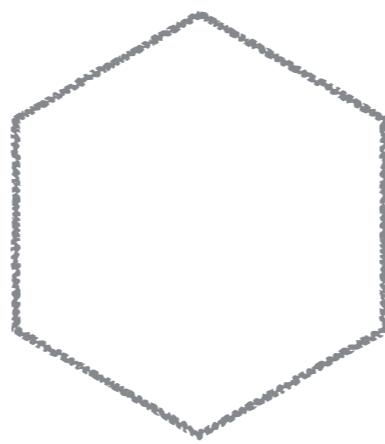


@samnewman

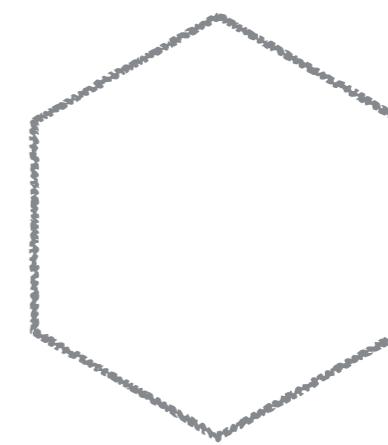
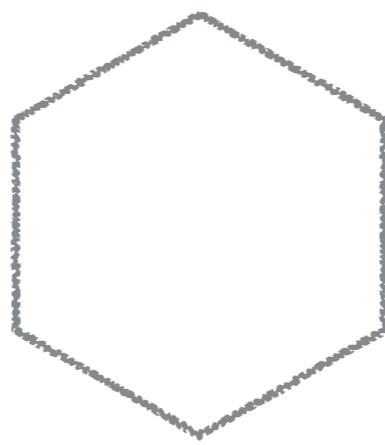
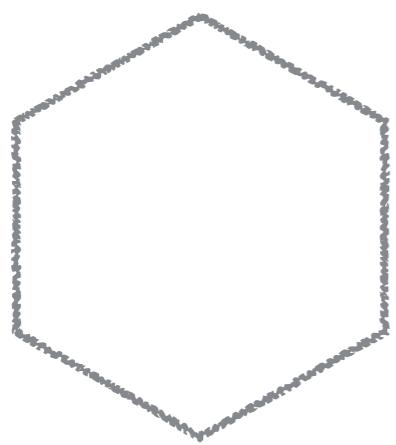


1 in 100

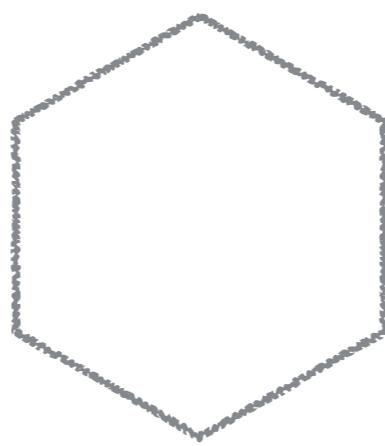
@samnewman



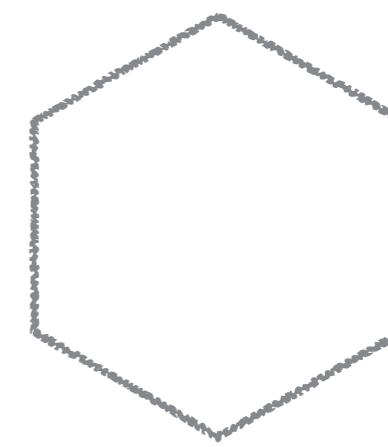
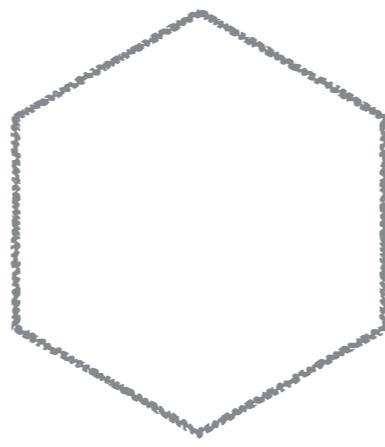
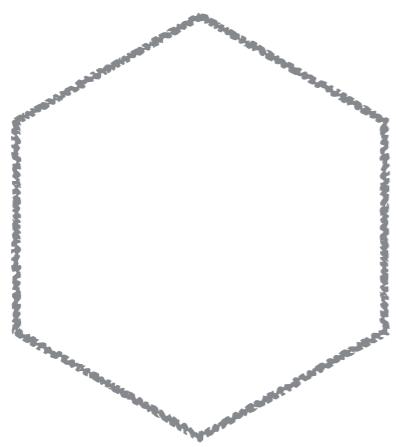
1 in 100



@samnewman



4 in 100



@samnewman

Strangler App

@samnewman

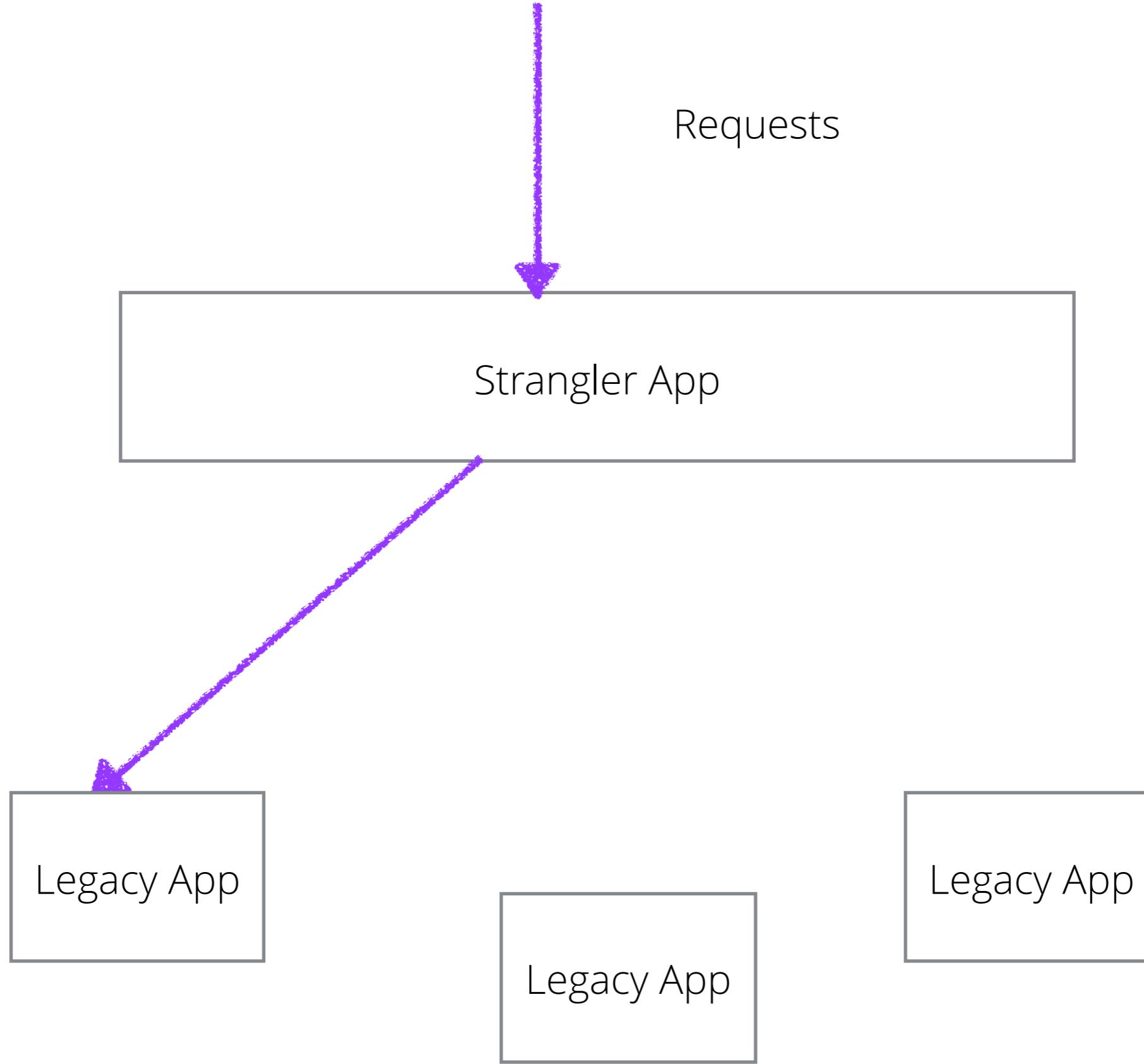
Strangler App

Legacy App

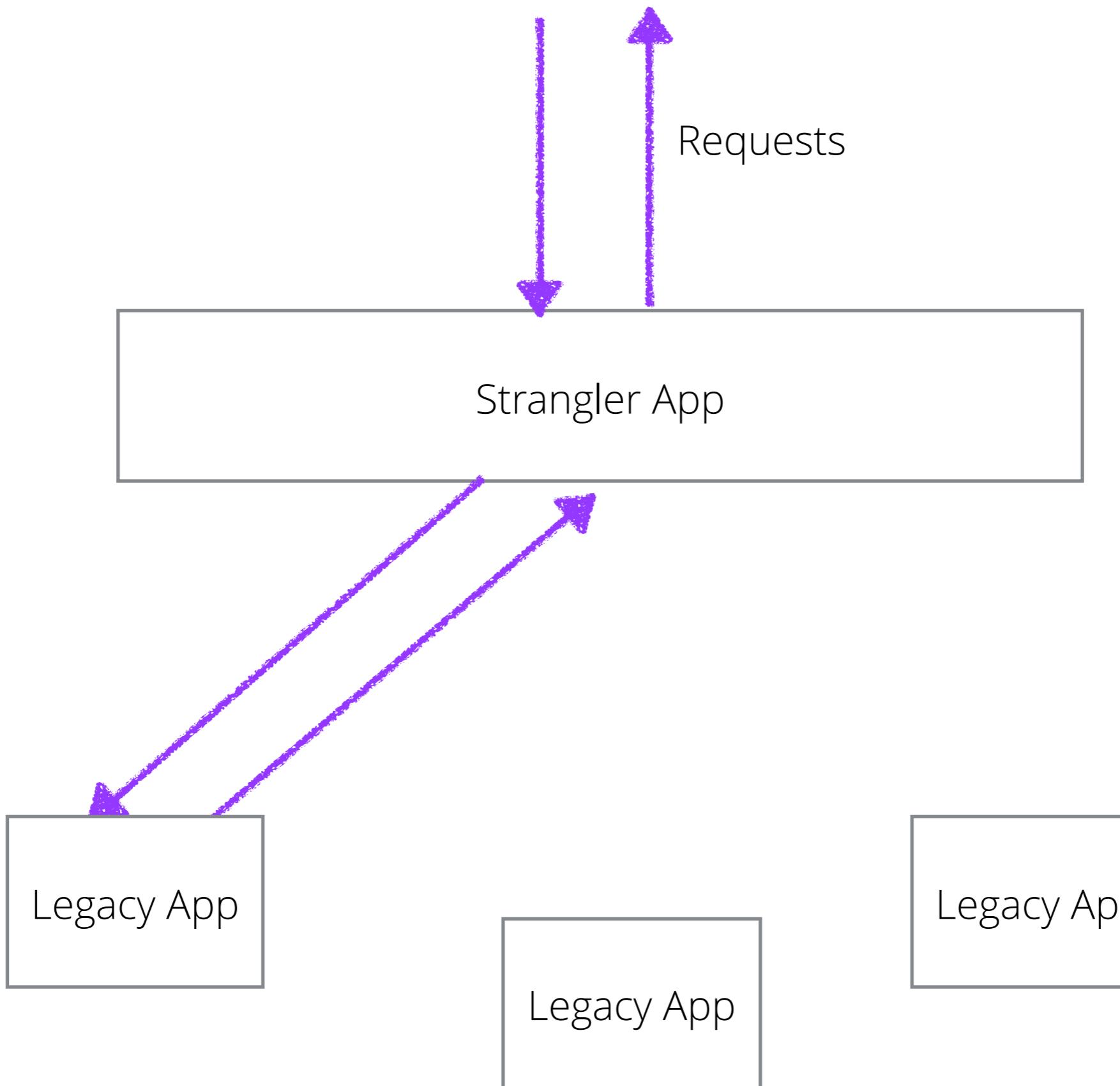
Legacy App

Legacy App

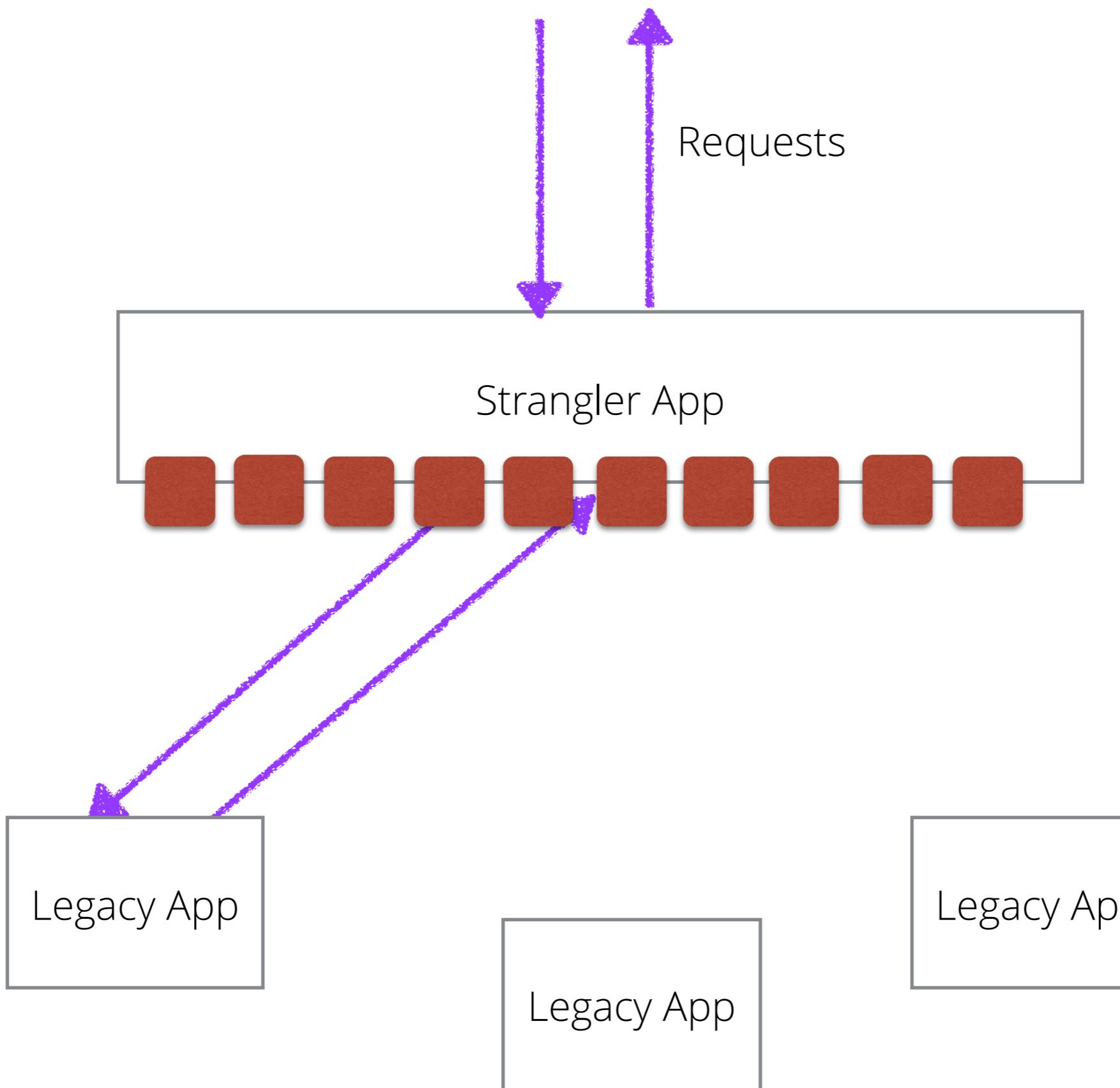
@samnewman



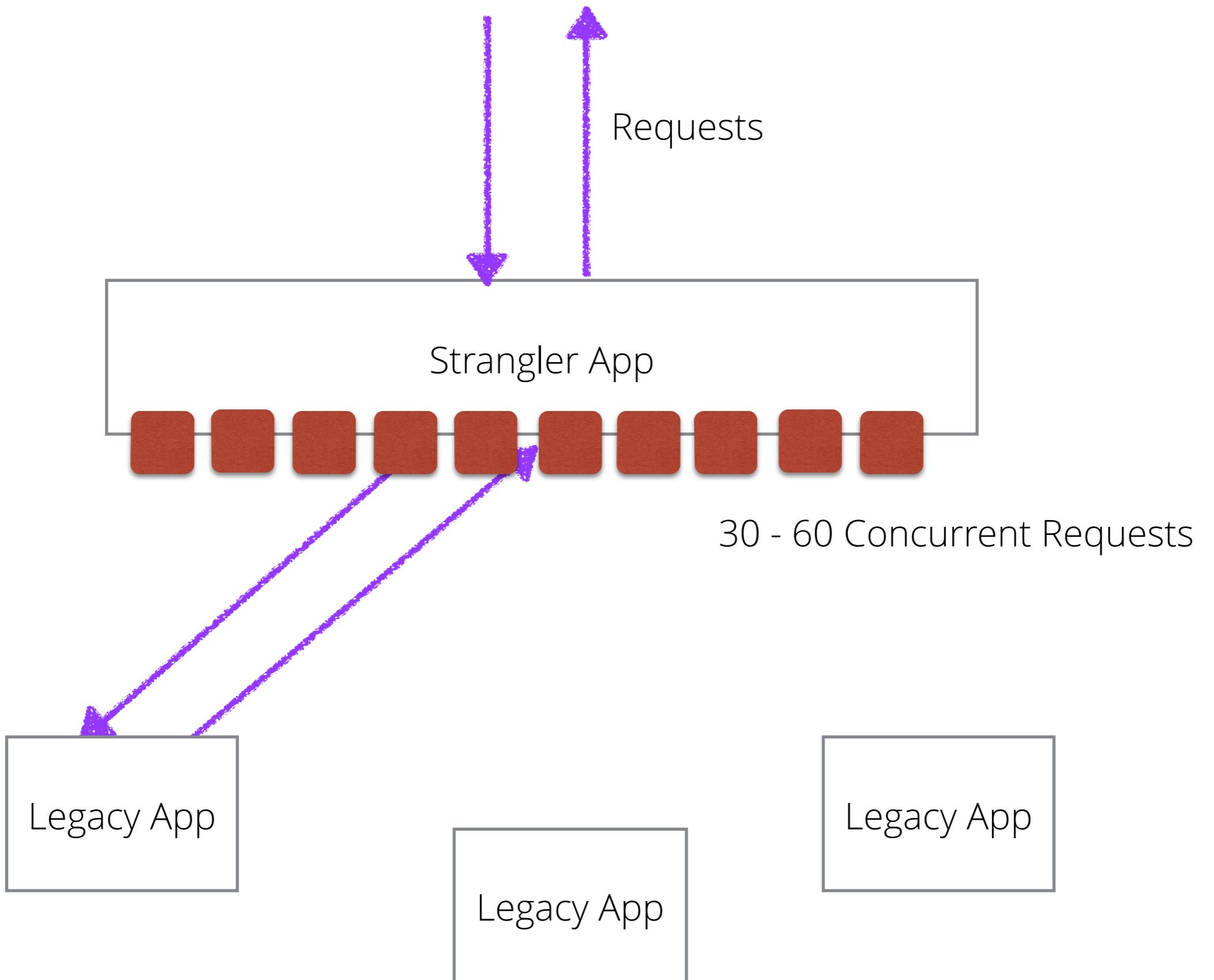
@samnewman

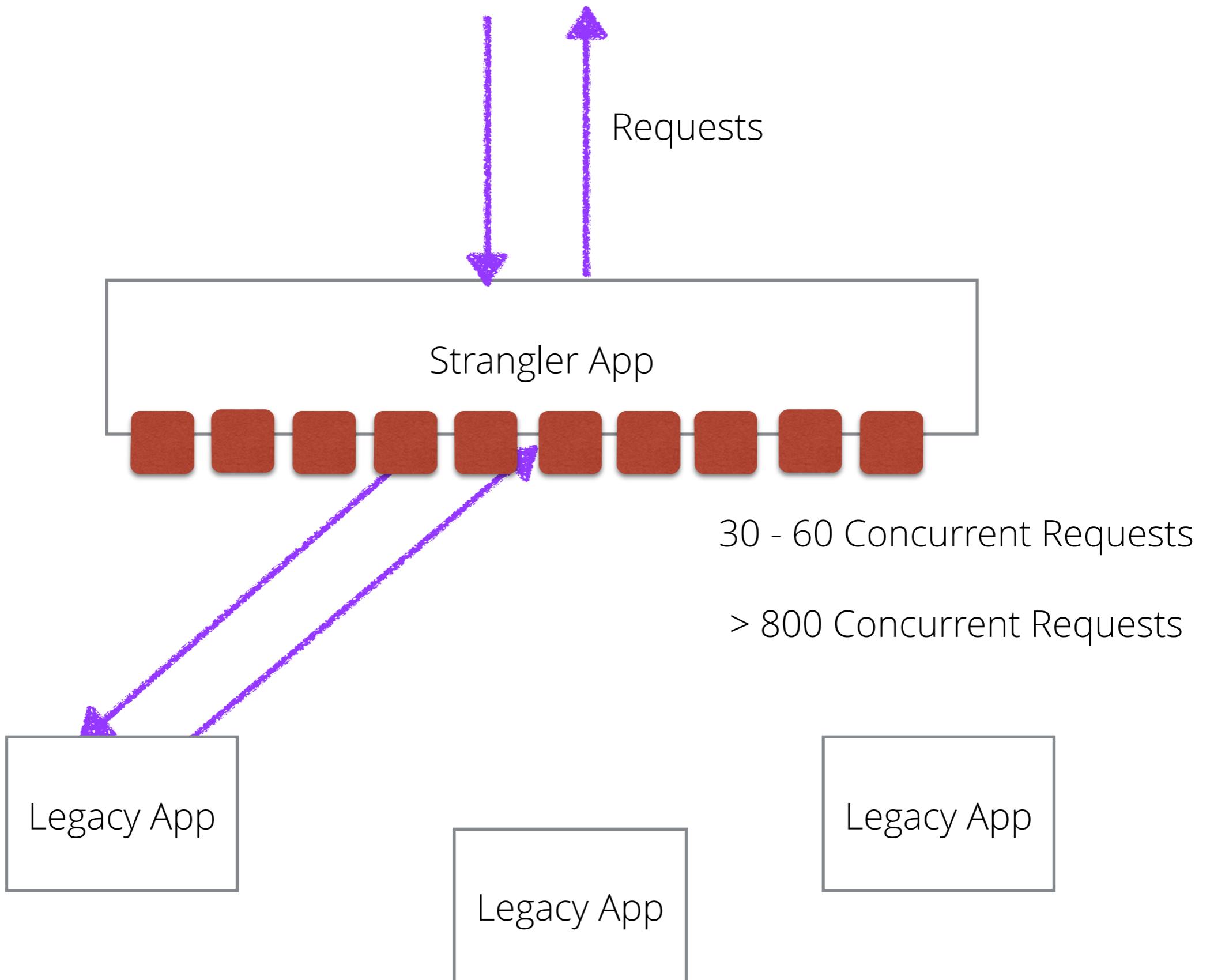


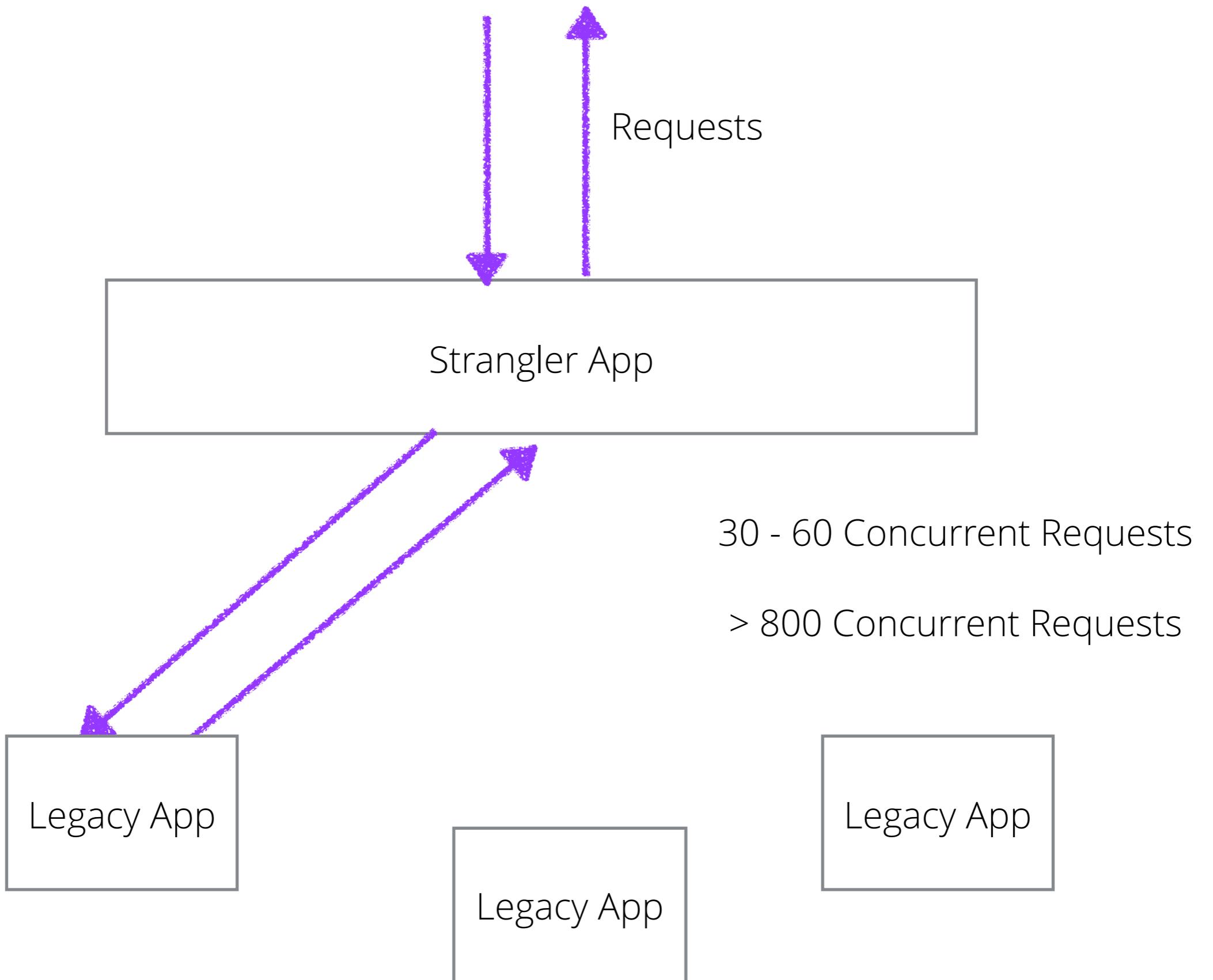
@samnewman



@samnewman

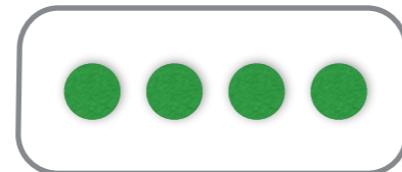






Strangler App

Thread Pool



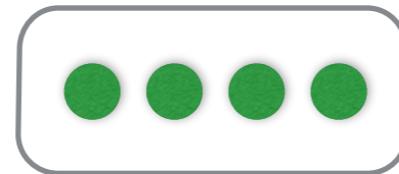
Legacy App

Legacy App

Legacy App

Strangler App

Thread Pool



Failing...slowly!

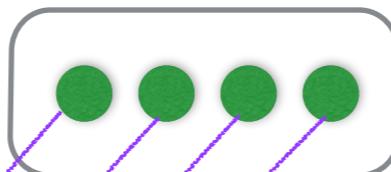
Legacy App

Legacy App

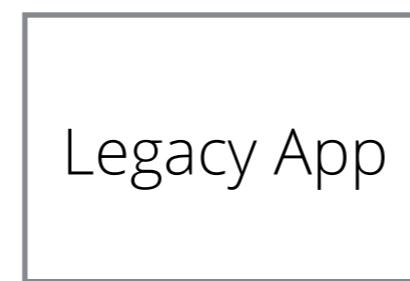
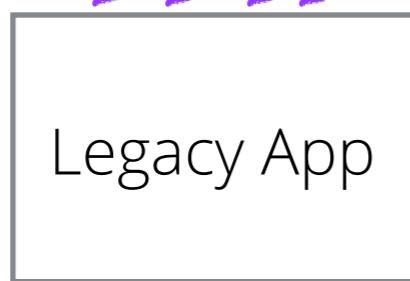
Legacy App

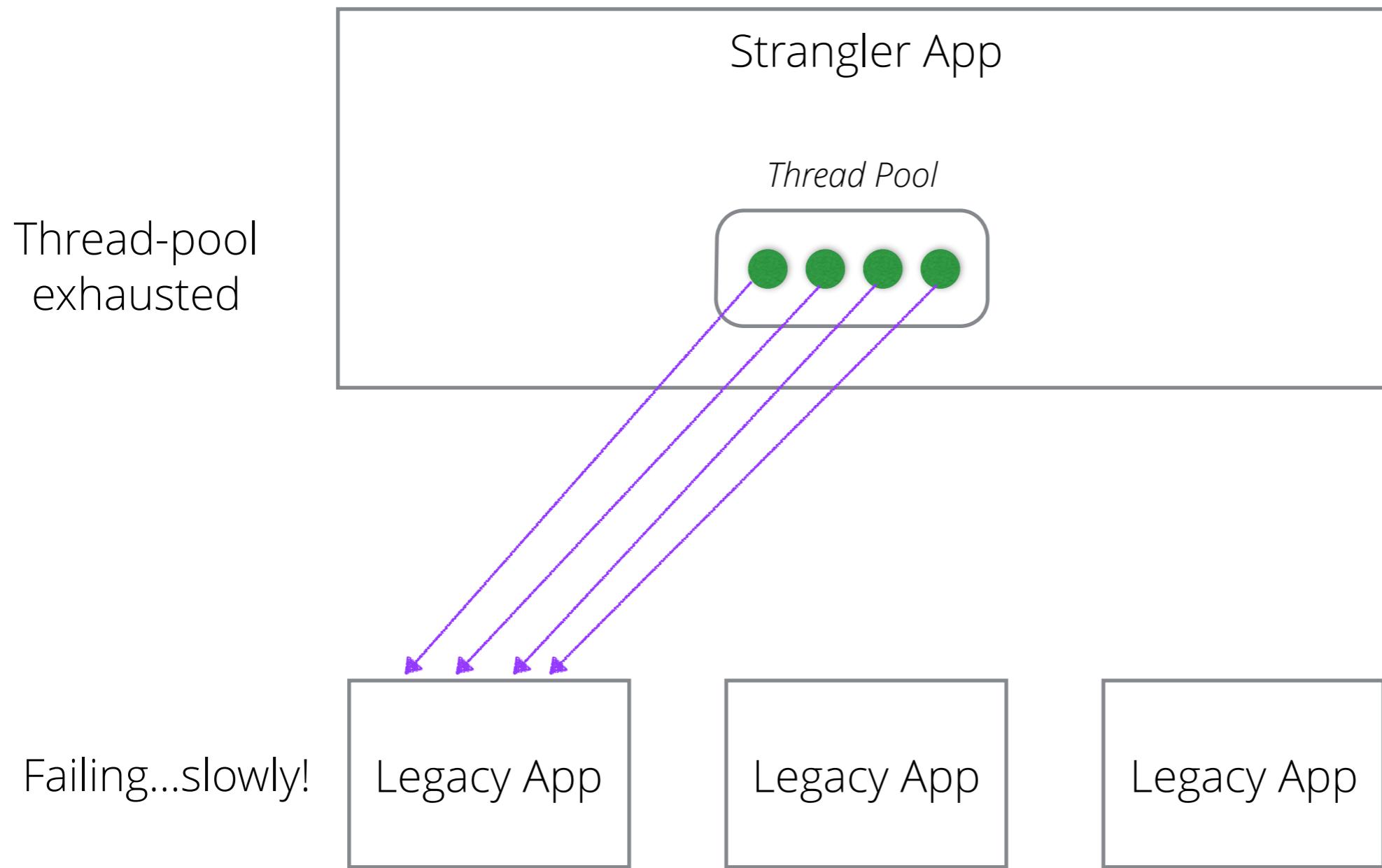
Strangler App

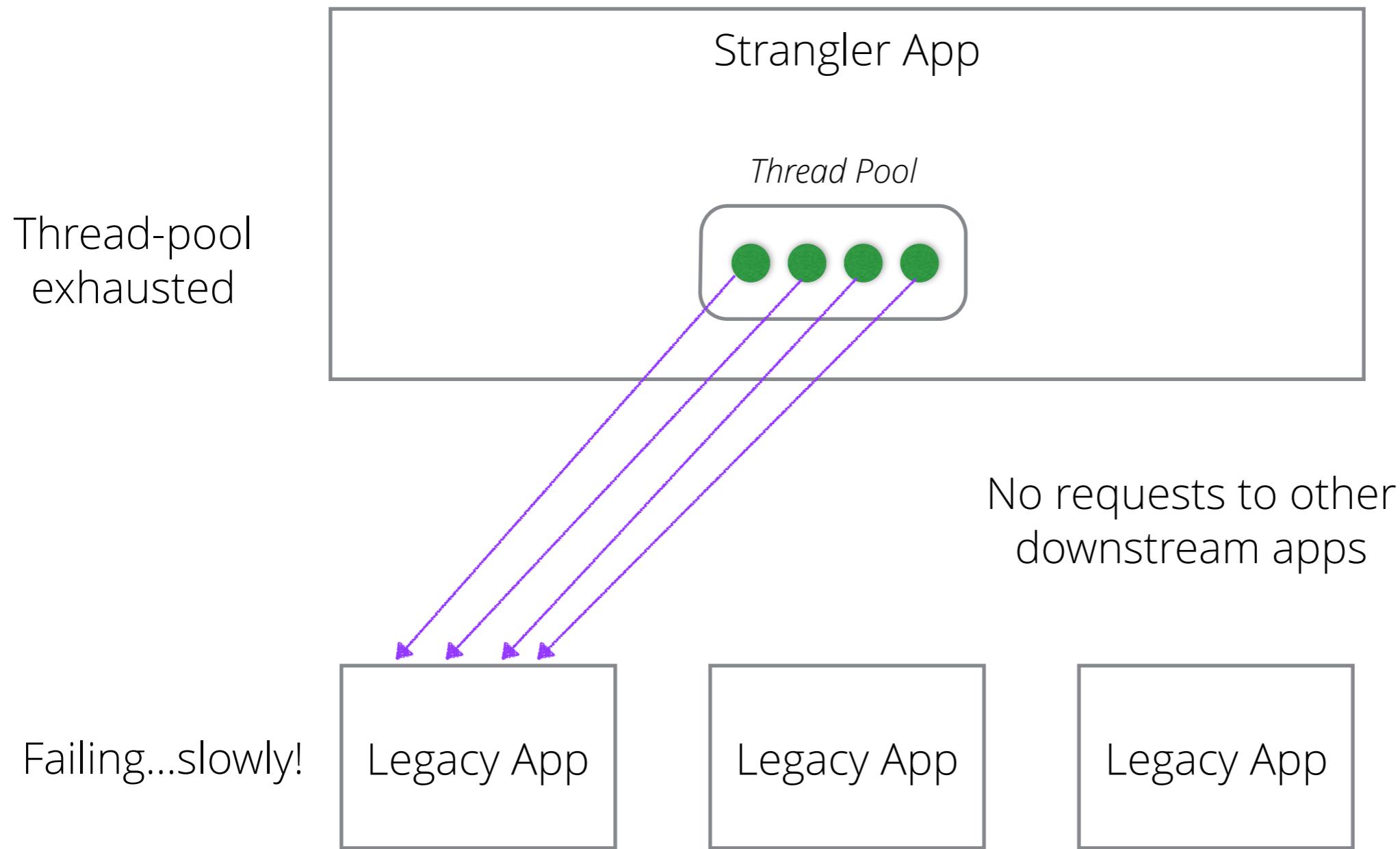
Thread Pool

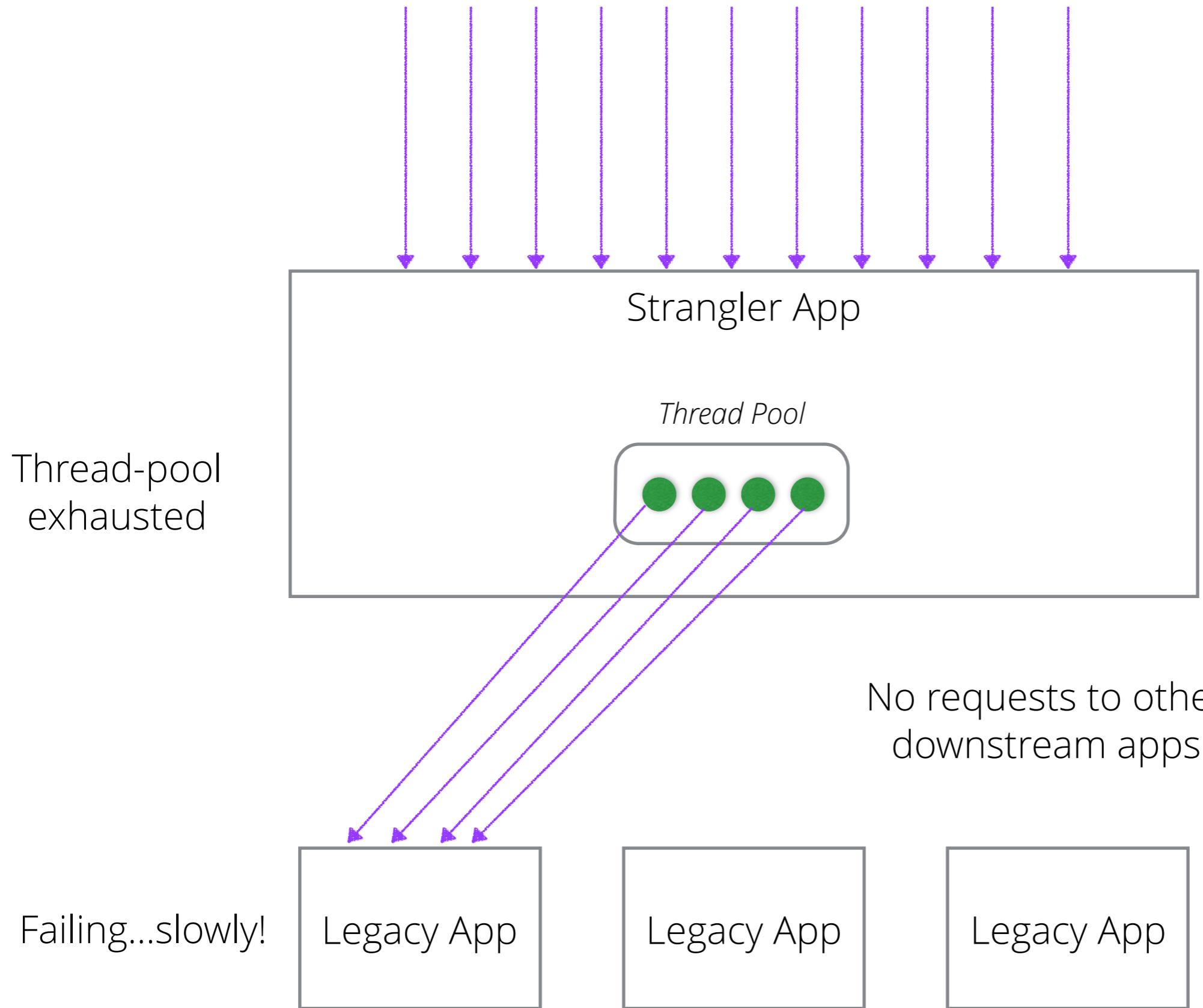


Failing...slowly!

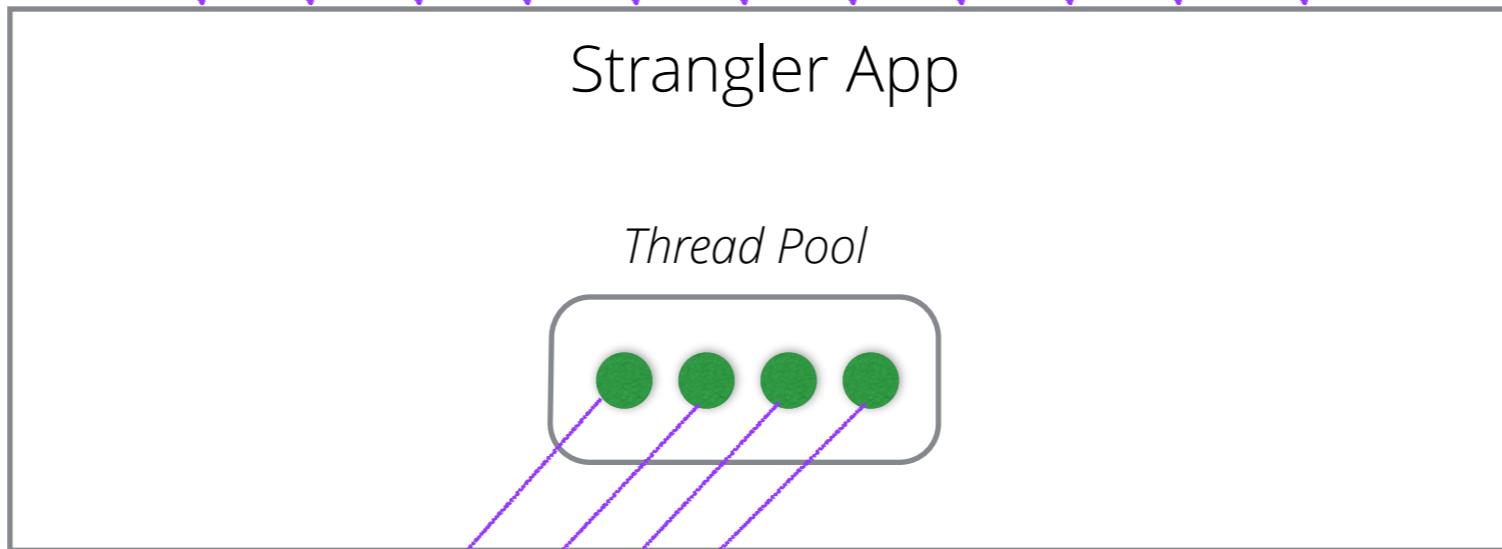






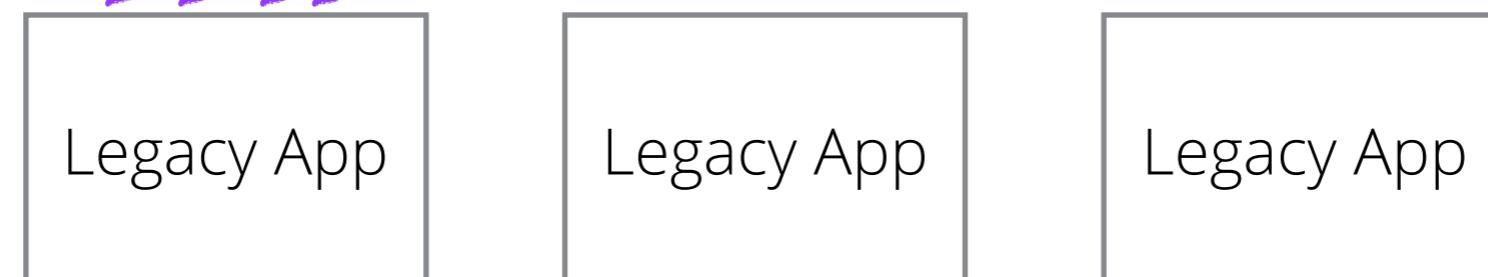


Requests
Building Up



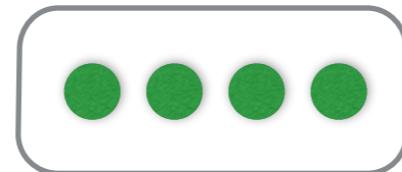
No requests to other
downstream apps

Failing...slowly!



Strangler App

Thread Pool

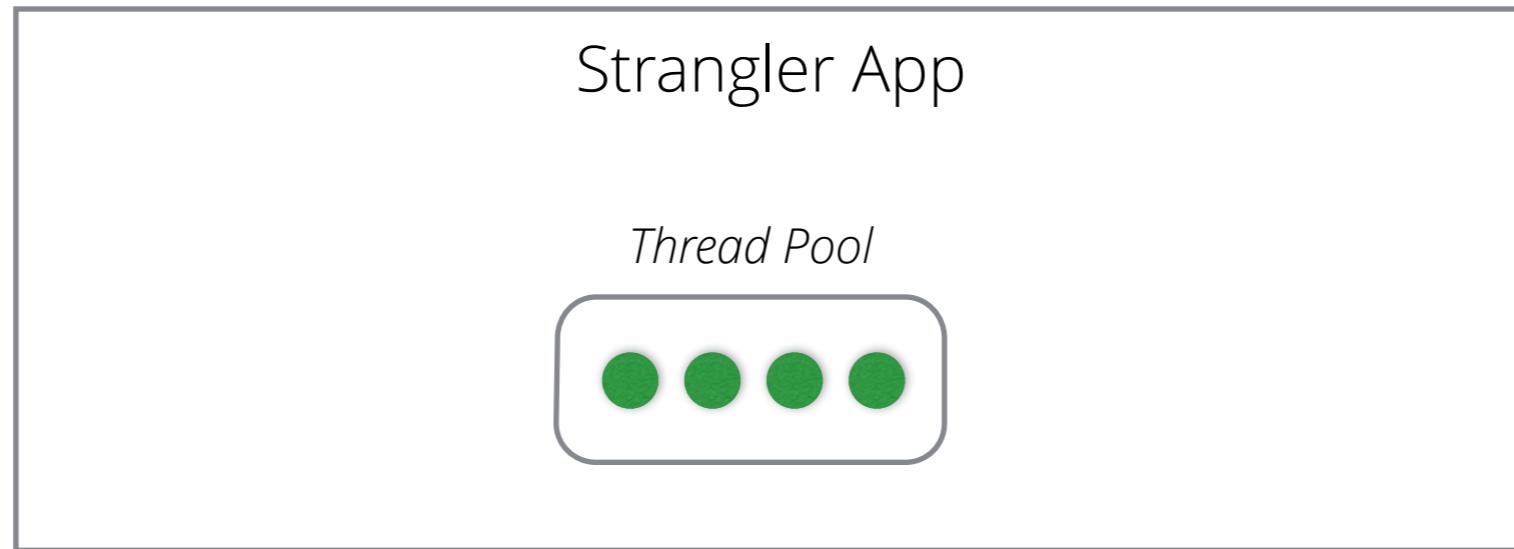


Legacy App

Legacy App

Legacy App

Fix **Timeouts**



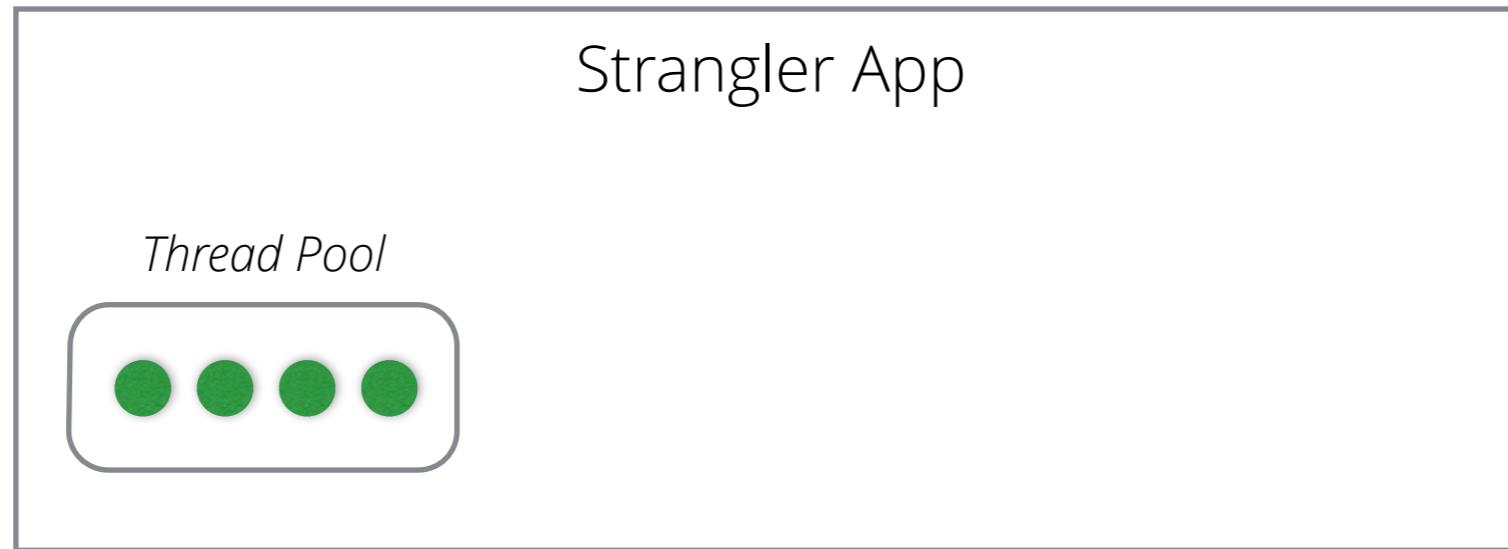
Legacy App

Legacy App

Legacy App

@samnewman

Fix **Timeouts**

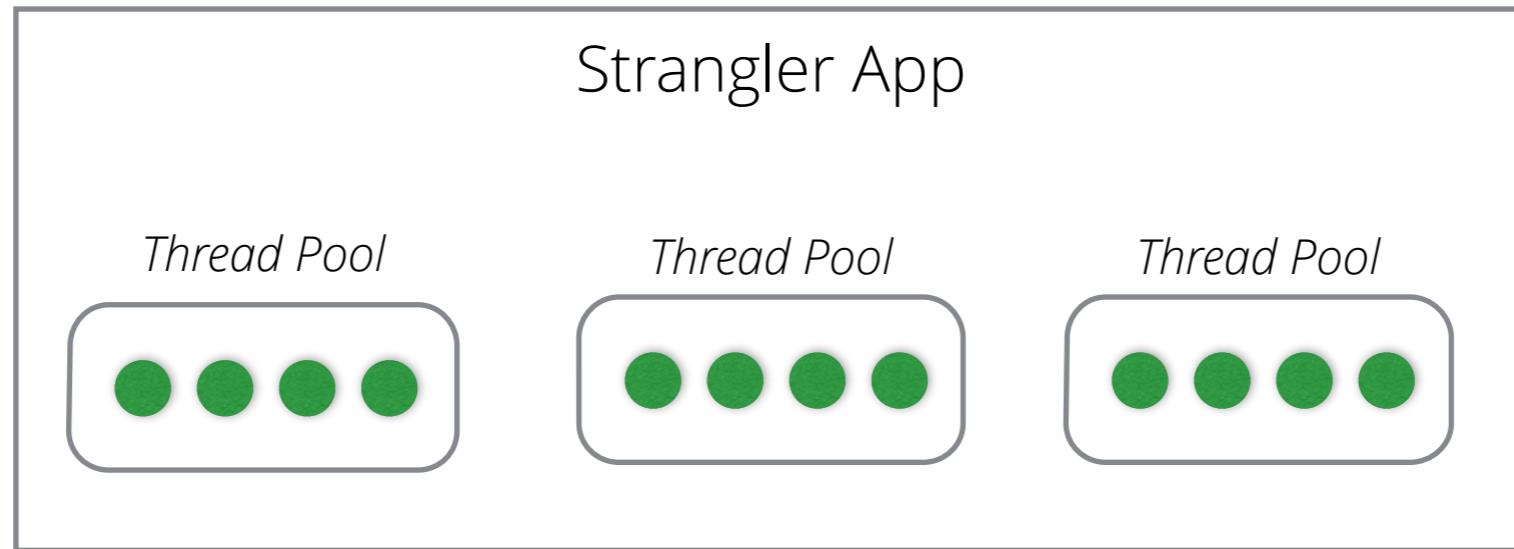


Legacy App

Legacy App

Legacy App

Fix **Timeouts**

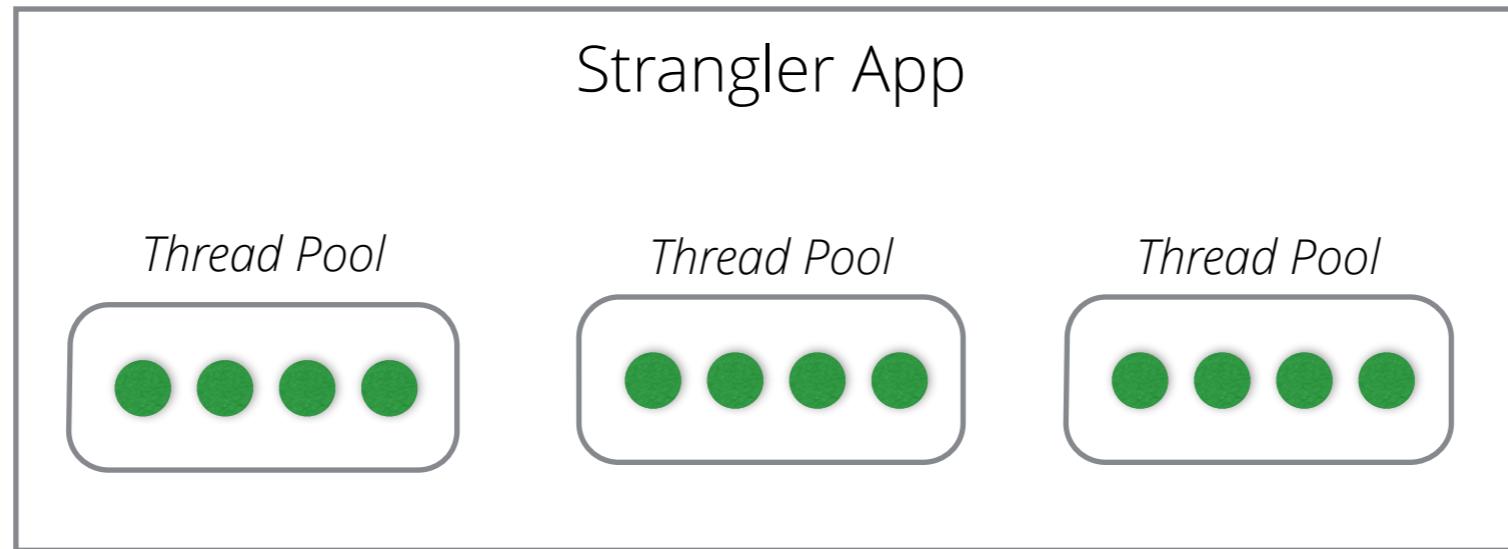


Legacy App

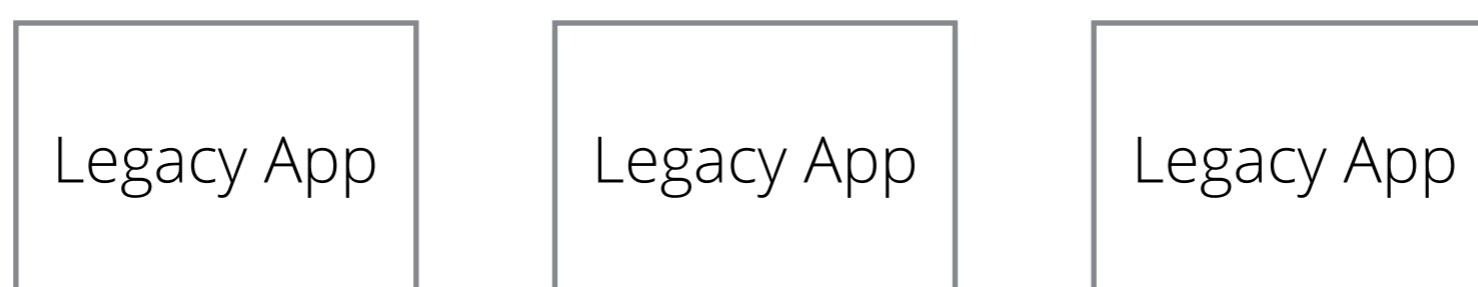
Legacy App

Legacy App

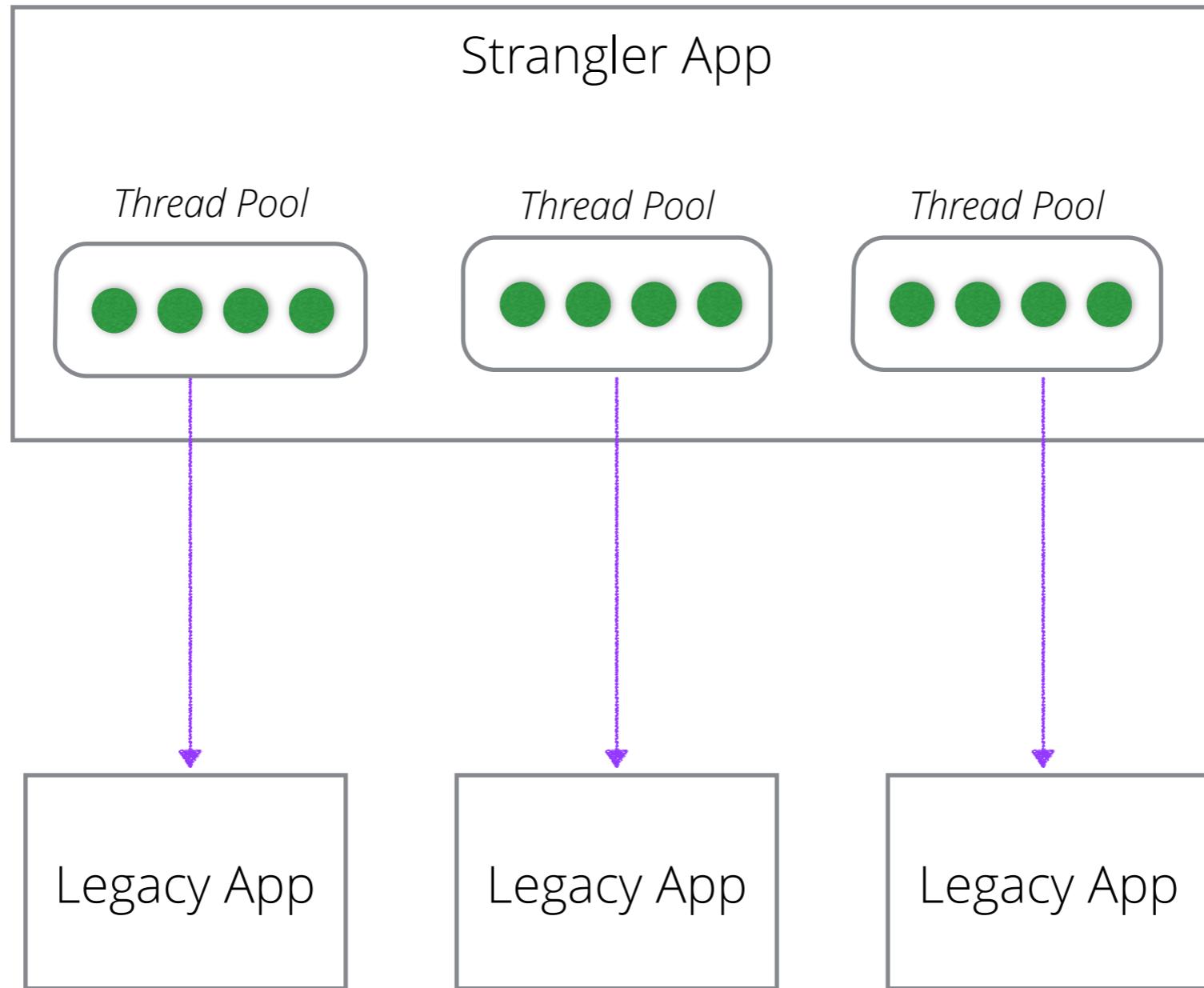
Fix **Timeouts**



Bulkhead
Downstream
Connections



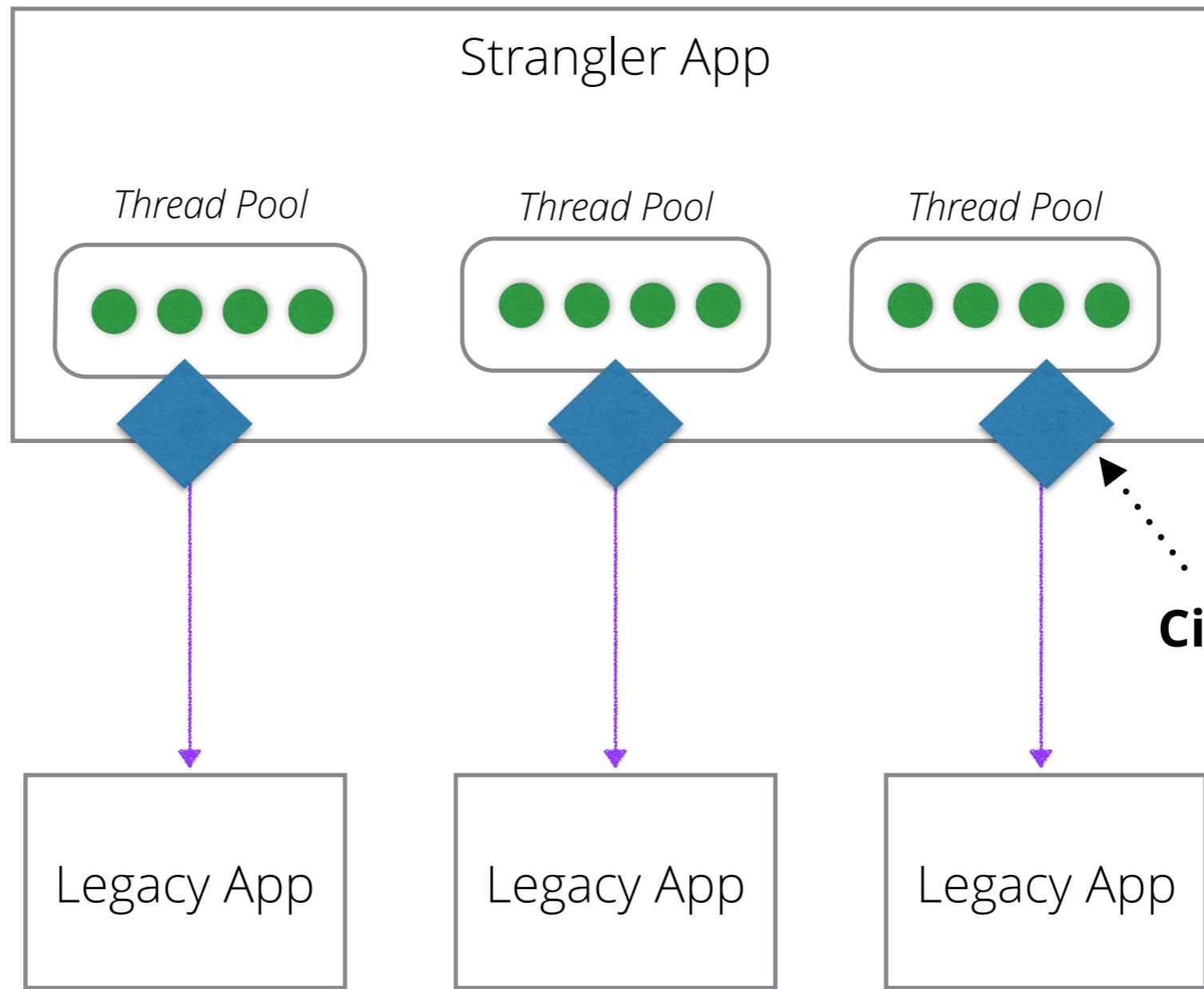
Fix **Timeouts**



Bulkhead
Downstream
Connections

@samnewman

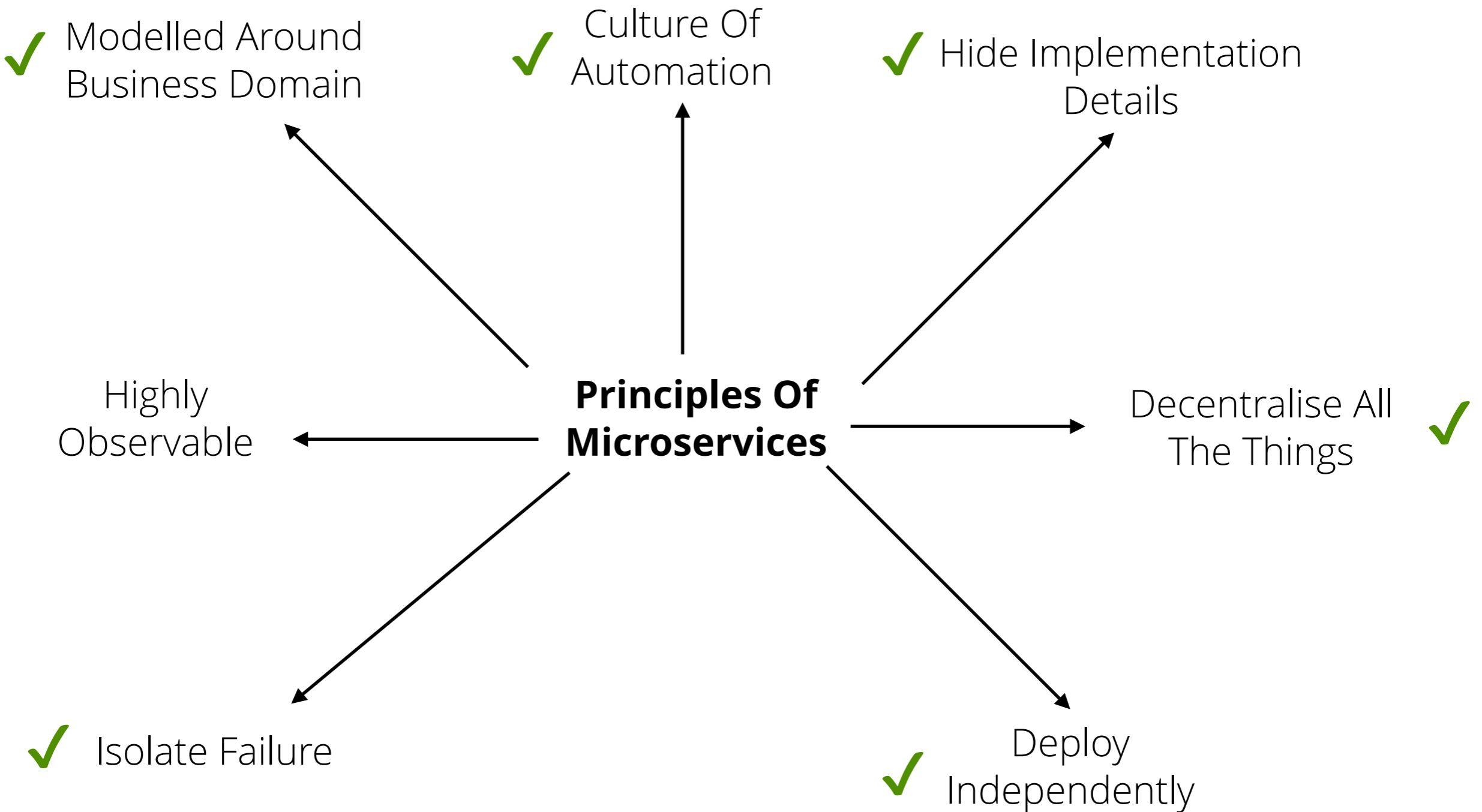
Fix Timeouts

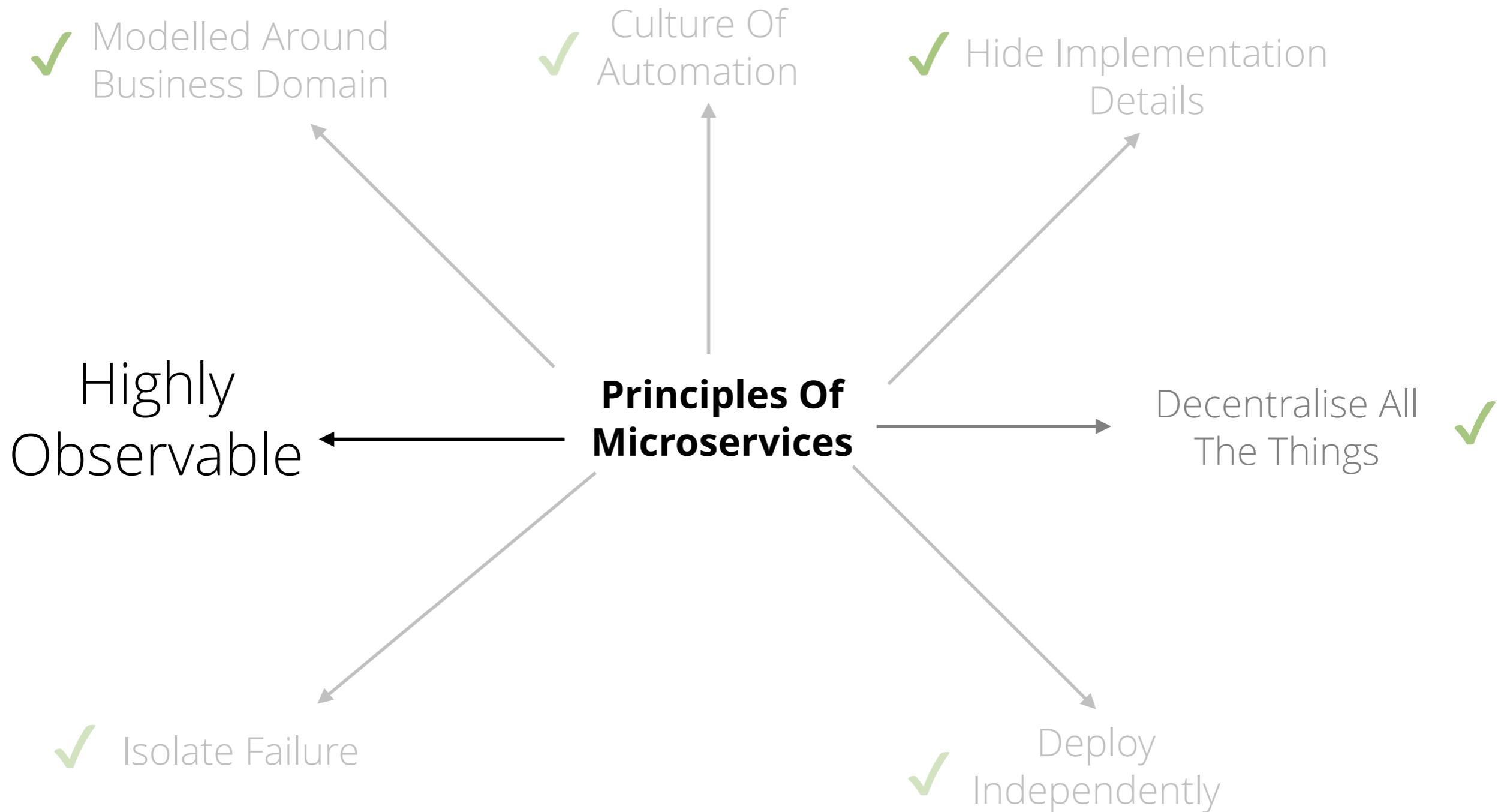


Bulkhead
Downstream
Connections

Circuit Breakers

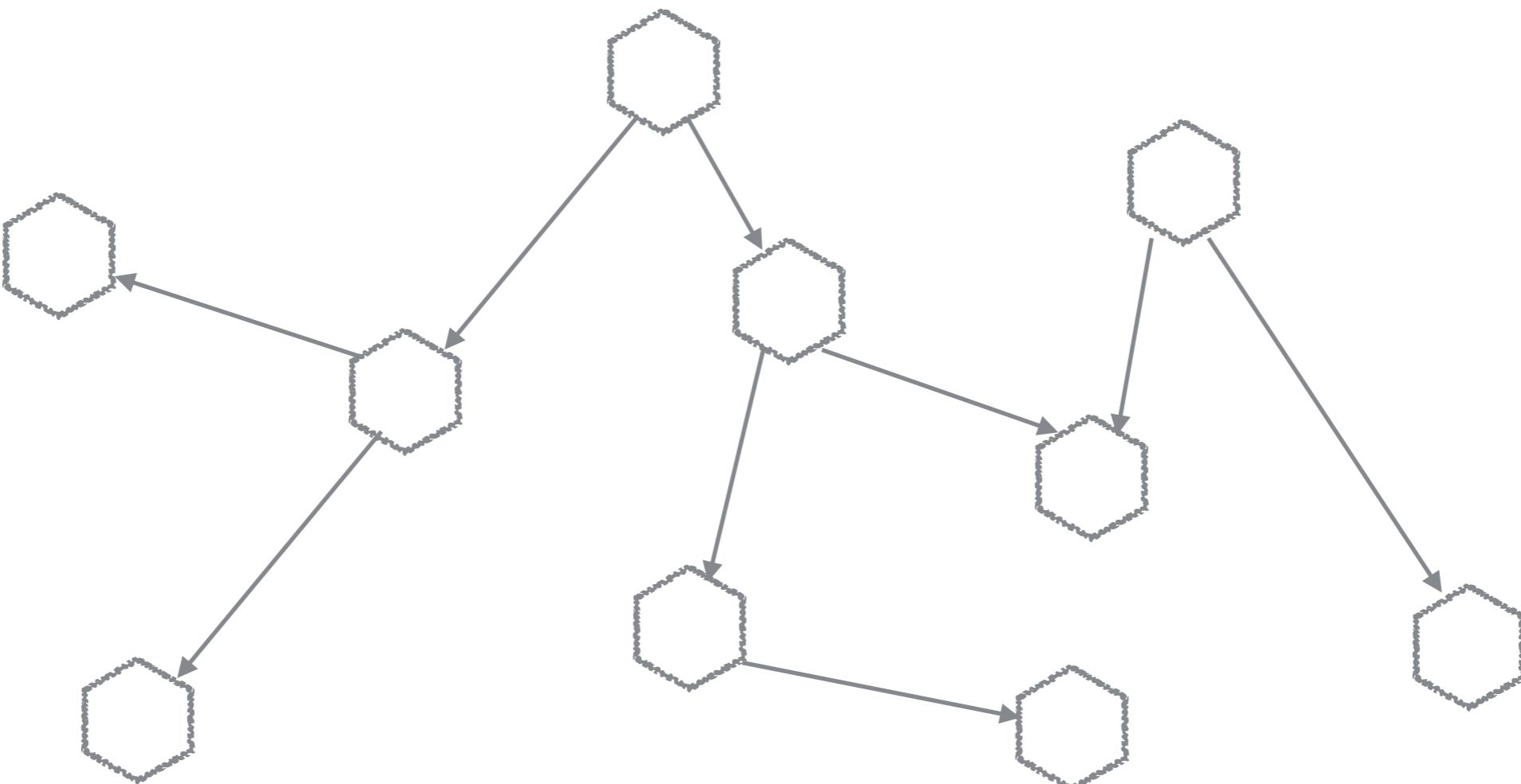
@samnewman







AGGREGATION



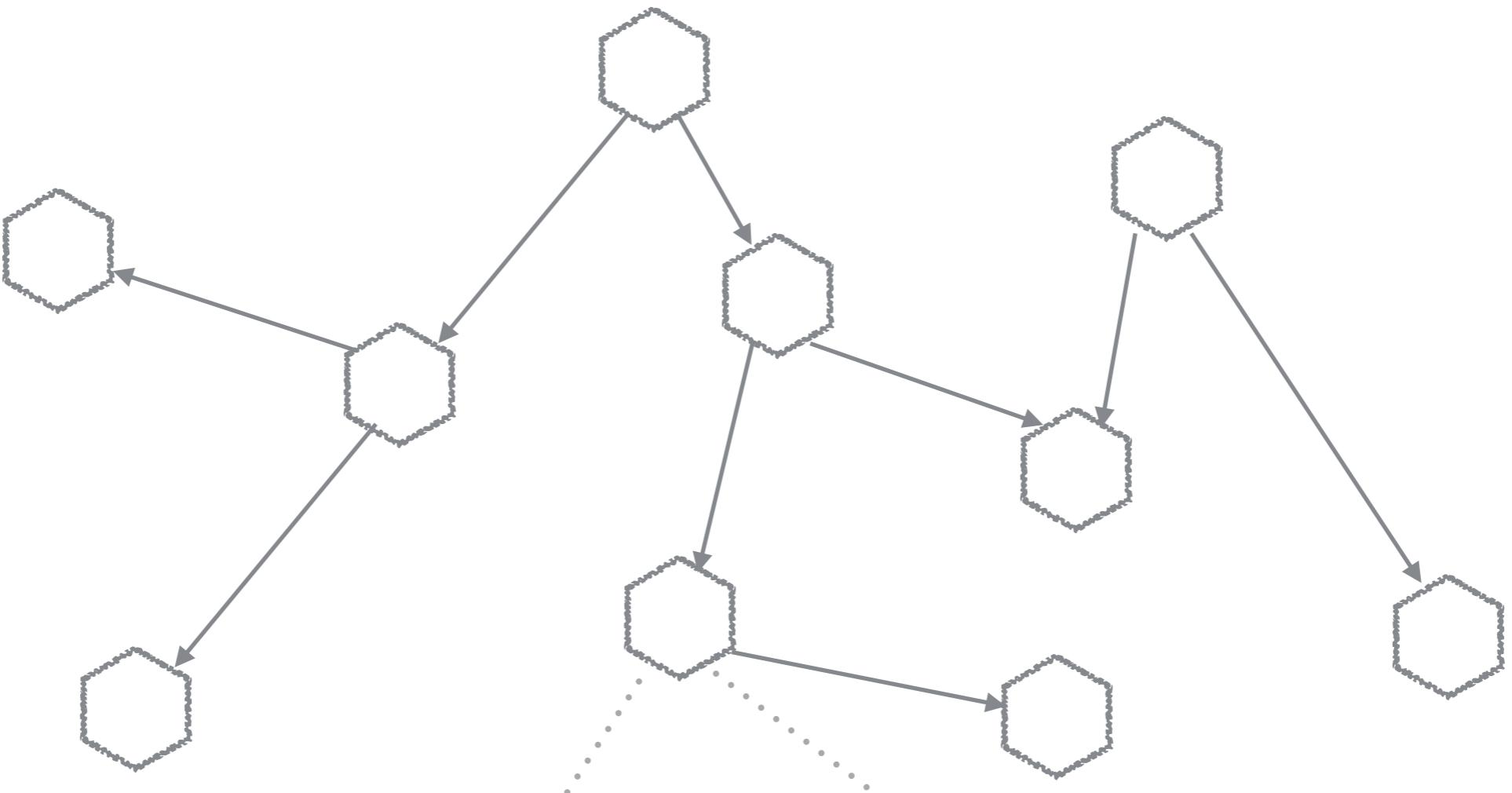
@samnewman

AGGREGATION



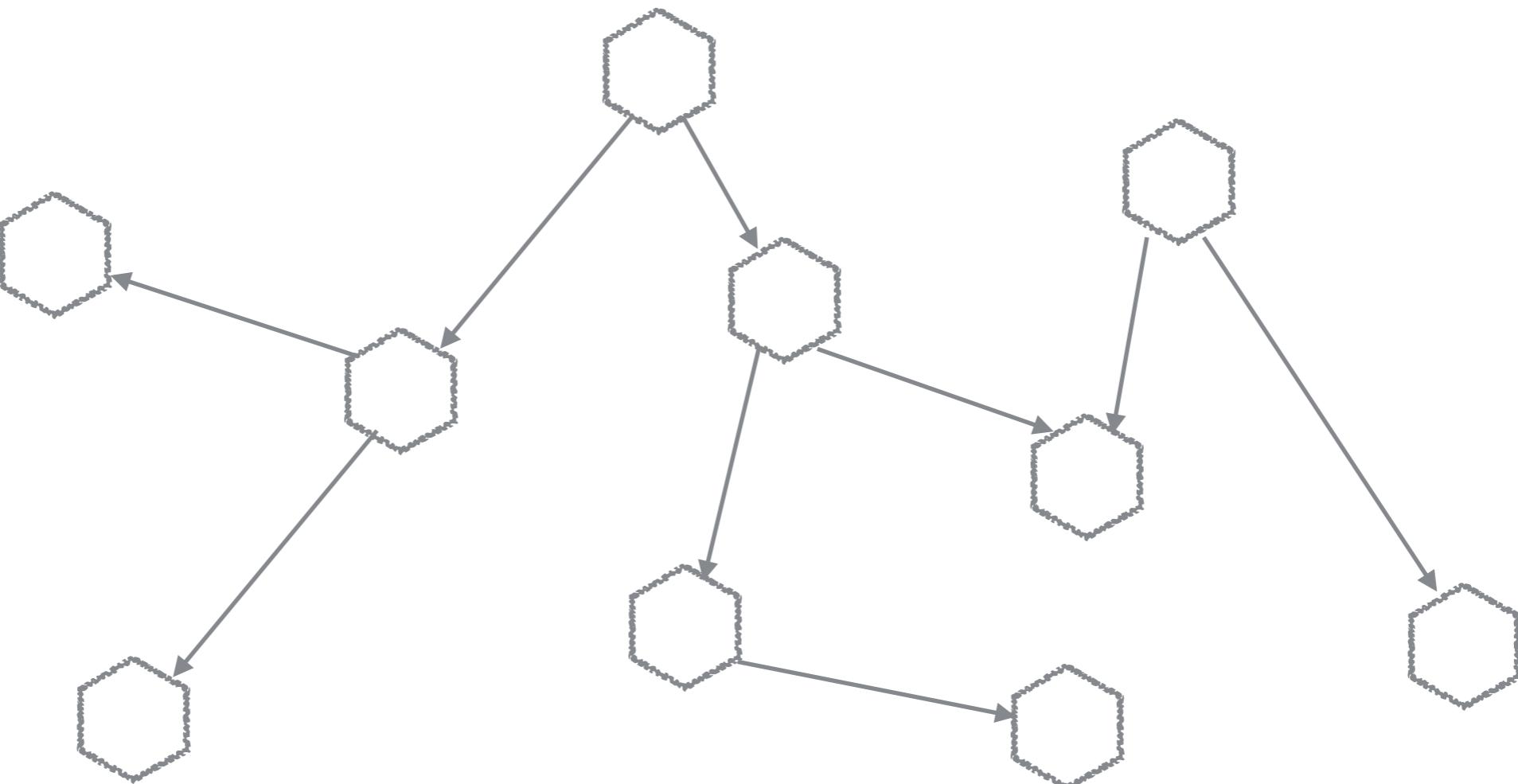
@samnewman

AGGREGATION



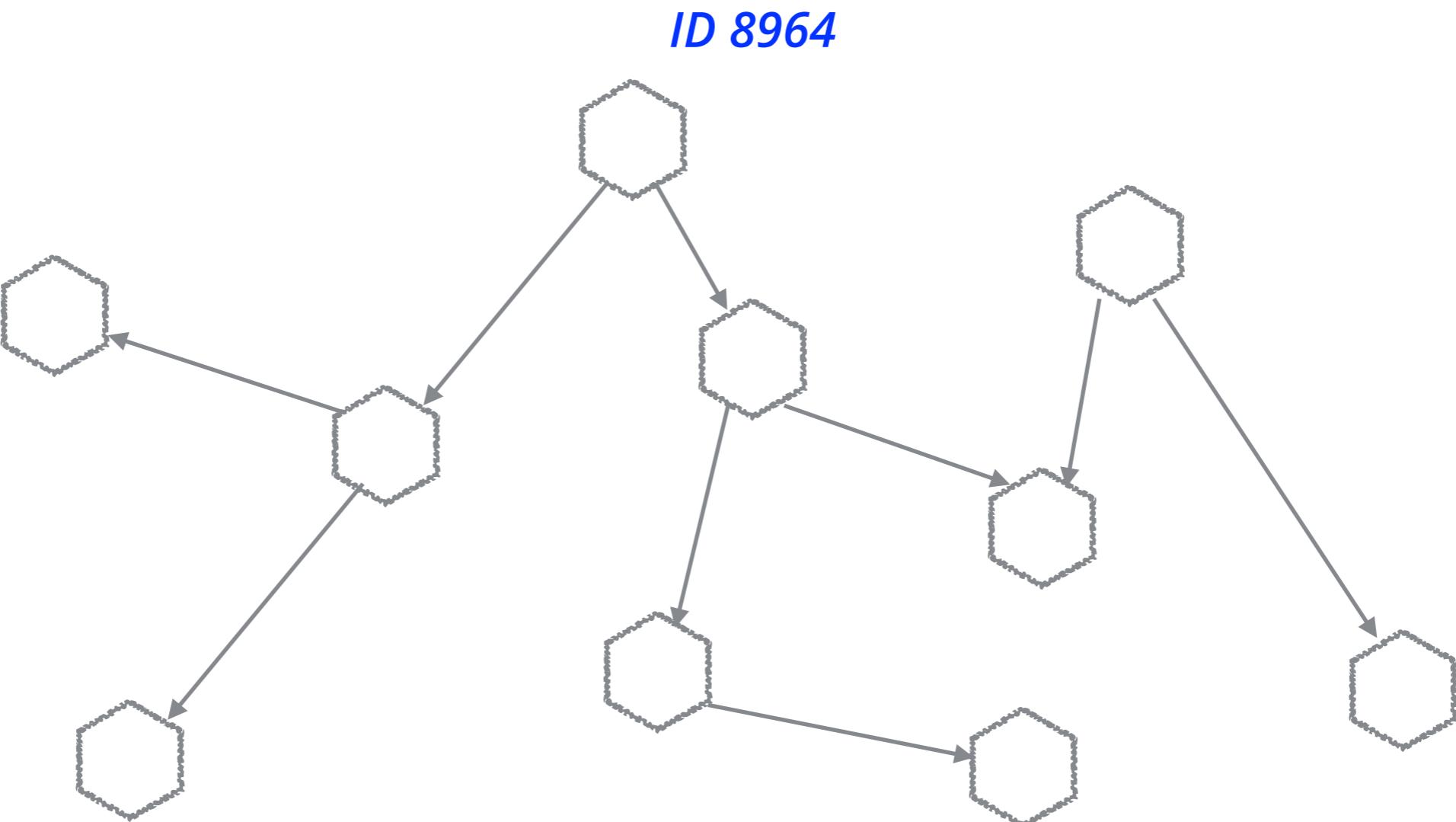
@samnewman

CORRELATION IDS



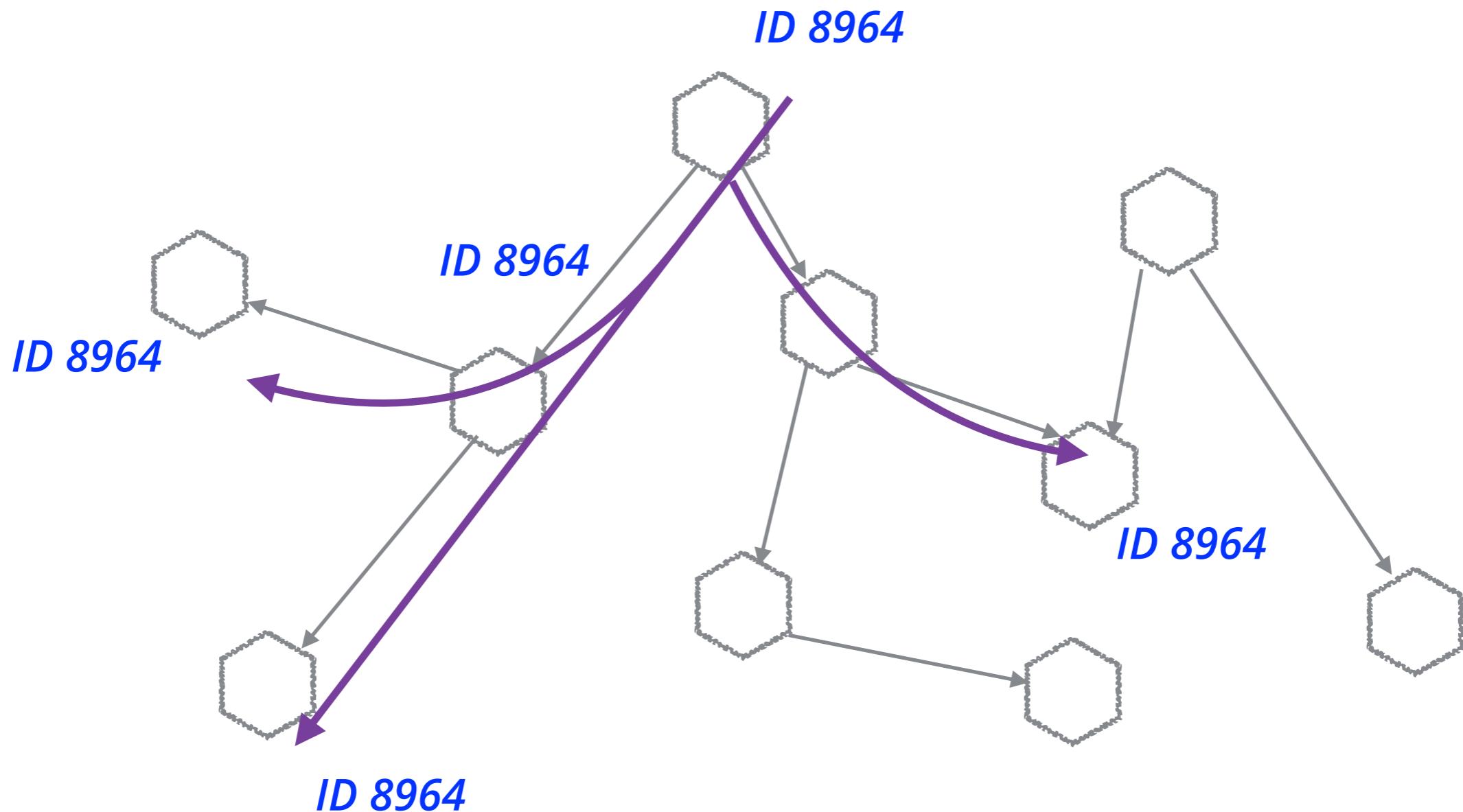
@samnewman

CORRELATION IDS



@samnewman

CORRELATION IDS

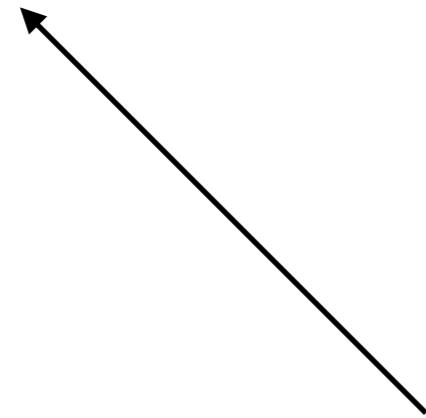


@samnewman

Principles Of Microservices

@samnewman

Modelled Around
Business Domain

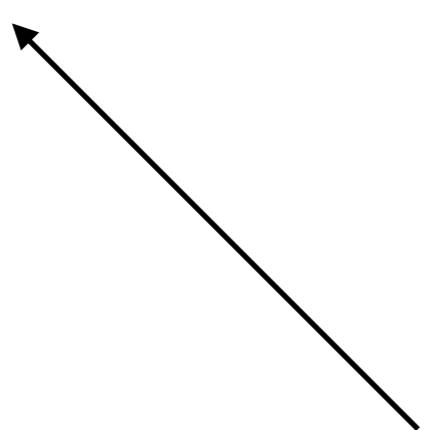


Principles Of Microservices

Modelled Around
Business Domain

Culture Of
Automation

**Principles Of
Microservices**

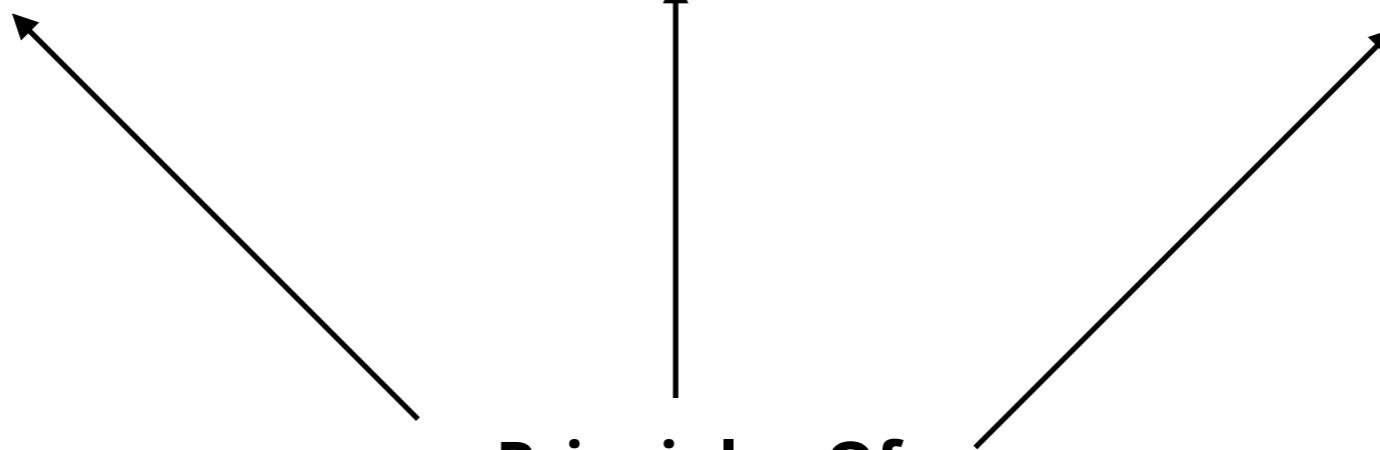


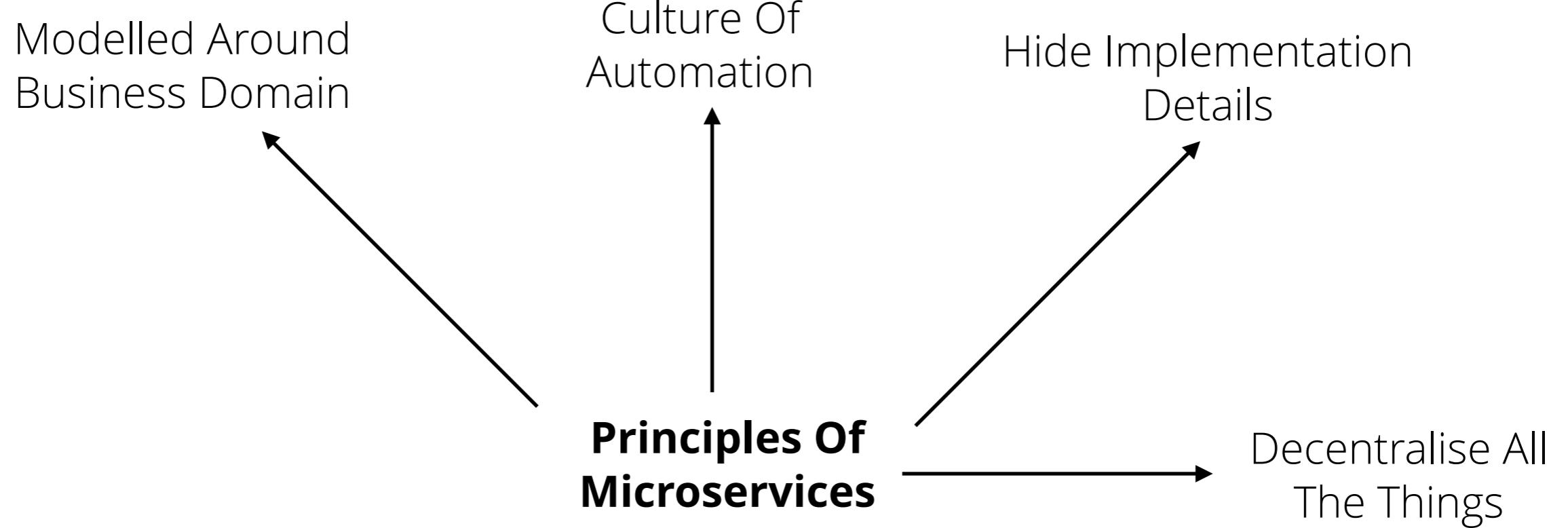
Modelled Around
Business Domain

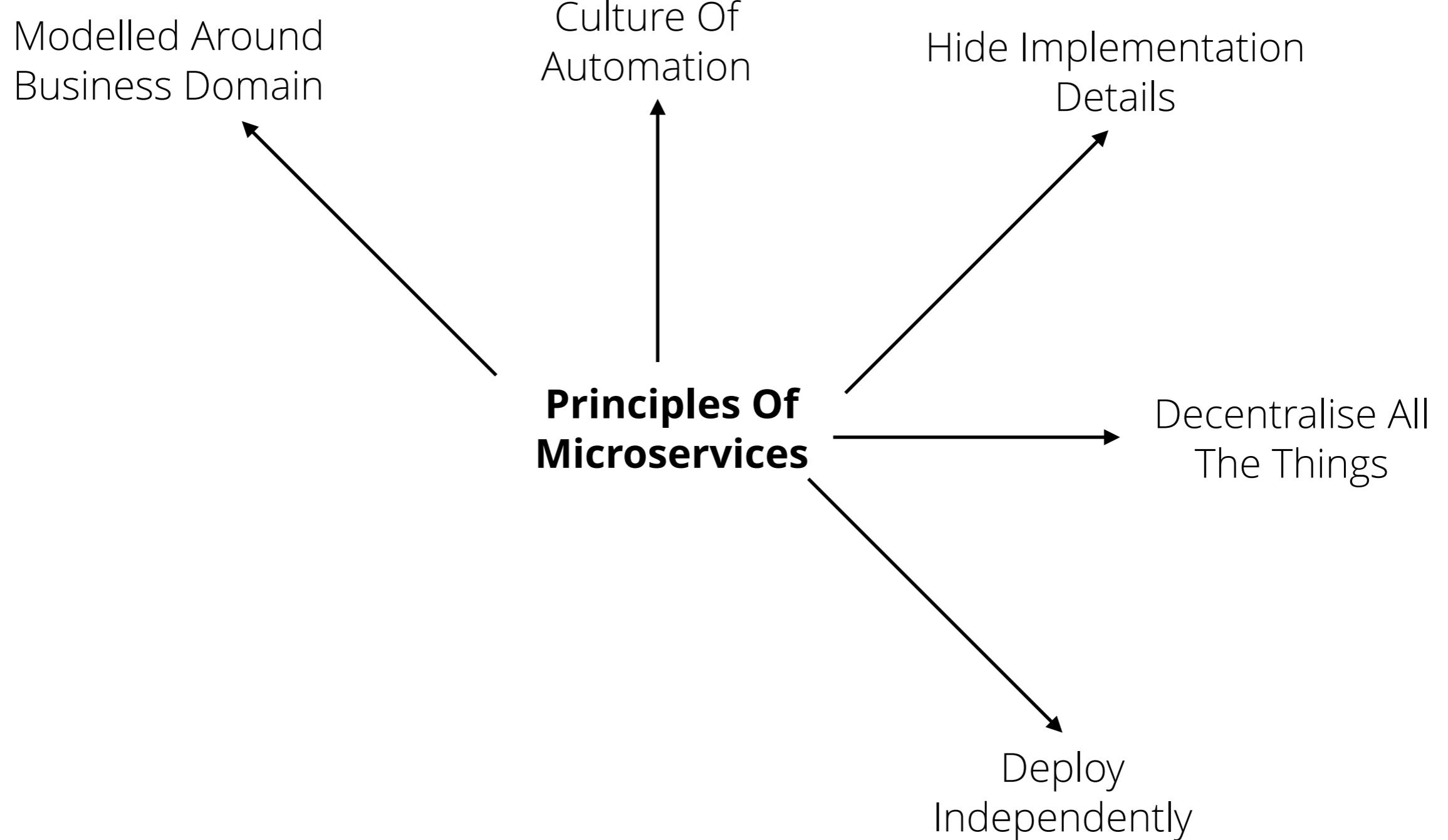
Culture Of
Automation

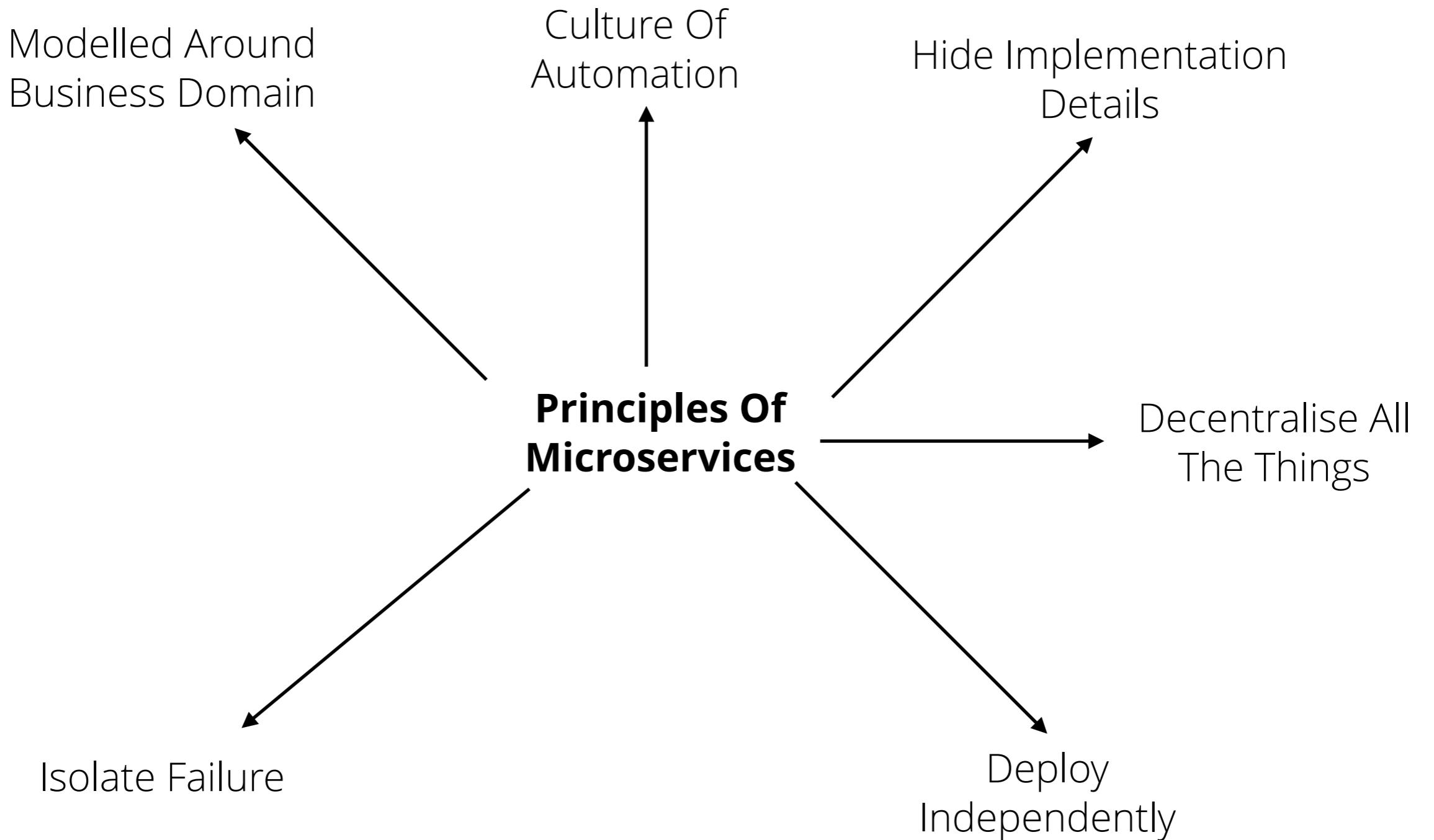
Hide Implementation
Details

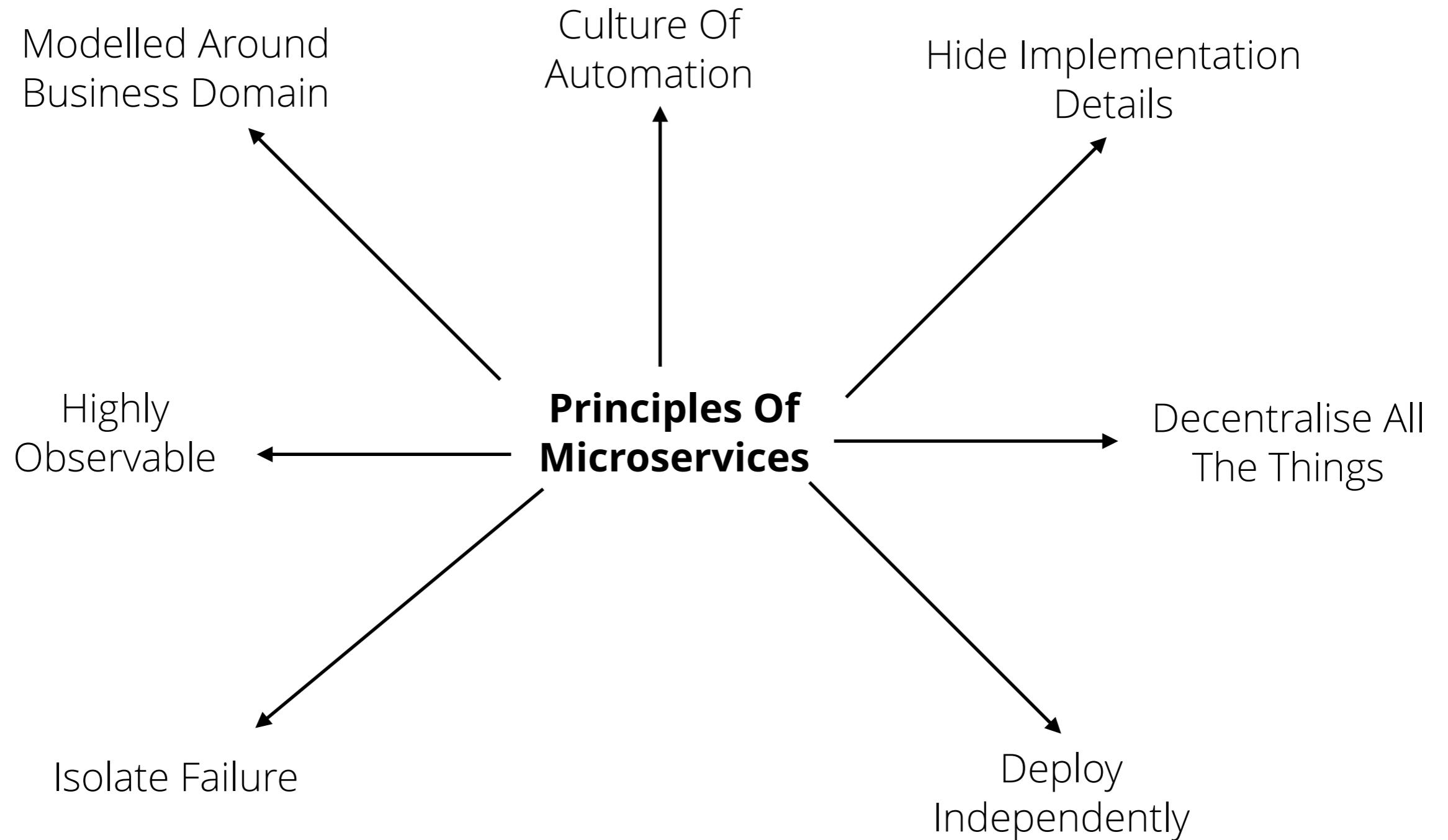
Principles Of Microservices











Natural FEEDBACK MACHINE

1 SELECT TALK



RELEVANCE OF CONTENT

not relevant

PRESERATION

poor

OVERALL IMPRESSION

Leave your feedback
about the presentation
and the speaker

no way



3 SUBMIT FEEDBACK

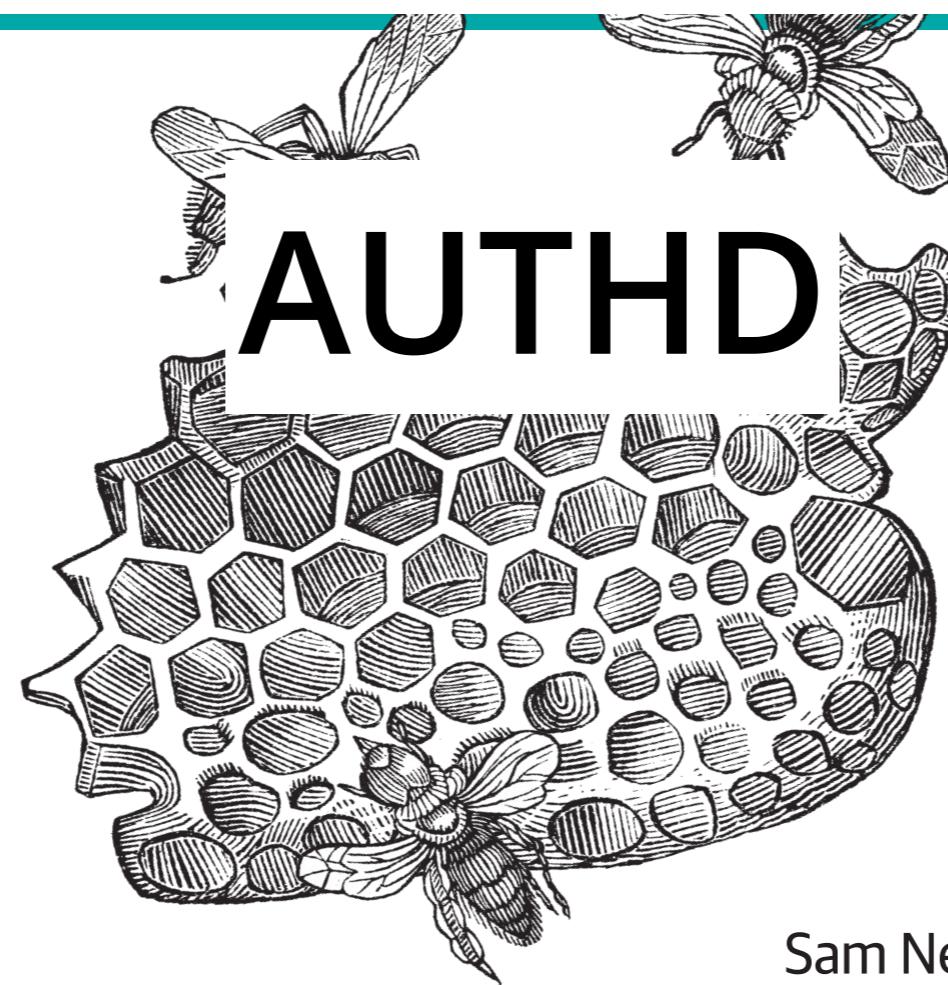
OX

@samnewman

O'REILLY®

Building Microservices

<http://buildingmicroservices.com/>



@samnewman



Sam Newman

Techie at ThoughtWorks, based nominally in Australia. Also food & boardgames. Author of Building Microservices from O'Reilly

 [bio from Twitter](#)

 [Sydney in Australia](#)

Spoken at 24 events in 9 countries



[Edit YOUR PROFILE](#) 

[Summary](#)

[Upcoming events 5](#)

[Past events 33](#)

[People](#)

[Messages](#)



tWorks



tWorks

PRACTICAL CONSIDERATIONS
Practical Considerations For Microservices

PRINCIPLES OF MICROSERVICES
The Principles Of Microservices



Focused around a business domain
Technology Agnostic API
Small

1

Practical Considerations For Microservic...



Deploying And Te Microservices

[@samnewman](#)

24 EVENTS spoken at
1 EVENT involved with

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

@samnewman

THANKS!

Sam Newman
@samnewman

ThoughtWorks®