PRINCIPLES OF MICROSERVICES

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

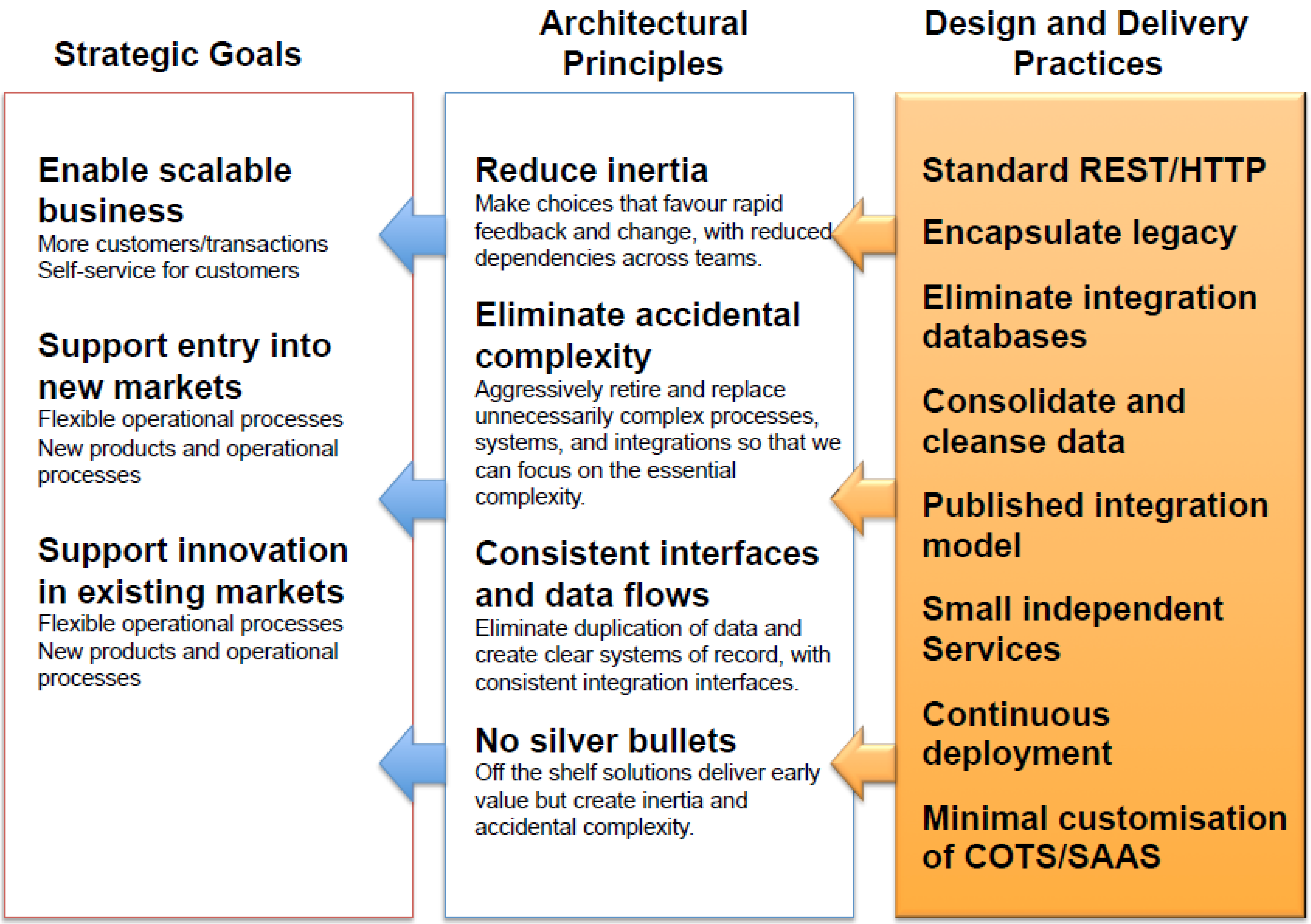
VII. Port binding
Export services via port binding

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



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

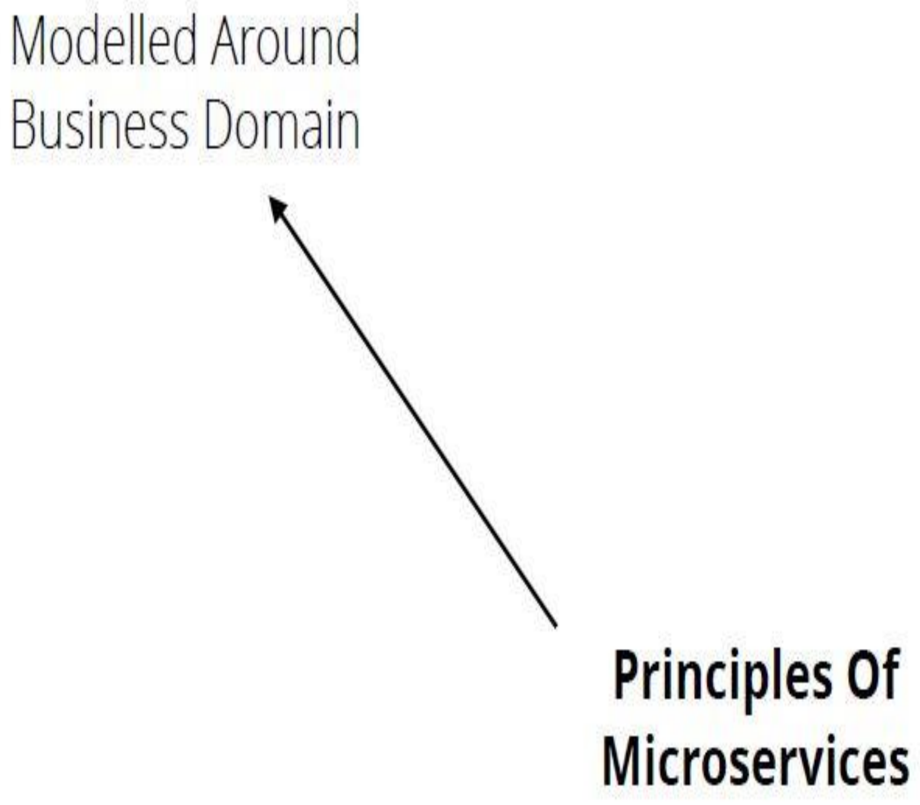
Small independent Services

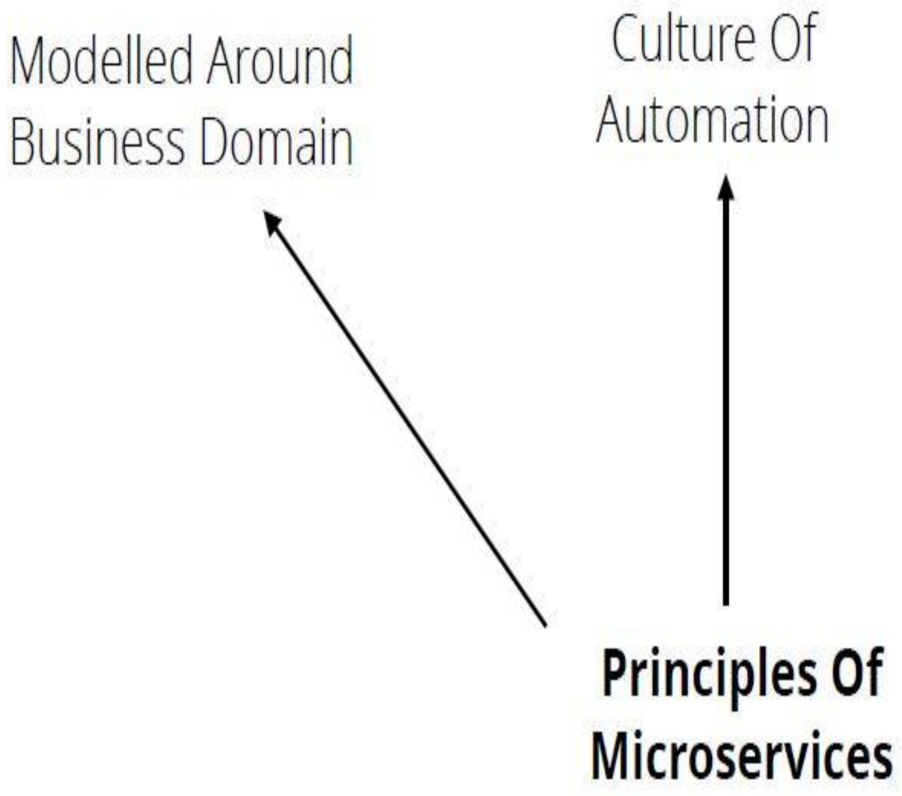
Continuous deployment

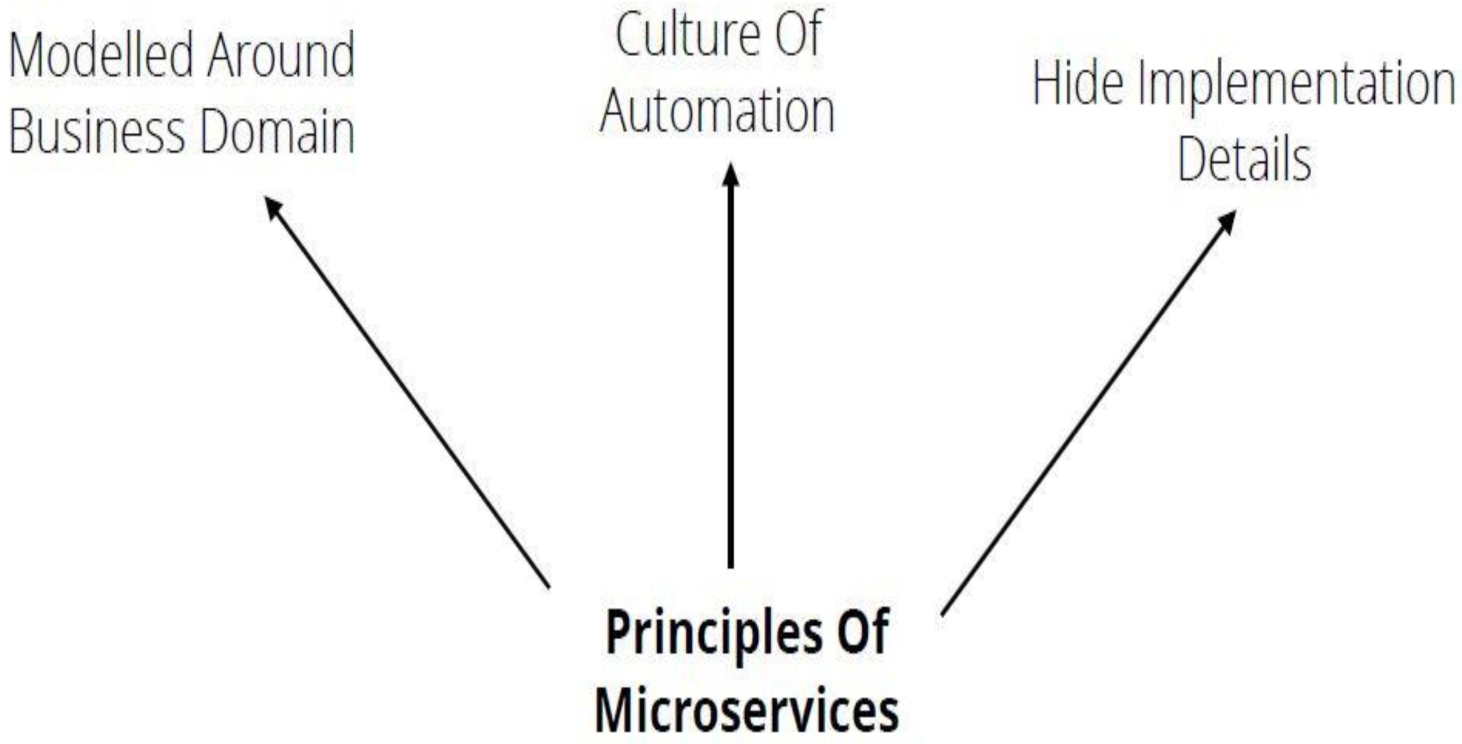
Minimal customisation of COTS/SAAS

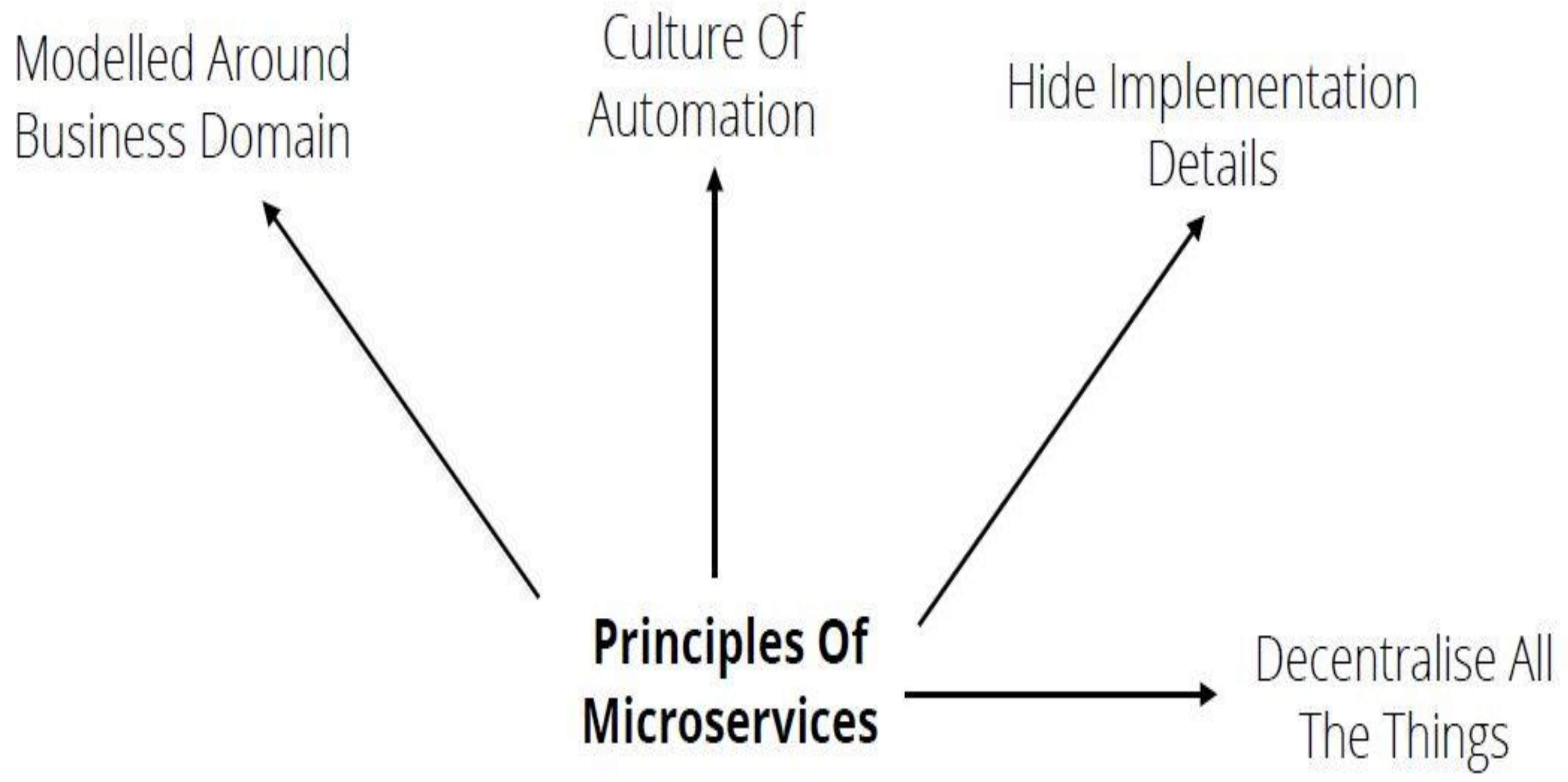
Small ***Autonomous*** services
that ***work together***

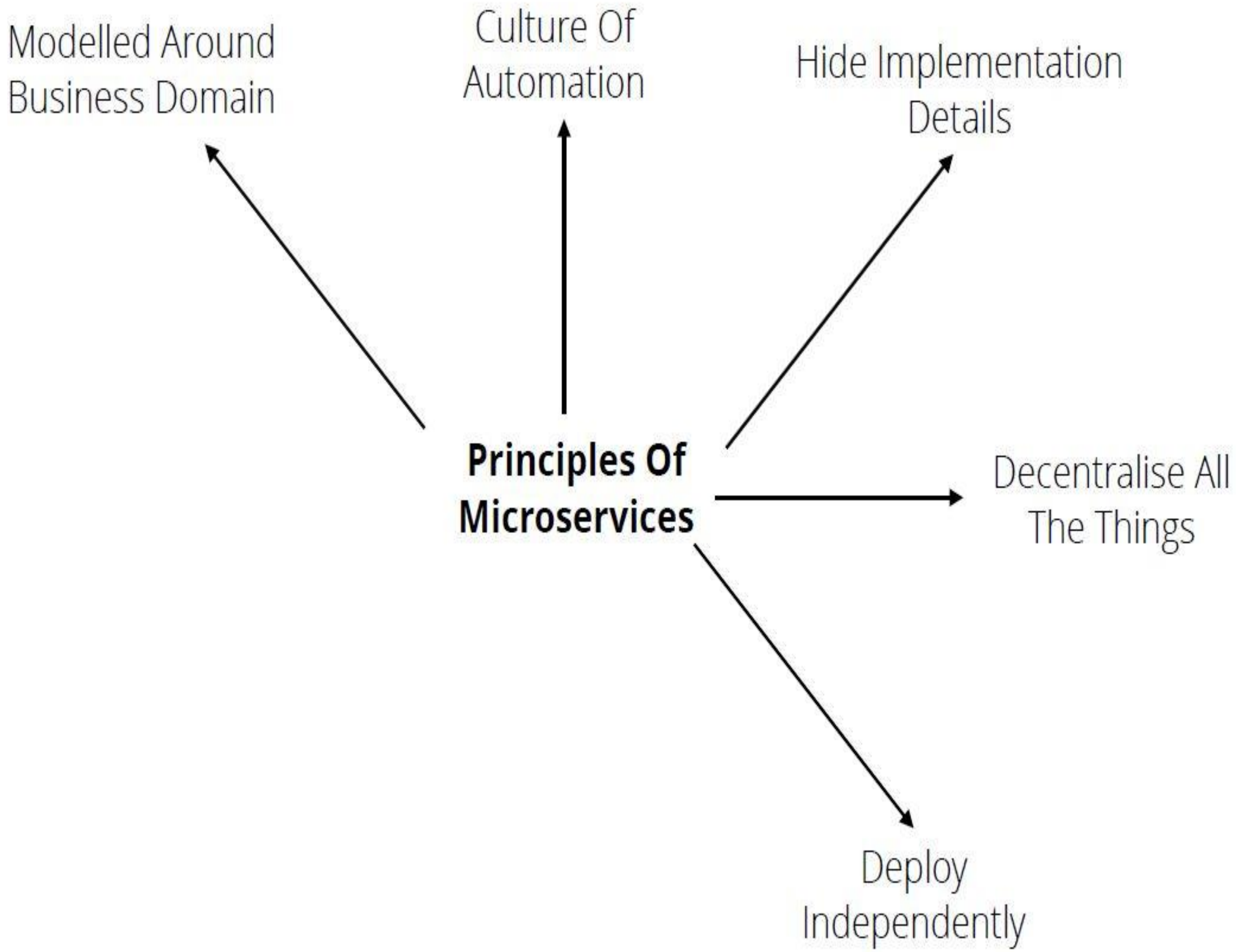
Principles Of Microservices

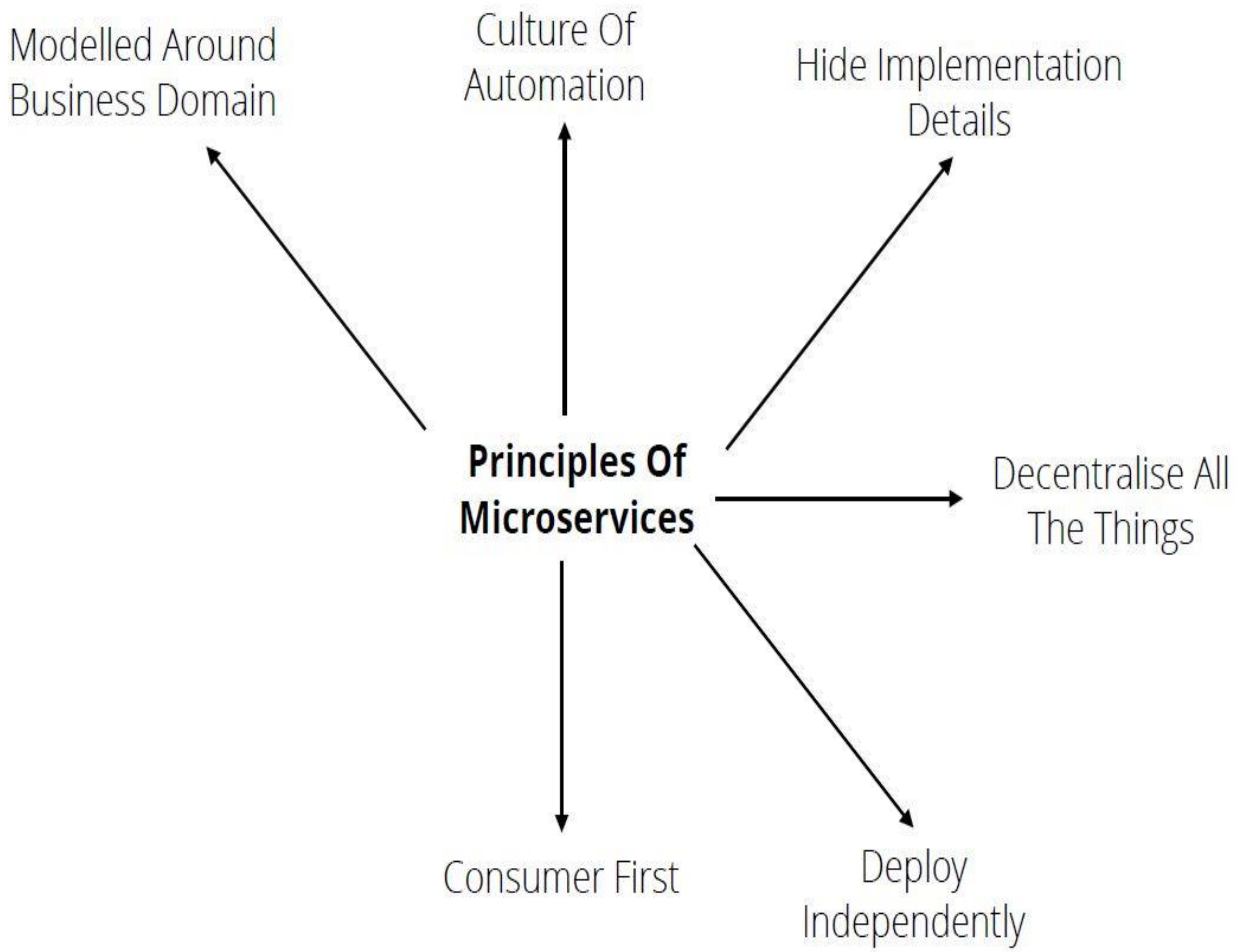

Modelled Around Business Domain
Principles Of Microservices

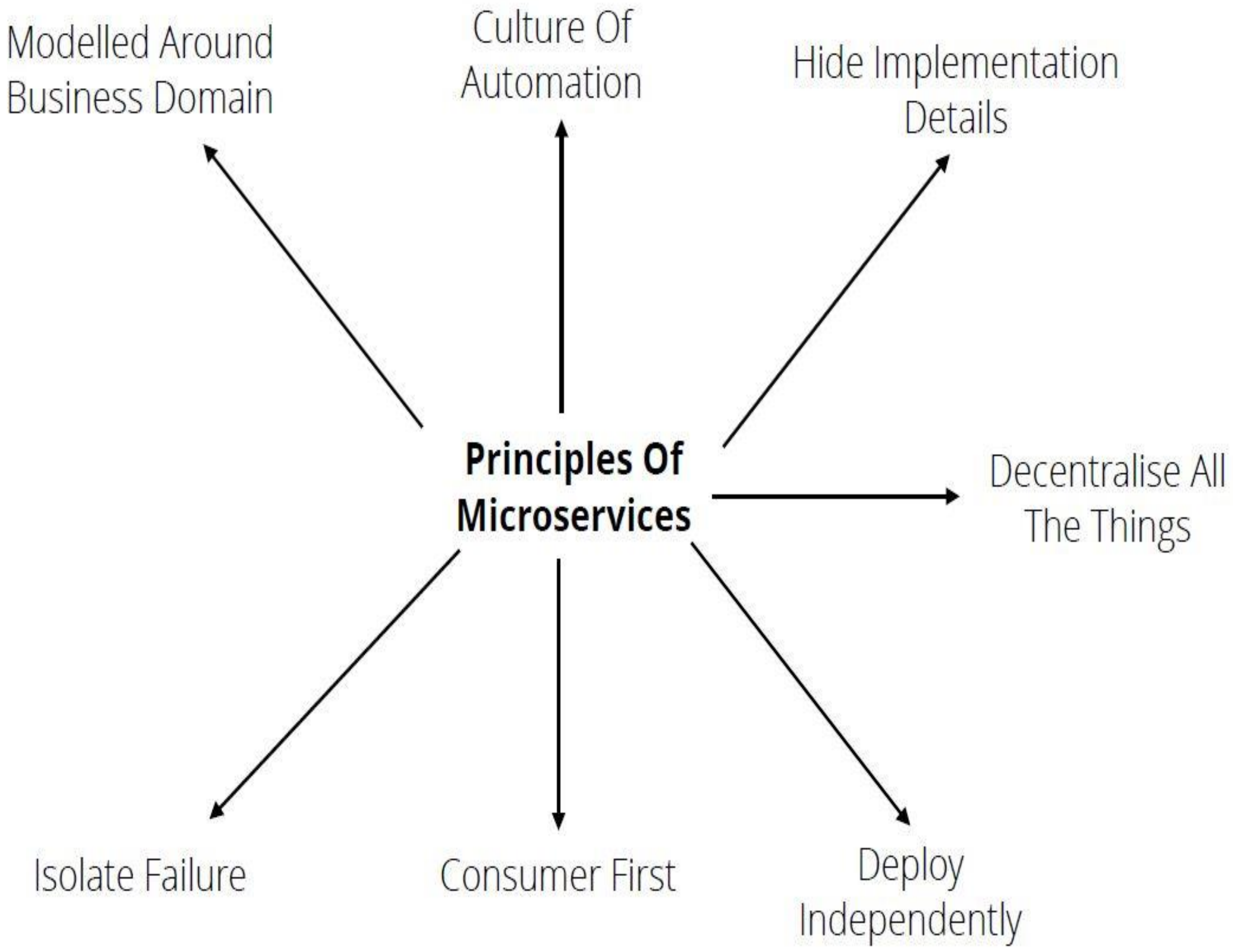

Modelled Around Business Domain
Culture Of Automation
Principles Of Microservices

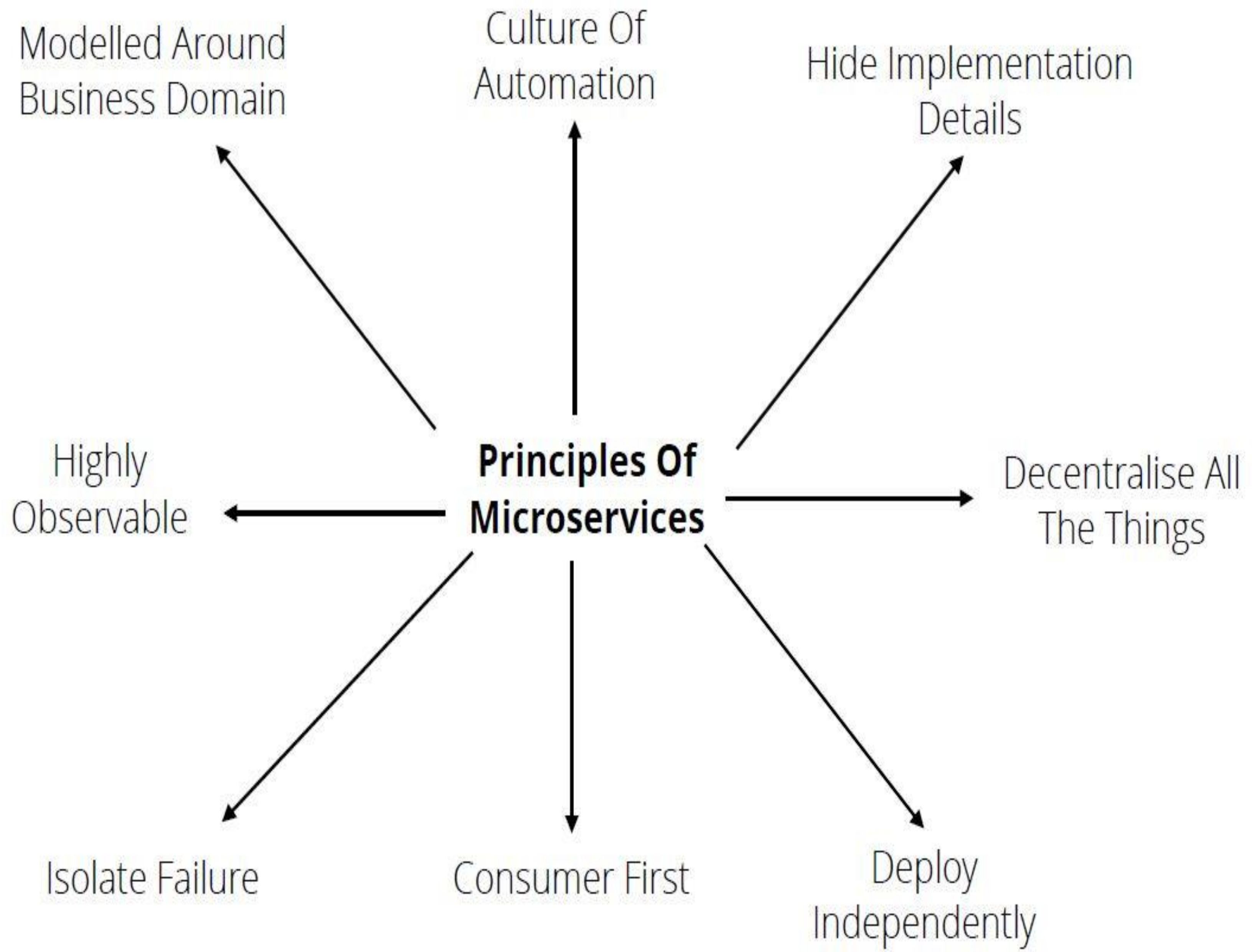

Modelled Around Business Domain
Culture Of Automation
Hide Implementation Details
Principles Of Microservices

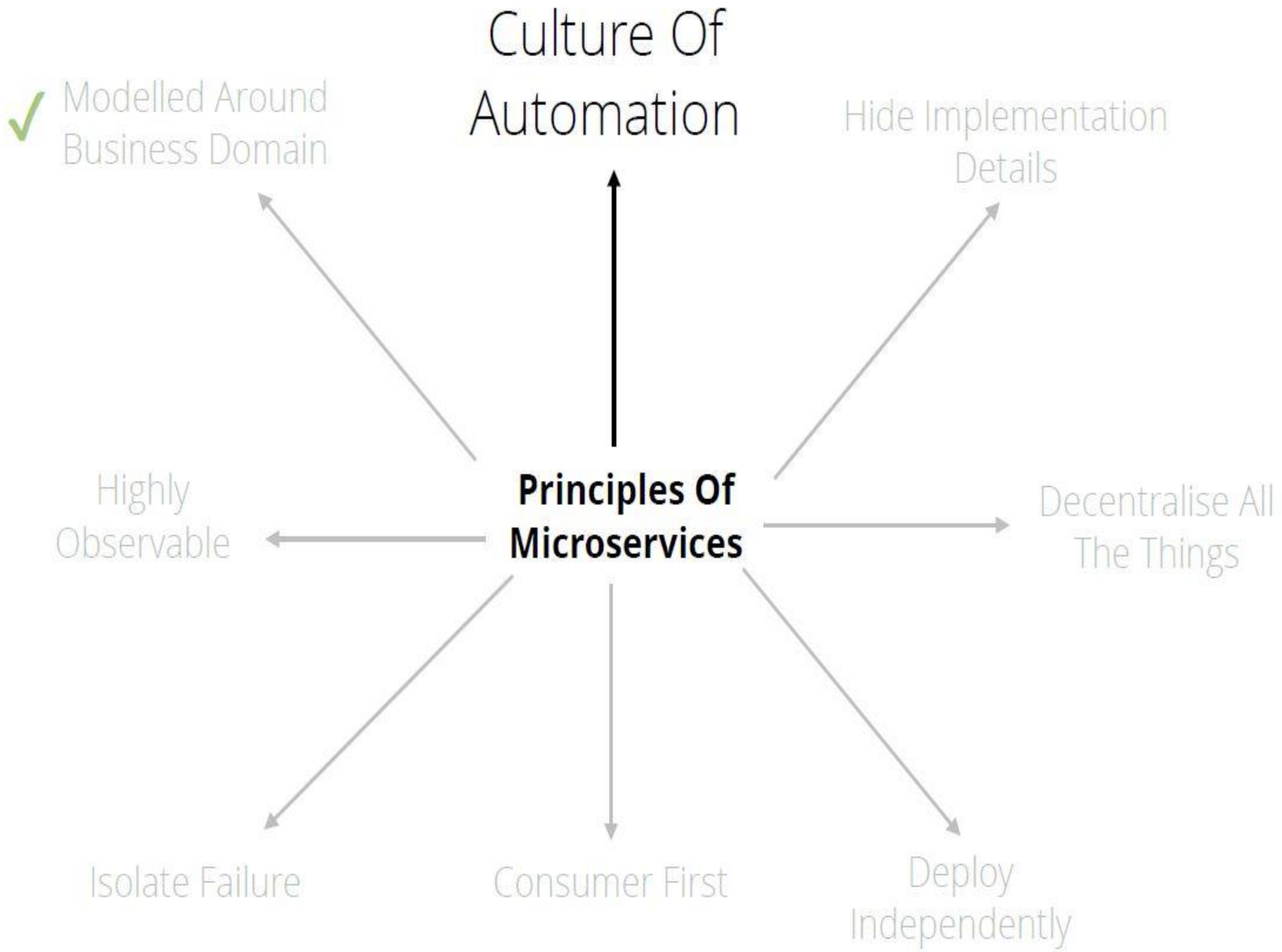

Modelled Around Business Domain
Culture Of Automation
Hide Implementation Details
Principles Of Microservices
Decentralise All The Things


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


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


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


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


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

2 Microservices

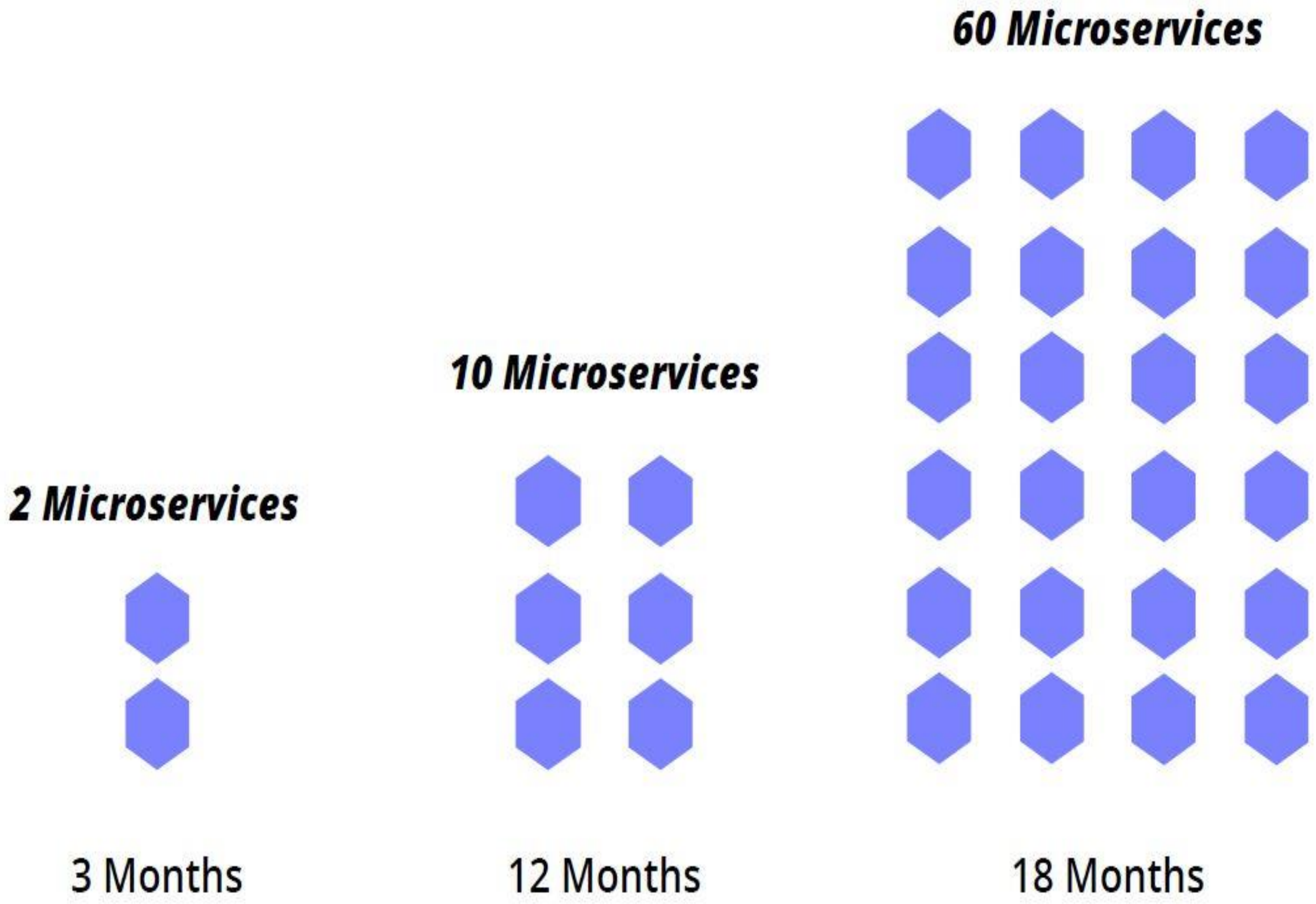
3 Months

10 Microservices

2 Microservices

3 Months

12 Months


2 Microservices
10 Microservices
60 Microservices
3 Months
12 Months
18 Months

Infrastructure Automation

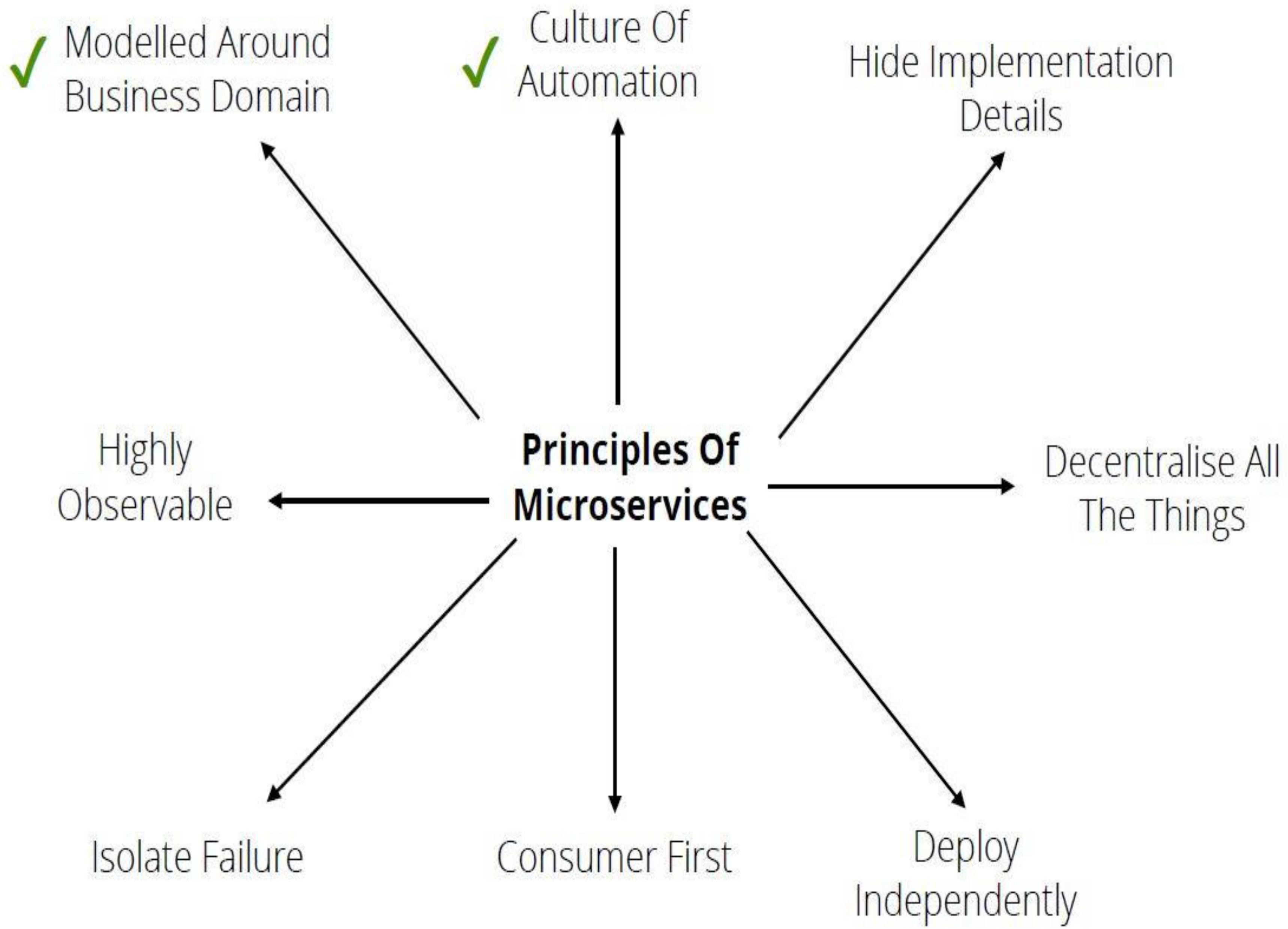
Infrastructure Automation

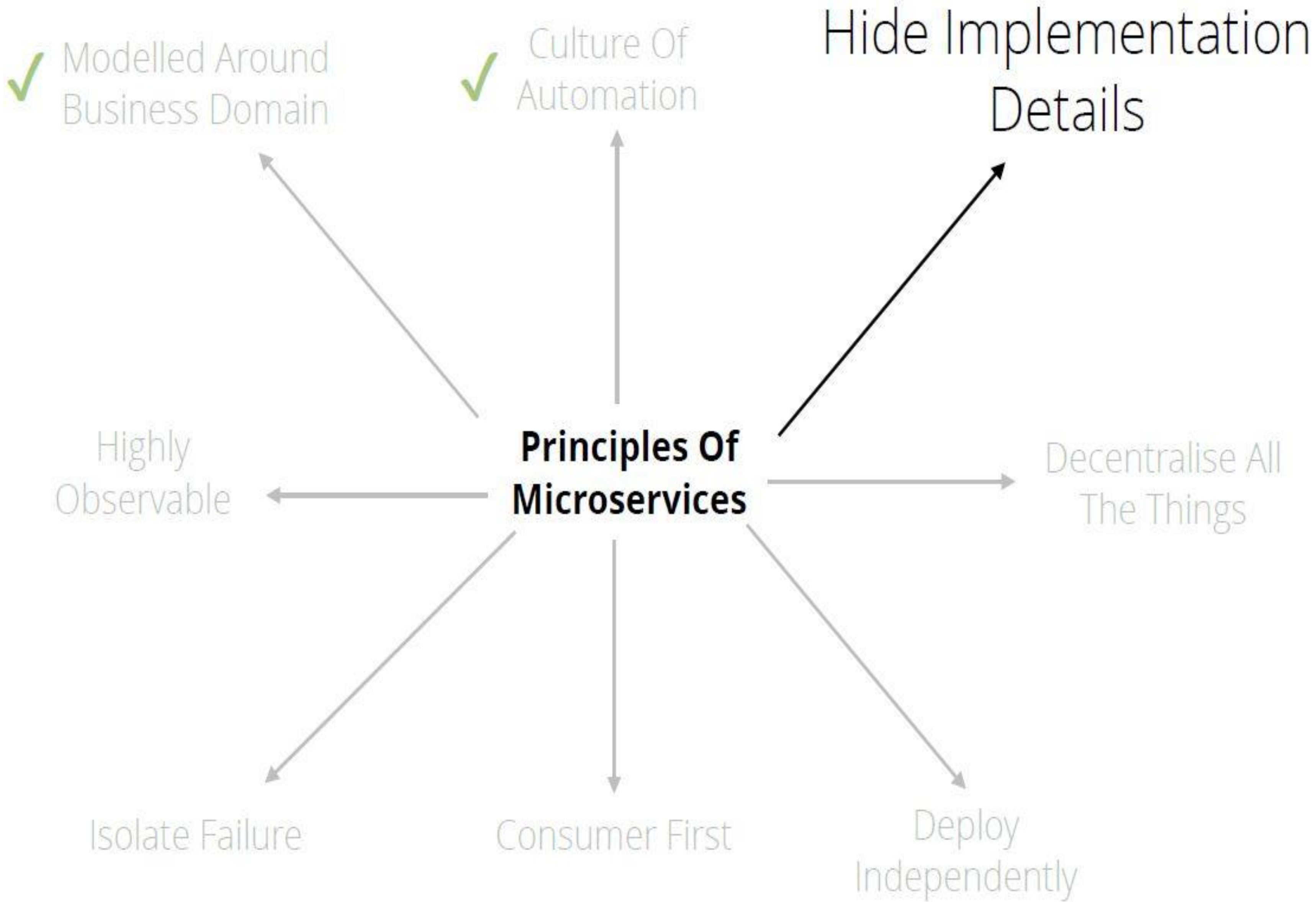
Automated Testing

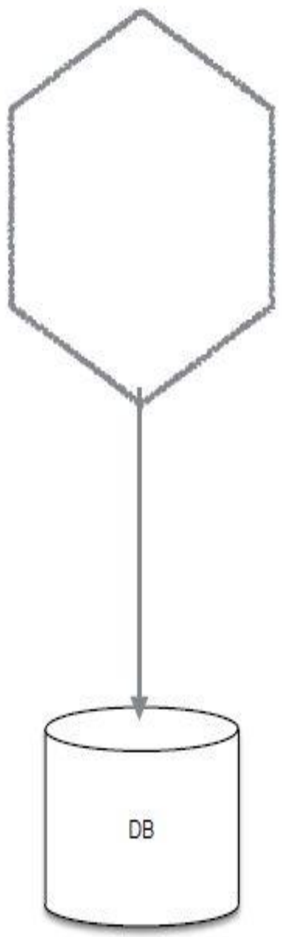
Infrastructure Automation

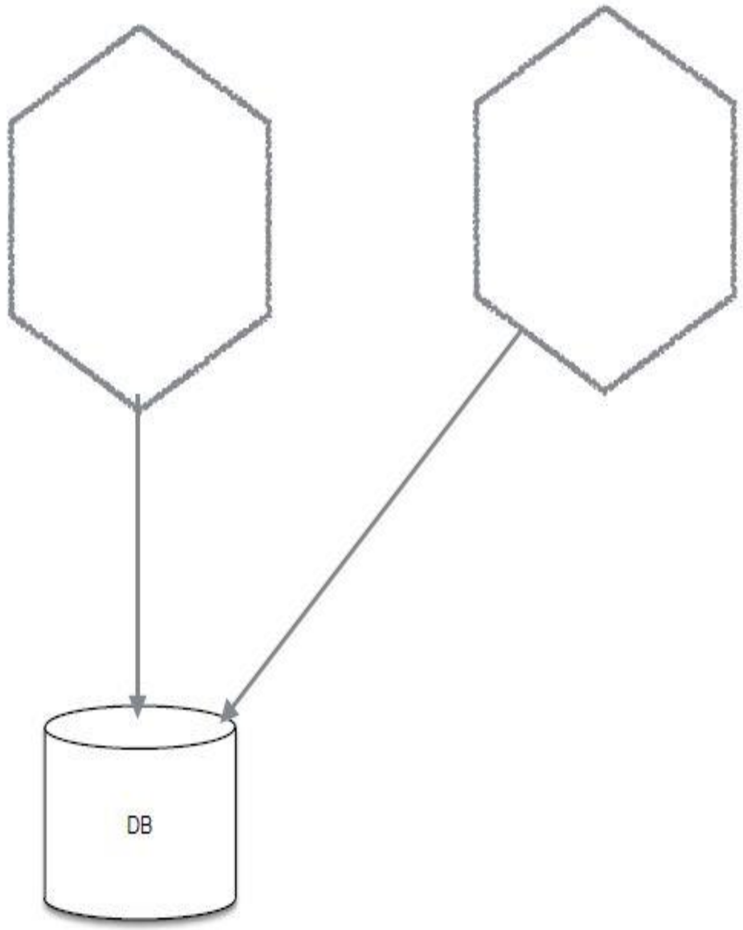
Automated Testing

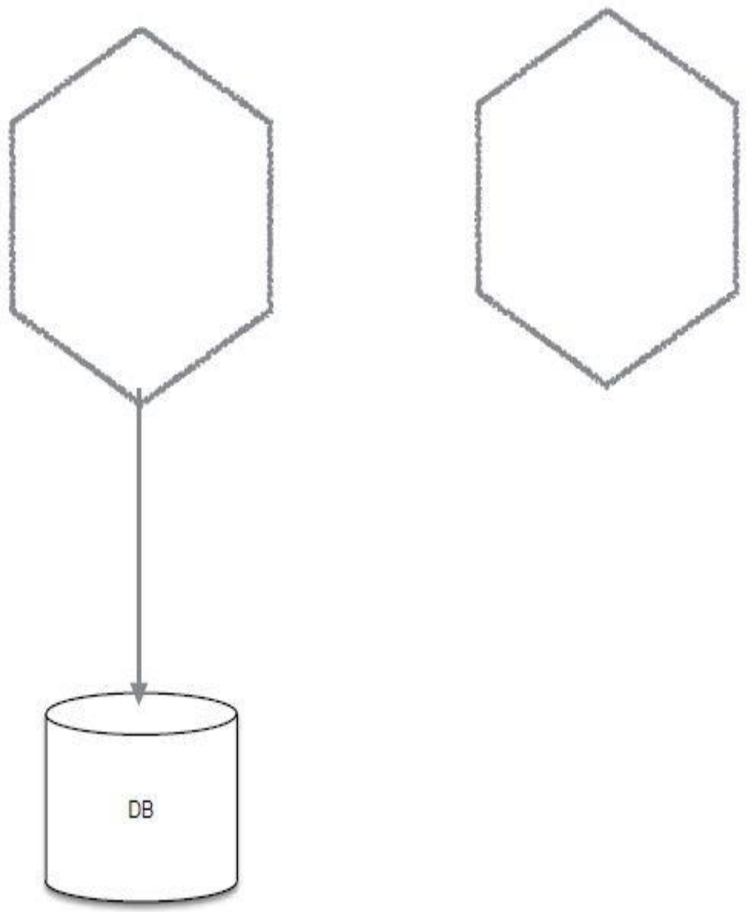
Continuous Delivery

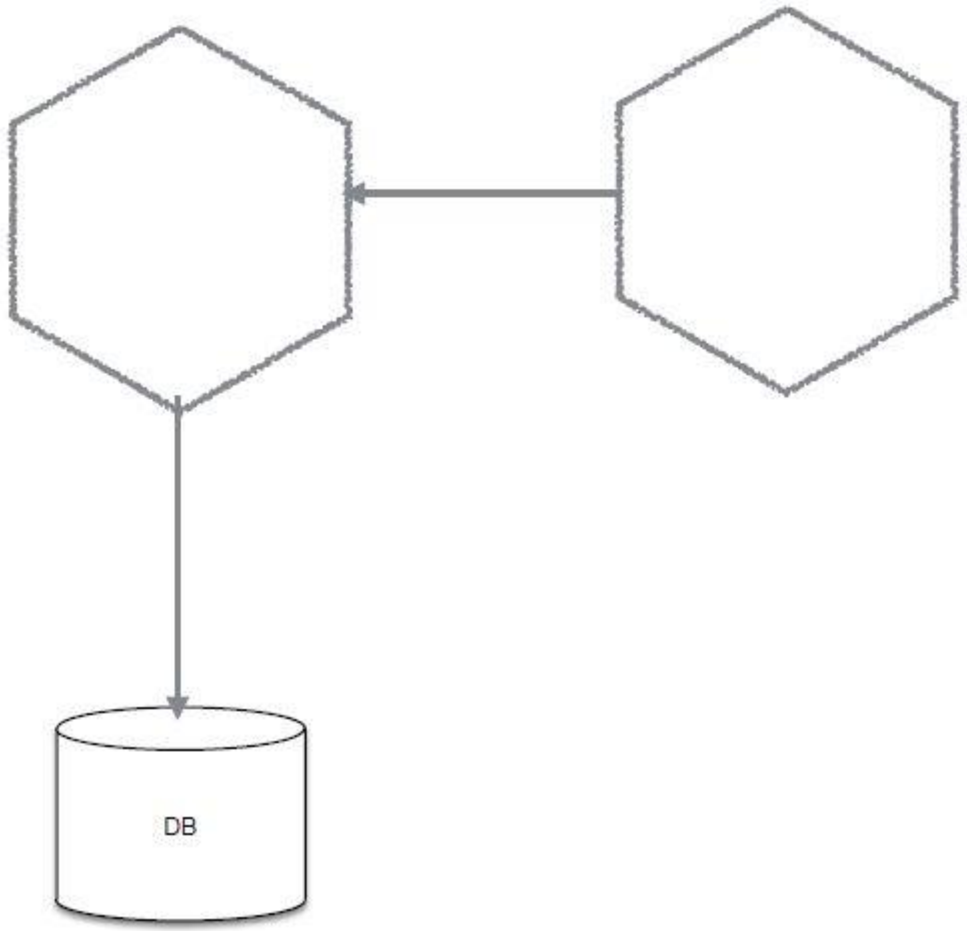

✓ Modelled Around Business Domain
✓ Culture Of Automation
Hide Implementation Details
Highly Observable
Principles Of Microservices
Decentralise All The Things
Isolate Failure
Consumer First
Deploy Independently


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

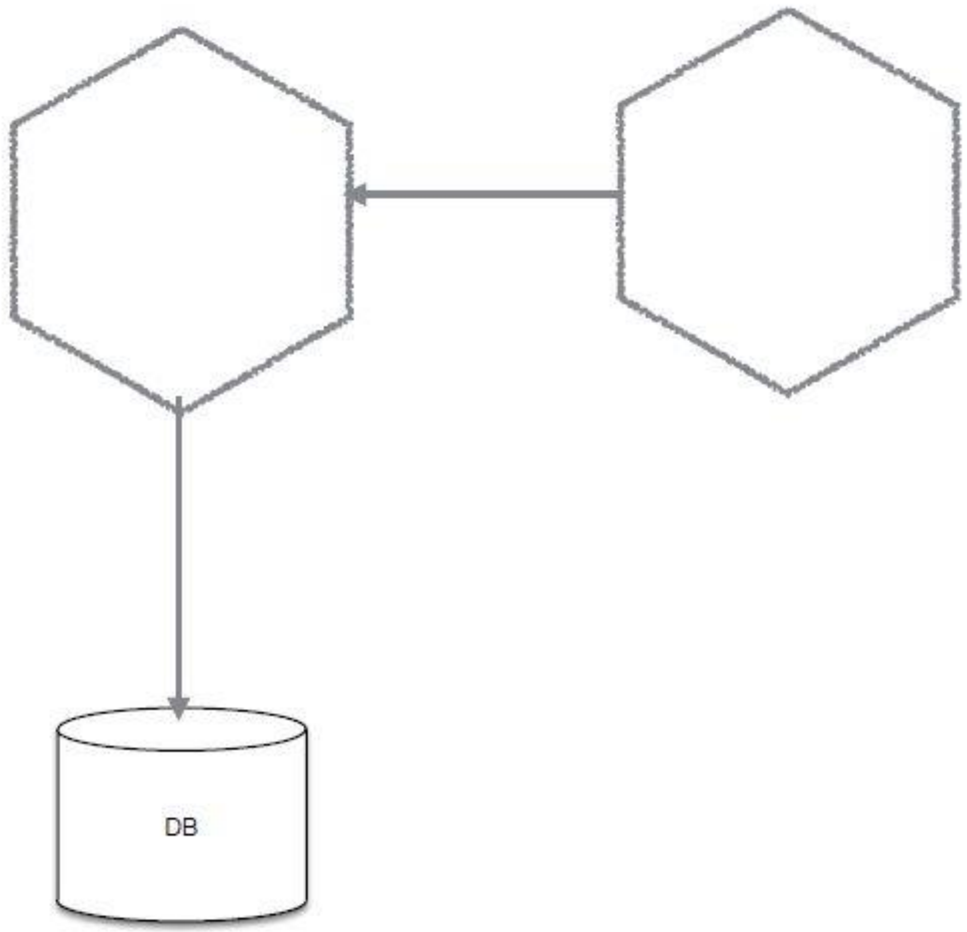

DB


DB

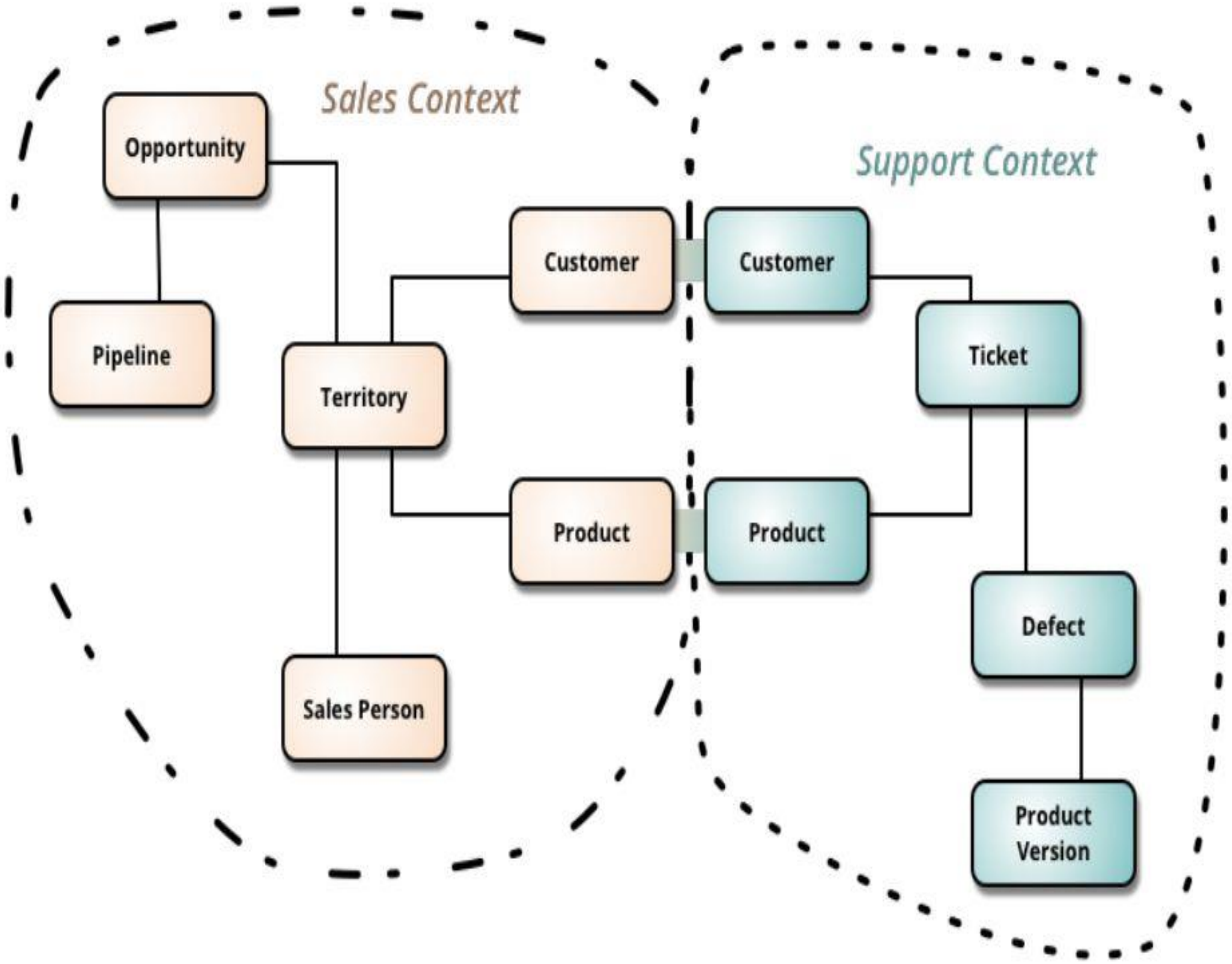

DB


DB

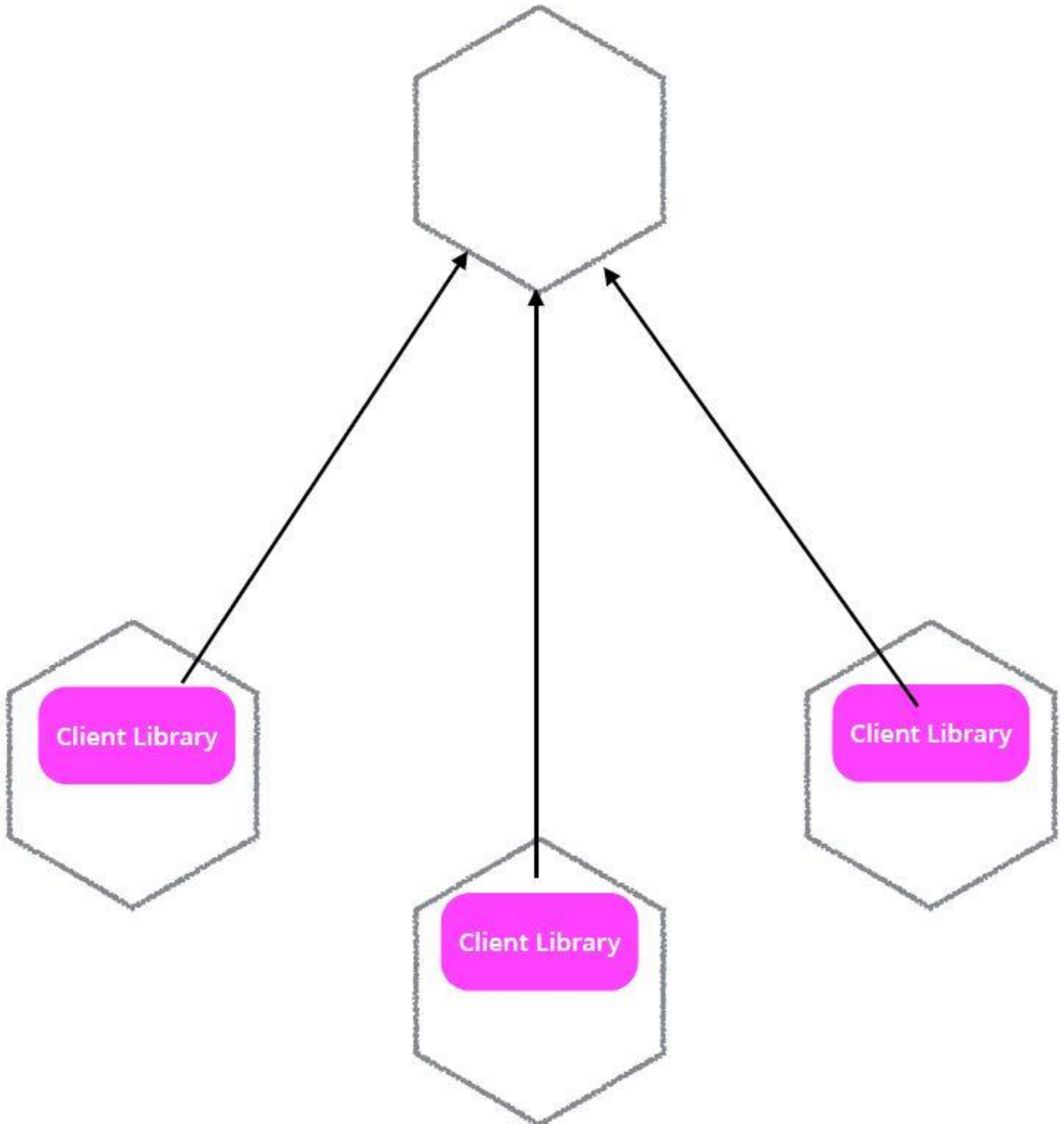
HIDE YOUR DATABASE

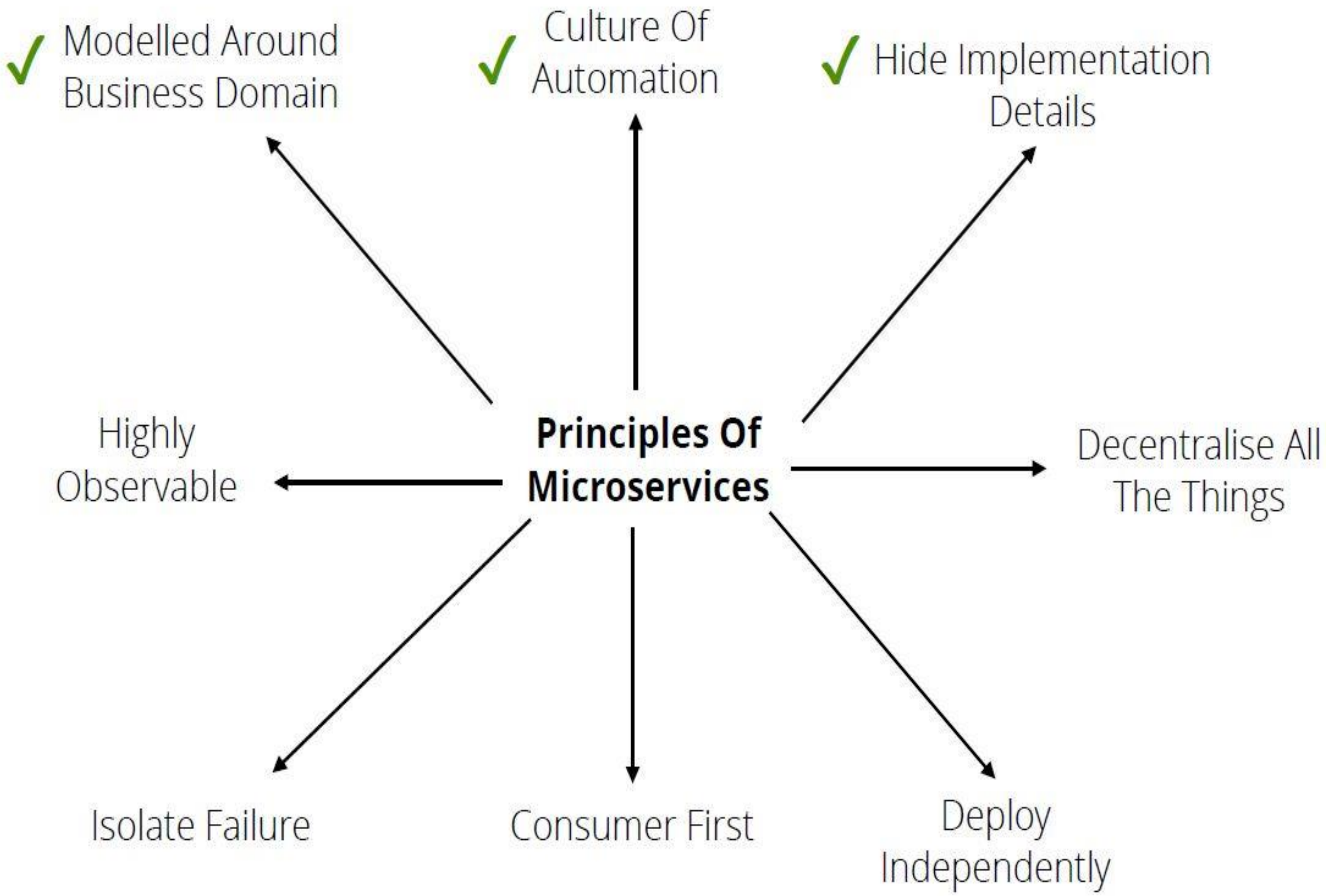


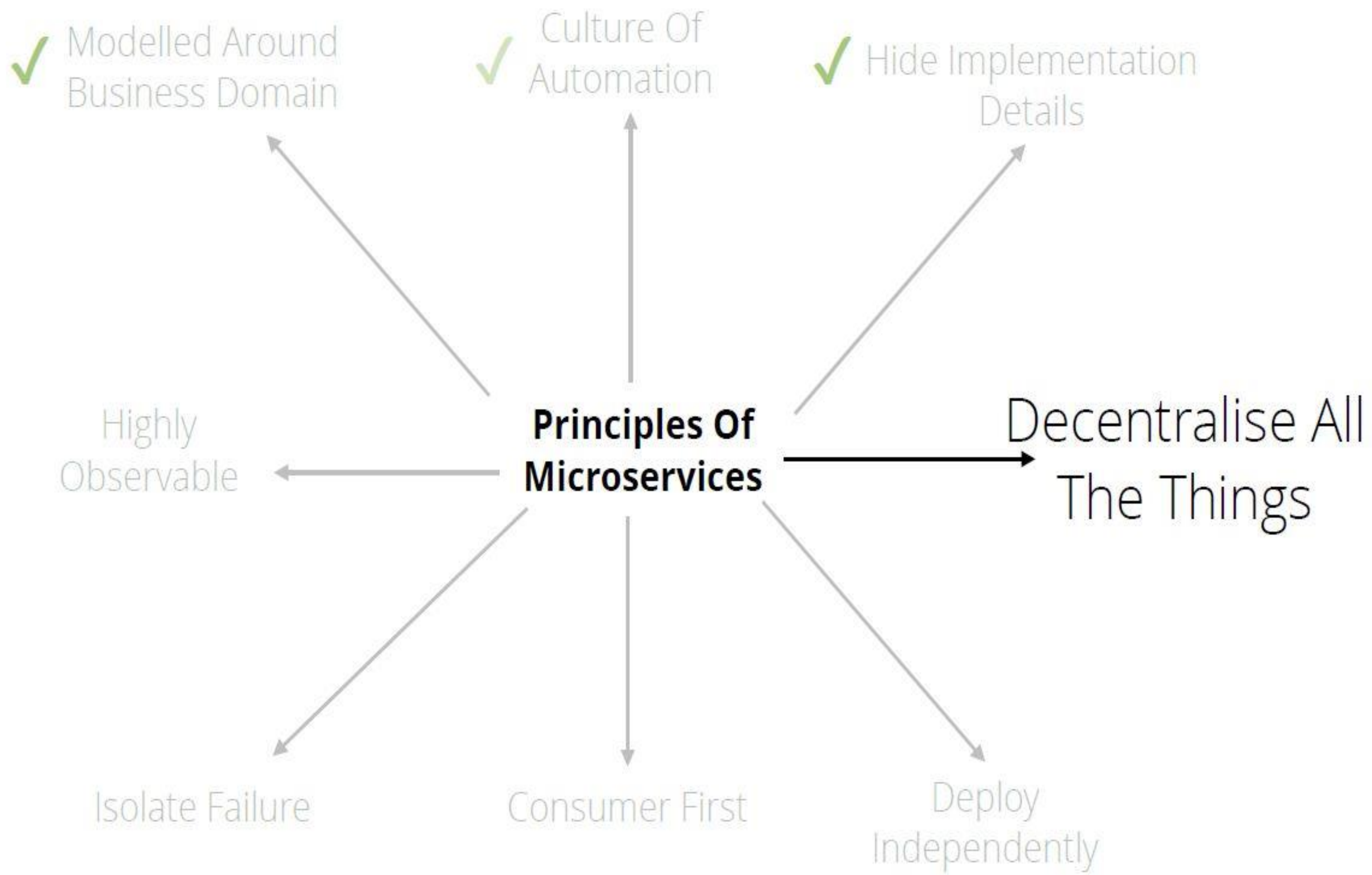
BOUNDED CONTEXTS



BE CAREFUL OF CLIENT LIBRARIES



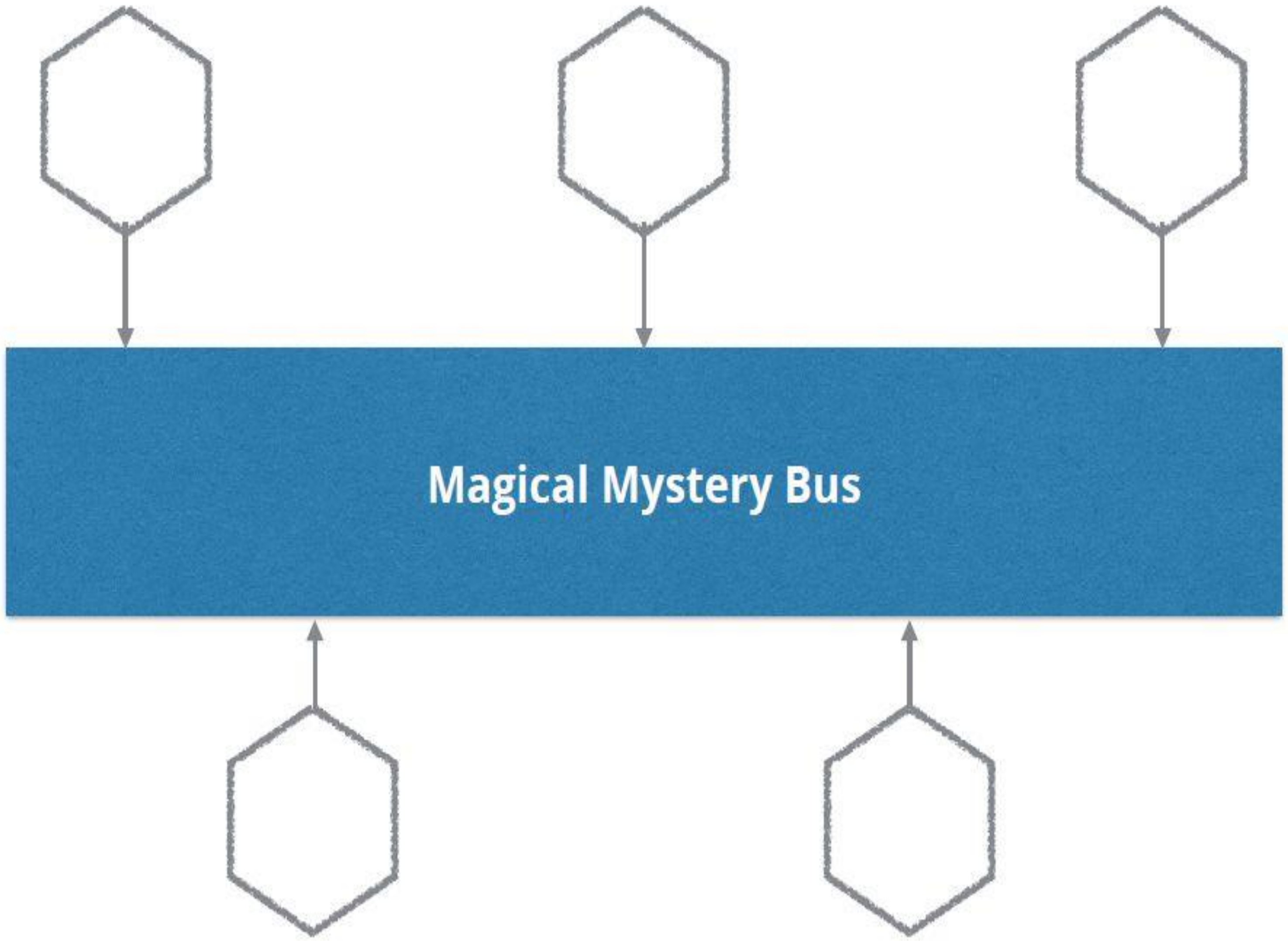

✓ Modelled Around Business Domain
✓ Culture Of Automation
✓ Hide Implementation Details
Highly Observable
Principles Of Microservices
Decentralise All The Things
Isolate Failure
Consumer First
Deploy Independently

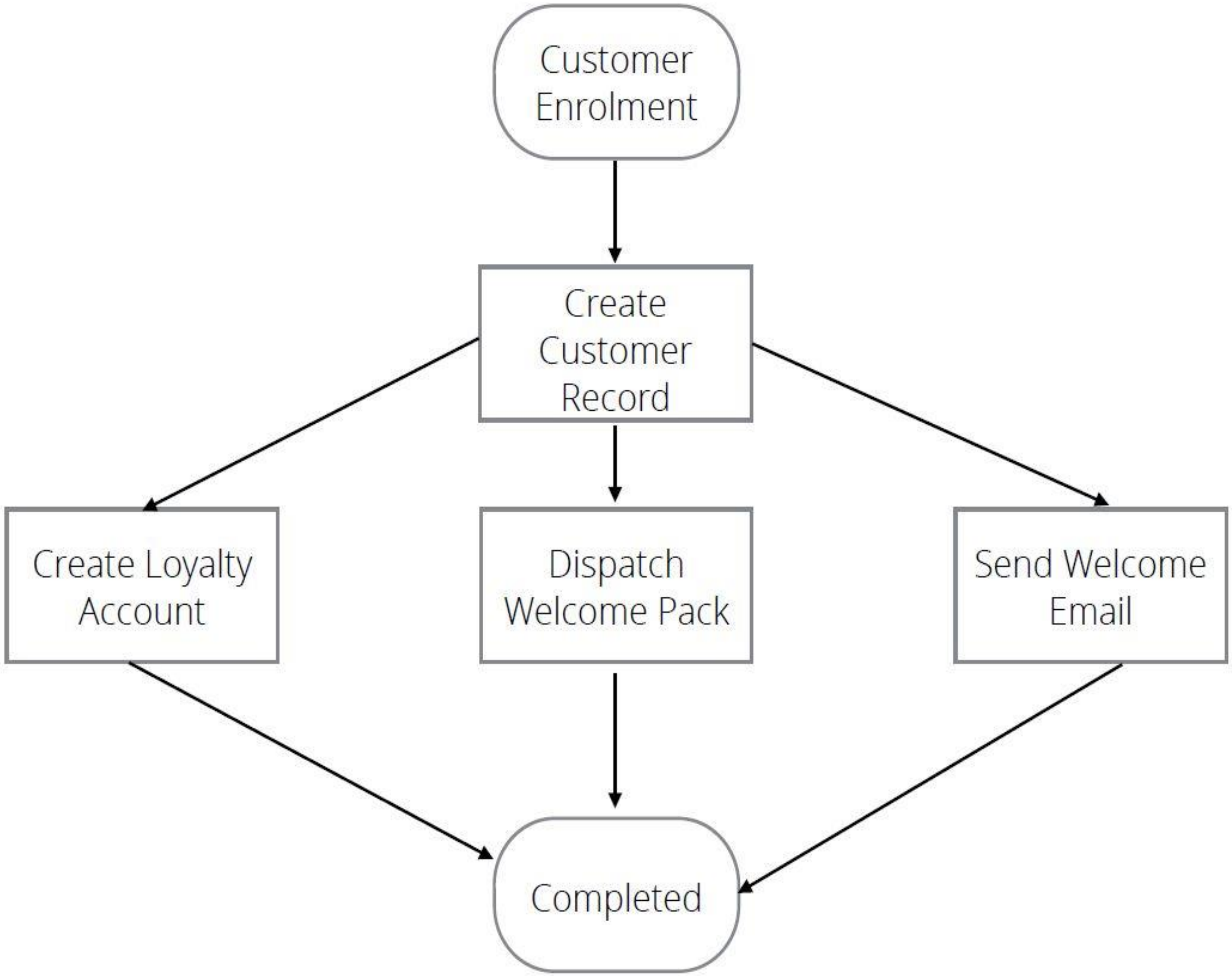

✓ Modelled Around Business Domain
✓ Culture Of Automation
✓ Hide Implementation Details
Highly Observable
Principles Of Microservices
Decentralise All The Things
Isolate Failure
Consumer First
Deploy Independently

What is autonomy?

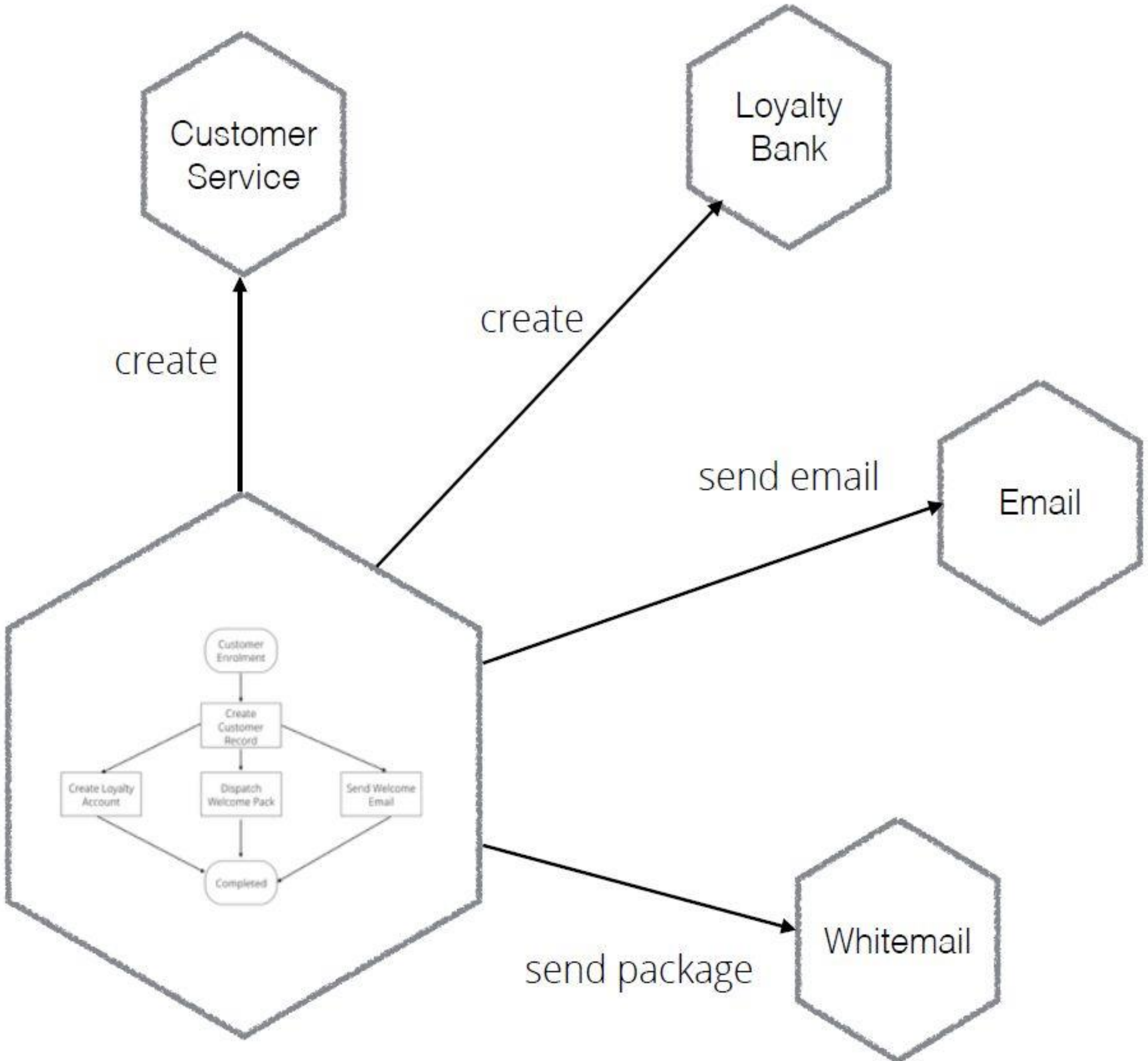
Giving people as much freedom as possible to do the job at hand

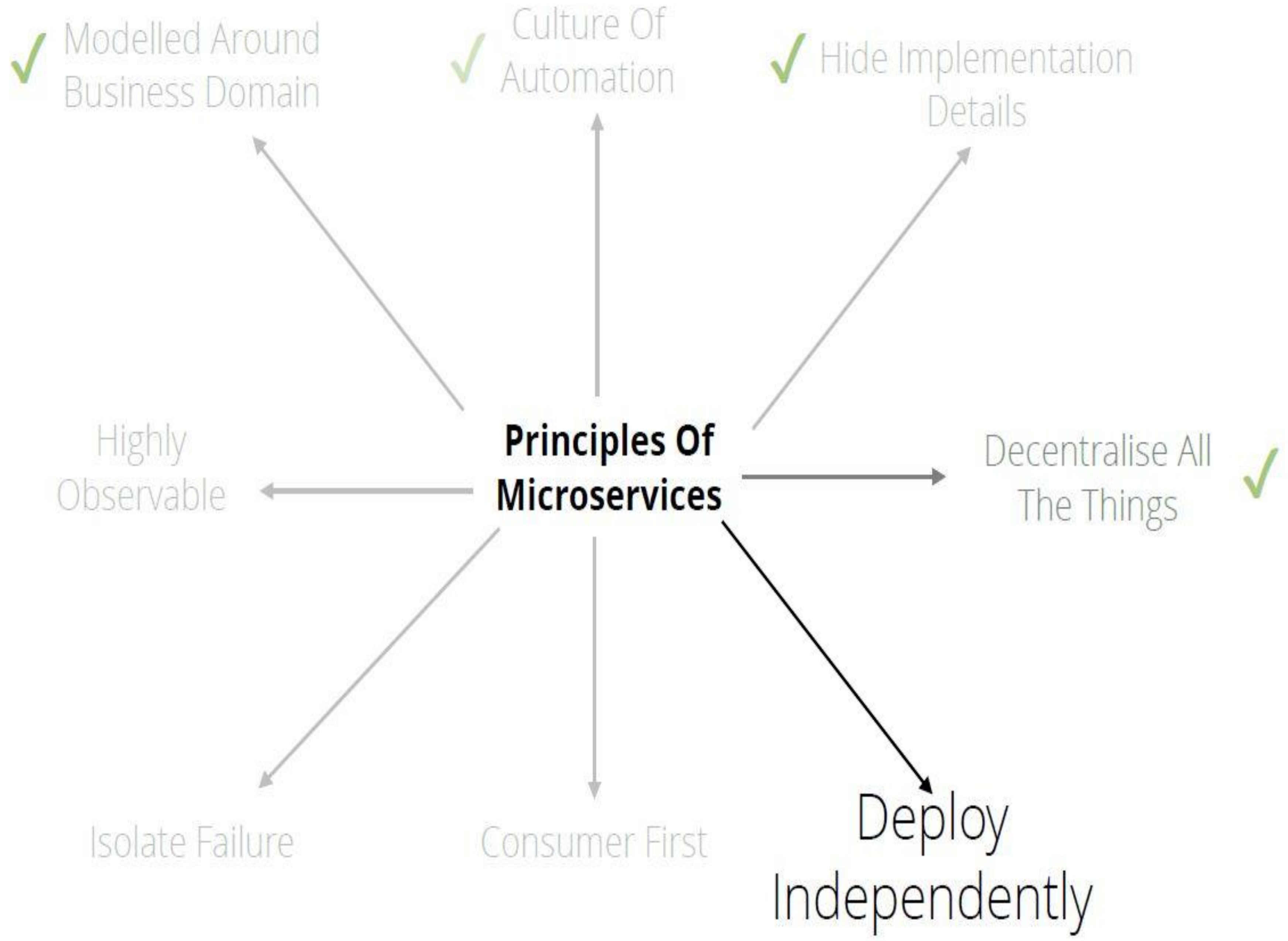
DUMB-PIPES, SMART ENDPOINTS



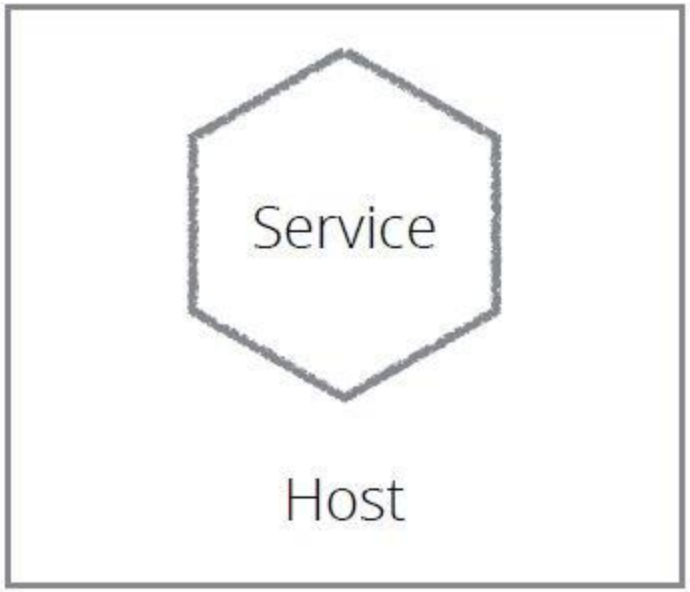

Customer Enrolment
Create Customer Record
Create Loyalty Account
Dispatch Welcome Pack
Send Welcome Email
Completed

ORCHESTRATION

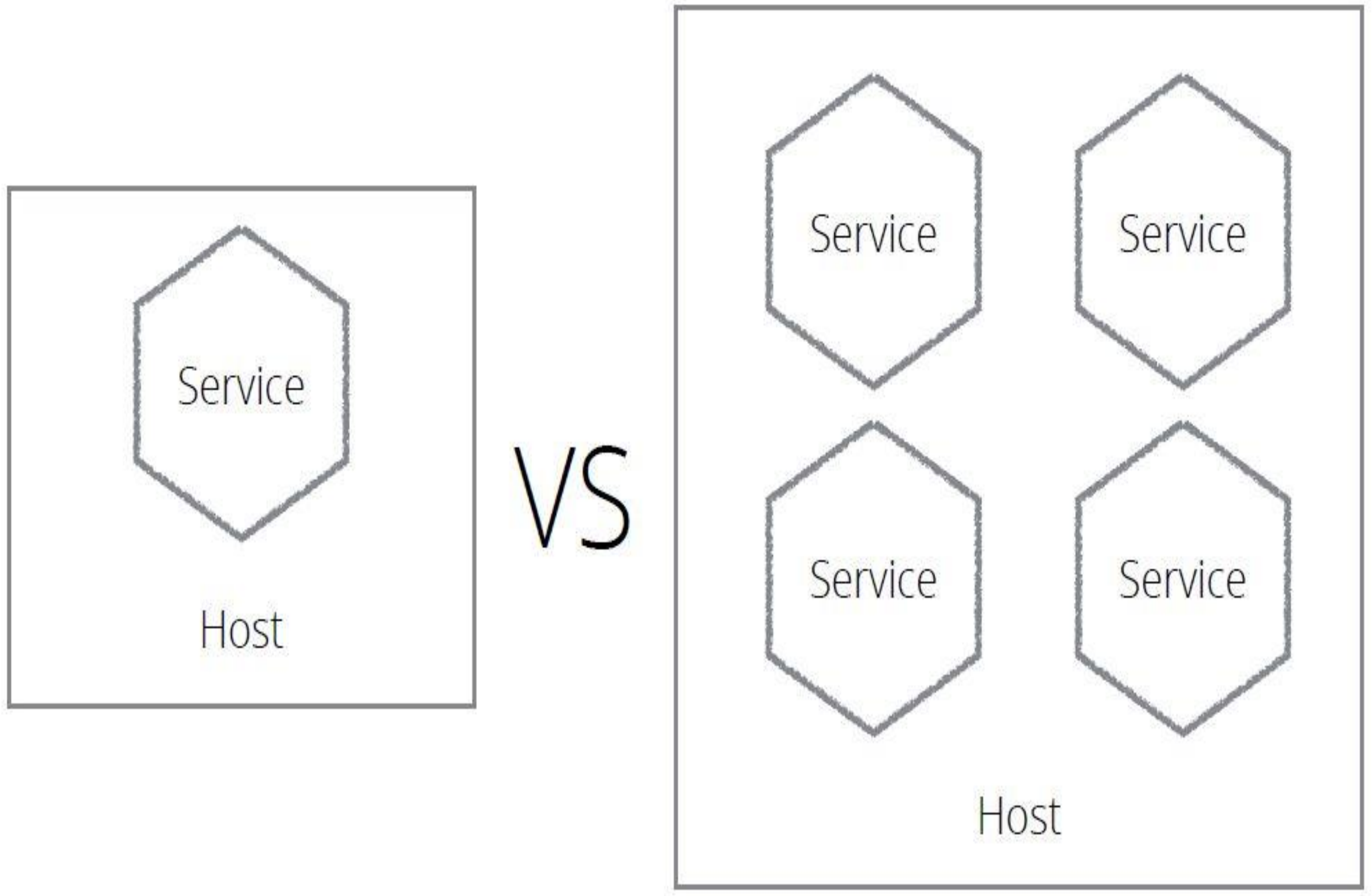



✓ Modelled Around Business Domain
✓ Culture Of Automation
✓ Hide Implementation Details
Highly Observable
Principles Of Microservices
Decentralise All The Things ✓
Isolate Failure
Consumer First
Deploy Independently

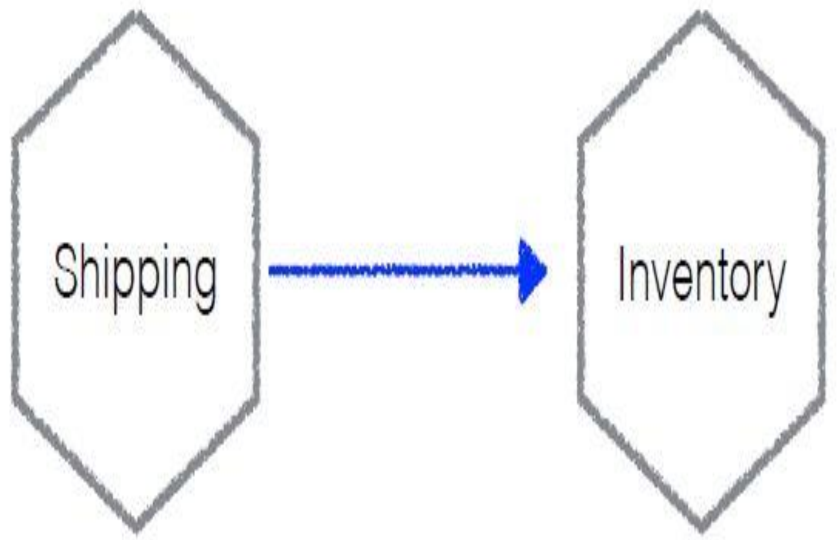
ONE SERVICE PER-HOST



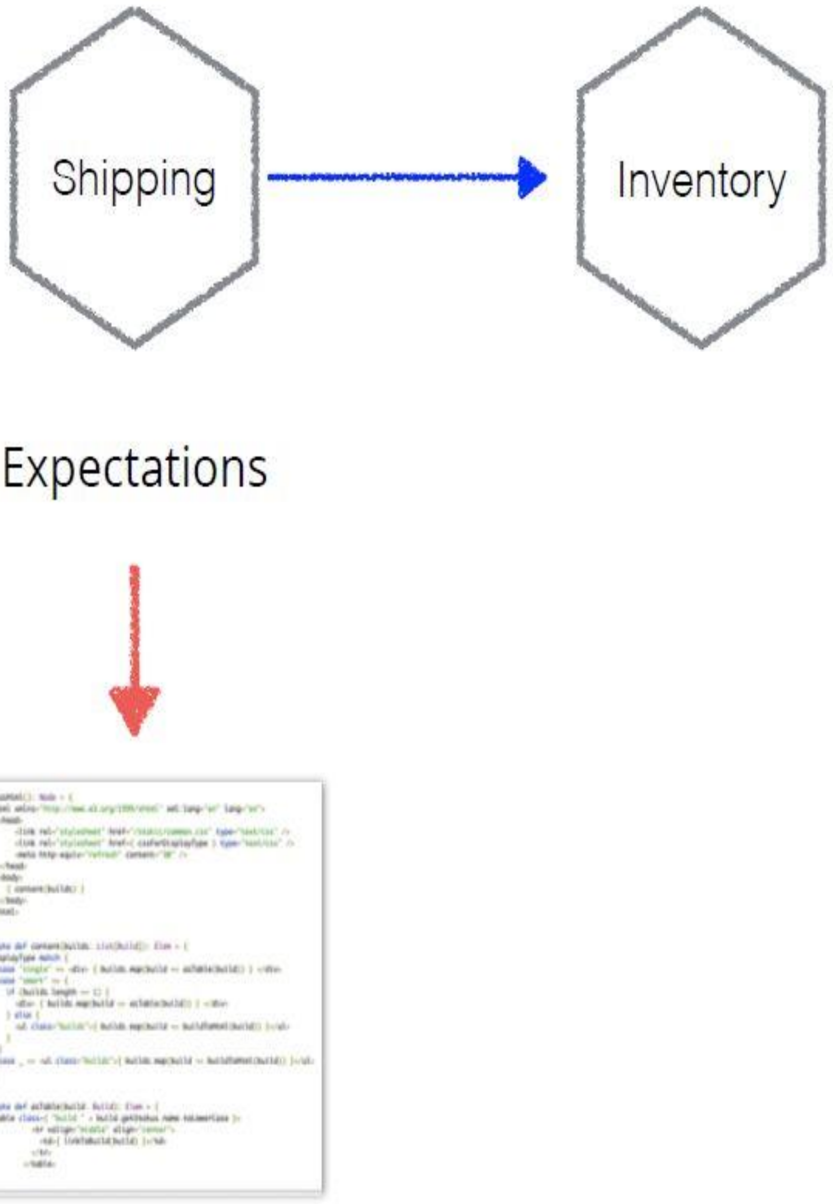
ONE SERVICE PER-HOST



CONSUMER-DRIVEN CONTRACTS

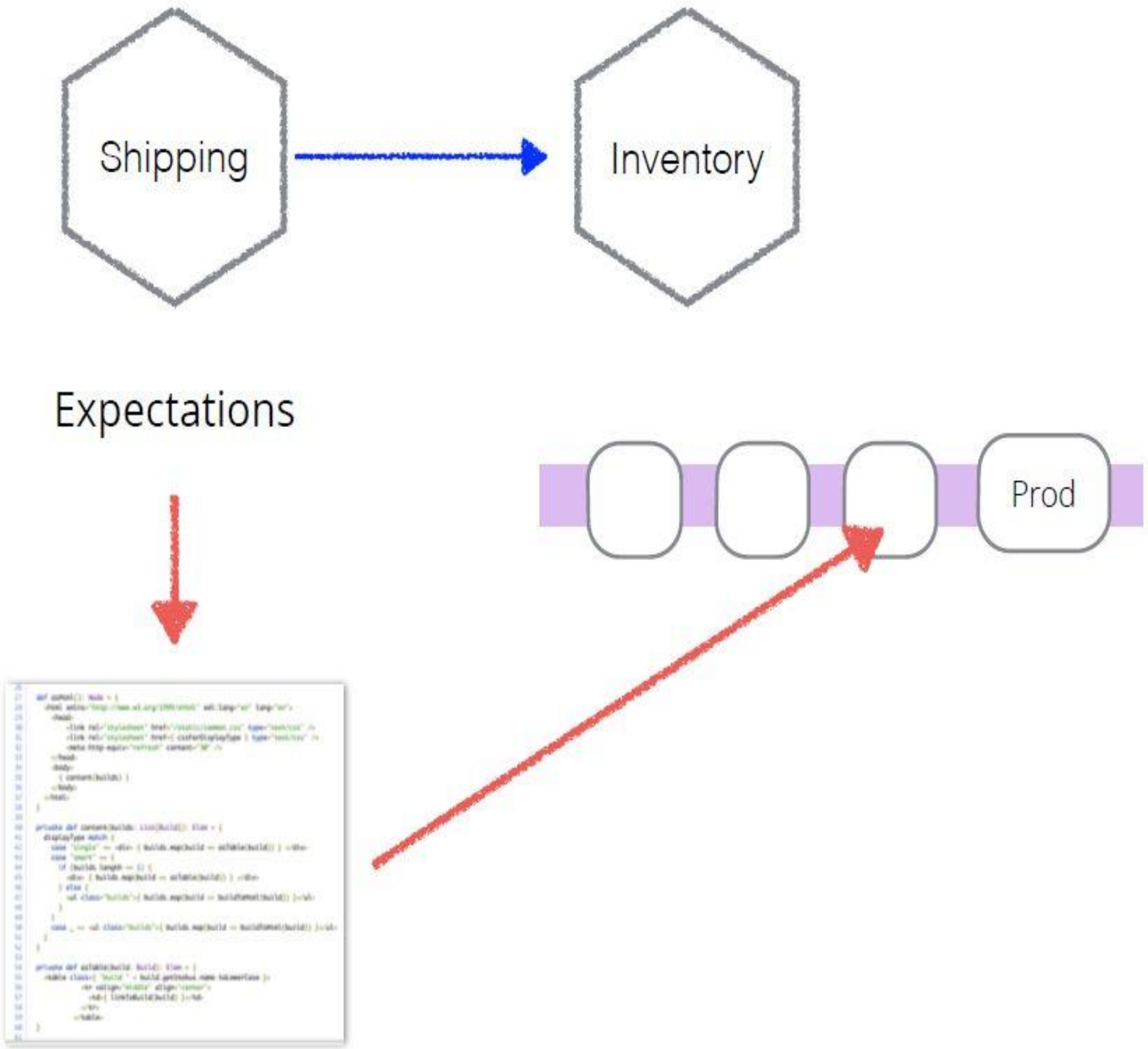


CONSUMER DRIVEN CONTRACTS

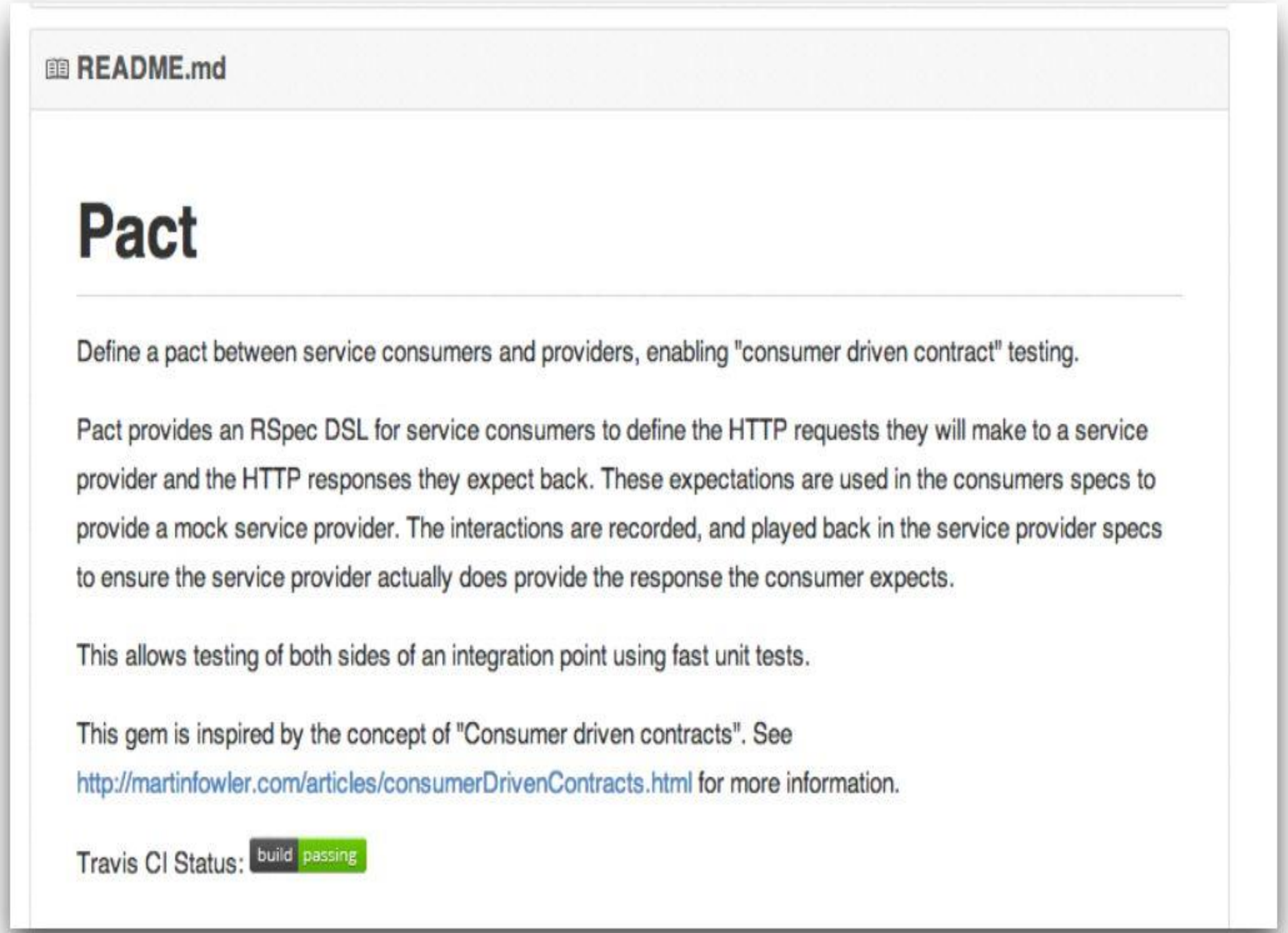


```
def asHtml(): Node = {
  <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
    <head>
        <link rel="stylesheet" href="/static/common.css" type="text/css" />
        <link rel="stylesheet" href={ cssForDisplayType } type="text/css" />
        <meta http-equiv="refresh" content="30"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

CONSUMER-DRIVEN CONTRACTS



```
[illegible]
```


README.md

Pact

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

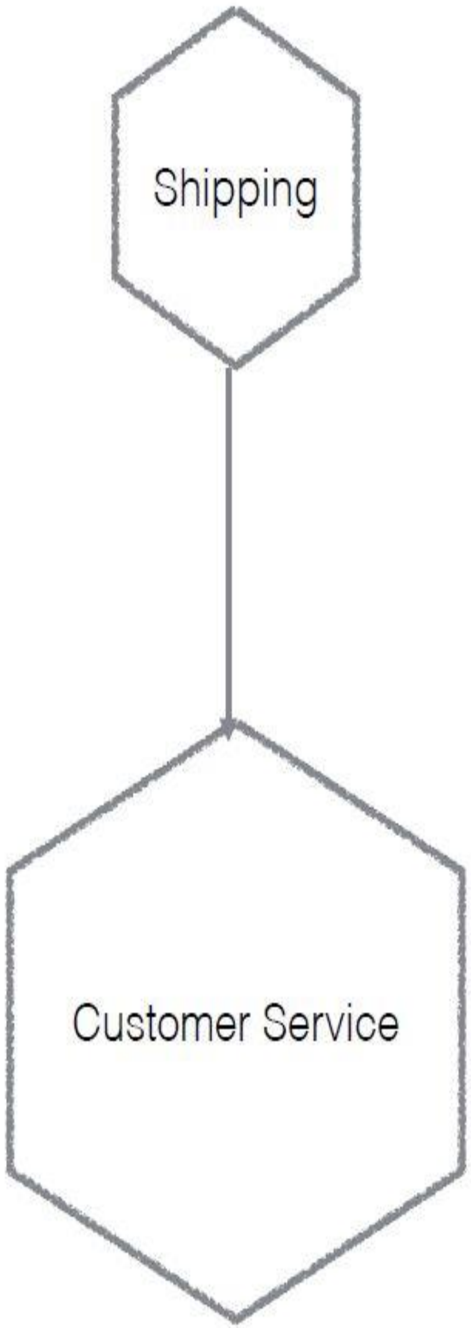
Pact provides an RSpec DSL for service consumers to define the HTTP requests they will make to a service provider and the HTTP responses they expect back. These expectations are used in the consumers specs to provide a mock service provider. The interactions are recorded, and played back in the service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

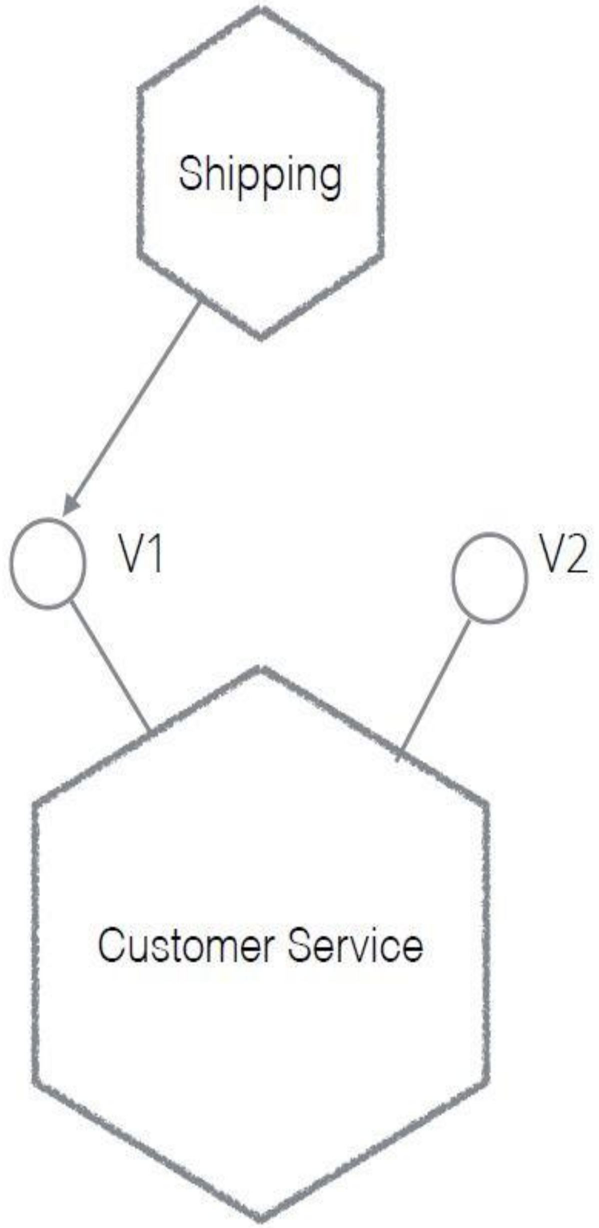
This gem is inspired by the concept of "Consumer driven contracts". See http://martinfowler.com/articles/consumerDrivenContracts.html for more information.

Travis CI Status: build passing

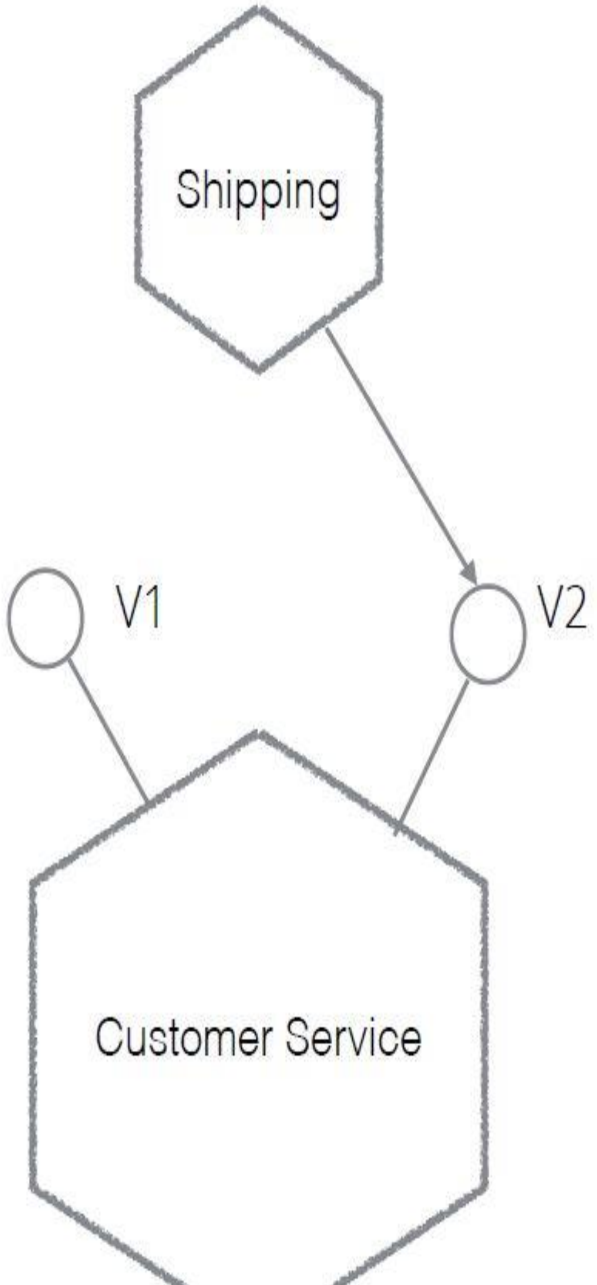
https://github.com/realestate-com-au/pact


Shipping
Customer Service

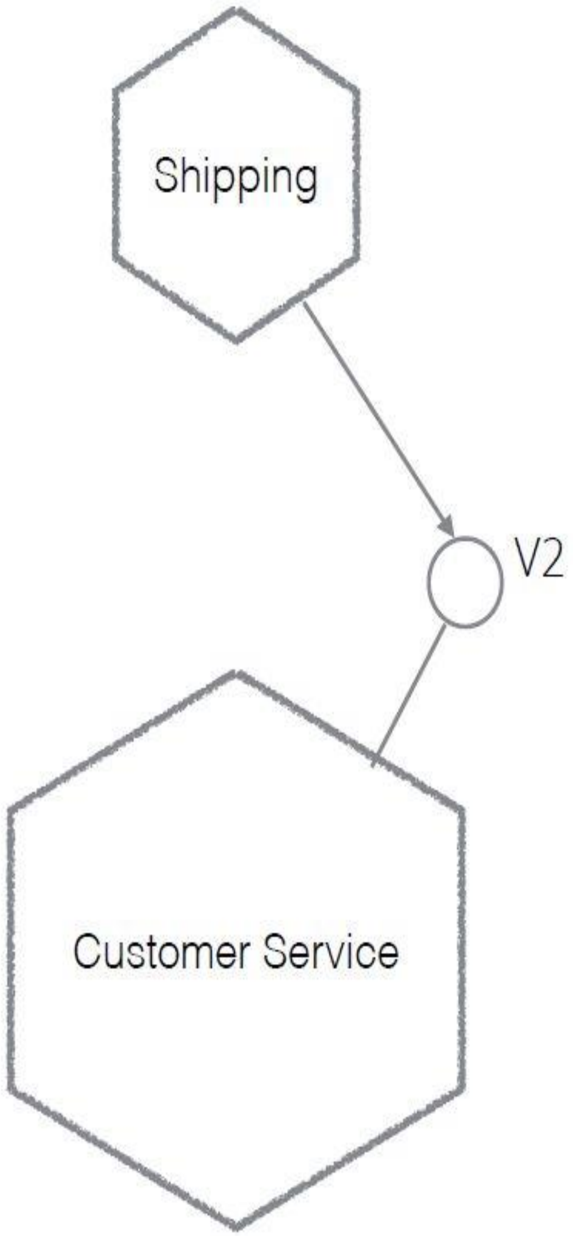
CO-EXIST ENDPOINTS

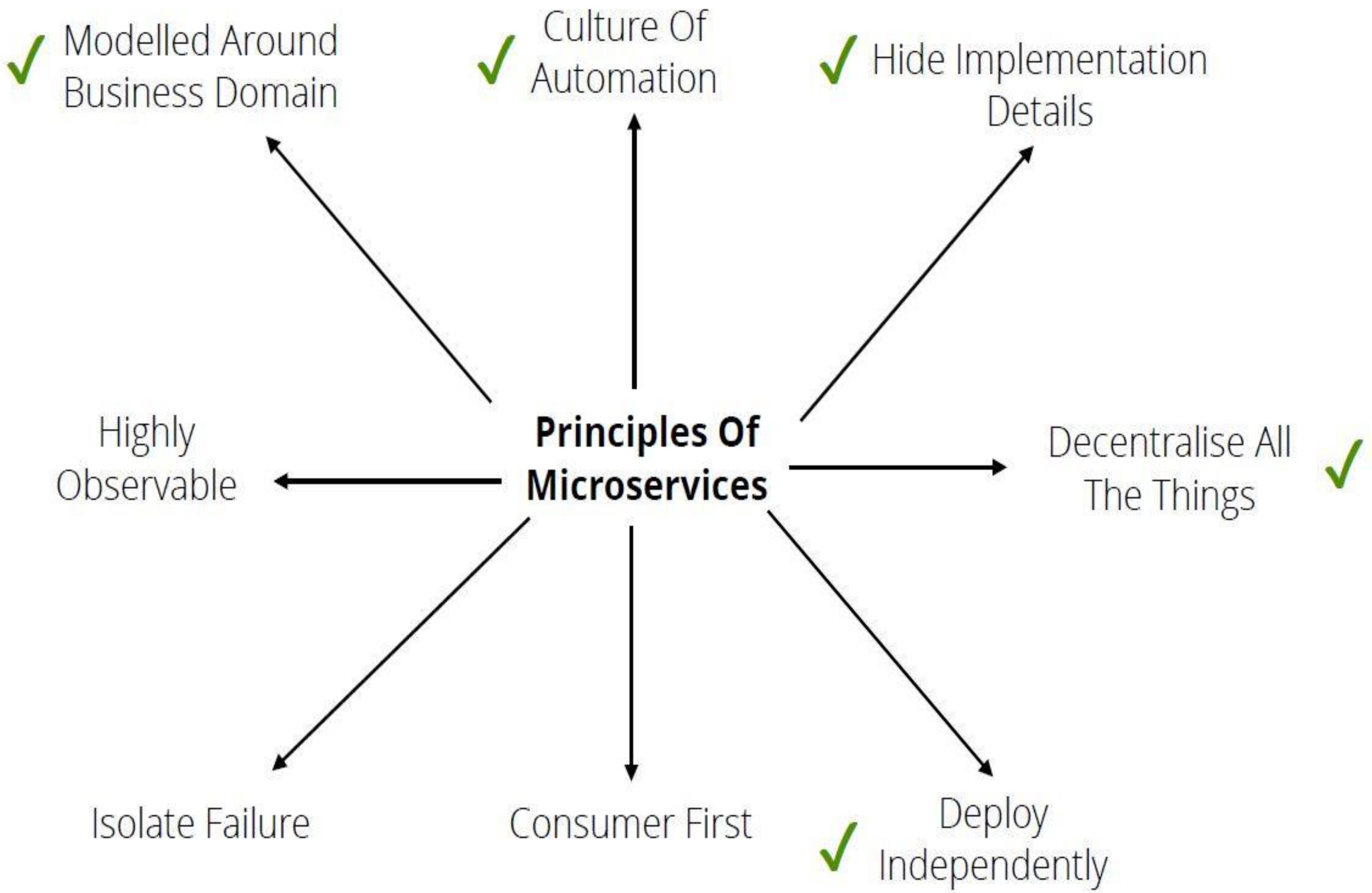


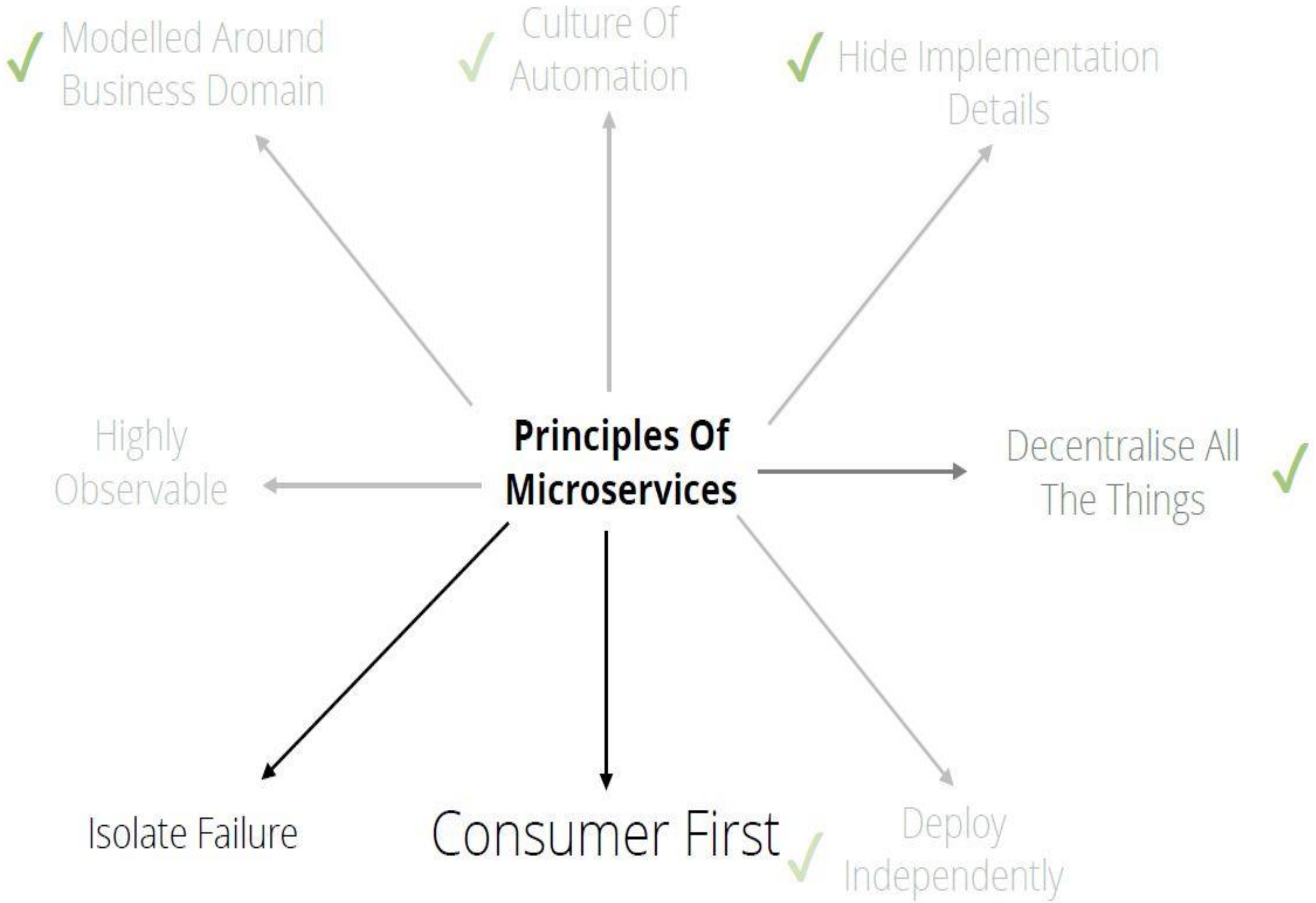
CO-EXIST ENDPOINTS



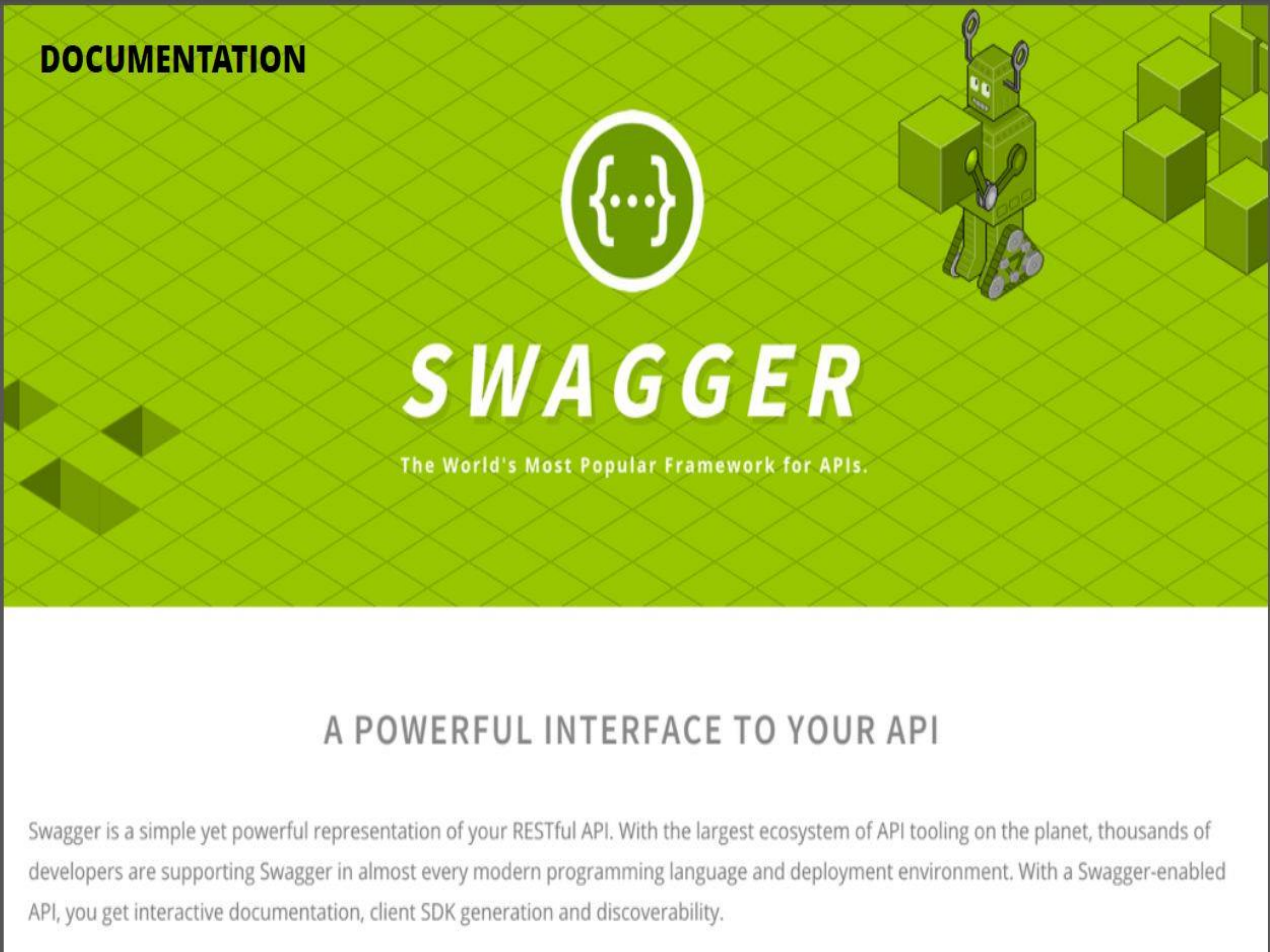
CO-EXIST ENDPOINTS




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


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

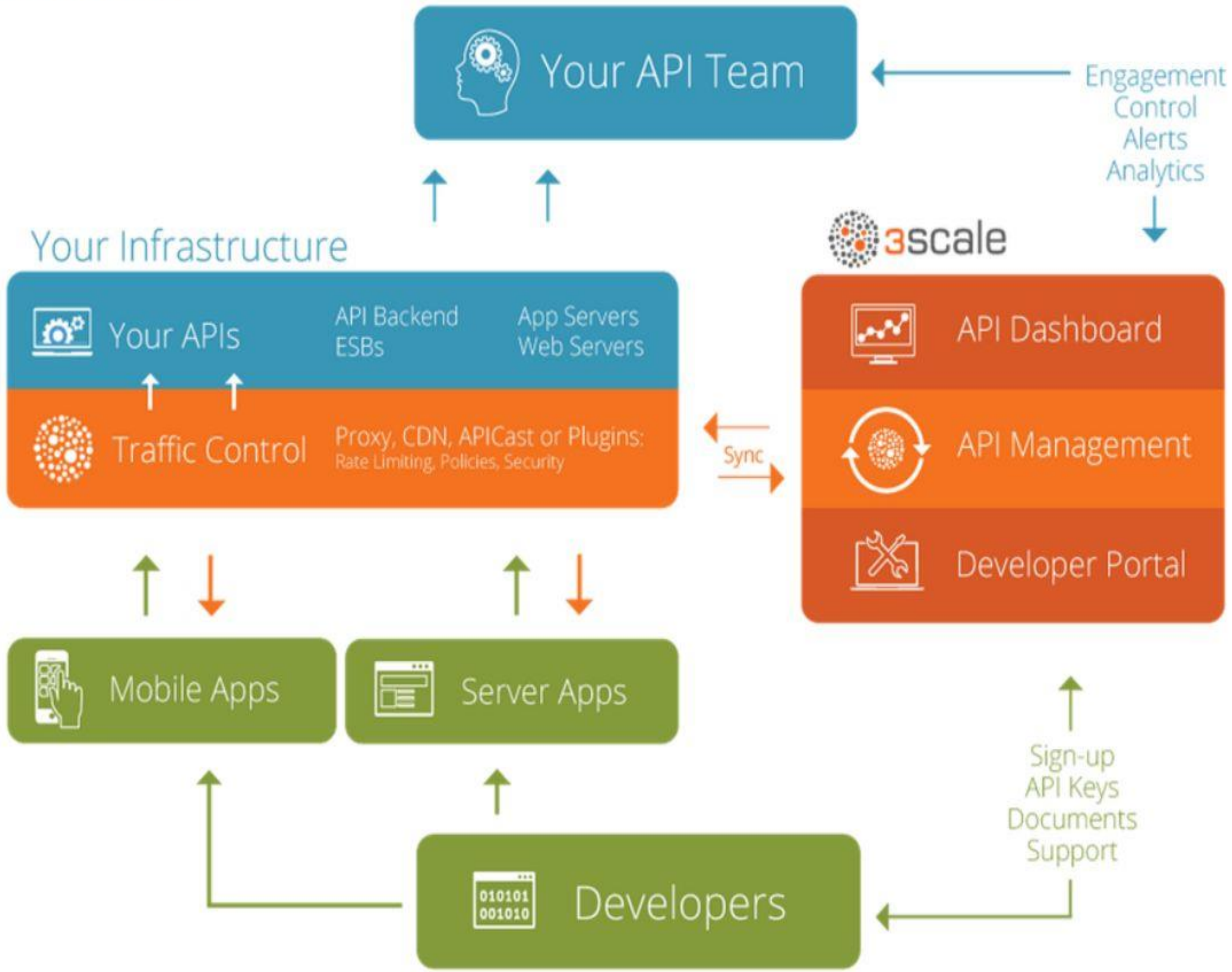
DOCUMENTATION



A POWERFUL INTERFACE TO YOUR API

Swagger is a simple yet powerful representation of your RESTful API. With the largest ecosystem of API tooling on the planet, thousands of developers are supporting Swagger in almost every modern programming language and deployment environment. With a Swagger-enabled API, you get interactive documentation, client SDK generation and discoverability.

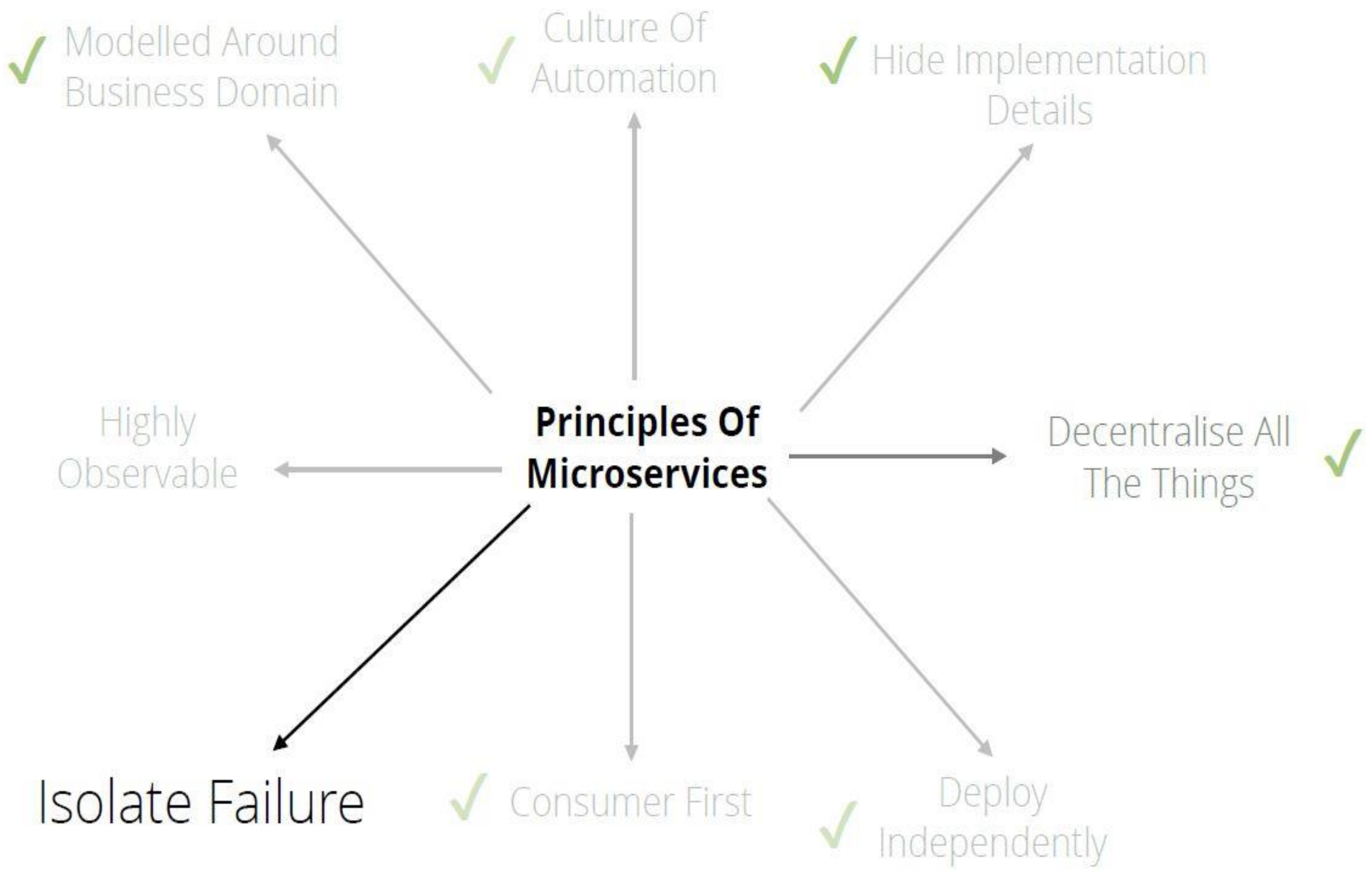
API GATEWAYS



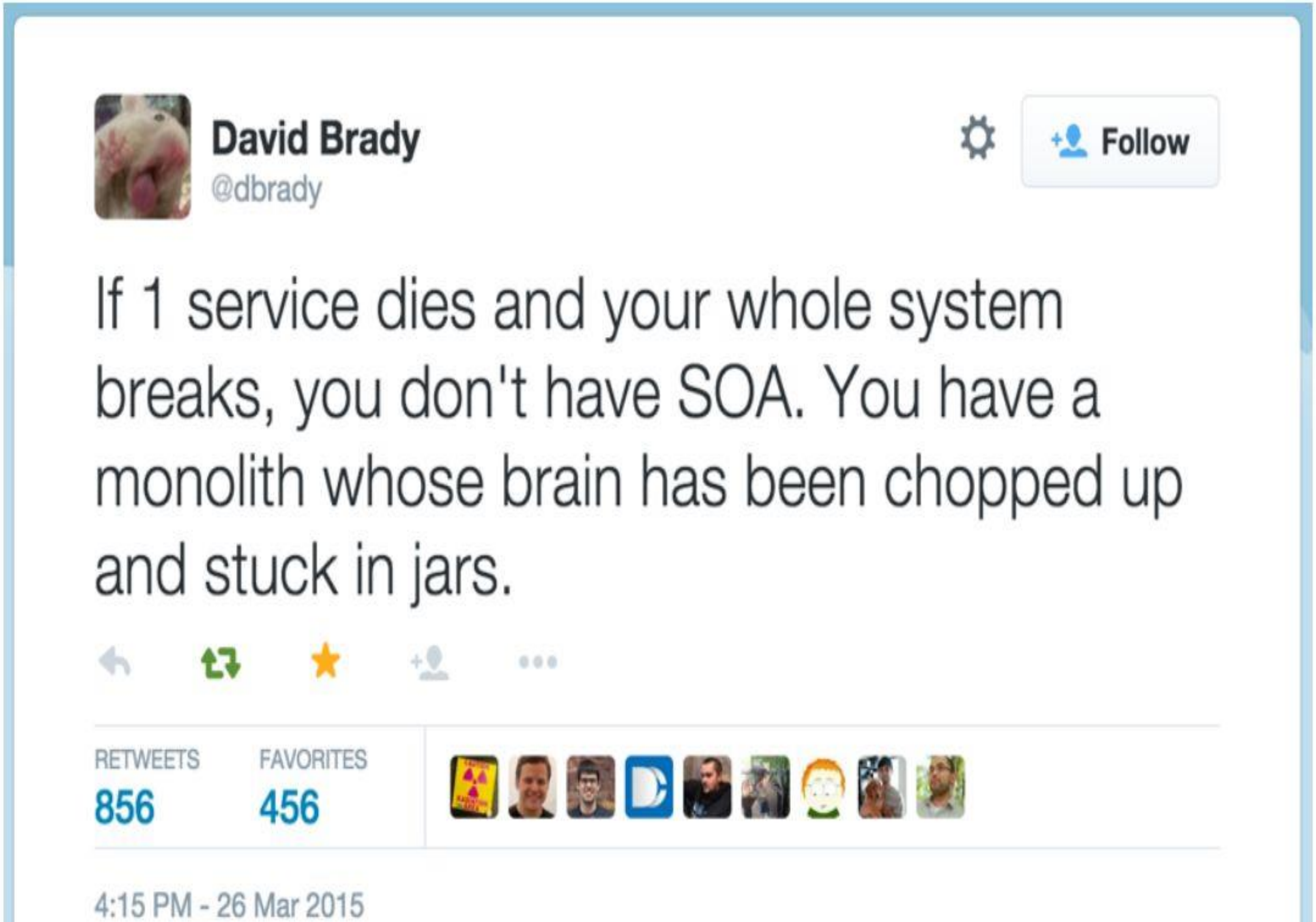
SERVICE DISCOVERY

etcd

CONSUL



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



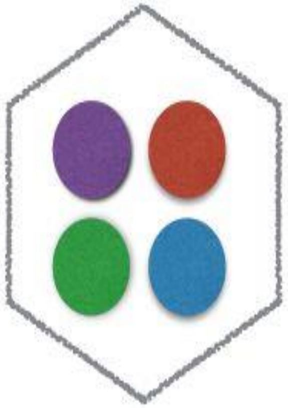
David Brady
@dbrady

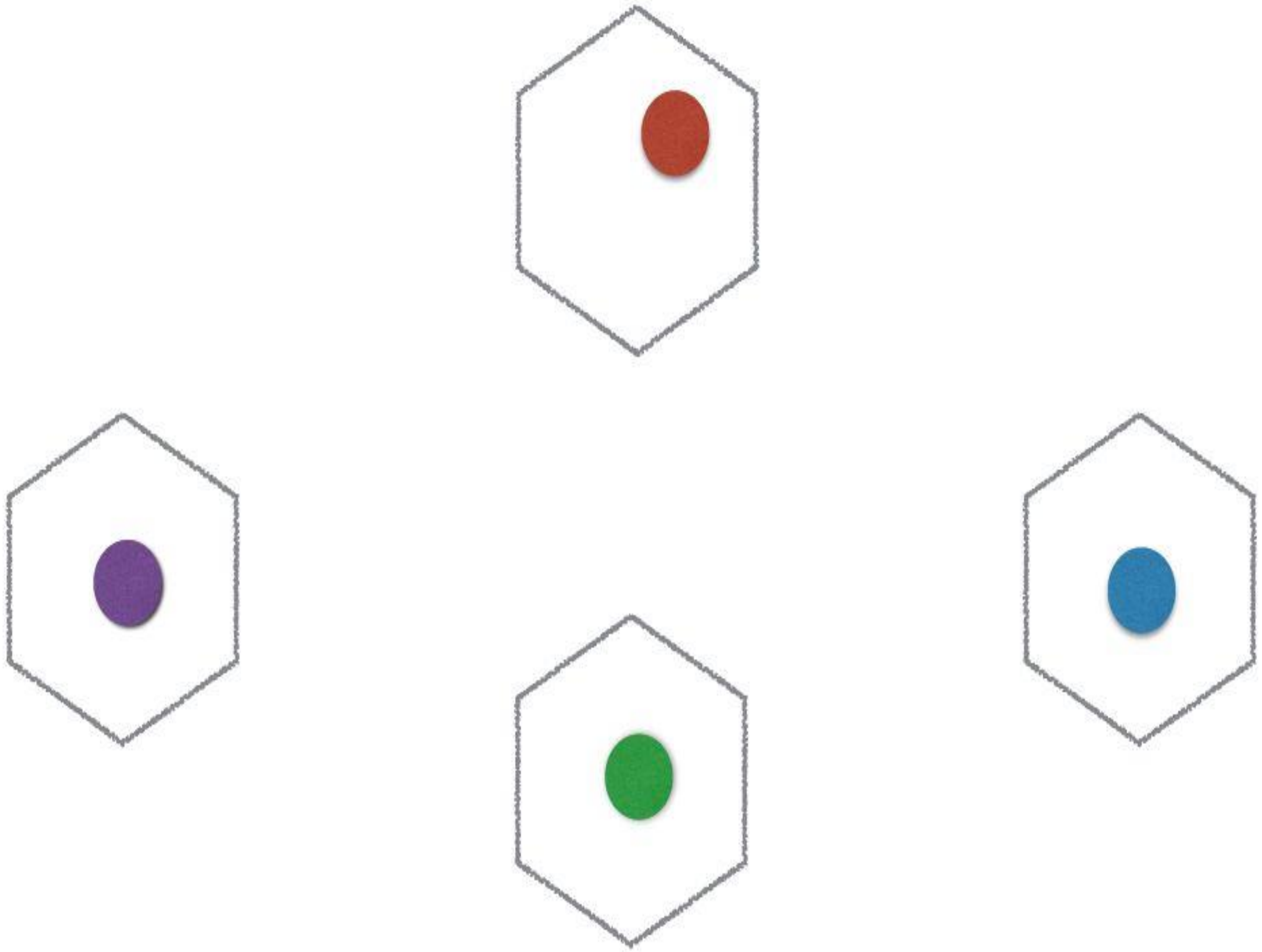
Follow

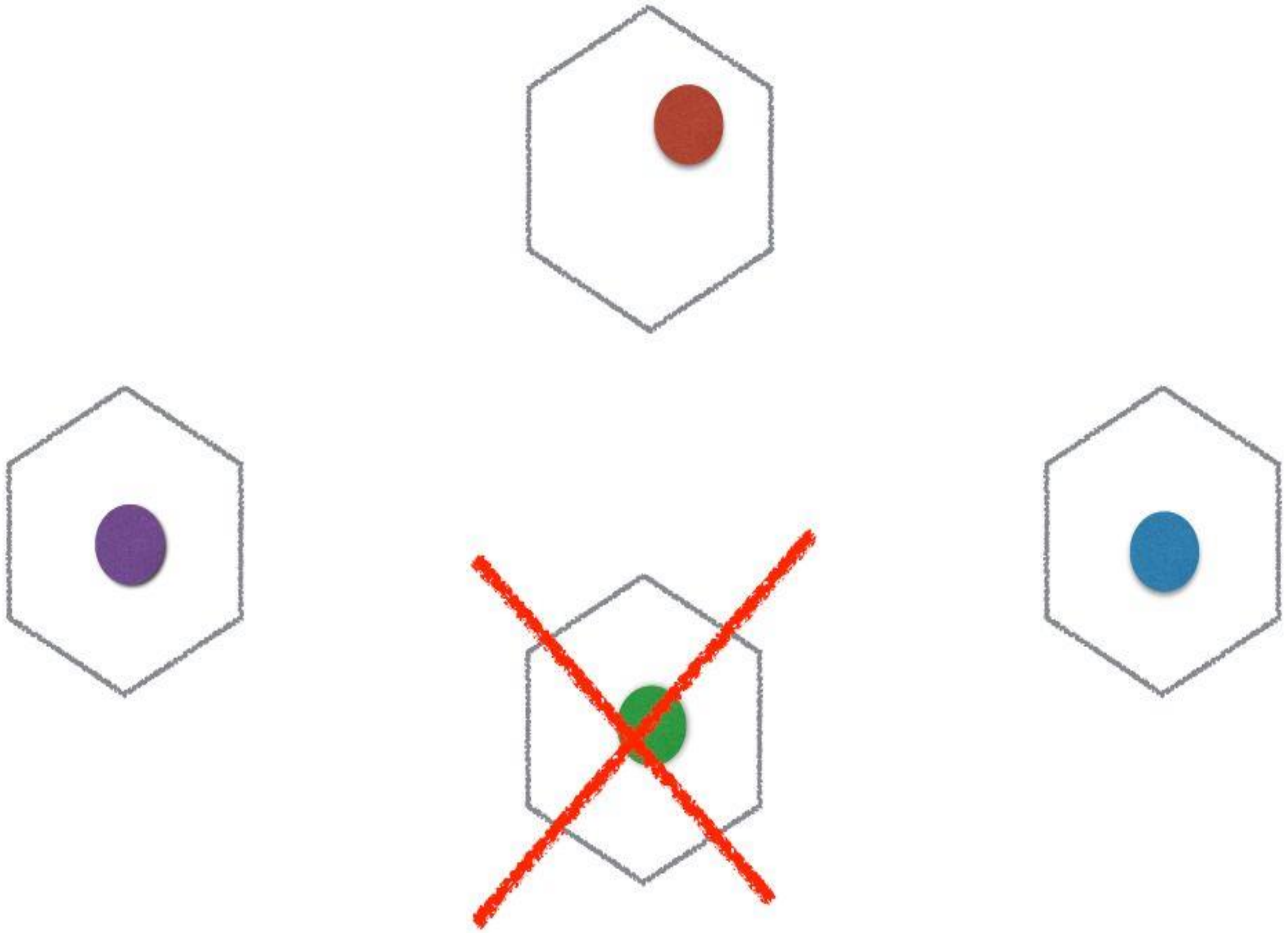
If 1 service dies and your whole system breaks, you don't have SOA. You have a monolith whose brain has been chopped up and stuck in jars.

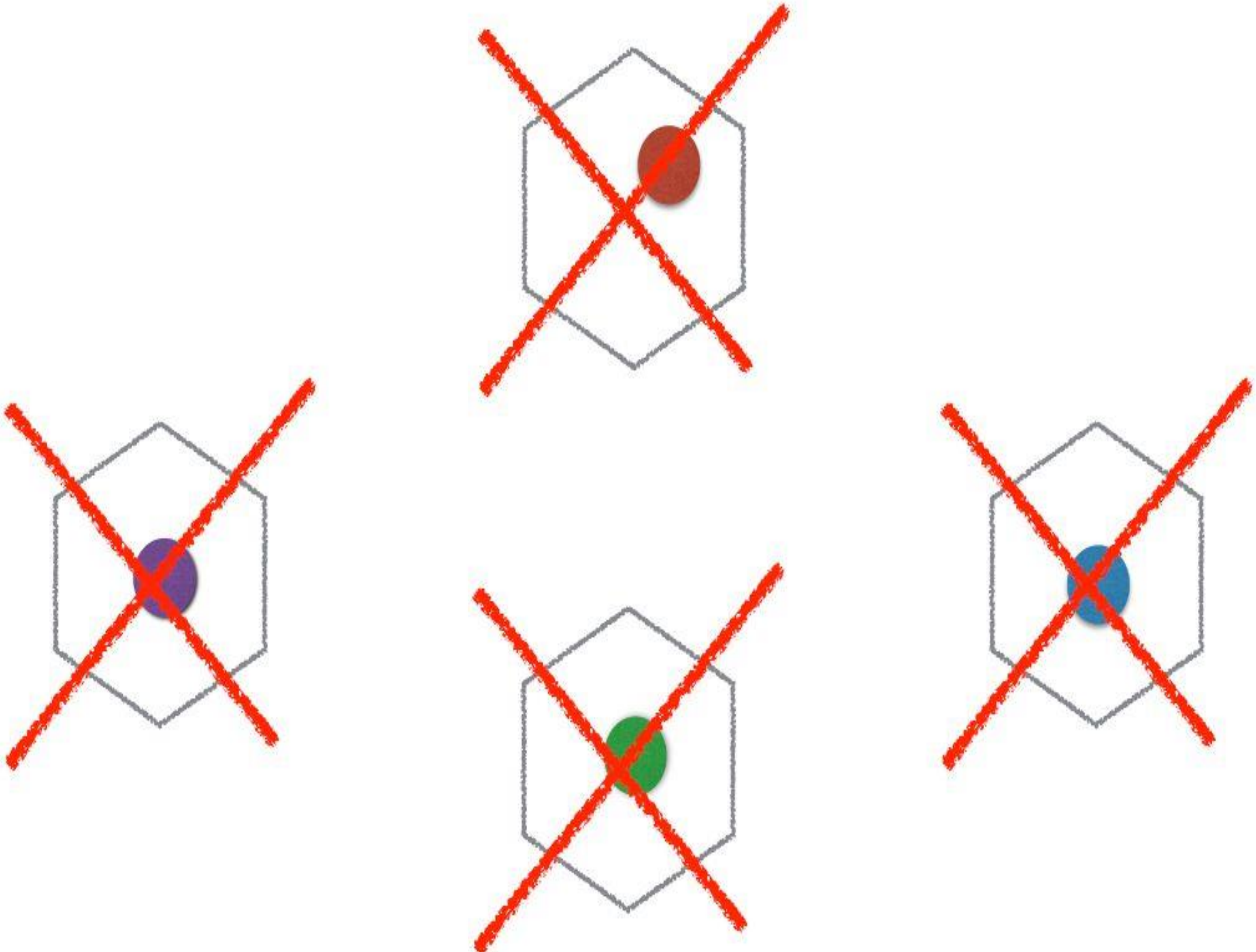
RETWEETS 856
FAVORITES 456

4:15 PM - 26 Mar 2015

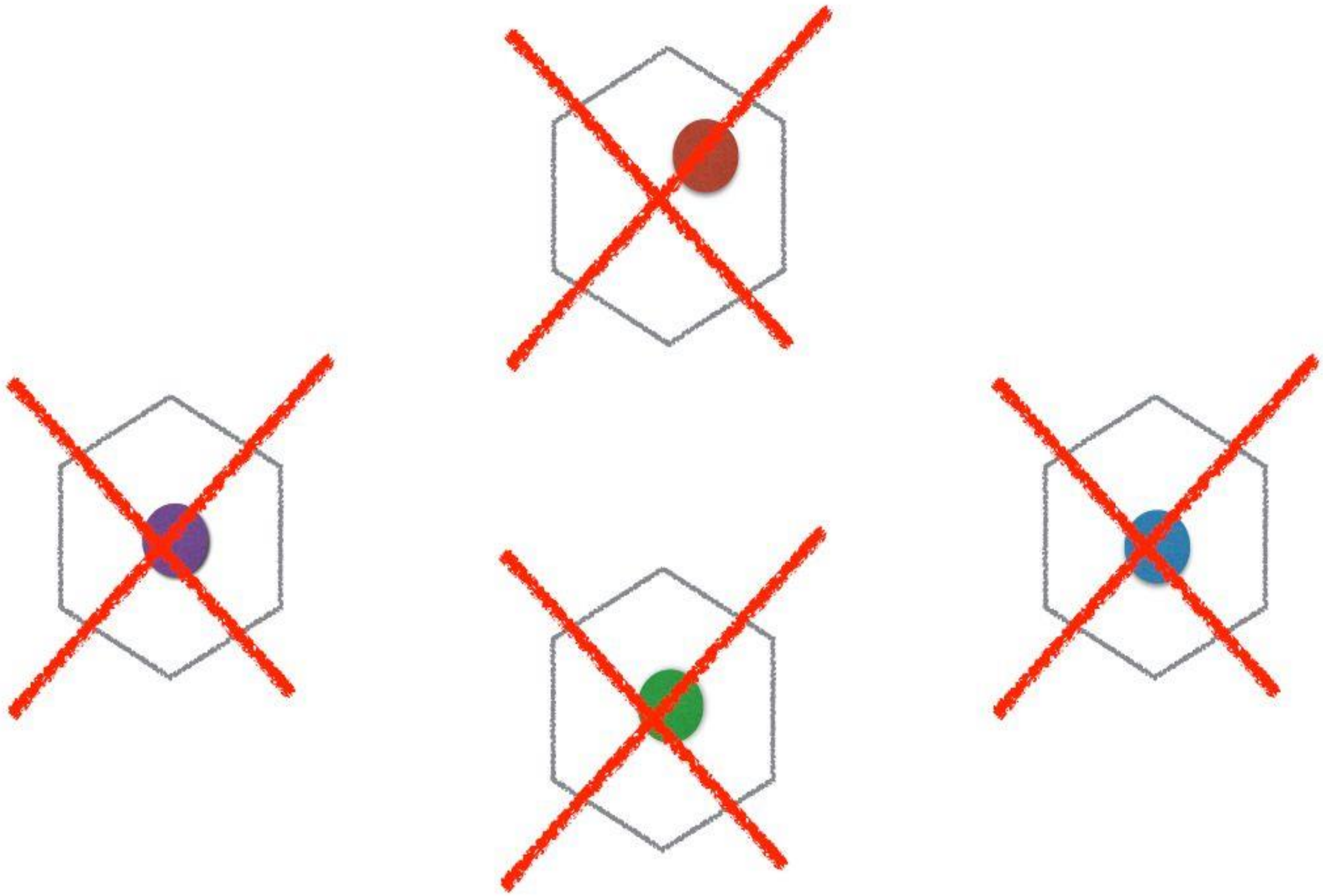


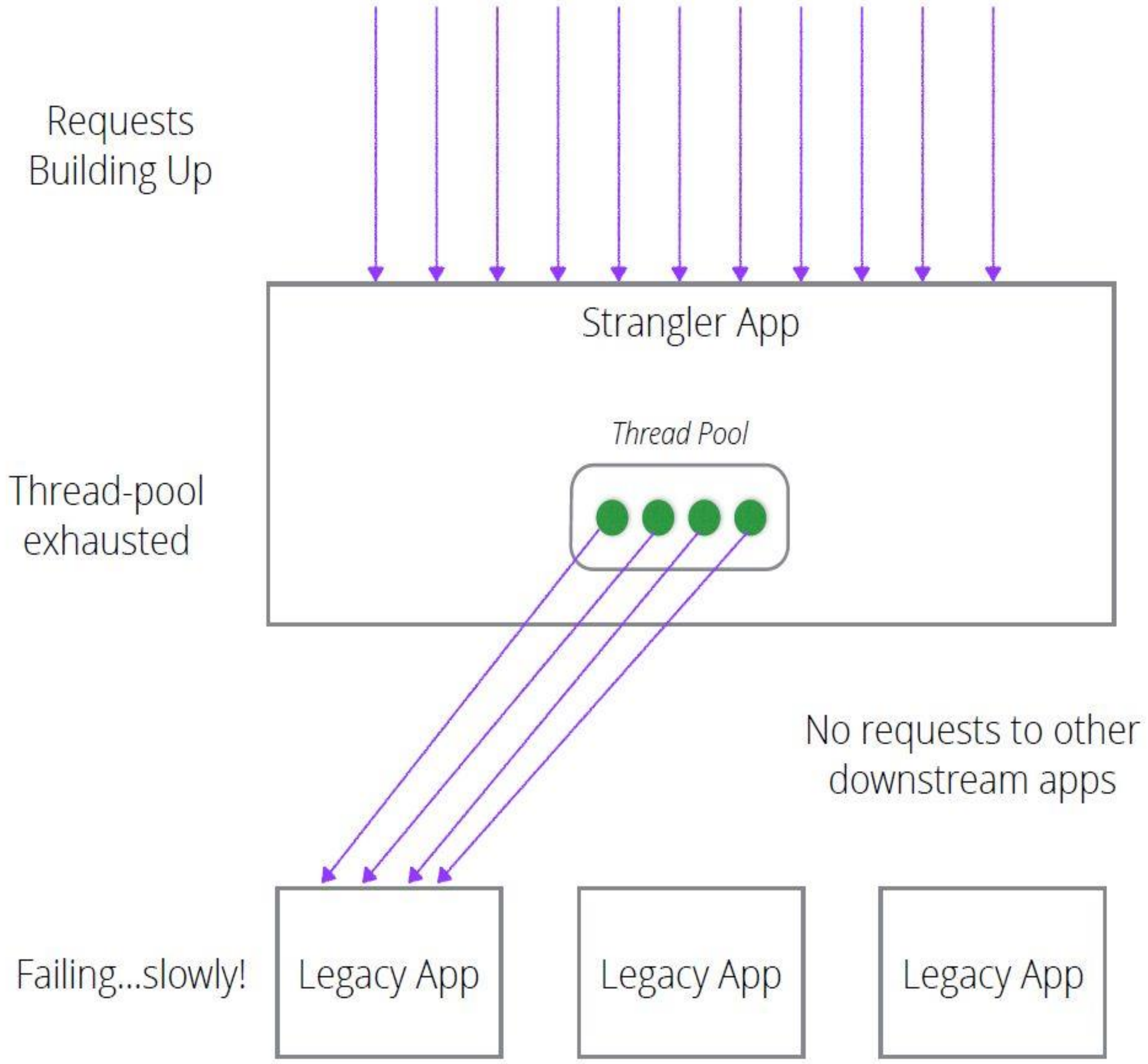




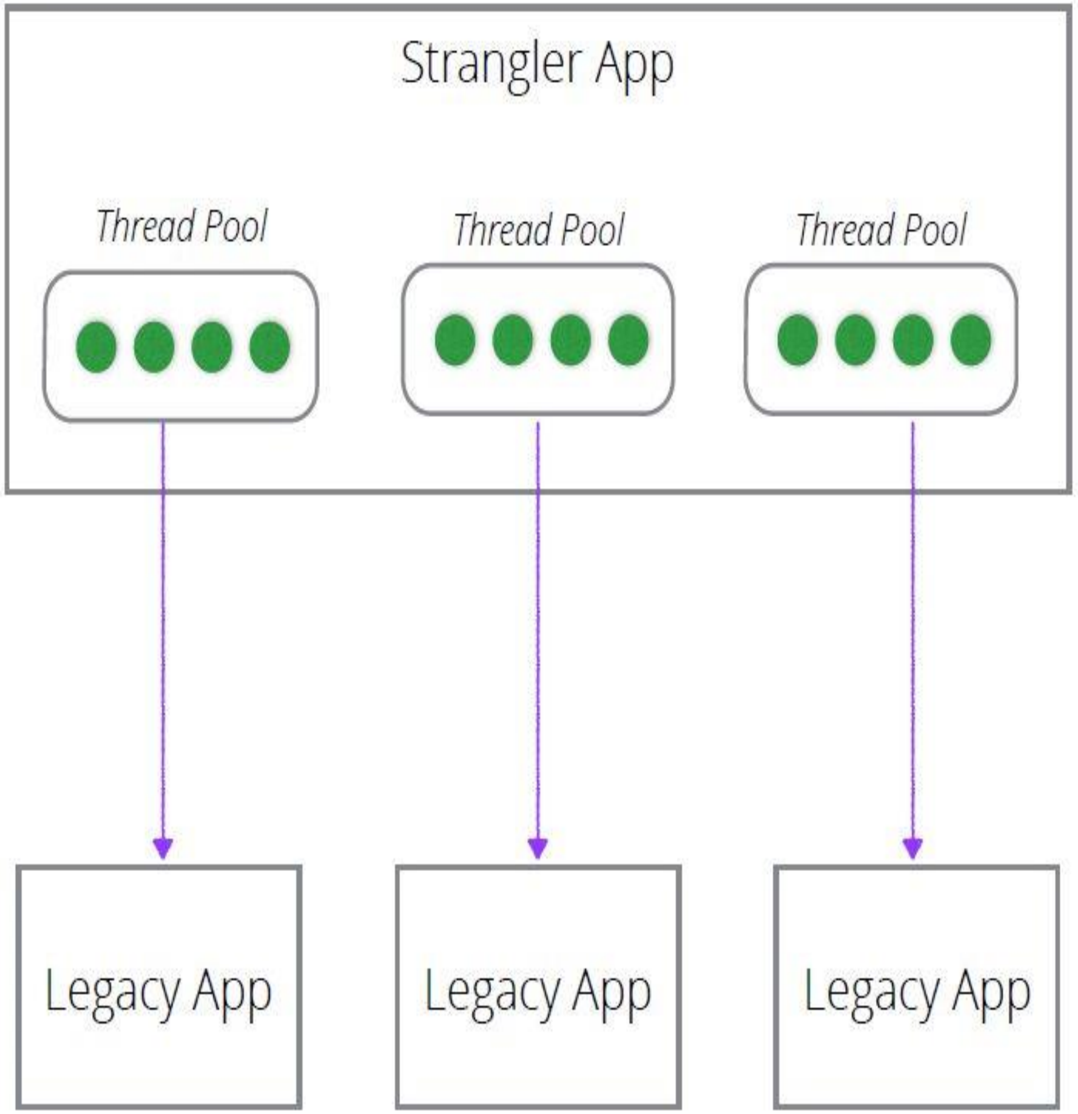


AVOID THE DISTRIBUTED SINGLE POINT OF FAILURE!

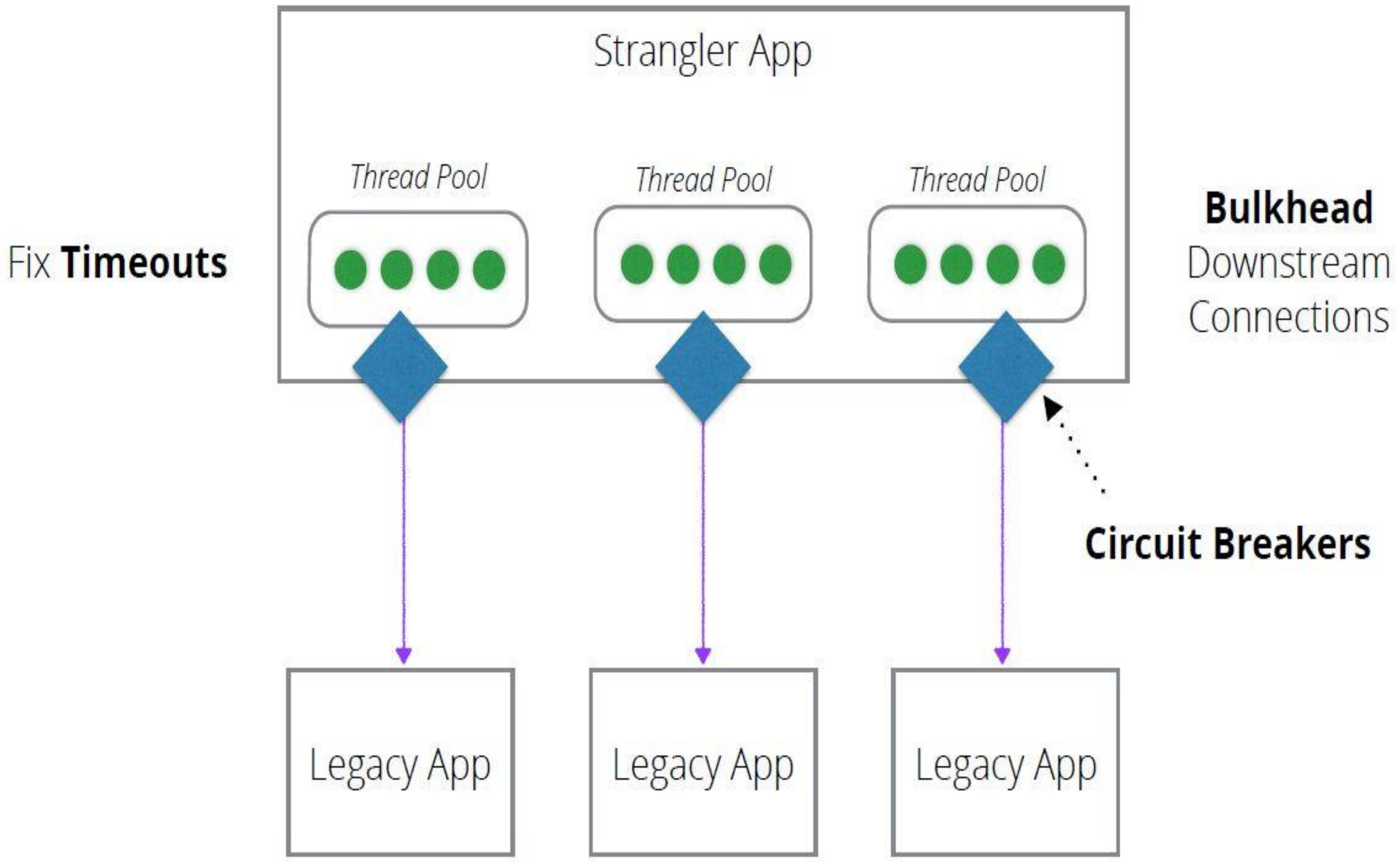


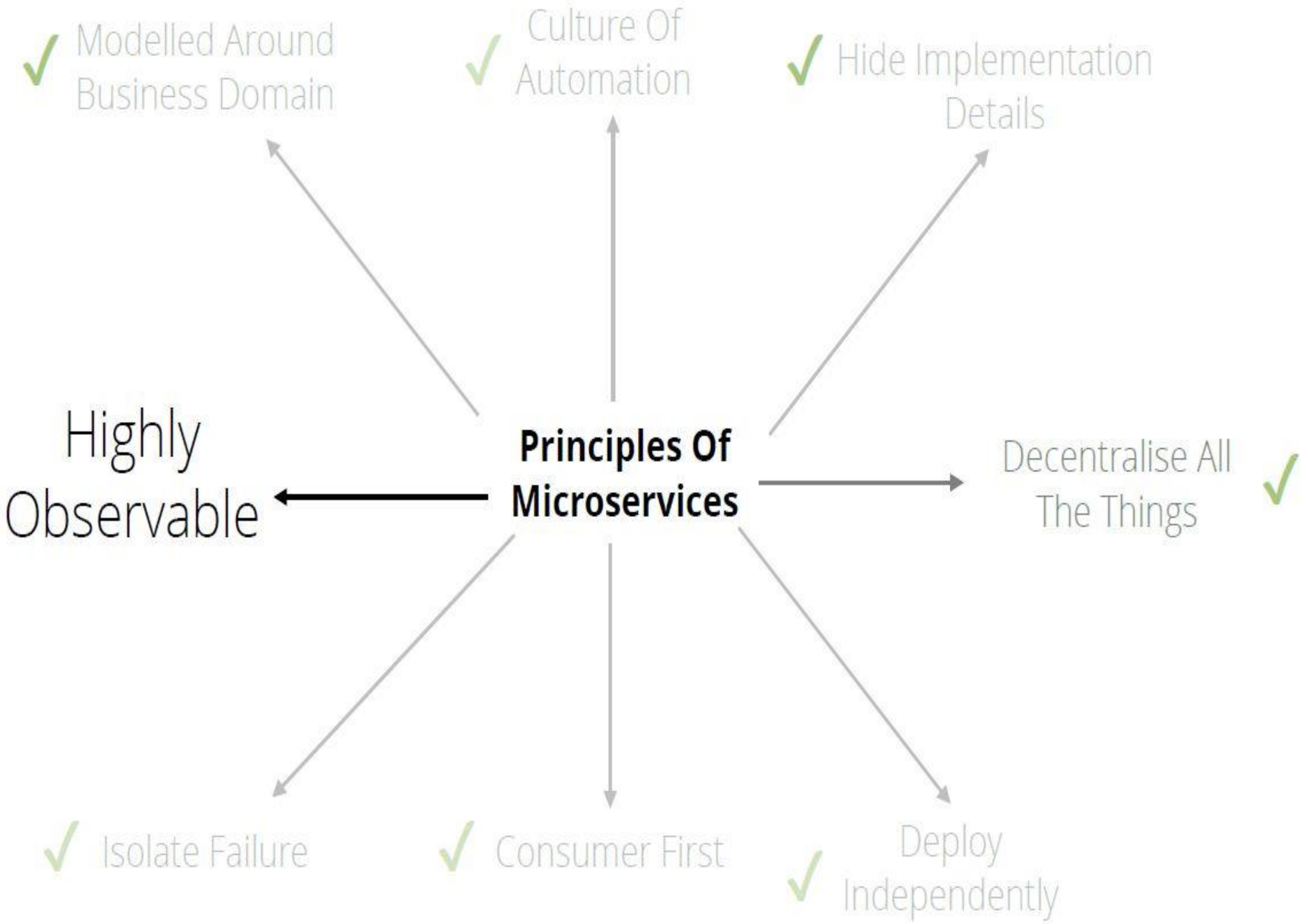

Requests Building Up
Strangler App
Thread Pool
Thread-pool exhausted
No requests to other downstream apps
Failing...slowly!
Legacy App
Legacy App
Legacy App

Fix **Timeouts**

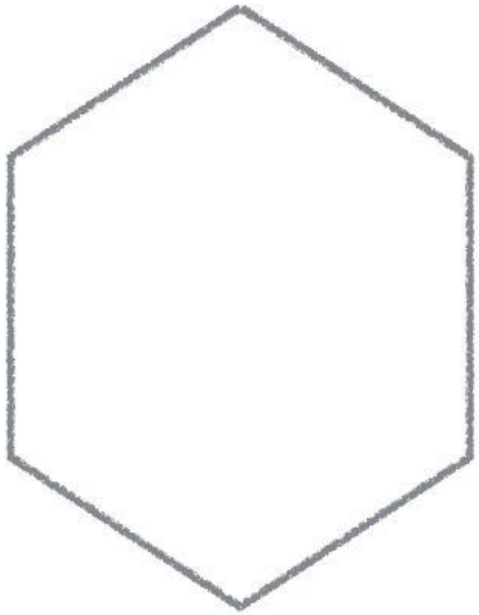


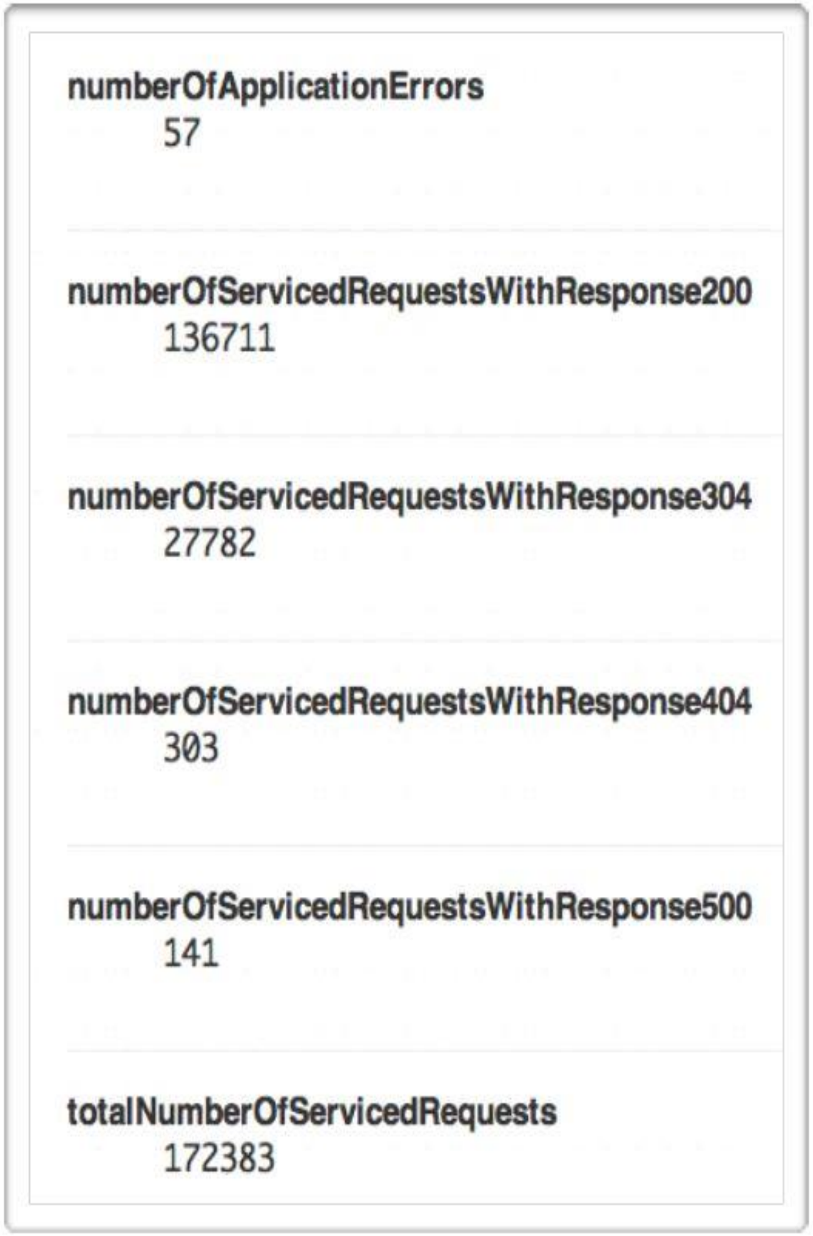
Bulkhead
Downstream
Connections


Strangler App
Thread Pool
Thread Pool
Thread Pool
Fix Timeouts
Bulkhead Downstream Connections
Circuit Breakers
Legacy App
Legacy App
Legacy App


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

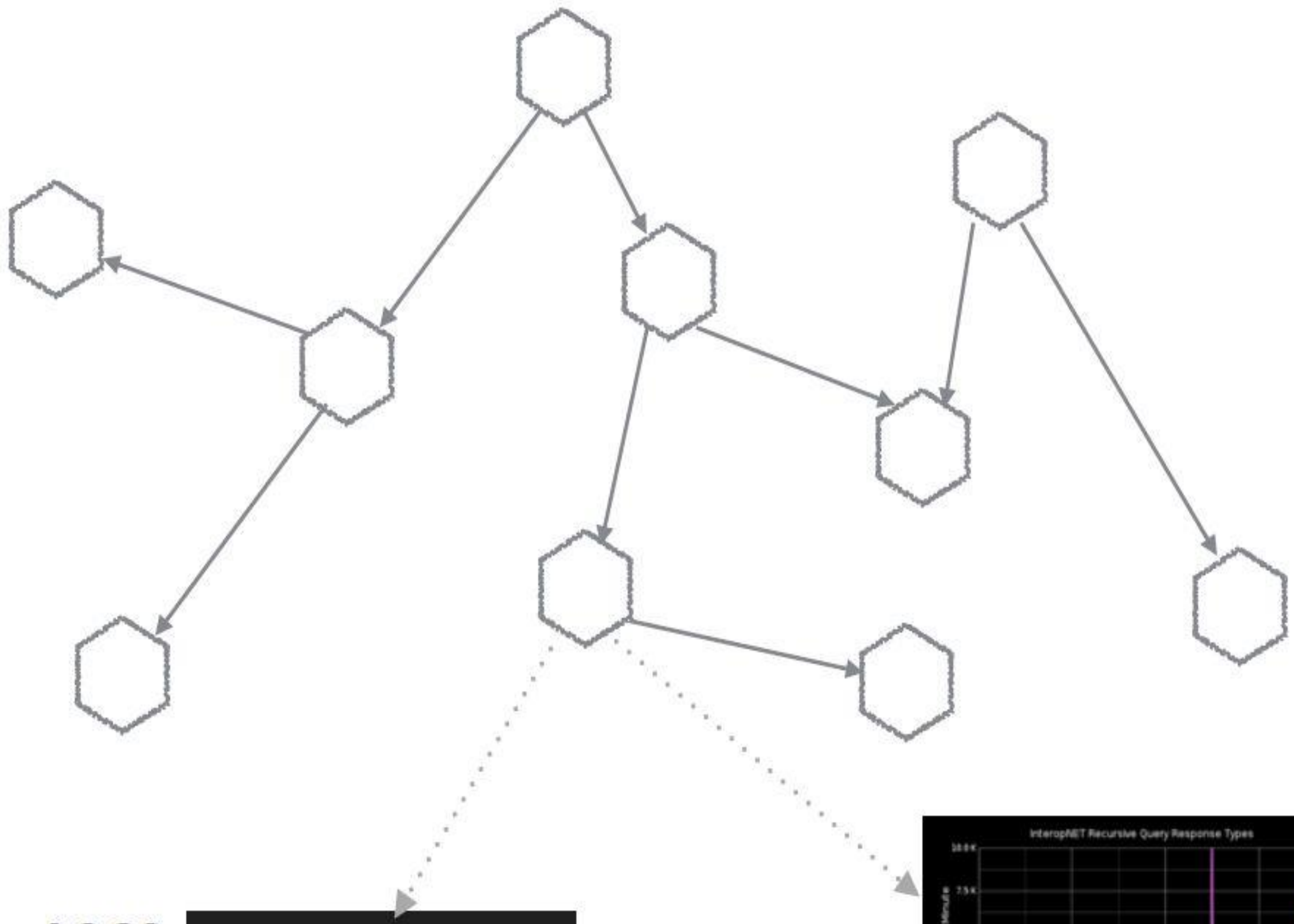
STATS PAGES





```
numberOfApplicationErrors
        57

numberOfServicedRequestsWithResponse200
        136711

numberOfServicedRequestsWithResponse304
        27782

numberOfServicedRequestsWithResponse404
        303

numberOfServicedRequestsWithResponse500
        141

totalNumberOfServicedRequests
        172383
```

AGGREGATION



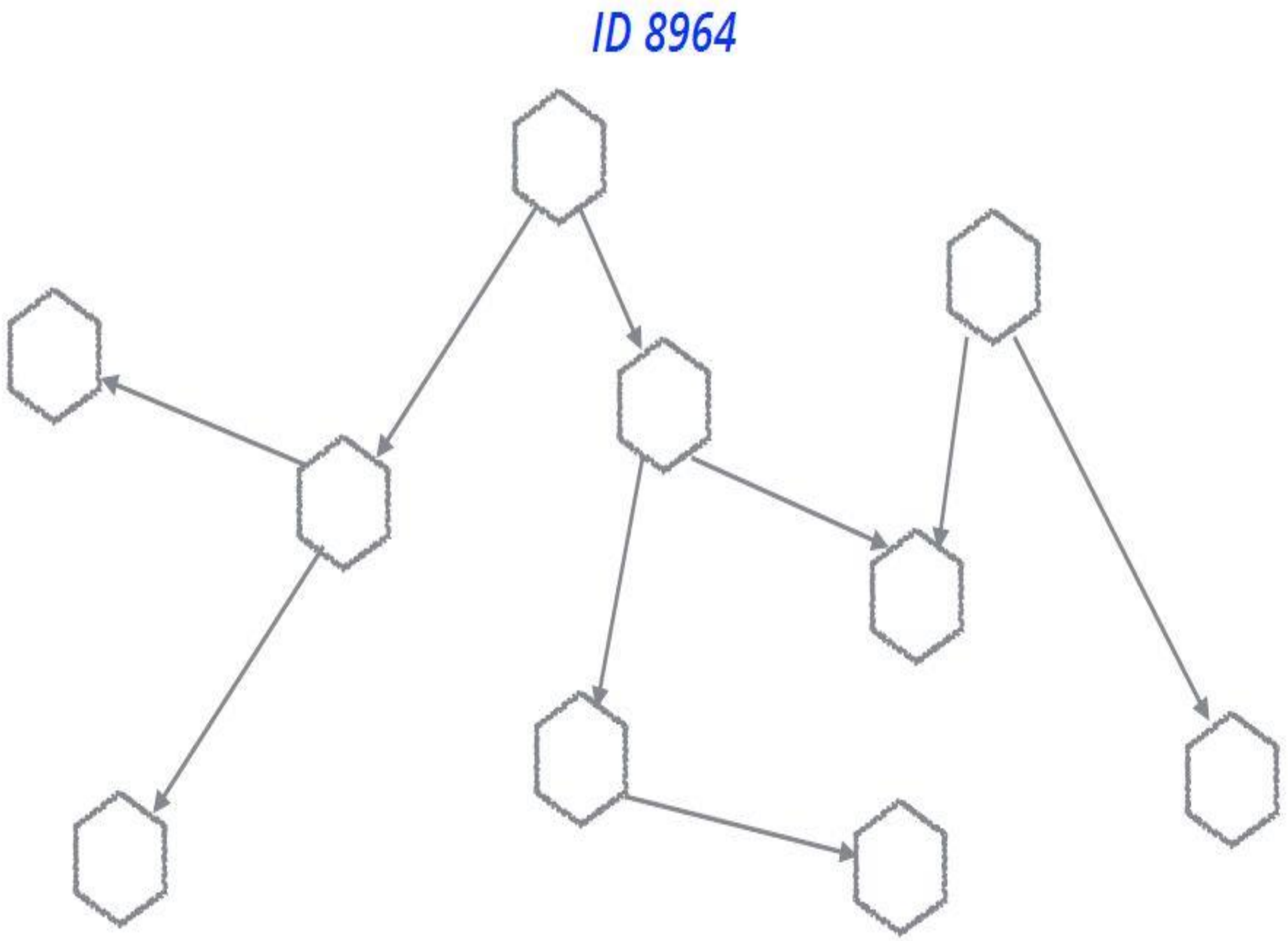
LOGS

kibana

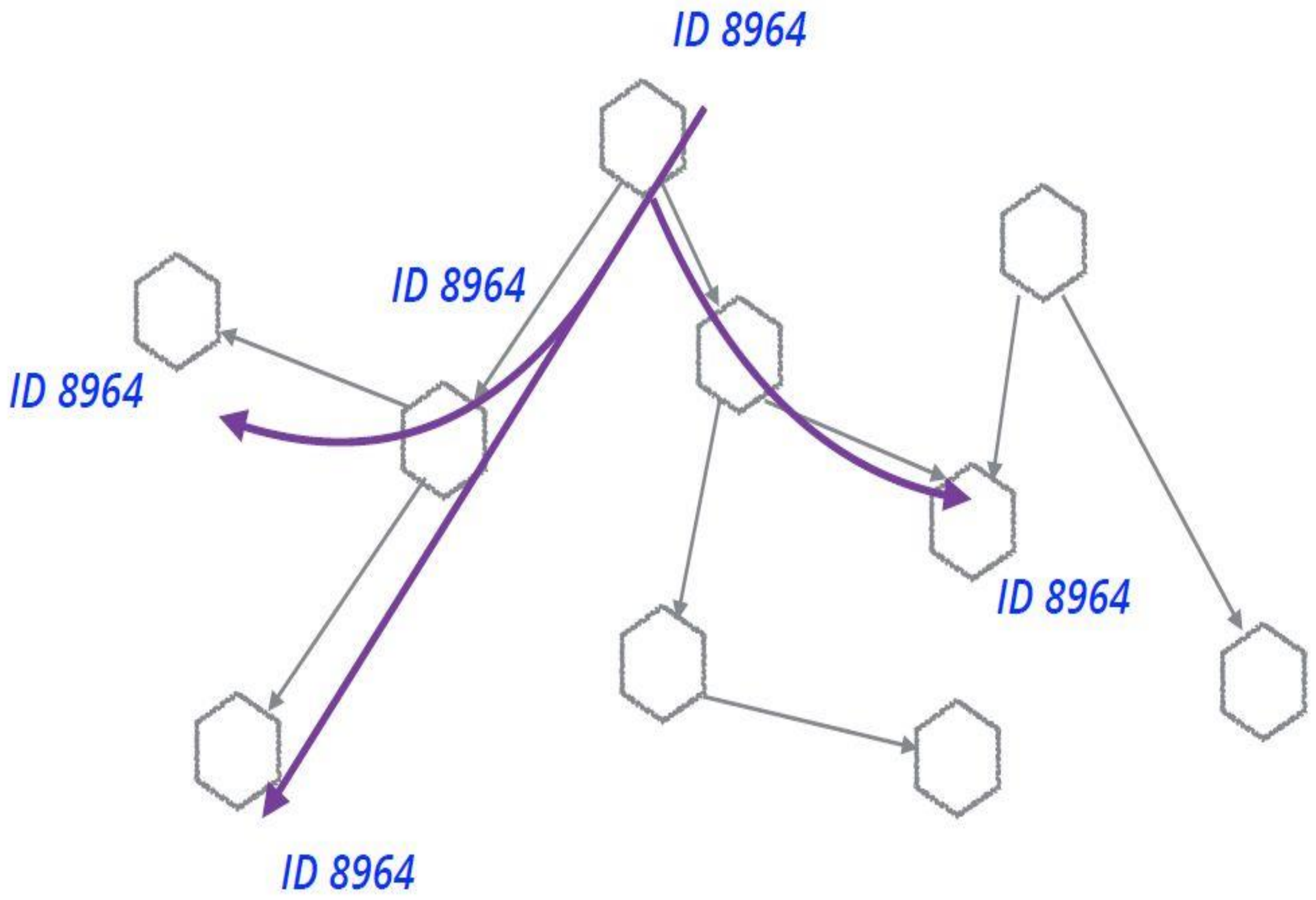
STATS

InteropNET Recursive Query Response Types
Queries Per Minute
10.0 K
7.5 K
5.0 K
2.5 K
0
05/11 12pm
05/12 12am
05/12 12pm
AAAA
A
PTR
NXDomain

CORRELATION IDS



CORRELATION IDS


ID 8964
ID 8964
ID 8964
ID 8964
ID 8964