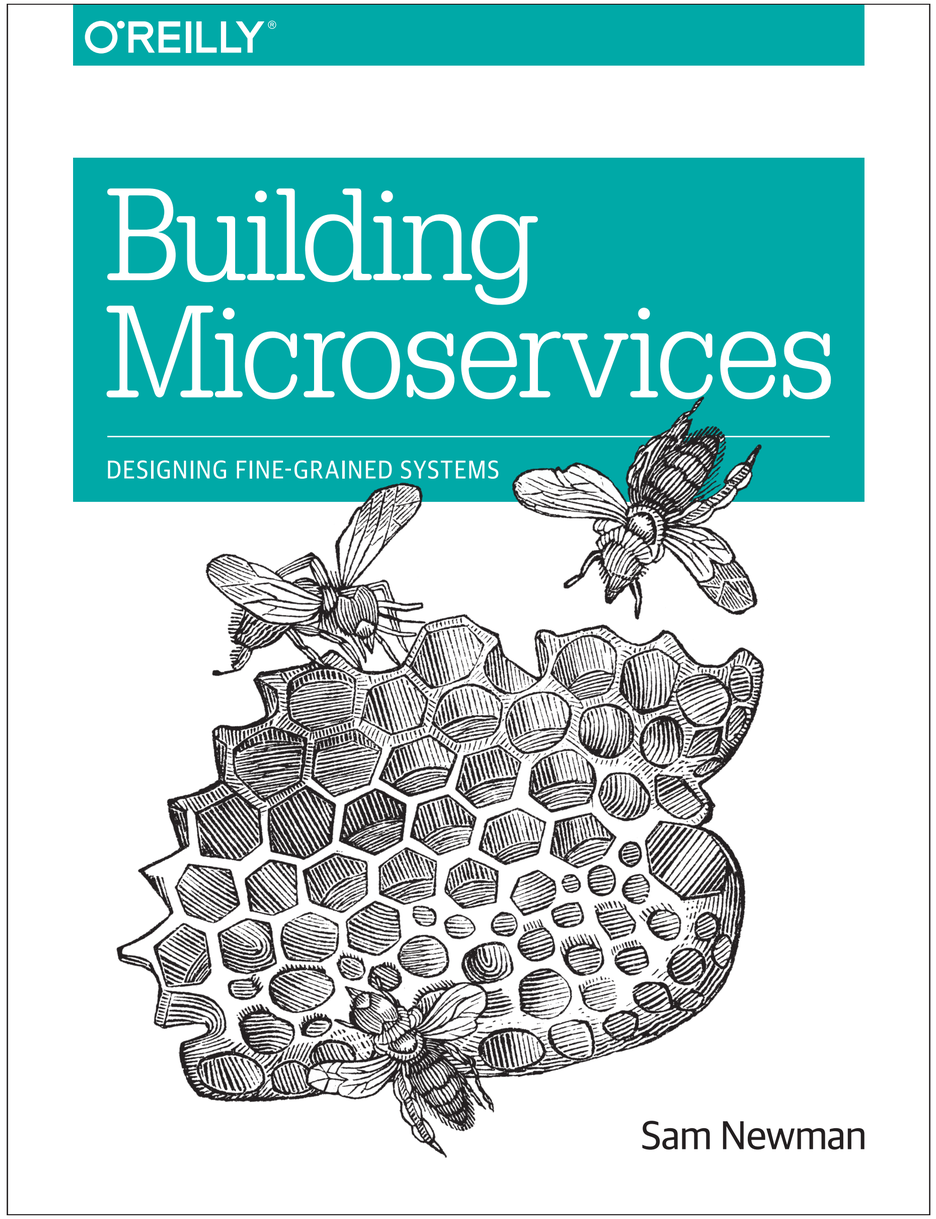
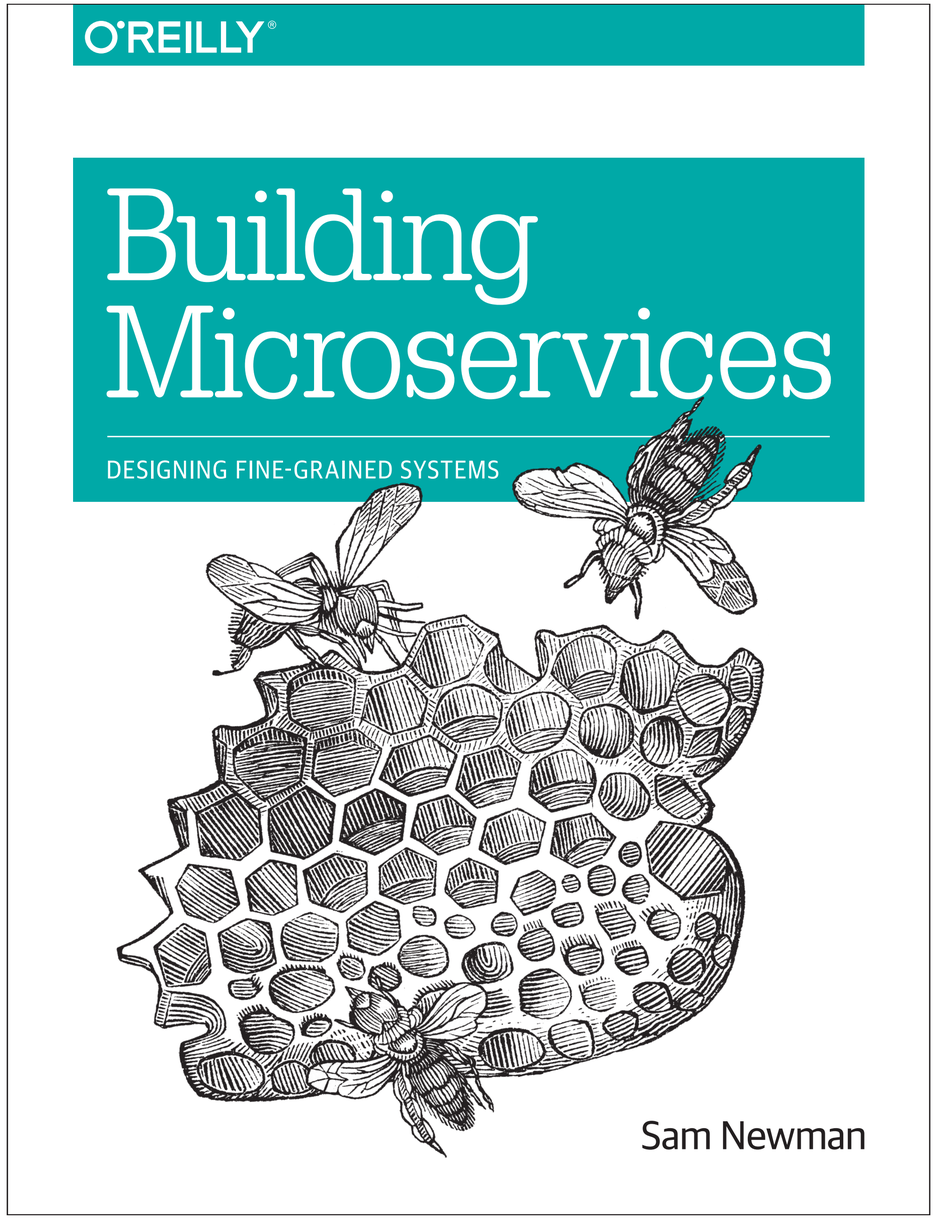
MONOLITH TO MICROSERVICES
APPLICATION DECOMPOSITION PATTERNS

Sam Newman

https://www.flickr.com/photos/140077762@N04/37364813975/


O'REILLY®
Building Microservices
DESIGNING FINE-GRAINED SYSTEMS
Sam Newman



Sam Newman
& Associates

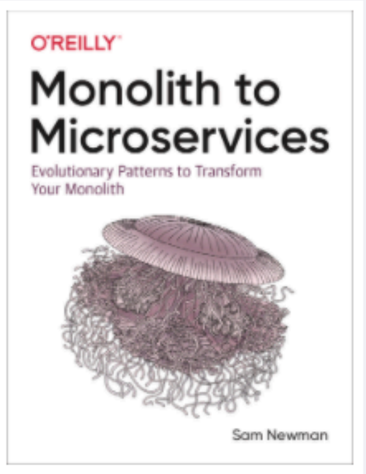
@samnewman

NEW BOOK!

Monolith To Microservices.

Monolith To Microservices is a new book on system decomposition from O'Reilly

How do you detangle a monolithic system and migrate it to a microservices architecture? How do you do it while maintaining business-as-usual? As a companion to Building Microservices, this new book details multiple approaches for helping you transition from existing monolthic systems to microservice architectures. This book is ideal if you're looking to evolve your current systems, rather than just rewriting everything from scratch.



How To Get The Book.

The book is now available, and you can order the dead-tree and Kindle versions over at Amazon. You can also read the book online on O'Reilly's online learning platform.

Read on O'Reilly Learning Online | Order at Amazon.com

Order at Amazon.co.uk

https://samnewman.io/books/monolith-to-microservices/

OTHER COURSES

http://bit.ly/snewman-olt

OTHER COURSES

http://bit.ly/snewman-olt

OTHER COURSES

http://bit.ly/snewman-olt

3 hour session

3 hour session

Two 10-15 breaks

3 hour session

Two 10-15 breaks

Please ask questions using the Q&A widget

The session will be recorded, and recordings will be available 24-48 hours after the class ends

AGENDA

AGENDA

Introduction

AGENDA

Introduction

Migration Approach

AGENDA

Introduction

Migration Approach

Application Refactoring

AGENDA

Introduction

Migration Approach

Application Refactoring

UI Decomposition

AGENDA

Introduction

Migration Approach

Application Refactoring

UI Decomposition

POLL: WHAT IS YOUR EXPERIENCE LEVEL WITH MICROSERVICES?

Only thinking about using them

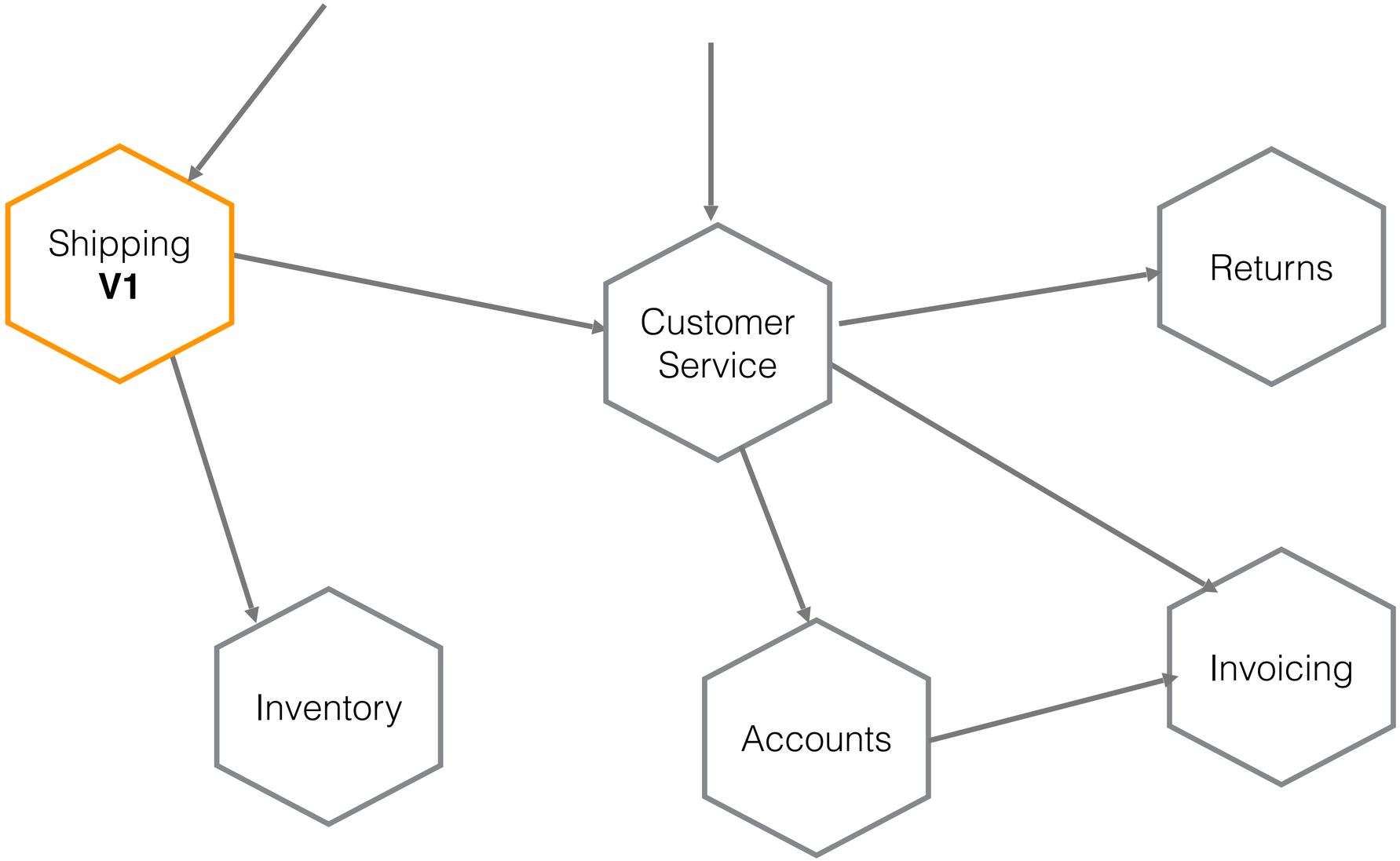
Just started using them

1-2 years experience

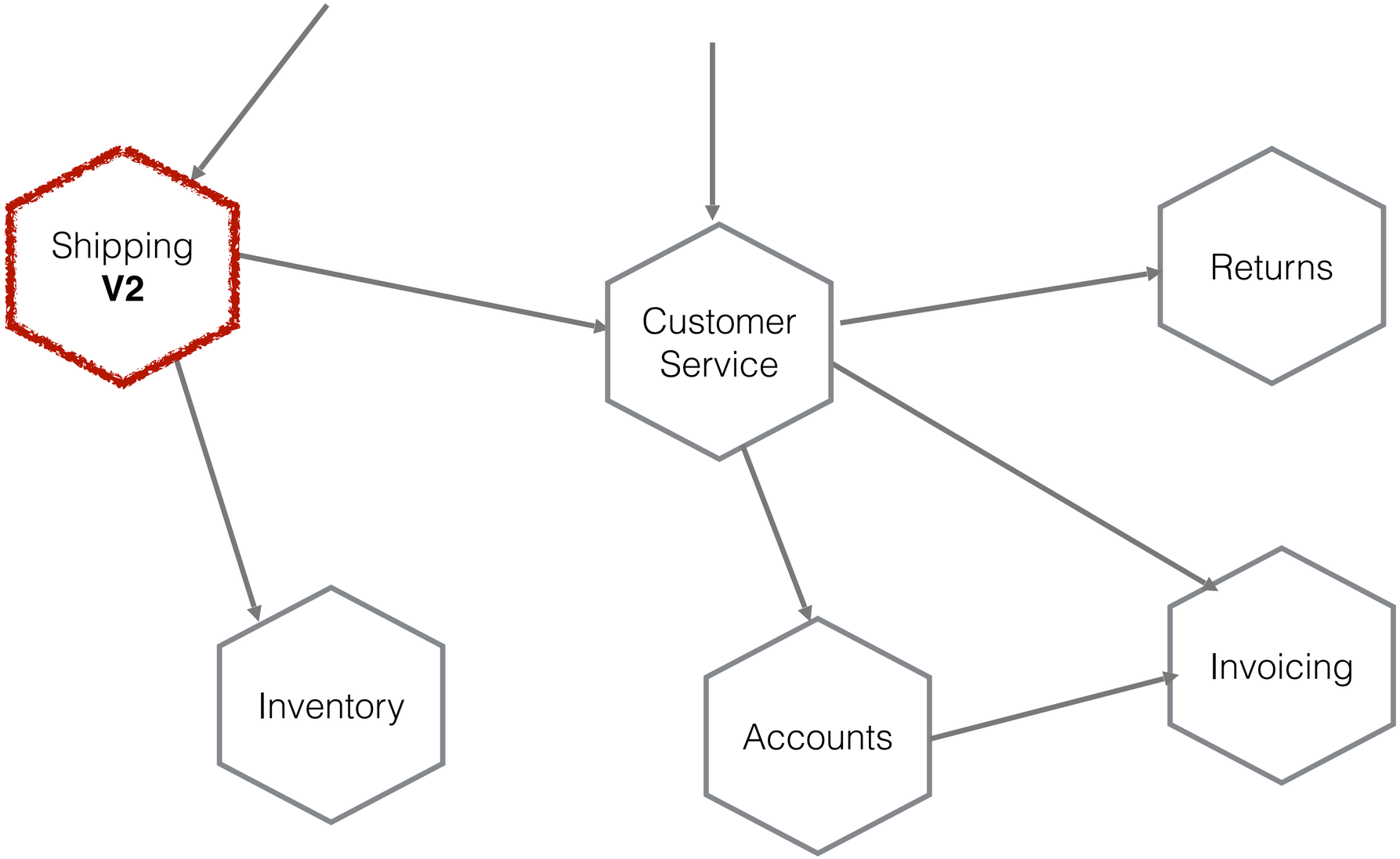
2+ years experience

What are microservices?

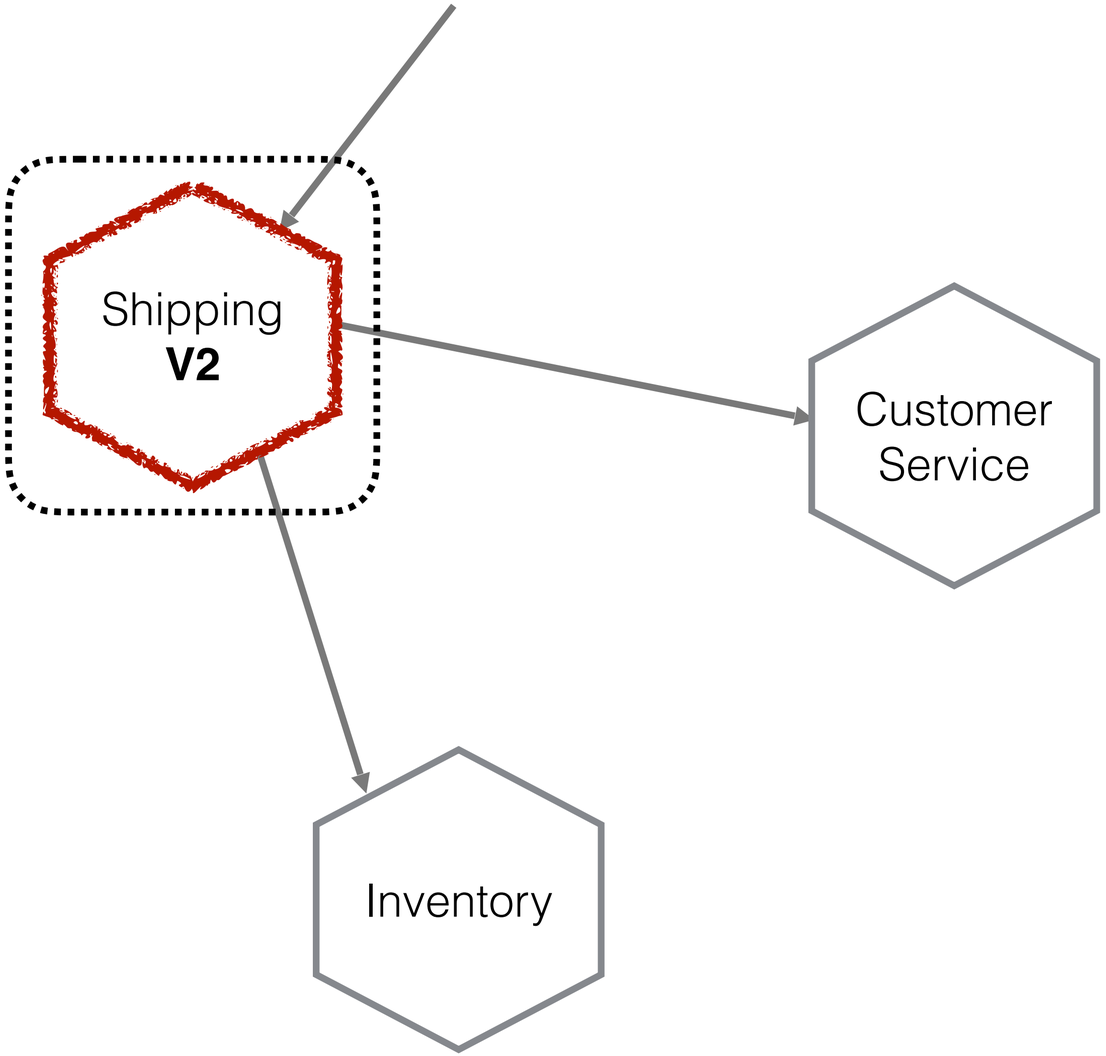
INDEPENDENT DEPLOYABILITY



INDEPENDENT DEPLOYABILITY

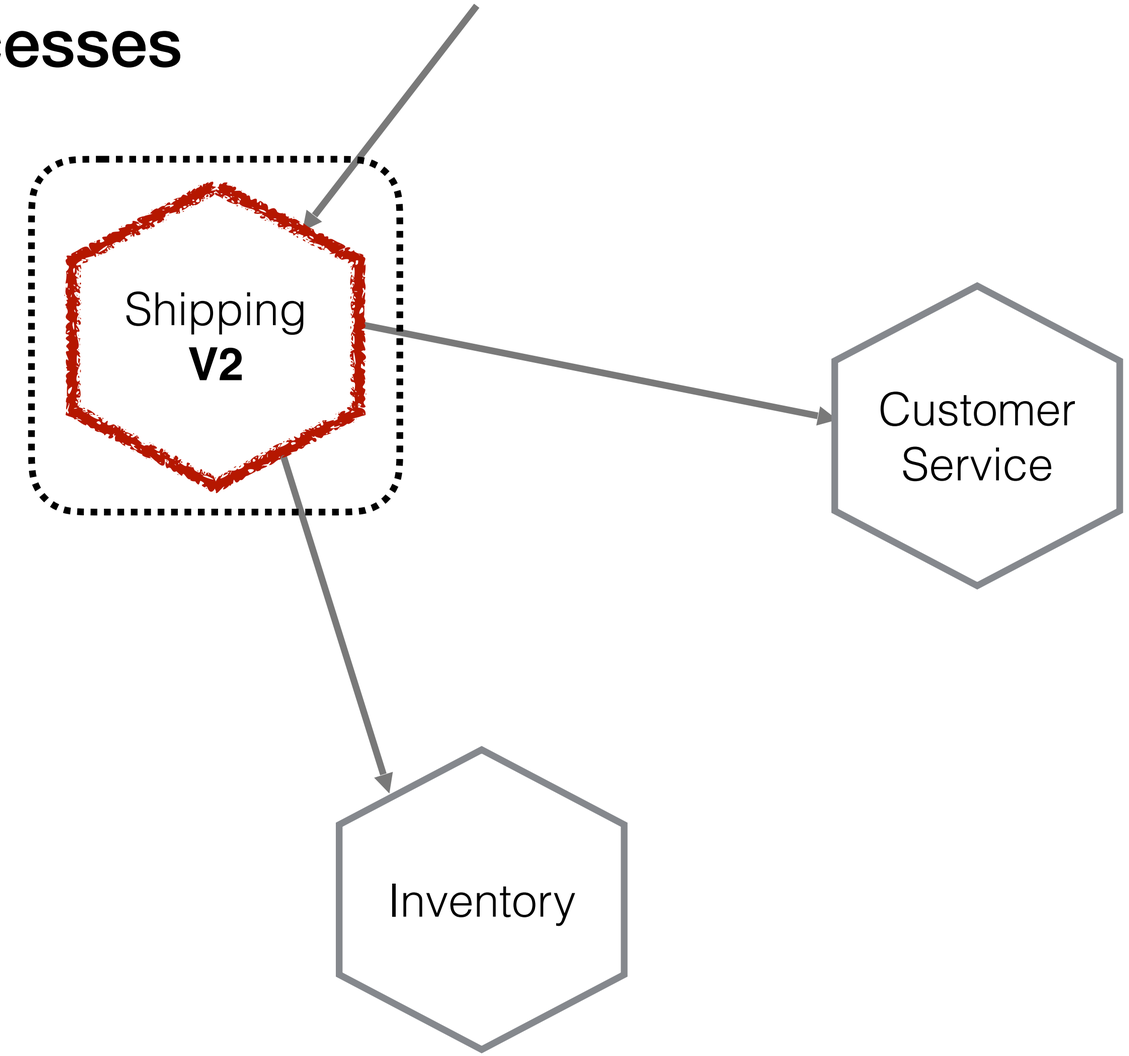


ANATOMY OF A MICROSERVICE



ANATOMY OF A MICROSERVICE

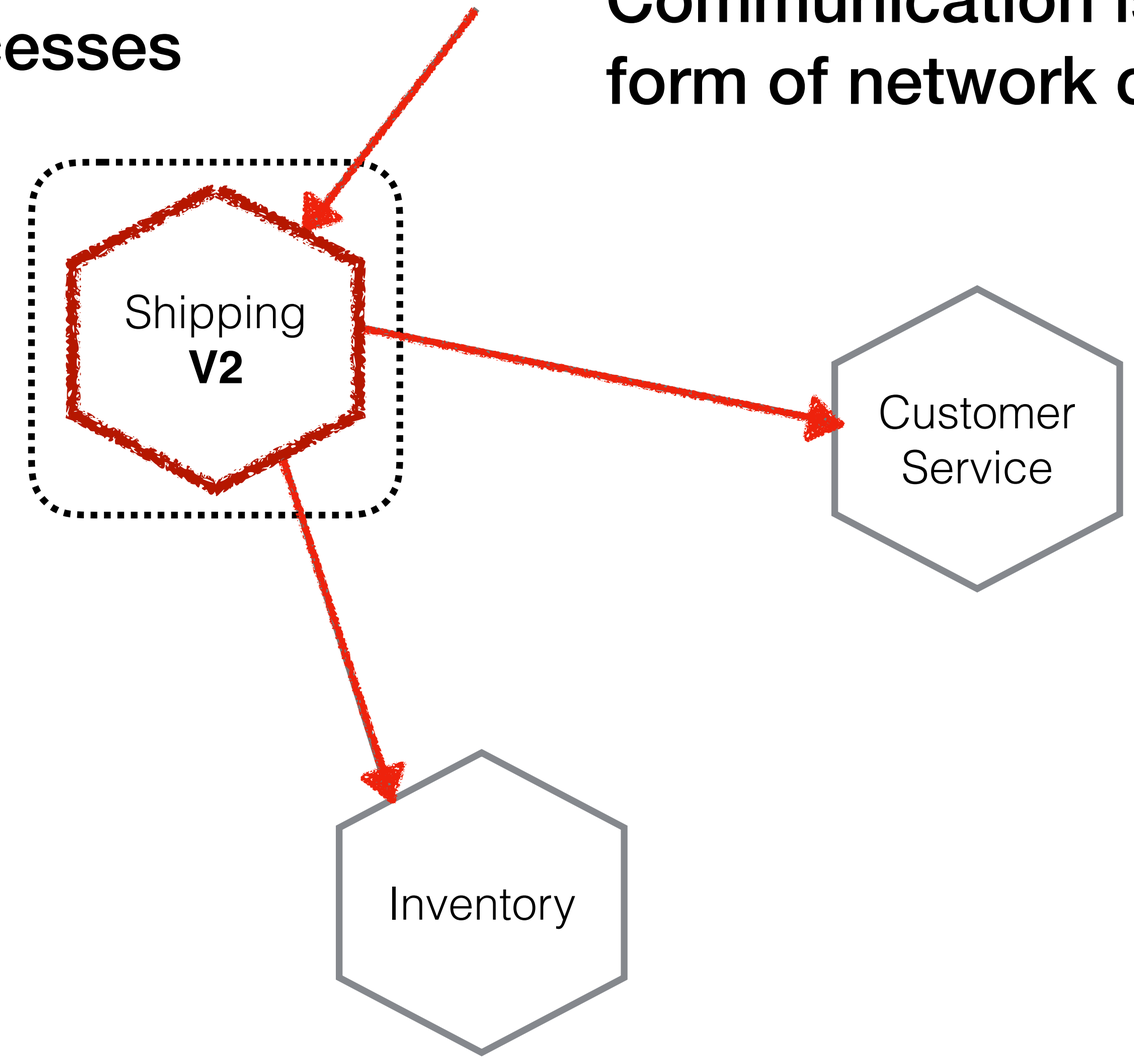
Microservice instances are typically separate processes



ANATOMY OF A MICROSERVICE

Microservice instances are typically separate processes

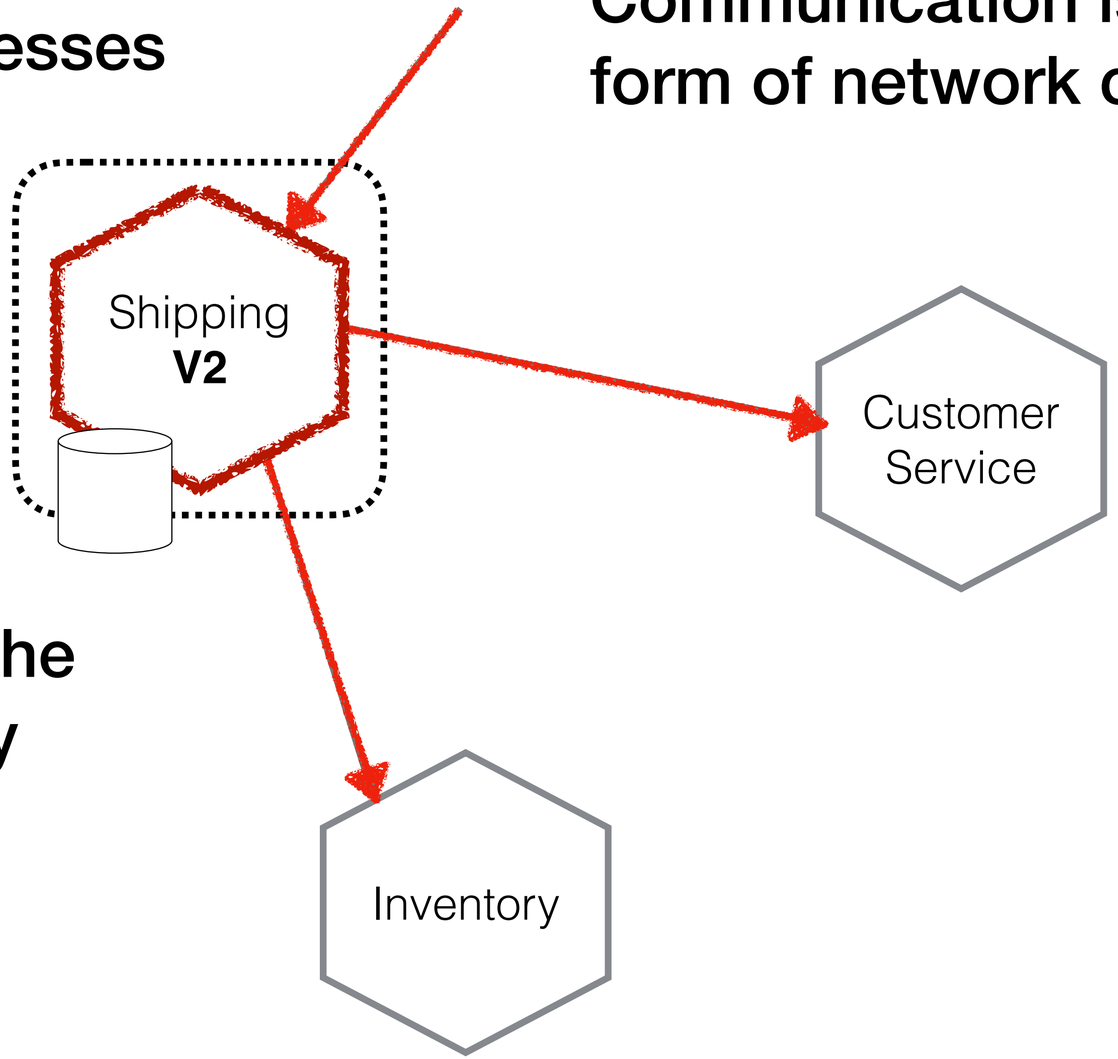
Communication is via some form of network call



ANATOMY OF A MICROSERVICE

Microservice instances are typically separate processes

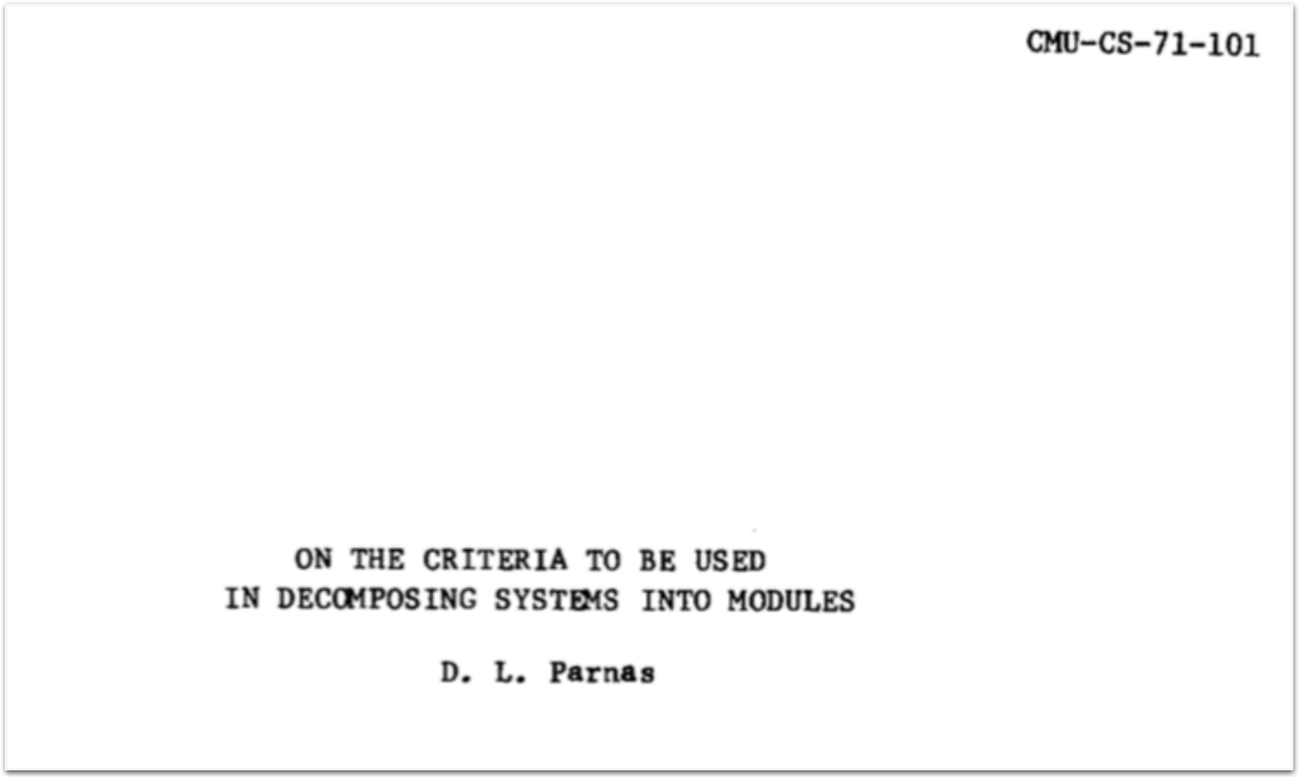
Communication is via some form of network call



Data is hidden inside the microservice boundary

Why separate database?

INFORMATION HIDING



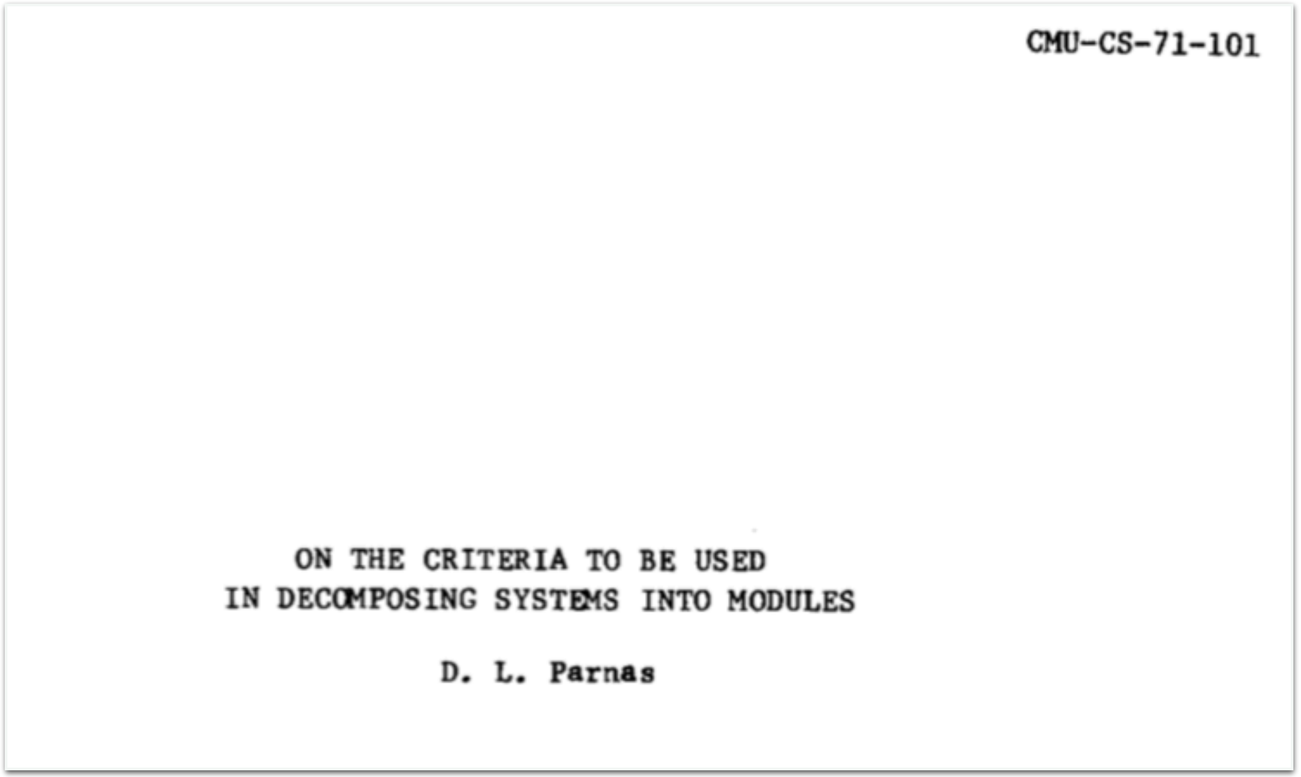
CMU-CS-71-101

ON THE CRITERIA TO BE USED
IN DECOMPOSING SYSTEMS INTO MODULES

D. L. Parnas

http://repository.cmu.edu/cgi/viewcontent.cgi?article=2979&context=compsci

INFORMATION HIDING



CMU-CS-71-101

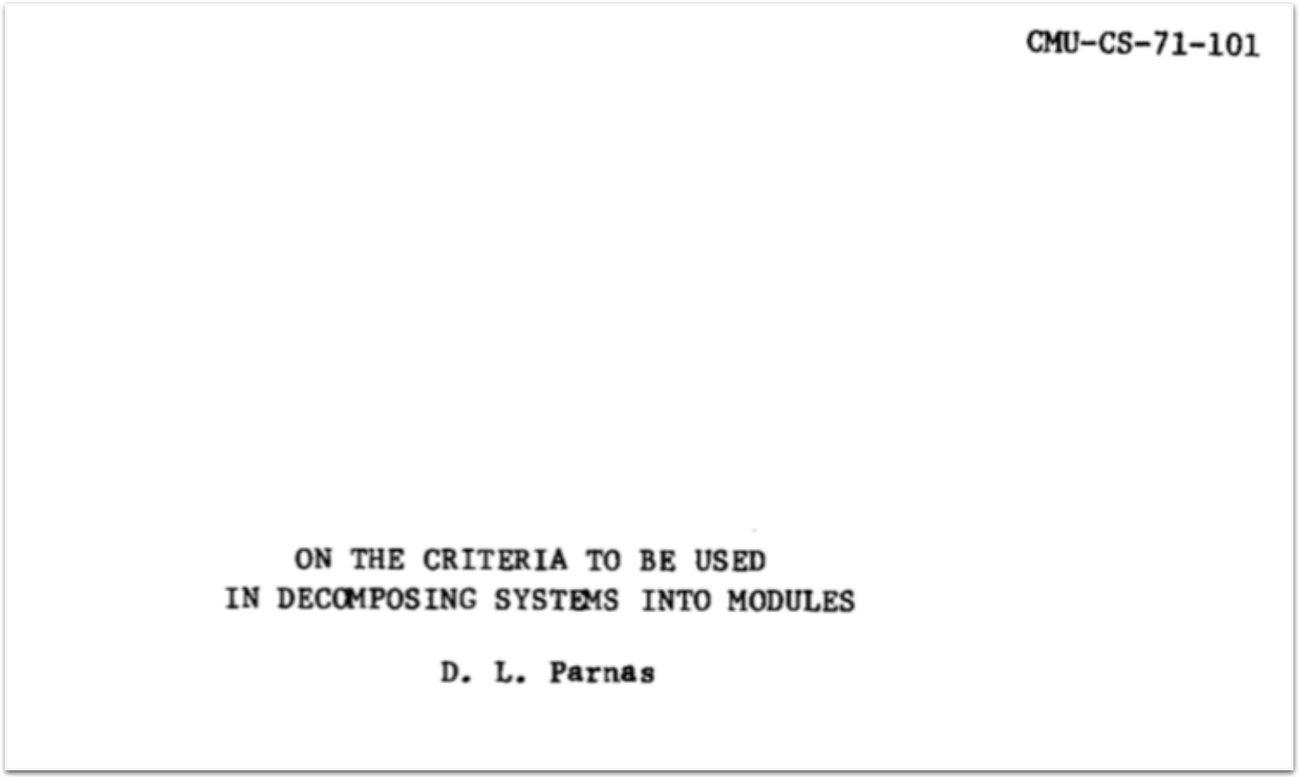
ON THE CRITERIA TO BE USED
IN DECOMPOSING SYSTEMS INTO MODULES

D. L. Parnas

Published in 1971

http://repository.cmu.edu/cgi/viewcontent.cgi?article=2979&context=compsci

INFORMATION HIDING



CMU-CS-71-101

ON THE CRITERIA TO BE USED
IN DECOMPOSING SYSTEMS INTO MODULES

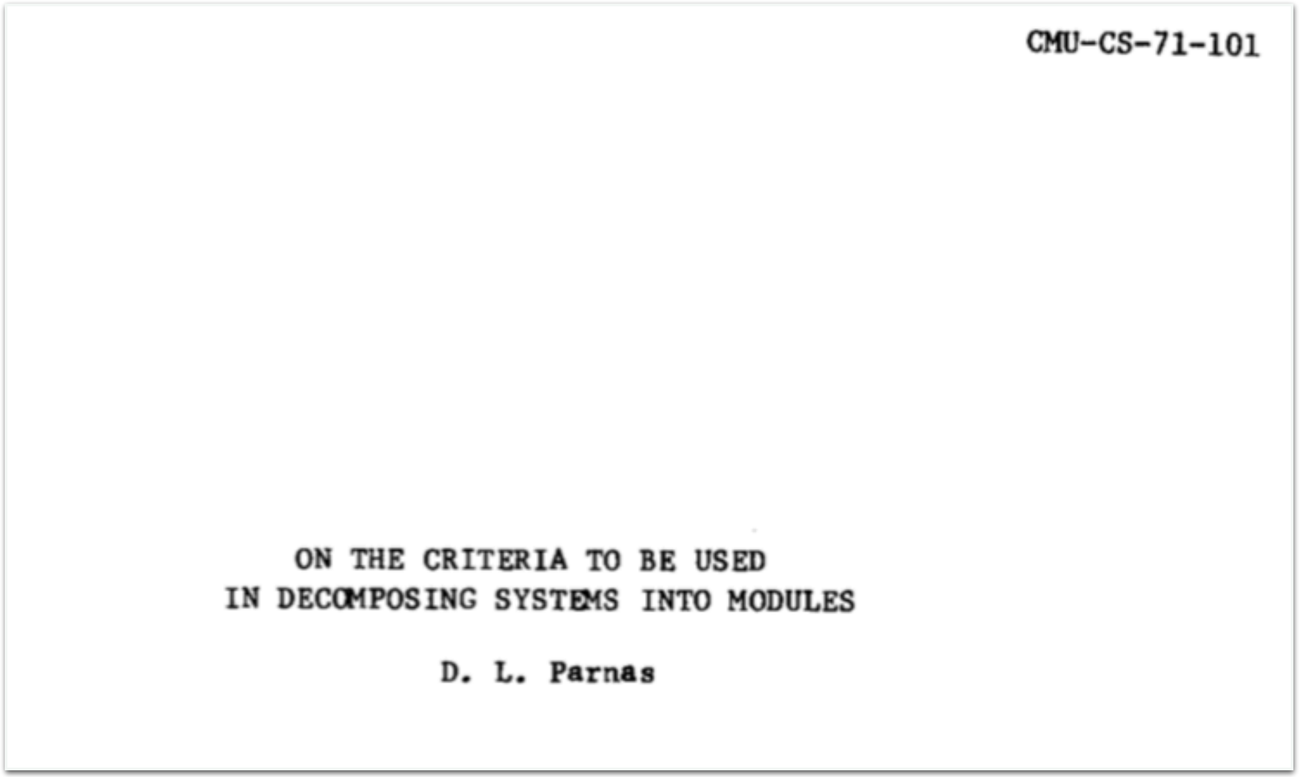
D. L. Parnas

Published in 1971

Looked at how best to define module boundaries

http://repository.cmu.edu/cgi/viewcontent.cgi?article=2979&context=compsci

INFORMATION HIDING



CMU-CS-71-101

ON THE CRITERIA TO BE USED
IN DECOMPOSING SYSTEMS INTO MODULES

D. L. Parnas

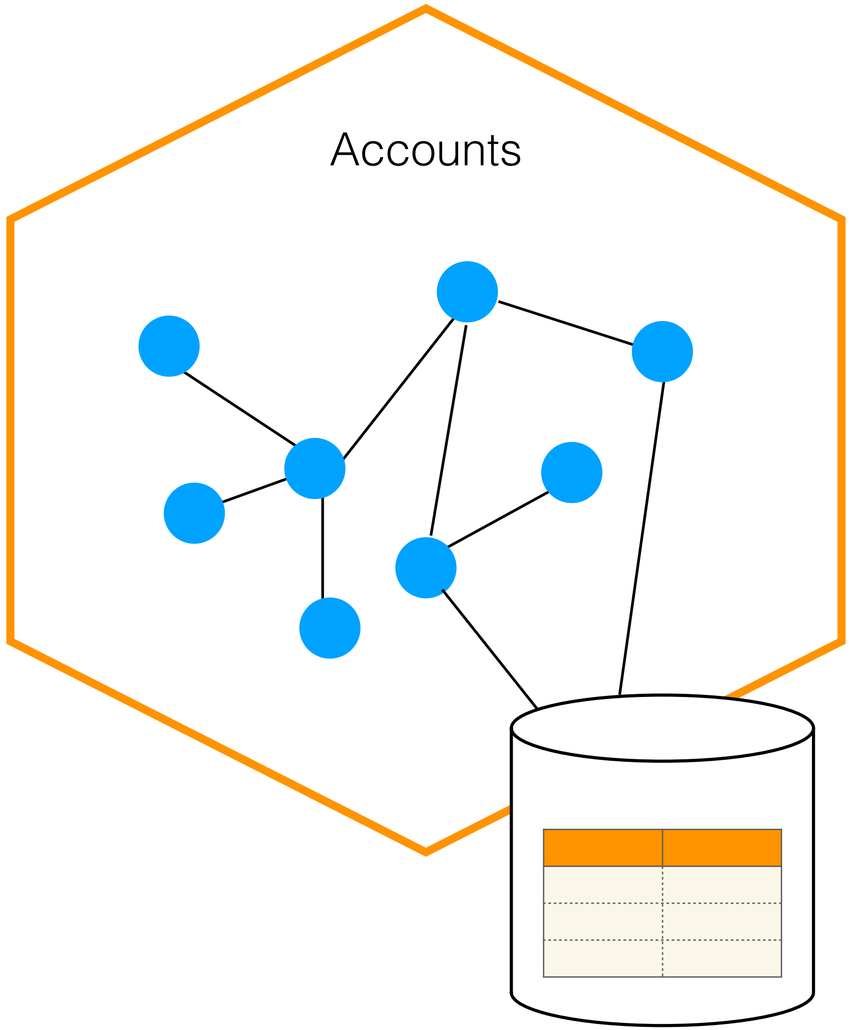
Published in 1971

Looked at how best to define module boundaries

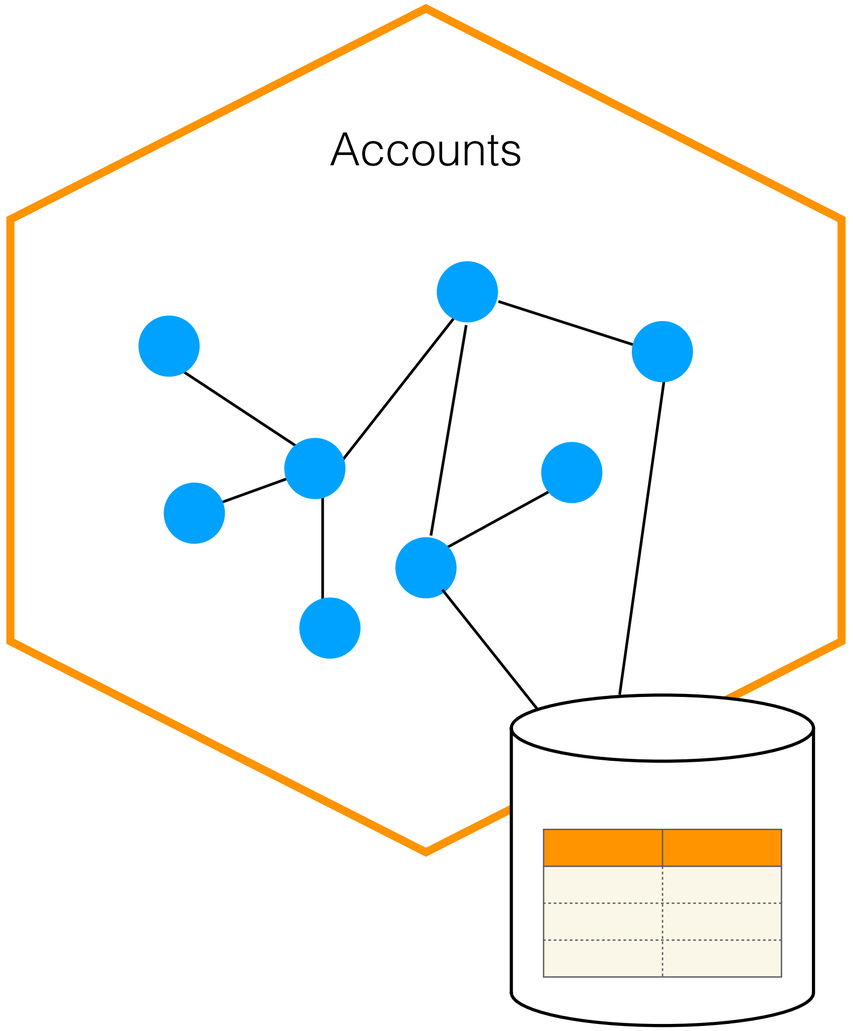
Found that “information hiding” worked best

http://repository.cmu.edu/cgi/viewcontent.cgi?article=2979&context=compsci

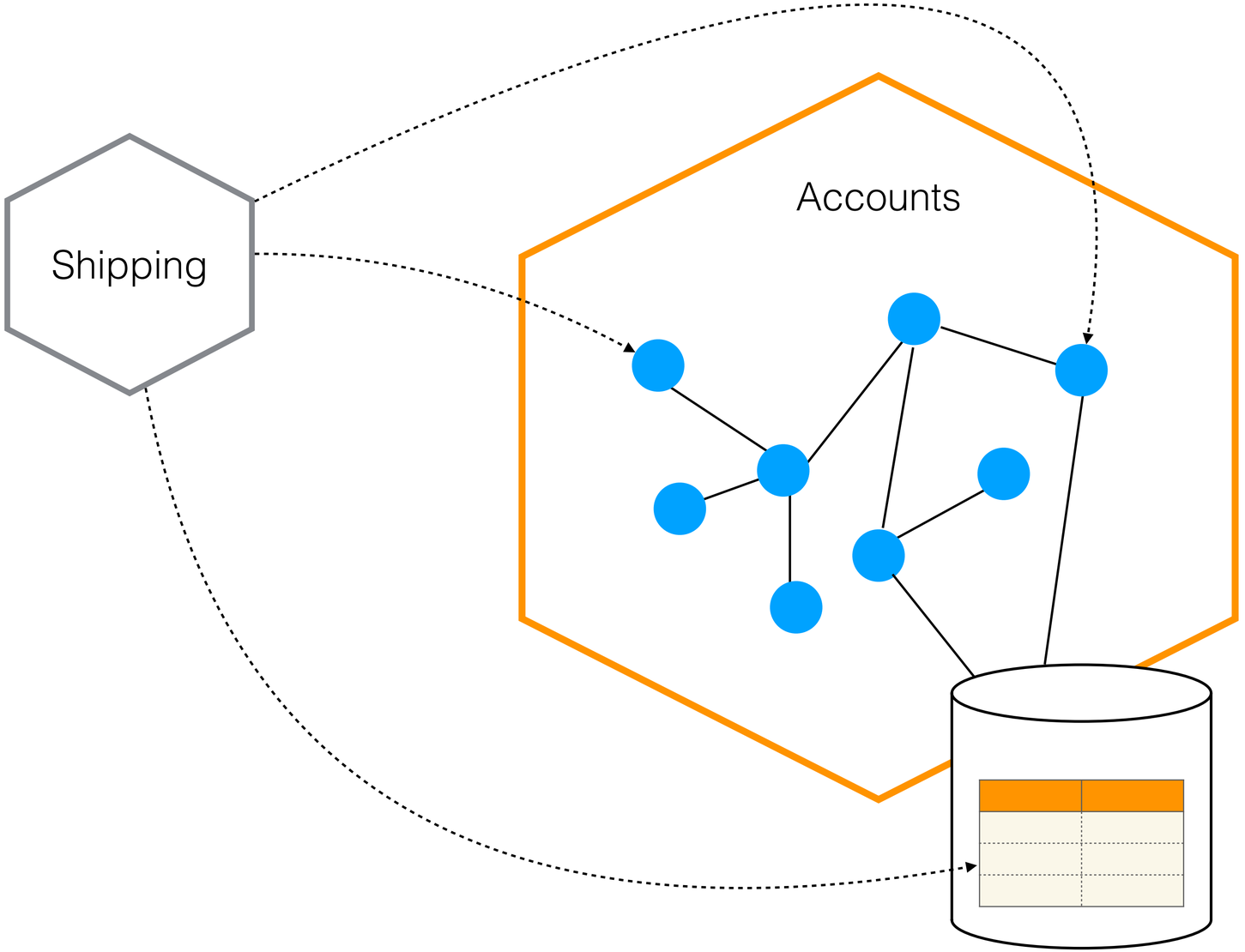
NOT HIDING?



NOT HIDING?

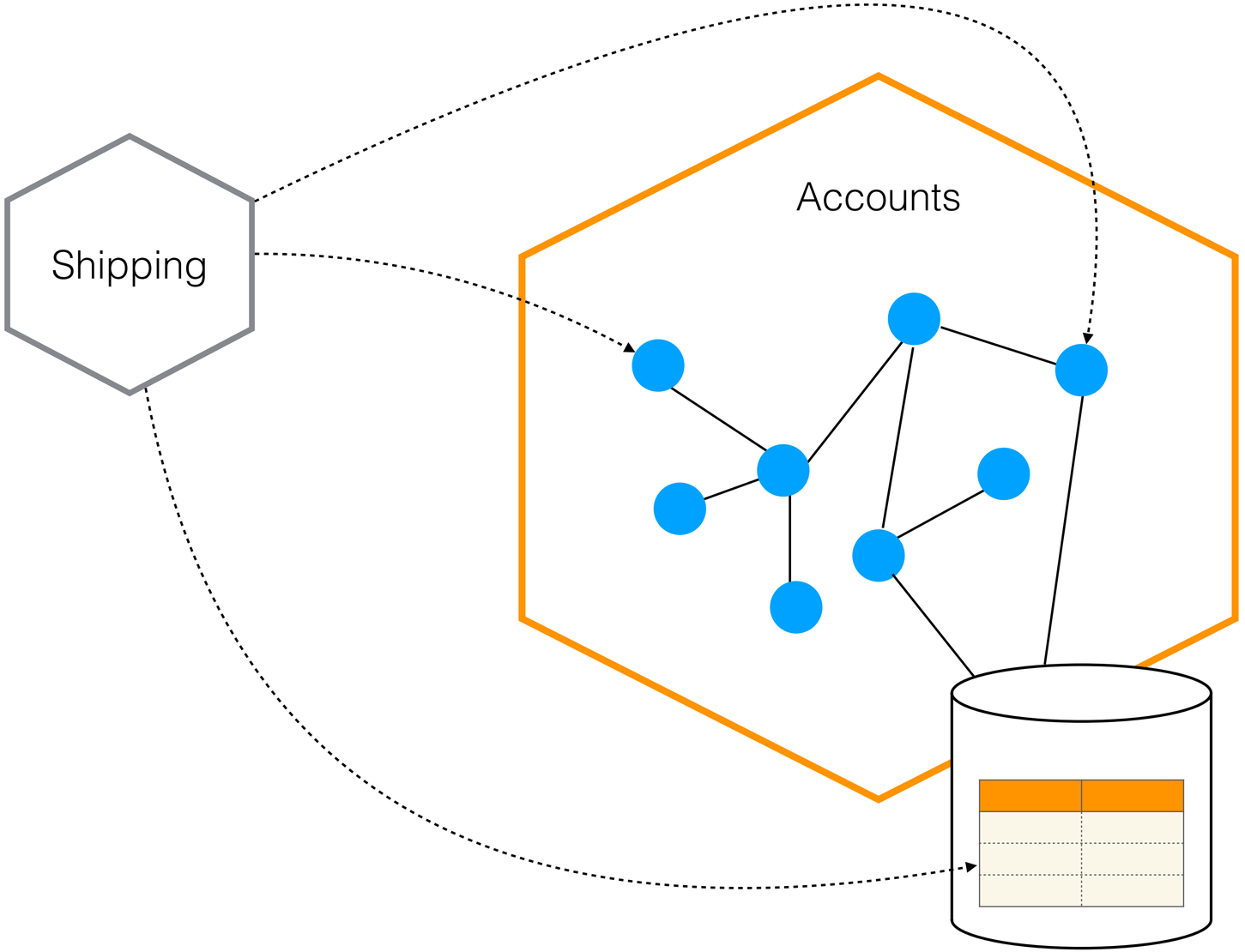


NOT HIDING?



If an upstream consumer can reach into your internal implementation..

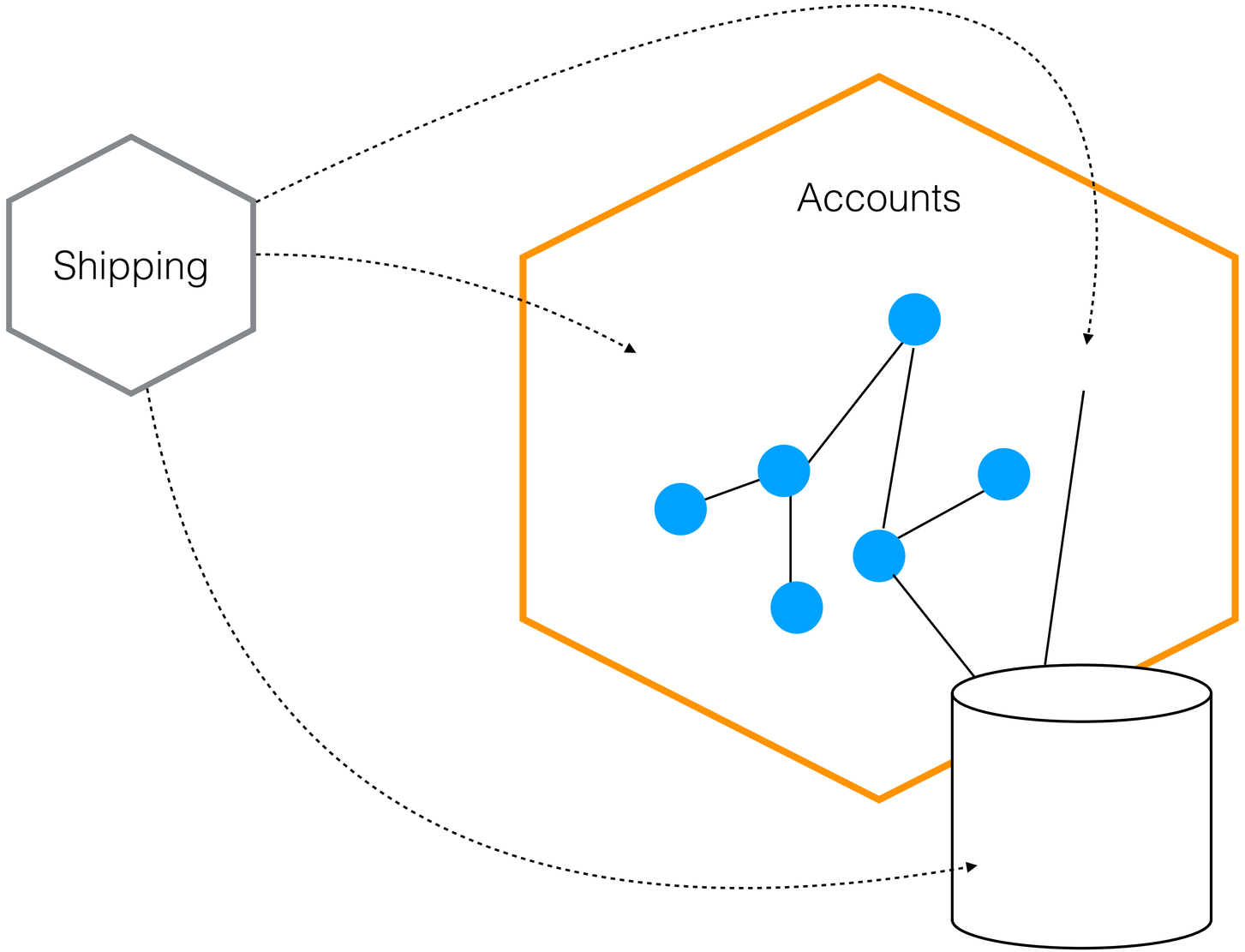
NOT HIDING?



If an upstream consumer can reach into your internal implementation..

…then you can't change this implementation without breaking the consumer

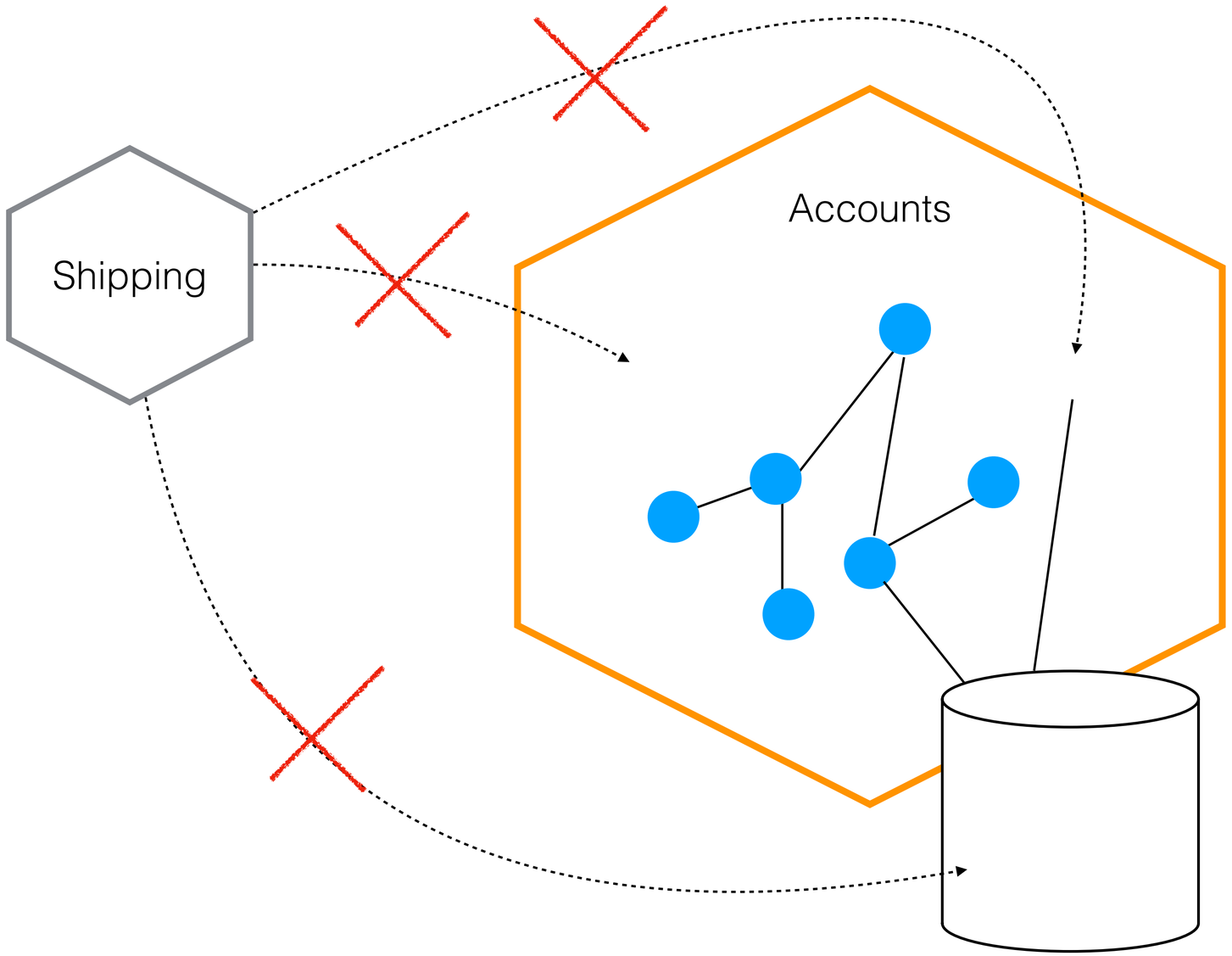
NOT HIDING?



If an upstream consumer can reach into your internal implementation..

…then you can't change this implementation without breaking the consumer

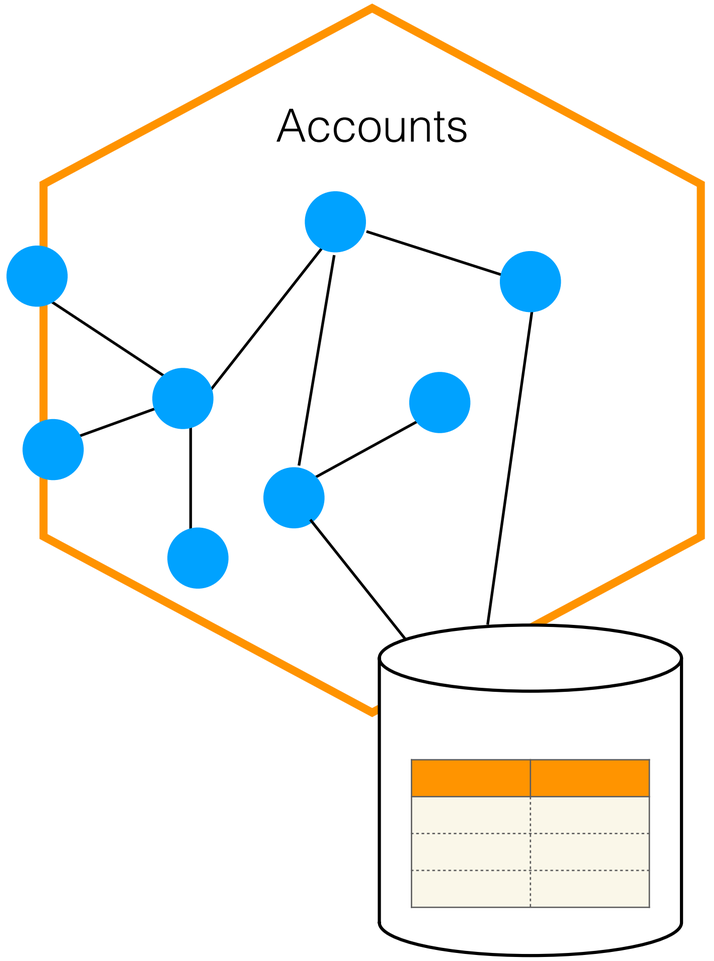
NOT HIDING?



If an upstream consumer can reach into your internal implementation..

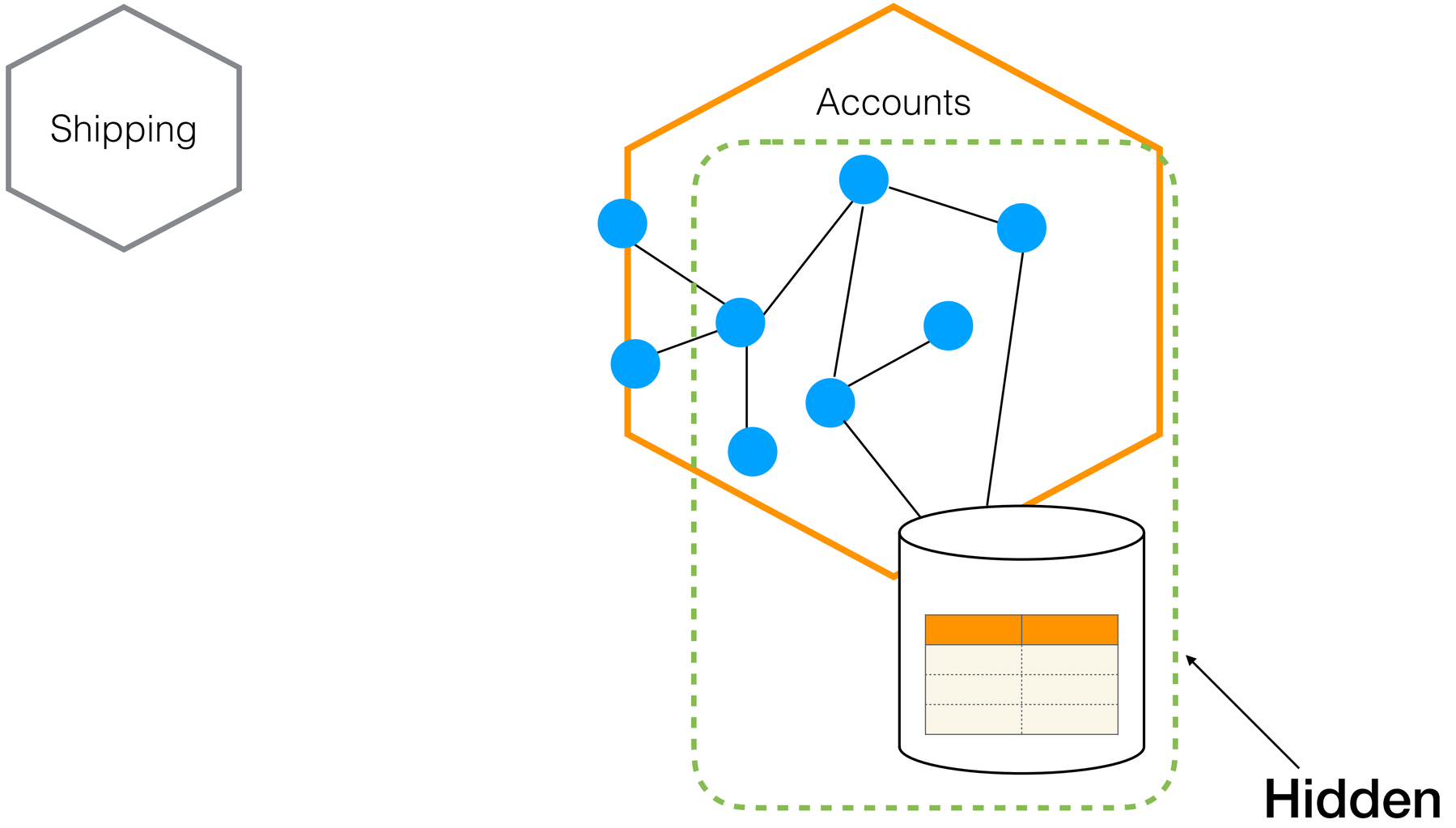
…then you can't change this implementation without breaking the consumer

HIDING!



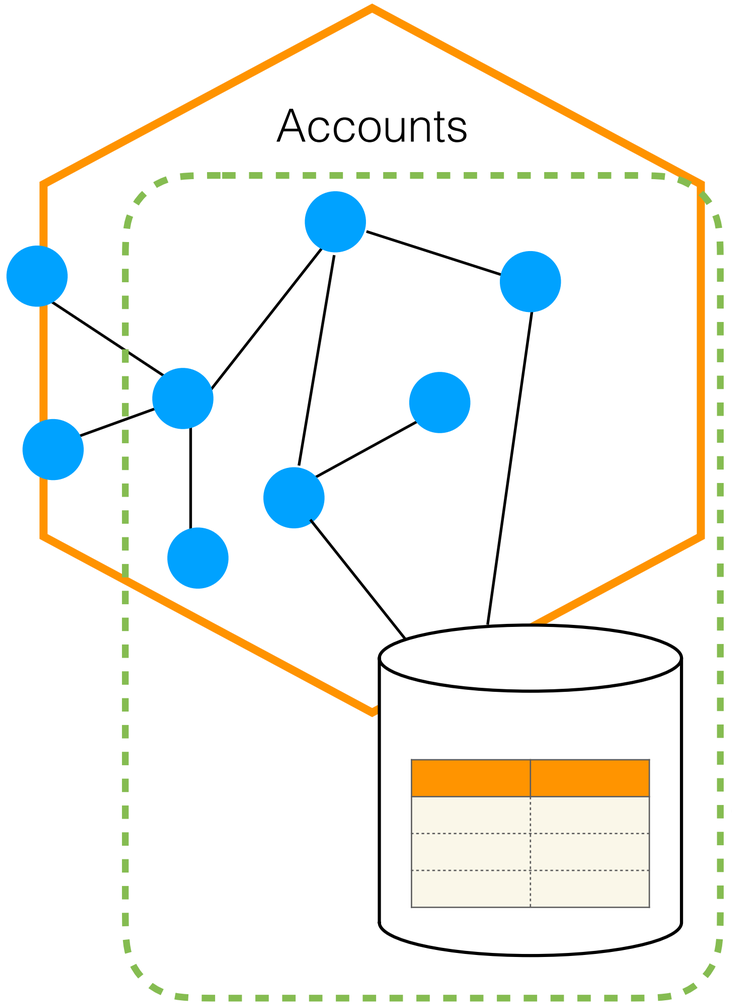
HIDING!

Hide your secrets!



HIDING!

Shipping



Hide your secrets!

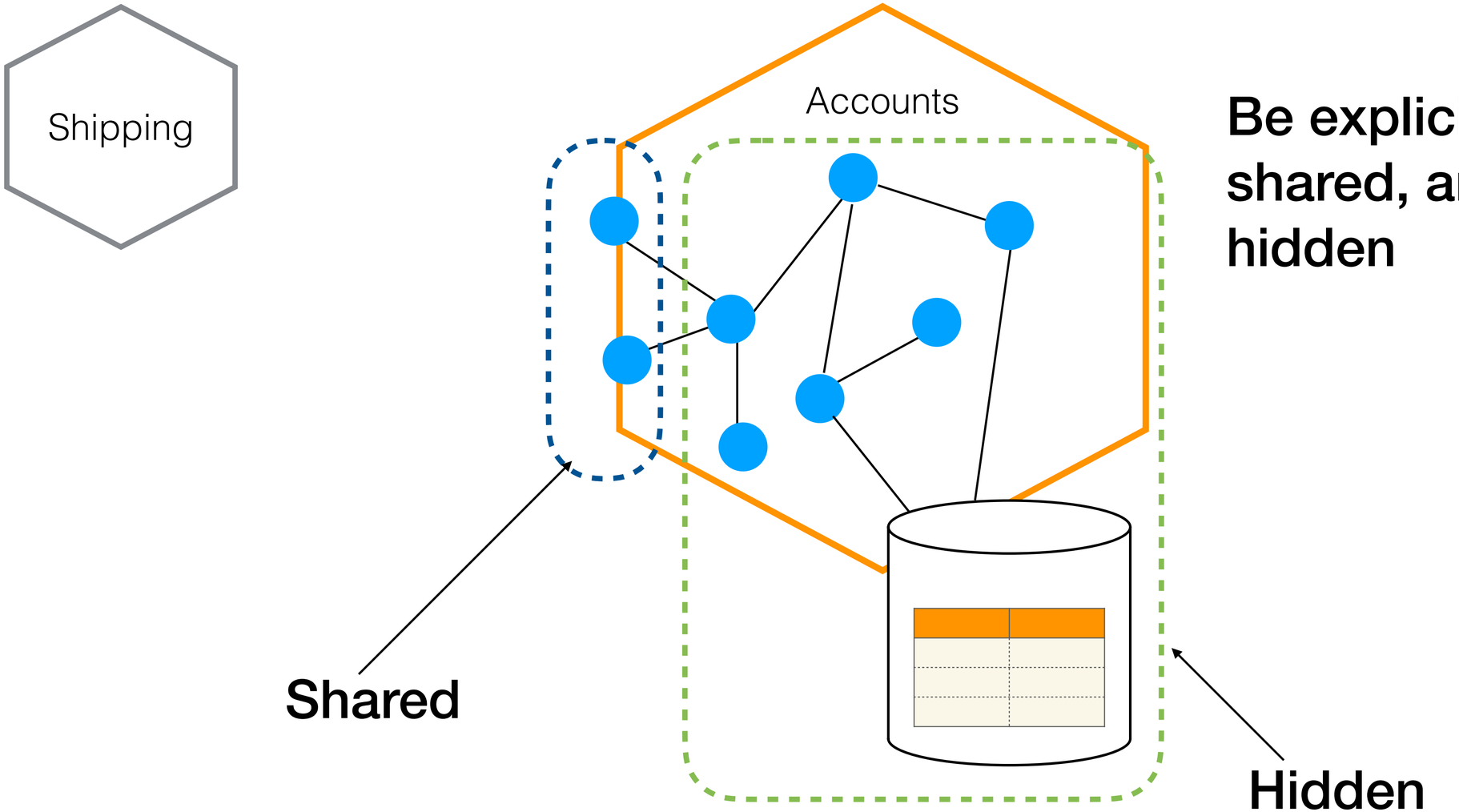
Be explicit about what is shared, and what is hidden

Hidden

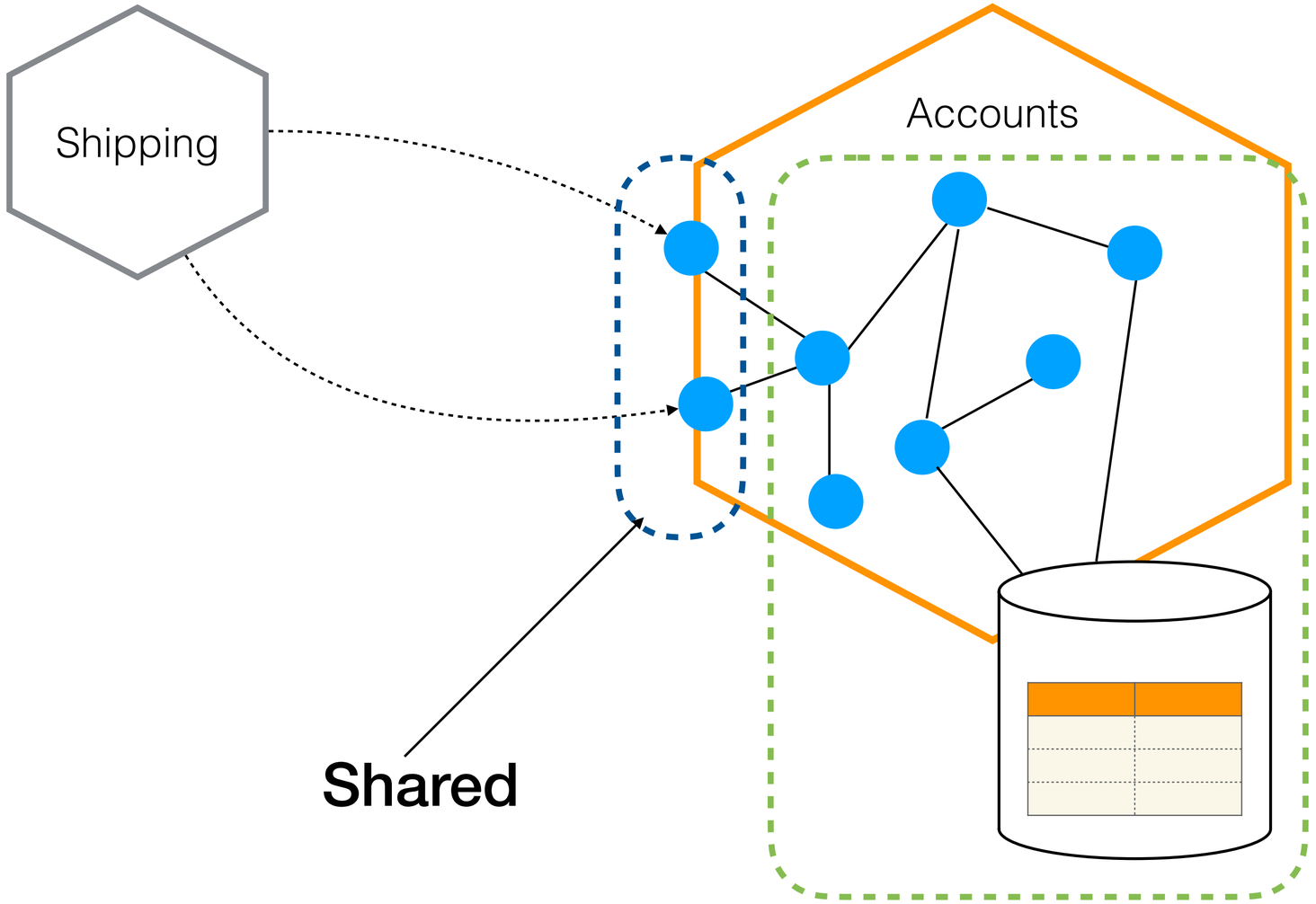
HIDING!

Hide your secrets!

Be explicit about what is shared, and what is hidden



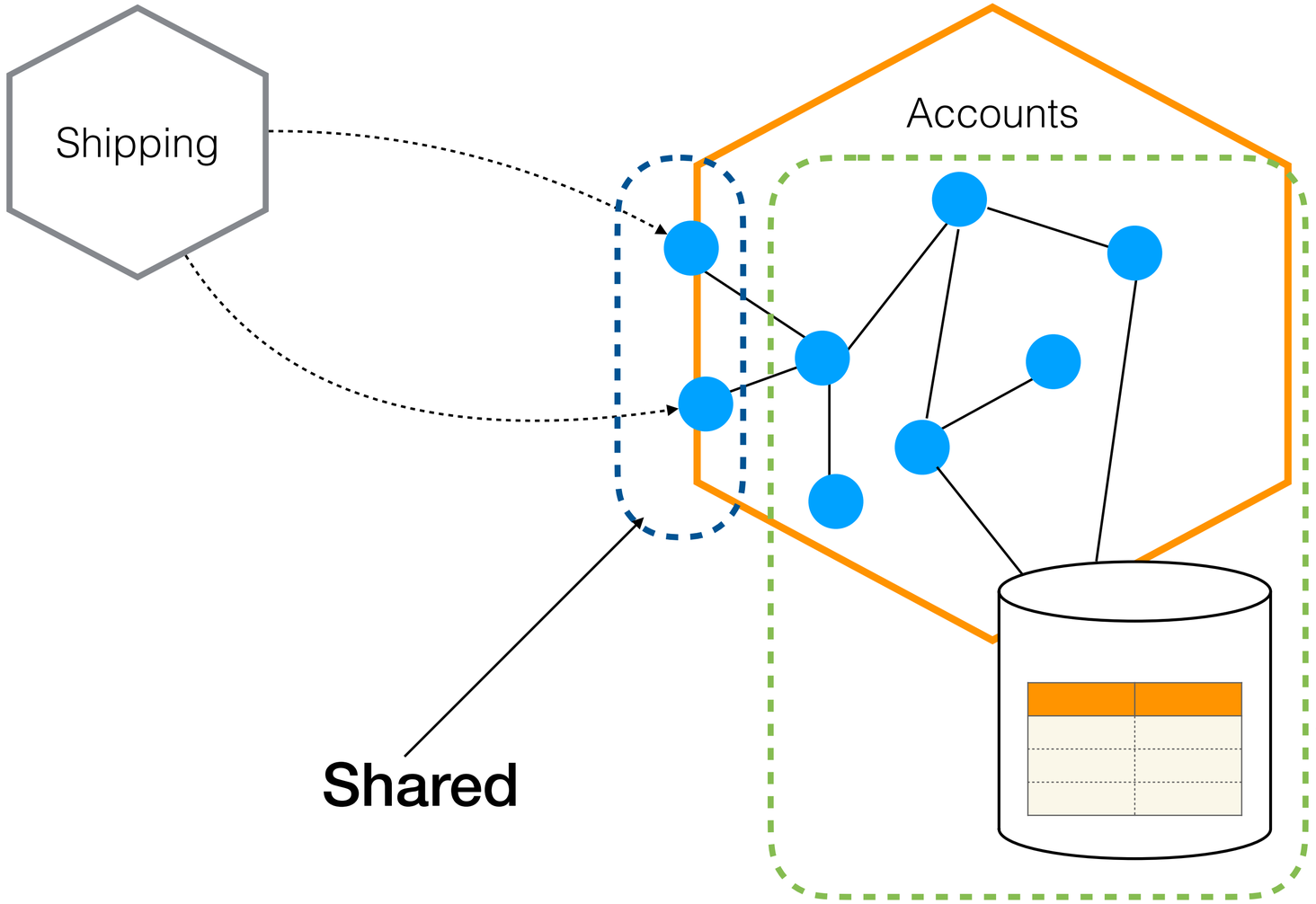
HIDING!



Hide your secrets!

Be explicit about what is shared, and what is hidden

HIDING!

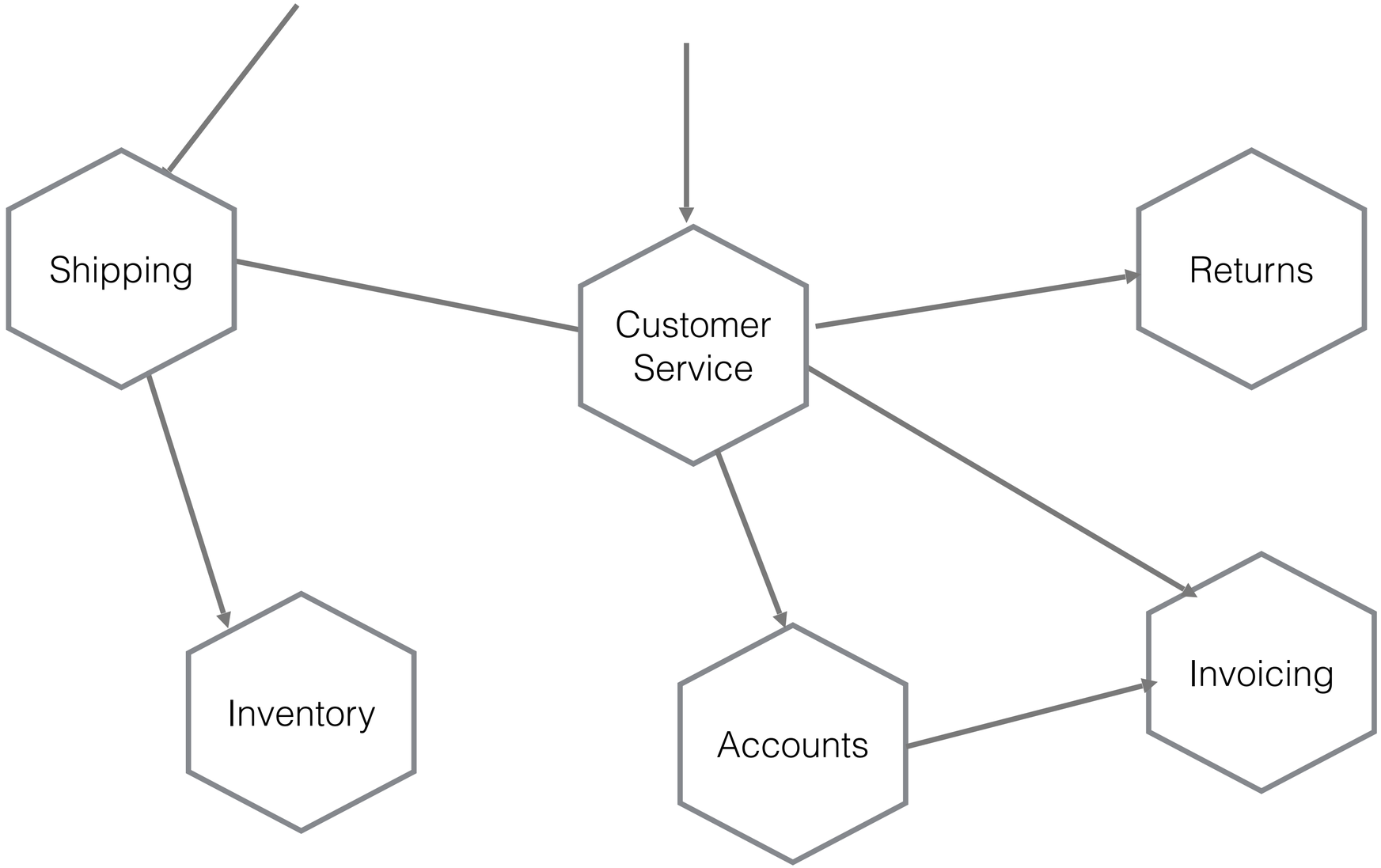


Hide your secrets!

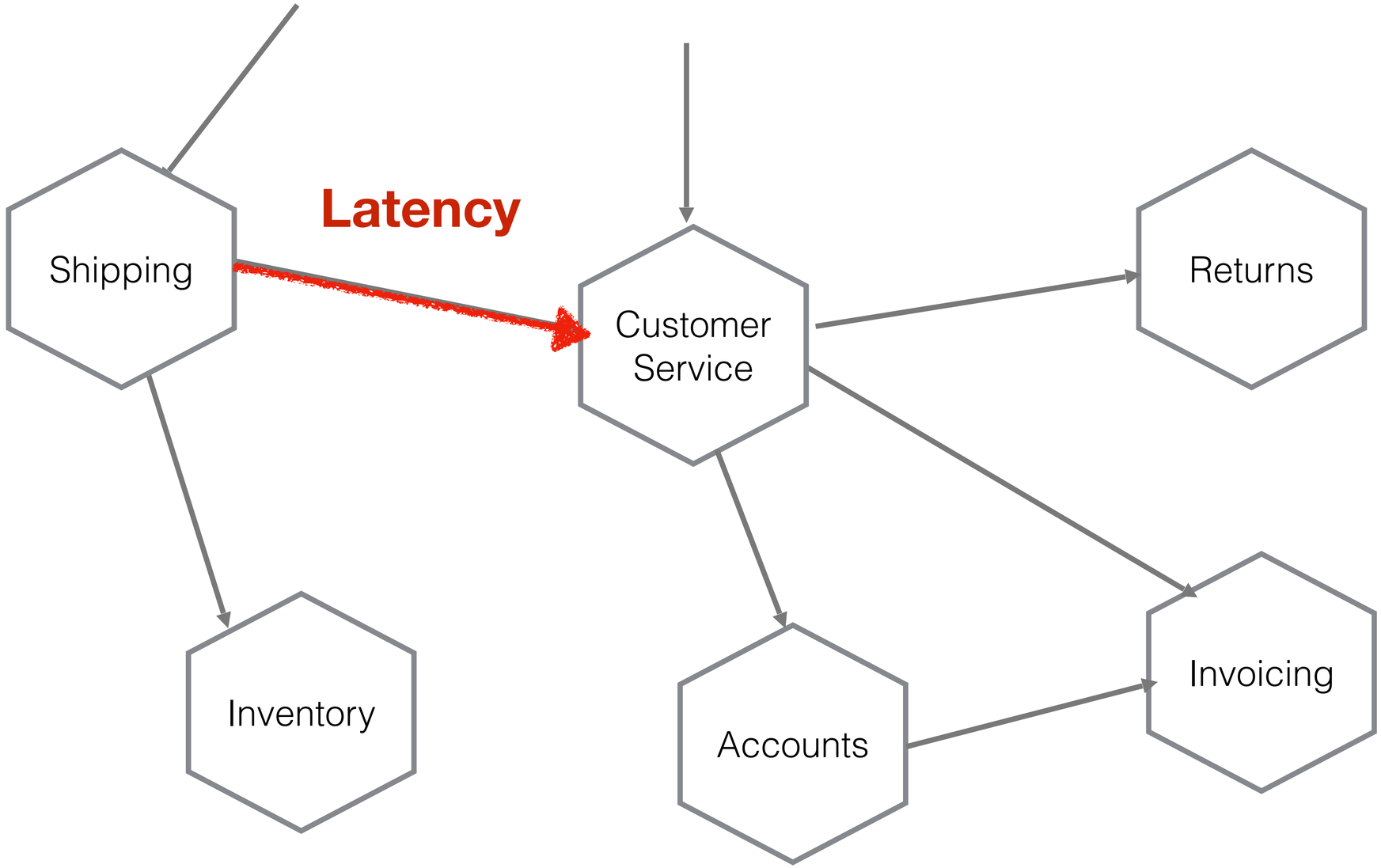
Be explicit about what is shared, and what is hidden

Hidden things can change, shared things can’t

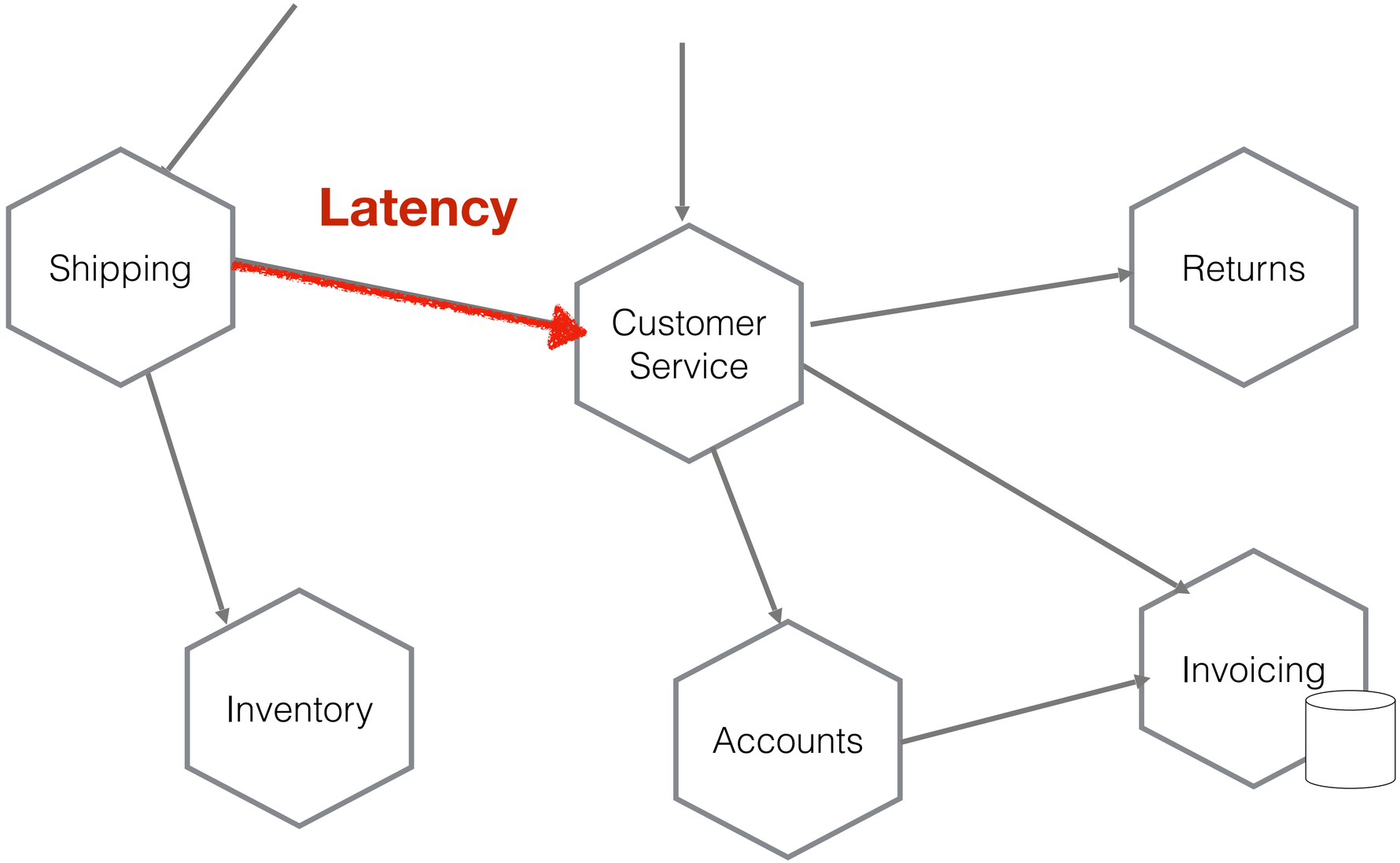
PROBLEMS WITH MICROSERVICES



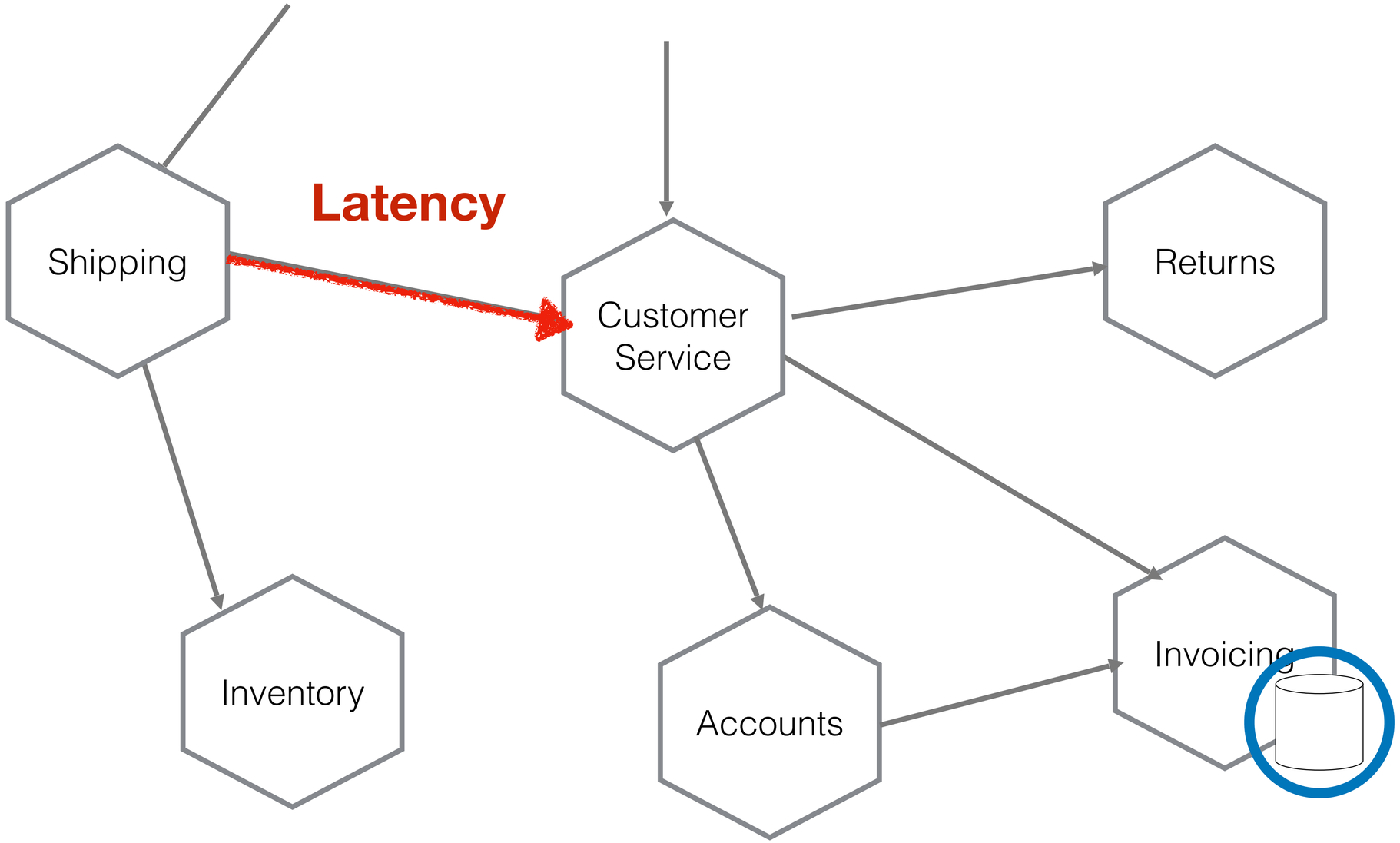
PROBLEMS WITH MICROSERVICES



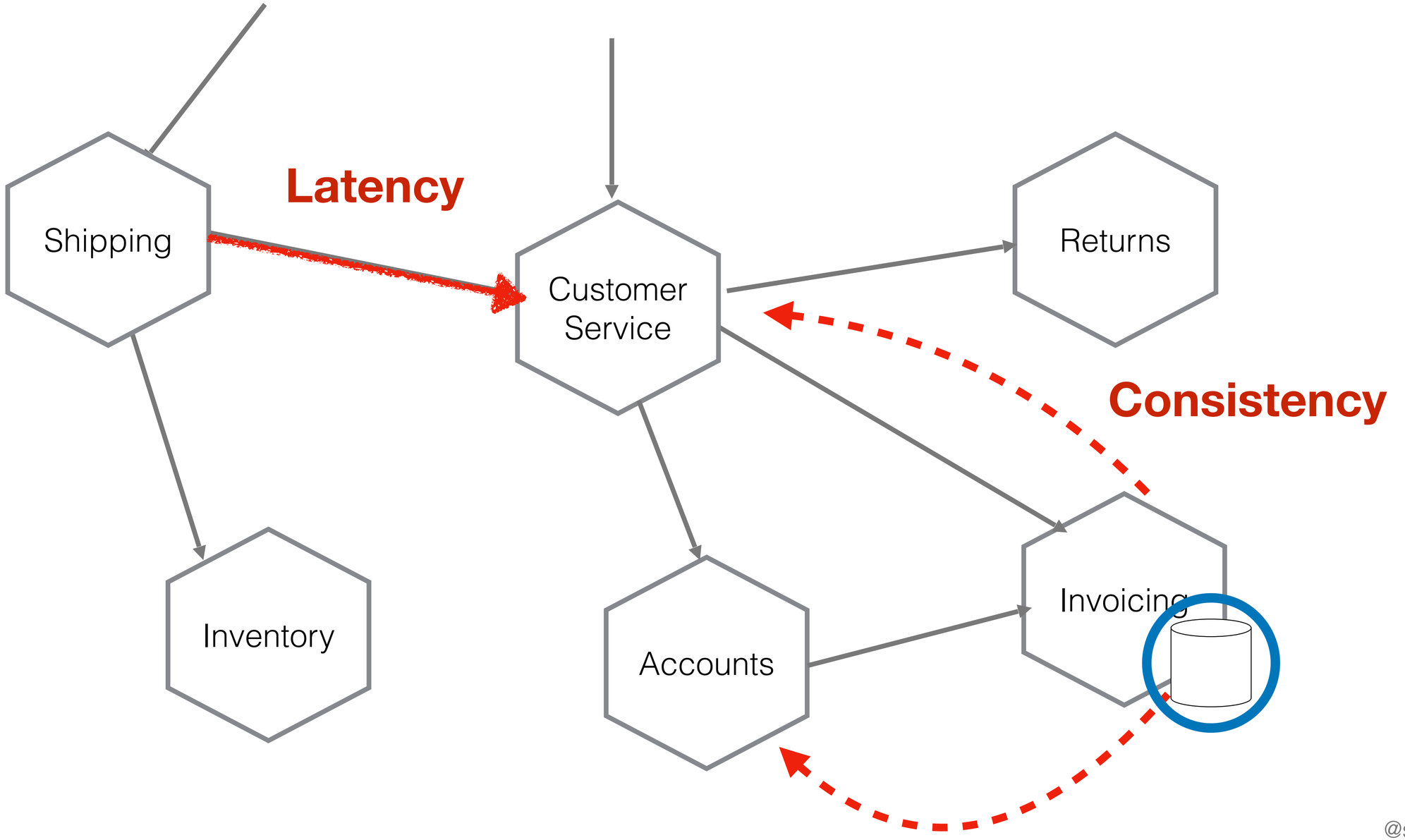
PROBLEMS WITH MICROSERVICES



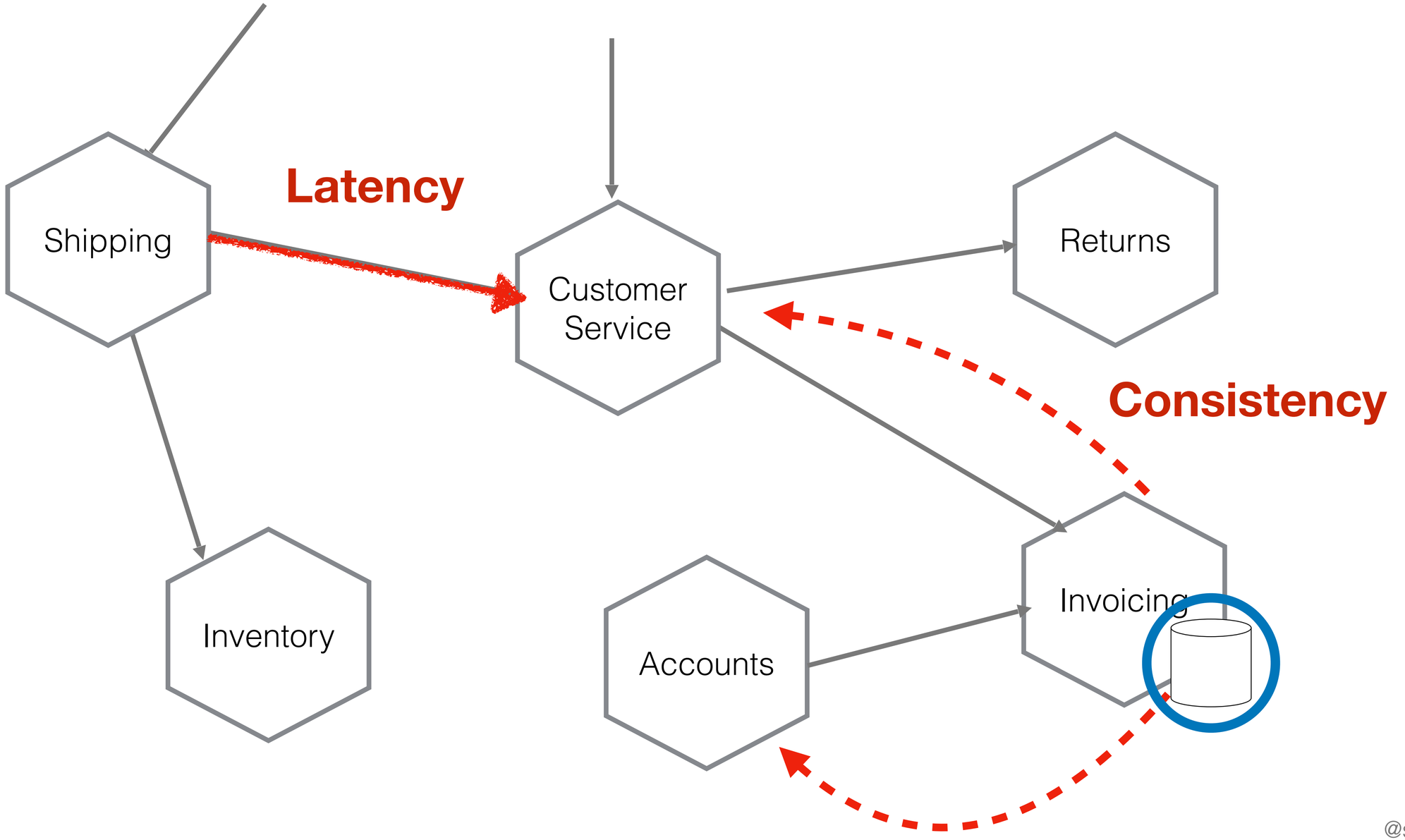
PROBLEMS WITH MICROSERVICES



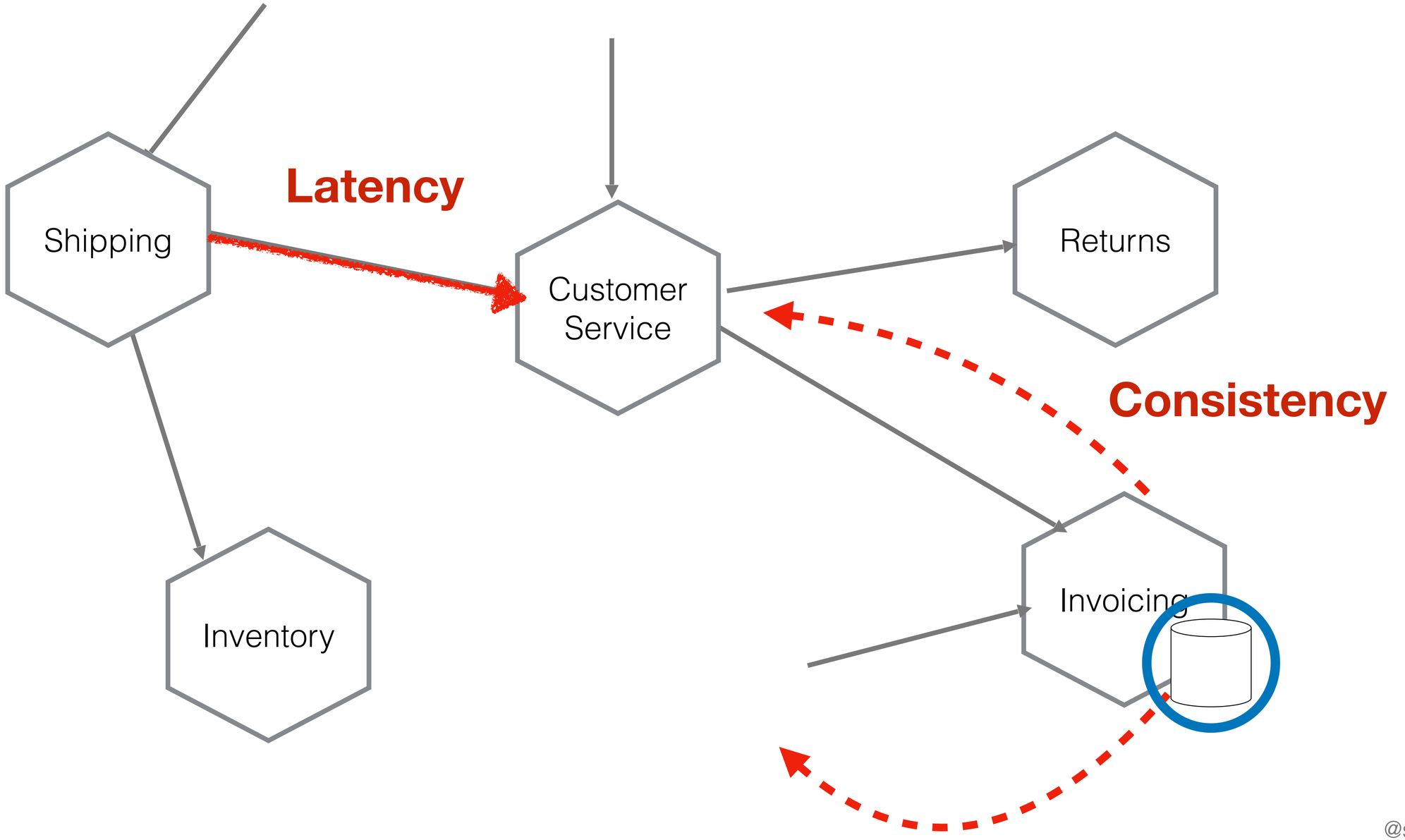
PROBLEMS WITH MICROSERVICES



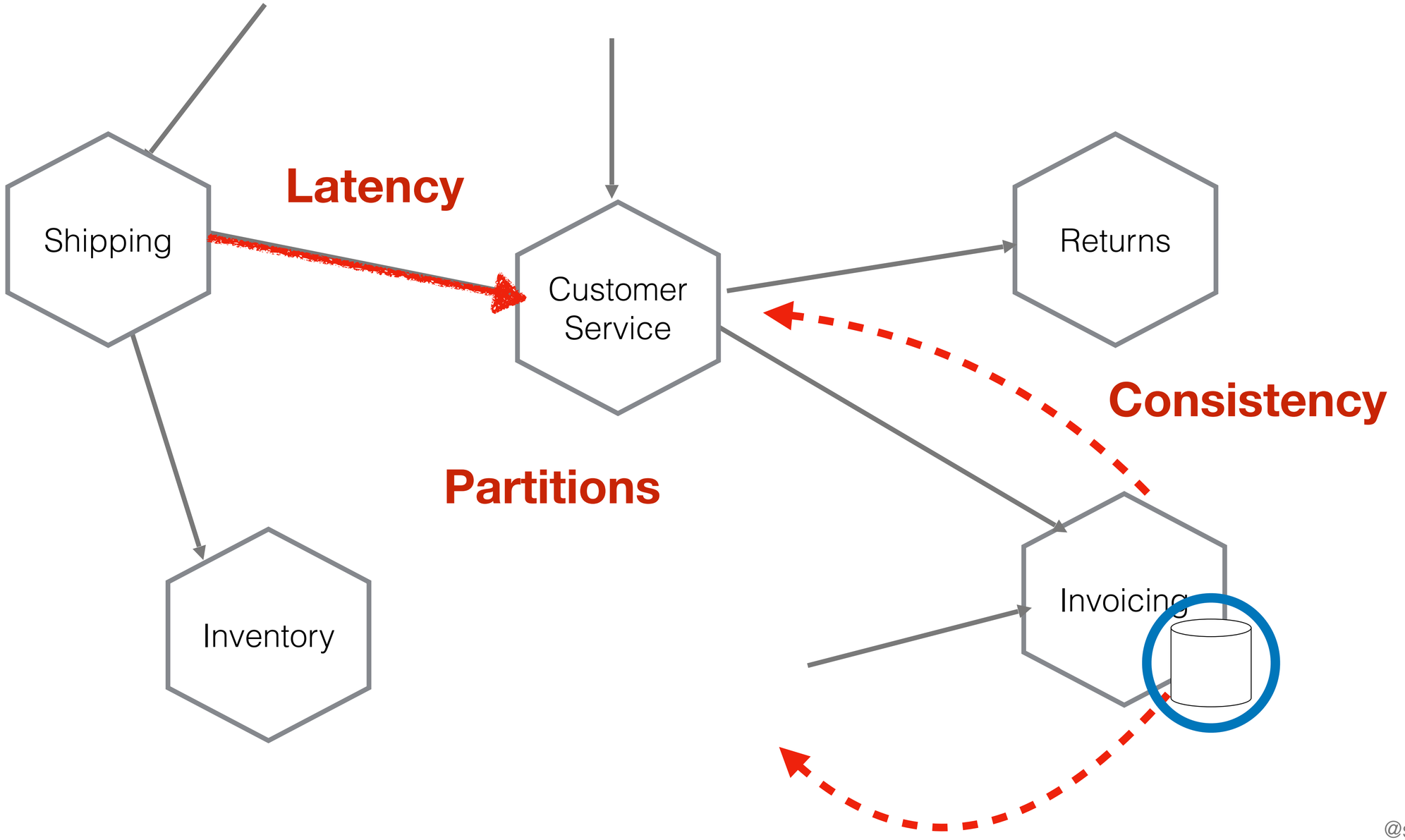
PROBLEMS WITH MICROSERVICES



PROBLEMS WITH MICROSERVICES



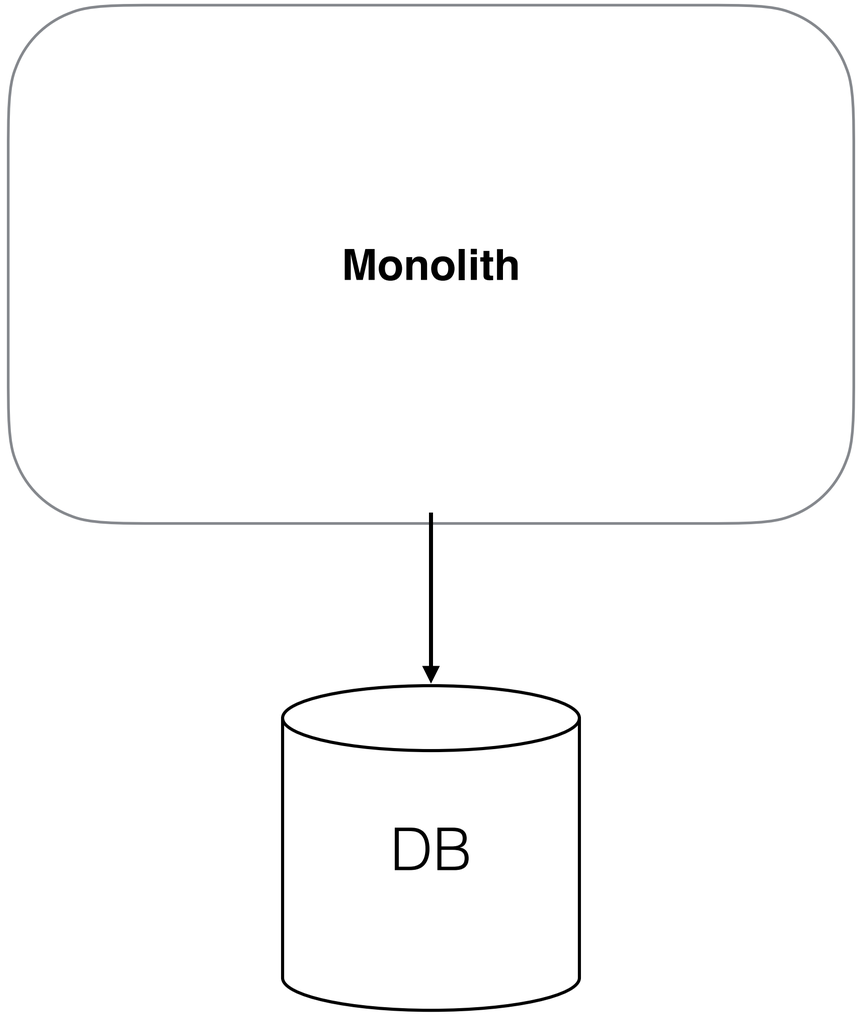
PROBLEMS WITH MICROSERVICES



https://www.flickr.com/photos/jon-wade/4769234322/

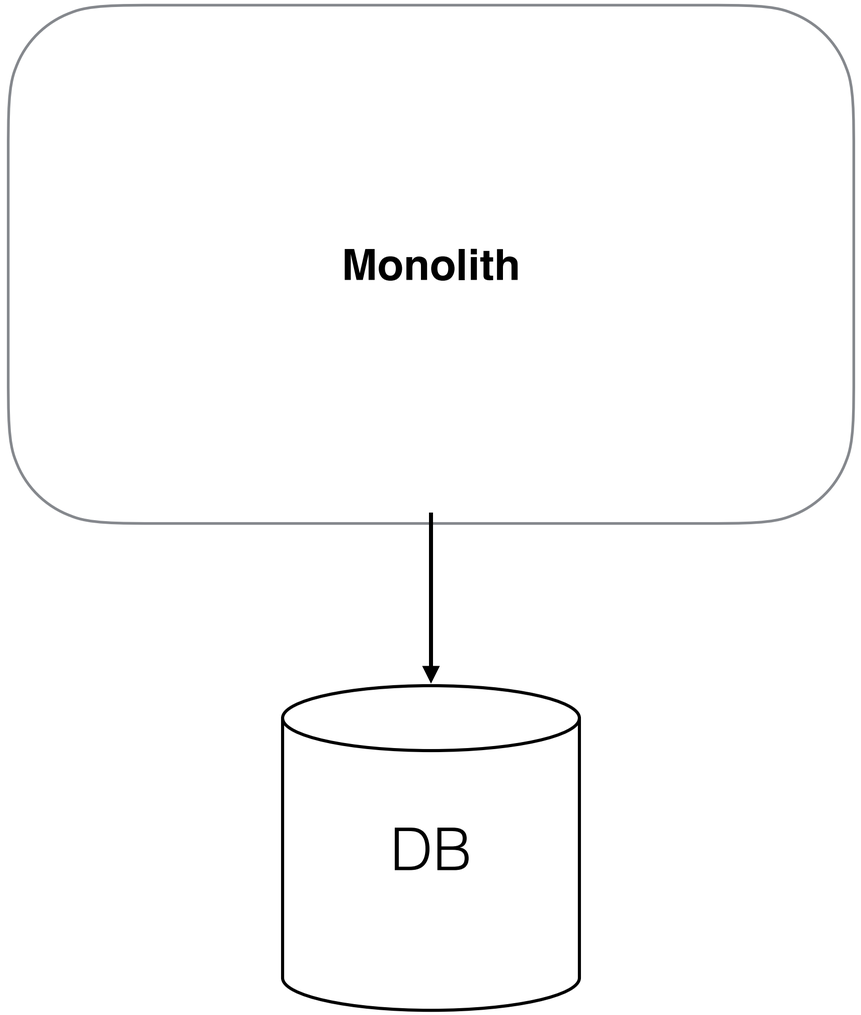
Monolith refers to a unit of deployment

SINGLE PROCESS MONOLITH



SINGLE PROCESS MONOLITH

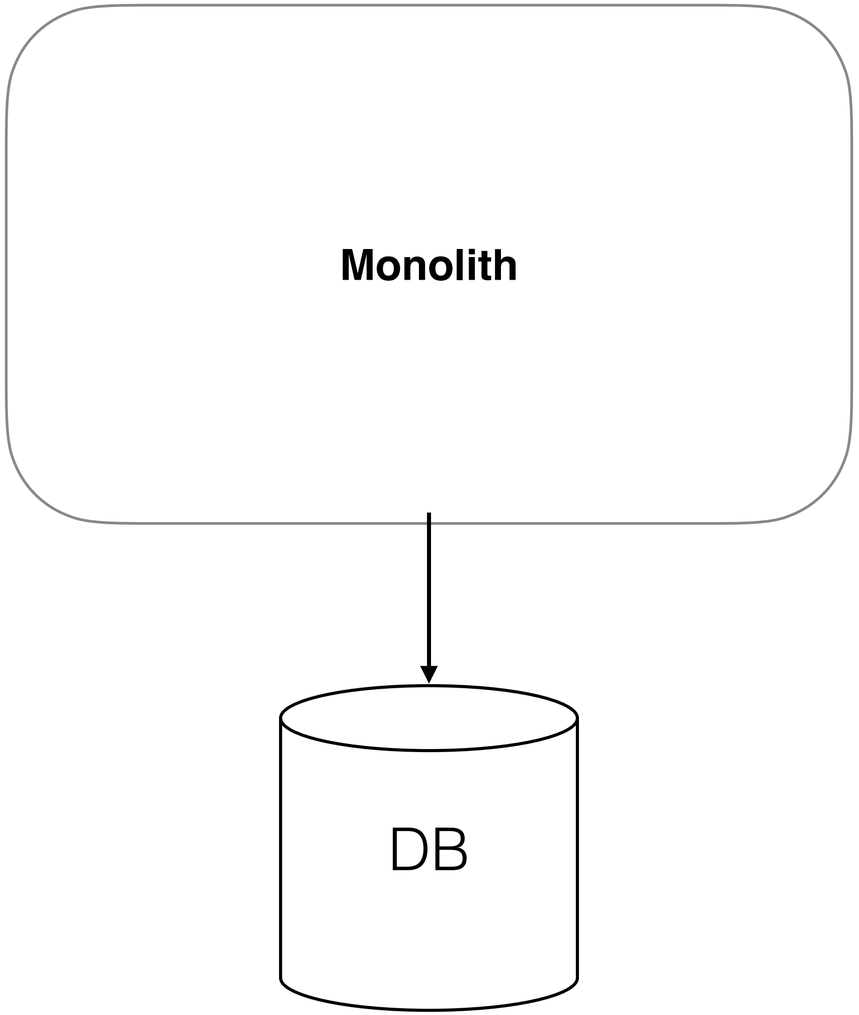
All code packaged into a single process



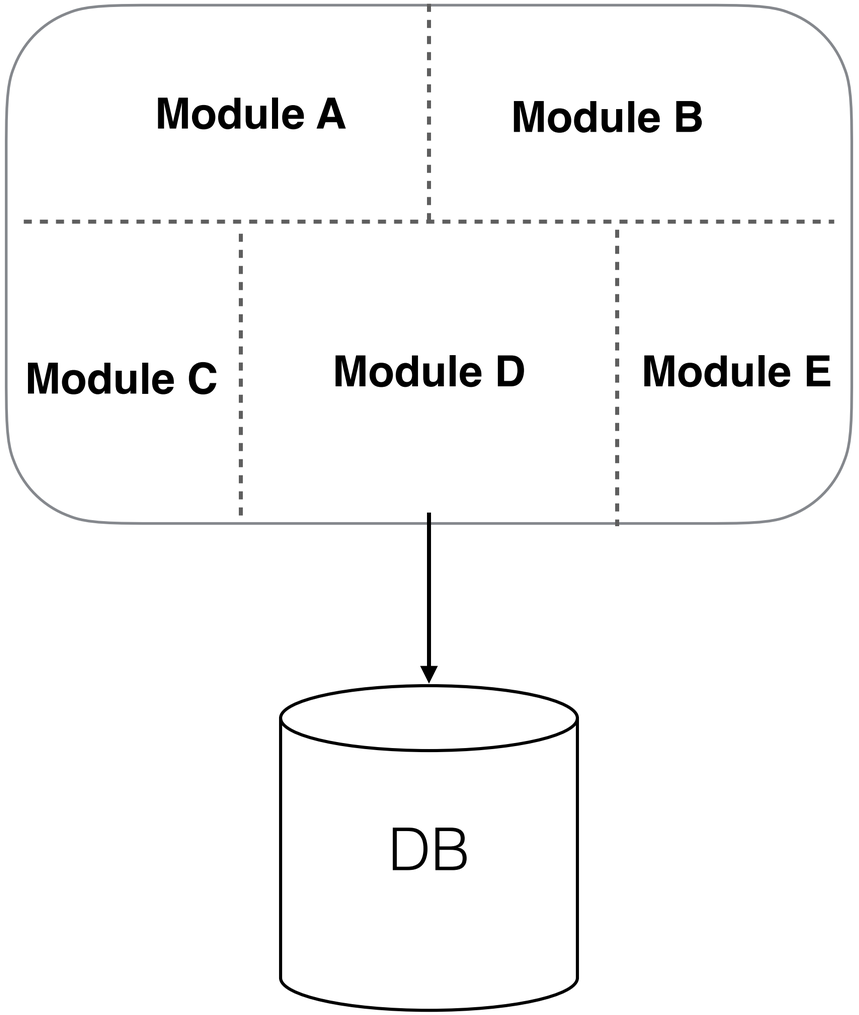
SINGLE PROCESS MONOLITH

All code packaged into a single process

All data stored in a single database

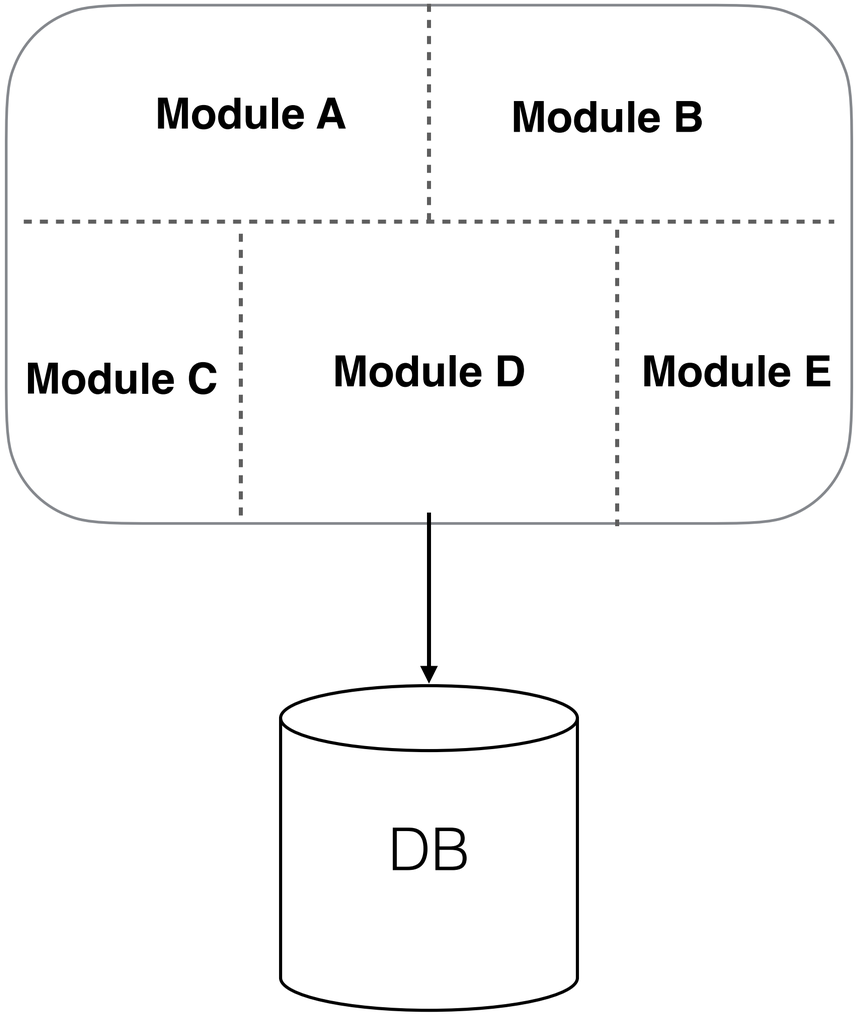


MODULAR MONOLITH



MODULAR MONOLITH

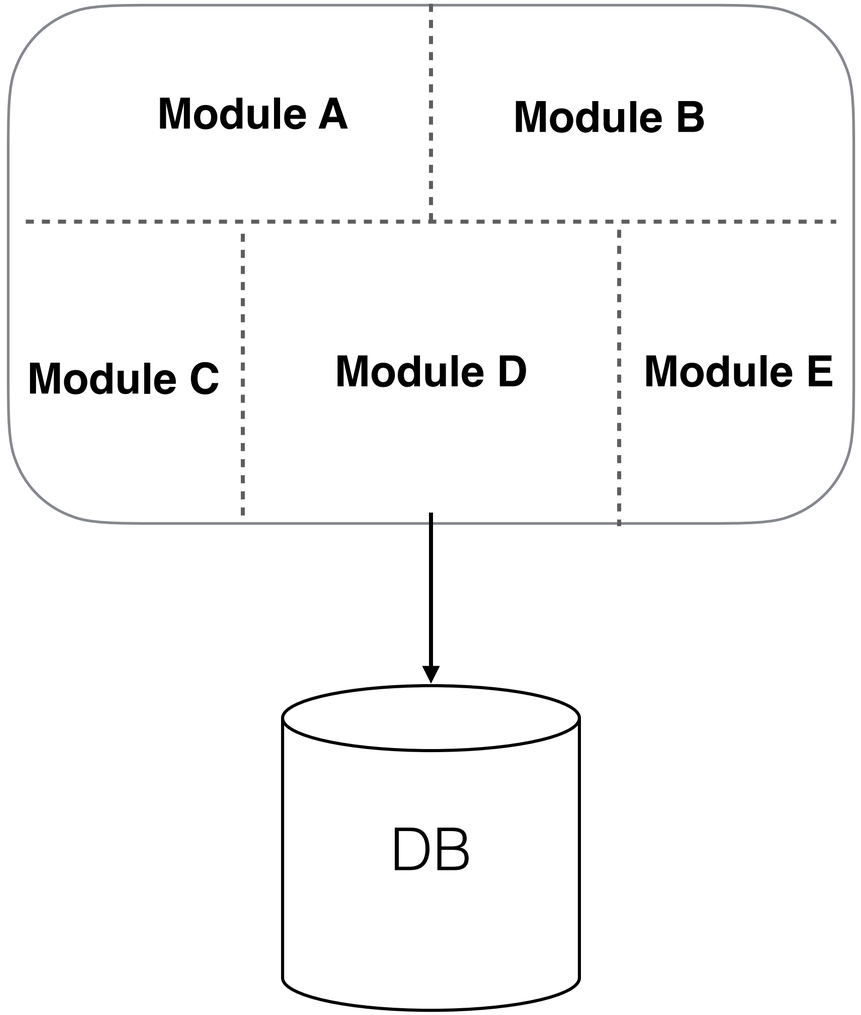
The code is broken into modules



MODULAR MONOLITH

The code is broken into modules

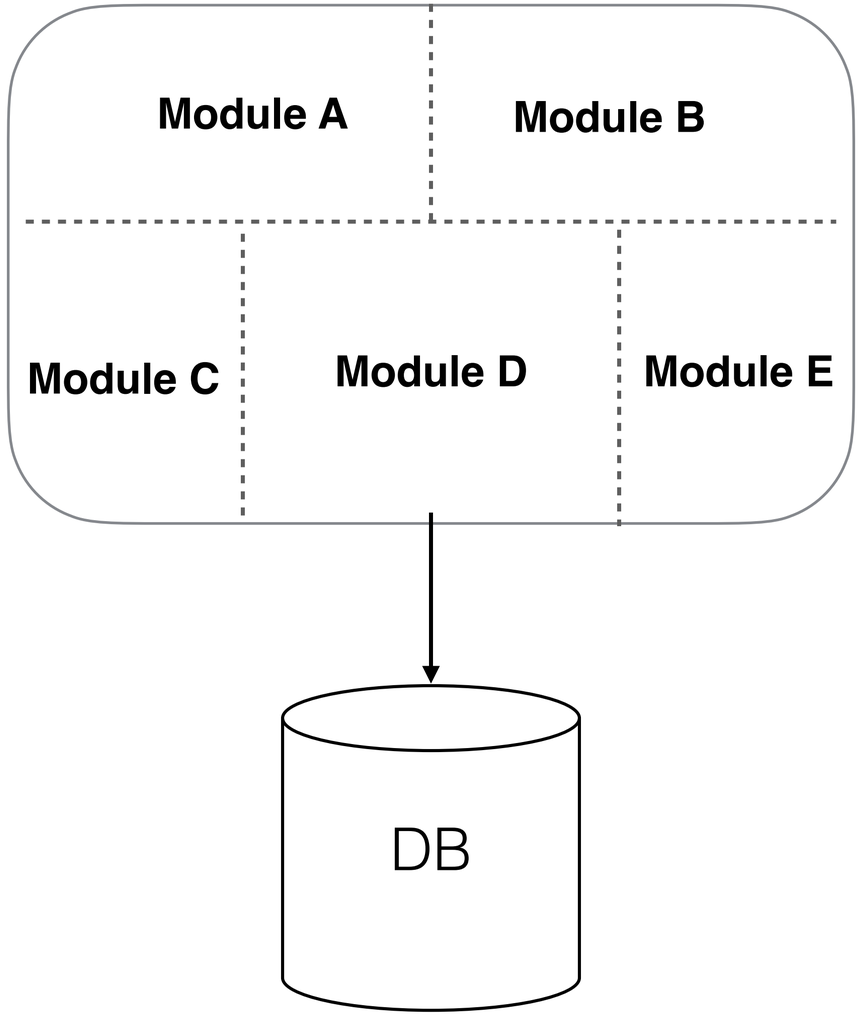
Each module packaged together into a single process



MODULAR MONOLITH

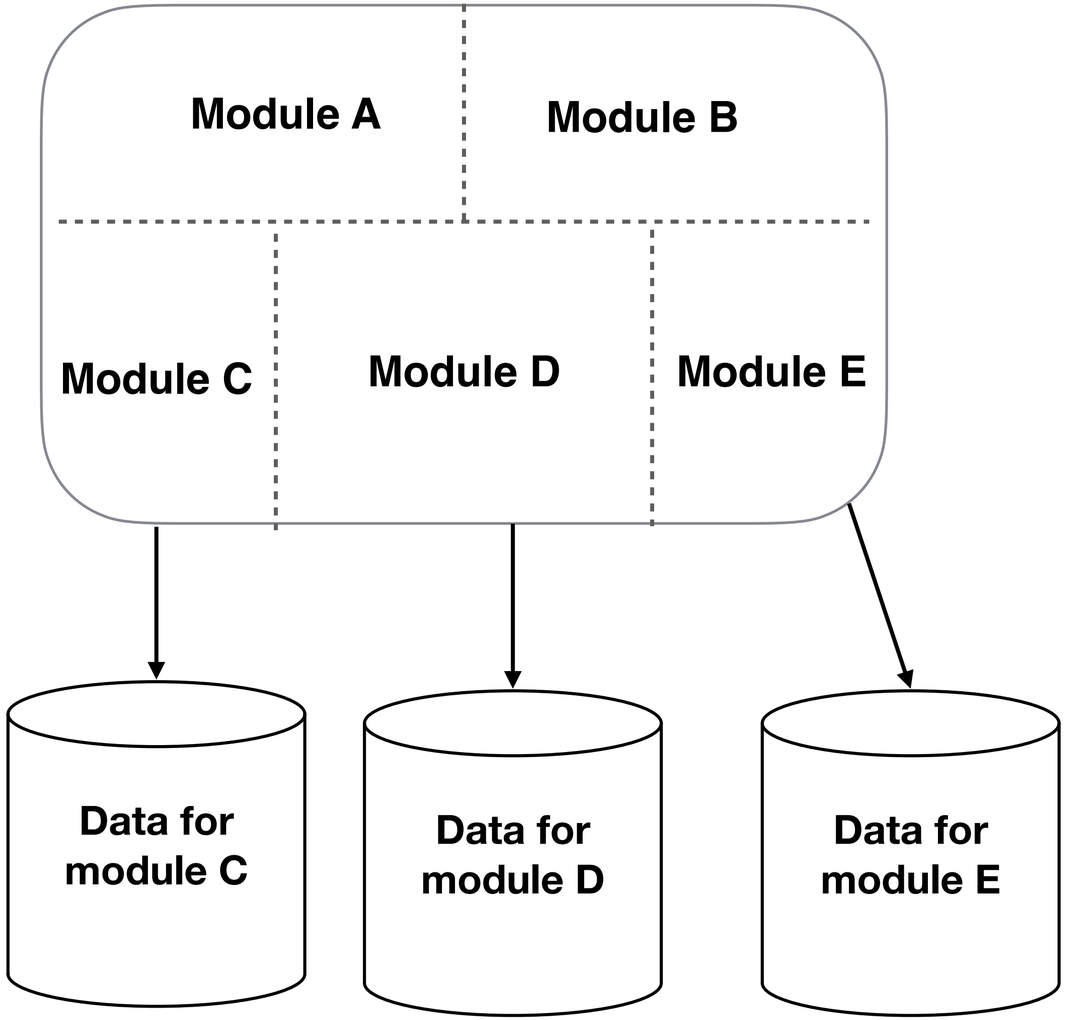
The code is broken into modules

Each module packaged together into a single process

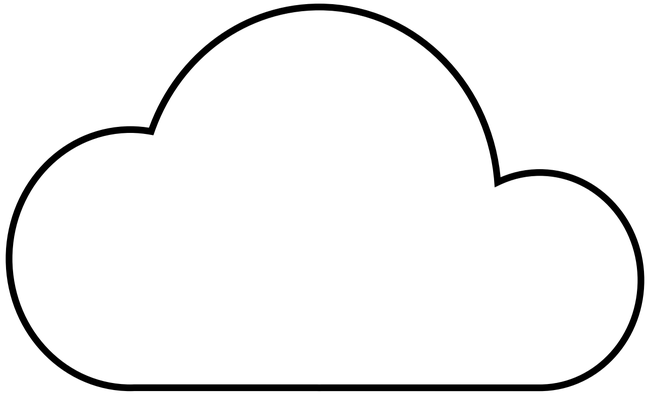


Highly underrated option

MODULAR DATA

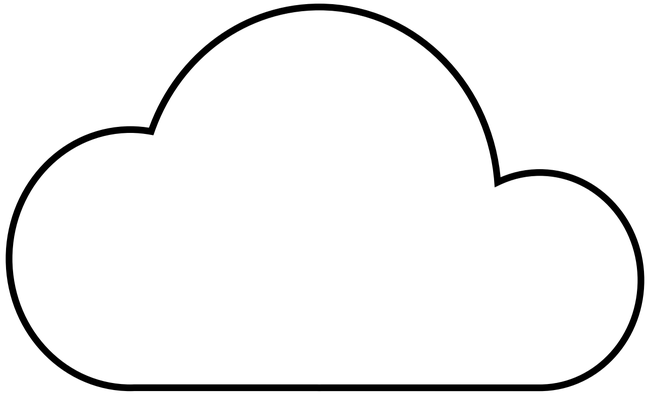


COTS/THIRD-PARTY MONOLITH



COTS/THIRD-PARTY MONOLITH

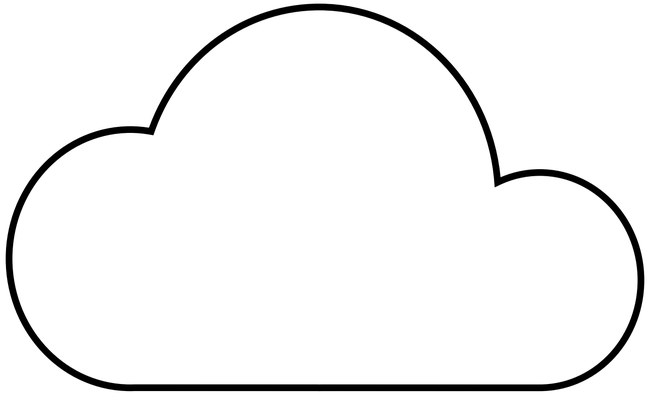
Could be on-prem software, or a SAAS product



COTS/THIRD-PARTY MONOLITH

Could be on-prem software, or a SAAS product

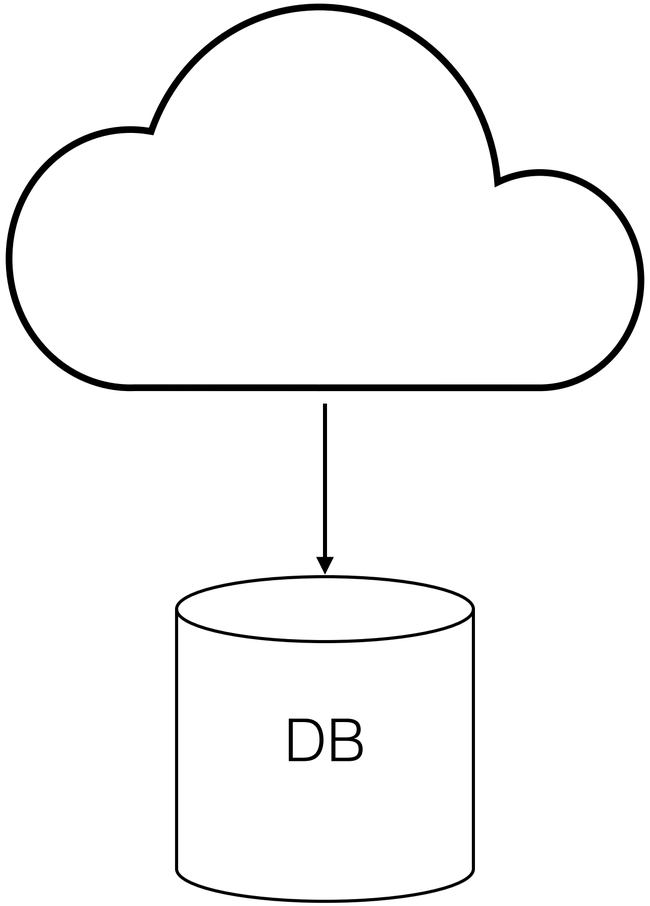
You have limited to no ability to change the core system



COTS/THIRD-PARTY MONOLITH

Could be on-prem software, or a SAAS product

You have limited to no ability to change the core system

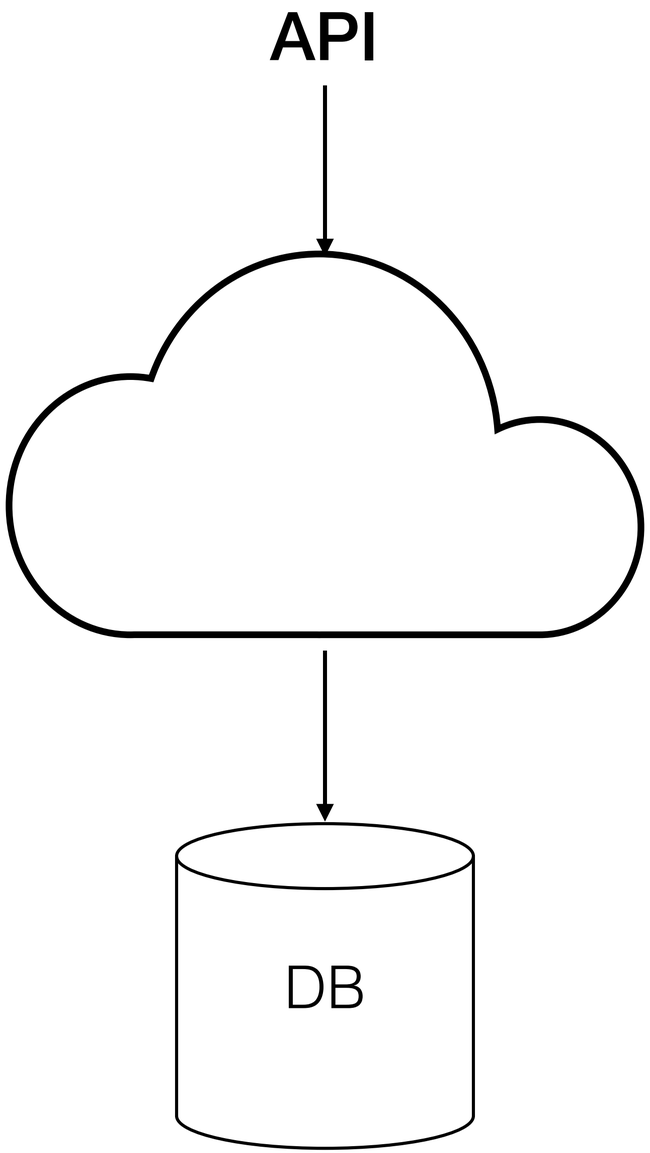


You *might* have access to underlying storage…

COTS/THIRD-PARTY MONOLITH

Could be on-prem software, or a SAAS product

You have limited to no ability to change the core system



You *might* have access to underlying storage…

…or perhaps APIs

AGENDA

Introduction

Migration Approach

Application Refactoring

UI Decomposition

AGENDA

Introduction

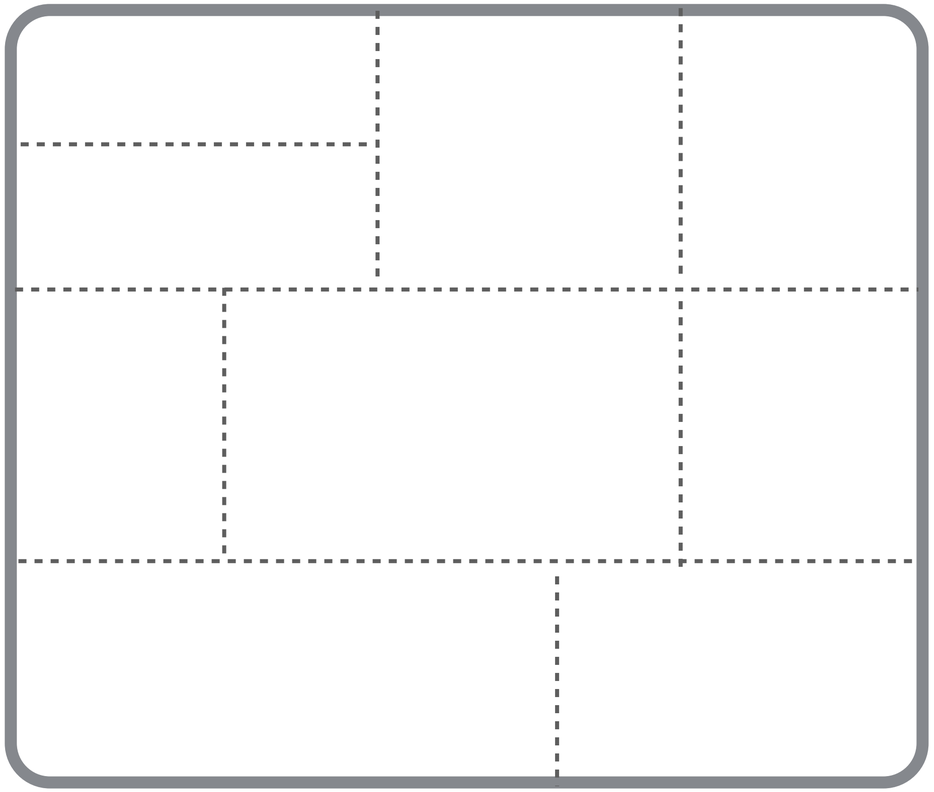
Migration Approach

Application Refactoring

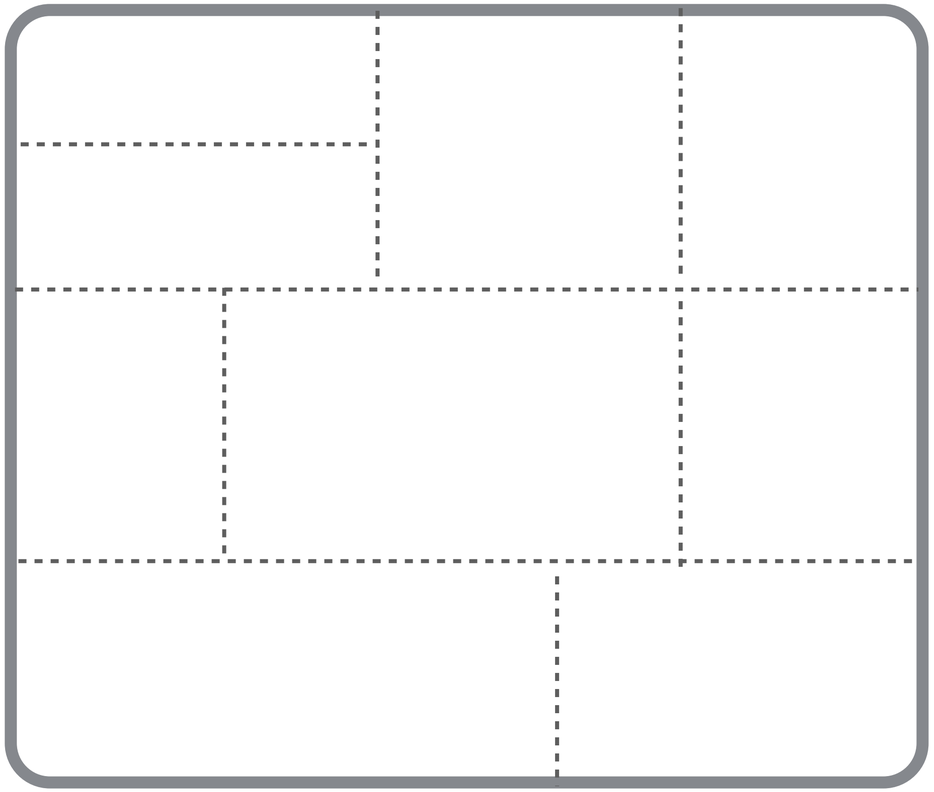
UI Decomposition

DECOMPOSITIONAL DRIVERS

DECOMPOSITIONAL DRIVERS

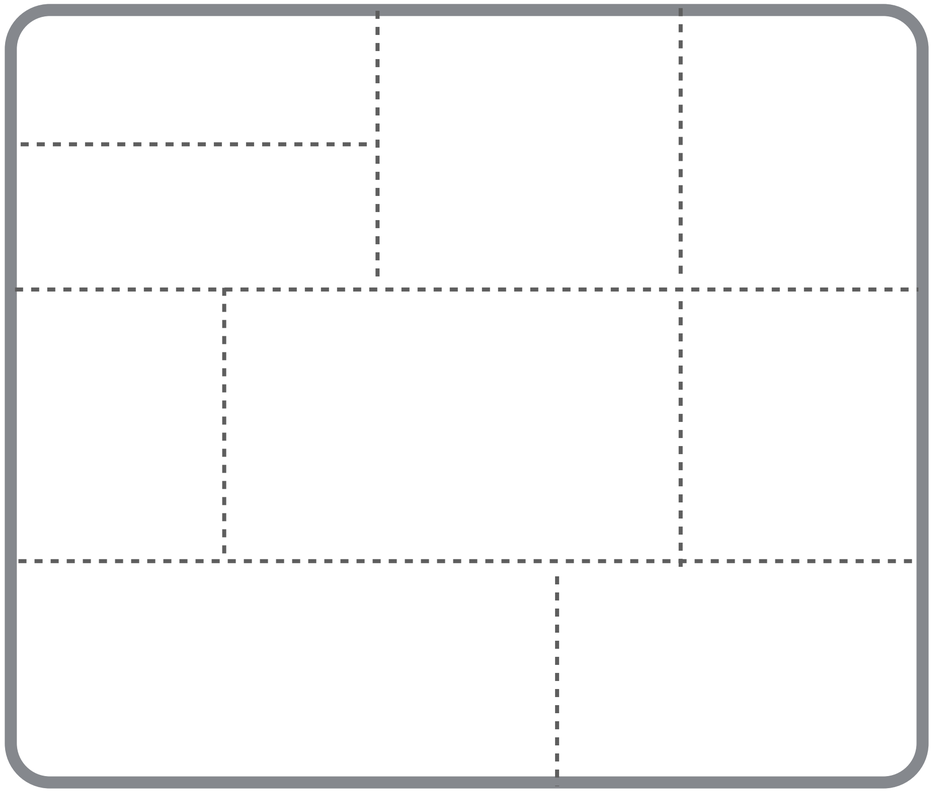


DECOMPOSITIONAL DRIVERS



How do you decide how to break apart your monolith?

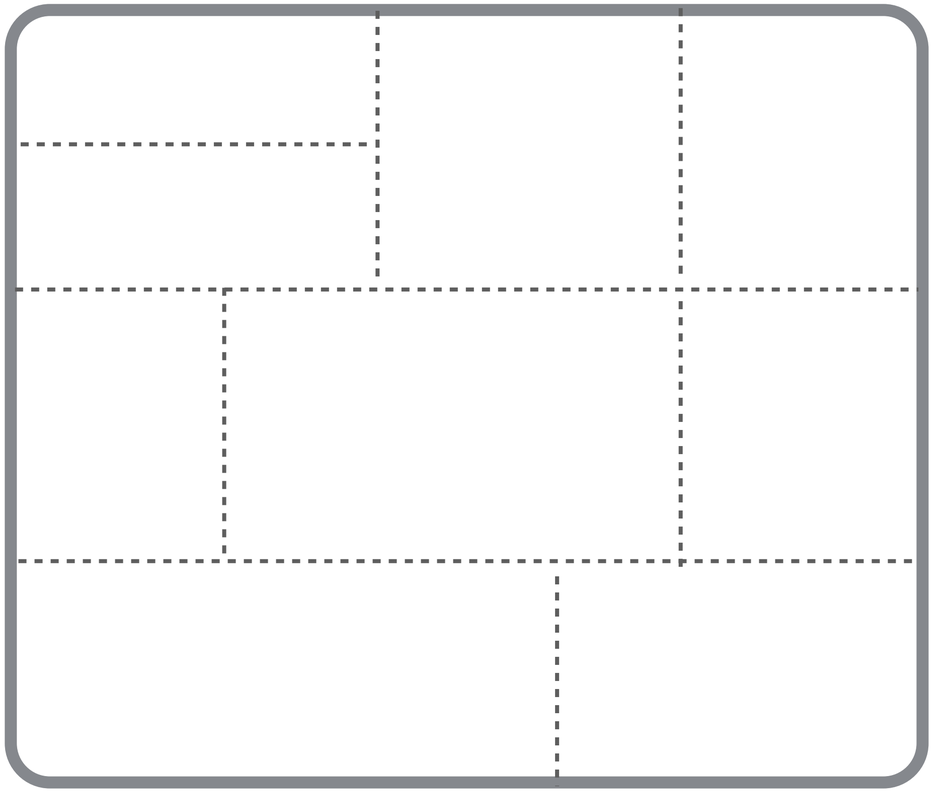
DECOMPOSITIONAL DRIVERS



How do you decide how to break apart your monolith?

Lots of different techniques…

DECOMPOSITIONAL DRIVERS



How do you decide how to break apart your monolith?

Lots of different techniques…

…but the goal is key

IT’S ALL ABOUT THE GOAL

IT'S ALL ABOUT THE GOAL

You don't "win" by having microservices

IT'S ALL ABOUT THE GOAL

You don't "win" by having microservices

Microservices are a means to an end

POLL: DO YOU HAVE A CLEAR UNDERSTANDING AS TO WHY YOU USING MICROSERVICES?

1. Yes! I know why we're using microservices

2. I've got a big of a vague idea…

3. I have no idea at all

EXAMPLE GOALS & CONCERNS

EXAMPLE GOALS & CONCERNS

Time to market

EXAMPLE GOALS & CONCERNS

Time to market

Team autonomy

EXAMPLE GOALS & CONCERNS

Time to market

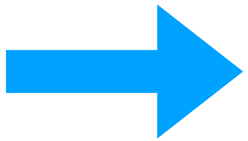
Team autonomy

Scale up size of delivery organisation

EXAMPLE GOALS & CONCERNS

Time to market

Team autonomy



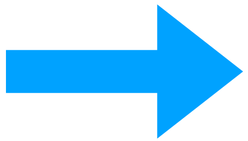
Scale up size of delivery organisation

EXAMPLE GOALS & CONCERNS

Time to market

Team autonomy

Scale up size of delivery organisation



Primary Concerns

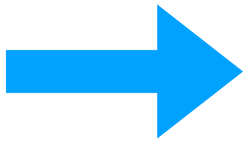
Domain-driven Design

EXAMPLE GOALS & CONCERNS

Time to market

Team autonomy

Scale up size of delivery organisation



Primary Concerns

Domain-driven Design

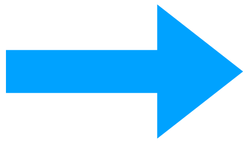
Organisational Structure

EXAMPLE GOALS & CONCERNS

Time to market

Team autonomy

Scale up size of delivery organisation



Primary Concerns

Domain-driven Design

Organisational Structure

Secondary Concern

Volatility

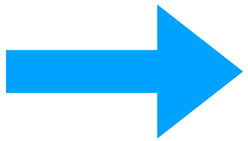
EXAMPLE GOALS & CONCERNS (CONT)

EXAMPLE GOALS & CONCERNS (CONT)

Application
Scale

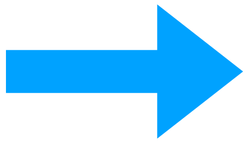
EXAMPLE GOALS & CONCERNS (CONT)

Application Scale



EXAMPLE GOALS & CONCERNS (CONT)

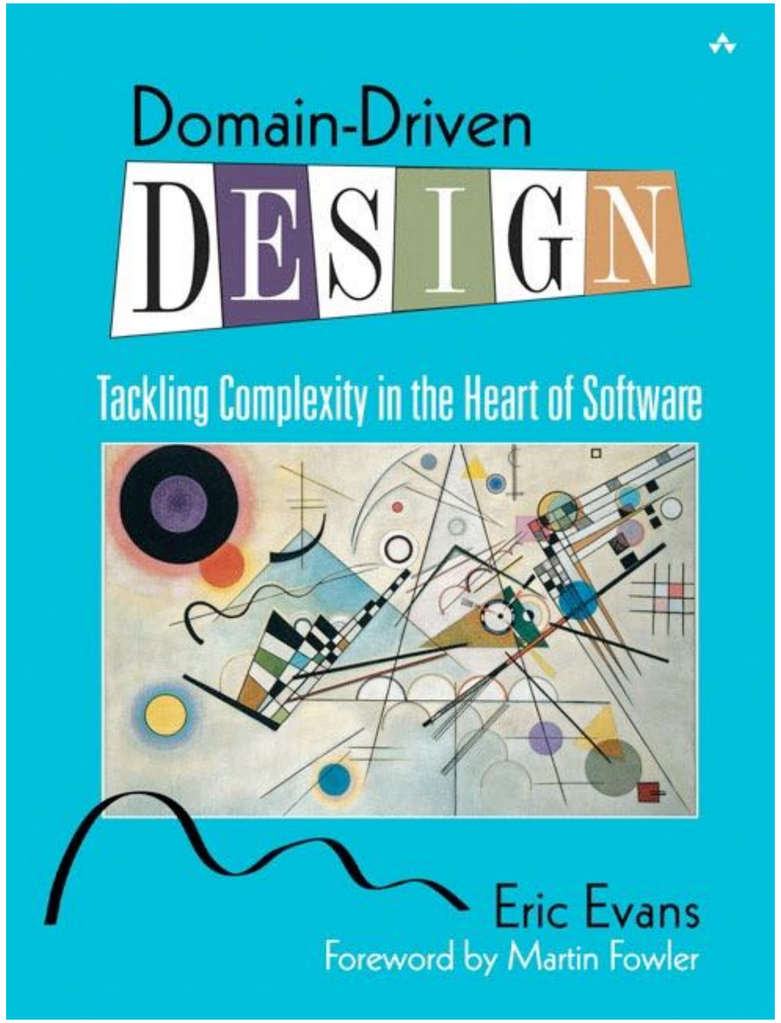
Application Scale



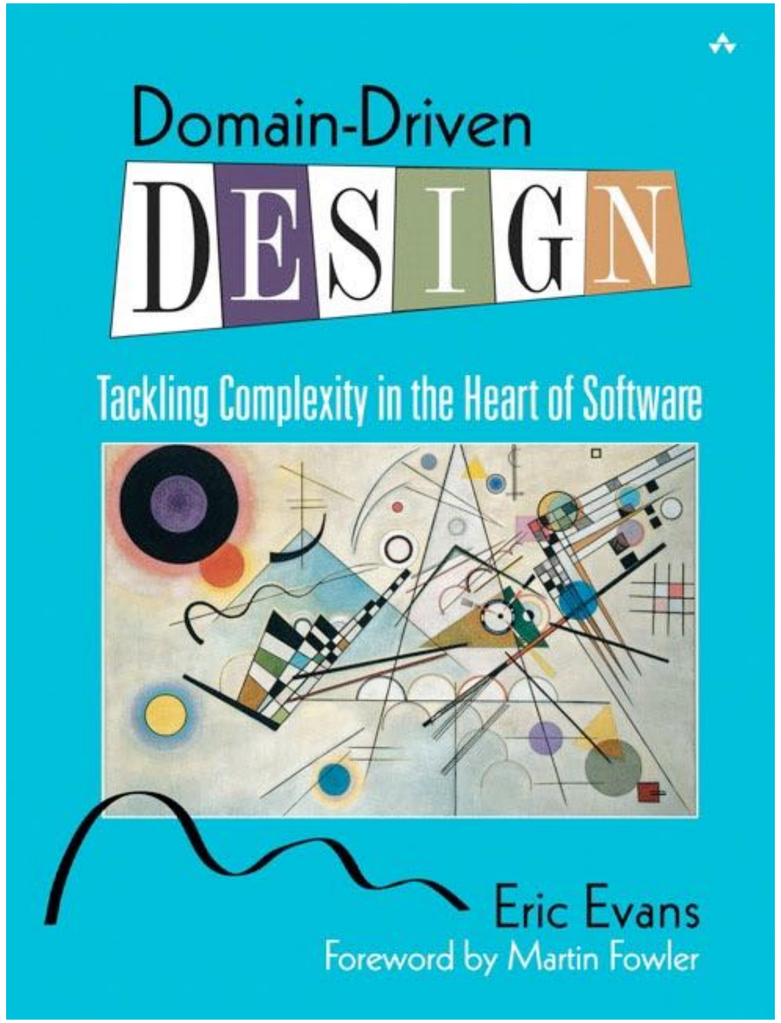
Primary Concern

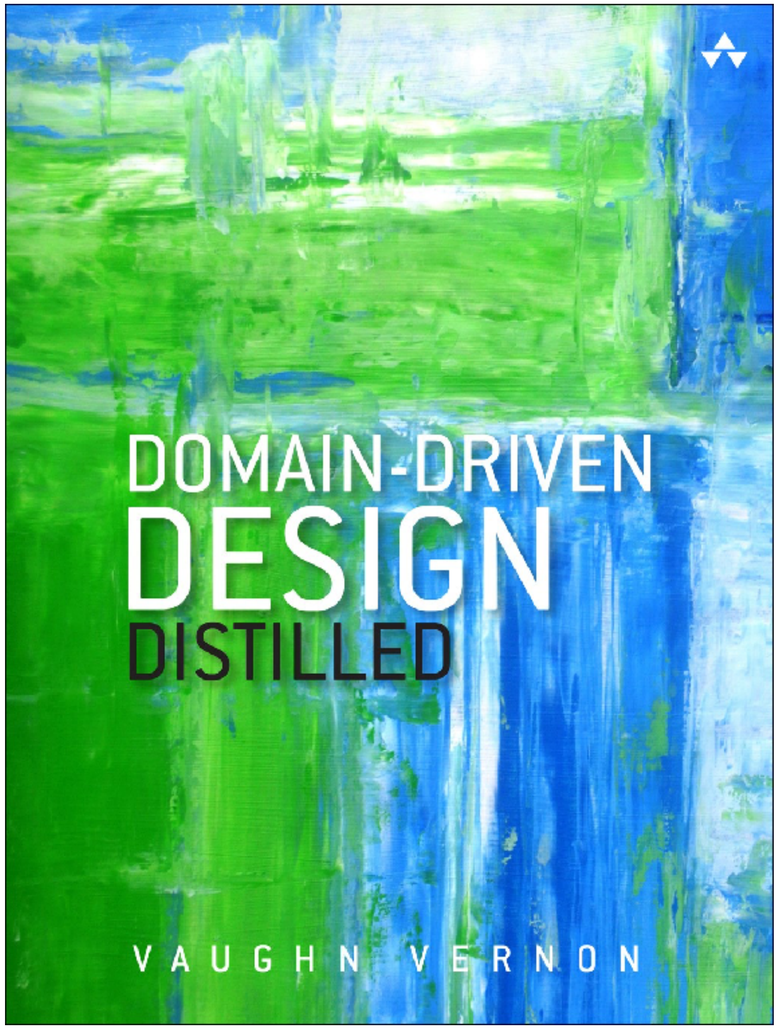
Bottlenecks

DOMAIN-DRIVEN DESIGN



DOMAIN-DRIVEN DESIGN





Poll

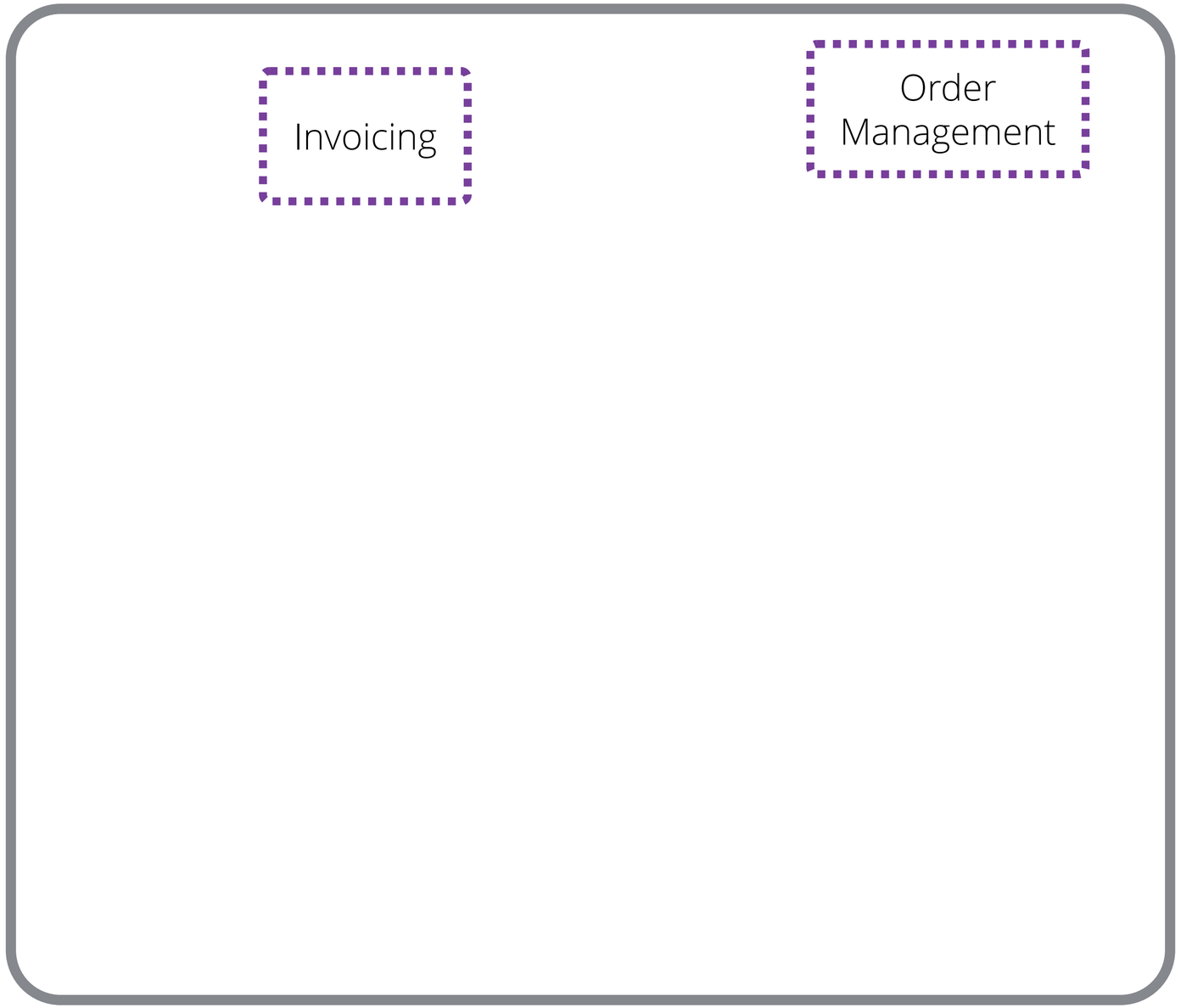
How many of you know about Domain-Driven Design?

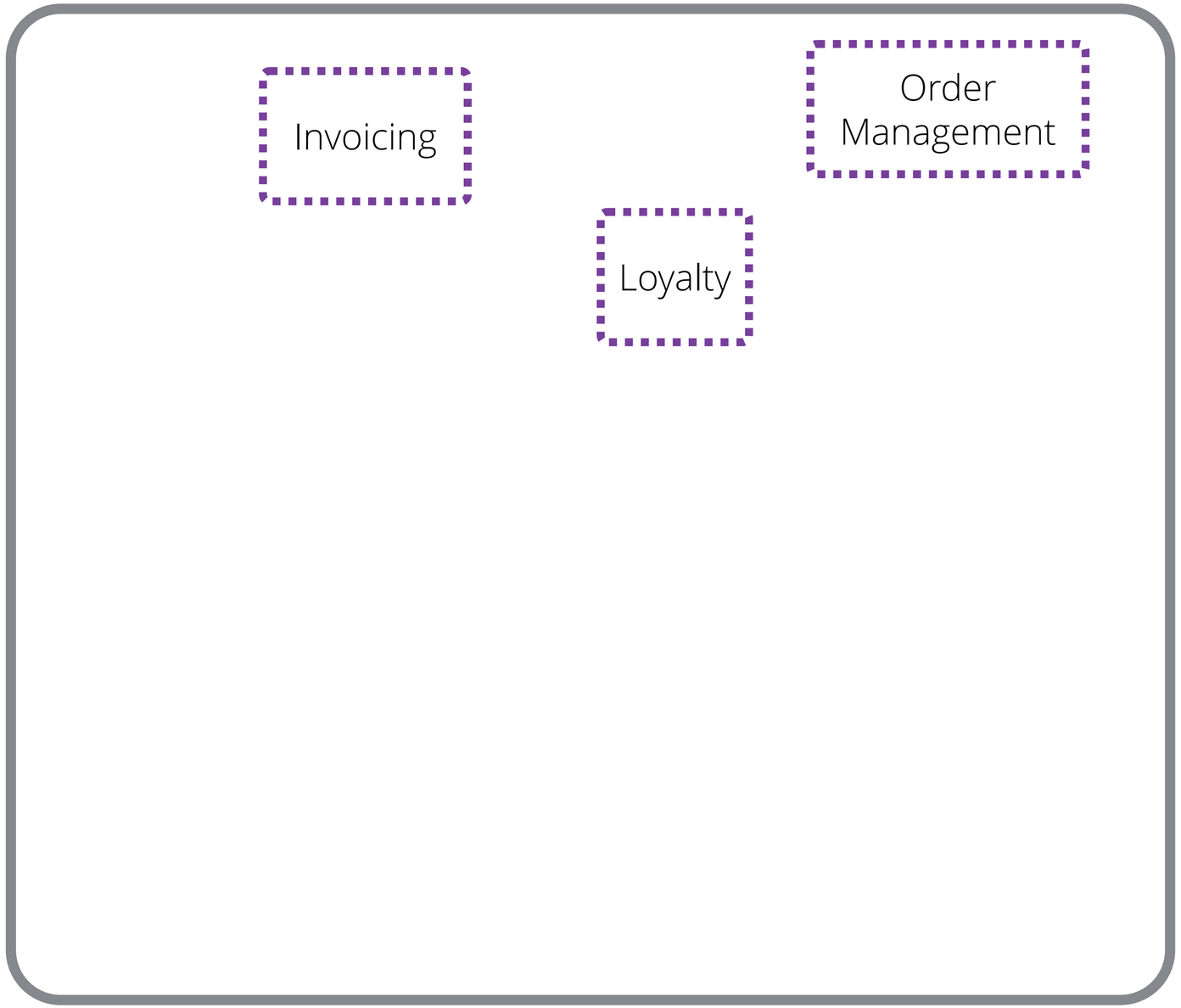
a.) I don’t know about it

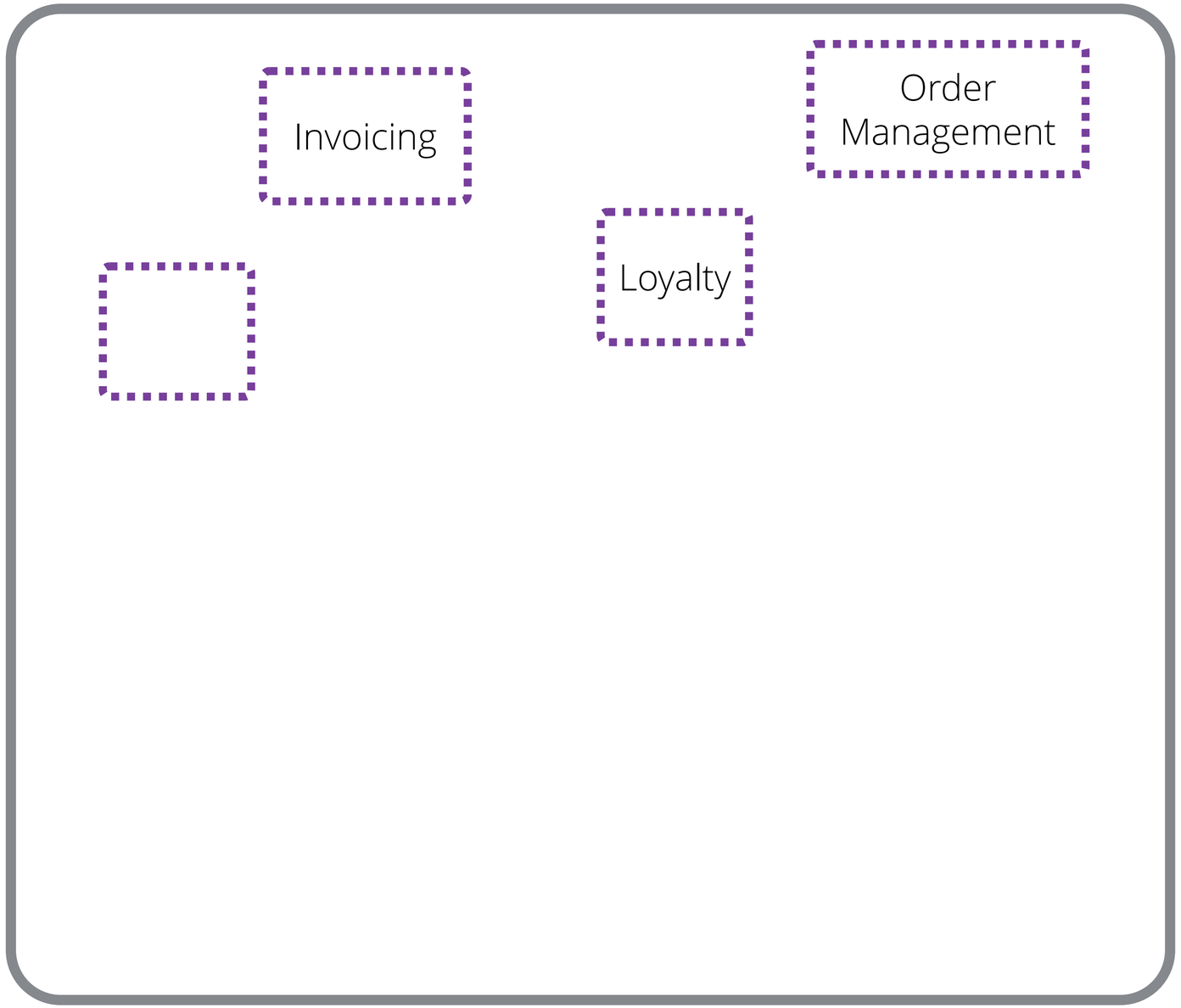
b.) I know a bit about it, but haven’t used it

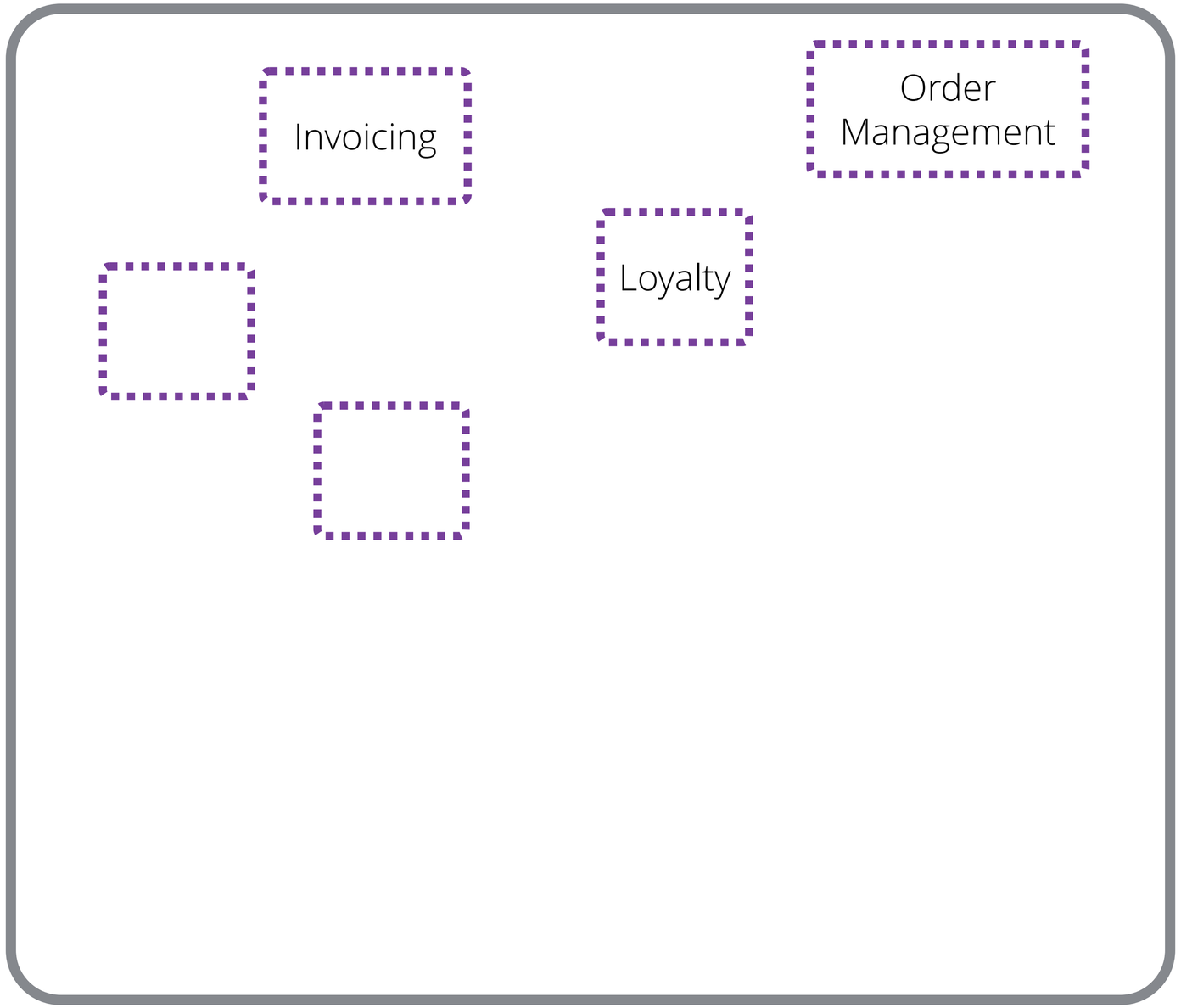
c.) We use it!

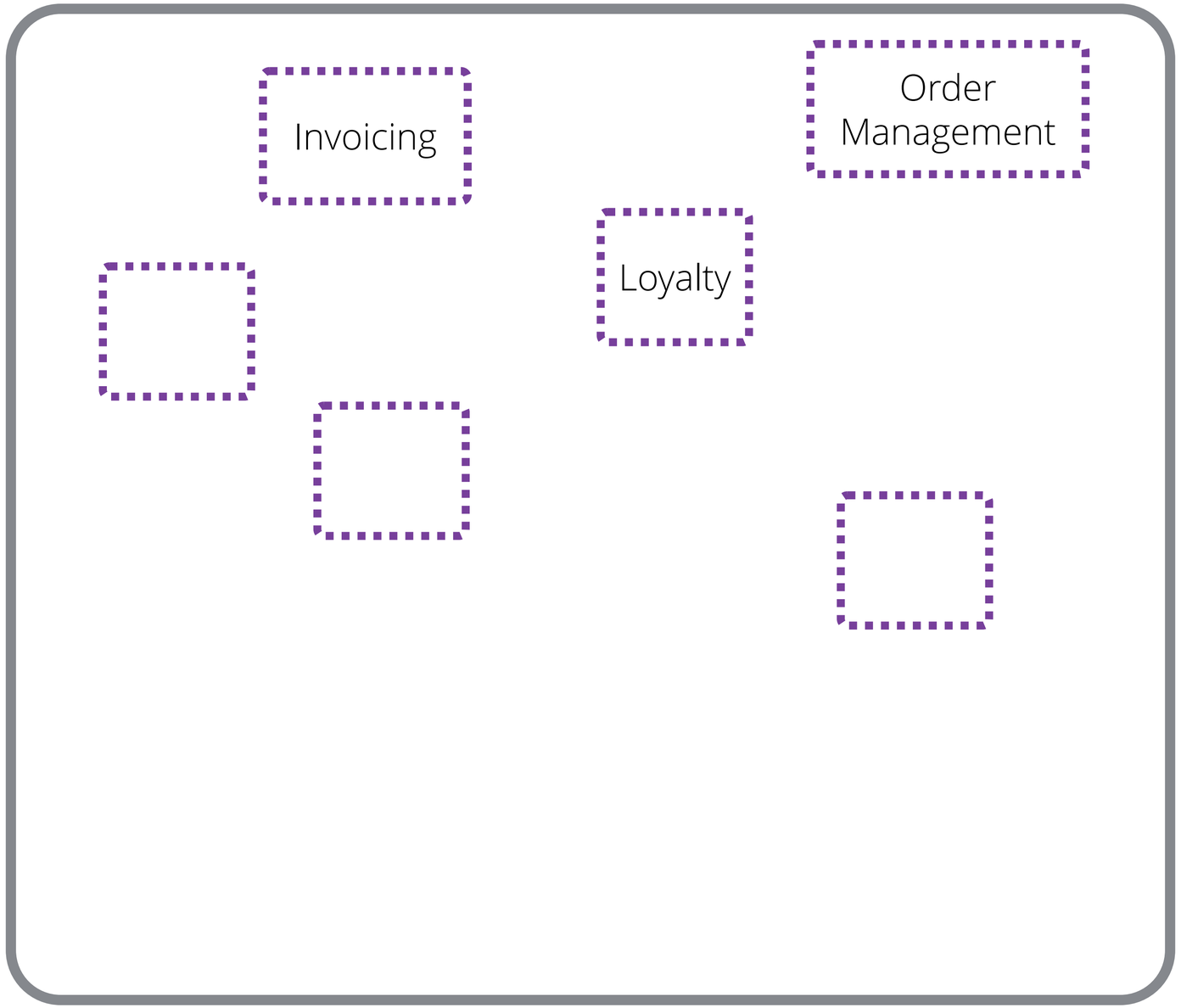

Order
Management

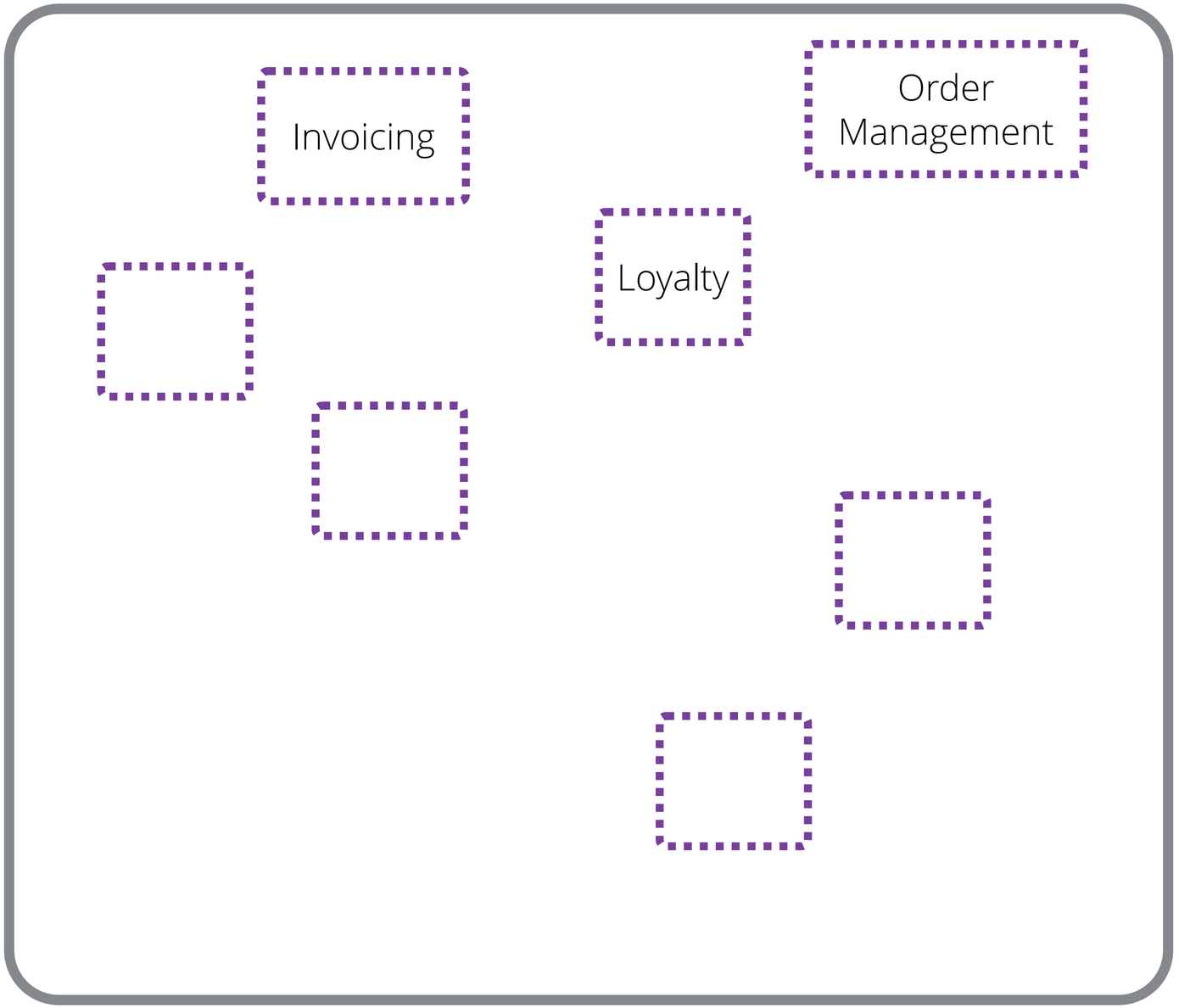

Invoicing
Order Management

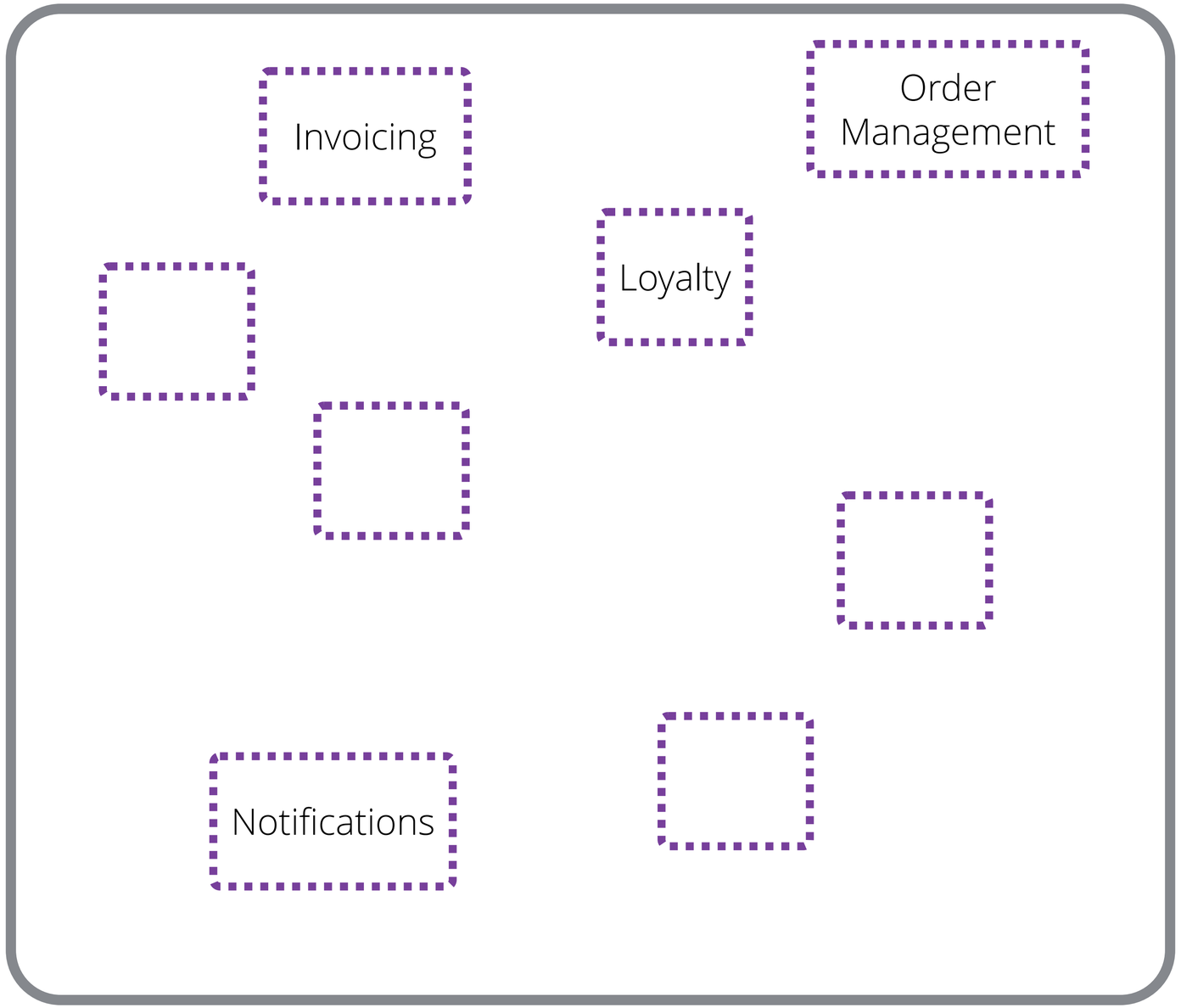

Invoicing
Order Management
Loyalty

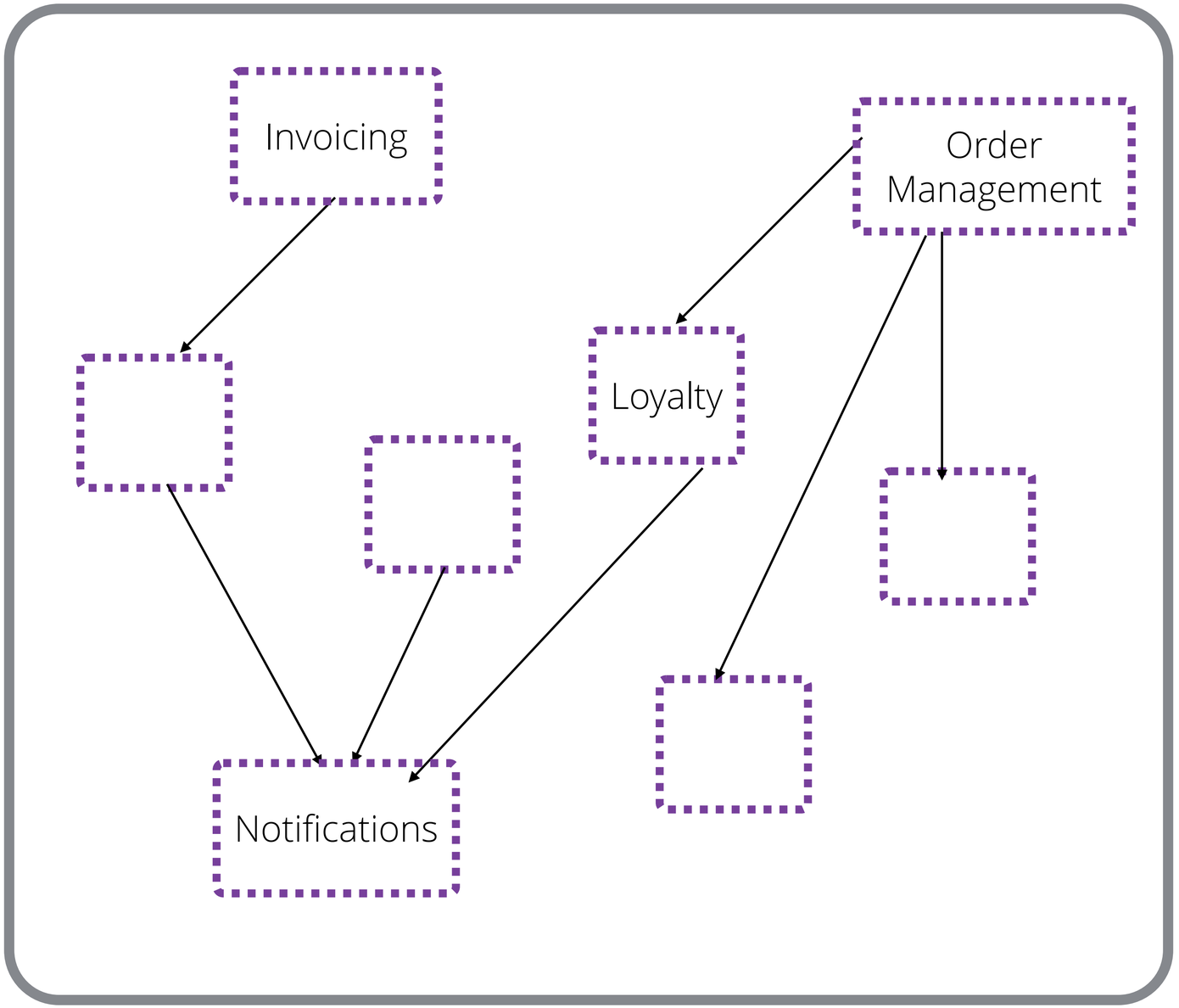

Invoicing
Order Management
Loyalty

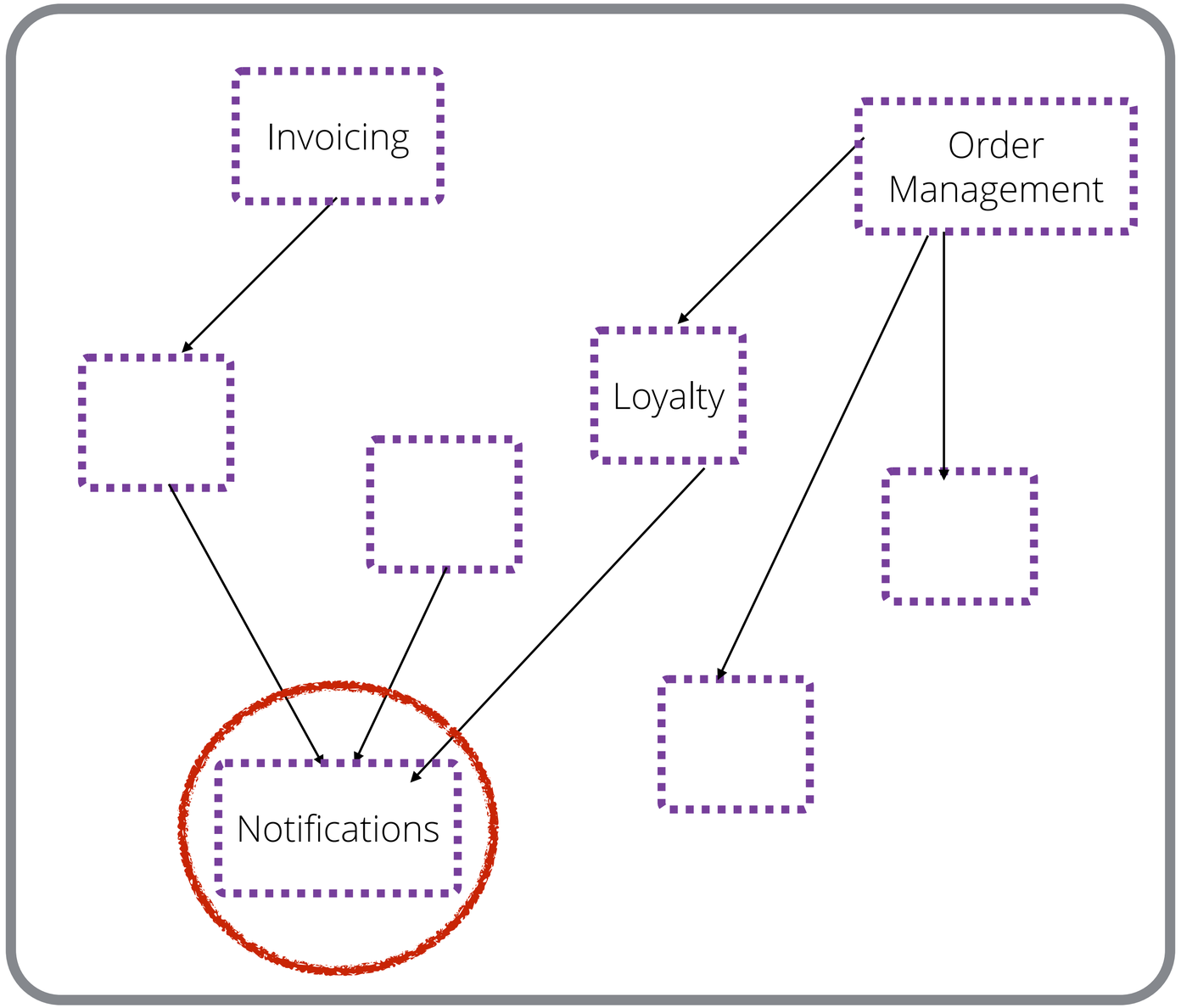

Invoicing
Order Management
Loyalty

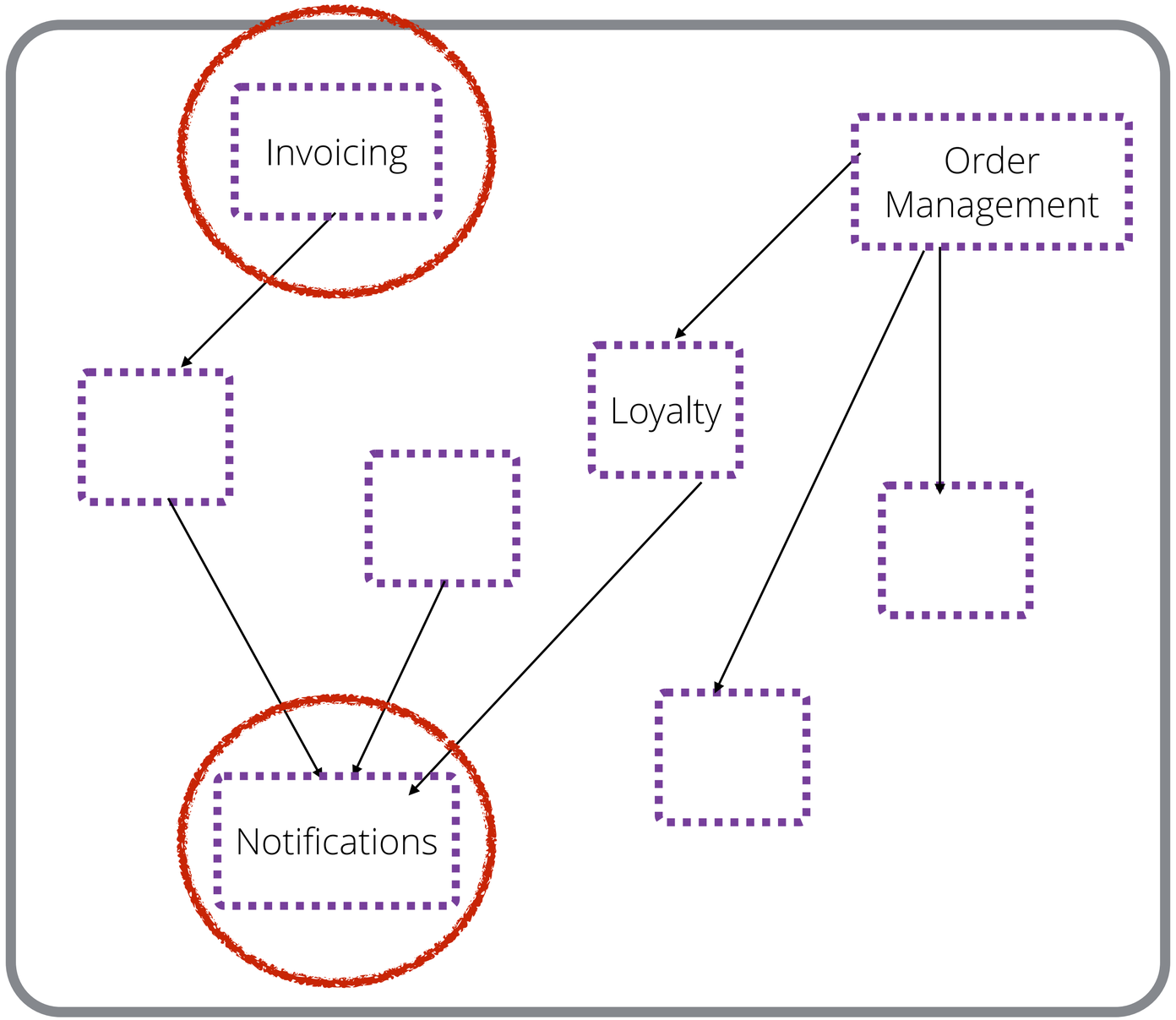

Invoicing
Order Management
Loyalty


Invoicing
Order Management
Loyalty


Invoicing
Order Management
Loyalty
Notifications


Invoicing
Order Management
Loyalty
Notifications

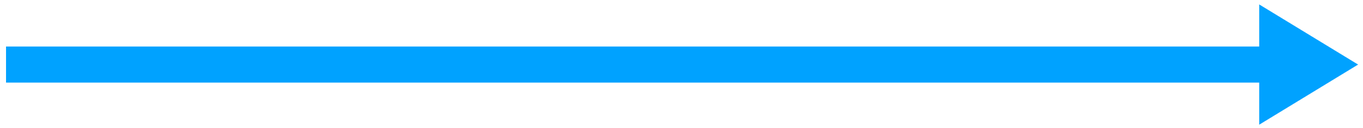

Invoicing
Order Management
Loyalty
Notifications


Invoicing
Order Management
Loyalty
Notifications

IT’S ALL ABOUT TRADEOFFS

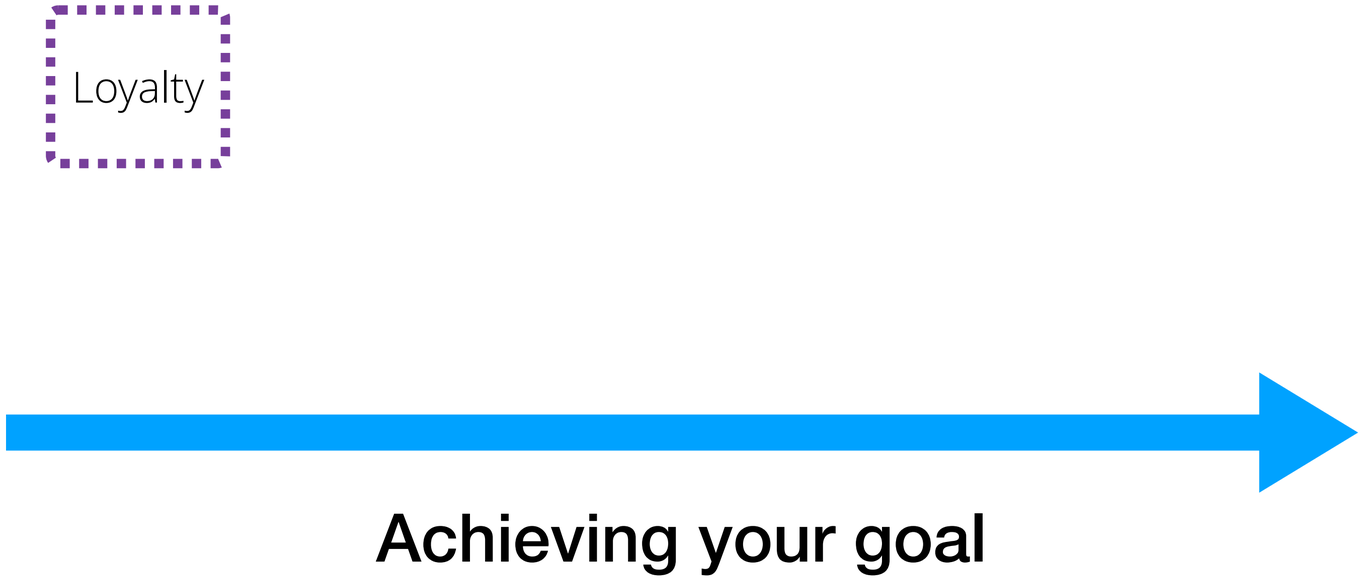
Achieving your goal

IT’S ALL ABOUT TRADEOFFS

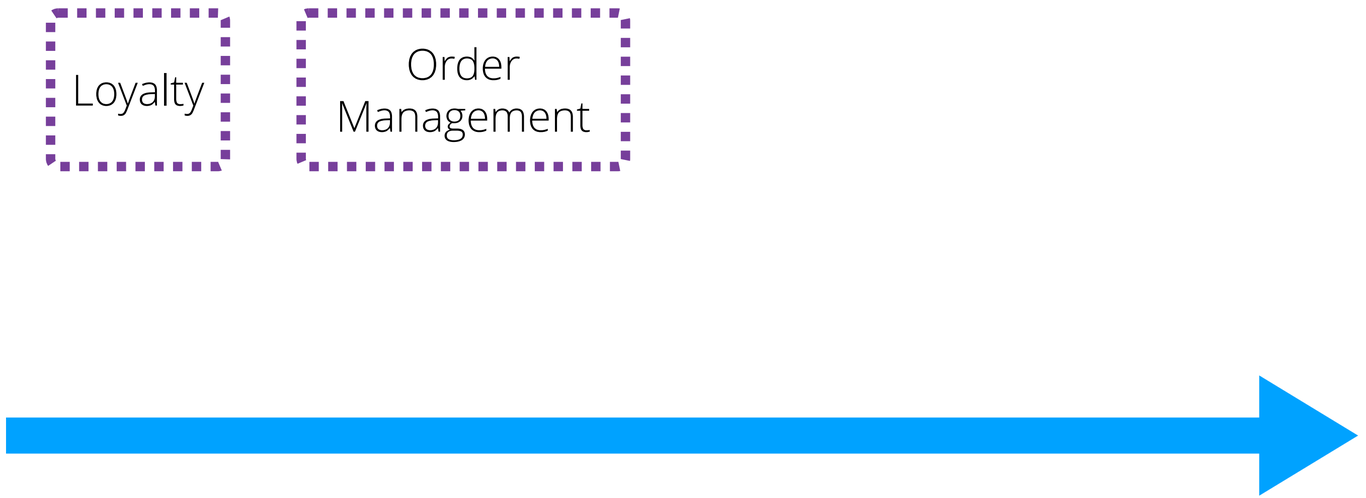


Achieving your goal

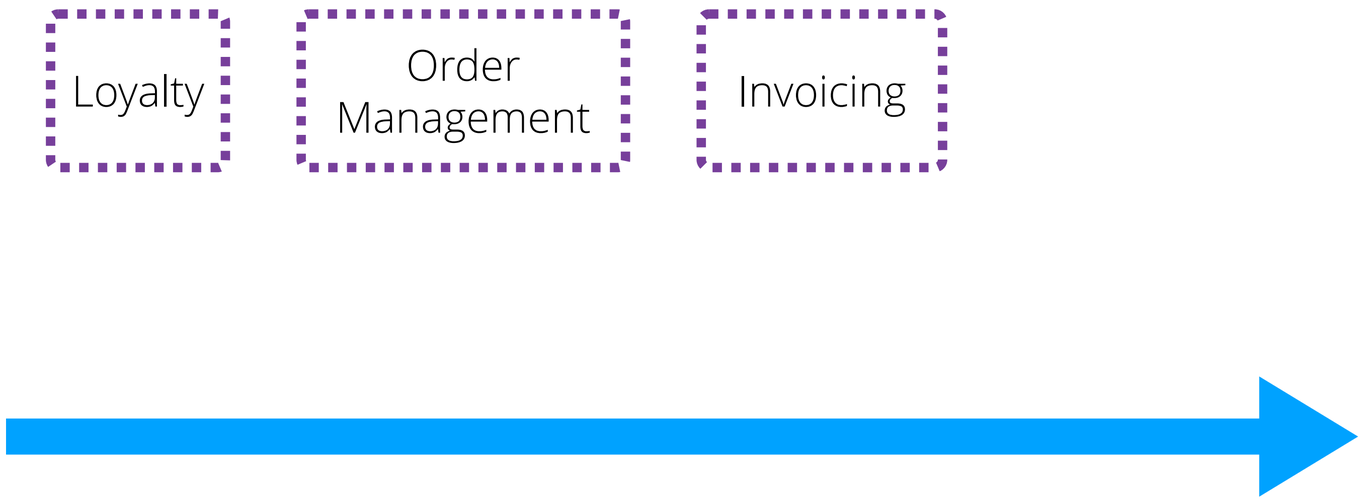
IT’S ALL ABOUT TRADEOFFS



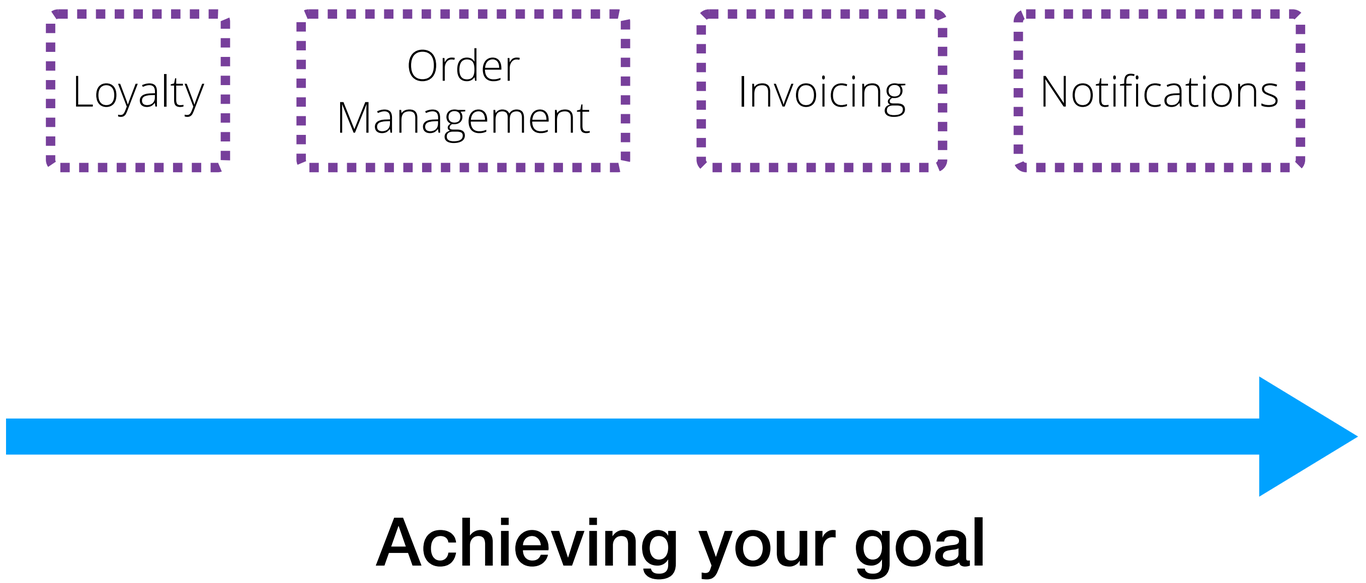
IT'S ALL ABOUT TRADEOFFS



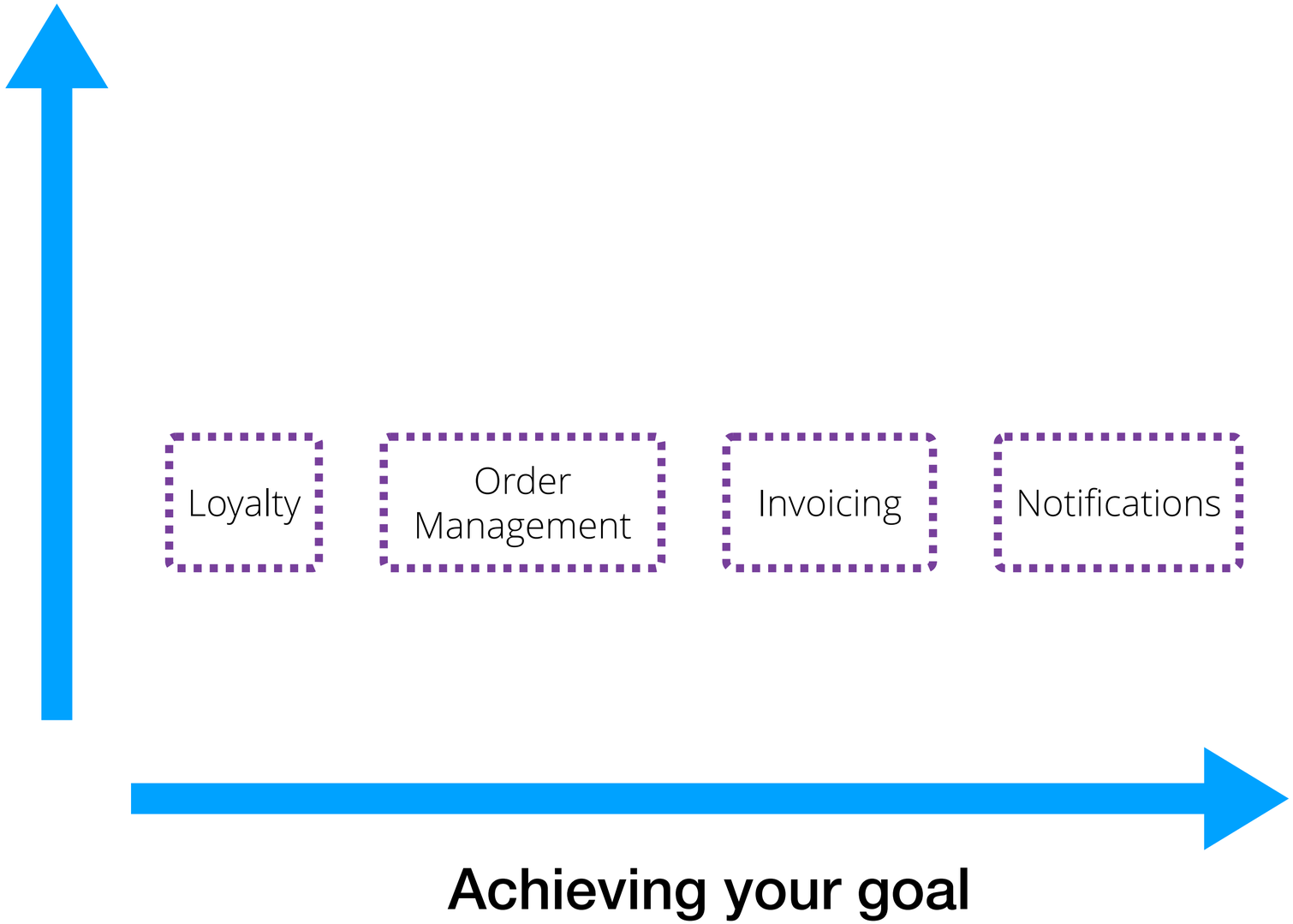
IT'S ALL ABOUT TRADEOFFS



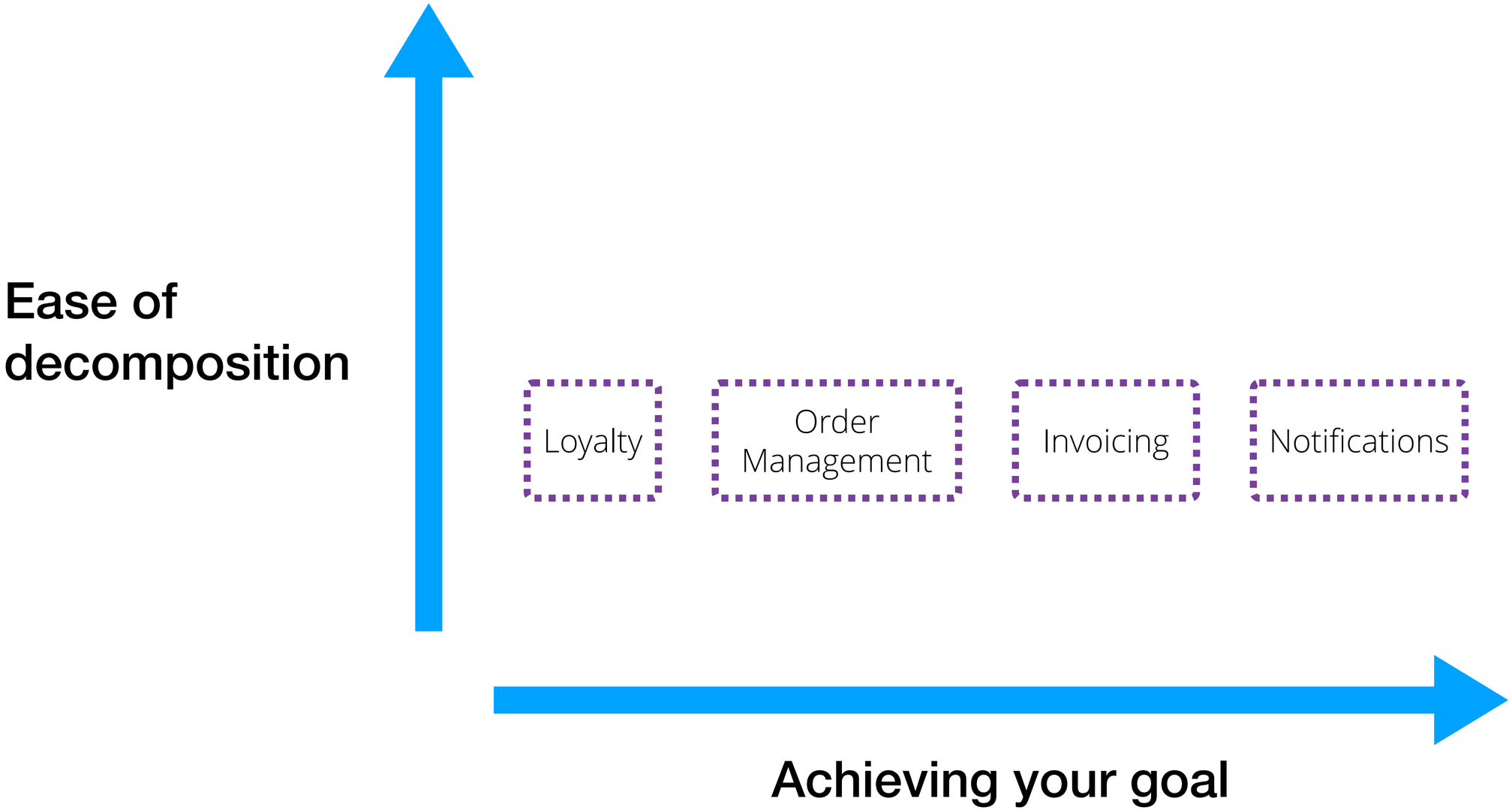
IT'S ALL ABOUT TRADEOFFS



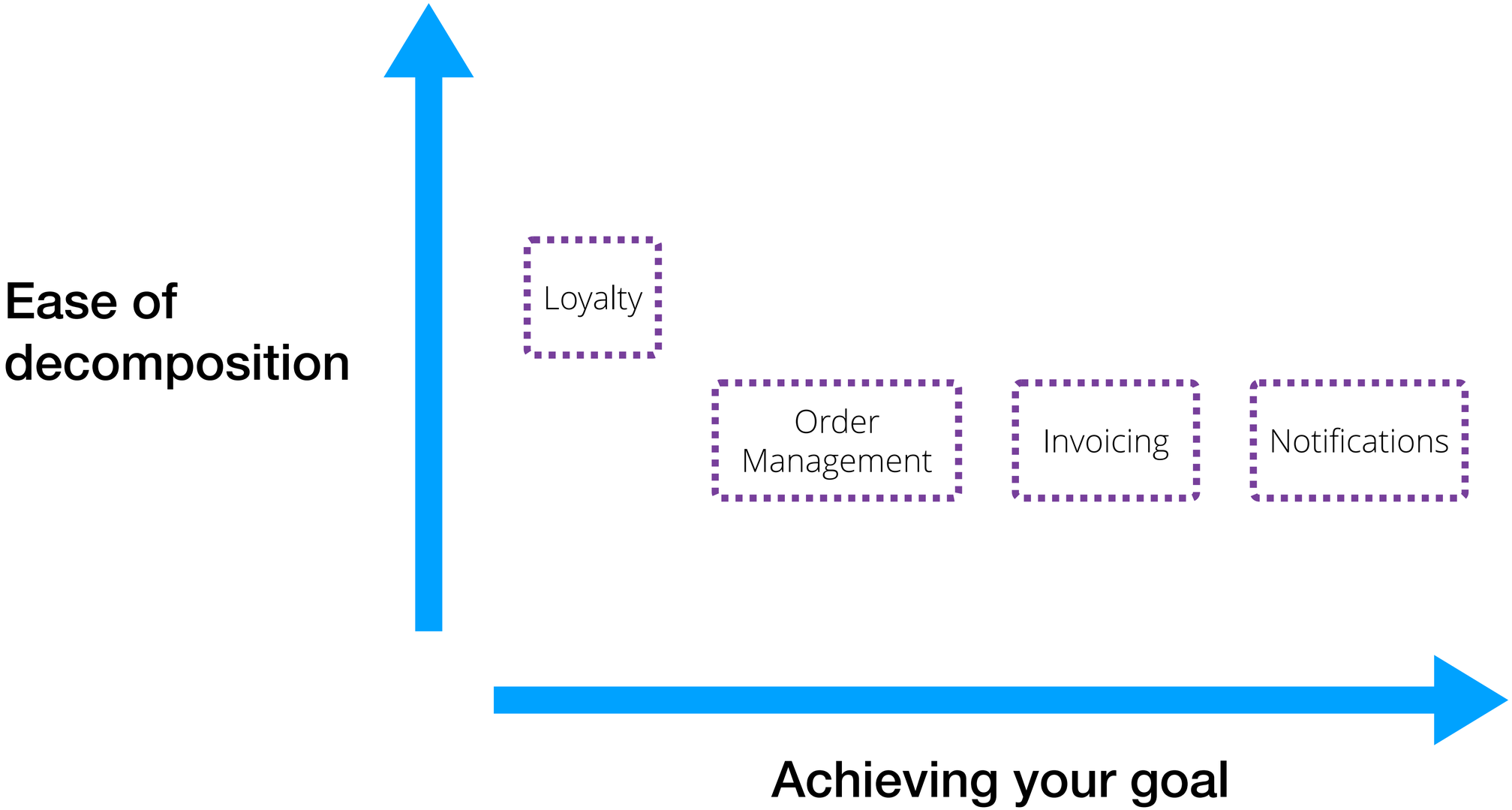
IT'S ALL ABOUT TRADEOFFS



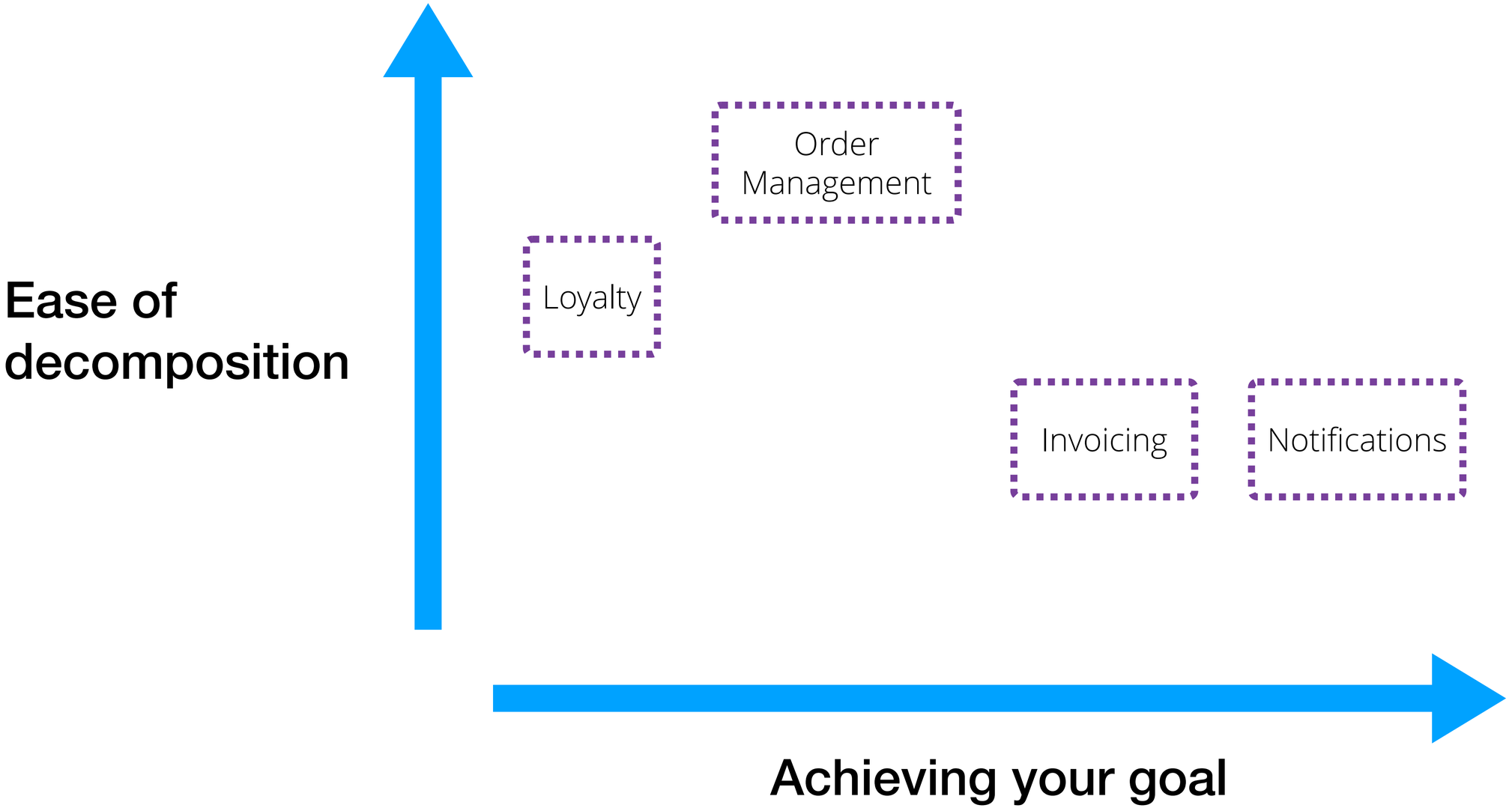
IT'S ALL ABOUT TRADEOFFS



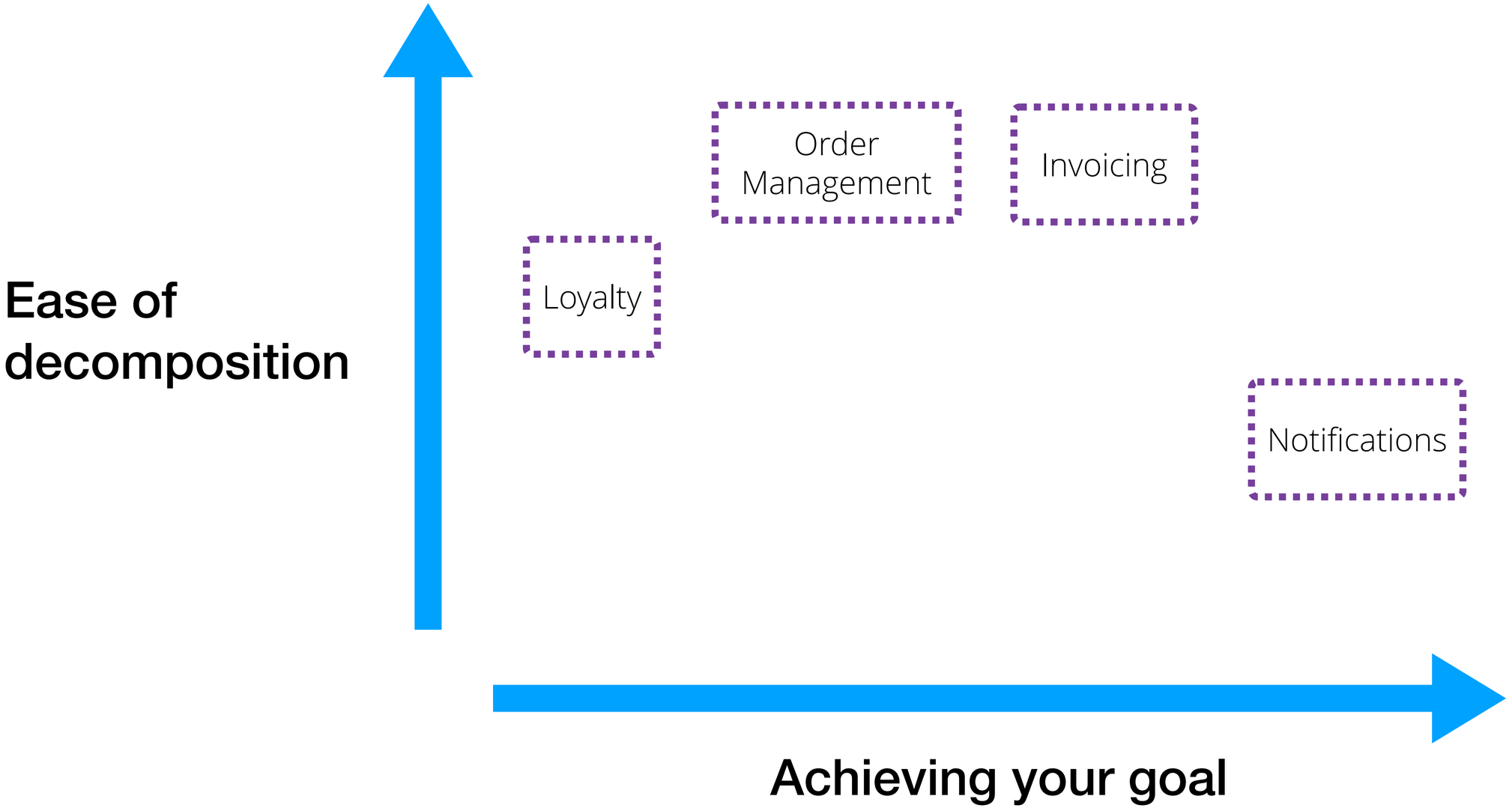
IT'S ALL ABOUT TRADEOFFS



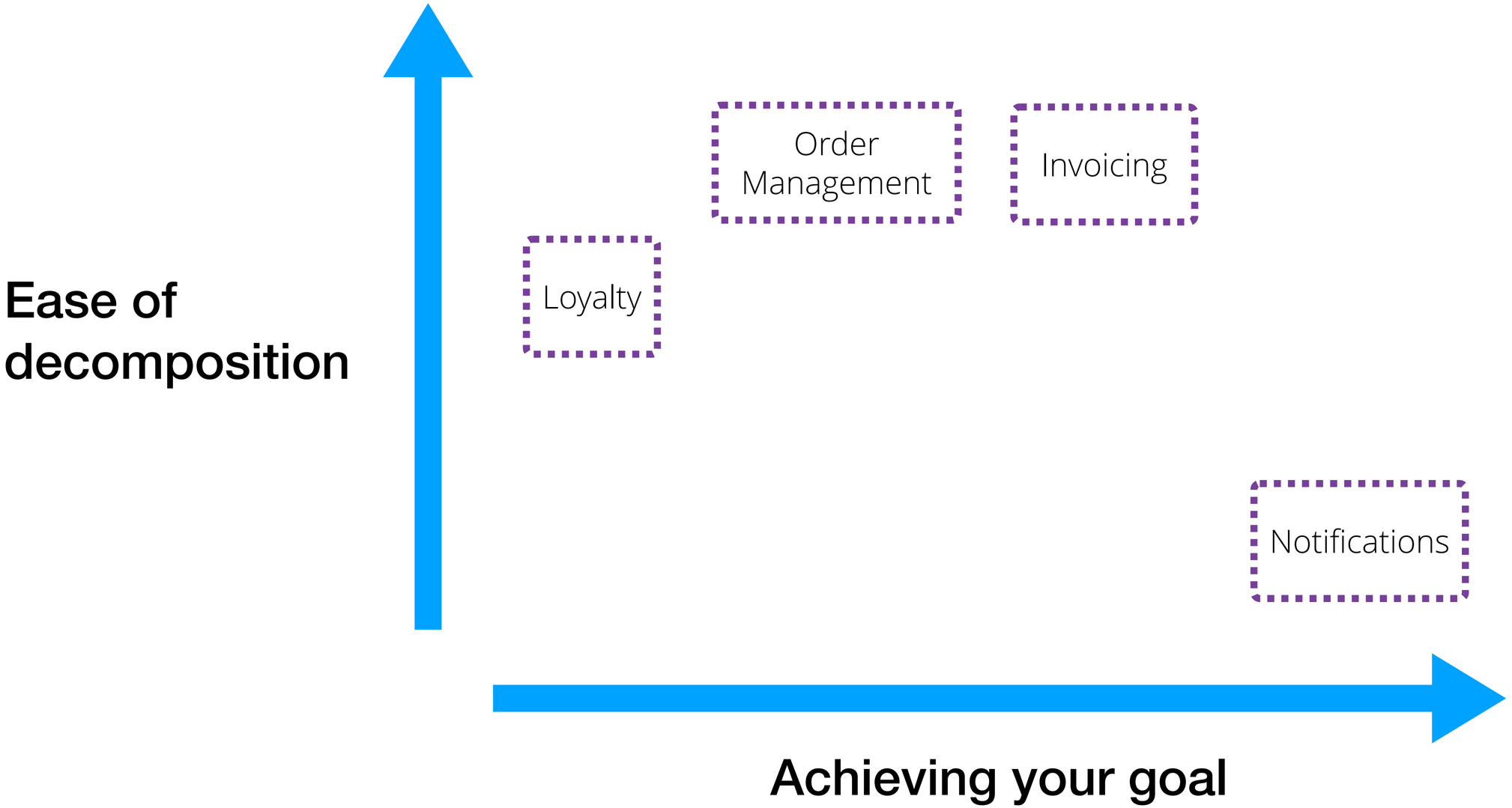
IT'S ALL ABOUT TRADEOFFS



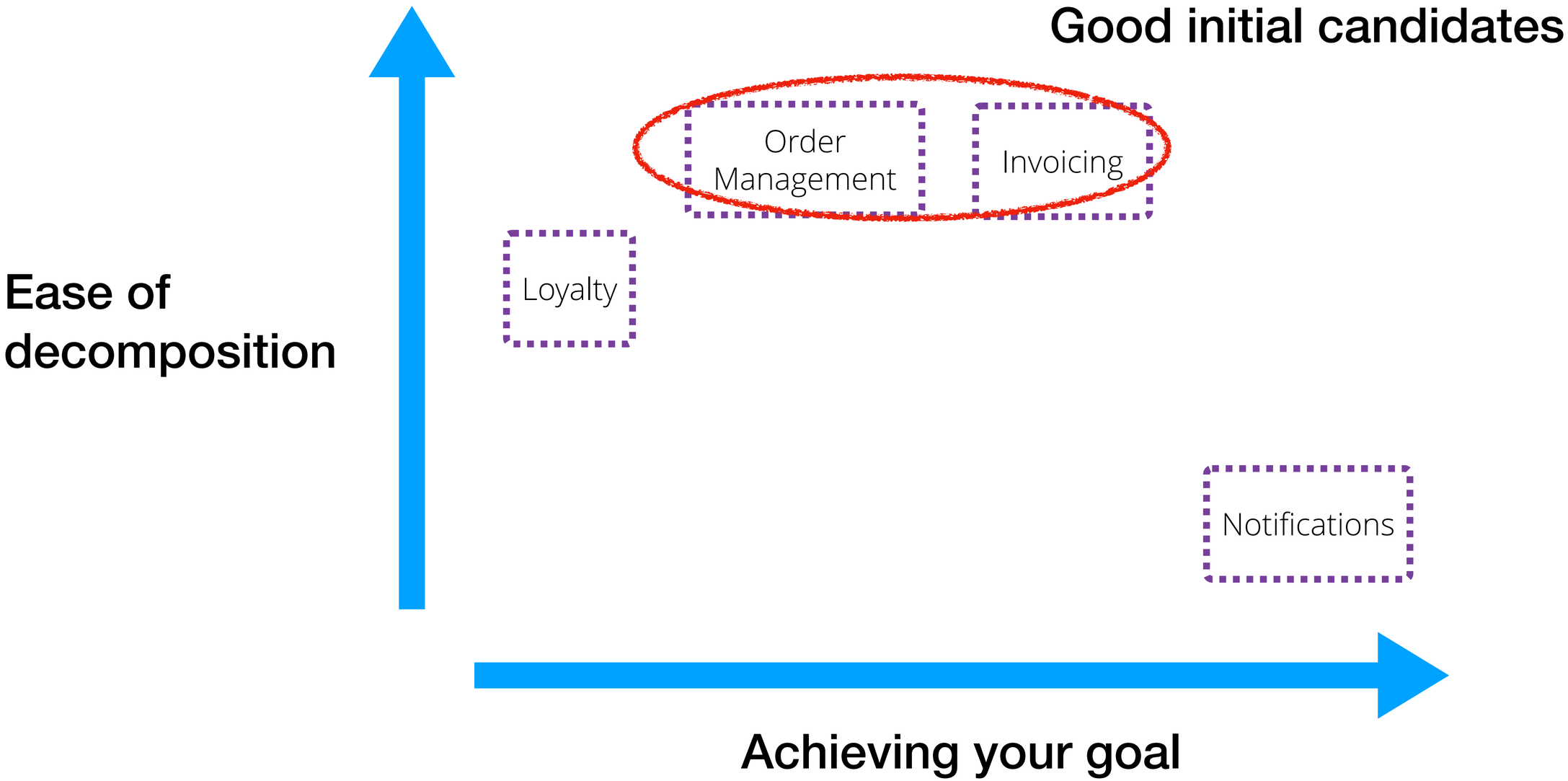
IT'S ALL ABOUT TRADEOFFS



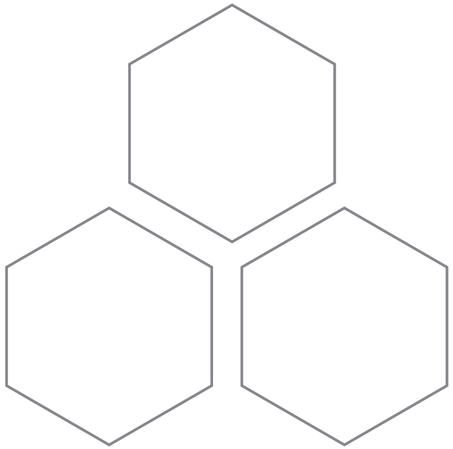
IT'S ALL ABOUT TRADEOFFS

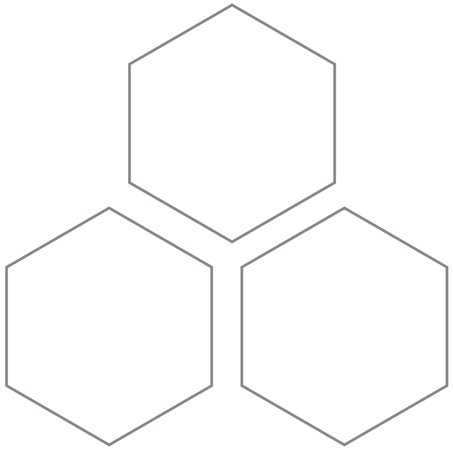


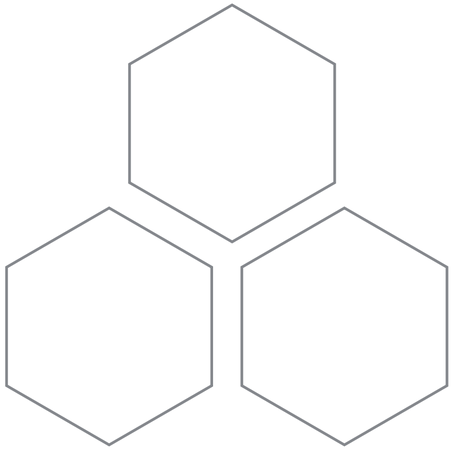
IT'S ALL ABOUT TRADEOFFS

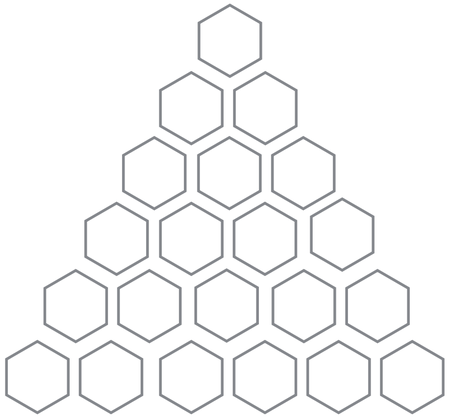


https://www.flickr.com/photos/orinrobertjohn/2911248047/

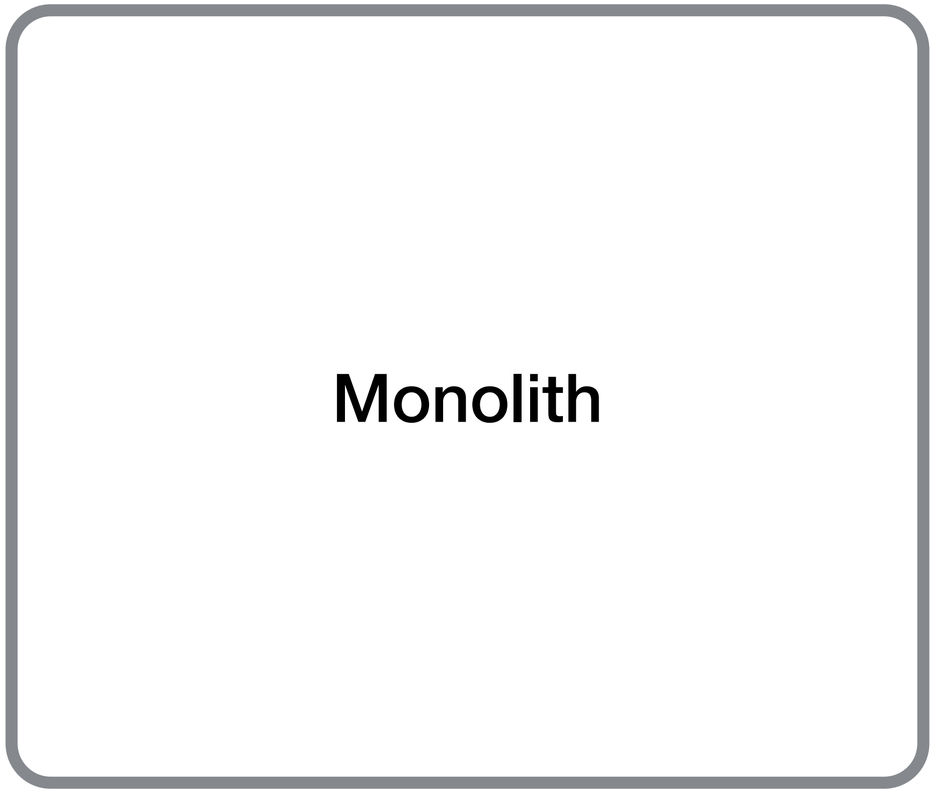






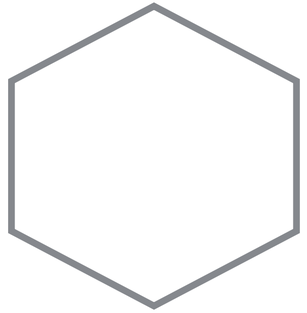


INCREMENTAL DECOMPOSITION FTW

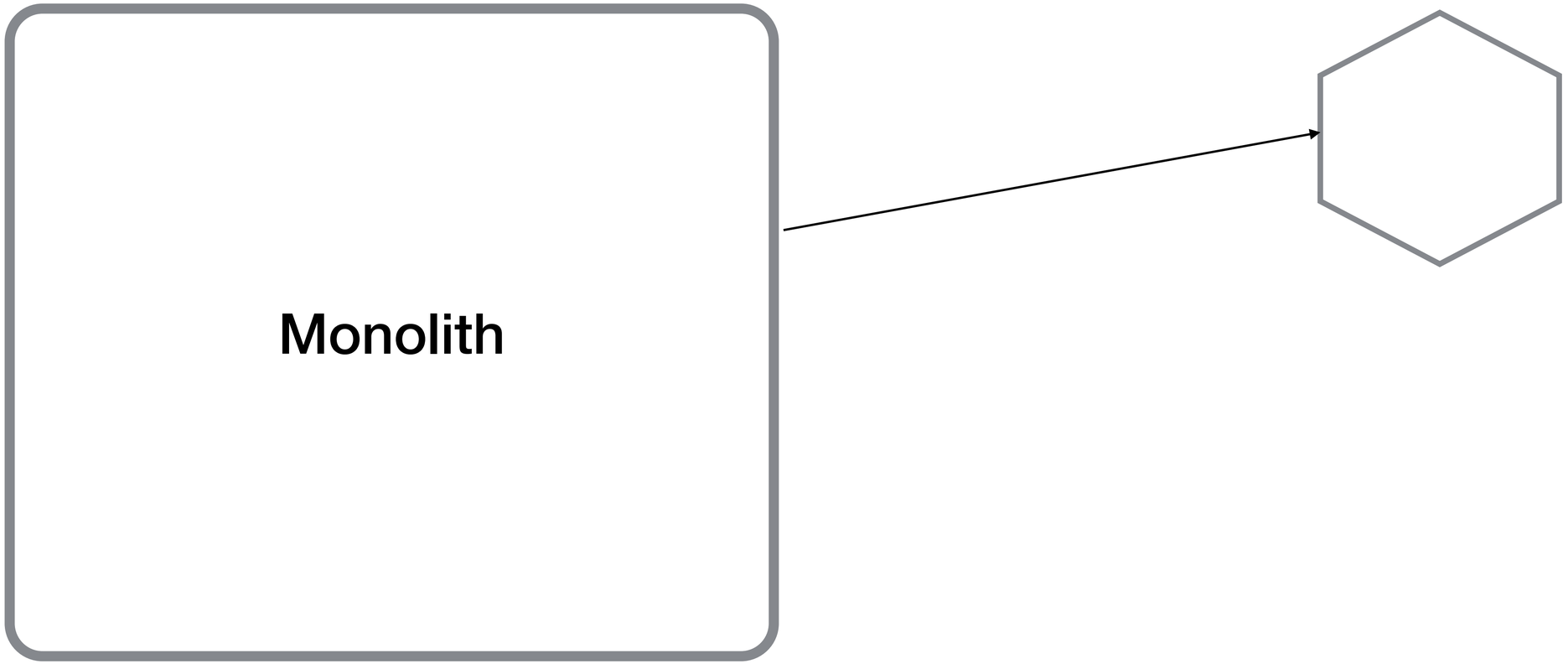


INCREMENTAL DECOMPOSITION FTW

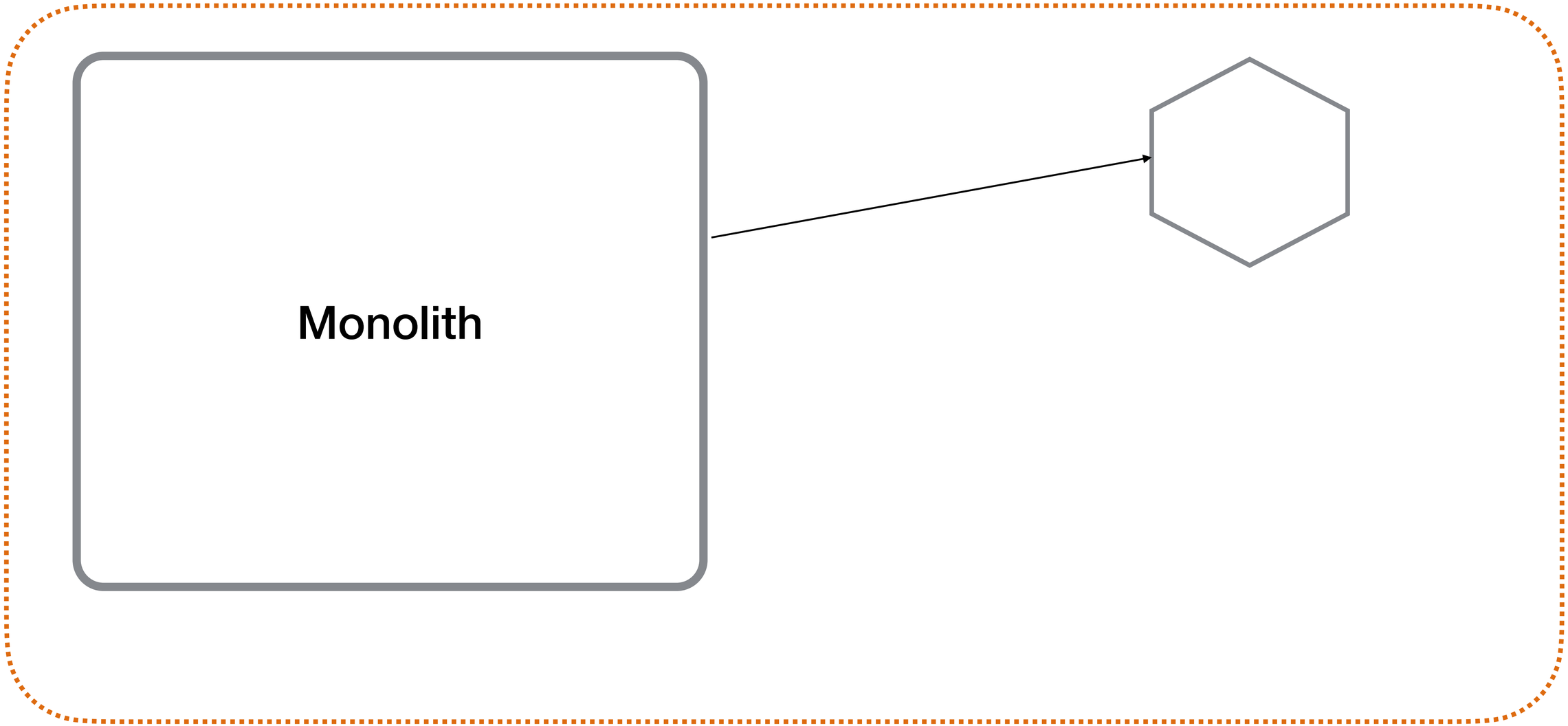
Monolith



INCREMENTAL DECOMPOSITION FTW



INCREMENTAL DECOMPOSITION FTW



You won’t appreciate the true horror, pain and suffering of microservices until you’re running them in production

“If you do a big bang rewrite, the only thing you’re certain of is a big bang”

- Martin Fowler (paraphrased)

Move functionality to microservices a piece at a time

Move functionality to microservices a piece at a time

Get it into production to start getting value from it, and learning from the experience

AGENDA

Introduction

Migration Approach

Application Refactoring

UI Decomposition

AGENDA

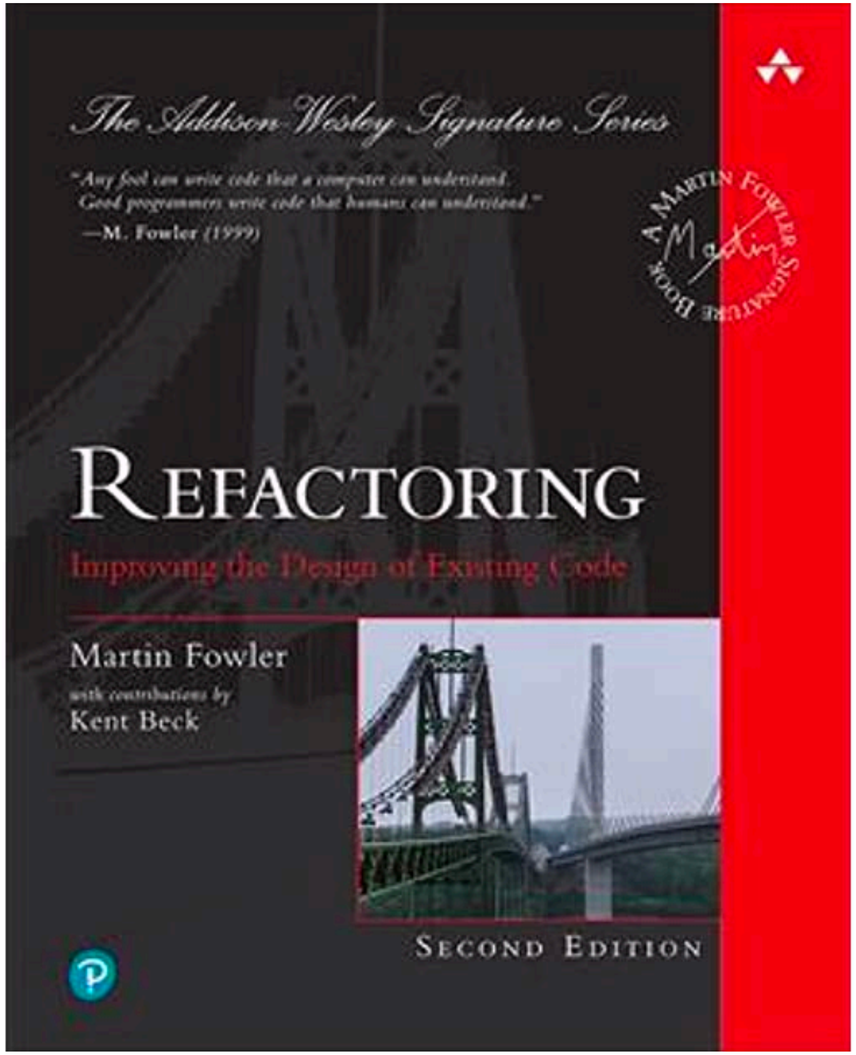
Introduction

Migration Approach

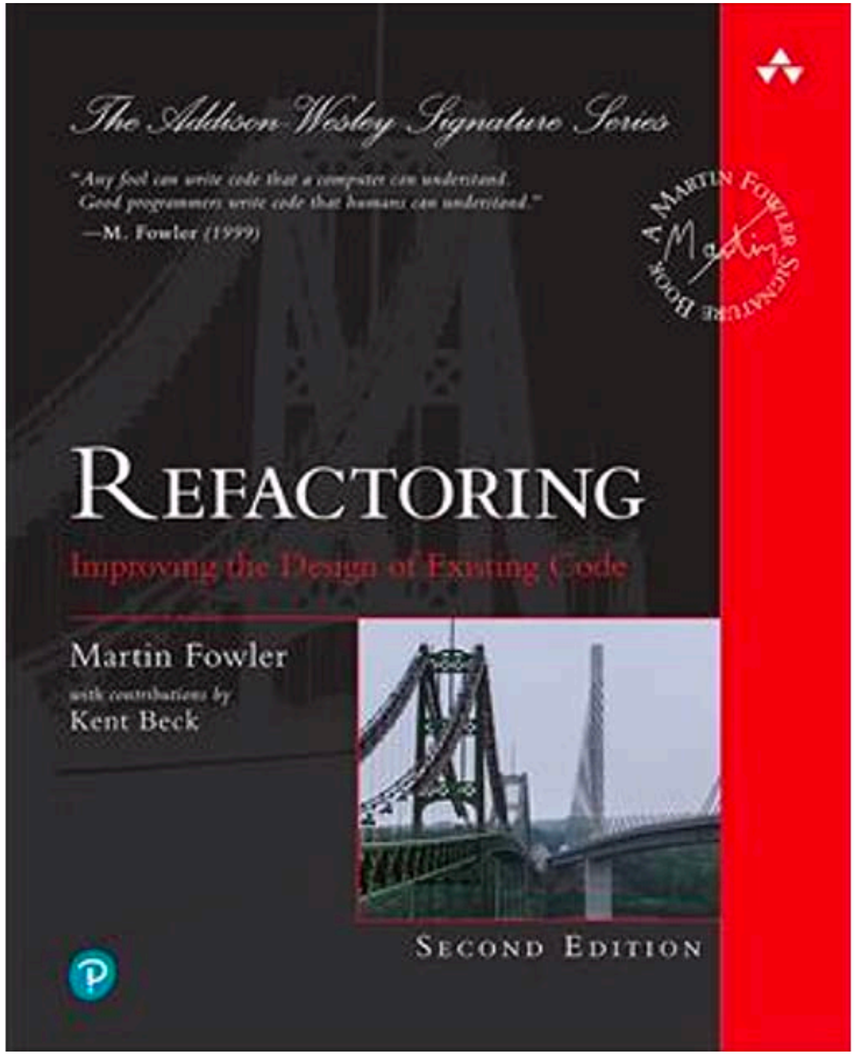
Application Refactoring

UI Decomposition

ON REFACTORING

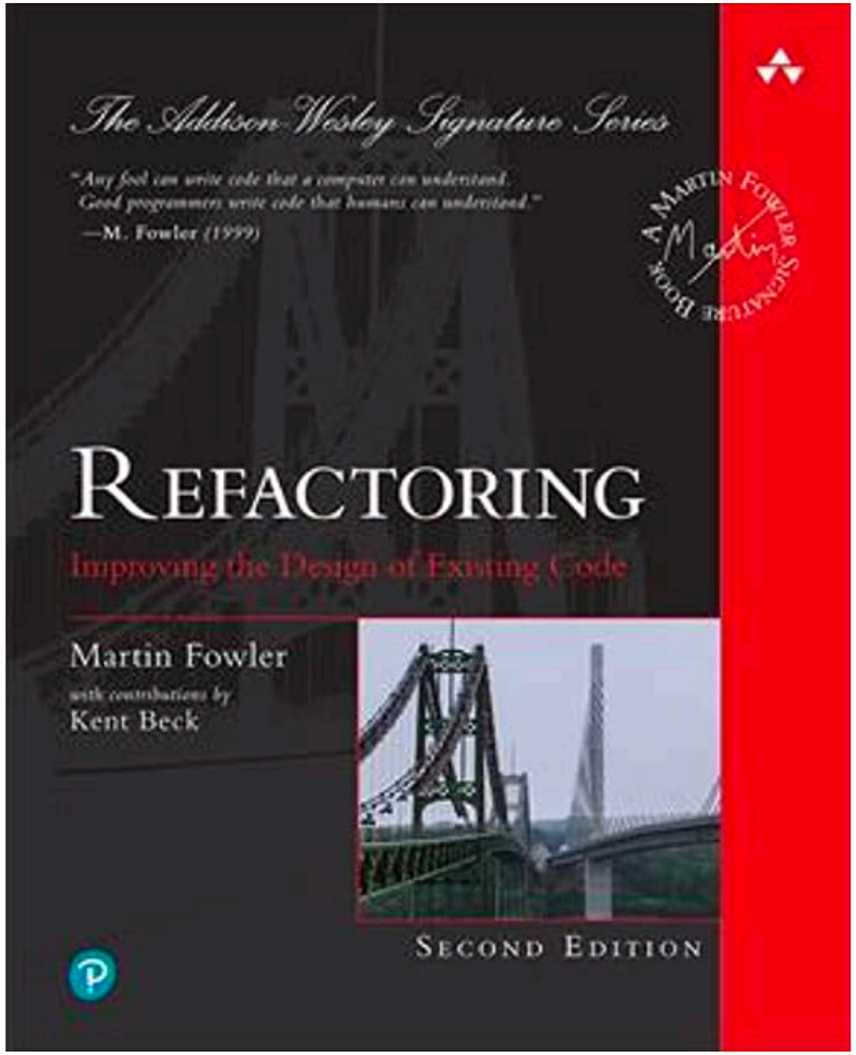


ON REFACTORING



Changing the structure of the code without changing behaviour

ON REFACTORING



Changing the structure of the code without changing behaviour

By not changing behaviour, we can compare the software before and after the change to make sure it still works

PATTERN: STRANGLER FIG

By Poyt448 Peter Woodard - Own work, Public Domain, https://commons.wikimedia.org/w/index.php?curid=5920414

PATTERN: STRANGLER FIG

By Poyt448 Peter Woodard - Own work, Public Domain, https://commons.wikimedia.org/w/index.php?curid=5920414

Wrap a new system around the old system

PATTERN: STRANGLER FIG

By Poyt448 Peter Woodard - Own work, Public Domain, https://commons.wikimedia.org/w/index.php?curid=5920414

Wrap a new system around the old system

The new system doesn't need to fully replace the old

PATTERN: STRANGLER FIG

By Poyt448 Peter Woodard - Own work, Public Domain, https://commons.wikimedia.org/w/index.php?curid=5920414

Wrap a new system around the old system

The new system doesn't need to fully replace the old

Calls to functionality in the new system are intercepted, other calls are served by the old system

STRANGLER FIG IMPLEMENTATION

Intercept calls on permitter of old system (the monolith)

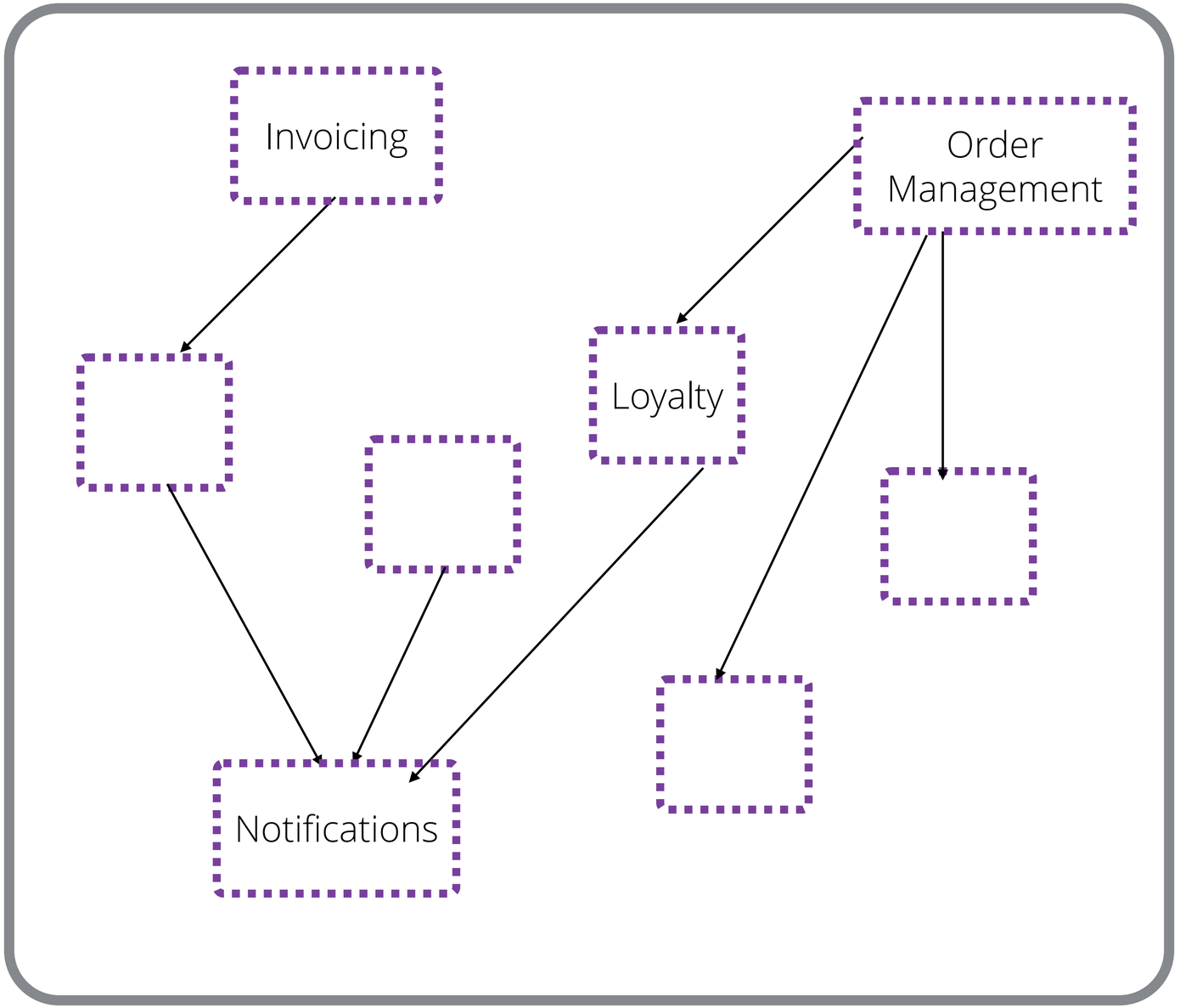
STRANGLER FIG IMPLEMENTATION

Intercept calls on permitter of old system (the monolith)

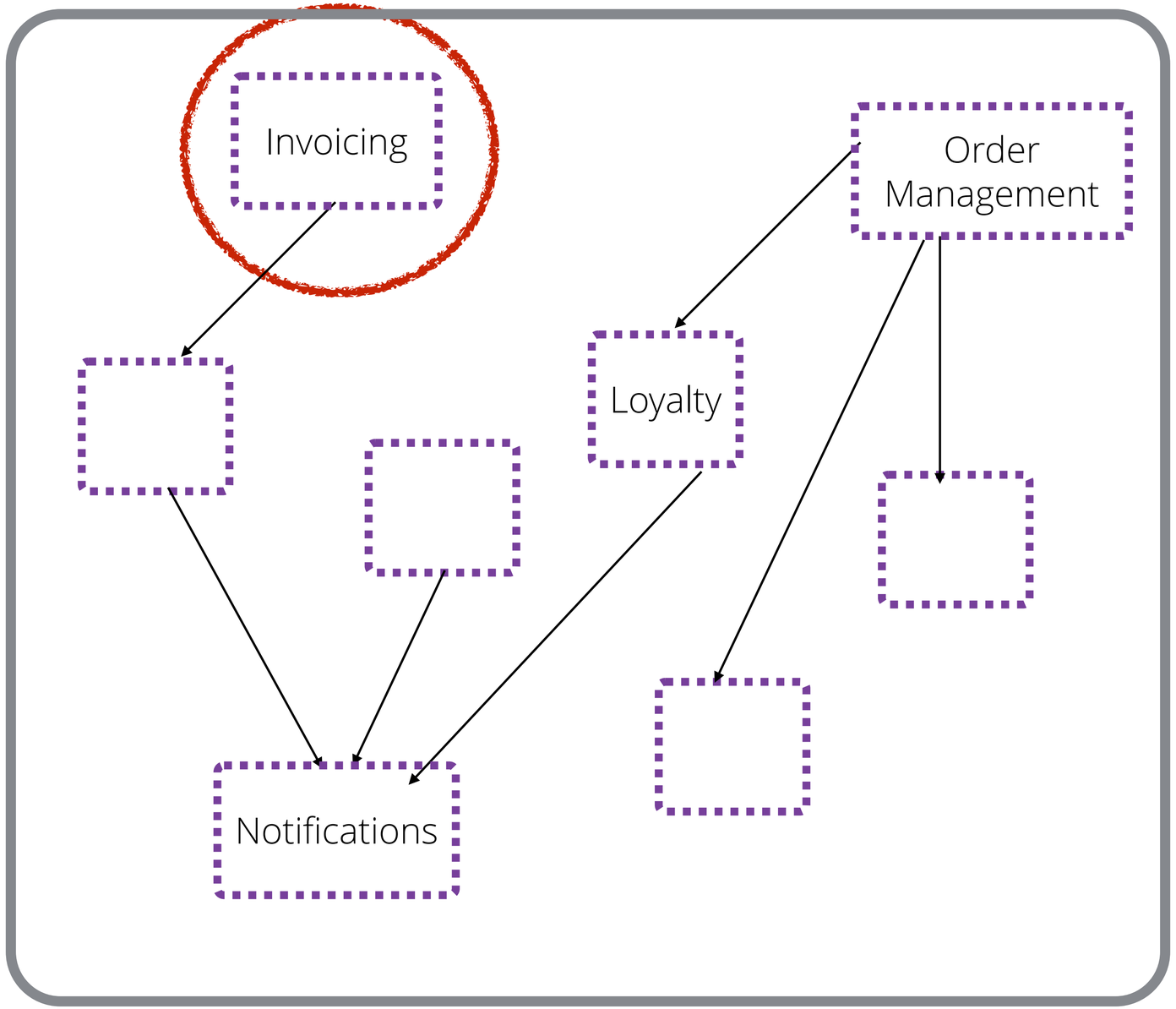
Divert calls to where the functionality is located

CANDIDATES

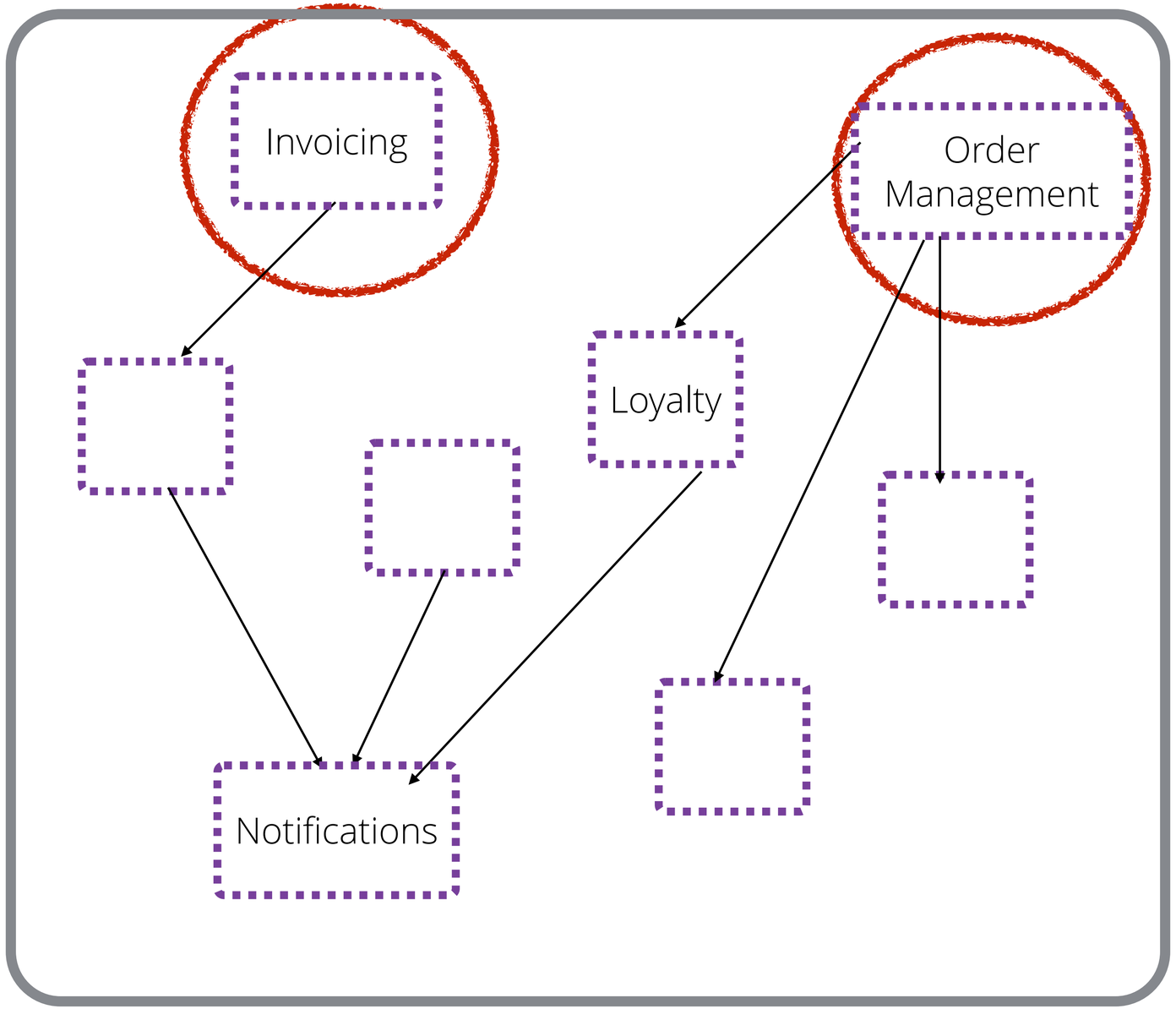
CANDIDATES



CANDIDATES

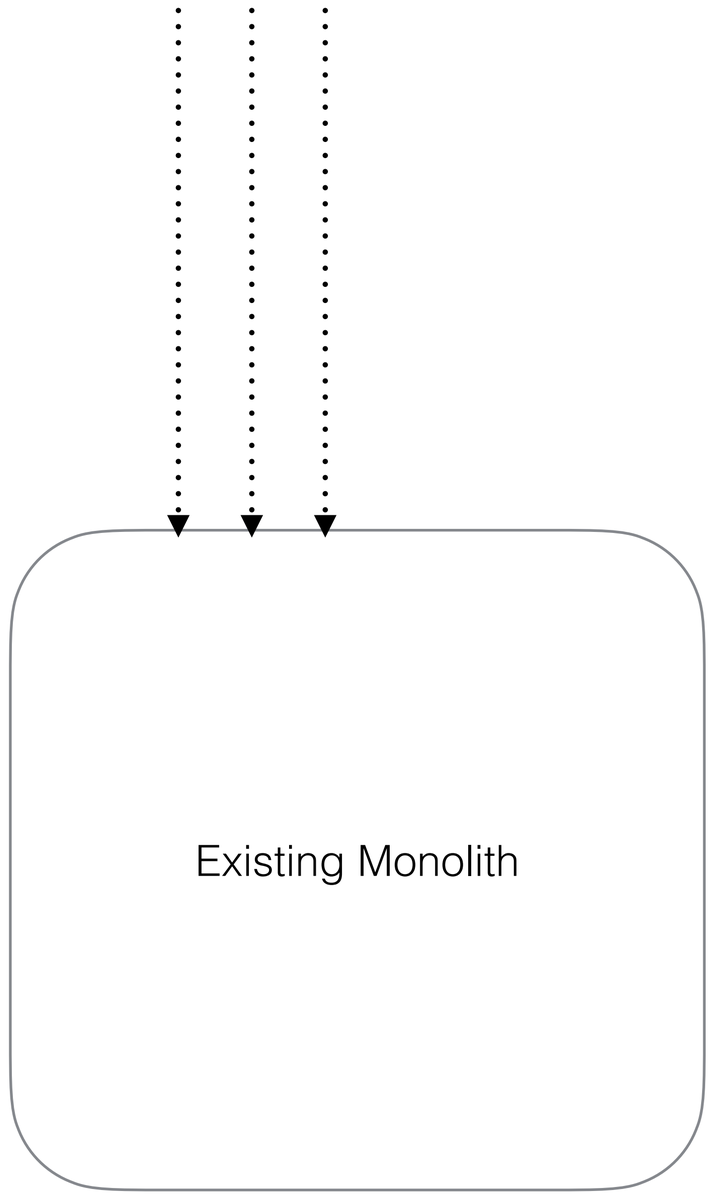


CANDIDATES

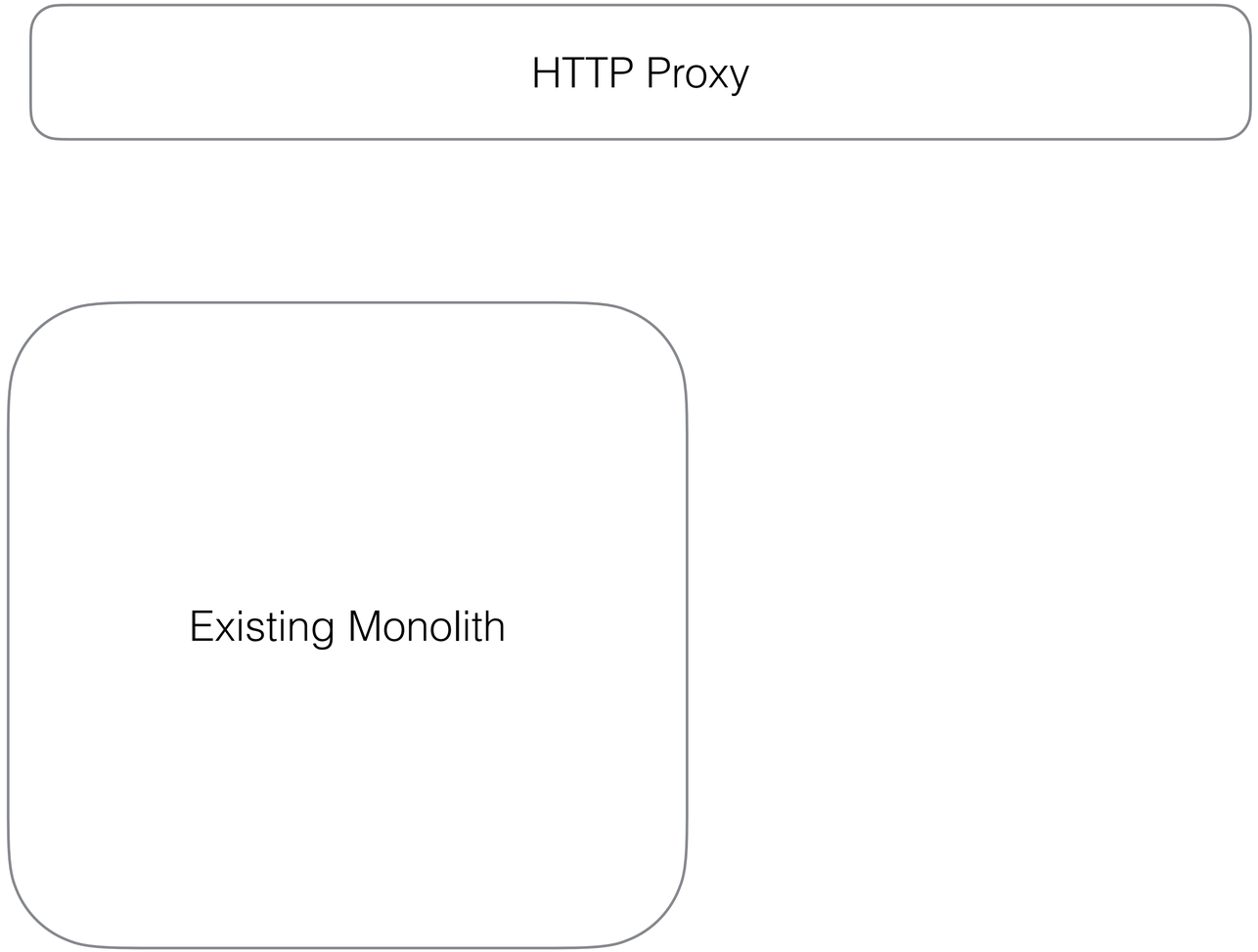


INTERCEPTION

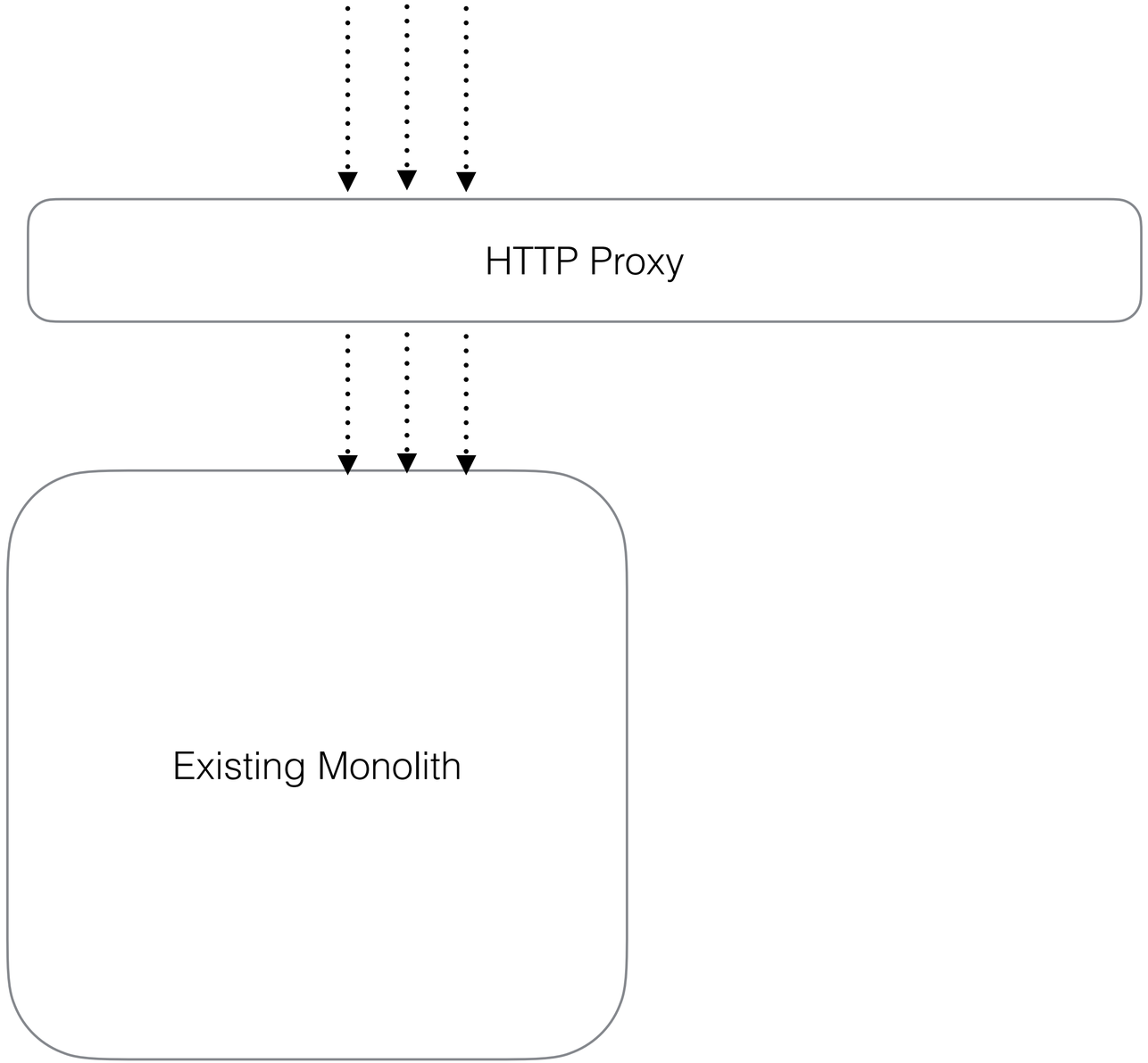
INTERCEPTION



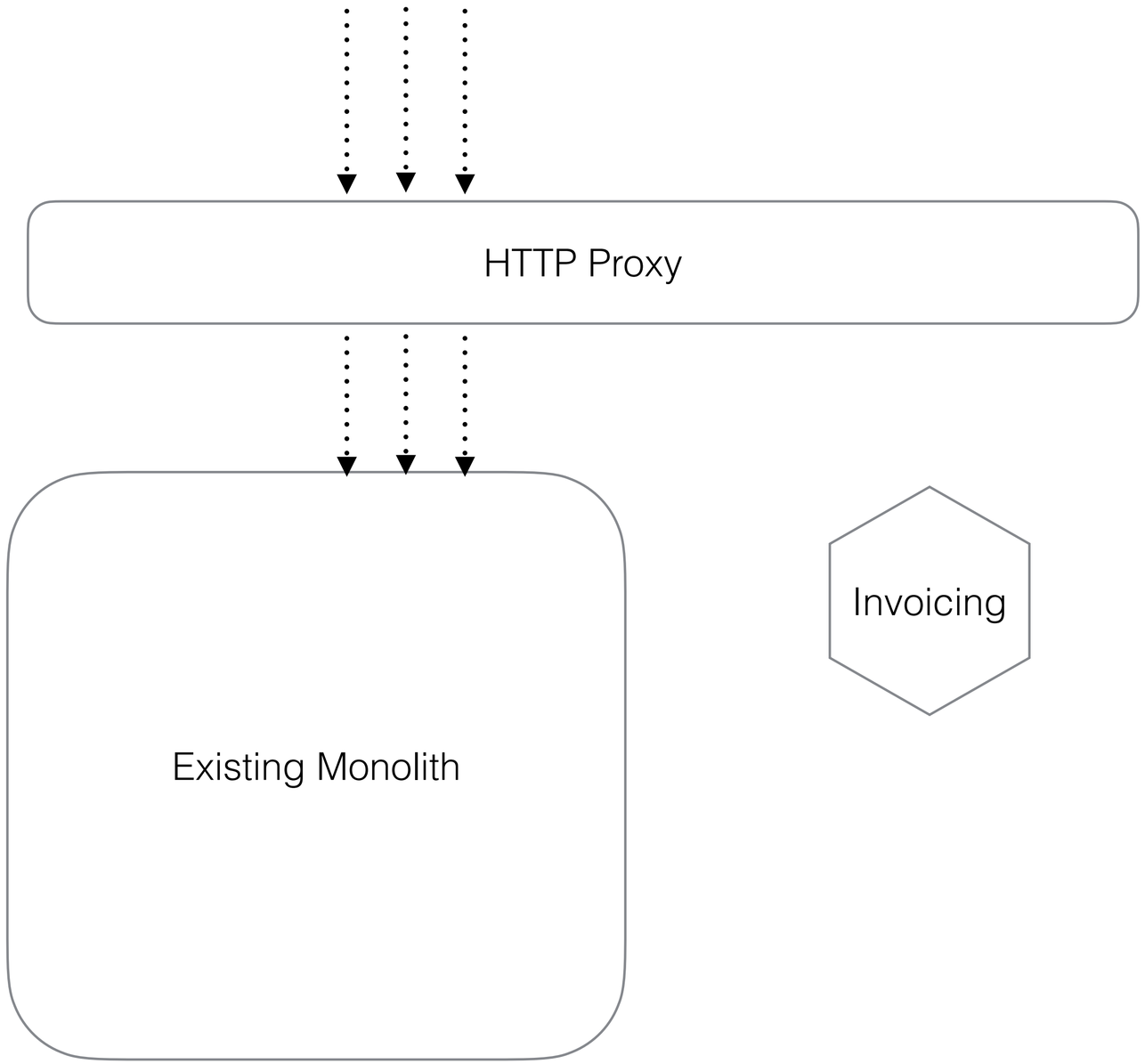
INTERCEPTION



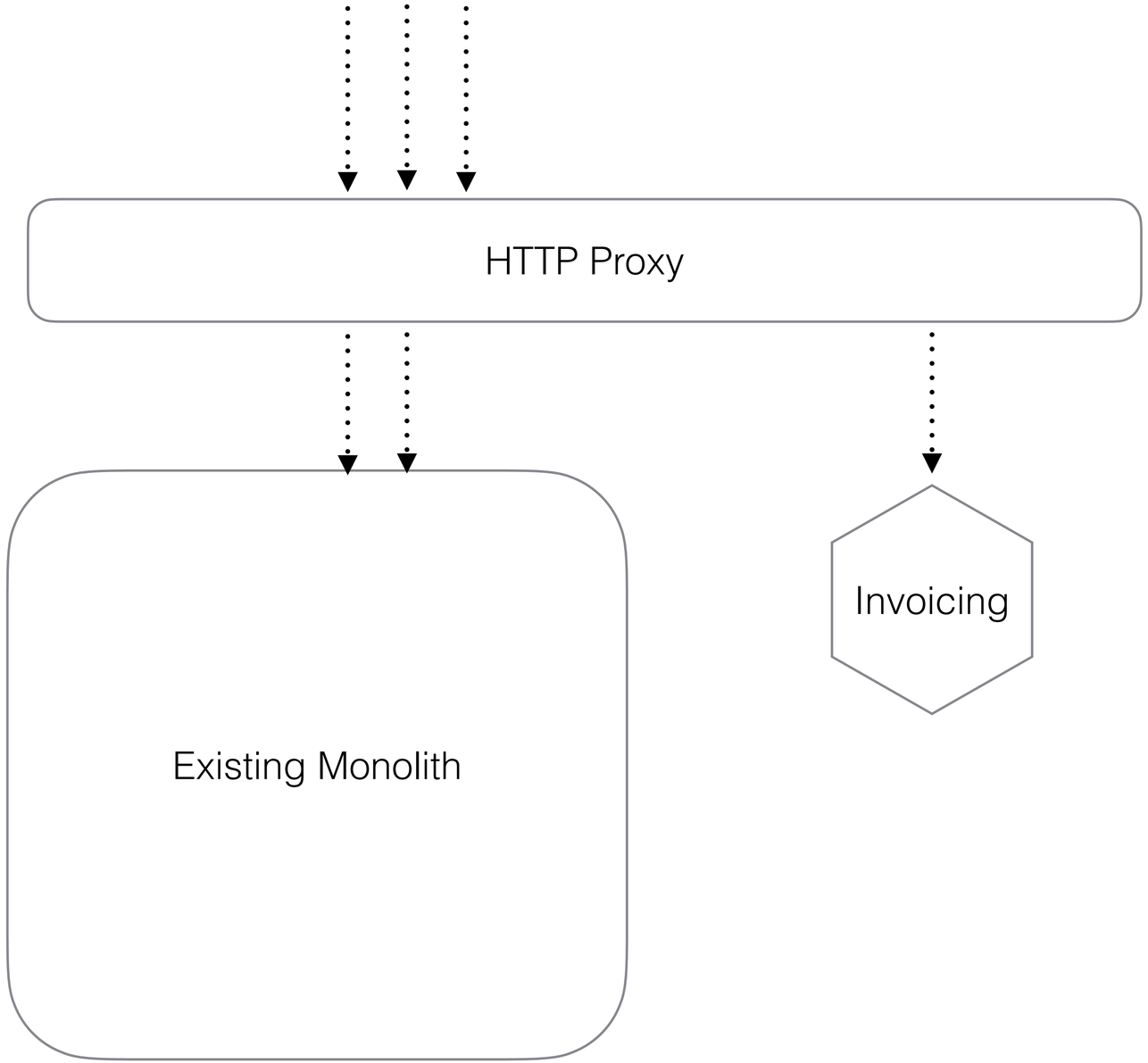
INTERCEPTION



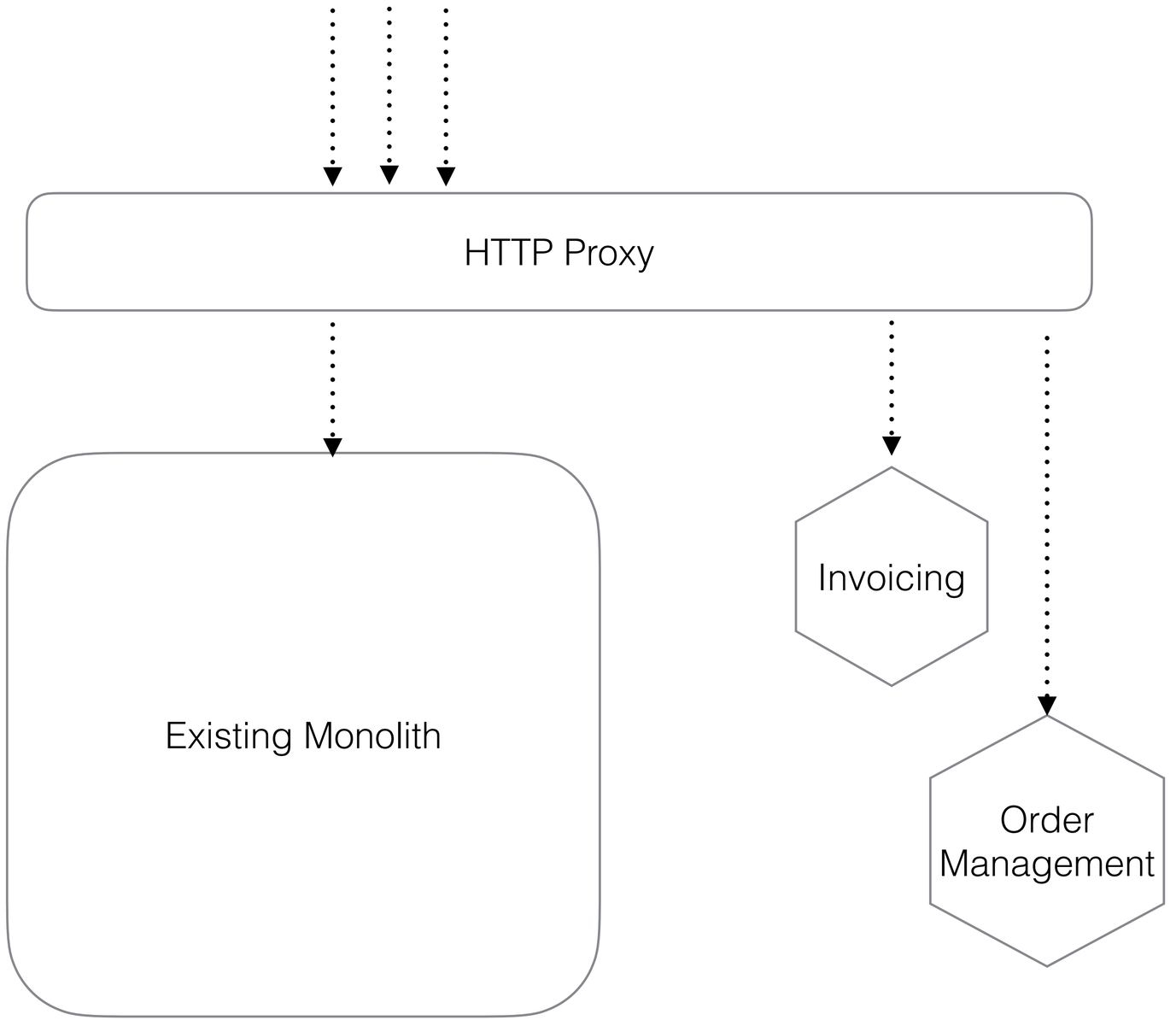
INTERCEPTION



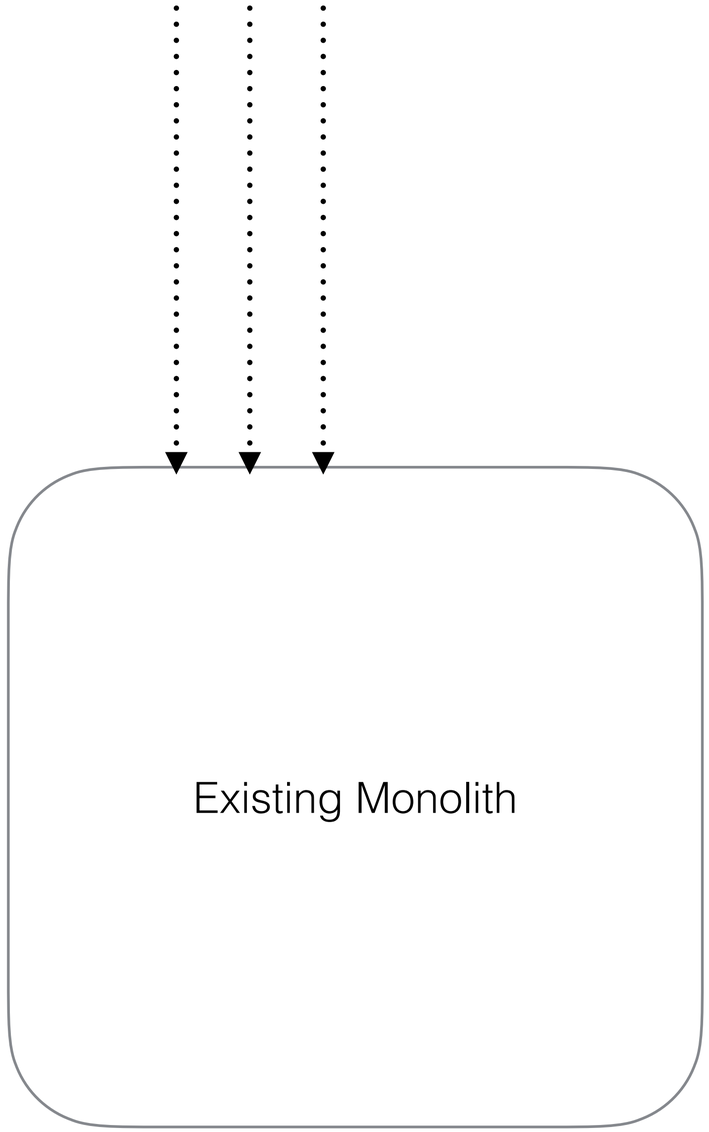
INTERCEPTION



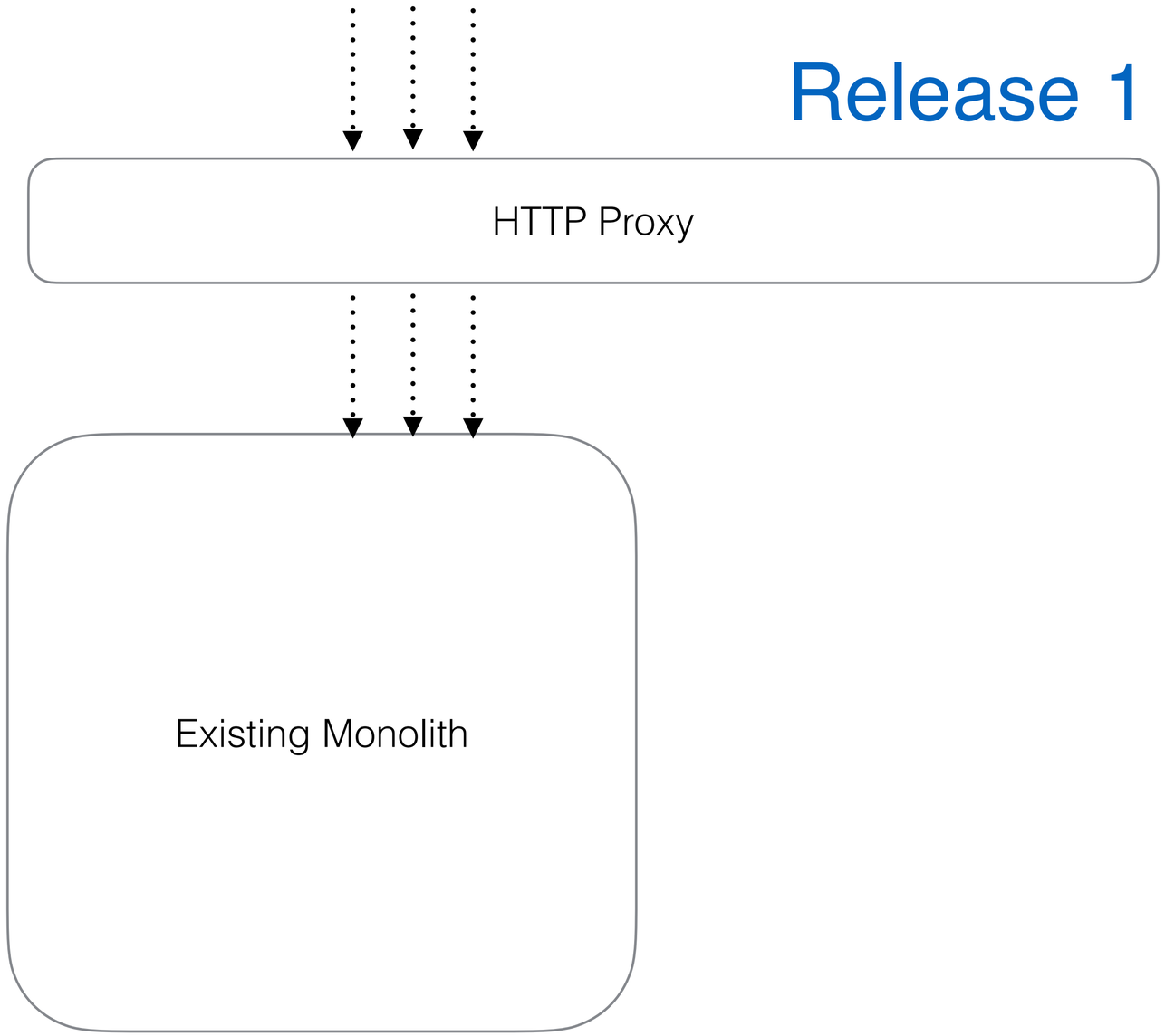
INTERCEPTION



INCREMENTAL ROLLOUT

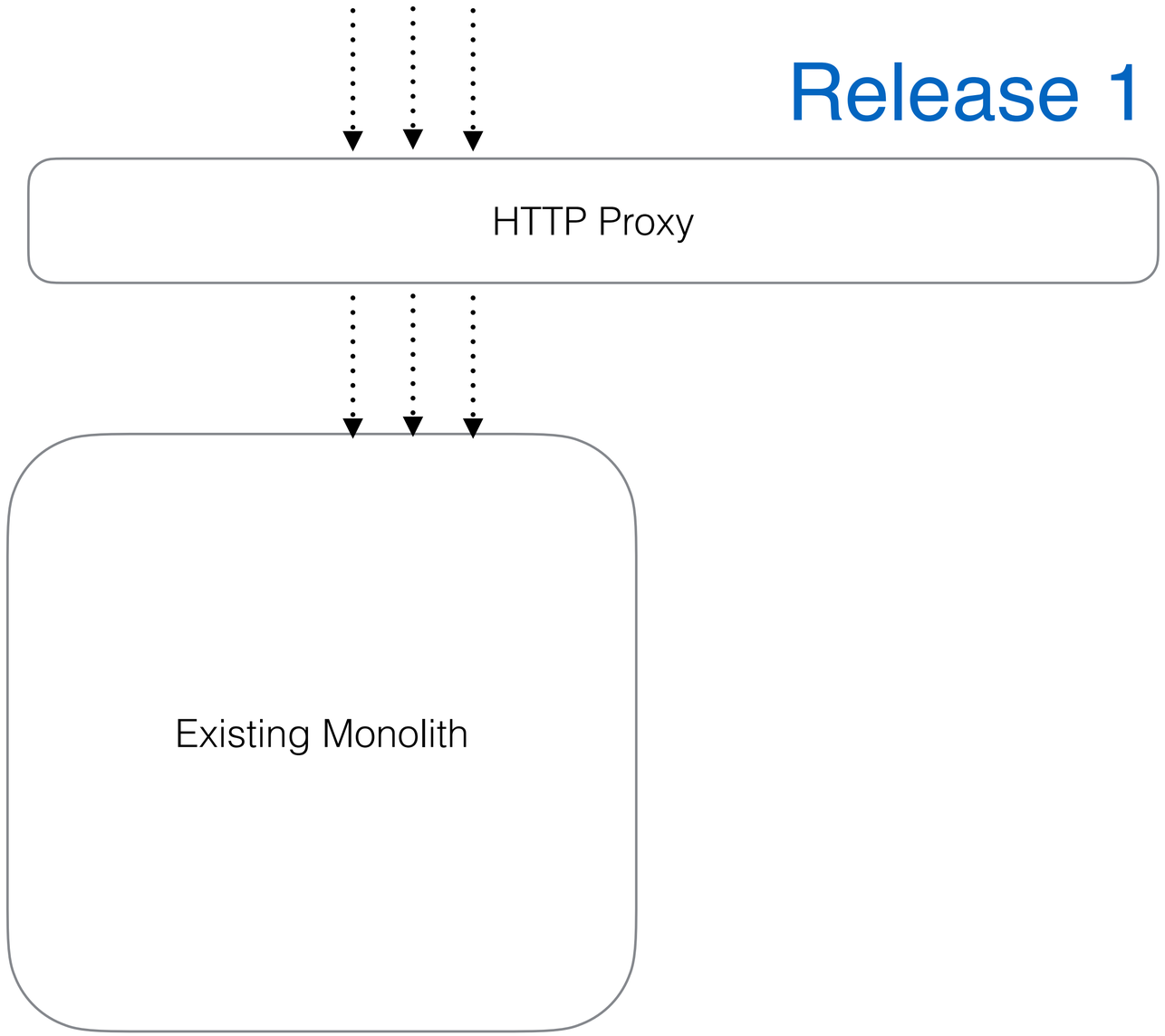


INCREMENTAL ROLLOUT



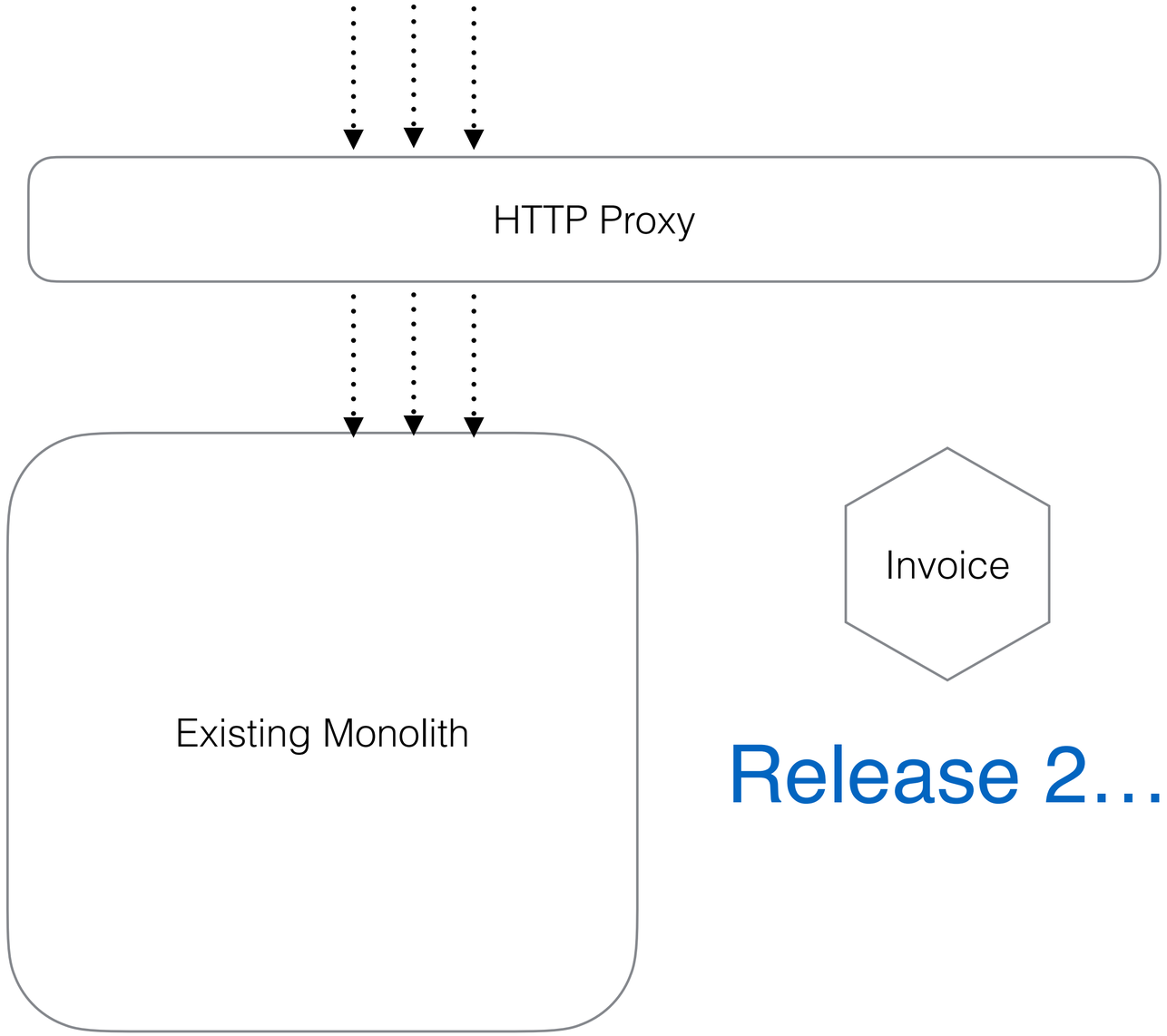
INCREMENTAL ROLLOUT

Assess impact of adding a network hop early



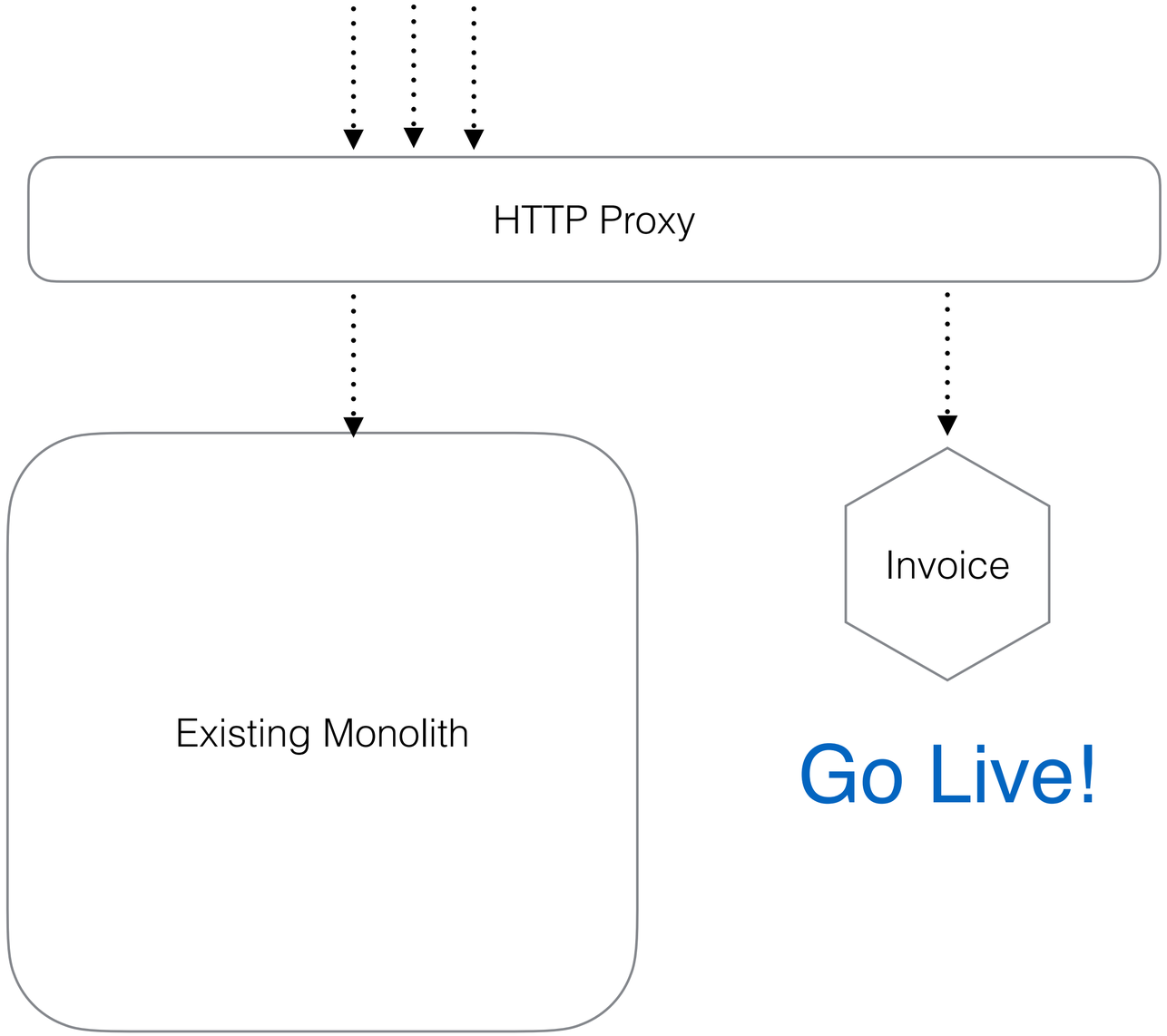
INCREMENTAL ROLLOUT

Assess impact of adding a network hop early

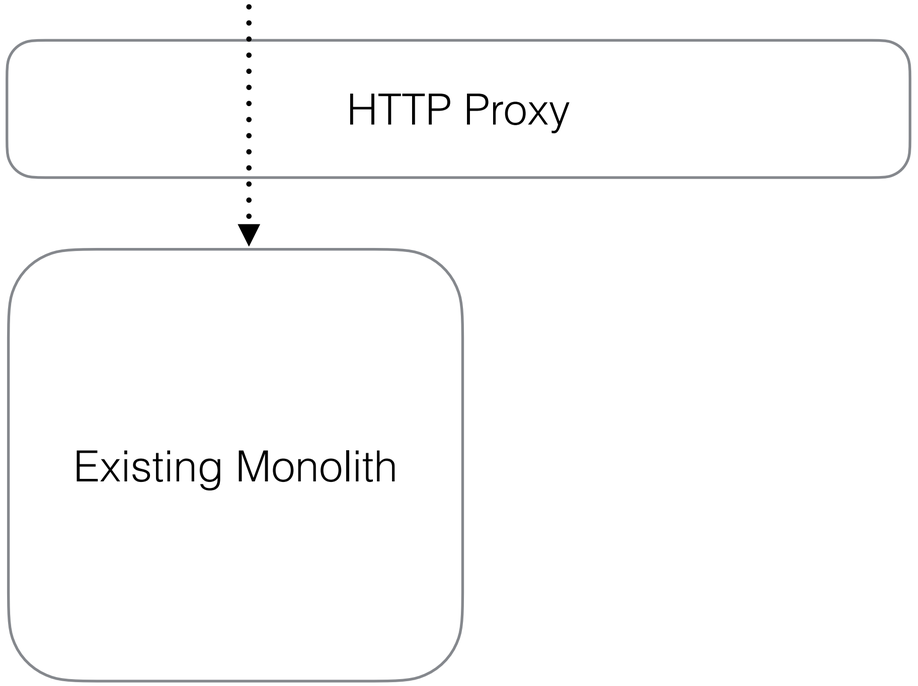


INCREMENTAL ROLLOUT

Assess impact of adding a network hop early

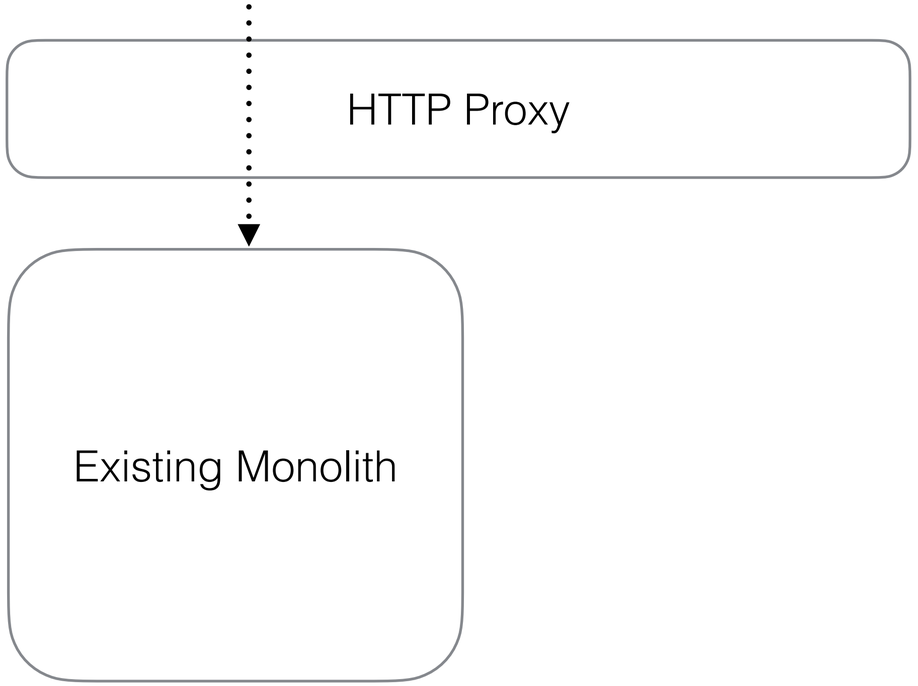


DEPLOYING WORK THAT ISN'T READY?



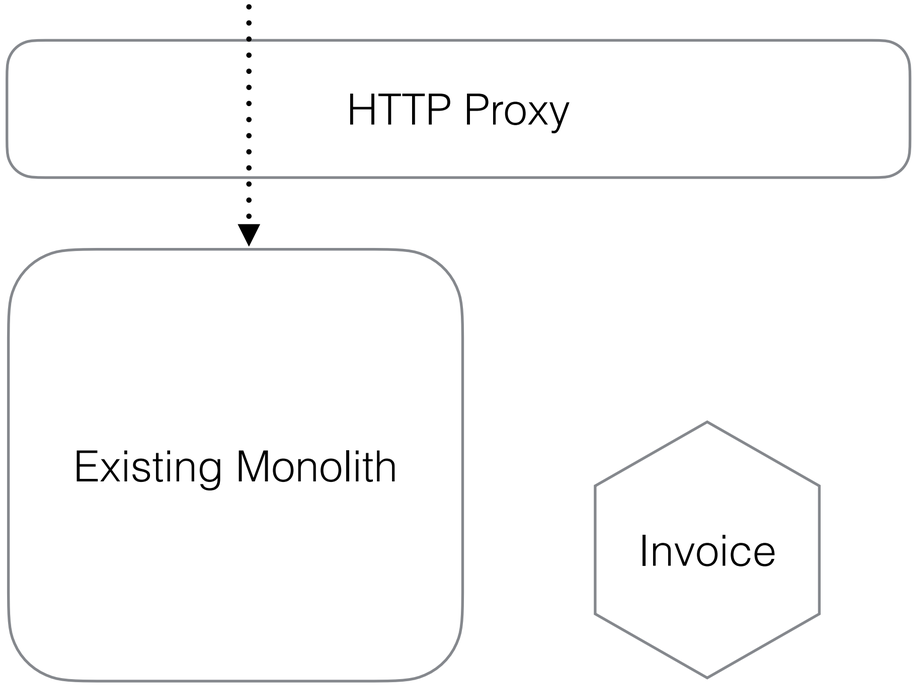
DEPLOYING WORK THAT ISN'T READY?

Deploying software doesn't need to be conflated with "release"



DEPLOYING WORK THAT ISN'T READY?

Deploying software doesn't need to be conflated with "release"

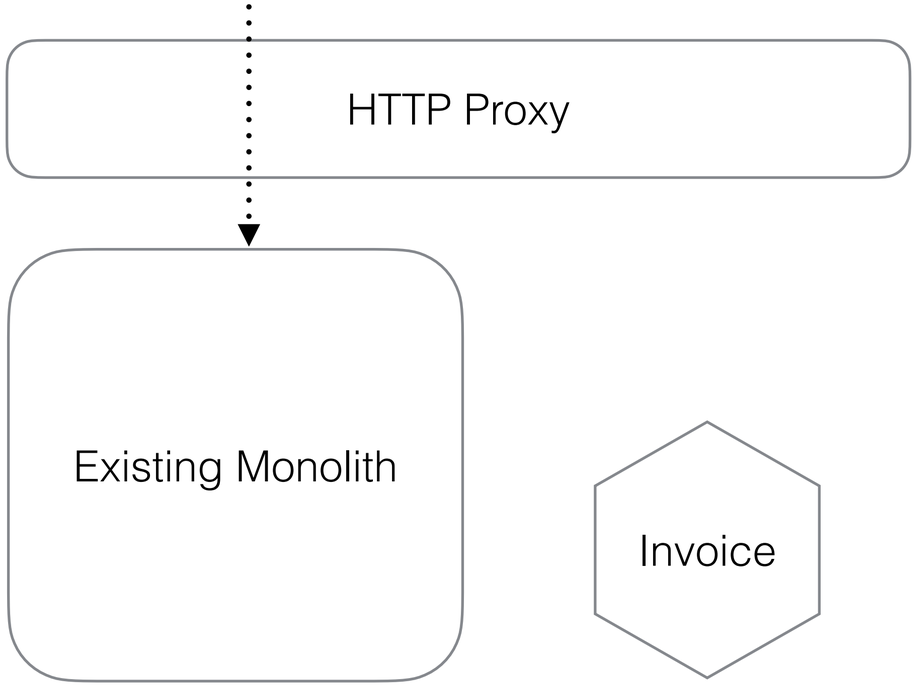


Invoice microservice is deployed

DEPLOYING WORK THAT ISN'T READY?

Deploying software doesn't need to be conflated with "release"

Separating these two steps allows you to do things like validate the newly deployed software prior to release

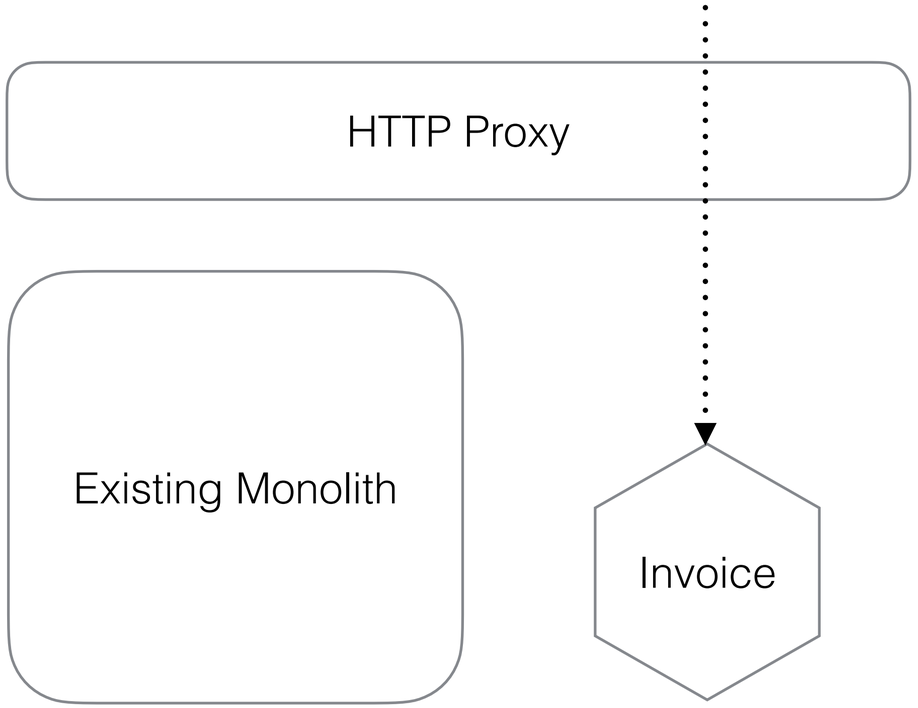


Invoice microservice is deployed

DEPLOYING WORK THAT ISN'T READY?

Deploying software doesn't need to be conflated with "release"

Separating these two steps allows you to do things like validate the newly deployed software prior to release



Now the new Invoice microservice is released

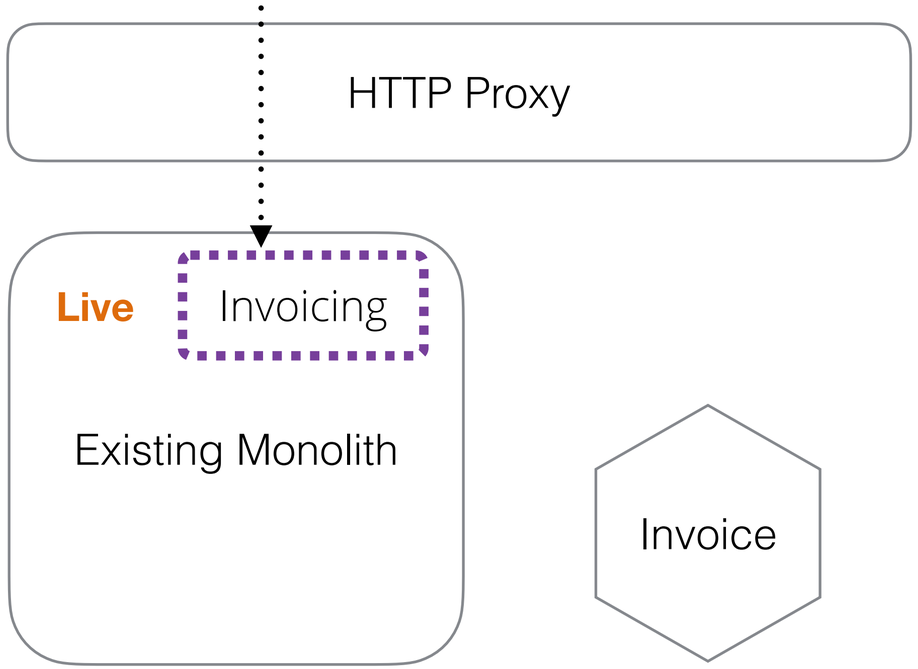
DON’T REMOVE THE OLD FUNCTIONALITY TOO SOON!

DON’T REMOVE THE OLD FUNCTIONALITY TOO SOON!

If we leave the old invoicing functionality in the monolith, we have an easy rollback

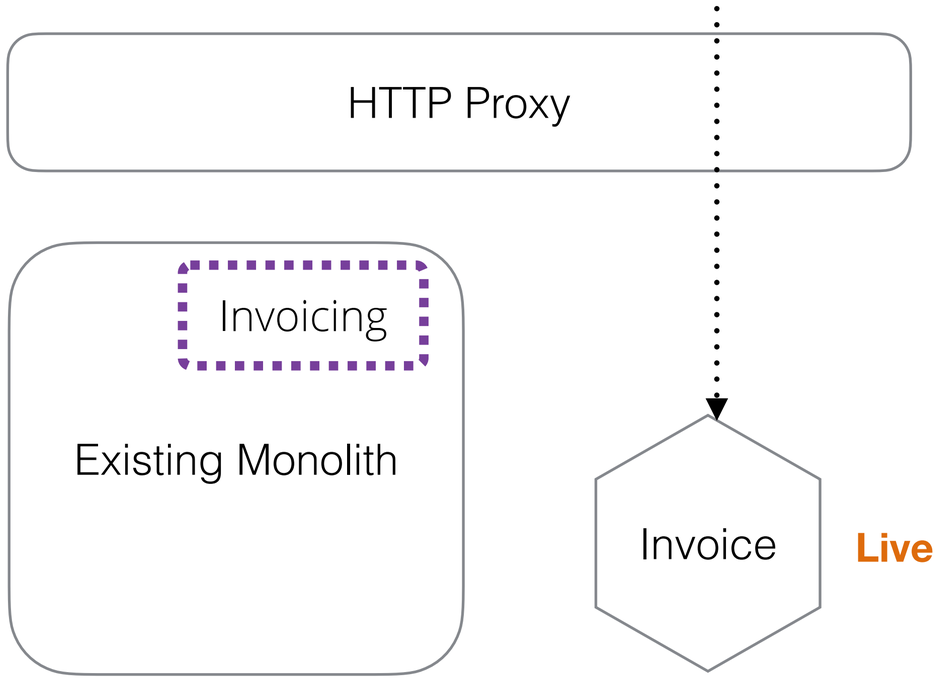
DON’T REMOVE THE OLD FUNCTIONALITY TOO SOON!

If we leave the old invoicing functionality in the monolith, we have an easy rollback



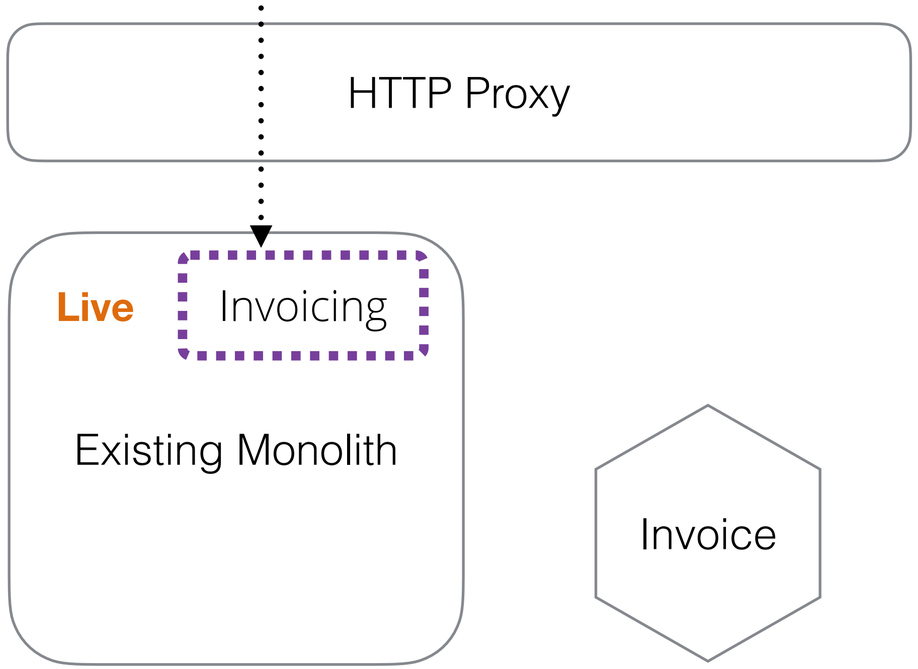
DON’T REMOVE THE OLD FUNCTIONALITY TOO SOON!

If we leave the old invoicing functionality in the monolith, we have an easy rollback



DON’T REMOVE THE OLD FUNCTIONALITY TOO SOON!

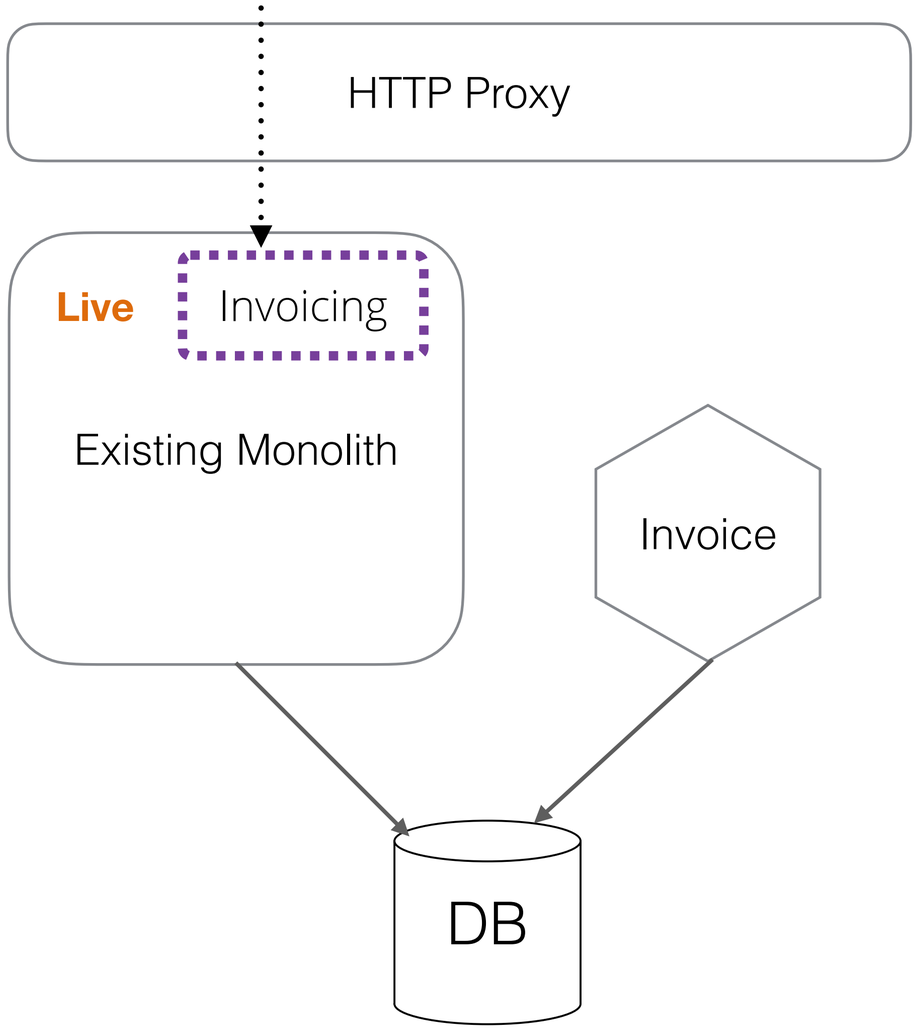
If we leave the old invoicing functionality in the monolith, we have an easy rollback



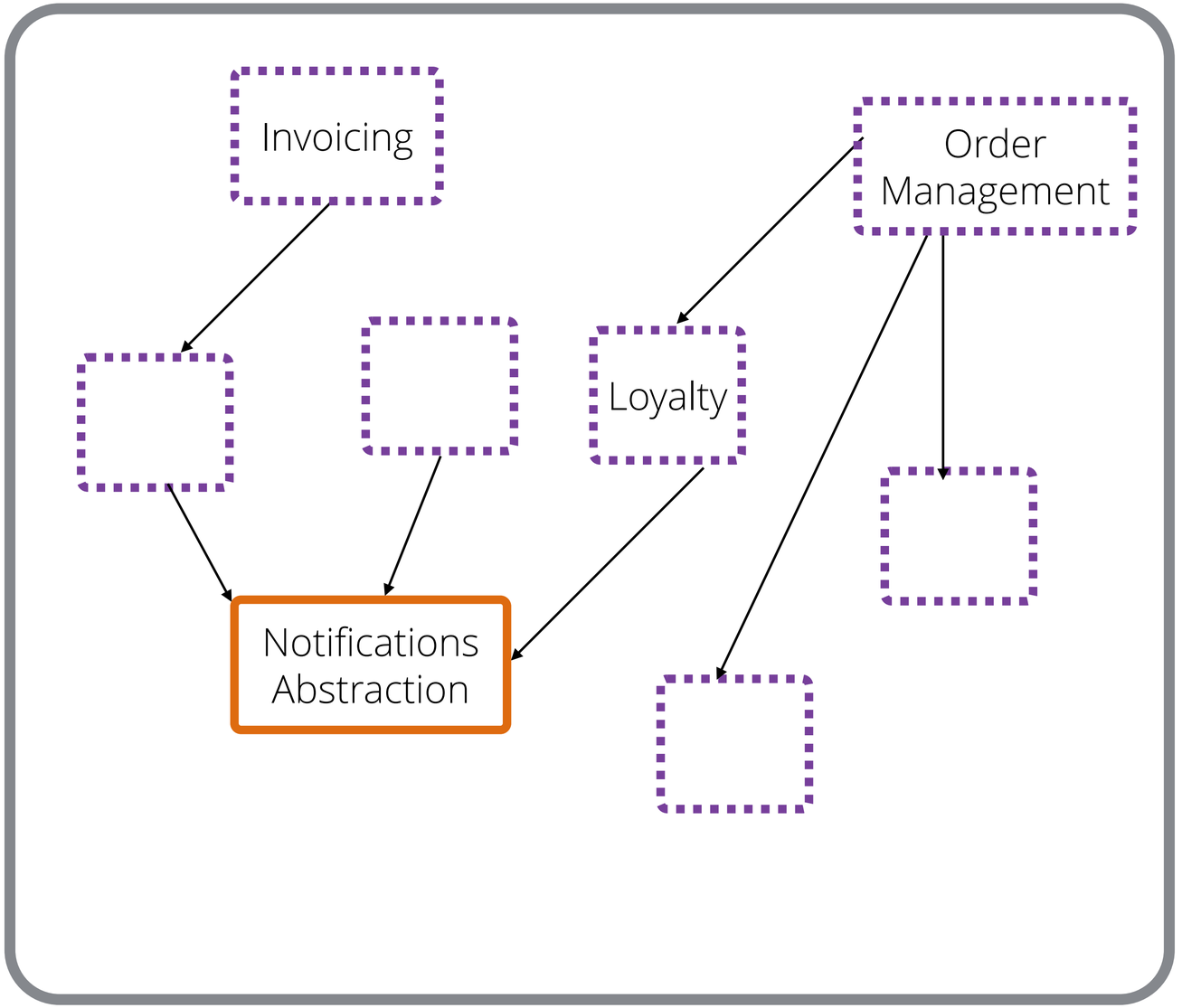
DON’T REMOVE THE OLD FUNCTIONALITY TOO SOON!

If we leave the old invoicing functionality in the monolith, we have an easy rollback

Would likely need to use the same DB during this transition

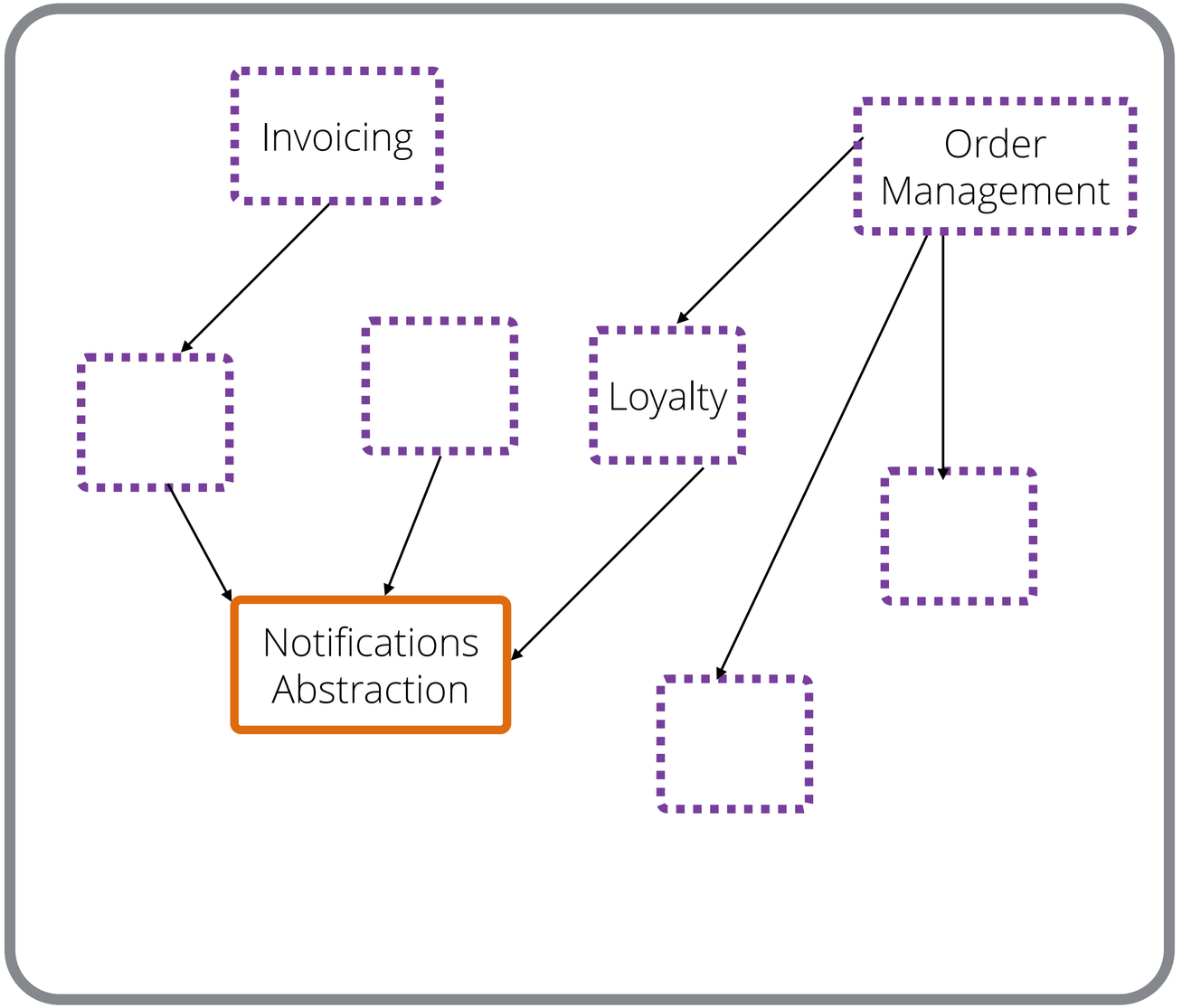


BRANCH BY ABSTRACTION



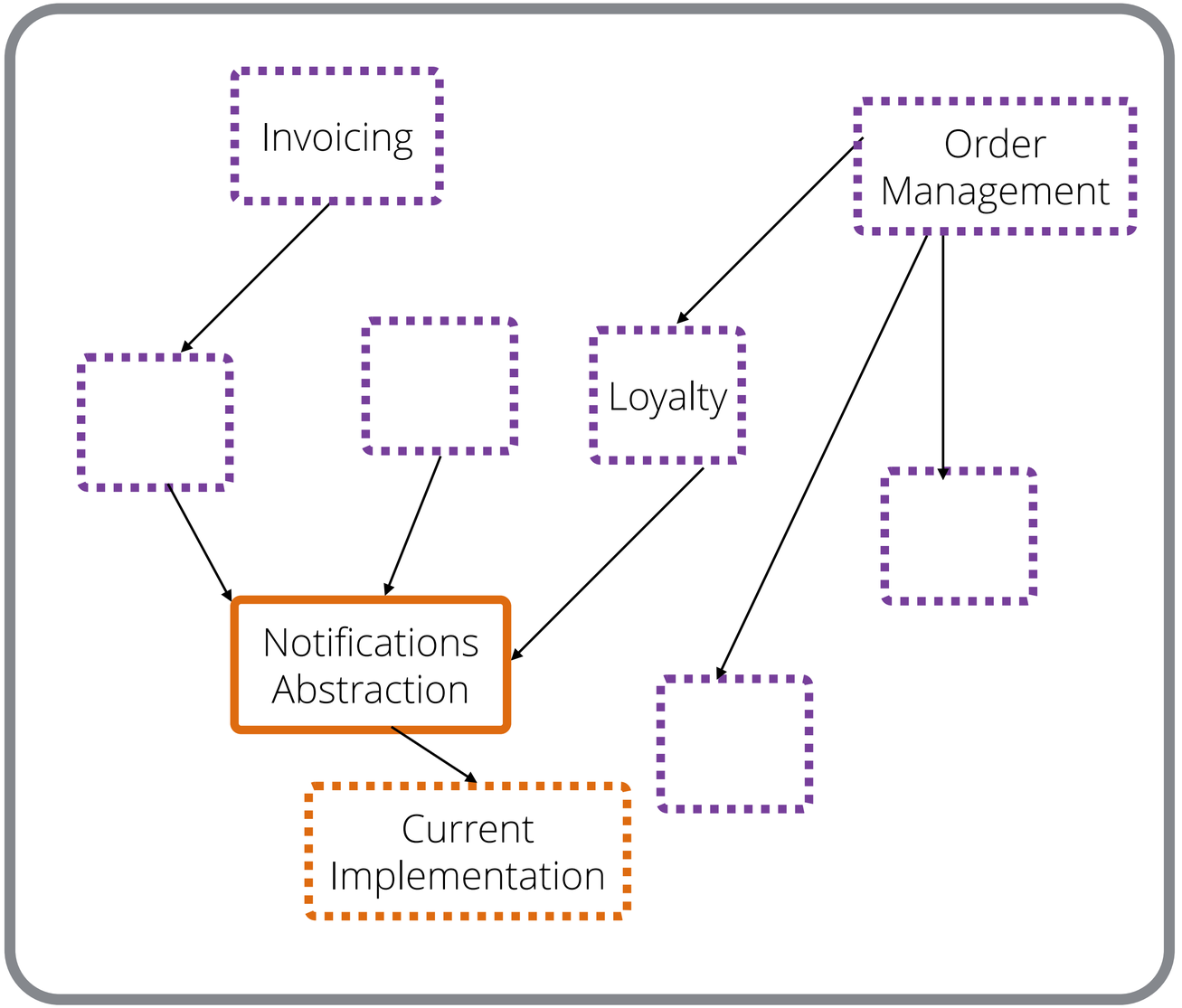
BRANCH BY ABSTRACTION

Gives you an abstraction point which can be used to “hide” ongoing development



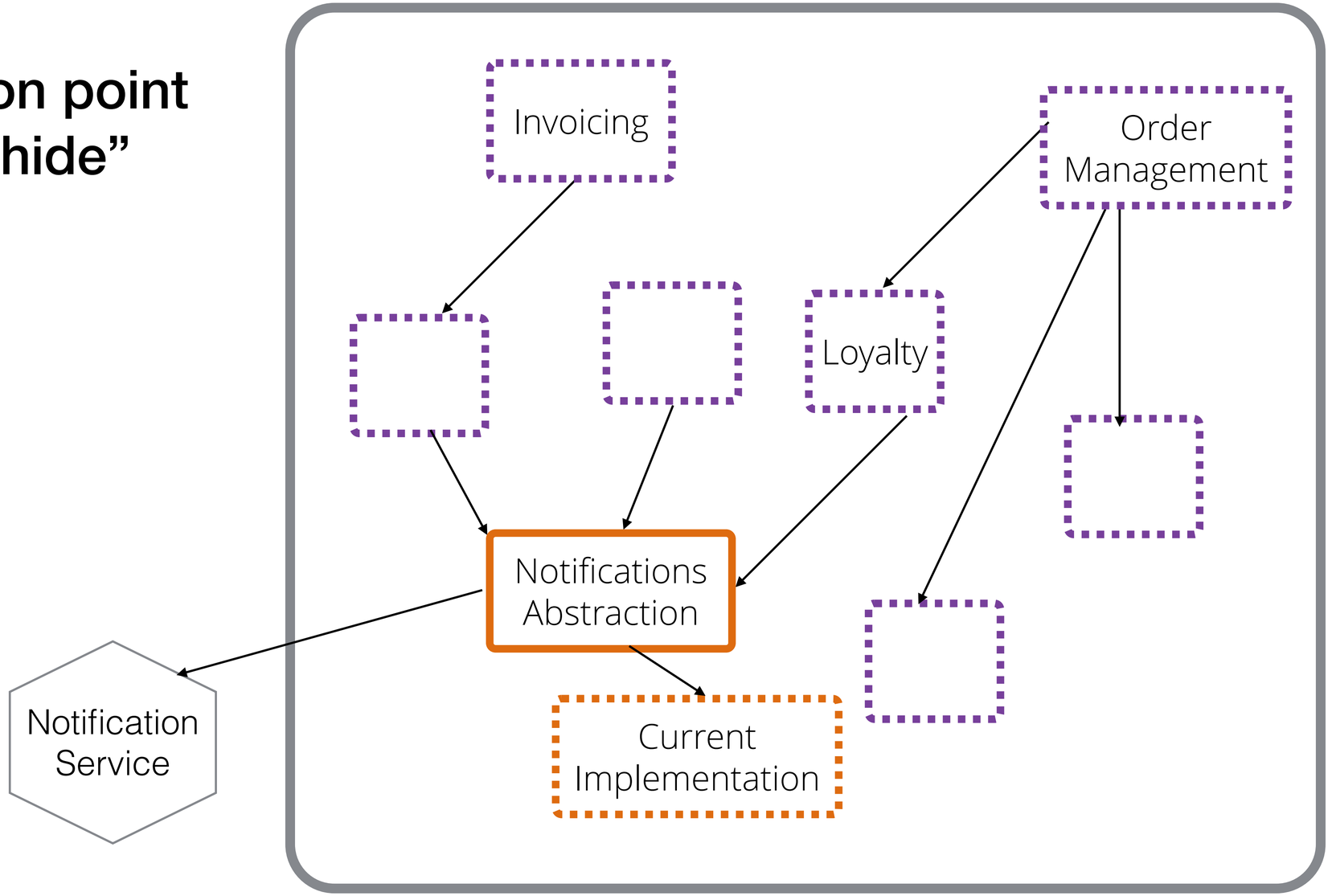
BRANCH BY ABSTRACTION

Gives you an abstraction point which can be used to “hide” ongoing development



BRANCH BY ABSTRACTION

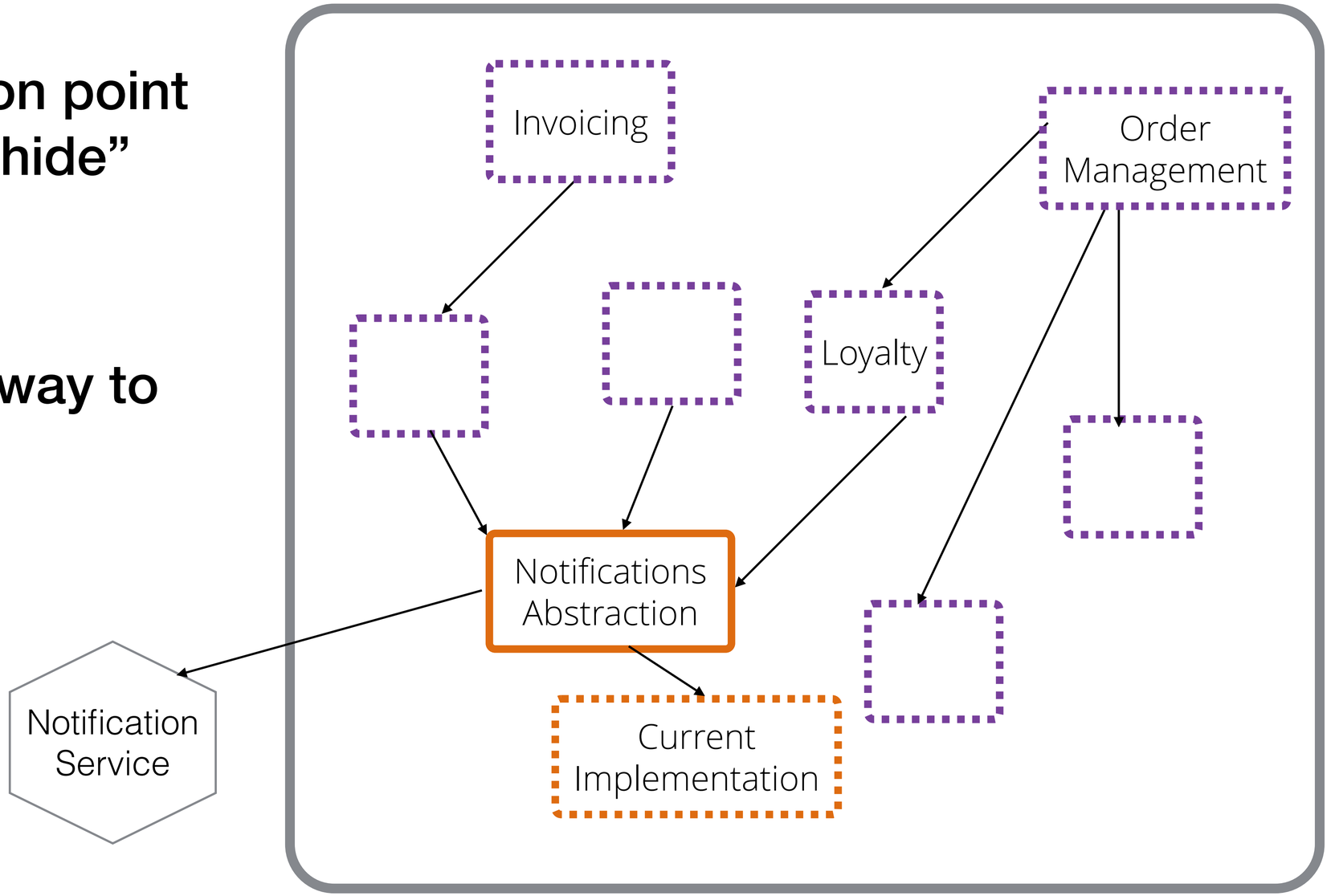
Gives you an abstraction point which can be used to “hide” ongoing development



BRANCH BY ABSTRACTION

Gives you an abstraction point which can be used to “hide” ongoing development

Can also be used as a way to toggle between implementations



WHY NOT BRANCH IN SOURCE CODE?

WHY NOT BRANCH IN SOURCE CODE?

Requires more branches

WHY NOT BRANCH IN SOURCE CODE?

Requires more branches

Can’t switch between implementations without a redeploy

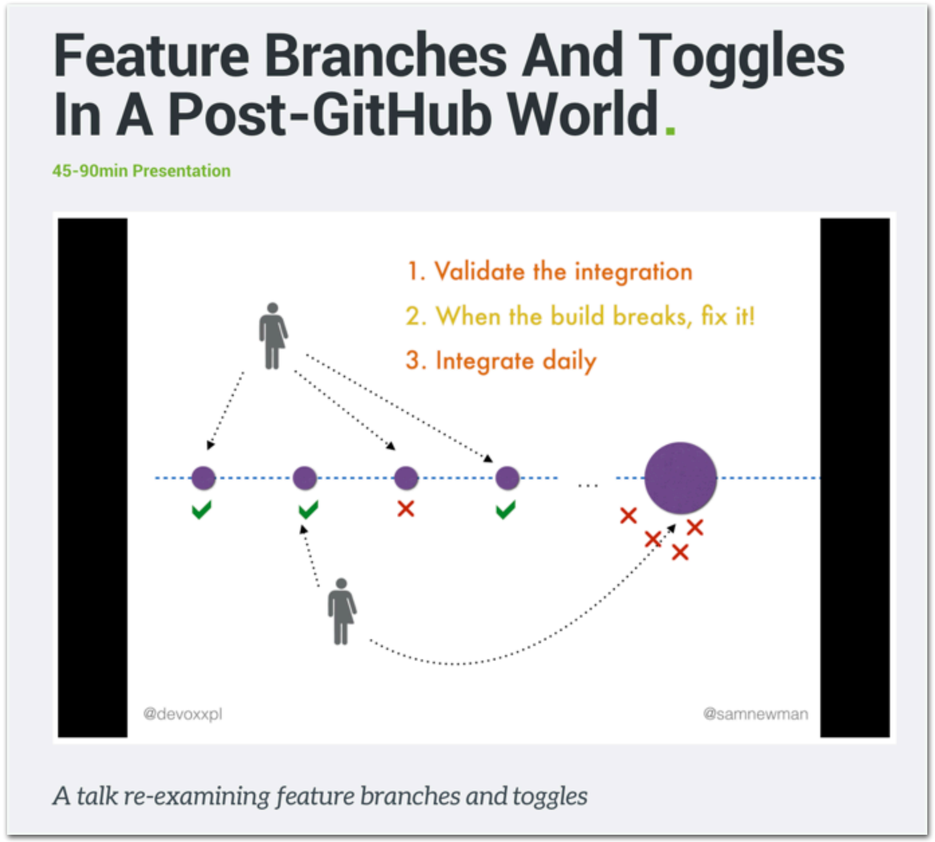
WHY NOT BRANCH IN SOURCE CODE?

Requires more branches

Can’t switch between implementations without a redeploy

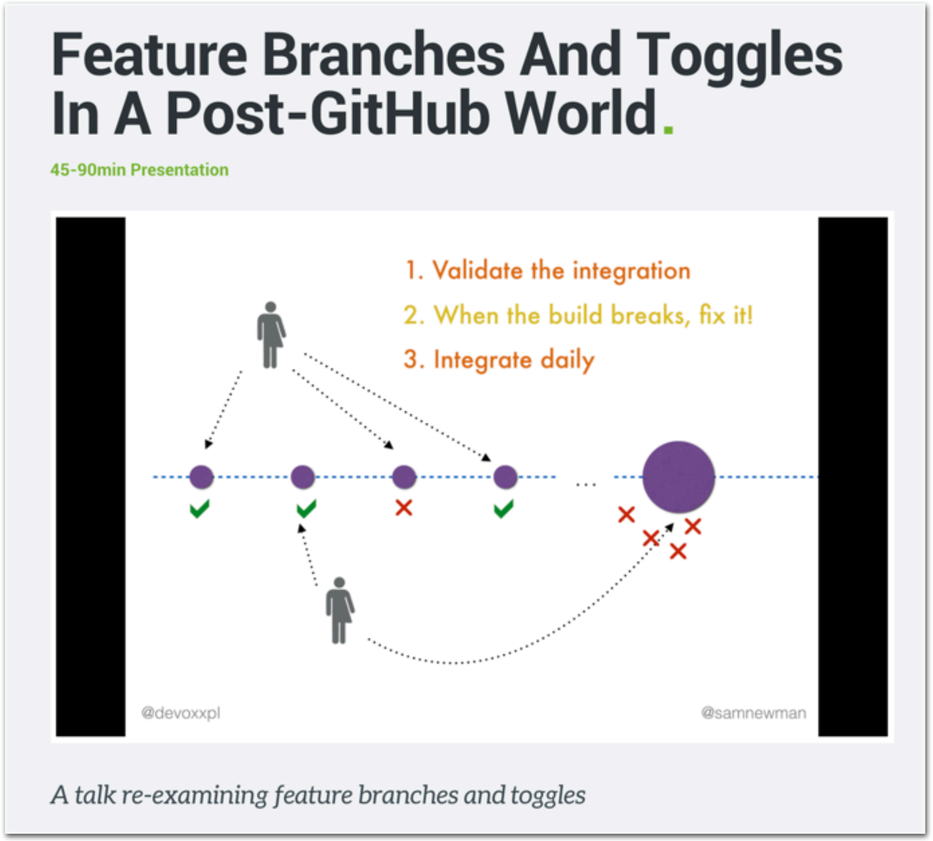
Can’t integrate and test the changes in a timely fashion

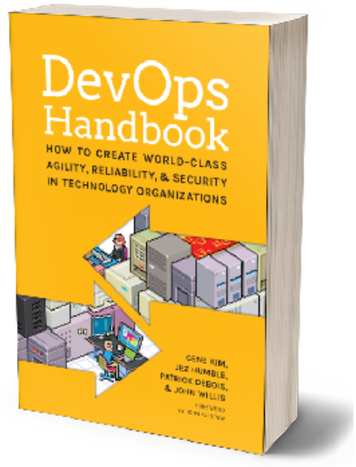
AVOID LONG-LIVED SOURCE CODE BRANCHING



https://samnewman.io/talks/branching-and-feature-toggles/

AVOID LONG-LIVED SOURCE CODE BRANCHING

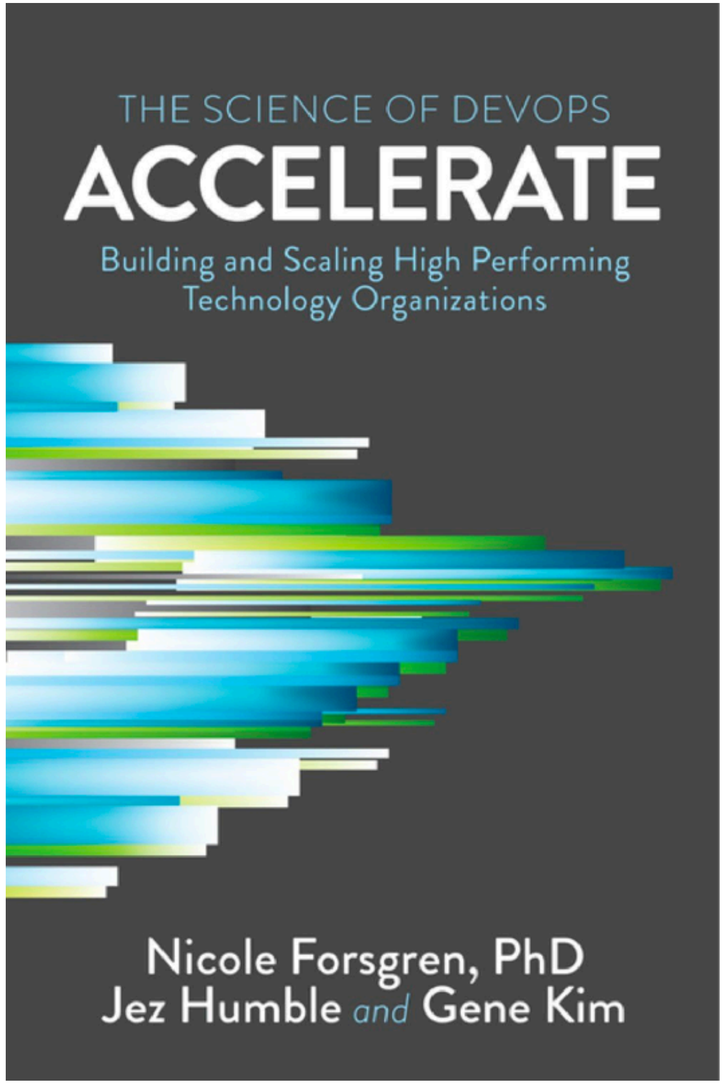




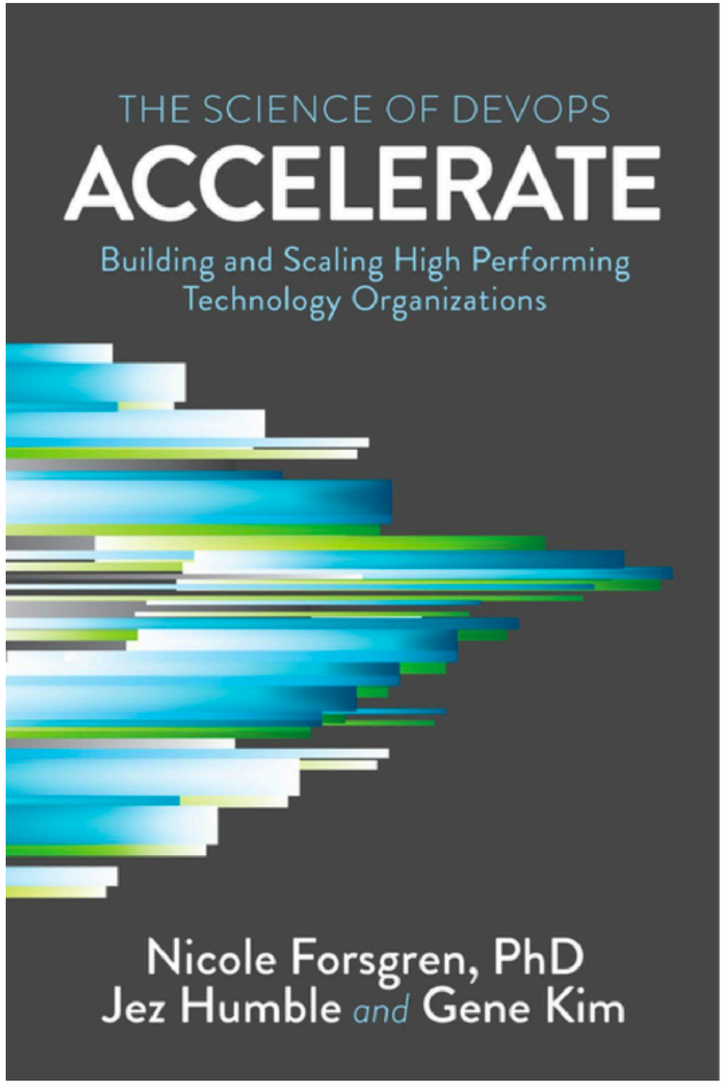
"…trunk-based development predicts higher throughput and better stability, and even higher job satisfaction and lower rates of burnout."

https://samnewman.io/talks/branching-and-feature-toggles/

ACCELERATE (READ THIS BOOK)

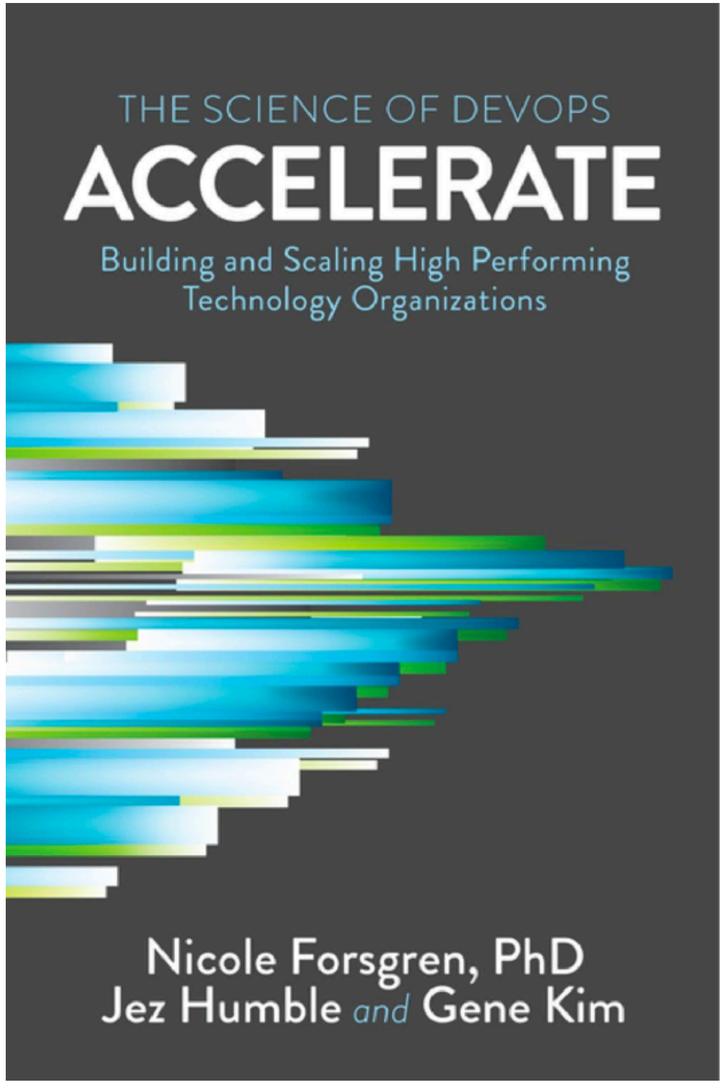


ACCELERATE (READ THIS BOOK)



"Our research also found that developing off trunk/master rather than on long-lived feature branches was correlated with higher delivery performance."

ACCELERATE (READ THIS BOOK)

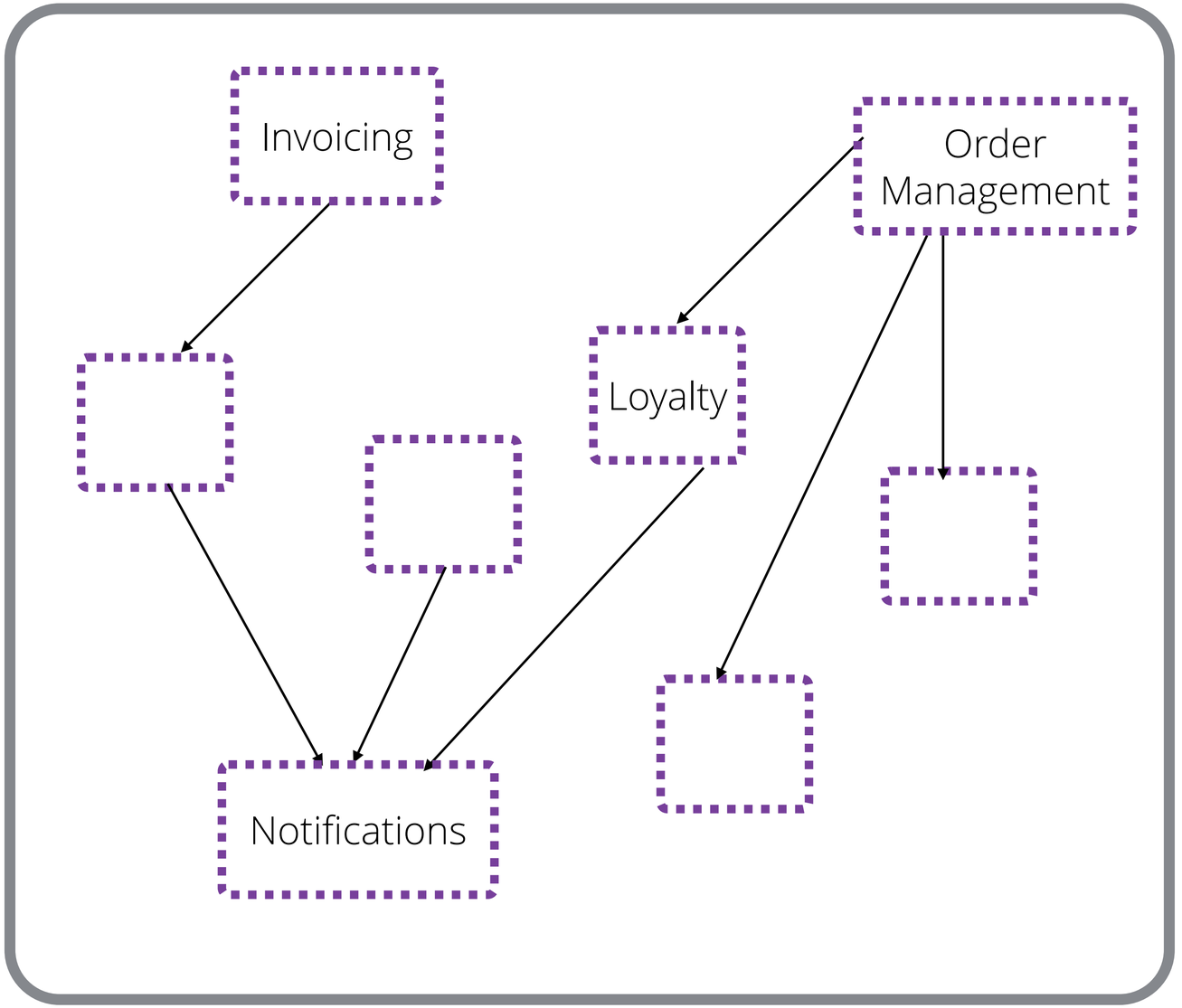


"Our research also found that developing off trunk/master rather than on long-lived feature branches was correlated with higher delivery performance."

"Teams that did well had fewer than three active branches at any time, their branches had very short lifetimes (less than a day) before being merged into trunk and never had "code freeze" or stabilization periods."

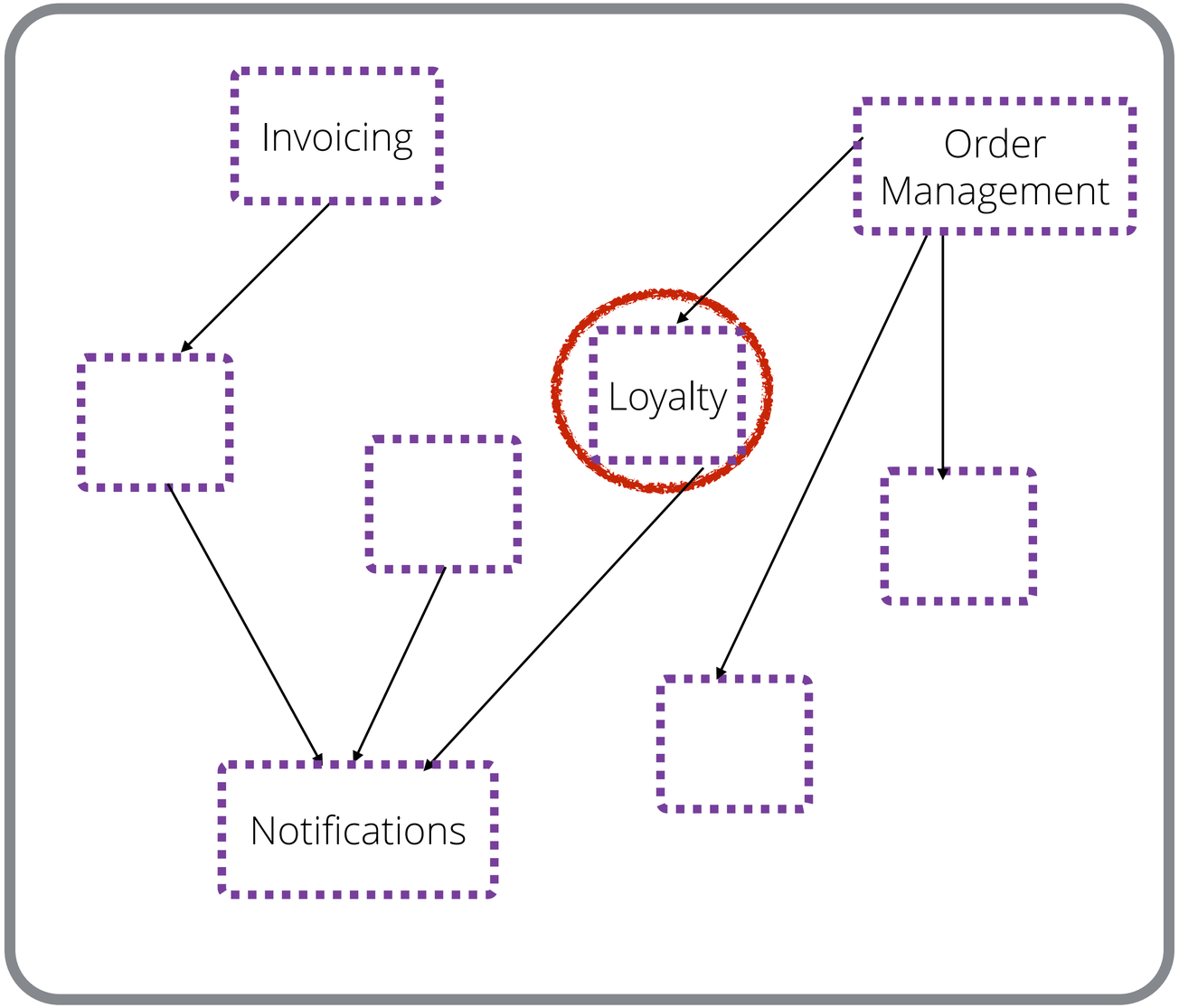
CANDIDATES FOR BRANCH BY ABSTRACTION

Can be used to replace functionality deeper inside the monolith



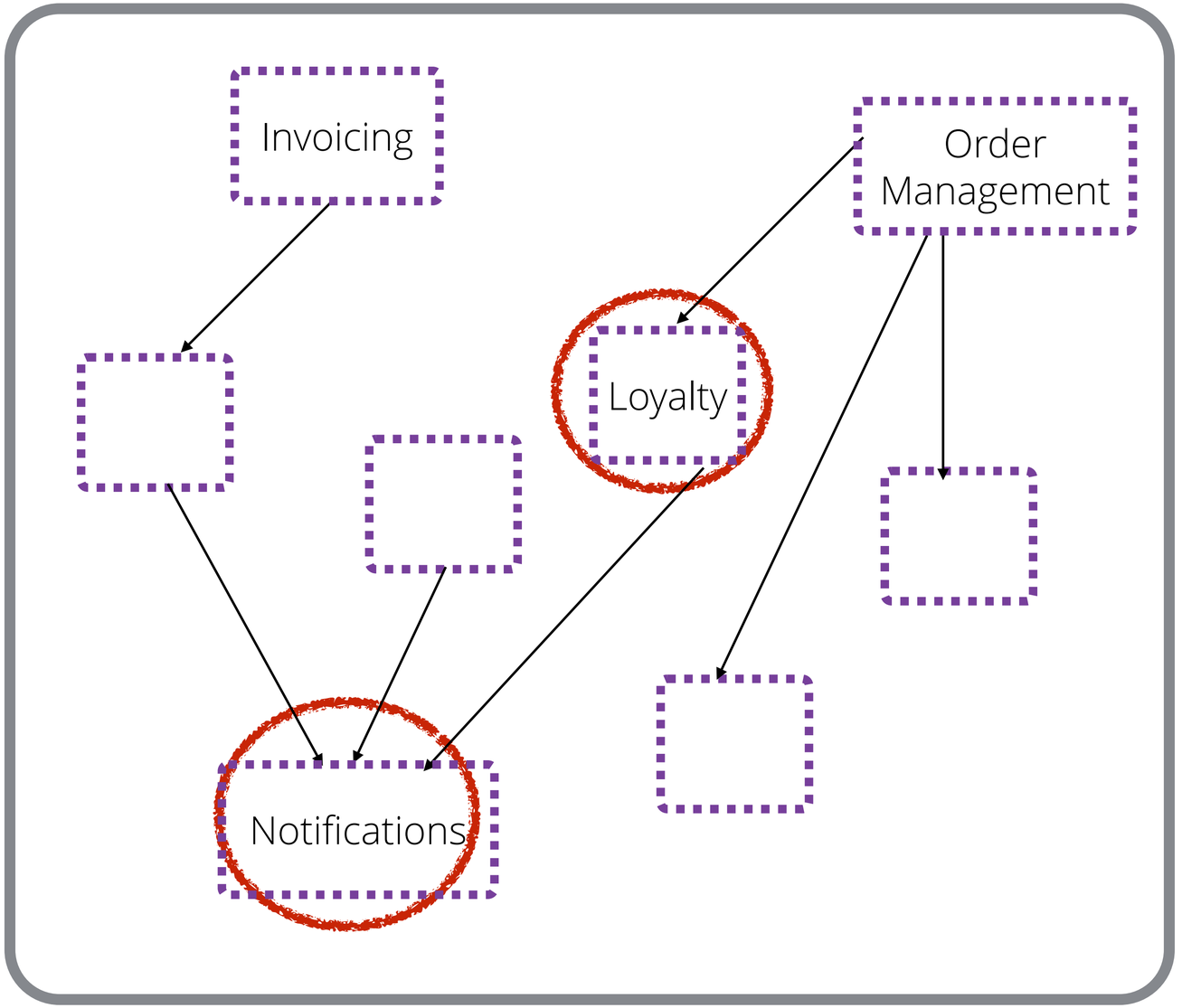
CANDIDATES FOR BRANCH BY ABSTRACTION

Can be used to replace functionality deeper inside the monolith



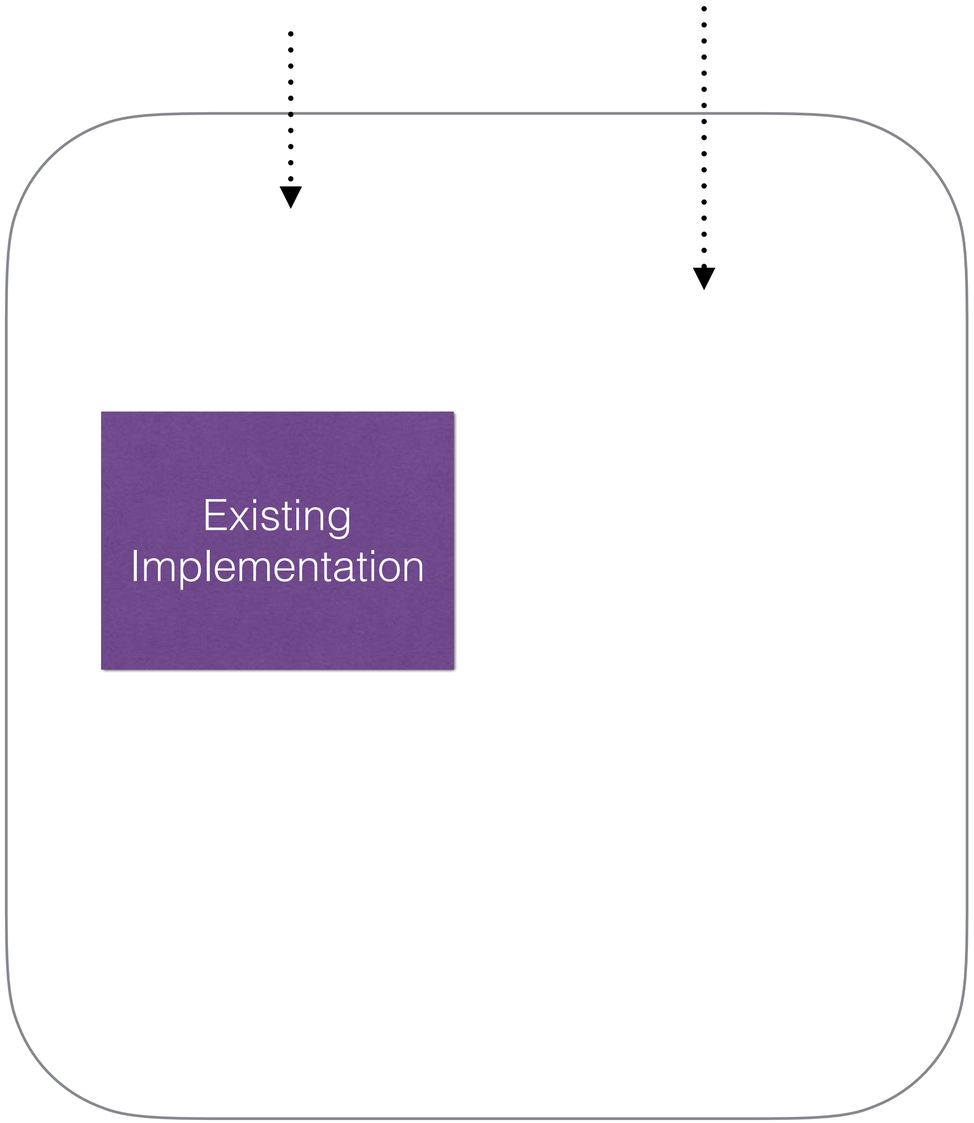
CANDIDATES FOR BRANCH BY ABSTRACTION

Can be used to replace functionality deeper inside the monolith

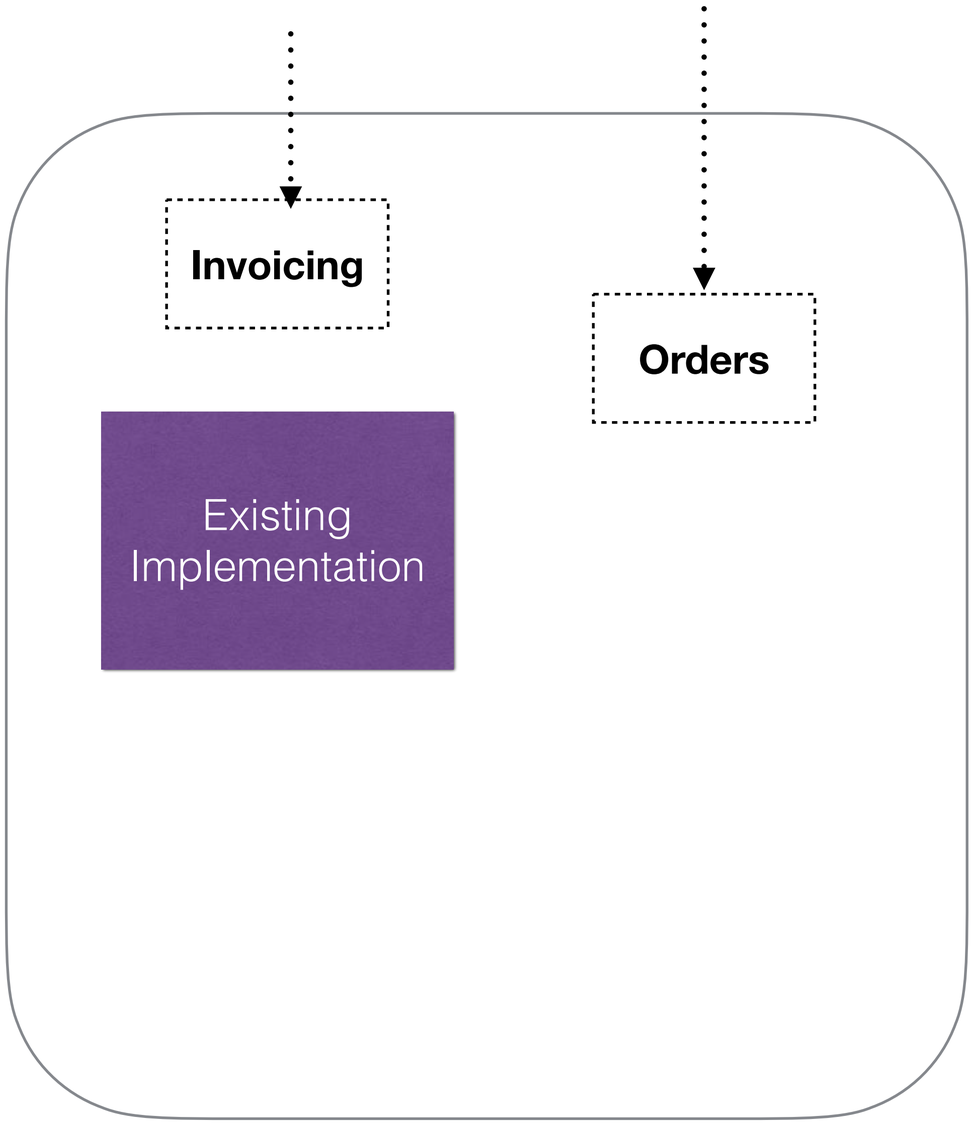


EXAMPLE

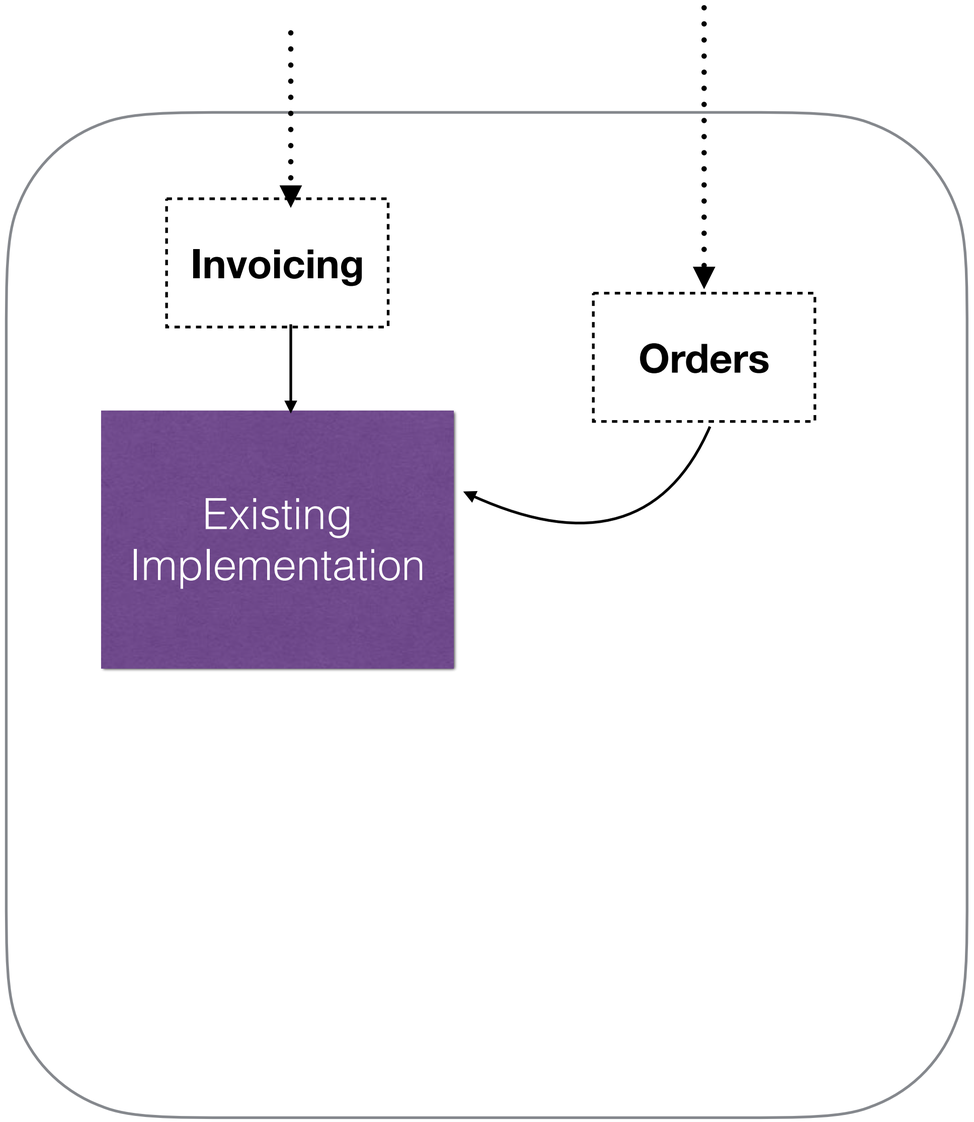
EXAMPLE



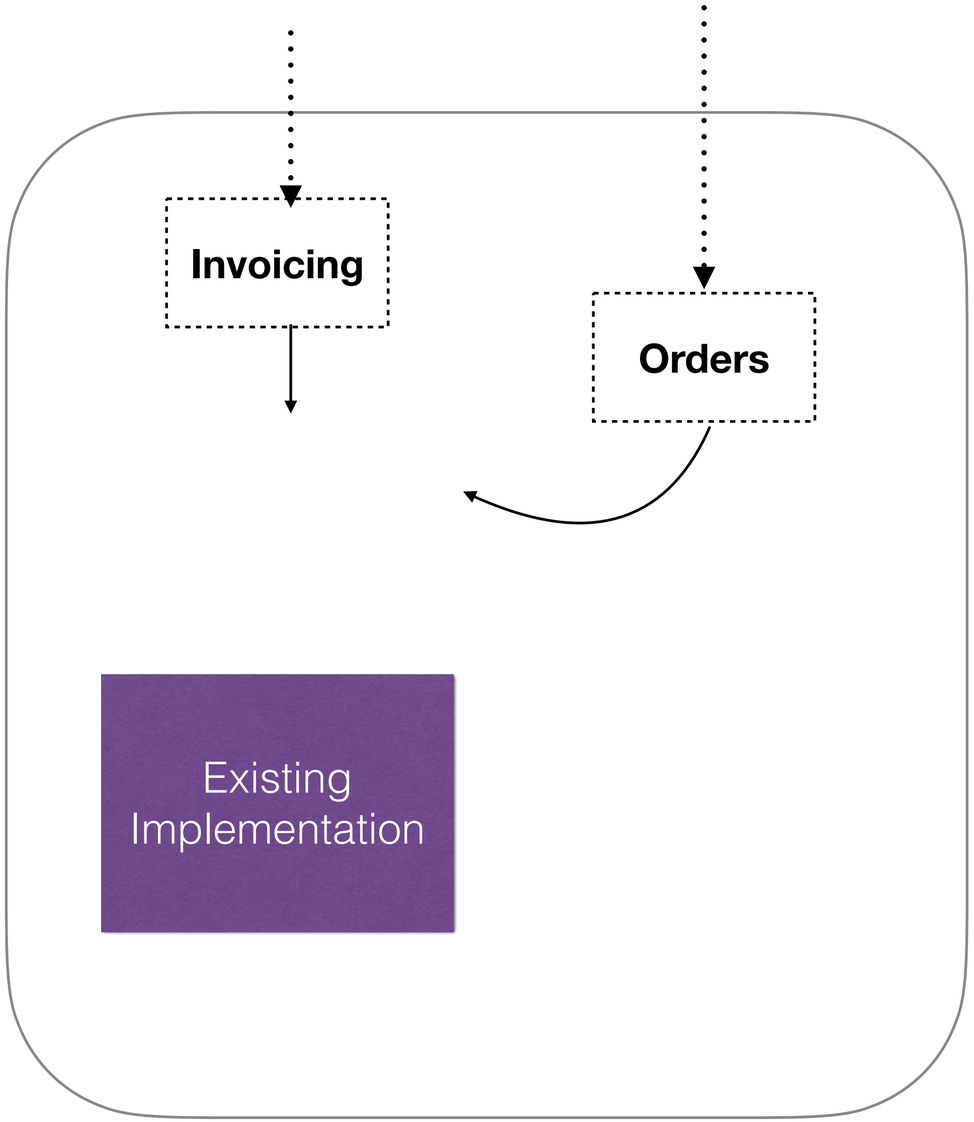
EXAMPLE



EXAMPLE

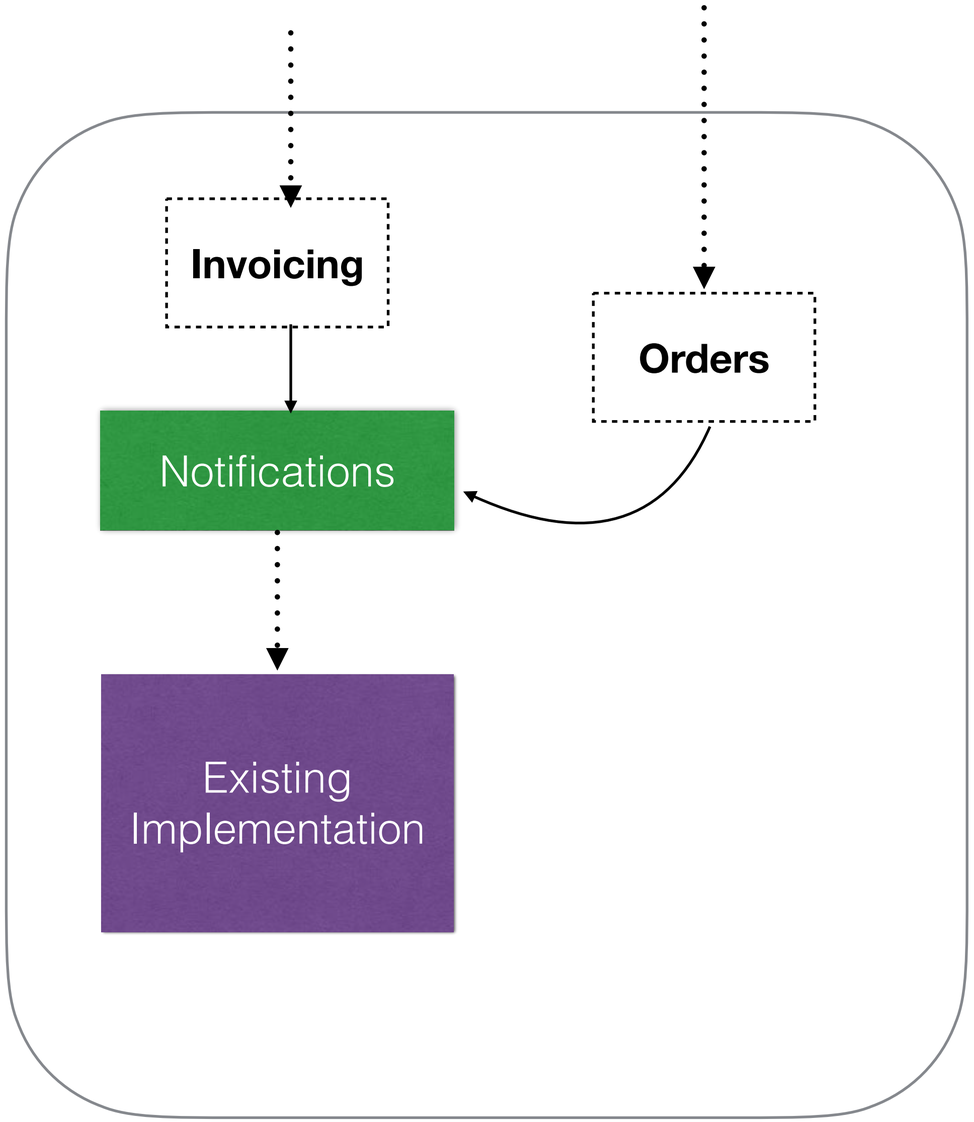


EXAMPLE

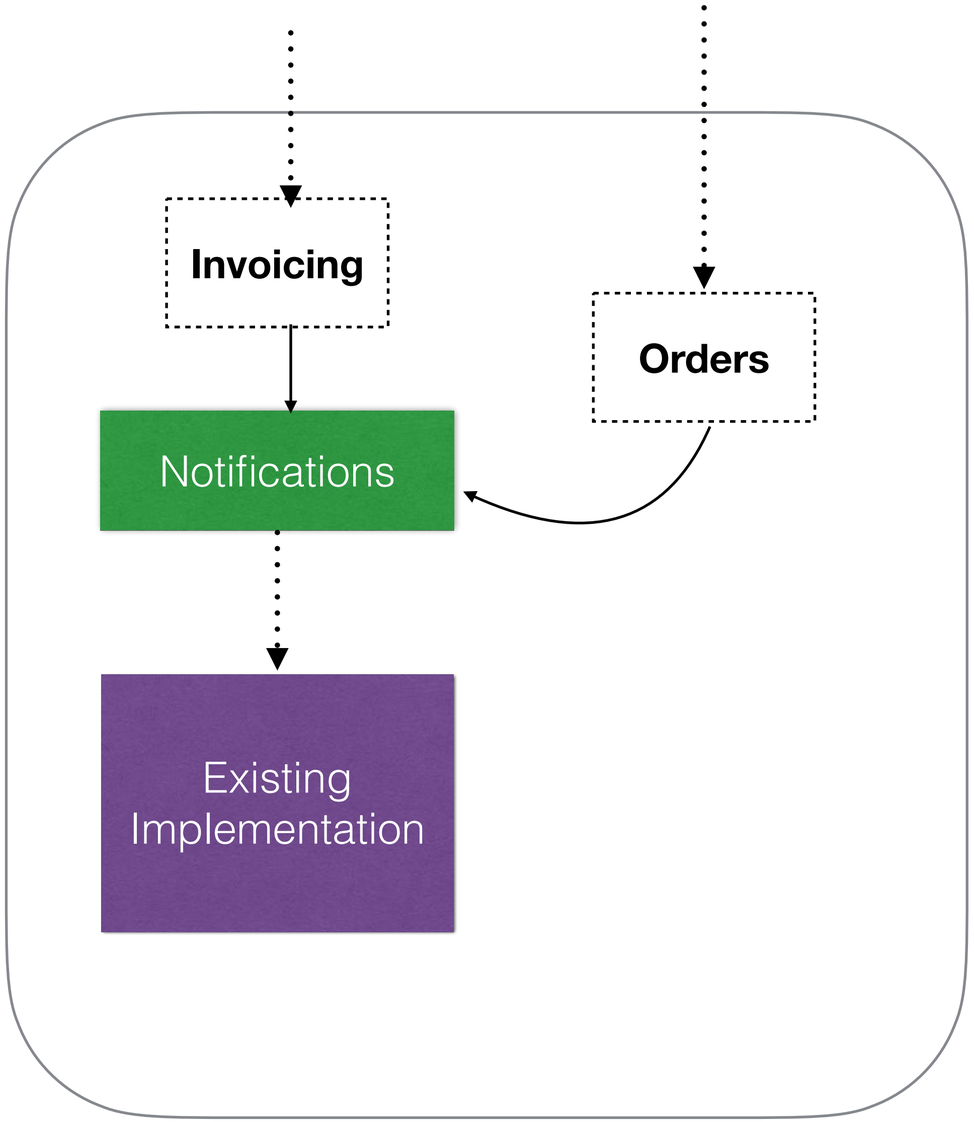


EXAMPLE

1. Create abstraction point



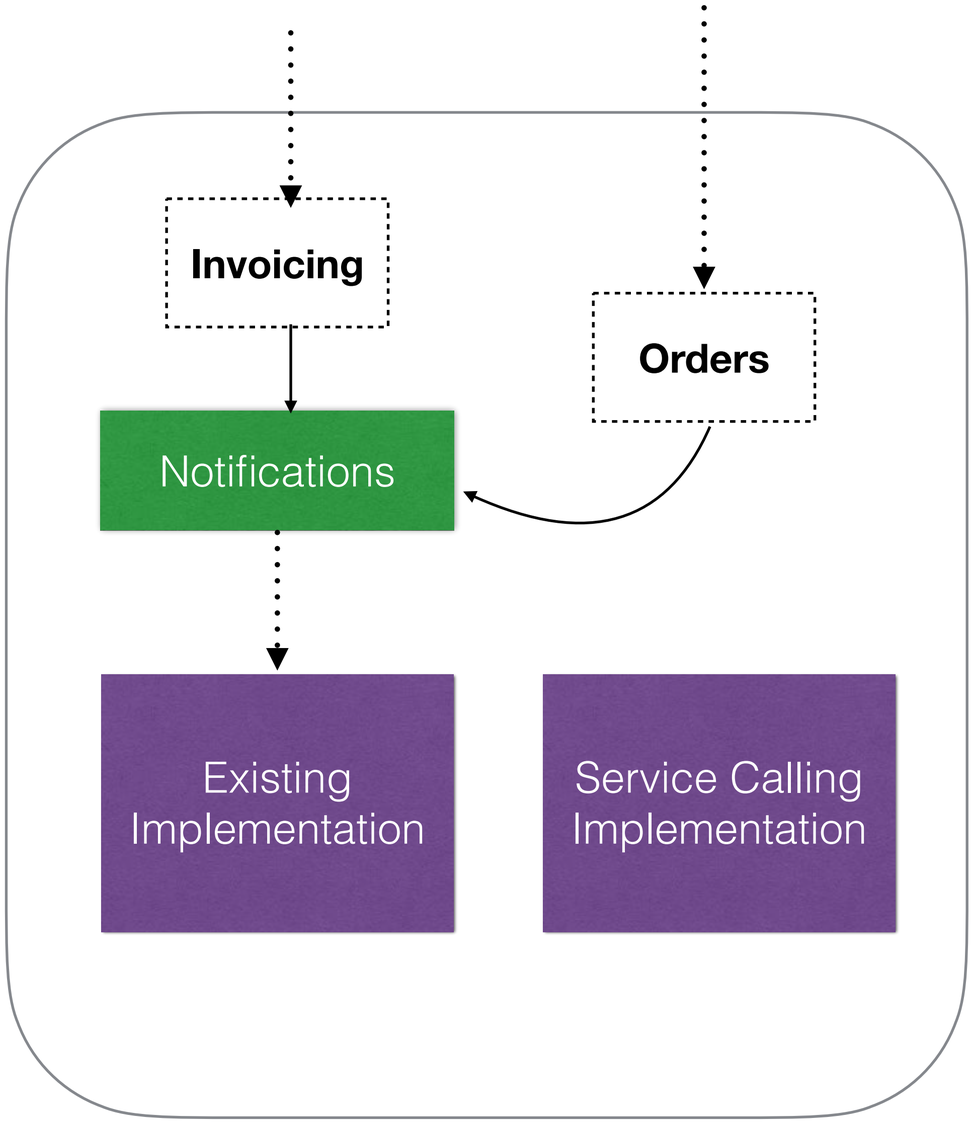
EXAMPLE



1. Create abstraction point

2. Start work on new service implementation

EXAMPLE



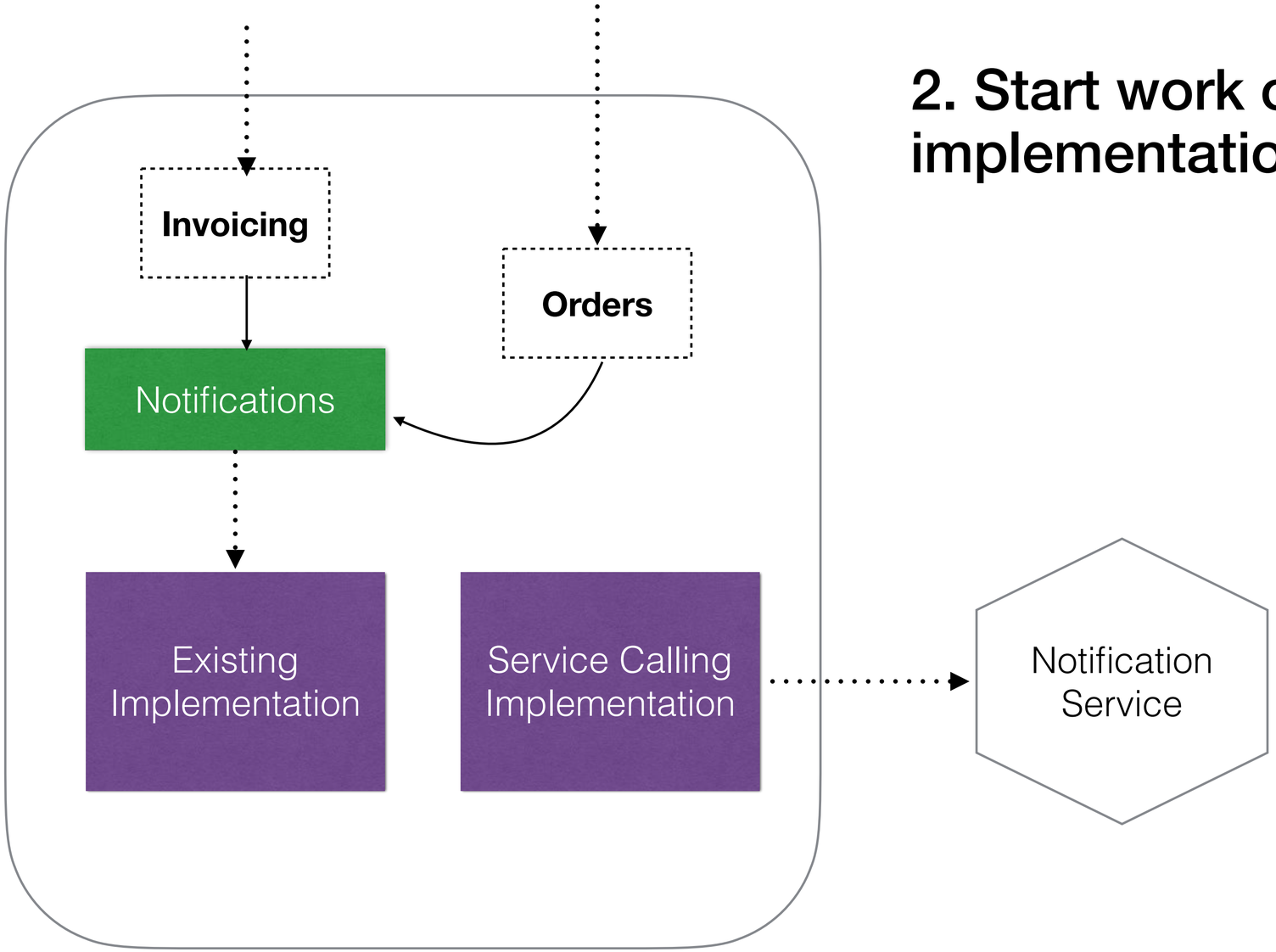
1. Create abstraction point

2. Start work on new service implementation

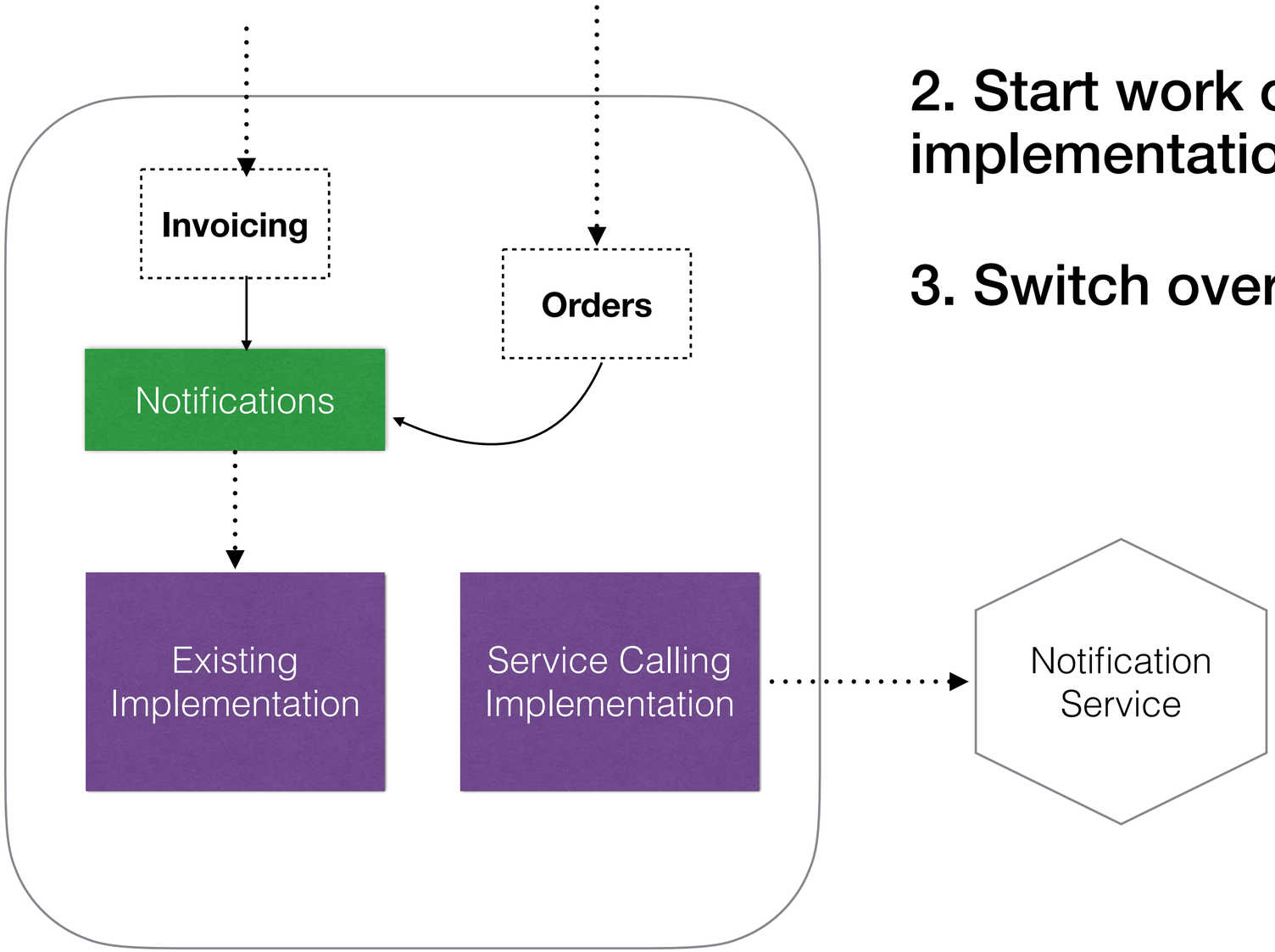
EXAMPLE

1. Create abstraction point

2. Start work on new service implementation



EXAMPLE

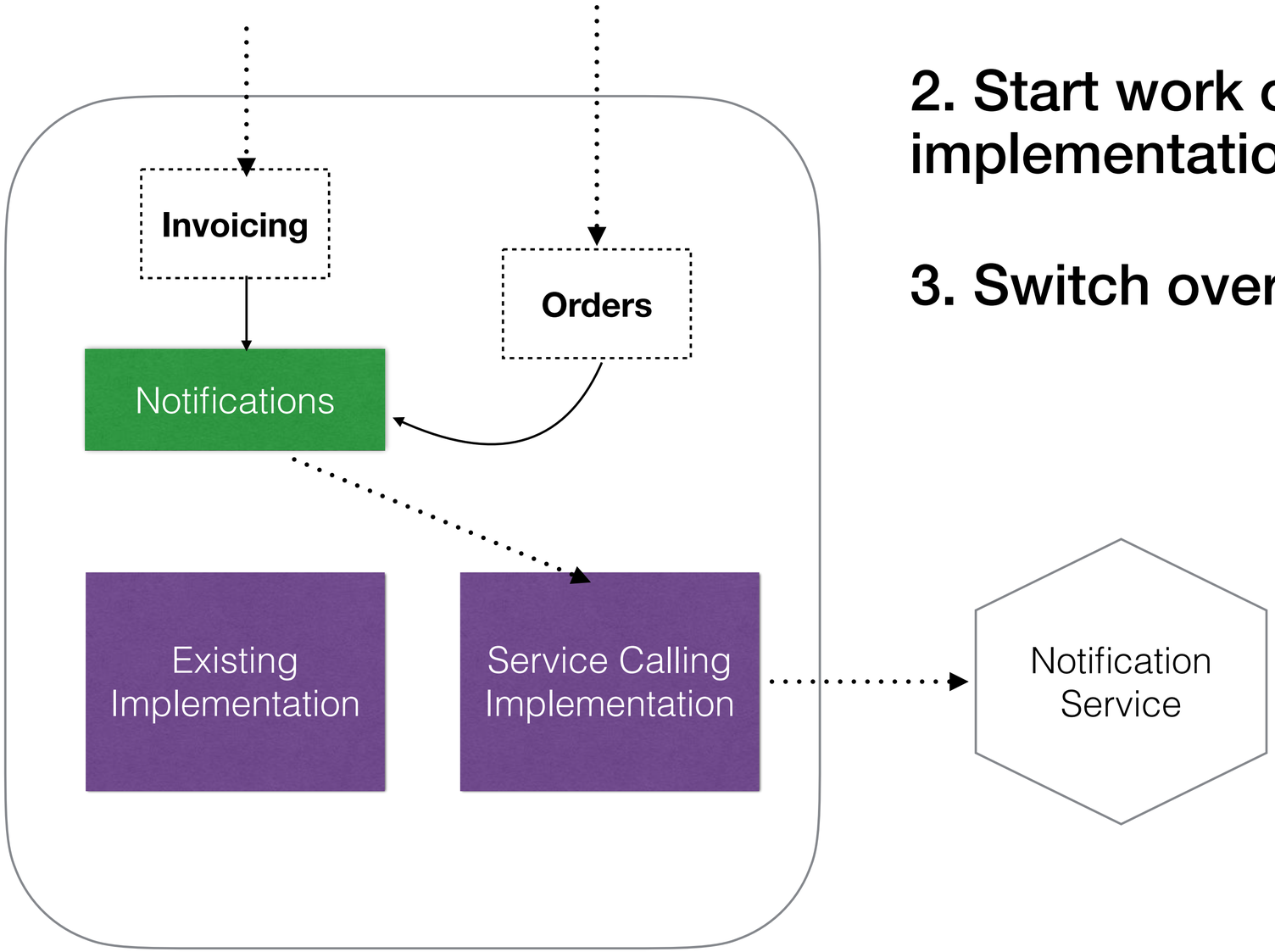


1. Create abstraction point

2. Start work on new service implementation

3. Switch over

EXAMPLE

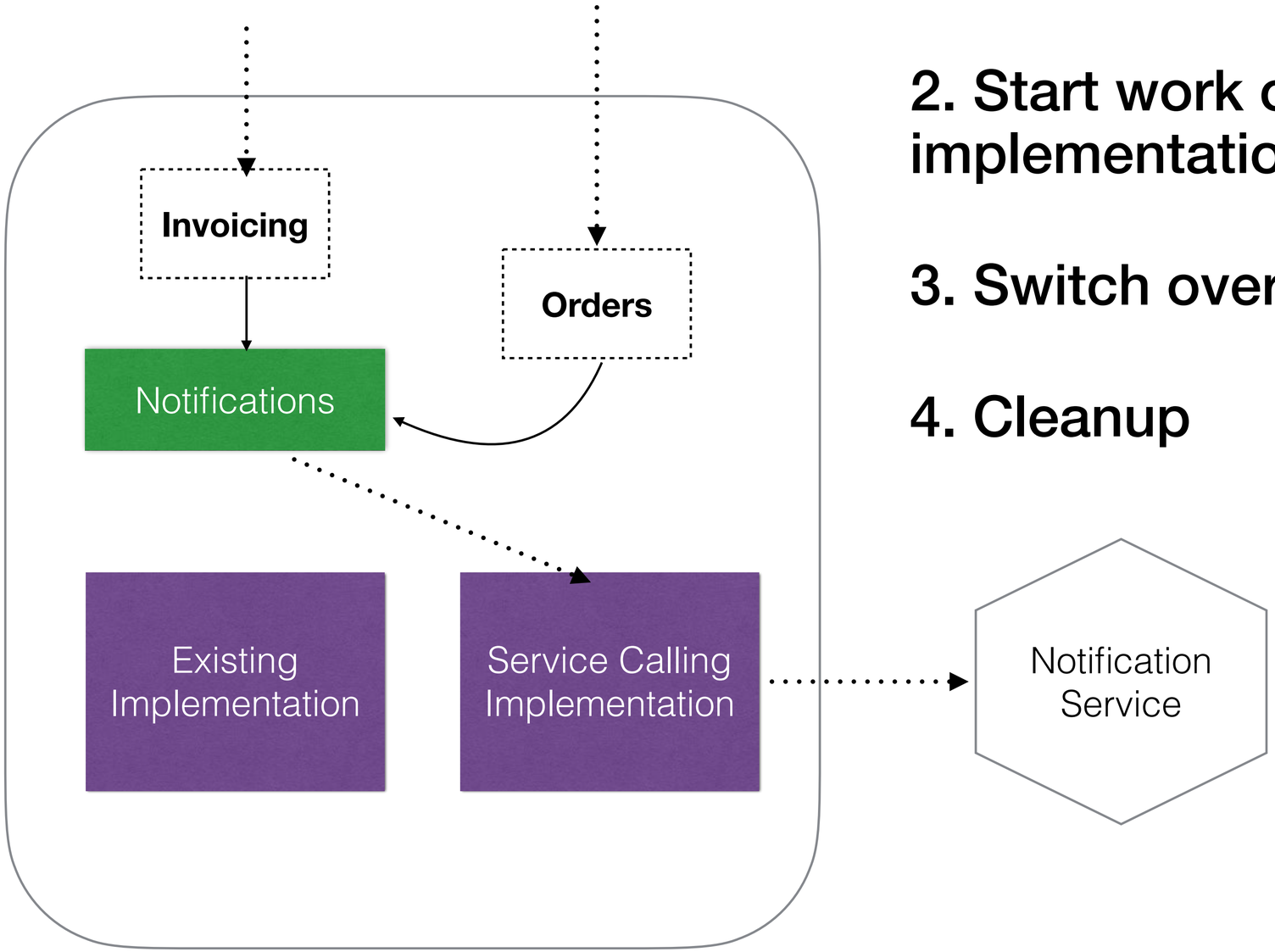


1. Create abstraction point

2. Start work on new service implementation

3. Switch over

EXAMPLE



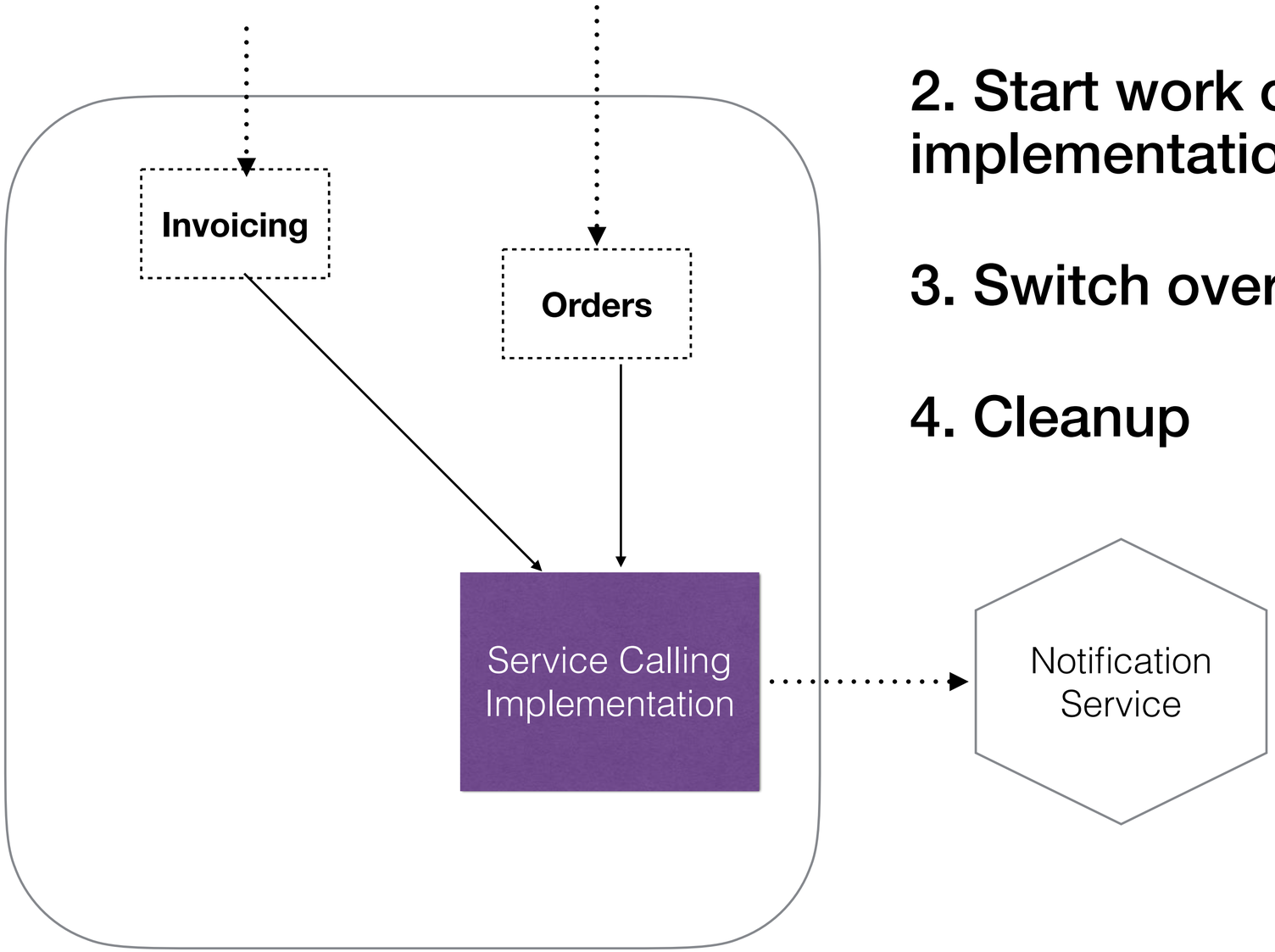
1. Create abstraction point

2. Start work on new service implementation

3. Switch over

4. Cleanup

EXAMPLE

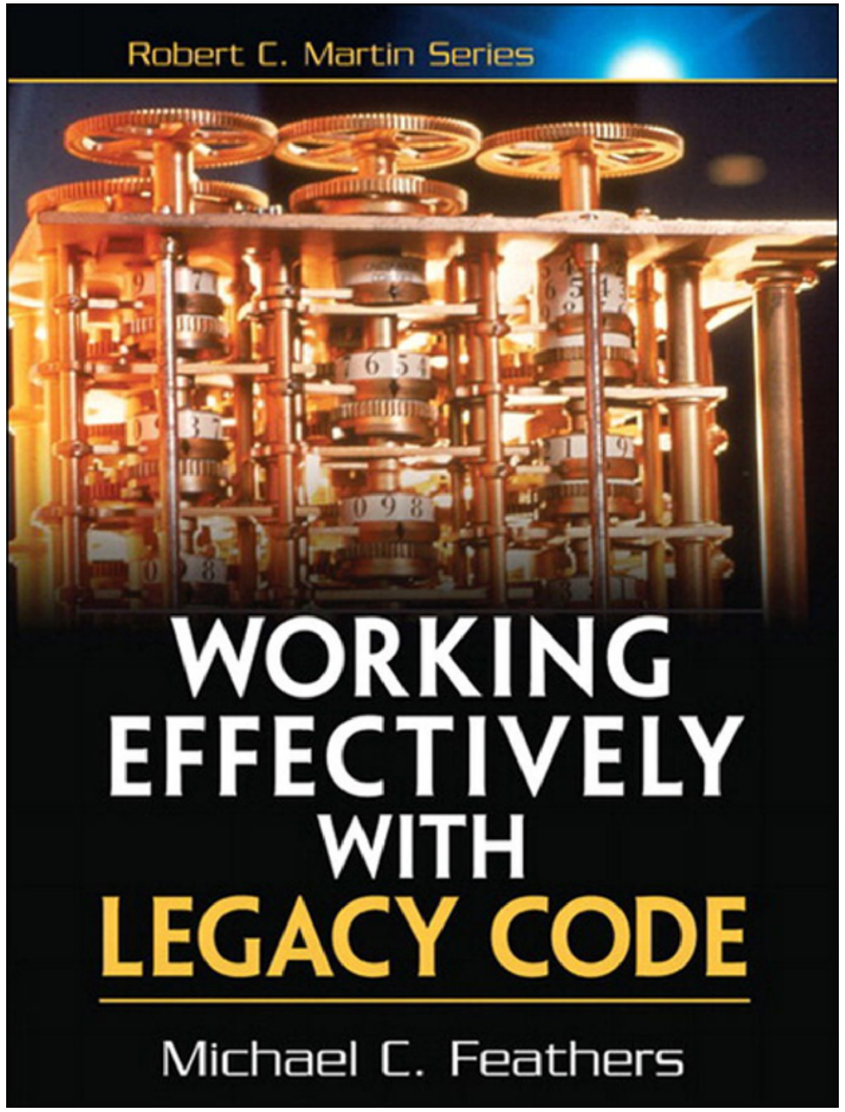


1. Create abstraction point

2. Start work on new service implementation

3. Switch over

4. Cleanup


Robert C. Martin Series
WORKING EFFECTIVELY WITH LEGACY CODE
Michael C. Feathers

PATTERN: FEATURE TOGGLES

https://www.flickr.com/photos/paulcross/4333070249/

PATTERN: FEATURE TOGGLES

Feature Toggles (aka Feature Flags)

Feature Toggles (often also refered to as Feature Flags) are a powerful technique, allowing teams to modify system behavior without changing code. They fall into various usage categories, and it's important to take that categorization into account when implementing and managing toggles. Toggles introduce complexity. We can keep that complexity in check by using smart toggle implementation practices and appropriate tools to manage our toggle configuration, but we should also aim to constrain the number of toggles in our system.

09 October 2017

Pete Hodgson

Pete Hodgson is an independent software delivery consultant based in the San Francisco Bay Area. He specializes in helping startup

CONTENTS

expand

Allow for functionality to be toggled off and on

https://martinfowler.com/articles/feature-toggles.html

PATTERN: FEATURE TOGGLES

Feature Toggles (aka Feature Flags)

Feature Toggles (often also refered to as Feature Flags) are a powerful technique, allowing teams to modify system behavior without changing code. They fall into various usage categories, and it's important to take that categorization into account when implementing and managing toggles. Toggles introduce complexity. We can keep that complexity in check by using smart toggle implementation practices and appropriate tools to manage our toggle configuration, but we should also aim to constrain the number of toggles in our system.

09 October 2017

Pete Hodgson

Pete Hodgson is an independent software delivery consultant based in the San Francisco Bay Area. He specializes in helping startup

CONTENTS

expand

Allow for functionality to be toggled off and on

Can also be used to switch between alternative implementations

https://martinfowler.com/articles/feature-toggles.html

PATTERN: FEATURE TOGGLES

Feature Toggles (aka Feature Flags)

Feature Toggles (often also refered to as Feature Flags) are a powerful technique, allowing teams to modify system behavior without changing code. They fall into various usage categories, and it's important to take that categorization into account when implementing and managing toggles. Toggles introduce complexity. We can keep that complexity in check by using smart toggle implementation practices and appropriate tools to manage our toggle configuration, but we should also aim to constrain the number of toggles in our system.

09 October 2017

Pete Hodgson

Pete Hodgson is an independent software delivery consultant based in the San Francisco Bay Area. He specializes in helping startup

CONTENTS

expand

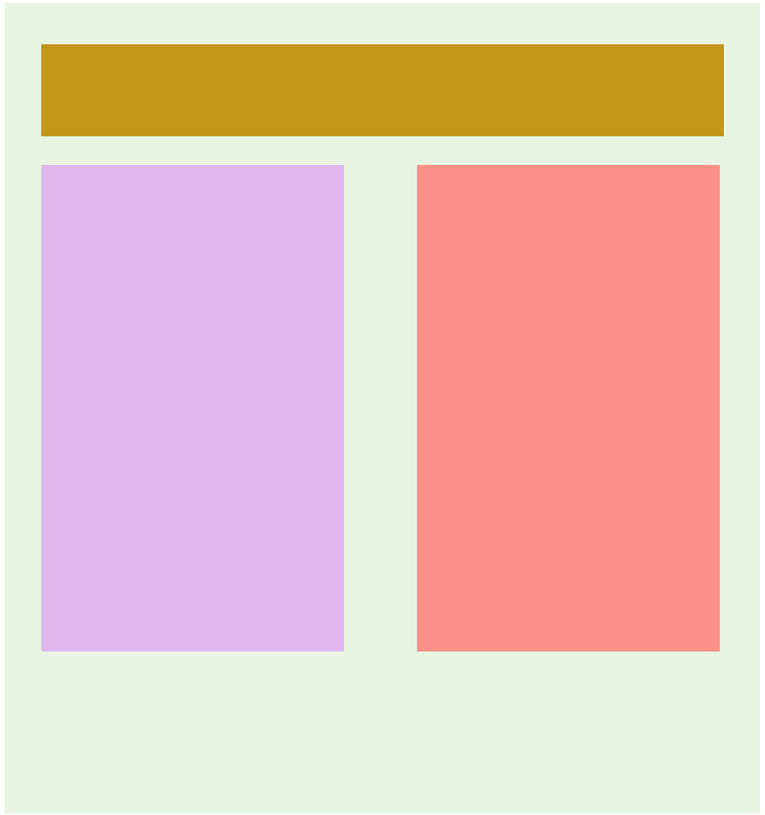
Allow for functionality to be toggled off and on

Can also be used to switch between alternative implementations

Changing toggled values doesn't require a rebuild or redeploy

https://martinfowler.com/articles/feature-toggles.html

FEATURE TOGGLES (CONT)

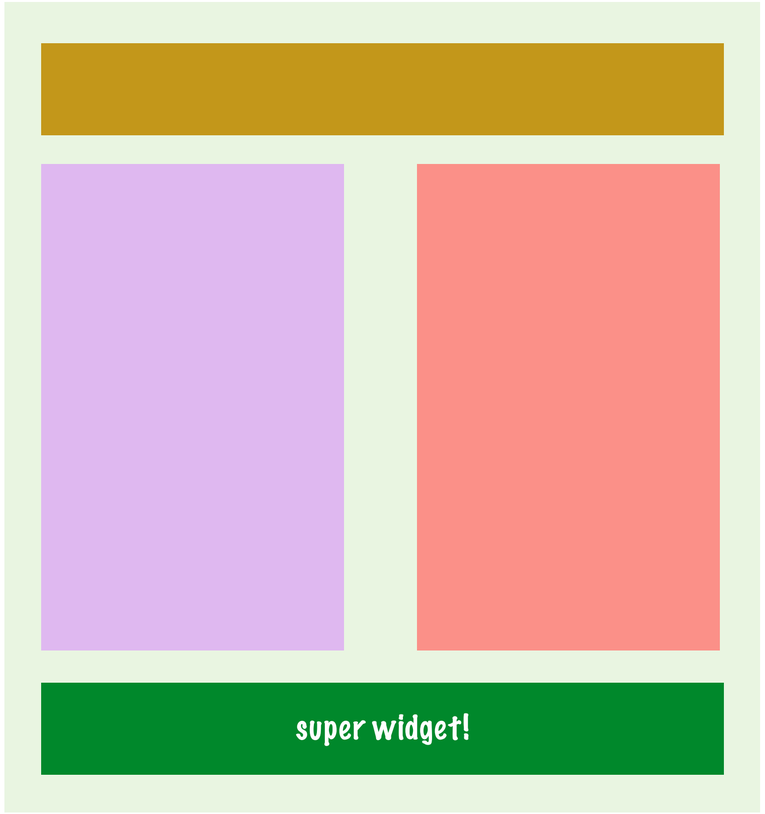


Specified in configuration file

```
…
super_widget = off
…
```

Specified in configuration file

FEATURE TOGGLES (CONT)

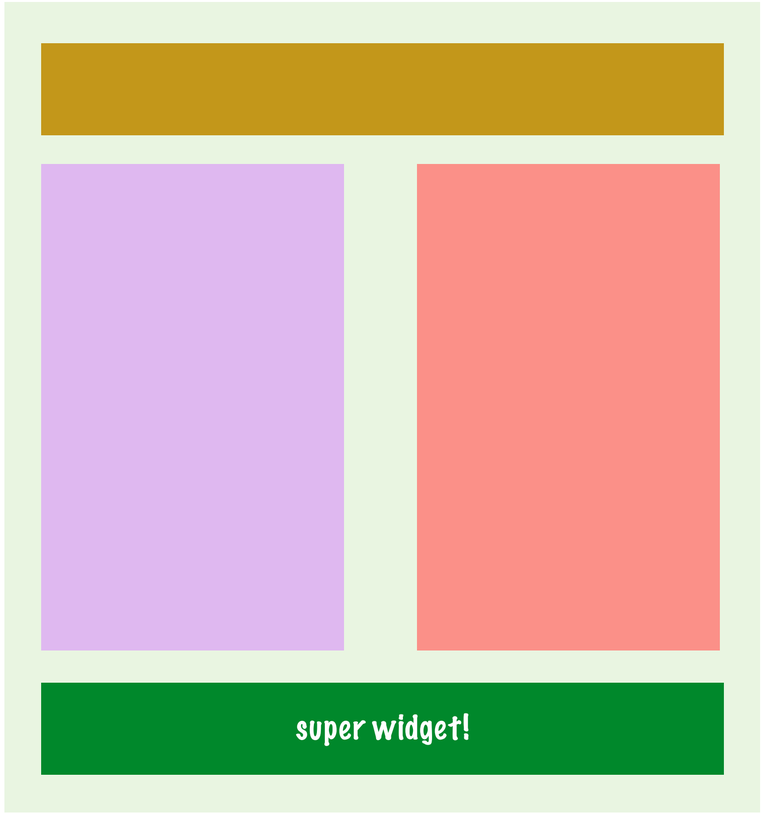


Specified in configuration file

```
...
  super_widget = on
...
```

Specified in configuration file

FEATURE TOGGLES (CONT)



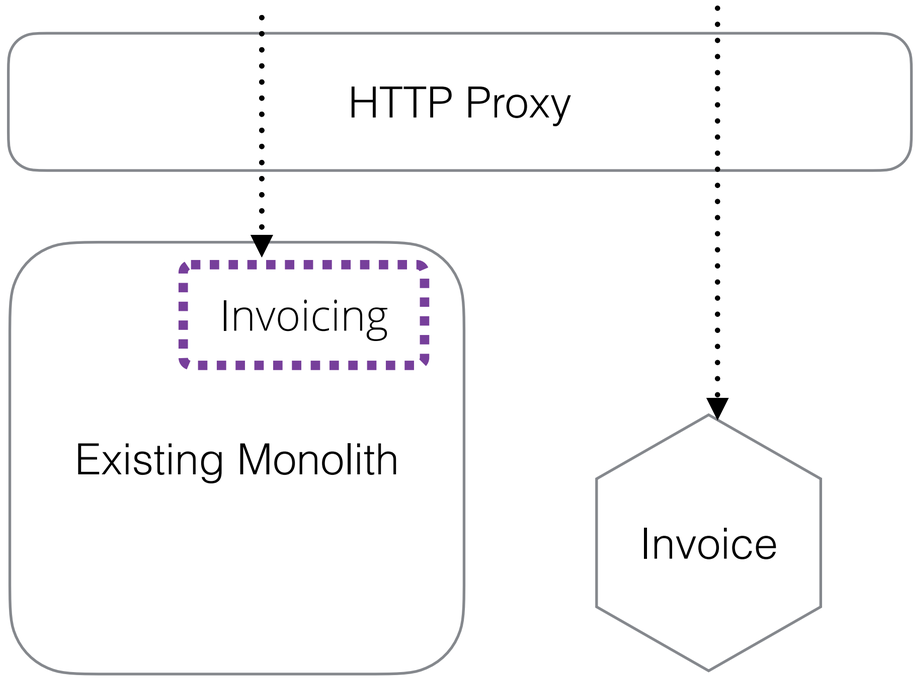
Specified in configuration file

```
...
  super_widget = on
...
```

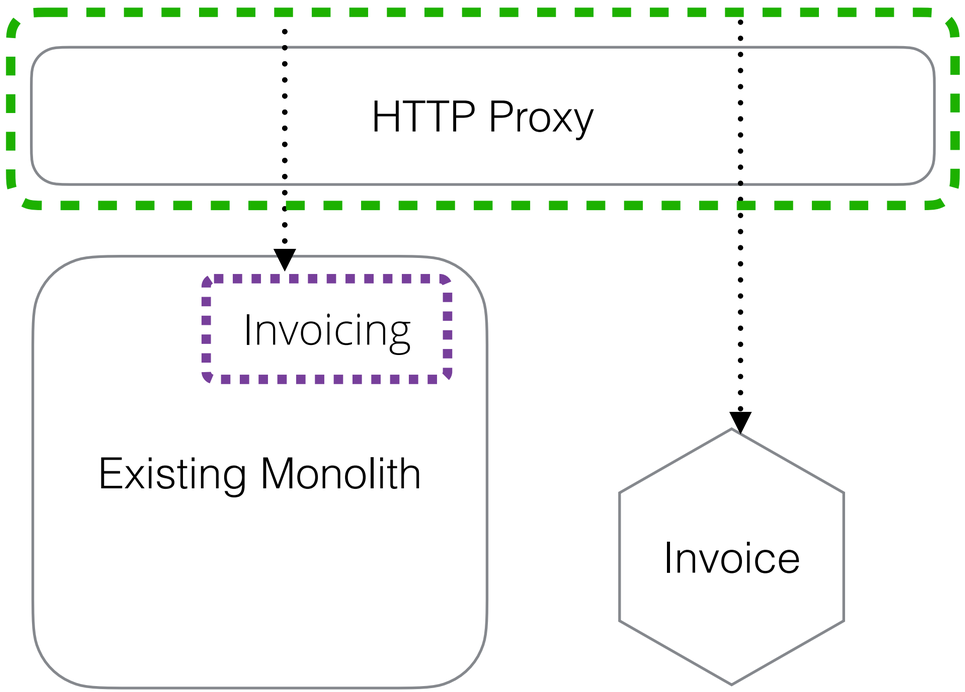
Specified in configuration file

```
$. run -Dsuper_widget=on
```

WITH THE STRANGLER FIG PATTERN

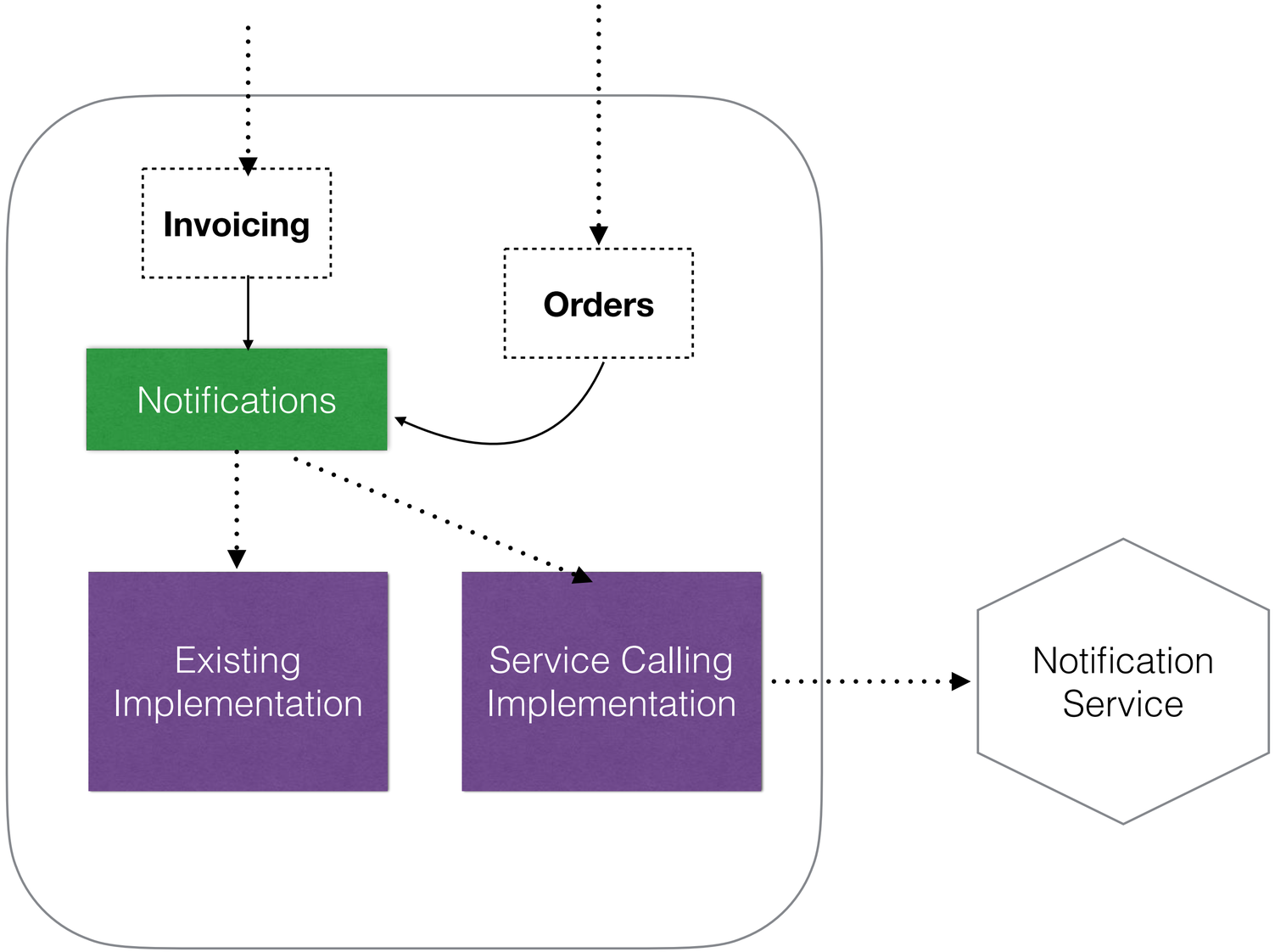


WITH THE STRANGLER FIG PATTERN



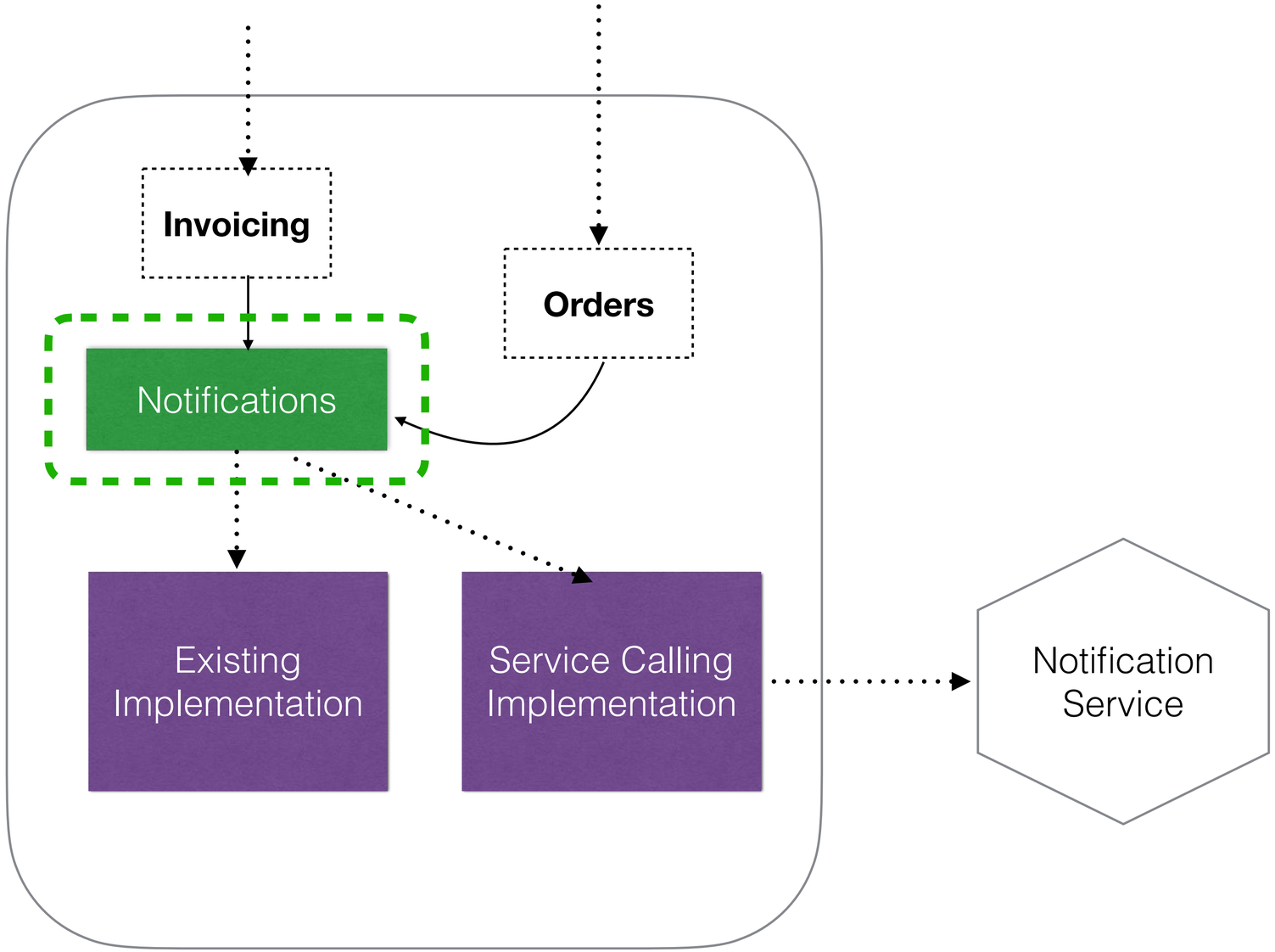
Implement a simple config-based toggle in the HTTP Proxy configuration

WITH BRANCH BY ABSTRACTION



WITH BRANCH BY ABSTRACTION

Use the abstraction to divert calls based on toggle value



PROGRESSIVE DELIVERY

https://redmonk.com/jgovernor/2018/08/06/towards-progressive-delivery/

PROGRESSIVE DELIVERY

Canary

https://redmonk.com/jgovernor/2018/08/06/towards-progressive-delivery/

PROGRESSIVE DELIVERY

Canary

A/B Testing

https://redmonk.com/jgovernor/2018/08/06/towards-progressive-delivery/

PROGRESSIVE DELIVERY

Canary

A/B Testing

Parallel Runs

https://redmonk.com/jgovernor/2018/08/06/towards-progressive-delivery/

PROGRESSIVE DELIVERY

JAMES GOVERNOR'S MONKCHIPS

Towards Progressive Delivery

By James Governor | @monkchips | August 6, 2018

At RedMonk we generally try to avoid coming up with new terms for technologies and

Canary

A/B Testing

Parallel Runs

Blue/Green

https://redmonk.com/jgovernor/2018/08/06/towards-progressive-delivery/

PROGRESSIVE DELIVERY

Canary

A/B Testing

Parallel Runs

Blue/Green

Dark Launching

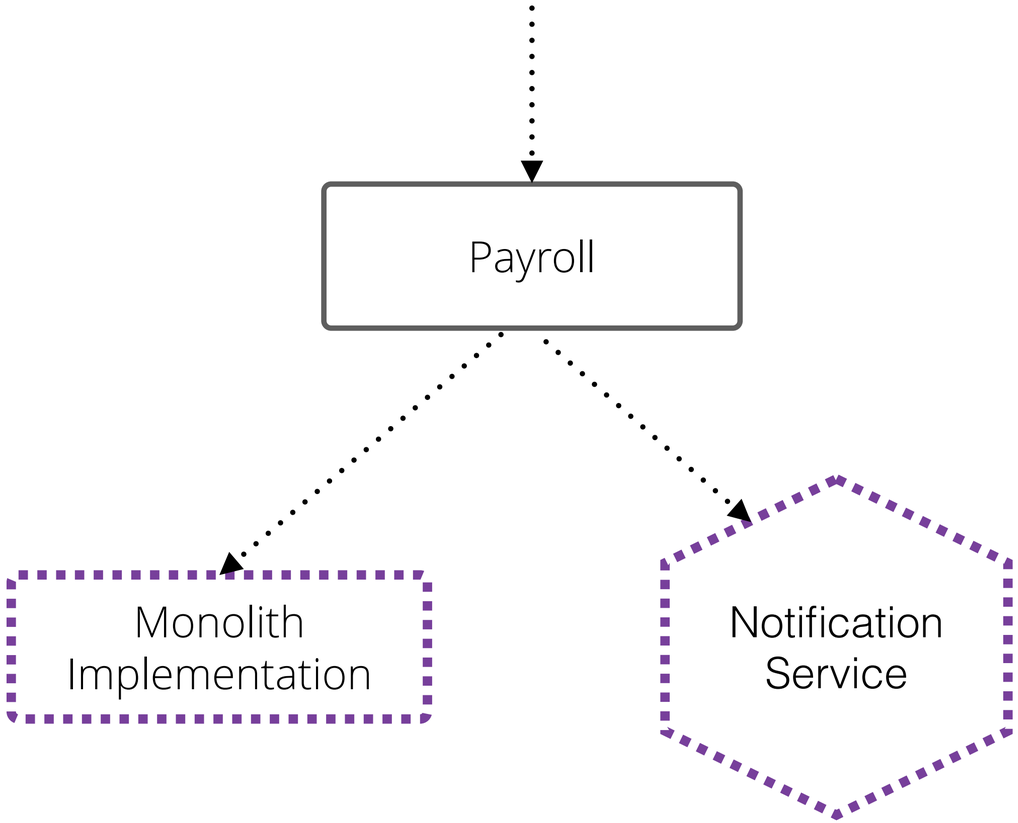
https://redmonk.com/jgovernor/2018/08/06/towards-progressive-delivery/

PATTERN: PARALLEL RUN

Rather than executing one implementation or another, execute both

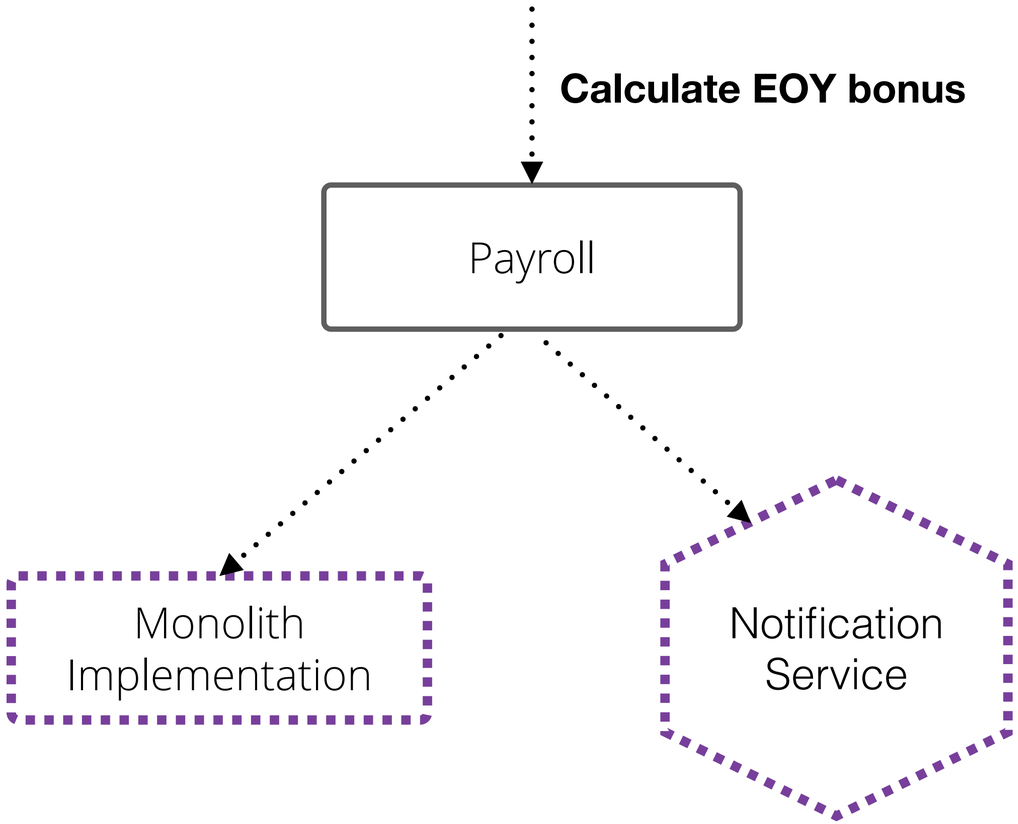
PATTERN: PARALLEL RUN

Rather than executing one implementation or another, execute both



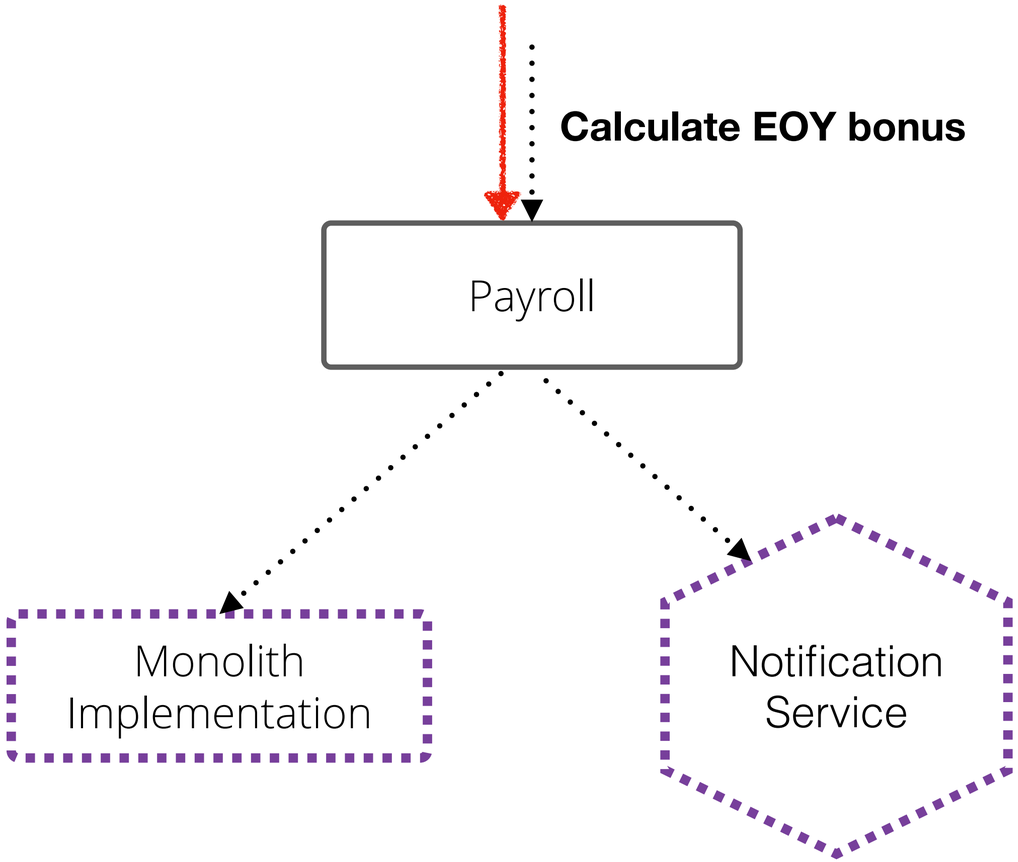
PATTERN: PARALLEL RUN

Rather than executing one implementation or another, execute both



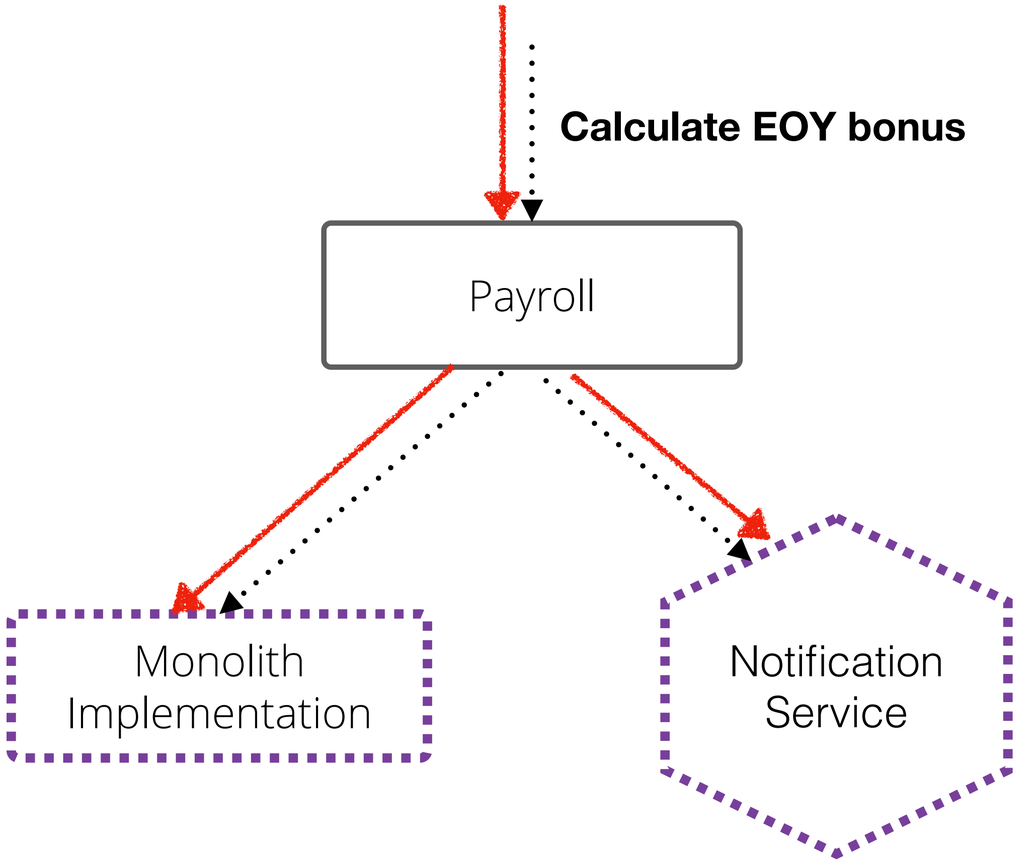
PATTERN: PARALLEL RUN

Rather than executing one implementation or another, execute both



PATTERN: PARALLEL RUN

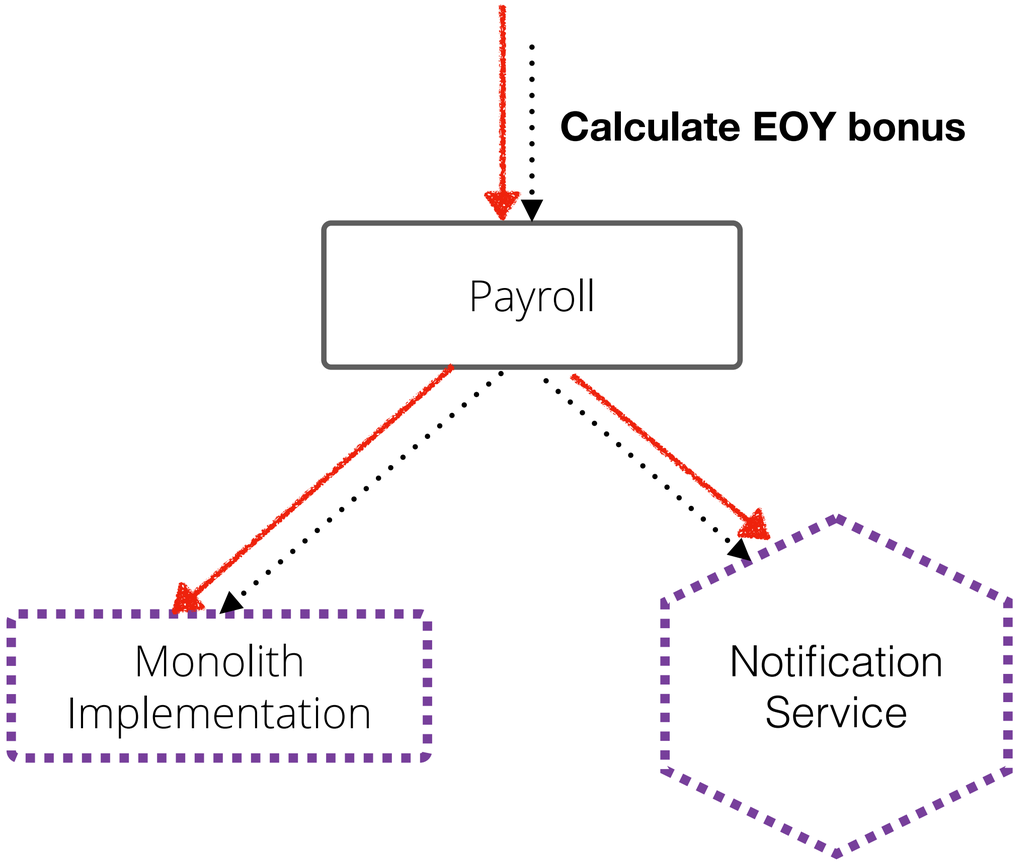
Rather than executing one implementation or another, execute both



PATTERN: PARALLEL RUN

Rather than executing one implementation or another, execute both

Compare the results, and ensure they are the same

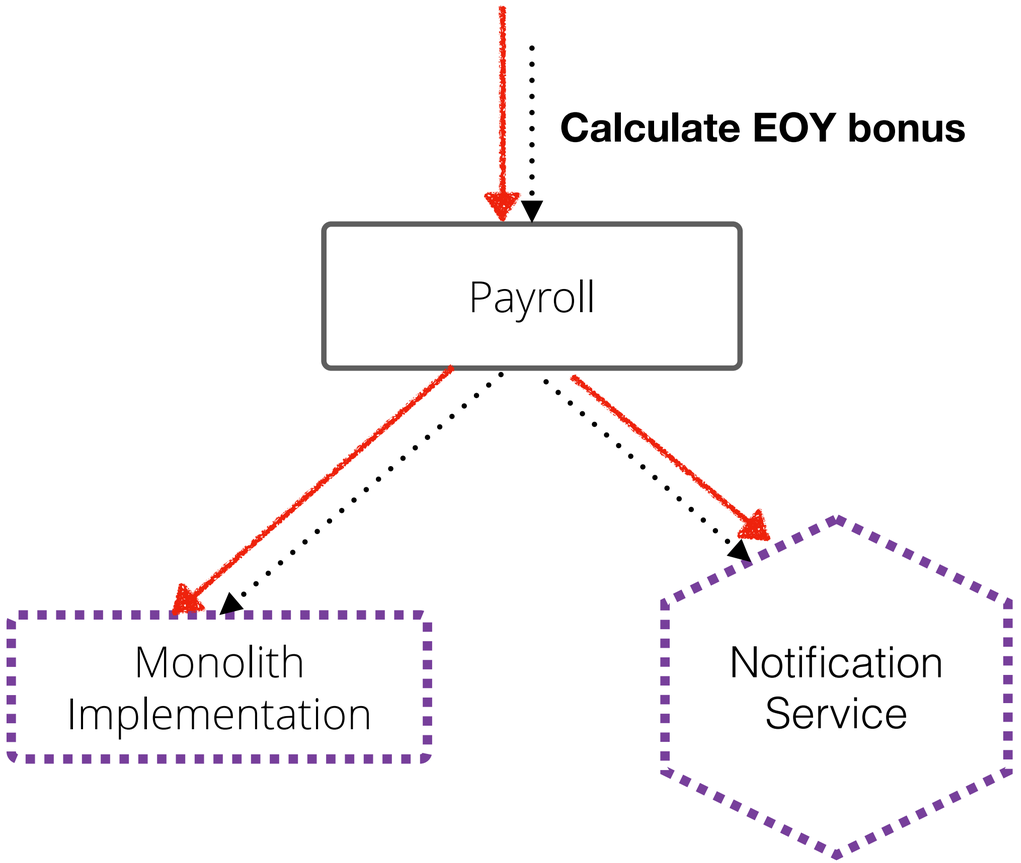


PATTERN: PARALLEL RUN

Rather than executing one implementation or another, execute both

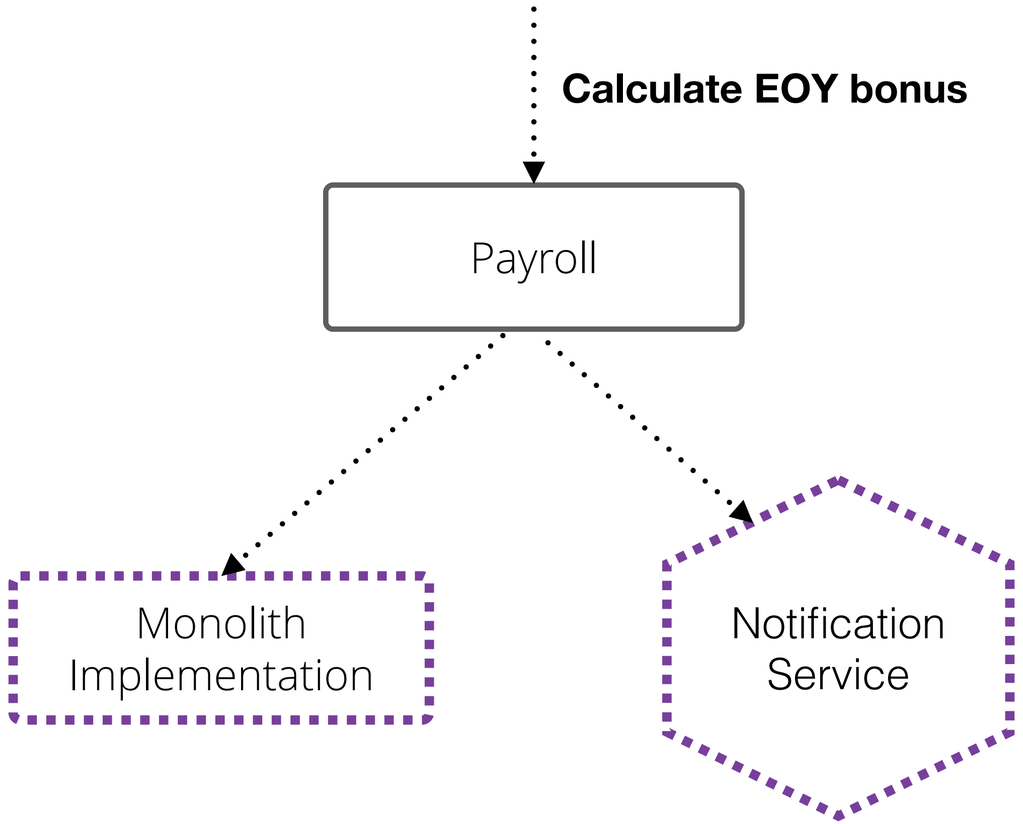
Compare the results, and ensure they are the same

Can be useful to ensure your refactoring doesn't change behaviour



SOURCE OF TRUTH

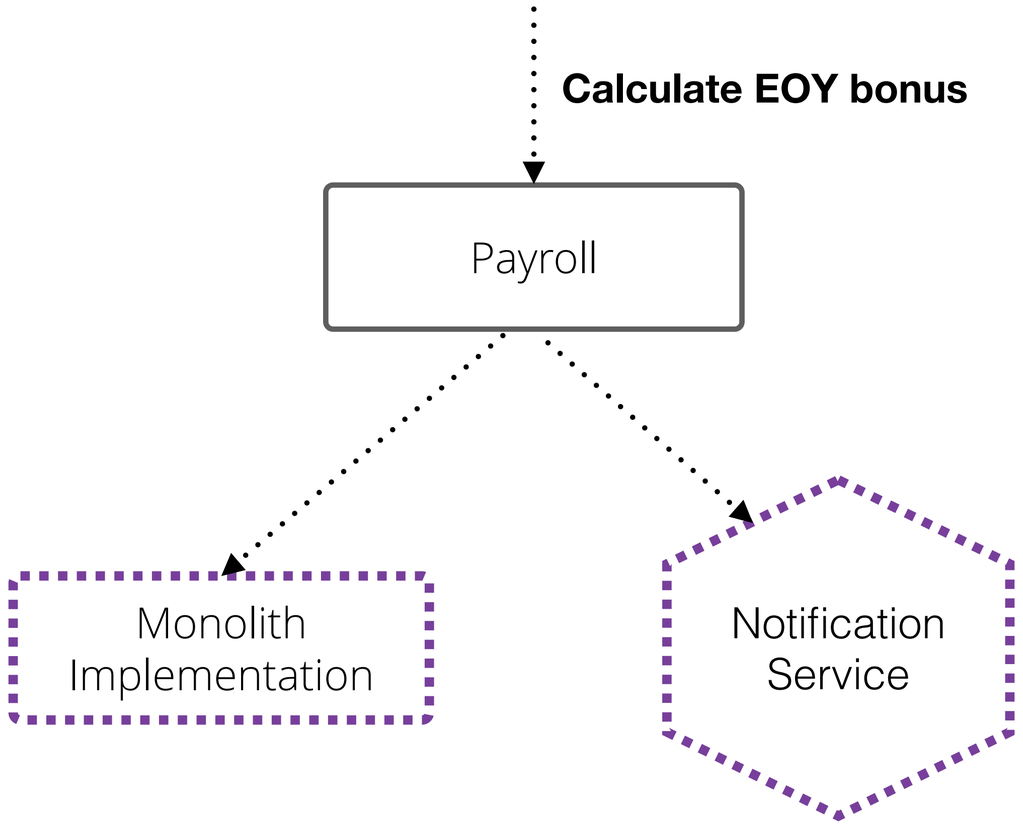
You'd only treat one implementation as the source of truth



SOURCE OF TRUTH

You'd only treat one implementation as the source of truth

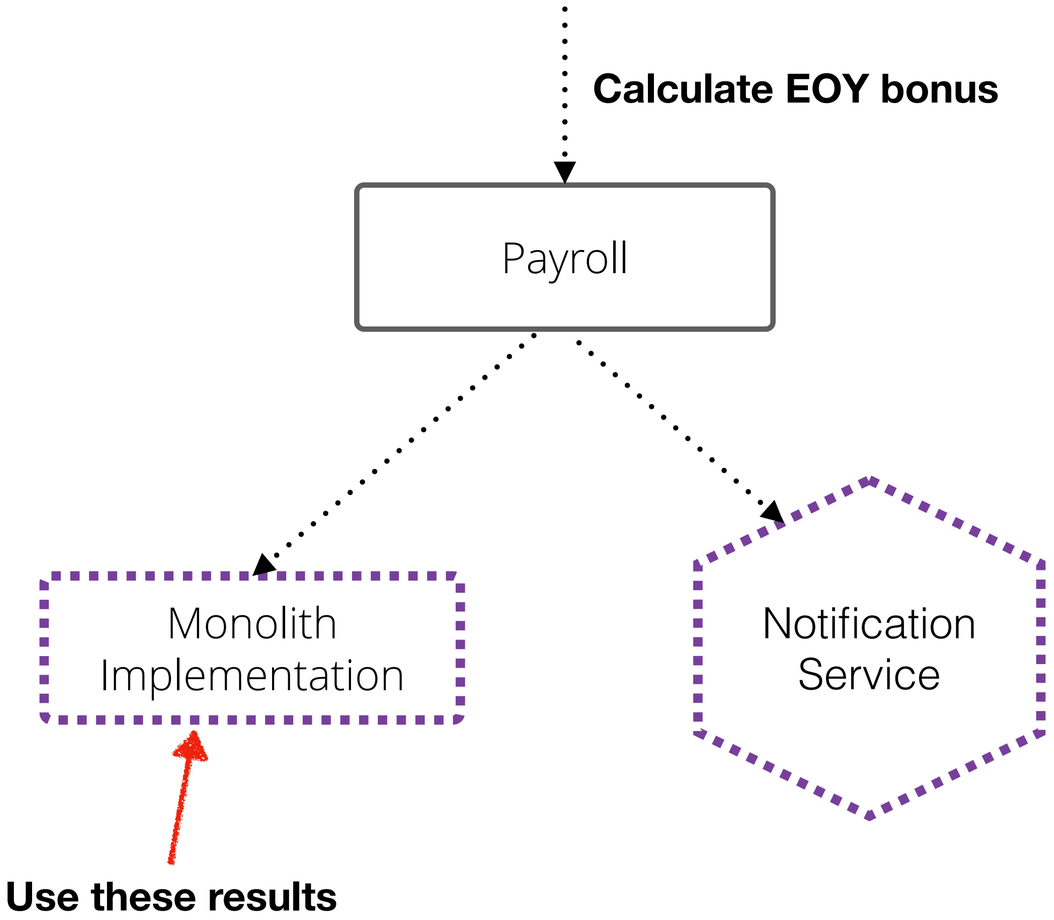
Initially, this would be the existing implementation



SOURCE OF TRUTH

You'd only treat one implementation as the source of truth

Initially, this would be the existing implementation

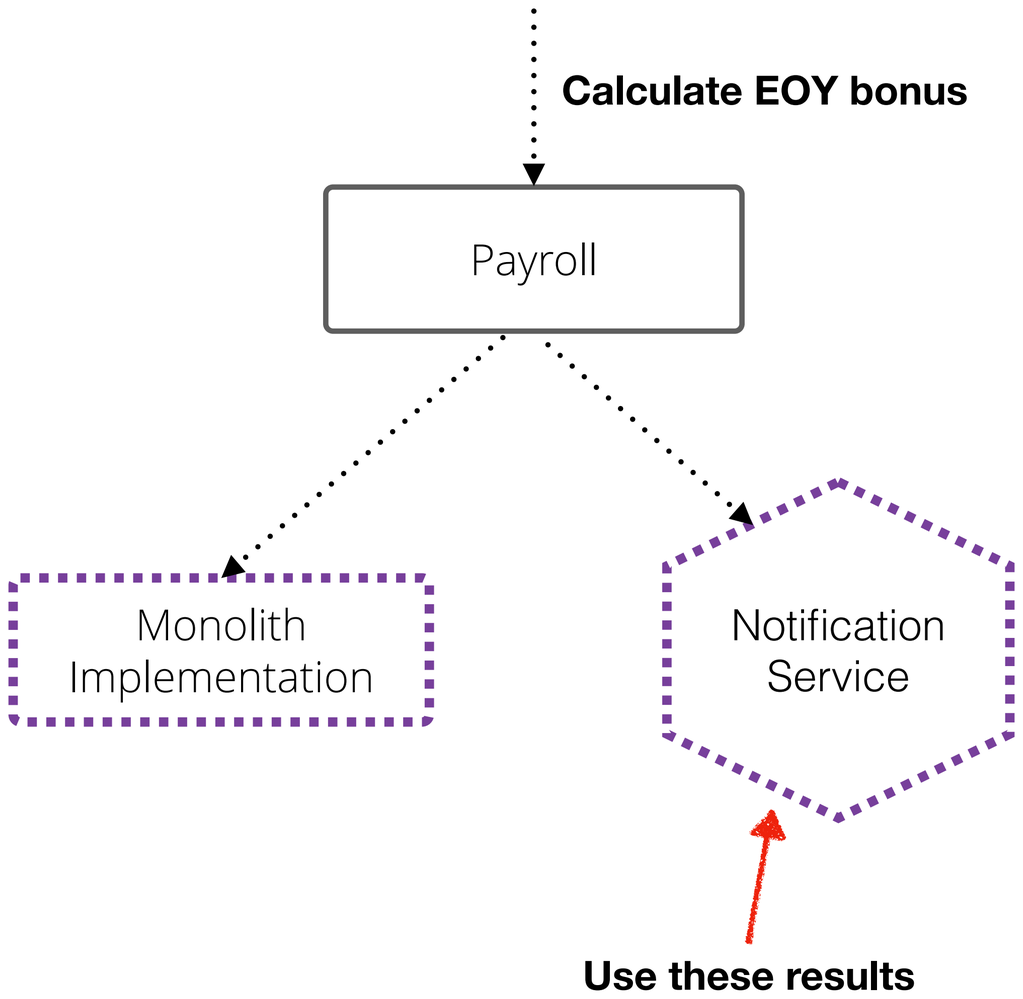


SOURCE OF TRUTH

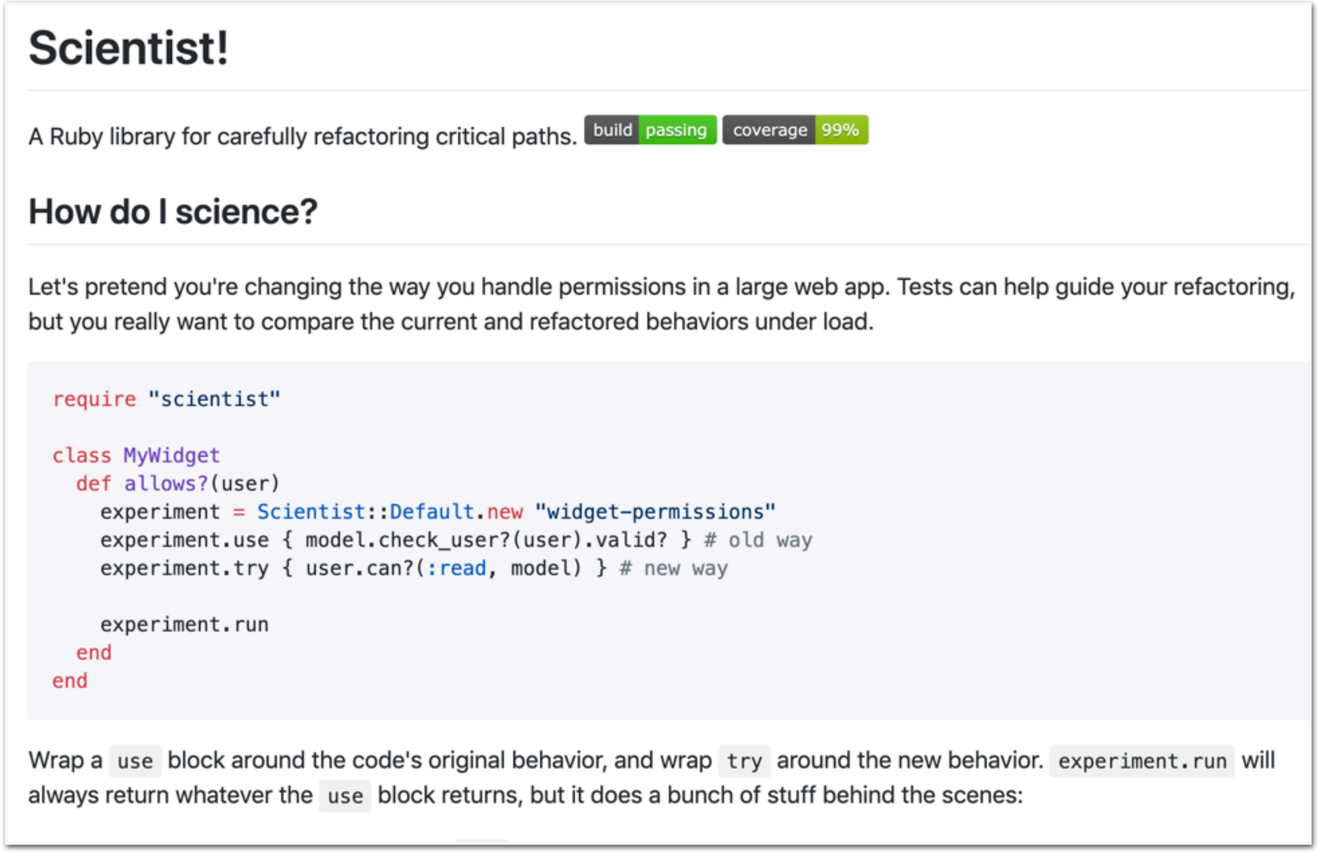
You'd only treat one implementation as the source of truth

Initially, this would be the existing implementation

Later, this could be your new microservice implementation



GITHUB SCIENTIST



Scientist!

A Ruby library for carefully refactoring critical paths. build passing coverage 99%

How do I science?

Let's pretend you're changing the way you handle permissions in a large web app. Tests can help guide your refactoring, but you really want to compare the current and refactored behaviors under load.

```
require "scientist"

class MyWidget
  def allows?(user)
    experiment = Scientist::Default.new "widget-permissions"
    experiment.use { model.check_user?(user).valid? } # old way
    experiment.try { user.can?(:read, model) } # new way

    experiment.run
  end
end
```

Wrap a `use` block around the code's original behavior, and wrap `try` around the new behavior. `experiment.run` will always return whatever the `use` block returns, but it does a bunch of stuff behind the scenes:

https://github.com/github/scientist

GITHUB SCIENTIST

Scientist!

A Ruby library for carefully refactoring critical paths. build passing coverage 99%

How do I science?

Let's pretend you're changing the way you handle permissions in a large web app. Tests can help guide your refactoring, but you really want to compare the current and refactored behaviors under load.

```
require "scientist"

class MyWidget
  def allows?(user)
    experiment = Scientist::Default.new "widget-permissions"
    experiment.use { model.check_user?(user).valid? } # old way
    experiment.try { user.can?(:read, model) } # new way

    experiment.run
  end
end
```

Wrap a `use` block around the code's original behavior, and wrap `try` around the new behavior. `experiment.run` will always return whatever the `use` block returns, but it does a bunch of stuff behind the scenes:

Alternatives

- daylerees/scientist (PHP)
- scientistproject/scientist.net (.NET)
- joealcorn/laboratory (Python)
- rawls238/Scientist4J (Java)
- tomiaijo/scientist (C++)
- trello/scientist (node.js)
- ziyasal/scientist.js (node.js, ES6)
- yeller/laboratory (Clojure)
- lancew/Scientist (Perl 5)
- lancew/ScientistP6 (Perl 6)
- MadcapJake/Test-Lab (Perl 6)
- cwbriones/scientist (Elixir)
- calavera/go-scientist (Go)
- jelmersnoeck/experiment (Go)
- spoptchev/scientist (Kotlin / Java)
- junkpiano/scientist (Swift)

https://github.com/github/scientist

AGENDA

Introduction

Migration Approach

Application Refactoring

UI Decomposition

AGENDA

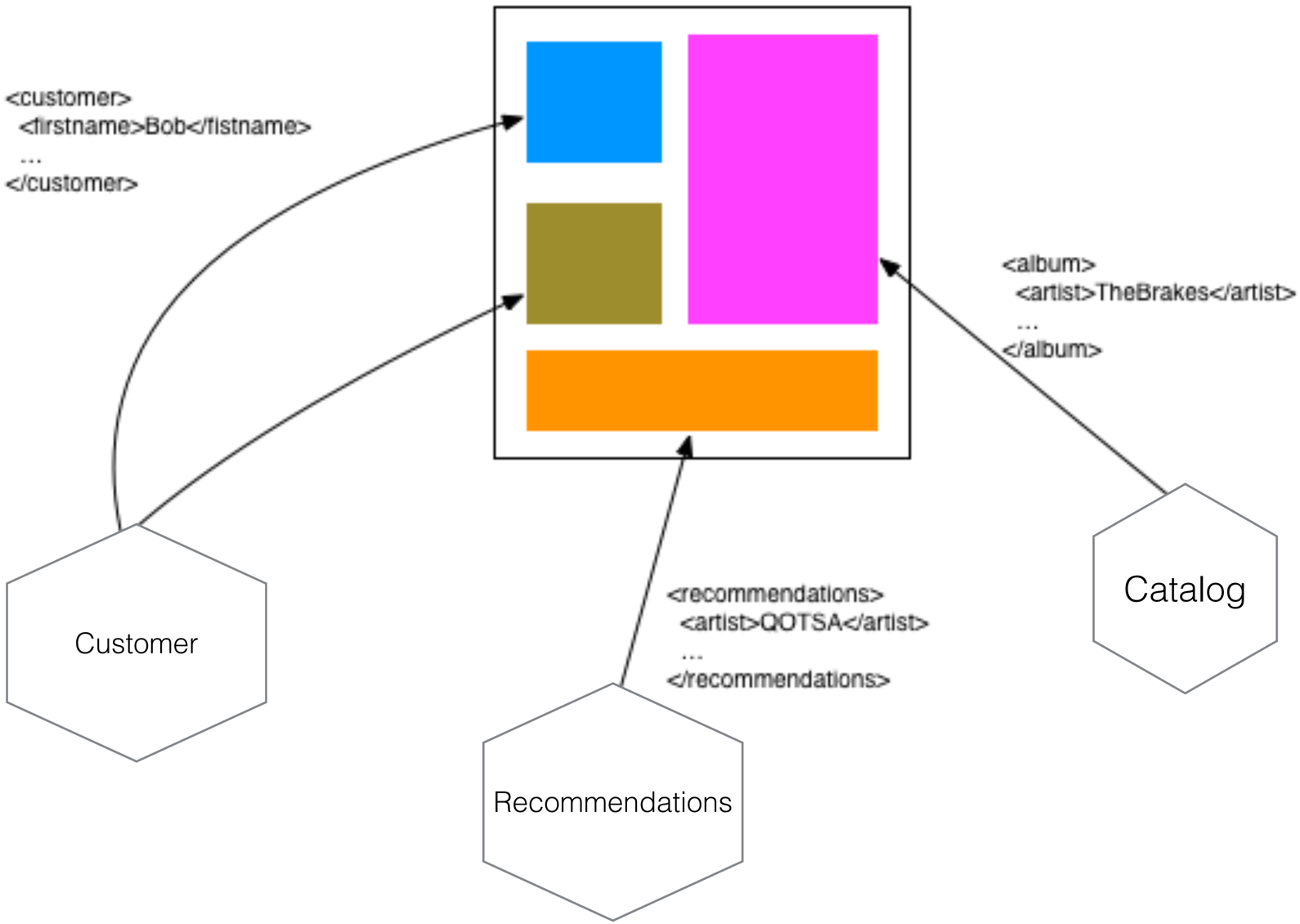
Introduction

Migration Approach

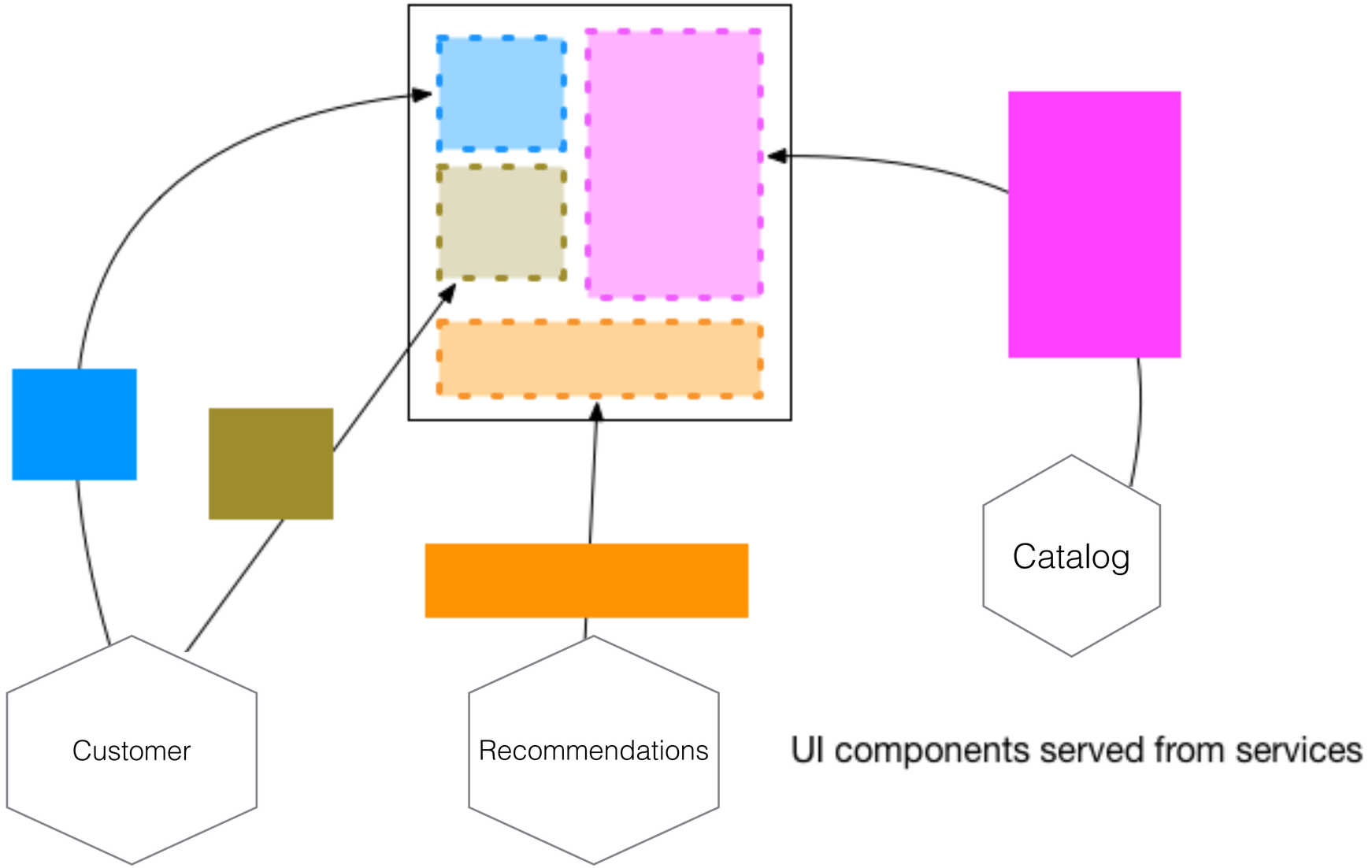
Application Refactoring

UI Decomposition

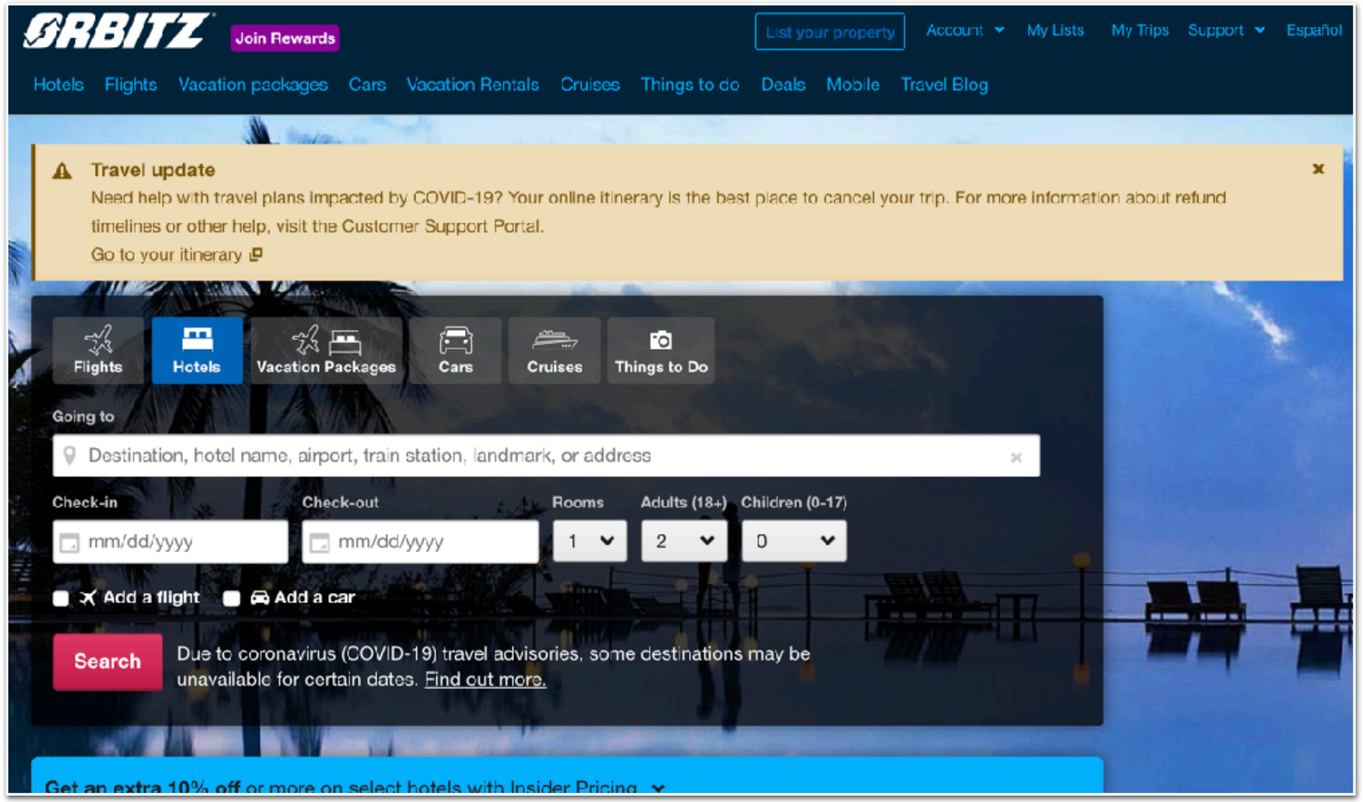
SINGLE-PAGE APP APPROACH



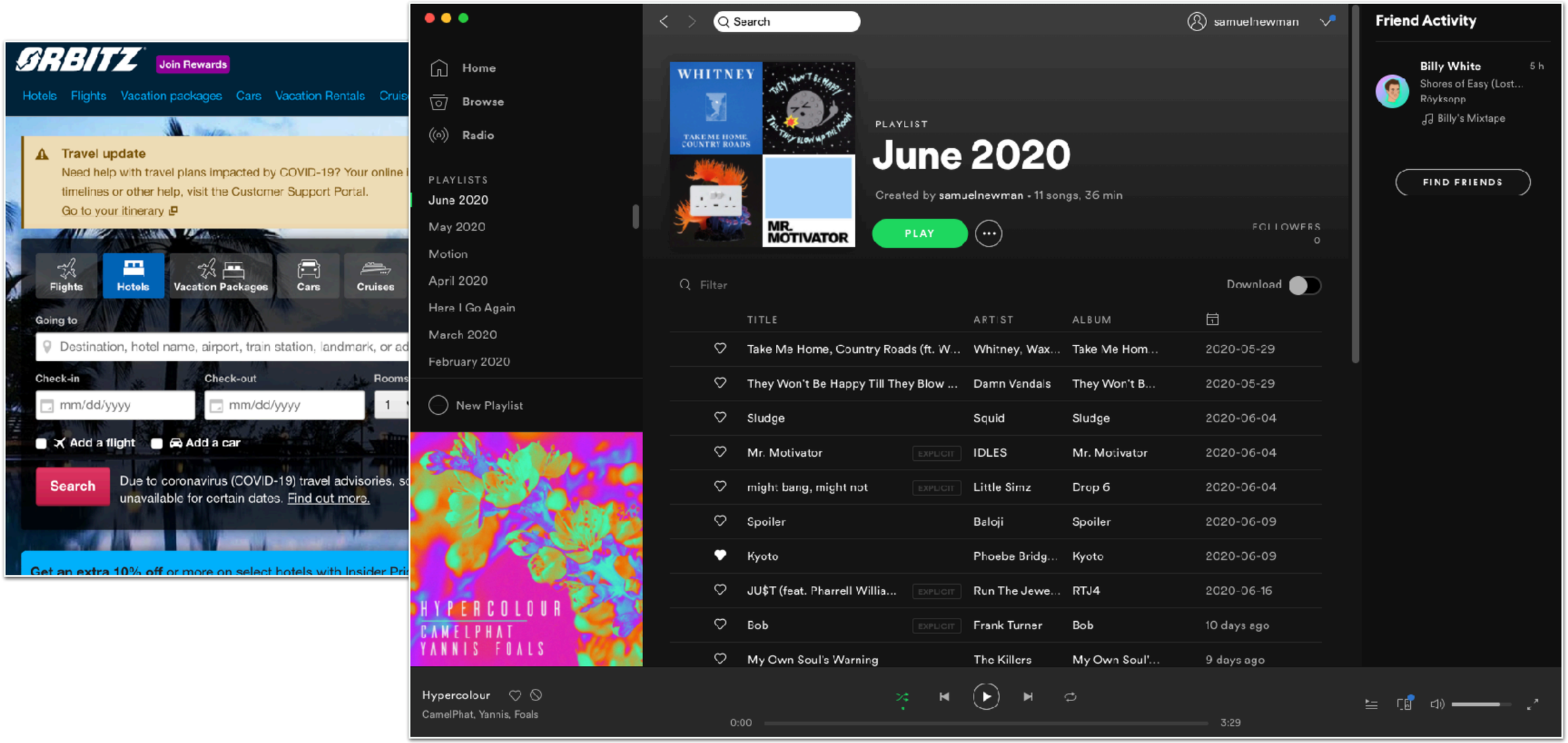
WIDGET-BASED ASSEMBLY



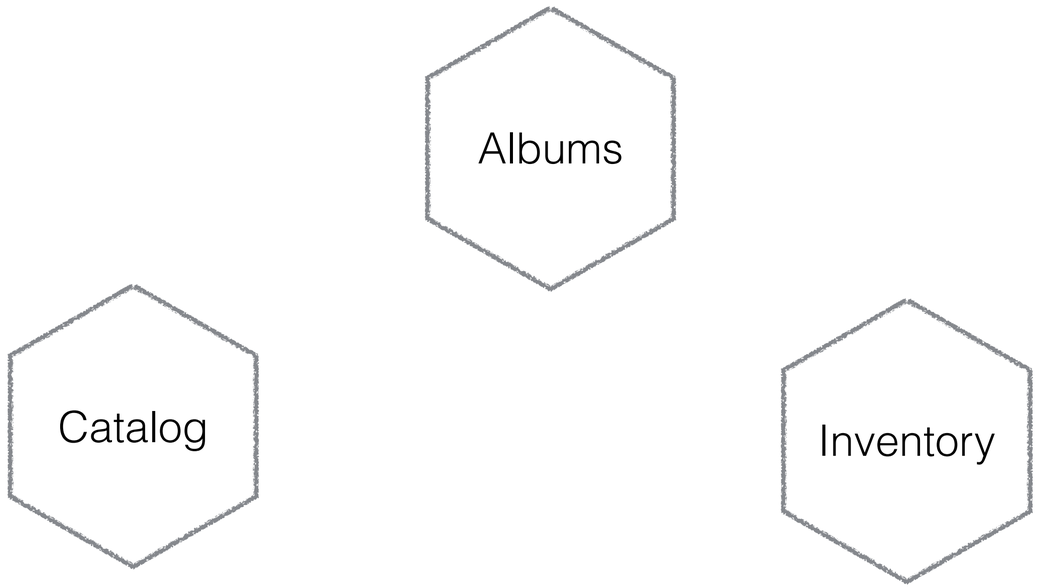
EXAMPLES OF WIDGET-BASED ASSEMBLY



EXAMPLES OF WIDGET-BASED ASSEMBLY

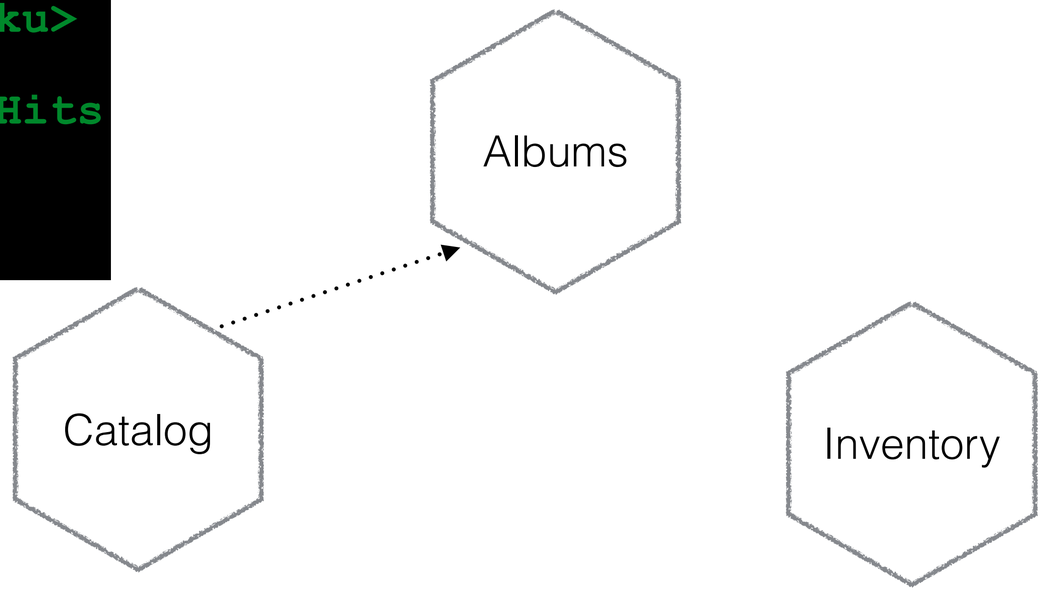


WIDGET-BASED ASSEMBLY - DETAILS

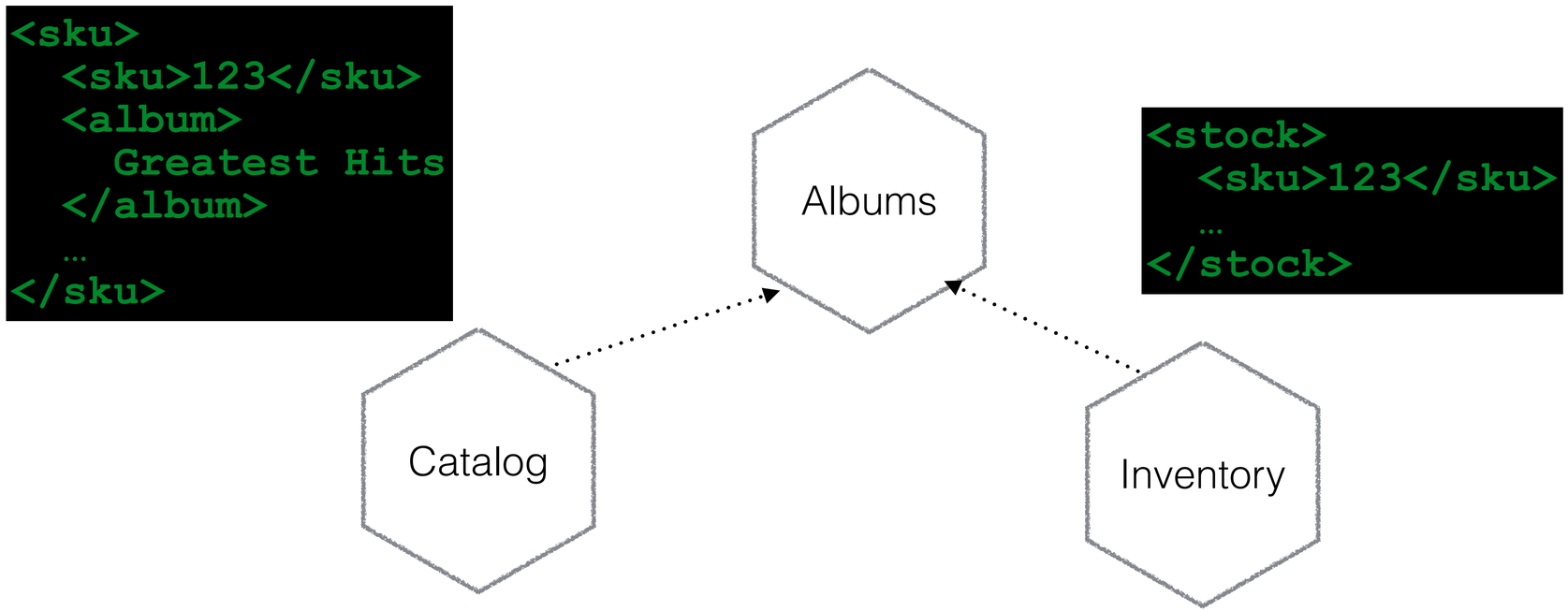


WIDGET-BASED ASSEMBLY - DETAILS

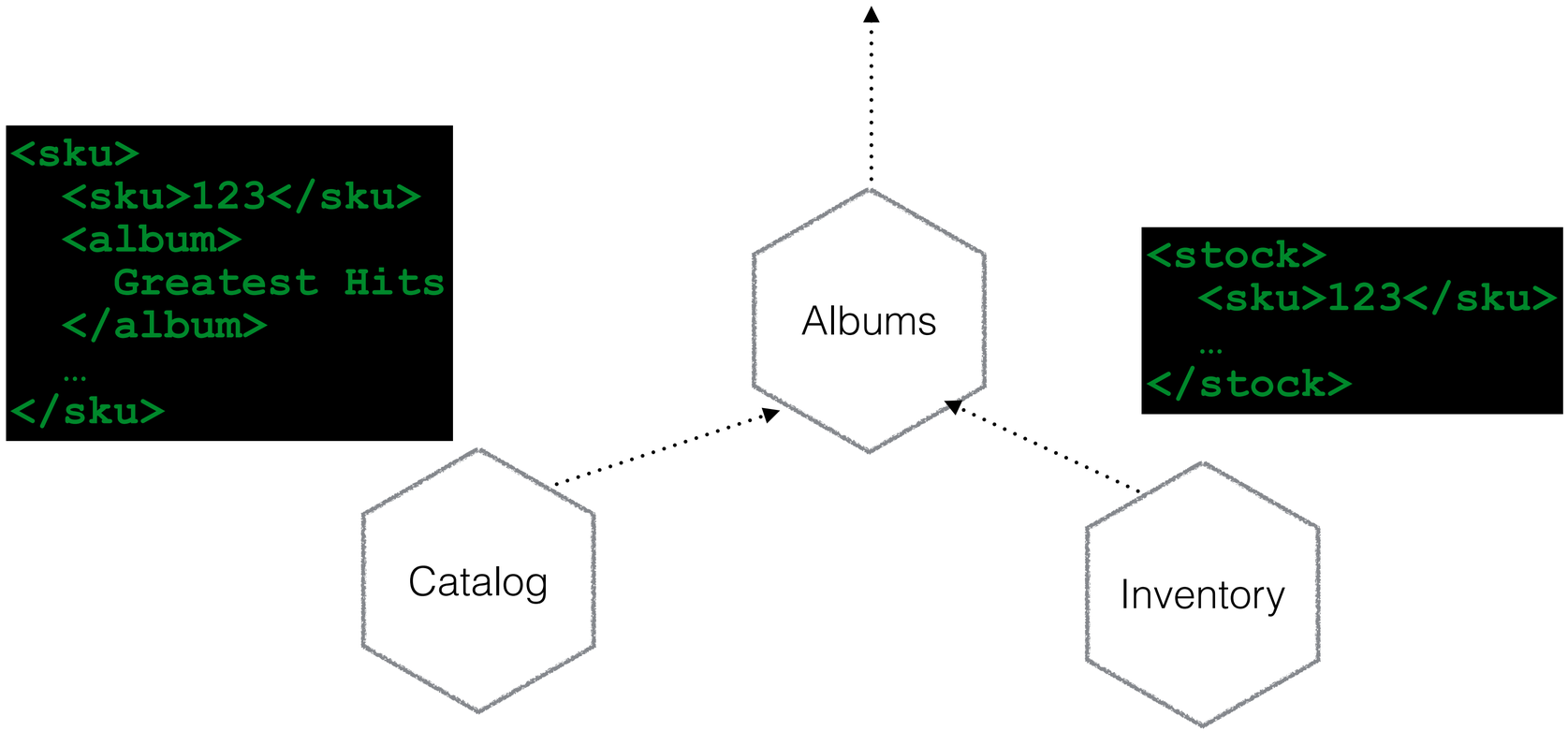
```
<sku>
  <sku>123</sku>
  <album>
    Greatest Hits
  </album>
  …
</sku>
```



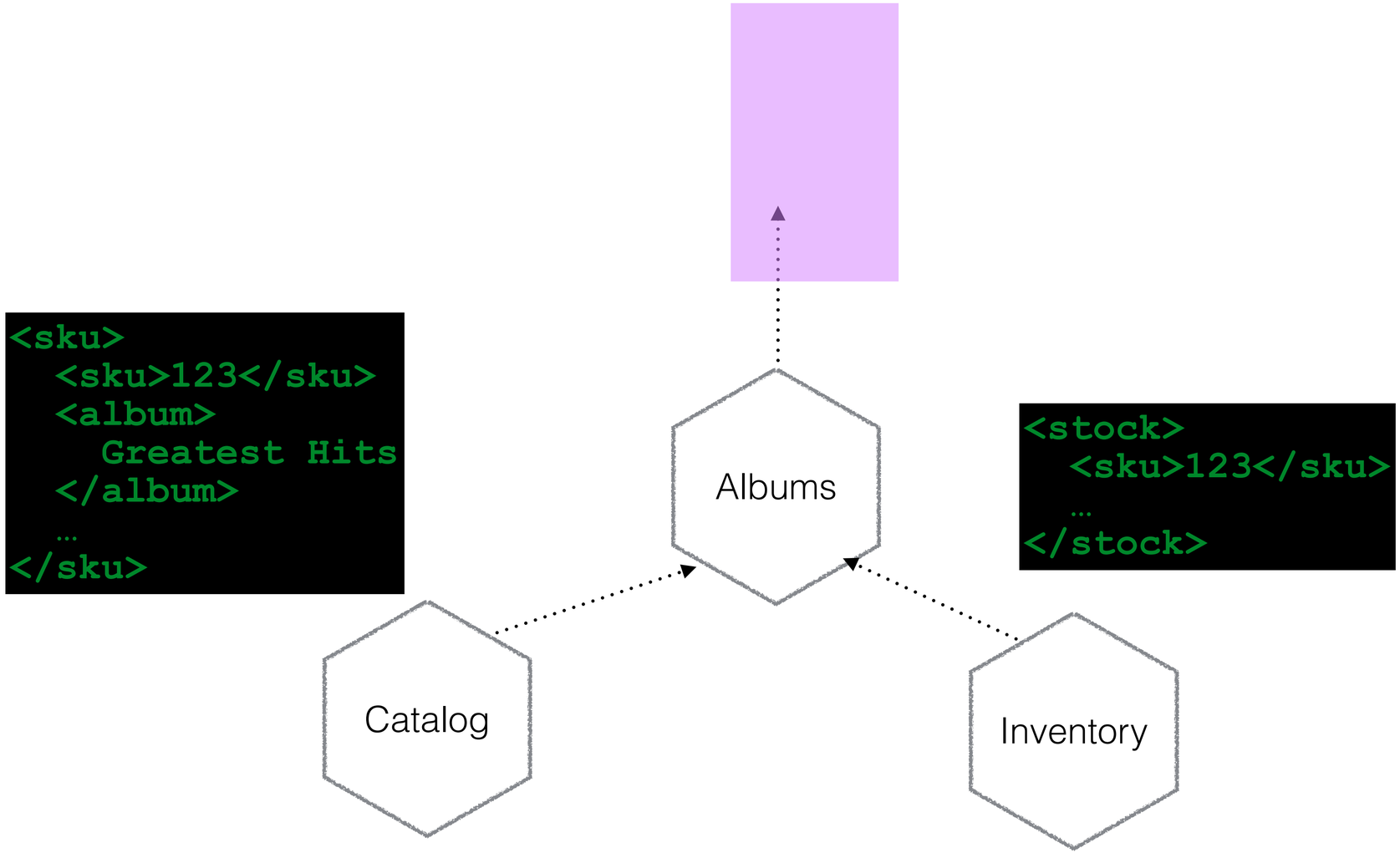
WIDGET-BASED ASSEMBLY - DETAILS



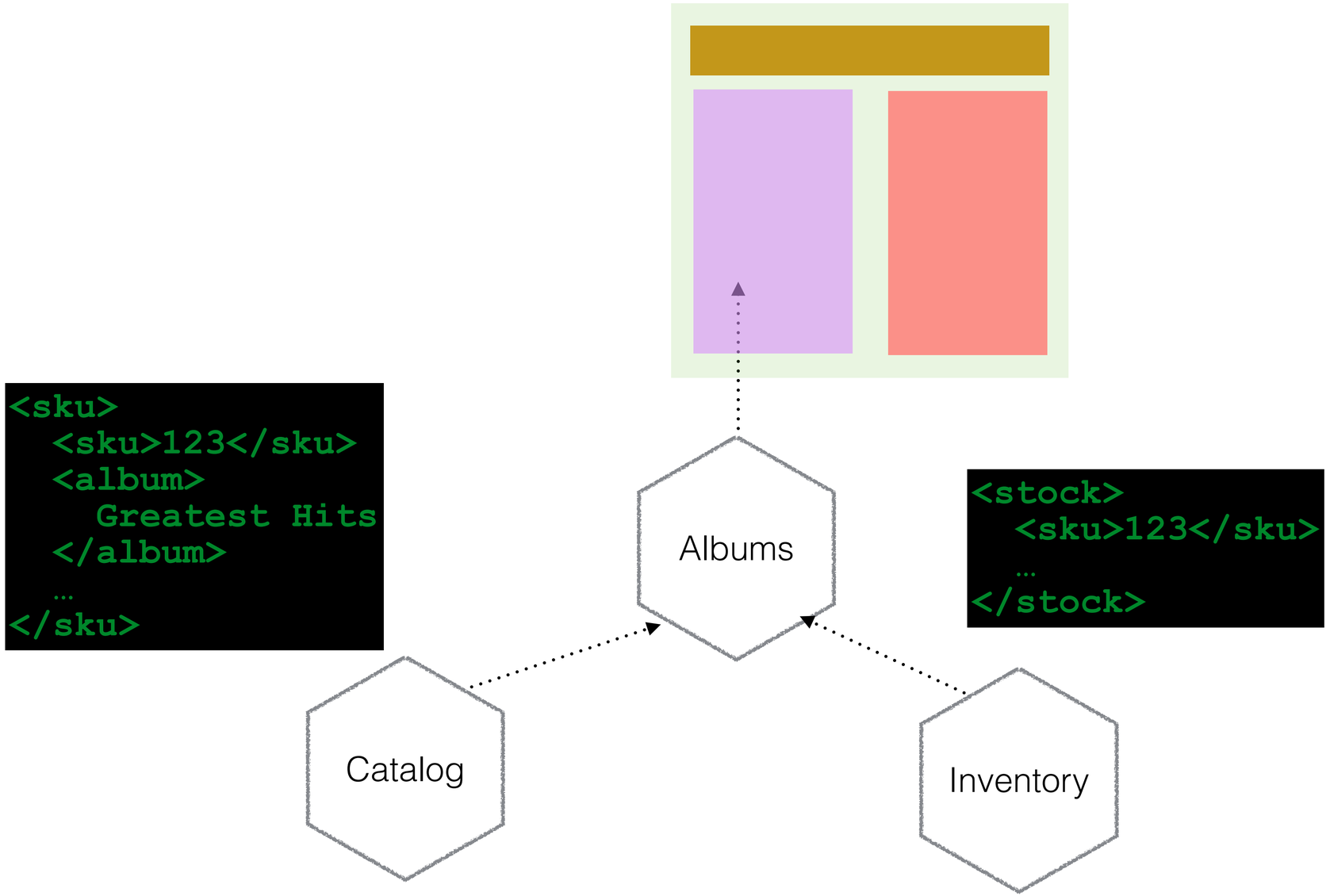
WIDGET-BASED ASSEMBLY - DETAILS



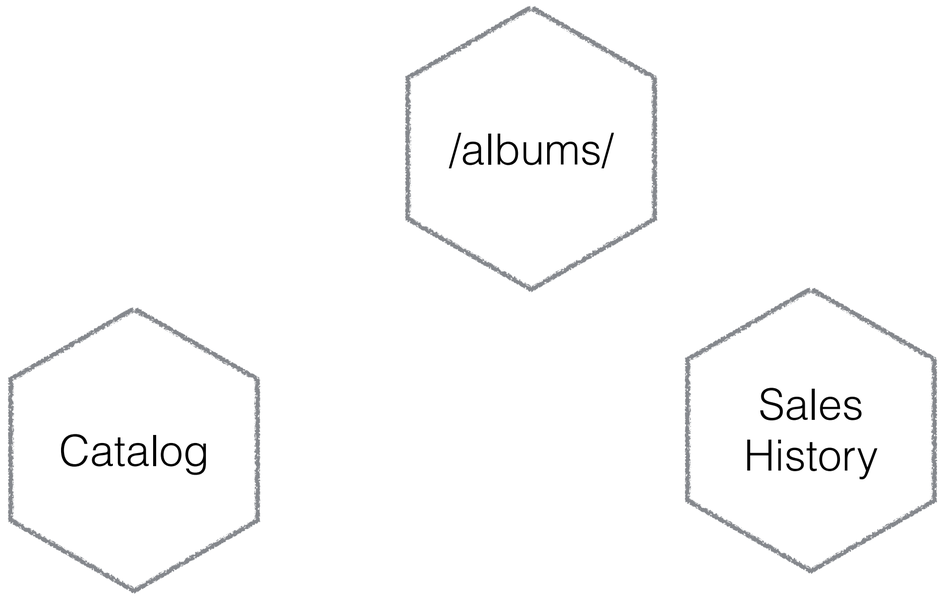
WIDGET-BASED ASSEMBLY - DETAILS



WIDGET-BASED ASSEMBLY - DETAILS

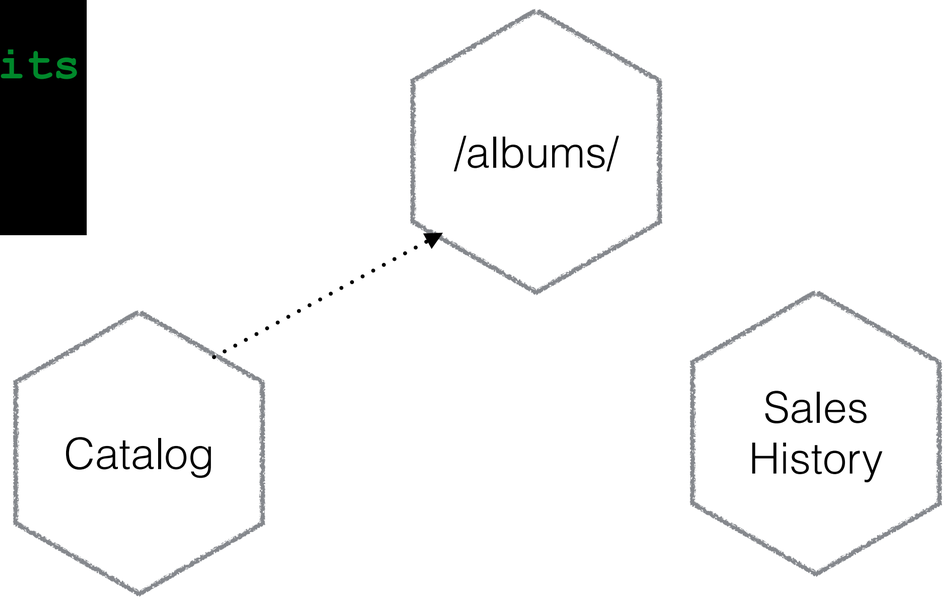


PAGE-BASED ASSEMBLY



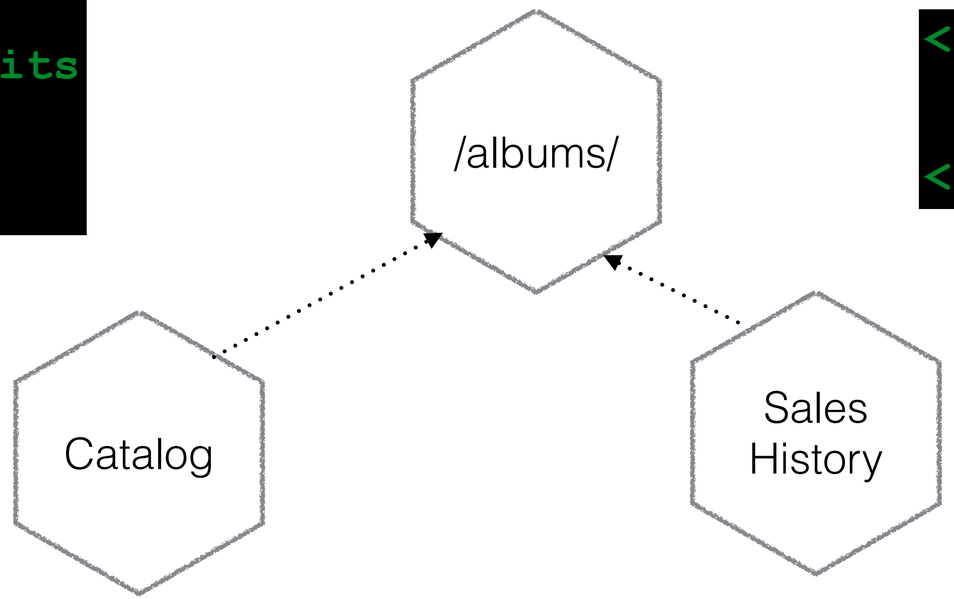
PAGE-BASED ASSEMBLY

```
<sku>
  <sku>123</sku>
  <album>
    Greatest Hits
  </album>
  …
</sku>
```



PAGE-BASED ASSEMBLY

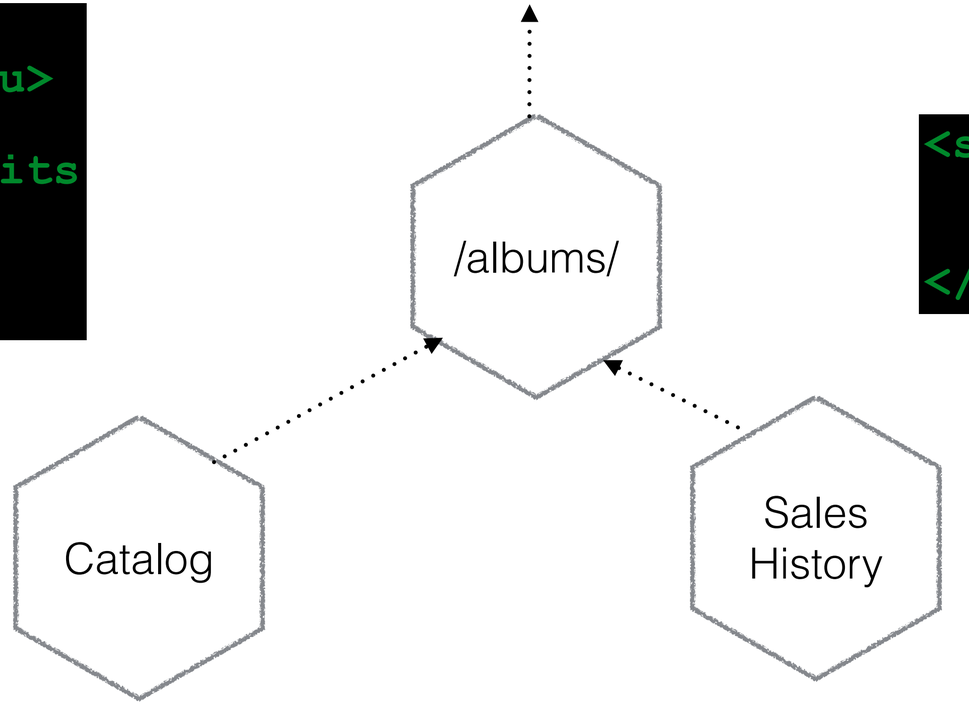
```
<sku>
  <sku>123</sku>
  <album>
    Greatest Hits
  </album>
  …
</sku>
```




```
<sales>
  <sku>123</sku>
  …
</sales>
```

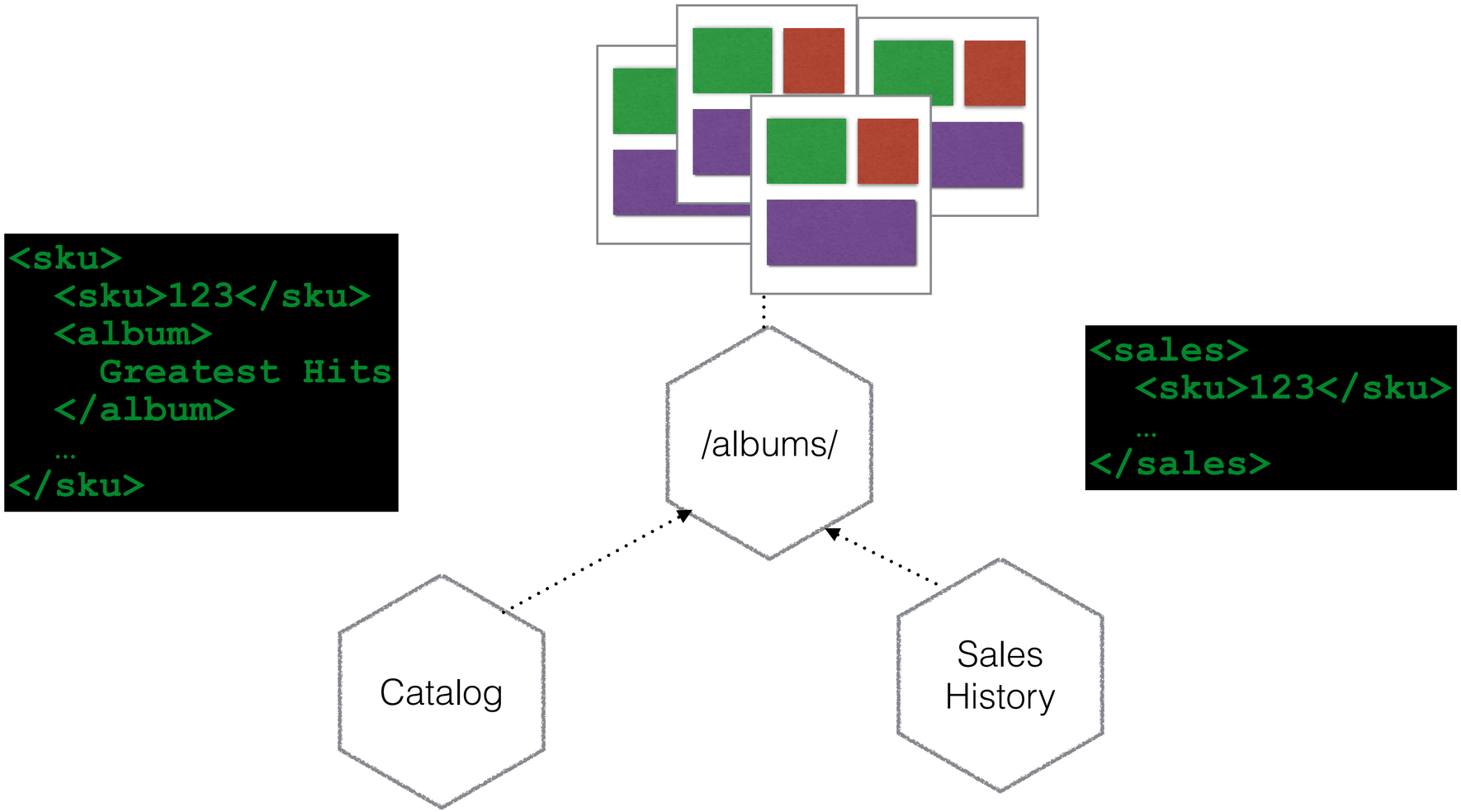
PAGE-BASED ASSEMBLY

```
<sku>
  <sku>123</sku>
  <album>
    Greatest Hits
  </album>
  …
</sku>
```



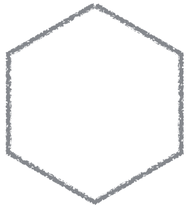

```
<sales>
  <sku>123</sku>
  …
</sales>
```

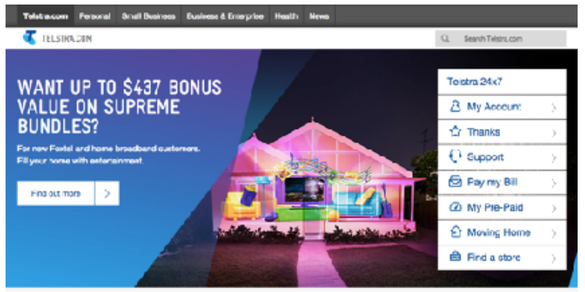
PAGE-BASED ASSEMBLY



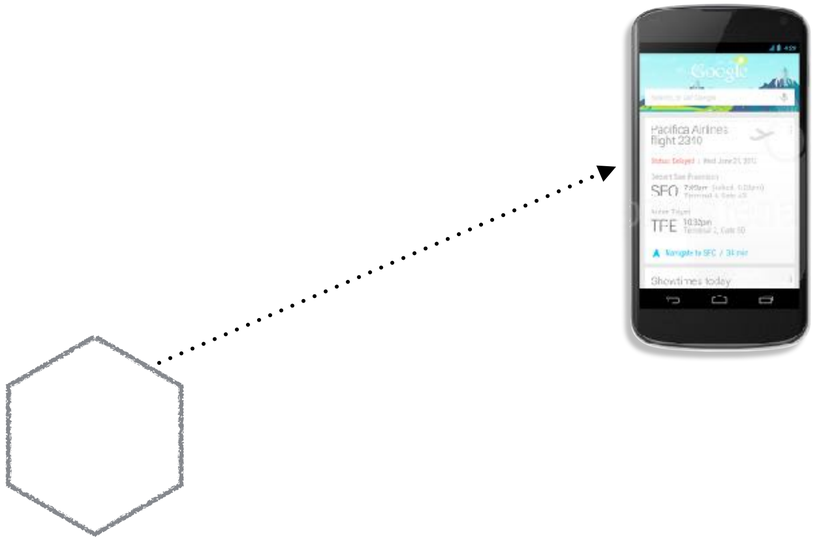
AND MOBILE?

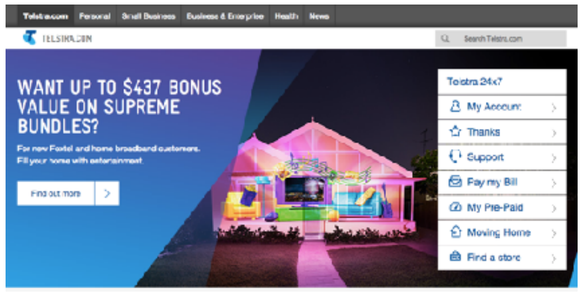

Google
Pacifica Airlines
flight 2340
SFO
TPE
Showtimes today



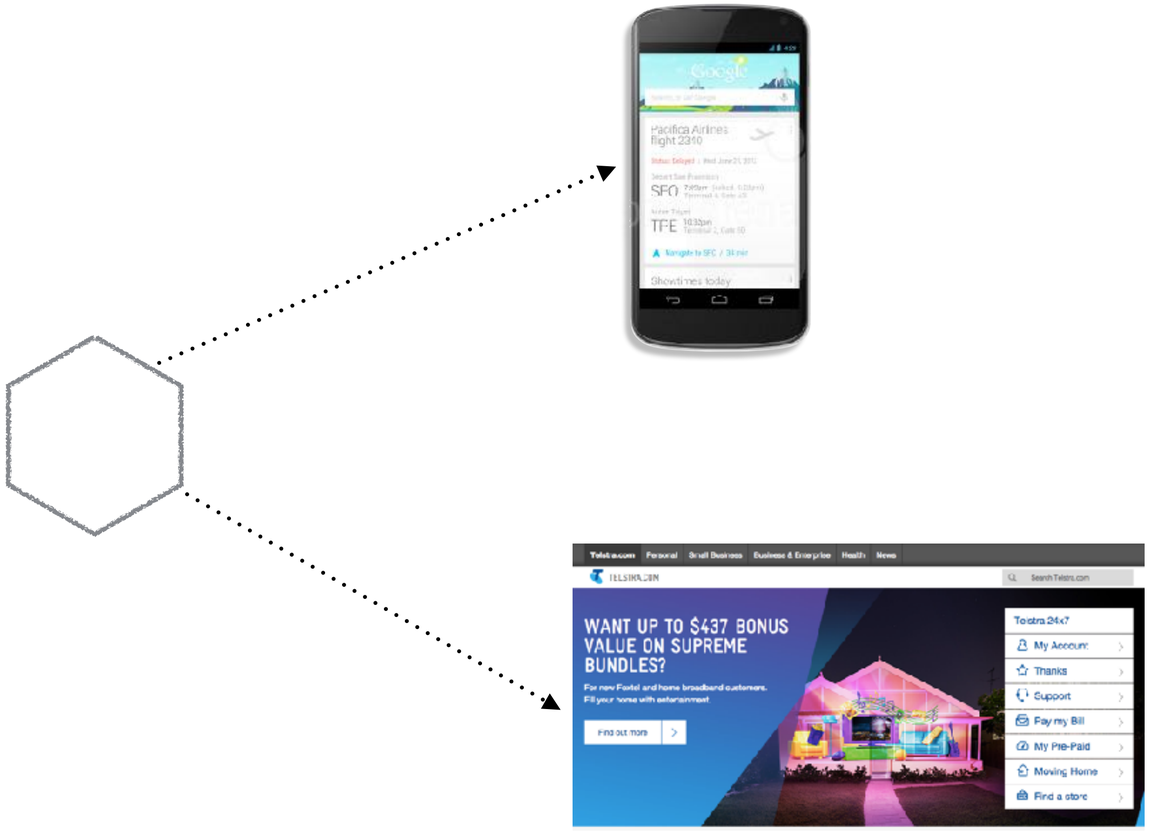

WANT UP TO $437 BONUS VALUE ON SUPREME BUNDLES?
Find out more
Telstra 24x7
My Account
Thanks
Support
Pay my Bill
My Pre-Paid
Moving Home
Find a store

AND MOBILE?

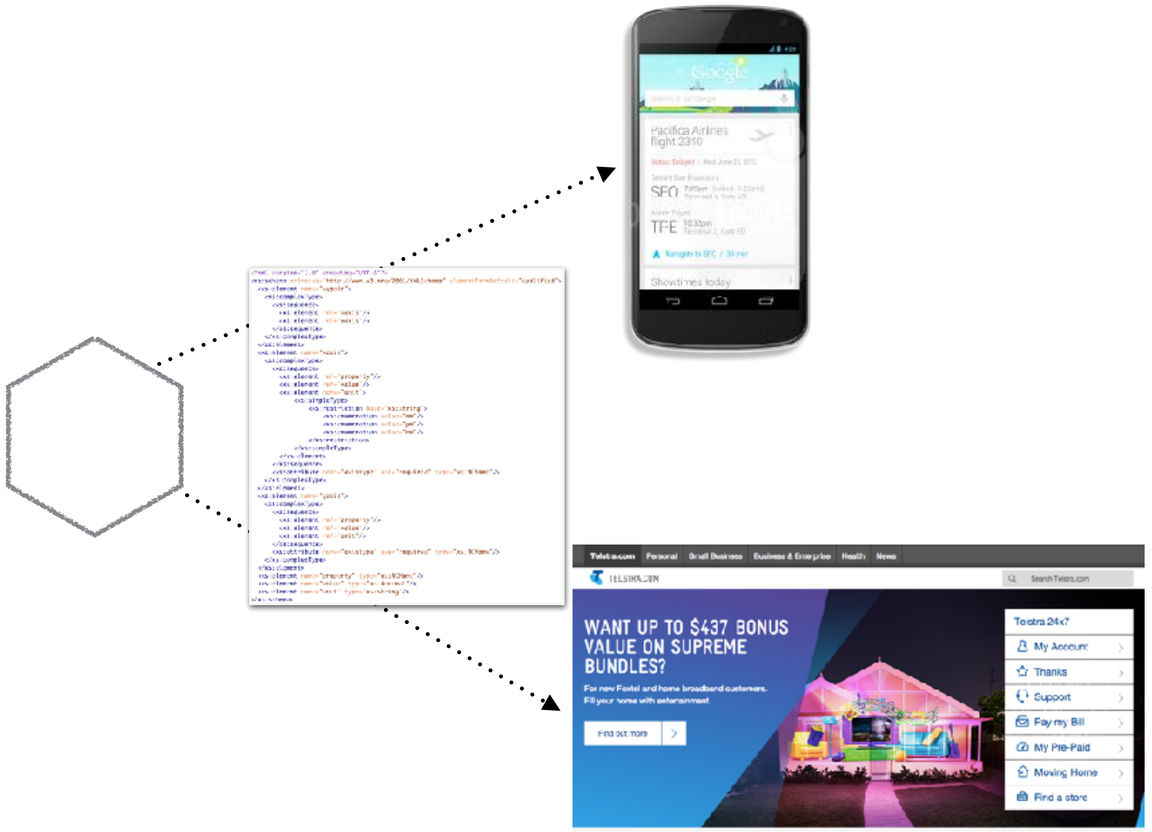

Pacifica Airlines
flight 2340
SFO
TPE
Showtimes today


WANT UP TO $437 BONUS VALUE ON SUPREME BUNDLES?
Find out more
Telstra 24x7
My Account
Thanks
Support
Pay my Bill
My Pre-Paid
Moving Home
Find a store

AND MOBILE?


Pacifica Airlines
flight 2340
SFO
TPE
Showtimes today
WANT UP TO $437 BONUS VALUE ON SUPREME BUNDLES?
Find out more
Telstra 24x7
My Account
Thanks
Support
Pay my Bill
My Pre-Paid
Moving Home
Find a store

AND MOBILE?


Pacifica Airlines
flight 2340
SFO
TPE
WANT UP TO $437 BONUS VALUE ON SUPREME BUNDLES?
Find out more
Telstra 24x7
My Account
Thanks
Support
Pay my Bill
My Pre-Paid
Moving Home
Find a store

CHALLENGING CONSTRAINTS

CHALLENGING CONSTRAINTS

Limited screen real-estate

CHALLENGING CONSTRAINTS

Limited screen real-estate

Fetching data can take longer, and drain battery (and credit!)

CHALLENGING CONSTRAINTS

Limited screen real-estate

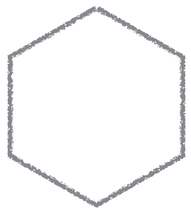
Fetching data can take longer, and drain battery (and credit!)

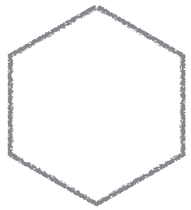
Use-cases can differ greatly

GENERAL-PURPOSE AGGREGATING GATEWAY

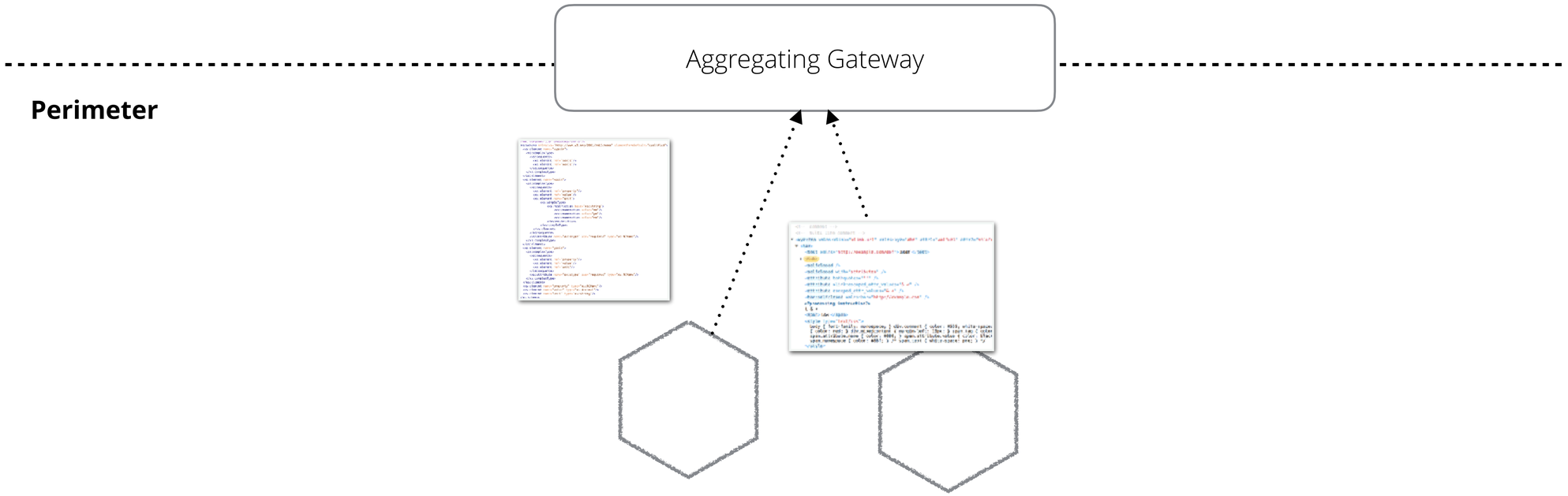
Aggregating Gateway

Perimeter

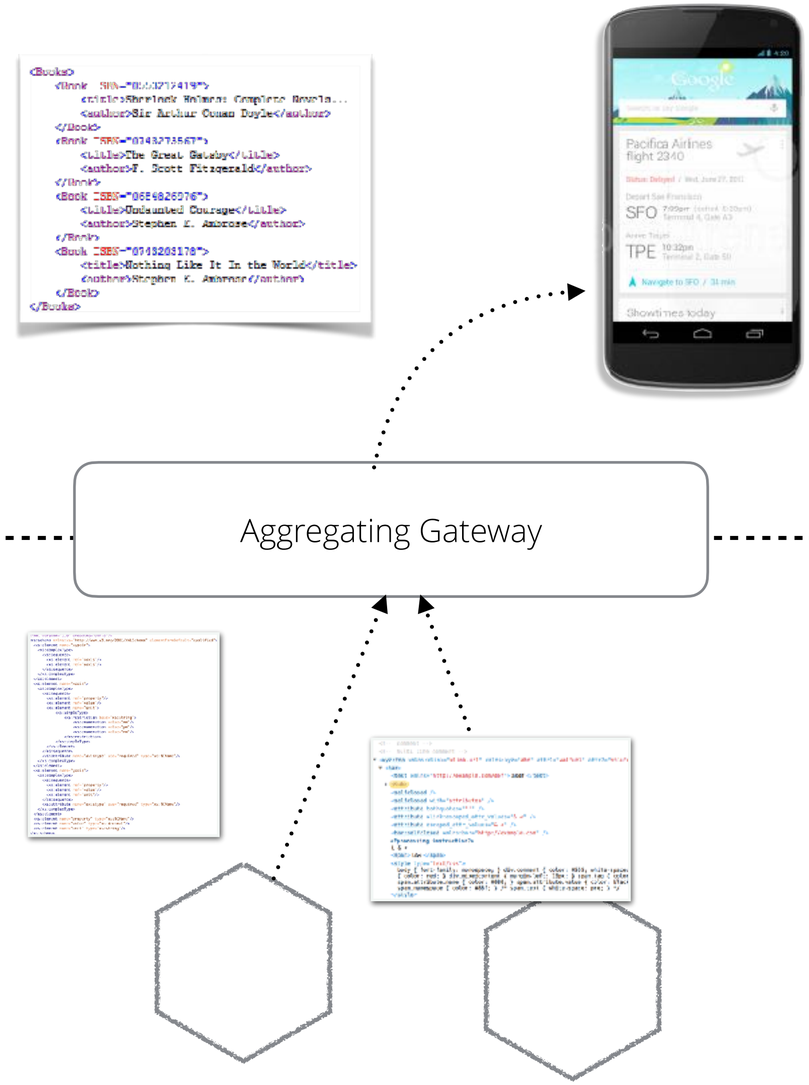




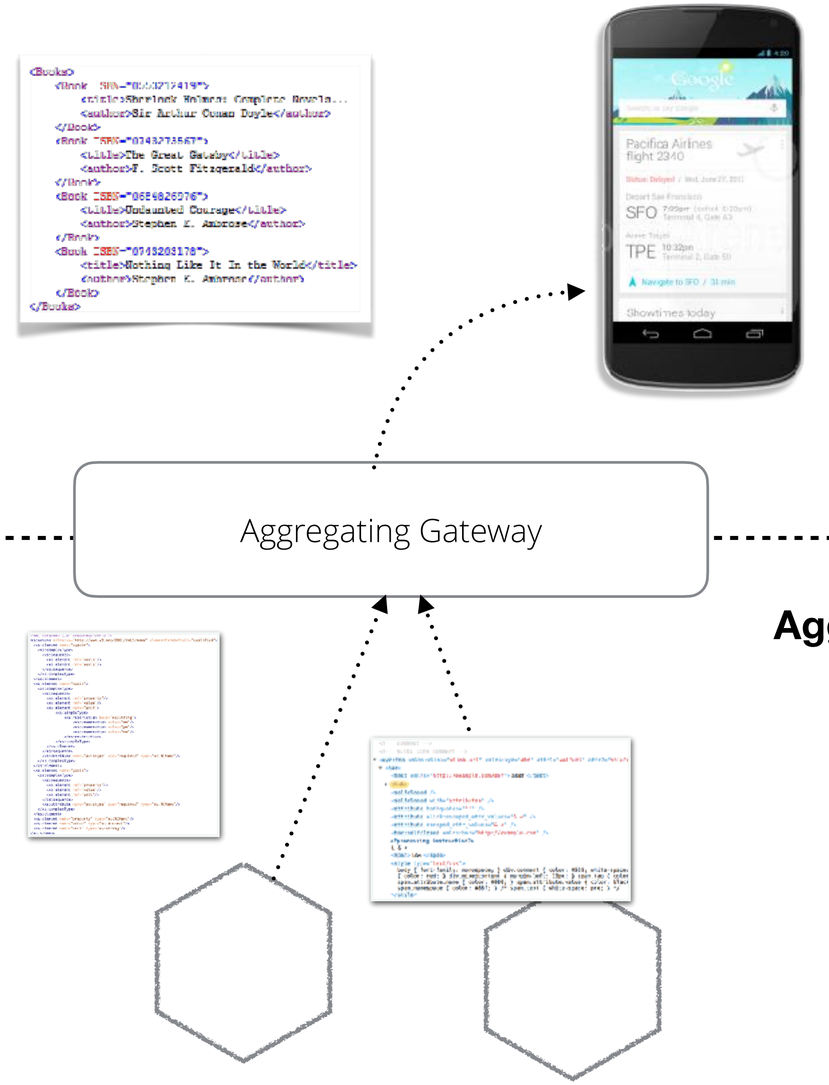
GENERAL-PURPOSE AGGREGATING GATEWAY



GENERAL-PURPOSE AGGREGATING GATEWAY

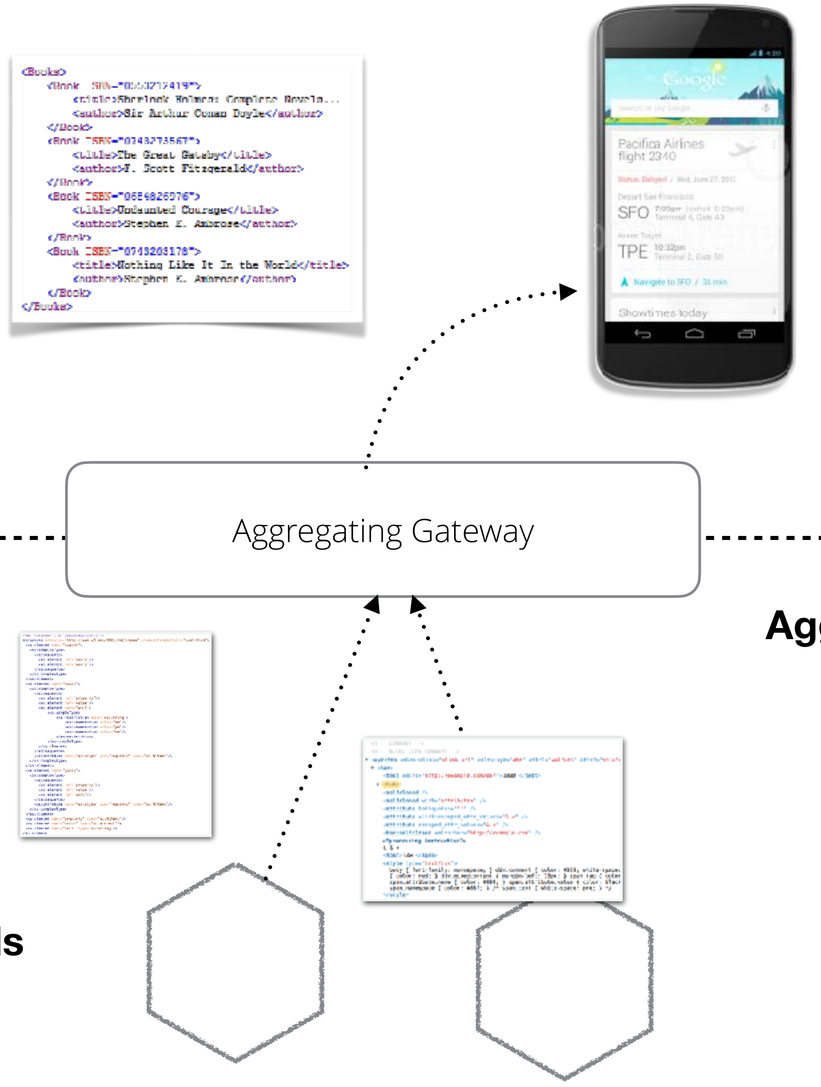


GENERAL-PURPOSE AGGREGATING GATEWAY

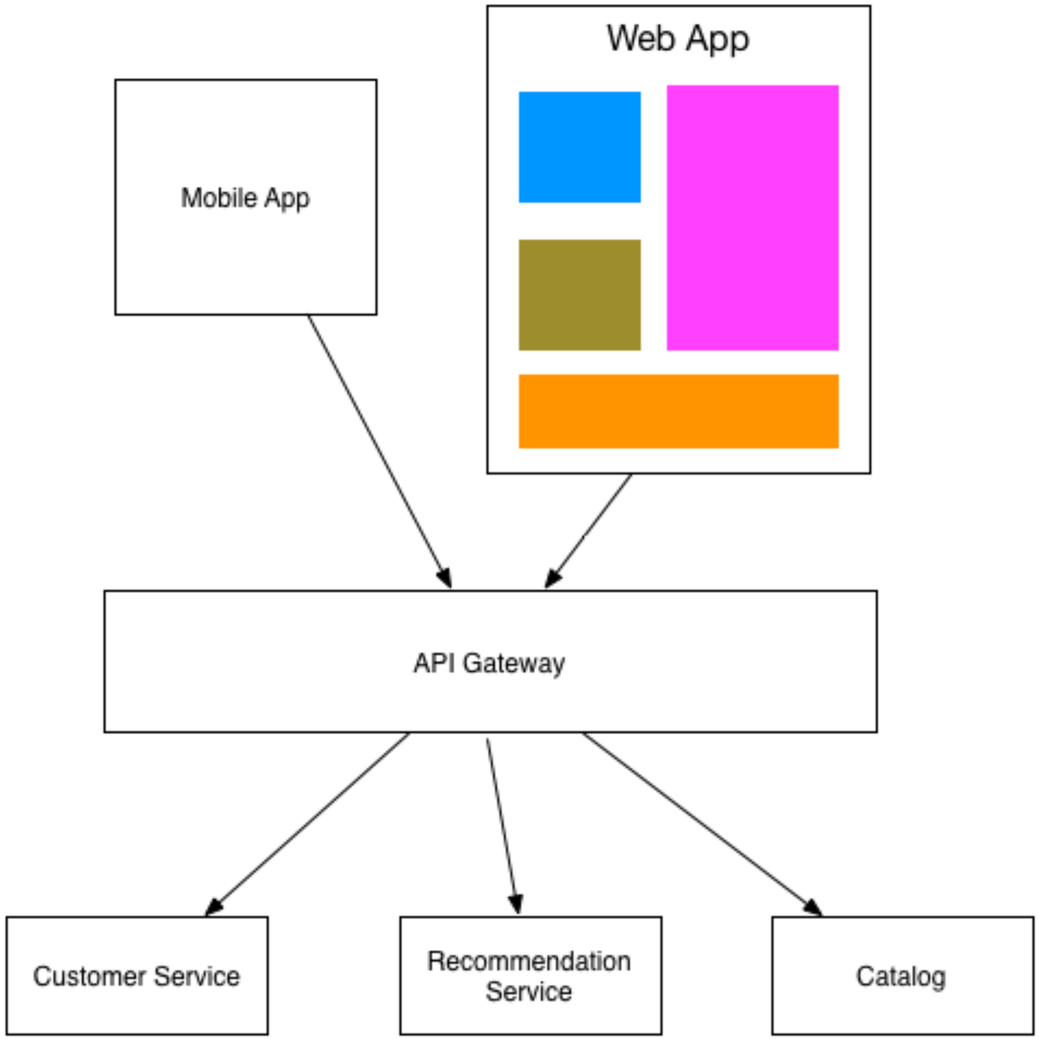


Aggregating gateway strips out unnecessary data

GENERAL-PURPOSE AGGREGATING GATEWAY



API GATEWAY


Web App
Mobile App
API Gateway
Customer Service
Recommendation Service
Catalog

API GATEWAY - COMPETING CONCERNS

API GATEWAY - COMPETING CONCERNS

Authorization and authentication

API GATEWAY - COMPETING CONCERNS

Authorization and authentication

API-rate limiting

API GATEWAY - COMPETING CONCERNS

Authorization and authentication

API-rate limiting

Request logging

API GATEWAY - COMPETING CONCERNS

Authorization and authentication

API-rate limiting

Request logging

Service discovery

API GATEWAY - COMPETING CONCERNS

Authorization and authentication

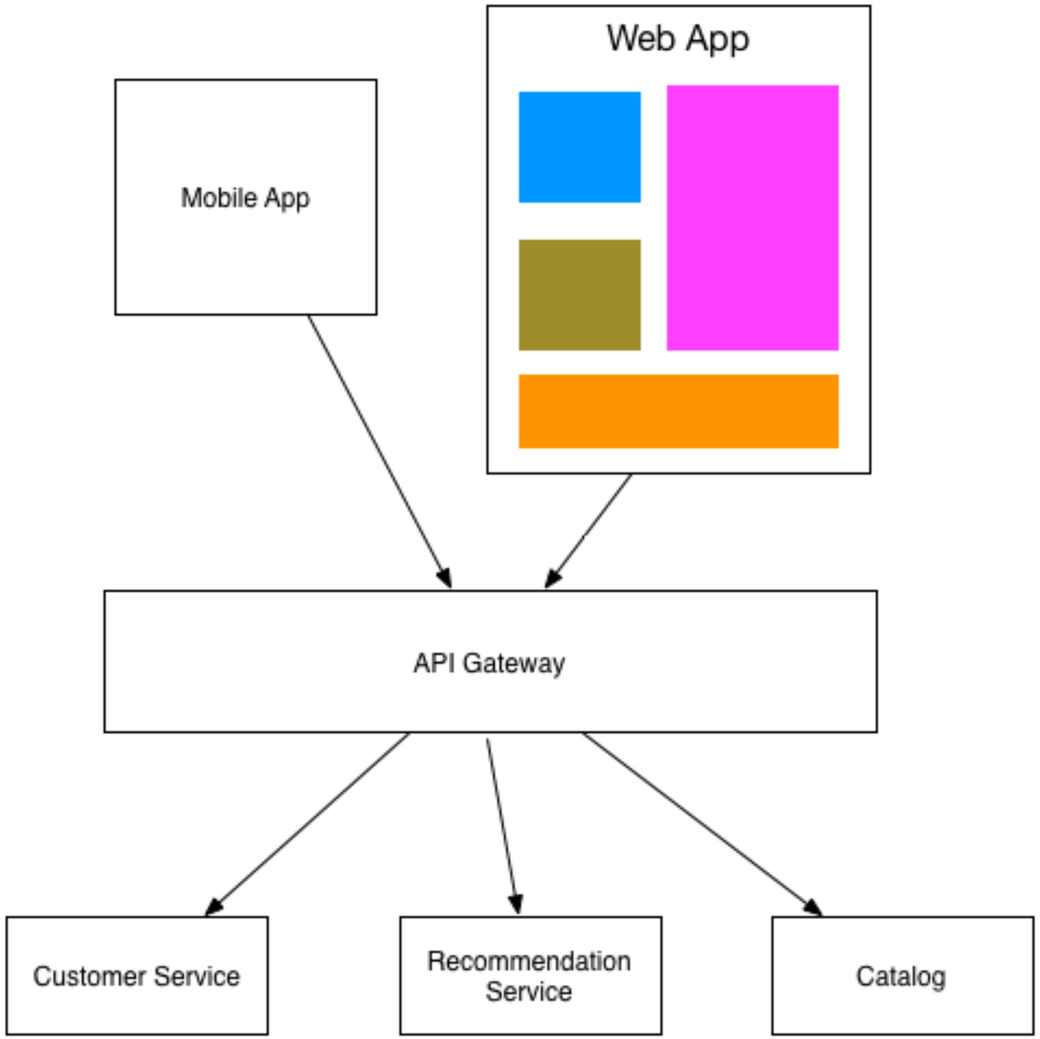
API-rate limiting

Request logging

Service discovery

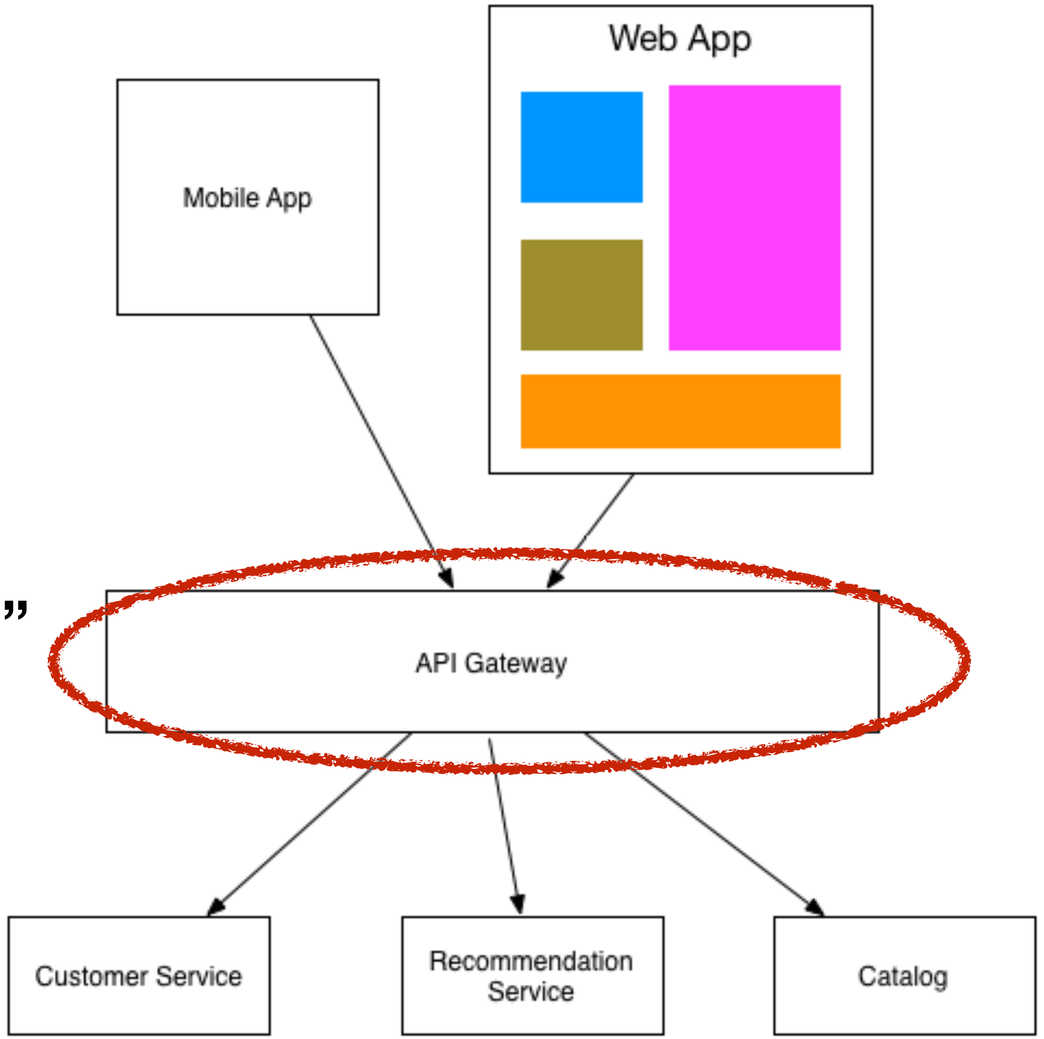
Call aggregation

API GATEWAY

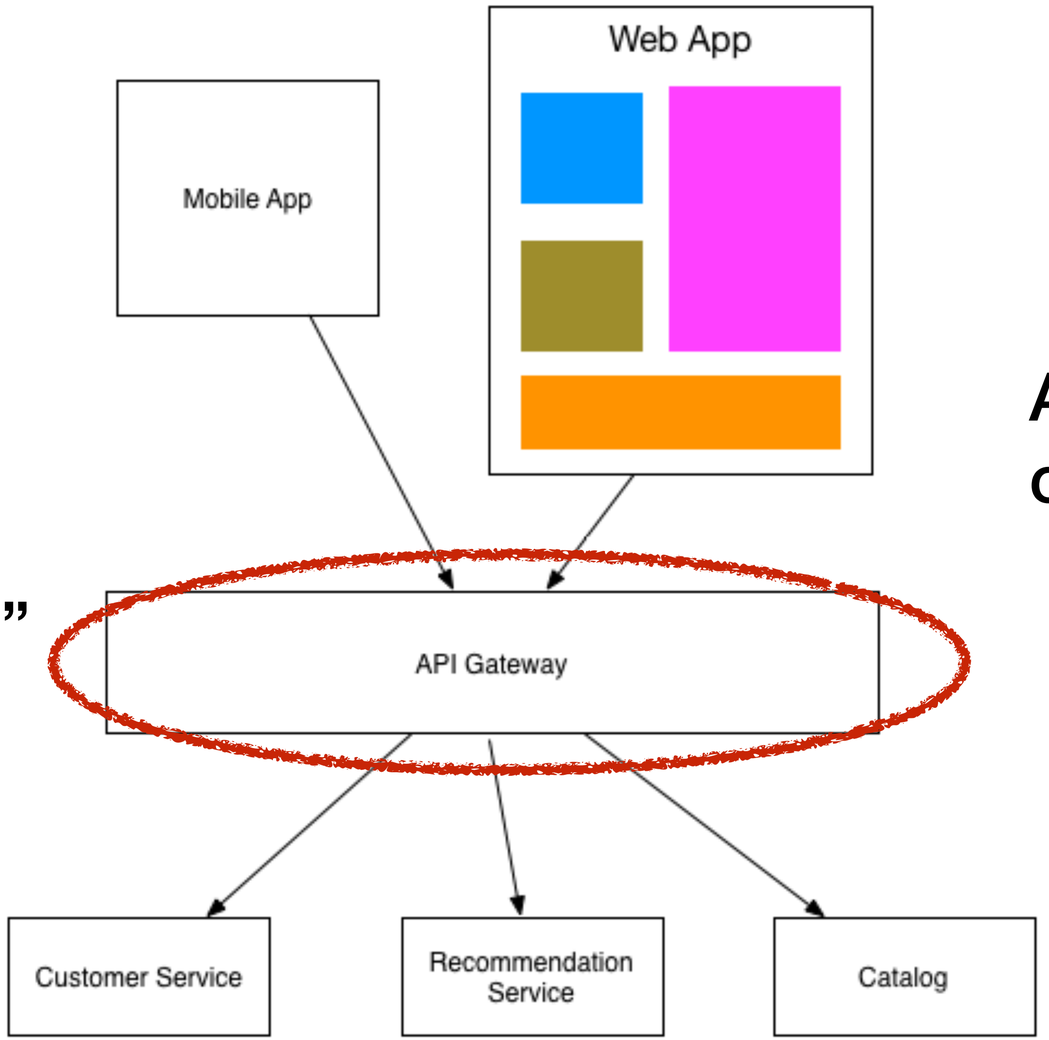

Web App
Mobile App
API Gateway
Customer Service
Recommendation Service
Catalog

API GATEWAY

Can become “bloated”



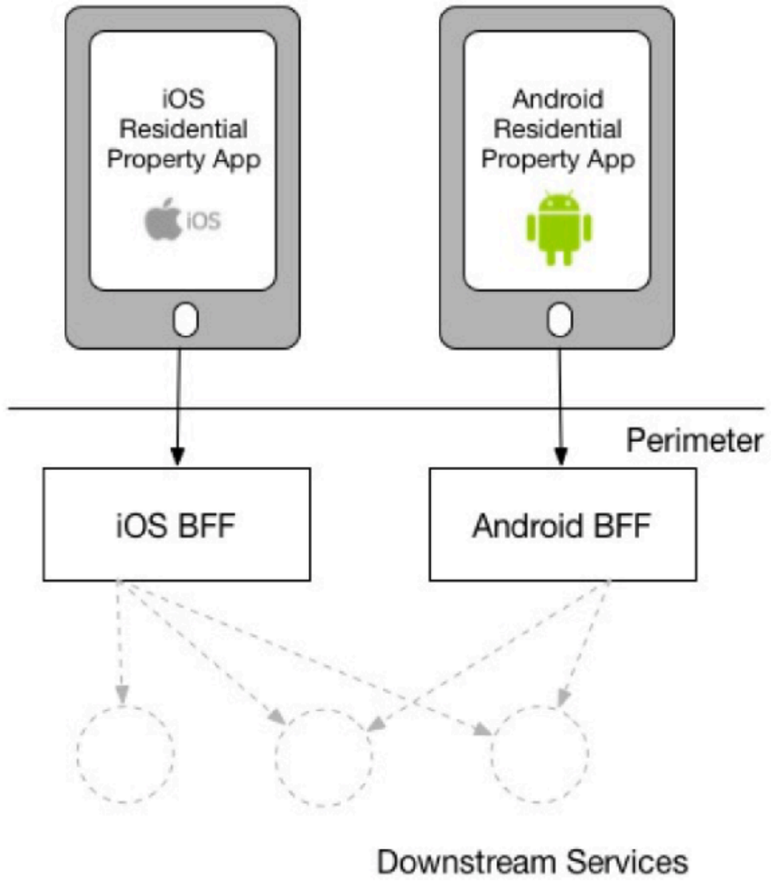
API GATEWAY



Can become “bloated”

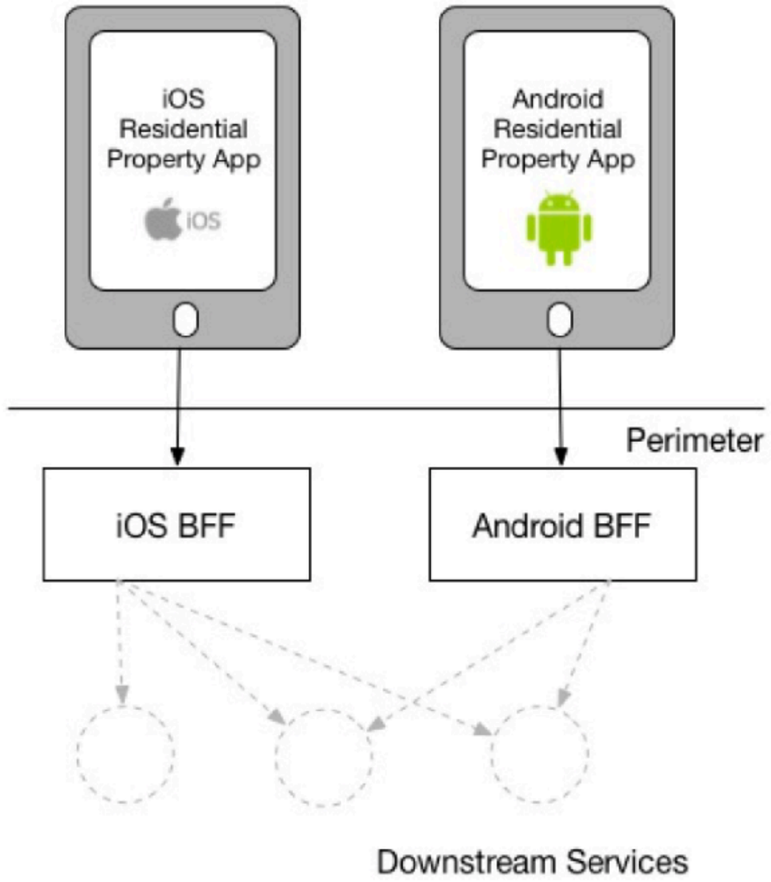
A single point of contention

BACKENDS FOR FRONTENDS



http://samnewman.io/patterns/architectural/bff/

BACKENDS FOR FRONTENDS



http://samnewman.io/patterns/architectural/bff/

Pattern: Backends For Frontends

Written on Nov 18 2015

Single-purpose Edge Services for UIs and external parties

Introduction

With the advent and success of the web, the de facto way of delivering user interfaces has shifted from thick-client applications to interfaces delivered via the web, a trend that has also enabled the growth of SAAS-based solutions in general. The benefits of delivering a user interface over the web were huge - primarily as the cost of releasing new functionality was significantly reduced as the cost of client-side installs was (in most cases) eliminated altogether.

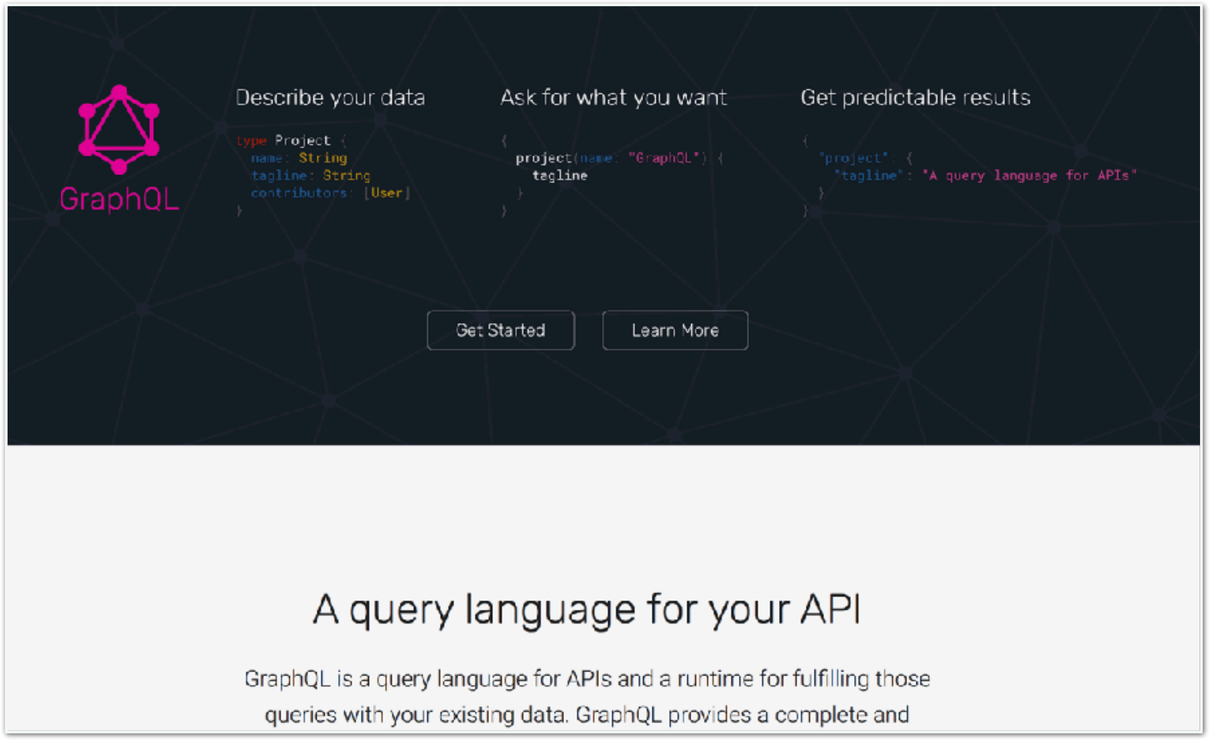
This simpler world didn't last long though, as the age of the mobile followed shortly afterwards. Now we had a problem. We had server-side functionality which we wanted to expose both via our desktop web UI, and via one or more mobile UIs. With a system that had initially been developed with a desktop-web UI in mind, we often faced a problem in accommodating these new types of user interface, often as we already had a tight coupling between the desktop web UI and our backed services.

The General-Purpose API Backend

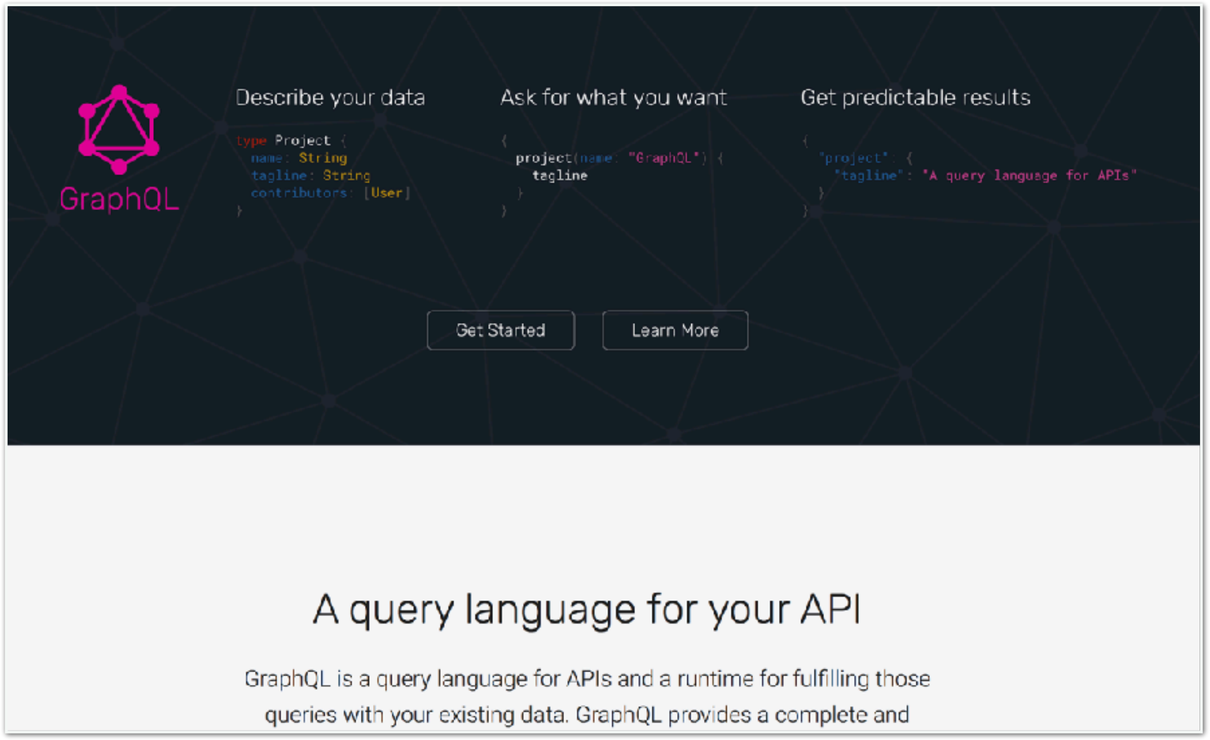
A first step in accommodating more than one type of UI is normally

http://samnewman.io/patterns/architectural/bff/

GRAPHQL


GraphQL
Describe your data
type Project {
name: String
tagline: String
contributors: [User]
}
Ask for what you want
{
project(name: "GraphQL") {
tagline
}
}
Get predictable results
{
"project": {
"tagline": "A query language for APIs"
}
}
Get Started
Learn More
A query language for your API
GraphQL is a query language for APIs and a runtime for fulfilling those queries with your existing data. GraphQL provides a complete and

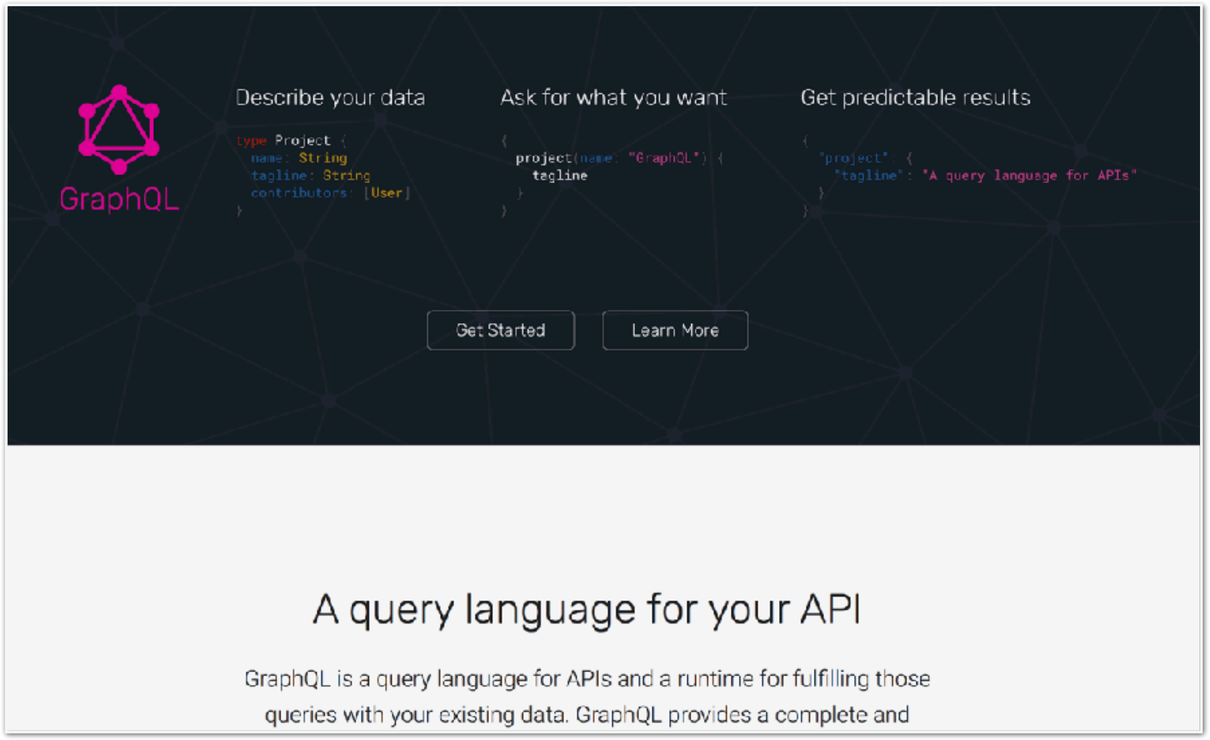
GRAPHQL




```
{
  hero {
    name
    # Queries can have comments!
    friends {
      name
    }
  }
}
```

```
{
  "data": {
    "hero": {
      "name": "R2-D2",
      "friends": [
        {
          "name": "Luke Skywalker"
        },
        {
          "name": "Han Solo"
        },
```

GRAPHQL

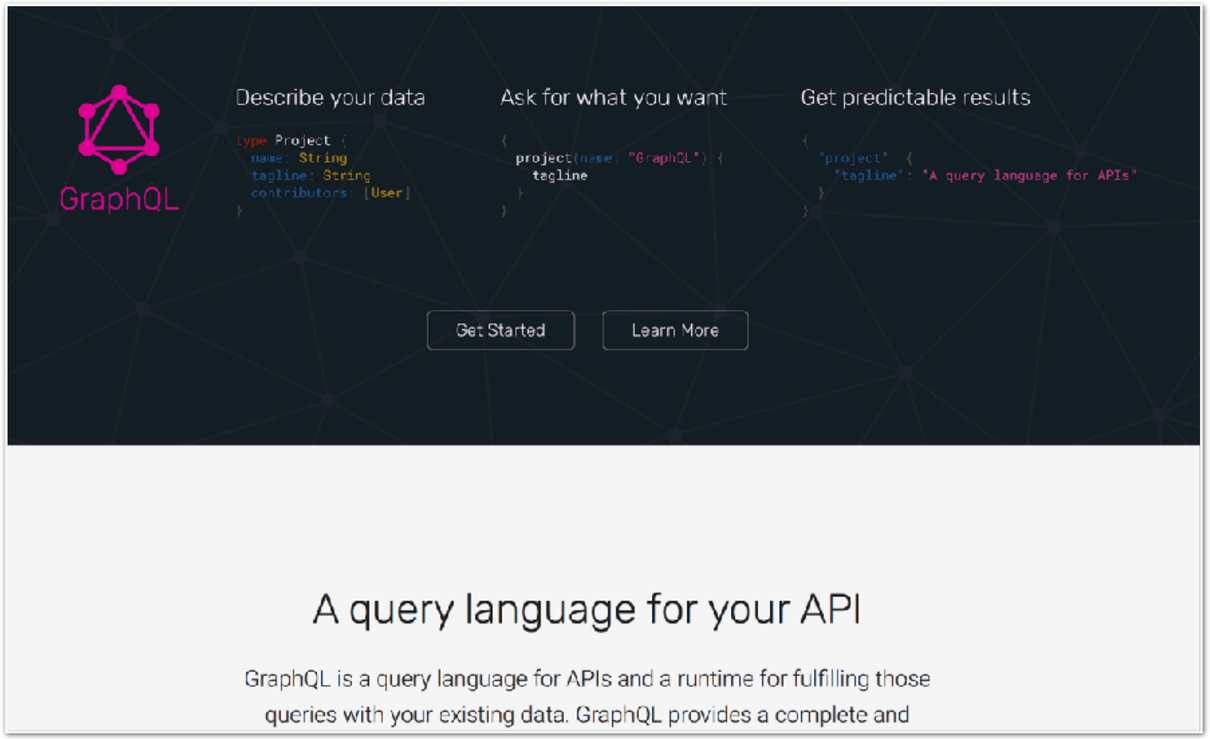



```
{
  hero {
    name
    # Queries can have comments!
    friends {
      name
    }
  }
}
```

```
{
  "data": {
    "hero": {
      "name": "R2-D2",
      "friends": [
        {
          "name": "Luke Skywalker"
        },
        {
          "name": "Han Solo"
        },
```

Allows the client to specify queries dynamically

GRAPHQL




```
{
  hero {
    name
    # Queries can have comments!
    friends {
      name
    }
  }
}
```

```
{
  "data": {
    "hero": {
      "name": "R2-D2",
      "friends": [
        {
          "name": "Luke Skywalker"
        },
        {
          "name": "Han Solo"
        },
```

Allows the client to specify queries dynamically

These queries can span microservices

Versioning?

GRAPHQL CAVEATS

Versioning?

Caching

GRAPHQL CAVEATS

Versioning?

Caching

Designed for the perimeter

GRAPHQL CAVEATS

Versioning?

Caching

Designed for the perimeter

Expensive queries

MICRO FRONTENDS

Micro Frontends

Good frontend development is hard. Scaling frontend development so that many teams can work simultaneously on a large and complex product is even harder. In this article we'll describe a recent trend of breaking up frontend monoliths into many smaller, more manageable pieces, and how this architecture can increase the effectiveness and efficiency of teams working on frontend code. As well as talking about the various benefits and costs, we'll cover some of the implementation options that are available, and we'll dive deep into a full example application that demonstrates the technique.

19 June 2019

Cam Jackson

Cam Jackson is a full-stack web developer and consultant at ThoughtWorks, with a particular interest in how large organisations scale their frontend development process and practices. He has worked with clients across multiple

CONTENTS

https://martinfowler.com/articles/micro-frontends.html

MICRO FRONTENDS

Micro Frontends

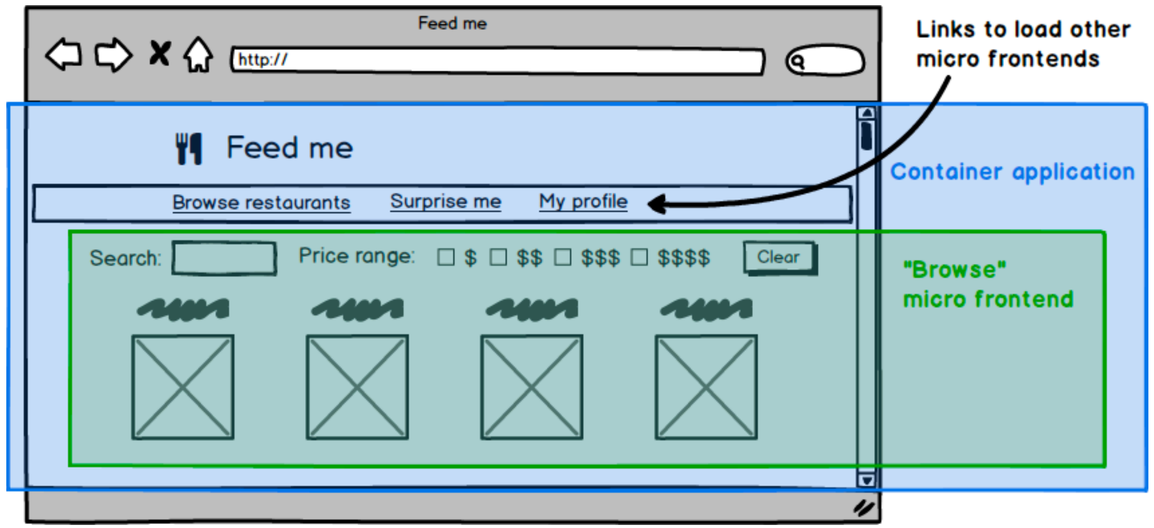
Good frontend development is hard. Scaling frontend development so that many teams can work simultaneously on a large and complex product is even harder. In this article we'll describe a recent trend of breaking up frontend monoliths into many smaller, more manageable pieces, and how this architecture can increase the effectiveness and efficiency of teams working on frontend code. As well as talking about the various benefits and costs, we'll cover some of the implementation options that are available, and we'll dive deep into a full example application that demonstrates the technique.

19 June 2019

Cam Jackson

Cam Jackson is a full-stack web developer and consultant at ThoughtWorks, with a particular interest in how large organisations scale their frontend development process and practices. He has worked with clients across multiple

CONTENTS



https://martinfowler.com/articles/micro-frontends.html

IN SUMMARY

IN SUMMARY

Adopt an incremental approach to decomposition

IN SUMMARY

Adopt an incremental approach to decomposition

Have a clear understanding as to what you're trying to achieve

IN SUMMARY

Adopt an incremental approach to decomposition

Have a clear understanding as to what you're trying to achieve

Even small changes can be broken up into smaller releases

IN SUMMARY

Adopt an incremental approach to decomposition

Have a clear understanding as to what you're trying to achieve

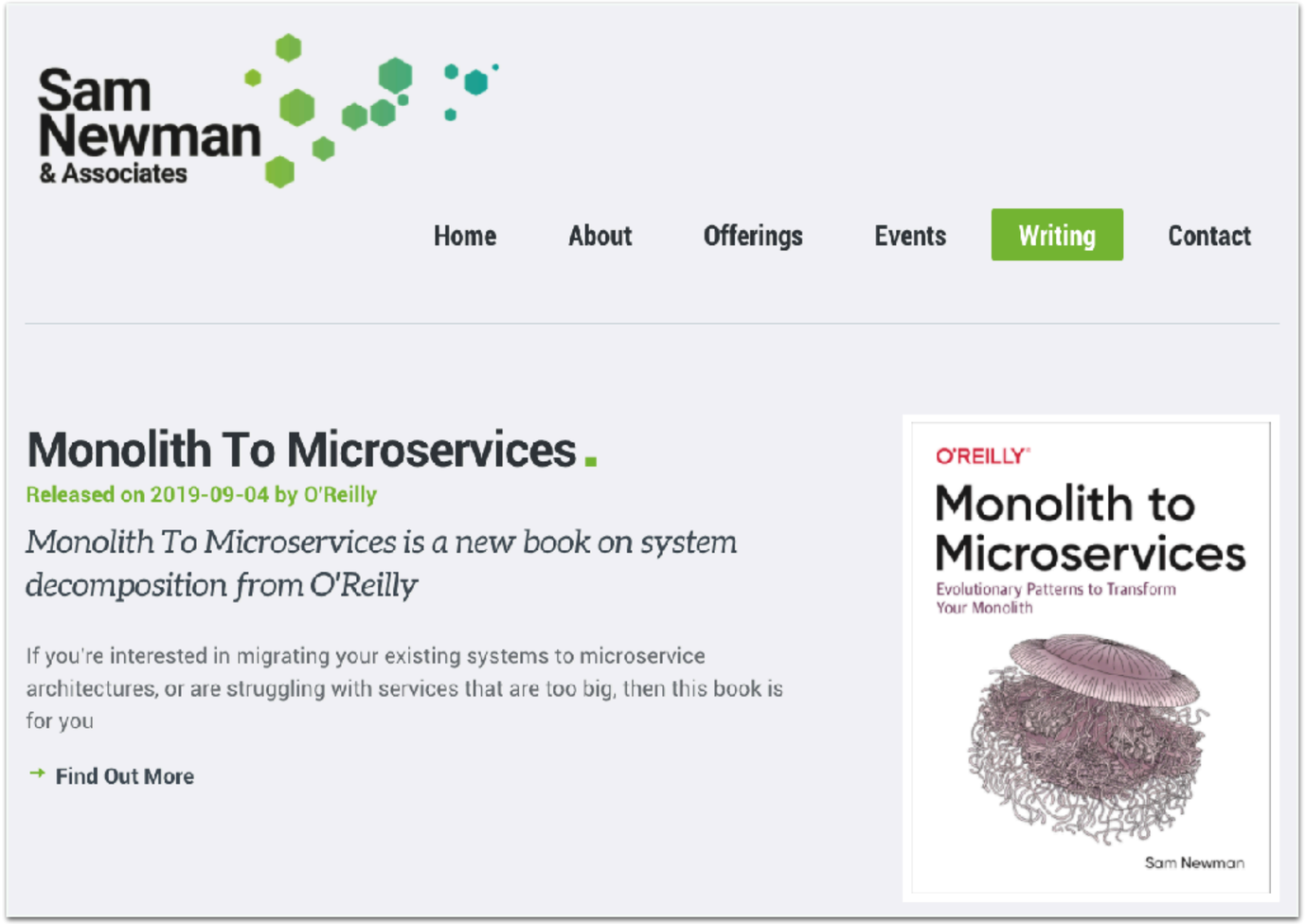
Even small changes can be broken up into smaller releases

Decomposition doesn't need to stop at the backend

OTHER COURSES

http://bit.ly/snewman-olt

THANKS!



Sam
Newman
& Associates

https://samnewman.io/