



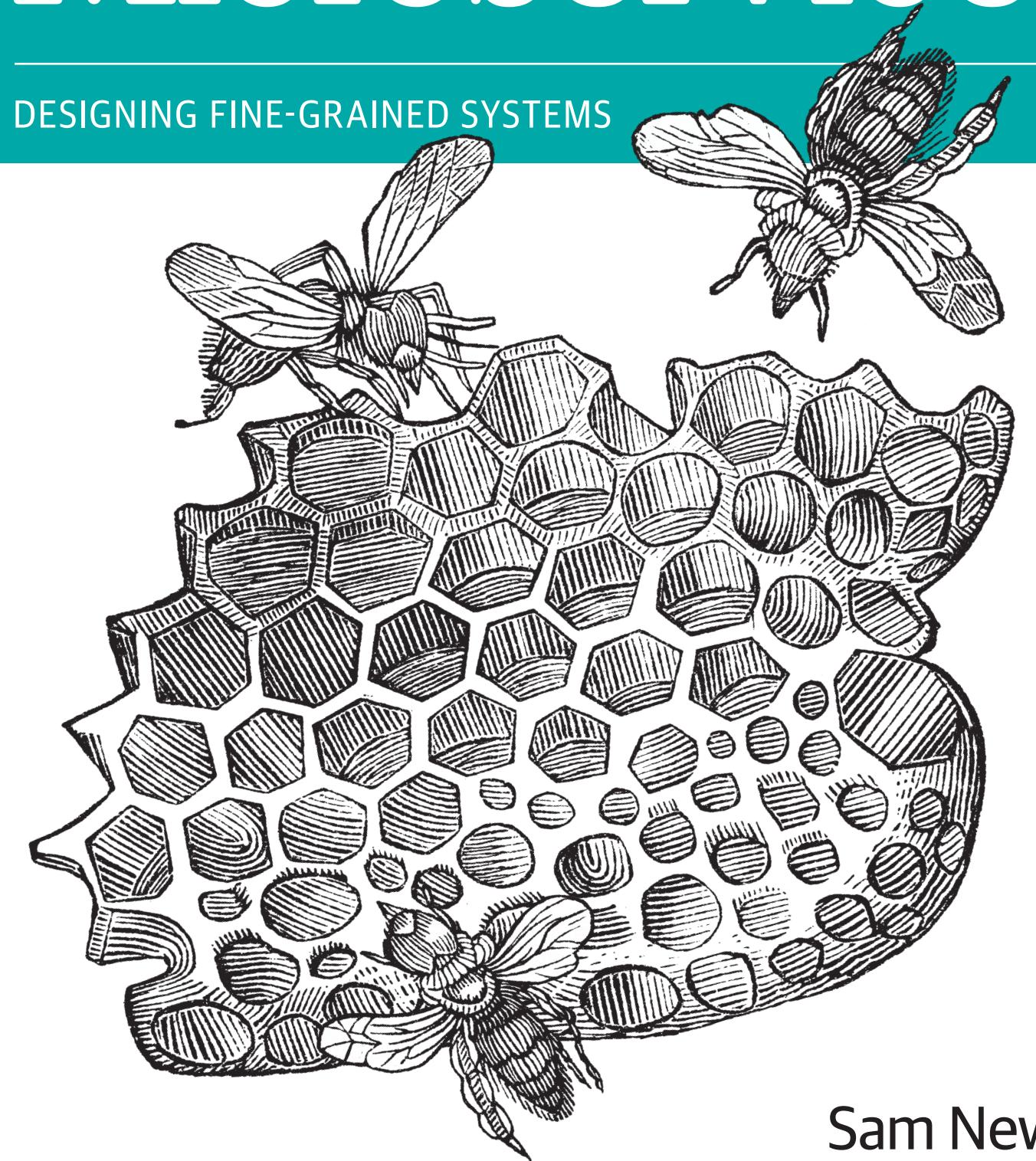
MONOLITH TO MICROSERVICES APPLICATION DECOMPOSITION PATTERNS

Sam Newman

O'REILLY®

Building Microservices

DESIGNING FINE-GRAINED SYSTEMS

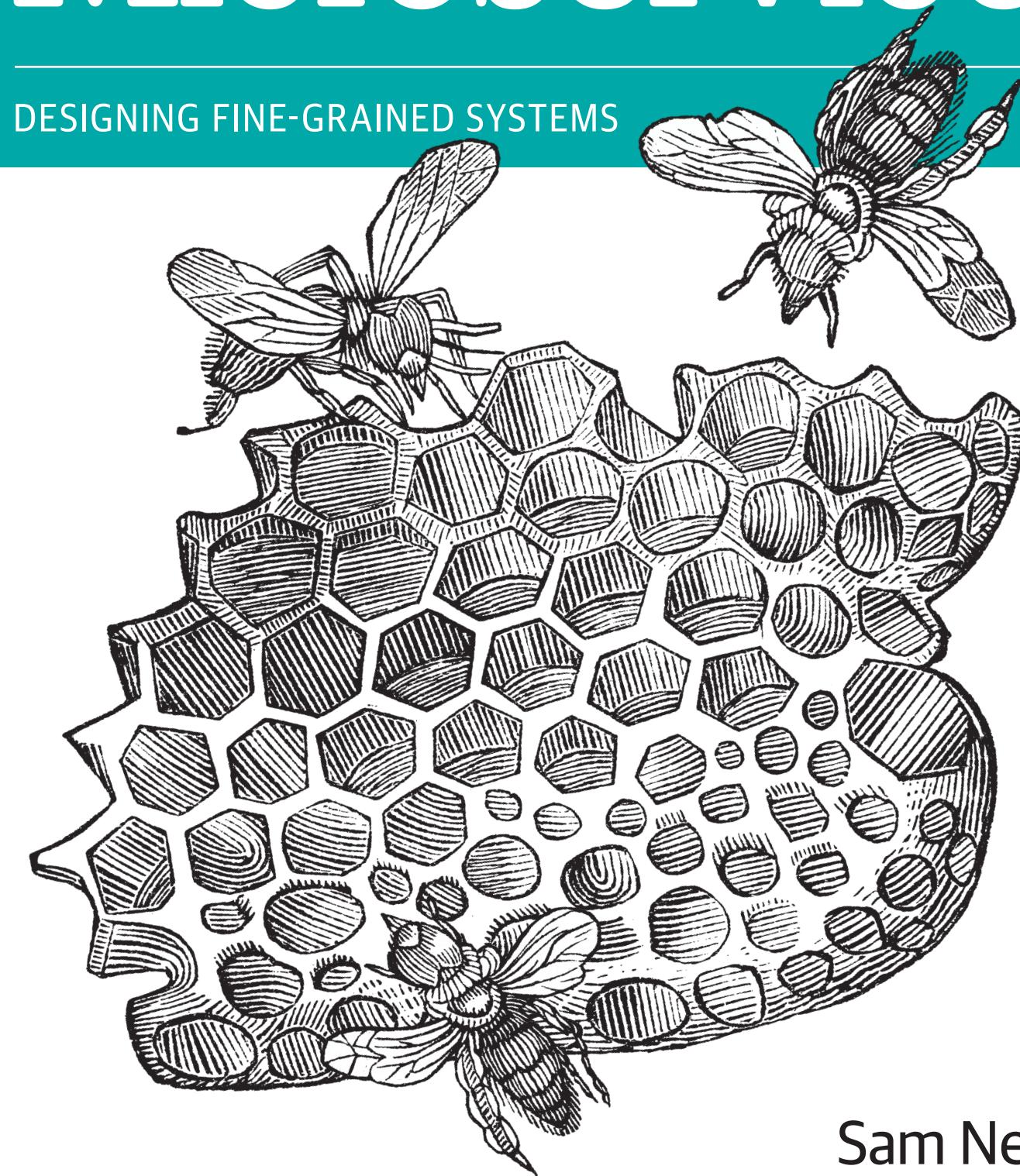


Sam Newman

O'REILLY®

Building Microservices

DESIGNING FINE-GRAINED SYSTEMS



Sam Newman

**Sam
Newman
& Associates**



@samnewman

NEW BOOK!

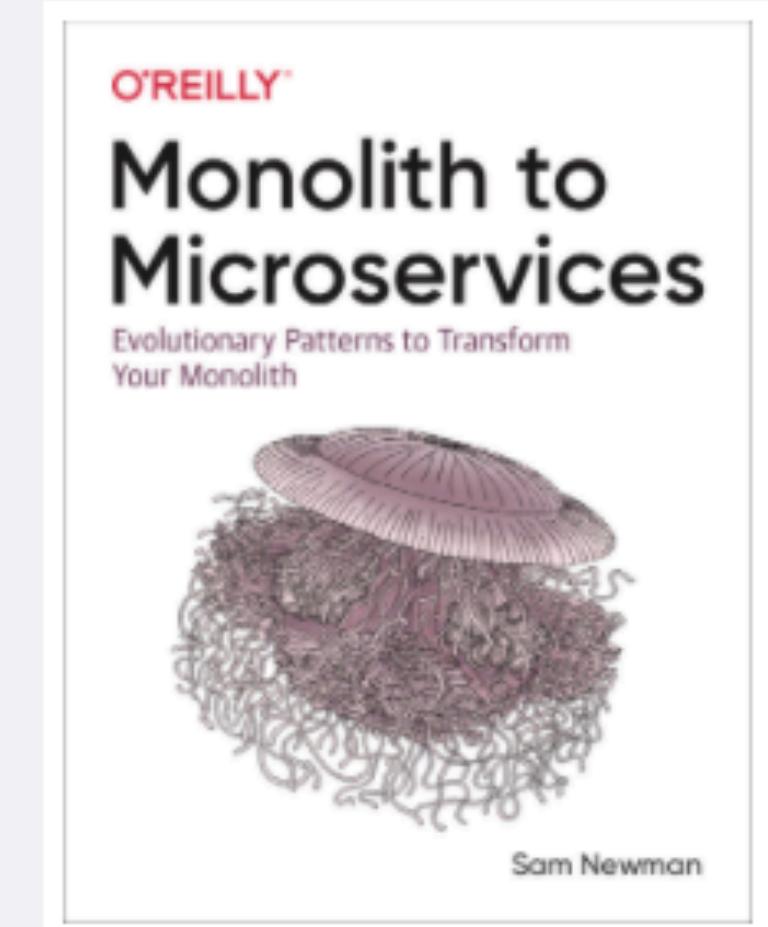
Monolith To Microservices.

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 monolithic 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



MICROSERVICE FUNDAMENTALS

Sam Newman

<https://www.flickr.com/photos/snowpeak/42068450985/>

<http://bit.ly/snewman-olt>

@samnewman

OTHER COURSES



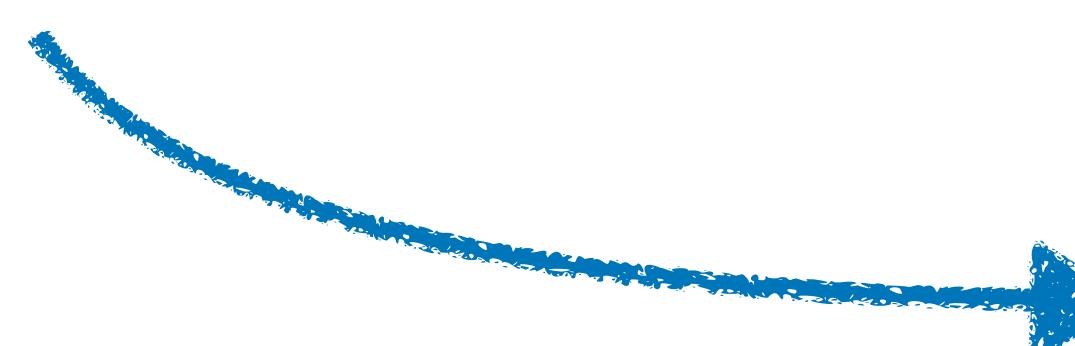
<http://bit.ly/snewman-olt>

@samnewman

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

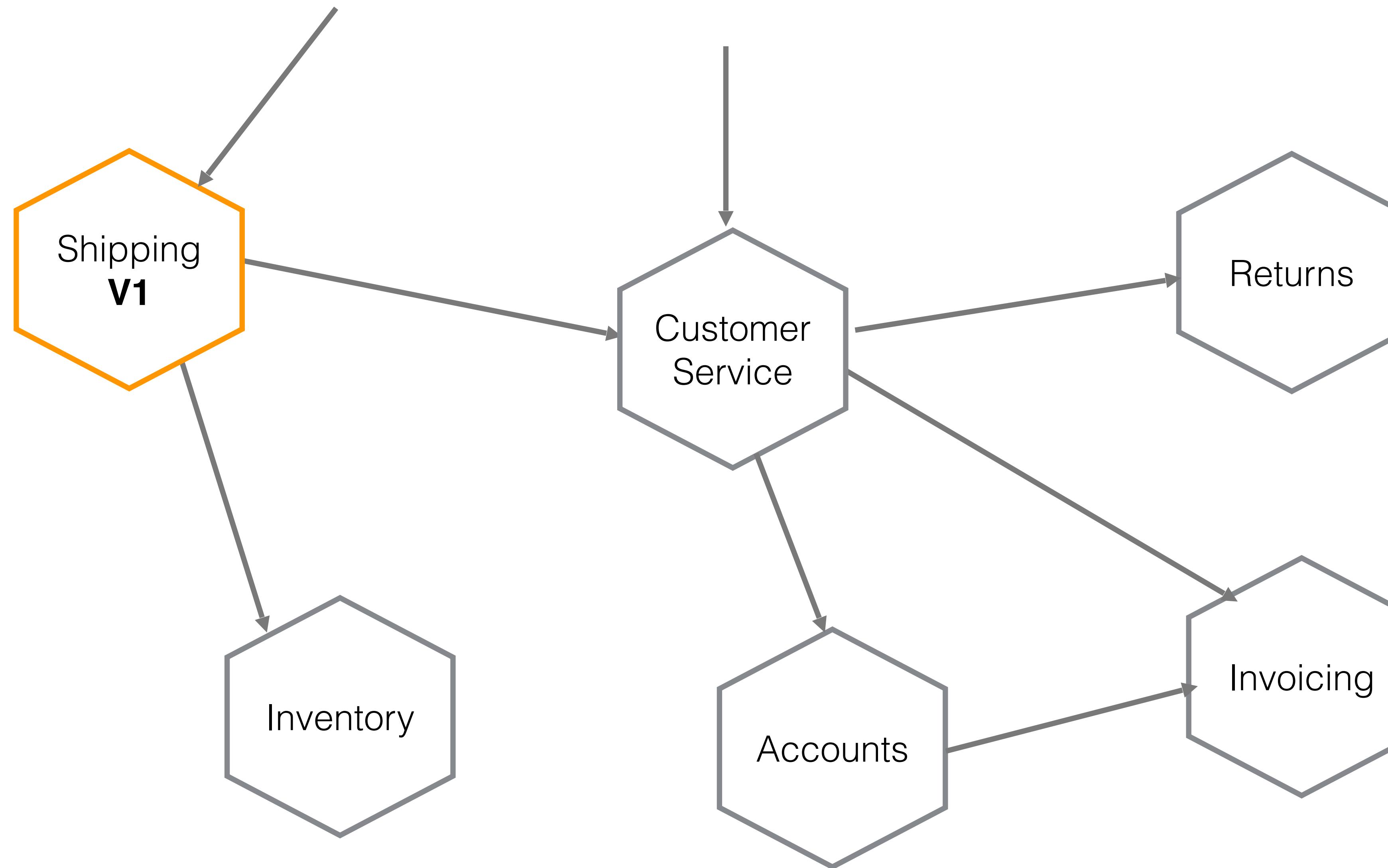
Just started using them

1-2 years experience

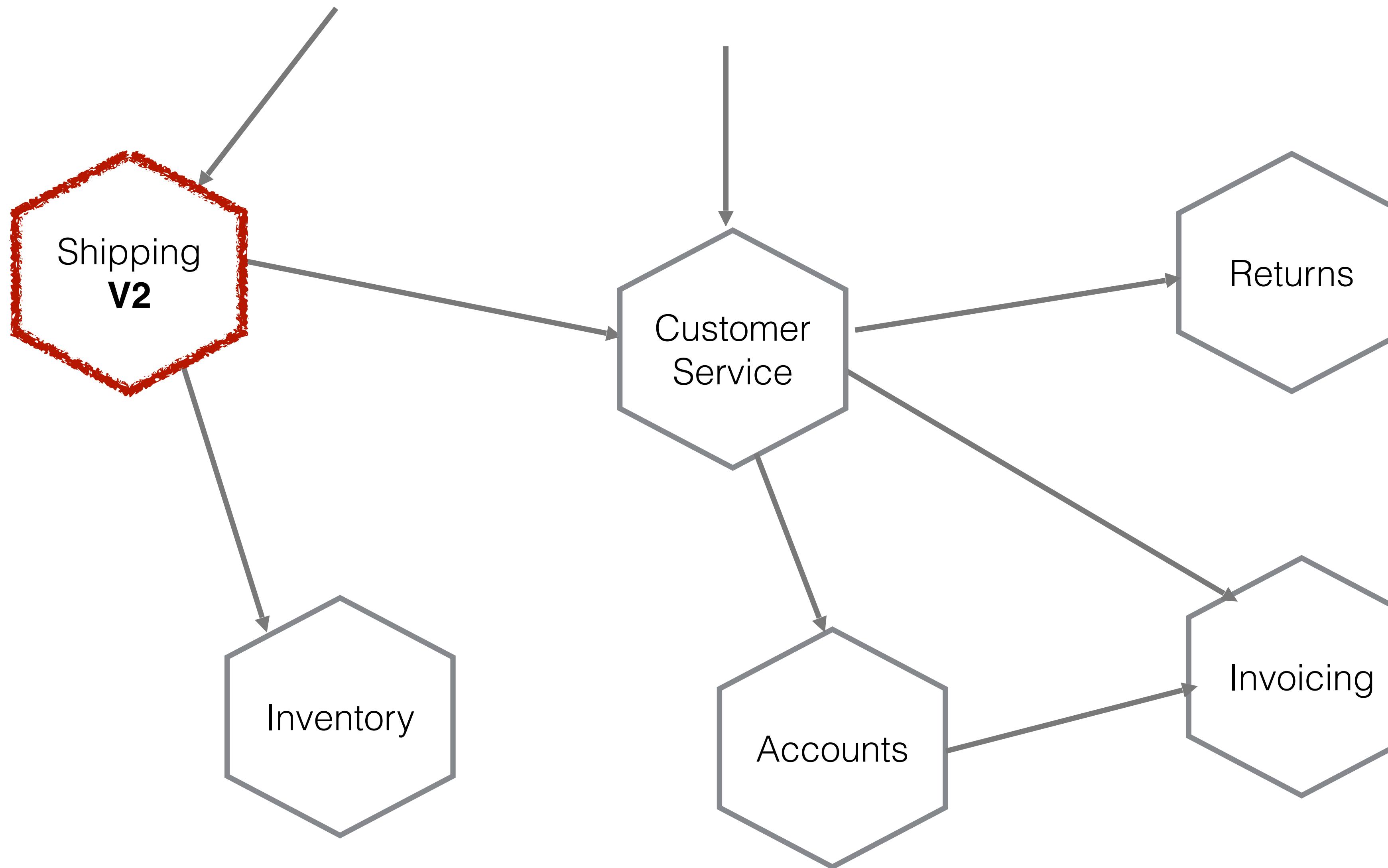
2+ years experience

What are microservices?

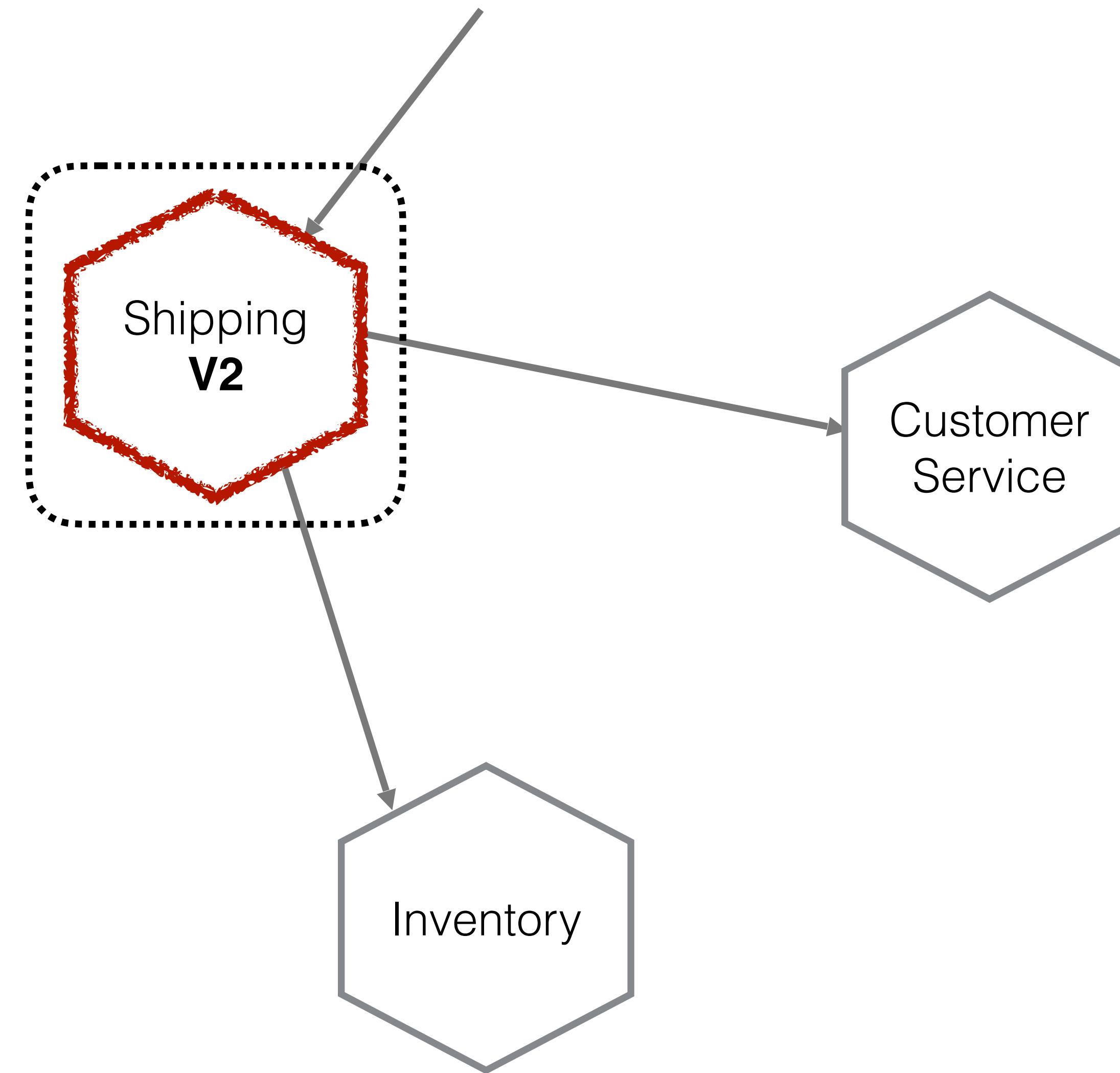
INDEPENDENT DEPLOYABILITY



INDEPENDENT DEPLOYABILITY

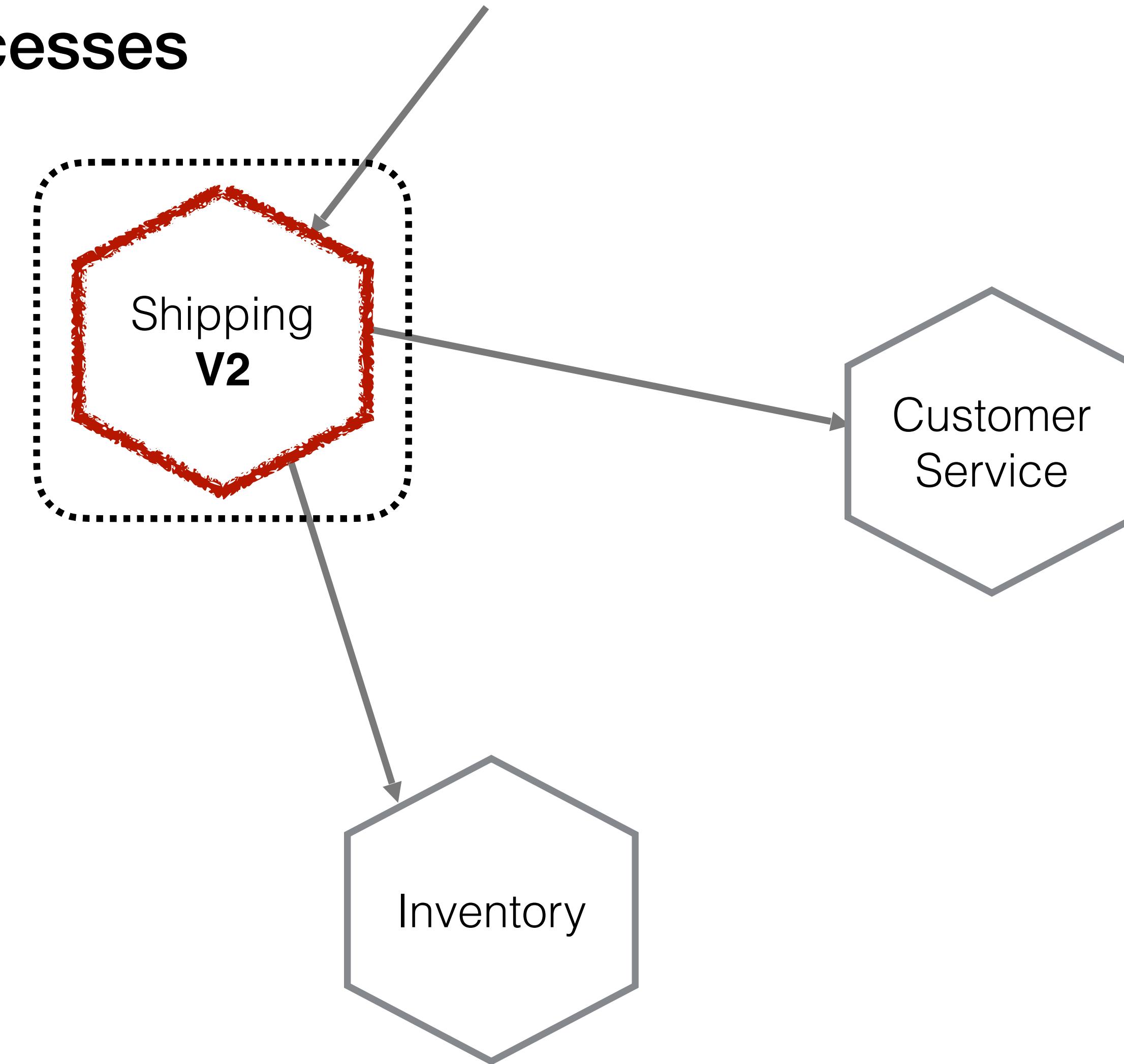


ANATOMY OF A MICROSERVICE



ANATOMY OF A MICROSERVICE

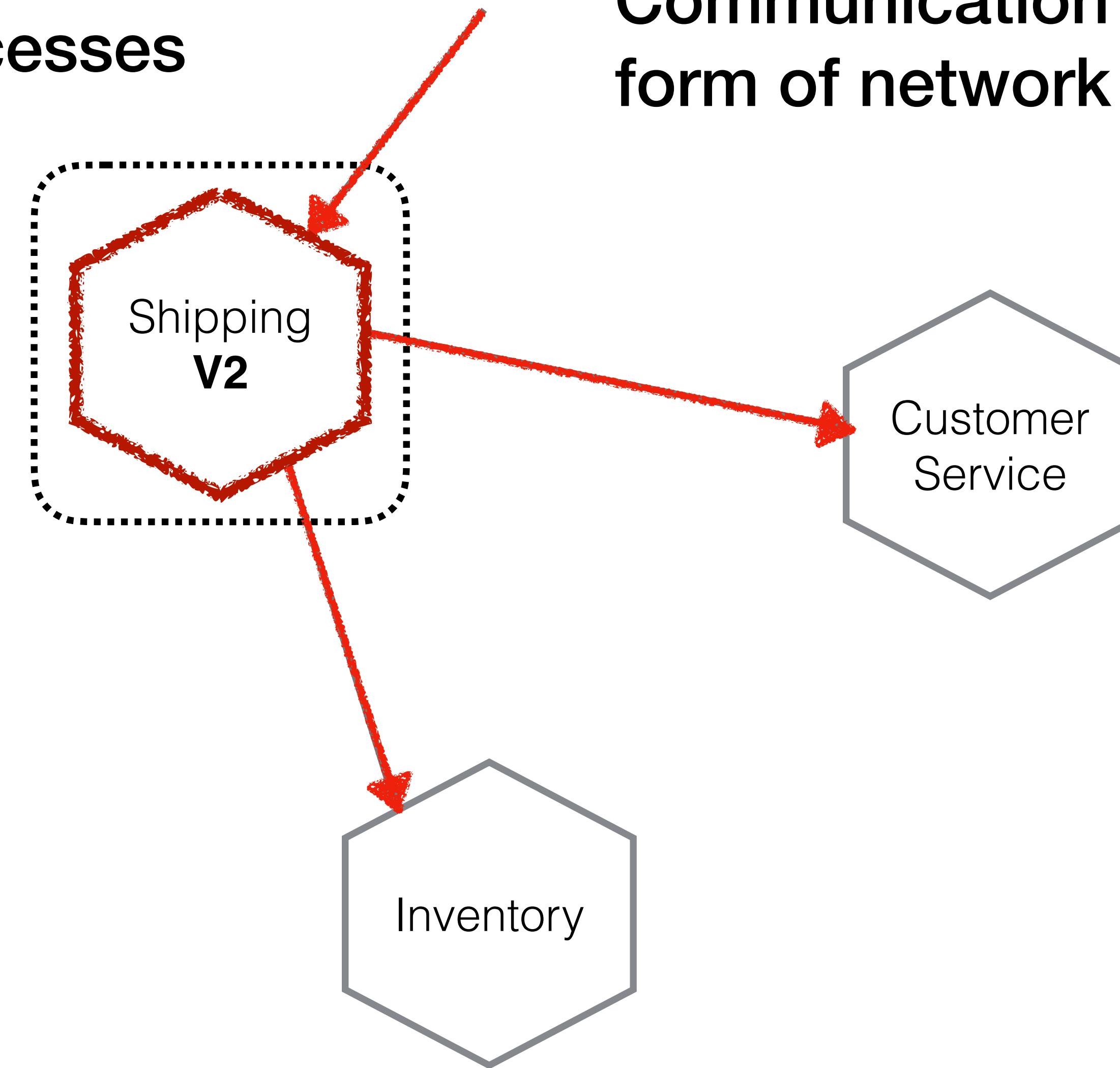
Microservice instances are typically separate processes



ANATOMY OF A MICROSERVICE

Microservice instances are typically separate processes

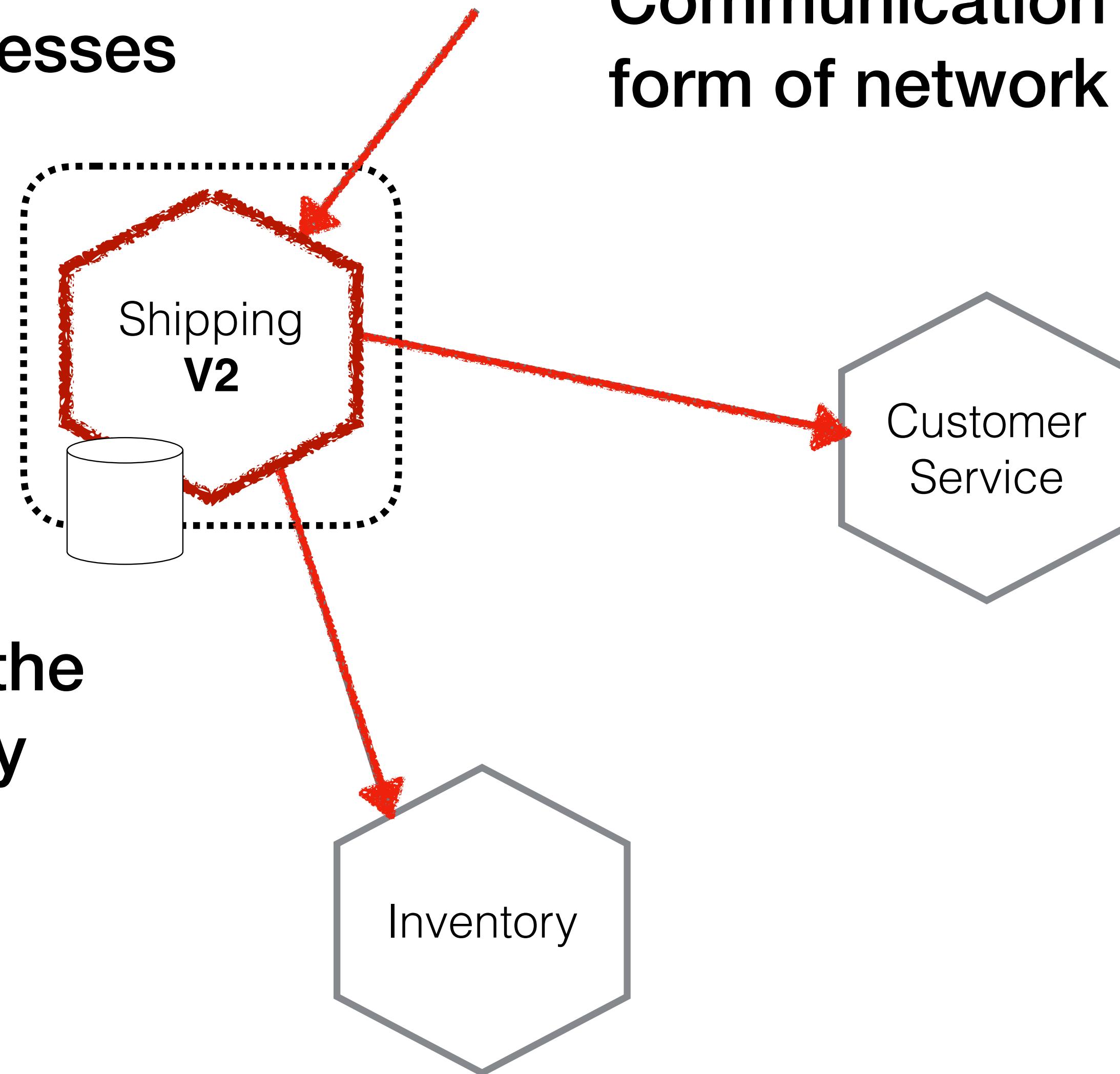
Communication is via some form of network call



ANATOMY OF A MICROSERVICE

Microservice instances are typically separate processes

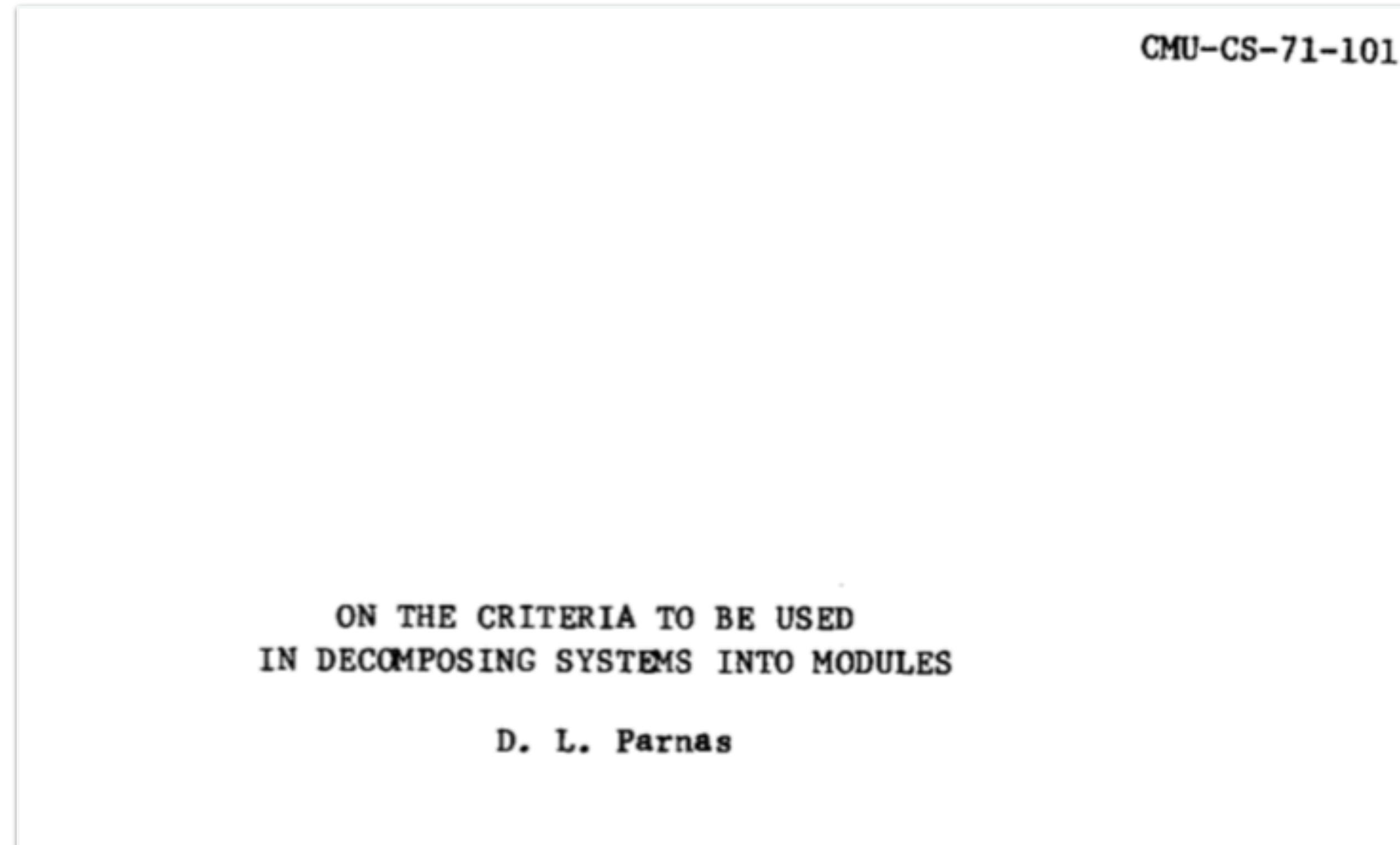
Communication is via some form of network call



Data is hidden inside the microservice boundary

Why separate database?

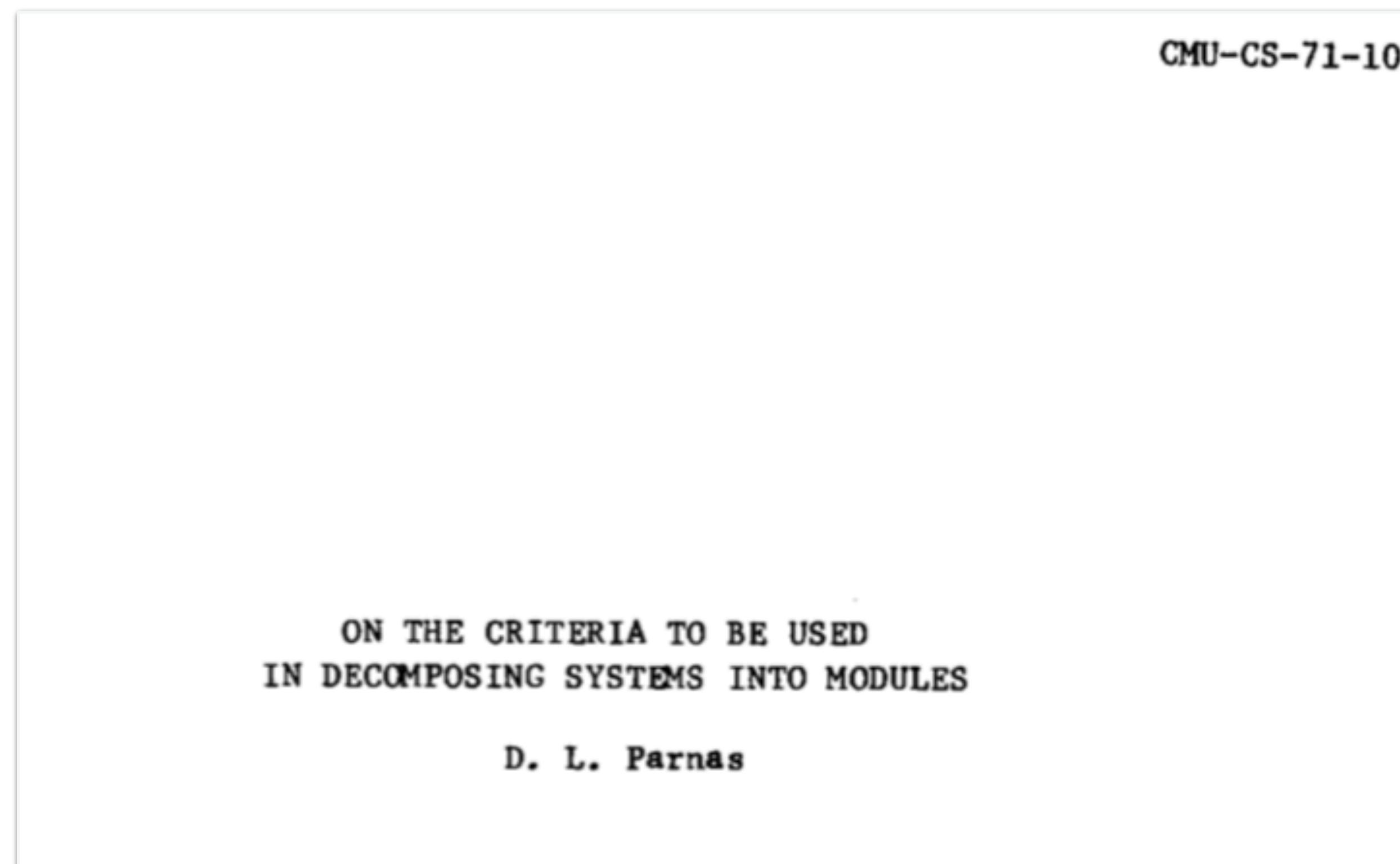
INFORMATION HIDING



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

@samnewman

INFORMATION HIDING

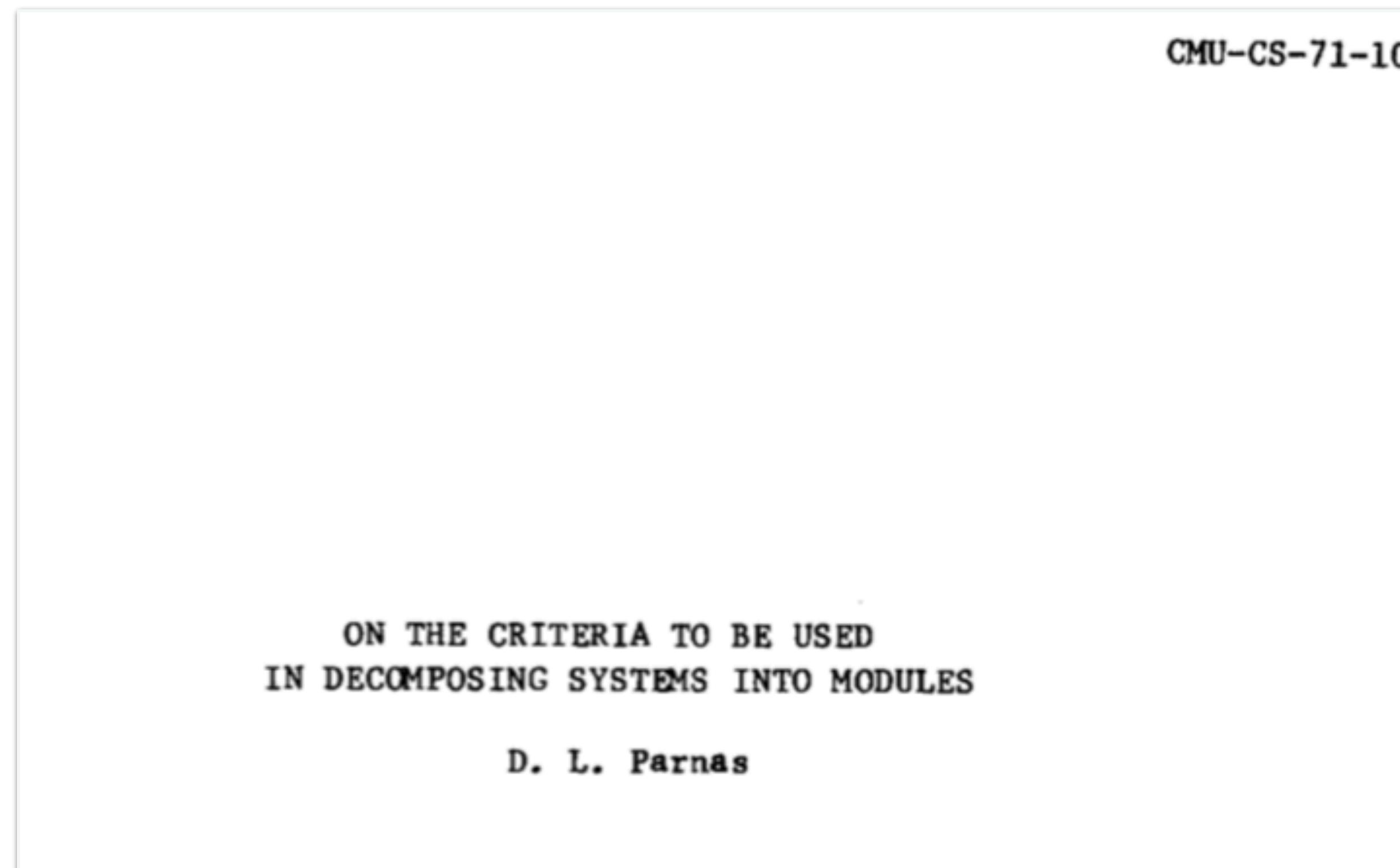


Published in 1971

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

@samnewman

INFORMATION HIDING



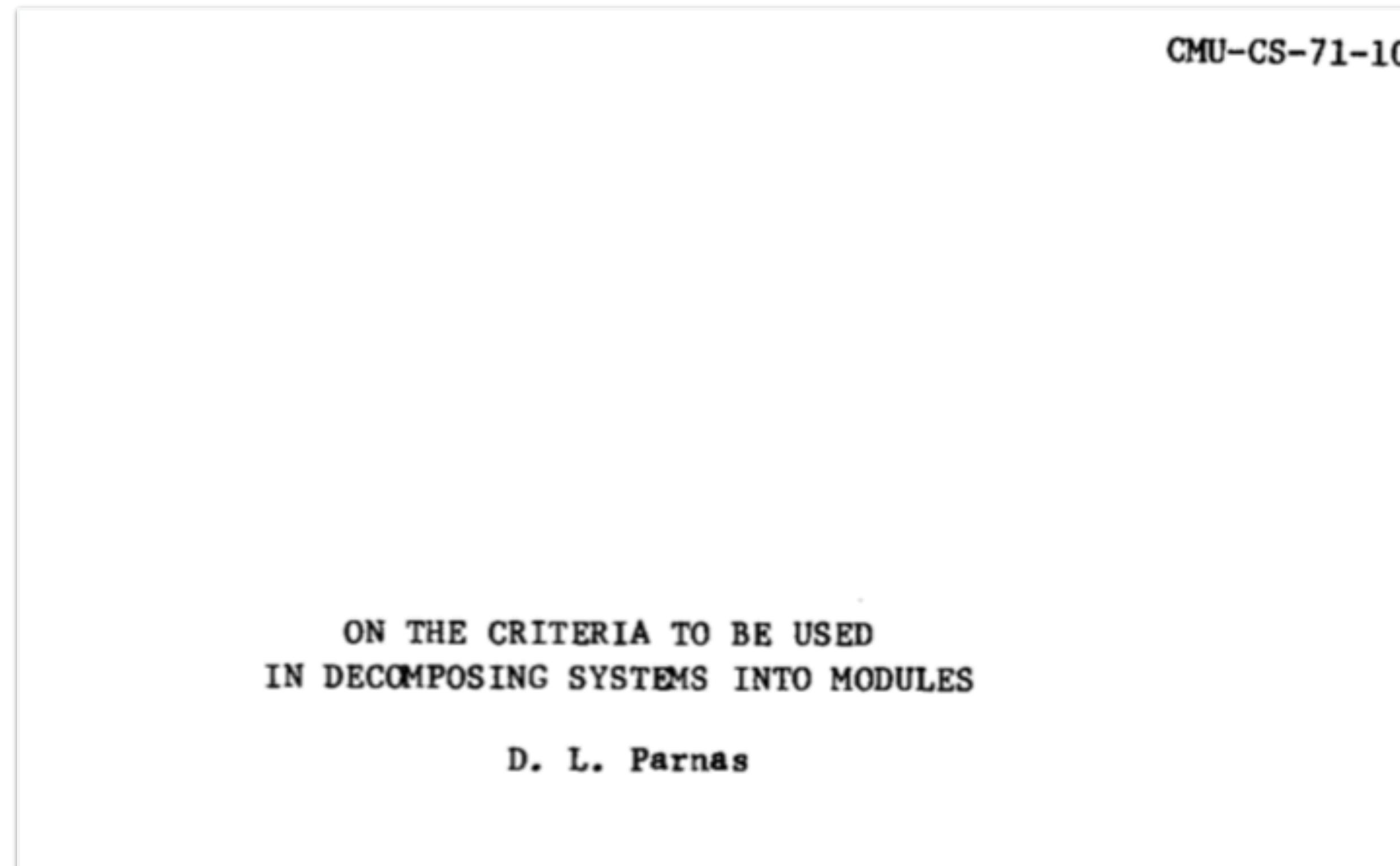
Published in 1971

Looked at how best to define
module boundaries

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

@samnewman

INFORMATION HIDING

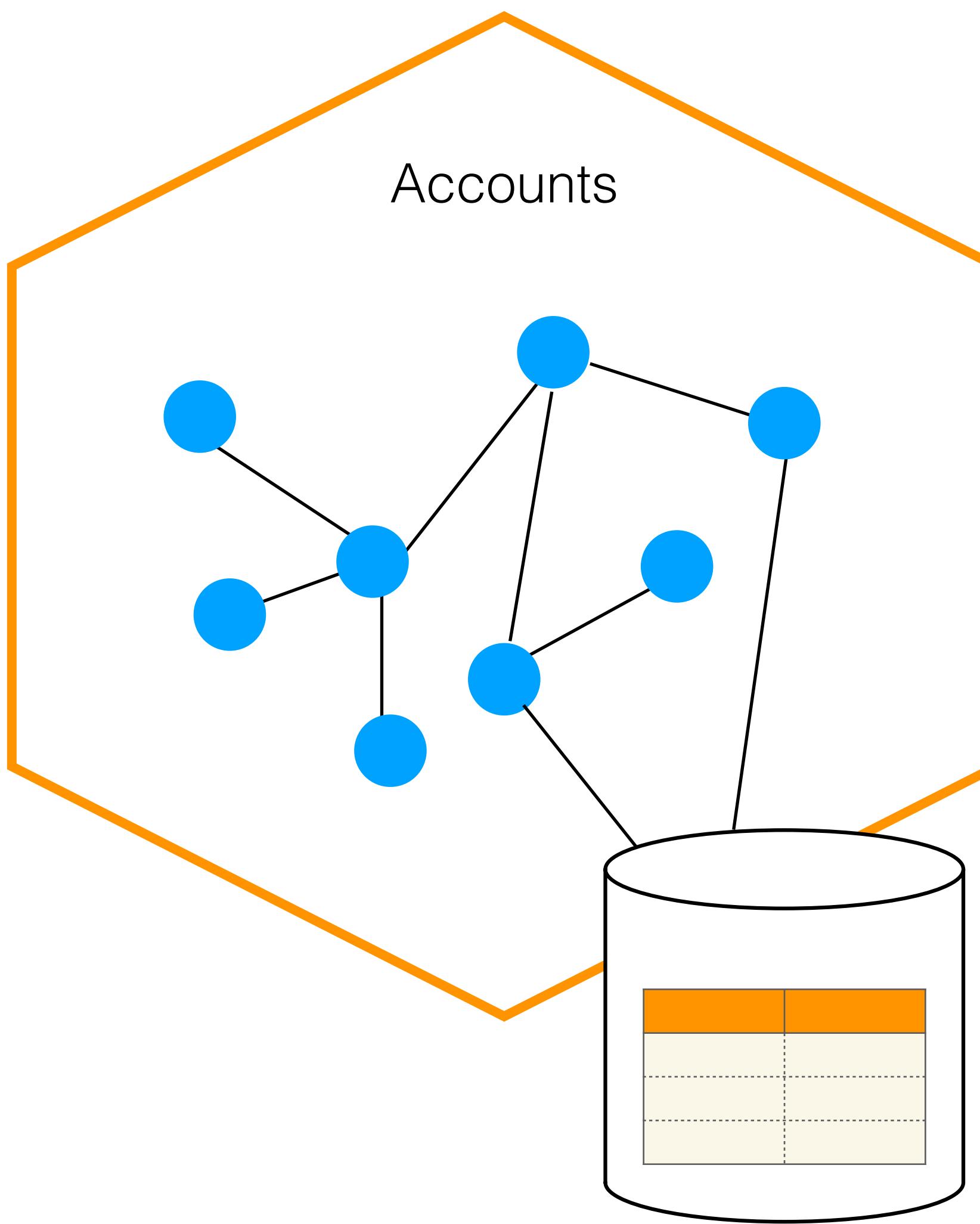


Published in 1971

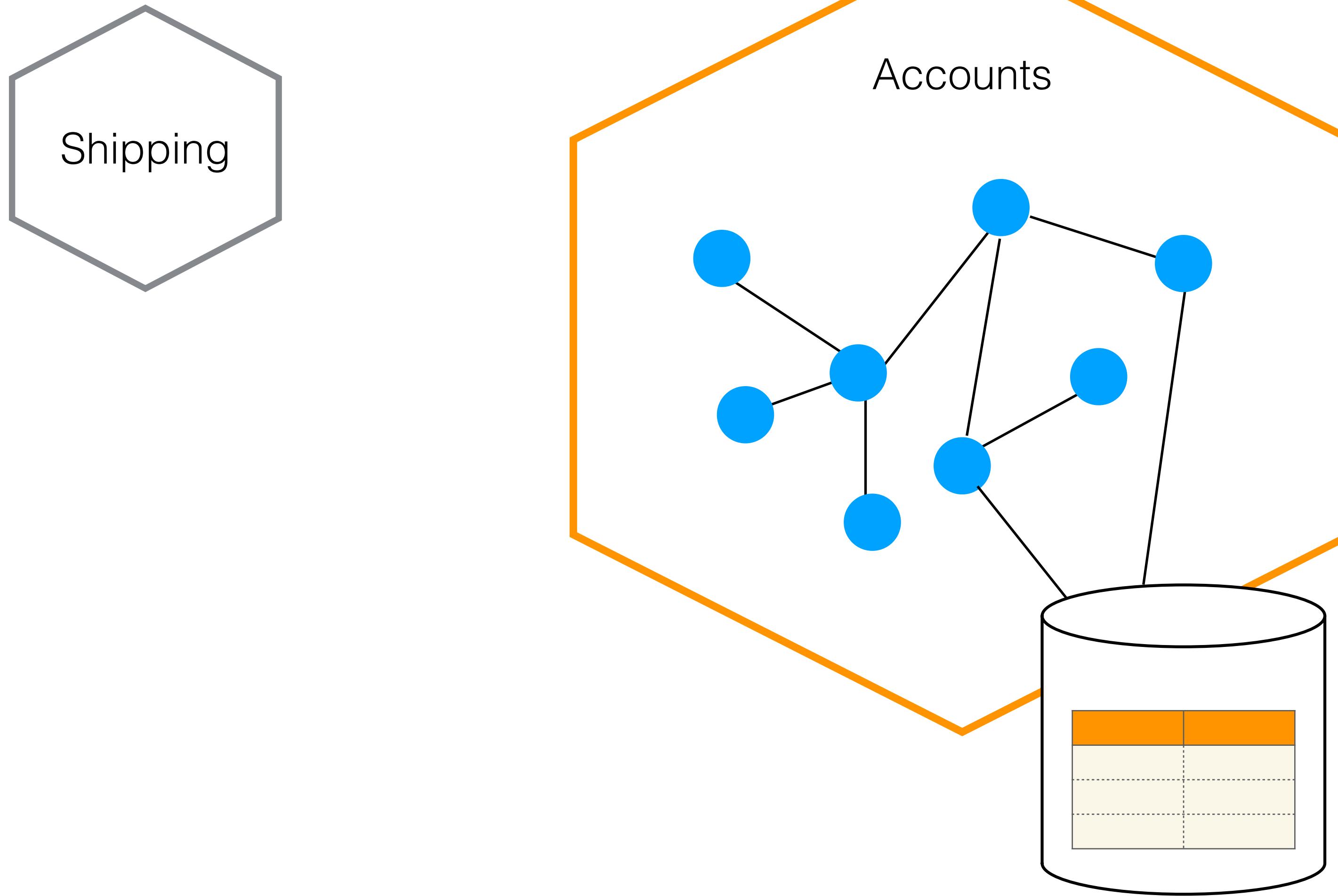
Looked at how best to define module boundaries

Found that “information hiding” worked best

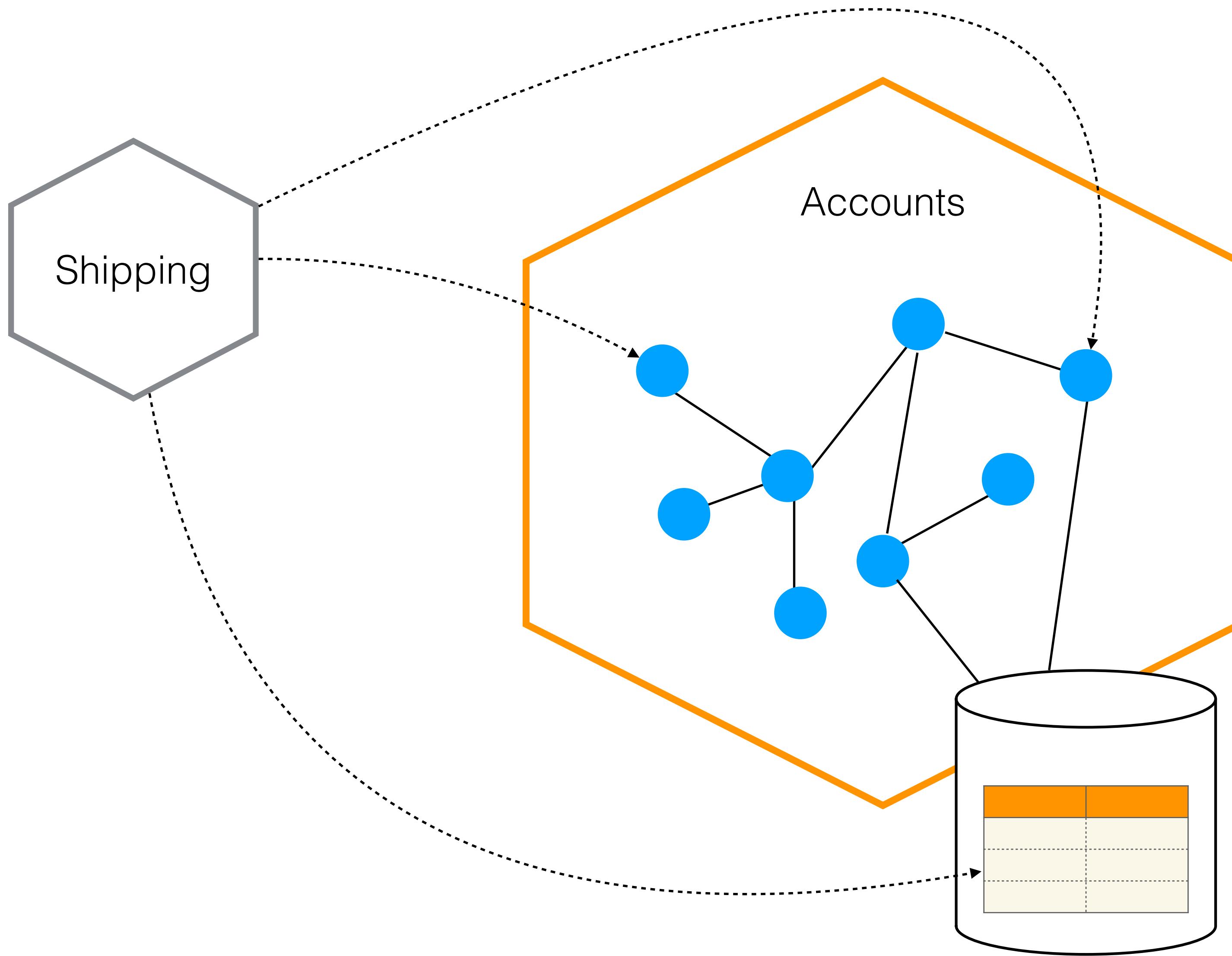
NOT HIDING?



NOT HIDING?

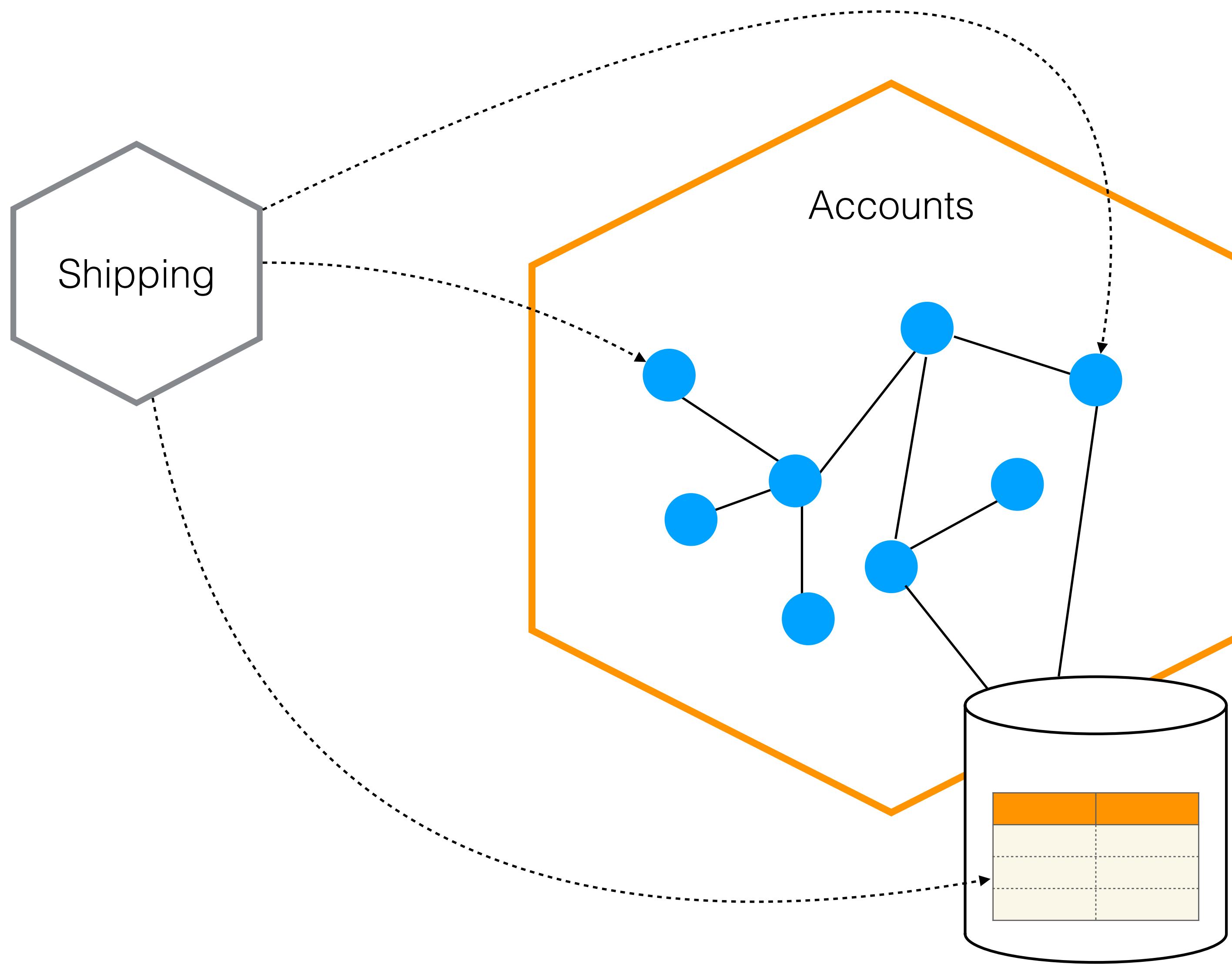


NOT HIDING?



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

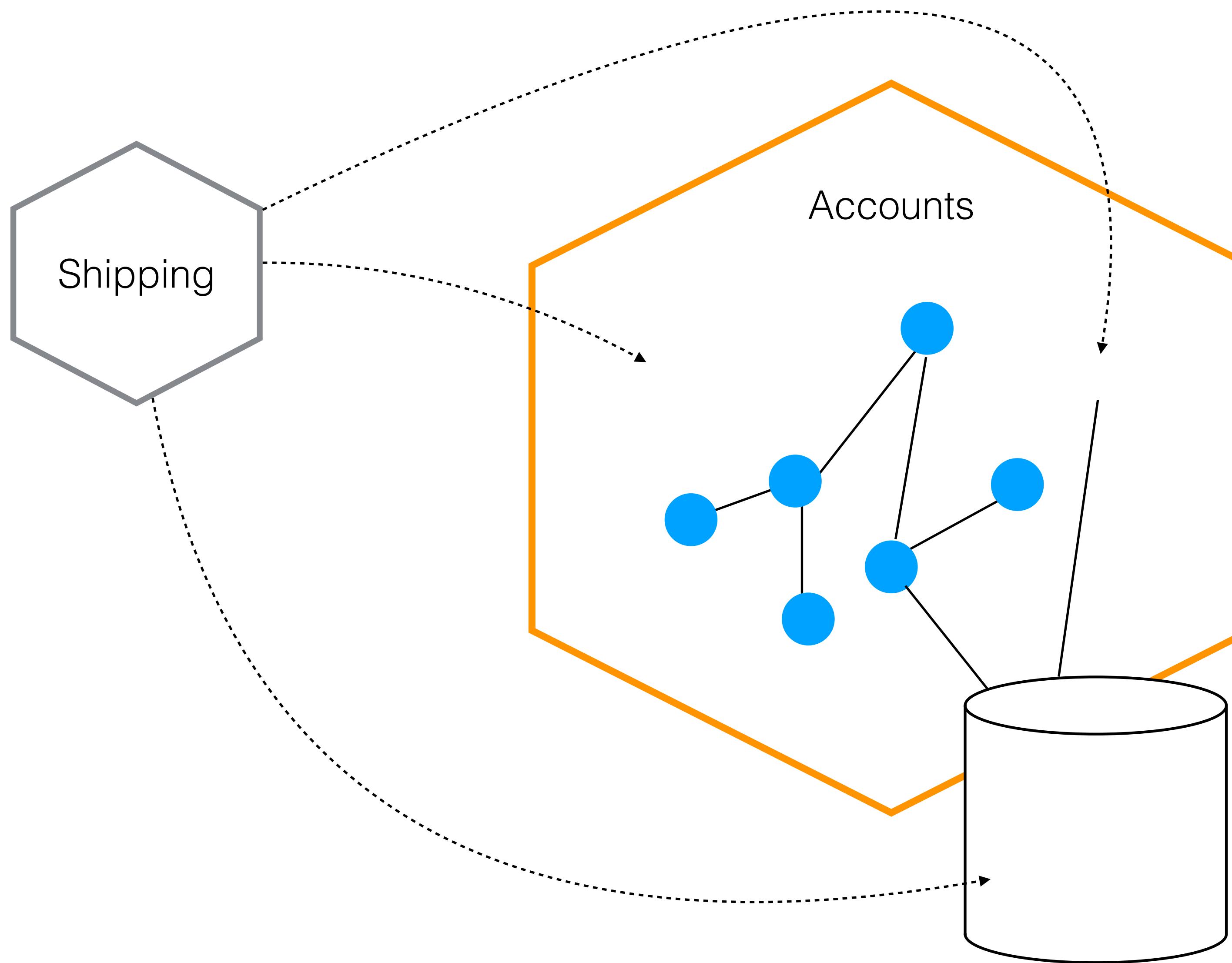
NOT HIDING?



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

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

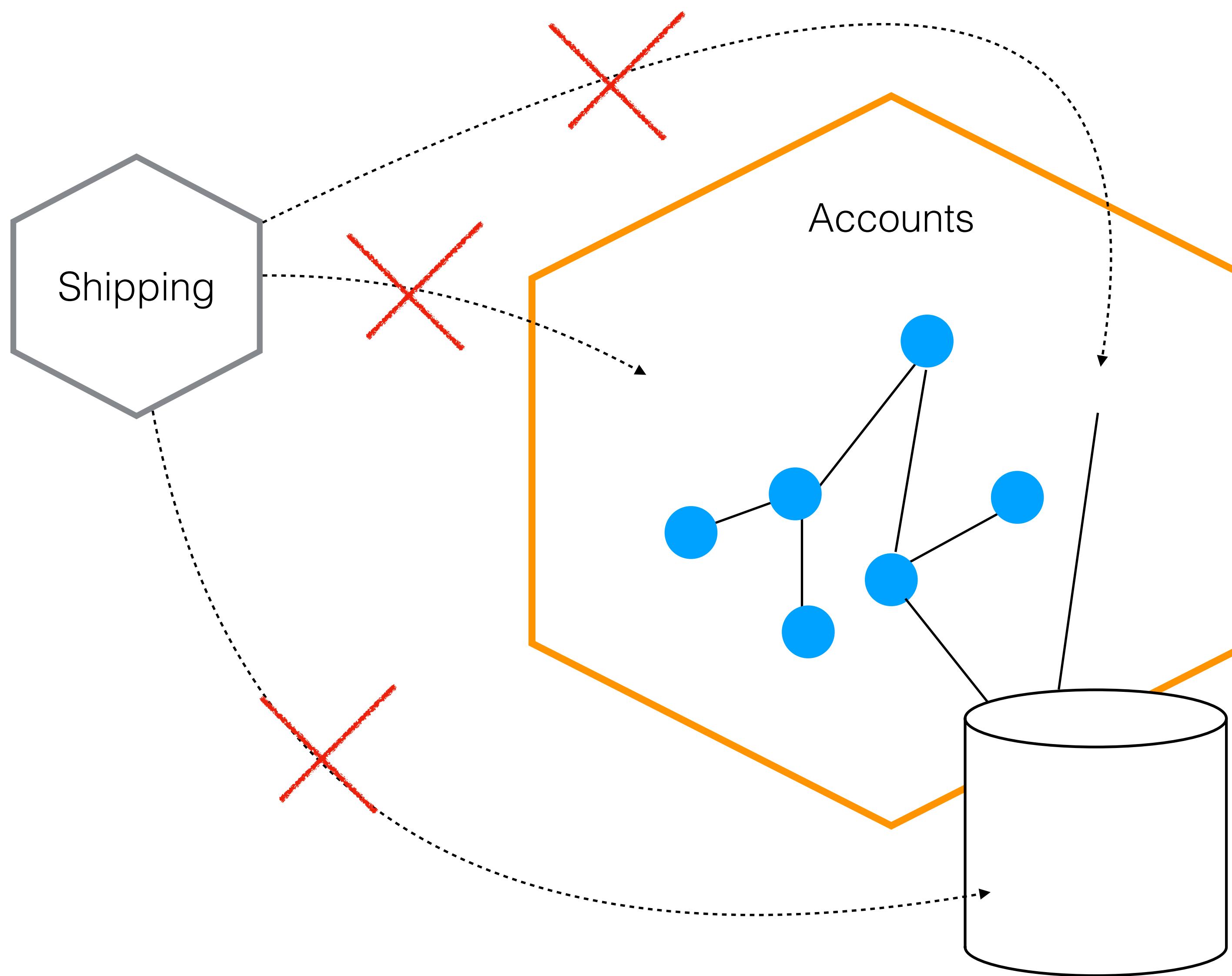
NOT HIDING?



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

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

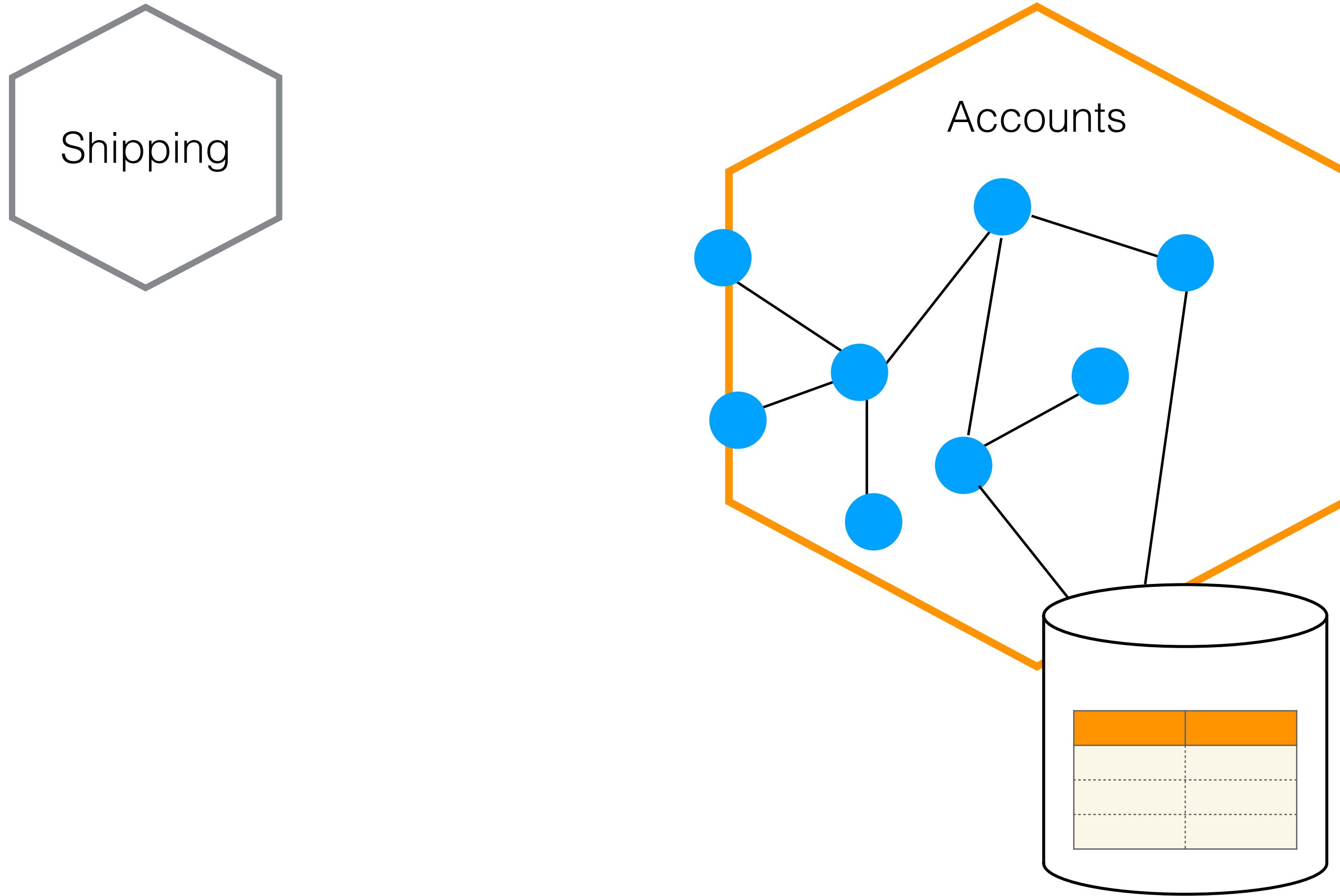
NOT HIDING?



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

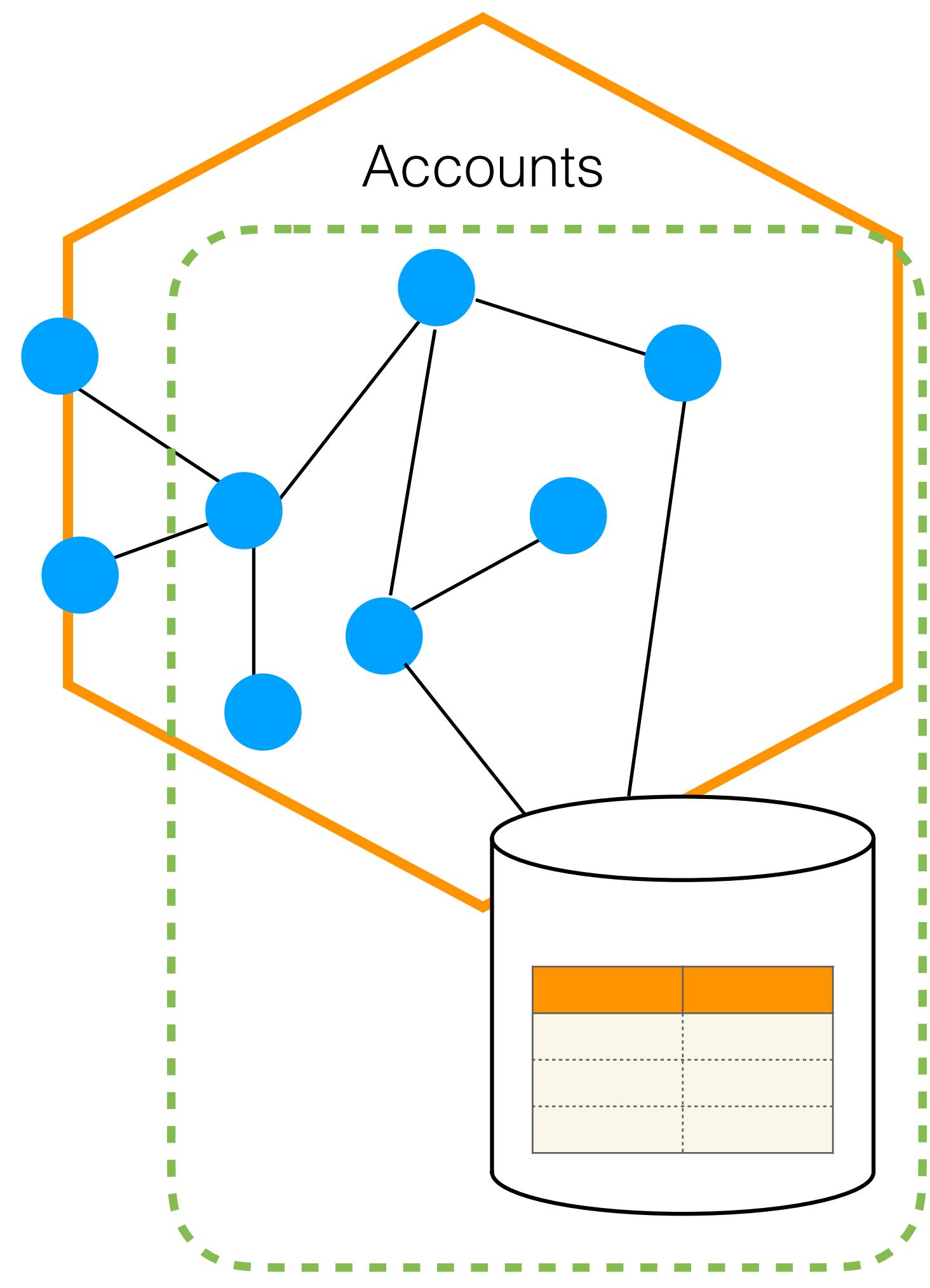
...then you can't change this implementation without breaking the consumer

HDING!



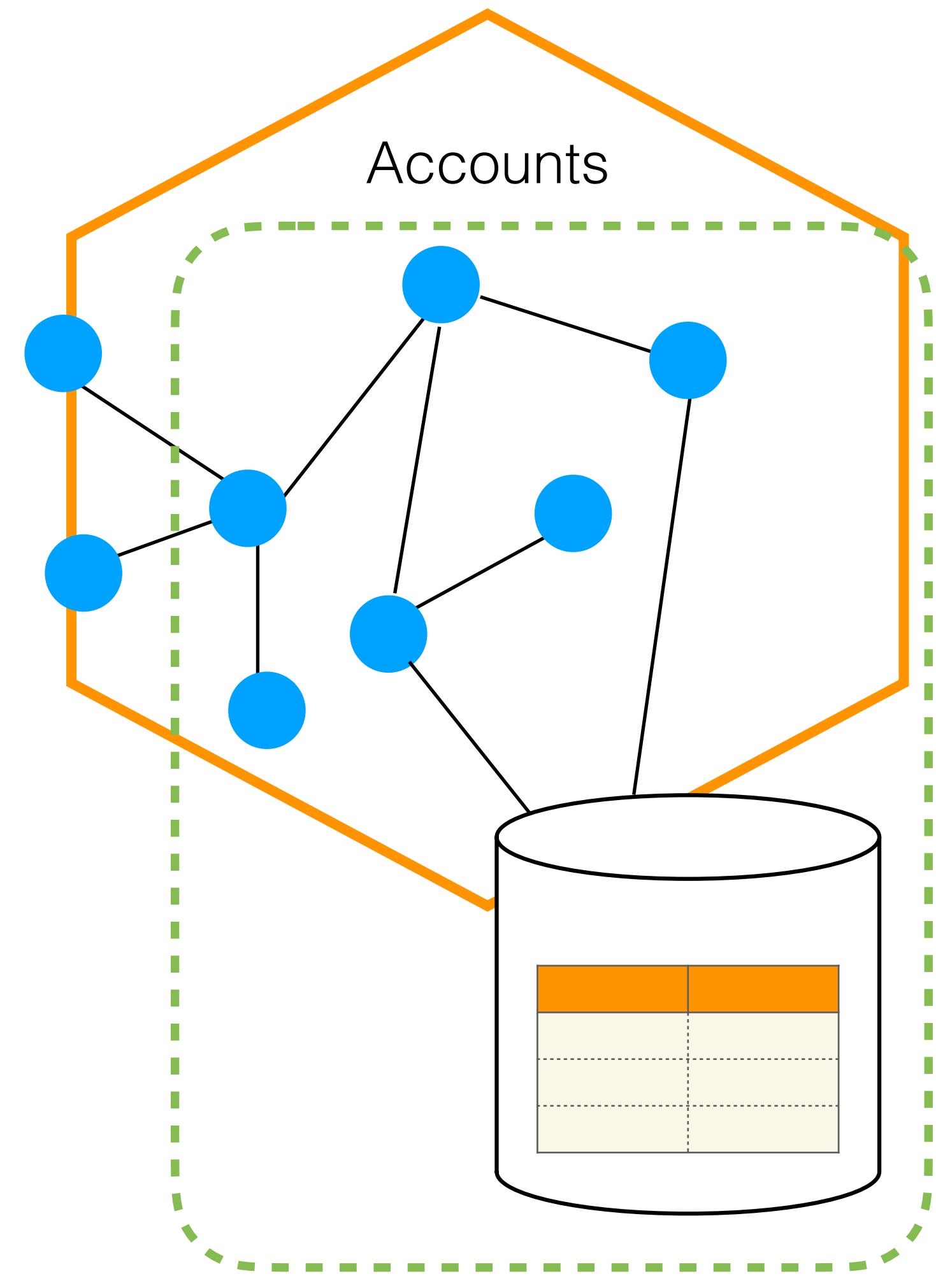
HIDING!

Hide your secrets!



Hidden

HIDING!

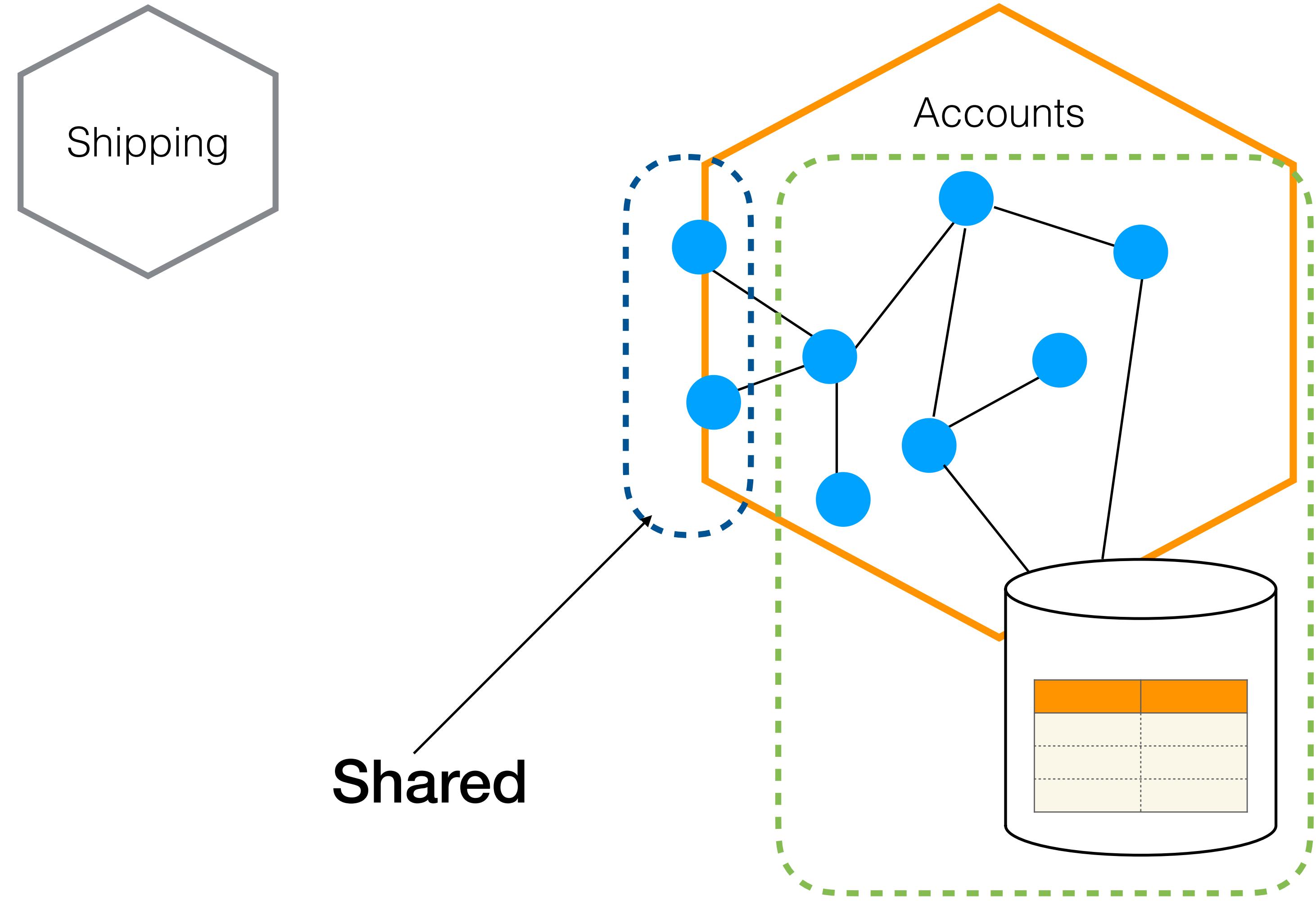


Hide your secrets!

Be explicit about what is shared, and what is hidden

Hidden

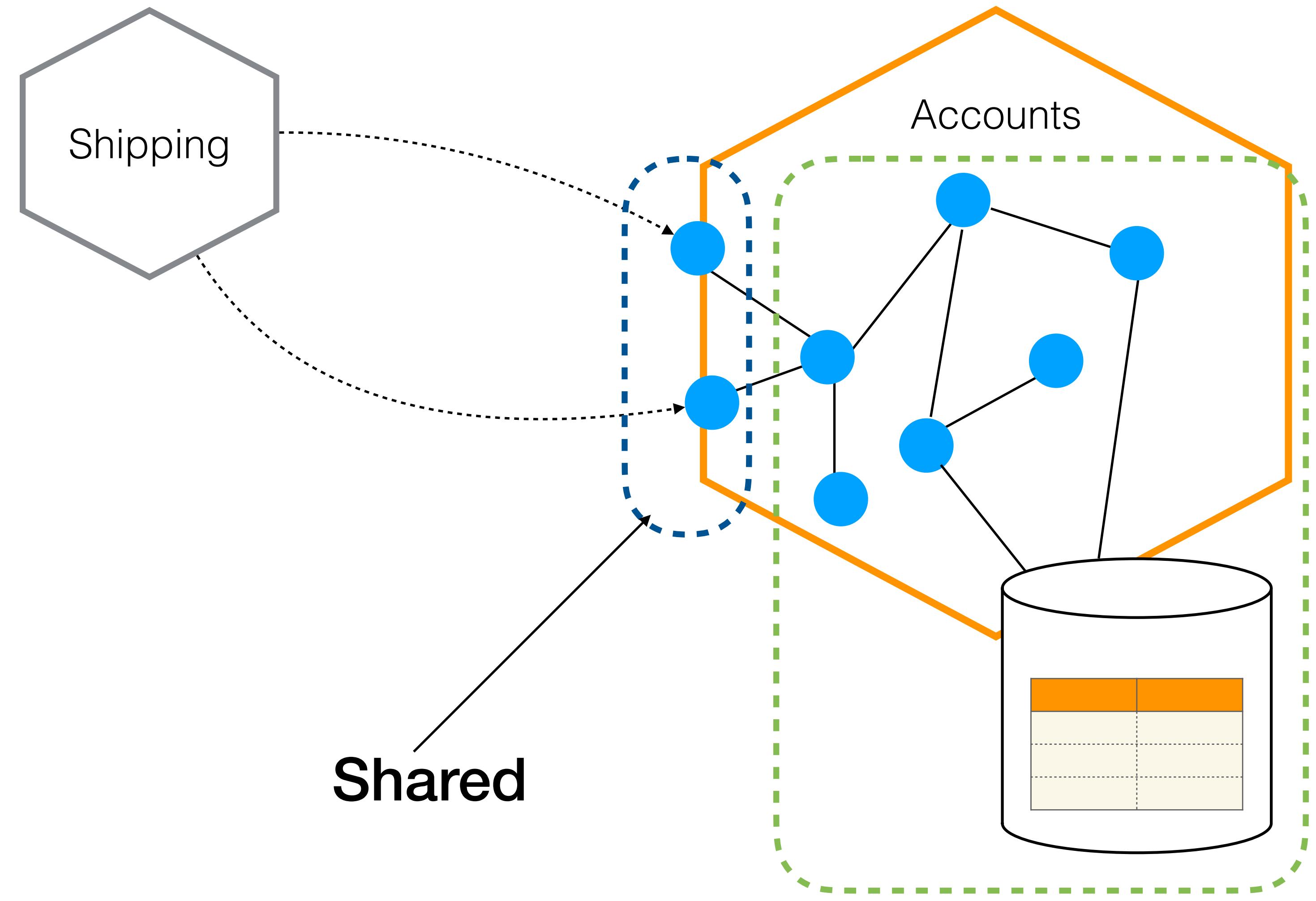
HIDING!



Hide your secrets!

Be explicit about what is shared, and what is hidden

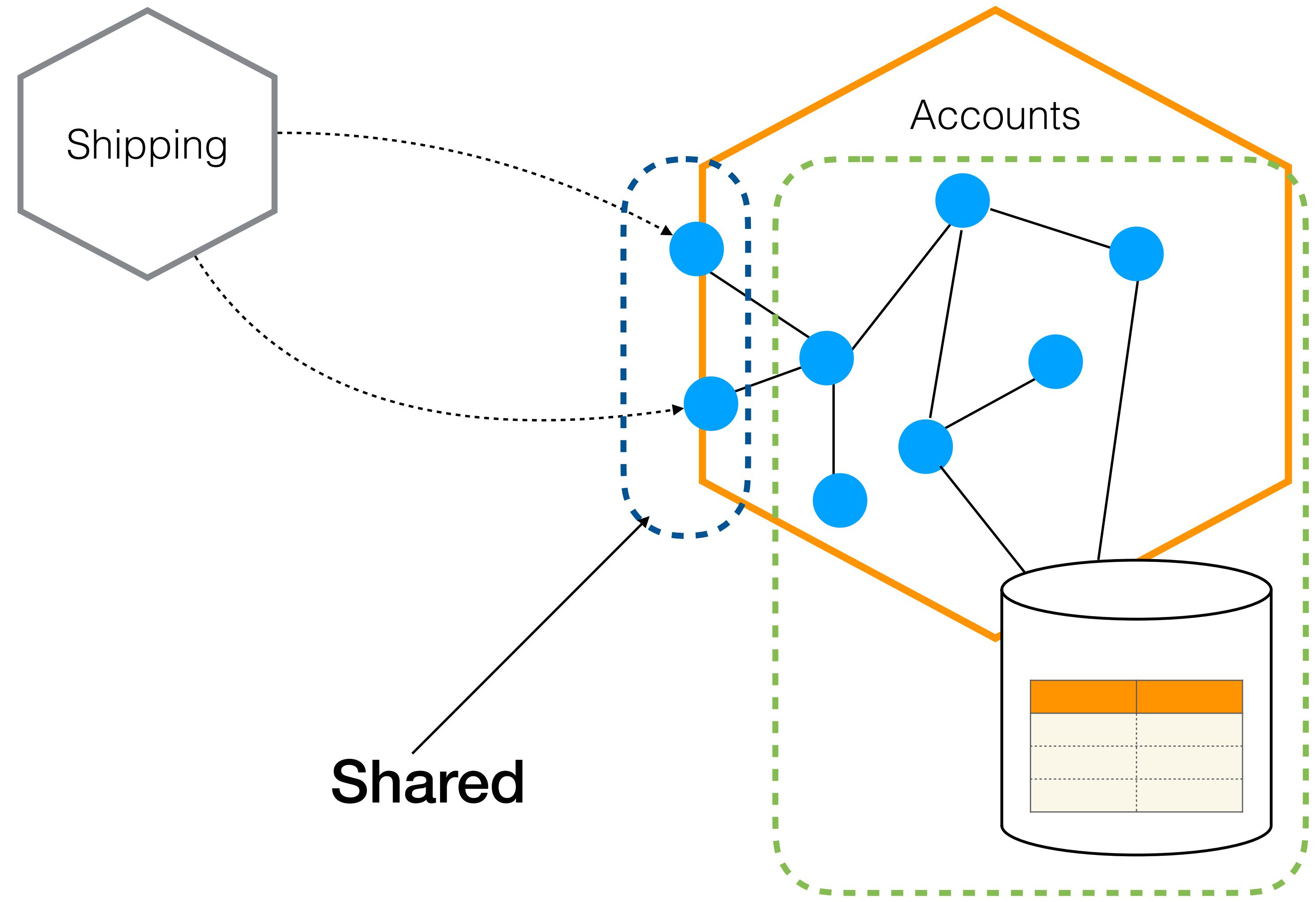
HIDING!



Hide your secrets!

Be explicit about what is shared, and what is hidden

HIDING!



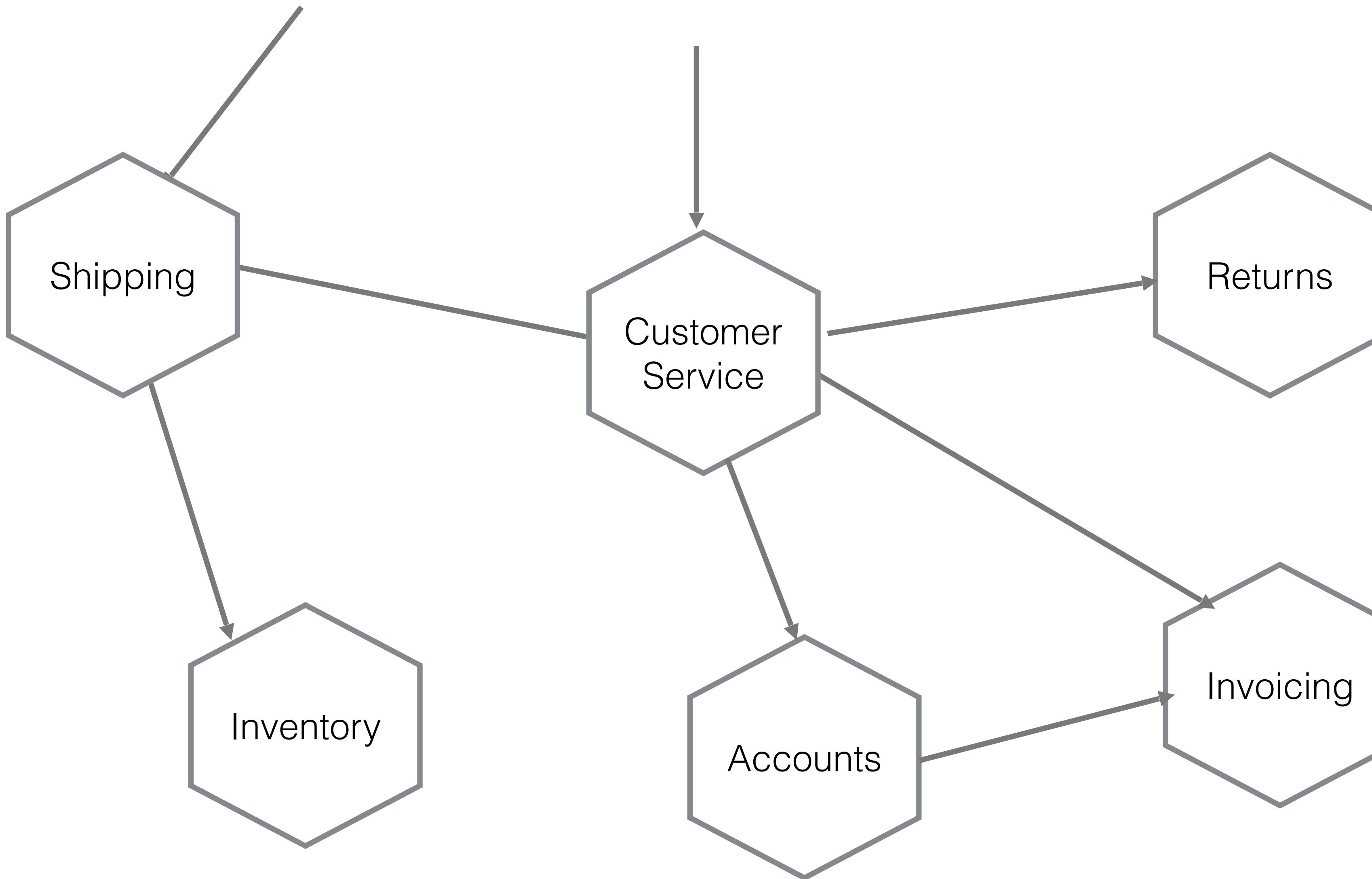
Hide your secrets!

Be explicit about what is shared, and what is hidden

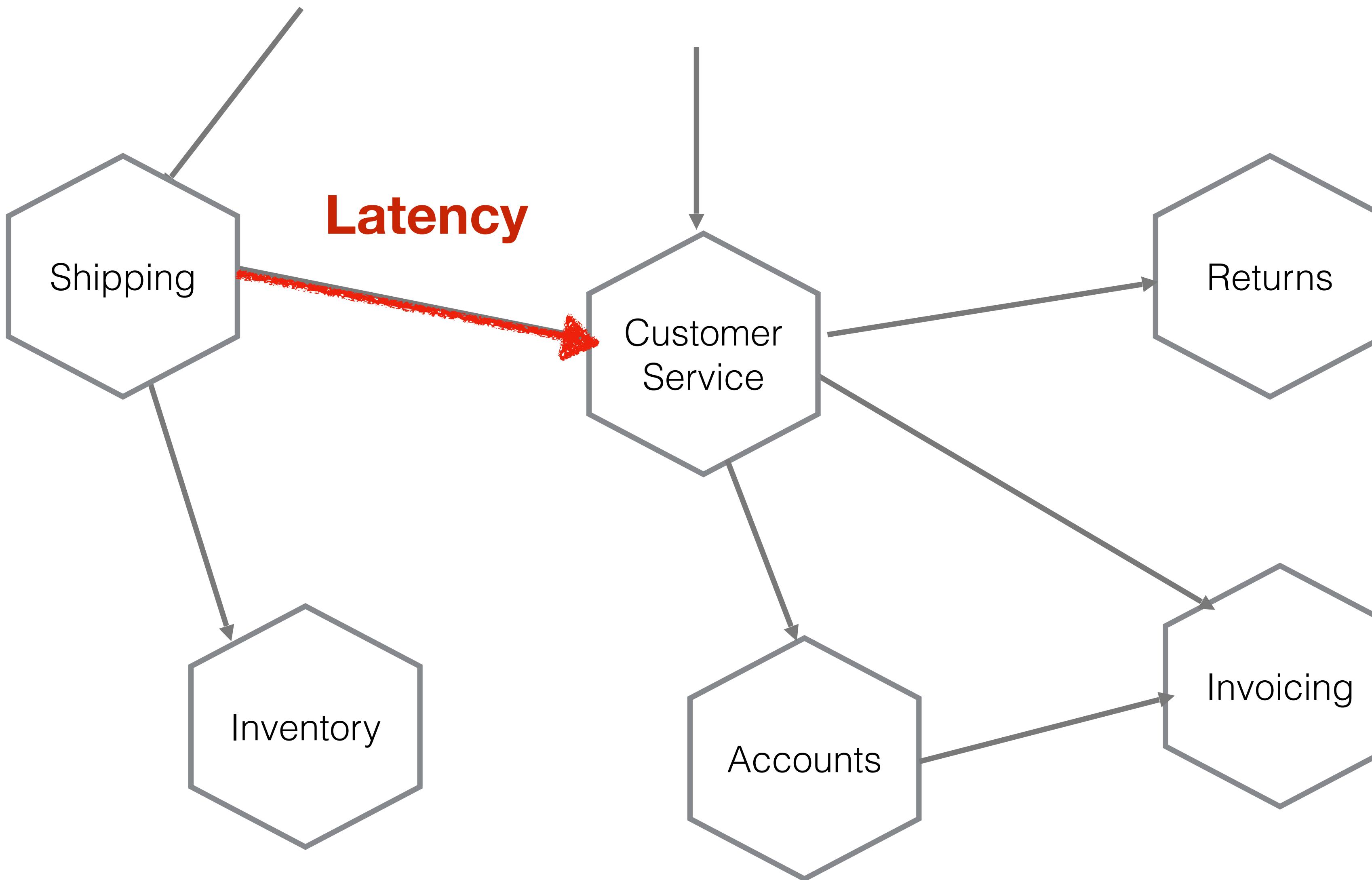
Hidden things can change, shared things can't

Hidden

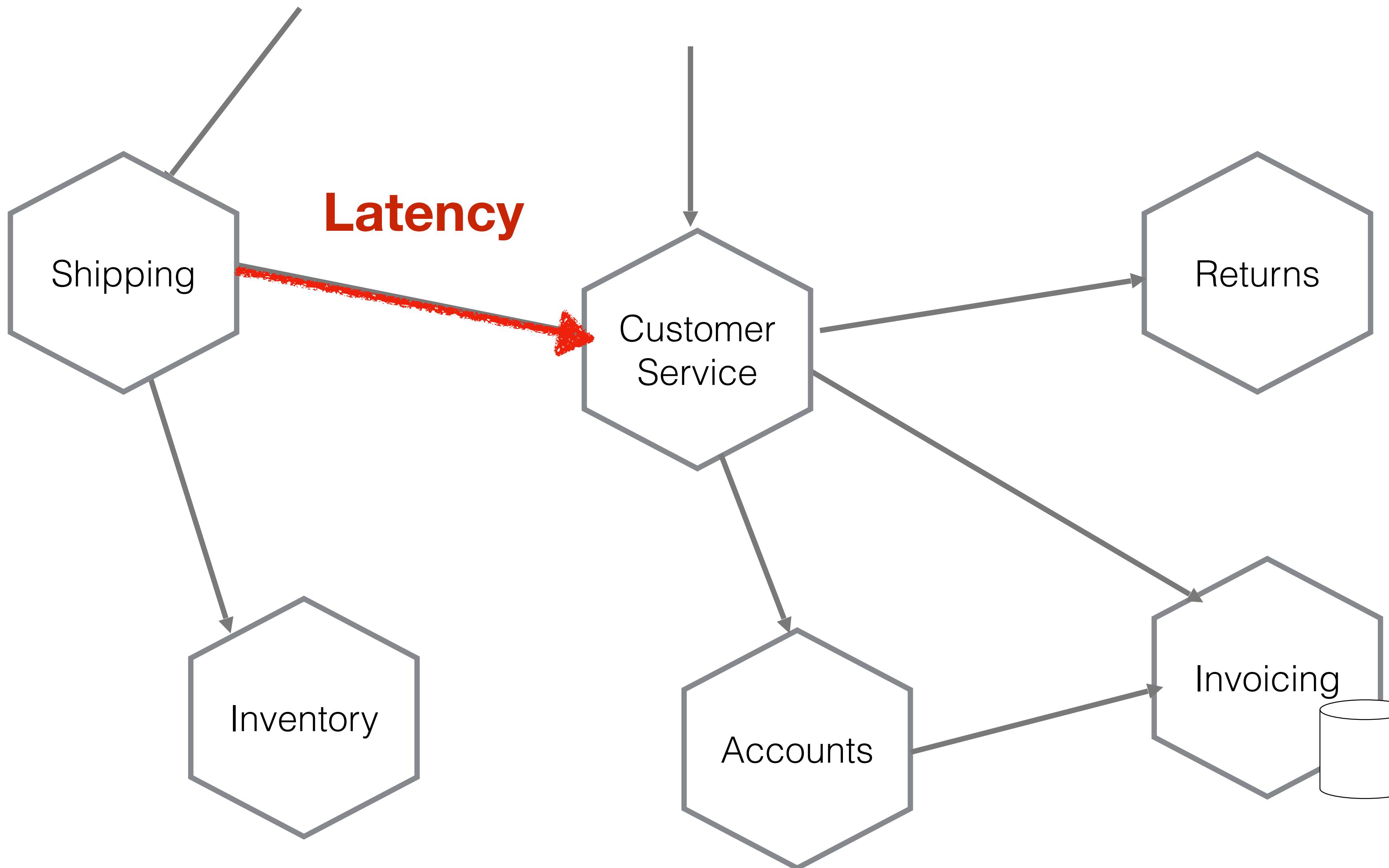
PROBLEMS WITH MICROSERVICES



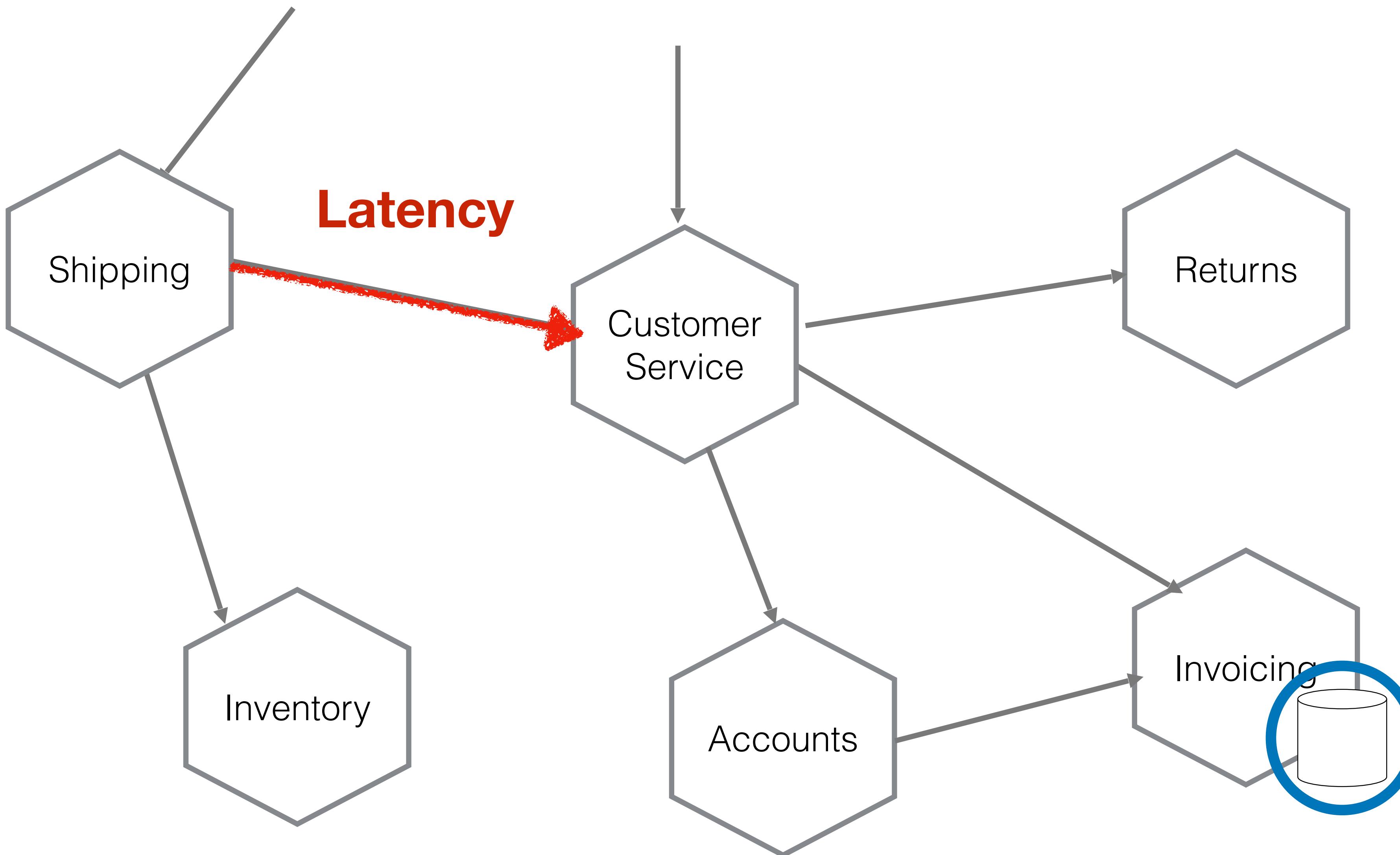
PROBLEMS WITH MICROSERVICES



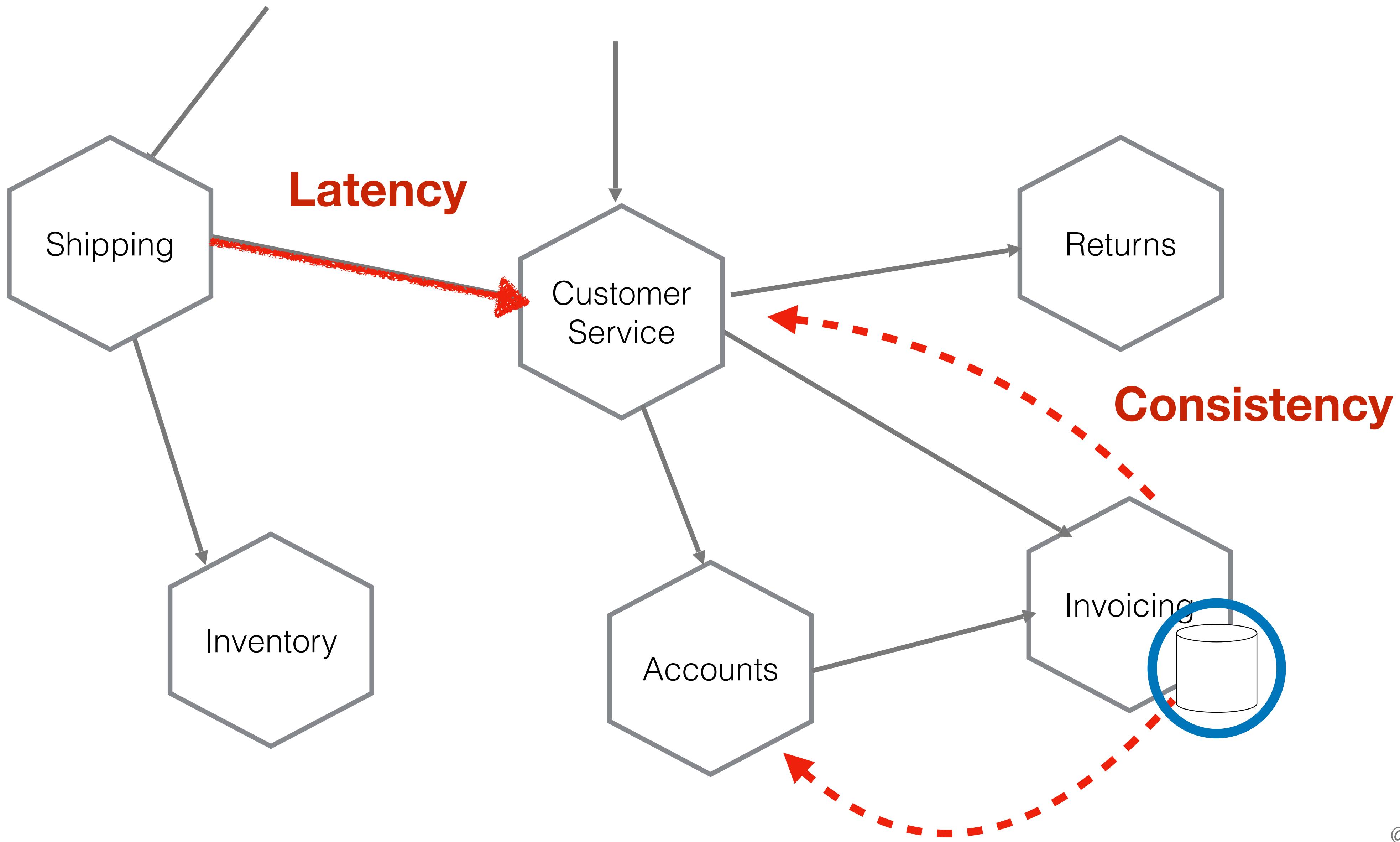
PROBLEMS WITH MICROSERVICES



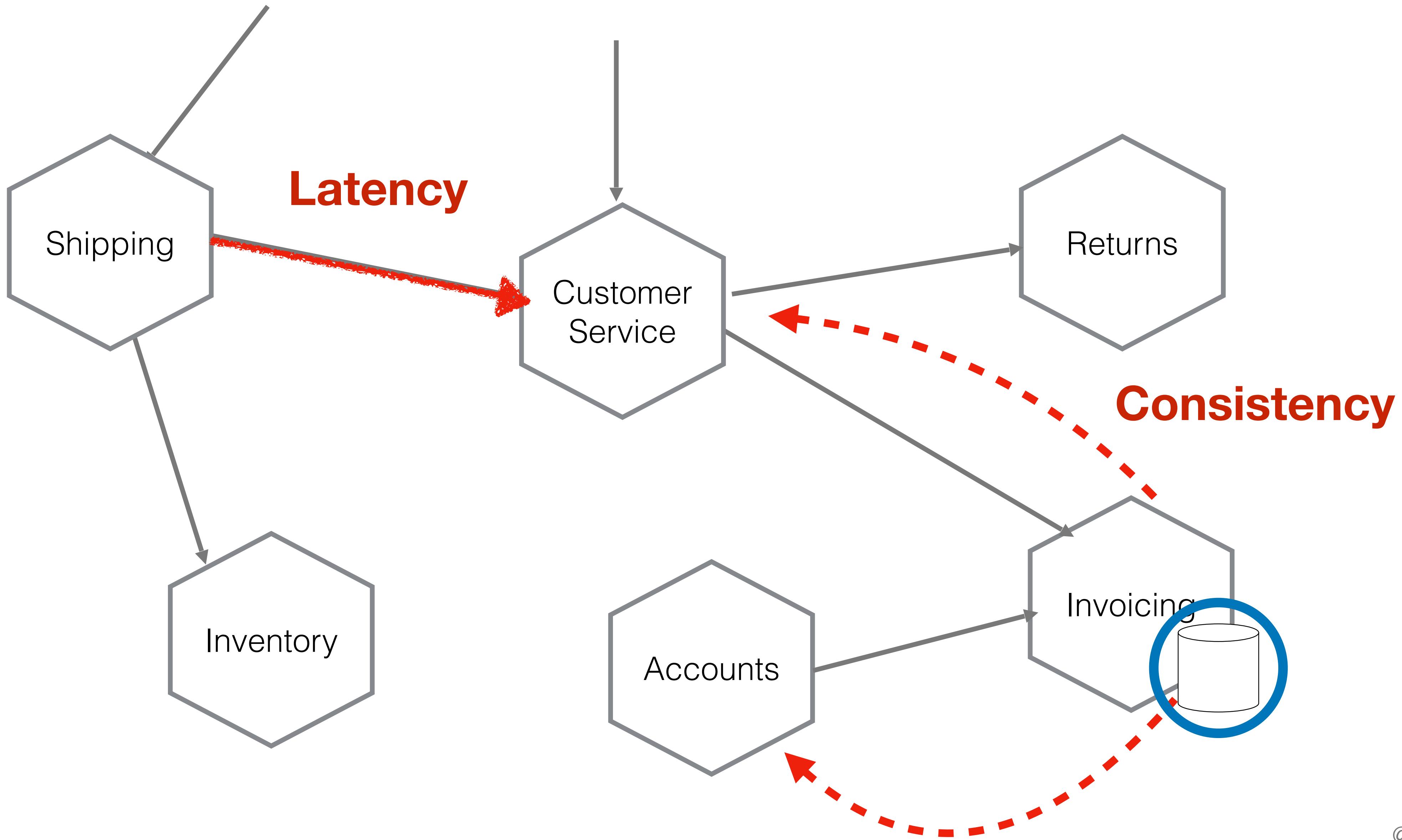
PROBLEMS WITH MICROSERVICES



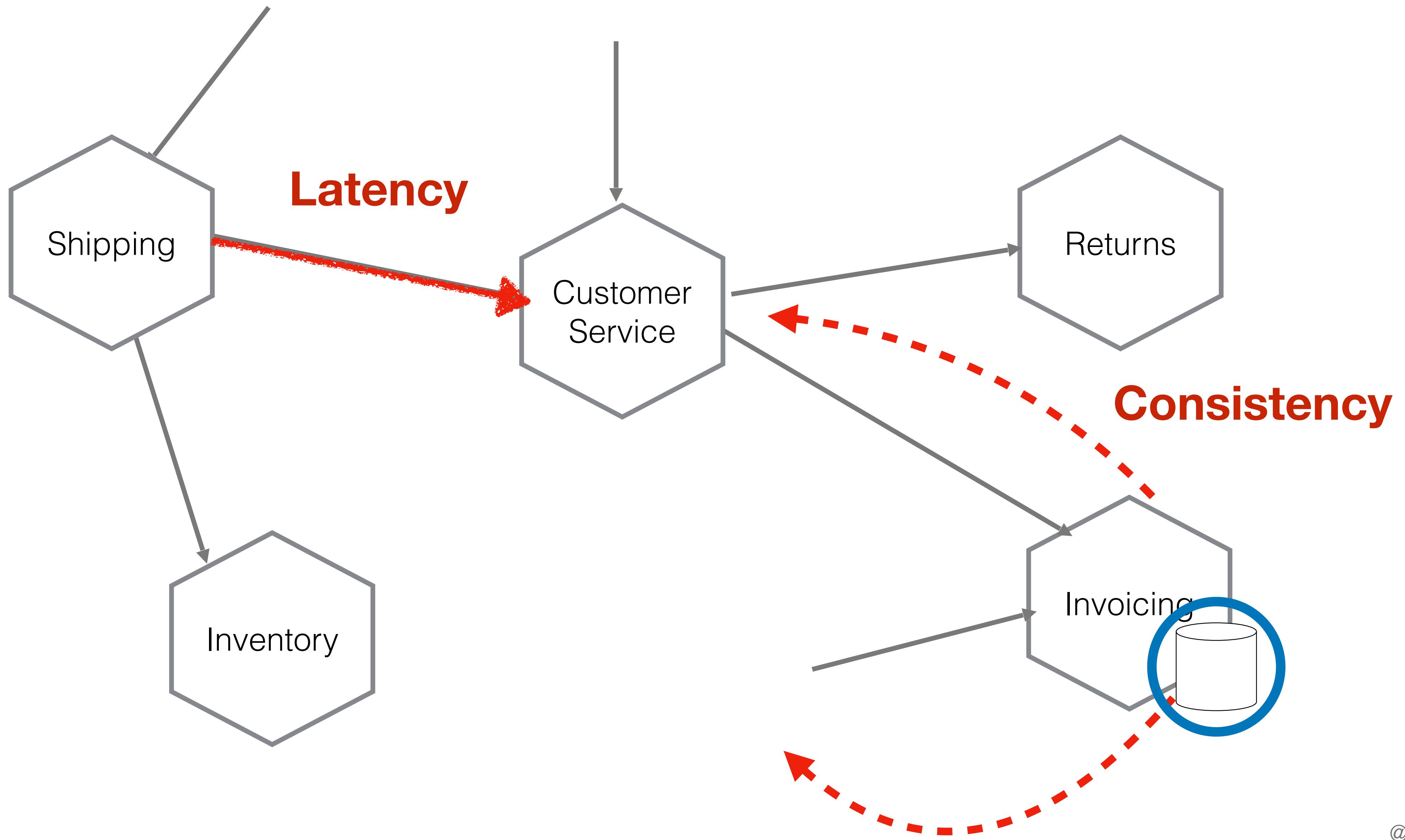
PROBLEMS WITH MICROSERVICES



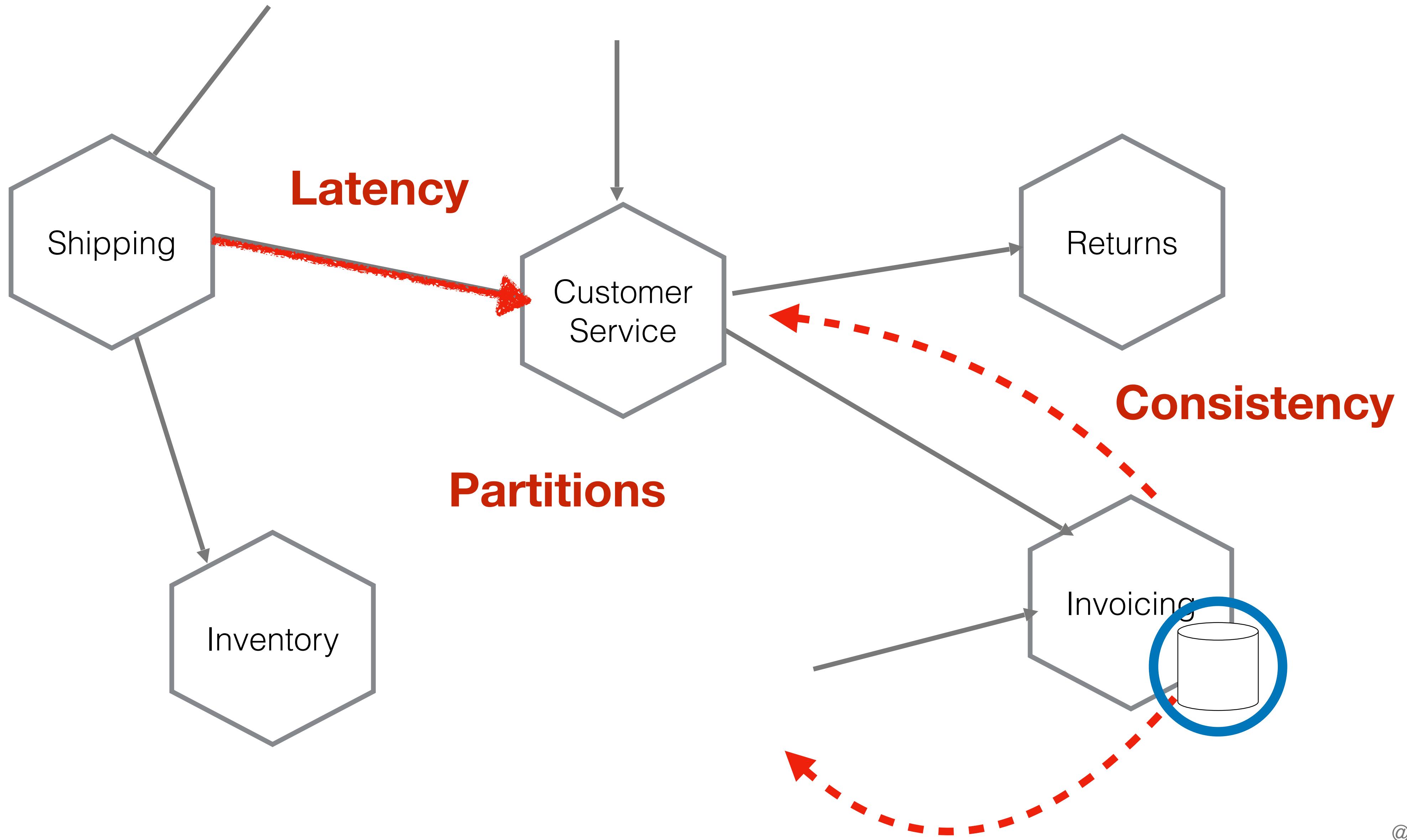
PROBLEMS WITH MICROSERVICES



PROBLEMS WITH MICROSERVICES



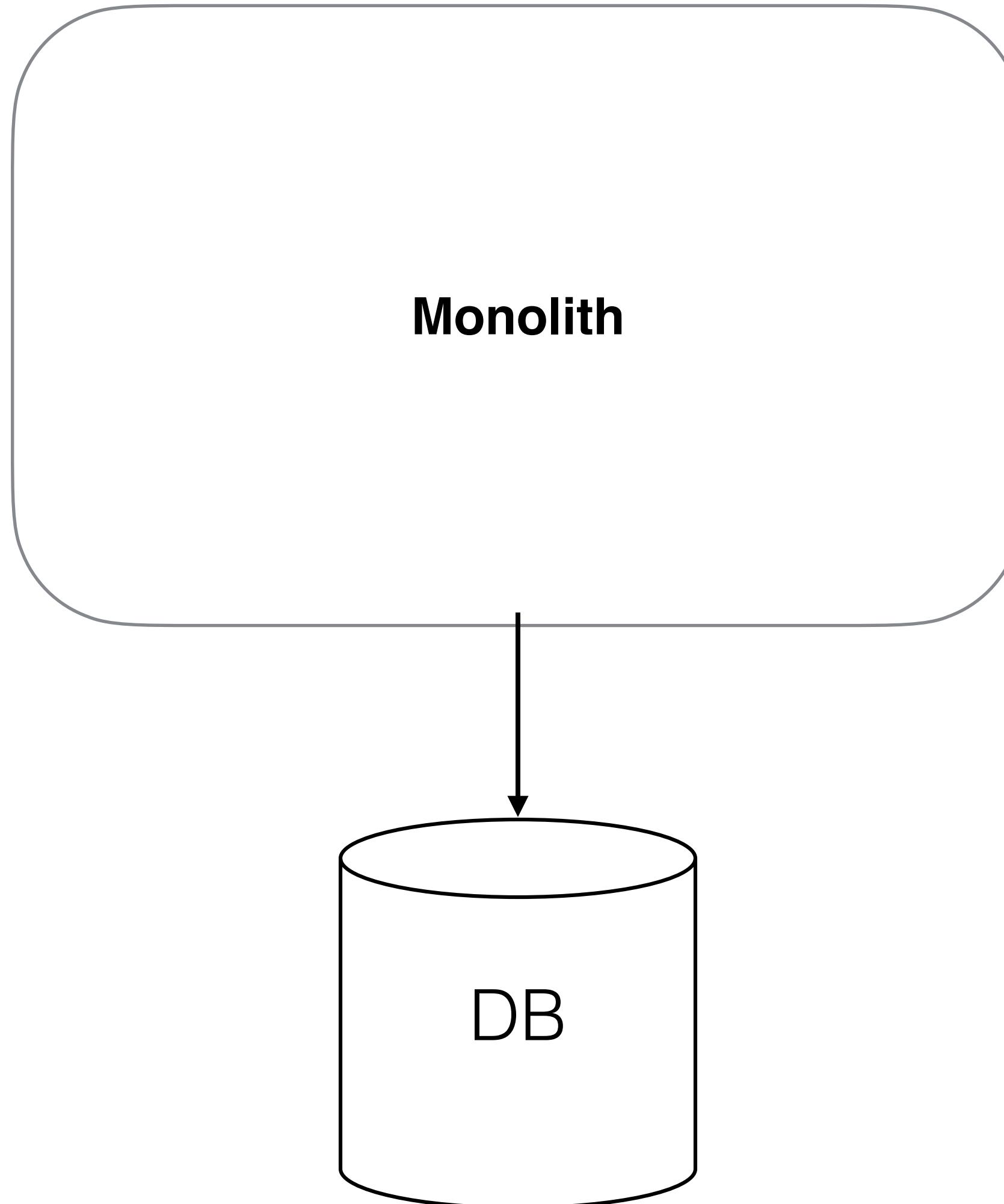
PROBLEMS WITH MICROSERVICES





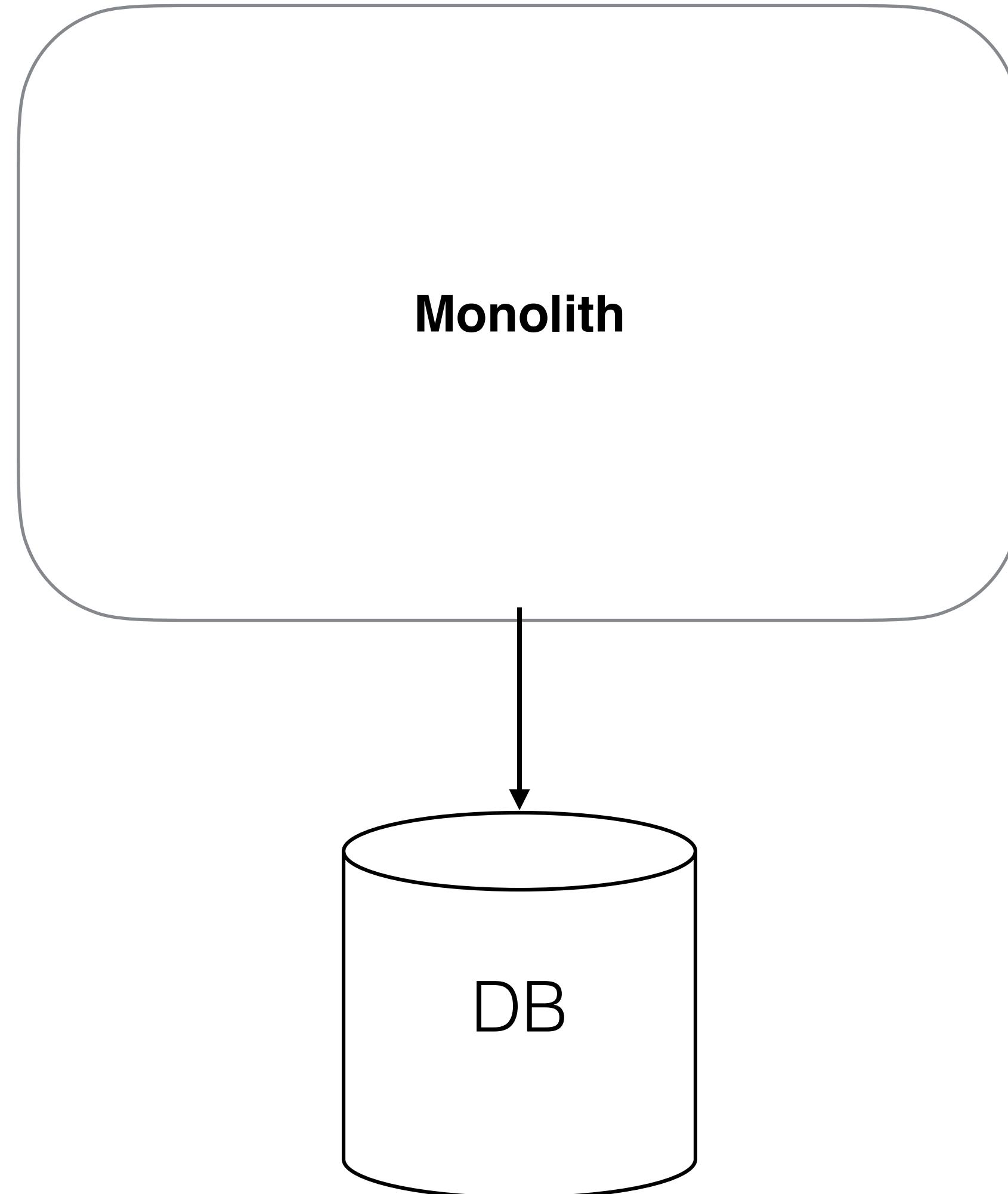
Monolith refers to a unit of deployment

SINGLE PROCESS MONOLITH



SINGLE PROCESS MONOLITH

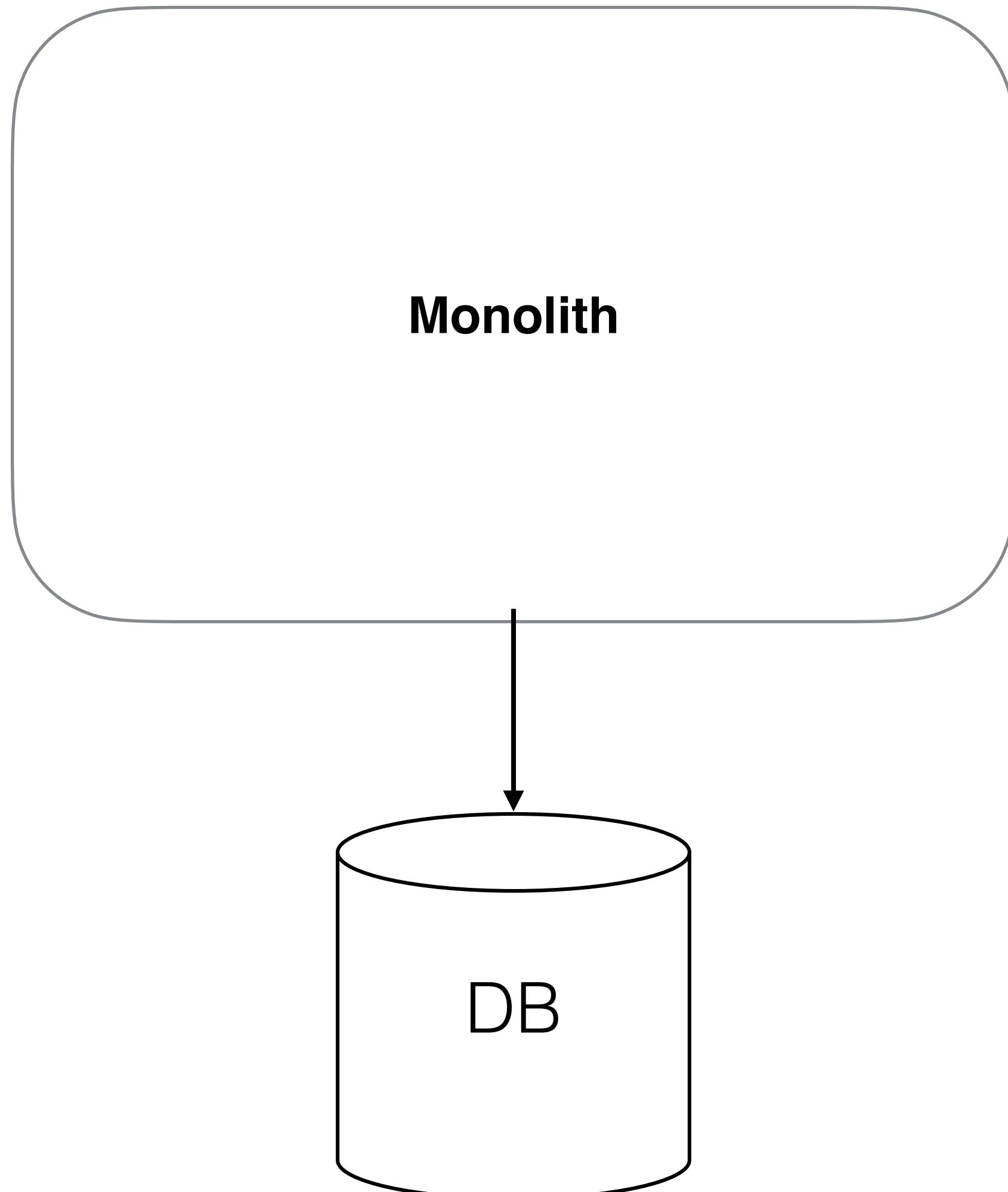
All code packaged
into a single process



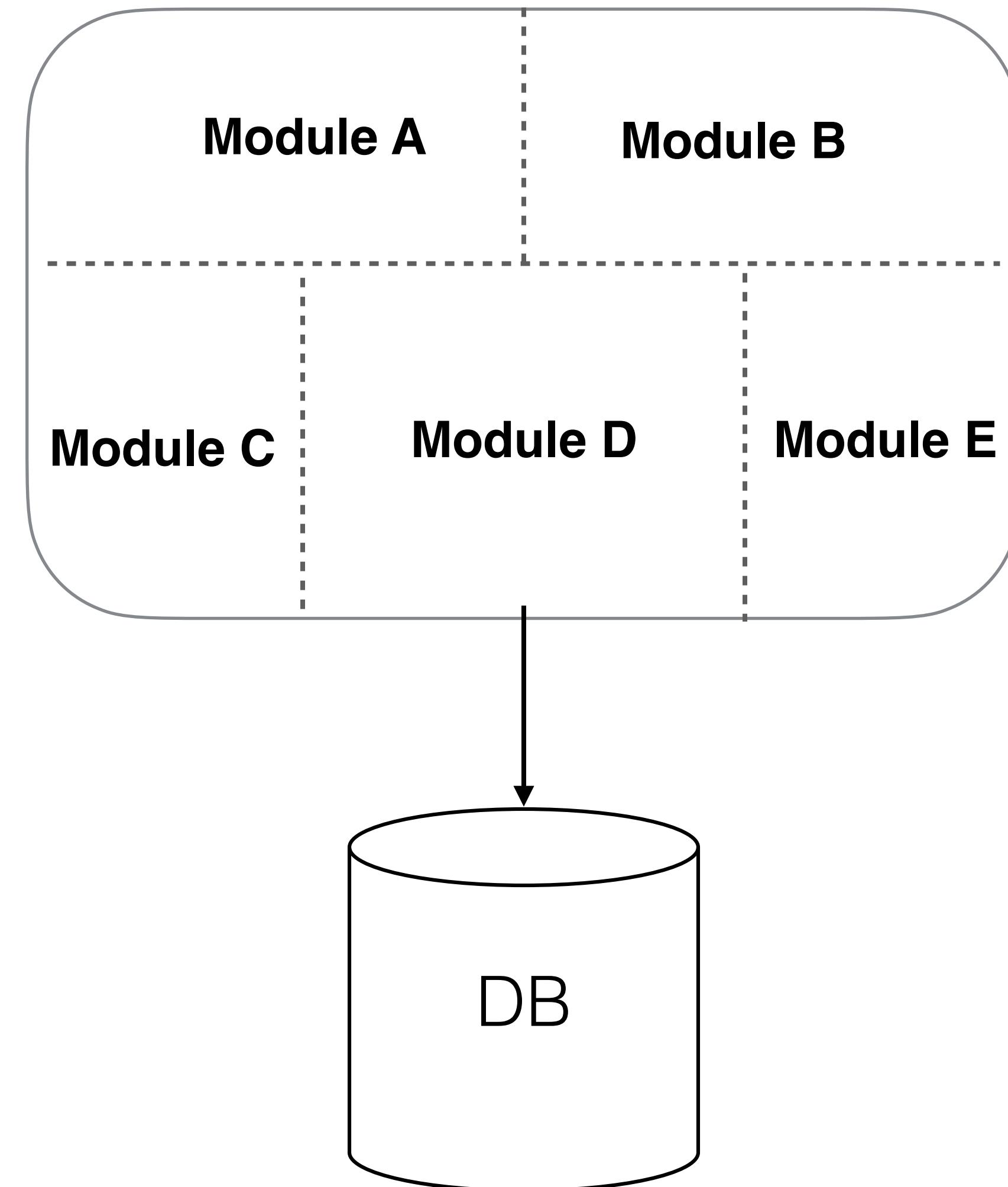
SINGLE PROCESS MONOLITH

All code packaged
into a single process

All data stored in a
single database

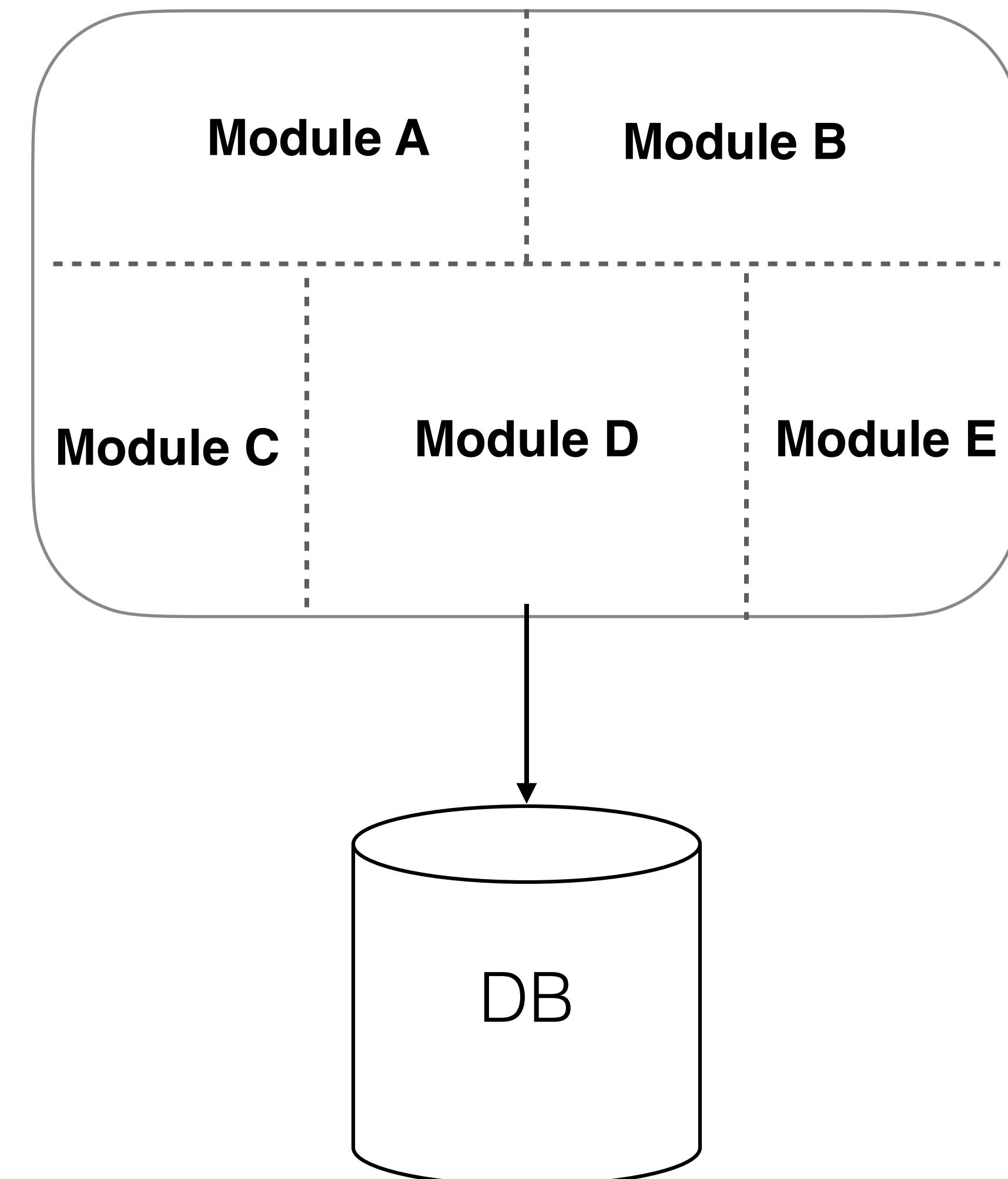


MODULAR MONOLITH



MODULAR MONOLITH

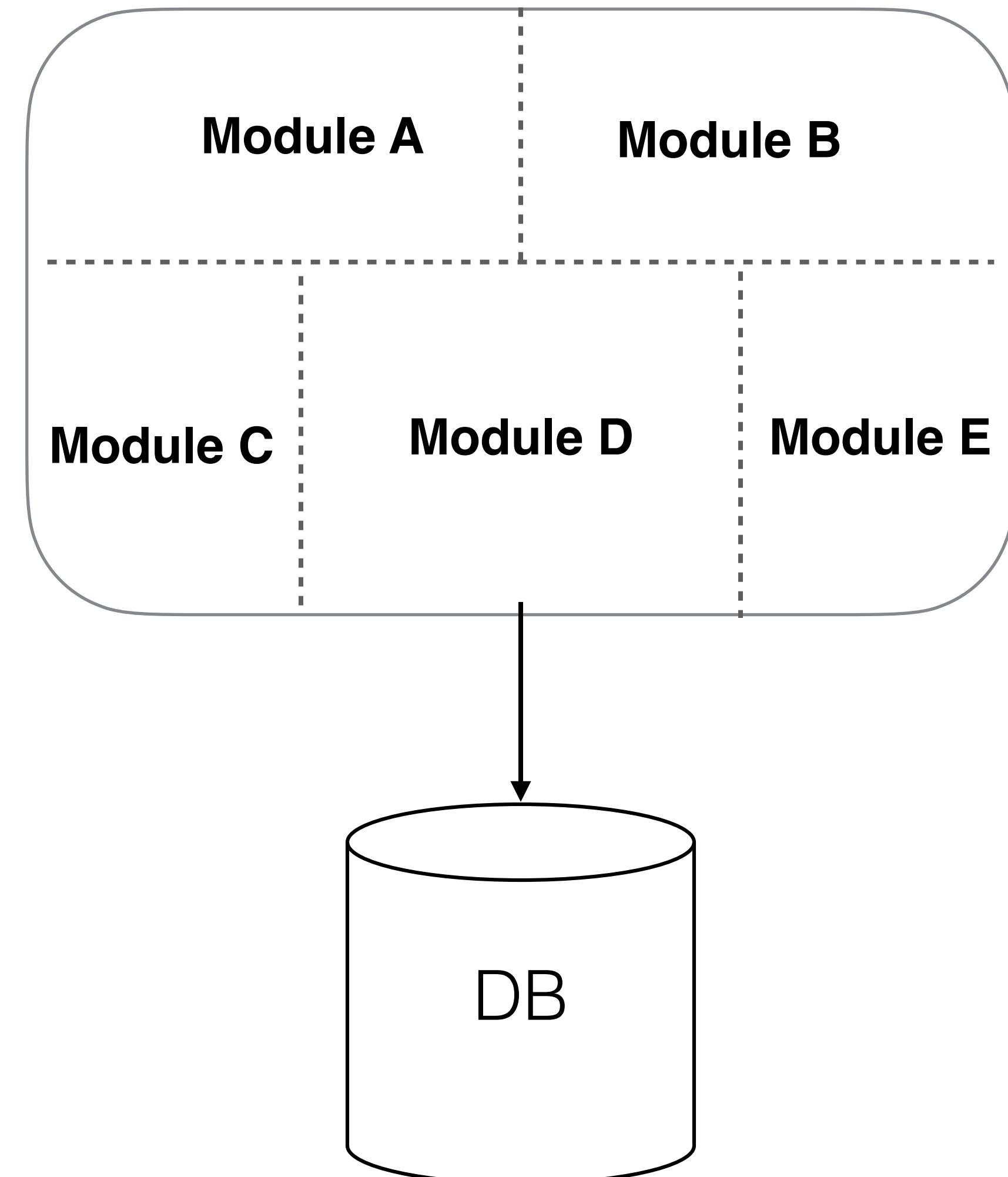
The code is broken
into modules



MODULAR MONOLITH

The code is broken
into modules

Each module
packaged together
into a single process

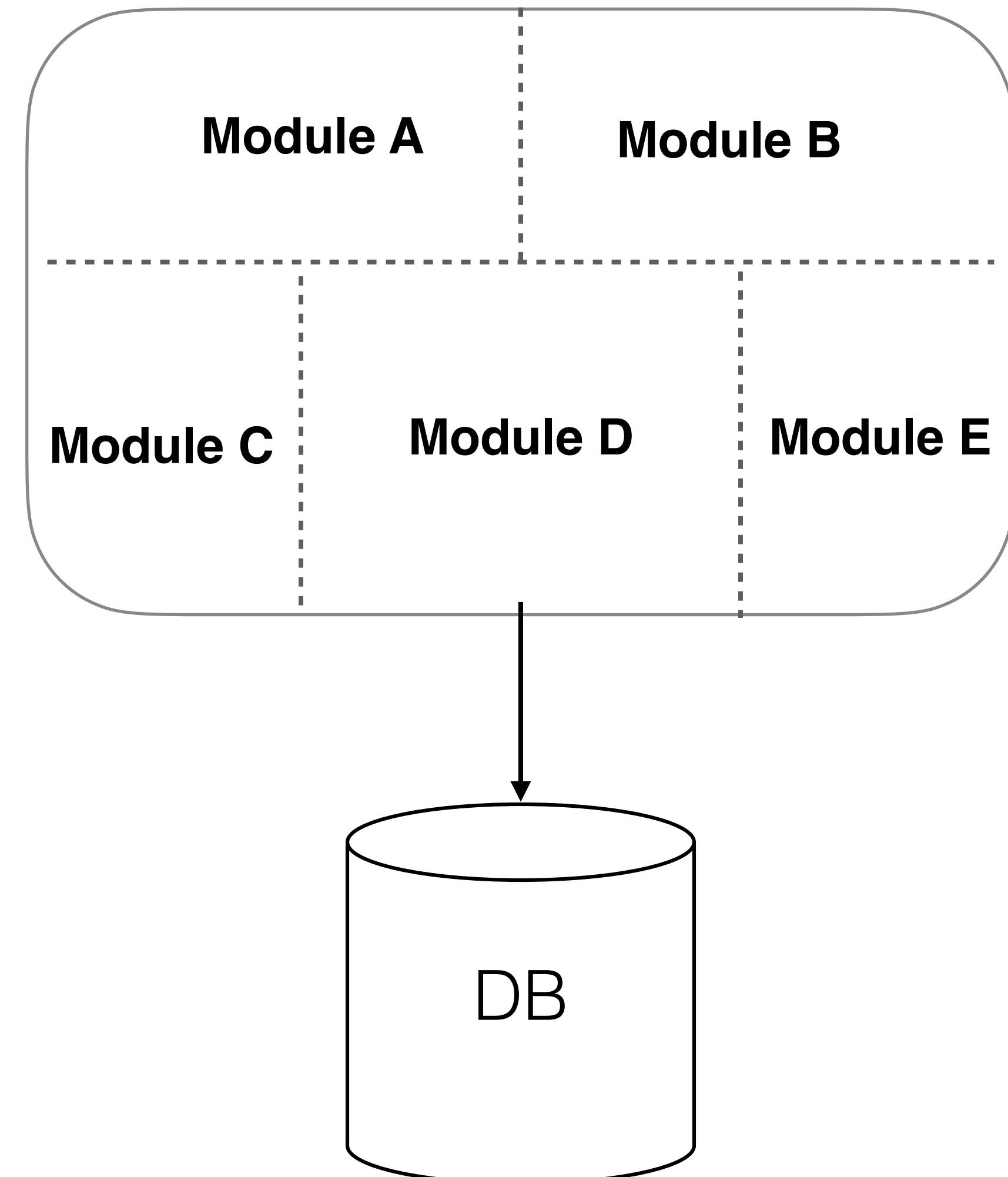


MODULAR MONOLITH

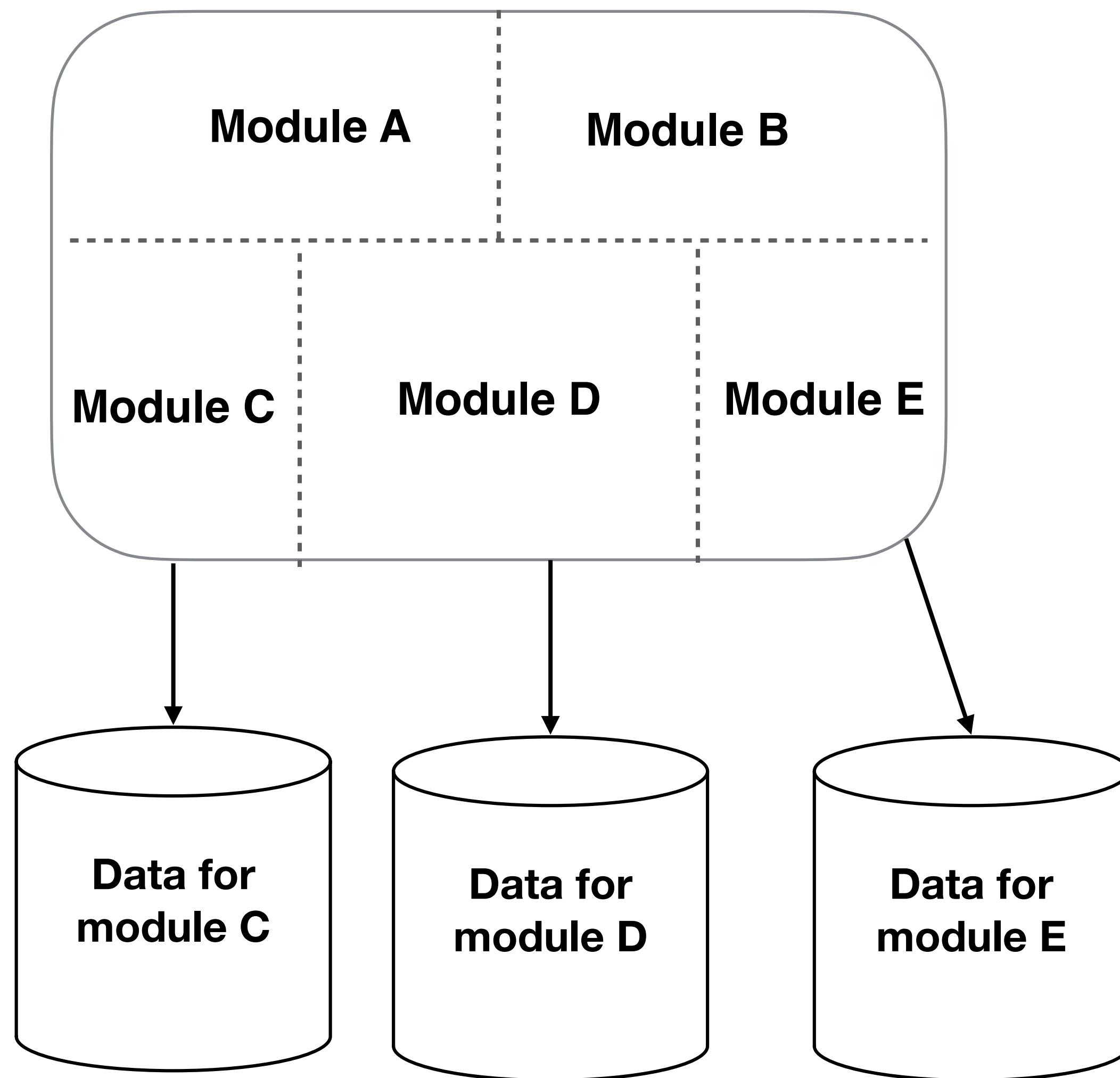
The code is broken
into modules

Each module
packaged together
into a single process

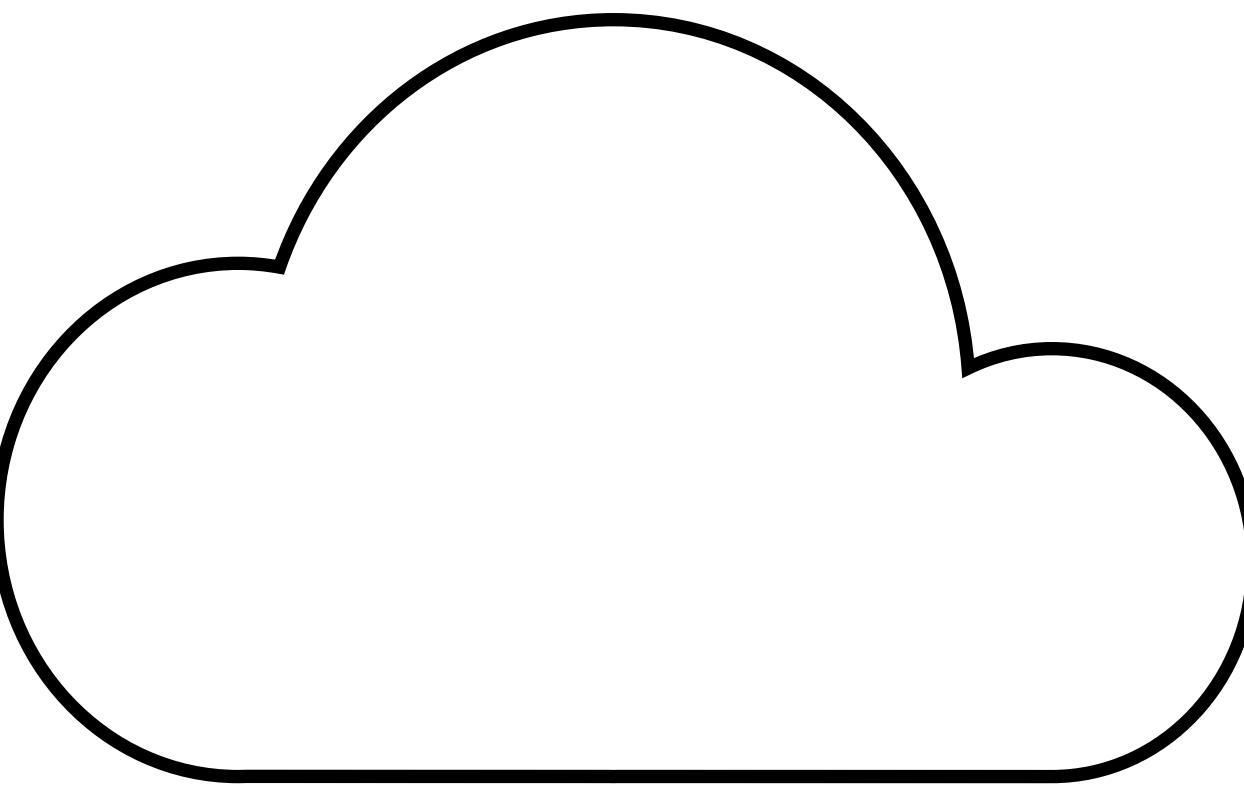
Highly underrated
option



MODULAR DATA

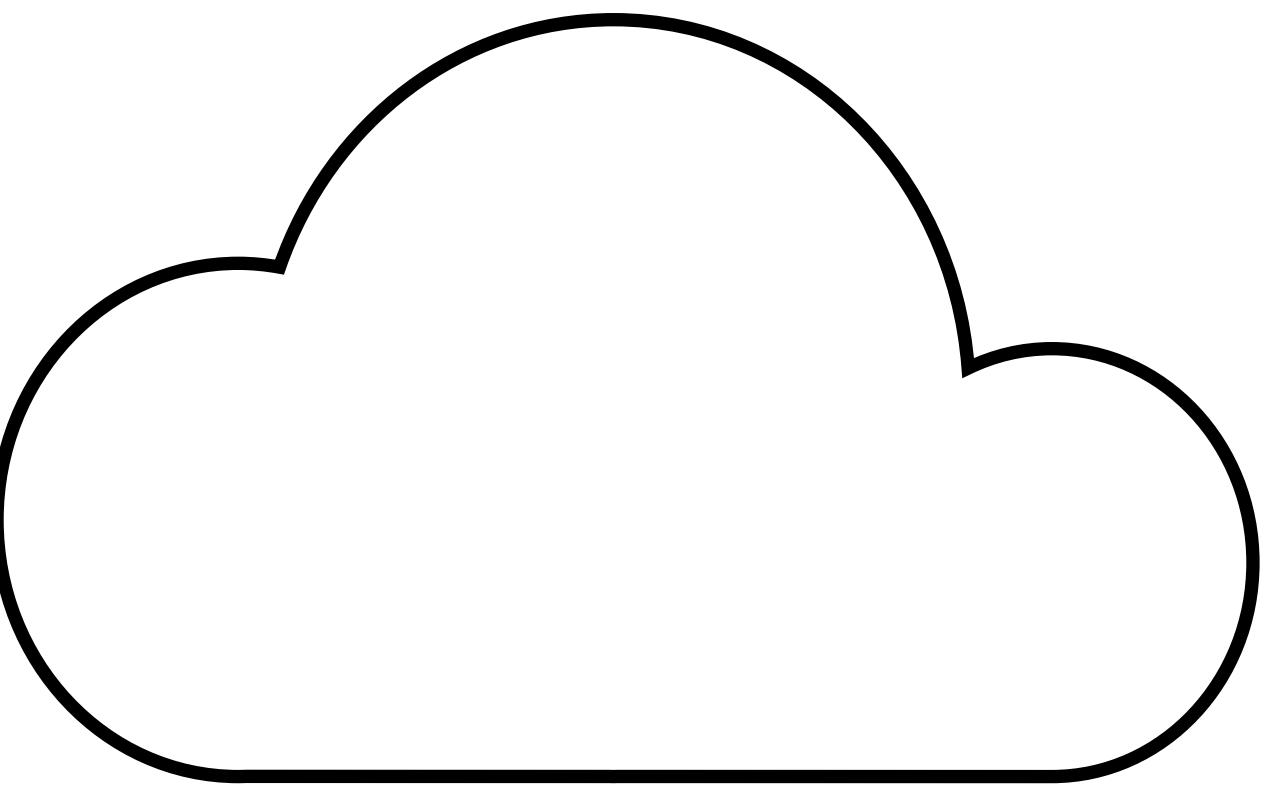


COTS/THIRD-PARTY MONOLITH



COTS/THIRD-PARTY MONOLITH

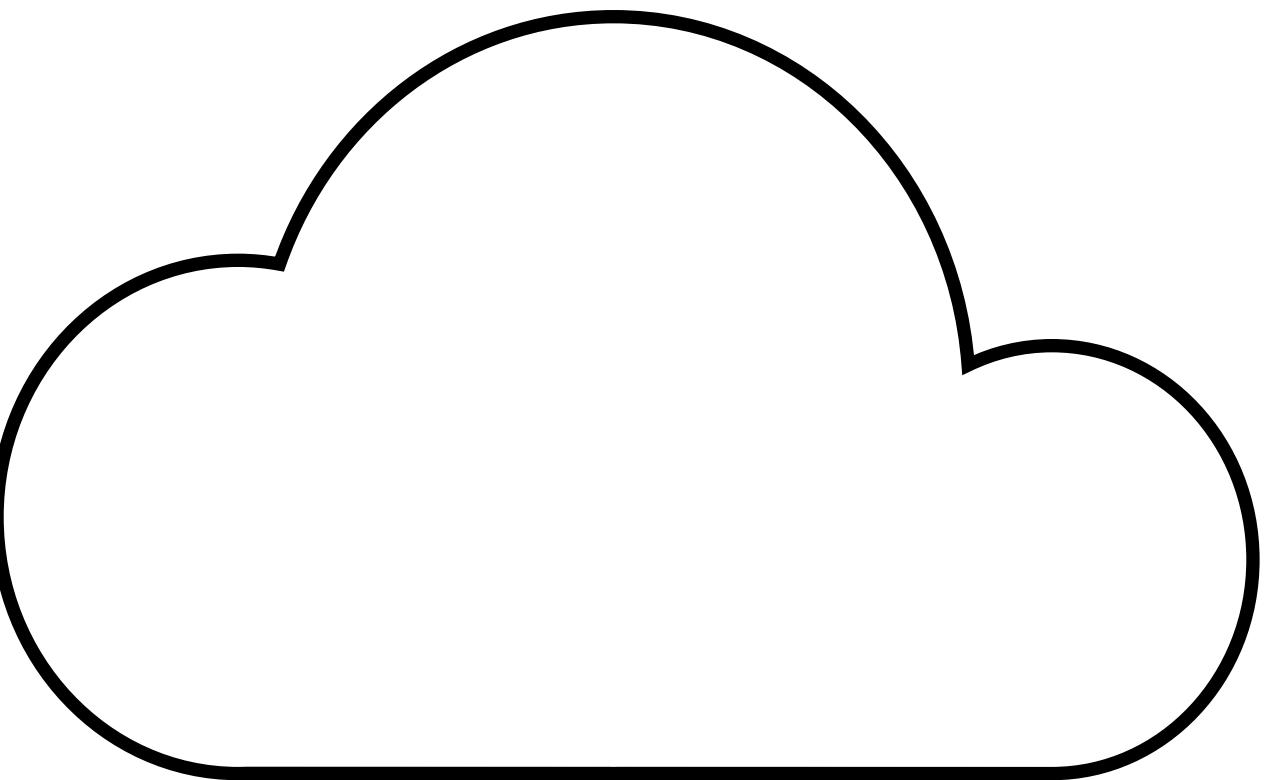
Could be on-prem
software, or a SaaS
product



COTS/THIRD-PARTY MONOLITH

Could be on-prem
software, or a SaaS
product

You have limited to
no ability to change
the core system

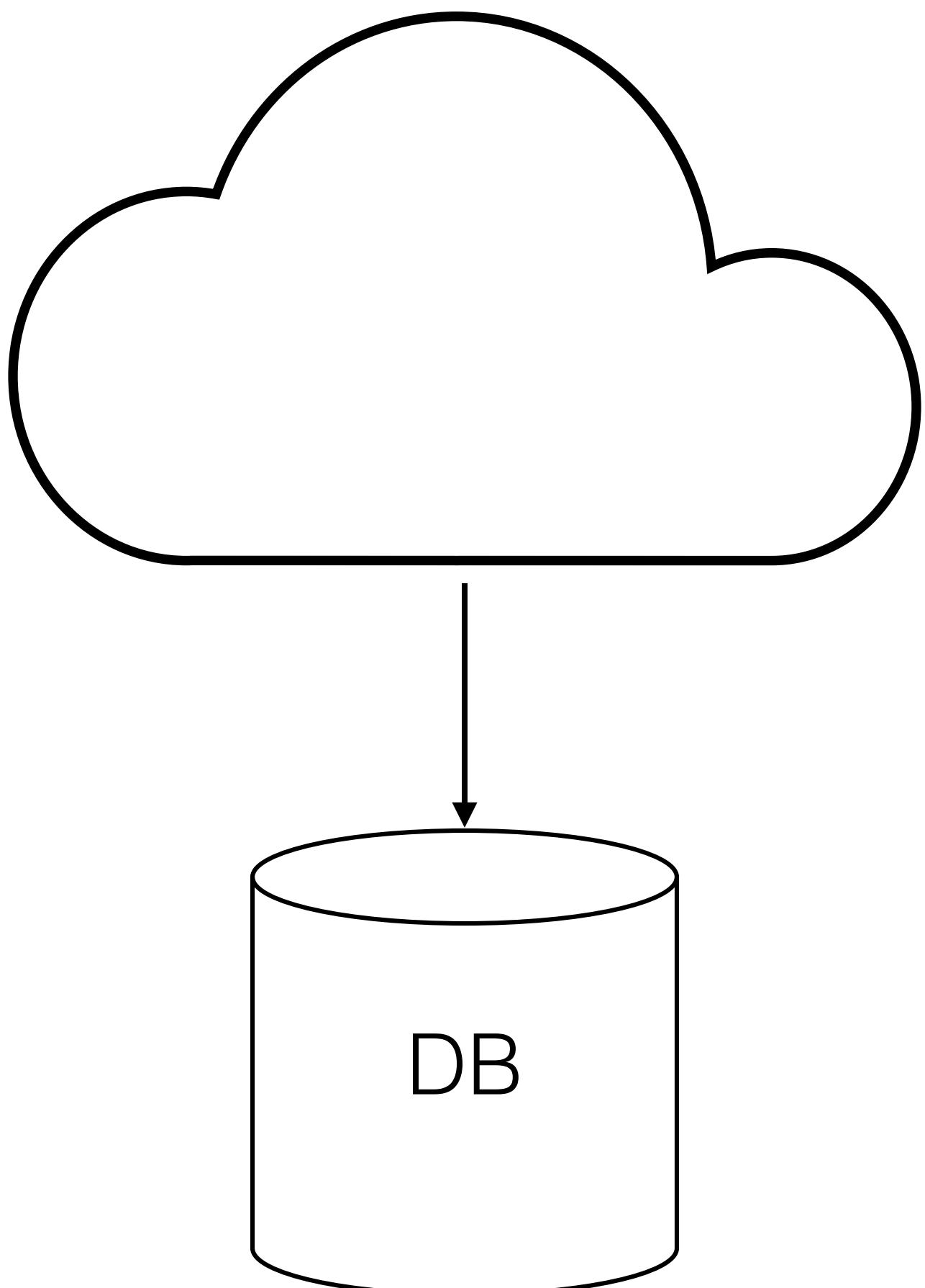


COTS/THIRD-PARTY MONOLITH

Could be on-prem software, or a SaaS product

You have limited to no ability to change the core system

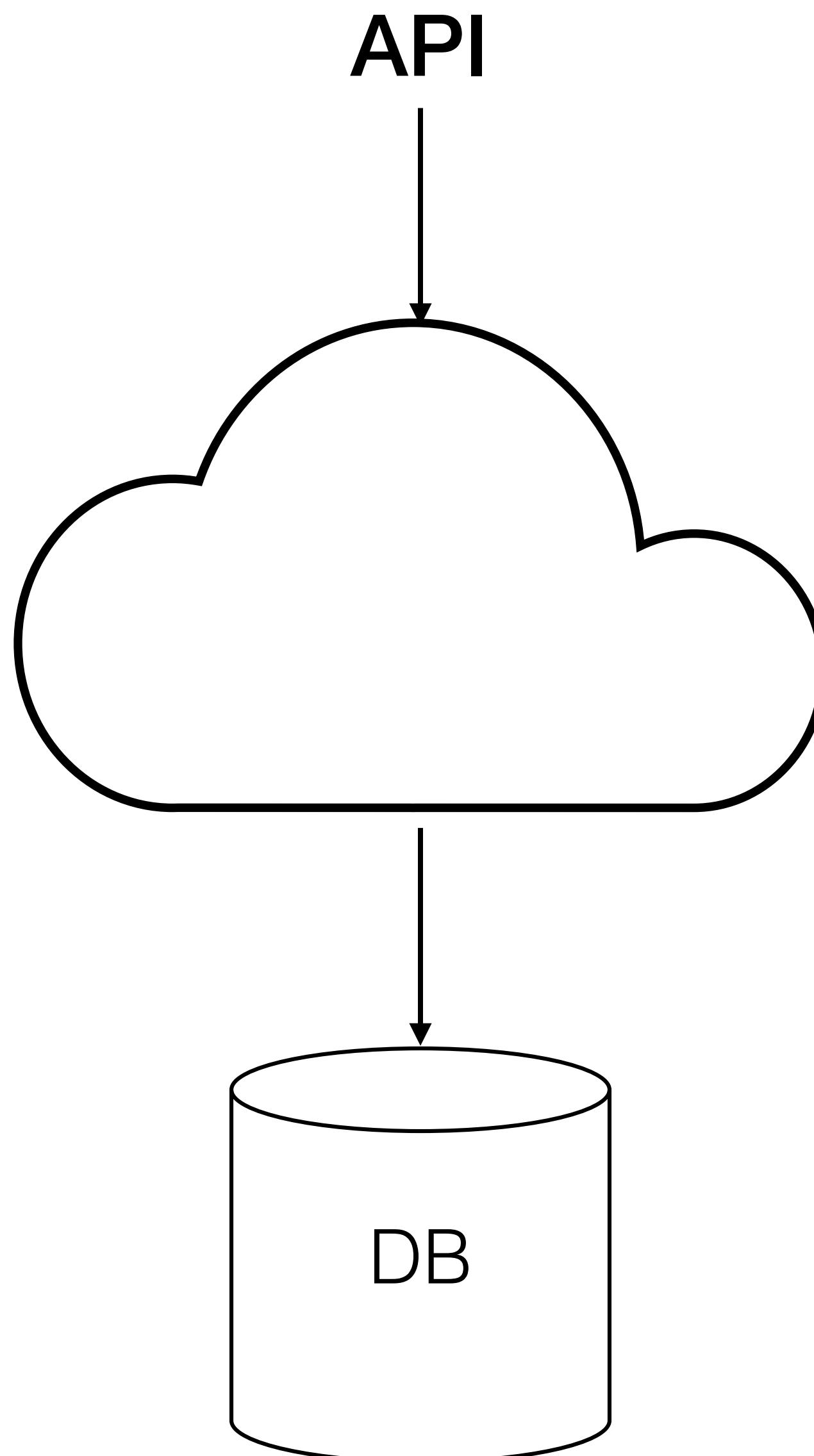
You ***might*** have access to underlying storage...



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

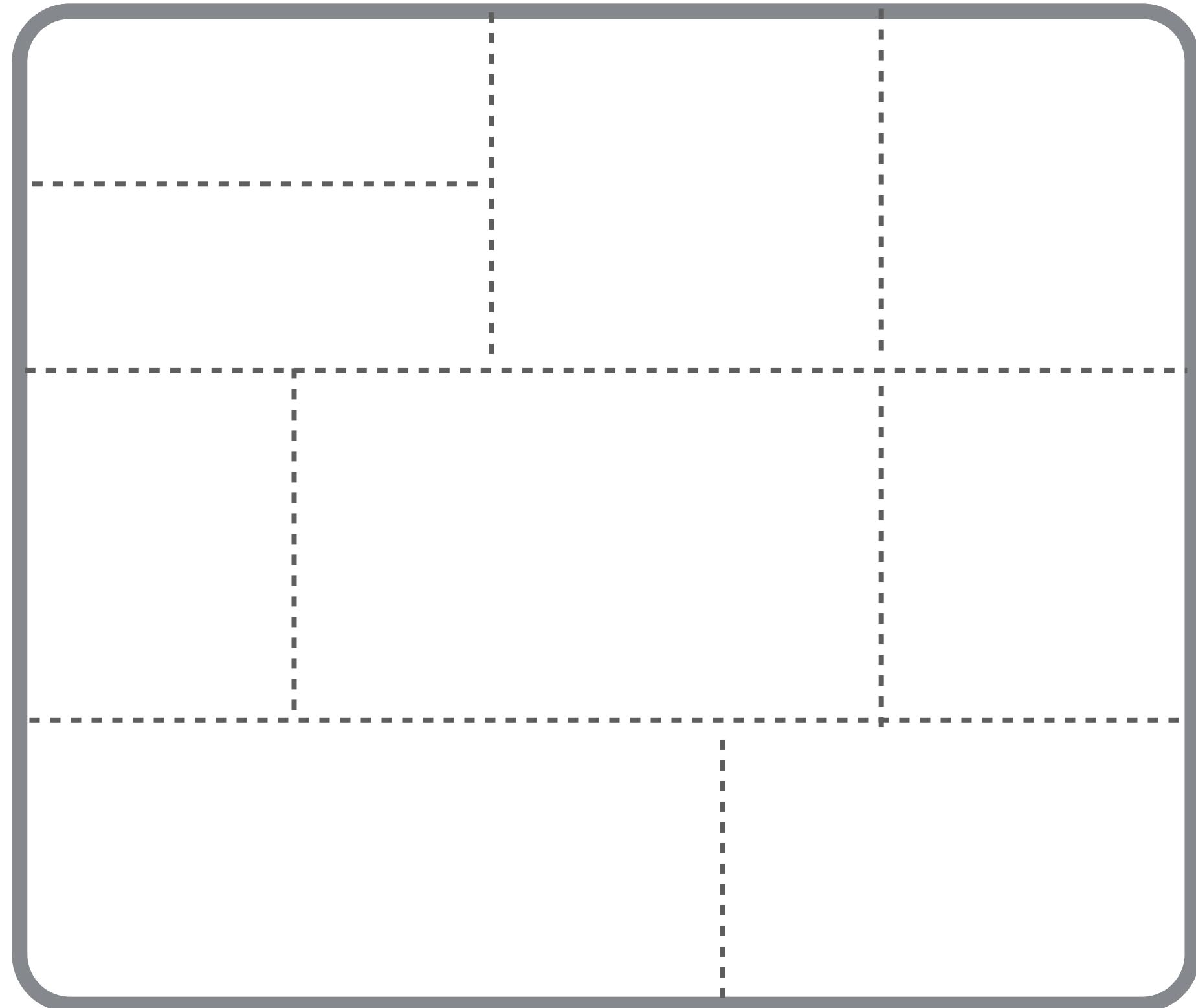
DECOMPOSITIONAL DRIVERS



DECOMPOSITIONAL DRIVERS

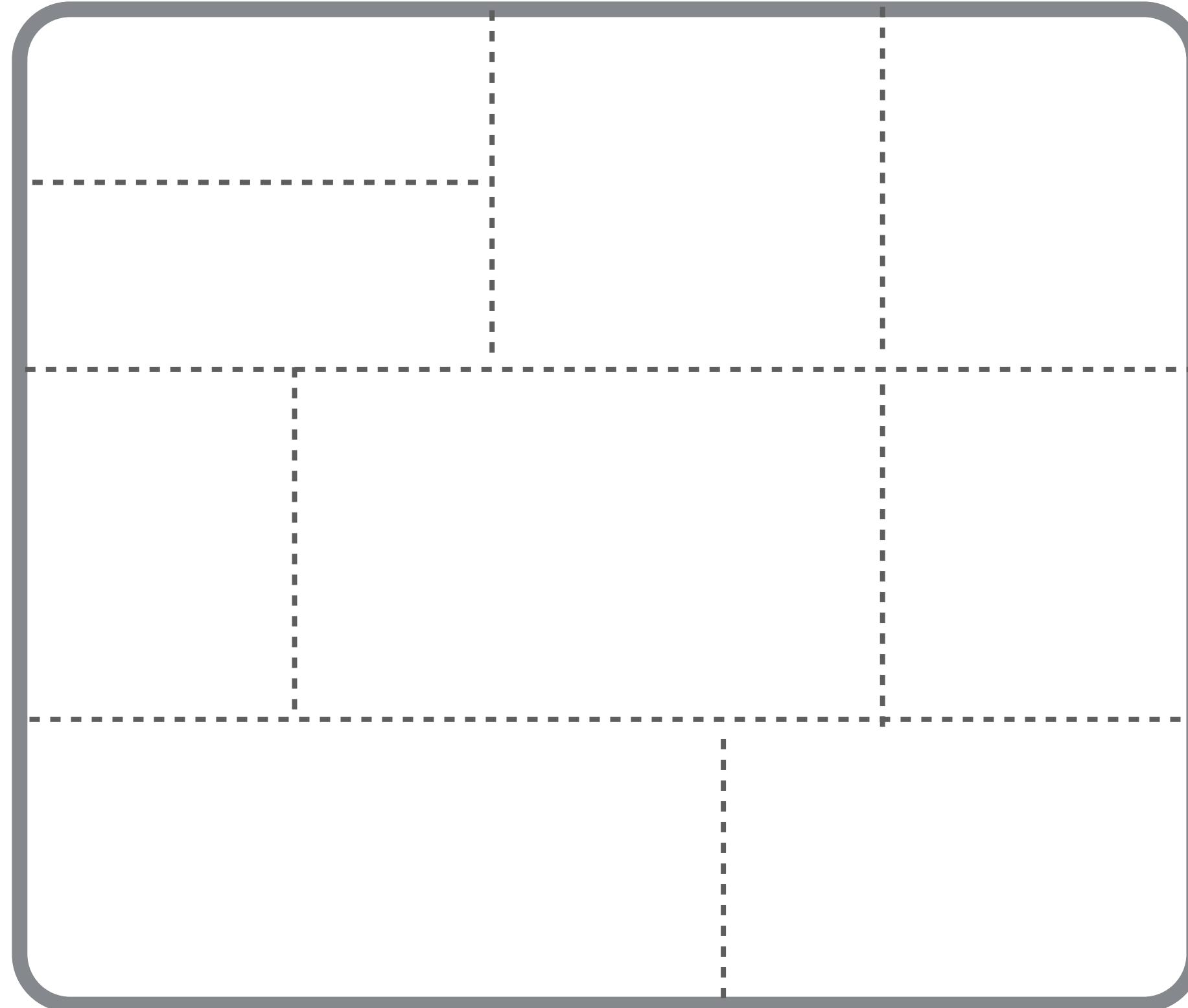


DECOMPOSITIONAL DRIVERS



How do you decide how to break apart your monolith?

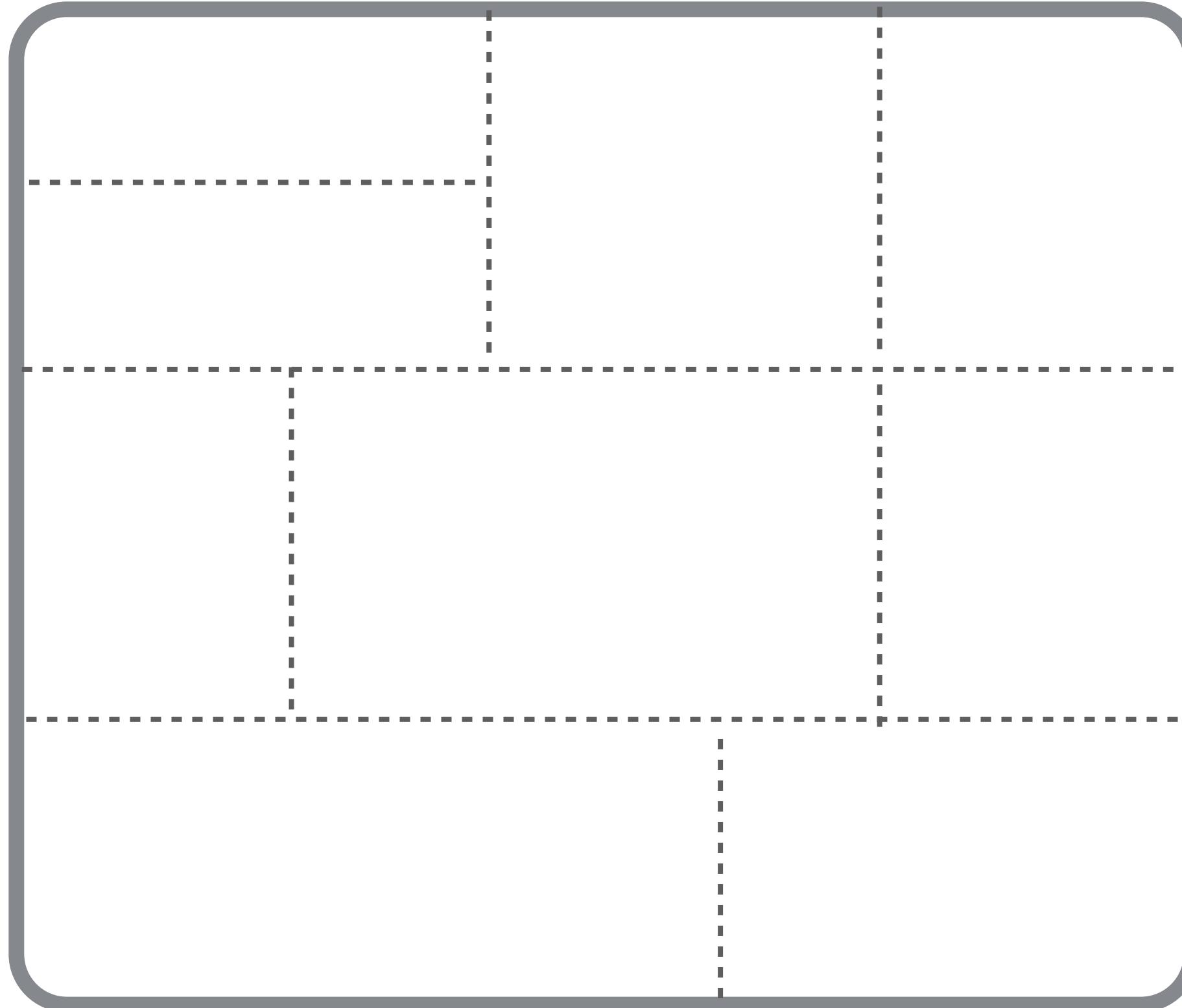
DECOMPOSITIONAL DRIVERS



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

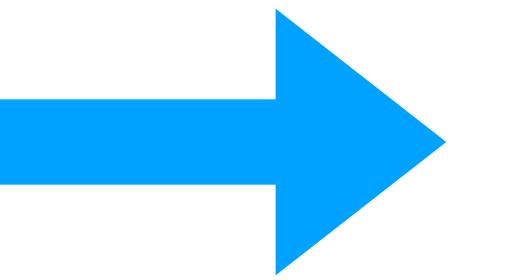
Team autonomy

Scale up size of delivery
organisation

EXAMPLE GOALS & CONCERNS

Time to market

Team autonomy



Scale up size of delivery
organisation

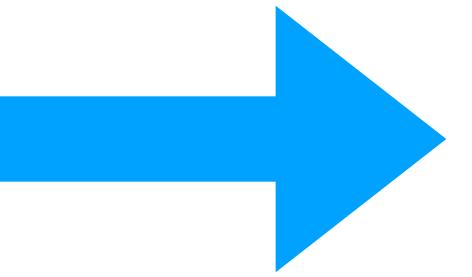
EXAMPLE GOALS & CONCERNS

Time to market

Team autonomy

Scale up size of delivery organisation

Primary Concerns
Domain-driven Design

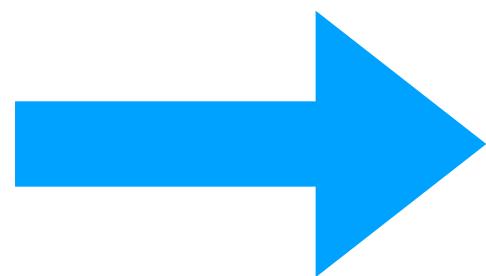


EXAMPLE GOALS & CONCERNS

Time to market

Team autonomy

Scale up size of delivery organisation



Primary Concerns

Domain-driven Design

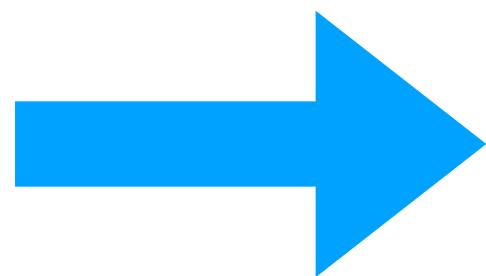
Organisational Structure

EXAMPLE GOALS & CONCERNS

Time to market

Team autonomy

Scale up size of delivery organisation



Primary Concerns

Domain-driven Design

Organisational Structure

Secondary Concern

Volatility

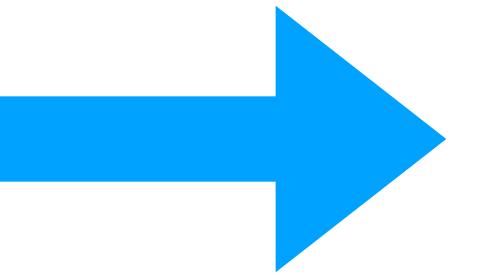
EXAMPLE GOALS & CONCERNS (CONT)

EXAMPLE GOALS & CONCERNS (CONT)

Application Scale

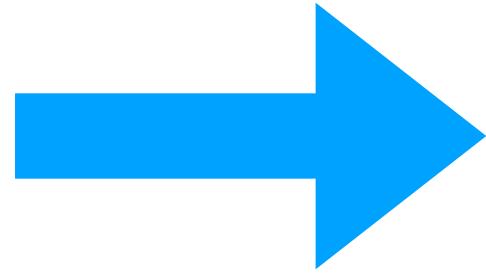
EXAMPLE GOALS & CONCERNS (CONT)

**Application
Scale**



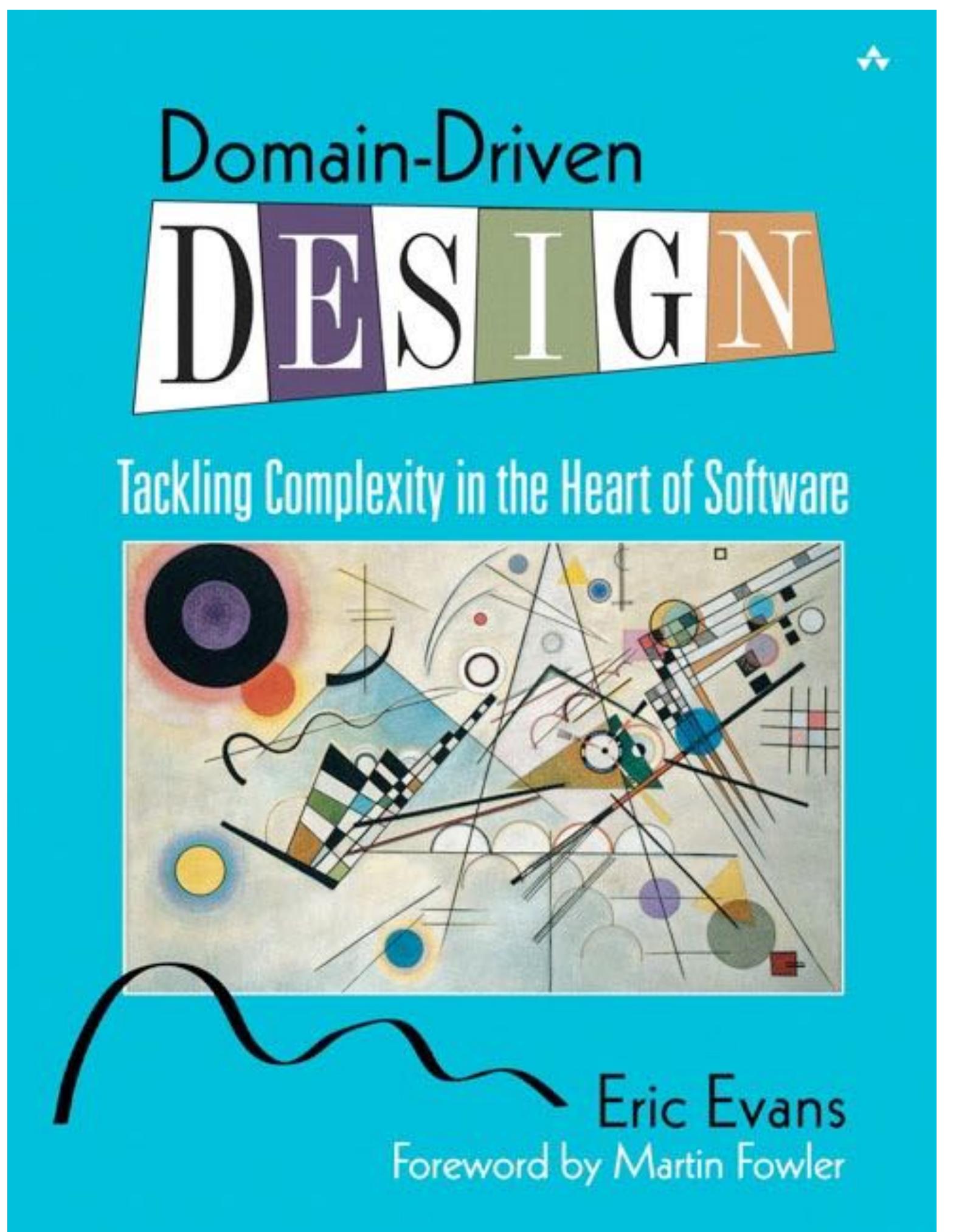
EXAMPLE GOALS & CONCERNS (CONT)

**Application
Scale**

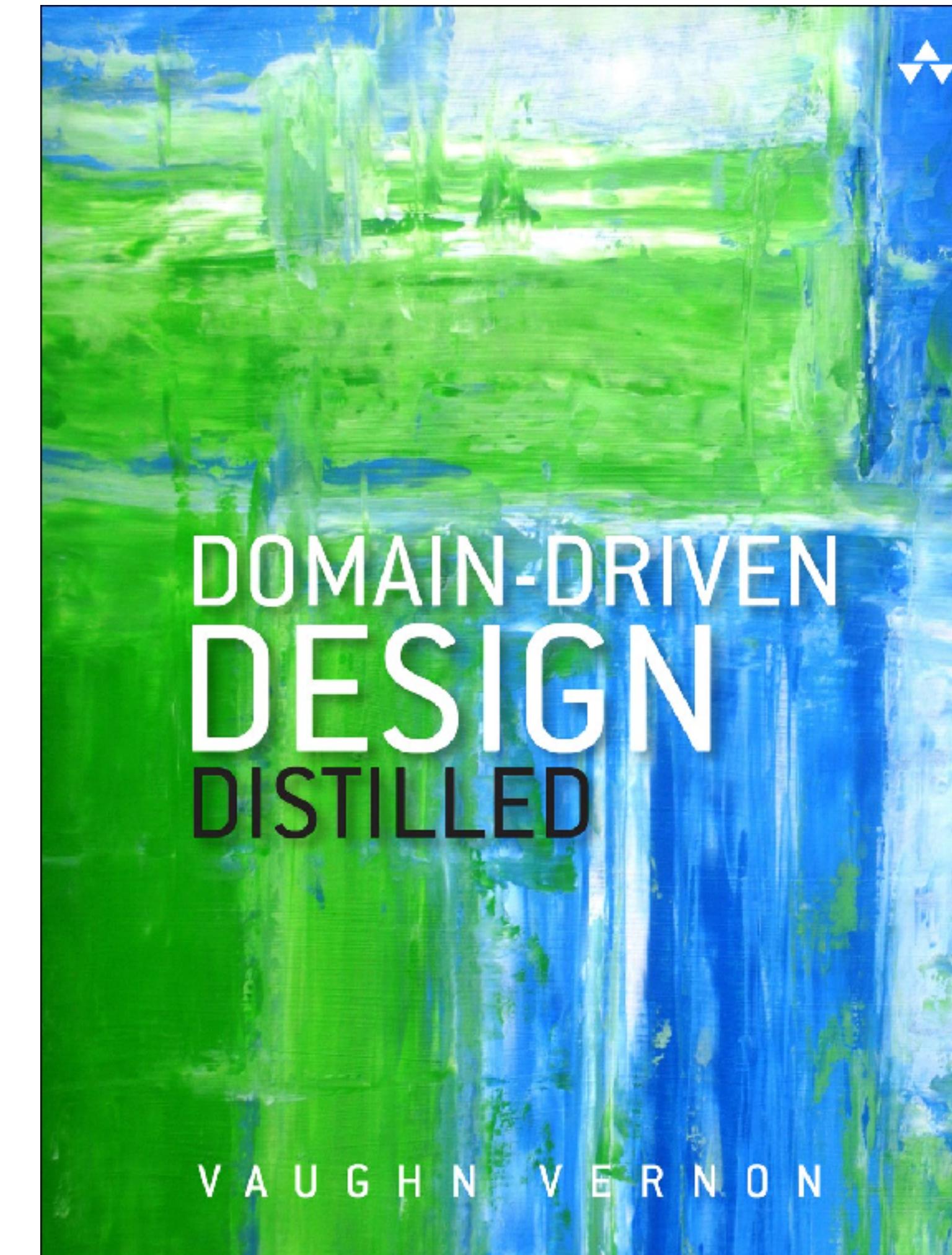
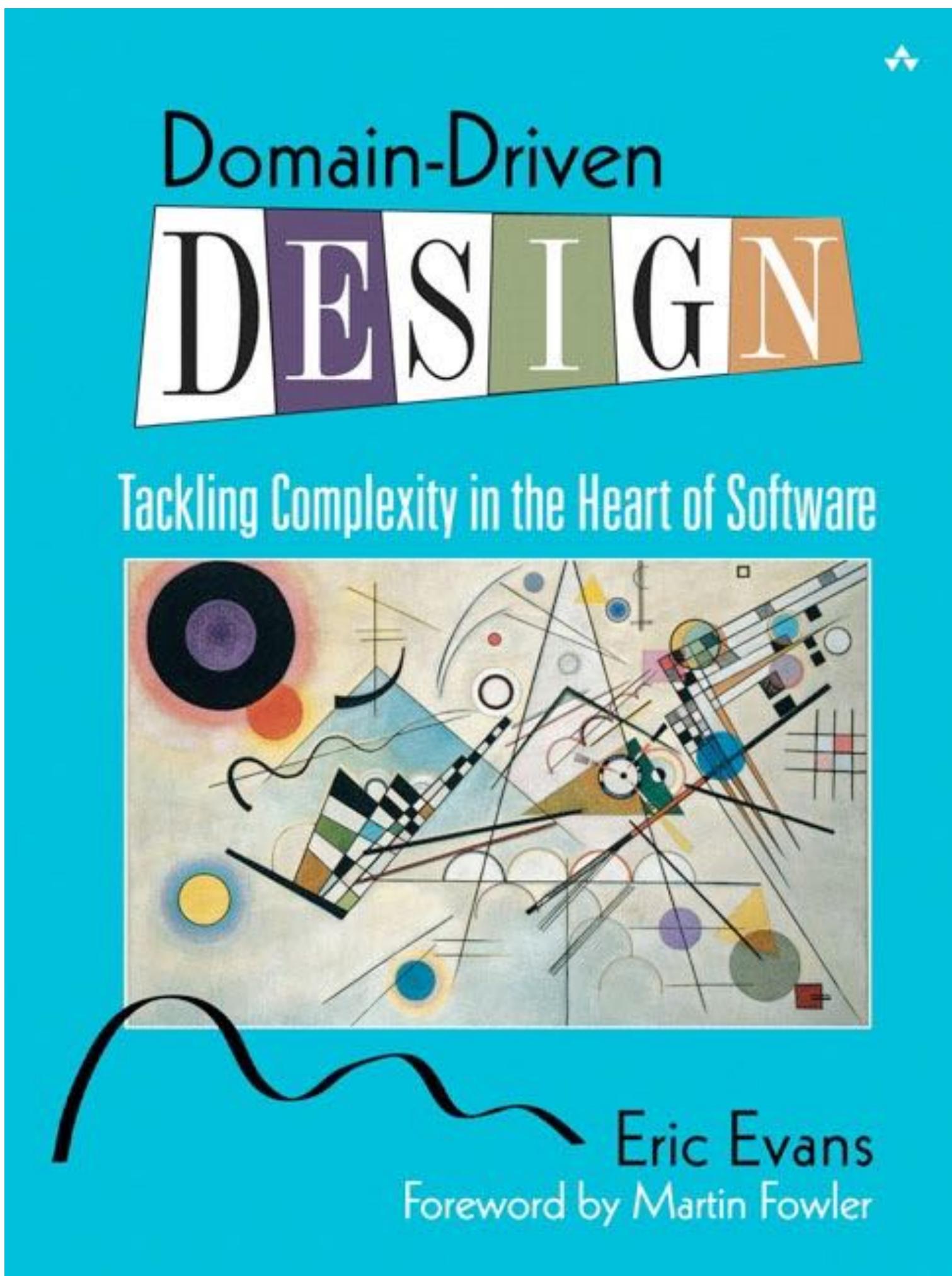


**Primary Concern
Bottlenecks**

DOMAIN-DRIVEN DESIGN



DOMAIN-DRIVEN DESIGN



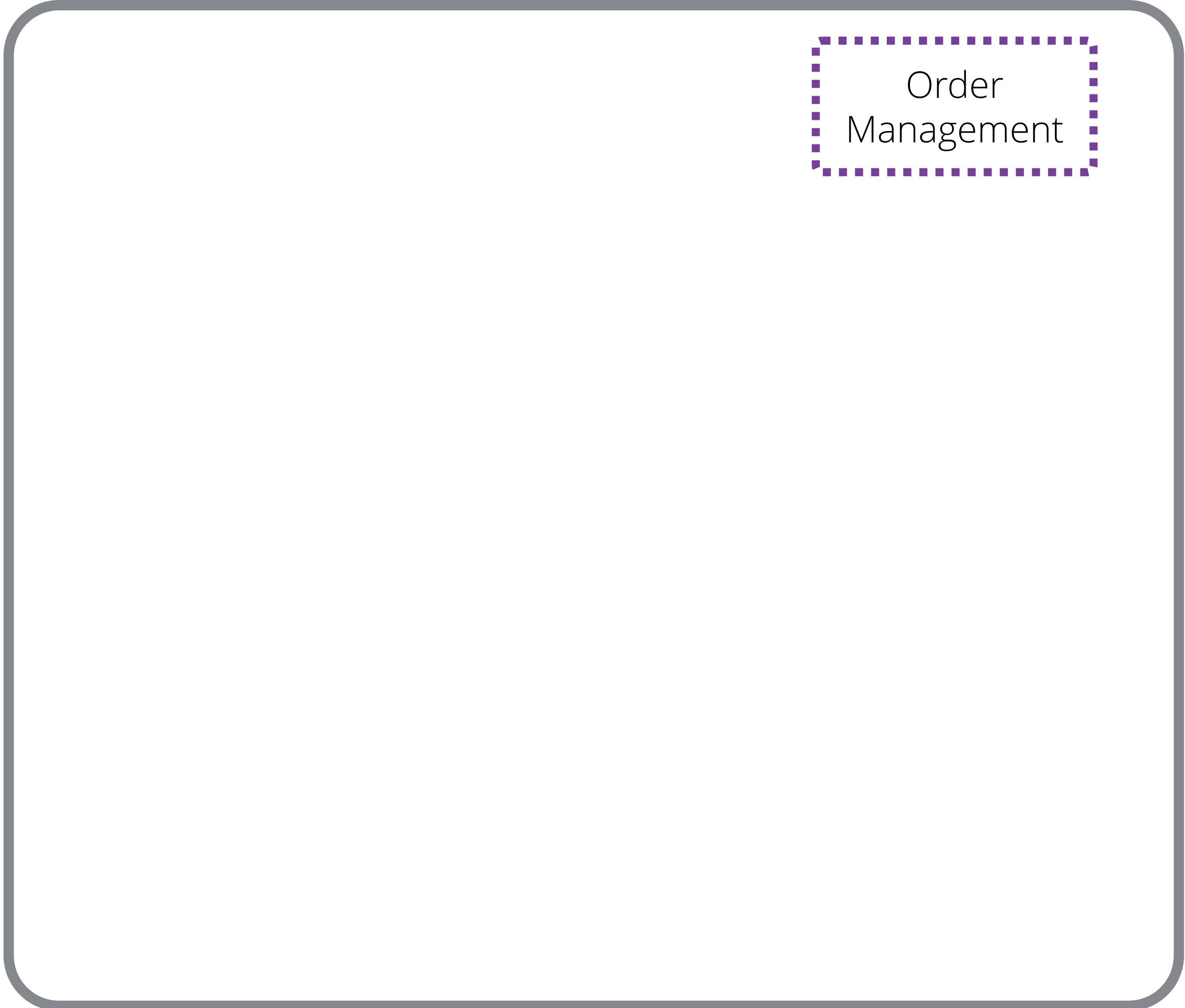
Poll

How many of you know about Domain-Driven Design?

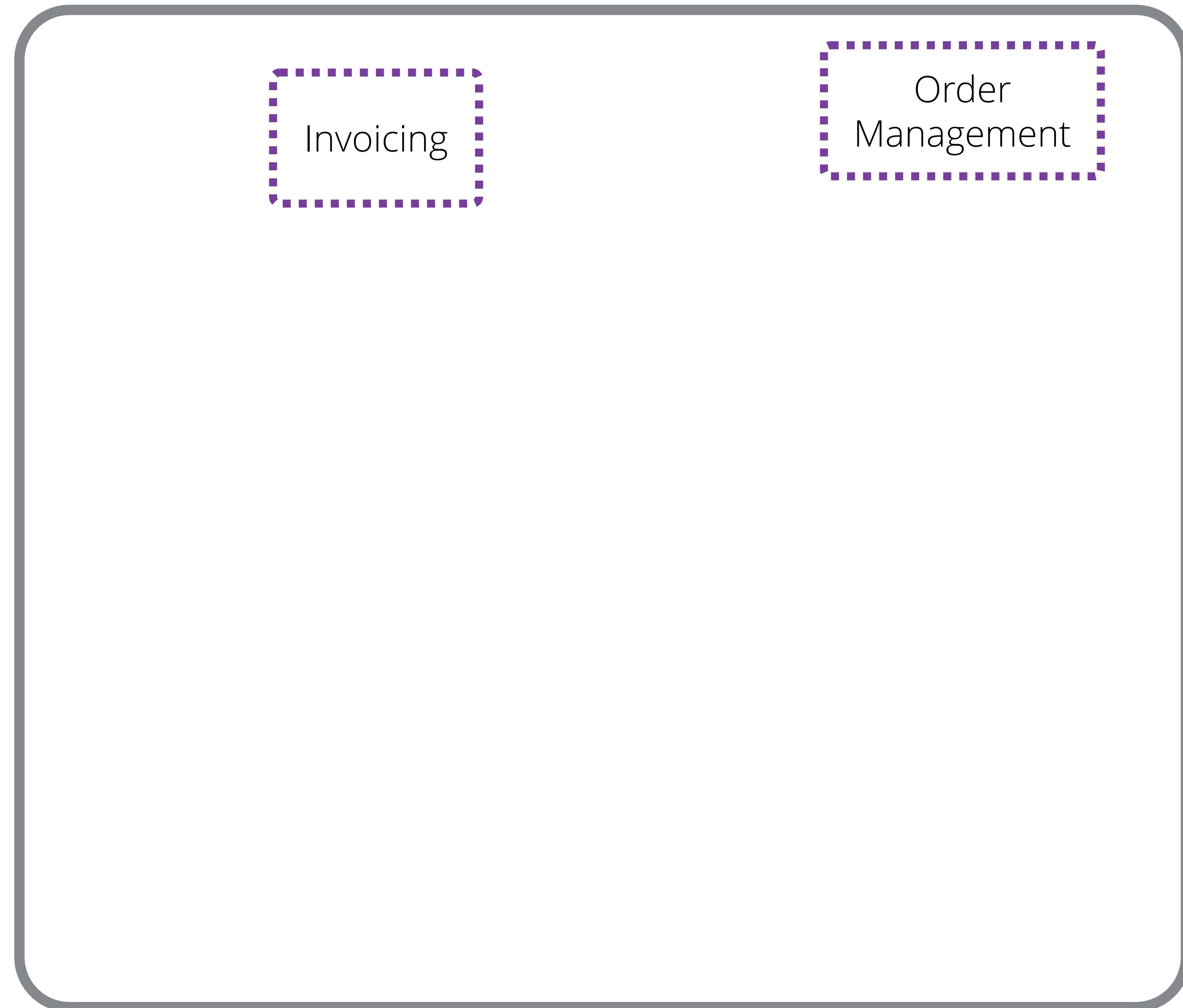
- a.) I don't know about it
- b.) I know a bit about it, but haven't used it
- c.) We use it!

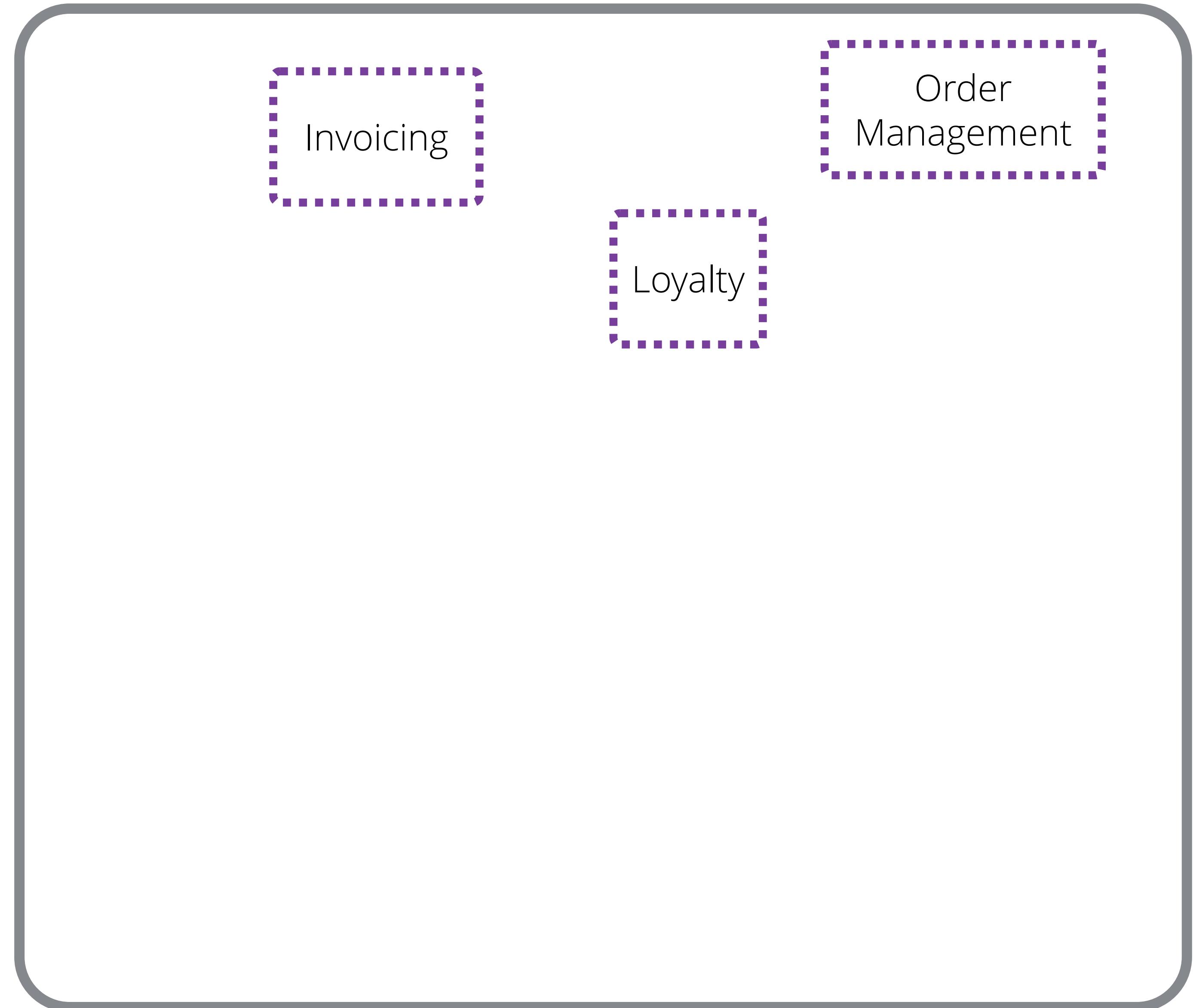


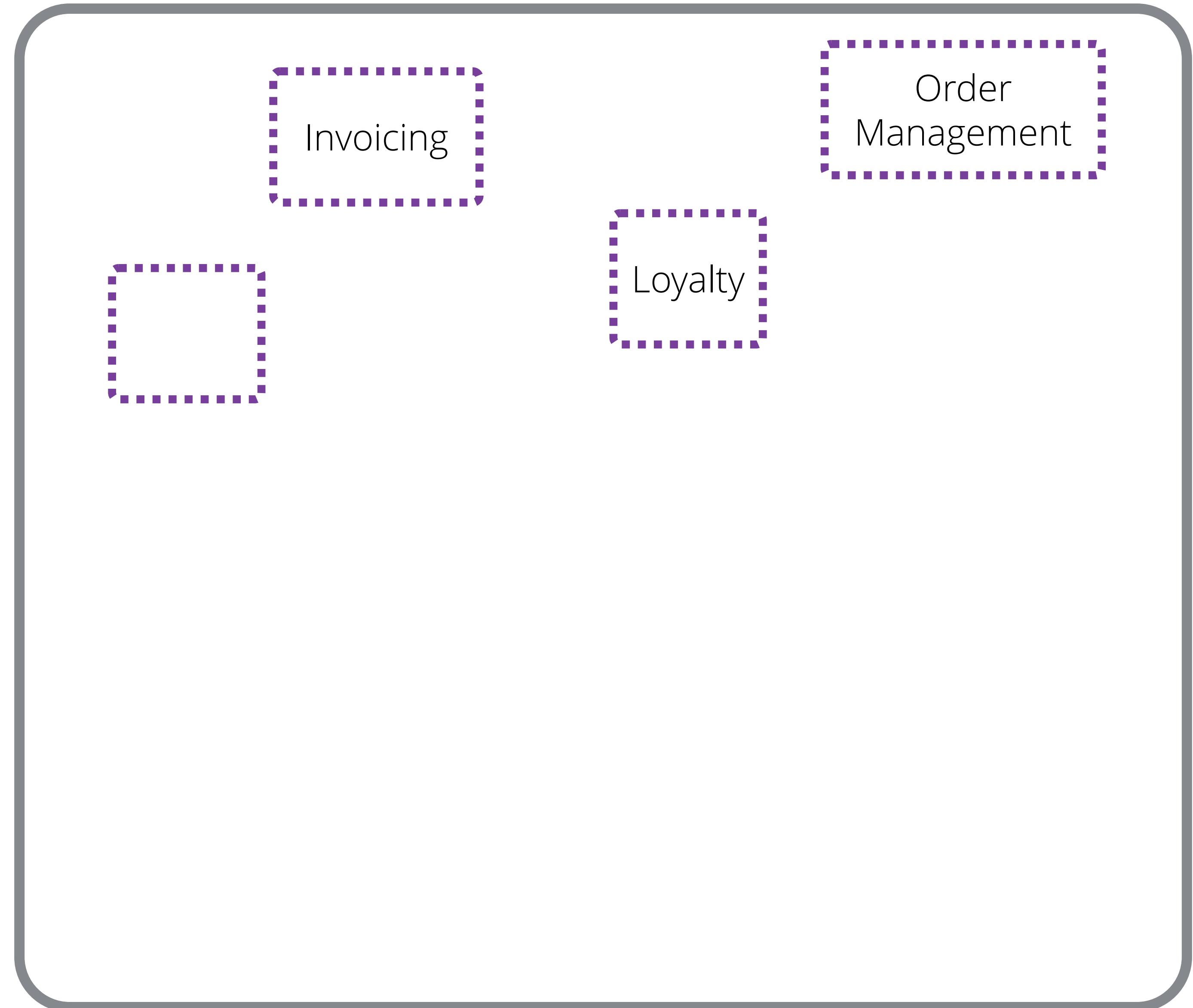
@samnewman

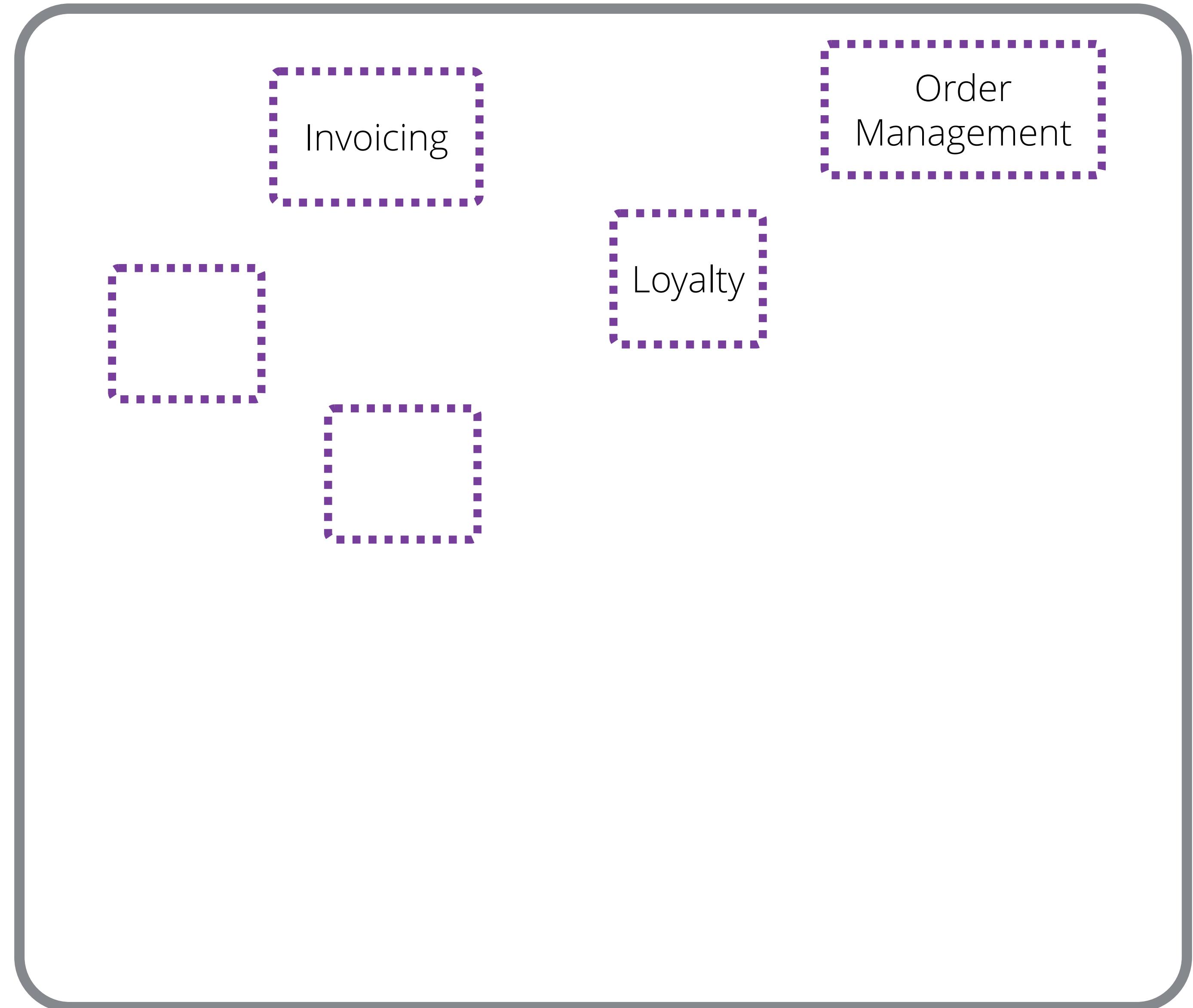


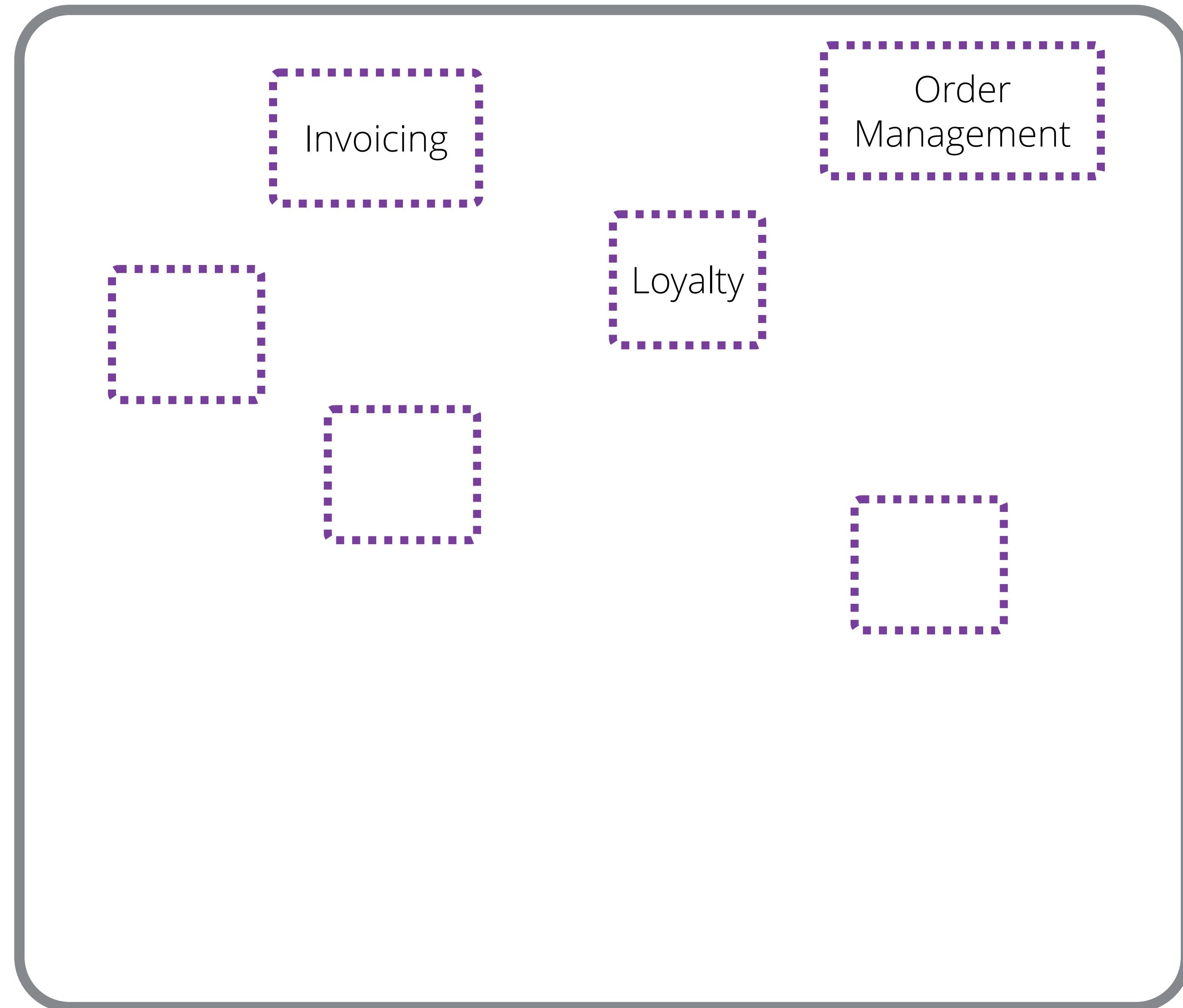
Order
Management

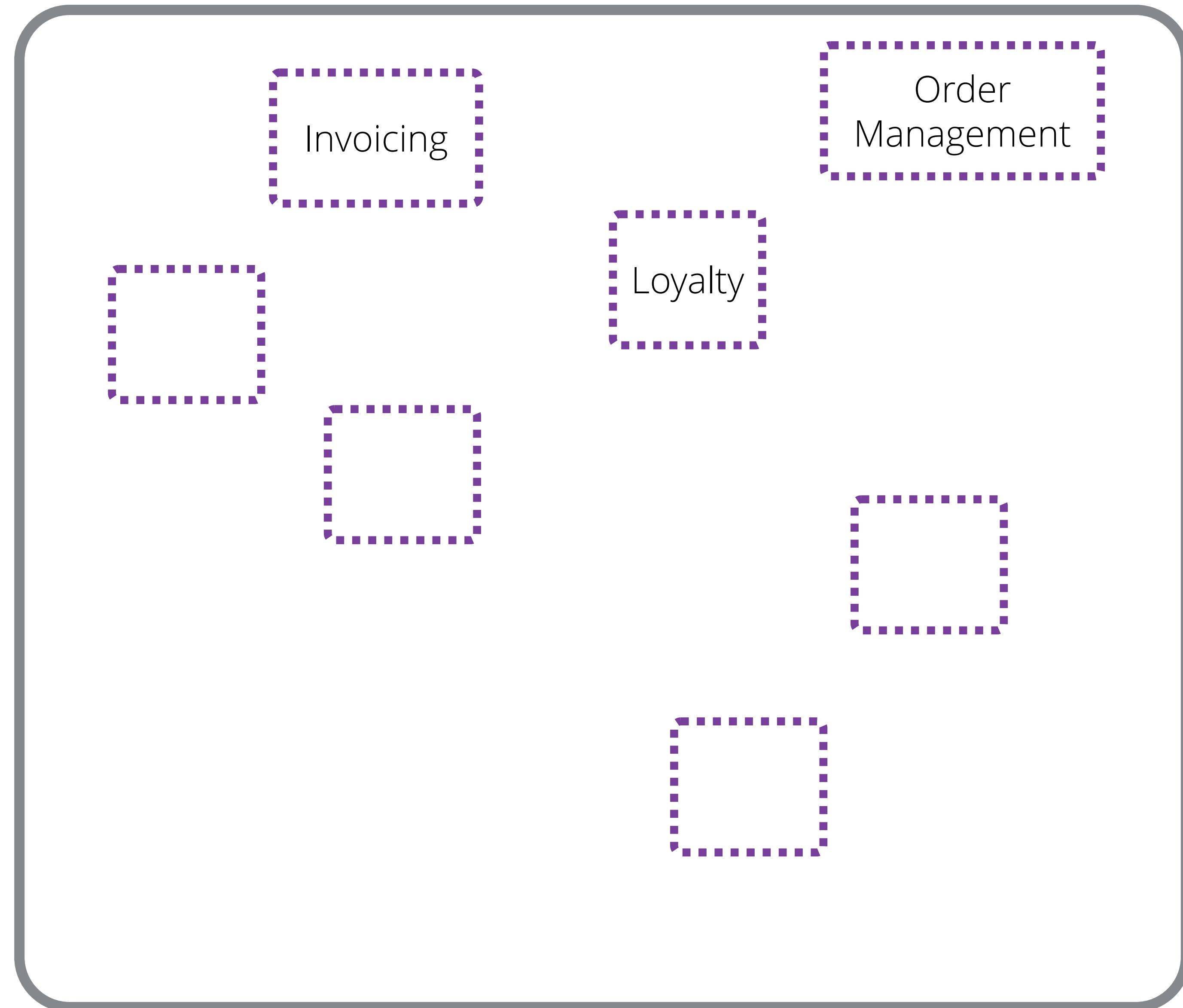


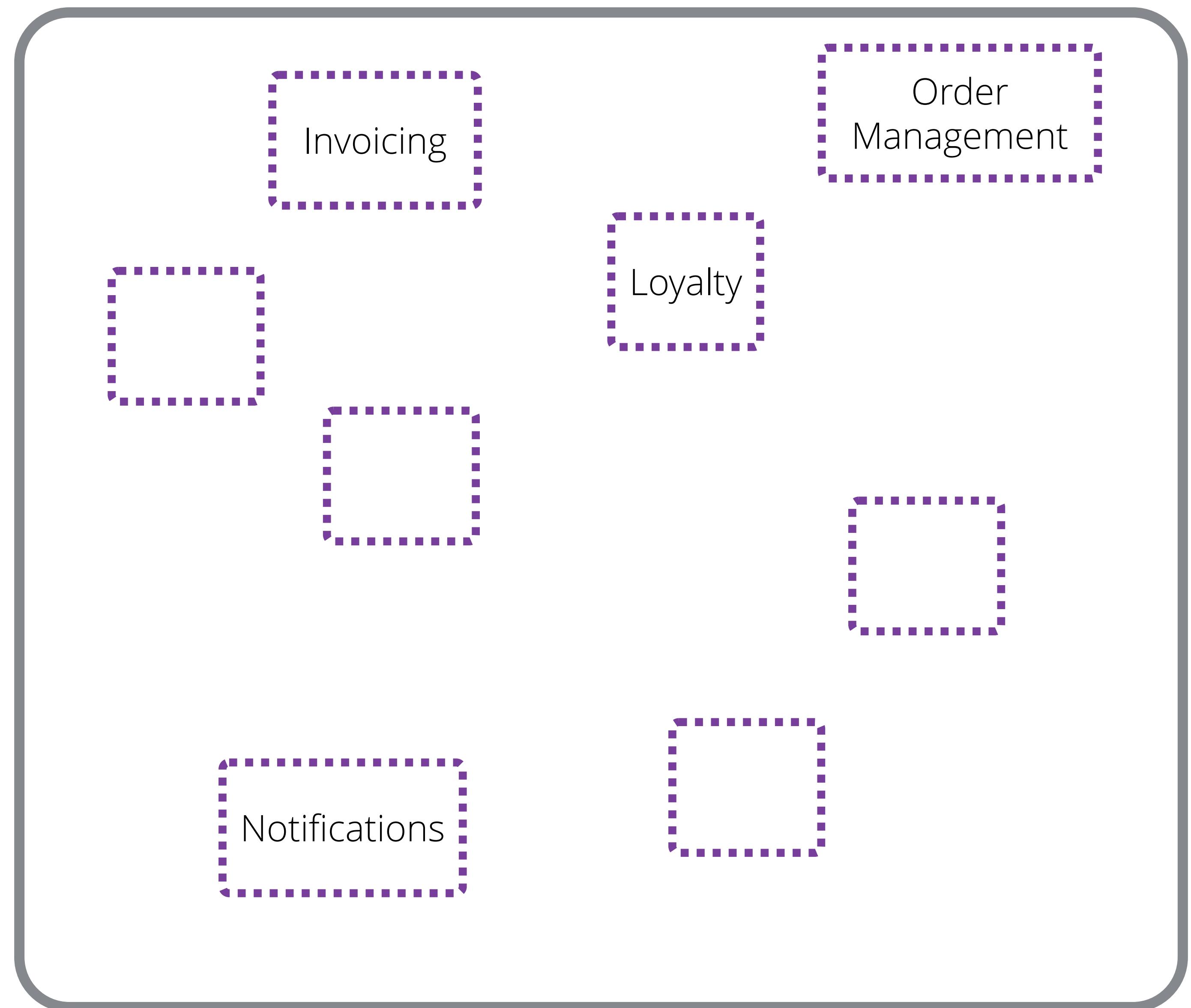


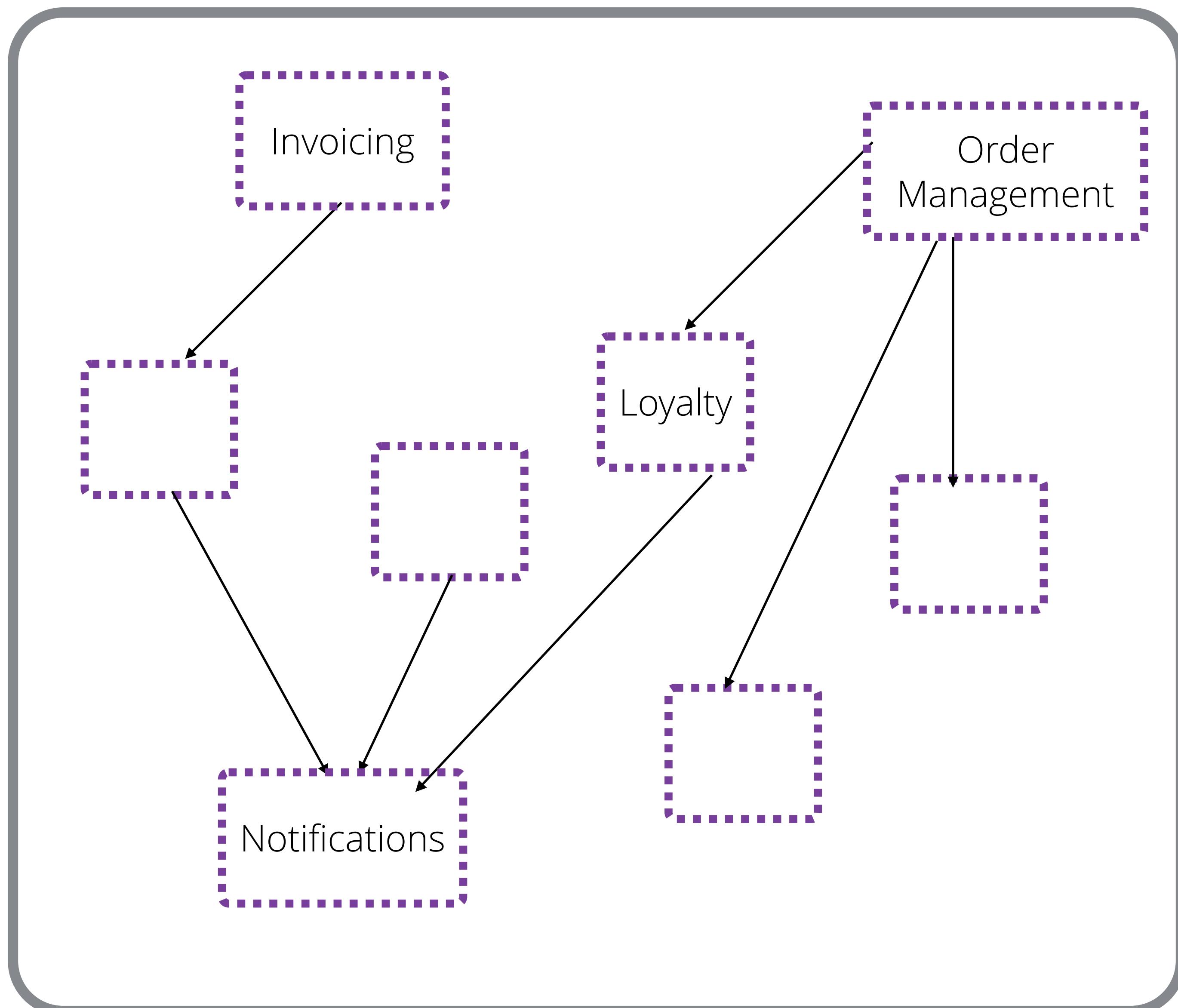


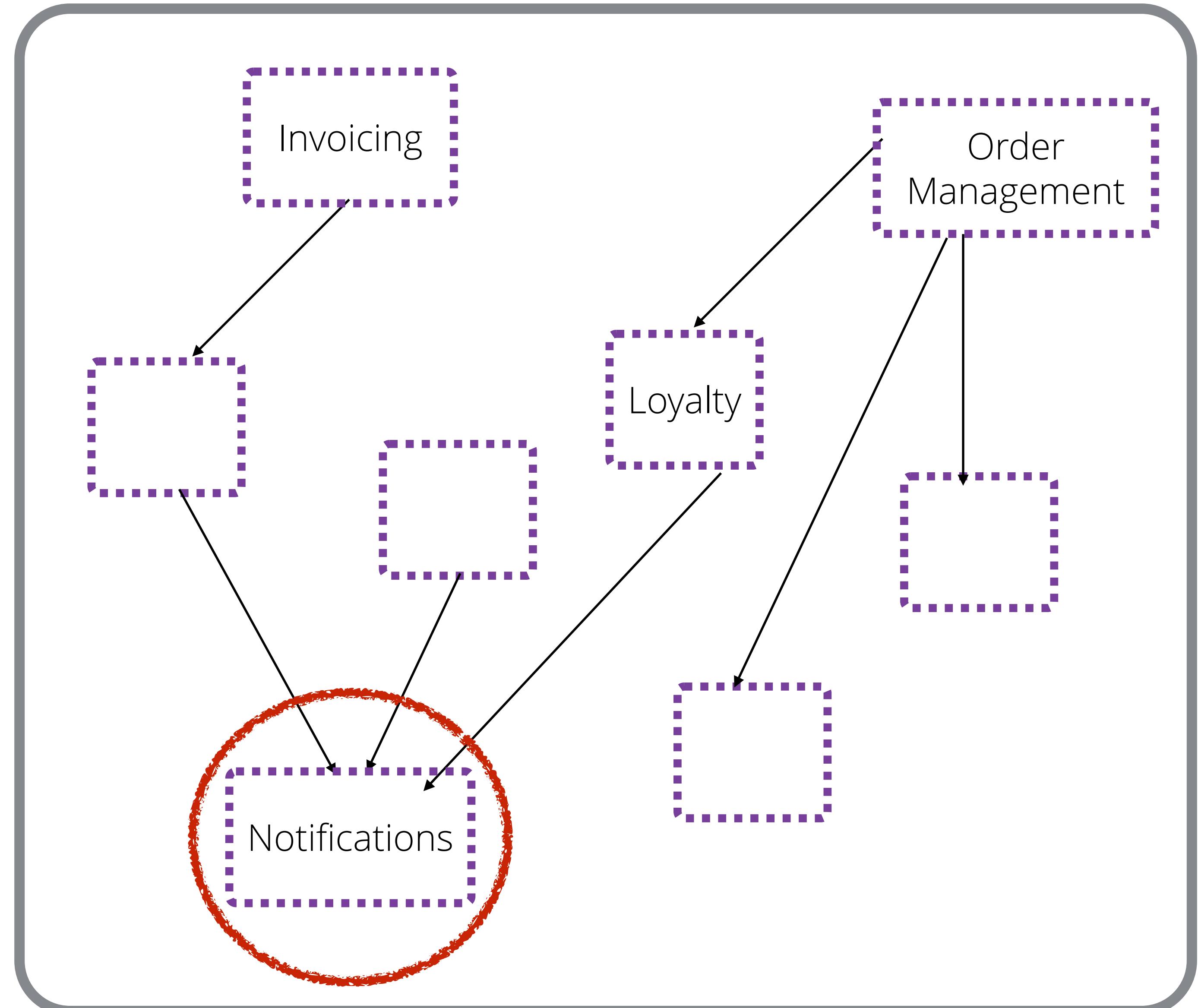


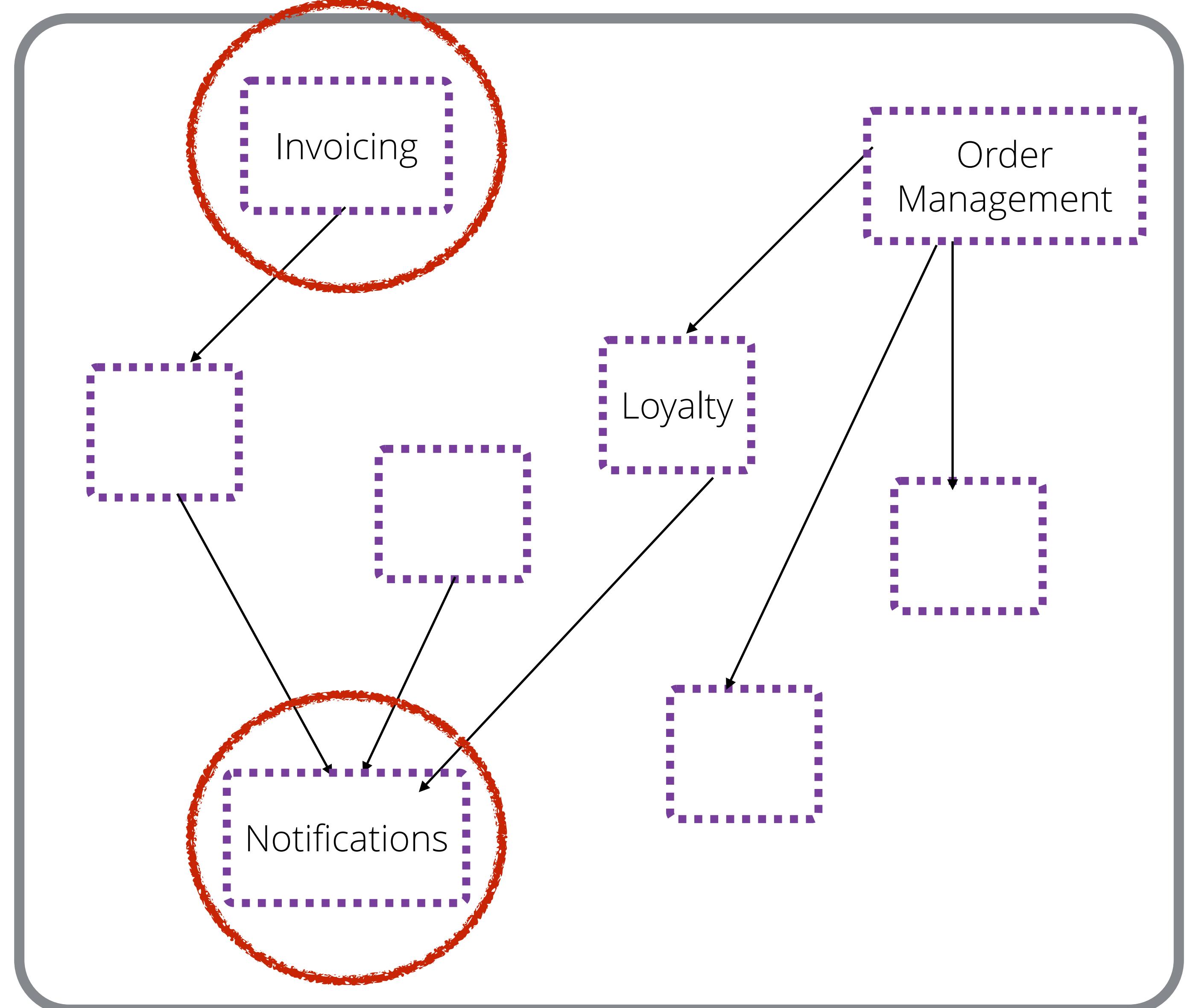












IT'S ALL ABOUT TRADEOFFS

Achieving your goal

IT'S ALL ABOUT TRADEOFFS



Achieving your goal

IT'S ALL ABOUT TRADEOFFS

Loyalty



Achieving your goal

IT'S ALL ABOUT TRADEOFFS



Achieving your goal

IT'S ALL ABOUT TRADEOFFS



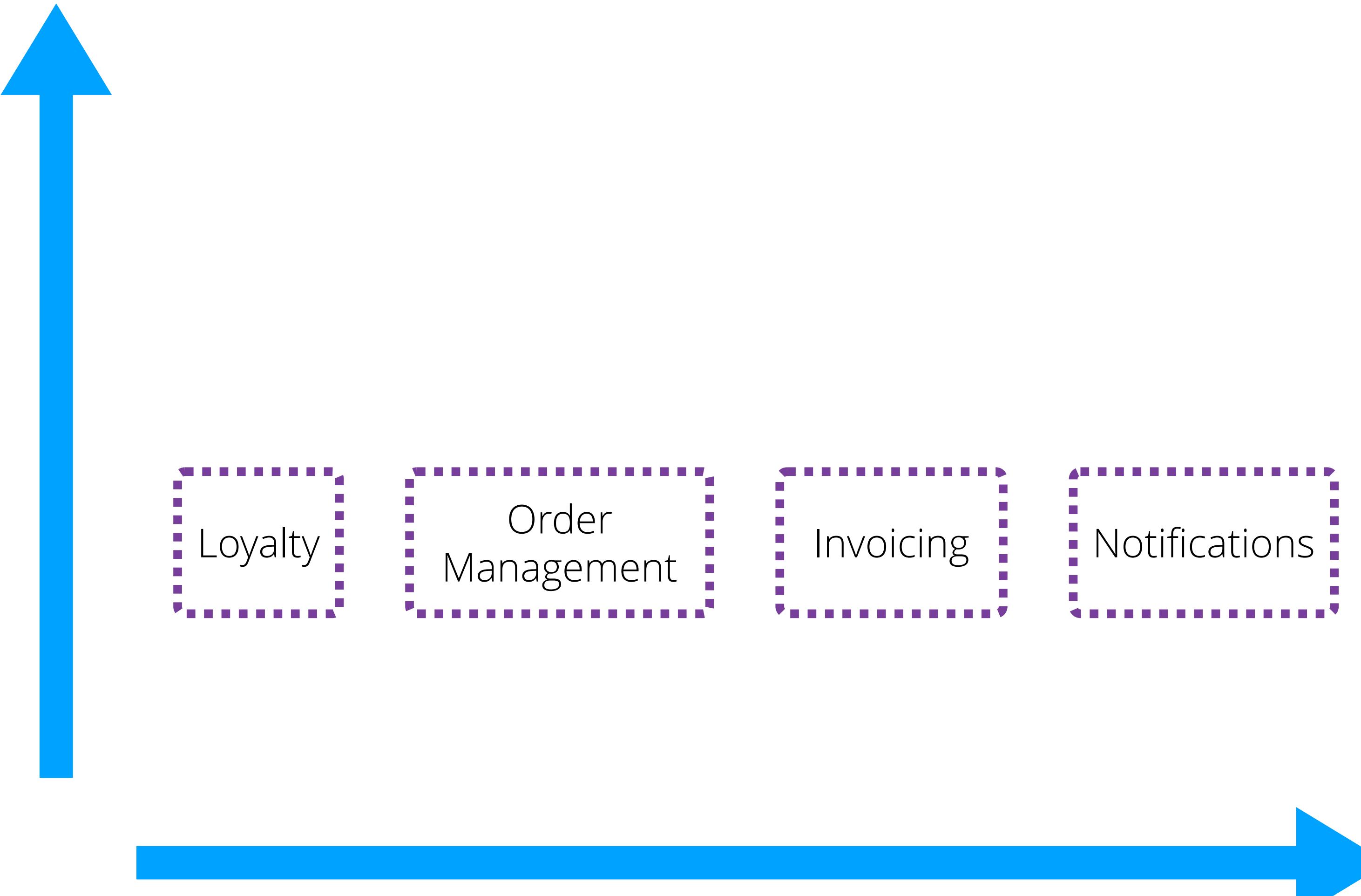
Achieving your goal

IT'S ALL ABOUT TRADEOFFS



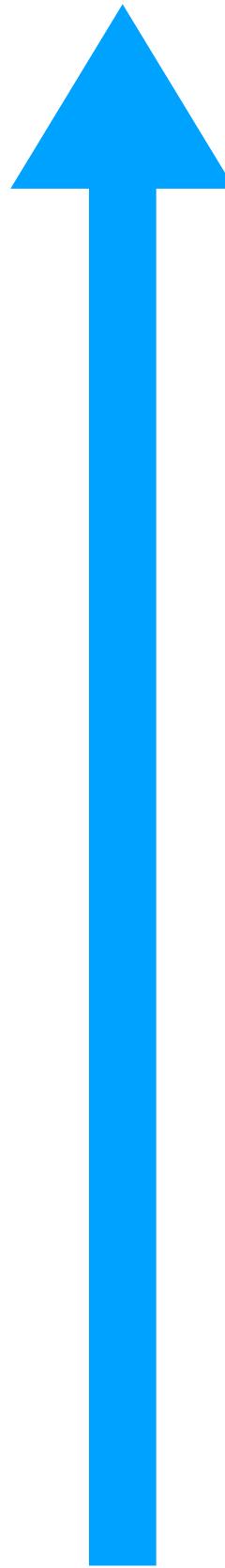
Achieving your goal

IT'S ALL ABOUT TRADEOFFS



IT'S ALL ABOUT TRADEOFFS

**Ease of
decomposition**



Loyalty

Order
Management

Invoicing

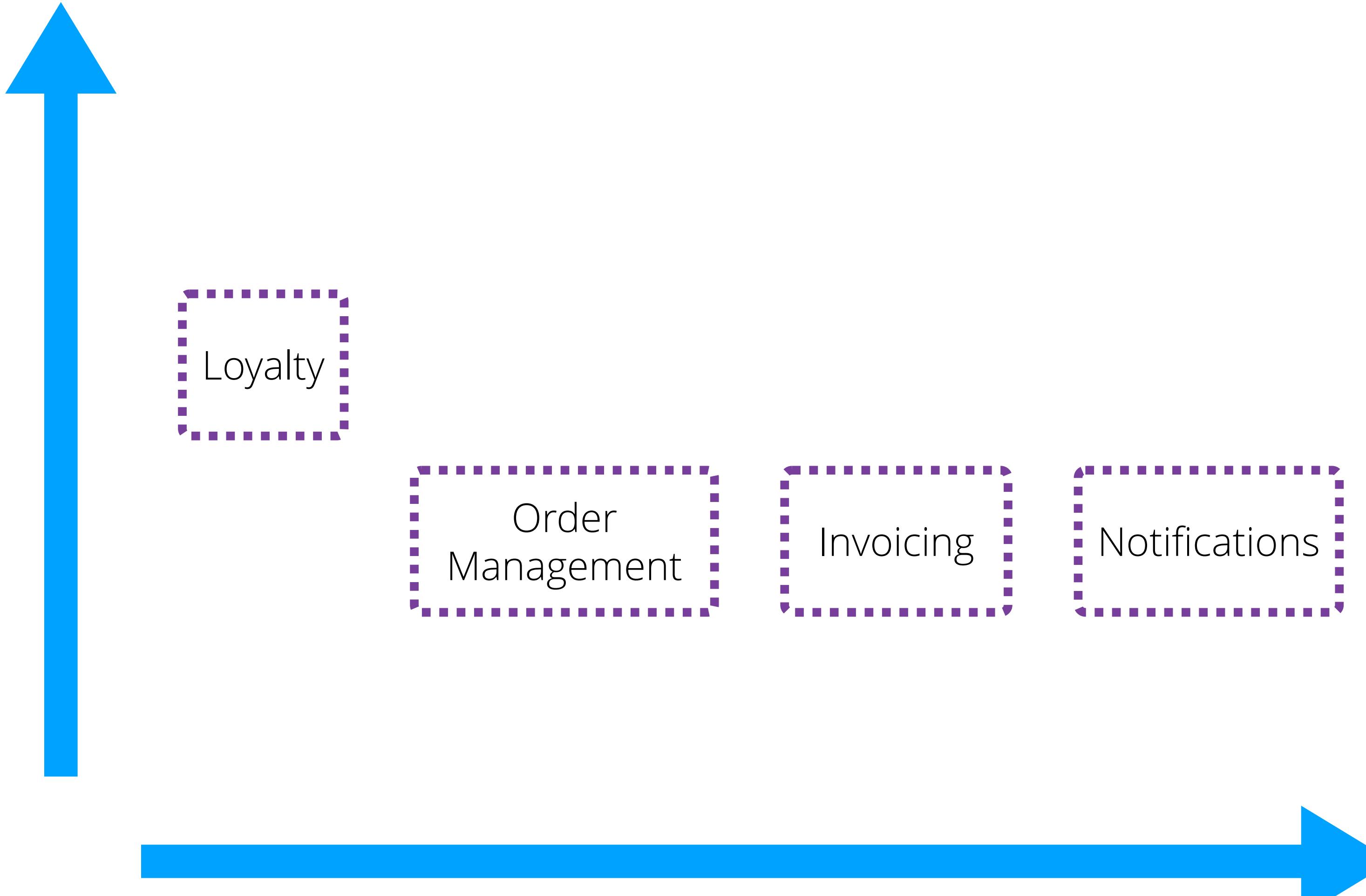
Notifications



Achieving your goal

IT'S ALL ABOUT TRADEOFFS

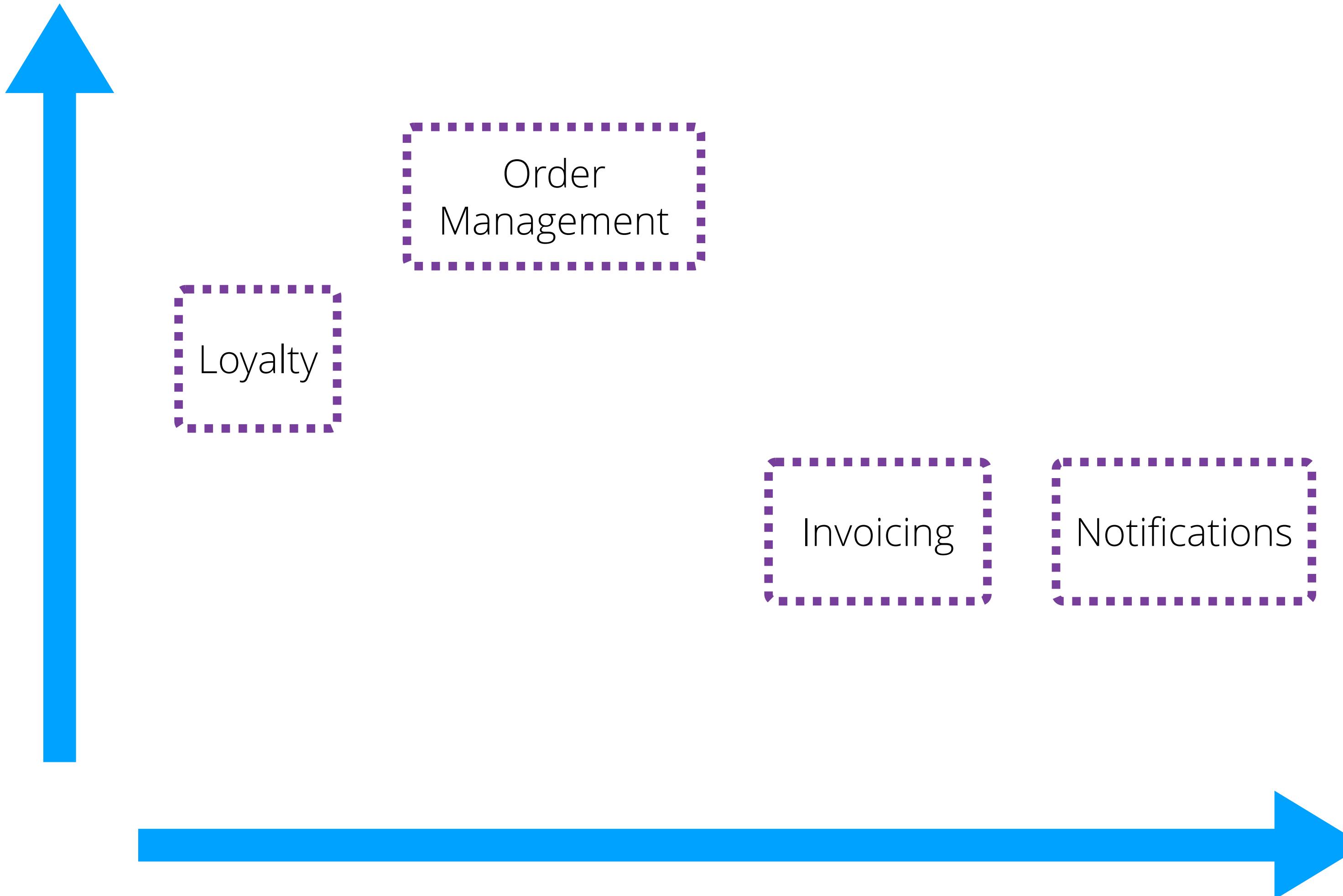
**Ease of
decomposition**



Achieving your goal

IT'S ALL ABOUT TRADEOFFS

**Ease of
decomposition**



Achieving your goal

IT'S ALL ABOUT TRADEOFFS

**Ease of
decomposition**



Order
Management

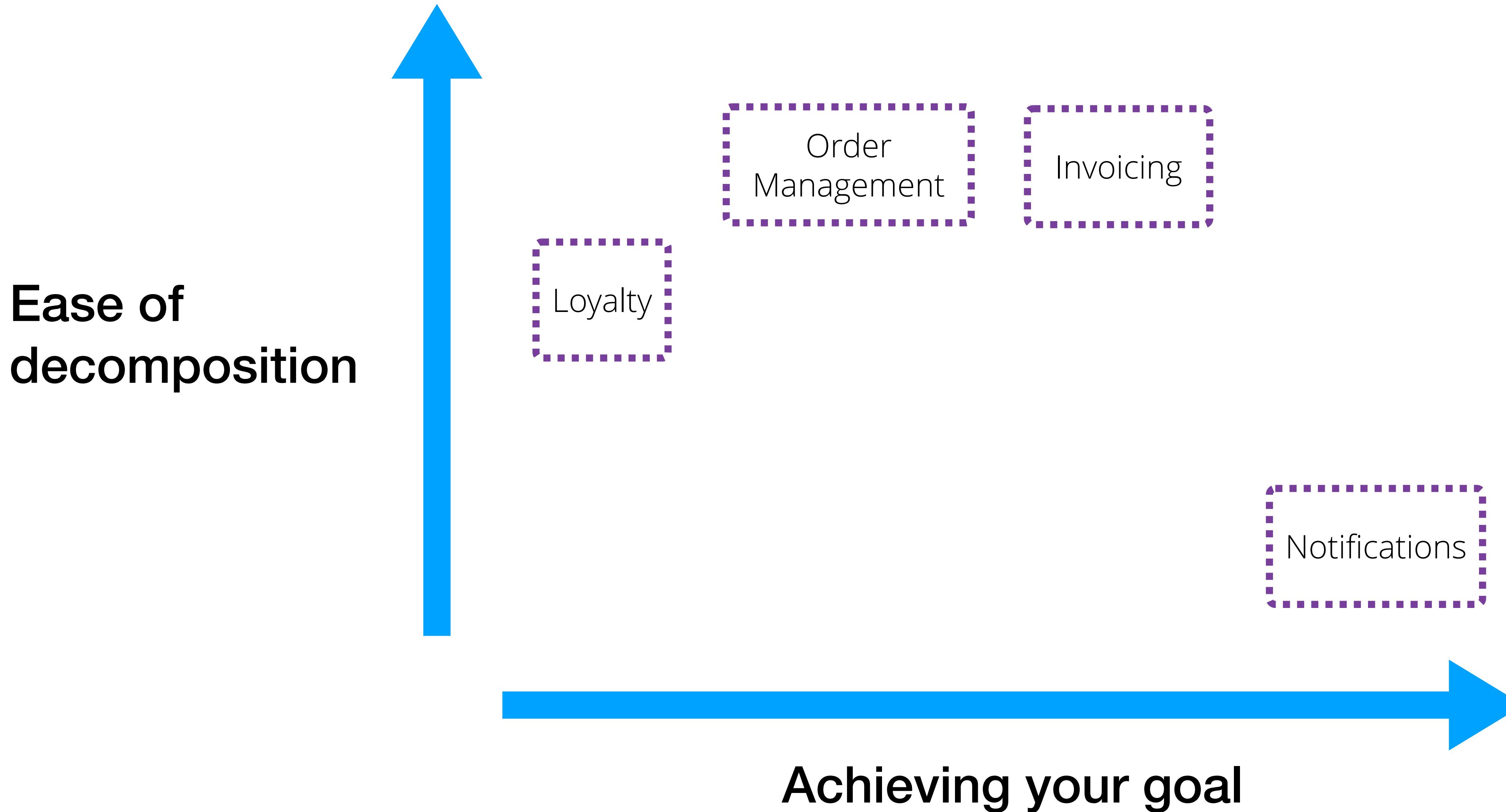
Invoicing

Notifications

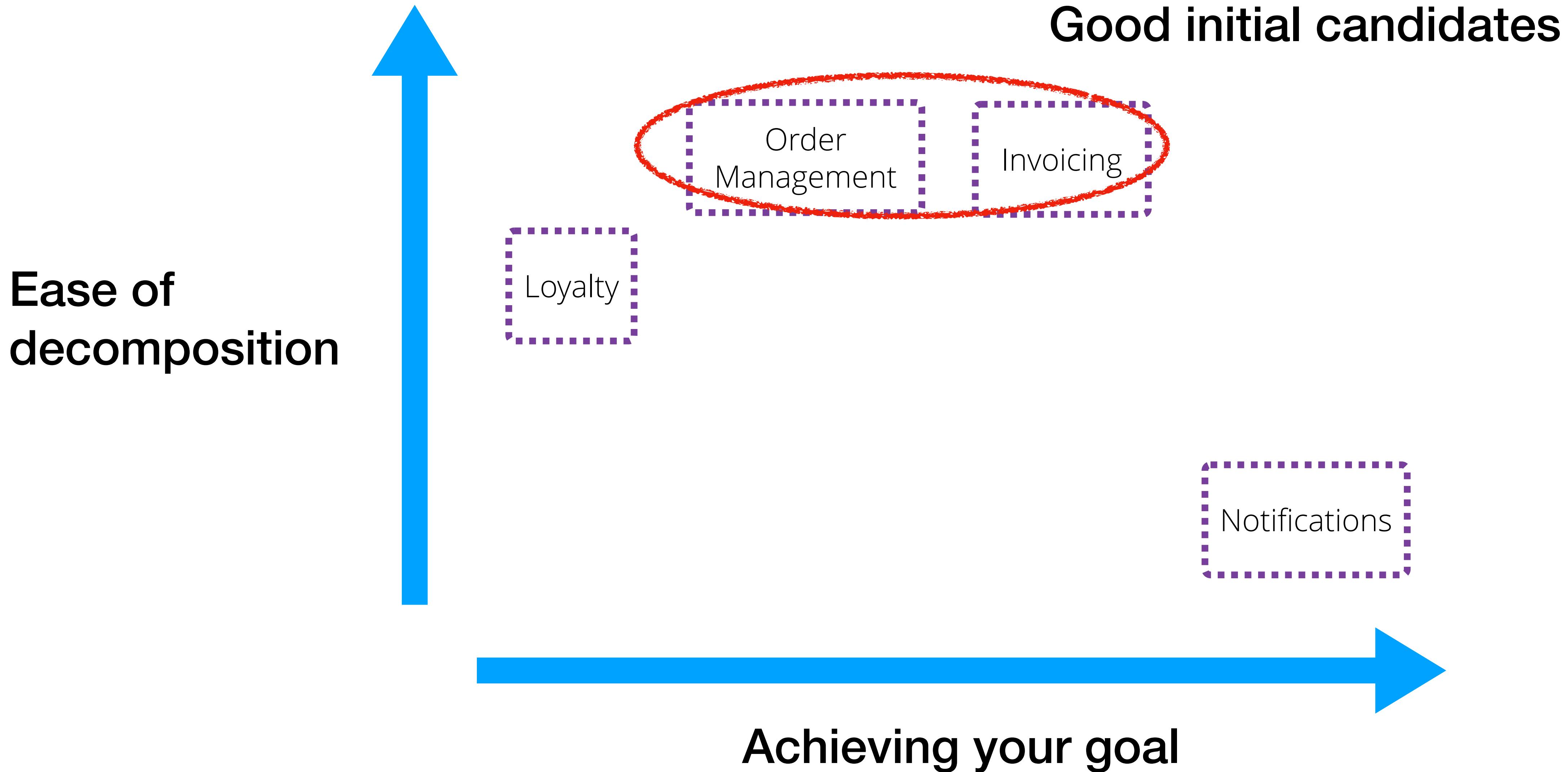


Achieving your goal

IT'S ALL ABOUT TRADEOFFS



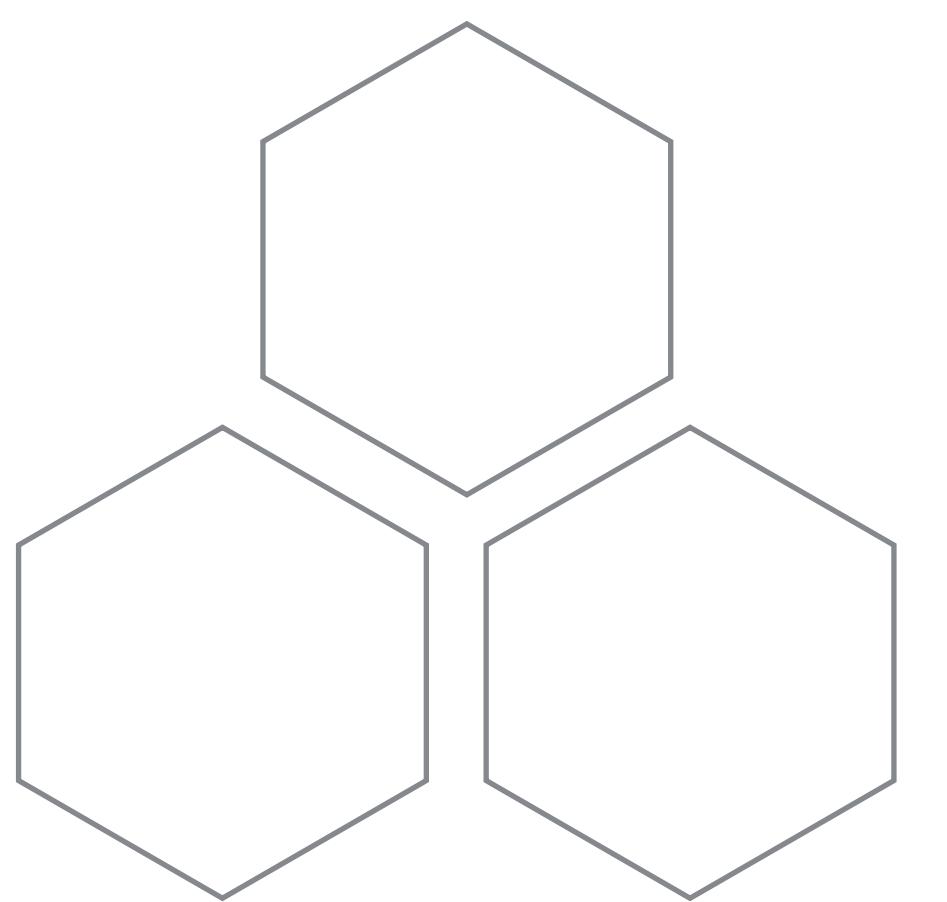
IT'S ALL ABOUT TRADEOFFS

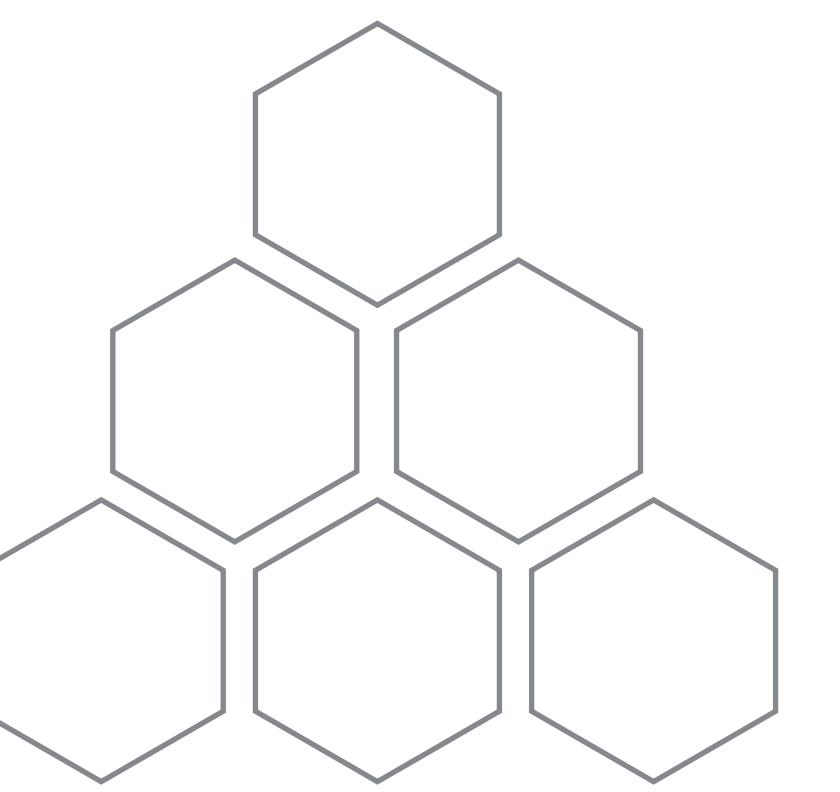
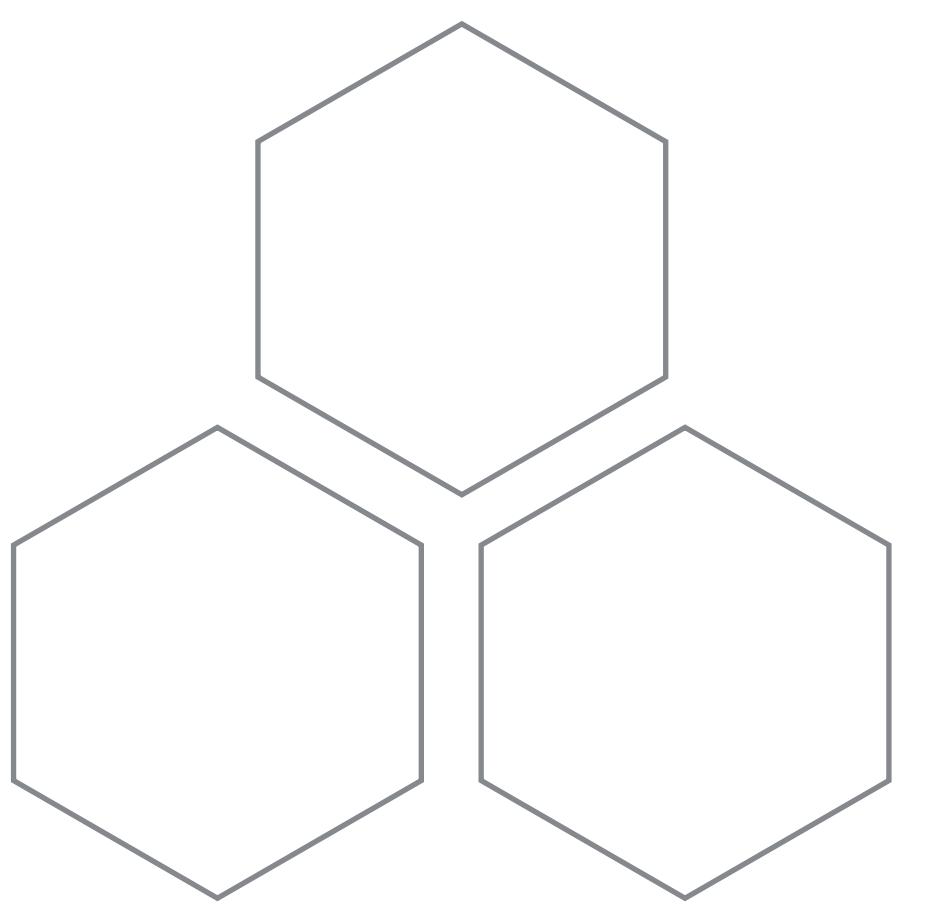


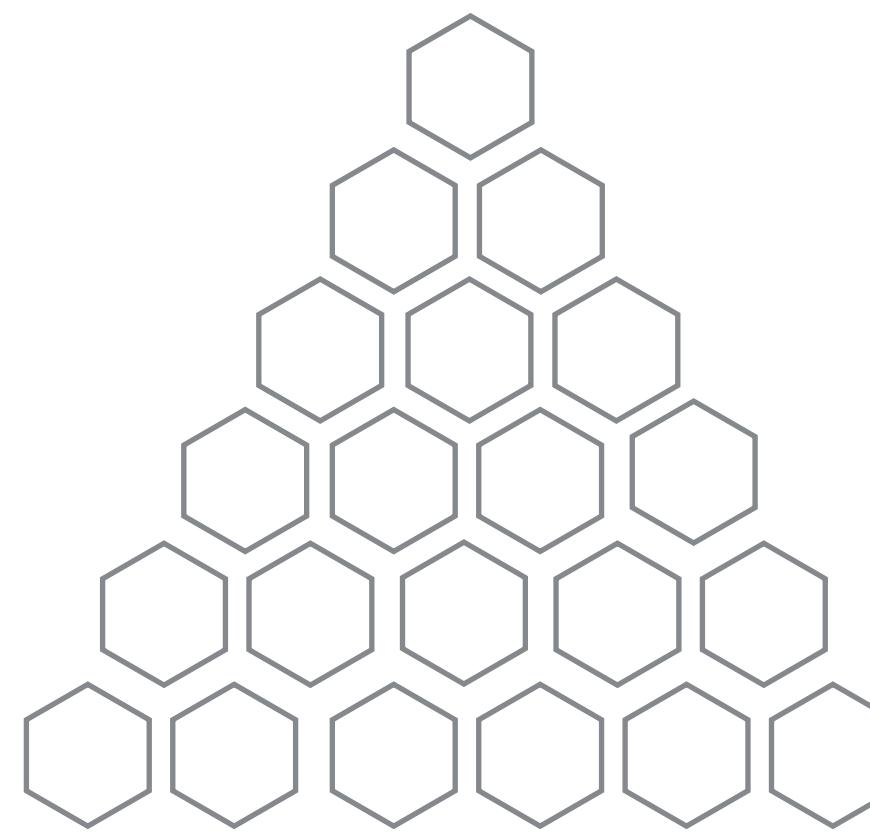
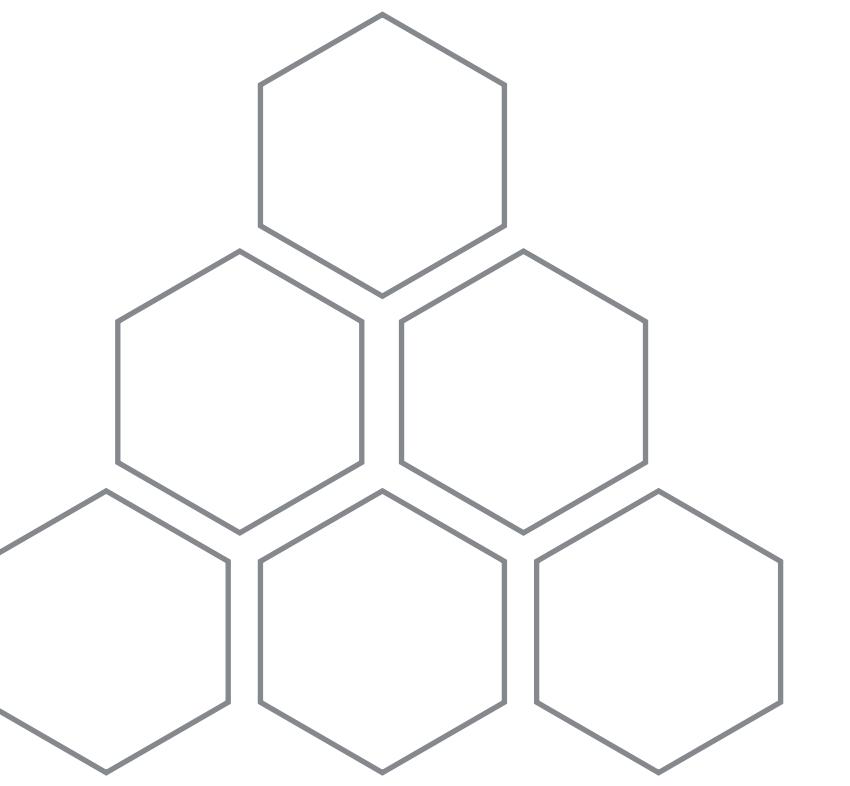
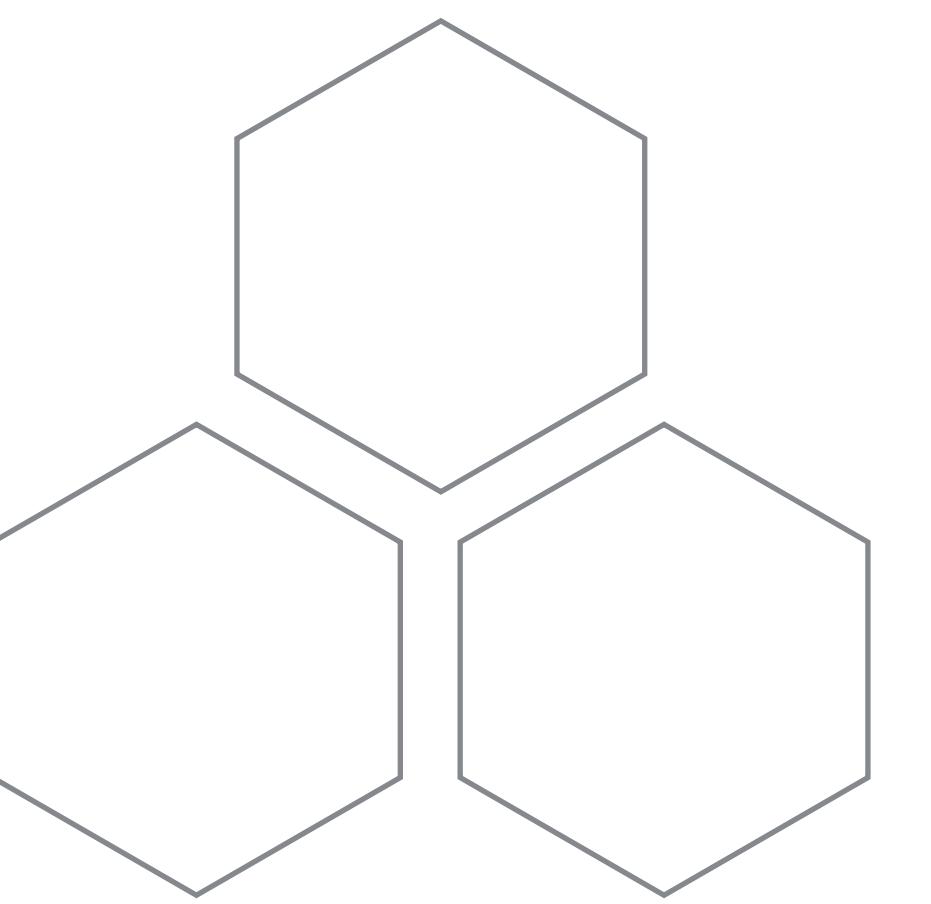
LOUDNESS

10

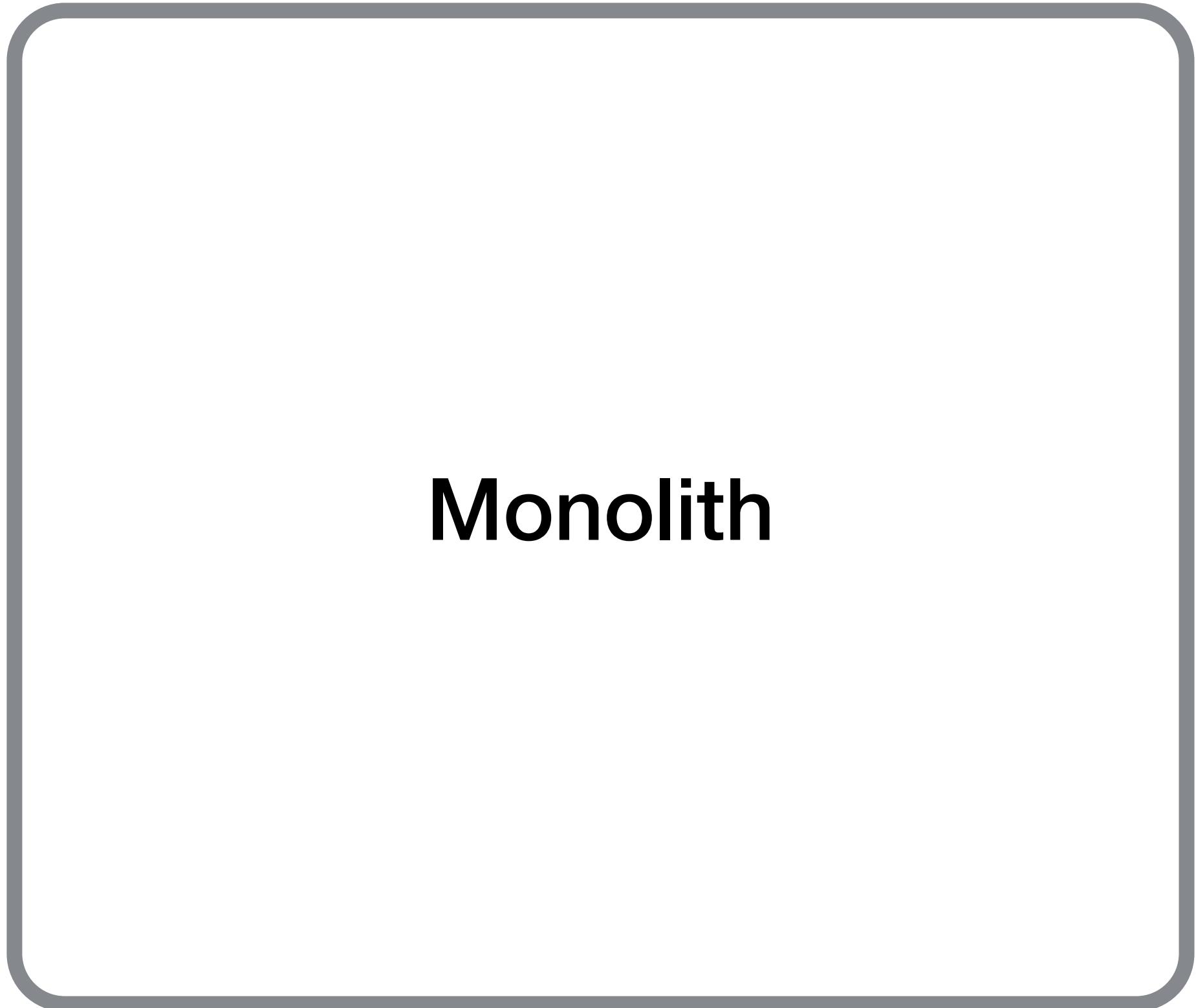






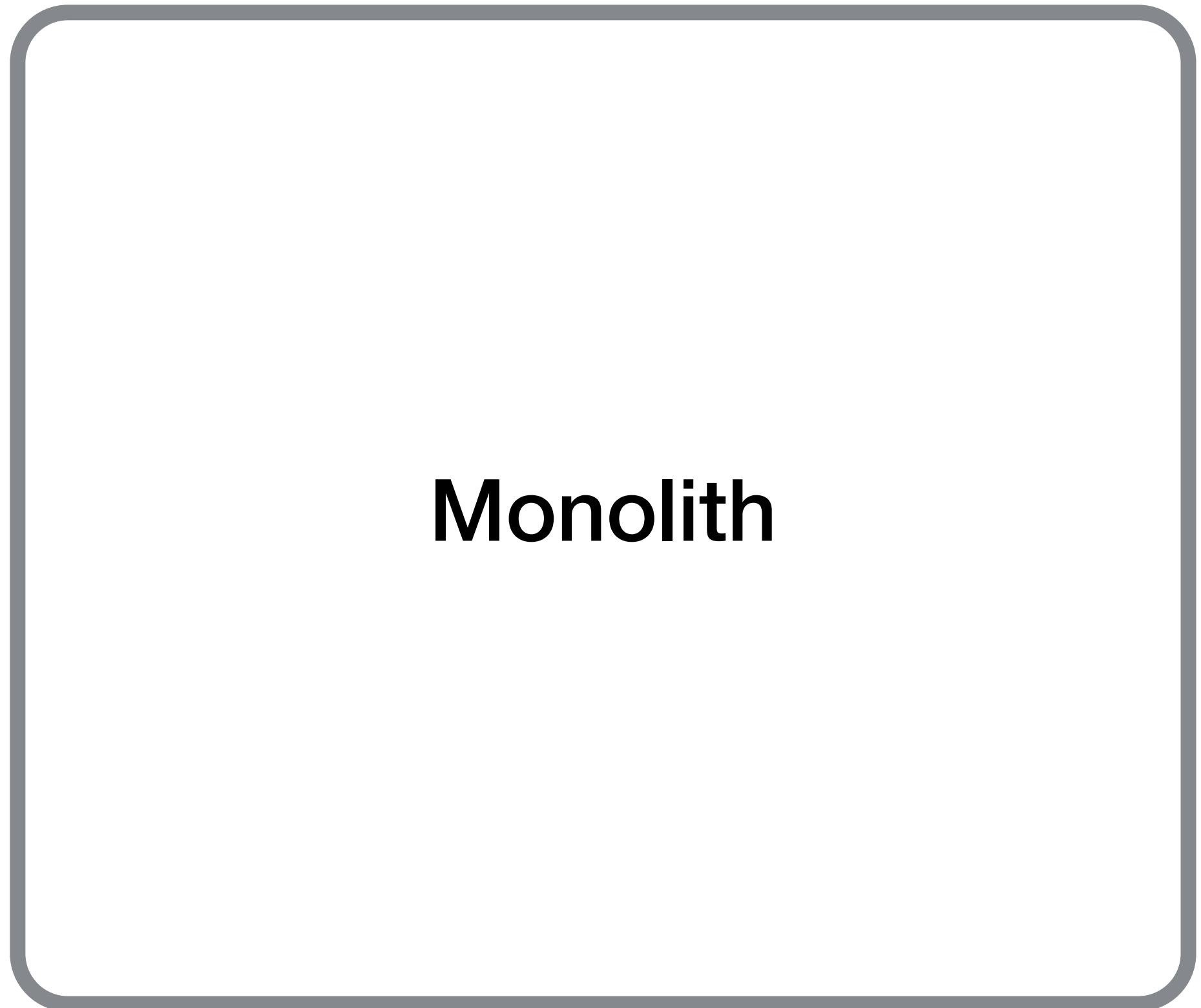


INCREMENTAL DECOMPOSITION FTW

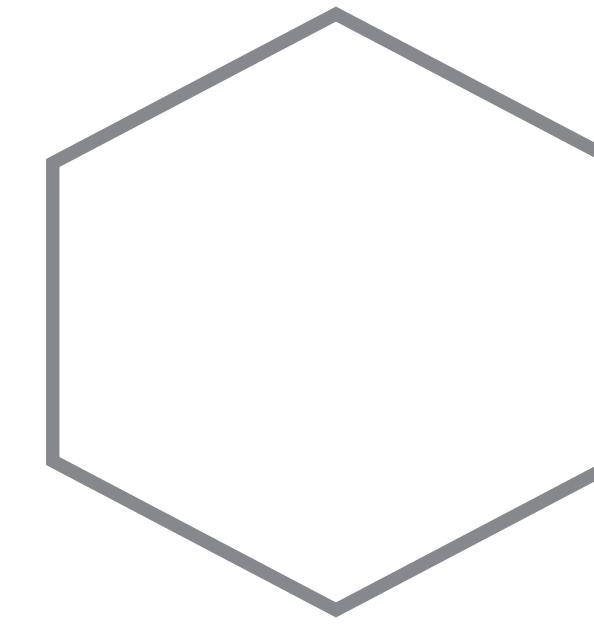


Monolith

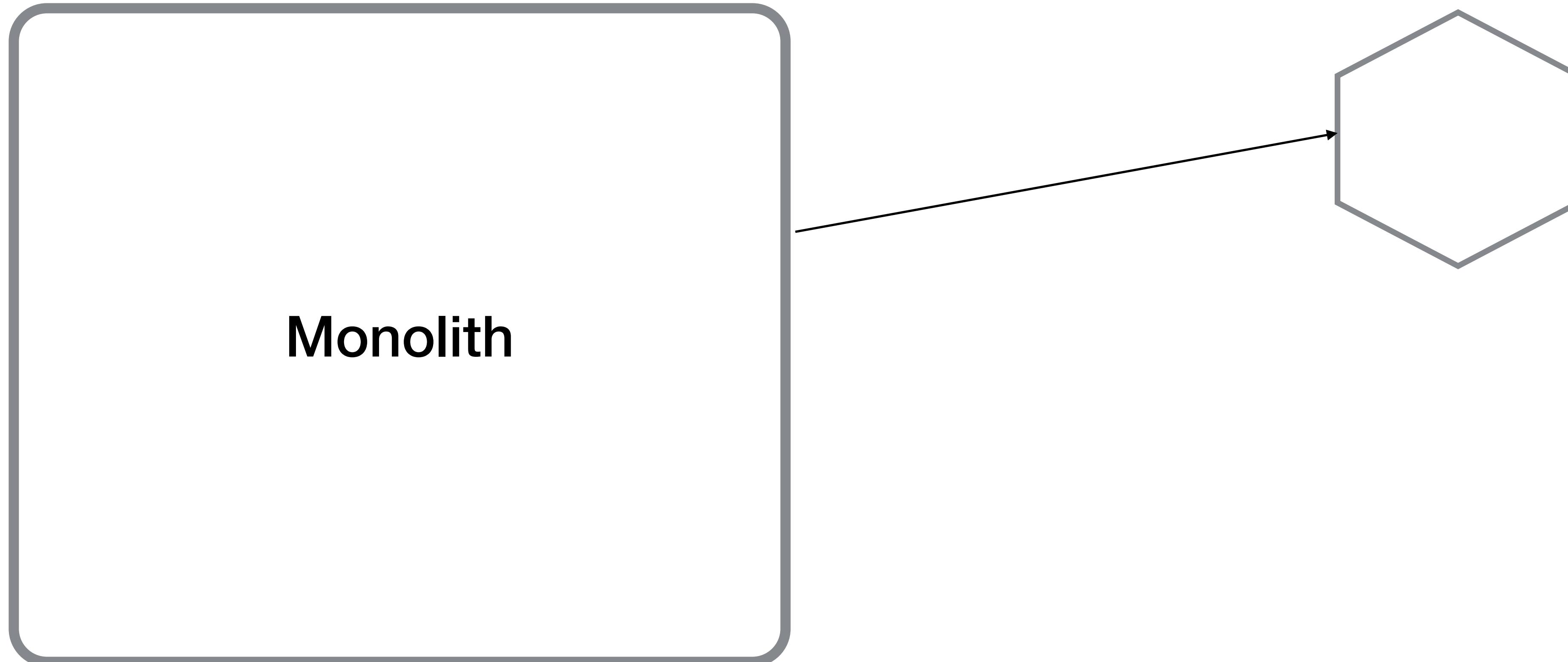
INCREMENTAL DECOMPOSITION FTW



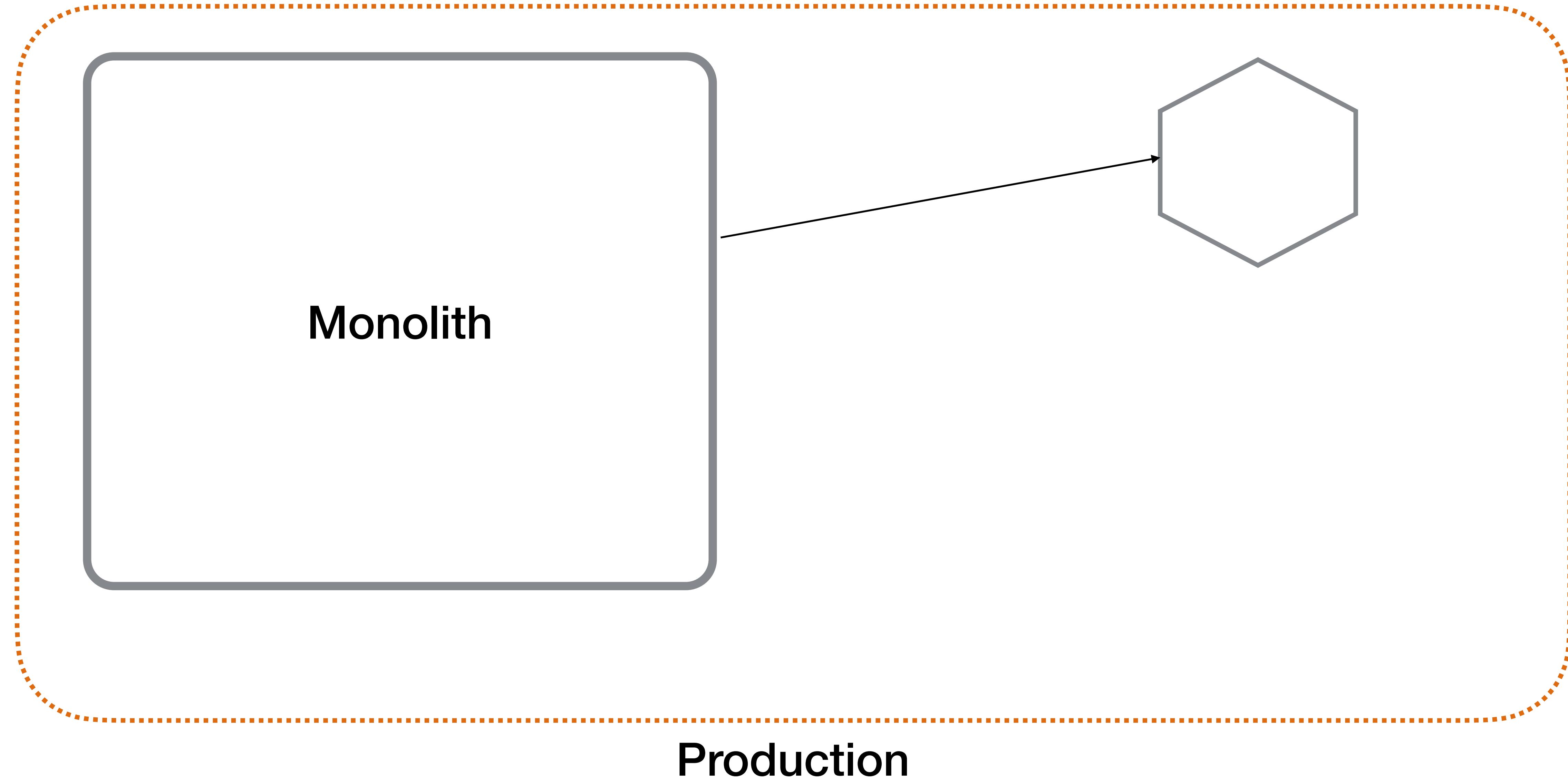
Monolith



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

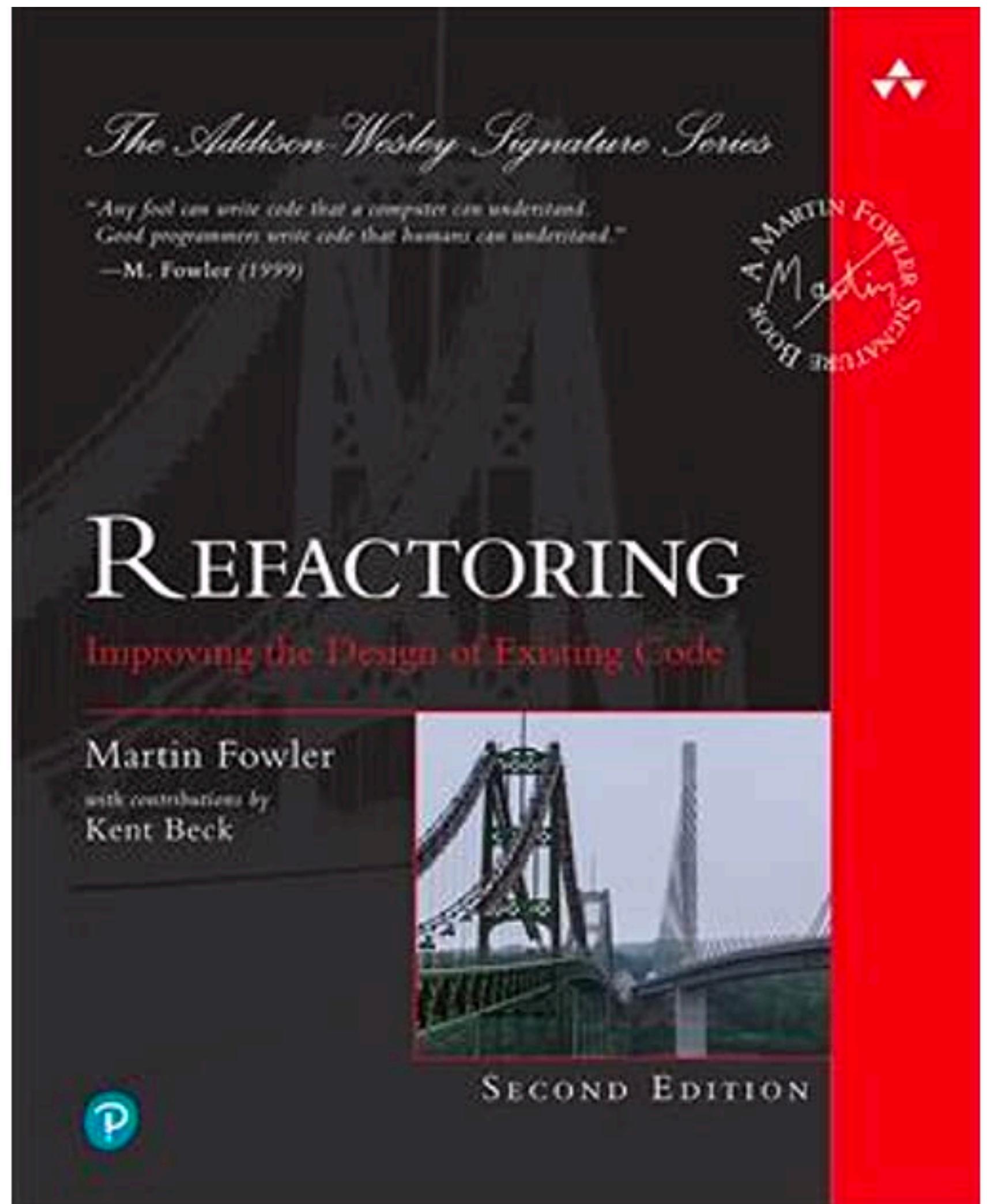
Introduction

Migration Approach

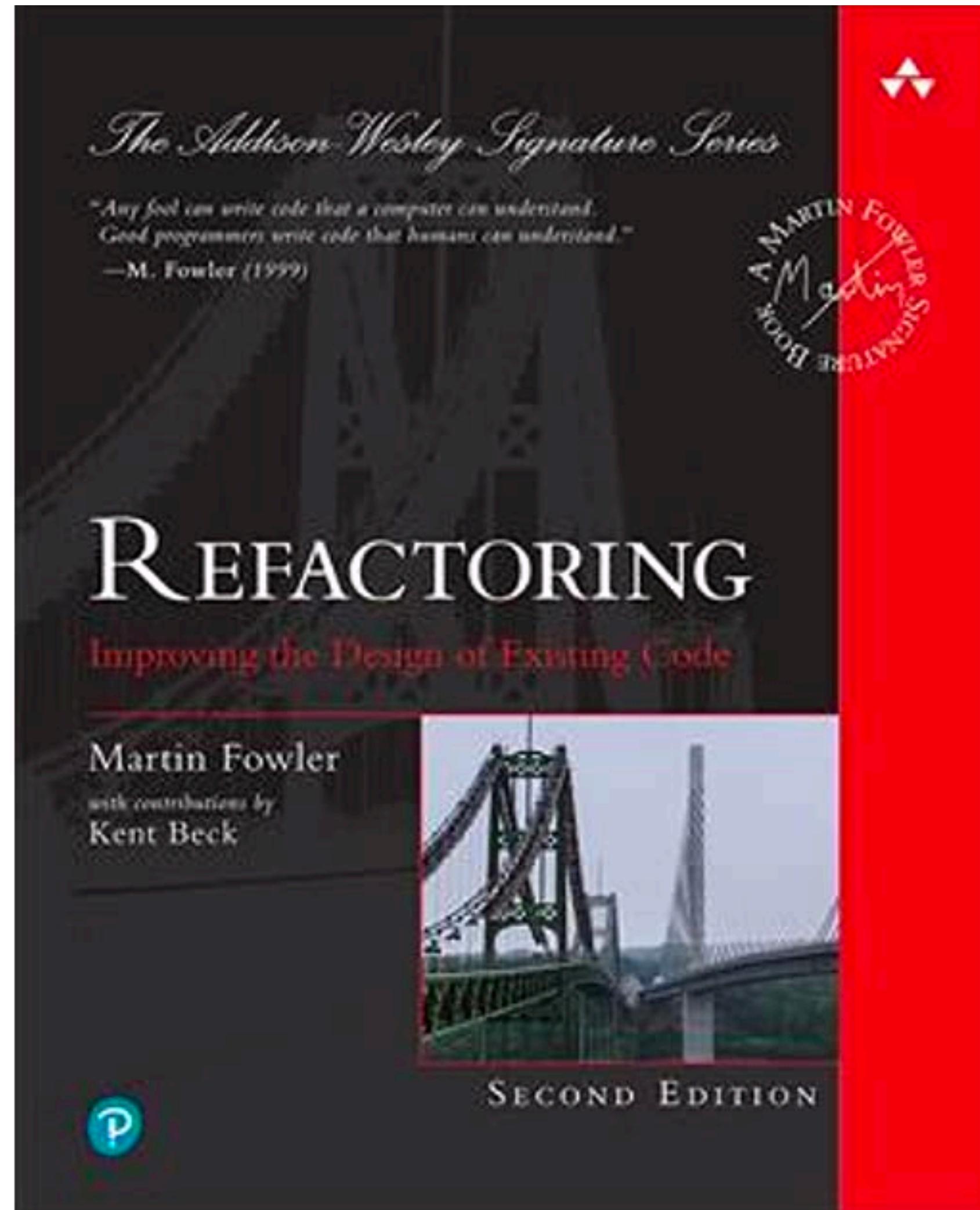
Application Refactoring

UI Decomposition

ON REFACTORING

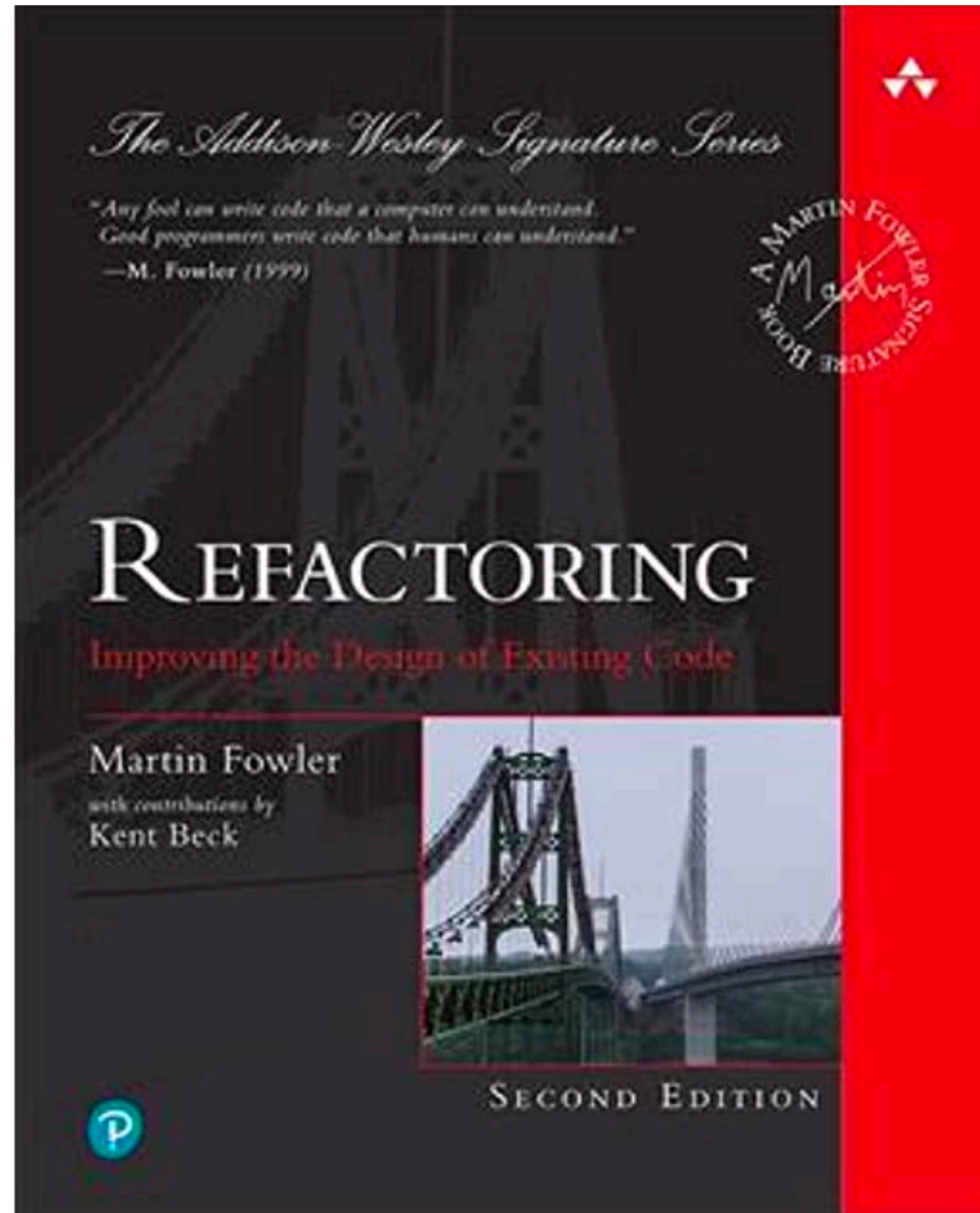


ON REFACTORING



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



Wrap a new system around the old system

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

PATTERN: STRANGLER FIG



Wrap a new system around the old system

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

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

PATTERN: STRANGLER FIG



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

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

STRANGLER FIG IMPLEMENTATION

Intercept calls on perimeter of old system (the monolith)

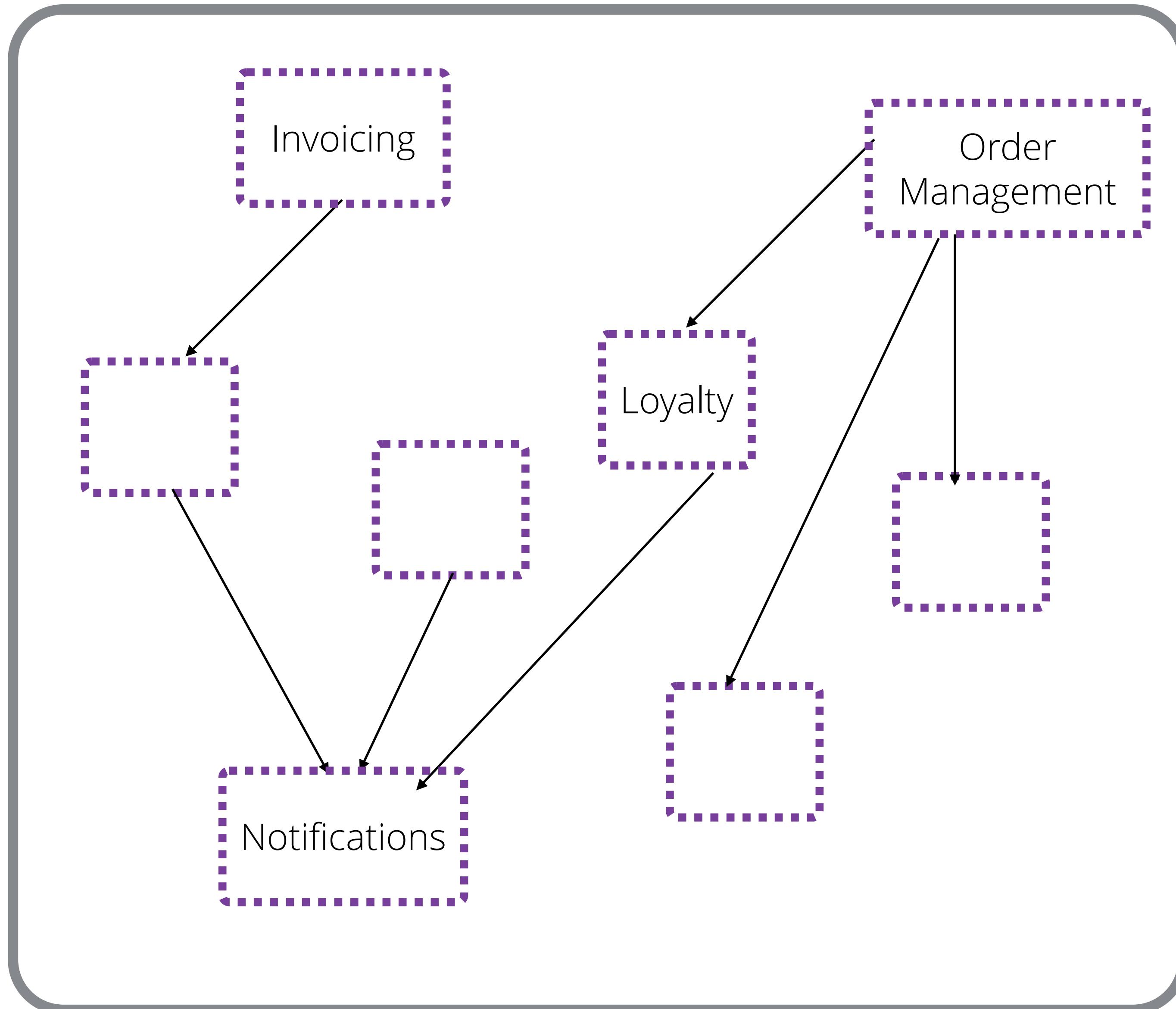
STRANGLER FIG IMPLEMENTATION

Intercept calls on perimeter of old system (the monolith)

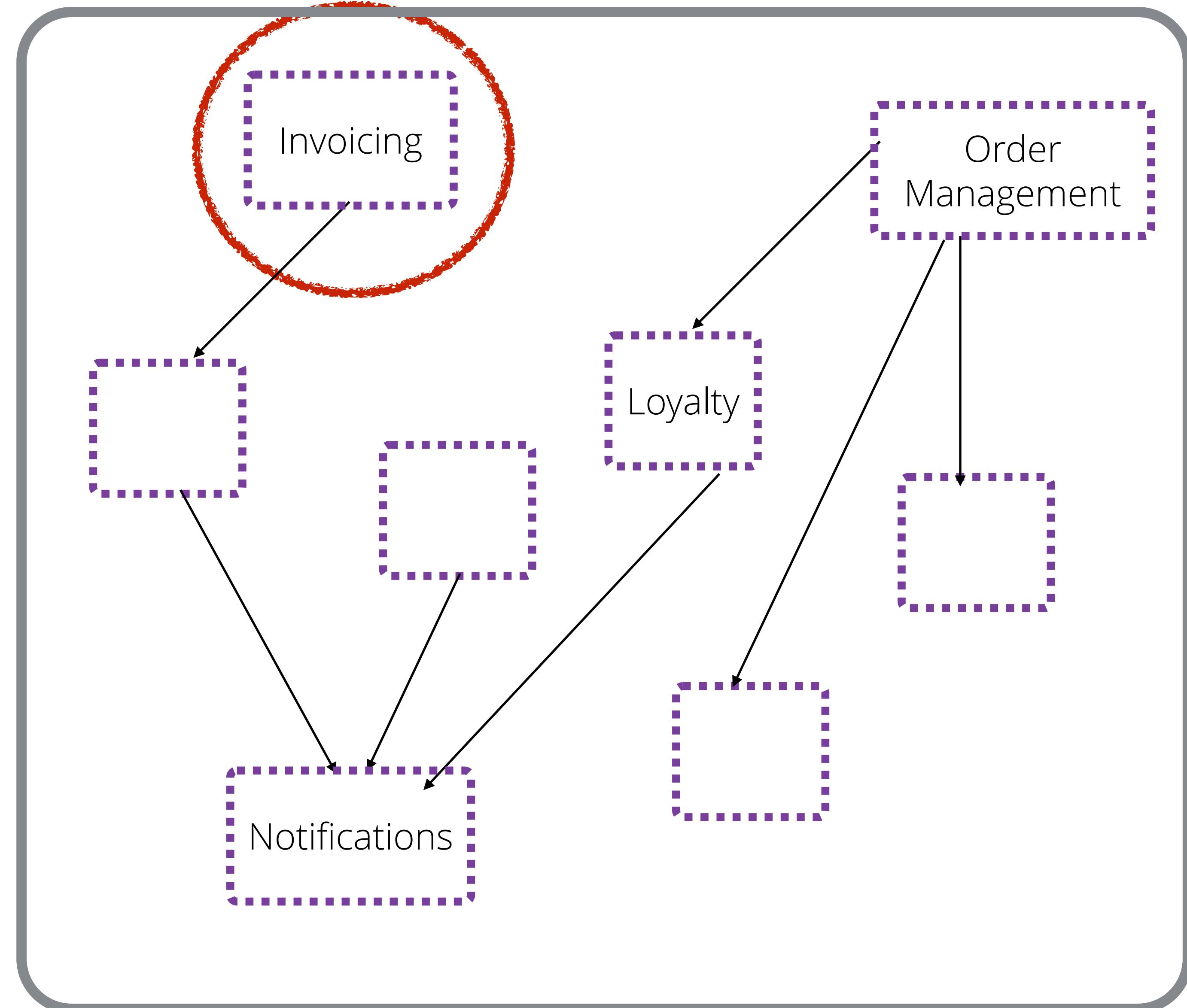
Divert calls to where the functionality is located

CANDIDATES

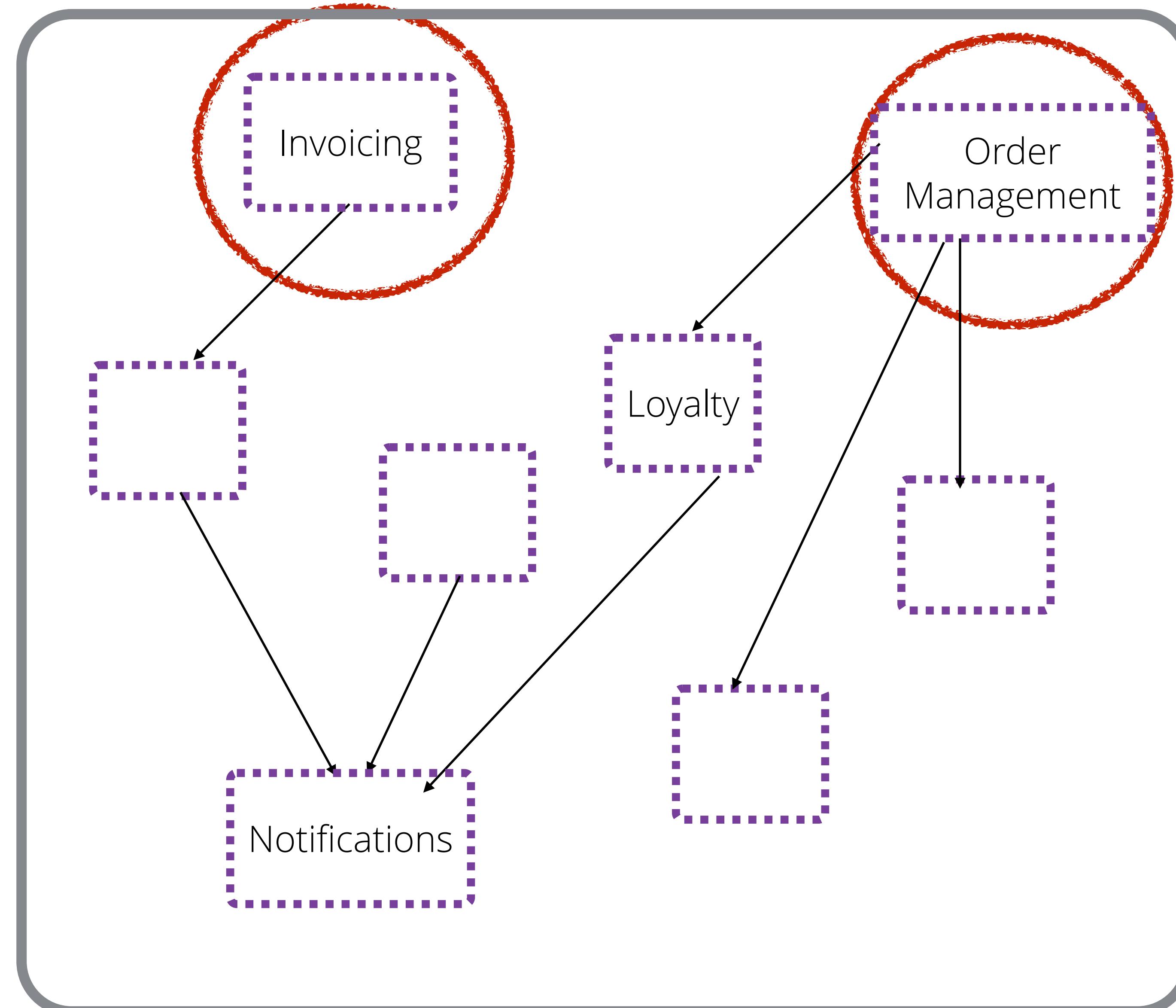
CANDIDATES



CANDIDATES



CANDIDATES

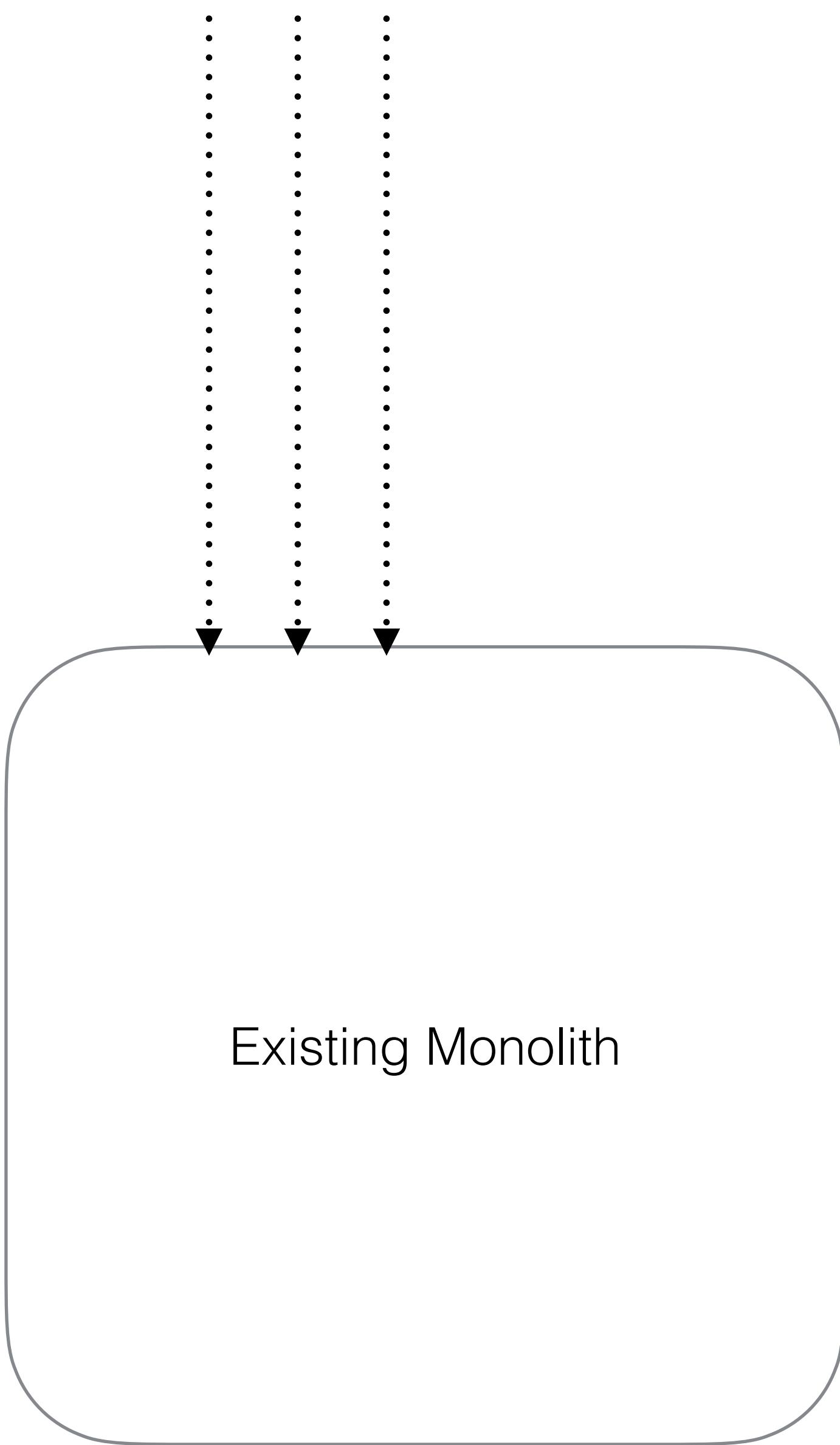


INTERCEPTION



Existing Monolith

INTERCEPTION

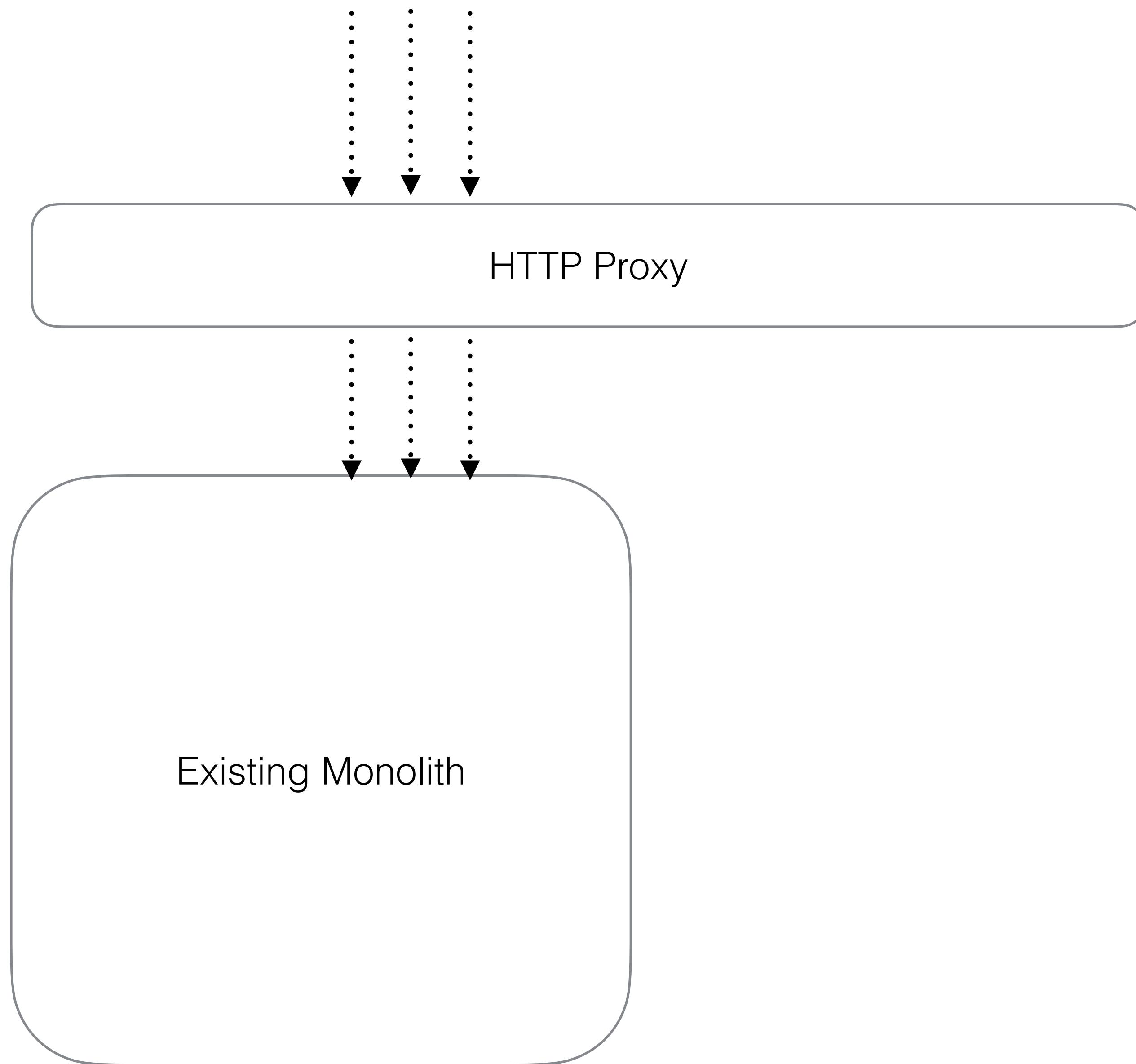


Existing Monolith

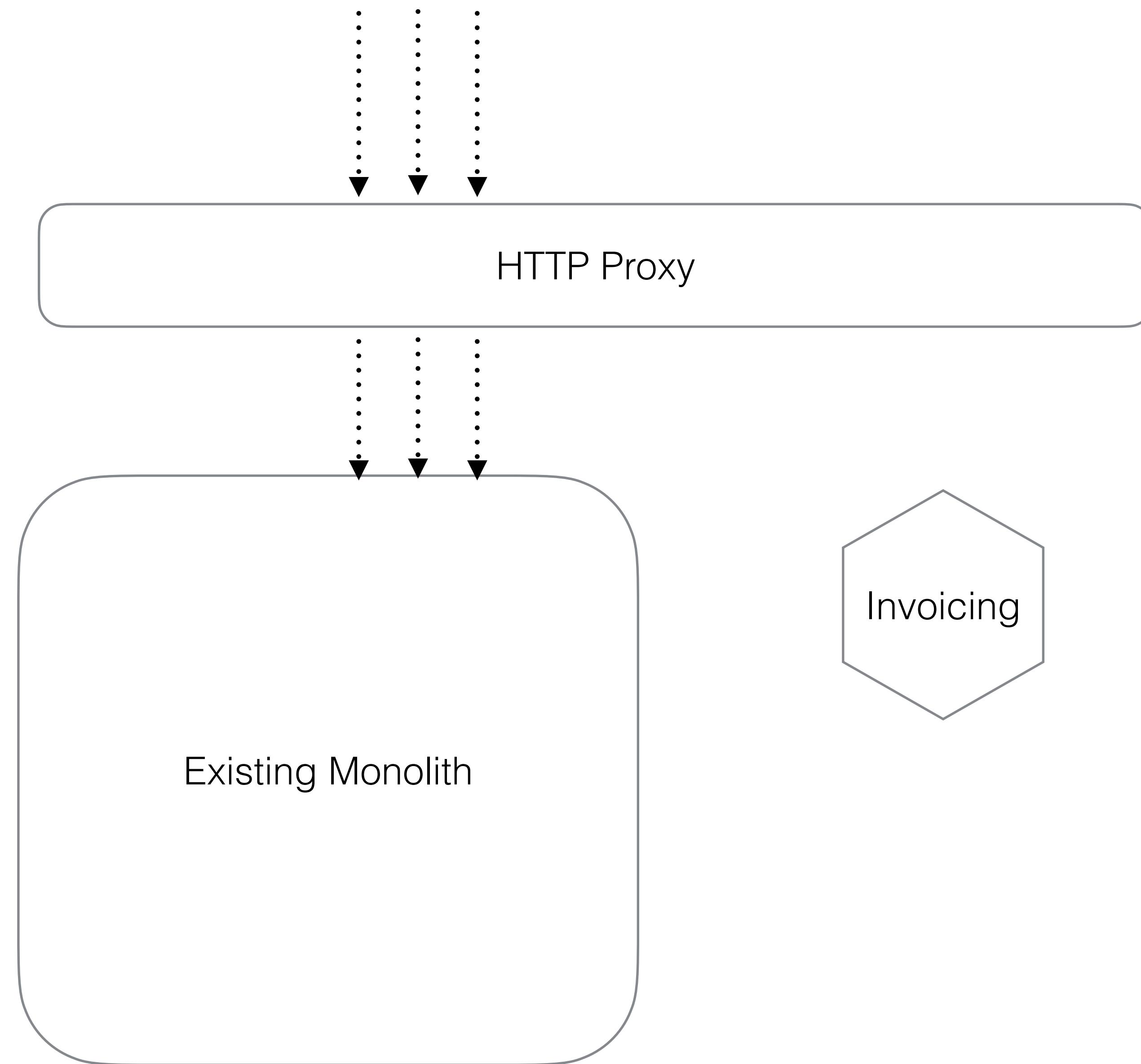
INTERCEPTION



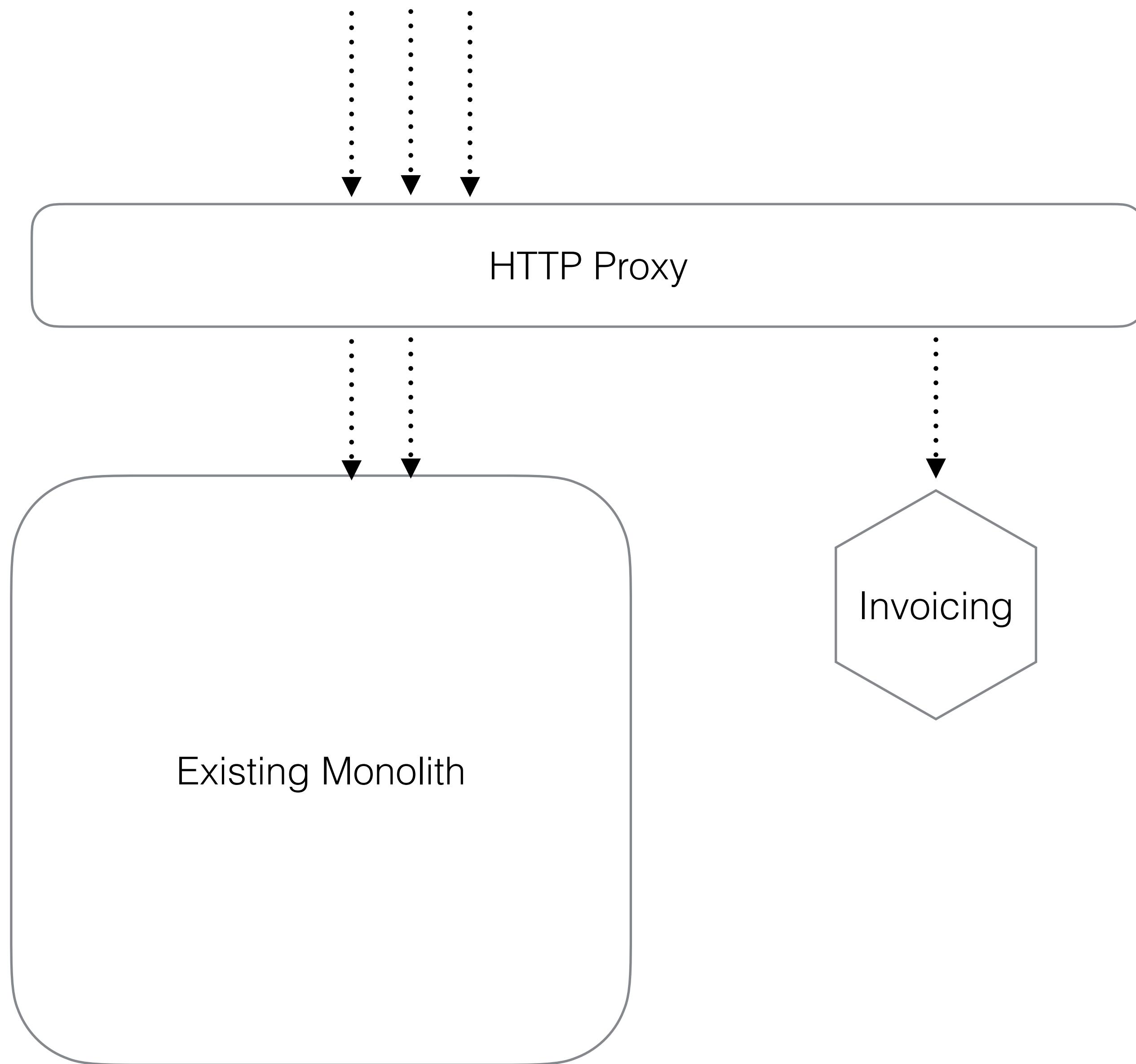
INTERCEPTION



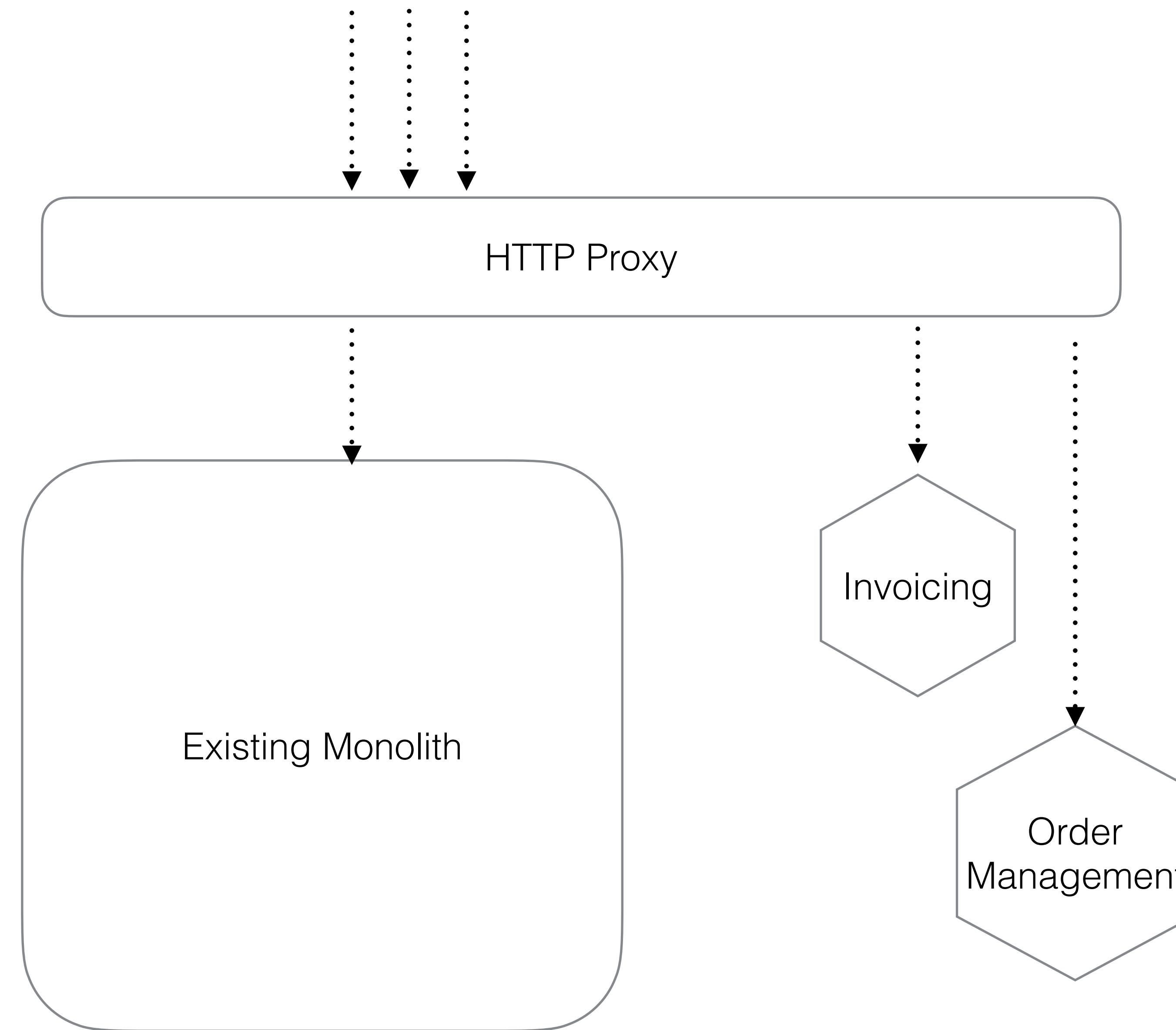
INTERCEPTION



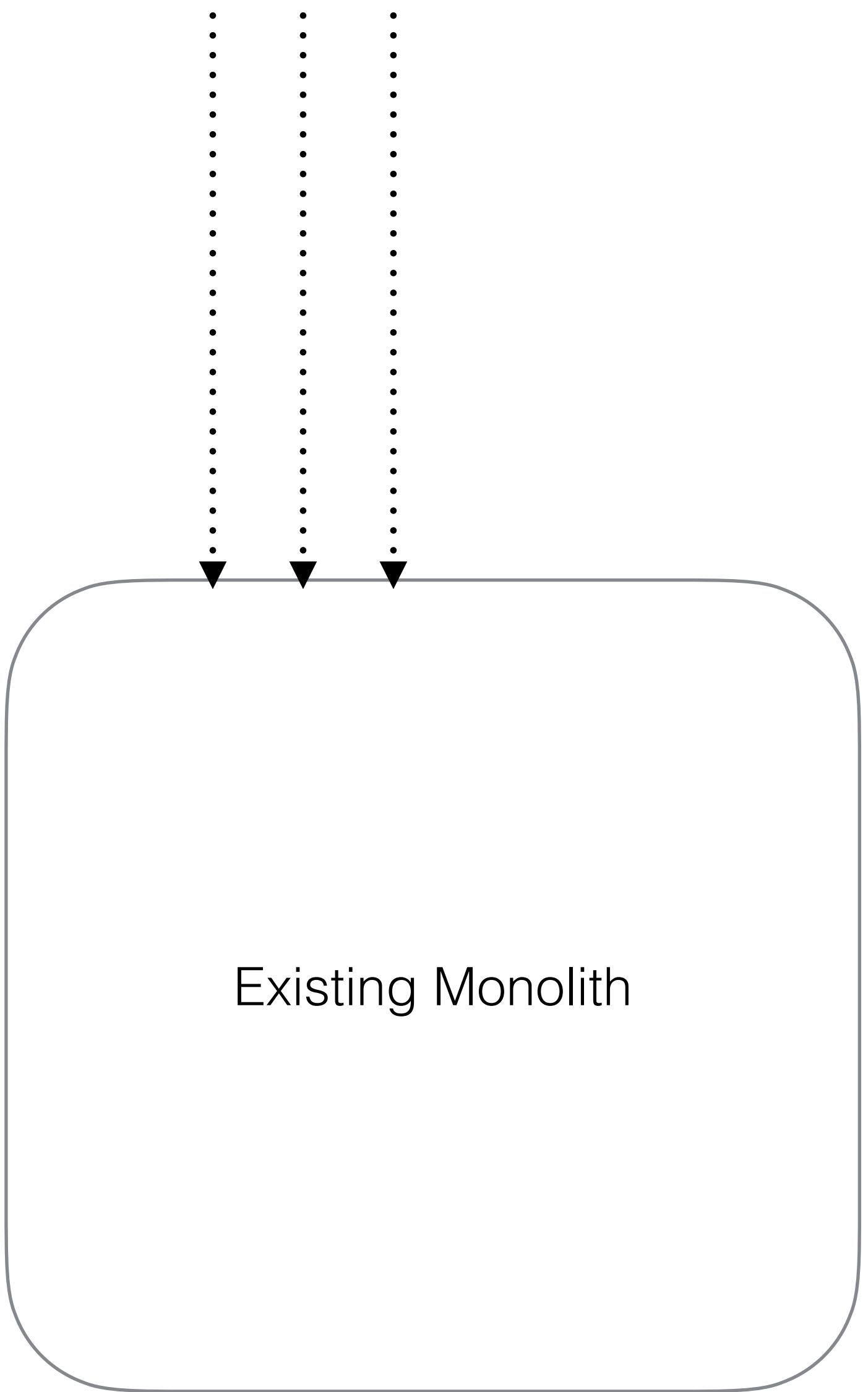
INTERCEPTION



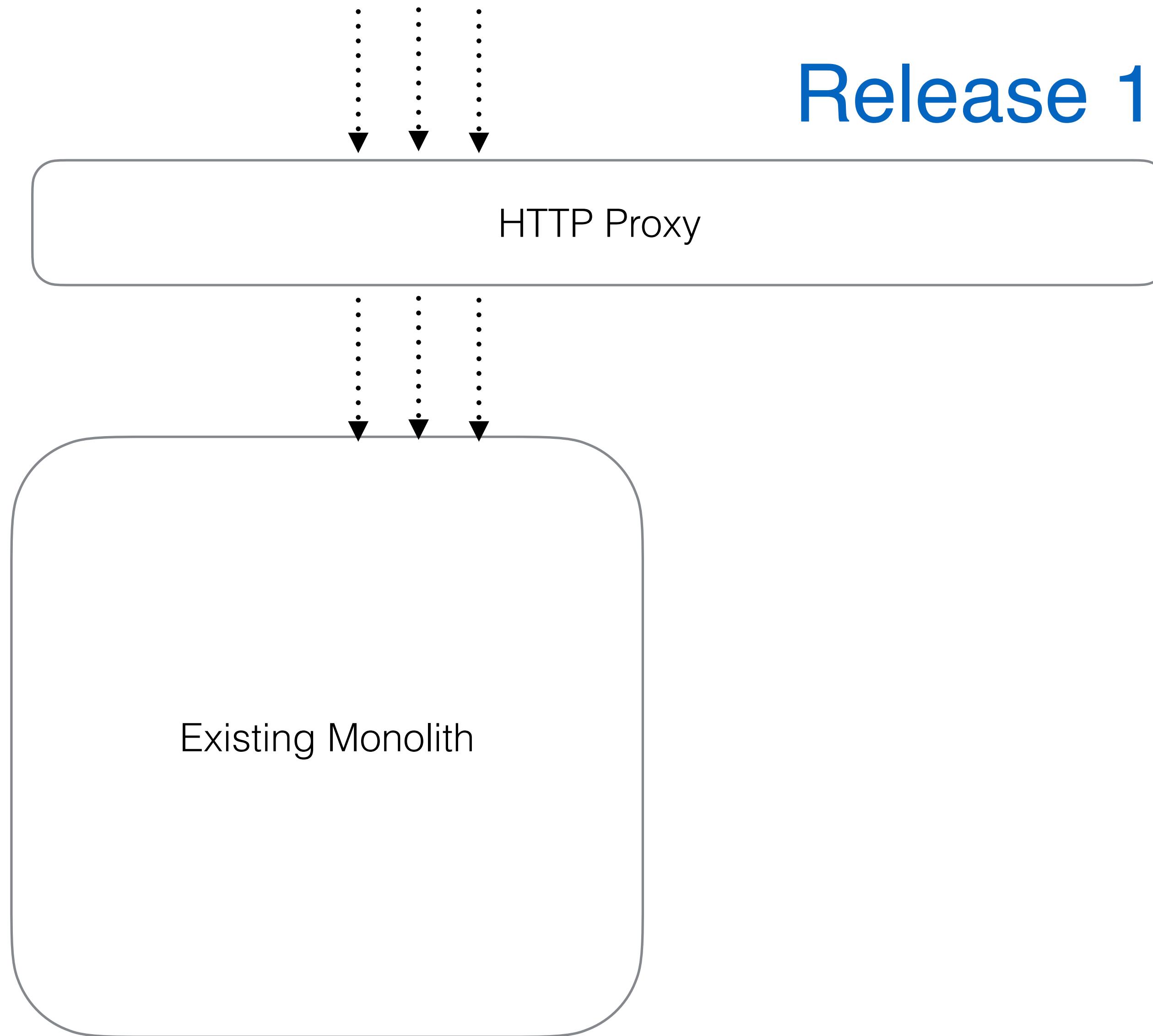
INTERCEPTION



INCREMENTAL ROLLOUT

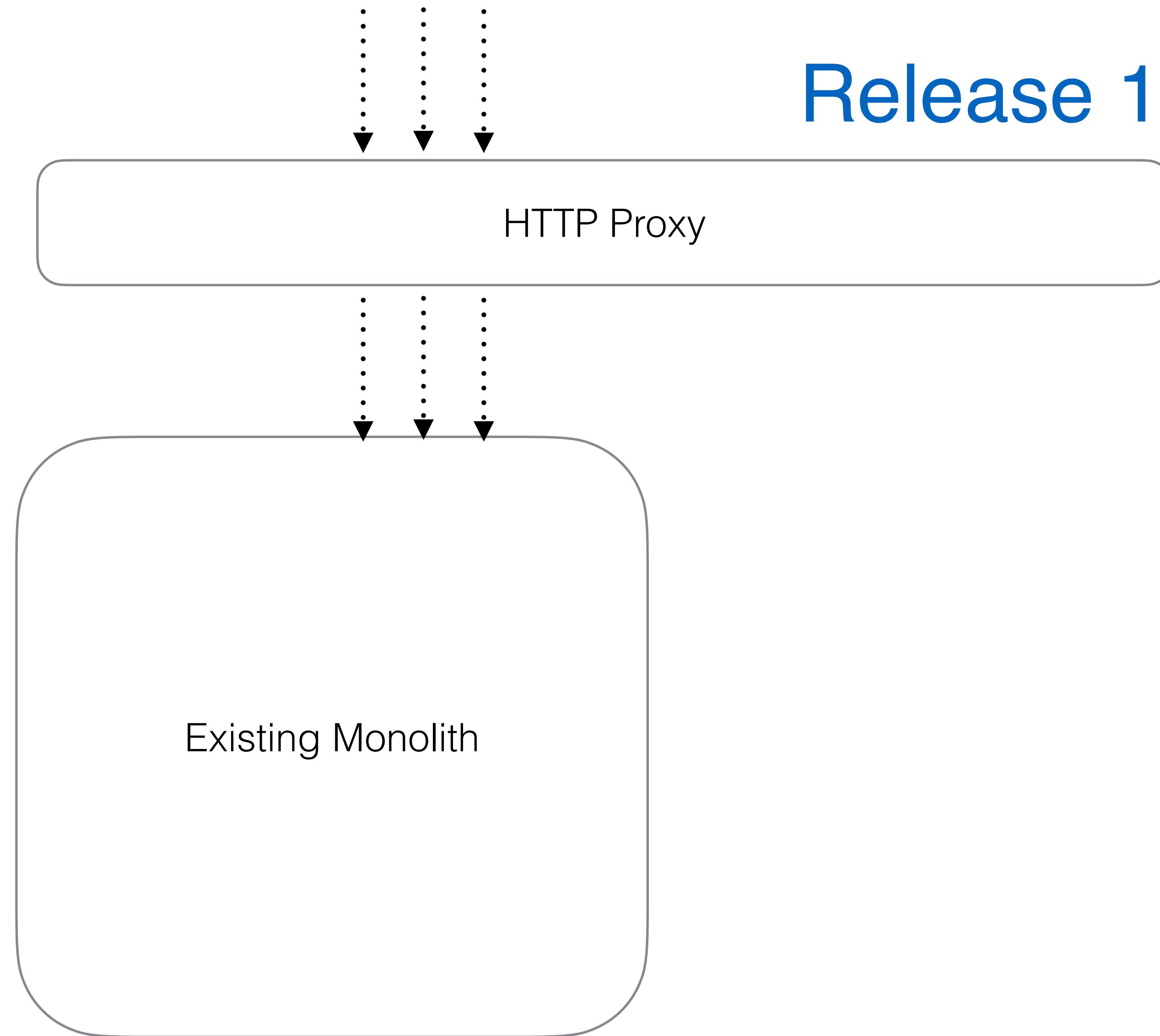


INCREMENTAL ROLLOUT



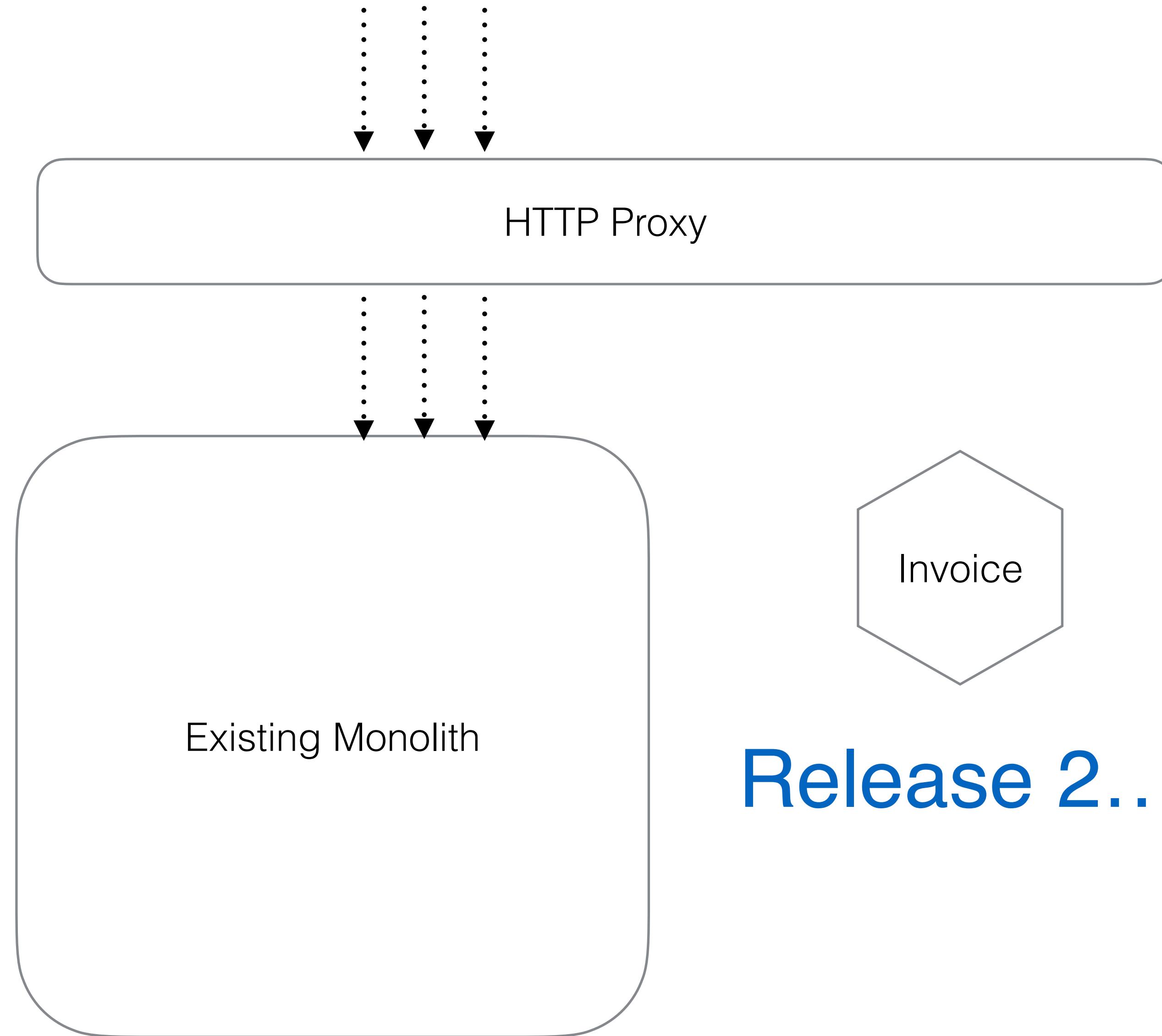
INCREMENTAL ROLLOUT

Assess impact of
adding a network
hop early



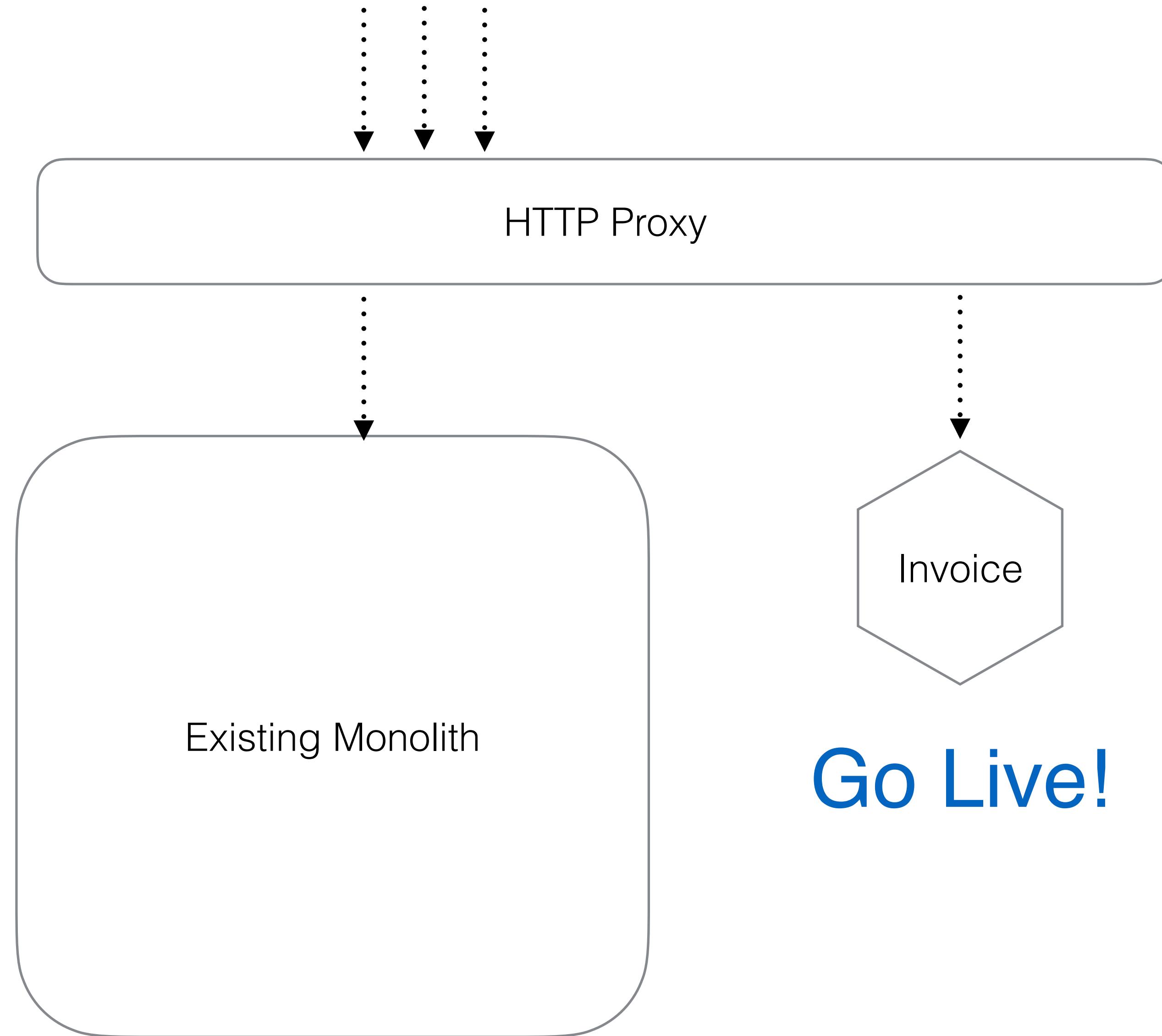
INCREMENTAL ROLLOUT

Assess impact of
adding a network
hop early

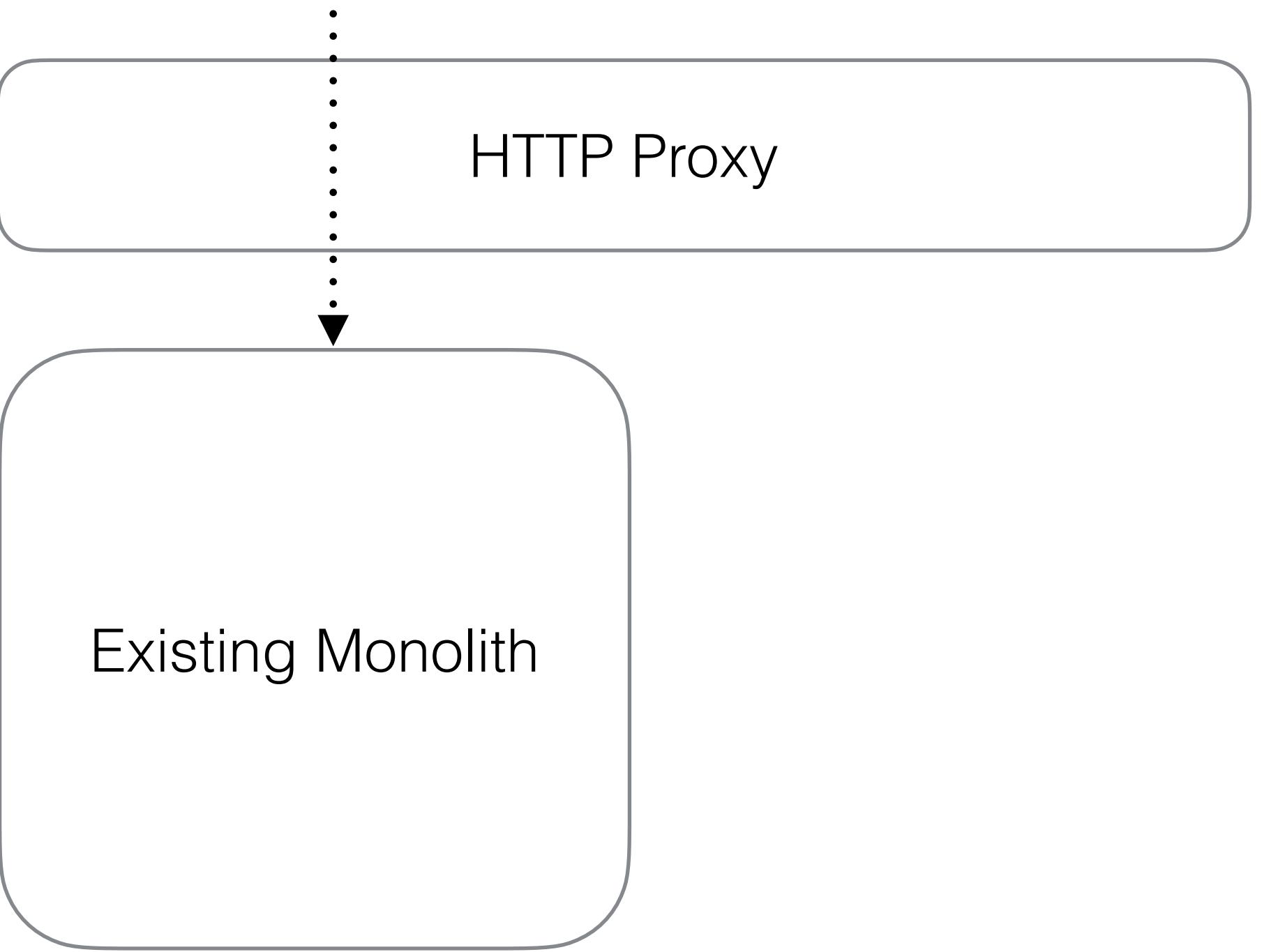


INCREMENTAL ROLLOUT

Assess impact of
adding a network
hop early

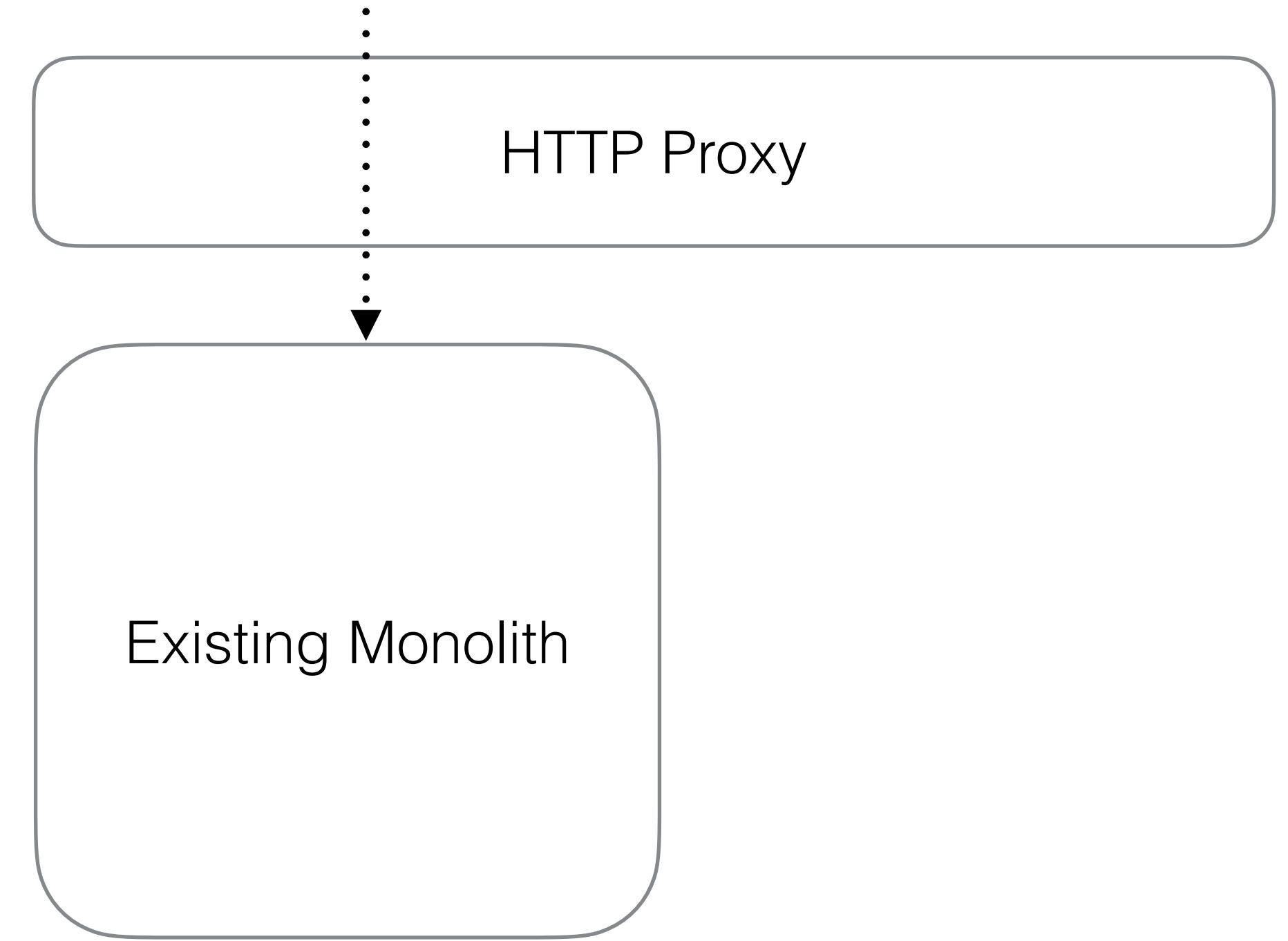


DEPLOYING WORK THAT ISN'T READY?



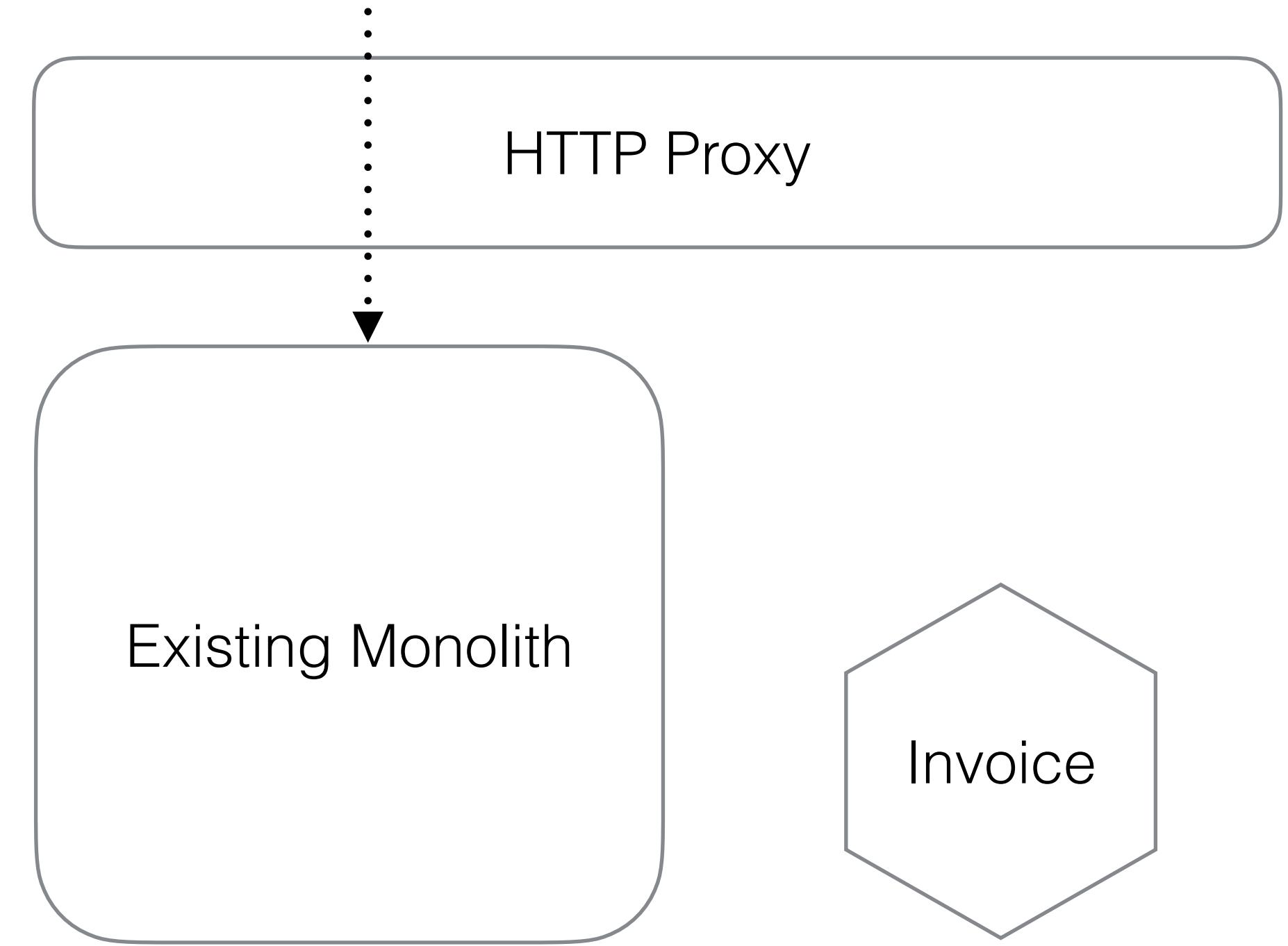
DEPLOYING WORK THAT ISN'T READY?

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”

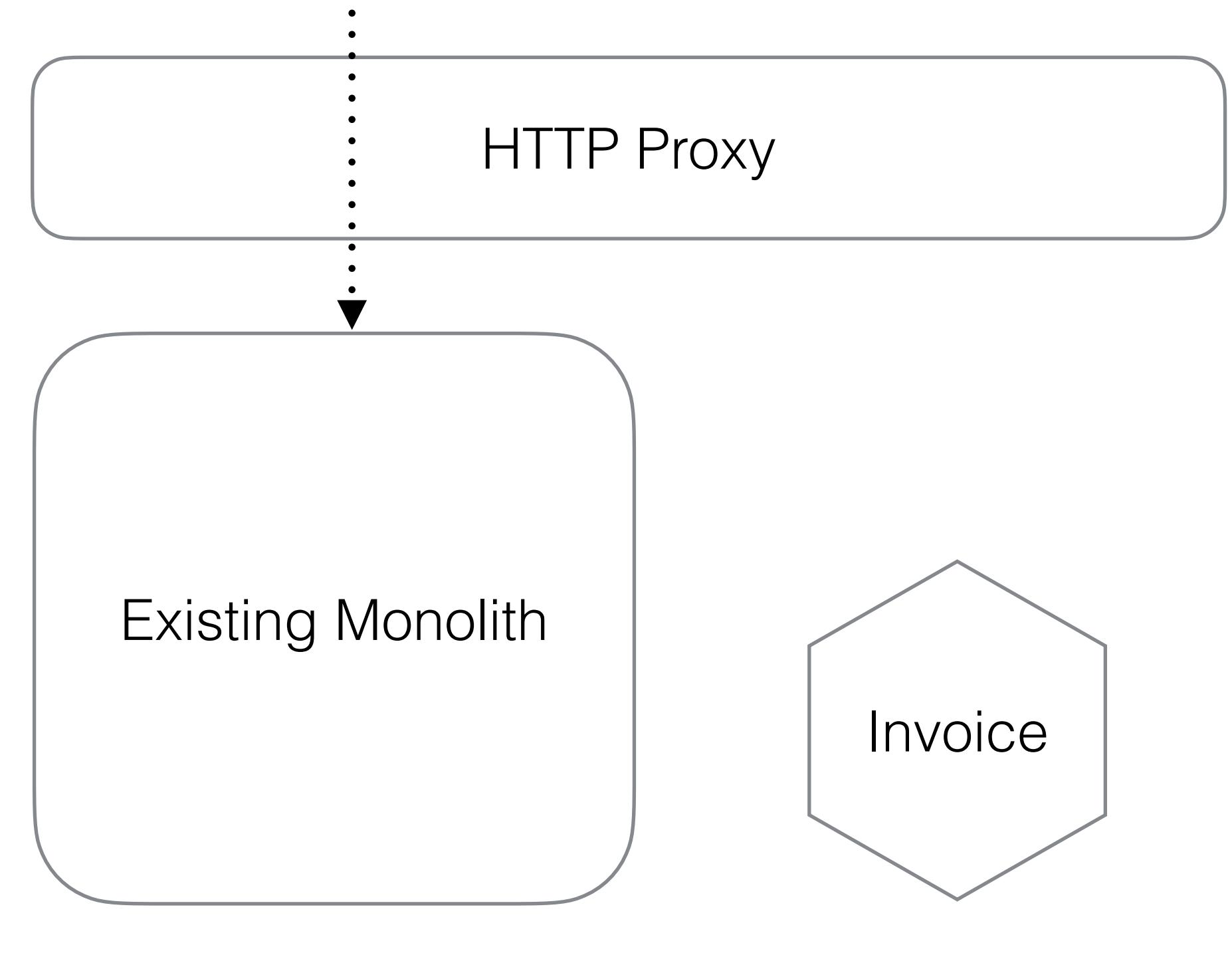


Invoice microservice is deployed

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

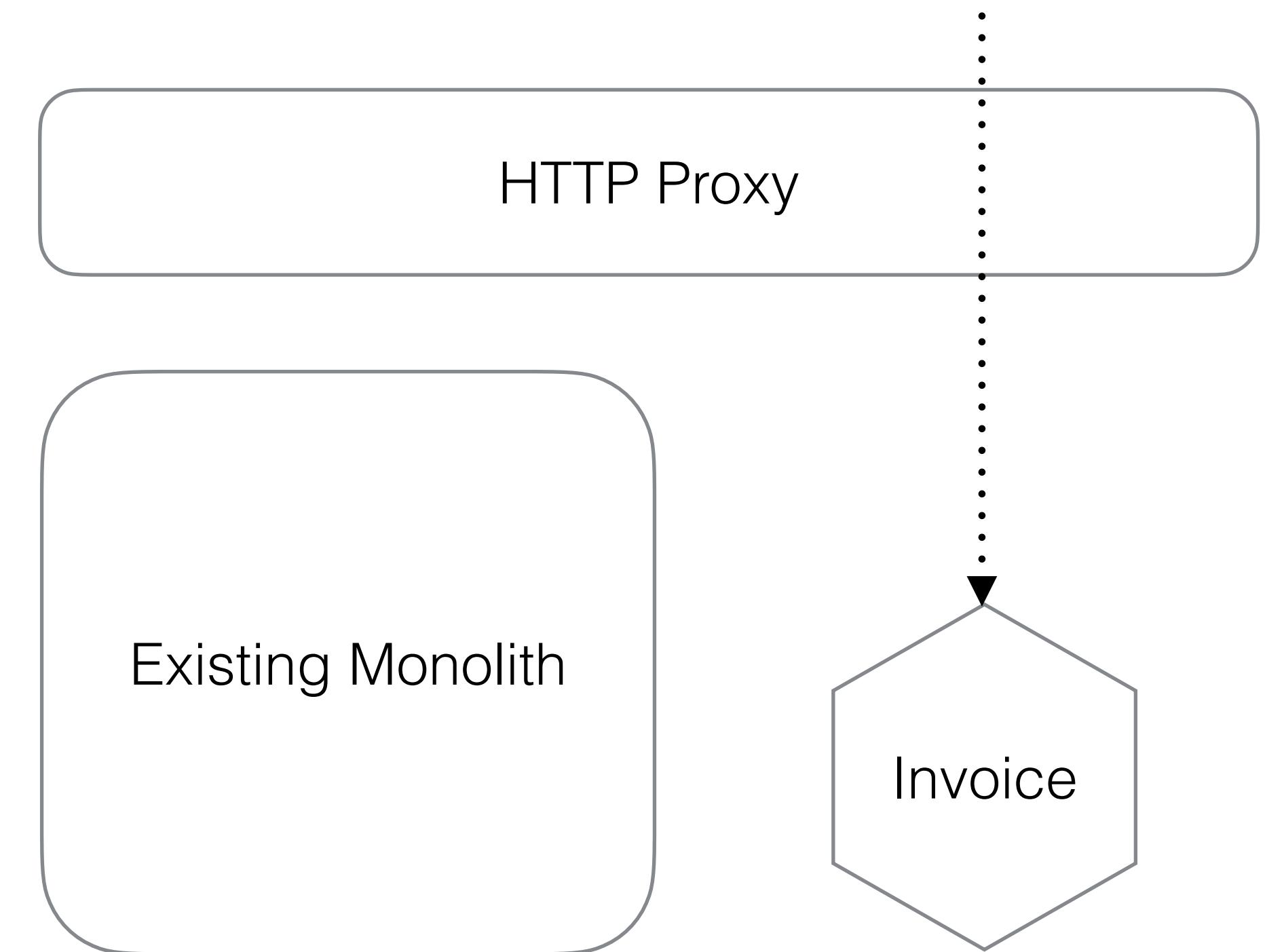


Invoice microservice is deployed

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

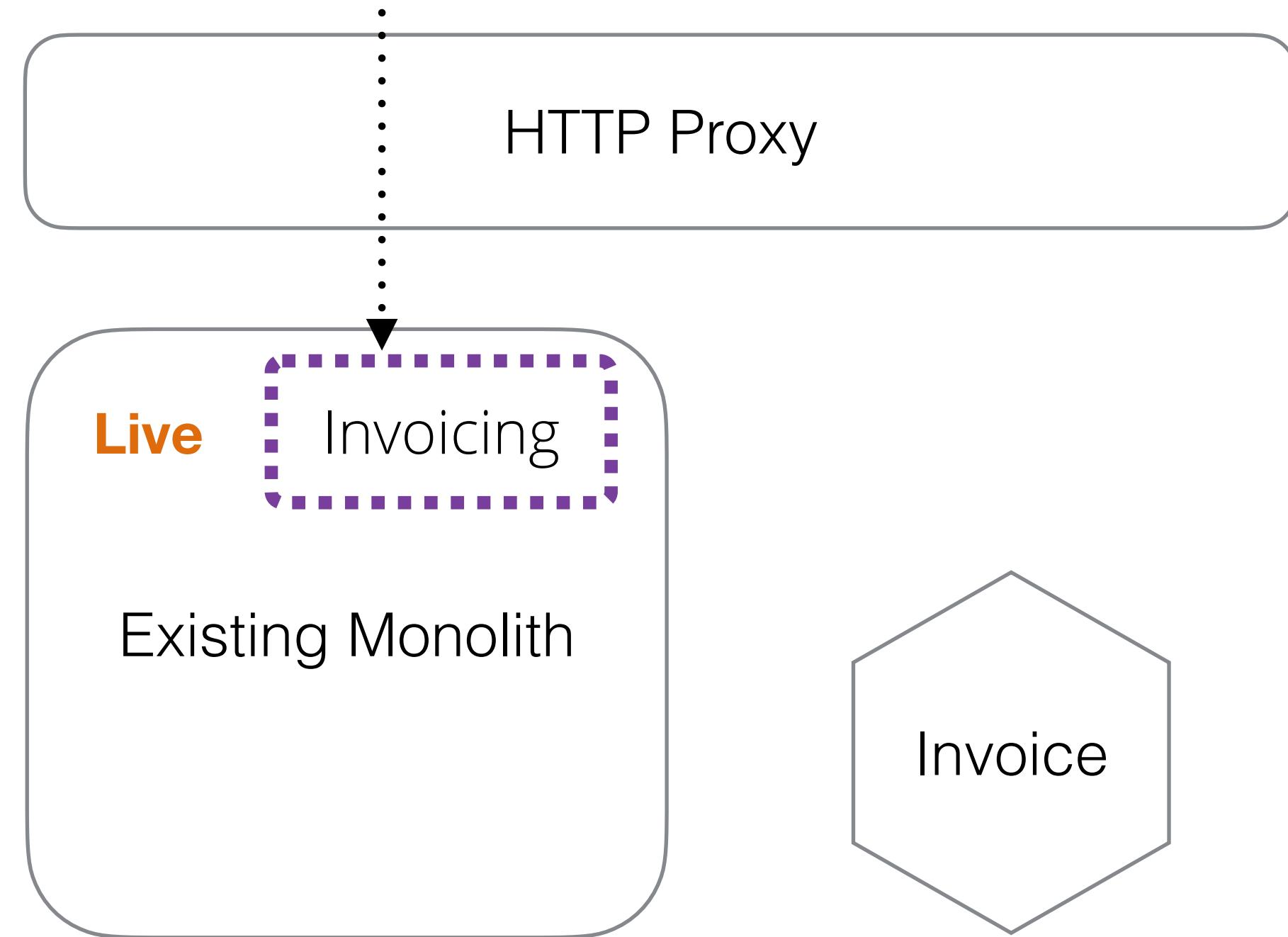
DON'T REMOVE THE OLD FUNCTIONALITY TOO SOON!

DON'T REMOVE THE OLD FUNCTIONALITY TOO SOON!

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

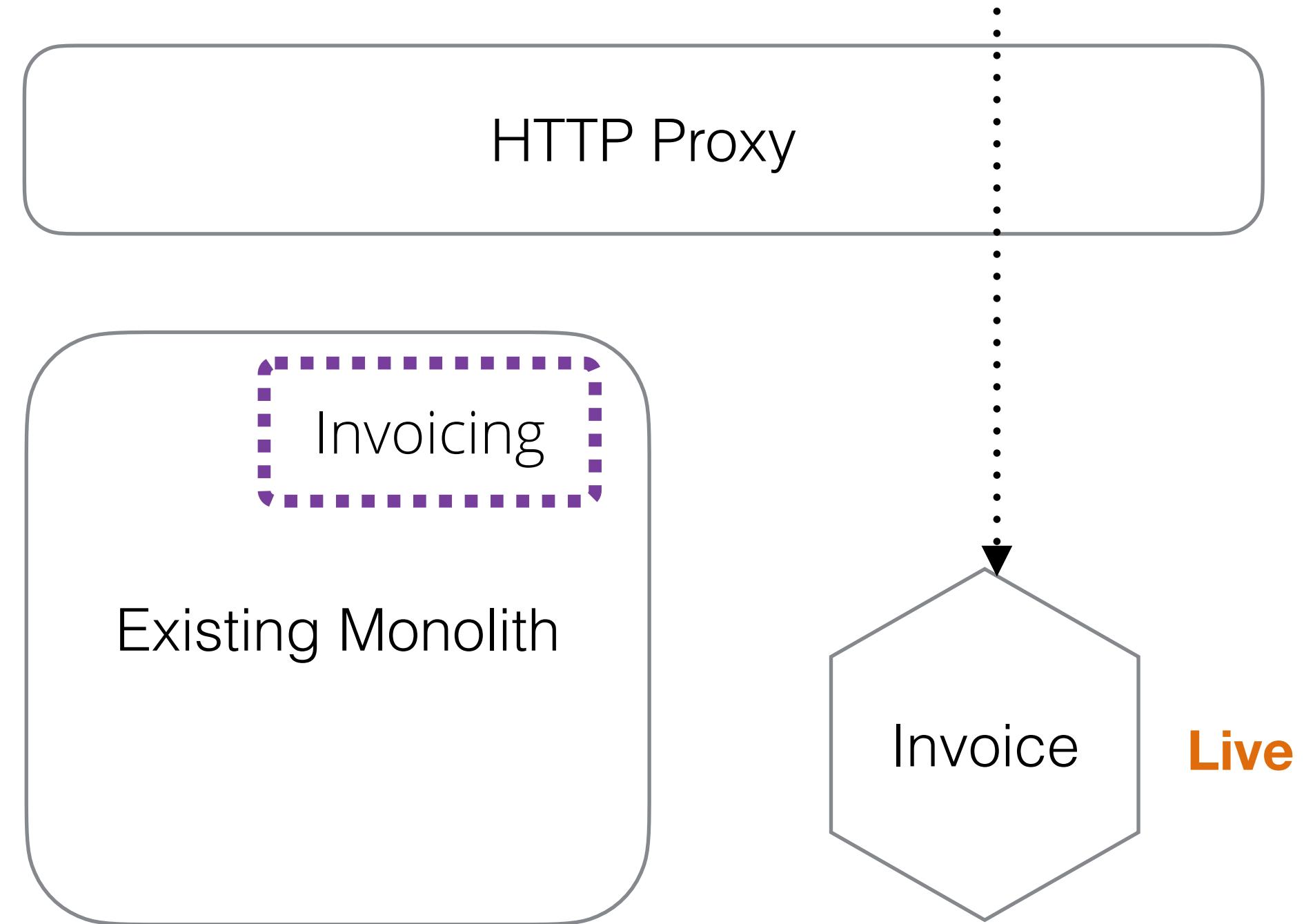
DON'T REMOVE THE OLD FUNCTIONALITY TOO SOON!

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



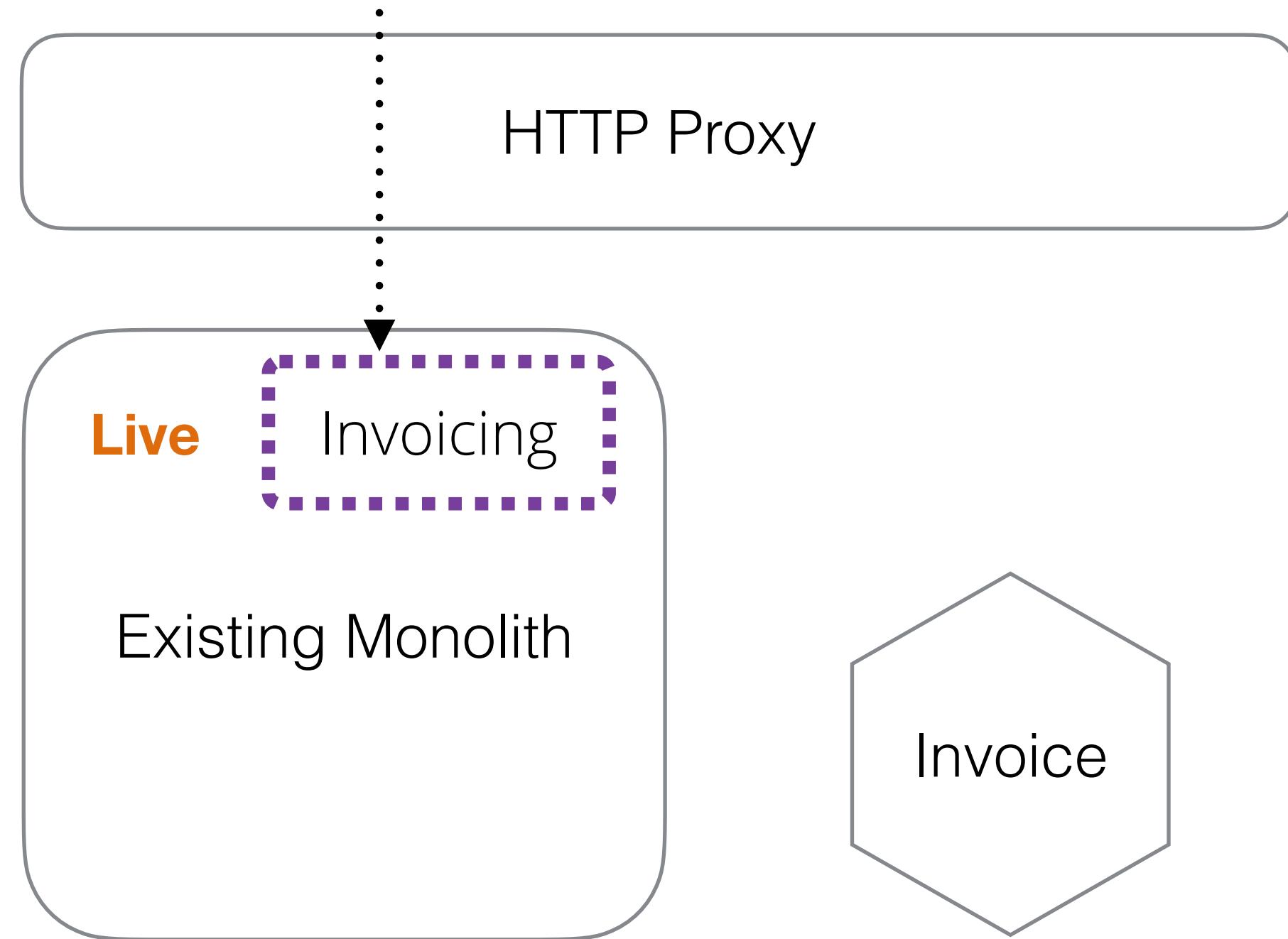
DON'T REMOVE THE OLD FUNCTIONALITY TOO SOON!

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



DON'T REMOVE THE OLD FUNCTIONALITY TOO SOON!

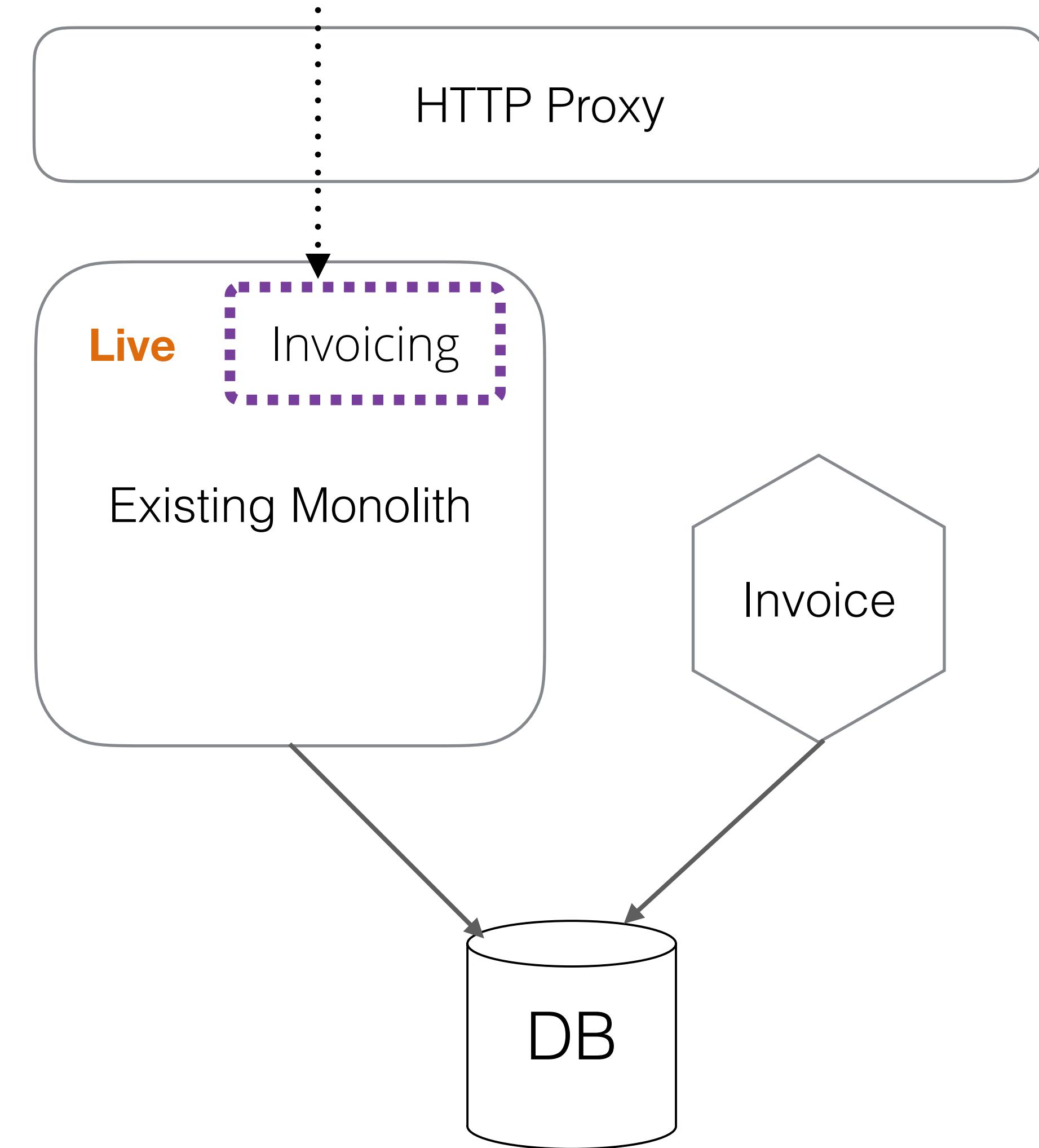
If we leave the old invoicing functionality in the monolith, we have an easy rollback



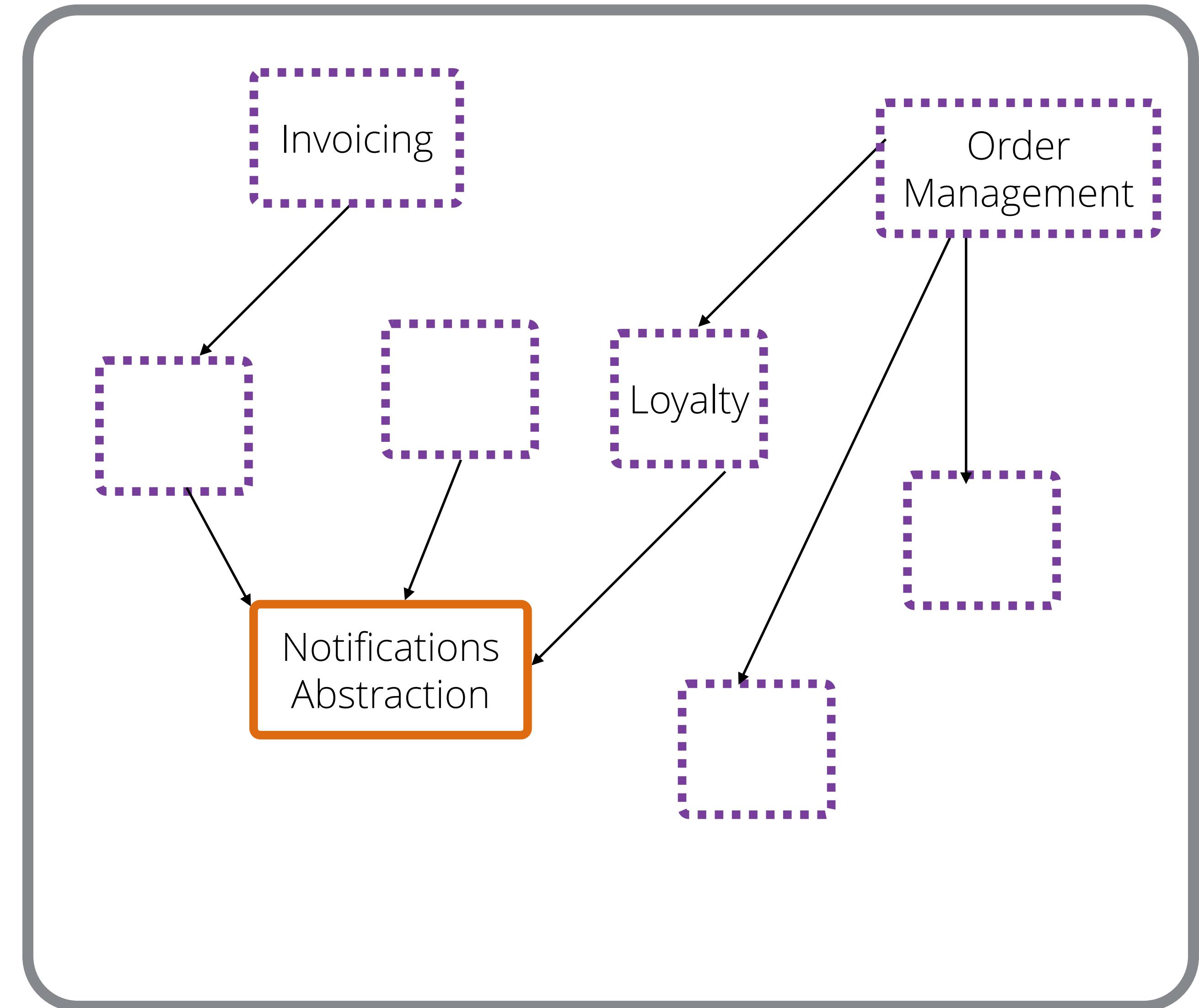
DON'T REMOVE THE OLD FUNCTIONALITY TOO SOON!

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

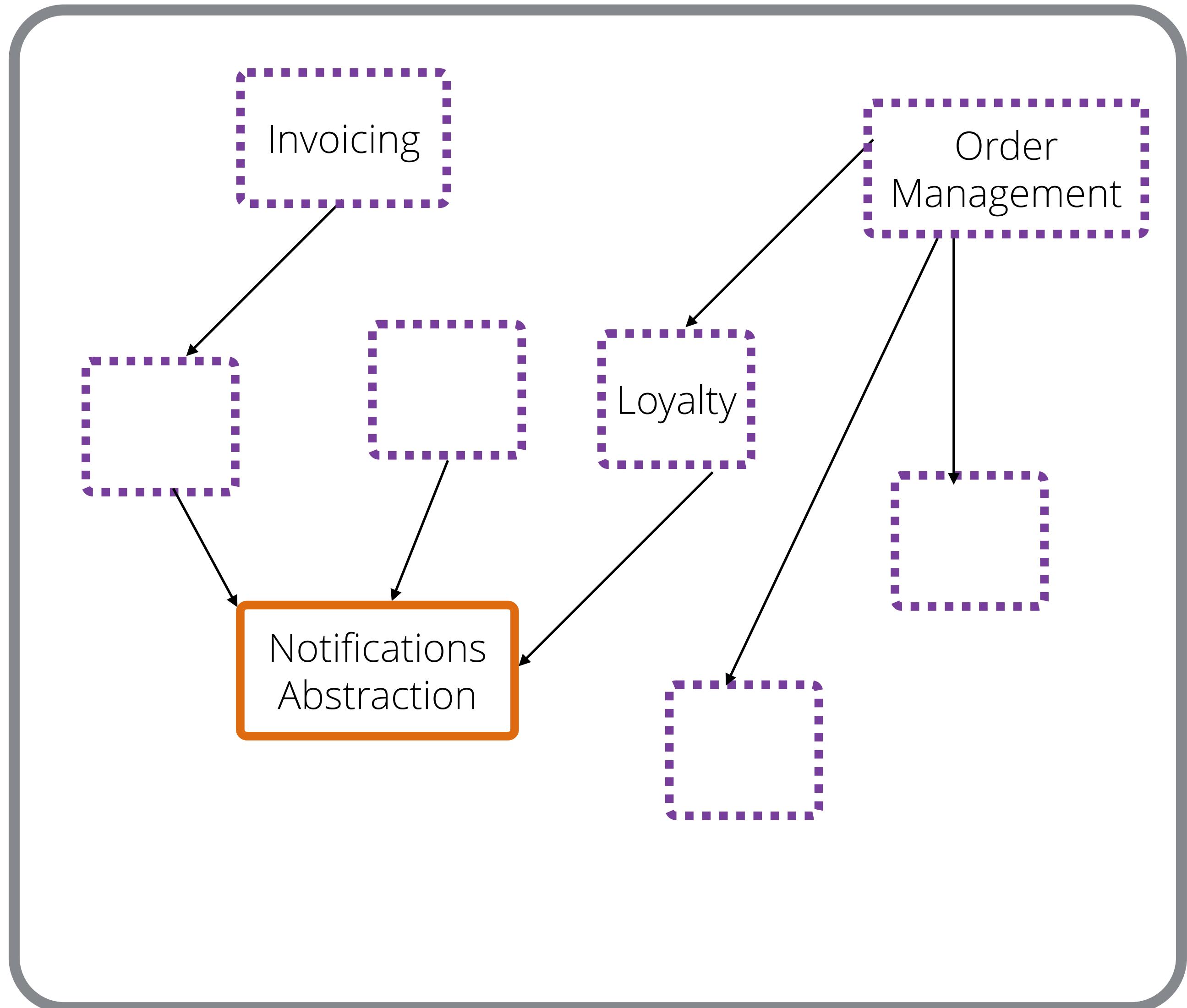


BRANCH BY ABSTRACTION



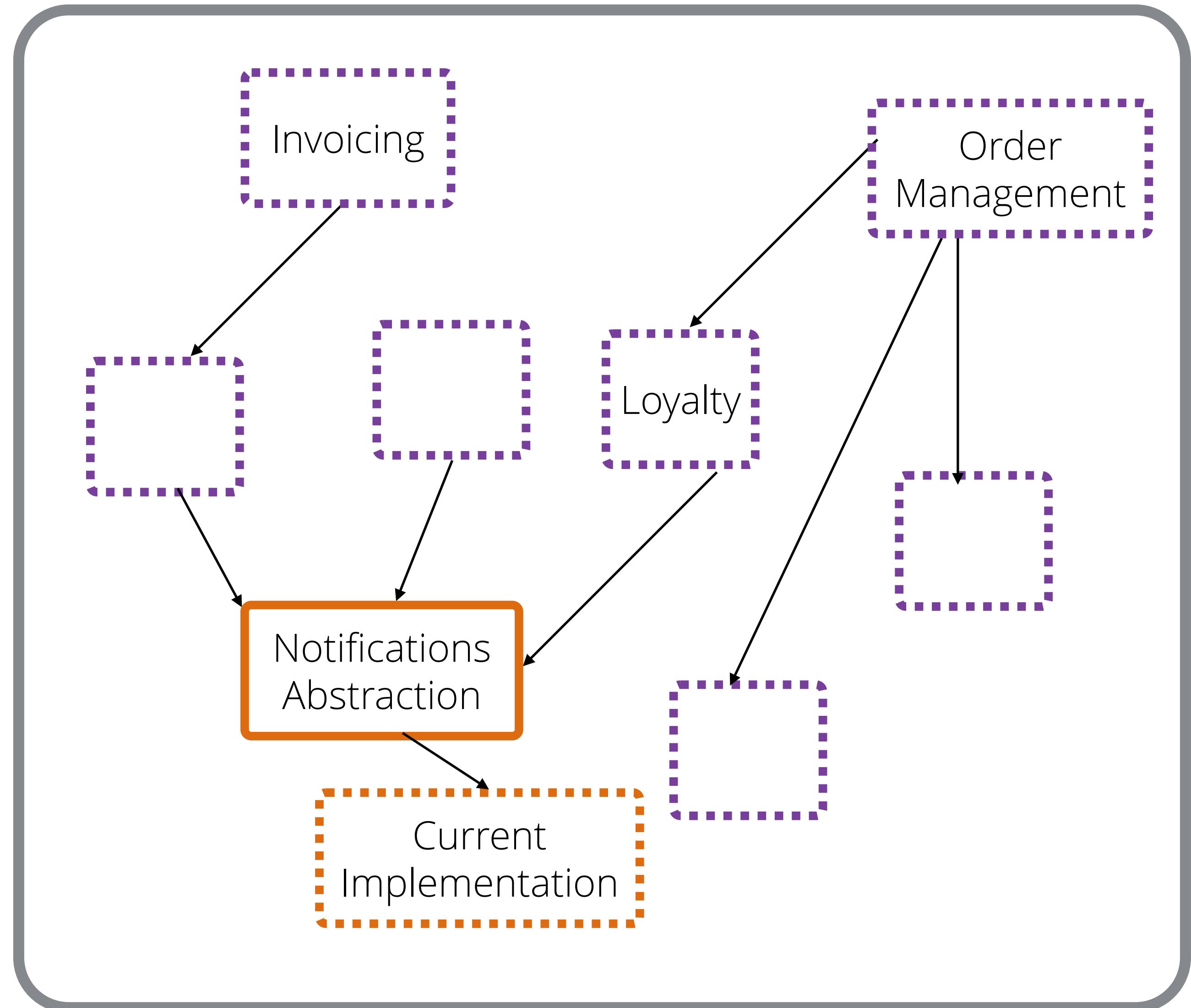
BRANCH BY ABSTRACTION

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



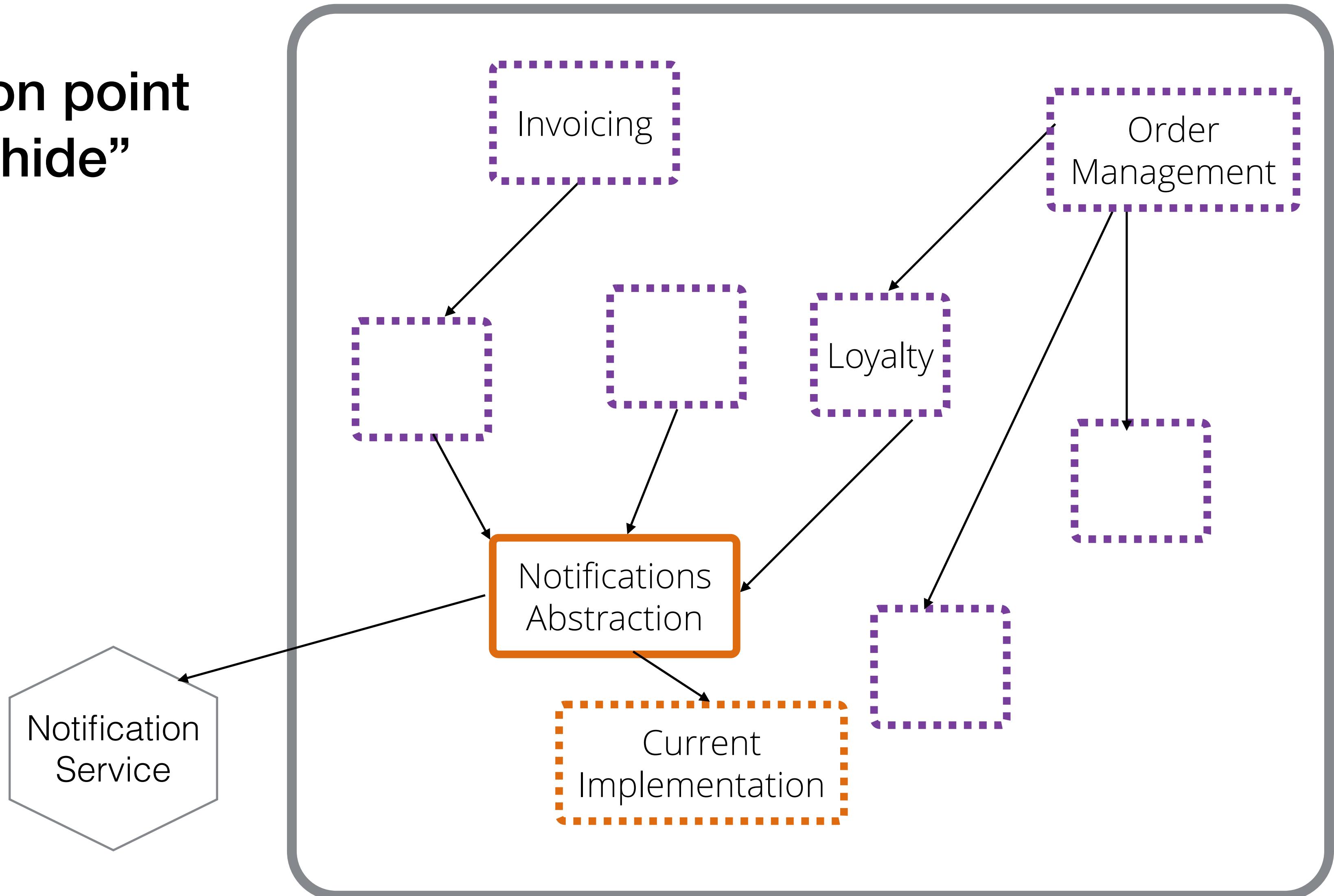
BRANCH BY ABSTRACTION

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



BRANCH BY ABSTRACTION

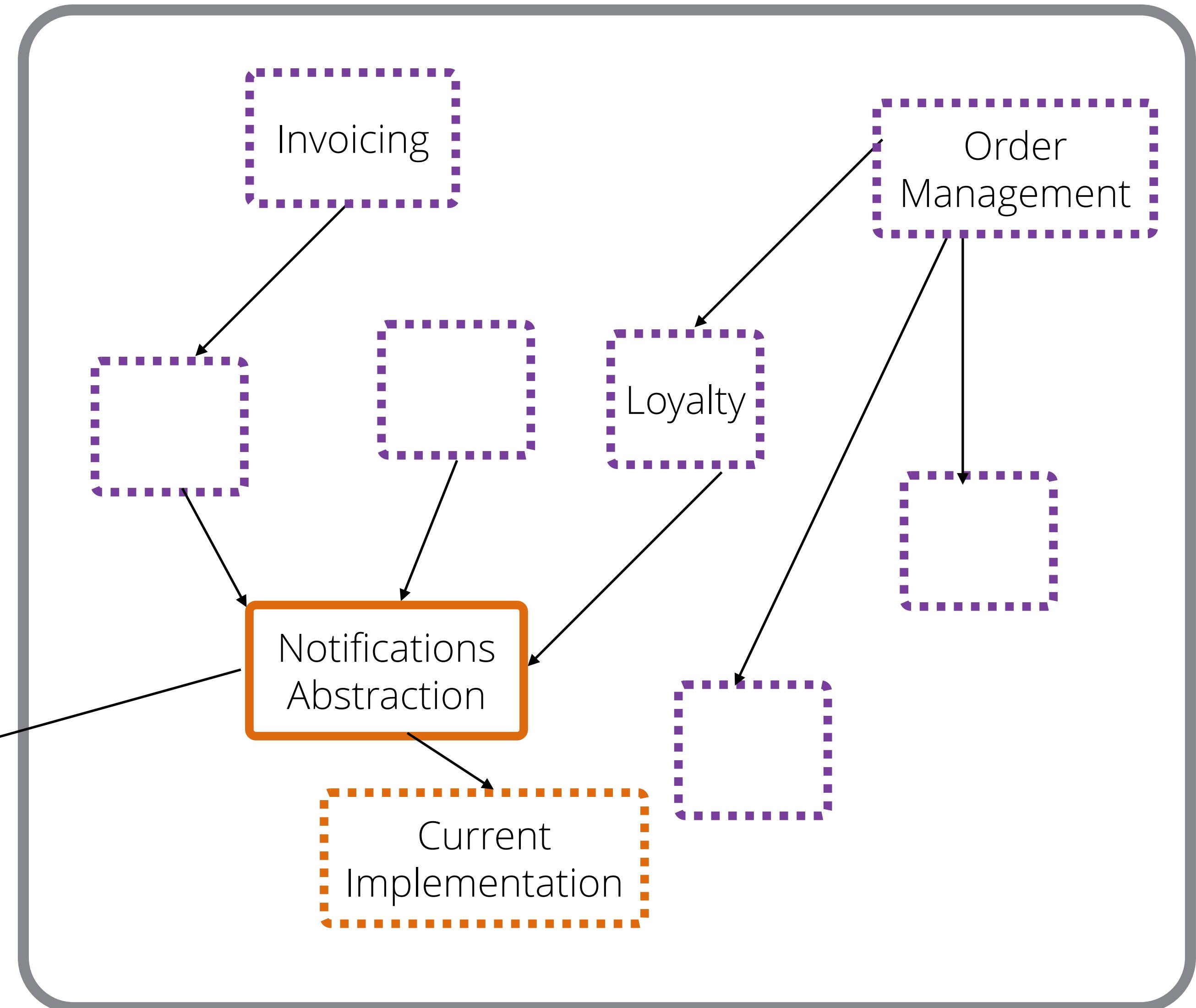
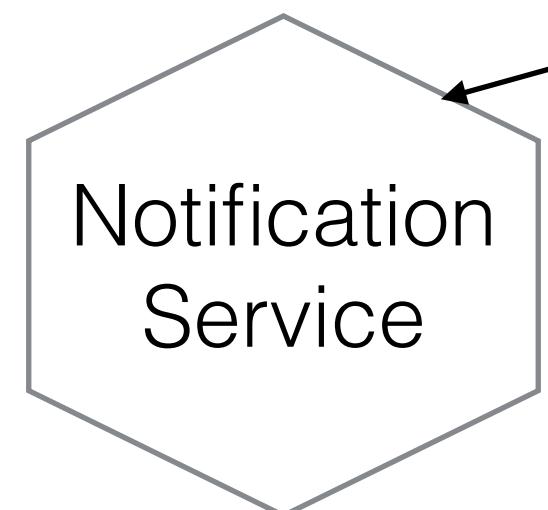
Gives you an abstraction point which can be used to “hide” ongoing development



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

Feature Branches And Toggles In A Post-GitHub World.

45-90min Presentation

The diagram illustrates a workflow for managing feature branches and toggles. It features two vertical black bars representing repositories. In the center, a horizontal dashed blue line represents the main trunk. Four purple circular nodes are connected by dotted lines to the trunk. The top node has three green checkmarks below it. The second node from the left has one green checkmark and one red X. The third node has one green checkmark. The bottom node has three red X's. Three human icons are positioned above the nodes: one above the top node, one above the second node, and one below the bottom node. To the right of the diagram, the text 'A talk re-examining feature branches and toggles' is written in a cursive font.

1. Validate the integration
2. When the build breaks, fix it!
3. Integrate daily

@devoxxpl @samnewman

A talk re-examining feature branches and toggles

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

AVOID LONG-LIVED SOURCE CODE BRANCHING

Feature Branches And Toggles In A Post-GitHub World.

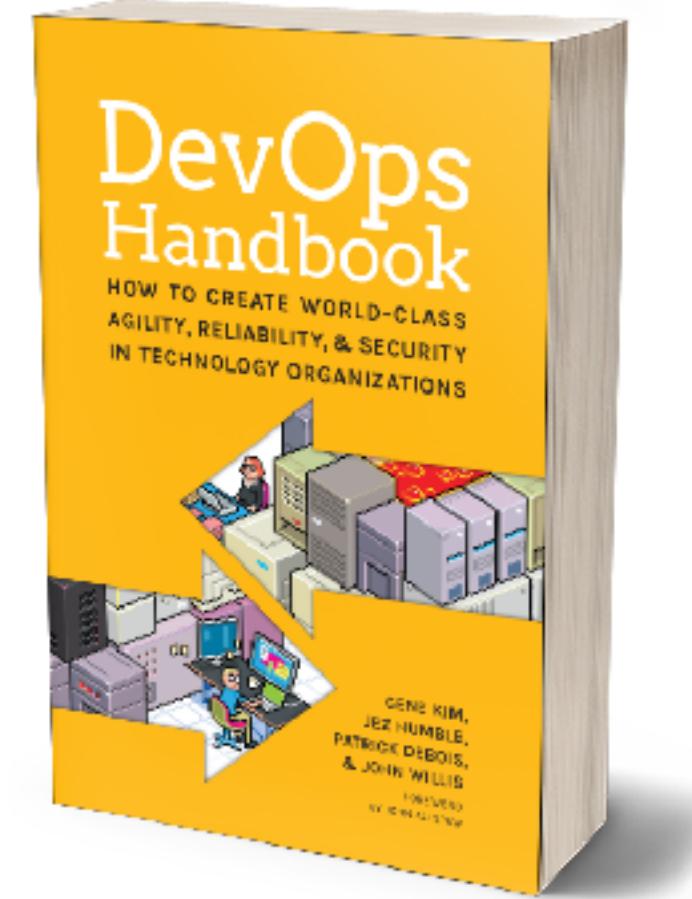
45-90min Presentation

The diagram shows two paths from a developer icon at the top to a large purple circle representing the main codebase. The left path, labeled 'feature branches', has several small purple circles along its way, each with a green checkmark or a red 'X'. The right path, labeled 'trunk-based development', is a single solid blue line with a green checkmark at its end. Above the paths, three steps are listed: 1. Validate the integration, 2. When the build breaks, fix it!, and 3. Integrate daily.

1. Validate the integration
2. When the build breaks, fix it!
3. Integrate daily

@devoxxpl @samnewman

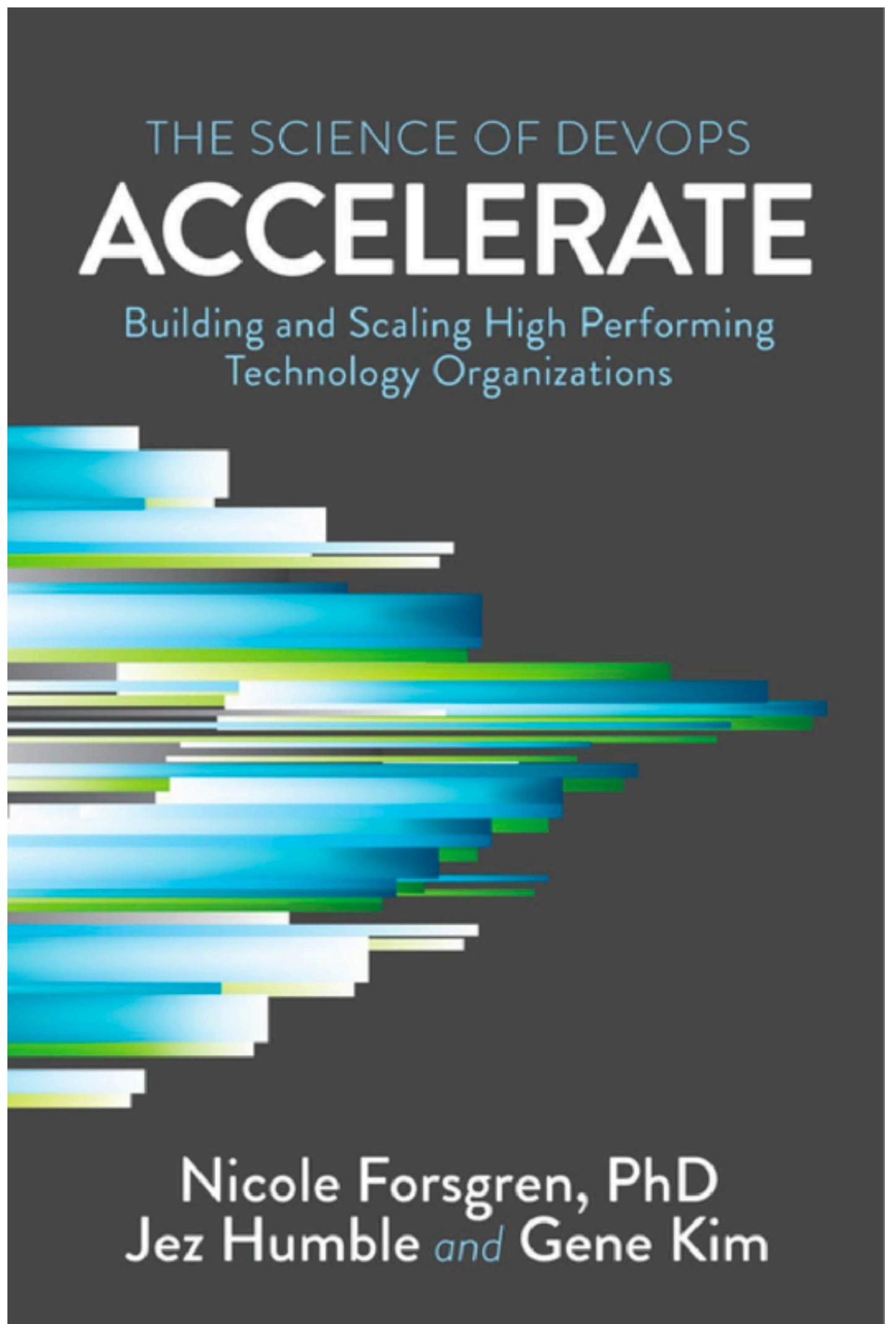
A talk re-examining feature branches and toggles



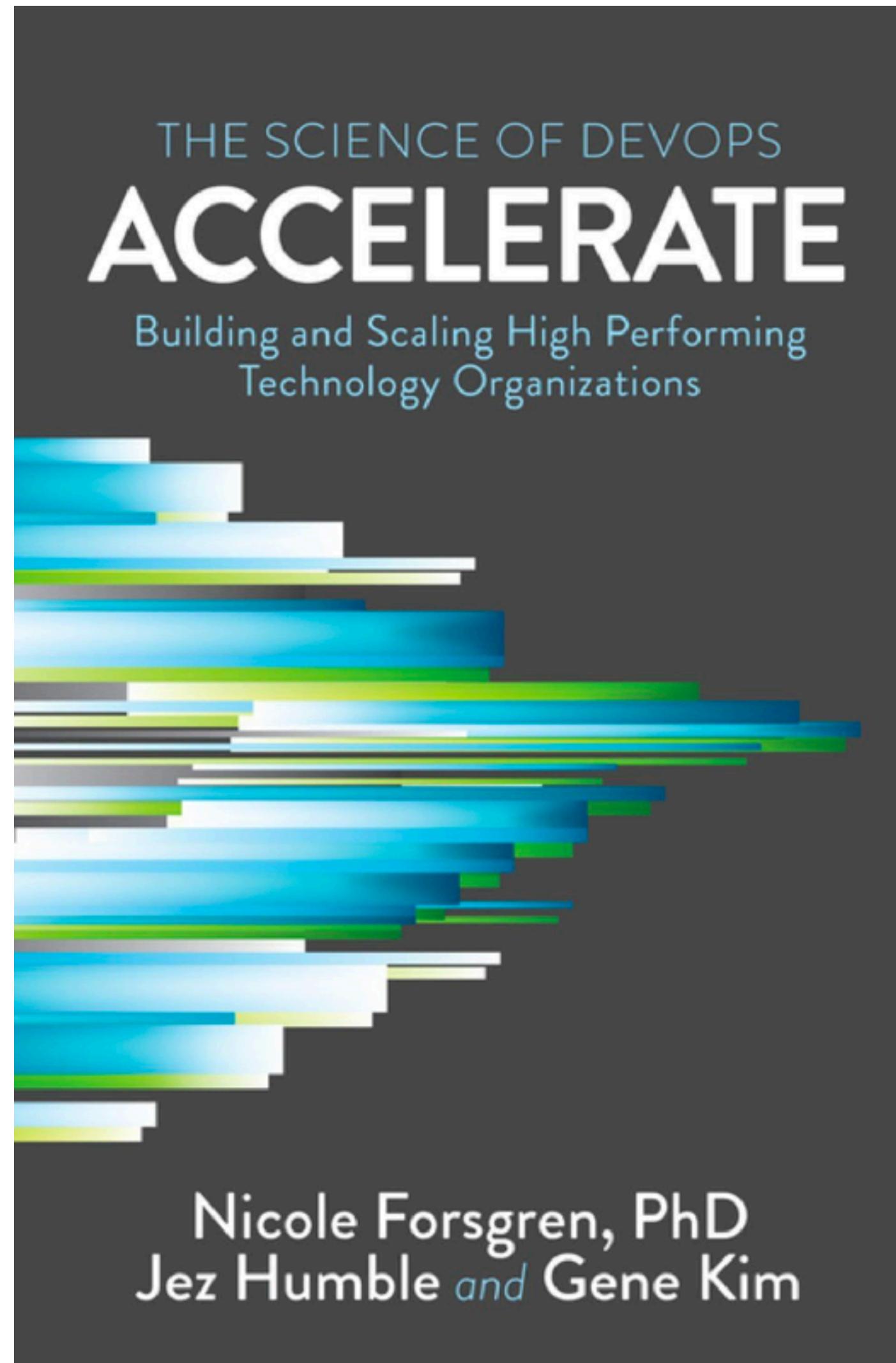
“...trunk-based development predicts higher throughput and better stability, and even higher job satisfaction and lower rates of burnout.”

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

ACCELERATE (READ THIS BOOK)

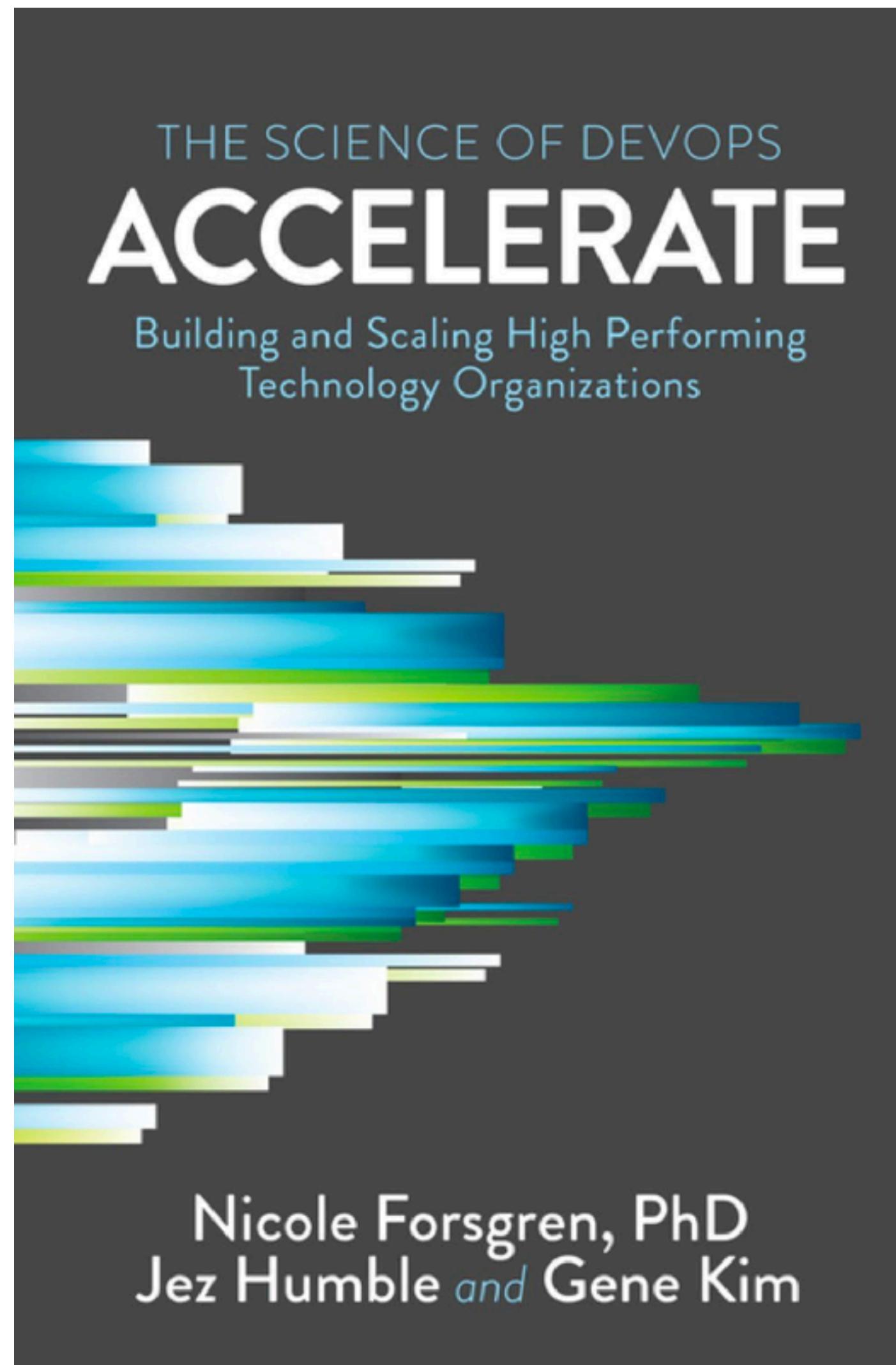


ACCELERATE (READ THIS BOOK)



“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)

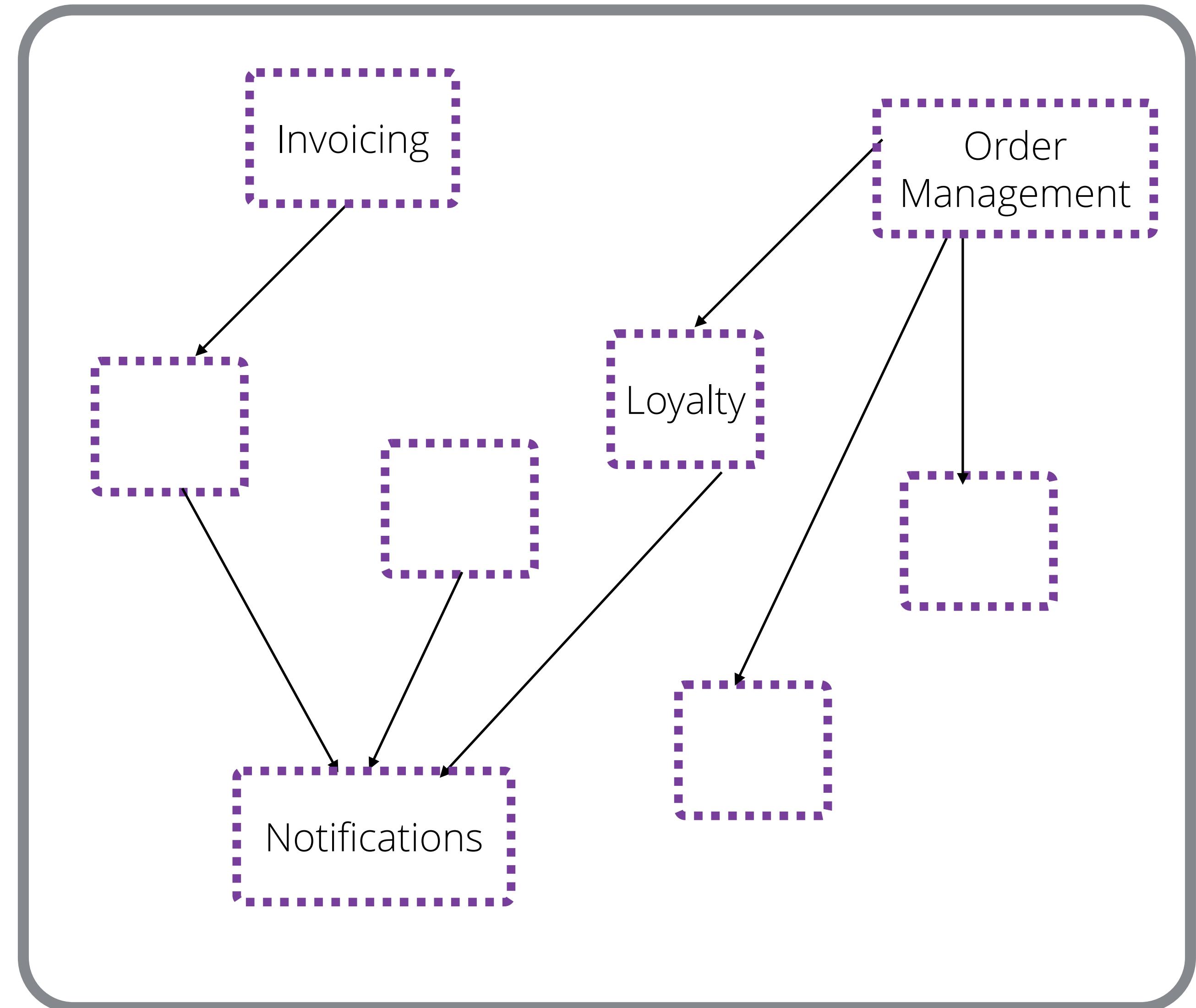


“Our research also found that developing off trunk/master rather than on long-lived feature branches was correlated with higher delivery performance.”

“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.”

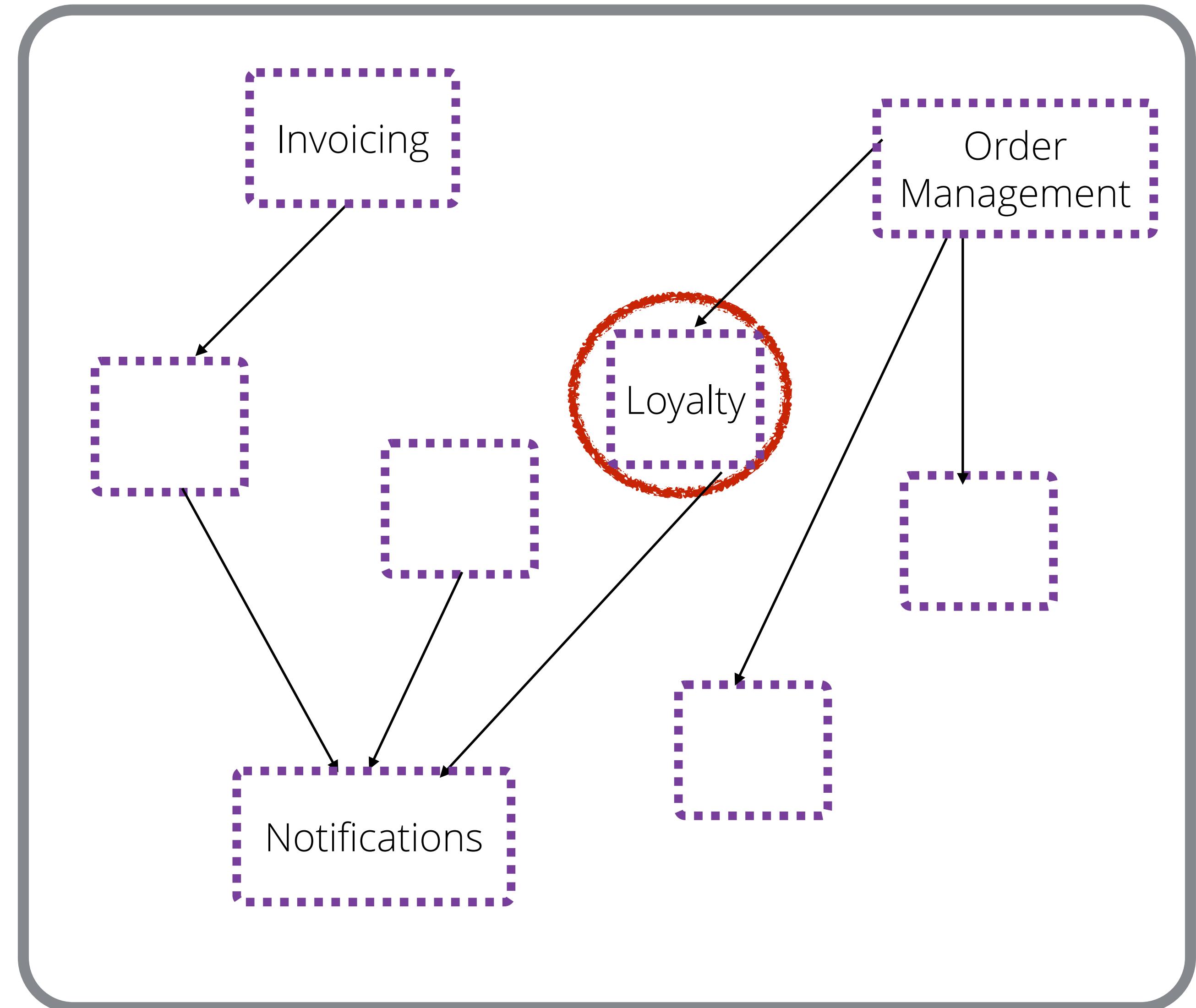
CANDIDATES FOR BRANCH BY ABSTRACTION

Can be used to replace functionality deeper inside the monolith



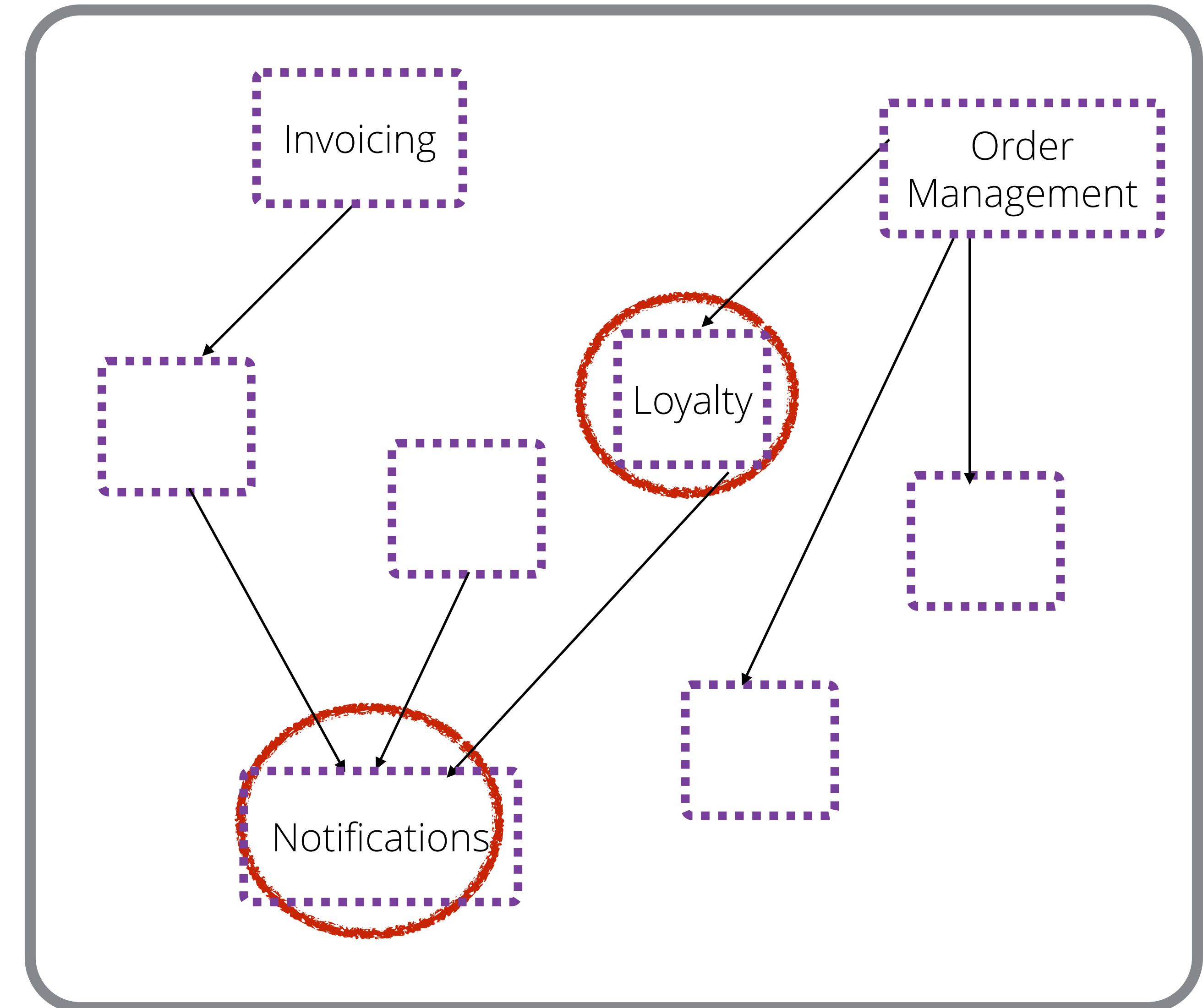
CANDIDATES FOR BRANCH BY ABSTRACTION

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



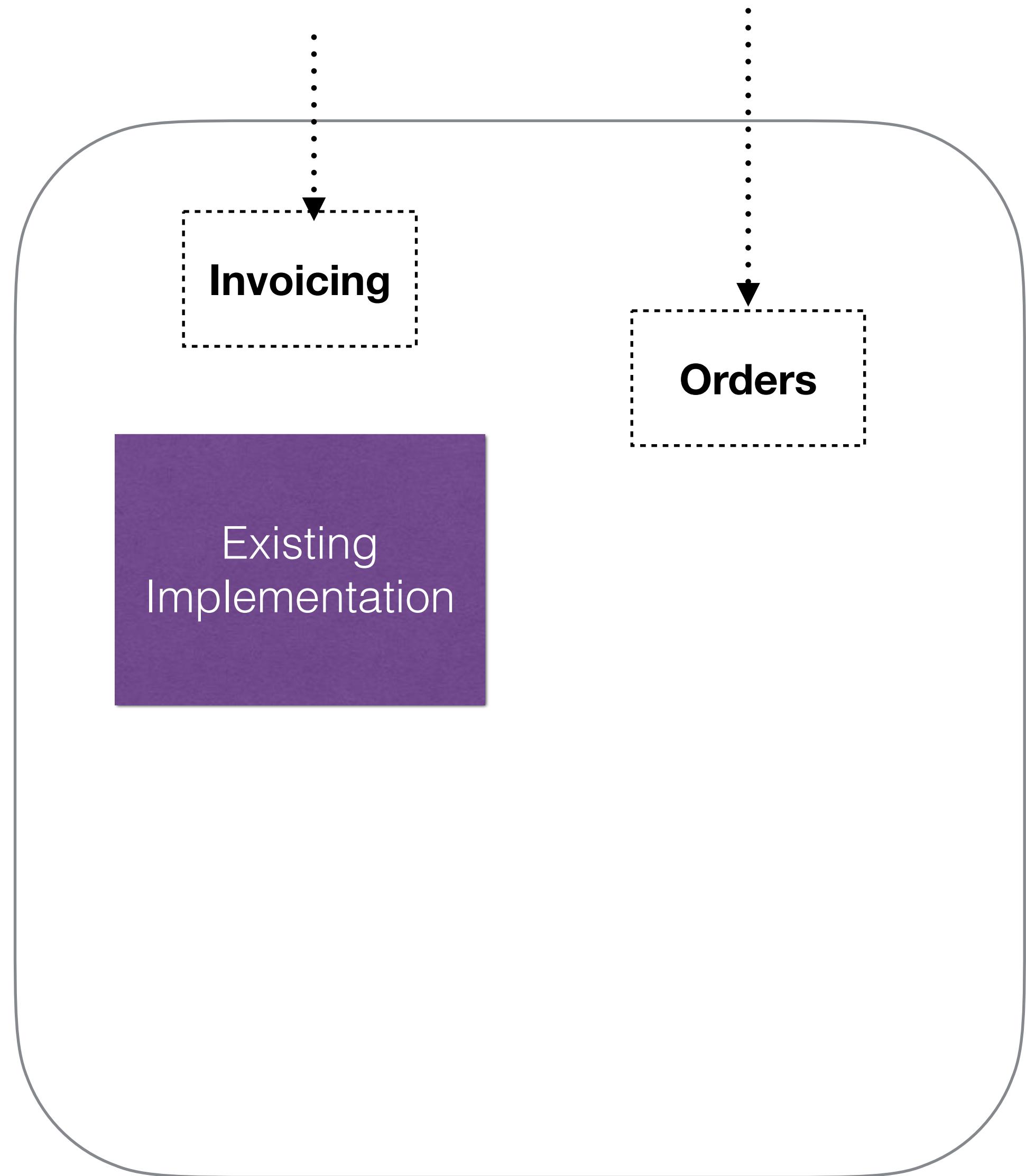
EXAMPLE



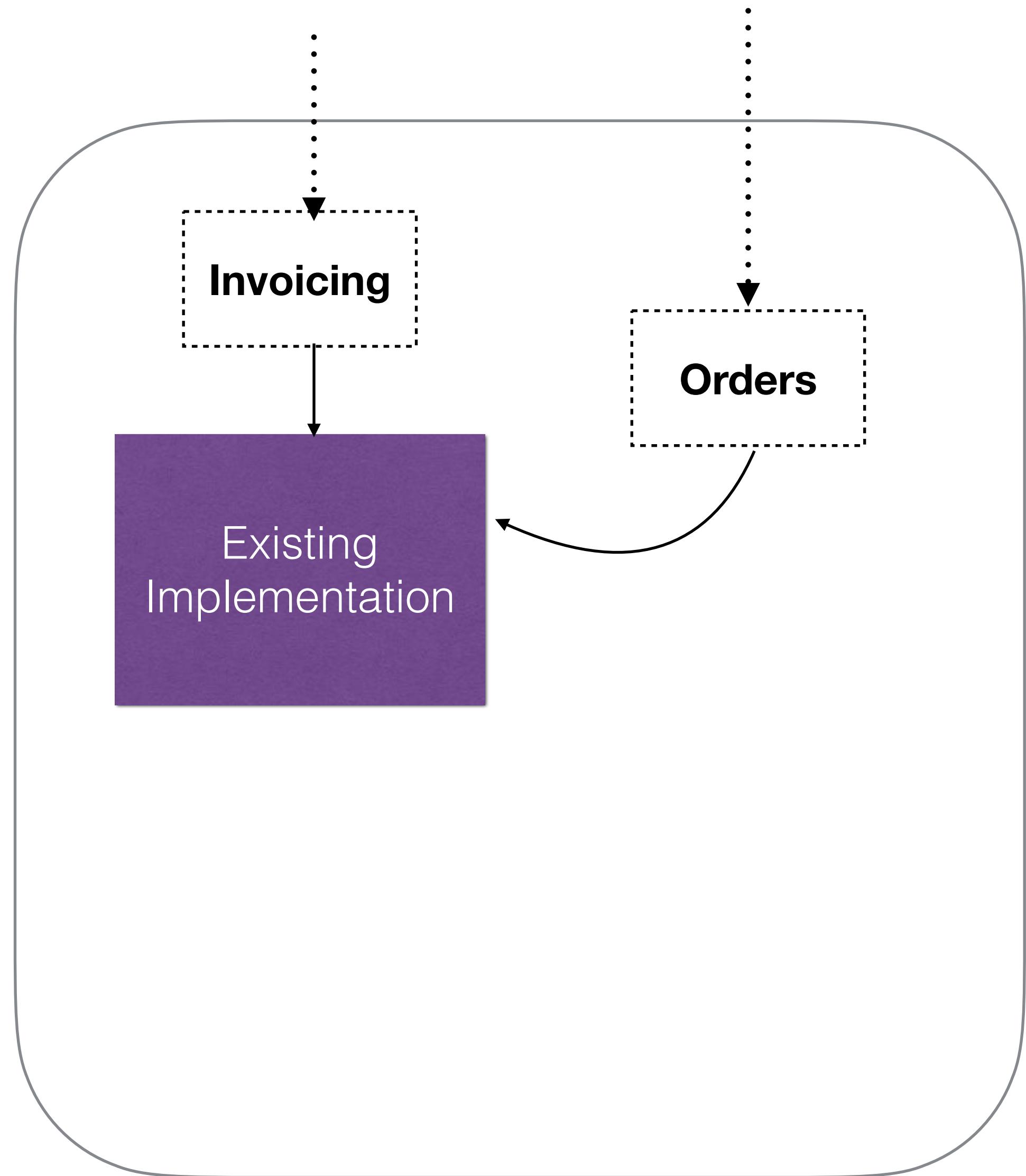
EXAMPLE



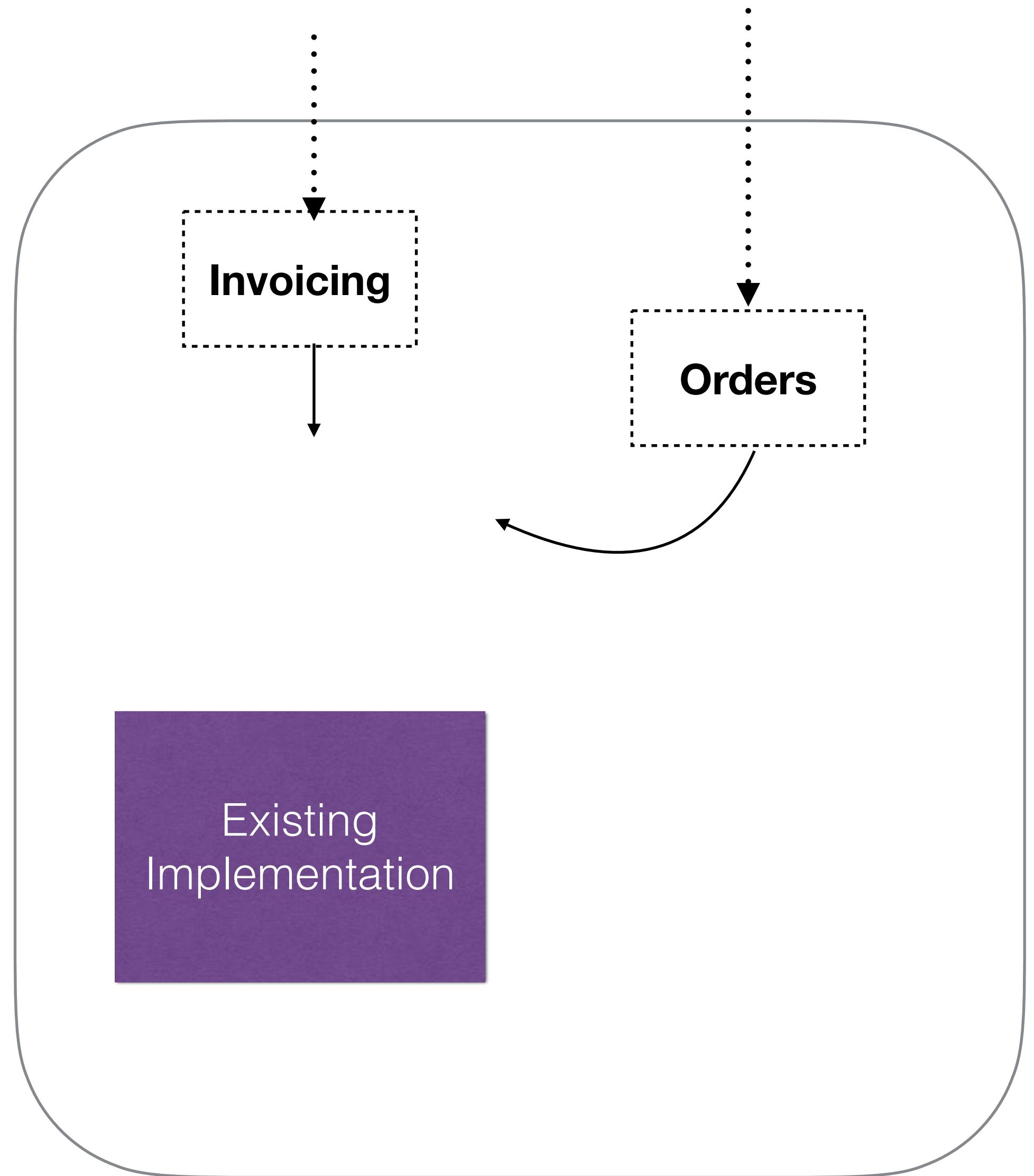
EXAMPLE



EXAMPLE

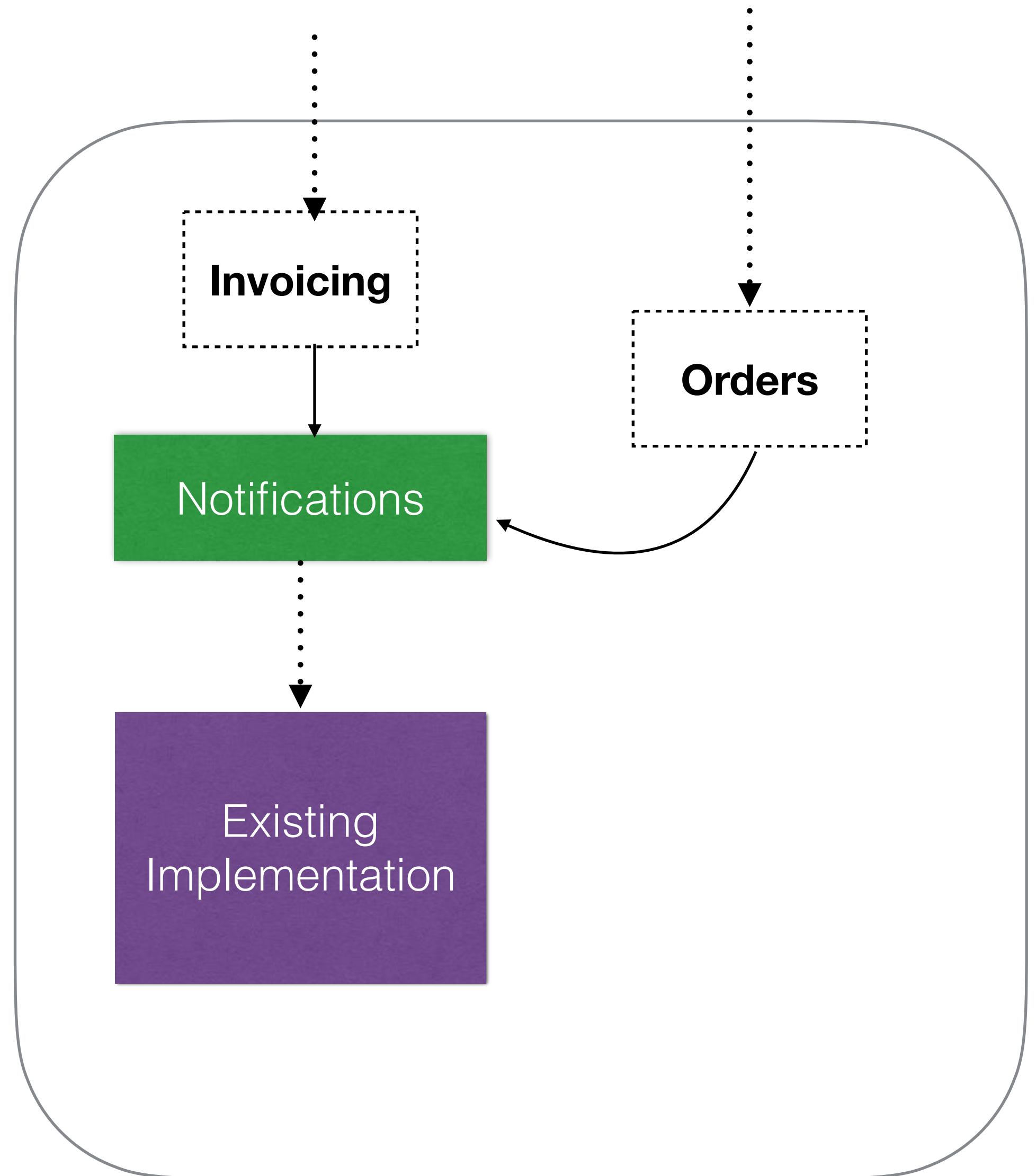


EXAMPLE

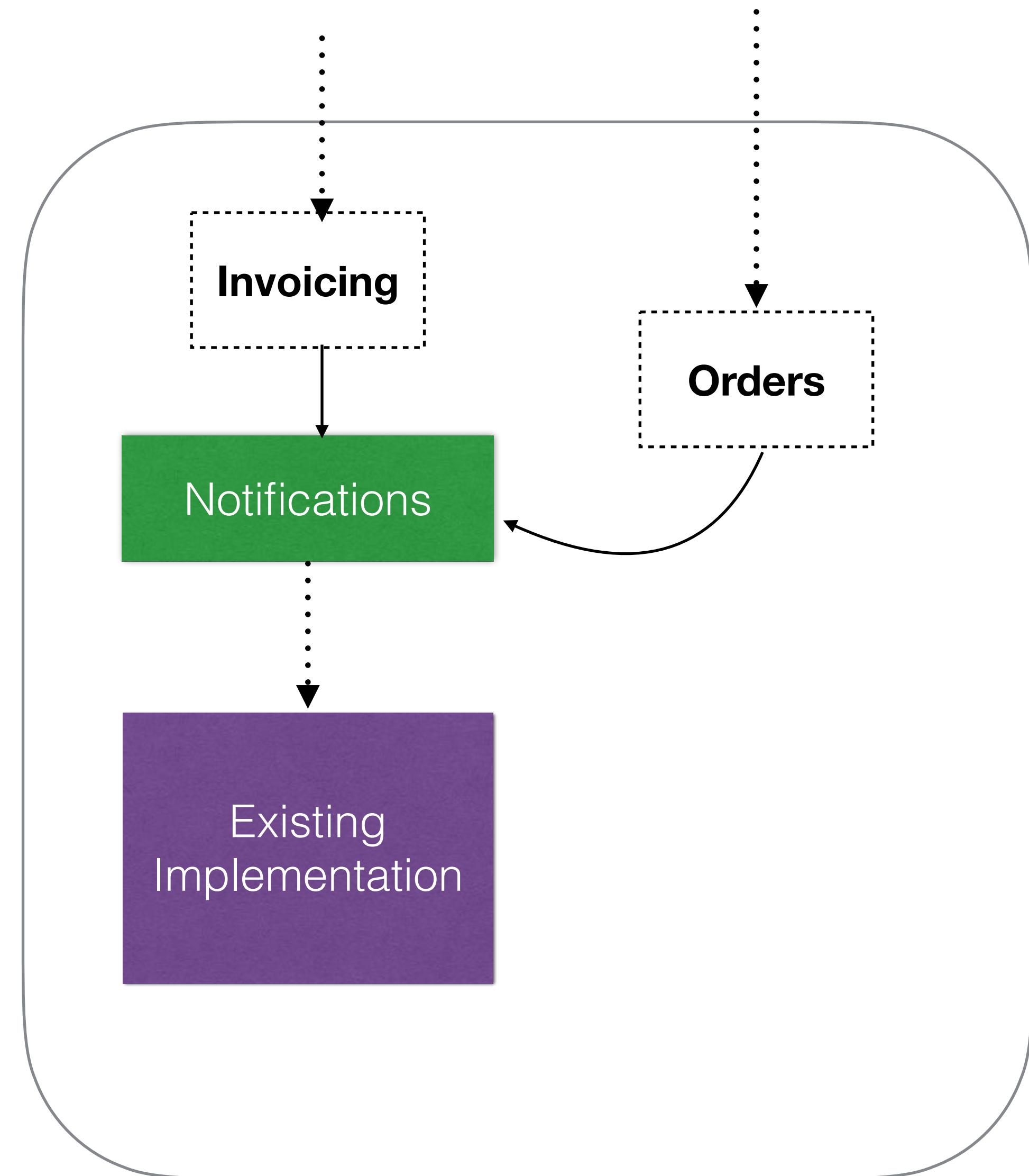


EXAMPLE

1. Create abstraction point



EXAMPLE

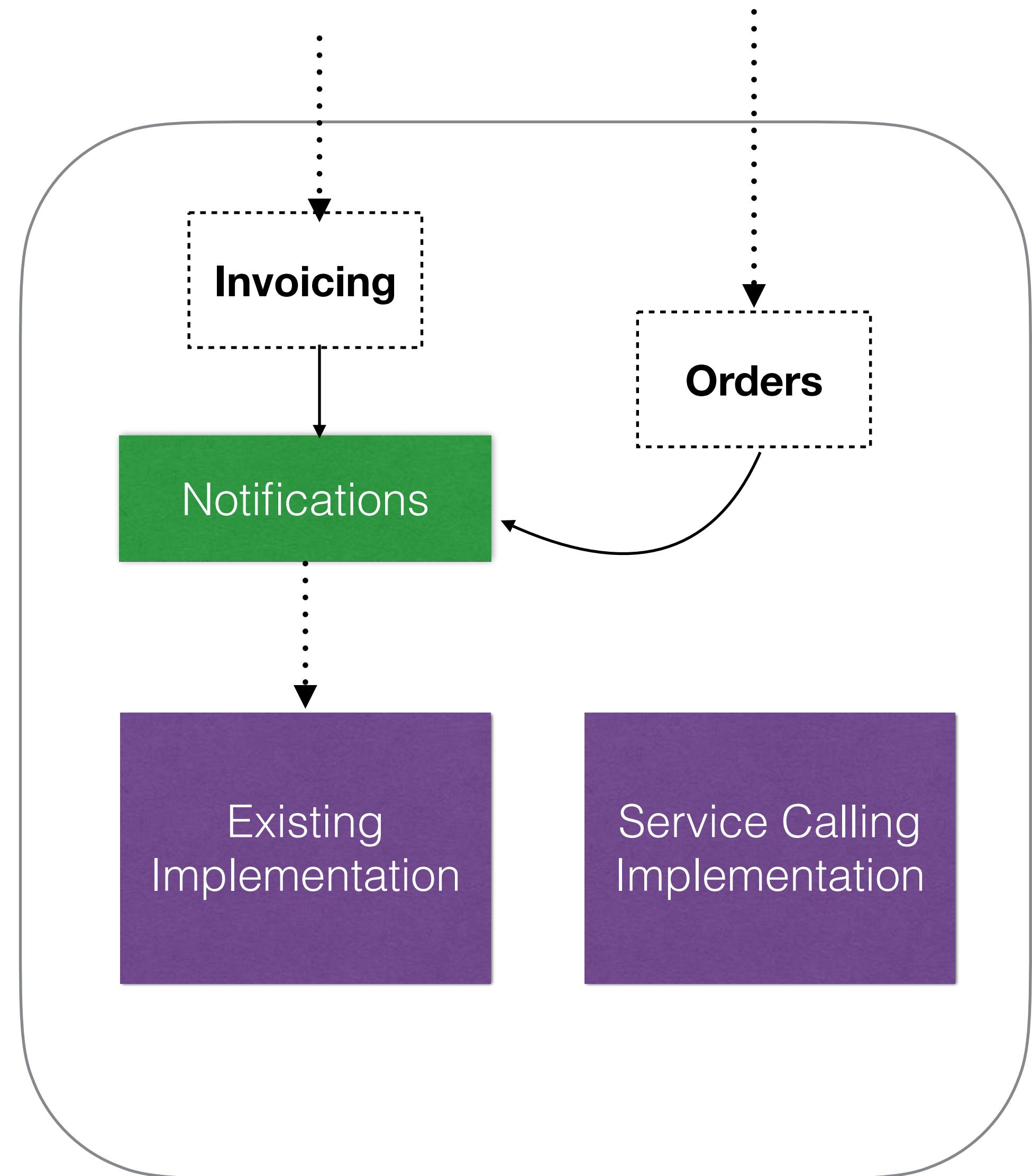


1. Create abstraction point
2. Start work on new service implementation

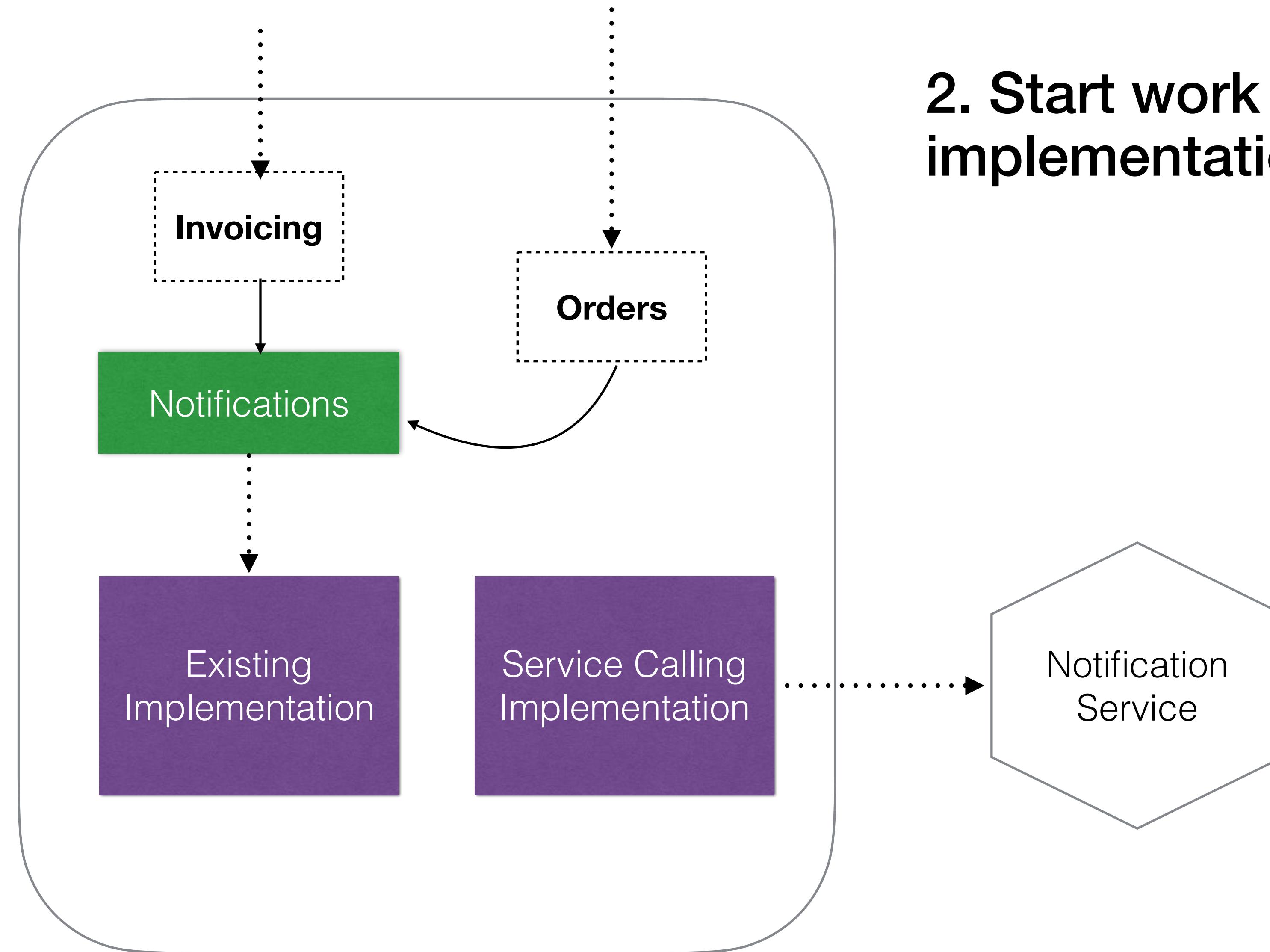
EXAMPLE

1. Create abstraction point

2. Start work on new service implementation



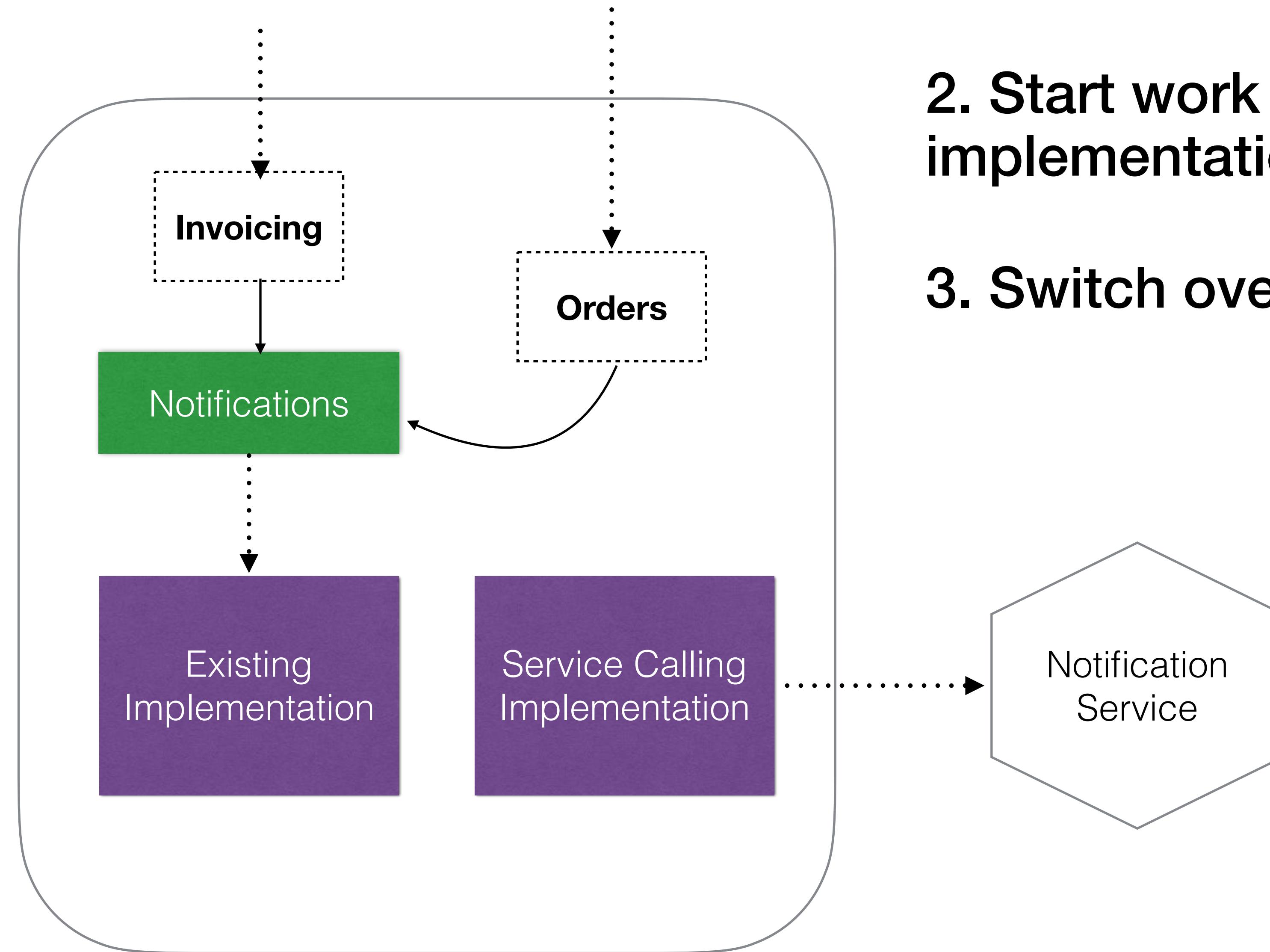
EXAMPLE



1. Create abstraction point

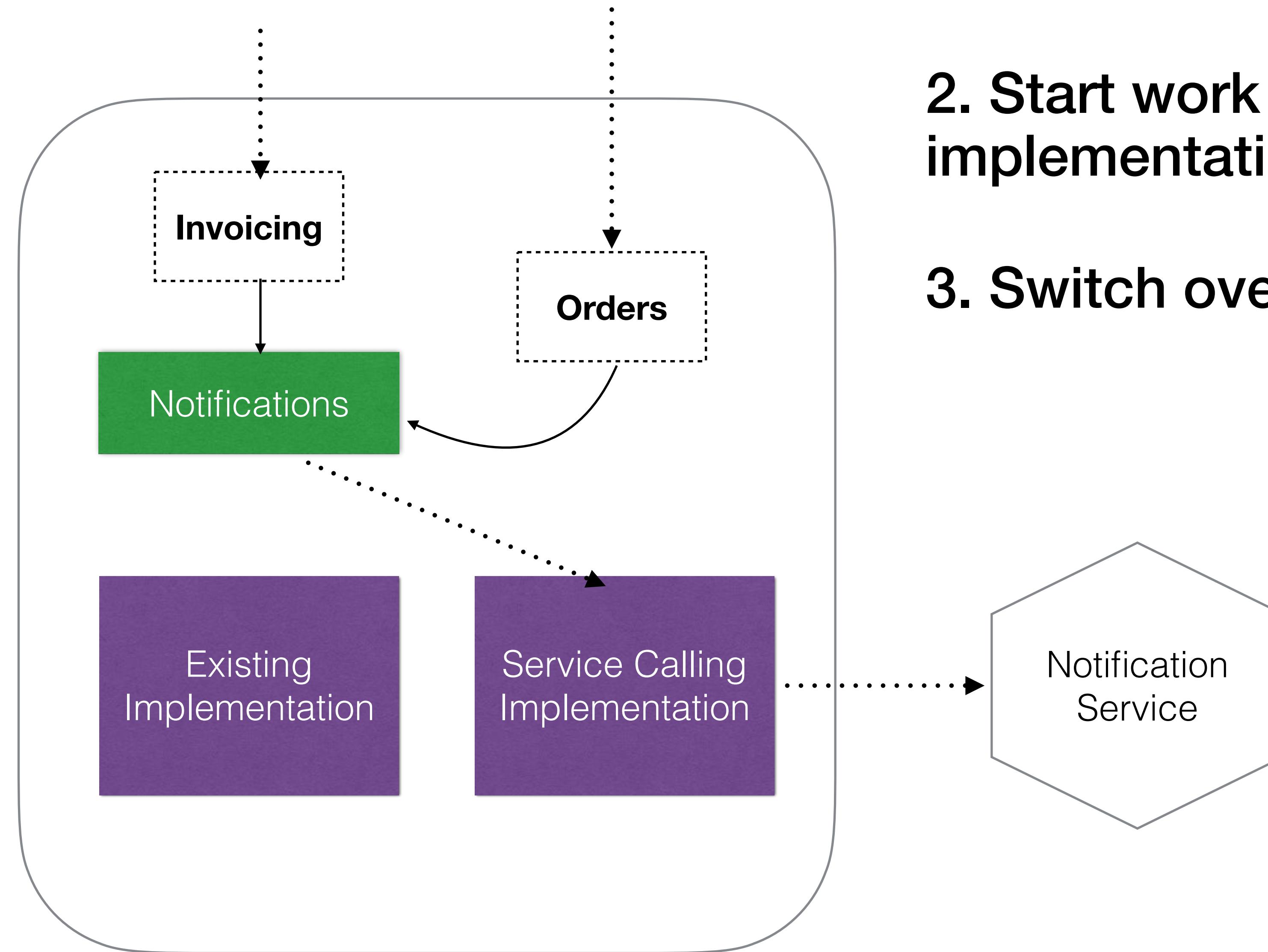
2. Start work on new service implementation

EXAMPLE



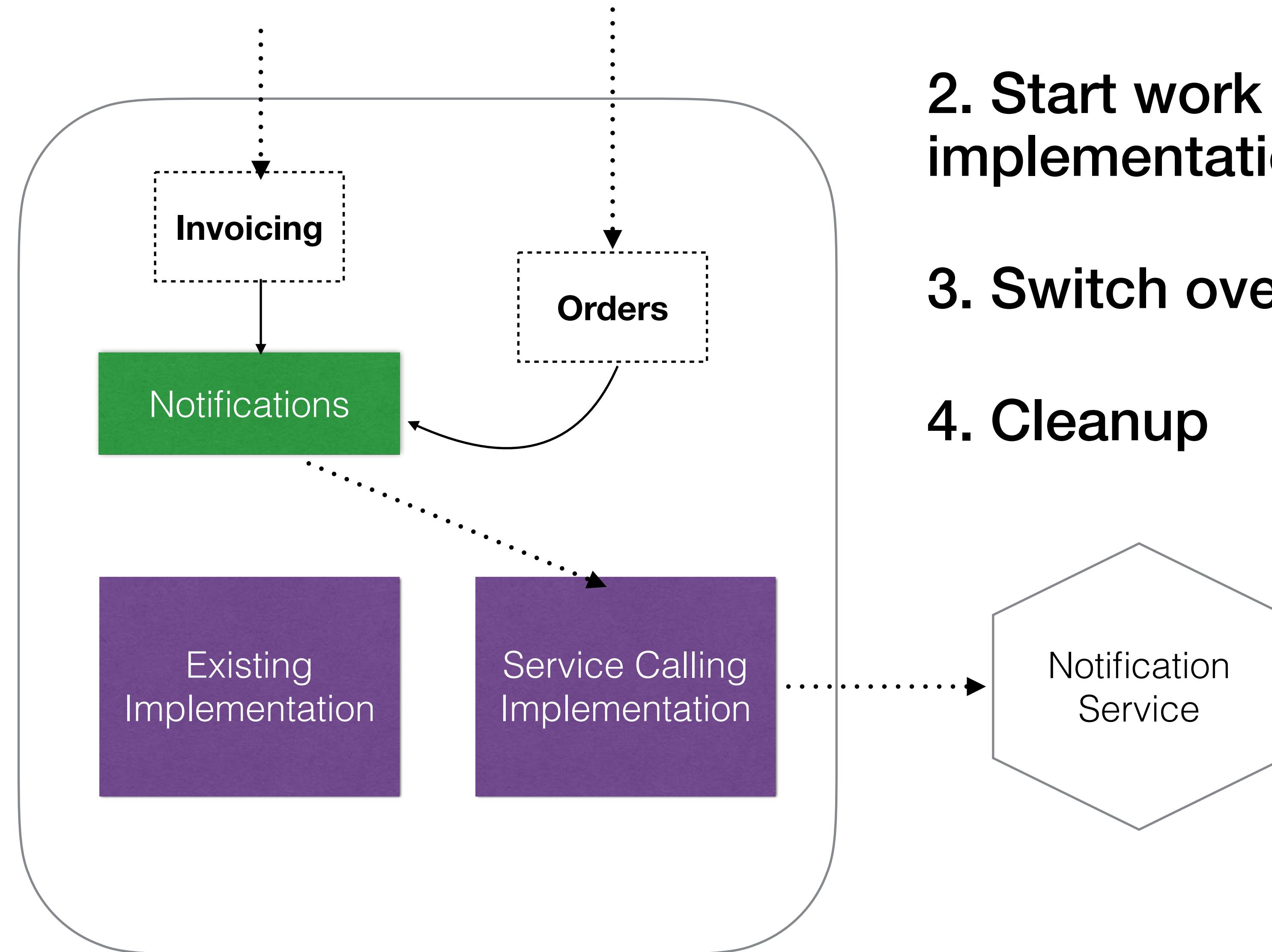
- 1. Create abstraction point**
- 2. Start work on new service implementation**
- 3. Switch over**

EXAMPLE



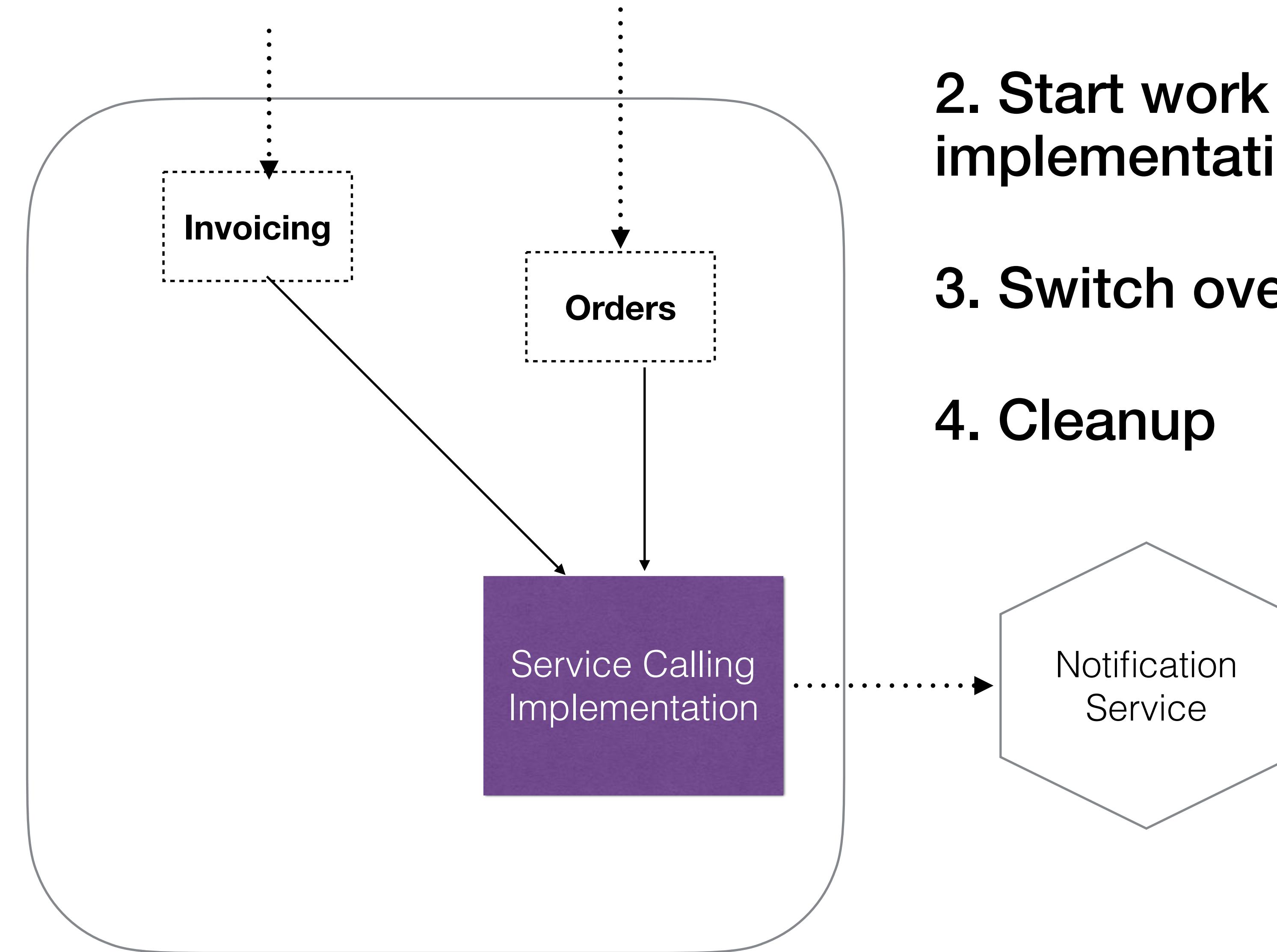
1. Create abstraction point
2. Start work on new service implementation
3. Switch over

EXAMPLE

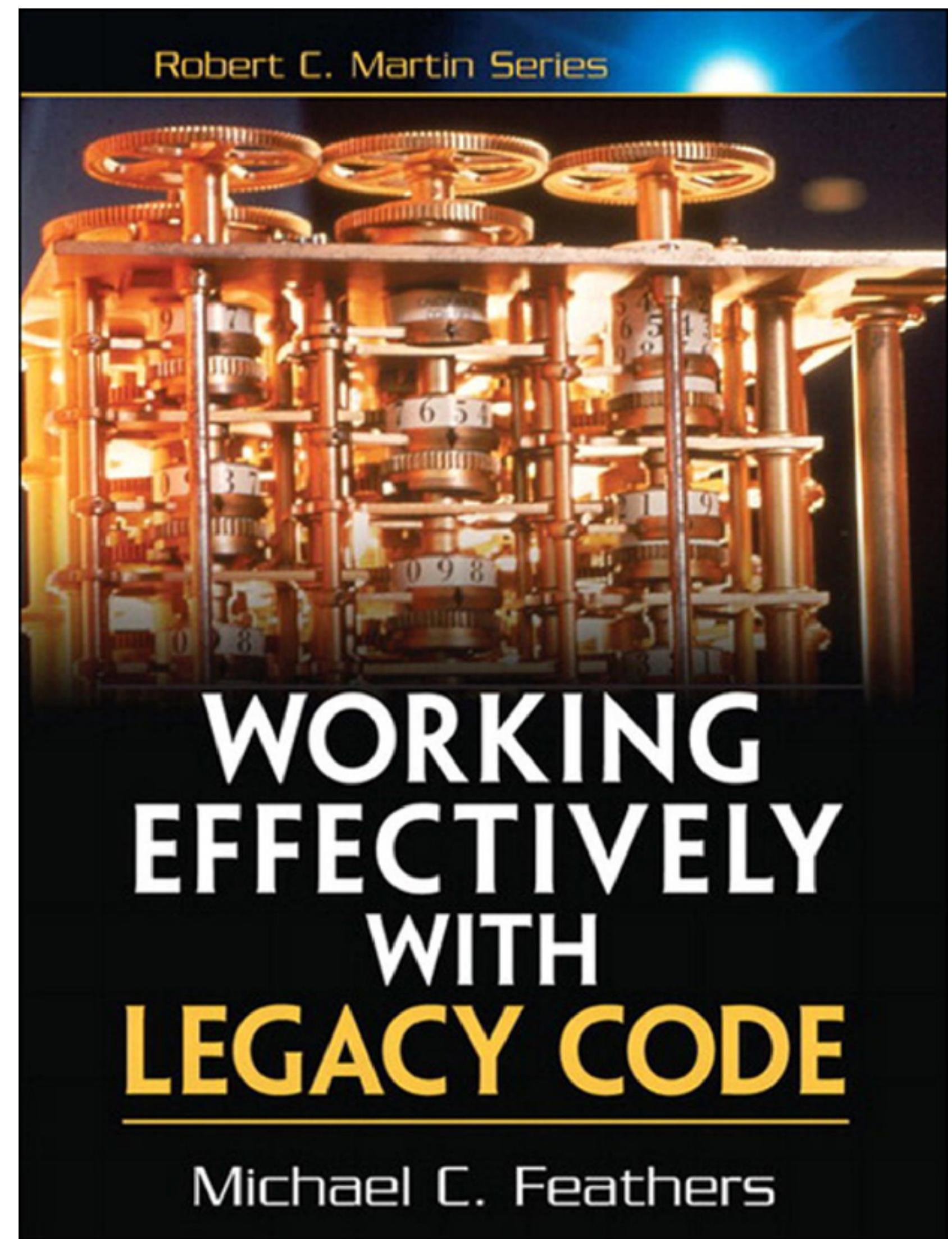


1. Create abstraction point
2. Start work on new service implementation
3. Switch over
4. Cleanup

EXAMPLE



1. Create abstraction point
2. Start work on new service implementation
3. Switch over
4. Cleanup



PATTERN: FEATURE TOGGLES



PATTERN: FEATURE TOGGLES

Feature Toggles (aka Feature Flags)

Feature Toggles (often also referred 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



CONTENTS

[expand](#)

- [A Toggling Tale](#)
- [Categories of toggles](#)
- [Implementation Techniques](#)
- [Toggle Configuration](#)
- [Working with feature-flagged systems](#)

Pete Hodgson

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

Allow for functionality to be toggled off and on

PATTERN: FEATURE TOGGLES

Feature Toggles (aka Feature Flags)

Feature Toggles (often also referred 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



CONTENTS

[expand](#)

- [A Toggling Tale](#)
- [Categories of toggles](#)
- [Implementation Techniques](#)
- [Toggle Configuration](#)
- [Working with feature-flagged systems](#)

Pete Hodgson

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

Allow for functionality to be toggled off and on

Can also be used to switch between alternative implementations

PATTERN: FEATURE TOGGLES

Feature Toggles (aka Feature Flags)

Feature Toggles (often also referred 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

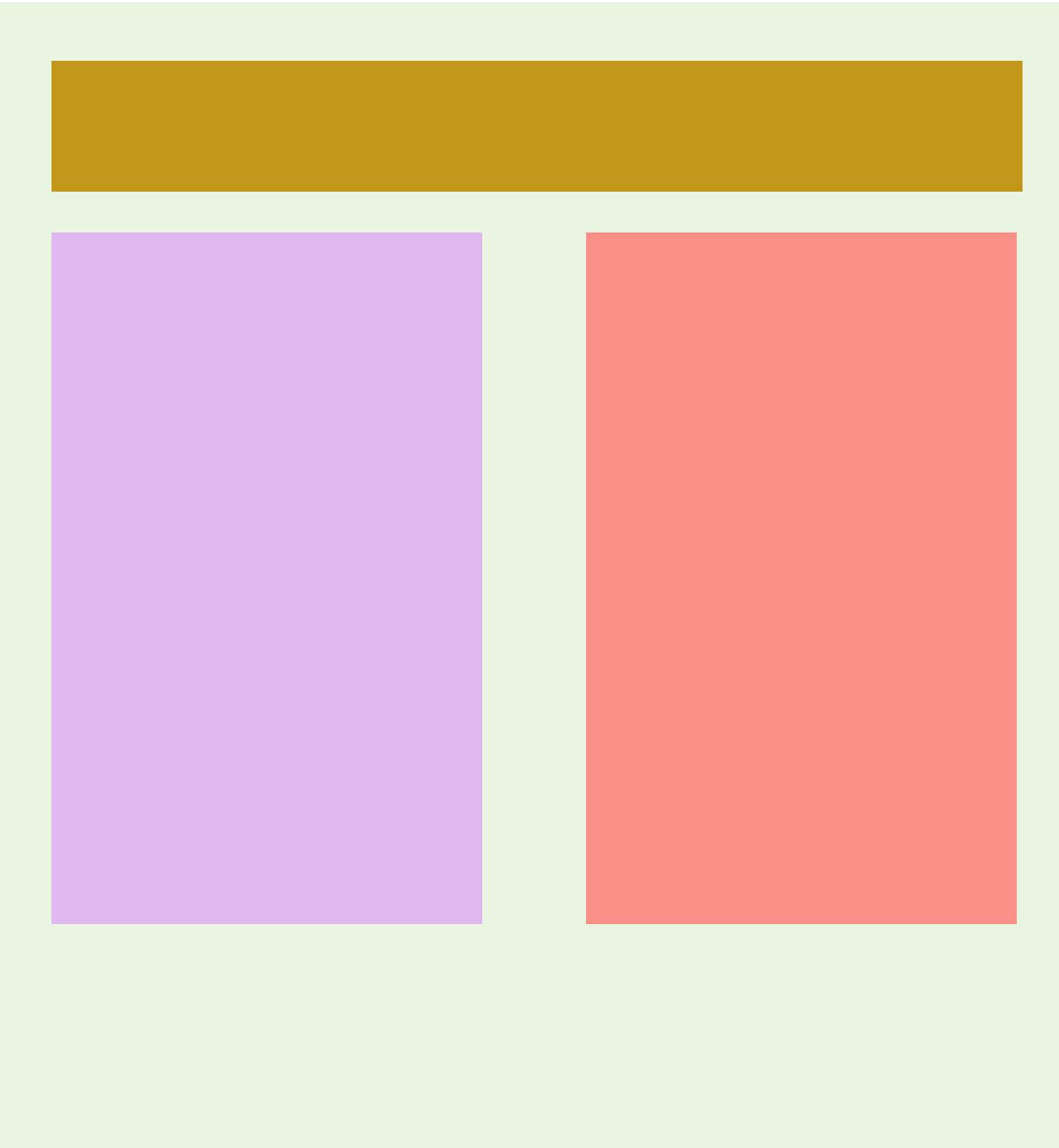
- [A Toggling Tale](#)
- [Categories of toggles](#)
- [Implementation Techniques](#)
- [Toggle Configuration](#)
- [Working with feature-flagged systems](#)

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

FEATURE TOGGLES (CONT)

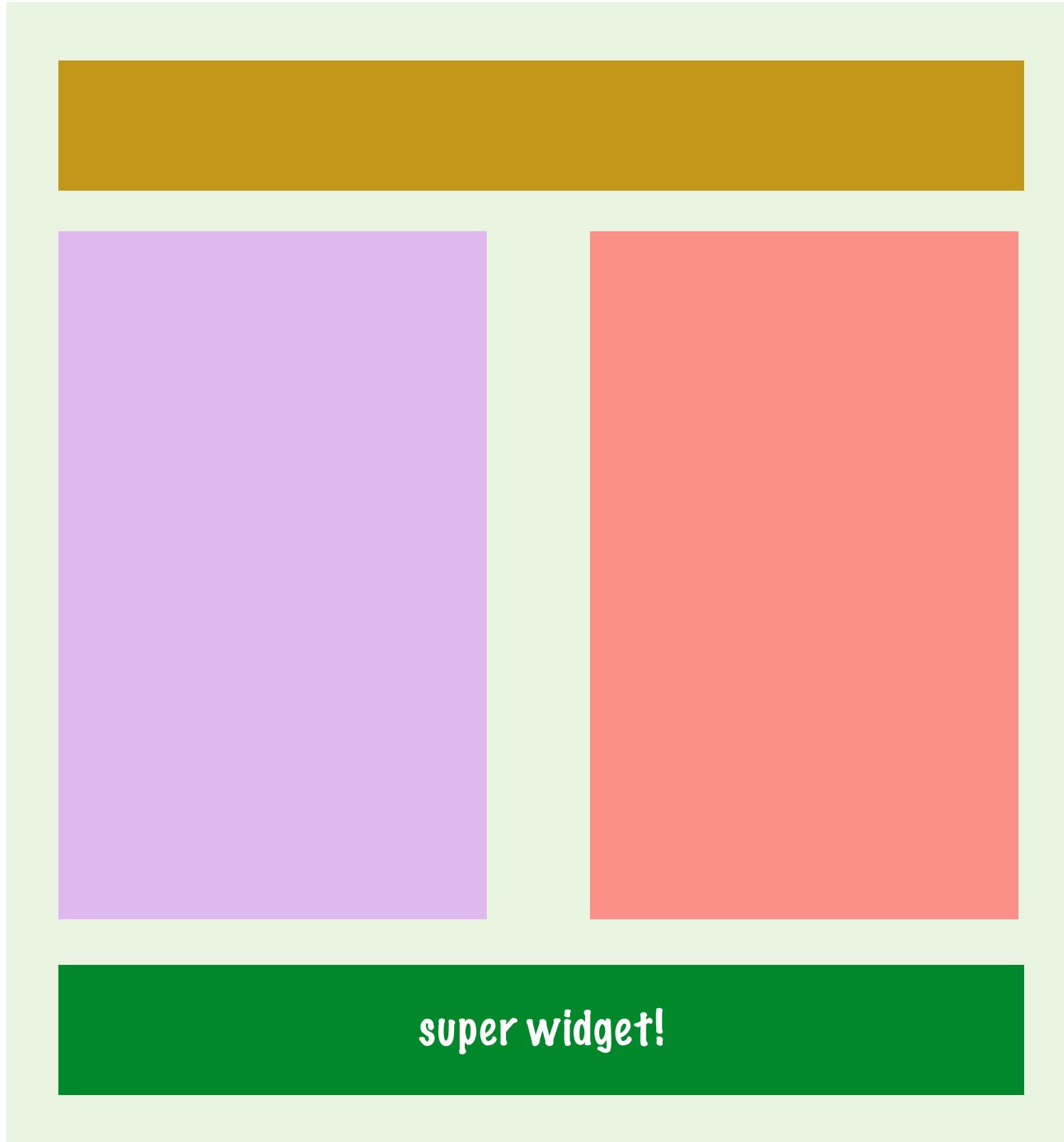


Specified in configuration file

```
...  
super_widget = off  
...
```

Specified in configuration file

FEATURE TOGGLES (CONT)

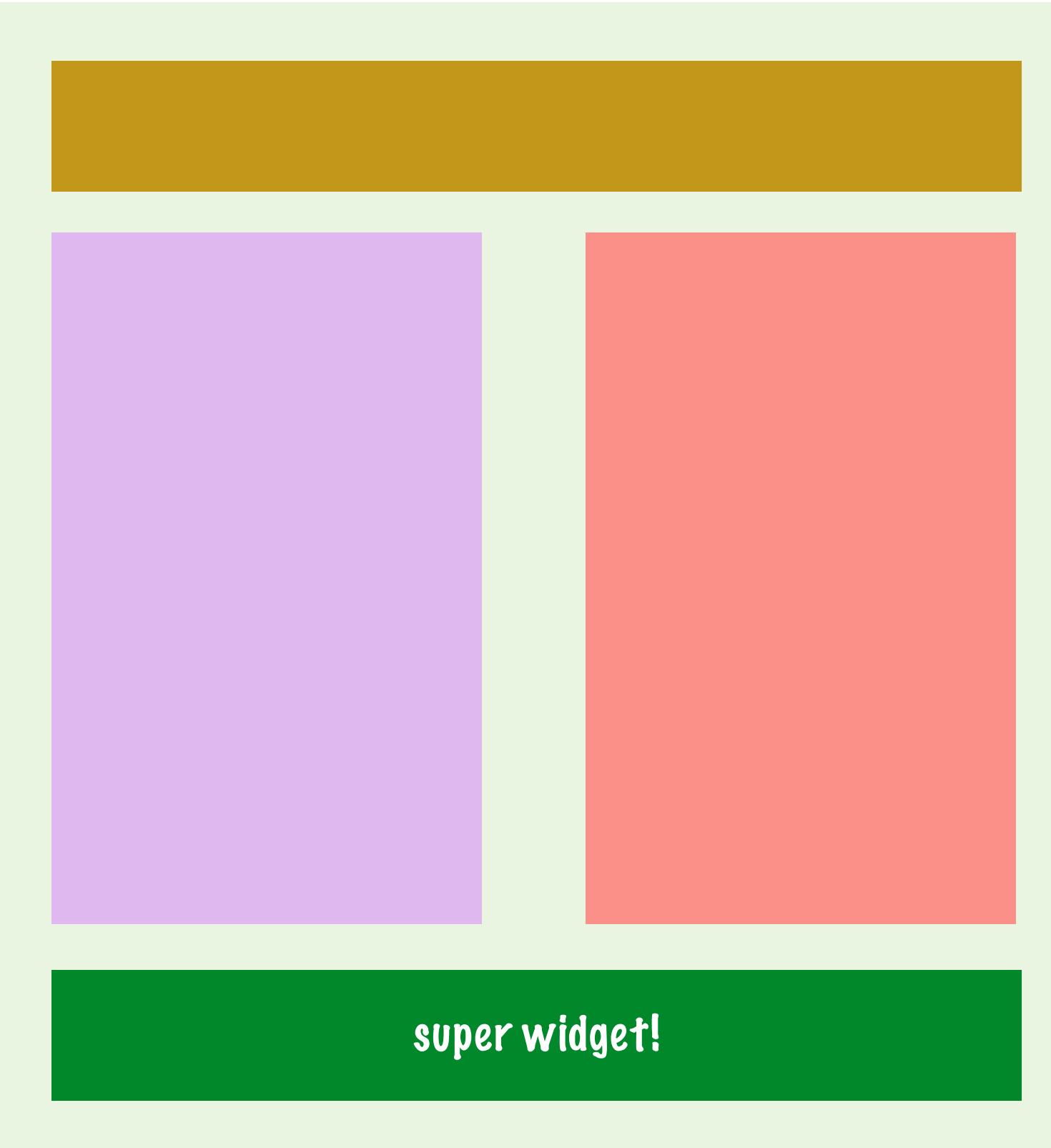


Specified in configuration file

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

Specified in configuration file

FEATURE TOGGLES (CONT)



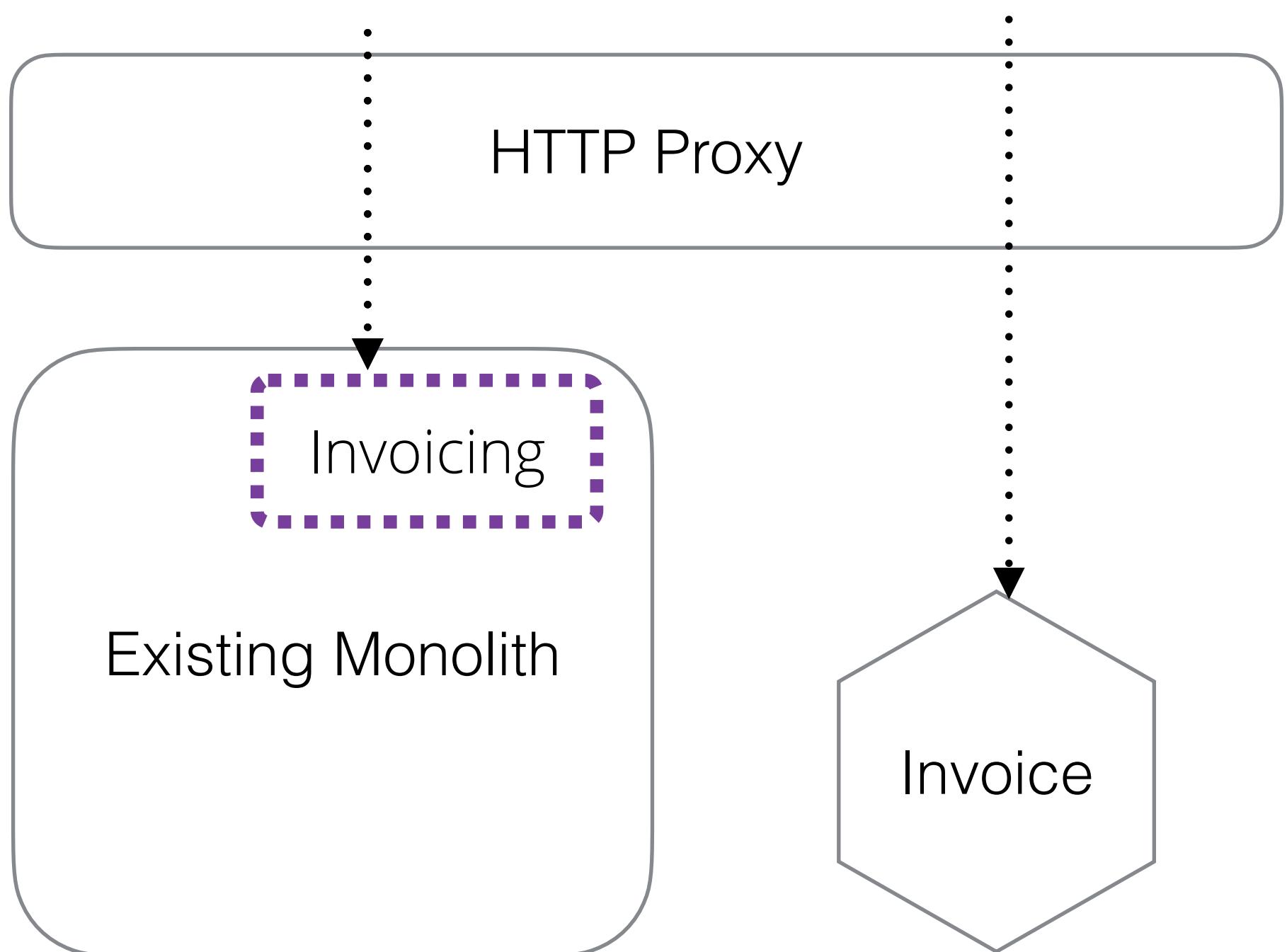
Specified in configuration file

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

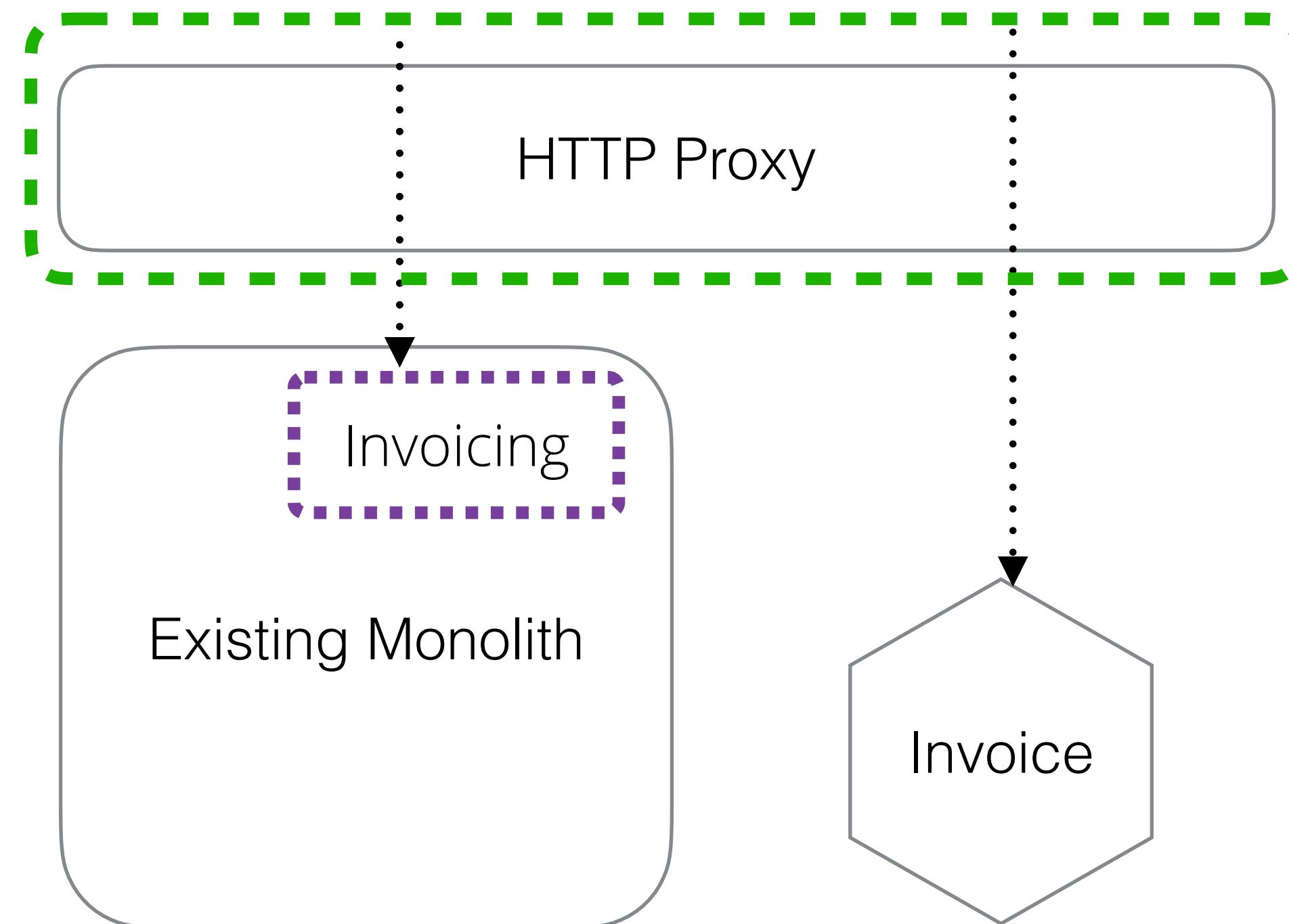
Specified in configuration file

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

WITH THE STRANGLER FIG PATTERN

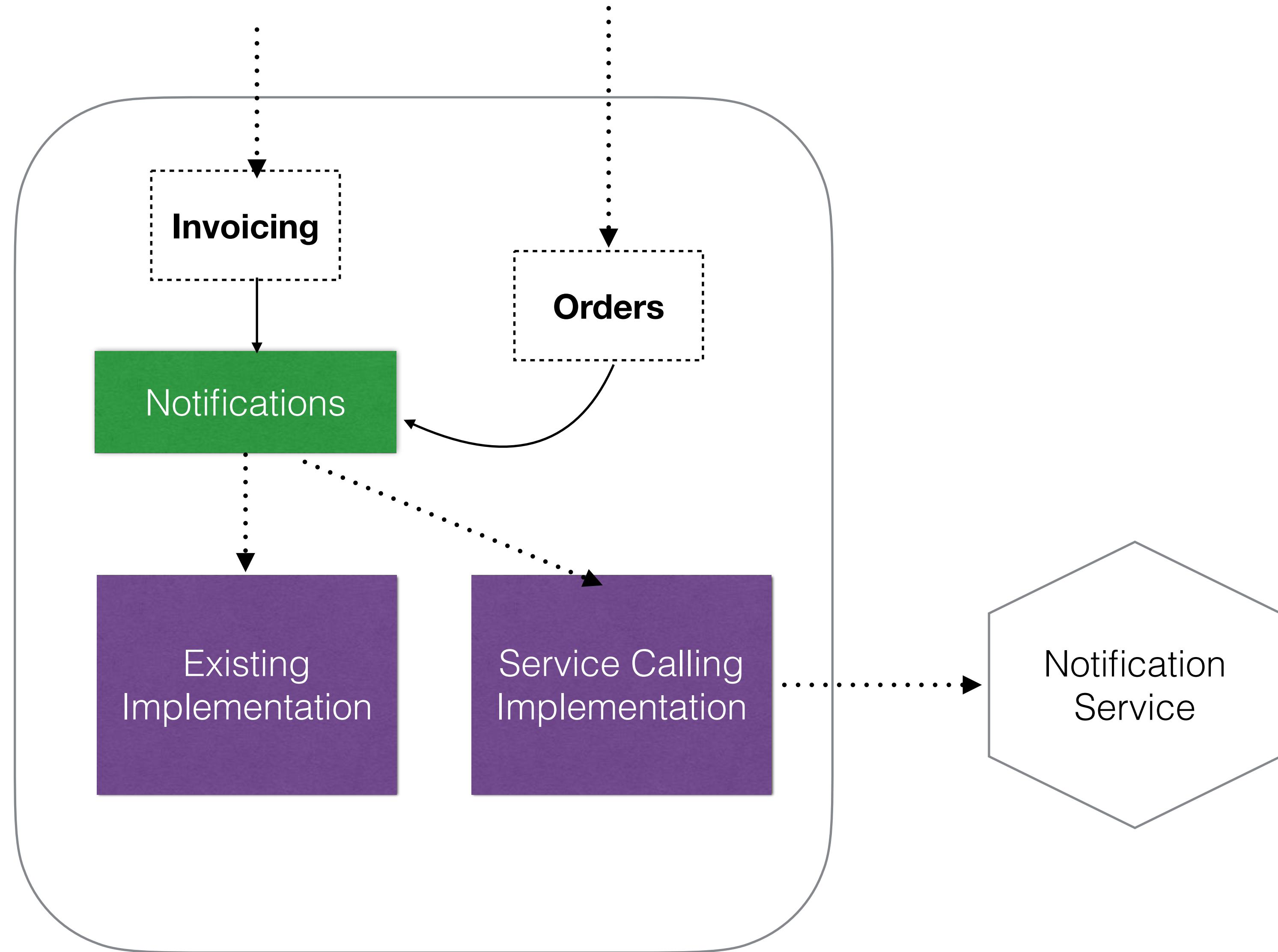


WITH THE STRANGLER FIG PATTERN



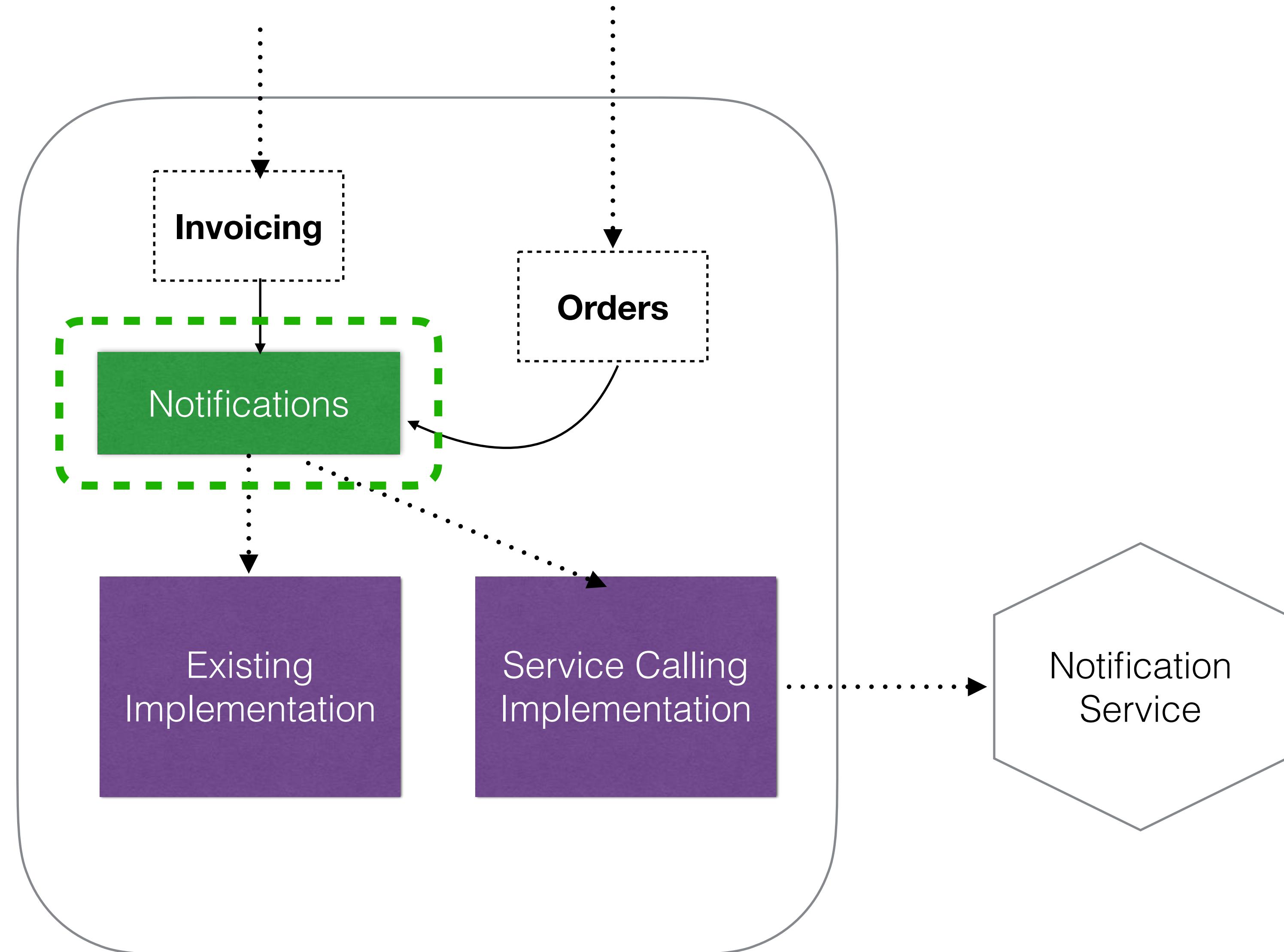
Implement a simple config-based toggle in the HTTP Proxy configuration

WITH BRANCH BY ABSTRACTION



WITH BRANCH BY ABSTRACTION

Use the abstraction to divert calls based on toggle value



PROGRESSIVE DELIVERY

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

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

@samnewman

PROGRESSIVE DELIVERY

JAMES GOVERNOR'S MONKCHIPS

Towards Progressive Delivery

By James Governor | @monkchips | August 6, 2018

[!\[\]\(326384fe05c9766da603f3dc6c206179_img.jpg\)](#) [!\[\]\(eb7b812bf6b21af7a49a6a6c0a04a64a_img.jpg\)](#) [!\[\]\(10550a1d2ec5ff07f7456e8ed268fa5f_img.jpg\)](#) [!\[\]\(d2072014c13227f52b9912d9893d20f3_img.jpg\)](#)



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

Canary

PROGRESSIVE DELIVERY

JAMES GOVERNOR'S MONKCHIPS

Towards Progressive Delivery

By James Governor | @monkchips | August 6, 2018

[!\[\]\(808369bed0da04037d1199bb0683502e_img.jpg\)](#) [!\[\]\(ce76d7fc49fdd79e5f8a40bde0c63a5c_img.jpg\)](#) [!\[\]\(bdc4892ef64ef5b158579ff0dc49ce56_img.jpg\)](#) [!\[\]\(32b92149f1ea9881c0dab2ec72af42d0_img.jpg\)](#)



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

Canary

A/B Testing

PROGRESSIVE DELIVERY

JAMES GOVERNOR'S MONKCHIPS

Towards Progressive Delivery

By James Governor | @monkchips | August 6, 2018

[!\[\]\(92f97183a67cf6dc8568744a405d025b_img.jpg\)](#) [!\[\]\(f2e6b026b3aa46772951df791ae6b455_img.jpg\)](#) [!\[\]\(f86bea020728bf8ed3ee074aa8922e78_img.jpg\)](#) [!\[\]\(166cb1299f4f2646ffa2e0cd45a90a9e_img.jpg\)](#)



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

Canary

A/B Testing

Parallel Runs

PROGRESSIVE DELIVERY

JAMES GOVERNOR'S MONKCHIPS

Towards Progressive Delivery

By James Governor | @monkchips | August 6, 2018

[!\[\]\(9b0c3cd9b0b01cd007ada8f9f5543cba_img.jpg\)](#) [!\[\]\(9dc50d513276b990b472f78ddaee825f_img.jpg\)](#) [!\[\]\(cf8d77b4c49e062a573a6edd51cdfea0_img.jpg\)](#) [!\[\]\(f82503fd949ef3b49729f709b5af44c1_img.jpg\)](#)



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

Canary

A/B Testing

Parallel Runs

Blue/Green

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

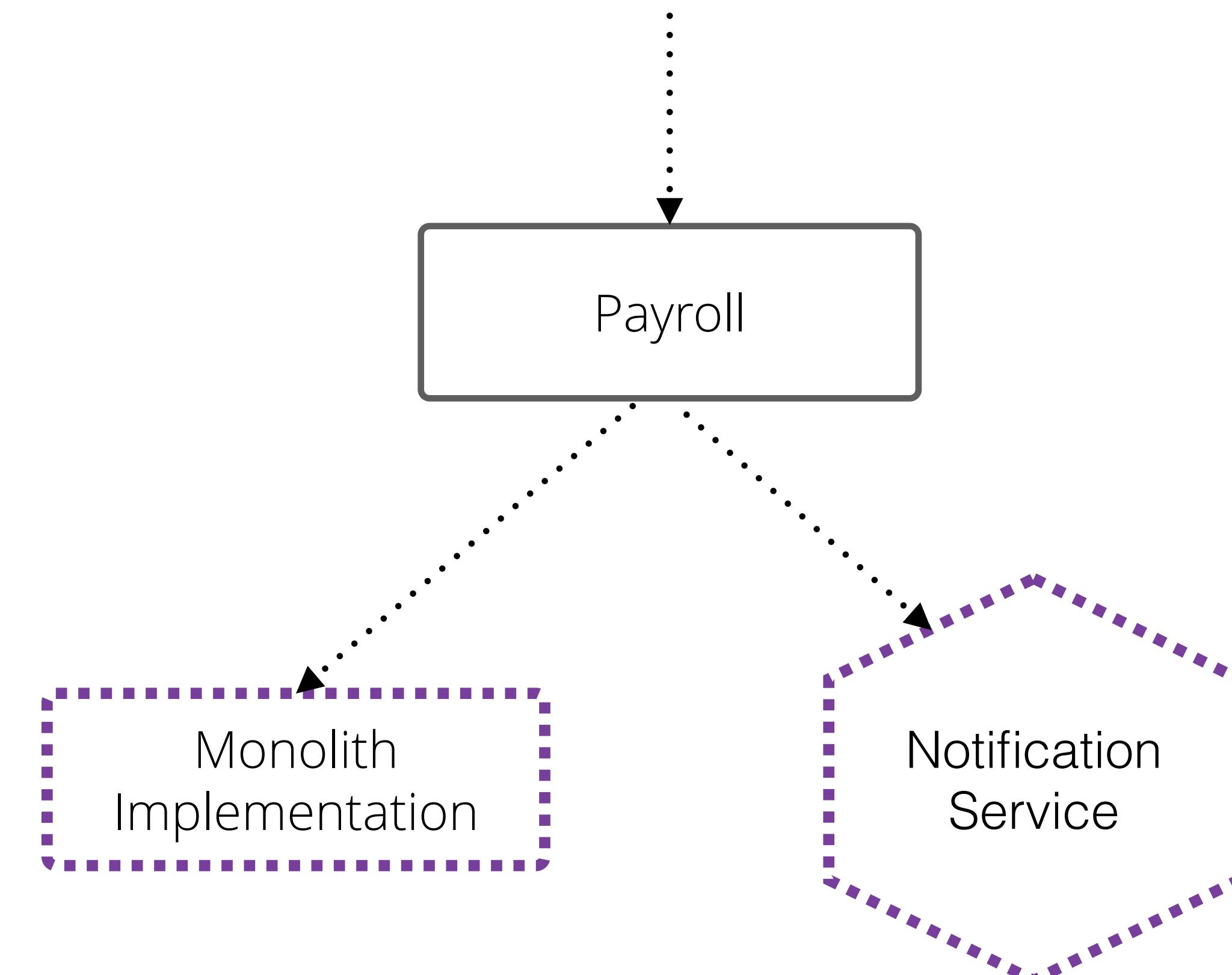
Dark Launching

PATTERN: PARALLEL RUN

Rather than executing one implementation or another, execute both

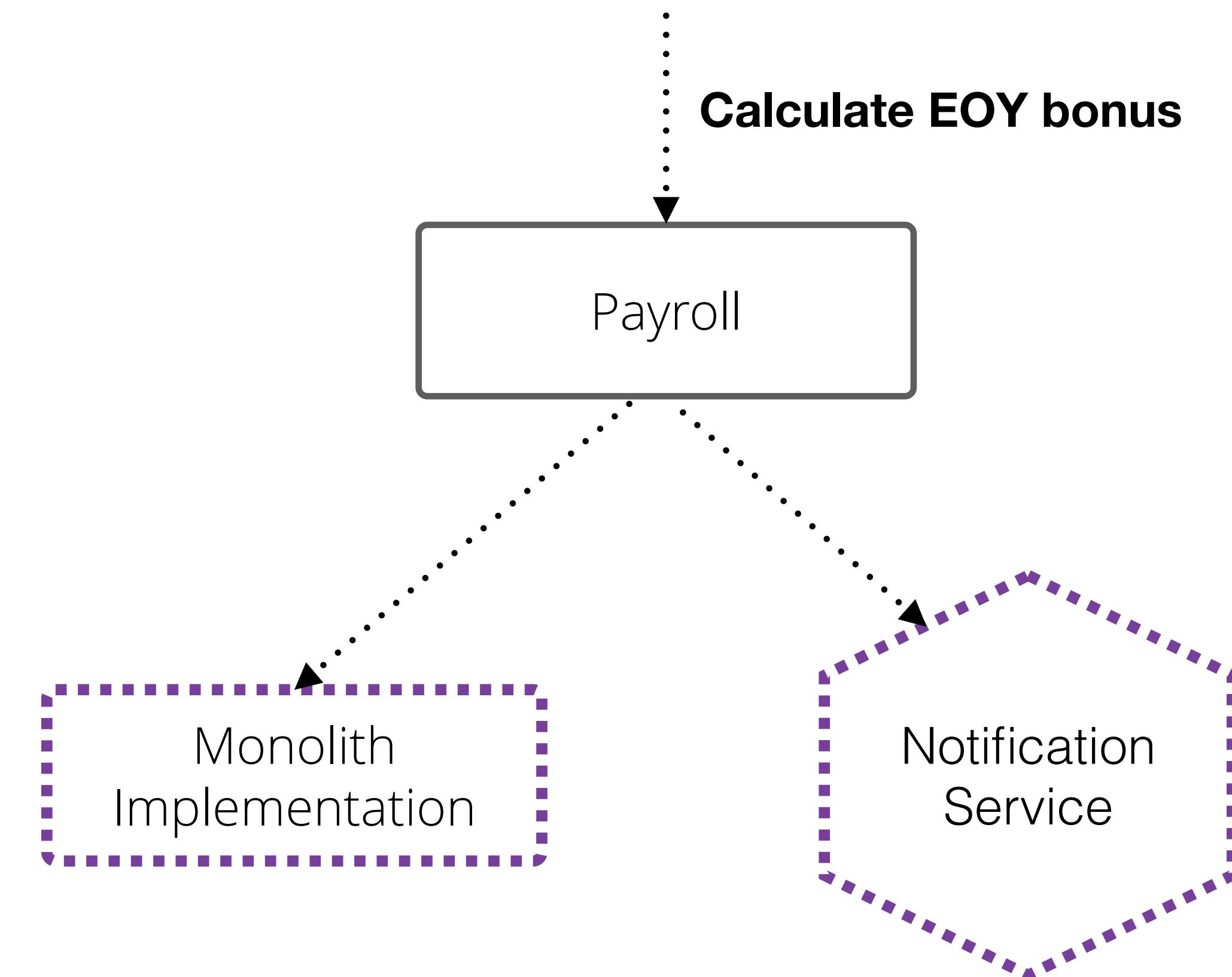
PATTERN: PARALLEL RUN

Rather than executing one implementation or another, execute both



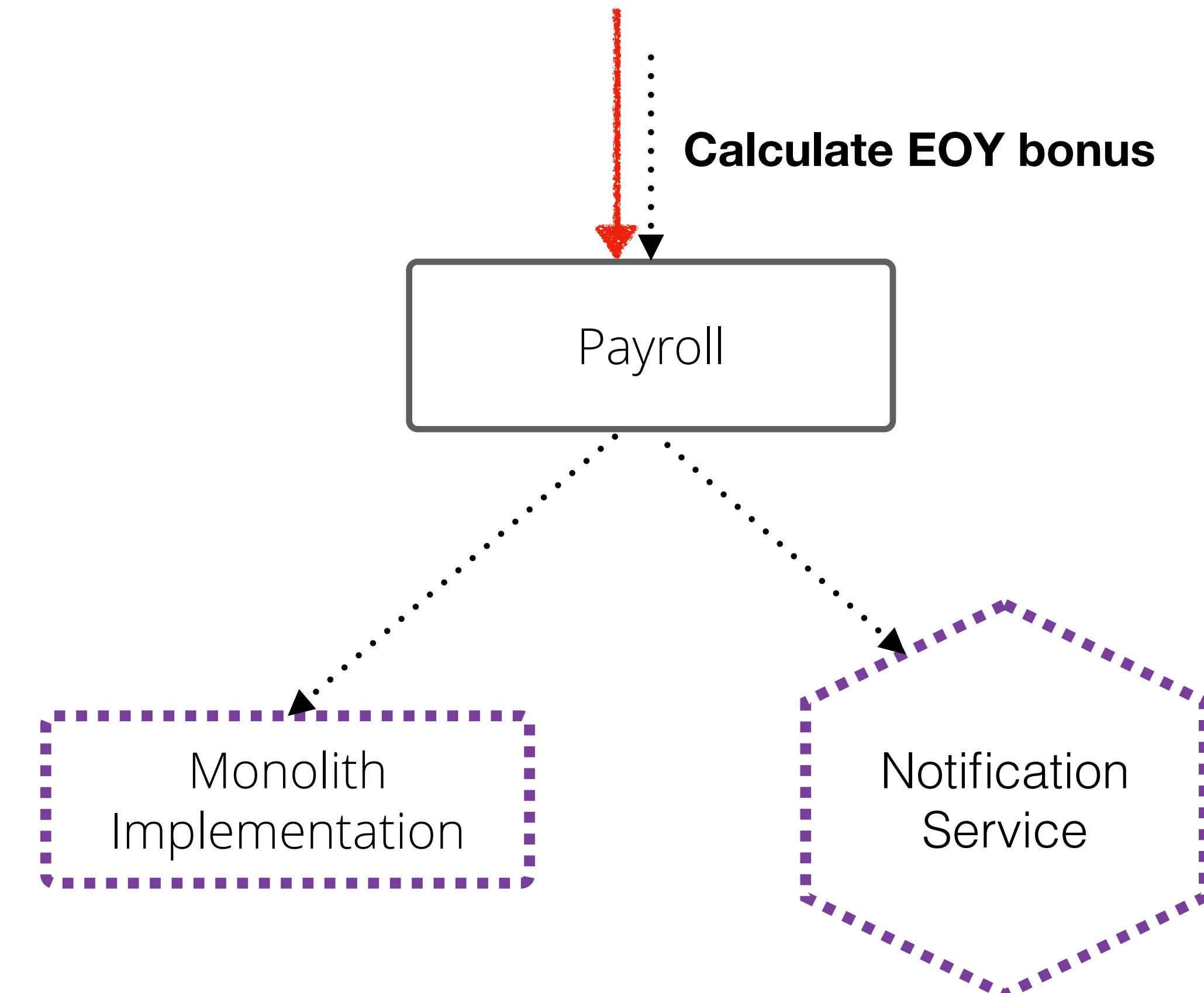
PATTERN: PARALLEL RUN

Rather than executing one implementation or another, execute both



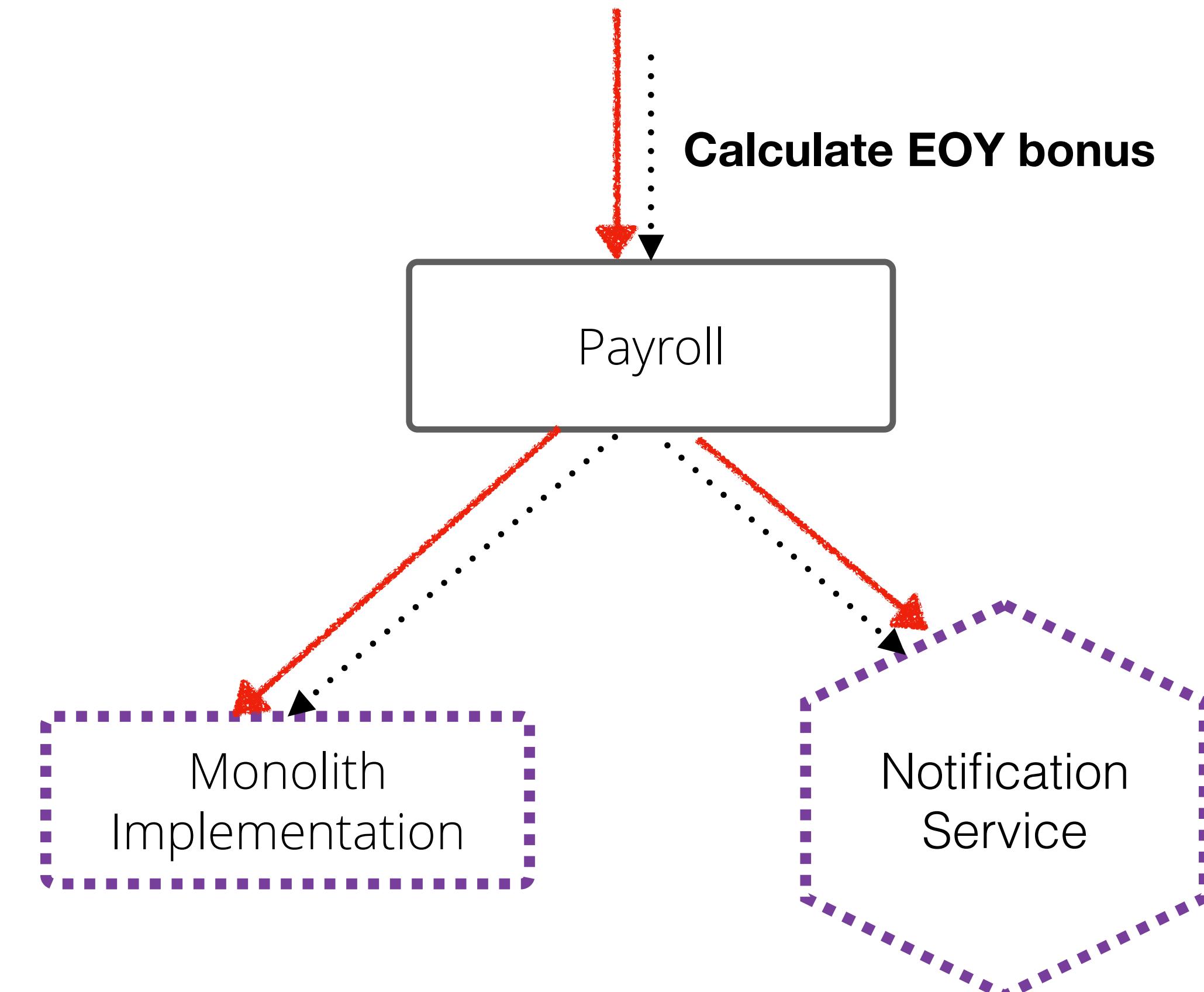
PATTERN: PARALLEL RUN

Rather than executing one implementation or another, execute both



PATTERN: PARALLEL RUN

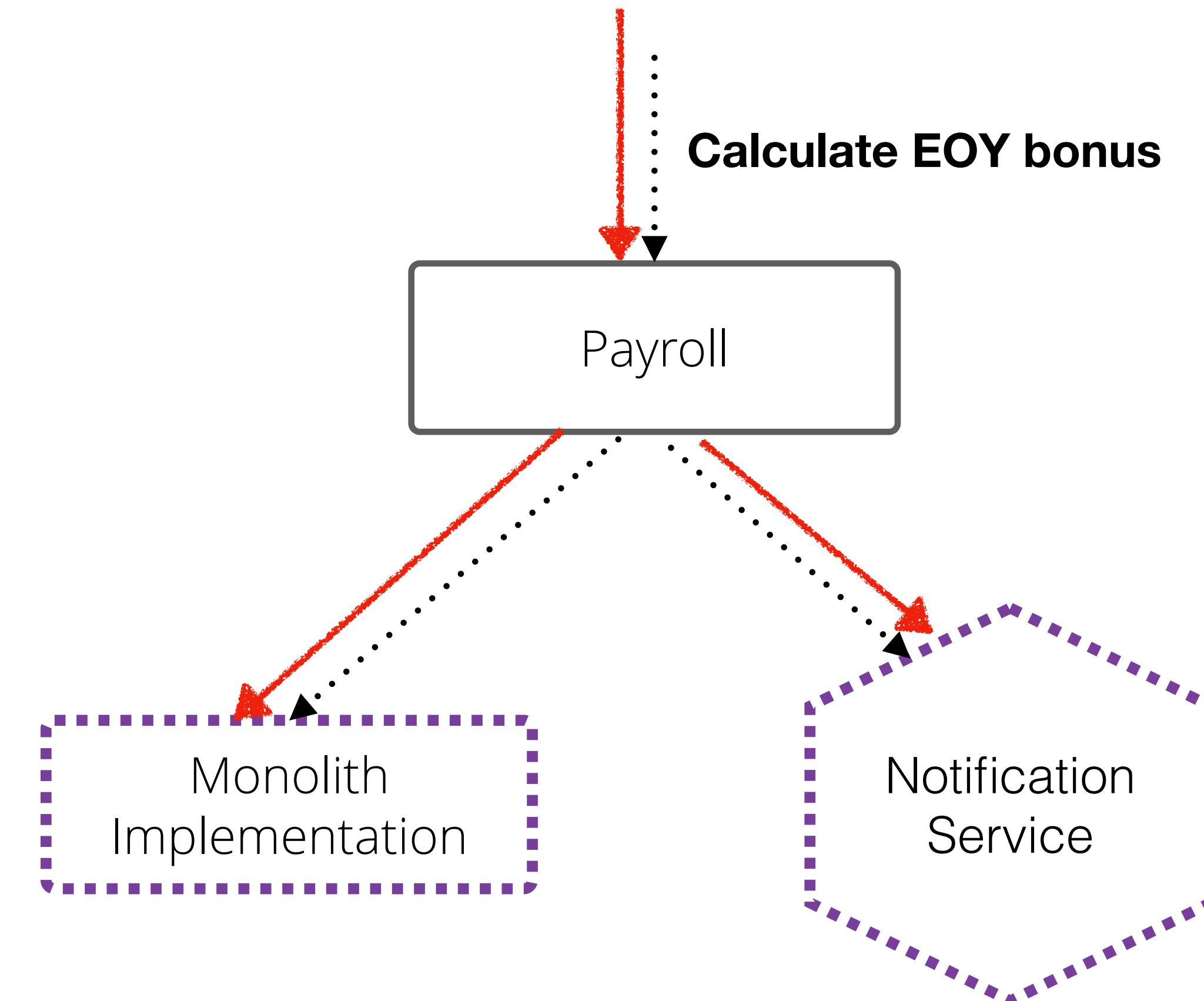
Rather than executing one implementation or another, execute both



PATTERN: PARALLEL RUN

Rather than executing one implementation or another, execute both

Compare the results, and ensure they are the same

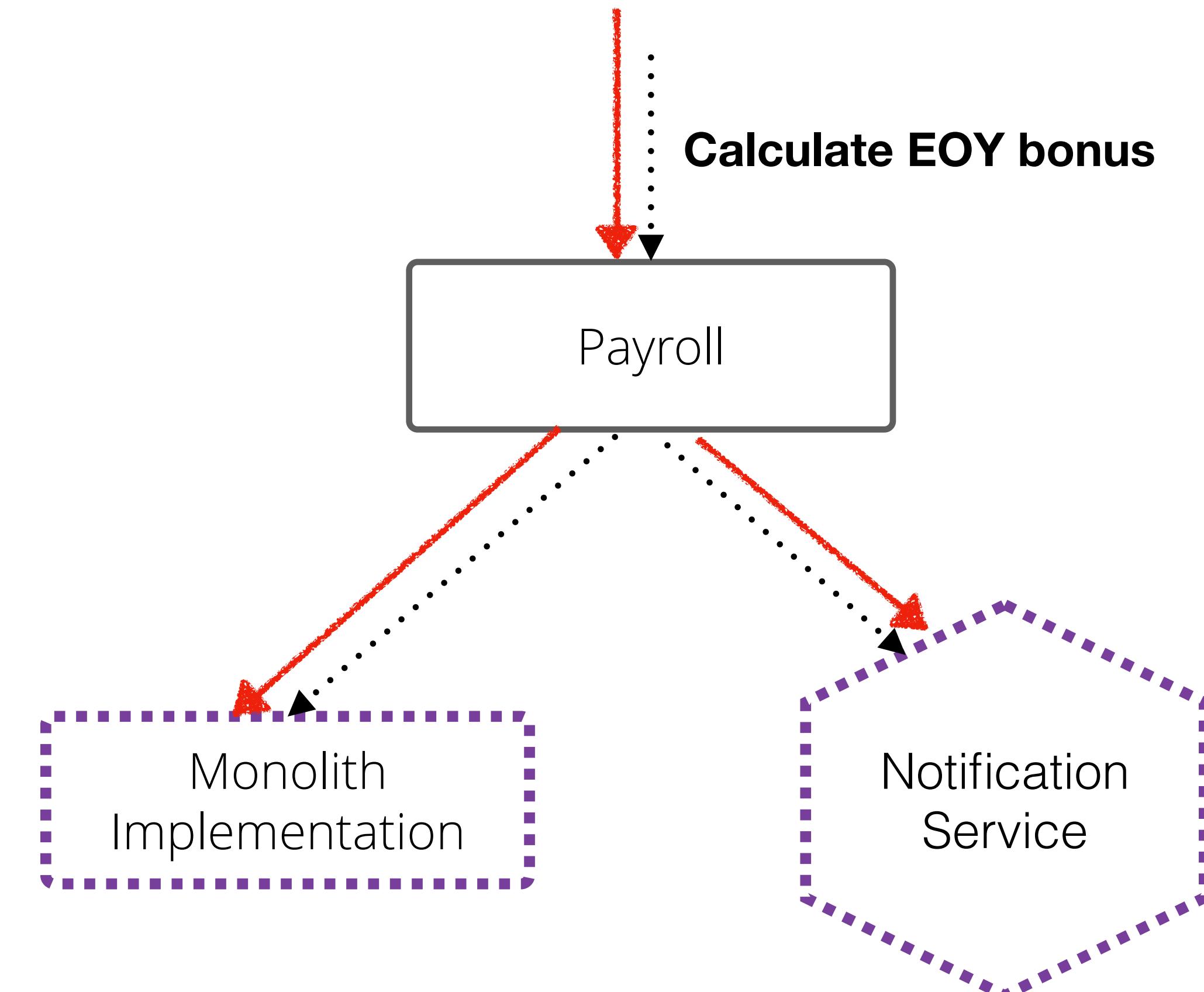


PATTERN: PARALLEL RUN

Rather than executing one implementation or another, execute both

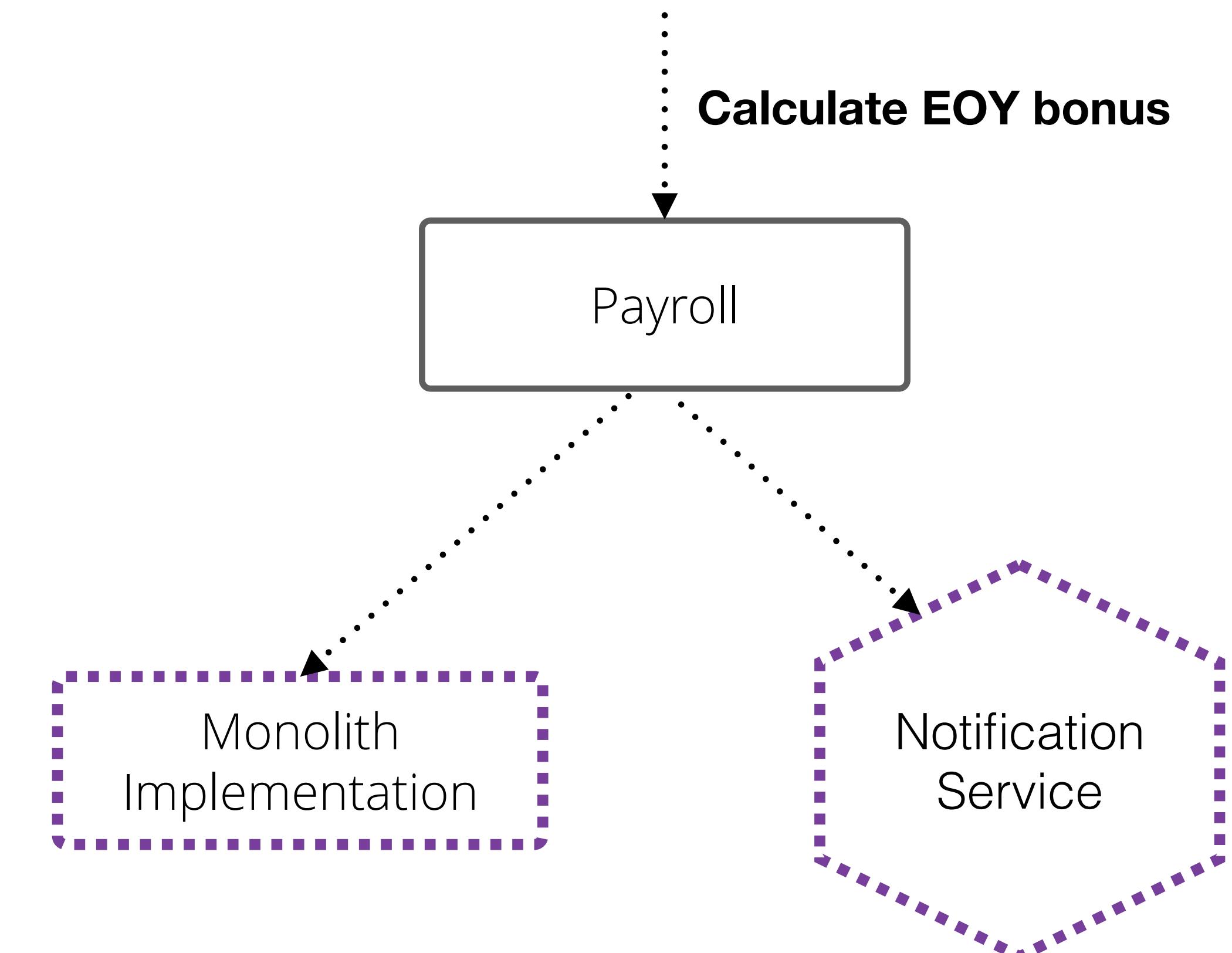
Compare the results, and ensure they are the same

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



SOURCE OF TRUTH

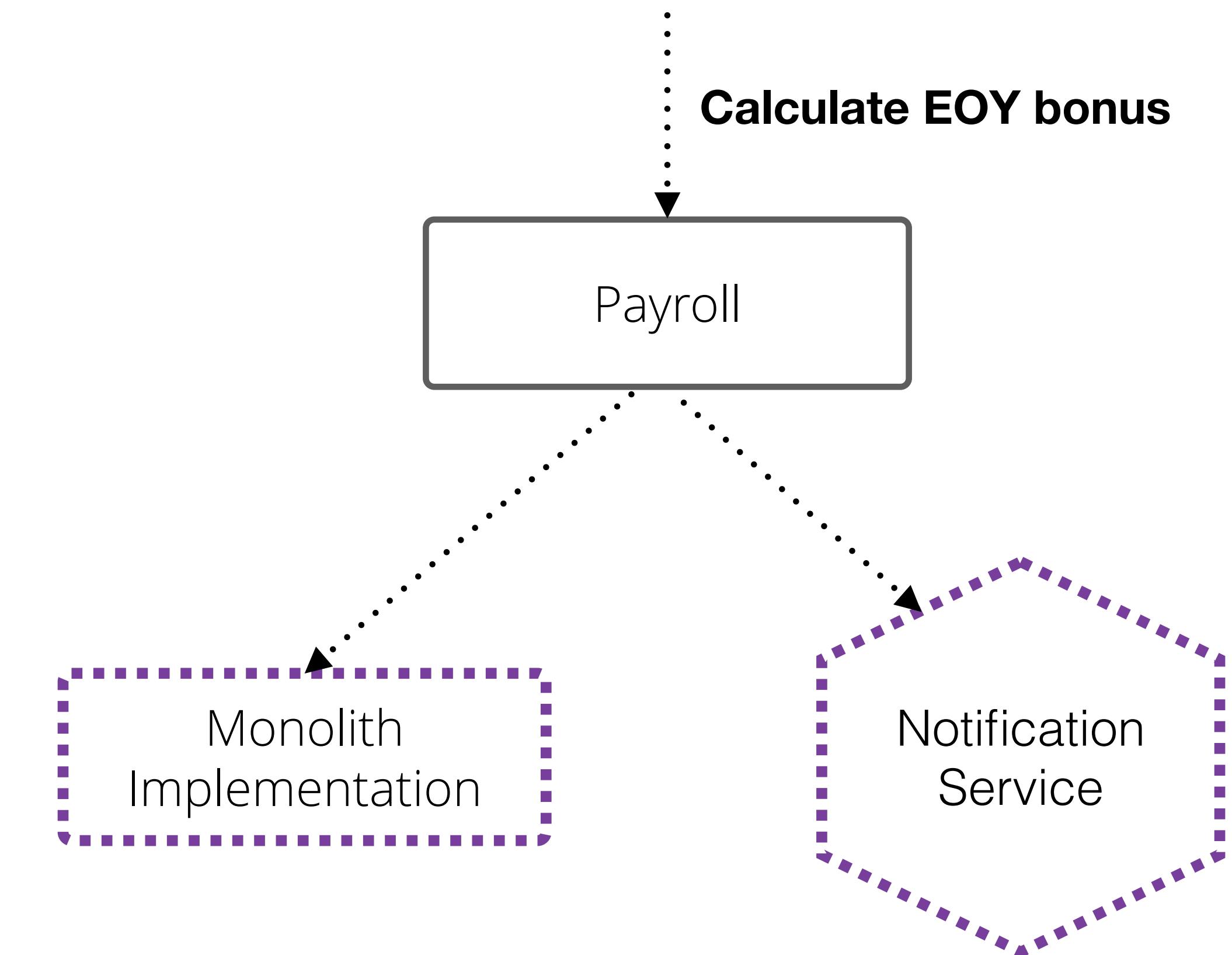
You'd only treat one implementation as the source of truth



SOURCE OF TRUTH

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

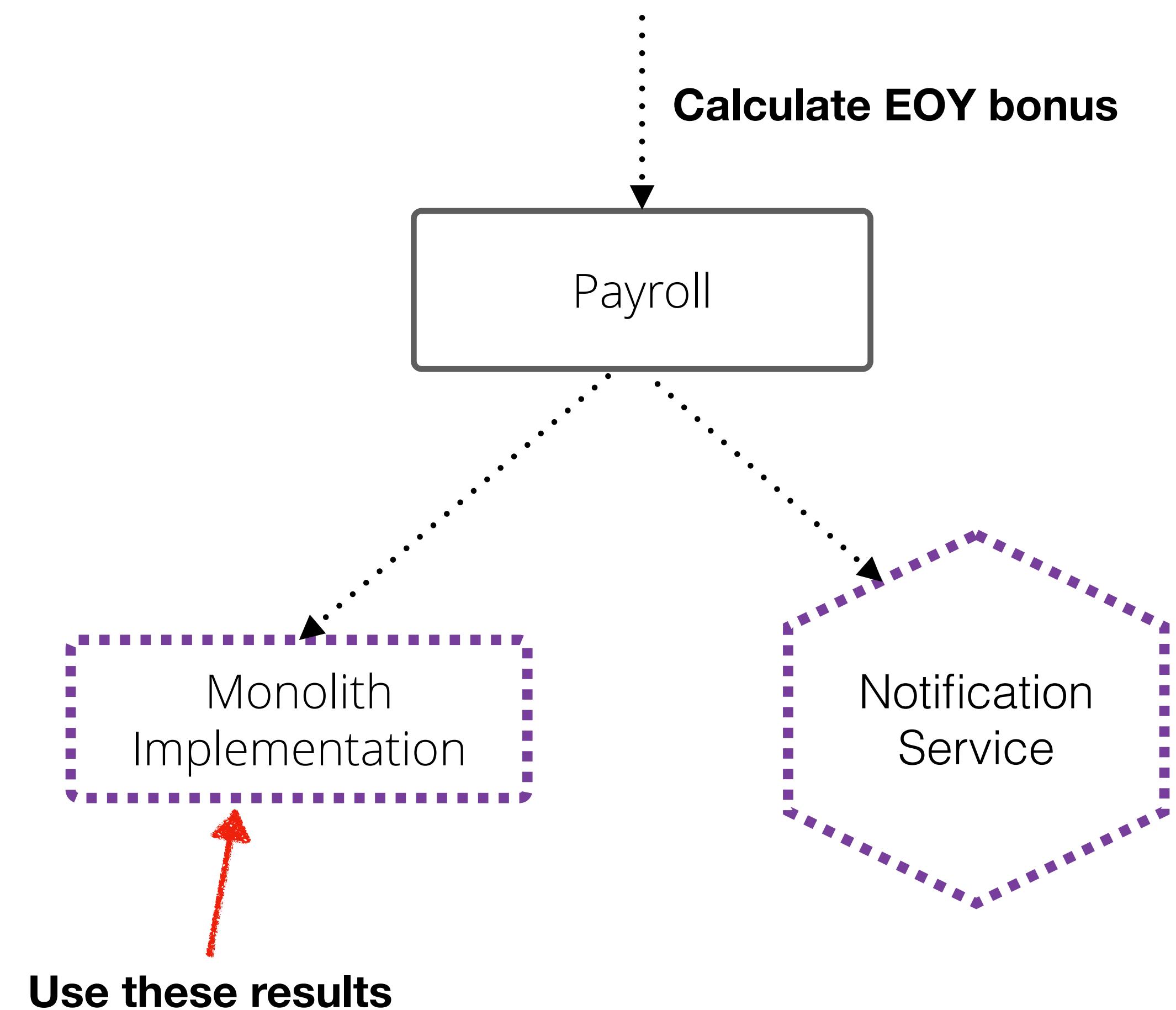
Initially, this would be the existing implementation



SOURCE OF TRUTH

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

Initially, this would be the existing implementation

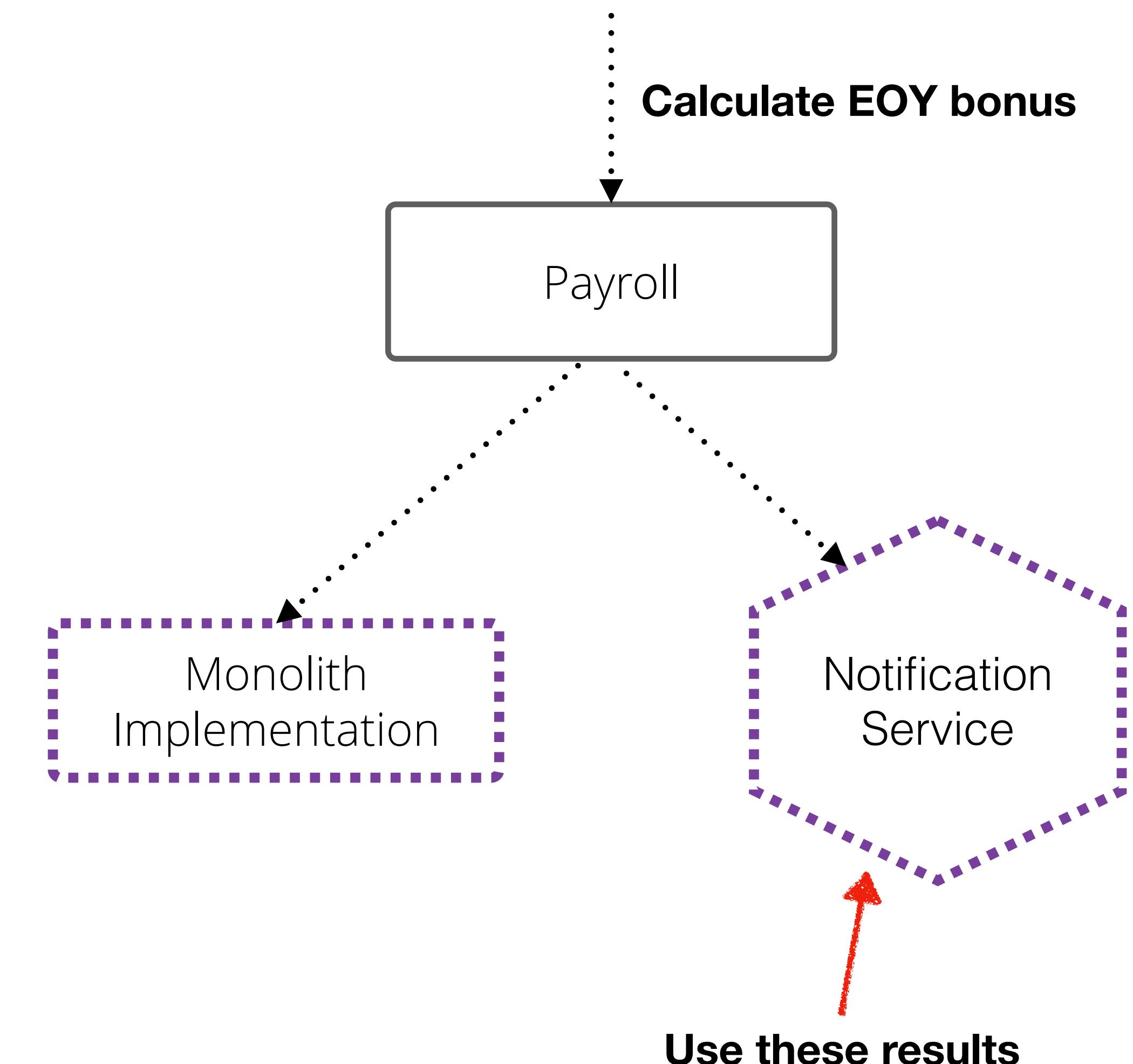


SOURCE OF TRUTH

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)
- [sophtchev/scientist](#) (Kotlin / Java)
- [junkpiano/scientist](#) (Swift)

<https://github.com/github/scientist>

AGENDA

Introduction

Migration Approach

Application Refactoring

UI Decomposition

AGENDA

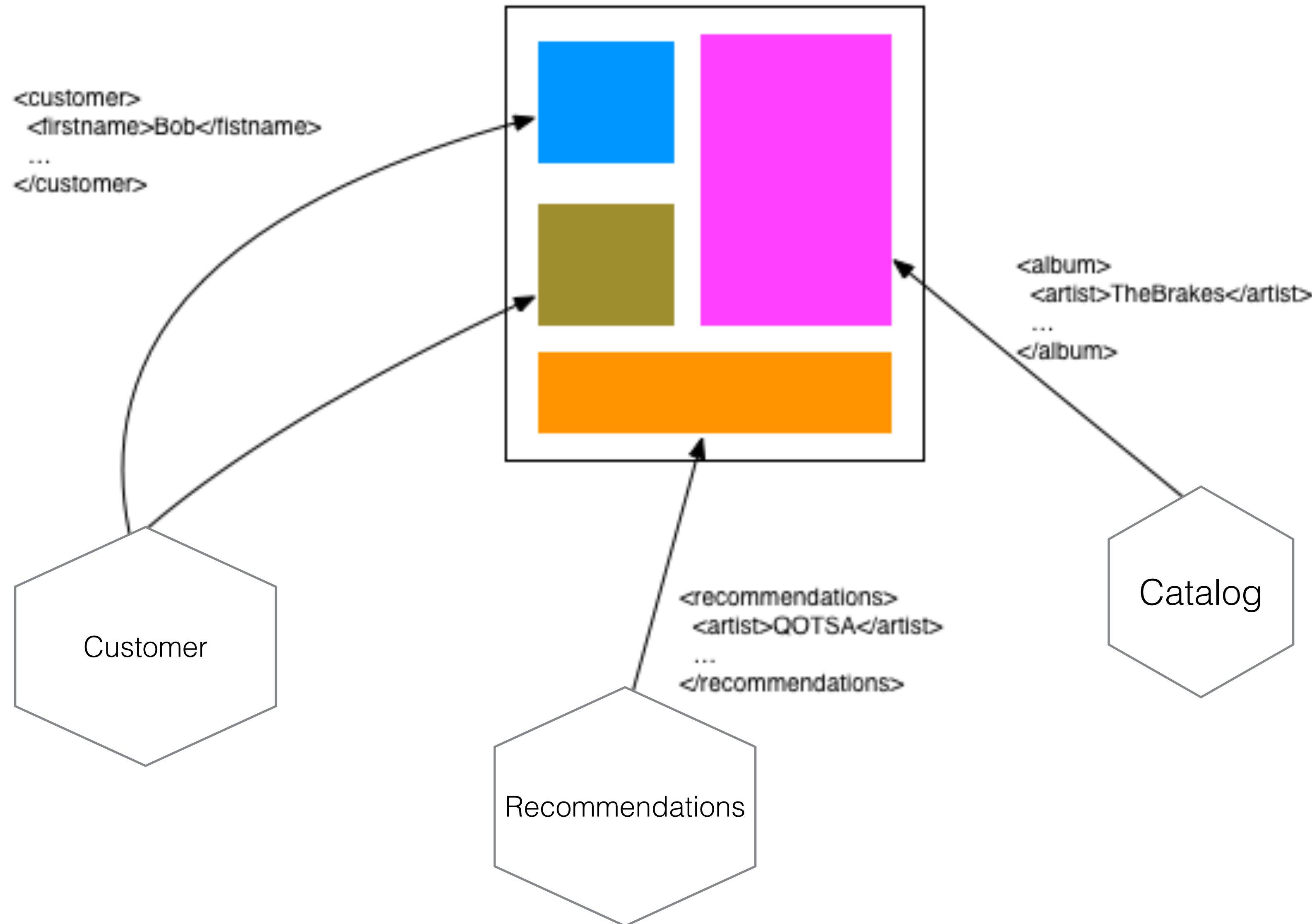
Introduction

Migration Approach

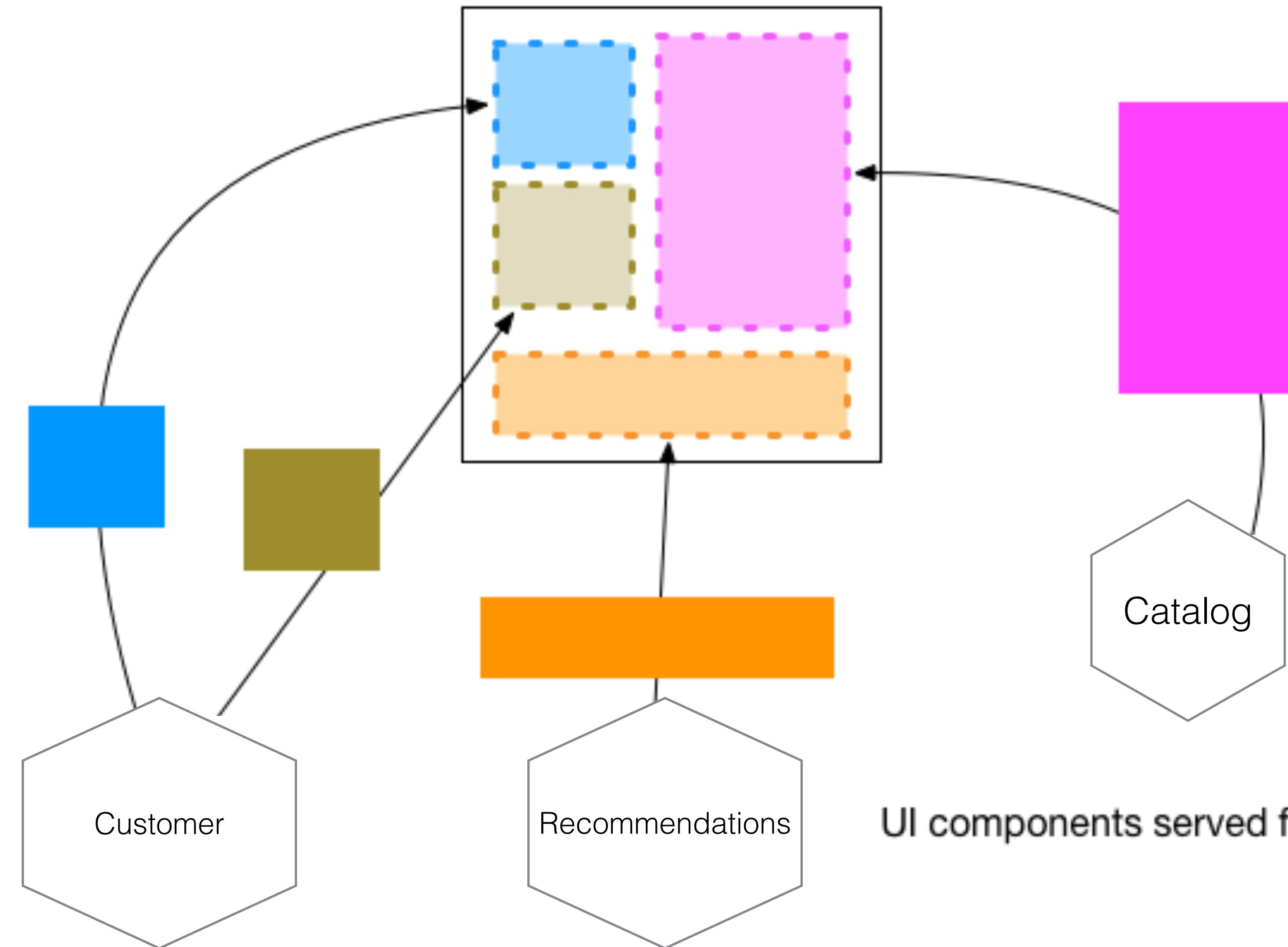
Application Refactoring

UI Decomposition

SINGLE-PAGE APP APPROACH



WIDGET-BASED ASSEMBLY



EXAMPLES OF WIDGET-BASED ASSEMBLY

The screenshot shows the Orbitz website interface. At the top, there's a navigation bar with links for "Join Rewards", "List your property", "Account", "My Lists", "My Trips", "Support", and "Español". Below the navigation is a secondary menu with links for "Hotels", "Flights", "Vacation packages", "Cars", "Vacation Rentals", "Cruises", "Things to do", "Deals", "Mobile", and "Travel Blog". A prominent yellow banner at the top left says "Travel update" with a warning about COVID-19 and a link to the Customer Support Portal. Below the banner is a search form with tabs for "Flights", "Hotels" (which is selected), "Vacation Packages", "Cars", "Cruises", and "Things to Do". The search form includes fields for "Going to" (destination), "Check-in" (mm/dd/yyyy), "Check-out" (mm/dd/yyyy), "Rooms" (1), "Adults (18+)" (2), "Children (0-17)" (0), and checkboxes for "Add a flight" and "Add a car". A red "Search" button is located at the bottom left of the search form. A note below the search form states: "Due to coronavirus (COVID-19) travel advisories, some destinations may be unavailable for certain dates. [Find out more.](#)" At the very bottom of the page, there's a blue banner with the text "Get an extra 10% off or more on select hotels with Insider Pricing".

EXAMPLES OF WIDGET-BASED ASSEMBLY

The image displays two distinct user interfaces demonstrating widget-based assembly:

Orbitz Travel Website (Left Side):

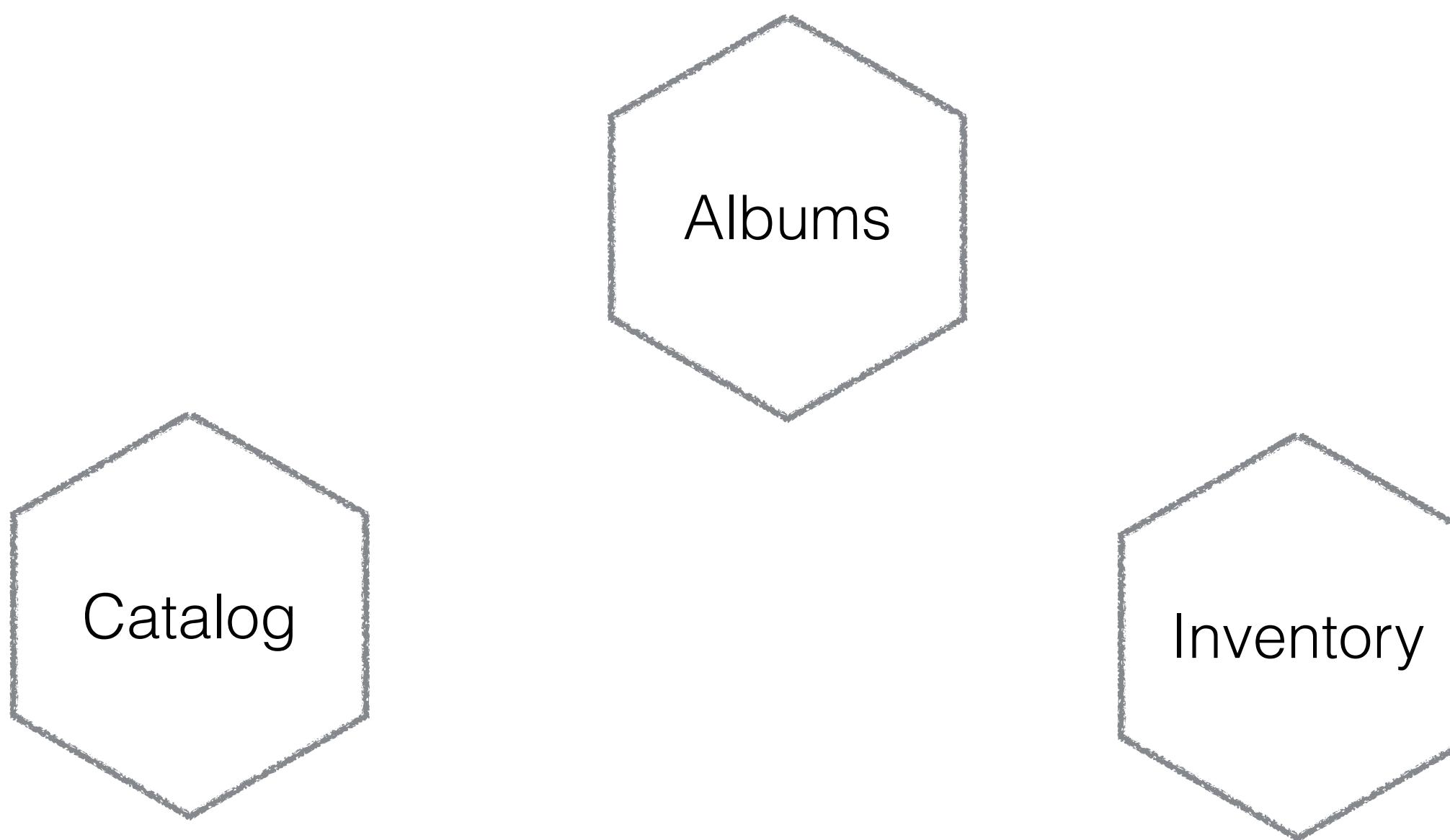
- Header:** Orbitz logo, Join Rewards button, and navigation links for Hotels, Flights, Vacation packages, Cars, Vacation Rentals, and Cruises.
- Alert:** A yellow banner titled "Travel update" informs users about COVID-19 travel impacts and directs them to the Customer Support Portal.
- Search:** A large search bar with placeholder text "Going to" and dropdown fields for Destination, Check-in (mm/dd/yyyy), and Check-out (mm/dd/yyyy).
- Filters:** Options to "Add a flight" or "Add a car".
- Search Button:** A prominent red "Search" button.
- Message:** A note about COVID-19 travel advisories.
- Offer:** A blue banner at the bottom offering an extra 10% off.

Spotify Playlist Page (Right Side):

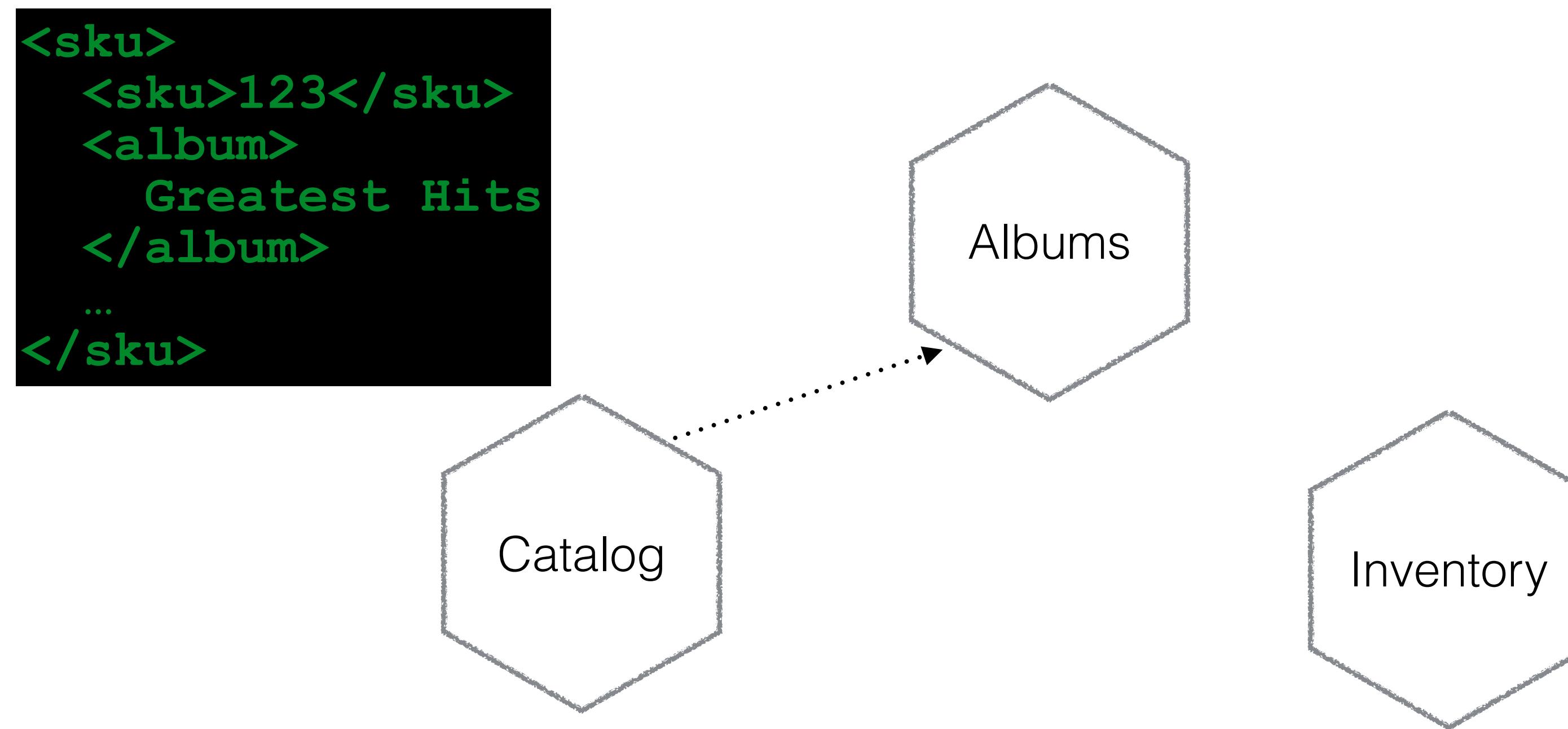
- Header:** Search bar, profile icon for samuelnewman, and a "Friend Activity" sidebar.
- Left Sidebar:** Home, Browse, Radio, and PLAYLISTS sections listing June 2020, May 2020, Motion, April 2020, Here I Go Again, March 2020, and February 2020.
- Center Content:** A large title "June 2020" indicating the playlist's name and creation date by samuelnewman (11 songs, 36 min). It includes a "PLAY" button and a "Download" toggle.
- Table:** A list of songs in the playlist, showing columns for TITLE, ARTIST, ALBUM, and DATE.
- Bottom:** A media player interface showing the current song "Hypercolour" by CamelPhat, Yannis, Foals, with playback controls and a progress bar.

TITLE	ARTIST	ALBUM	DATE
Take Me Home, Country Roads (ft. W...	Whitney, Wax...	Take Me Hom...	2020-05-29
They Won't Be Happy Till They Blow ...	Damn Vandals	They Won't B...	2020-05-29
Sludge	Squid	Sludge	2020-06-04
Mr. Motivator	IDLES	Mr. Motivator	2020-06-04
might bang, might not	Little Simz	Drop 6	2020-06-04
Spoiler	Baloji	Spoiler	2020-06-09
Kyoto	Phoebe Bridg...	Kyoto	2020-06-09
JU\$T (feat. Pharrell Willia...	Run The Jewe...	RTJ4	2020-06-16
Bob	Frank Turner	Bob	10 days ago
My Own Soul's Warning	The Killers	My Own Soul'...	9 days ago

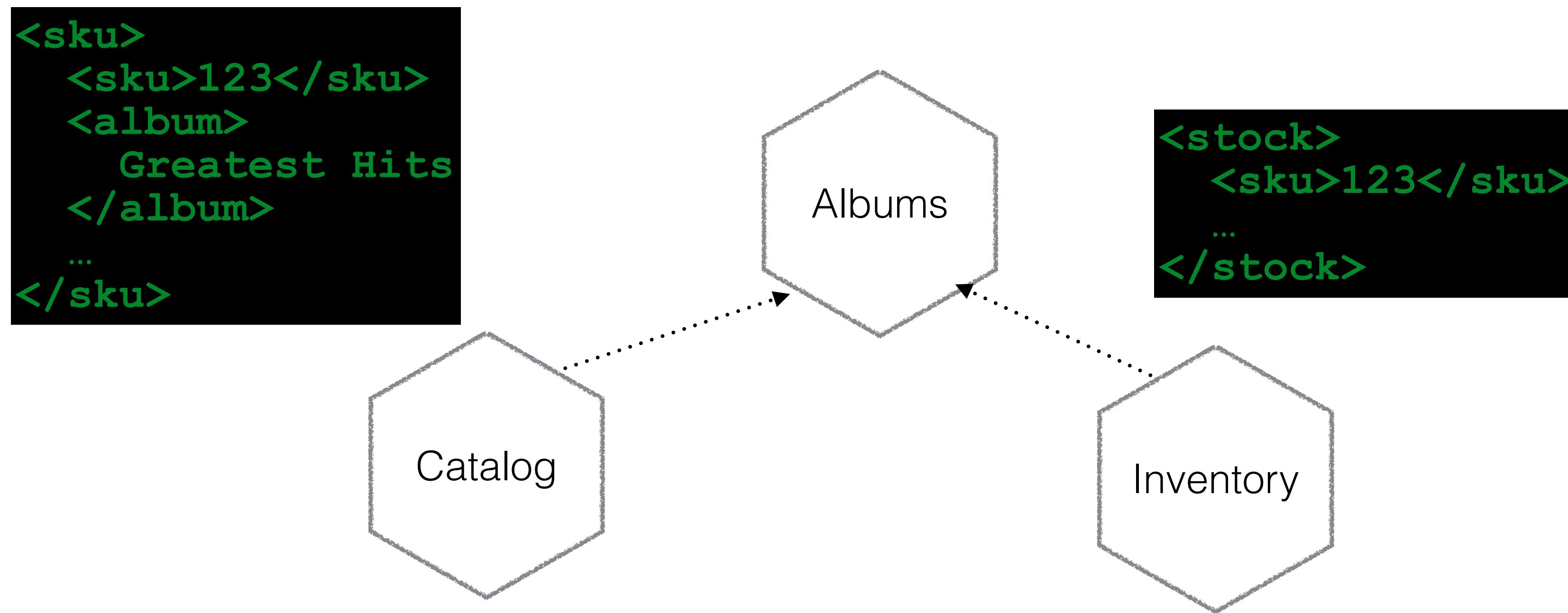
WIDGET-BASED ASSEMBLY - DETAILS



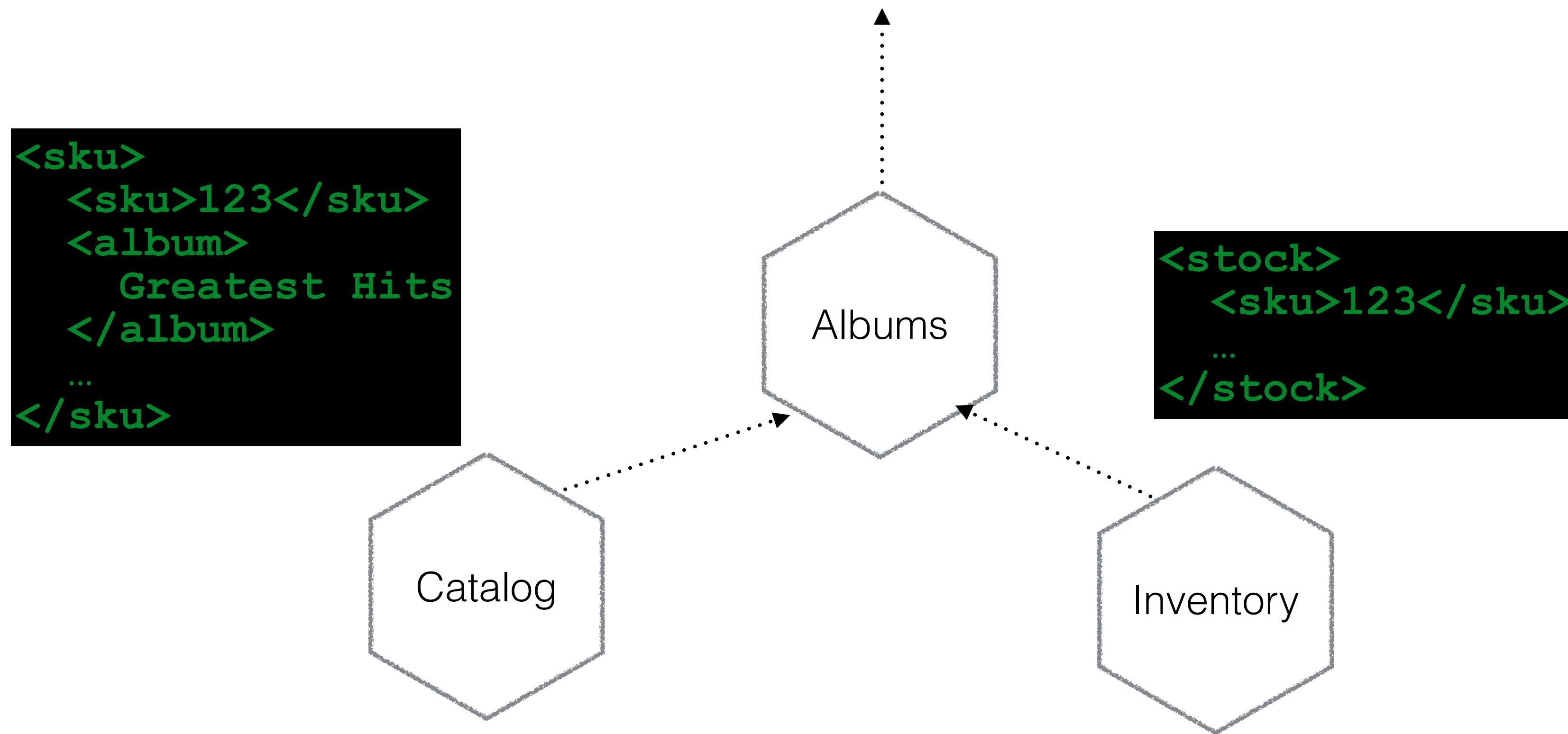
WIDGET-BASED ASSEMBLY - DETAILS



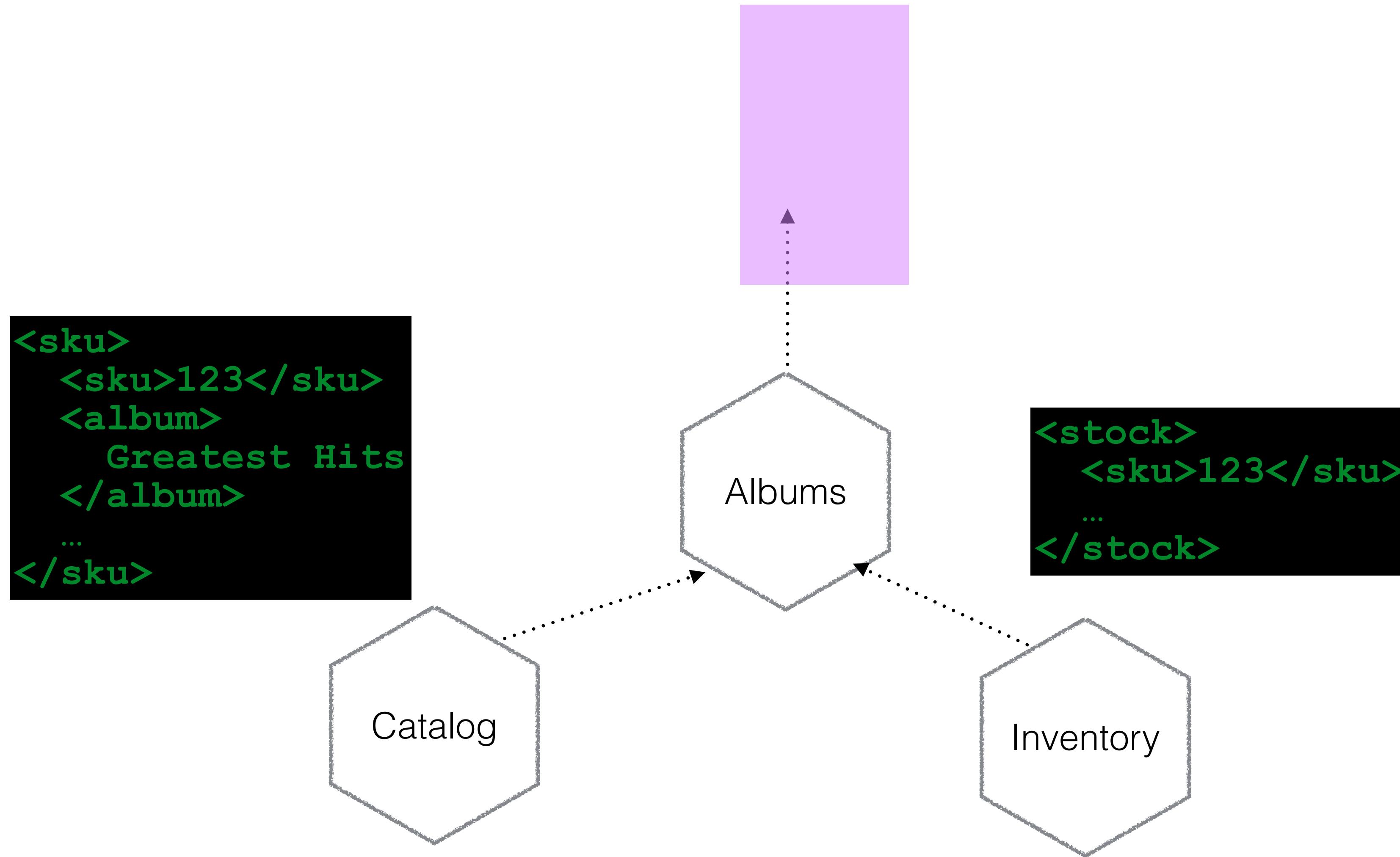
WIDGET-BASED ASSEMBLY - DETAILS



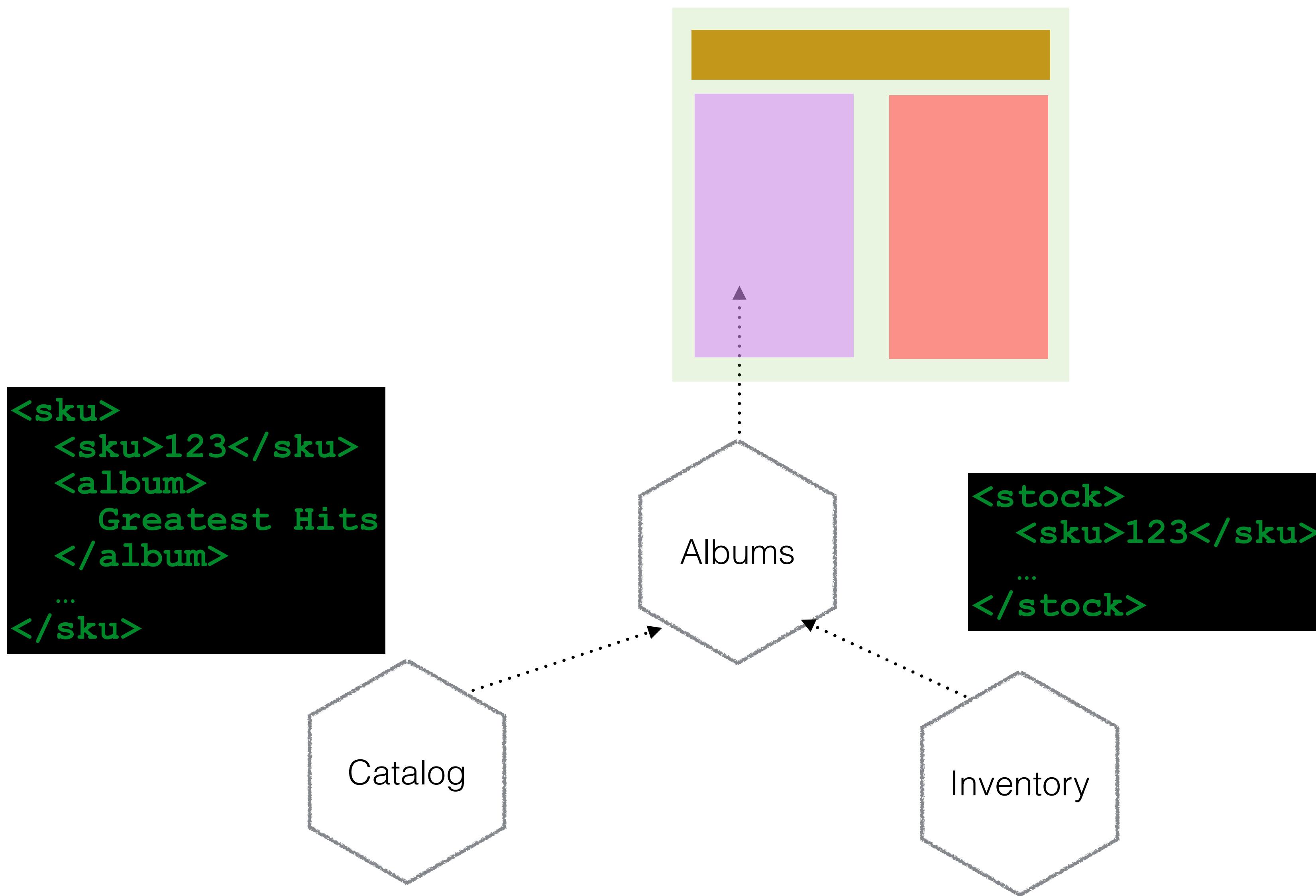
WIDGET-BASED ASSEMBLY - DETAILS



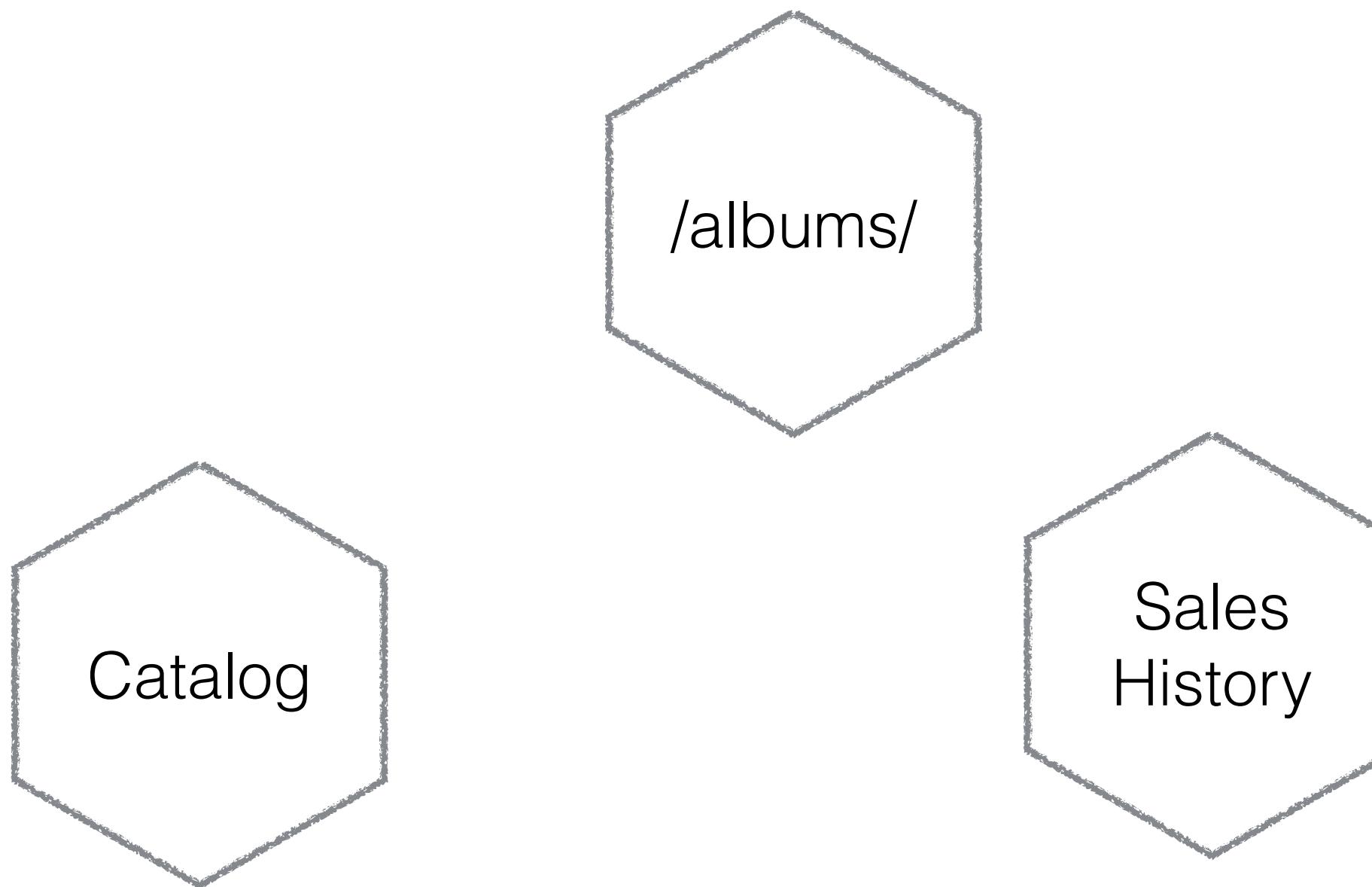
WIDGET-BASED ASSEMBLY - DETAILS



WIDGET-BASED ASSEMBLY - DETAILS

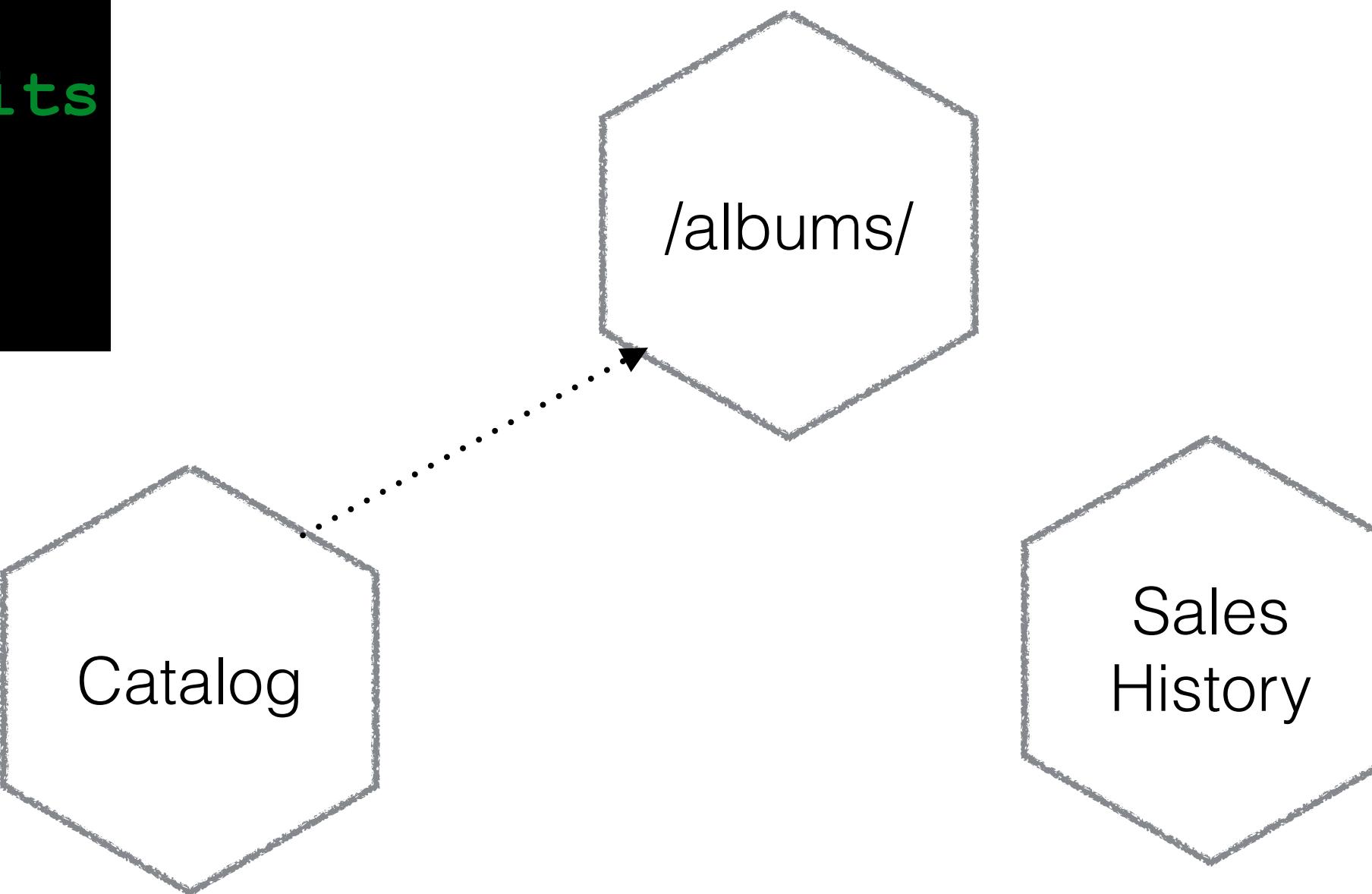


PAGE-BASED ASSEMBLY

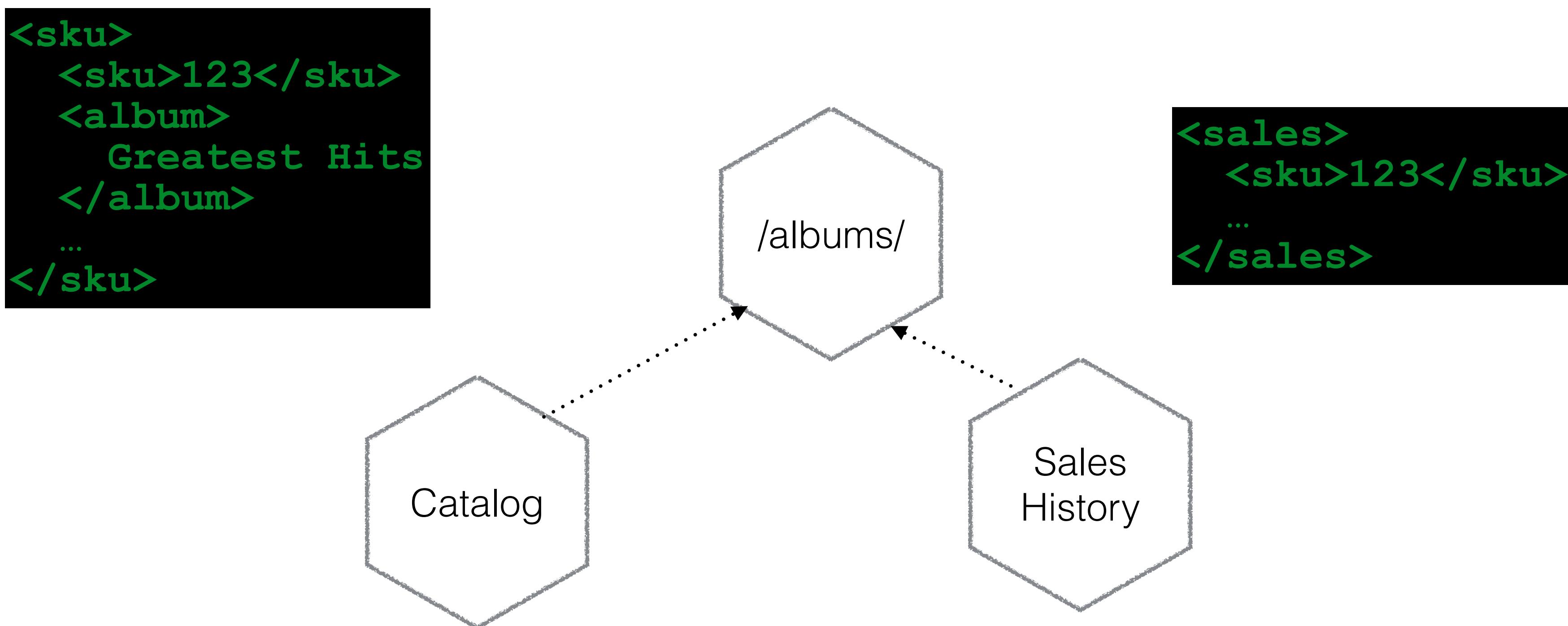


PAGE-BASED ASSEMBLY

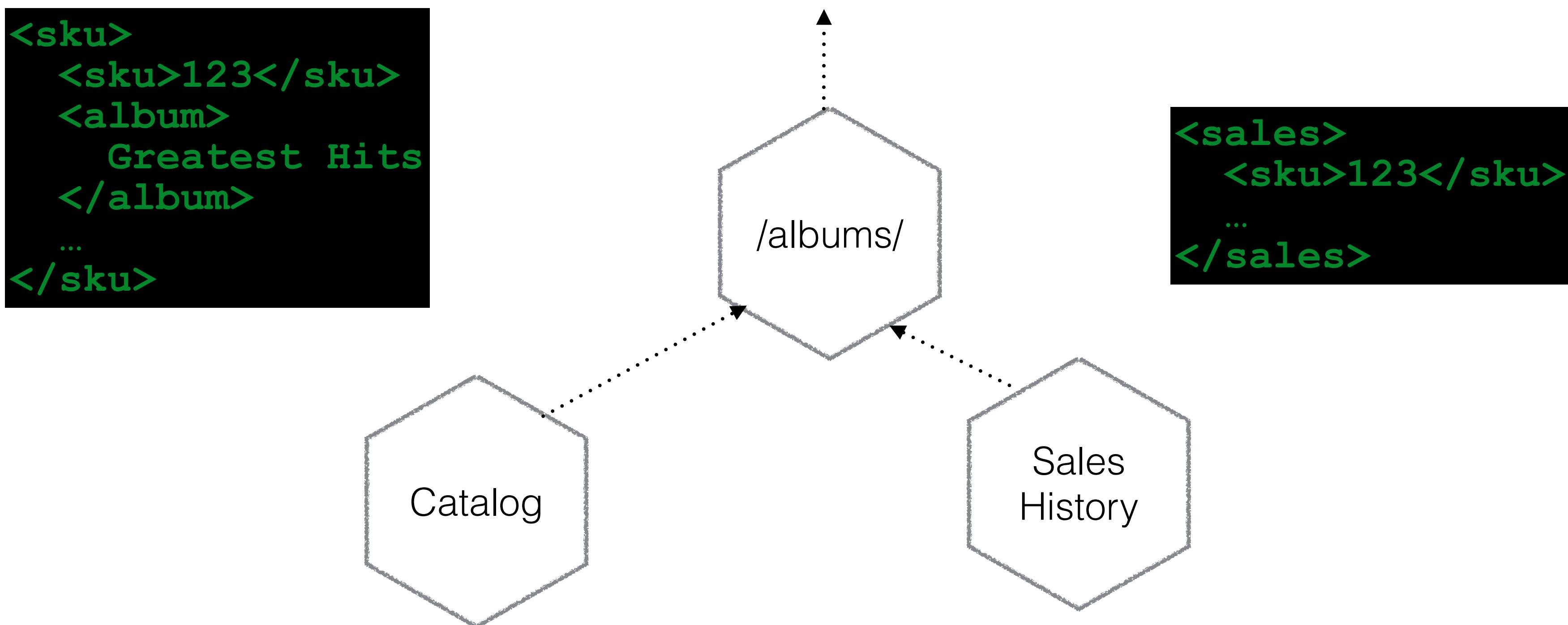
```
<sku>
  <sku>123</sku>
  <album>
    Greatest Hits
  </album>
  ...
</sku>
```



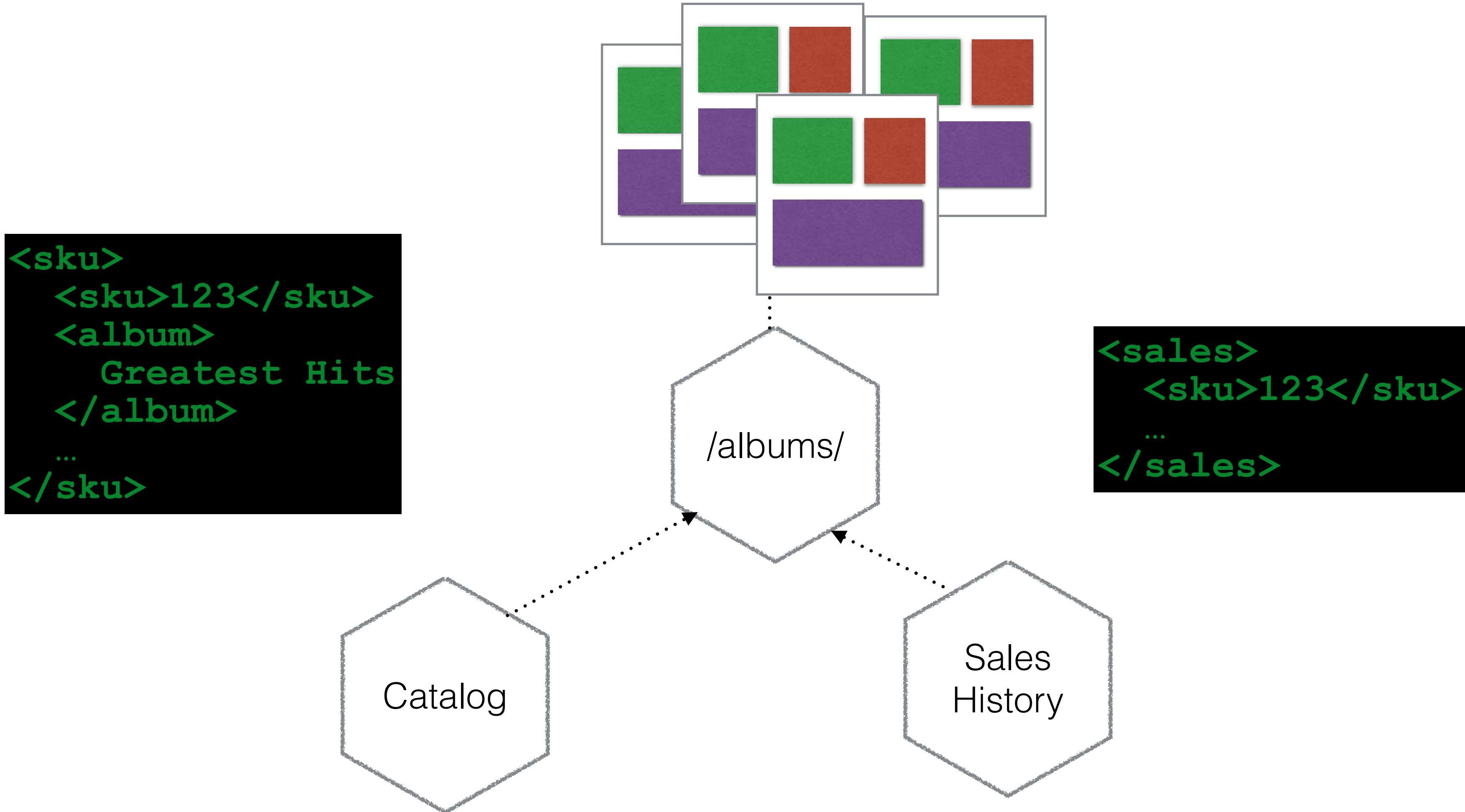
PAGE-BASED ASSEMBLY



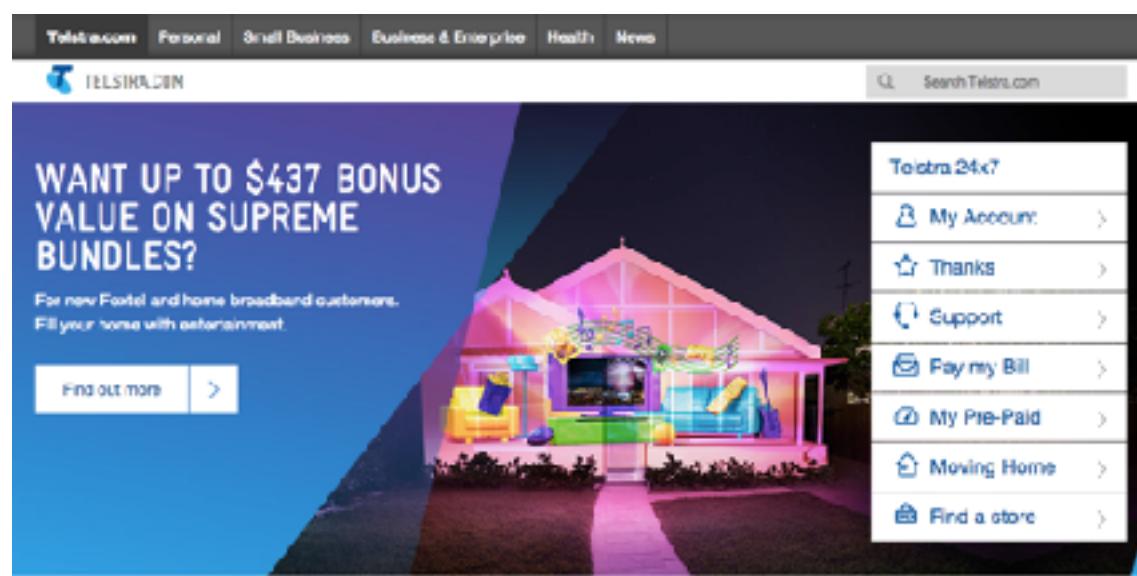
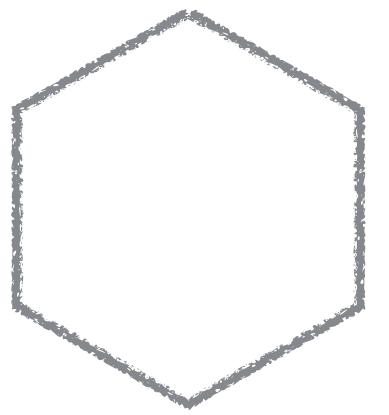
PAGE-BASED ASSEMBLY



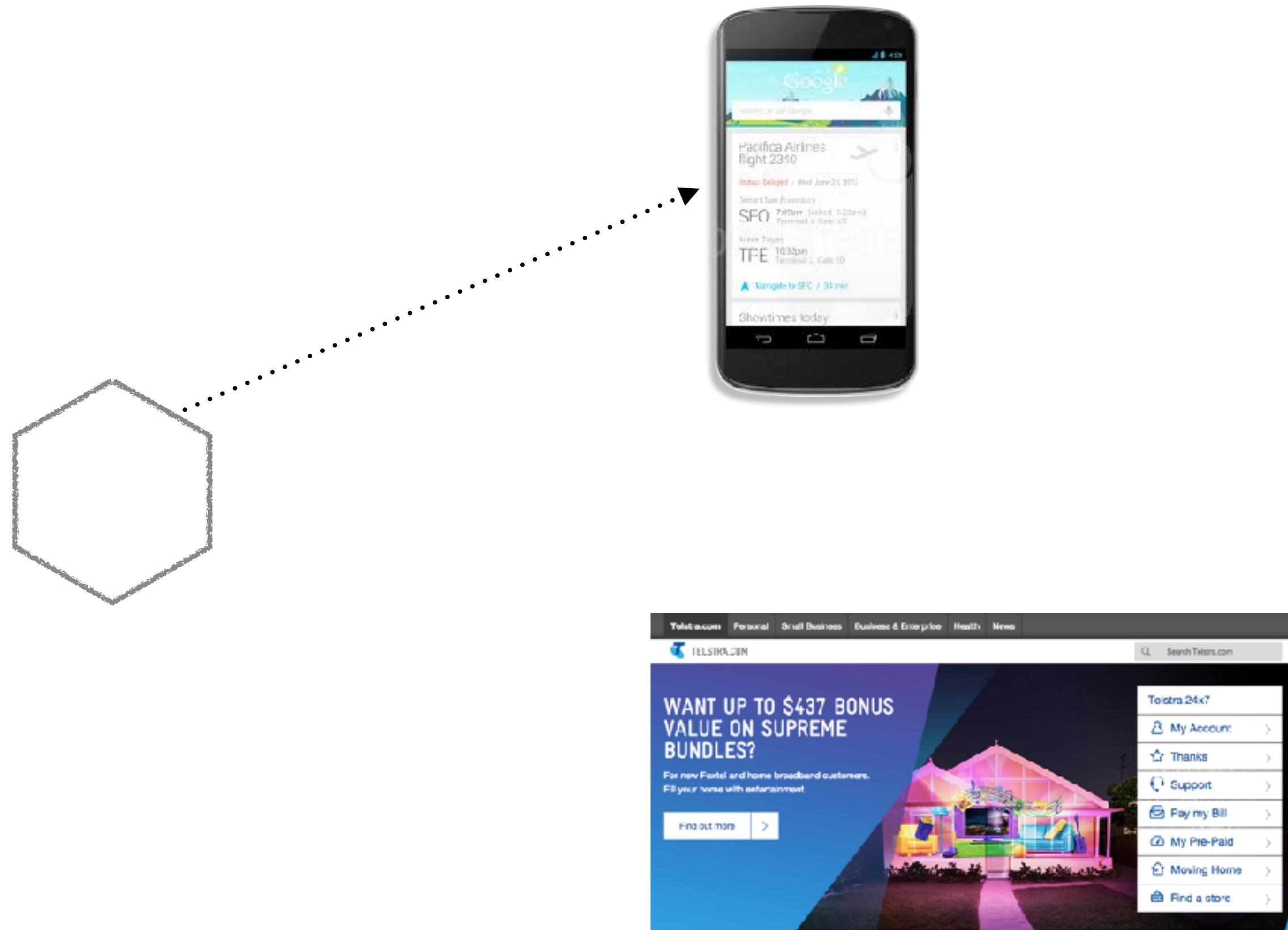
PAGE-BASED ASSEMBLY



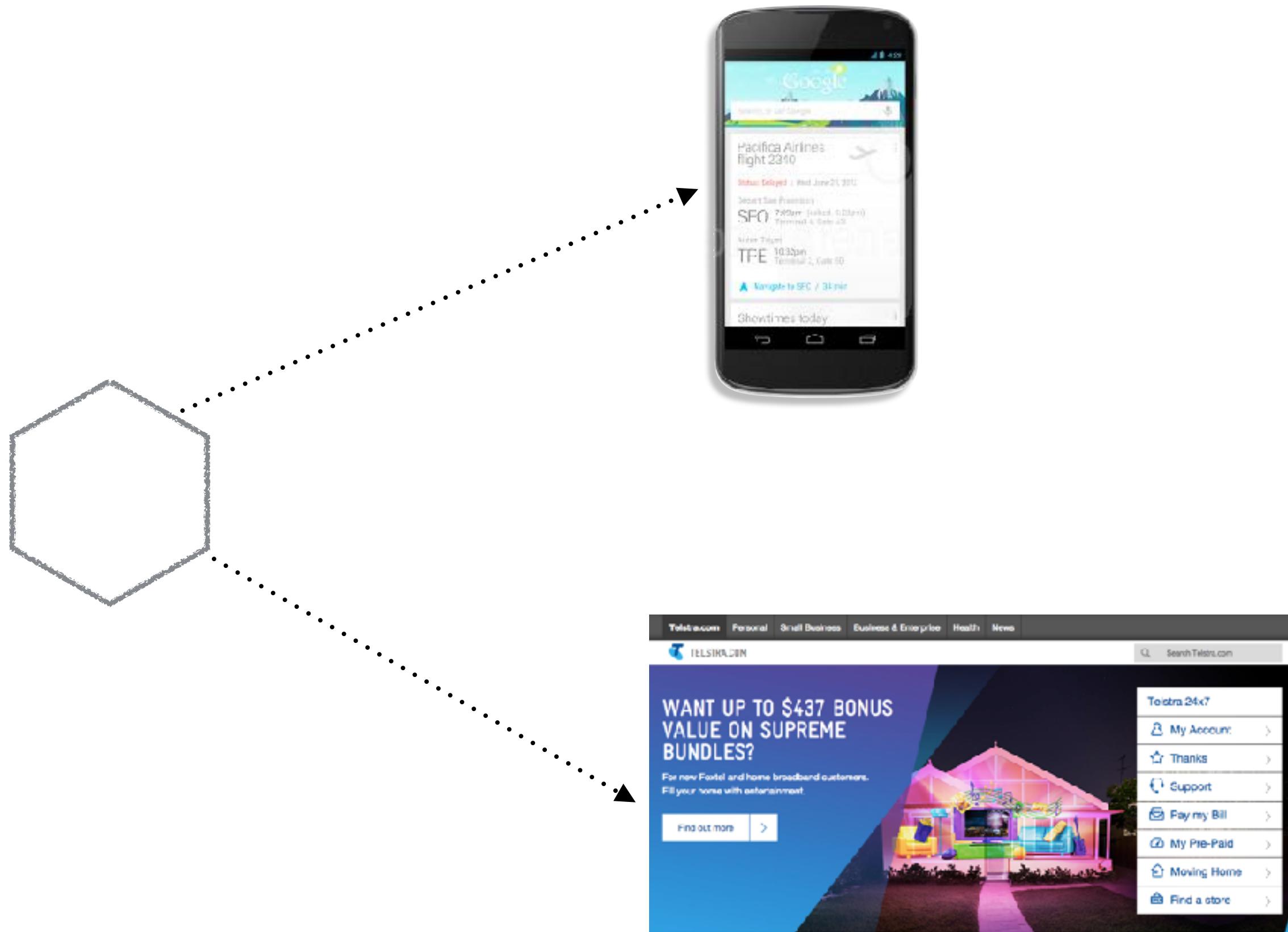
AND MOBILE?



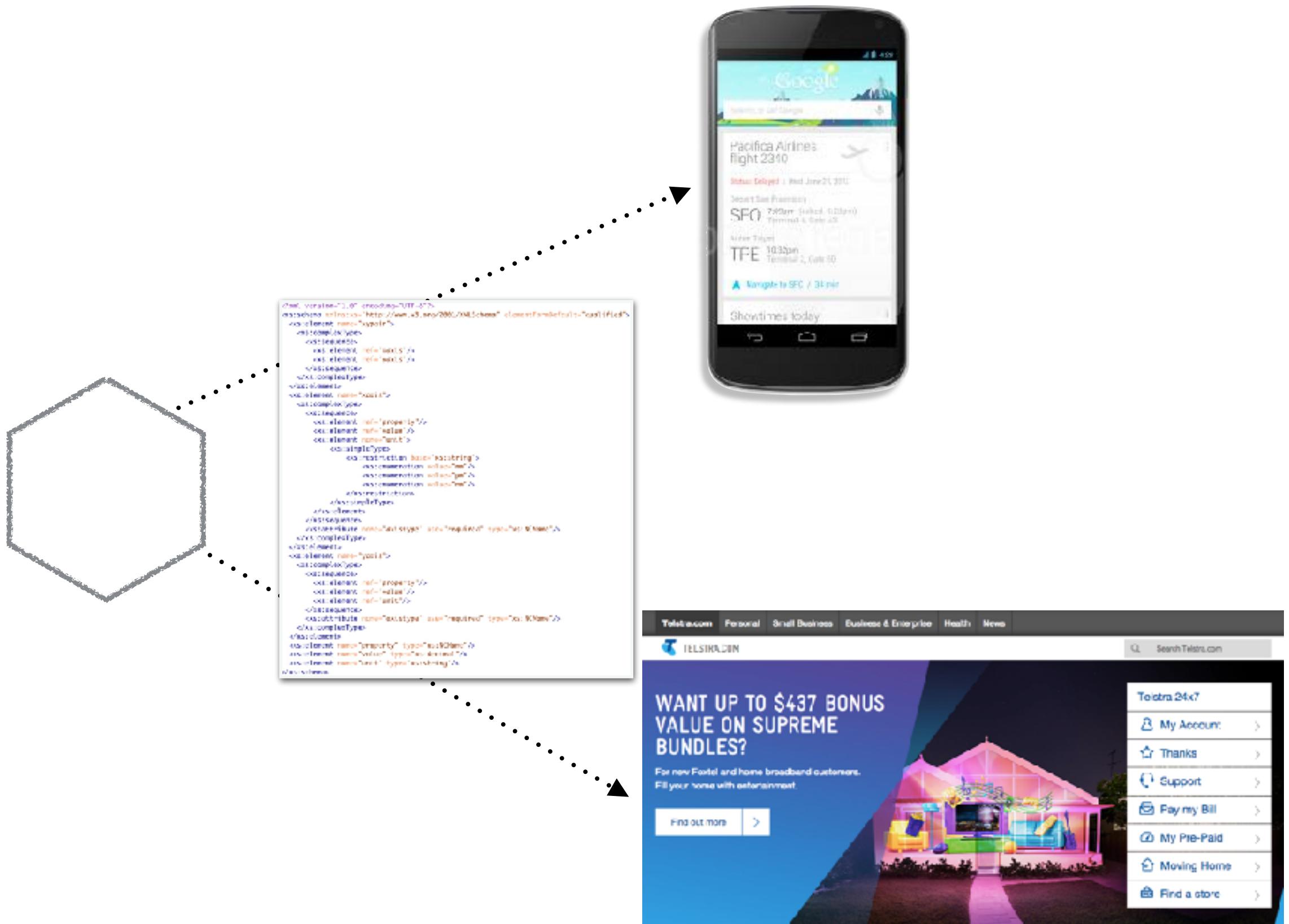
AND MOBILE?



AND MOBILE?



AND MOBILE?



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

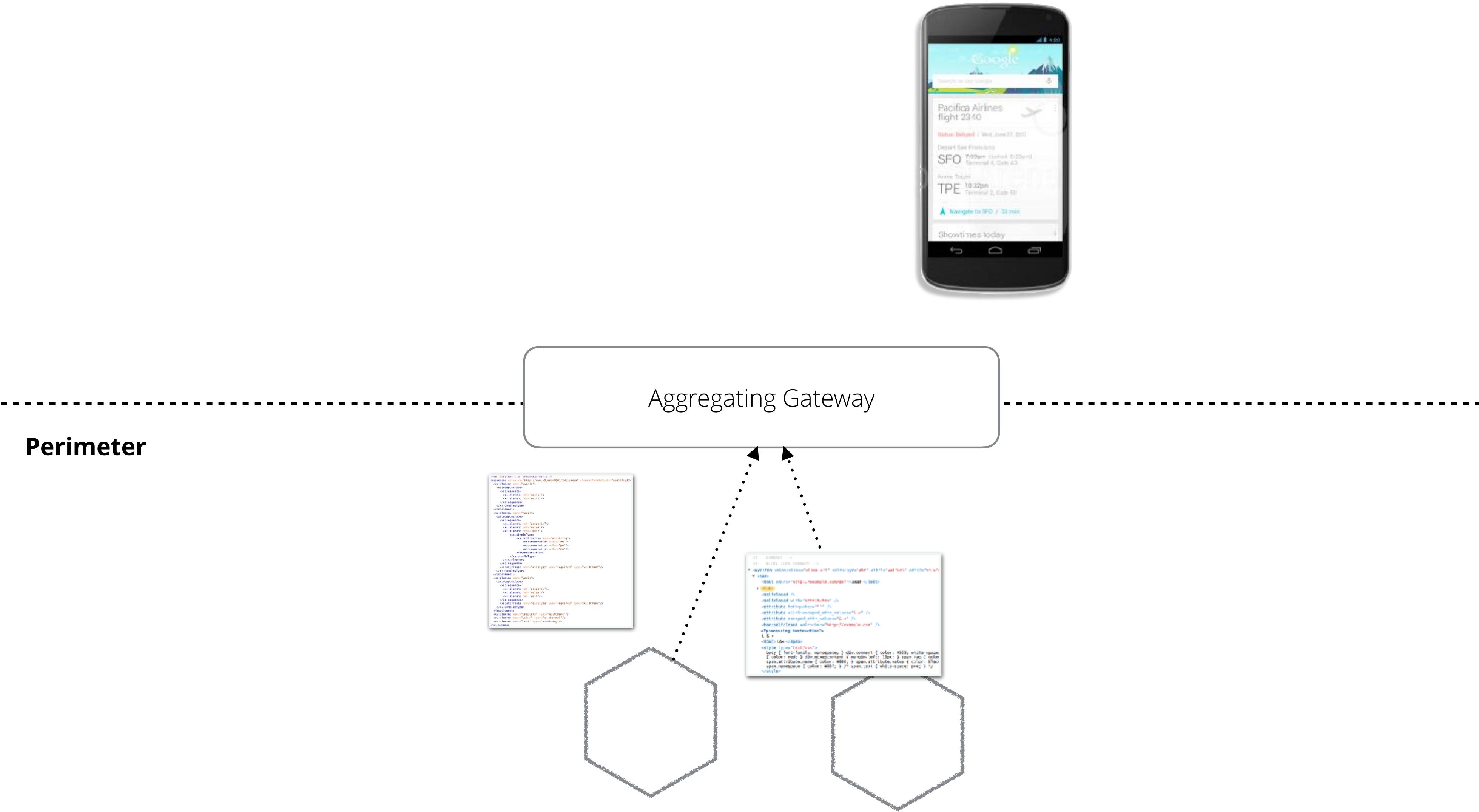
Fetching data can take longer, and drain battery (and credit!)

Use-cases can differ greatly

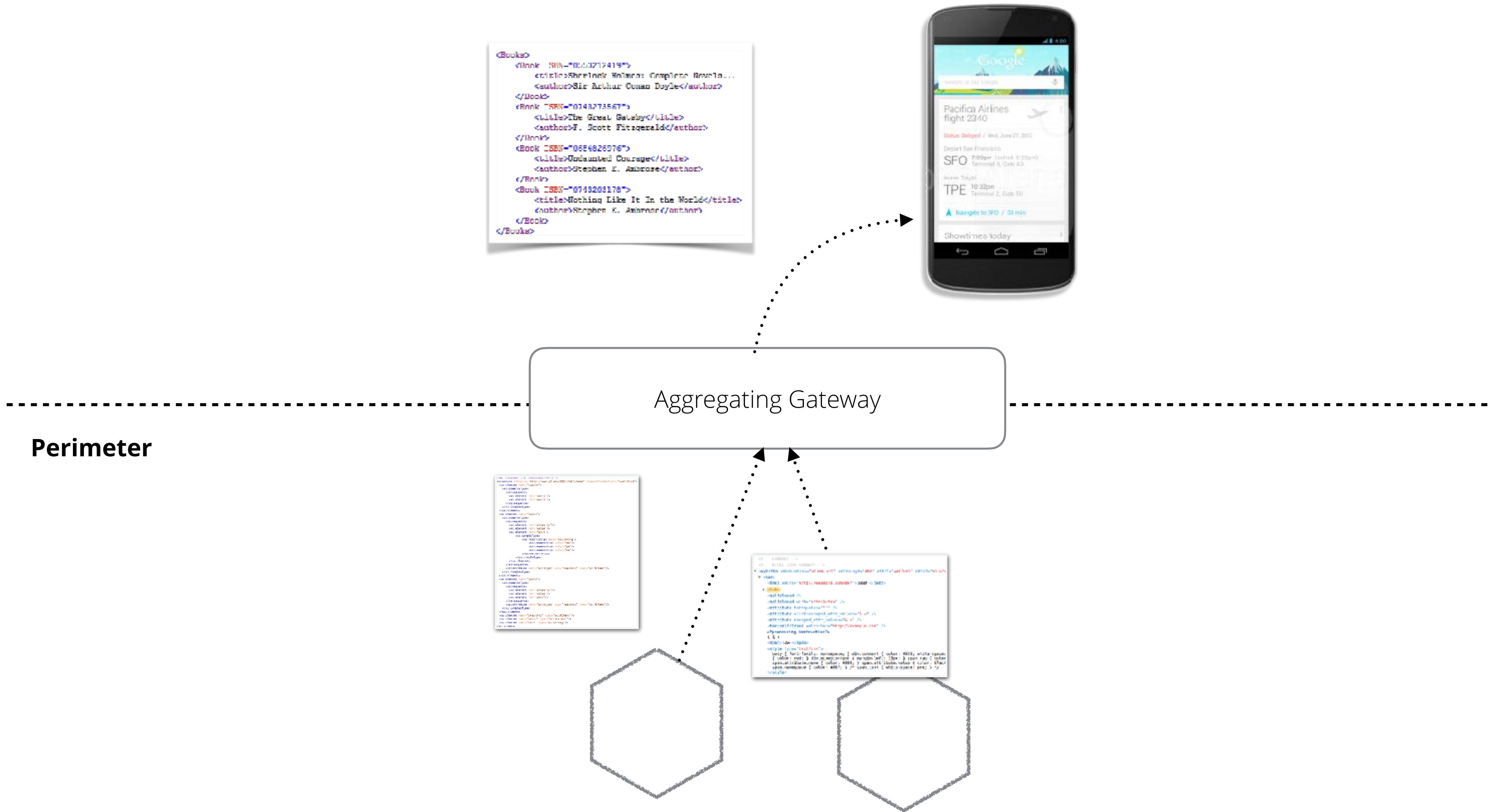
GENERAL-PURPOSE AGGREGATING GATEWAY



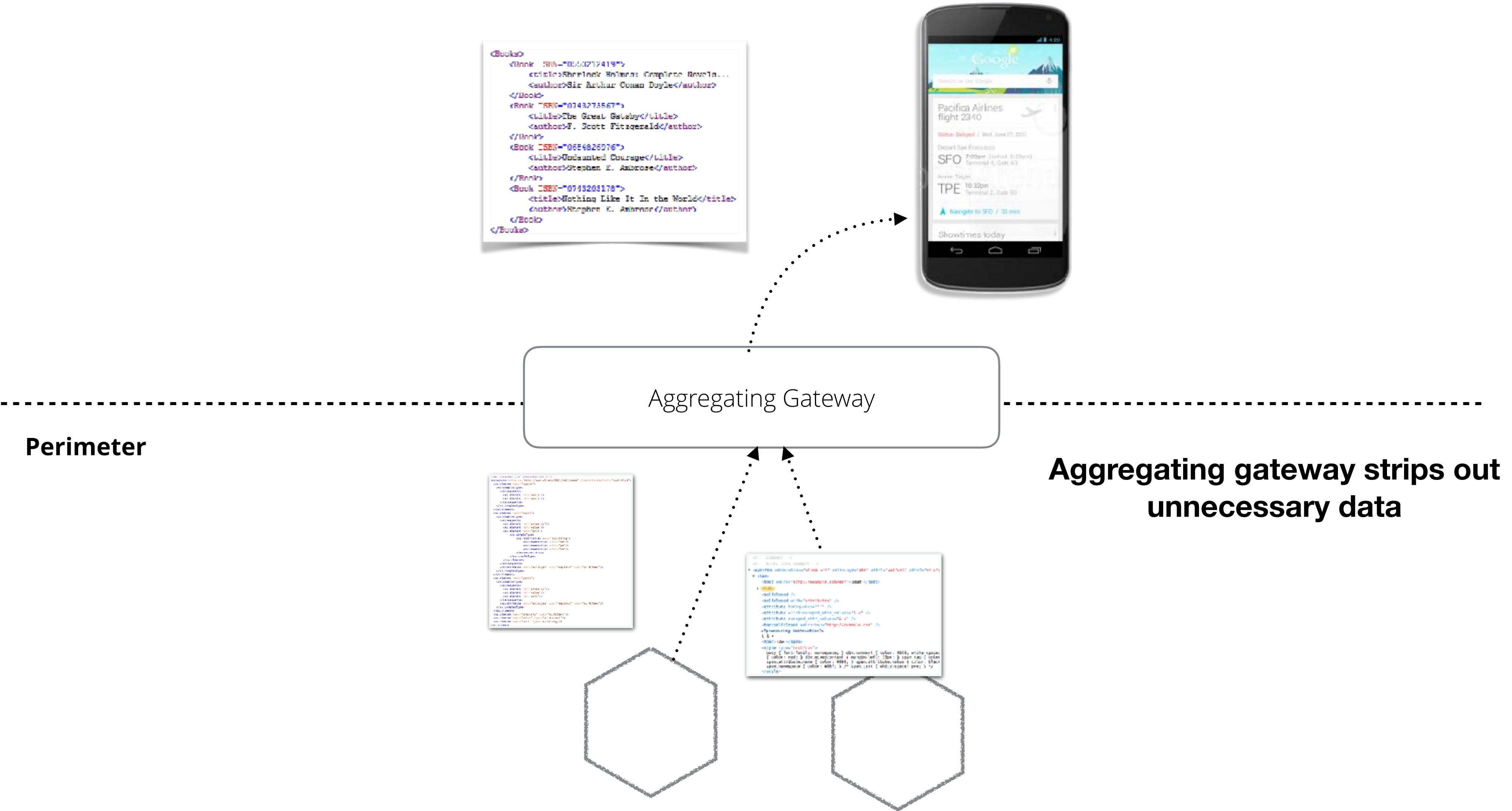
GENERAL-PURPOSE AGGREGATING GATEWAY



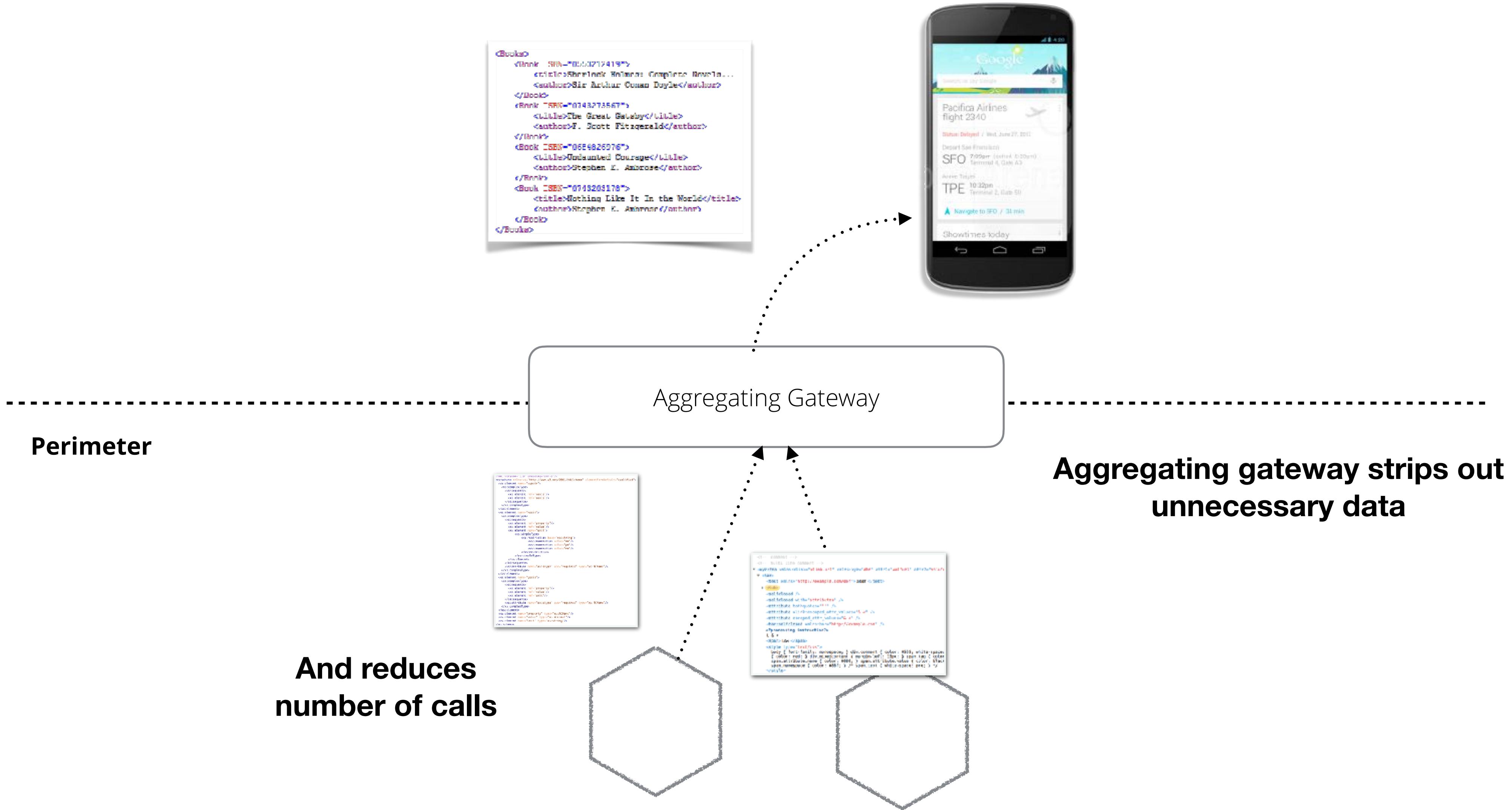
GENERAL-PURPOSE AGGREGATING GATEWAY



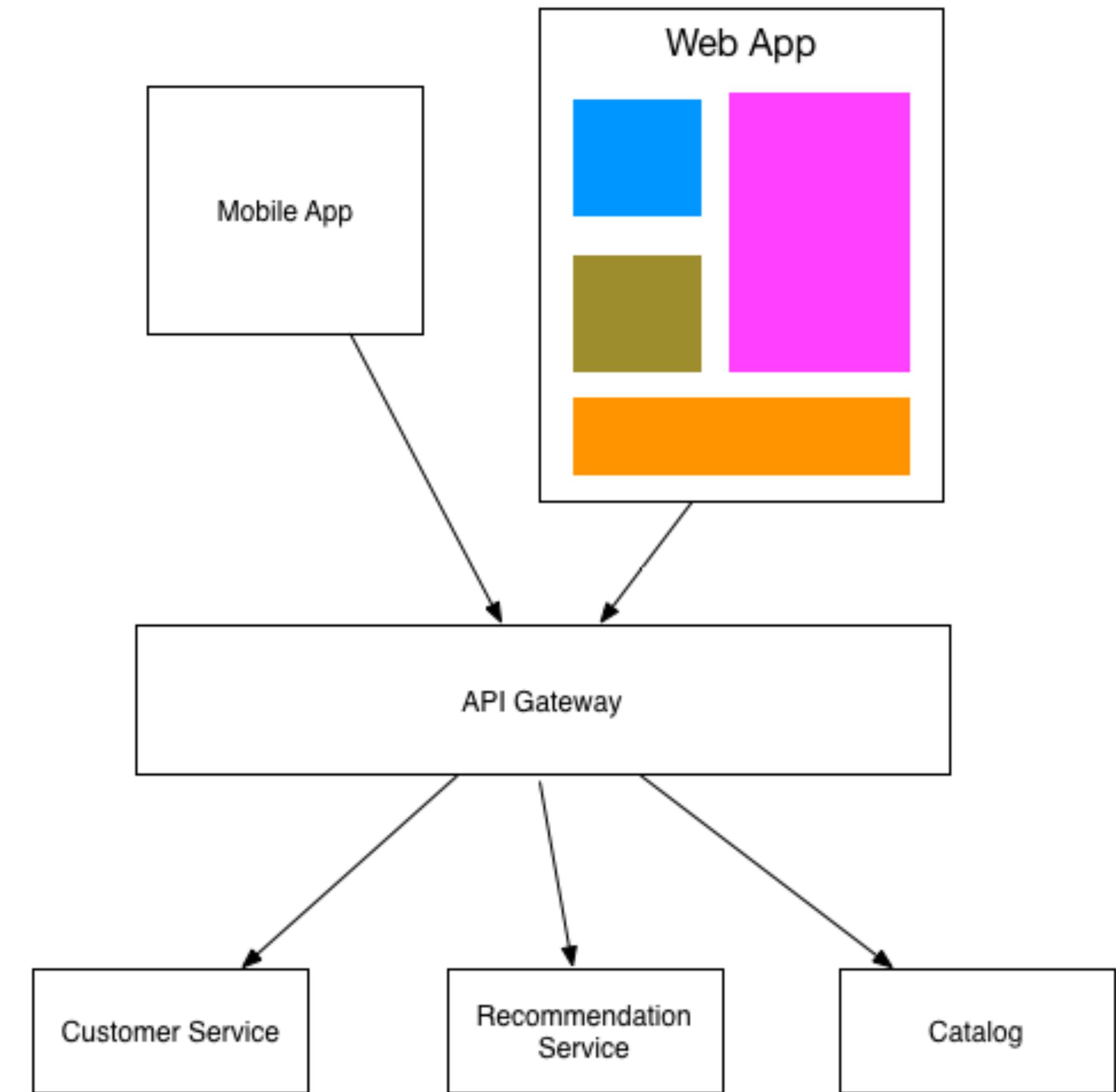
GENERAL-PURPOSE AGGREGATING GATEWAY



GENERAL-PURPOSE AGGREGATING GATEWAY



API GATEWAY



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

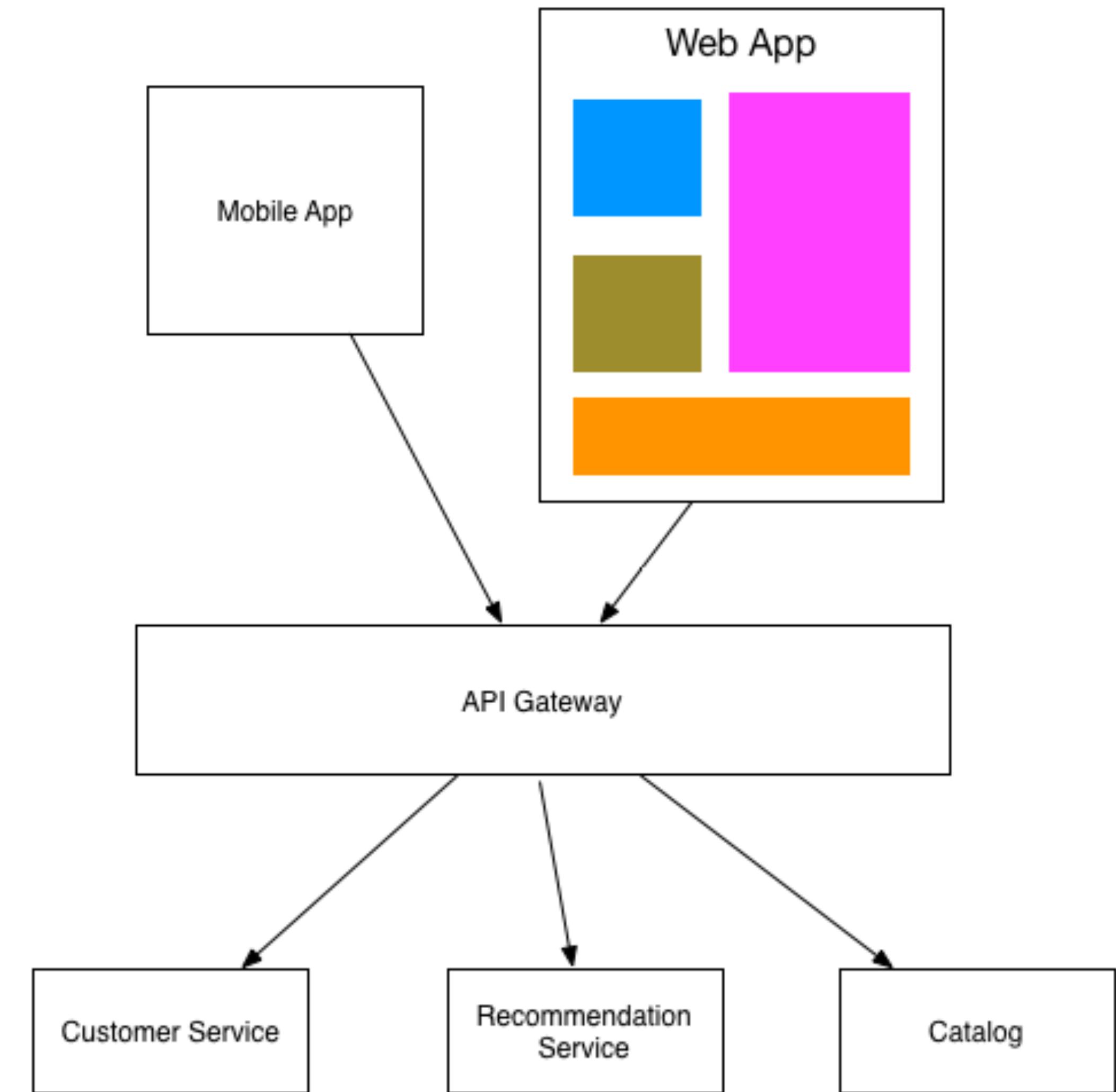
API-rate limiting

Request logging

Service discovery

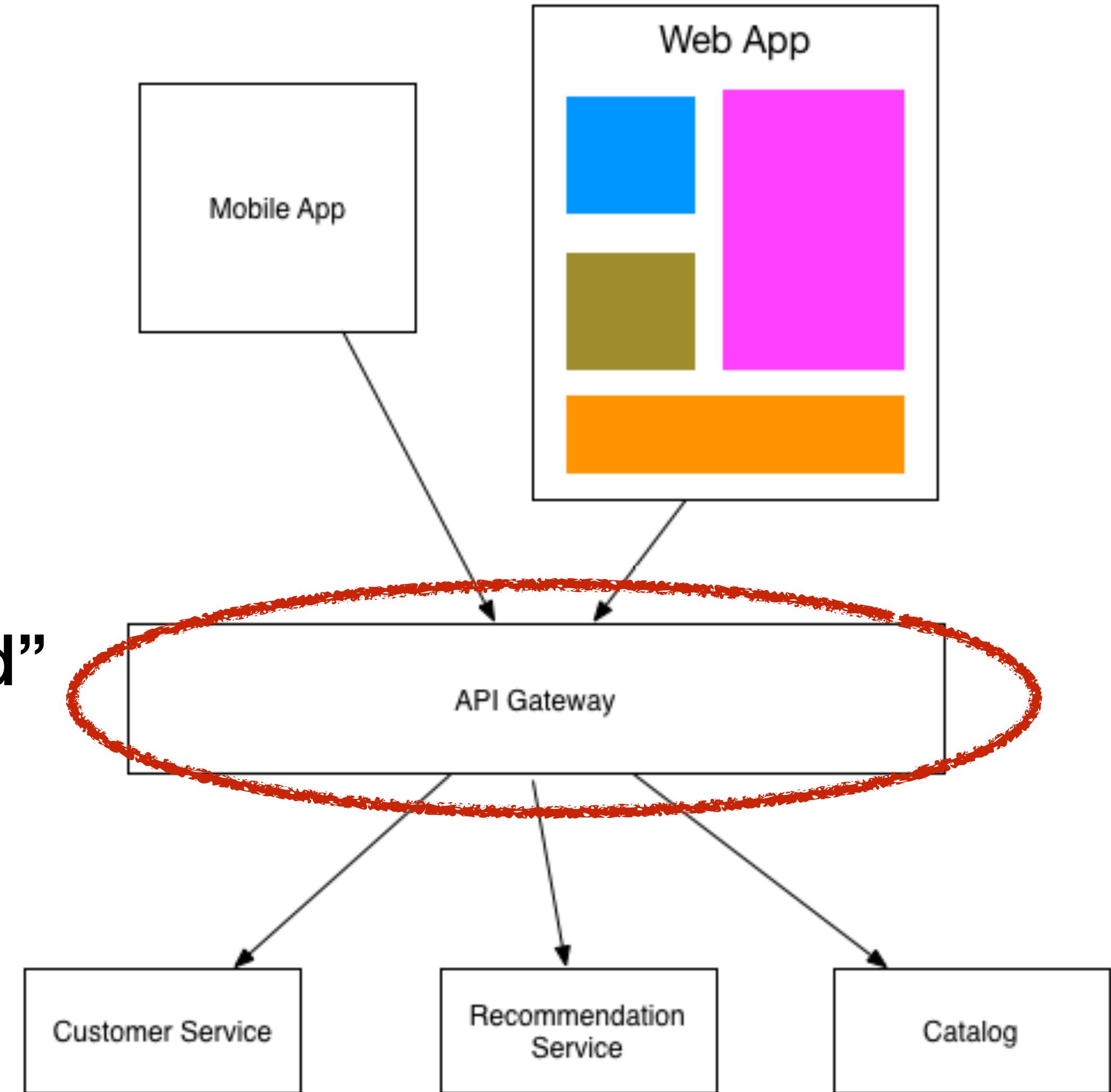
Call aggregation

API GATEWAY

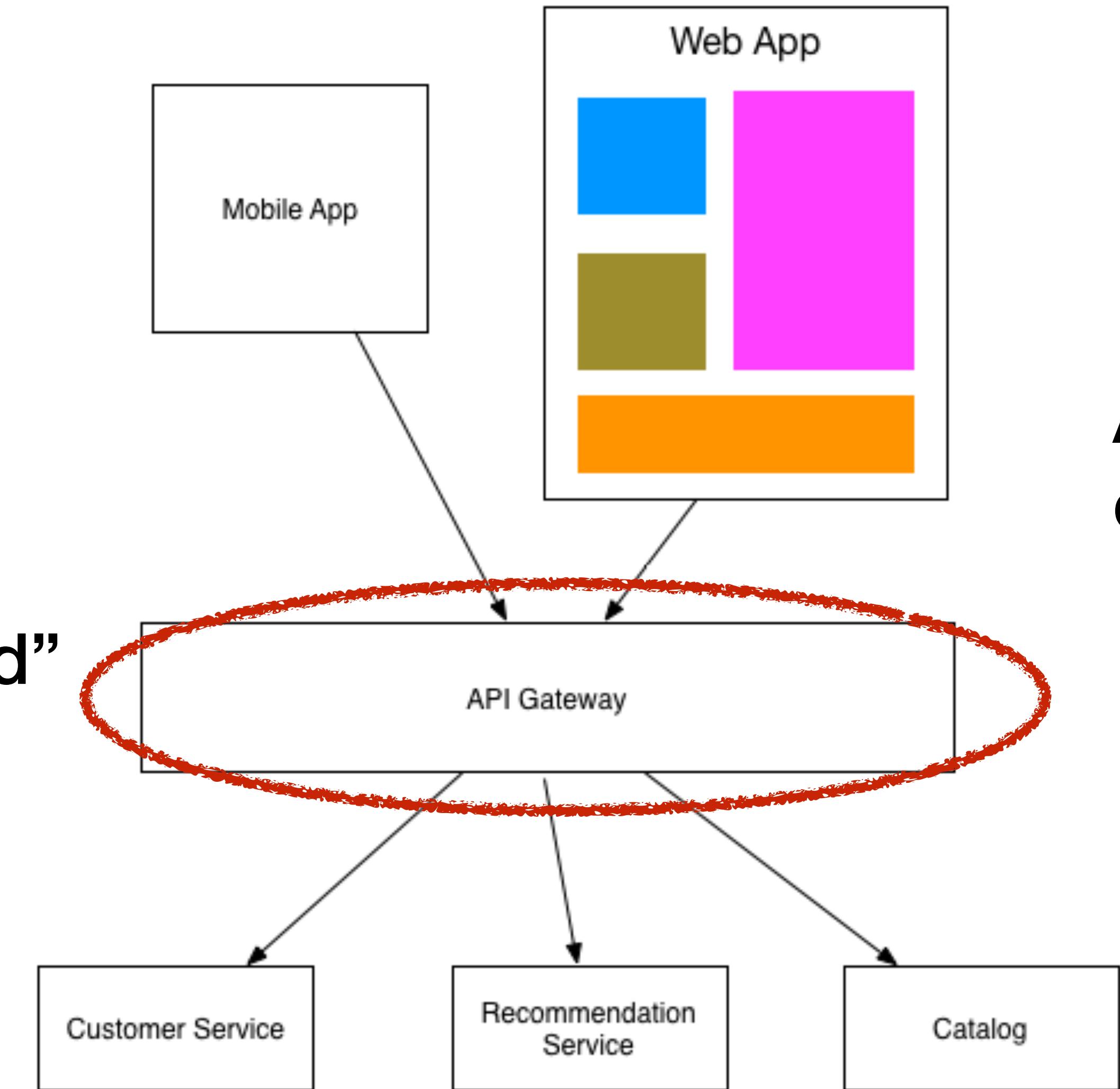


API GATEWAY

Can become “bloated”



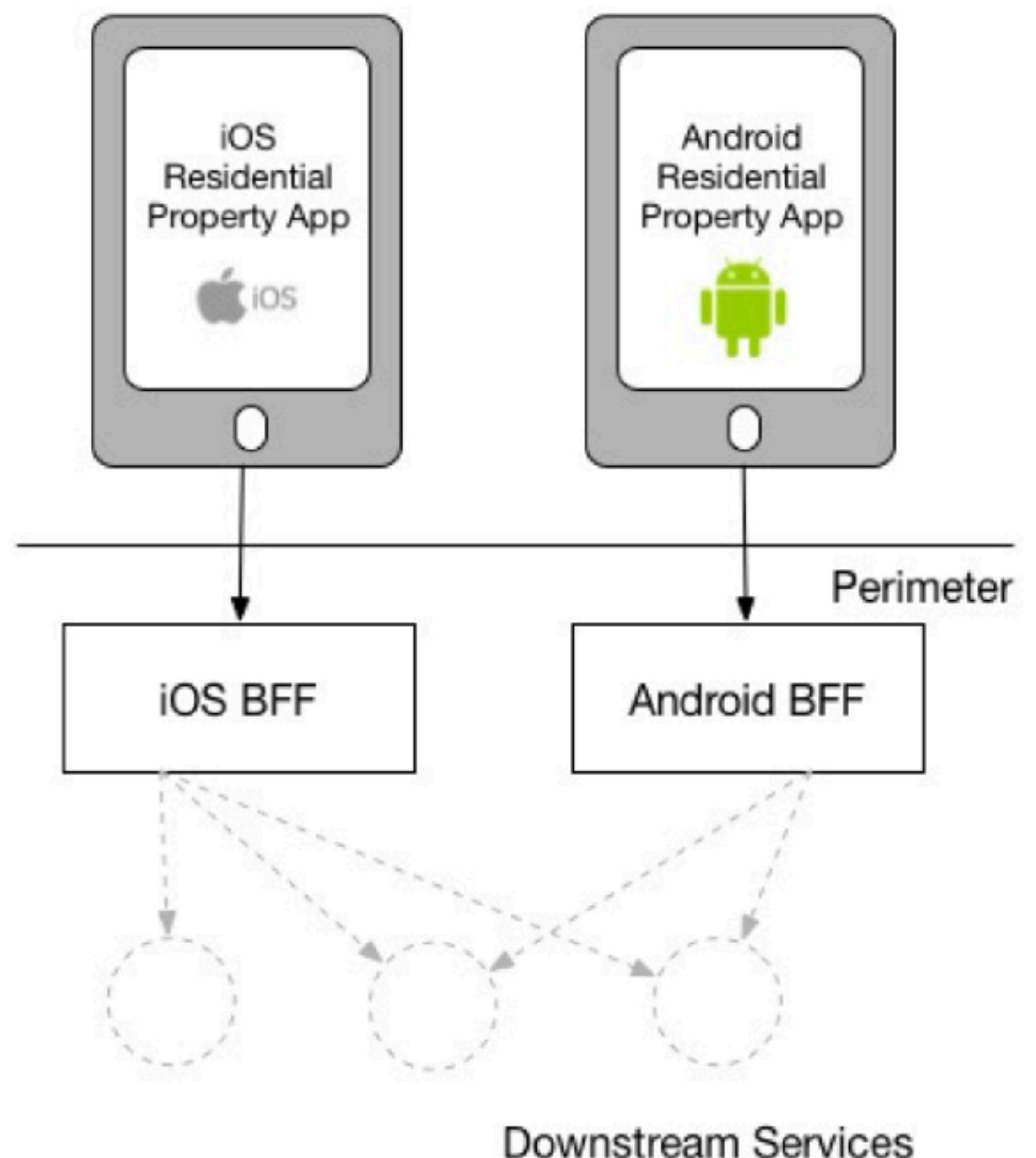
API GATEWAY



Can become “bloated”

A single point of contention

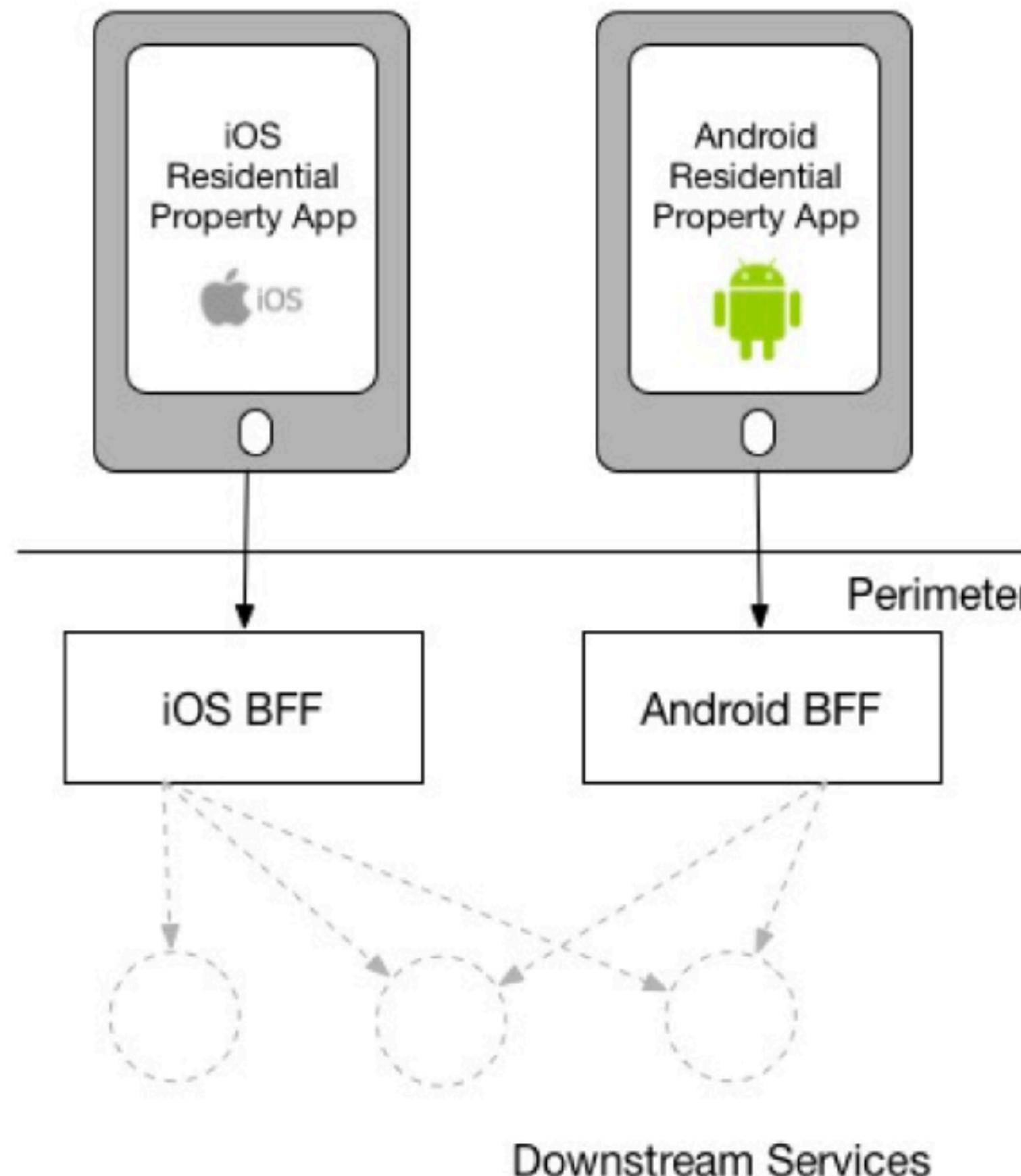
BACKENDS FOR FRONTENDS



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

@samnewman

BACKENDS FOR FRONTENDS



Pattern: Backends For Frontends.

Written on Nov 18 2015

Single-purpose Edge Services for UIs and external parties

- [1. Introduction](#)
- [2. The General-Purpose API Gateway](#)
- [3. Introducing The Backend For Frontend](#)
- [4. How Many?](#)
- [5. Handling Multiple Downstream Calls](#)
- [6. And Reuse](#)
- [7. For Desktop Web & Other Devices](#)
- [8. And Autonomy](#)
- [9. General Perimeter Concerns](#)
- [10. When To Use](#)
- [11. Further Reading](#)
- [12. Conclusion](#)

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

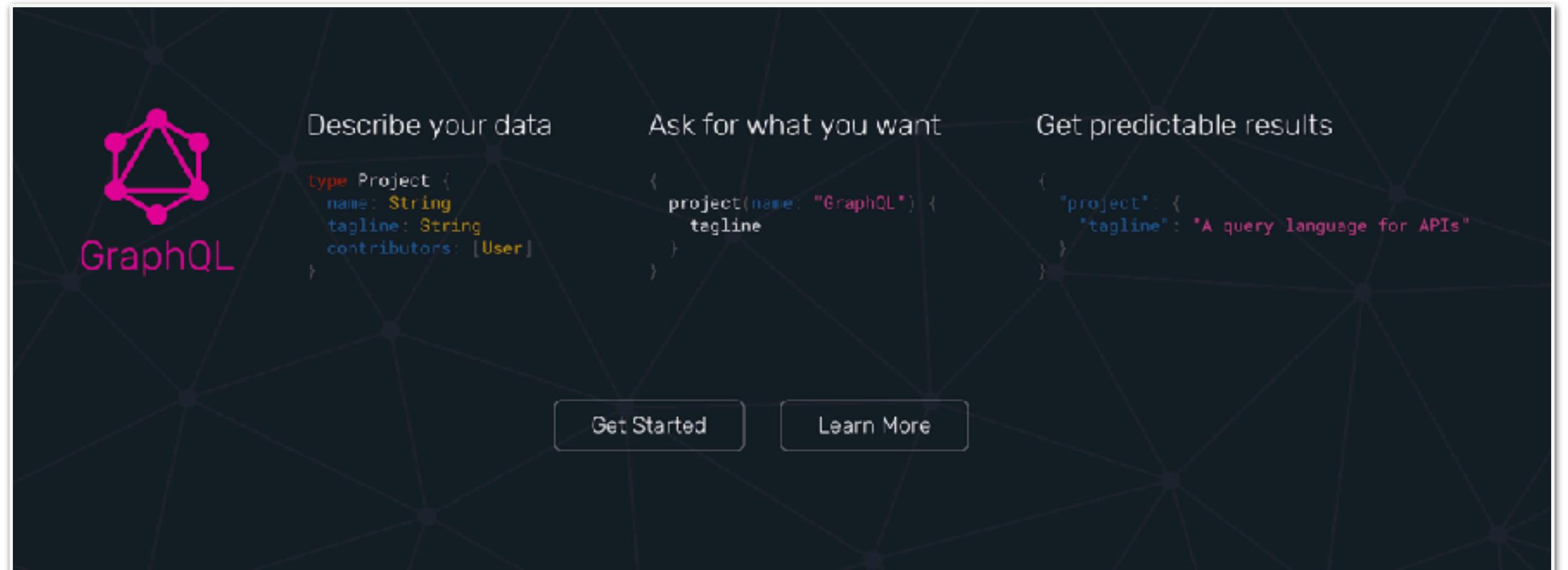
A first step in accommodating more than one type of UI is normally

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

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

@samnewman

GRAPHQL



The diagram illustrates the GraphQL ecosystem. At the top left is the GraphQL logo (a hexagon of nodes). To its right, three main concepts are shown: "Describe your data" (with a schema snippet), "Ask for what you want" (with a query snippet), and "Get predictable results" (with a response snippet). Below these concepts are two buttons: "Get Started" and "Learn More".

GraphQL

Describe your data

```
Type Project {  
  name: String  
  tagline: String  
  contributors: [User]  
}
```

Ask for what you want

```
{  
  project(name: "GraphQL") {  
    tagline  
  }  
}
```

Get predictable results

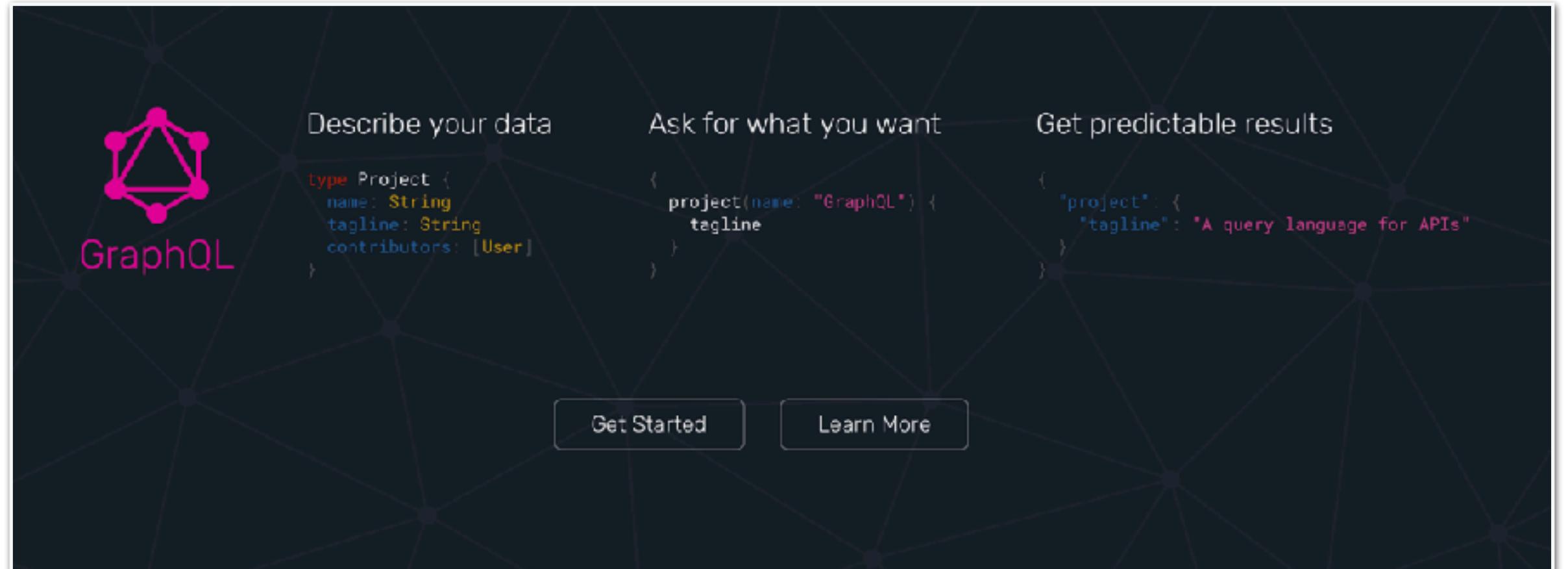
```
{  
  "project": {  
    "tagline": "A query language for APIs"  
  }  
}
```

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



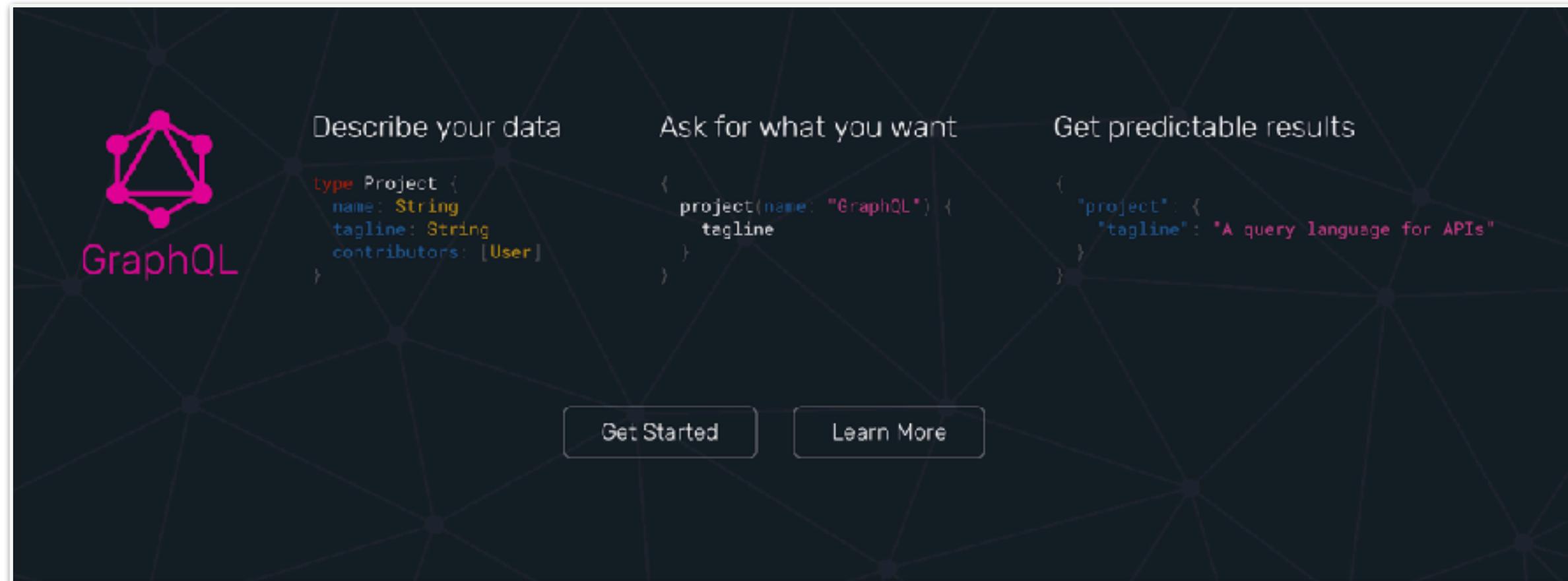
A query language for your API

GraphQL is a query language for APIs and a runtime for fulfilling those queries with your existing data. GraphQL provides a complete and

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

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

GRAPHQL



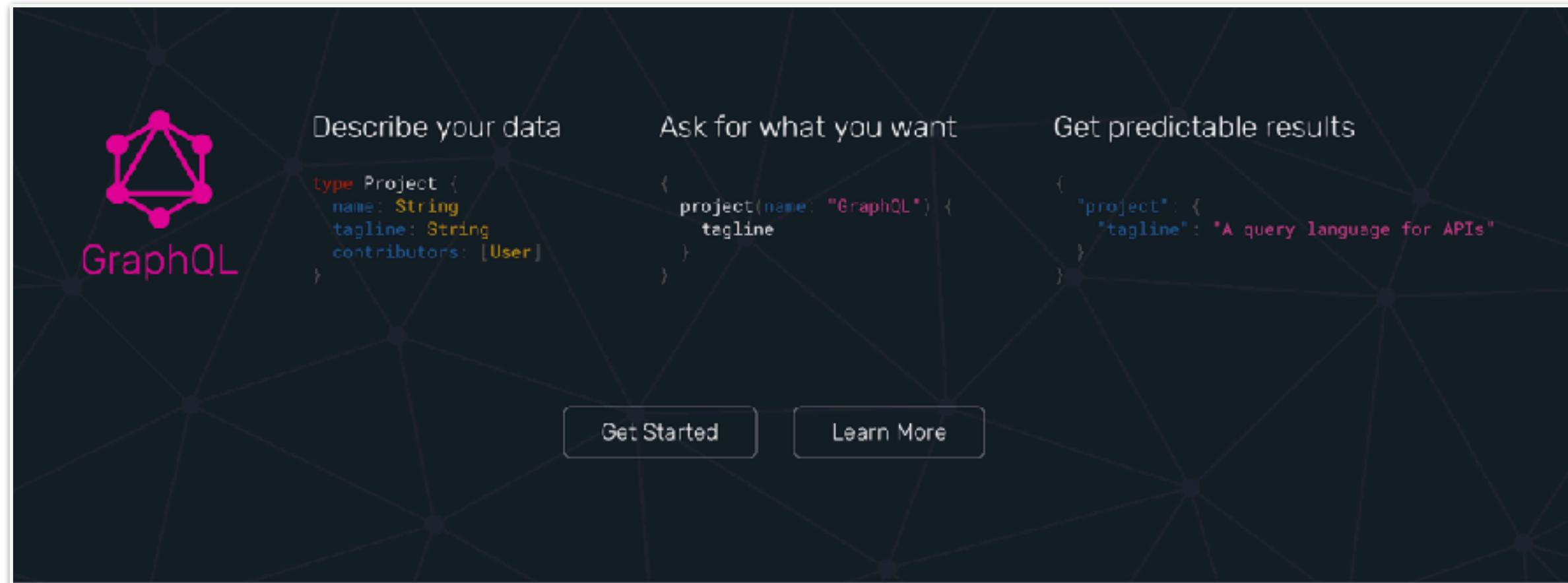
A query language for your API

GraphQL is a query language for APIs and a runtime for fulfilling those queries with your existing data. GraphQL provides a complete and

```
{  
  hero {  
    name  
    # Queries can have comments!  
    friends {  
      name  
    }  
  }  
}  
  
{  
  "data": {  
    "hero": {  
      "name": "R2-D2",  
      "friends": [  
        {  
          "name": "Luke Skywalker"  
        },  
        {  
          "name": "Han Solo"  
        },  
        {  
          "name": "Leia Organa"  
        },  
        {  
          "name": "Obi-Wan Kenobi"  
        },  
        {  
          "name": "Yoda"  
        }  
      ]  
    }  
  }  
}
```

Allows the client to specify queries dynamically

GRAPHQL



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

```
{  
  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

GRAPHQL CAVEATS

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



CONTENTS

Benefits

[Incremental upgrades](#)

[Simple, decoupled codebases](#)

[Independent deployment](#)

[Autonomous teams](#)

[In a nutshell](#)

The example

[Integration approaches](#)

[Server-side template composition](#)

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

<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

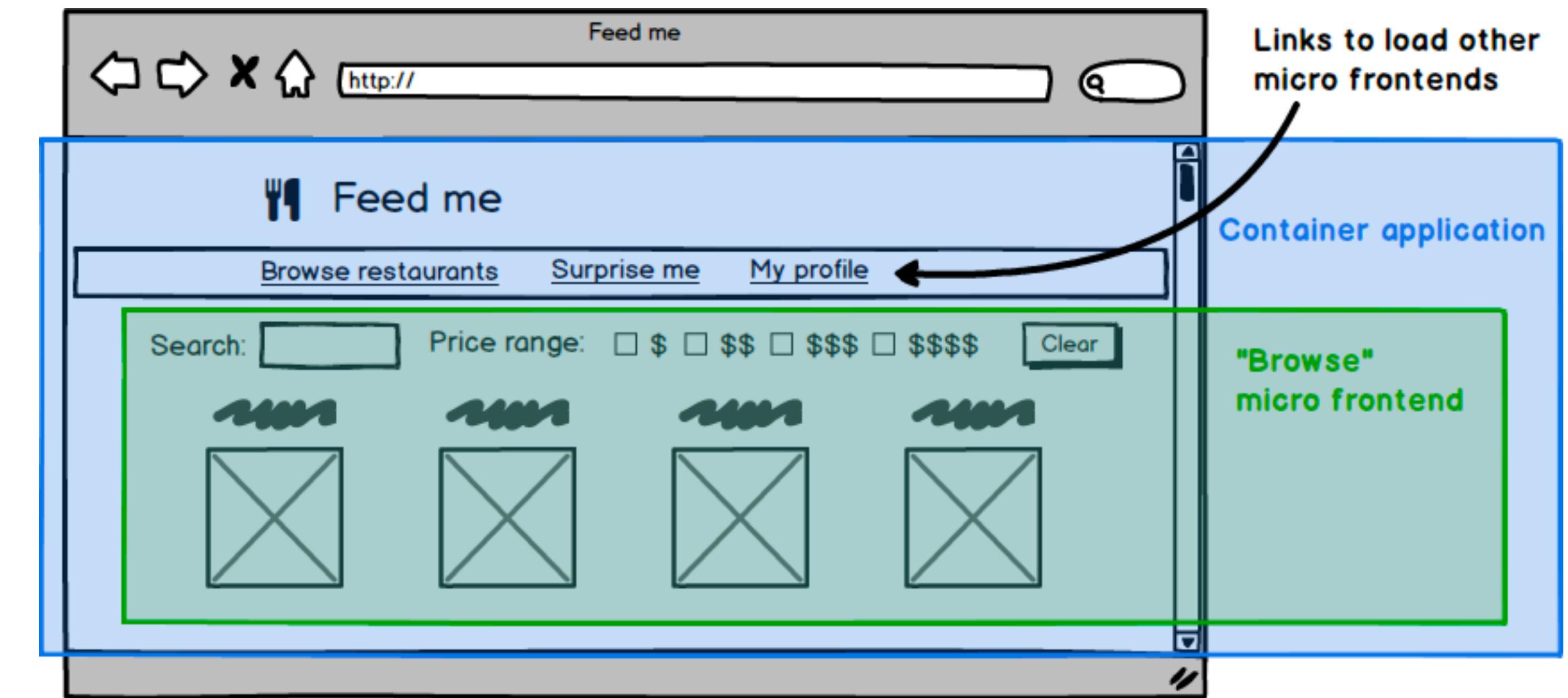
Cam Jackson is a full-stack web developer and consultant at ThoughtWorks, with a particular interest in how large organisations scale their frontend development process and practices. He has worked with clients across multiple

CONTENTS

Benefits

- Incremental upgrades
- Simple, decoupled codebases
- Independent deployment
- Autonomous teams
- In a nutshell
- The example
- Integration approaches
- Server-side template composition

<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

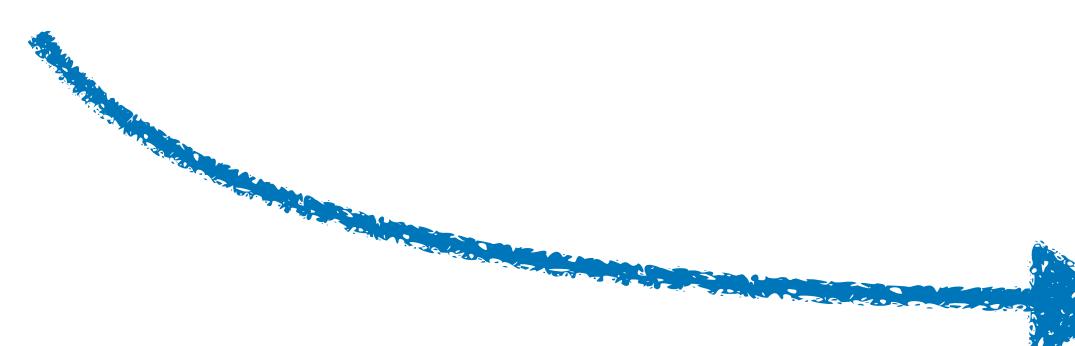
Even small changes can be broken up into smaller releases

Decomposition doesn't need to stop at the backend

OTHER COURSES



<http://bit.ly/snewman-olt>



THANKS!

Sam Newman & Associates



Home About Offerings Events **Writing** Contact

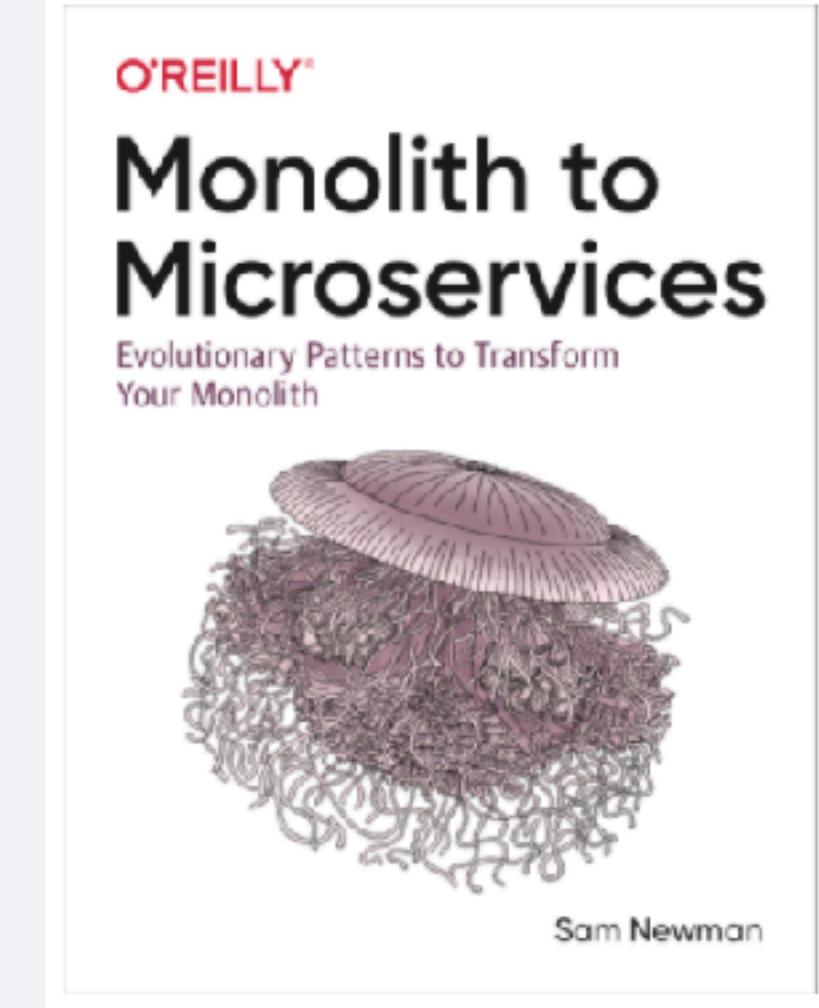
Monolith To Microservices.

Released on 2019-09-04 by O'Reilly

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

If you're interested in migrating your existing systems to microservice architectures, or are struggling with services that are too big, then this book is for you

[→ Find Out More](#)



<https://samnewman.io/>

@samnewman