

SERVICE-ORIENTED ARCHITECTURE

a practical introduction to the
service-oriented mind set

Thorben Schröder
FH Salzburg 2014

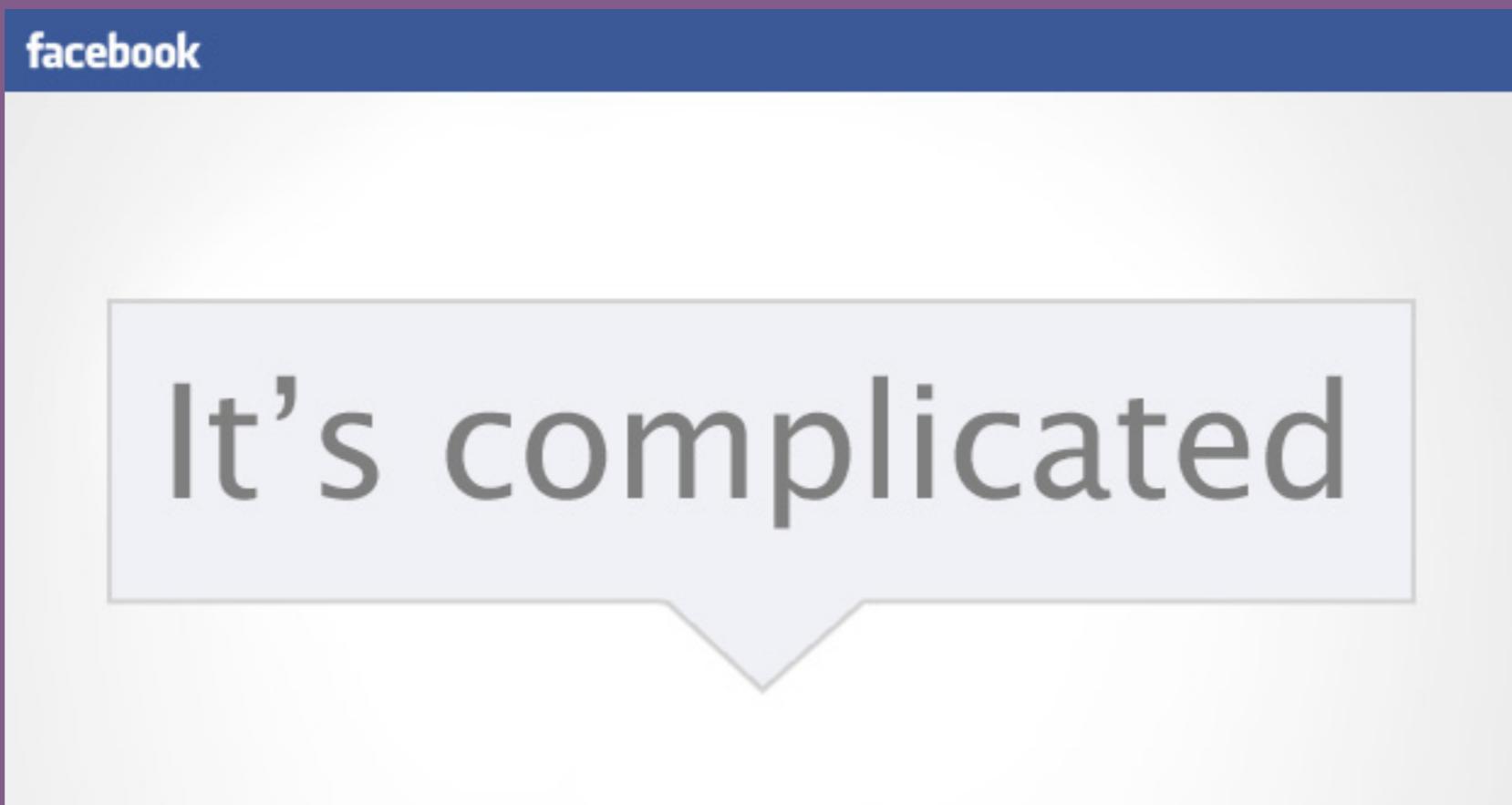
HI THERE

I am Thorben.

Reach me at info@thorbenschroeder.de

Or find me on Twitter, GitHub & Co as @walski.

WHAT I DO



WHAT I DO



WHAT DO YOU DO?

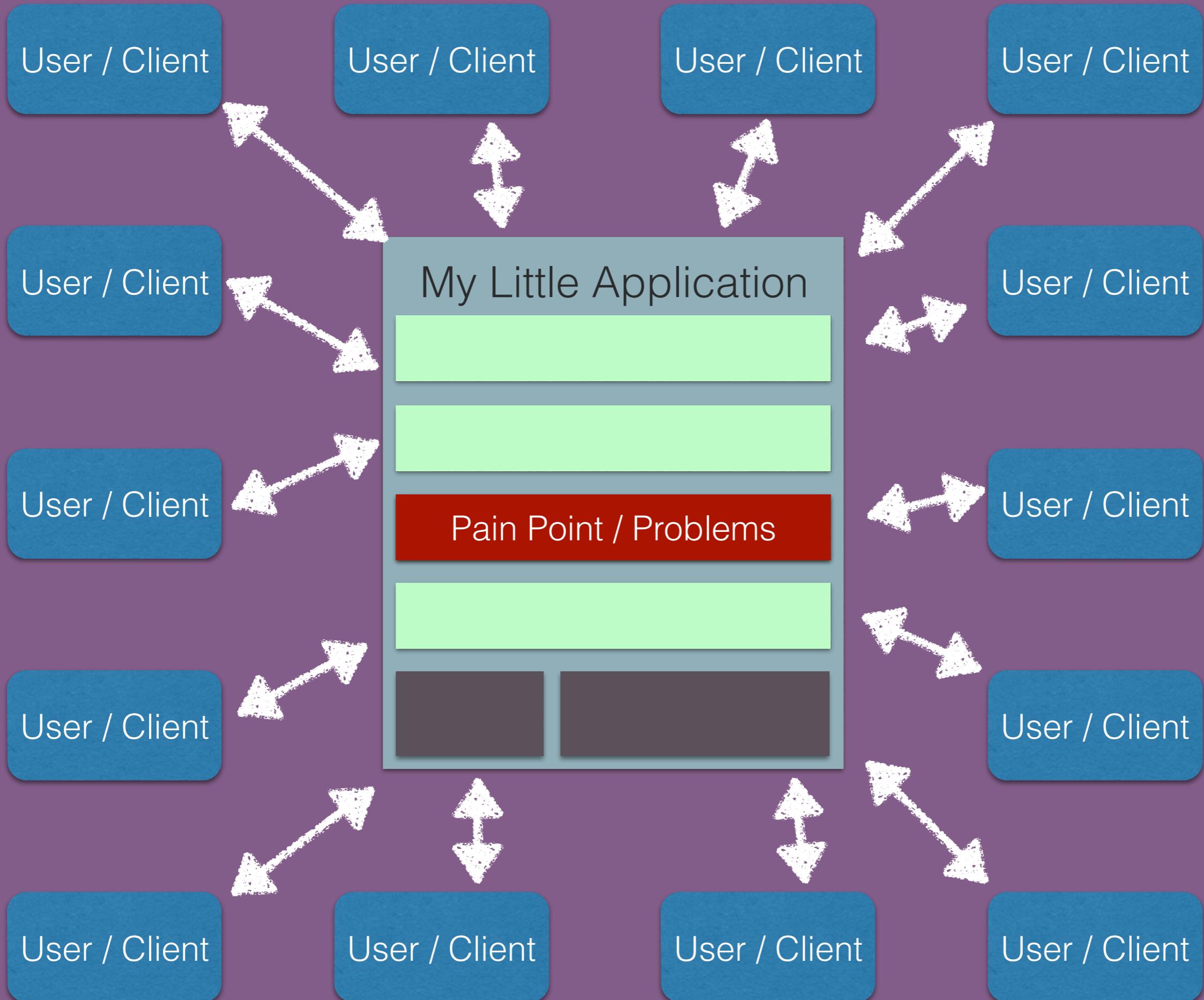
GOOD TO KNOW

- **Basics**
 - **git, (a little) UNIX, vocabulary (deployments, ...)**
- **Ruby (≥ 2.0)**
- **Ruby ecosystem**
 - **RubyGems, bundler, RVM**
- **Test driven methodology**

AGENDA

- 1. Intro to SOA**
- 2. Modular Web Applications**
- 3. Documentation**
- 4. Testing**
- 5. Development Setups**
- 6. Service Authentication**
- 7. Inter Service Communication**
- 8. Deployments**
- 9. Browsers as Clients**

SERVICES!



My Little This

Another Thing

Pain Point / Problems

My Little Application

Pain Point / Problems

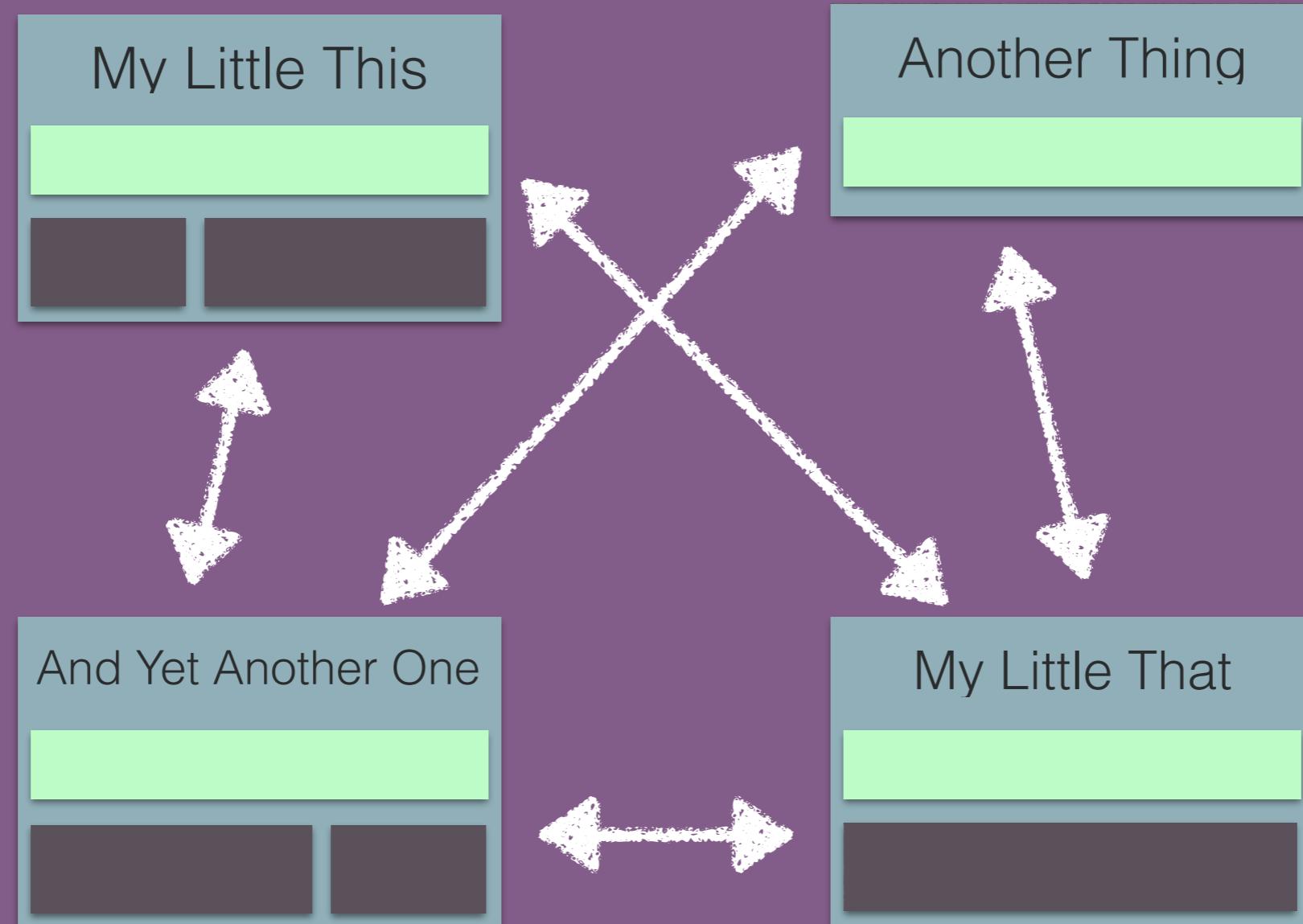
And Yet Another One

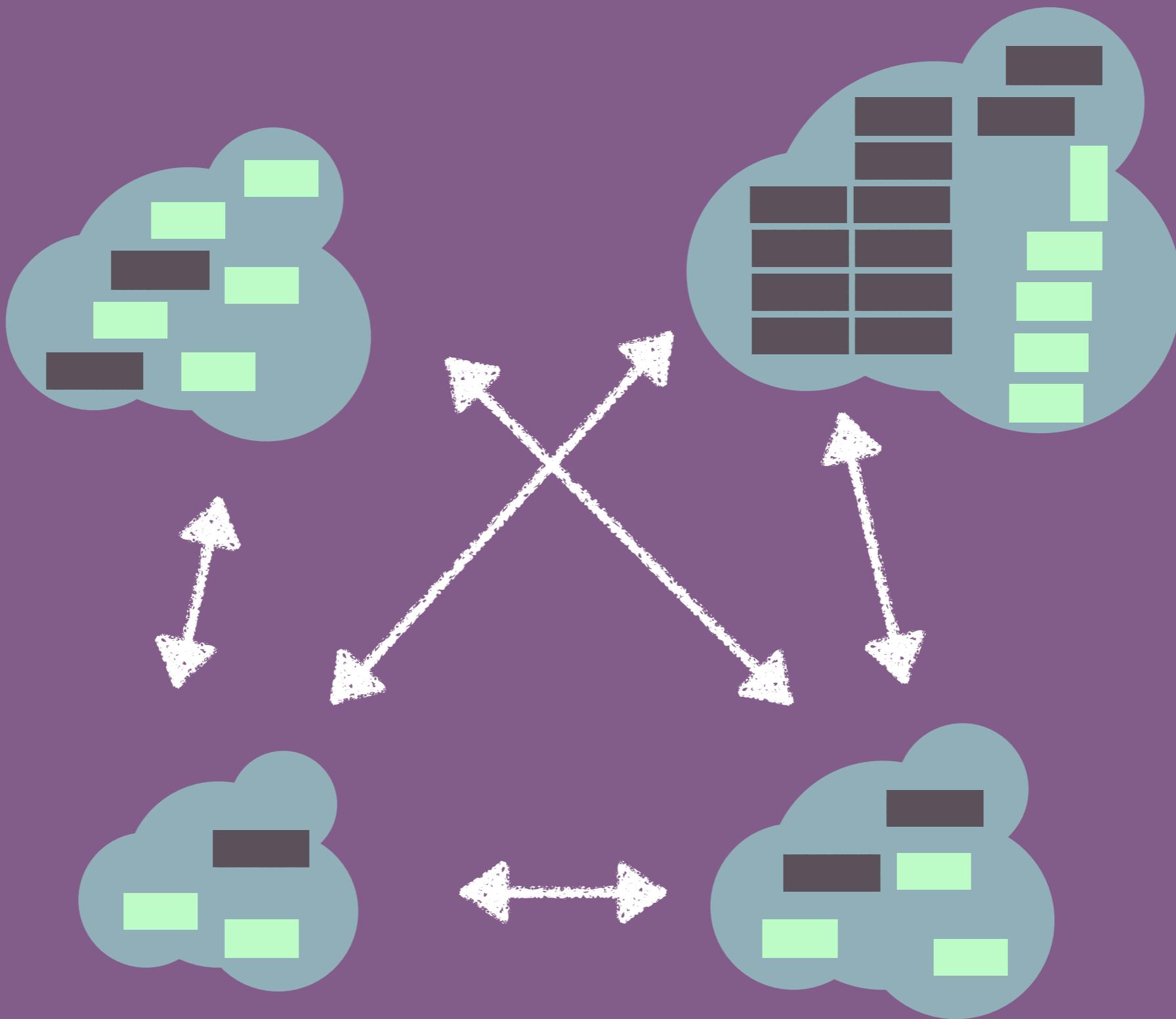
My Little That

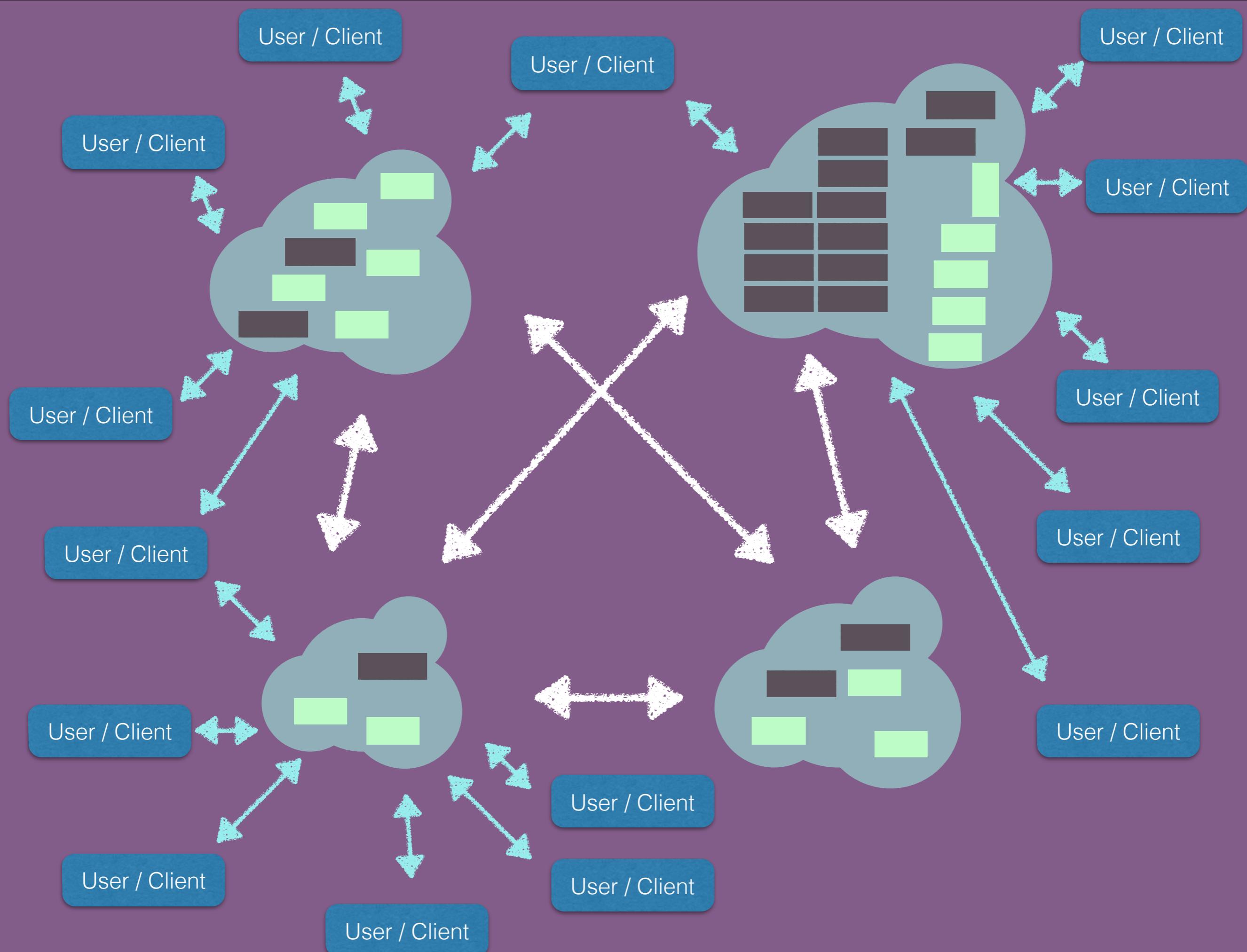


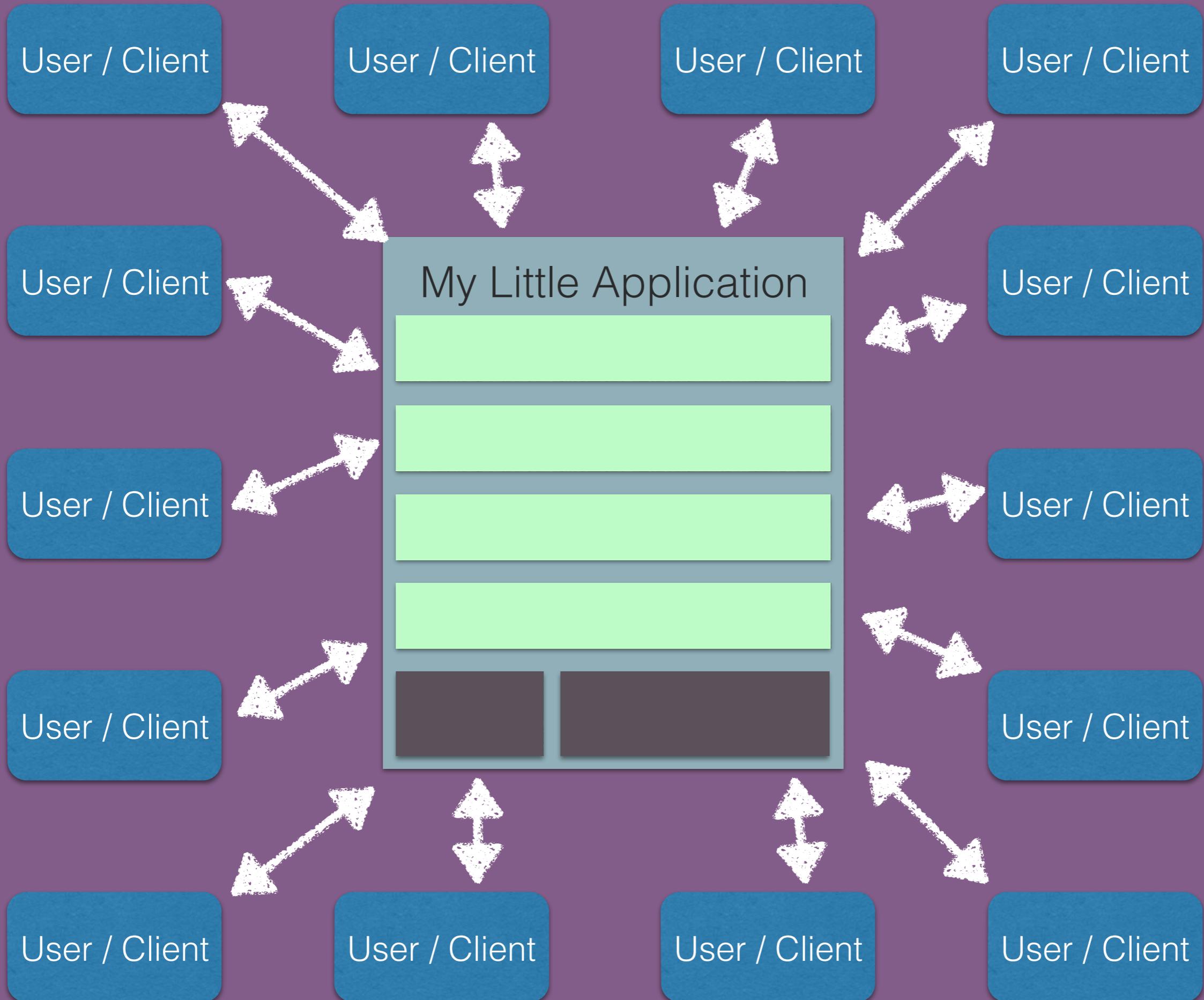
WHY?

- Applications' requirements keep growing
- Big applications are hard to maintain & debug
- Splitting up helps to focus!
- Bonus points: An internal API can be used externally at very little extra cost









SERVICE ORIENTED ARCHITECTURE IN BIG STROKES

- Move away from monolithic applications
- Change to clusters of communicating apps
- UNIX philosophy:
“Make each program do one thing well.”

HISTORY

- term coined by 1996
- got popular with the web as it's foundation by around 2001
- not very well defined

FOUR TENETS OF SOA

Don Box of Microsoft:

1. **Boundaries are explicit**
2. **Services are autonomous**
3. **Services share schema and contract, not class**
4. **Service compatibility is based on policy**

DOESN'T SAY MUCH...

BUT

My Little Application

Shop

Billing

Warehouse

Users

Redis

Database

My Billing

Billing

Database

My Shop

Shop

Warehouse

Redis

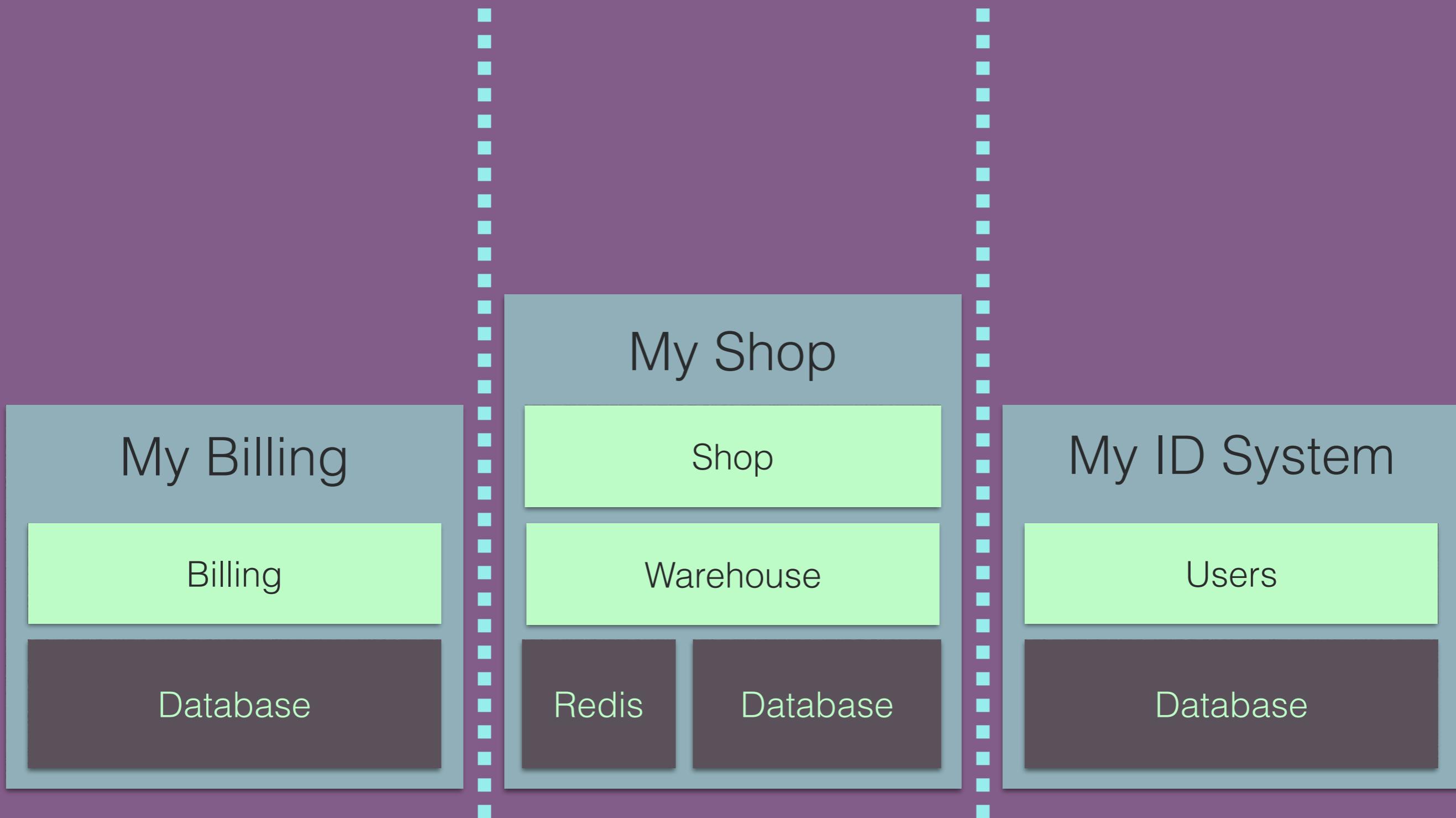
Database

My ID System

Users

Database

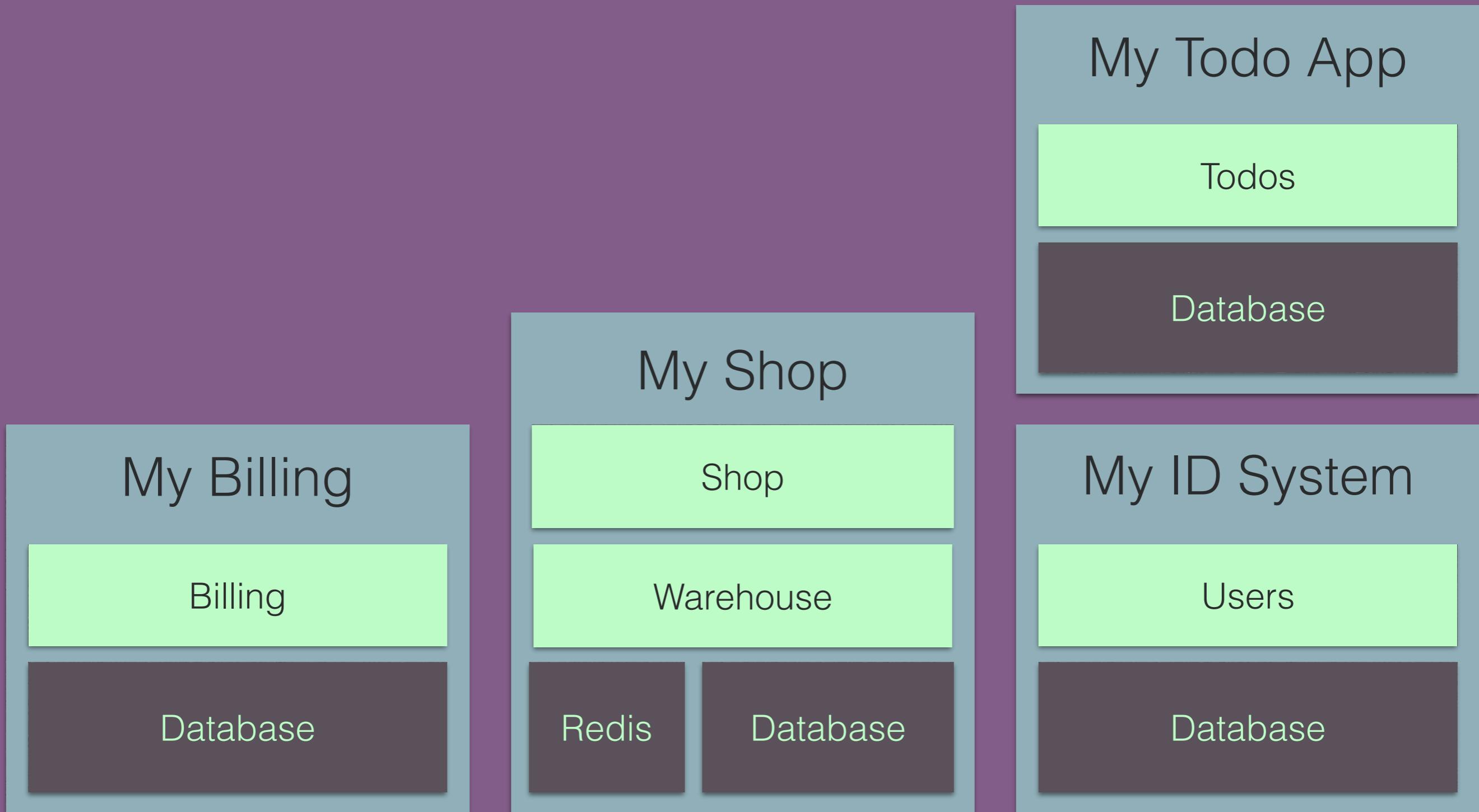
1. BOUNDARIES ARE EXPLICIT



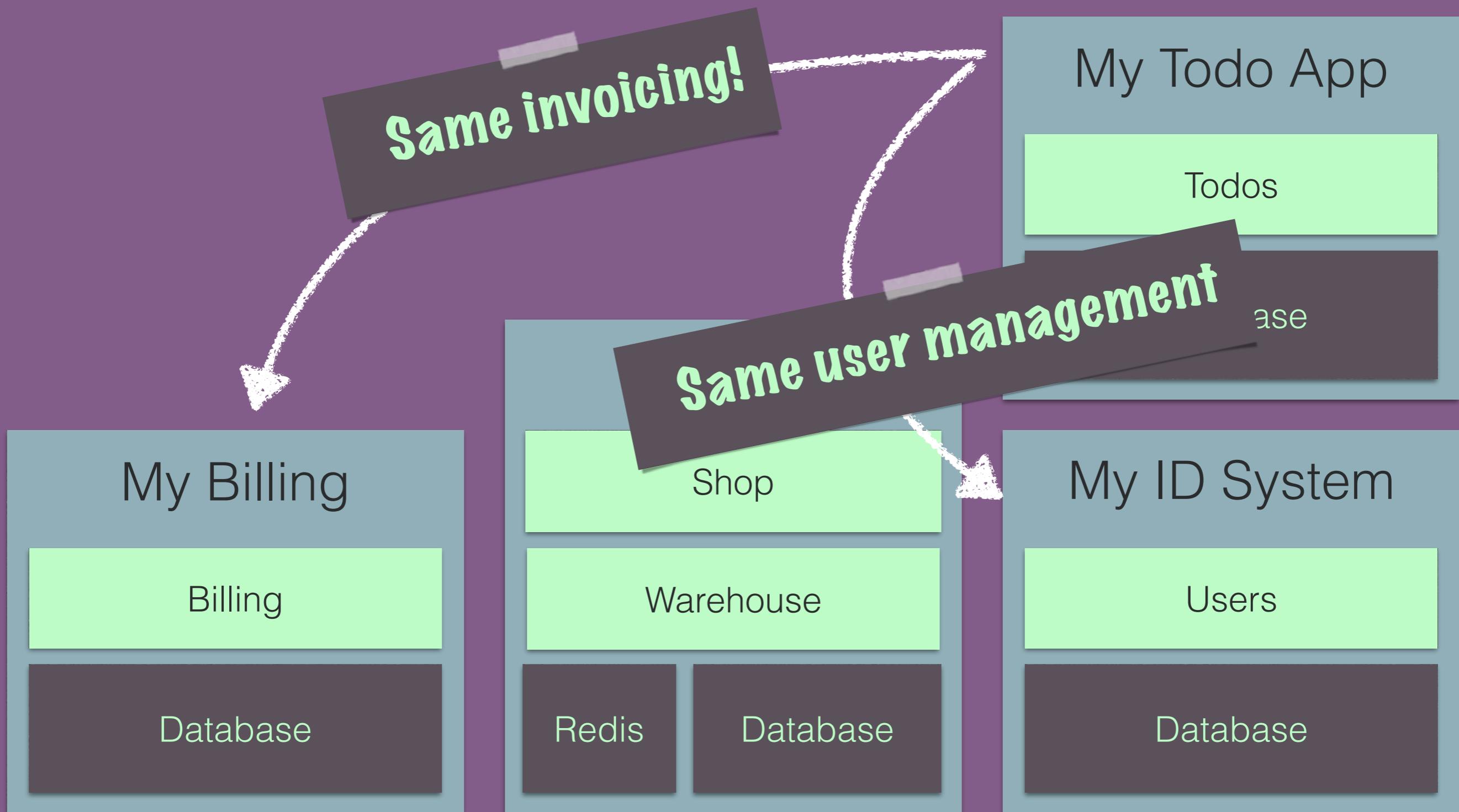
2. SERVICES ARE AUTONOMOUS



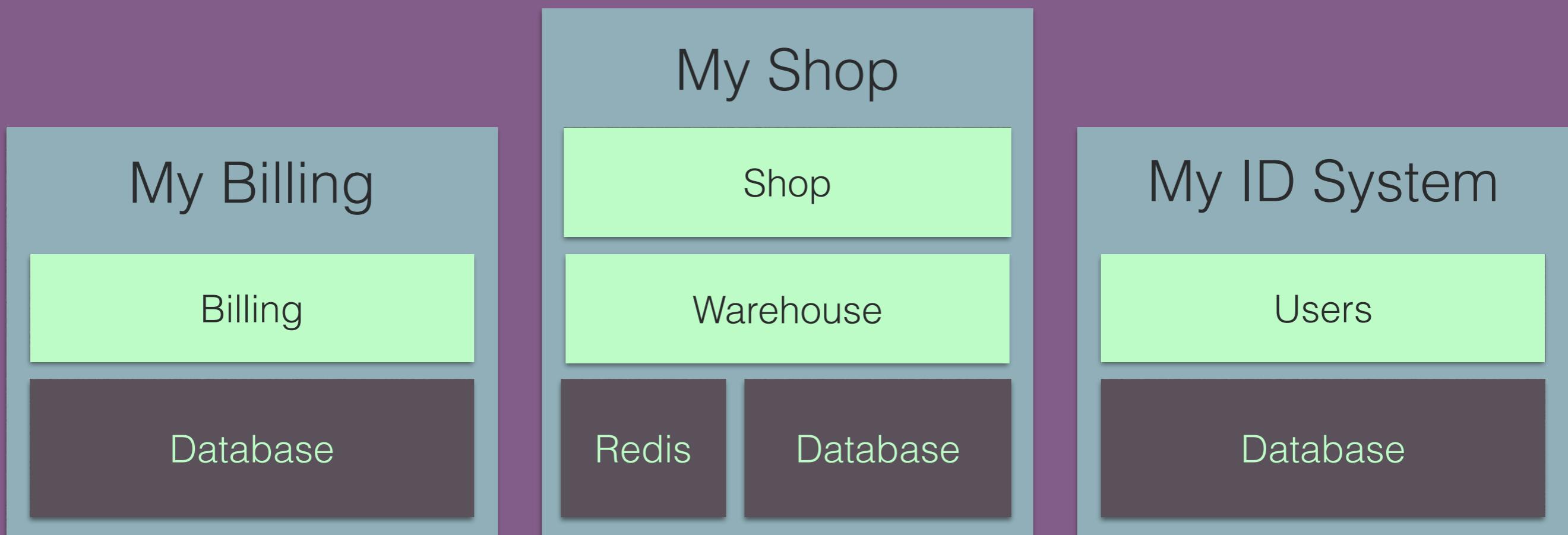
2. SERVICES ARE AUTONOMOUS



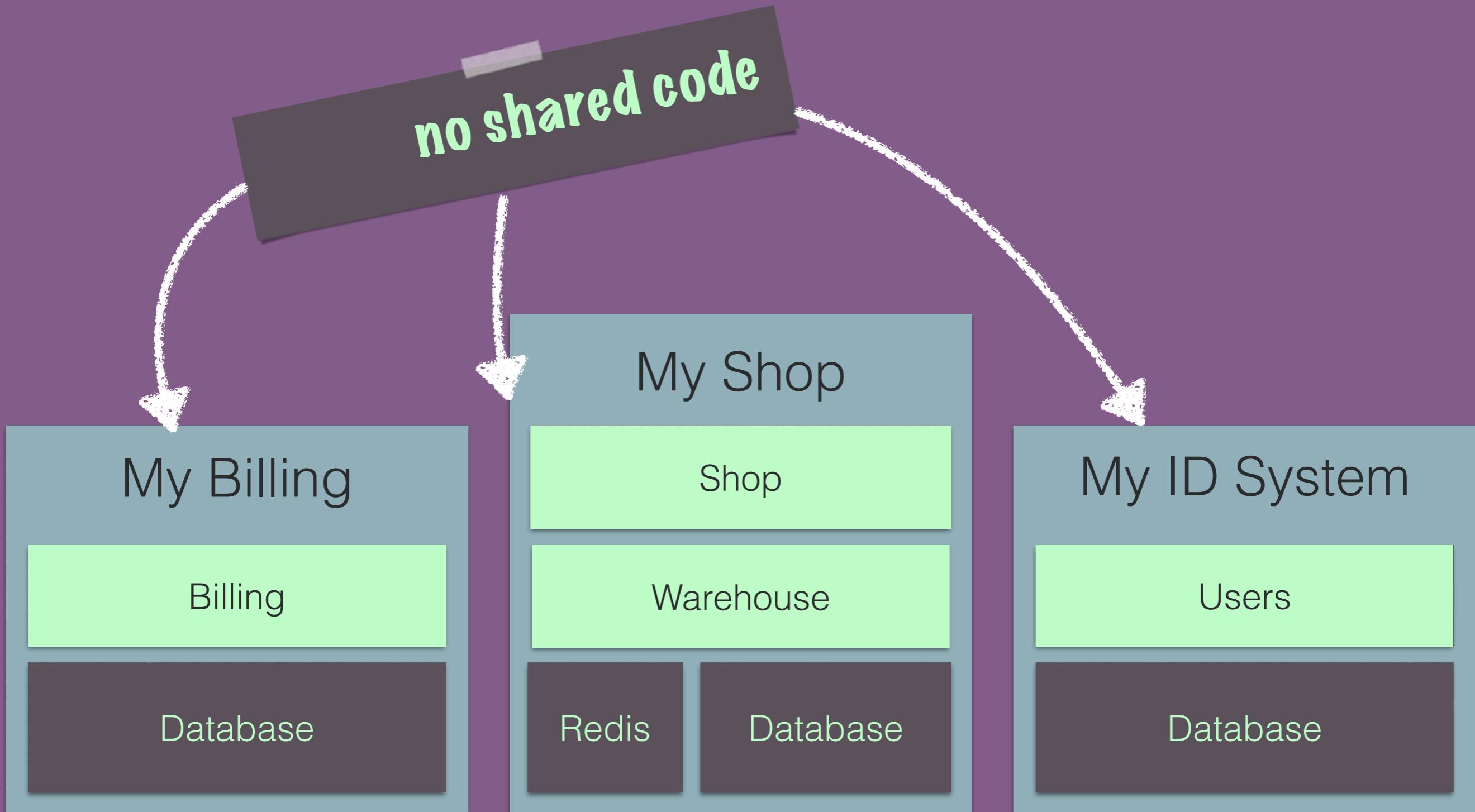
2. SERVICES ARE AUTONOMOUS



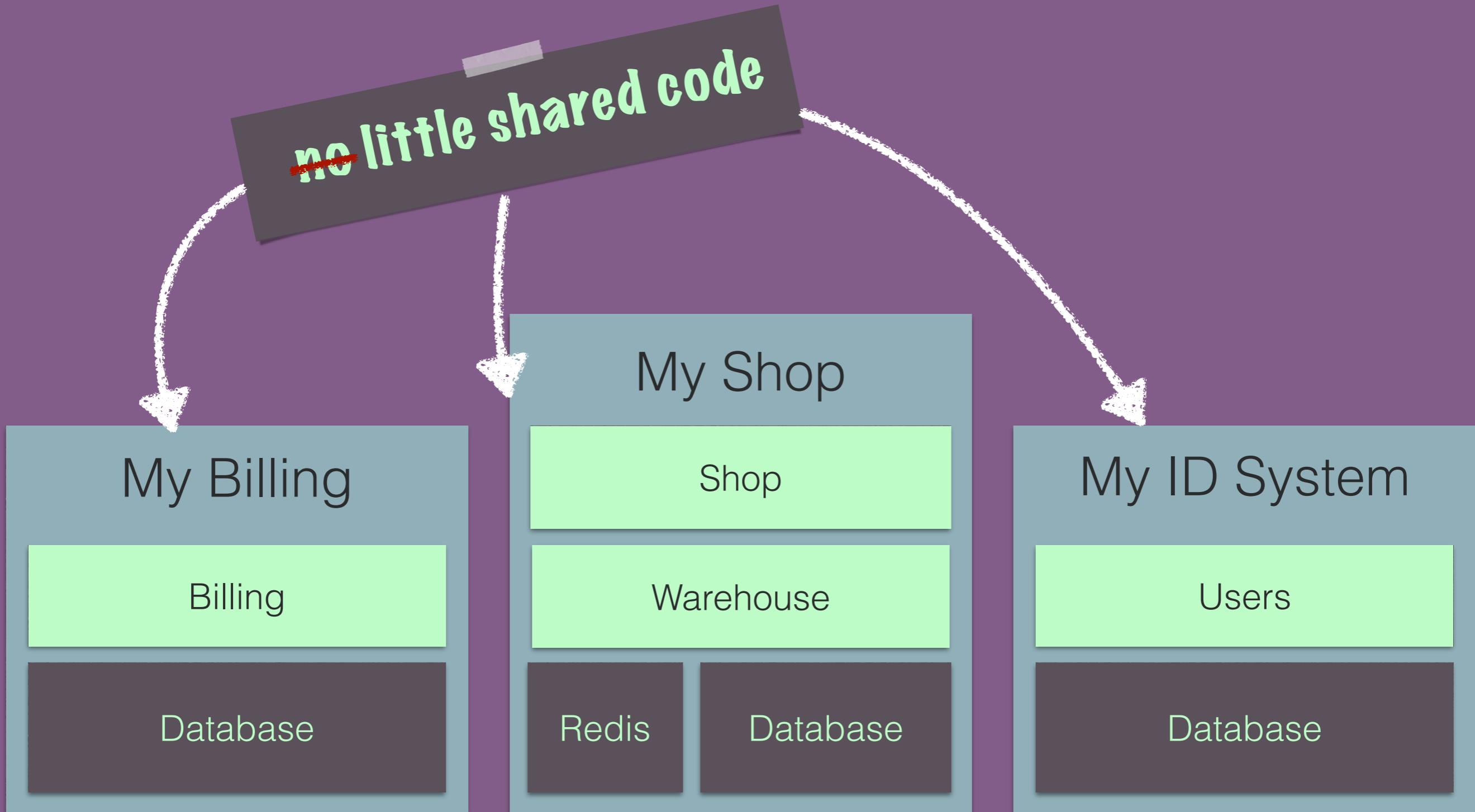
3. SERVICES SHARE SCHEMA AND CONTRACT, NOT CLASS



3. SERVICES SHARE SCHEMA AND CONTRACT, NOT CLASS



3. SERVICES SHARE SCHEMA AND CONTRACT, NOT CLASS



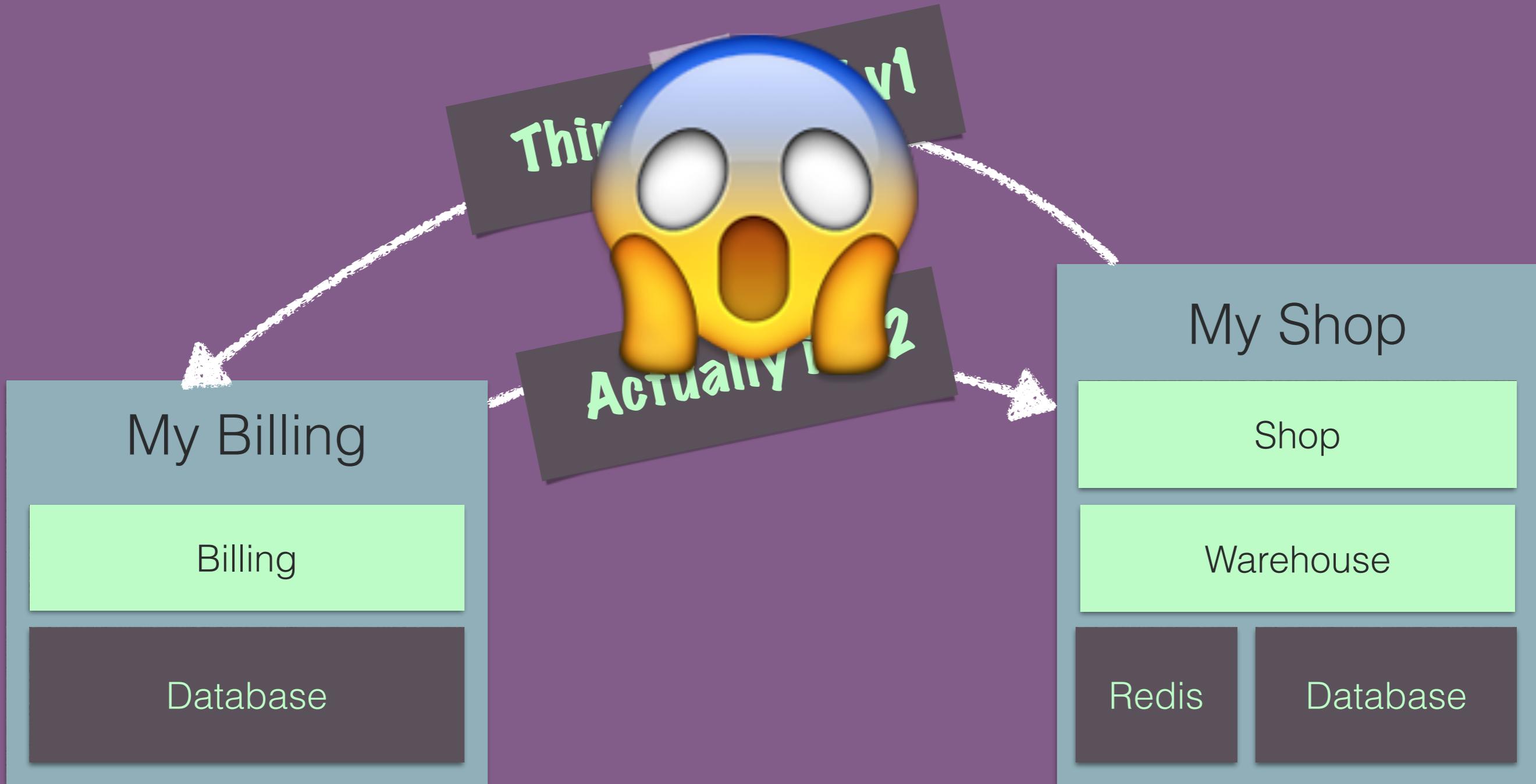
4. SERVICE COMPATIBILITY IS BASED ON POLICY



4. SERVICE COMPATIBILITY IS BASED ON POLICY



4. SERVICE COMPATIBILITY IS BASED ON POLICY



FOUR TENETS OF SOA

Thorben's version:

1. Think in autonomous apps
2. Make them communicate by messages only
3. Integration test everything

MODULAR WEB APPLICATIONS

THE TWELVE-FACTOR APP



<http://12factor.net/>

THE TWELVE-FACTOR APP

- “**a methodology for building software-as-a-service apps**”
- **made for “cloud” deployments**
- **scalability in mind**
- **very practical approach to SOA development**

THE TWELVE-FACTOR APP

I. Codebase

II. Dependencies

III. Config

IV. Backing Services

V. Build, release, run

VI. Processes

VII. Port binding

VIII. Concurrency

IX. Disposability

X. Dev/prod parity

XI. Logs

XII. Admin processes

THE TWELVE-FACTOR APP

I. **Codebase**

Use git

II. **Dependencies**

III. **Config**

IV. **Backing Services**

V. **Build, release, run**

VI. **Processes**

VII. **Port binding**

VIII. **Concurrency**

IX. **Disposability**

X. **Dev/prod parity**

XI. **Logs**

XII.

THE TWELVE-FACTOR APP

I. Codebase

II. Dependencies

III. Config

IV. Backing Services

V. Build, release, run

VI. Processes

VII. Port binding

Use bundler

Concurrency

IX. Disposability

X. Dev/prod parity

XI. Logs

XII.

THE TWELVE-FACTOR APP

I. Codebase

II. Dependencies

III. Config

IV. Backing Services

V. Build, release, run

VI. Processes

VII. Port binding

VIII. Concurrency

Sposability

X. Dev/prod parity

XI. Logs

XII.

Configuration is part
of the environment

THE TWELVE-FACTOR APP

I. Codebase

II. Dependencies

III. Config

IV. Backing Services

V. Build, release, run

VI. Processes

VII. Port binding

VIII. Concurrency

IX. — Disposability

**Make services
configurable**

XI. Logs

XII.

THE TWELVE-FACTOR APP

I. Codebase

II. Dependencies

III. Config

IV. Backing Services

V. Build, release, run

VI. Processes

VII. Port binding

VIII. Concurrency

IX. Disposability

X. Dev/prod parity

Turn Code & Config
into a deployment

XII.

THE TWELVE-FACTOR APP

I. Codebase

II. Dependencies

III. Config

IV. Backing Services

V. Build, release

VI. Processes

VII. Port binding

VIII. Concurrency

IX. Disposability

X. Dev/prod parity

Processes are stateless

Any “state” belongs in
a backing service

THE TWELVE-FACTOR APP

Apps are self-contained

**Communicate over a
configurable port**

III. Config

IV. Backing Services

V. Build, release, run

VI. Processes

Port binding

II. Concurrency

Disposability

X. Dev/prod parity

XI. Logs

XII.

THE TWELVE-FACTOR APP

I. Codebase

II. Dependencies

III. Configuration

IV. Backing Services

V. Build, release, run

VI. Processes

VII. Port binding

III. Concurrency

IX. Disposability

X. Dev/prod parity

XI. Logs

XII.

*Scale by processes
Define process types*

THE TWELVE-FACTOR APP

I. Codebase

II. Dependencies

III. Config

IV. Backing Services

V. Build, release, run

VI. Processes

VII. Port binding

VIII. Concurrency

Disposability

X. Dev/prod parity

XI. Logs

XII.

Quick startup
Graceful shutdown

THE TWELVE-FACTOR APP

I. Codebase

II. Dependencies

III. Configuration

IV. Backing services

V. Build, release, run

VI. Processes

VII. Port binding

VIII. Concurrency

IX. Disposability

X. Dev/prod parity

XI. Logs

XII.

Development == Production
Same backing services
Continuous deployments

THE TWELVE-FACTOR APP

I. Codebase

II. Dependencies

III. Config

IV. Backing Services

V. Build, release

VI. Processes

VII. Port binding

VIII. Concurrency

IX. Disposability

X. Dev/prod parity

XI. Logs

XII.

Log to STDOUT

THE TWELVE-FACTOR APP

I. Codebase

II. Dependencies

III. Config

IV. Backing services

V. Build, release, and deployment keys

VI. Processes

VII. Port binding

VIII. Concurrency

IX. Disposability

Dev/prod parity

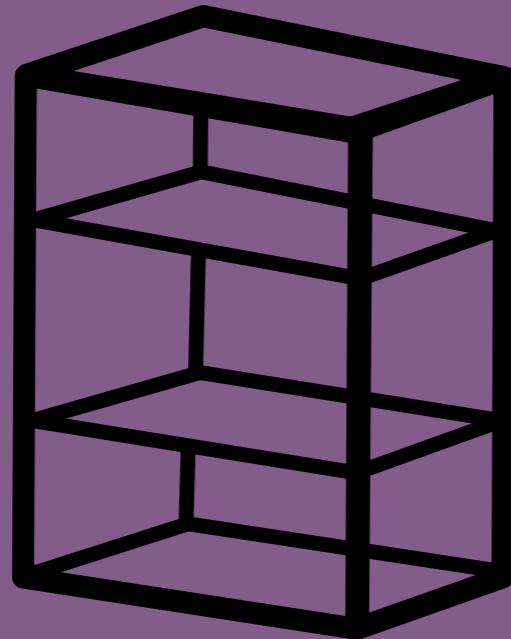
Logs

Admin processes

*Same execution as
app's processes*

*REPL should be part
of the app*

A RACK 101



rack
powers web applications

<http://rack.github.io/>

WHAT IS RACK?

Rack provides a minimal interface between web servers supporting Ruby and Ruby frameworks.

WHAT IS RACK?

- **Rack Application**
- **Rack Middleware**
- **That is it**

WHAT IS RACK?

- Hash in
- Array out

WHAT IS RACK?

- Hash in
- Array out

Rack environment

```
{  
  'REQUEST_METHOD' => 'POST',  
  'PATH_INFO' => '/api/v1/todos',  
  #...  
  'HTTP_AUTHORIZATION' => 'Basic bGFzdDp1bmIjb3Ju',  
}
```

WHAT IS RACK?

in
HTTP Header

- Array out

```
{  
  'REQUEST_METHOD' => 'POST',  
  'PATH_INFO' => '/api/v1/todos',  
  ...  
  'HTTP_AUTHORIZATION' => 'Basic bGFzdDp1bmljb3Ju',  
}
```

Rack environment

WHAT IS RACK?

- Hash in
- Array out

Iterable
to allow
streaming

Rack response

```
[200, {"Content-Type" => "text/plain"}, ['Body']]
```

HTTP Status

HTTP Response Header

Response Body

WHAT IS RACK?

- Hash in
- Array out

Iterable
to allow
streaming

Rack response

```
[200, {"Content-Type" => "text/plain"}, ['Body']]
```

HTTP Status

HTTP Response Header

Response Body

A RACK APP

- must respond to “call” with the Rack environment as the only argument

A RACK APP

Rack environment

```
class MyApp
  def call(env)
    [200, {'Content-Type' => 'text/plain'}, ['Hey there']]
  end
end
```

Rack response array

BUT WAIT...

CALL YOU SAID?

```
$ ri Proc#call
```

```
= Proc#call
```

```
(from ruby core)
```

```
-----  
prc.call(params,...)    -> obj  
prc[params,...]         -> obj  
prc.(params,...)        -> obj  
-----
```

Invokes the block, setting the block's parameters to the values in params using something close to method calling semantics....

PROCS ARE RACK APPS!

ALSO A RACK APP

Rack environment

```
-> (env) { [200, { 'Content-Type' => 'text/plain' }, ['...']] }
```

Rack response array

CHALLENGE ACCEPTED



SETTING UP OUR LAB

```
$ git clone https://github.com/walski/soa-workshop.git
$ cd soa-workshop/box
$ vagrant up
$ vagrant ssh
```

```
vagrant@...$ cd /vagrant/challenges/01
vagrant@...$ bundle install
vagrant@...$ bundle exec rake
```

SETTING UP OUR LAB

- **edit code on your machine (box/challenges)**
- **run the specs in Vagrant**

CHALLENGE ACCEPTED



RACK'S MIGHTY FRIENDS

RACK::REQUEST

RACK::RESPONSE

RACK::BUILDER

```
class MyAppA
  def call(env)
    [200, {'Content-Type' => 'text/plain'}, ['Hey A']]
  end
end

class MyAppB
  def call(env)
    [200, {'Content-Type' => 'text/plain'}, ['Hey B']]
  end
end

app = Rack::Builder.new {
  map("/a") { run MyAppA.new }
  map("/b") { run MyAppB.new }
  run -> (env) {[200, {'Content-Type' => 'text/plain'},
  ['What?']]}
```

RACKUP

CONFIG.RU

1. Create a rackup config

```
Bundler.require  
require 'myapp'
```

```
run MyApp.new
```

2. Run your web app

```
$ rackup
```

RTFM

rack.rubyforge.org/doc/README_rdoc.html

CHALLENGE ACCEPTED



A SINATRA PRIMER



WHAT IS SINATRA?

- **simple web framework**
- **routing / templates / “all we need”**
- **Sinatra apps are Rack apps**
- **good example**

WHAT IS SINATRA?

- **simple web framework**
- **routing / templates / “all we need”**
- **Sinatra apps are Rack apps**
- **good example**

WHAT IS SINATRA?

```
require 'sinatra/base'

class MyApp < Sinatra::Base
  get '/hossa' do
    'cool!'
  end
end
```

**SINATRA CAN DO SO MUCH MORE.
BUT SERIOUSLY...**

RTFM

www.sinatrarb.com/intro.html

CHALLENGE ACCEPTED



A RACK MIDDLEWARE

- **is a wrapper around an app**
- **gets the app passed as the only argument to its constructor**
- **must respond to “call” with the Rack environment as the only argument**

A RACK MIDDLEWARE

```
class MyMiddleware
  def initialize(app)
    @app = app
  end

  def call(env)
    if env['PATH_INFO'] == '/yay'
      [200, {'Content-Type' => 'text/plain'}, ['yay']]
    return
  end
  @app.call(env)
end
end
```

Parent Rack app



CHALLENGE ACCEPTED



I HATE RUBY!

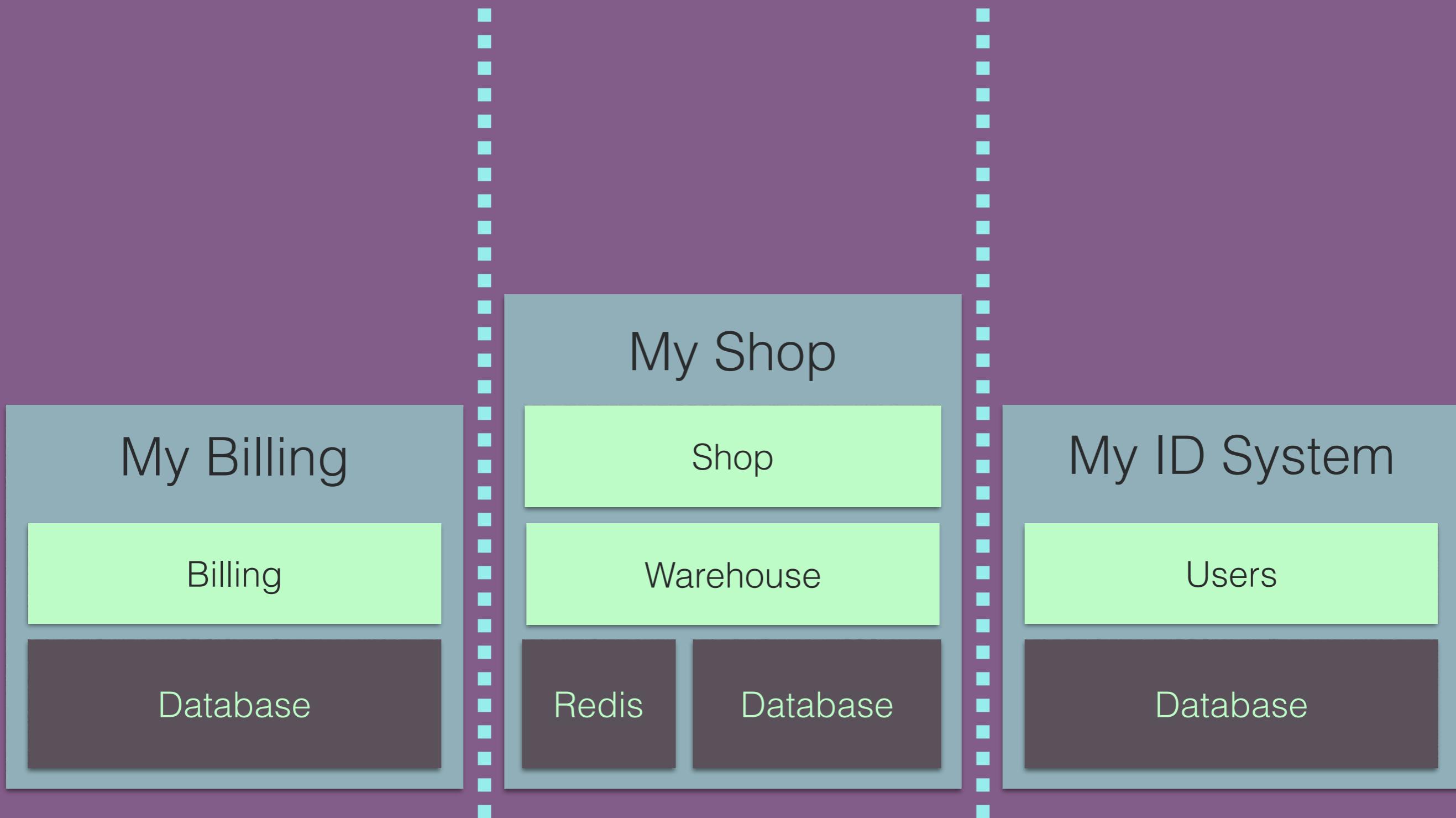
- **Don't you worry:**
 - **Python WSGI**
 - **Perl Web Server Gateway Interface**
 - **JavaScript Gateway Interface**
 - ...

SOA

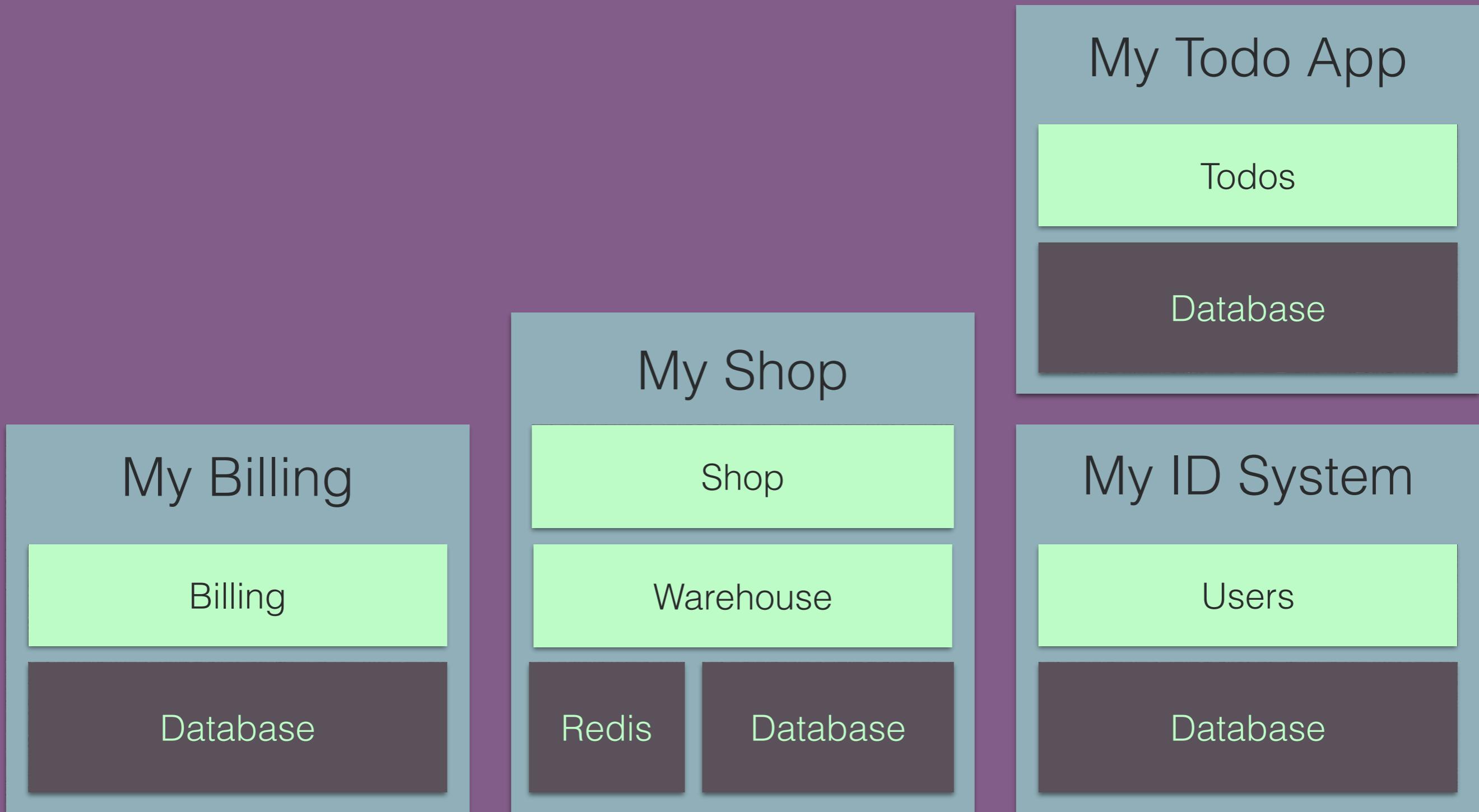
DOCUMENTATION

QUICK RECAP

1. BOUNDARIES ARE EXPLICIT



2. SERVICES ARE AUTONOMOUS



3. SERVICES SHARE SCHEMA AND CONTRACT, NOT CLASS



4. SERVICE COMPATIBILITY IS BASED ON

4. SERVICE

COMPATIBILITY IS



4. SERVICE COMPATIBILITY IS BASED ON POLICY

DOCUMENT YOUR APIs

- APIs are your policy / contract
- Helps to frame your mindset

DOCUMENT YOUR APIs

- APIs are your policy / contract
- Helps to frame your mindset

README DRIVEN DEVELOPMENT

- **Write your Readme before writing code**
- **Before ANY code!**
- **Forces you to think what you are about to build**
- **Enables team discussions**

READMES IN SOA

- The Readme can be / include your API documentation
- Good for cross functional teams

WHAT NEEDS TO BE DOCUMENTED?

- Take advantage of the simplicity of HTTP:
 - Request
 - Response

WHAT NEEDS TO BE DOCUMENTED?

- Request
- Verb
- Path
- Headers
- Body

WHAT NEEDS TO BE DOCUMENTED?

- Response
- Status
- Headers
- Body
- One documented request might lead to several different responses depending on the request specifics (unauthenticated vs. authenticated requests...)

WHAT NEEDS TO BE DOCUMENTED?

- **Miscellaneous**
- **Versioning**
- **Authentication**
- **Pagination**
- ...

SAAAAAAAMPLES



developer.github.com/v3/

SAAAAAAAMPLES



<https://devcenter.heroku.com/articles/platform-api-reference>

HOW TO DOCUMENT?

- Write it yourself
- Generated from code / comments
- Generated from tests / specs

HOW TO DOCUMENT?

- Write it yourself
- Generated from code / comments
- Generated from tests / specs



No documentation
challenge :/

CHALLENGE DENIED.

SOA TESTING

WHY IS THIS EVEN A THING?

INTEGRATION TESTS

vs.

UNIT TESTS

Monolithic apps
can be tested with
unit tests

SOA apps can't

They rely on the
integration with
each other!

YOUR OPTIONS

1. Have a mock for every single service
2. Find a way to run the real systems

MANAGE DEPENDENCIES

Each app is a gem

CREATING GEMS

```
$ bundle gem my_gem
  create  my_gem/Gemfile
  create  my_gem/Rakefile
  create  my_gem/LICENSE.txt
  create  my_gem/README.md
  create  my_gem/.gitignore
  create  my_gem/hello-world.gemspec
  create  my_gem/lib/my_gem.rb
  create  my_gem/lib/my_gem/version.rb
```

CREATING GEMS

Pretty much empty

```
$ bundle gem my_gem
  create  my_gem/Gemfile
  create  my_gem/Rakefile
  create  my_gem/LICENSE.txt
  create  my_gem/README.md
  create  my_gem/.gitignore
  create  my_gem/hello-world.gemspec
  create  my_gem/lib/my_gem.rb
  create  my_gem/lib/my_gem/version.rb
```

CREATING GEMS

Place to track your
dependencies, edit name,
description, etc...

```
$ bundle gem my_gem
      create  my_gem/Gemfile
      create  my_gem/Rakefile
      create  my_gem/LICENSE.txt
      create  my_gem/README.md
      create  my_gem/.gitignore
      create  my_gem/hello-world.gemspec
      create  my_gem/lib/my_gem.rb
      create  my_gem/lib/my_gem/version.rb
```

CREATING GEMS

```
Gem::Specification.new do |spec|  
  ...  
  spec.add_development_dependency "bundler", "~> 1.3"  
  spec.add_development_dependency "rake"  
end
```



Use `spec.add_dependency` for runtime dependencies

CHALLENGE ACCEPTED



COMMUNICATION

*Wrap your communication
between apps*

FARADAY

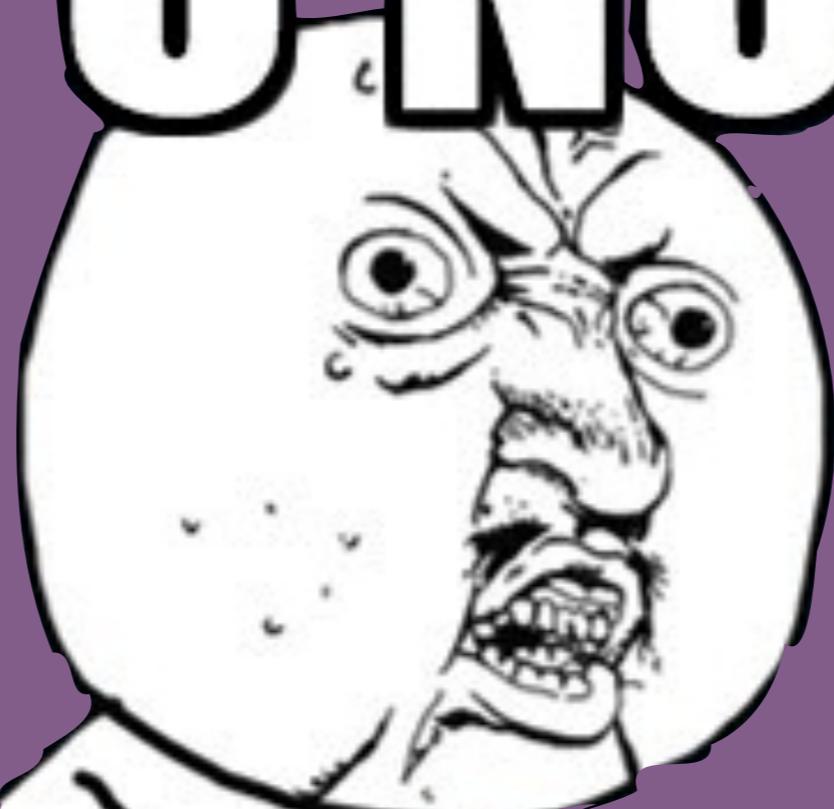
```
conn = Faraday.new(:url => 'http://example.com')
response = conn.get '/'
puts response.body
```

<https://github.com/lostisland/faraday>

CHALLENGE ACCEPTED



YUNO



CONFIG

COMMUNICATION

*Wrap your communication
between apps*

CONNECTION WRAPPER

```
class MyApp::Connection
  def self.billing
    # ... create Faraday connection
  end
end

class MyApp::MyApp < Sinatra::Base
  get '/some_route' do
    Connection.billing.post '/invoice'
  end
end
```

CHALLENGE ACCEPTED



DOTENV

```
# .env.test
HELLO_WORLD_URL=http://example.com
HELLO_WORLD_USER=last
HELLO_WORLD_PASSWORD=unicorn

# in the specs
require 'dotenv'
Dotenv.load!('.env.test')
```

BUNDLER NICENESS

```
# Gemfile
source 'https://rubygems.org'

gem 'my_gem', path: '../my_gem'
```

LOOKING AT THE SPECS

BUILDING A TEST HARNESS

- **Unify test dependency declarations**
- **Inject test connections**
- **Way to create & delete test data**

BUILDING A TEST HARNESS

```
# spec_helper.rb
require 'my_app-test_harness'

MyApp::TestHarness.activate do |harness|
  harness.augment MyApp::Shop::App
  harness.enable :datastore, :billing, :authentication
end

MyApp::TestHarness.data.reset(:invoices)
MyApp::TestHarness.data.generate(:user)
```

DEVELOPMENT SETUP

FOREMAN

- Manage process types
- Easily start all of your app locally
- Makes use of .env files

<https://github.com/ddollar/foreman>

FOREMAN

```
# Procfile
web: bundle exec puma --workers 3 --threads 64:64 --port $PORT
worker: bundle exec sidekiq --concurrency 64
```

```
# Start your app
$ bundle exec foreman start
# Stop your app
Ctrl + c
```

<https://github.com/ddollar/foreman>

CHALLENGE ACCEPTED



VAGRANT

- Not perfect...
- Best way to setup backing services consistently
- It's hard to make a designer learn how to deal with a single Rails application
- It's impossible with 10

VAGRANT BONUS POINTS

- Automate code/git stuff
- Automate migrations
- Make your apps cluster manageable
- Centralize local logs

TRAVIS CI

- **Test your code on each push**
- **See at a glance if a Pull Request is good to merge**
- **Use it for more than just running tests**
- **Free for open-source projects**

```
language: ruby
cache: bundler
rvm:
- 2.0.0
services:
- redis
addons:
  postgresql: 9.3

before_script:
- createuser -U postgres soa --superuser
- createdb -U postgres soa_test --owner=soa
- bundle exec rake db:structure:load
script:
- bundle exec rake

deploy:
- provider: heroku
  strategy: git
  api_key:
    secure: ...
run:
- rake db:migrate
app:
  master: soa-app
```



docs.travis-ci.com/user/build-configuration/

CHALLENGE ACCEPTED



Live Demo

SERVICE AUTHENTICATION

AUTHENTICATION VS. AUTHORIZATION

- **Authentication: Who are you?**
- **Authorization: Are you allowed to do XYZ?**

AUTHENTICATION

- OpenID
- OAuth
- OAuth 2
- Mozilla Persona
- ...

Nobody uses it anymore

Give me a break!

Lets see

Nobody uses it yet

OAUTH 2

- OpenID
- OAuth
- OAuth 2
- Mozilla Persona
- ...

Nobody uses it anymore

Give me a break!

Lets see

Nobody uses it yet

GENERAL IDEA

- Make it easy to use 3rd party services
- No more:
“enter your Twitter username/password”
- Acting on behalf of users

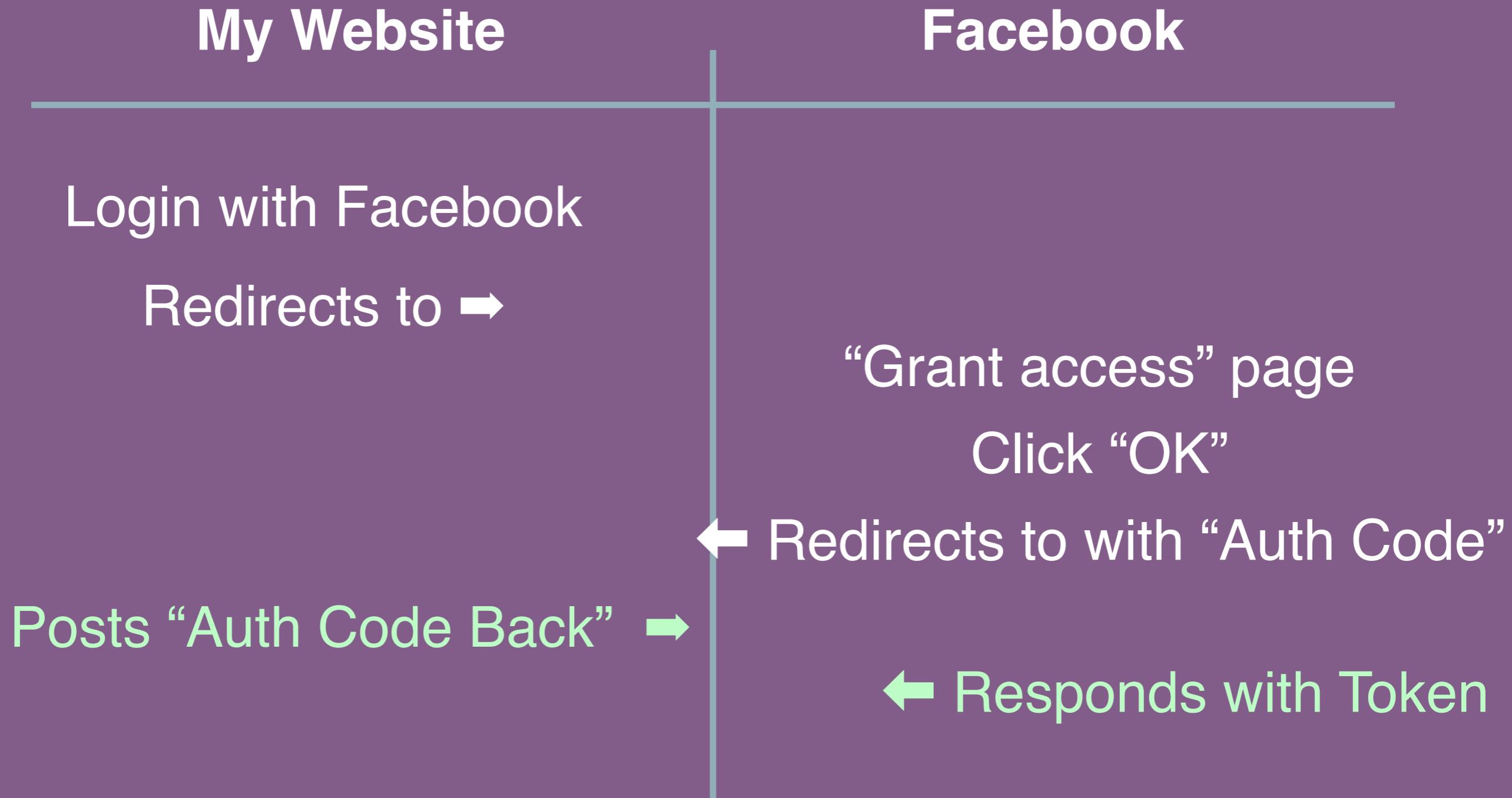
WHAT ABOUT v1?

- No good support for desktop clients
- Complicated cryptography involved
- Mostly long lived authorizations

SO WHAT HAS CHANGED?

- Introducing multiple “flows”
- No more cryptography
- Easier signatures (no more sorting, etc)
- Token can be refreshed

LET'S SEE HOW IT WORKS



WHAT'S THE TOKEN FOR?

WHAT'S THE TOKEN FOR?

```
curl -H "Authorization: Bearer YOURTOKEN" https://  
protected-page.com
```

SO IT'S NOT AN
AUTHENTICATION, RIGHT?

NO, BUT...

“NORMAL” FLOW

1. Get a token
2. Immediately query “/me.json” with the token
3. Get an identity through that endpoint

CHALLENGE ACCEPTED



Time to fool around:

<https://github.com/settings/applications/new>

[http://developer.github.com/v3/oauth_authorizations/
#create-a-new-authorization](http://developer.github.com/v3/oauth_authorizations/#create-a-new-authorization)

SO ARE WE GOOD?

NOT REALLY

hueniverse.com/2012/07/oauth-2-0-and-the-road-to-hell/

- No required token type
- No requirement to implement token expiration
- No strict requirement for registration
- Loose client type definition
- Lack of clear client security properties
- No required grant types
- No guidance on the suitability or applicability of grant types
- No useful support for native applications (but lots of lip service)
- No required client authentication method
- No limits on extensions

BUT IT'S THE BEST WE
HAVE

RTFM

<http://tools.ietf.org/html/draft-ietf-oauth-v2-31>

ROLL YOUR OWN PROVIDER

<https://github.com/applicake/doorkeeper>

<https://github.com/songkick/oauth2-provider>

INTER SERVICE COMMUNICATION

TRANSPORT AGNOSTIC APIs

**YOU TEND TO BUILD AN API AROUND
A CERTAIN MEANS OF
TRANSPORTATION**

**DECUPLE YOUR
BUSINESS LOGIC!**

My Little Application

HTTP

Business Logic

My Little Application

HTTP

SPDY

WebSocket

Business Logic

SUPER SKINNY CONTROLLERS FAT MODELS

[http://weblog.jamisbuck.org/2006/10/18/
skinny-controller-fat-model](http://weblog.jamisbuck.org/2006/10/18/skinny-controller-fat-model)

DEPLOYMENTS

BROWSERS AS CLIENTS

THE PROBLEM

- You have an API endpoint
- You have a HTML5 app that wants to access that endpoint
- You want to use XHR (AJAX) to do so
- You are NOT on the same domain

YOU ARE SCREWED!

BUSTED BY THE “SAME-ORIGIN POLICY”

[https://developer.mozilla.org/en-US/
docs/Web/JavaScript/
Same_origin_policy_for_JavaScript](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Same_origin_policy_for_JavaScript)

**“TWO PAGES HAVE THE SAME ORIGIN IF THE
PROTOCOL, PORT (IF ONE IS SPECIFIED), AND
HOST ARE THE SAME FOR BOTH PAGES. ”**

URL	Outcome	Reason
http://store.company.com/dir2/other.html	Success	
http://store.company.com/dir/inner/another.html	Success	
https://store.company.com/secure.html	Failure	Different protocol
http://store.company.com:81/dir/etc.html	Failure	Different port
http://news.company.com/dir/other.html	Failure	Different host

YOU HAVE 2 OPTIONS NOW

- JSONP
- CORS

JSONP

```
<script>
function myFunc(data) {
  console.log(data);
}
</script>

<script type="application/javascript" src="https://example.com/api/users/1?callback=myFunc"></script>
```

```
// response
myFunc({"your": "data", "has": "arrived"});
```

JSONP

- Works in all browsers!
- A bit tedious (jQuery can help)
- Only GET requests
- Security concerns

CORS