Gradually Modularizing your Monolith with Ruby Packs and Packwerk

Railsconf 2023 - Workshop

Alex Evanczuk and Stephan Hagemann

👋

I am Alex

Product Infrastructure,
Modularity at Gusto

alexevanczuk@gmail.com

👋

I am Stephan

Head of
Product Infrastructure at Gusto

https://ruby.social/@shageman
stephanhagemann.com
gradualmodularization.com

```
cd YOUR_WORKSPACE #ideally, next to your app
git clone --depth=1 https://github.com/gradual-systems/mastodon.git
git clone https://github.com/gradual-systems/railsconf-workshop.git
cd mastodon; ../railsconf-workshop/00.sh
```


```
cd YOUR_WORKSPACE #ideally, next to your app
git clone --depth=1 https://github.com/gradual-systems/mastodon.git
git clone https://github.com/gradual-systems/railsconf-workshop.git
cd mastodon; ../railsconf-workshop/00.sh
```

Gusto Context

Hundreds of engineers
Millions of lines of code
Thousands of commits per month
Packwerk in ~~production~~ dev since 2020
Developing github.com/rubyatscale since 2022

```
cd YOUR_WORKSPACE #ideally, next to your app
git clone --depth=1 https://github.com/gradual-systems/mastodon.git
git clone https://github.com/gradual-systems/railsconf-workshop.git
cd mastodon; ../railsconf-workshop/00.sh
```

Packages / Packs !?1!!

… like gems but with less overhead
… and specifically built for working within large apps
… and with an *extensible ecosystem* to gradually modularize over time

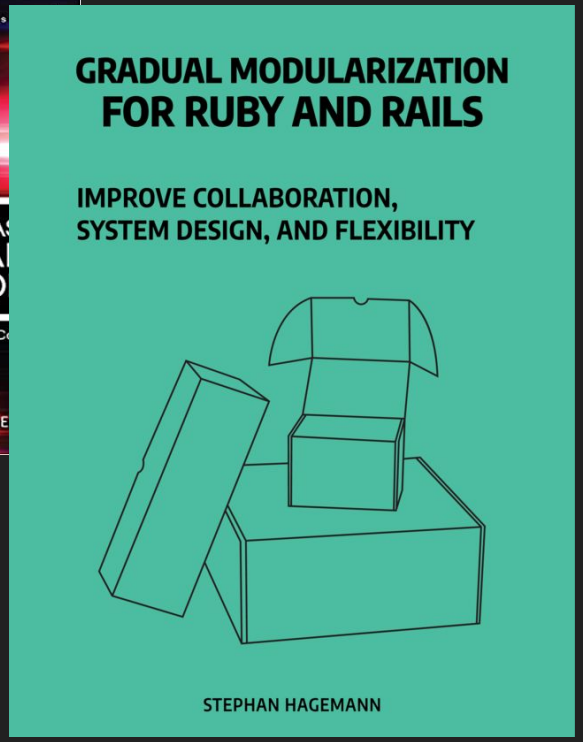
```
cd YOUR_WORKSPACE #ideally, next to your app
git clone --depth=1 https://github.com/gradual-systems/mastodon.git
git clone https://github.com/gradual-systems/railsconf-workshop.git
cd mastodon; ../railsconf-workshop/00.sh
```

Why packages?
Why gradual modularization?

- Create a *high-level structure* within your application
- Understand and manage *app-internal dependencies*
- *Reduce incidental complexity* by removing dependency cycles
- Create better-defined, *domain-specific internal APIs*
- *Reduce CI build time* and cost

- **Untangle the ball-of-mud**

```
cd YOUR_WORKSPACE #ideally, next to your app
git clone --depth=1 https://github.com/gradual-systems/mastodon.git
git clone https://github.com/gradual-systems/railsconf-workshop.git
cd mastodon; ../railsconf-workshop/00.sh
```



leanpub.com/package-based-rails-applications/c/railsconf2023

Ruby/Rails Modularity Slack Server:

tinyurl.com/rubyslack

```
cd YOUR_WORKSPACE #ideally, next to your app
git clone --depth=1 https://github.com/gradual-systems/mastodon.git
git clone https://github.com/gradual-systems/railsconf-workshop.git
cd mastodon; ../railsconf-workshop/00.sh
```

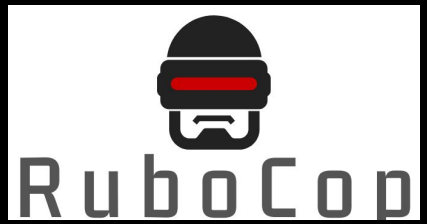
Gradual Modularization:

The Tools

```
cd YOUR_WORKSPACE #ideally, next to your app
git clone --depth=1 https://github.com/gradual-systems/mastodon.git
git clone https://github.com/gradual-systems/railsconf-workshop.git
cd mastodon; ../railsconf-workshop/00.sh
```

Standing on the shoulders of giants!

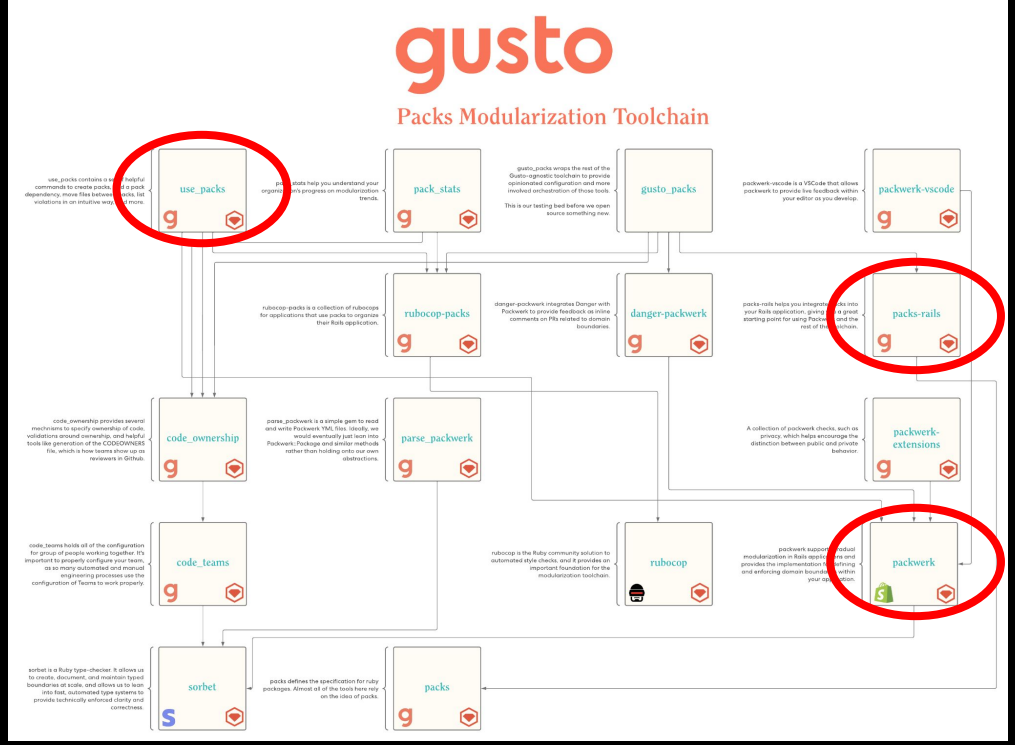
Packwerk



Rubocop

Sorbet

```
cd YOUR_WORKSPACE #ideally, next to your app
git clone --depth=1 https://github.com/gradual-systems/mastodon.git
git clone https://github.com/gradual-systems/railsconf-workshop.git
cd mastodon; ../railsconf-workshop/00.sh
```



https://github.com/rubyatscale

```
cd YOUR_WORKSPACE #ideally, next to your app
git clone --depth=1 https://github.com/gradual-systems/mastodon.git
git clone https://github.com/gradual-systems/railsconf-workshop.git
cd mastodon; ../railsconf-workshop/00.sh
```

Gradual Modularization - Tools

Packs Rails	Package Folders
Packwerk	Dependency checker
Rubocop Packs	Packs/TypedPublicApis cop Packs/DocumentedPublicApis cop Packs/ClassMethodsAsPublicApis cop Packs/RootNamespaceIsPackName cop
Packwerk extensions	Privacy checker Visibility checker Architecture checker
Code Ownership	Package-oriented ownership
Use Packs	CLI tools to manage and move files to packs
Pack Stats	Statistics reporting for packs and packwerk/rubocop usage

```
cd YOUR_WORKSPACE #ideally, next to your app
git clone --depth=1 https://github.com/gradual-systems/mastodon.git
git clone https://github.com/gradual-systems/railsconf-workshop.git
cd mastodon; ../railsconf-workshop/00.sh
```

Where should you start?

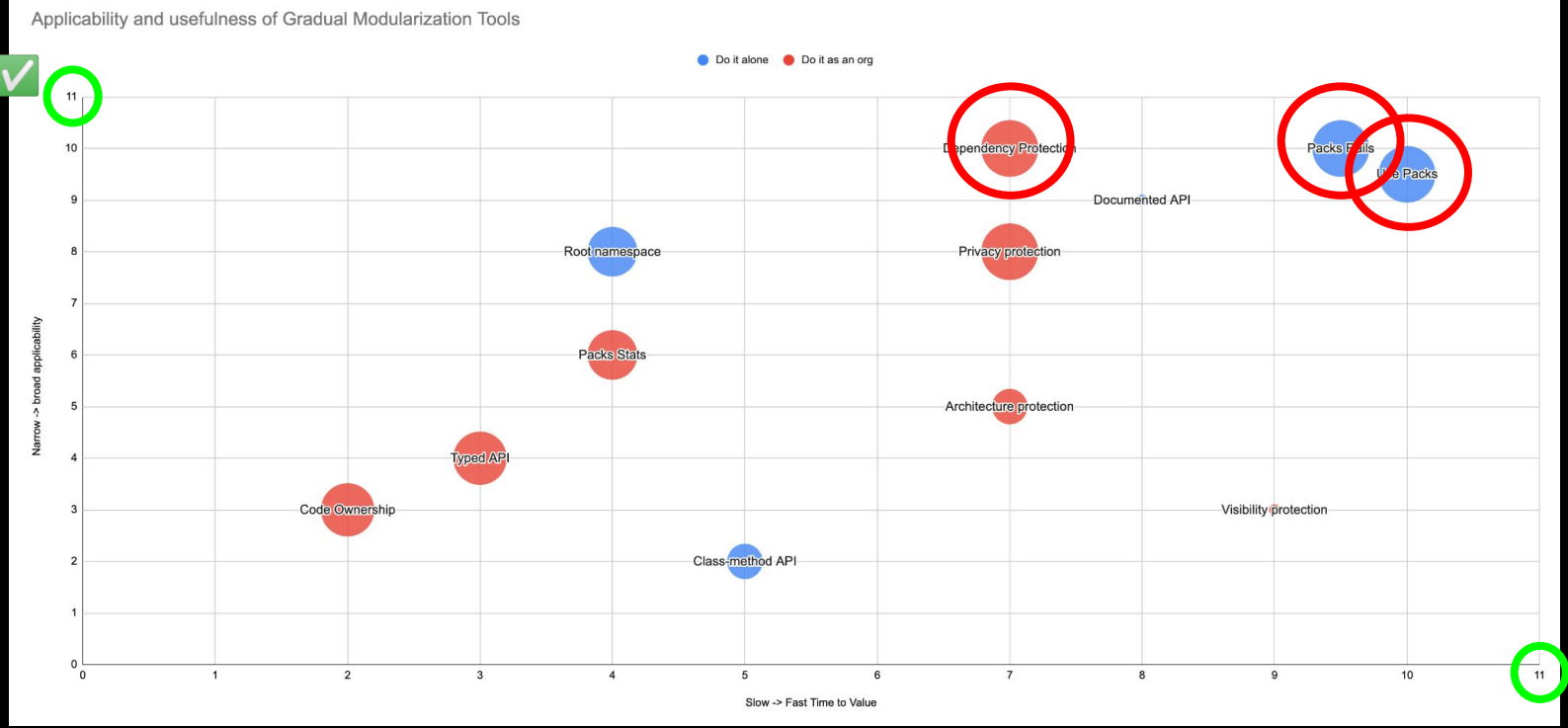
Gradual Modularization is a bit like
choose your own adventure

```
cd YOUR_WORKSPACE #ideally, next to your app
git clone --depth=1 https://github.com/gradual-systems/mastodon.git
git clone https://github.com/gradual-systems/railsconf-workshop.git
cd mastodon; ../railsconf-workshop/00.sh
```

Gradual Modularization - Tools

Packs Rails	Package Folders
Packwerk	Dependency checker
Rubocop Packs	Packs/TypedPublicApis cop Packs/DocumentedPublicApis cop Packs/ClassMethodsAsPublicApis cop Packs/RootNamespaceIsPackName cop
Packwerk extensions	Privacy checker Visibility checker Architecture checker
Code Ownership	Package-oriented ownership
Use Packs	CLI tools to manage and move files to packs
Pack Stats	Statistics reporting for packs and packwerk/rubocop usage

```
cd YOUR_WORKSPACE #ideally, next to your app
git clone --depth=1 https://github.com/gradual-systems/mastodon.git
git clone https://github.com/gradual-systems/railsconf-workshop.git
cd mastodon; ../railsconf-workshop/00.sh
```



https://tinyurl.com/mod-tools-analysis

```
cd YOUR_WORKSPACE #ideally, next to your app
git clone --depth=1 https://github.com/gradual-systems/mastodon.git
git clone https://github.com/gradual-systems/railsconf-workshop.git
cd mastodon; ../railsconf-workshop/00.sh
```

Gradual Modularization - Tools

Packs Rails	Package Folders
Packwerk	Dependency checker
Rubocop Packs	Packs/TypedPublicApis cop Packs/DocumentedPublicApis cop Packs/ClassMethodsAsPublicApis cop Packs/RootNamespaceIsPackName cop
Packwerk extensions	Privacy checker Visibility checker Architecture checker
Code Ownership	Package-oriented ownership
Use Packs	CLI tools to manage and move files to packs
Pack Stats	Statistics reporting for packs and packwerk/rubocop usage

```
cd YOUR_WORKSPACE #ideally, next to your app
git clone --depth=1 https://github.com/gradual-systems/mastodon.git
git clone https://github.com/gradual-systems/railsconf-workshop.git
cd mastodon; ../railsconf-workshop/00.sh
```

Gradual Modularization - Tools

Packs Rails	Package Folders
Packwerk	Dependency checker
Rubocop Packs	Packs/TypedPublicApis cop Packs/DocumentedPublicApis cop Packs/ClassMethodsAsPublicApis cop Packs/RootNamespaceIsPackName cop
Packwerk extensions	Privacy checker Visibility checker Architecture checker
Code Ownership	Package-oriented ownership
Use Packs	CLI tools to manage and move files to packs
Pack Stats	Statistics reporting for packs and packwerk/rubocop usage

```
cd YOUR_WORKSPACE #ideally, next to your app
git clone --depth=1 https://github.com/gradual-systems/mastodon.git
git clone https://github.com/gradual-systems/railsconf-workshop.git
cd mastodon; ../railsconf-workshop/00.sh
```

Let’s get practical!

```
cd YOUR_WORKSPACE #ideally, next to your app
git clone --depth=1 https://github.com/gradual-systems/mastodon.git
git clone https://github.com/gradual-systems/railsconf-workshop.git
cd mastodon; ../railsconf-workshop/00.sh
```

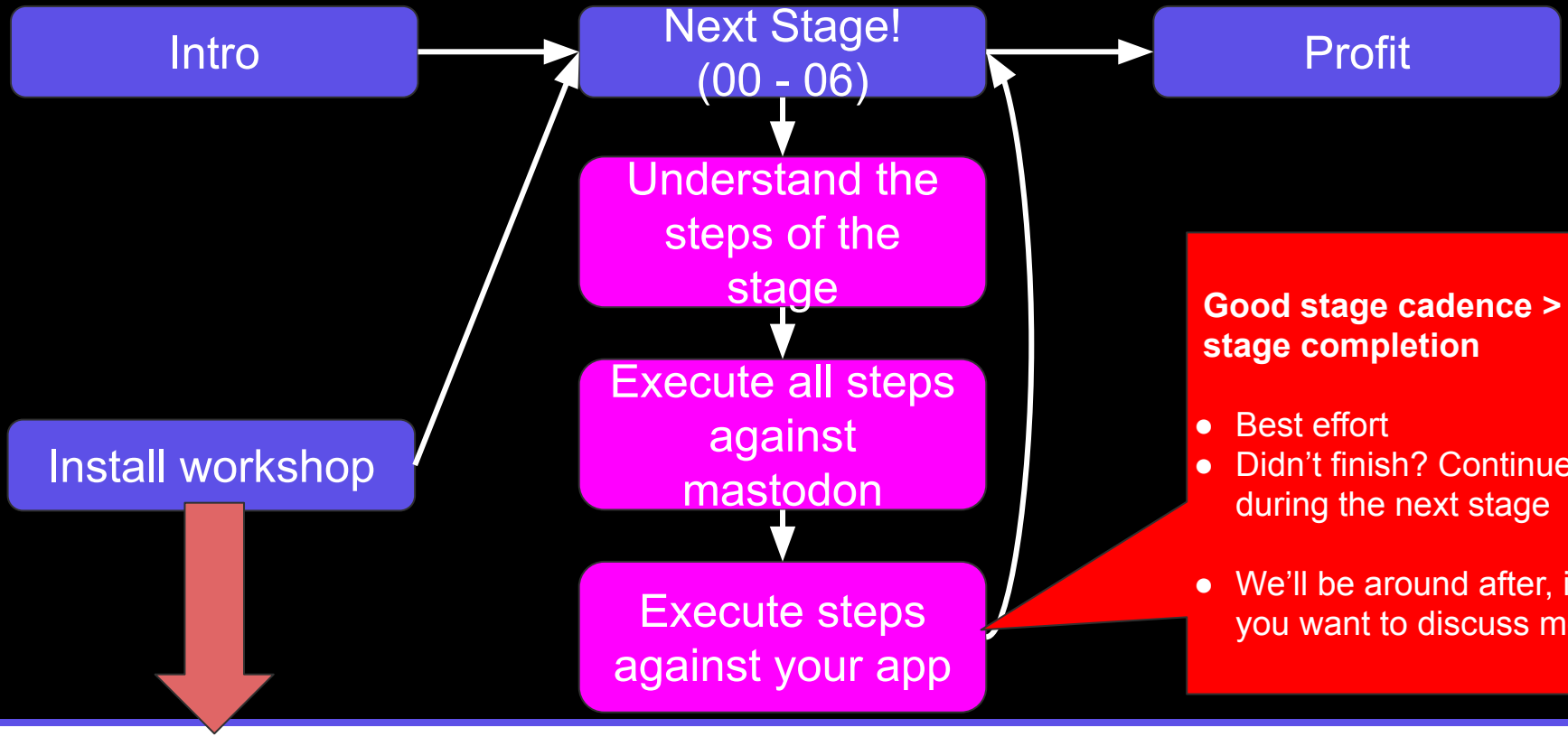
Say hi to our helpers!

```
cd YOUR_WORKSPACE #ideally, next to your app
git clone --depth=1 https://github.com/gradual-systems/mastodon.git
git clone https://github.com/gradual-systems/railsconf-workshop.git
cd mastodon; ../railsconf-workshop/00.sh
```

If you get stuck, send a message to our slack channel:

#ws-gradually-modularizing-your-monolith-with-ruby-packs-and-packwerk

```
cd YOUR_WORKSPACE #ideally, next to your app
git clone --depth=1 https://github.com/gradual-systems/mastodon.git
git clone https://github.com/gradual-systems/railsconf-workshop.git
cd mastodon; ../railsconf-workshop/00.sh
```




```
cd YOUR_WORKSPACE #ideally, next to your app
git clone --depth=1 https://github.com/gradual-systems/mastodon.git
git clone https://github.com/gradual-systems/railsconf-workshop.git
cd mastodon; ../railsconf-workshop/00.sh
```

Options for getting through the workshop

Step through script

```
cd mastodon
code ../railsconf-workshop/XX.sh
# => run individual transforms manually
```

Run & read script

```
cd mastodon
code ../railsconf-workshop/XX.sh
../railsconf-workshop/XX.sh
```

Checkout the code for the stage

```
cd mastodon
git clean -fd && git checkout . && git checkout XX
```

Goal of this workshop:

Let’s save mastodon some money on CI!

```
cd YOUR_WORKSPACE #ideally, next to your app
git clone --depth=1 https://github.com/gradual-systems/mastodon.git
git clone https://github.com/gradual-systems/railsconf-workshop.git
cd mastodon; ../railsconf-workshop/00.sh
```

00

Getting set up

```
cd YOUR_WORKSPACE #ideally, next to your app
git clone --depth=1 https://github.com/gradual-systems/mastodon.git
git clone https://github.com/gradual-systems/railsconf-workshop.git
cd mastodon; ../railsconf-workshop/00.sh
```

Stage 00 - part 1

```
echo "
gem 'packs-rails'
gem 'use_packs', group: %w(development test)
gem 'visualize_packwerk', group: %w(development test)
gem 'packwerk',
  github: 'gradual-systems/packwerk',
  branch: 'main', group: %w(development test)
" >> Gemfile

bundle install
```

Stage 00 - part 2

```
bundle binstubs bundler --force
bundle binstub use_packs packwerk

echo "pack_paths:
 - packs/*
 - .
" > packs.yml

echo "--require packs/rails/rspec" >> .rspec

bin/packwerk init
```

Packwerk might crash on your app

In `packwerk.yml`

```
exclude:
 - {bin,node_modules,script,tmp,vendor}/**/*
 - <relative path to offending file>
```

01

Extracting an admin pack

Admin interfaces… everyone’s got one?

How do you identify your admin files?

#1

Learn your domain, speak to the experts, have a meeting, do some analysis, make some proposals, draft a solution, make sure to get it right, …

#2

```
find . | grep admin
```

Stage 01

```
echo "Create admin pack"

bin/packs create packs/admin
bin/packs move packs/admin app/*/admin
bin/packs move . packs/admin/app/javascript

bin/packwerk update
which dot && bin/packs visualize
```

Why not move JS?

Because there is usually other stuff going on that is hard to see with ruby tooling

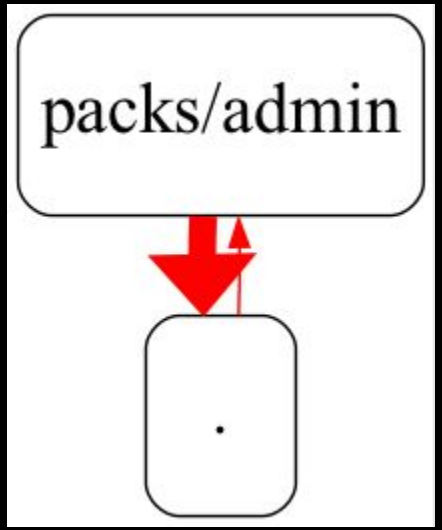
Because JS is often “it’s own application”

Moving files typically just works…

Except for when it doesn’t…

#1 Explicit, manual requires
#2 Autoloading failing to find stuff (metaprogramming)

```
bin/packs visualize #(for step 01)
```



02

Improve admin pack dependencies

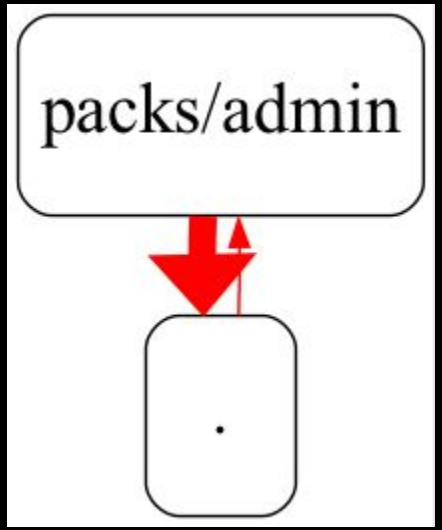
We can learn a lot from packwerk violations…

Do we see if any other code should go in the admin pack?

Stage 02

```
echo "Fix some admin package dependencies"

bin/packs move packs/admin app/controllers/api/*/admin

bin/packwerk update
which dot && bin/packs visualize
```

```
bin/packs visualize #(for step 02)
```



03

Create the “messy middle” pack

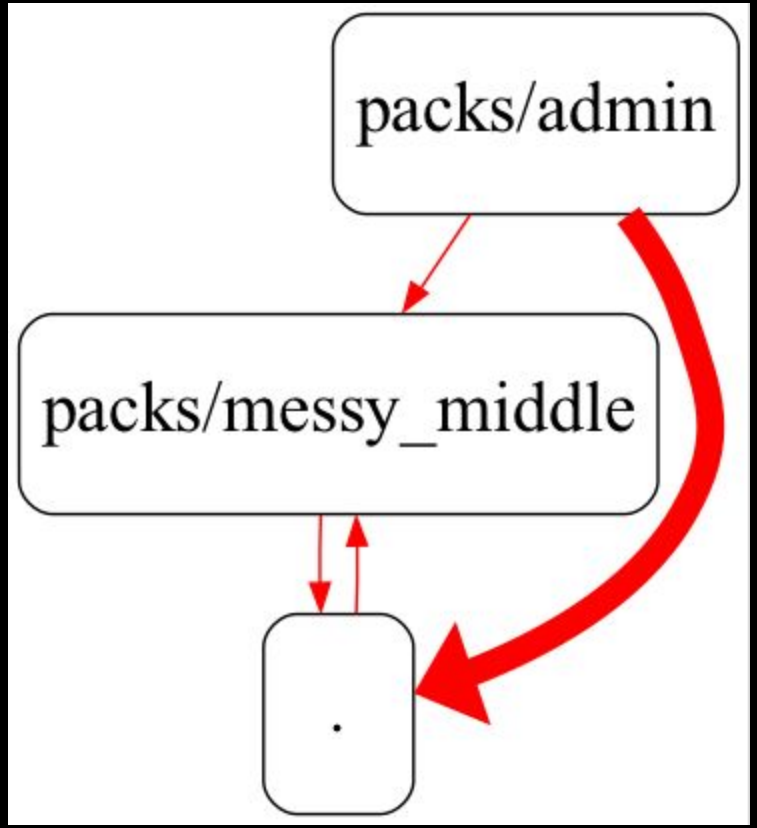
What do we do with the fact that there are SO MANY violations from the admin pack and (relatively) few from the root pack???

Moving the few violating files into their own pack would tell us more about their entanglement

Stage 03

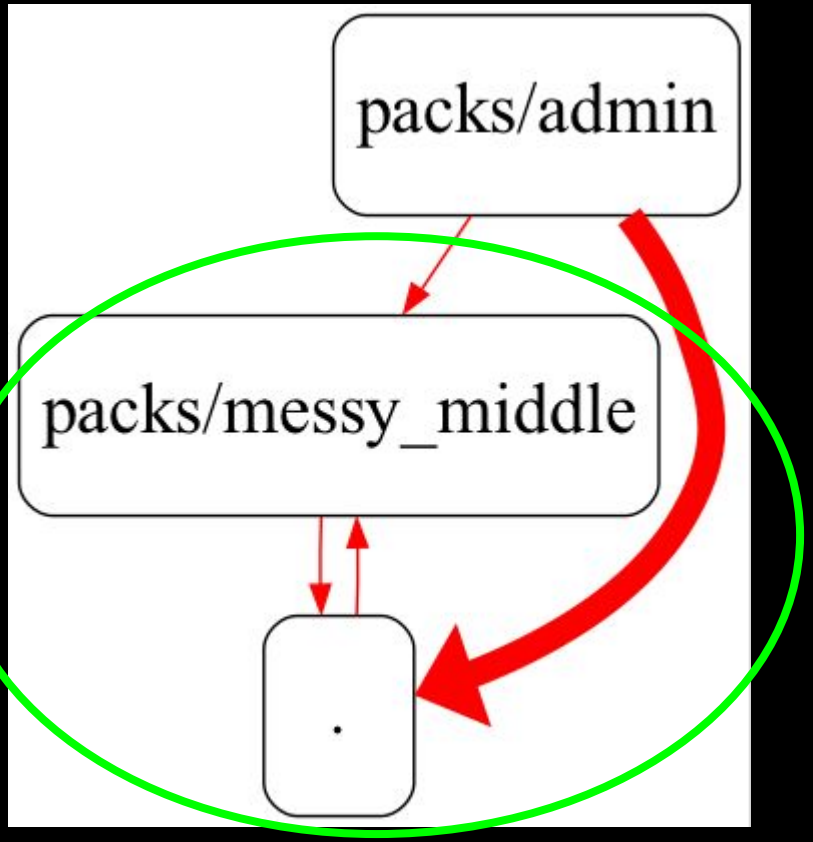
```
echo "Create pack for 'messy middle'"

bin/packs create packs/messy_middle
bin/packs move packs/messy_middle \

`../railsconf-workshop/03_find_files_to_root_violations.rb`

bin/packwerk update
which dot && bin/packs visualize
```

```
bin/packs visualize #(for step 03)
```



04

Analyze and empty out the “messy middle”



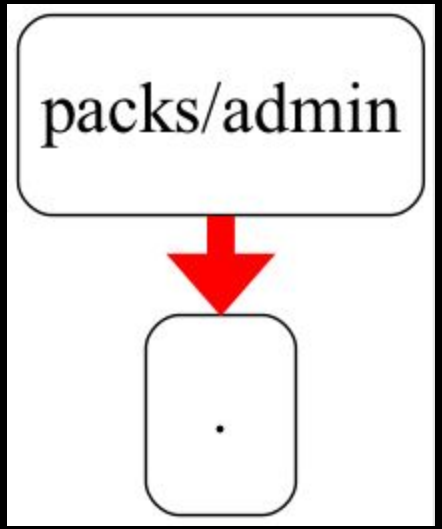
Get yourself a clean dependency graph with this one weird trick…

We can move the messy middle into the root package!

Stage 04

```
echo "Merge 'messy middle' with root pack"

bin/packs move . packs/messy_middle
rm -rf packs/messy_middle

bin/packwerk update
which dot && bin/packs visualize
```

bin/packs visualize #(for step 04)



05

Create user facing app pack

Alex and I have a long standing feud with code in the root pack…

If packages allow us to separate concepts within our domain, any domain code that is in the root package is entangled with the code that *configures the app and makes it runnable…*

We like the root pack to only do that: application configuration and setup

What would we name the pack we extract out of the root pack?

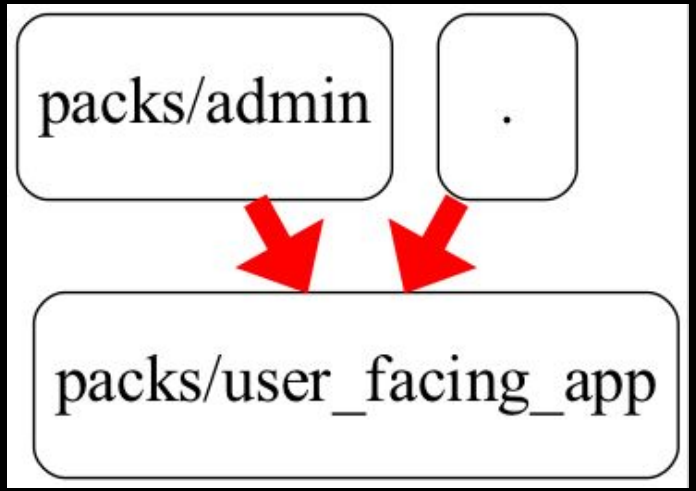
What is currently left there?

The app the user sees! Let’s make that explicit

Stage 05

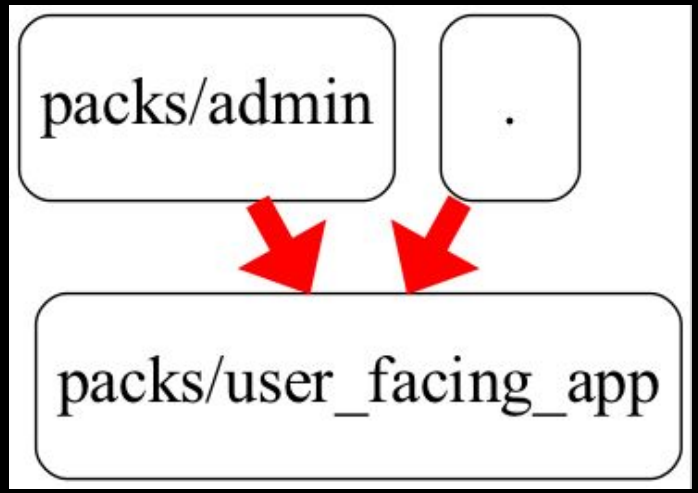
```
echo "Create user facing app pack"

bin/packs create packs/user_facing_app
bin/packs move packs/user_facing_app app
bin/packs move . packs/user_facing_app/app/javascript

bin/packwerk update
which dot && bin/packs visualize
```

```
bin/packs visualize #(for step 05)
```



06

Accept packwerk dependencies



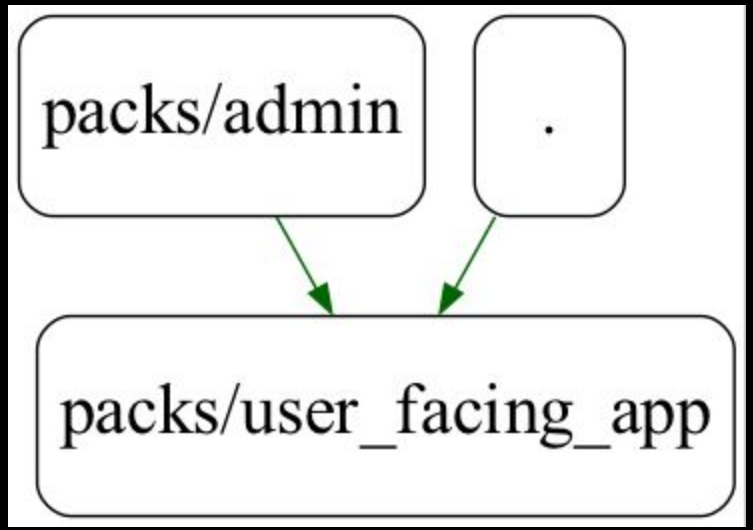
These dependencies look pretty good!

Let’s accept them.

Stage 06

```
echo "Accept packwerk dependencies"

bin/packs add_dependency . packs/user_facing_app
bin/packs add_dependency packs/admin packs/user_facing_app

bin/packwerk update
which dot && bin/packs visualize
```

```
bin/packs visualize #(for step 06)
```

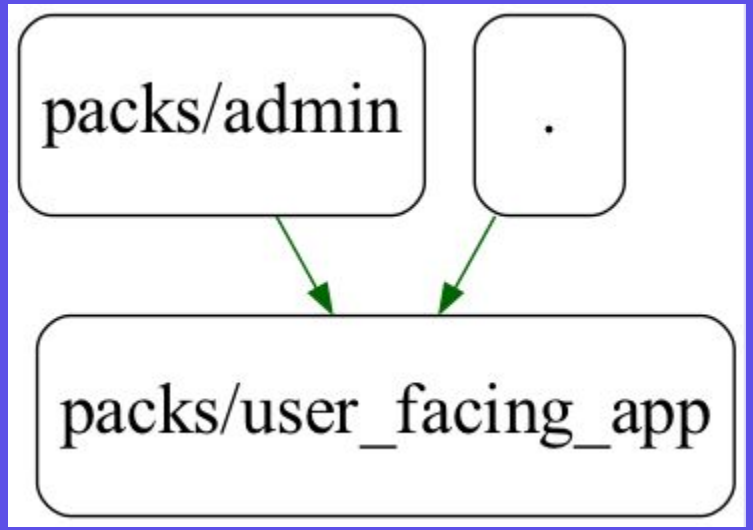


There is a step 07…

Did we save mastodon some money on CI?

Not yet… *but we're close*

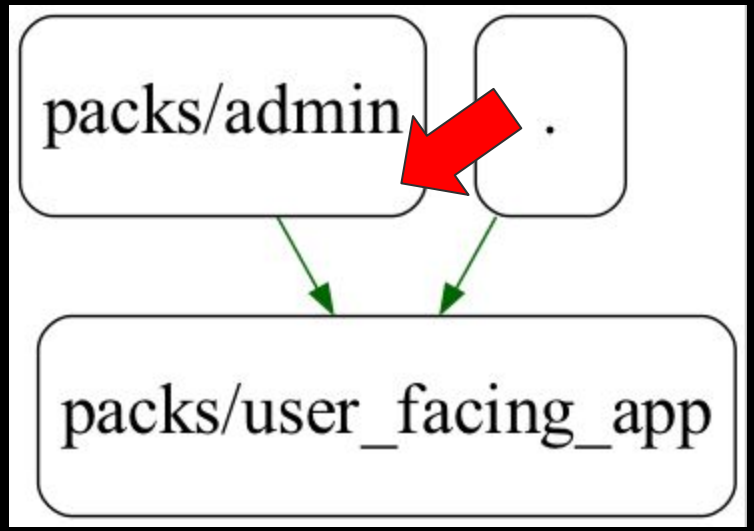
425 tests



158 tests

3559 tests

Does packwerk see all dependencies?



😬

❌ Dynamic method parameters
❌ Metaprogramming
❌ Monkey-patching

What about…

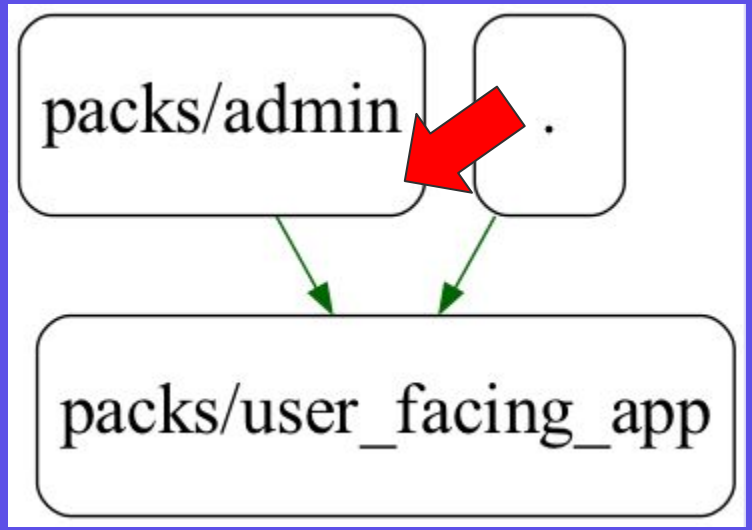
Routes?
Initializers?
Other metaprogramming?

Did we save mastodon some money on CI?

Not yet… *but we're close*

425 tests

425 + 158 = 583 tests to validate changes



158 tests

158 tests to validate package

3559 tests

3559 + 425 + 158 = 4142 tests to validate changes

At Gusto we save about 20% on CI doing this

How do you benefit from this?

😬 There is no open-source library for this *yet*

Care to contribute? → github.com/rubyatscale 😏

For now, this gist will do the basics 😁
https://tinyurl.com/conditionalpackbuilds

Recap and Summary

Moving files around (even loads of them) isn’t scary!

Moving files around leads to interesting insights!

Packages give helpful context to our code!

Packages can save money on CI!

Gradual Modularization - Tools

Packs Rails	Package Folders
Packwerk	Dependency checker
Rubocop Packs	Packs/TypedPublicApis cop Packs/DocumentedPublicApis cop Packs/ClassMethodsAsPublicApis cop Packs/RootNamespaceIsPackName cop
Packwerk extensions	Privacy checker Visibility checker Architecture checker
Code Ownership	Package-oriented ownership
Use Packs	CLI tools to manage and move files to packs
Pack Stats	Statistics reporting for packs and packwerk/rubocop usage

Gradual Modularization - Tools

Packs Rails	Package Folders
Packwerk	Dependency checker
Rubocop Packs	Packs/TypedPublicApis cop Packs/DocumentedPublicApis cop Packs/ClassMethodsAsPublicApis cop Packs/RootNamespaceIsPackName cop
Packwerk extensions	Privacy checker Visibility checker Architecture checker
Code Ownership	Package-oriented ownership
Use Packs	CLI tools to manage and move files to packs
Pack Stats	Statistics reporting for packs and packwerk/rubocop usage

We hope this got you excited to start your modularization journey!

👋

I am Alex

👋

I am Stephan

Thank you!

Wait… we have more time?

Step 07

Your own app!