Joel Biffin
Backend Engineer (Payments & Risk)
Cleo AI

github.com/joelbiffin
linkedin.com/in/joelbiffin

CLEO

LLGems

Re-Using Code the Ruby Way

CLEO

AGENDA

01 How Cleo's Rails app runs in production

02 What some of the trade-offs are in the chosen design

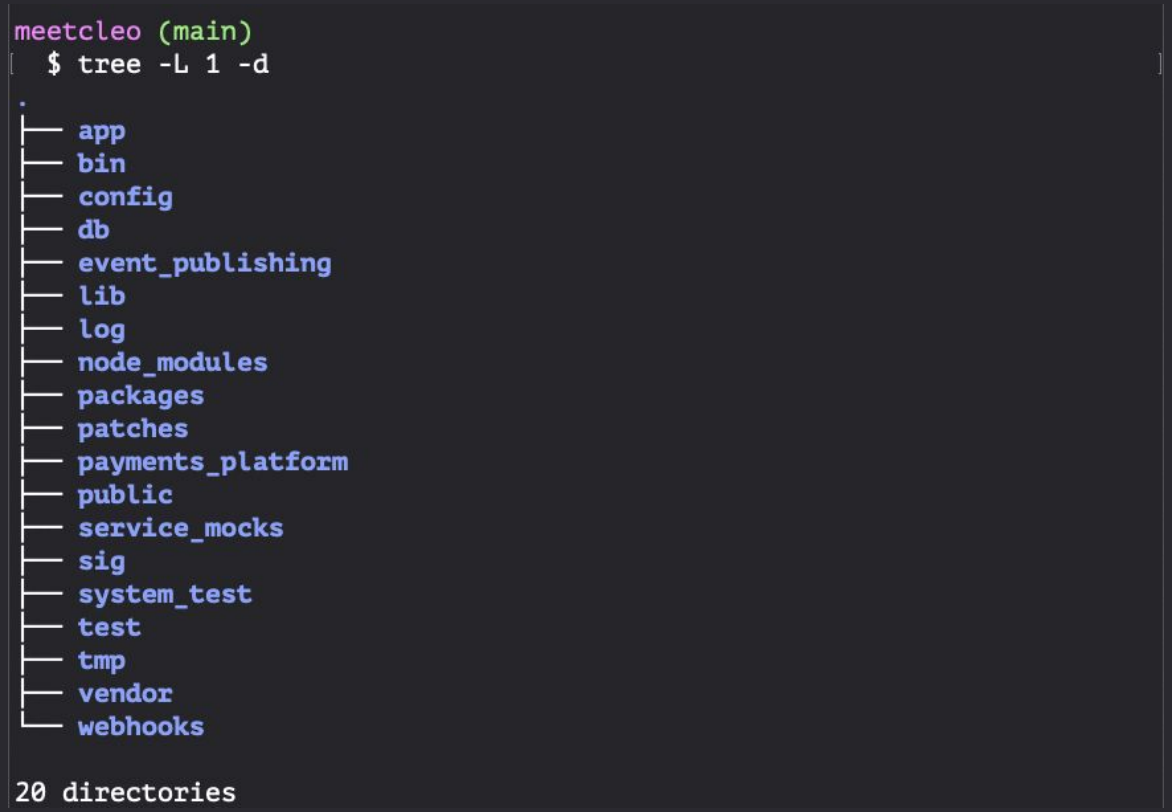
03 How we're trying to overcome one of the trade-offs using LLGems

01

Cleo in Production

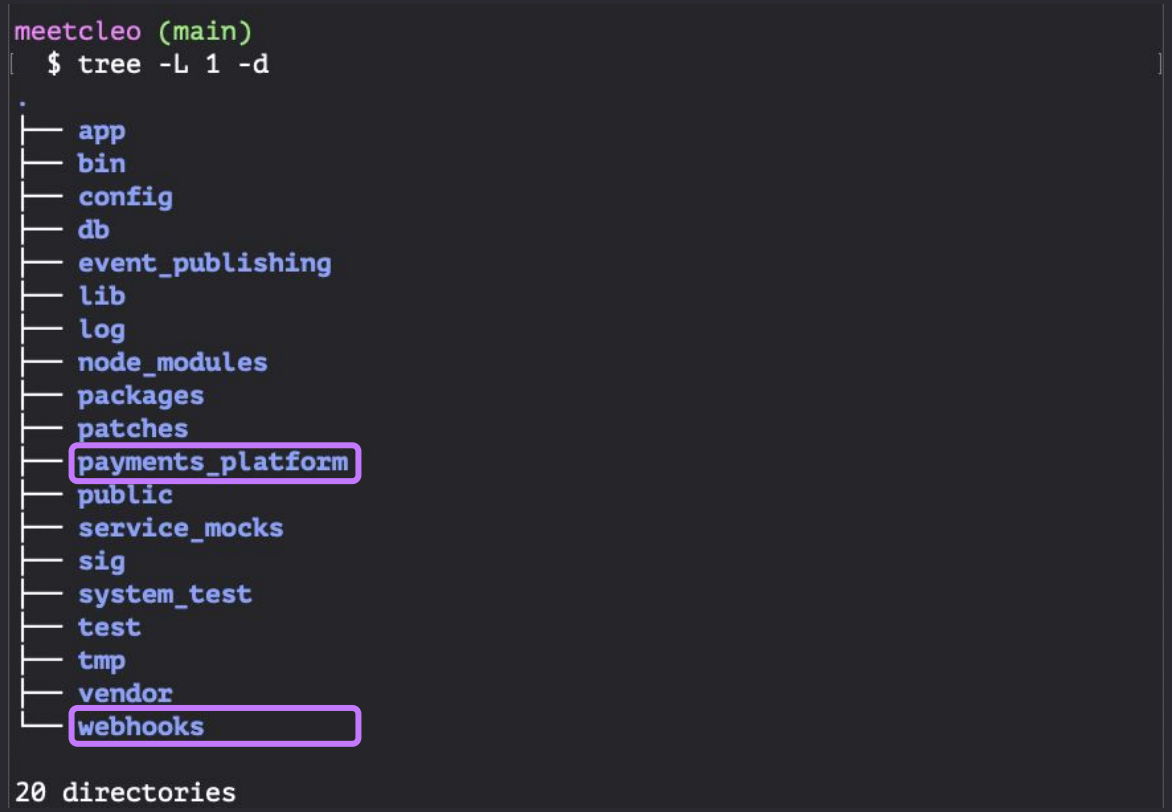
An overview of our Rails monolith

CLEO


meetcleo (main)
$ tree -L 1 -d
.
├── app
├── bin
├── config
├── db
├── event_publishing
├── lib
├── log
├── node_modules
├── packages
├── patches
├── payments_platform
├── public
├── service_mocks
├── sig
├── system_test
├── test
├── tmp
├── vendor
└── webhooks
20 directories

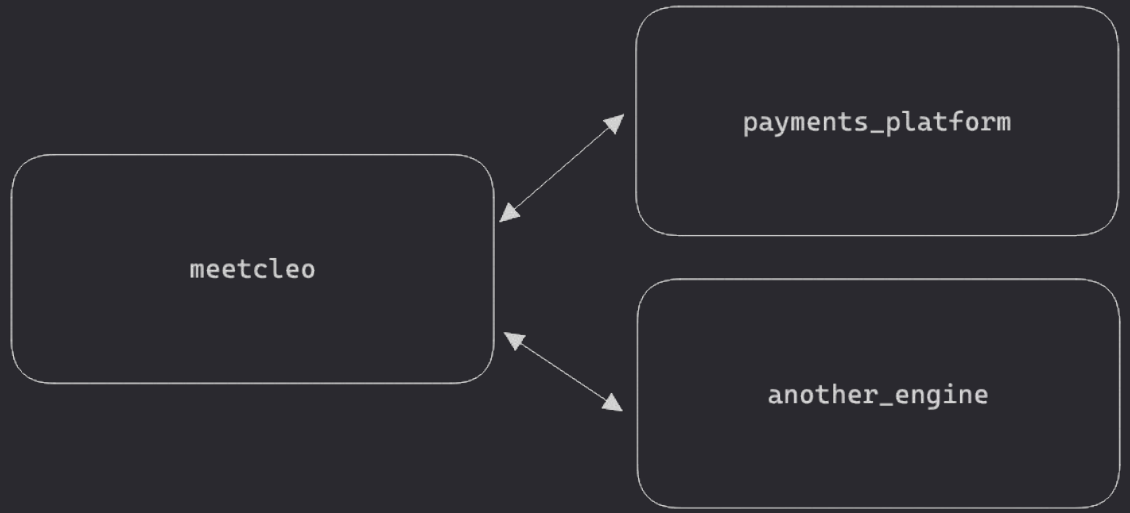
```
meetcleo (main)
  $ tree -L 1 -d app
app
├── assets
├── cleo_bot
├── controllers
├── events
├── helpers
├── javascript
├── lib
├── mailers
├── models
├── presenters
├── queries
├── serializers
├── services
├── validators
├── views
└── workers

17 directories
```

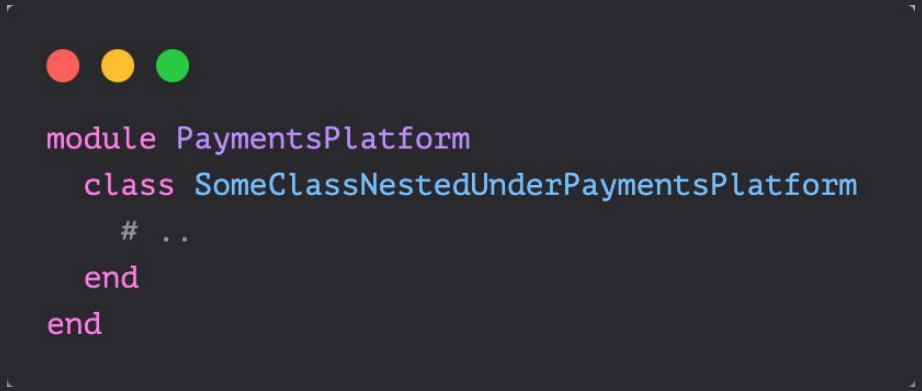
CLEO


meetcleo (main)
$ tree -L 1 -d
.
├── app
├── bin
├── config
├── db
├── event_publishing
├── lib
├── log
├── node_modules
├── packages
├── patches
├── payments_platform
├── public
├── service_mocks
├── sig
├── system_test
├── test
├── tmp
├── vendor
└── webhooks
20 directories

```
meetcleo (main)
  $ tree -L 1 -d payments_platform
payments_platform
├── app
├── bin
├── config
├── db
├── lib
└── test

7 directories

meetcleo (main)
  $ tree -L 1 -d payments_platform/app
payments_platform/app
├── apis
├── controllers
├── decorators
├── events
├── lib
├── models
├── services
├── webhook_subscribers
└── workers

10 directories
```

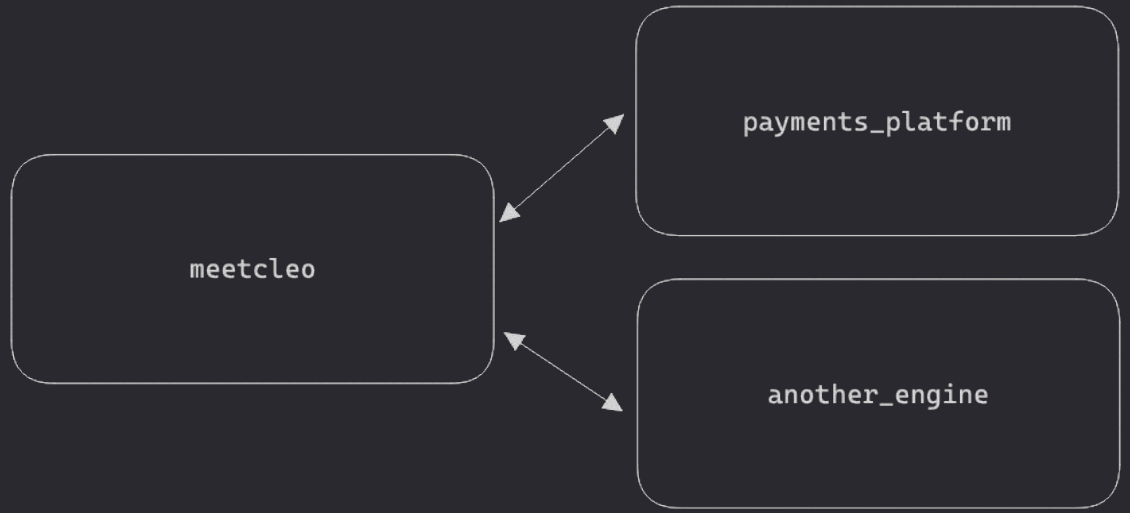
A Rails Engine is just a Rails app


payments_platform
meetcleo
another_engine



```ruby
module PaymentsPlatform
  class SomeClassNestedUnderPaymentsPlatform
    # ..
  end
end
```

CLEO


payments_platform
meetcleo
another_engine

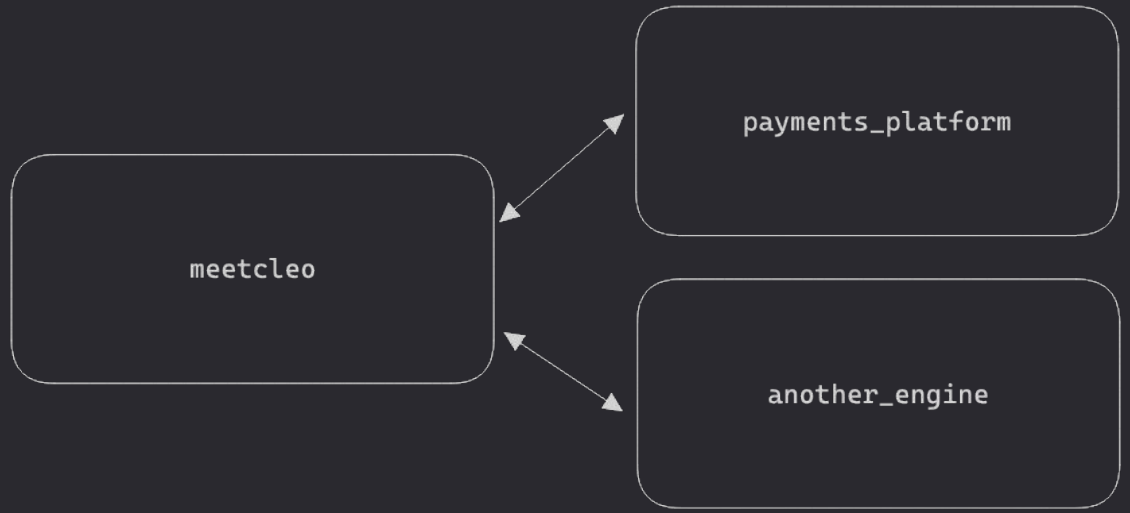
```ruby
class UserController < ApplicationController
  def show
    PaymentsPlatform::SomeClassNestedUnderPaymentsPlatform.find(params[:user_id])
  end
end
```

02

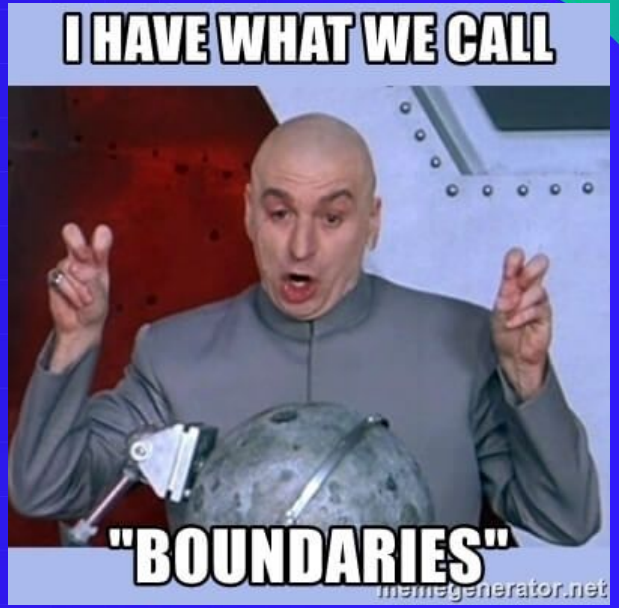
Boundaries are Hard

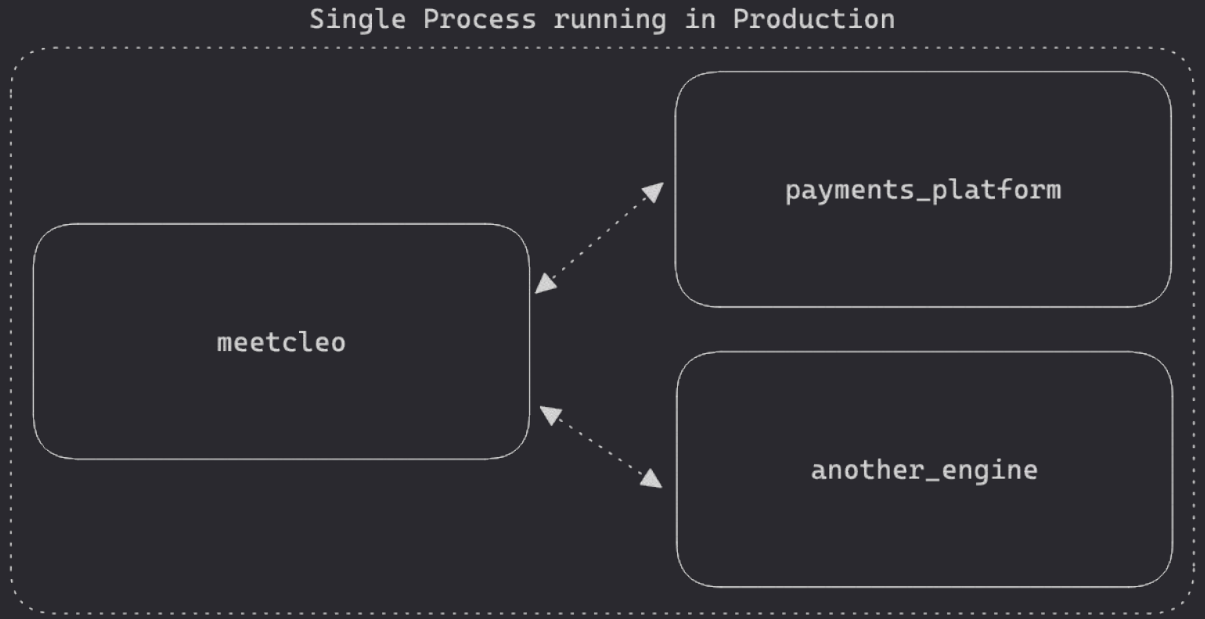
Some design challenges when using Rails engines

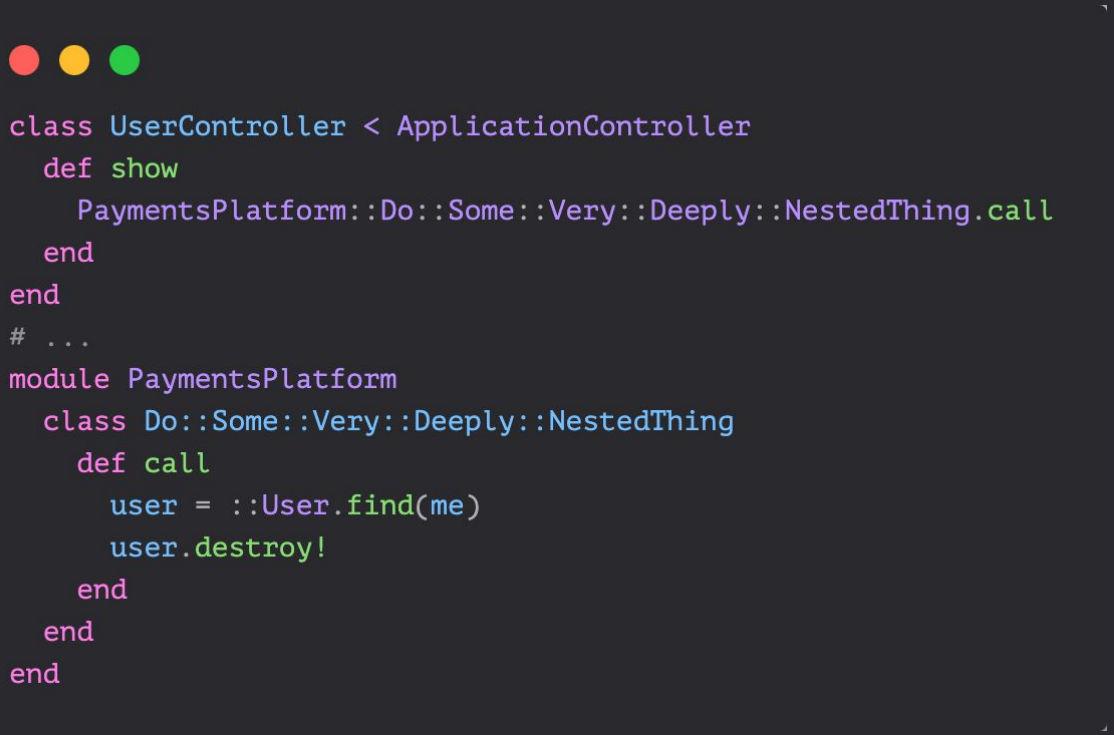
CLEO


payments_platform
meetcleo
another_engine

These boundaries are a lie




Single Process running in Production
payments_platform
meetcleo
another_engine



```ruby
class UserController < ApplicationController
  def show
    PaymentsPlatform::Do::Some::Very::Deeply::NestedThing.call
  end
end
# ...
module PaymentsPlatform
  class Do::Some::Very::Deeply::NestedThing
    def call
      user = ::User.find(me)
      user.destroy!
    end
  end
end
```

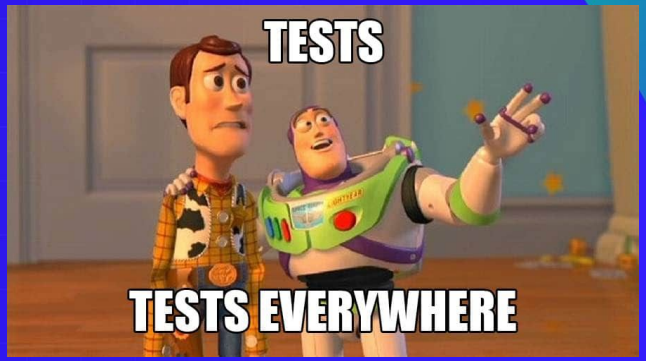
```ruby
class UserController < ApplicationController
  def show
    PaymentsPlatform::Do::Some::Very::Deeply::NestedThing.call
  end
end
# ...
module PaymentsPlatform
  class Do::Some::Very::Deeply::NestedThing
    def call
      user = ::User.find(me)
      user.destroy!
    end
  end
end
```

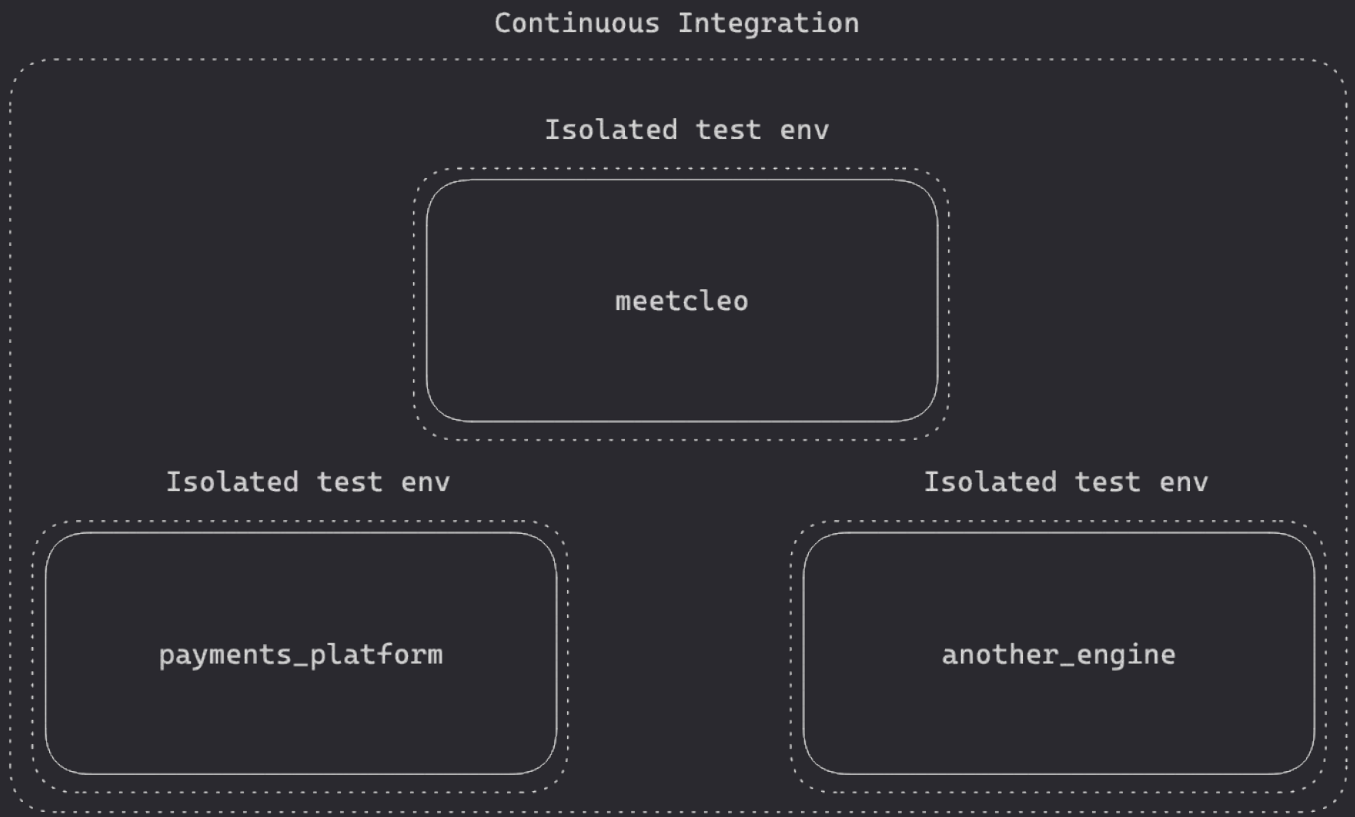
Feature Envy

CLEO

How can we ensure we respect the boundaries?

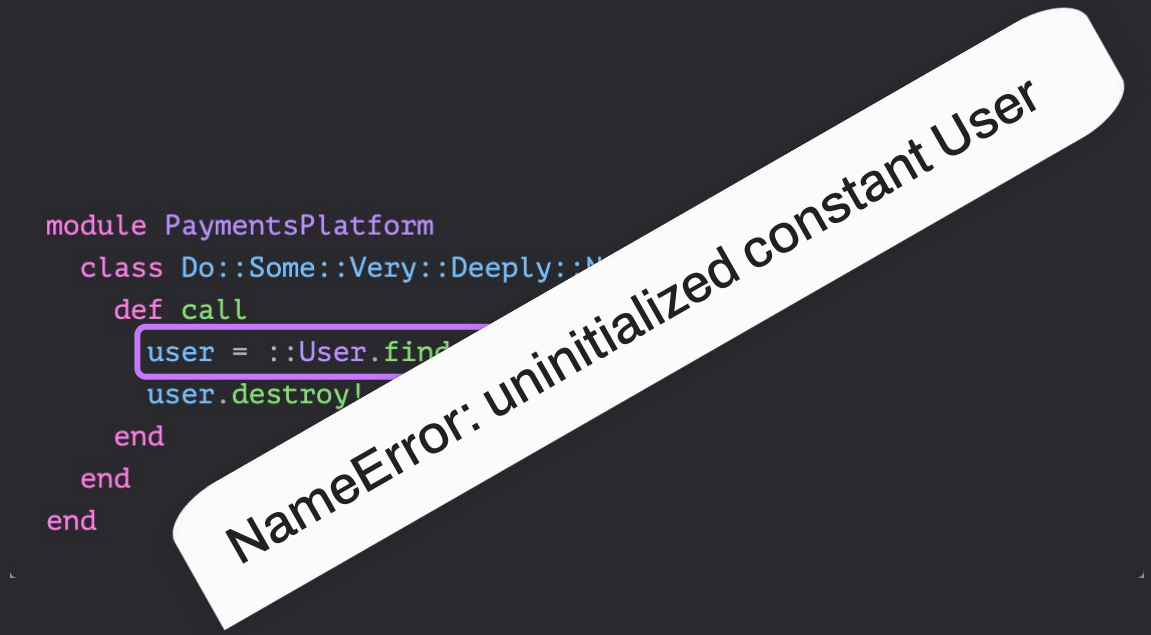
Your tests are your friend




Continuous Integration
Isolated test env
meetcleo
Isolated test env
payments_platform
Isolated test env
another_engine

```ruby
module PaymentsPlatform
  class Do::Some::Very::Deeply::NestedThing
    def call
      user = ::User.find(me)
      user.destroy!
    end
  end
end
```

CLEO


module PaymentsPlatform
class Do::Some::Very::Deeply::
def call
user = ::User.find
user.destroy!
end
end
end
NameError: uninitialized constant User

Dependencies are tough.

Engines impact our ability to share dependencies

1

We cannot co-depend on Constant references

2

We cannot co-depend on Ruby gems

3

We cannot leverage useful test helpers

03

Leveraging Localised Gems

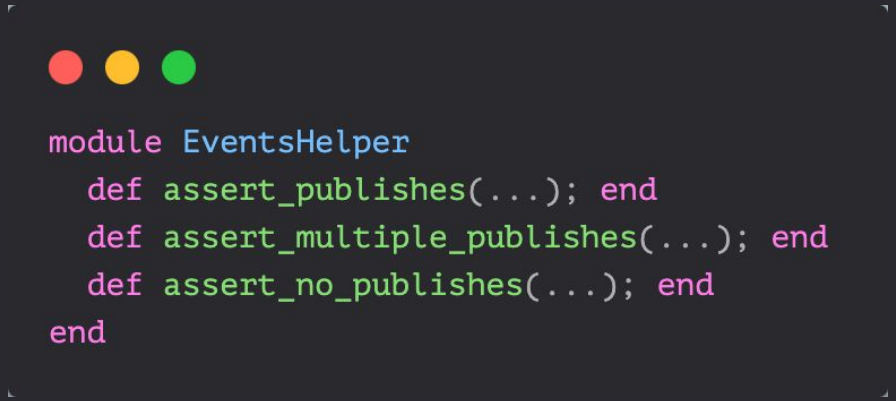
CLEO

Case Study: EventPublishing

Publishing Kafka Events from Rails

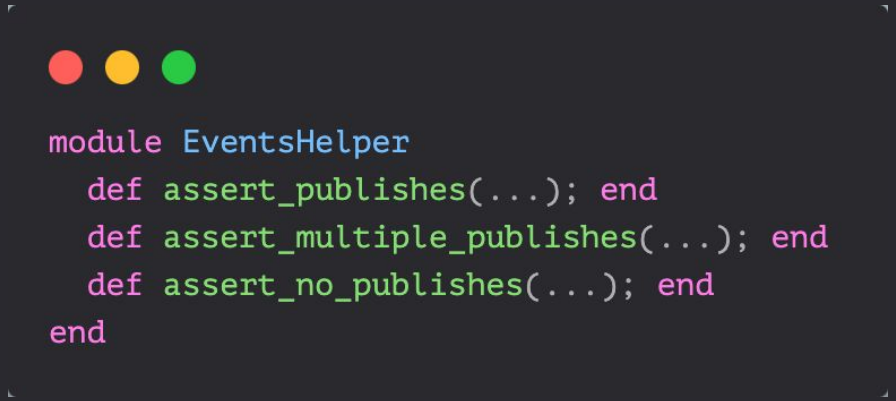
- We **live and breath** data
- We **love publishing events** to better understand the inner-workings of the system
- The Platforms team have created some great infrastructure to make publishing events easy, using **3rd party tools**:
  - Apache Kafka
  - DeliveryBoy
  - AvroTurf


EventPublishing Tooling
Events::Base
Eventable
Test::EventsHelper
SchemaRegistryWrapper
DeliveryBoyWrapper

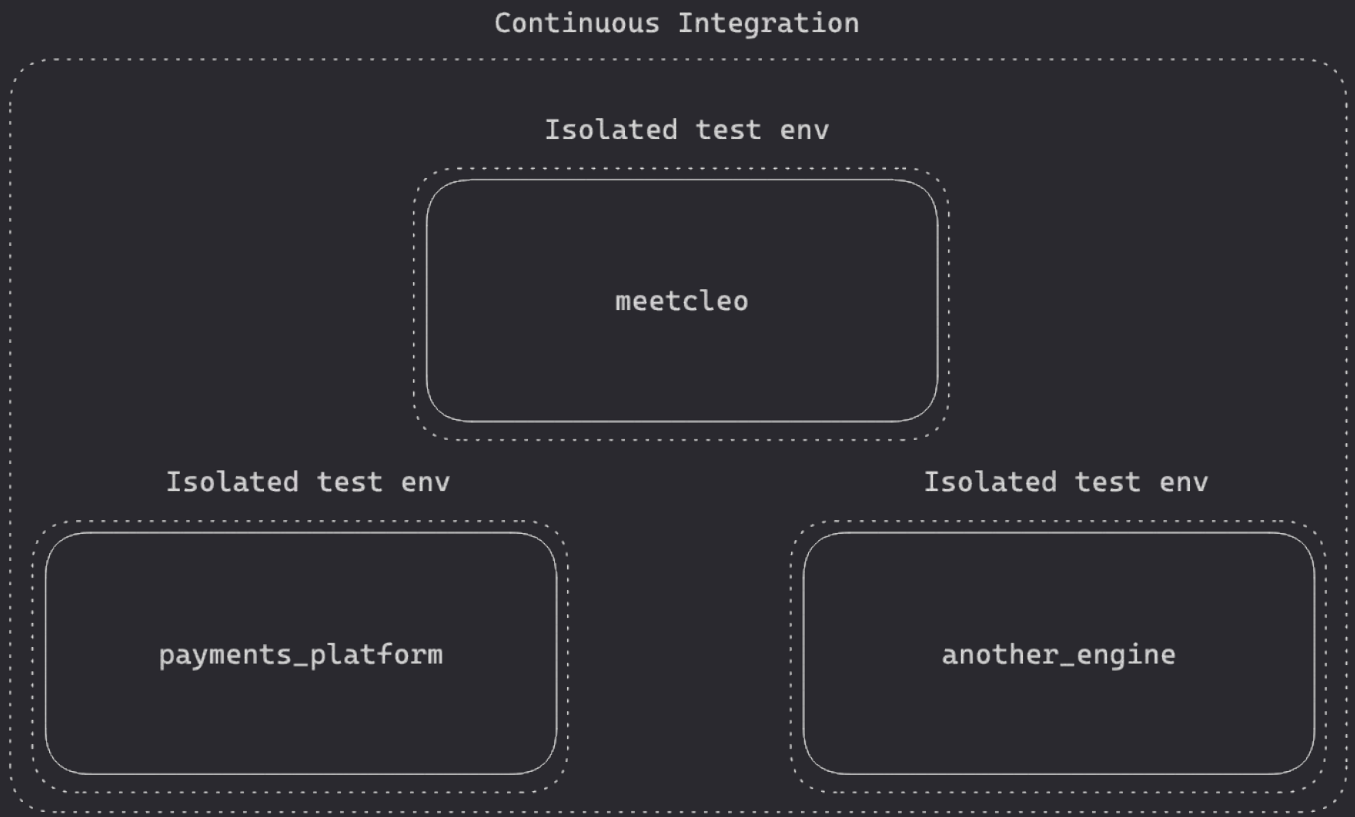


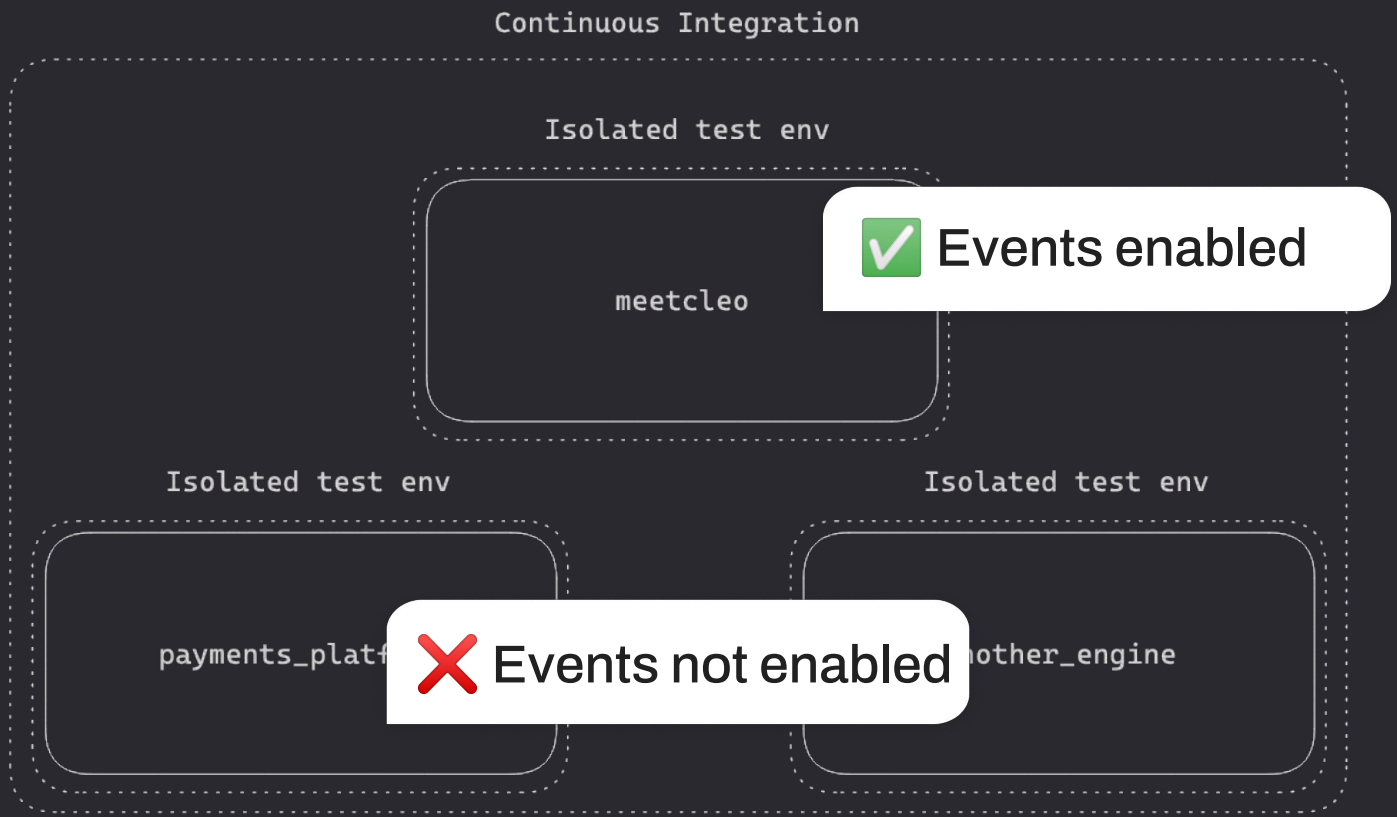
```ruby
module EventsHelper
  def assert_publishes(...); end
  def assert_multiple_publishes(...); end
  def assert_no_publishes(...); end
end
```

```ruby
class ExampleTest < ActiveSupport::TestCase
  test 'the method publishes 1 "foo" event' do
    assert_publishes 'events.foo' do
      Example.new.publish_foo
    end
  end
end
```



```ruby
module EventsHelper
  def assert_publishes(...); end
  def assert_multiple_publishes(...); end
  def assert_no_publishes(...); end
end
```


Continuous Integration
Isolated test env
meetcleo
Isolated test env
payments_platform
Isolated test env
another_engine


Continuous Integration
Isolated test env
meetcleo
✅ Events enabled
Isolated test env
Isolated test env
payments_platf
❌ Events not enabled
other_engine

How can we share the code with our Engines?

Ideation

1

Copy + Paste all of the code.

Because… why the heck not

2

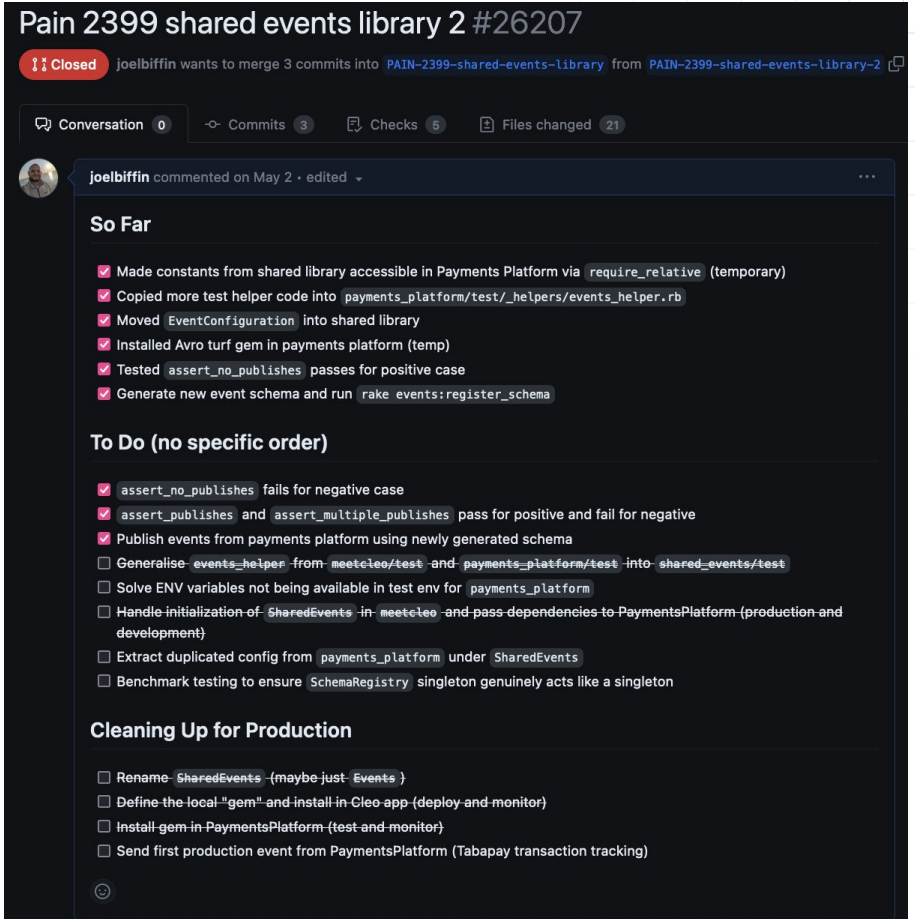
Move the code under some /lib module

3

Make a gem that's inside our repository so we can use a local install

Do the simple thing

Easier said than done



Pain 2399 shared events library 2 #26207

Closed joelbiffin wants to merge 3 commits into PAIN-2399-shared-events-library from PAIN-2399-shared-events-library-2

Conversation 0 Commits 3 Checks 5 Files changed 21

joelbiffin commented on May 2 • edited

So Far

- [x] Made constants from shared library accessible in Payments Platform via `require_relative` (temporary)
- [x] Copied more test helper code into `payments_platform/test/_helpers/events_helper.rb`
- [x] Moved `EventConfiguration` into shared library
- [x] Installed Avro turf gem in payments platform (temp)
- [x] Tested `assert_no_publishes` passes for positive case
- [x] Generate new event schema and run `rake events:register_schema`

To Do (no specific order)

- [x] `assert_no_publishes` fails for negative case
- [x] `assert_publishes` and `assert_multiple_publishes` pass for positive and fail for negative
- [x] Publish events from payments platform using newly generated schema
- [ ] ~~Generalise `events_helper` from `meetcleo/test` and `payments_platform/test` into `shared_events/test`~~
- [ ] Solve ENV variables not being available in test env for `payments_platform`
- [ ] ~~Handle initialization of `SharedEvents` in `meetcleo` and pass dependencies to PaymentsPlatform (production and development)~~
- [ ] Extract duplicated config from `payments_platform` under `SharedEvents`
- [ ] Benchmark testing to ensure `SchemaRegistry` singleton genuinely acts like a singleton

Cleaning Up for Production

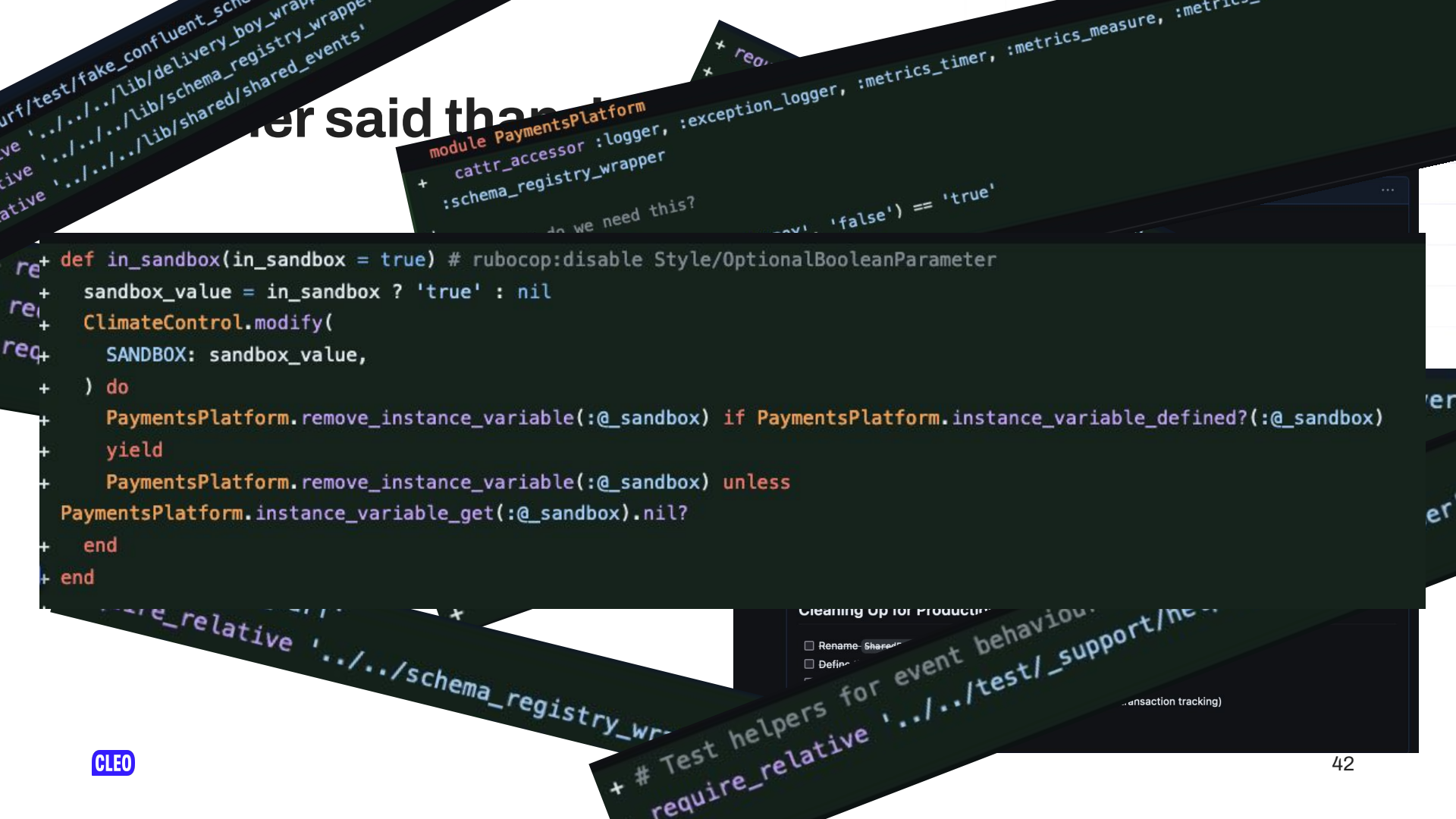
- [ ] ~~Rename `SharedEvents` (maybe just `Events`)~~
- [ ] ~~Define the local "gem" and install in Cleo app (deploy and monitor)~~
- [ ] ~~Install gem in PaymentsPlatform (test and monitor)~~
- [ ] Send first production event from PaymentsPlatform (Tabapay transaction tracking)

Easier said than don


module PaymentsPlatform
cattr_accessor :logger, :exception_logger, :metrics_timer, :metrics_measure,
:schema_registry_wrapper
TODO: do we need this?
def self.sandbox?
@_sandbox ||= ENV.fetch('SANDBOX', 'false') == 'true'
end
end
require_relative 'shared_events/even
require_relative 'shared_events/even
require_relative 'shared_events/even
require 'avro_turf/test/fake_confluent_schema_registry_server
require_relative '../../../lib/delivery_boy_wrapper'
require 'avro_turf'
require_relative '../../schema_registry_wr
Test helpers for event behaviour
require_relative '../../test/_support/helpers/events_helper
Cleaning Up for Productio
Tested
Generate ne
Handle initialization of SharedEvents in meetcleo and pass dependencies to Paym
development)
Extract duplicated config from payments_platform under Share
Benchmark testing to ensure SchemaRegistry singlet
Rename Share
Defin
ansaction tracking)
(temporary)

CLEO

...er said tha...


module PaymentsPlatform
cattr_accessor :logger, :exception_logger, :metrics_timer, :metrics_measure,
:schema_registry_wrapper
do we need this?
'false') == 'true'
def in_sandbox(in_sandbox = true) # rubocop:disable Style/OptionalBooleanParameter
sandbox_value = in_sandbox ? 'true' : nil
ClimateControl.modify(
SANDBOX: sandbox_value,
) do
PaymentsPlatform.remove_instance_variable(:@_sandbox) if PaymentsPlatform.instance_variable_defined?(:@_sandbox)
yield
PaymentsPlatform.remove_instance_variable(:@_sandbox) unless
PaymentsPlatform.instance_variable_get(:@_sandbox).nil?
end
end
require_relative '../../schema_registry_wr
Rename
Defin
ansaction tracking)
Test helpers for event behaviou
require_relative '../../test/_support/he

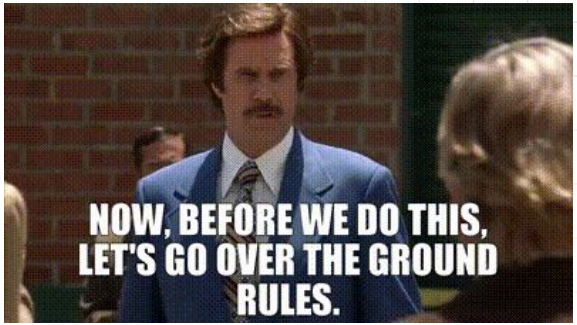
03 (part 2)

Gems, Gems, Gems 💎

2nd time lucky

CLEO

The ground rules



1

No huge checklists on PRs lying about the progress I was making

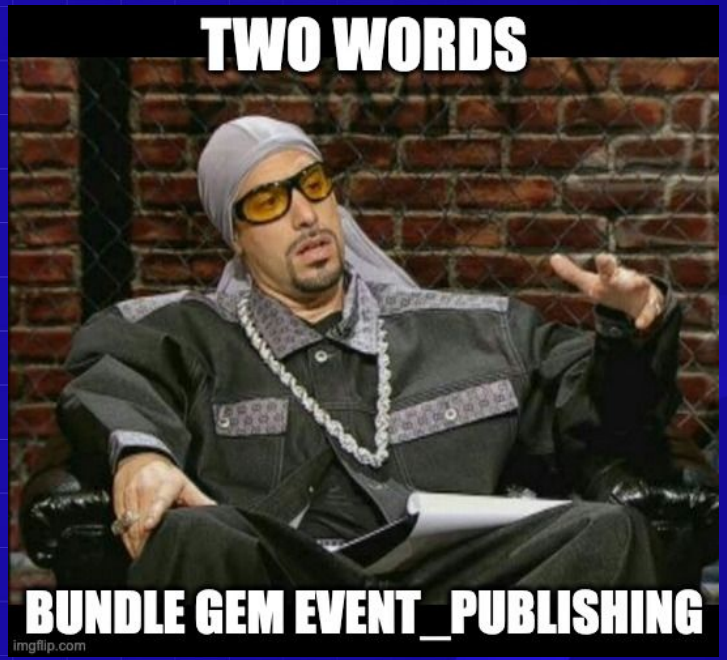
2

Incrementally deployable code to avoid the big-bang disaster

3

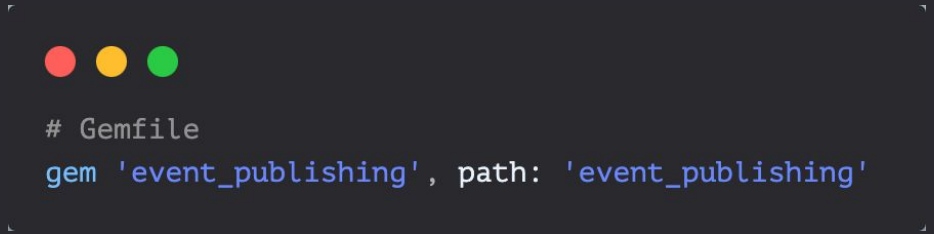
Refrain from solving problem using copious require_relative statements

Step 1: Gem init



```
meetcleo (main)
  $ bundle gem event_publishing
Creating gem 'event_publishing'...
```

```
meetcleo (main)
  $ tree -L 1 event_publishing
event_publishing
├── Gemfile
├── Gemfile.lock
├── LICENSE.txt
├── README.md
├── Rakefile
├── bin
├── event_publishing.gemspec
├── lib
└── test

4 directories, 6 files
```



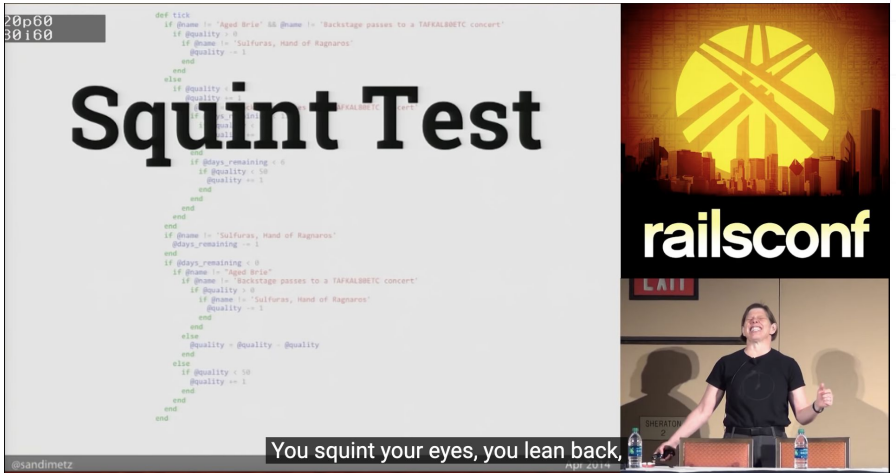
```
# Gemfile
gem 'event_publishing', path: 'event_publishing'
```

Step 2: Moving code into the gem


EventPublishing Tooling
Events::Base
Eventable
Test::EventsHelper
SchemaRegistryWrapper
DeliveryBoyWrapper

The Squint Test

As described by Sandi Metz in *All The Little Things*



```ruby
module Events
  class Base
    include ActiveModel::Validations

    NAMESPACES = %w[cleo events].freeze
    NAMESPACE = NAMESPACES.join('.')
    PLACEHOLDER = '?'.freeze
    SANDBOX = 'sandbox.'.freeze
    KEY_ID_FIELD_NAMES = %i[subject_id object_id complement_id].freeze

    validates :object, :verb, :version, :id, :triggered_at, presence: true
    validates_each :object, :subject, :complement do |record, attr, value|
      next if value.nil?

      if value.is_a?(Eventable)
        record.errors.add(attr, 'must have an `eventable_type_name`') unles
      else
        record.errors.add(attr, 'must be an instance of Eventable')
      end
    end
    validate :does_not_use_reserved_fields
    validate :key_contains_an_id

    def initialize(object:, verb:, subject:, version: 'latest', prepositior
      @object = object
      @verb = verb
      @subject = subject
      @version = version
      @preposition = preposition
      @complement = complement
```

Events::Base

```ruby
module Eventable
  extend ActiveSupport::Concern

  included do
    class_attribute :eventable_type_name
    class_attribute :complement_state_column_name, default: :state

    def eventable_id
      self[:id]
    end

    def complement_state
      self[self.class.complement_state_column_name]
    end
  end

  class_methods do
    def eventable_source_name
      table_name&.sub('.', '__')
    end
  end
end
```

Eventable

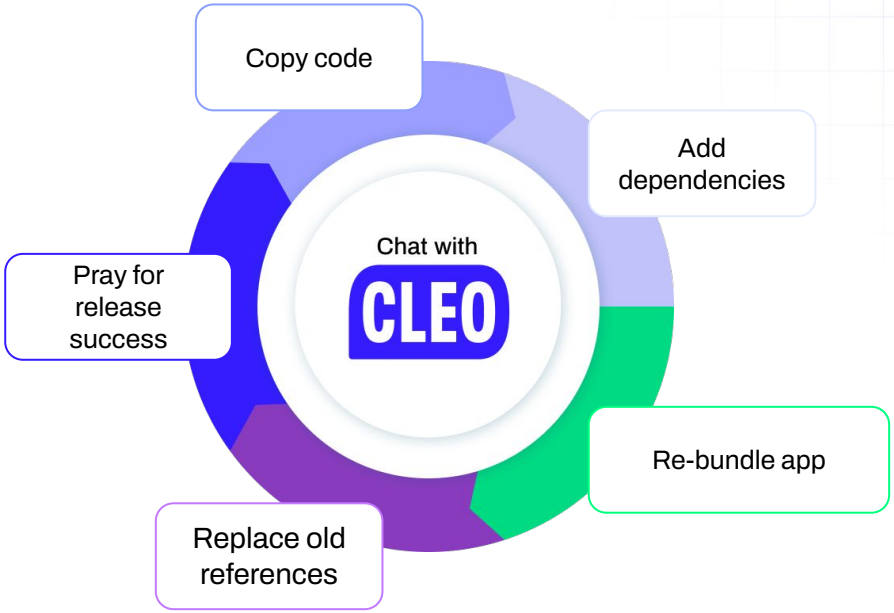
```ruby
require 'avro_turf/test/fake_confluent_schema_registry_server'

module EventsHelper
  class SchemaDefinitionNotFound < StandardError
    def self.message(event_type)
      "You're expecting to publish an event of type `#{raw_event_type_name(
    end

    def self.raw_event_type_name(event_type)
      event_type.delete_prefix("#{Events::Base.namespace}.")
    end
  end

  def setup_event_publishing
    ClimateControl.modify(EVENT_PUBLISHING_ENABLED: 'true', RAISE_ROLLBAR_W
      in_sandbox do
        stub_request(:any, /schema-registry/).to_rack(FakeConfluentSchemaRe
        register_schemas

        yield
      end
    end
  end

  def assert_publishes(event_type, expected_fields: {}, expected_key_fields
    setup_event_publishing do
      DeliveryBoyWrapper.expects(:deliver_async).with do |encoded_data, arg
        assert_equal event_type, args[:topic]

        data = SchemaRegistryWrapper.instance.decode(encoded_data).with_inc
```

EventsHelper

CLEO

Repeatable extraction process

CARRP



Eventable ✅

CLEO

```ruby
module Events
  class Base
    include ActiveModel::Validations

    NAMESPACES = %w[cleo events].freeze
    NAMESPACE = NAMESPACES.join('.')
    PLACEHOLDER = '?'.freeze
    SANDBOX = 'sandbox.'.freeze
    KEY_ID_FIELD_NAMES = %i[subject_id object_id complement_id].freeze

    validates :object, :verb, :version, :id, :triggered_at, presence: true
    validates_each :object, :subject, :complement do |record, attr, value|
      next if value.nil?

      if value.is_a?(Eventable)
        record.errors.add(attr, 'must have an `eventable_type_name`') unles
      else
        record.errors.add(attr, 'must be an instance of Eventable')
      end
    end
    validate :does_not_use_reserved_fields
    validate :key_contains_an_id

    def initialize(object:, verb:, subject:, version: 'latest', prepositior
      @object = object
      @verb = verb
      @subject = subject
      @version = version
      @preposition = preposition
      @complement = complement
```

Events::Base

```ruby
require 'avro_turf/test/fake_confluent_schema_registry_server'

module EventsHelper
  class SchemaDefinitionNotFound < StandardError
    def self.message(event_type)
      "You're expecting to publish an event of type `#{raw_event_type_name(
    end

    def self.raw_event_type_name(event_type)
      event_type.delete_prefix("#{Events::Base.namespace}.")
    end
  end

  def setup_event_publishing
    ClimateControl.modify(EVENT_PUBLISHING_ENABLED: 'true', RAISE_ROLLBAR_W
      in_sandbox do
        stub_request(:any, /schema-registry/).to_rack(FakeConfluentSchemaRe
        register_schemas

        yield
      end
    end
  end

  def assert_publishes(event_type, expected_fields: {}, expected_key_fields
    setup_event_publishing do
      DeliveryBoyWrapper.expects(:deliver_async).with do |encoded_data, arg
        assert_equal event_type, args[:topic]

        data = SchemaRegistryWrapper.instance.decode(encoded_data).with_inc
```

EventsHelper

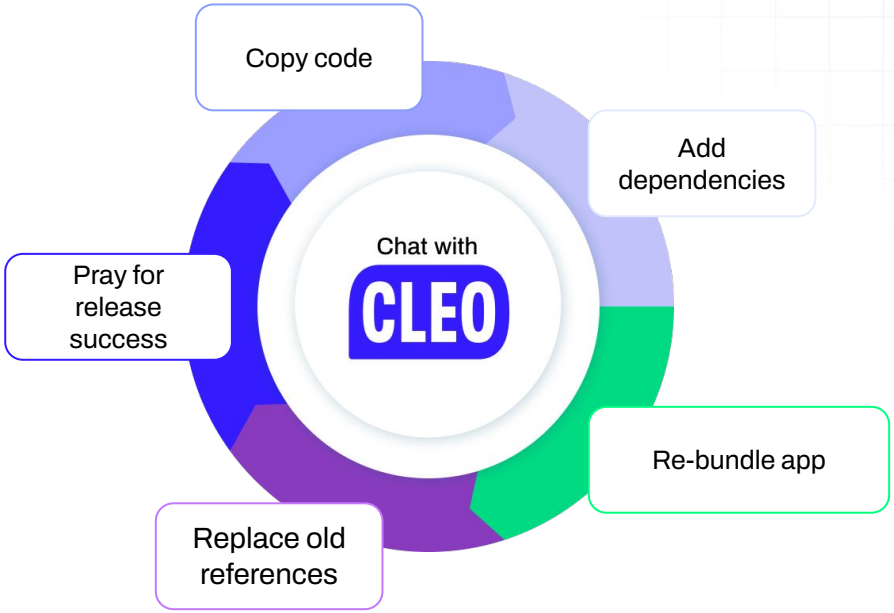
CLEO

Events::Base dependencies

1. DeliveryBoy Wrapper
2. SchemaRegistry Wrapper
3. Event Configuration

Repeatable extraction process

CARRP



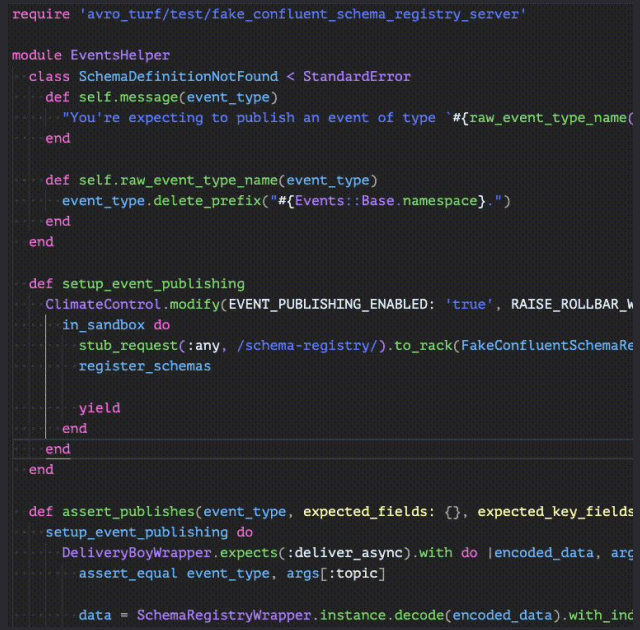
Events::Base ✅

CLEO

Where are we up to?

- ✅ We've installed the local gem in meetcleo
- ✅ All production code & dependencies are under new gem
- ⬜ We've moved the test helper under the gem
- ⬜ We're publishing events from our Rails engines

Step 3: Moving test helper into the gem



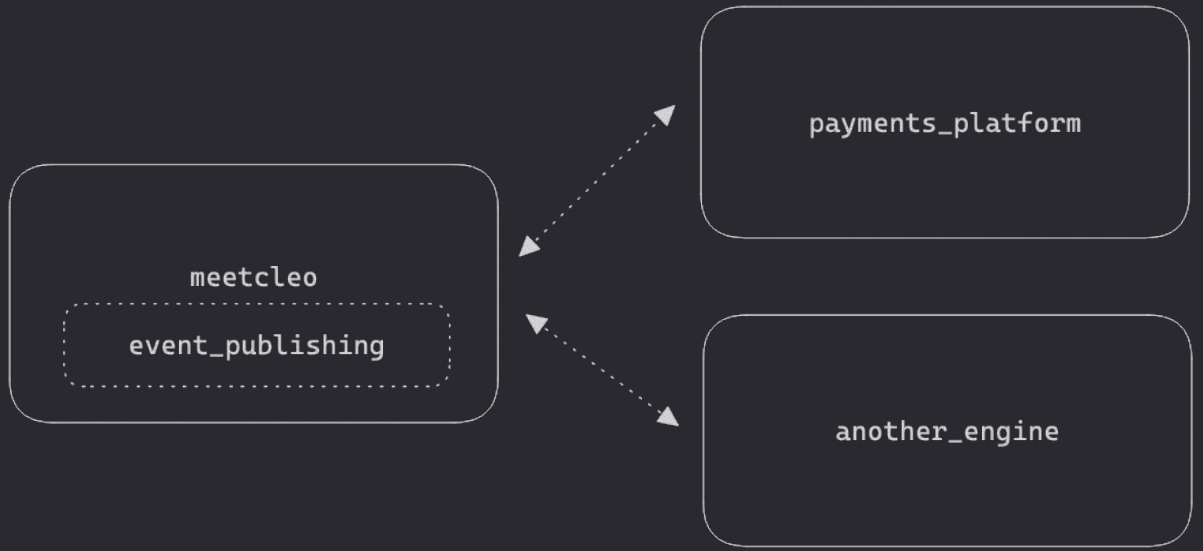
EventsHelper

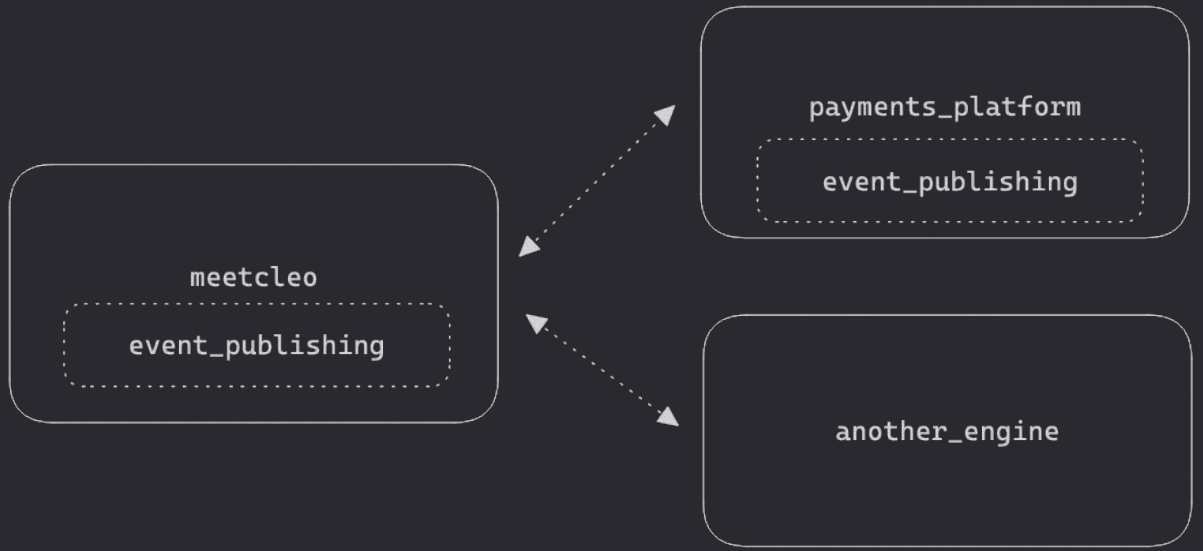
```ruby
# test/test_helper.rb
# or payments_platform/test/test_helper.rb
require 'event_publishing'
EventPublishing.testing
```

Where are we up to?

- ✅ We've installed the local gem in meetcleo
- ✅ All production code & dependencies are under new gem
- ✅ We've moved the test helper under the gem
- ⬜ We're publishing events from our Rails engines

Step 4: Sharing gem with Rails Engines


payments_platform
meetcleo
event_publishing
another_engine


payments_platform
event_publishing
meetcleo
event_publishing
another_engine

Where are we up to?

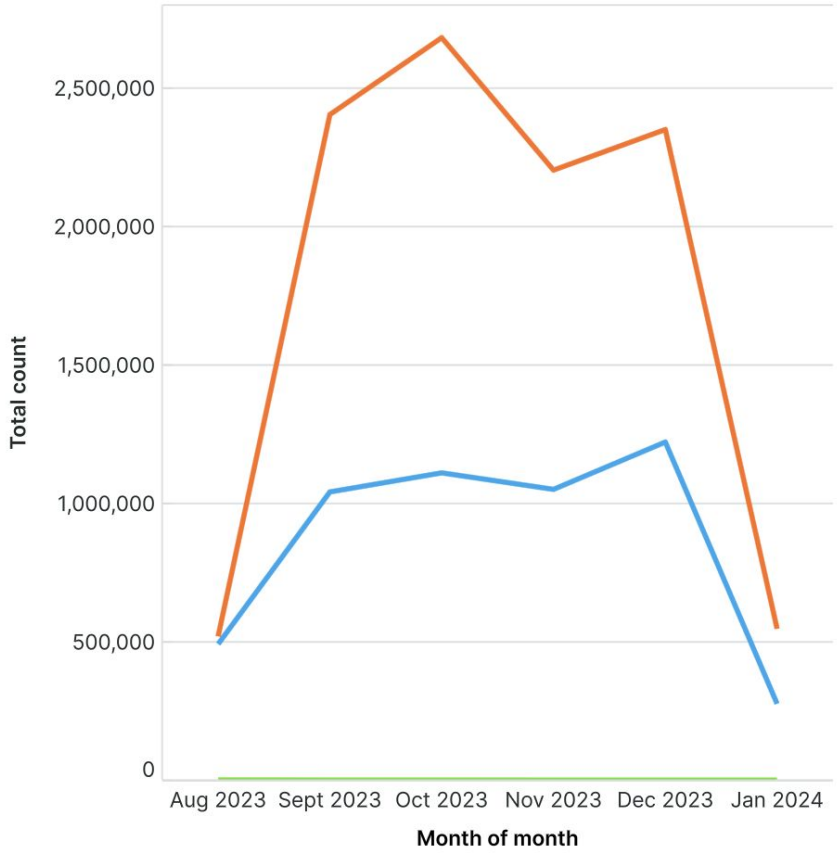
- ✅ We've installed the local gem in meetcleo
- ✅ All production code & dependencies are under new gem
- ✅ We've moved the test helper under the gem
- ✅ We're publishing events from our Rails engines

04

Evaluating the Solution

Was the refactor successful for engineering?
Was it successful for our stakeholders?

CLEO


event
payments_event_0
payments_event_1
payments_event_2
2,500,000
2,000,000
1,500,000
1,000,000
500,000
0
Total count
Aug 2023
Sept 2023
Oct 2023
Nov 2023
Dec 2023
Jan 2024
Month of month

15,904,943

```
meetcleo (main)
$ git log "3a2d60b28b9..HEAD" --oneline event_publishing/ | wc -l
      40
```

```
meetcleo (main)
$ git log "3a2d60b28b9..HEAD" --oneline event_publishing/ | wc -l
      40
```

40 Changes in Gem

0 Downtime

CLEO

Key Takeaways

01 Don’t be too quick to divide up your Rails app, in-process calls are a super-power

02 Boundaries are useful but remember that, in Ruby, nothing is really private so look to use tools to help you not create the big ball of mud

03 Making a gem is not as hard as you think, making a local gem is easier still and often a great place to start

04 Incremental deployments are **well** worth the effort

CLEO

THANK YOU

twitter.com/meetcleo

facebook.com/MeetCleo

instagram.com/meetcleo

linkedin.com/company/cleo-ai

tiktok.com/@meetcleo