

The Magic of Rails

exploring the principles & techniques
behind the framework







Eileen M. Uchitelle

eileencodes.com

@eileencodes





The logo for Ruby on Rails features the word "Rails" in a bold, white, sans-serif font. The letter "R" has a small red vertical bar inside its top loop. Above the text, there is a white graphic element consisting of a semi-circle with several small white diamond shapes at regular intervals.

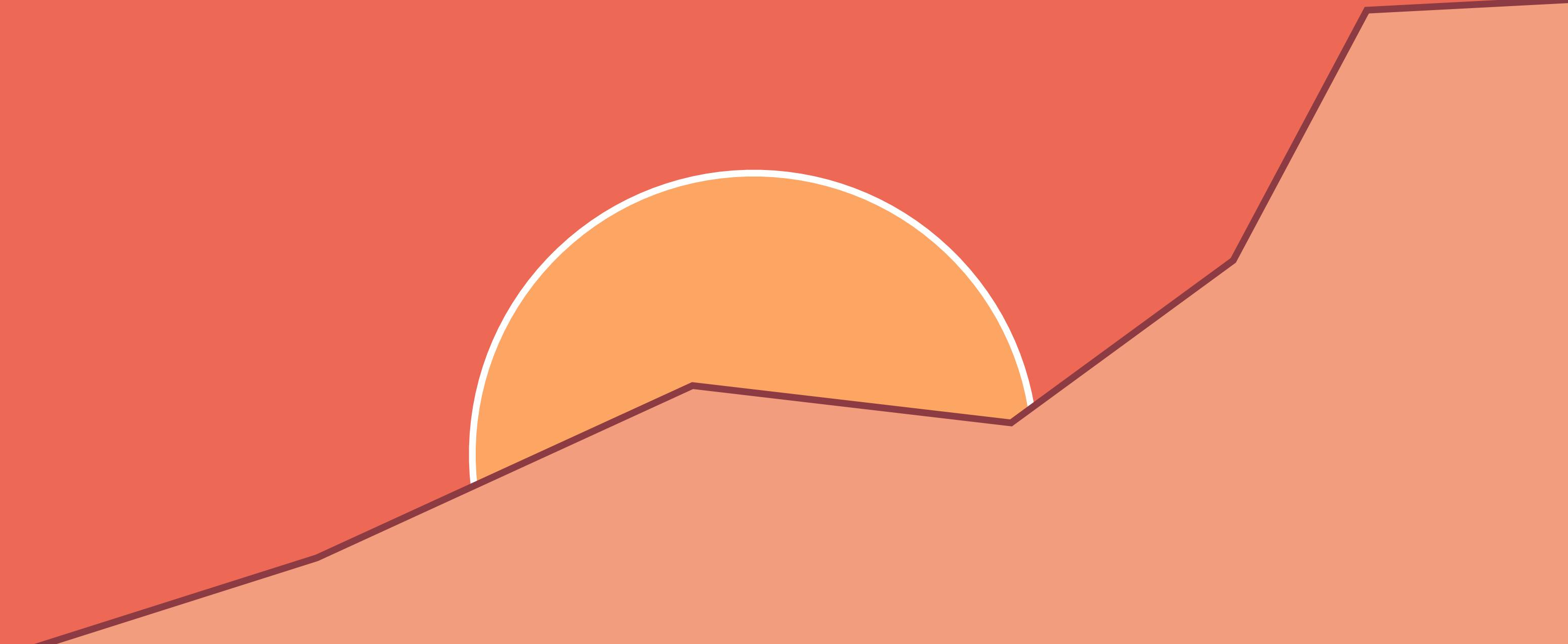
Rails

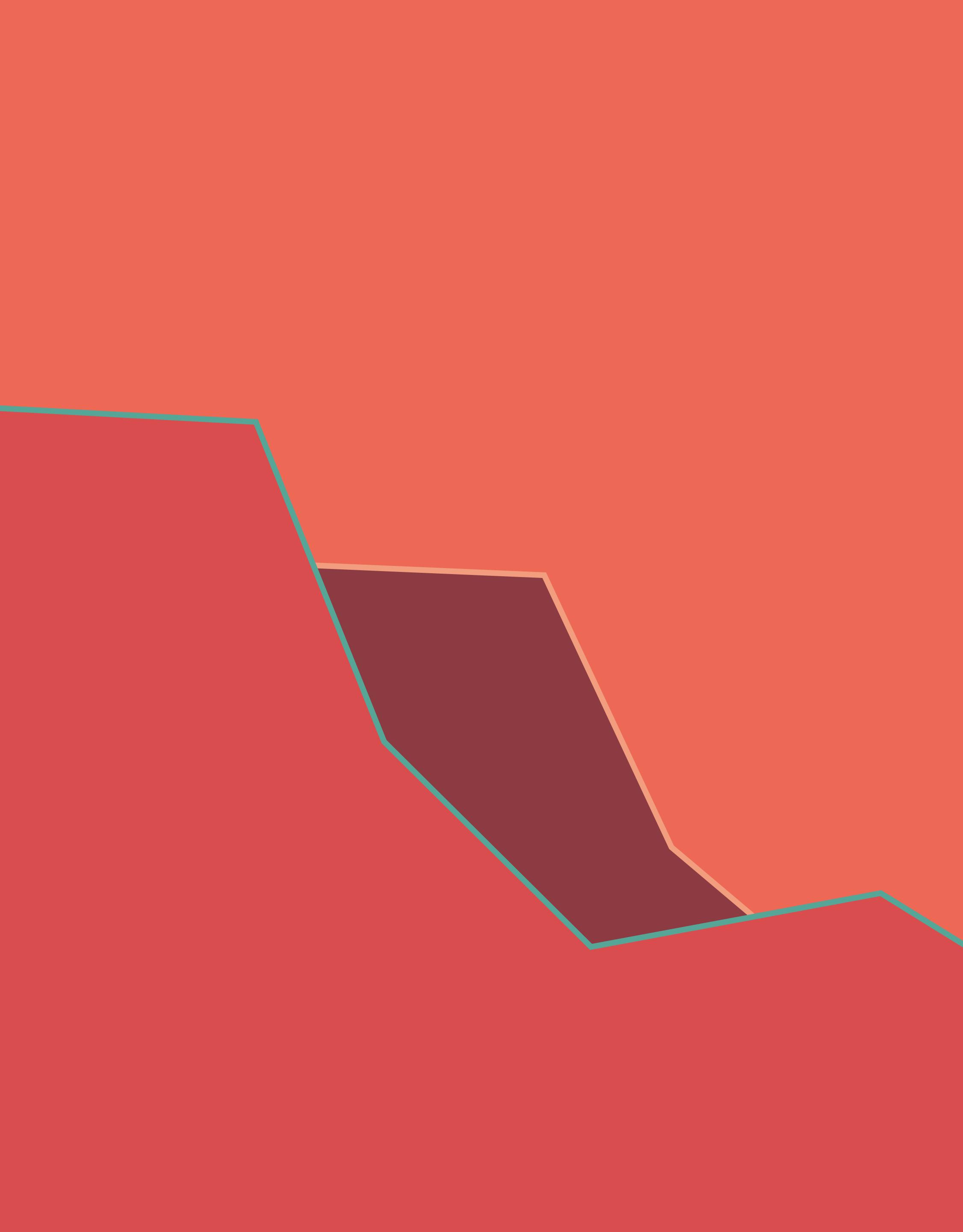
The Magic of Rails

exploring the principles & techniques
behind the framework

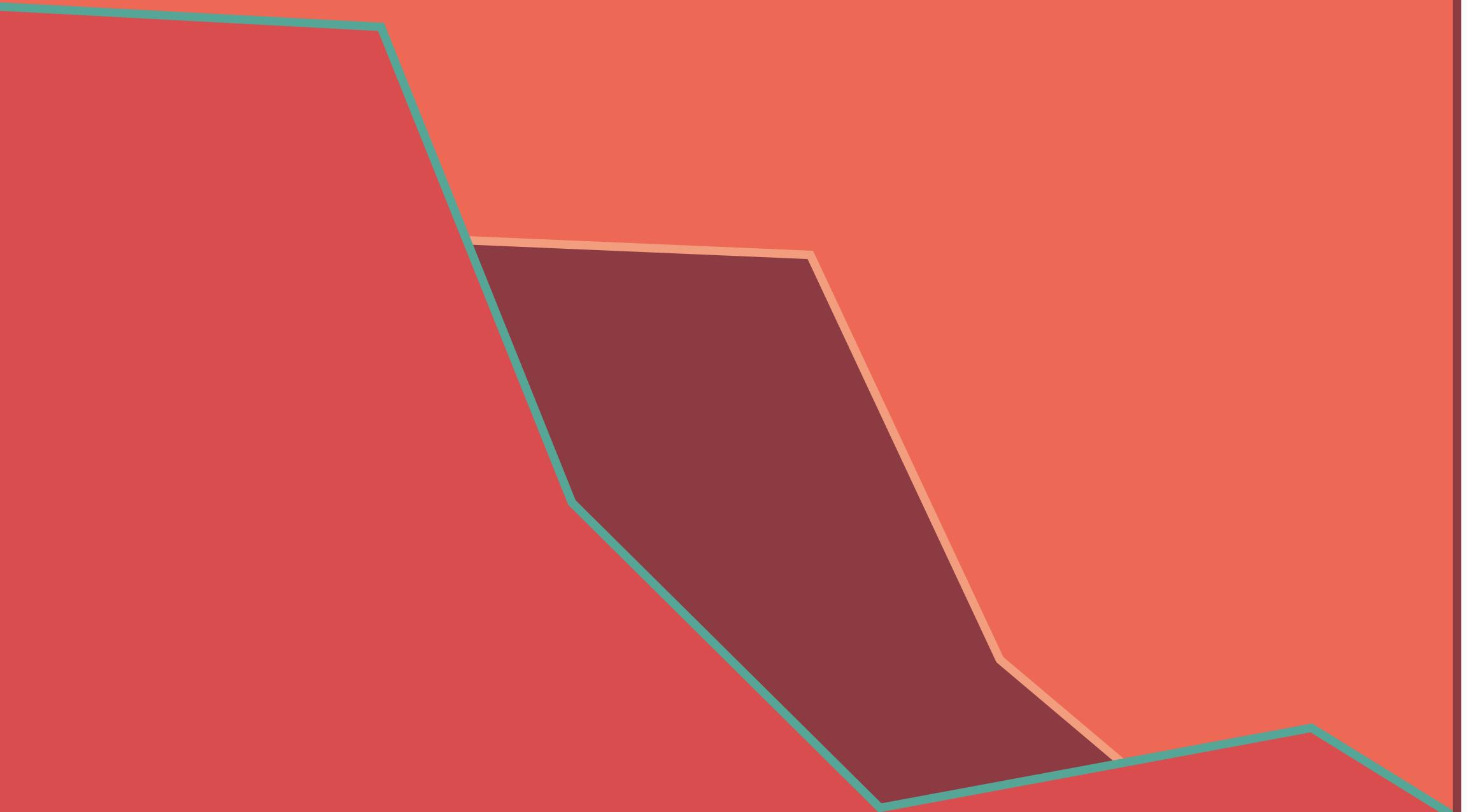


What is Ruby on Rails?

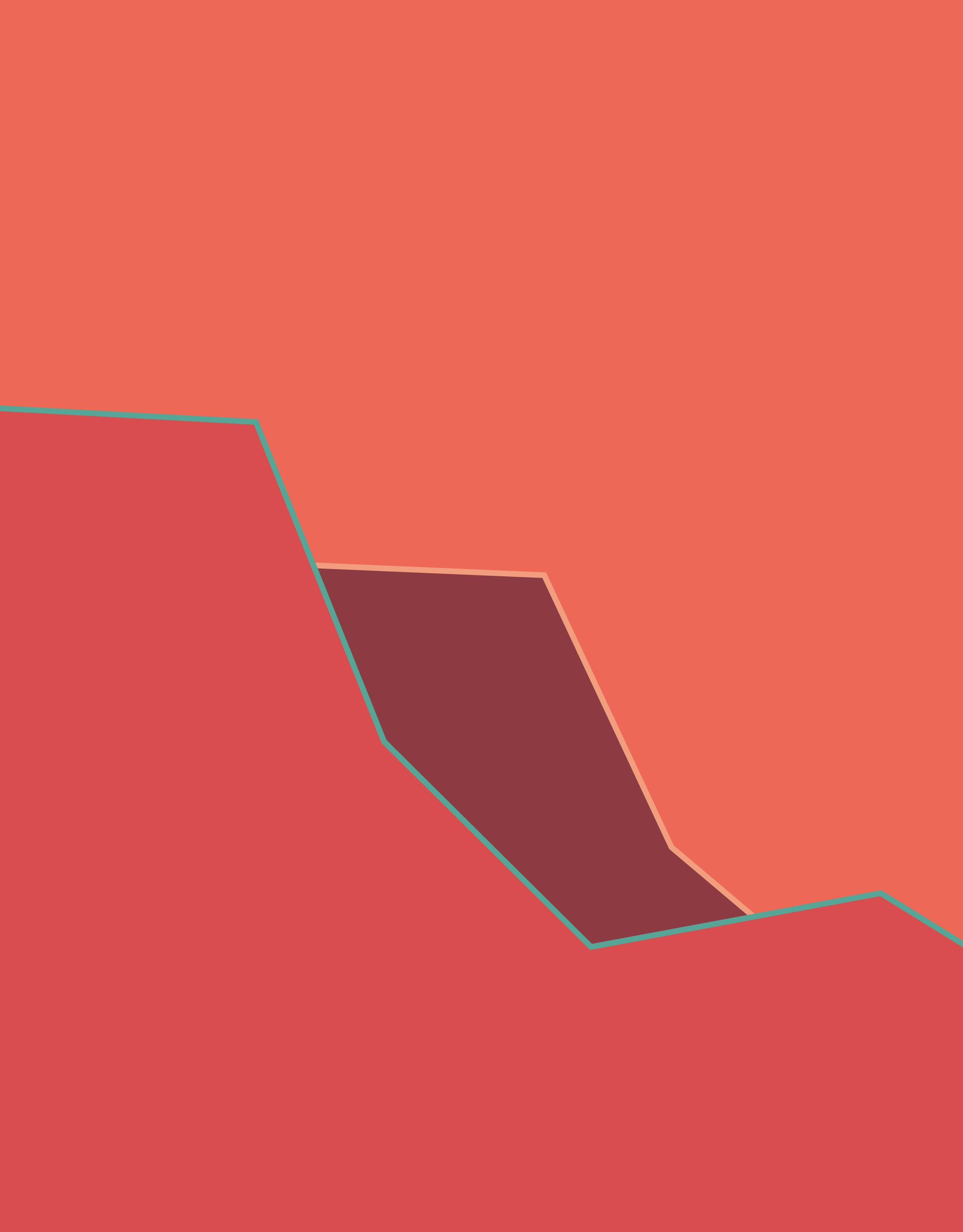


The left side of the slide features a solid red background with abstract geometric shapes in teal, dark brown, and light orange. A teal line forms a right-angled triangle at the top left. A dark brown parallelogram is positioned below it, partially overlapping the teal line. To the right of the parallelogram is a light orange trapezoid. At the bottom center, there is a small teal triangle pointing upwards. The overall aesthetic is minimalist and modern.

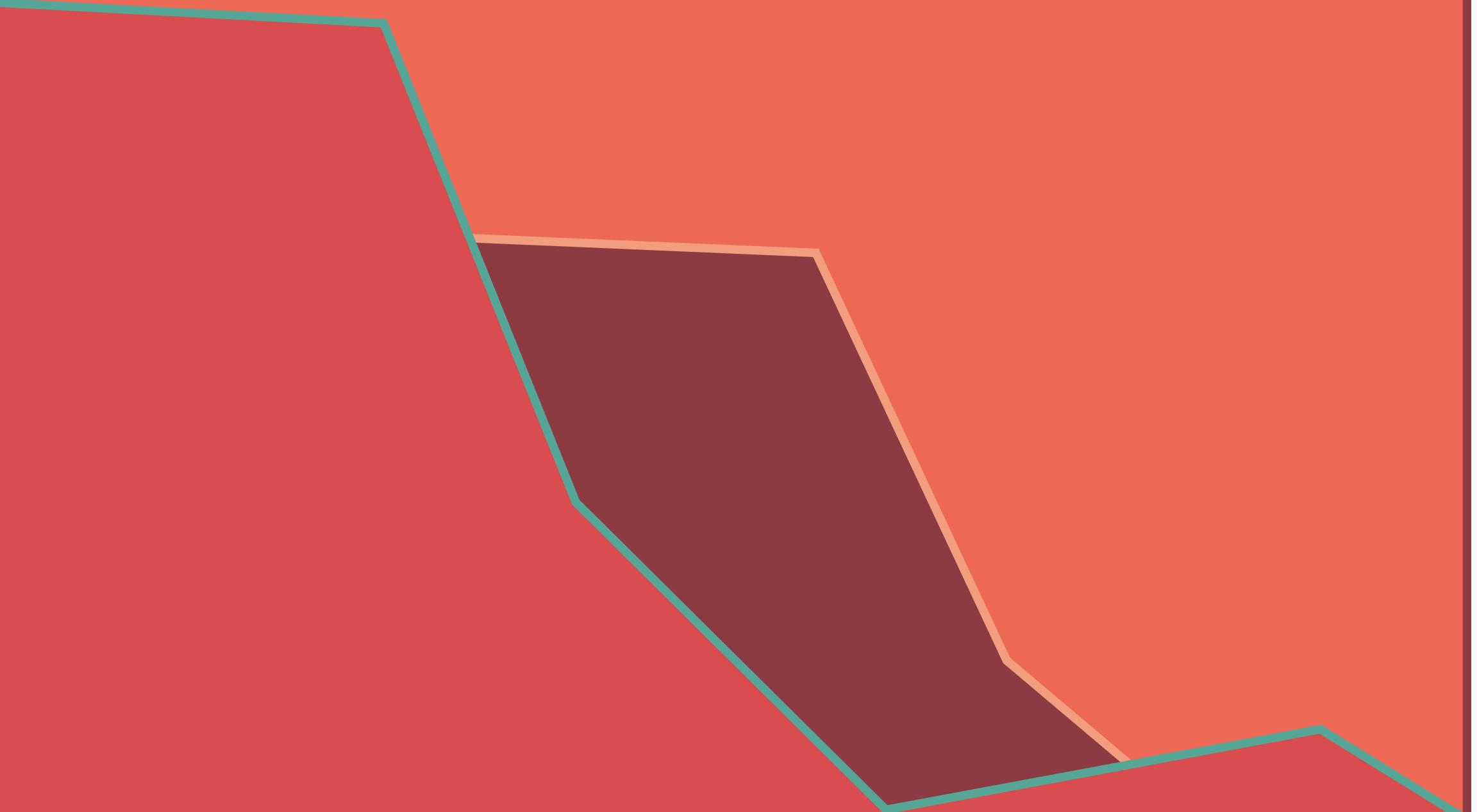
Rails is
modular, but
not fractured



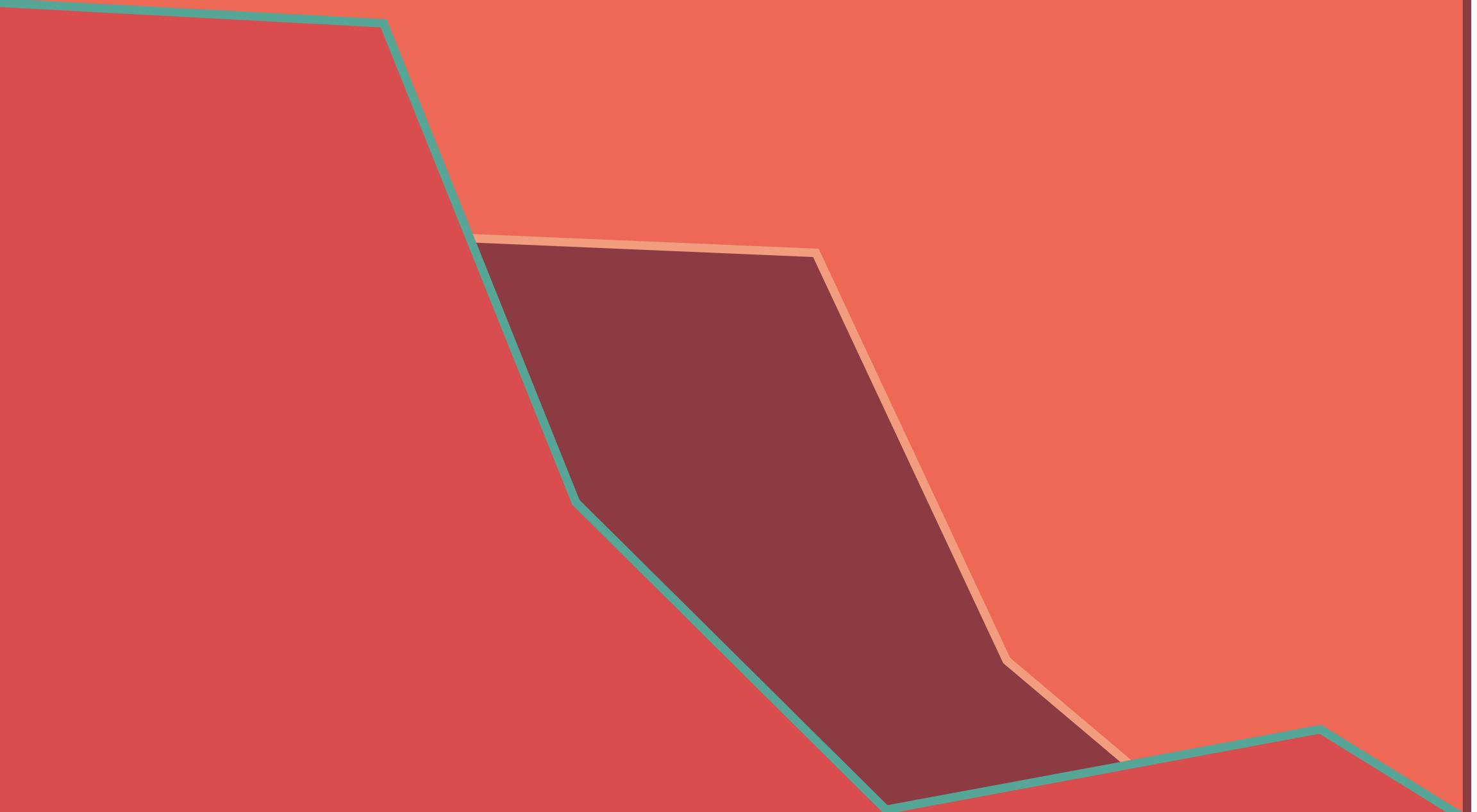
Rails is
designed to
have agnostic
interfaces

The left side of the slide features a solid red background with abstract white geometric shapes. These shapes include a large right-angled triangle at the top-left, a smaller right-angled triangle below it, and a trapezoid in the center. A thin teal line runs diagonally from the top-left corner towards the bottom-right.

Rails is
extracted from
applications

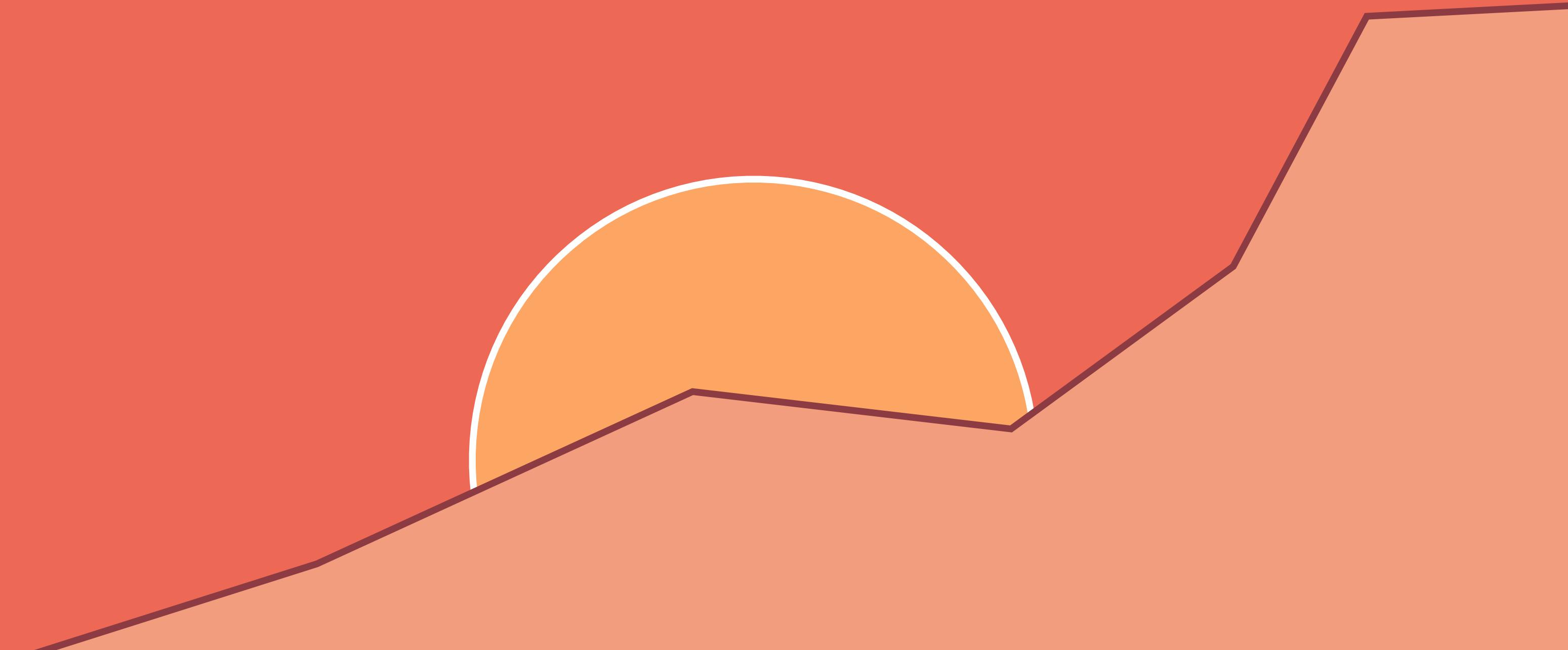


Rails is
made of
simple and
aesthetic APIs

A graphic element in the bottom-left corner consists of several overlapping geometric shapes. It includes a large red triangle pointing upwards, a smaller teal triangle nested within it, and a dark brown trapezoid. The shapes are set against a light orange background.

Rails is
a framework
that takes on
complexity to
empower you

How Rails components are structured



RUBY ON RAILS

Action Cable

Action Mailbox

Action Mailer

Action Pack

Action Text

Action View

Active Job

Active Model

Active Record

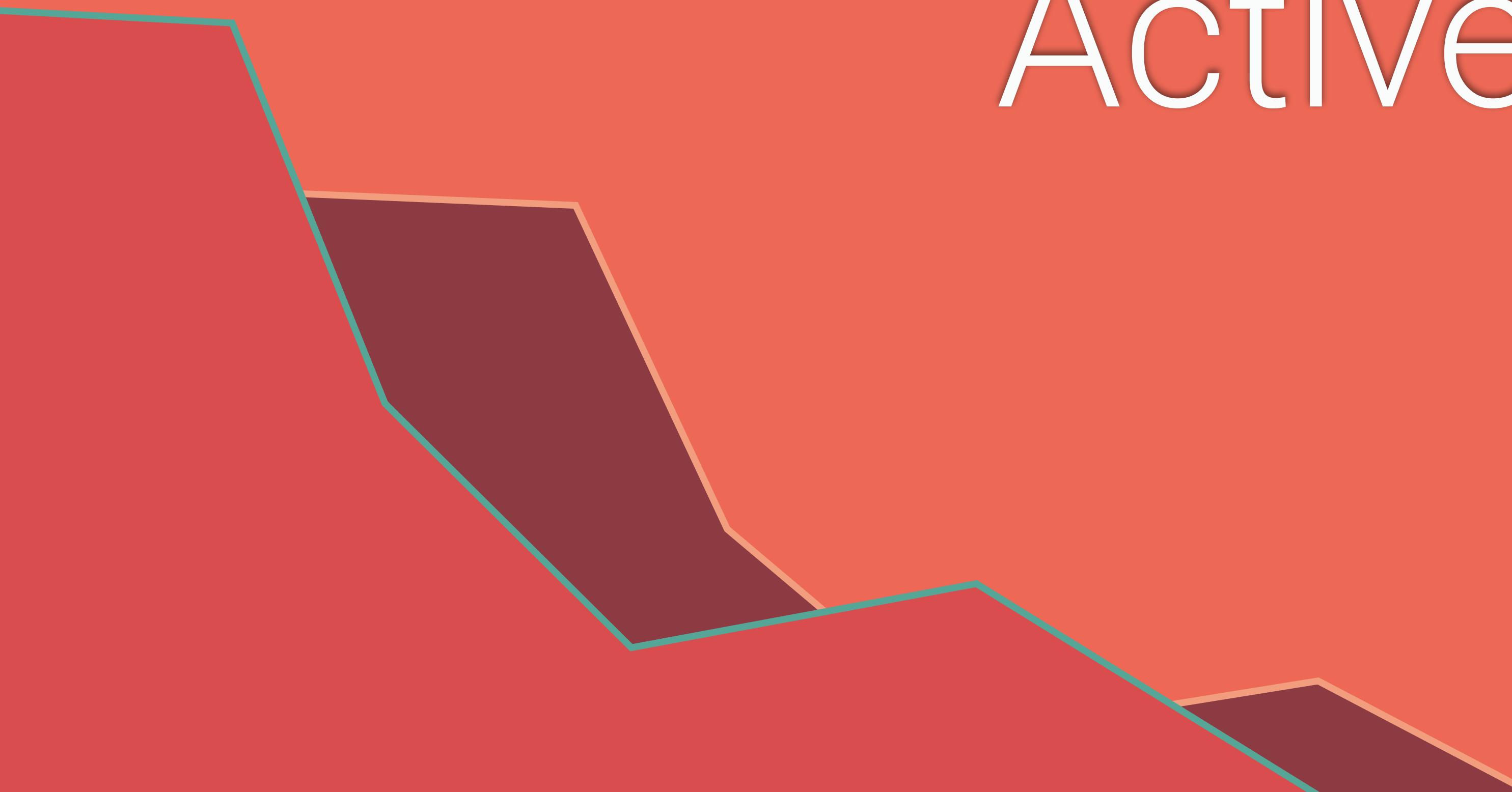
Active Storage

Active Support

Railties

Naming Convention

Active vs Action



NAMING CONVENTION

BACKEND

- Active Record
- Active Support
- Active Model
- Active Job
- Active Storage

NAMING CONVENTION

BACKEND

Active Record
Active Support
Active Model
Active Job
Active Storage

USER FACING

Action Mailer
Action Pack
Action View
Action Cable
Action Text
Action Mailbox

NAMING CONVENTION

BACKEND

Active Record
Active Support
Active Model
Active Job
Active Storage

GLUE

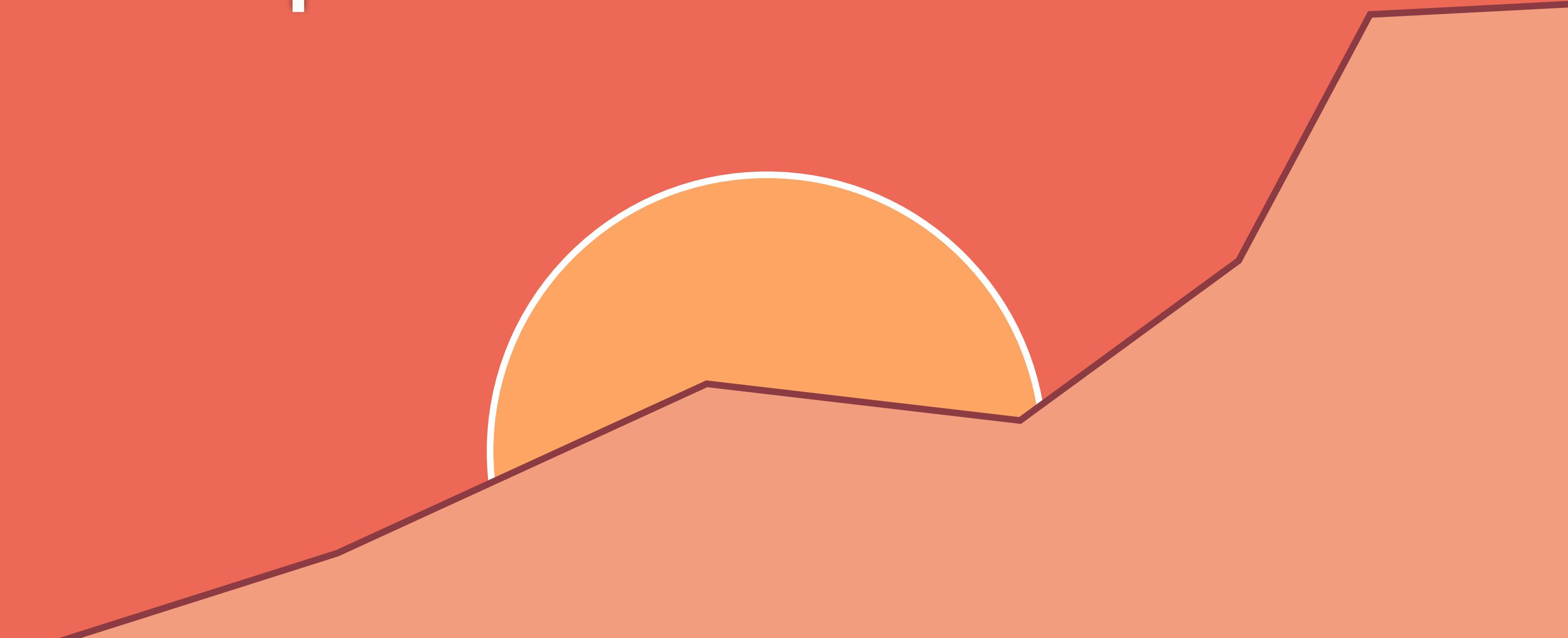
Railties

USER FACING

Action Mailer
Action Pack
Action View
Action Cable
Action Text
Action Mailbox



Architecture & Patterns of Rails components



Architecture & Patterns

the role of Railties

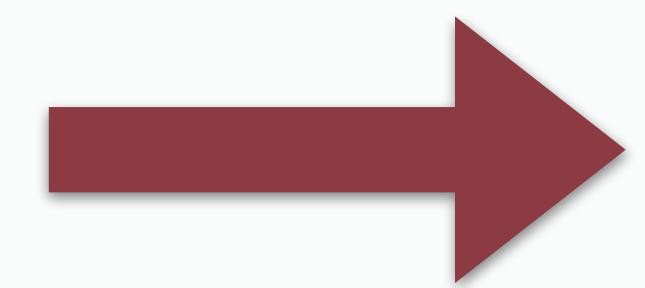


Application



Application

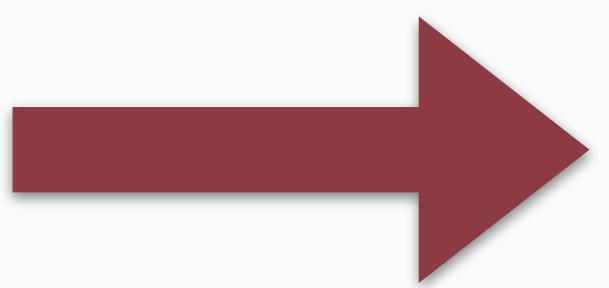
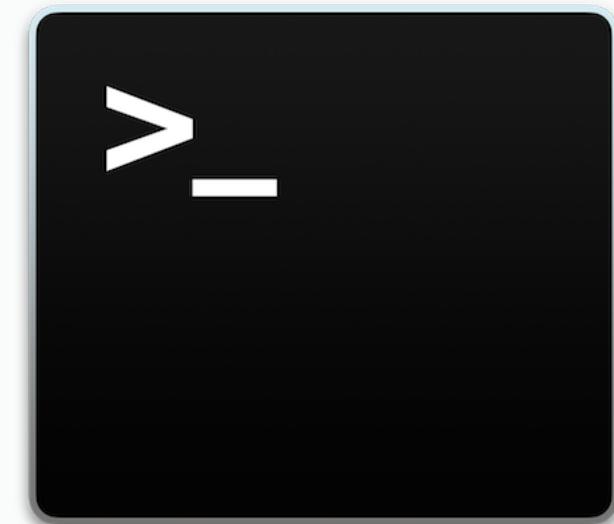
Register hooks



Application

Register hooks

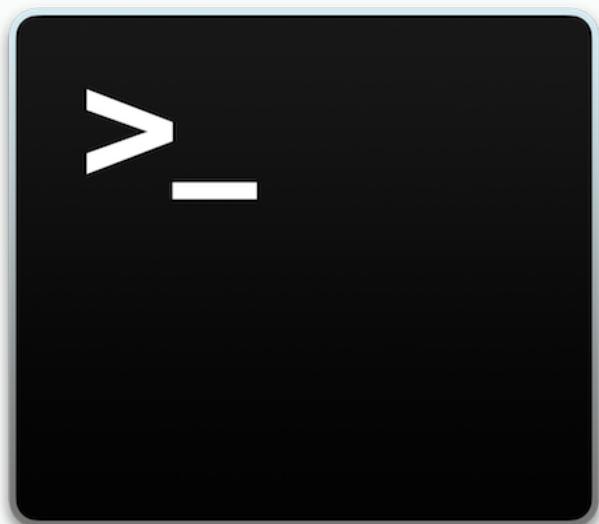
Load components



Application



Register hooks



Load components



Run hooks



```
# railtie.rb

initializer "initializer.name" do
  # do something at initialization
end
```

```
# railties/lib/rails/application.rb

def initializer(name, opts = {}, &block)
  self.class.initializer(name, opts, &block)
end
```

```
# railtie.rb

initializer "initializer.name" do |app|
  app.do_something
  app.config.do_something
end
```

```
# railtie.rb

initializer "initializer.name" do
  ActiveSupport.on_load(:active_record) do
    # do something at initialization
  end
end
```

```
# activerecord/lib/active_record/railtie.rb

initializer "active_record.initialize_database" do
  ActiveSupport.on_load(:active_record) do
    self.configurations =
      Rails.application.config.database_configuration

    establish_connection
  end
end
```

```
# activerecord/lib/active_record/railtie.rb

initializer "active_record.initialize_database" do
  ActiveSupport.on_load(:active_record) do
    self.configurations =
      Rails.application.config.database_configuration

    establish_connection
  end
end
```

```
# activerecord/lib/active_record/railtie.rb

initializer "active_record.initialize_database" do
  ActiveSupport.on_load(:active_record) do
    self.configurations =
      Rails.application.config.database_configuration

    establish_connection
  end
end
```

```
# activerecord/lib/active_record/railtie.rb

initializer "active_record.log_runtime" do
  require "active_record/railties/controller_runtime"
  ActiveSupport.on_load(:action_controller) do
    include ActiveRecord::Railties::ControllerRuntime
  end

  require "active_record/railties/job_runtime"
  ActiveSupport.on_load(:active_job) do
    include ActiveRecord::Railties::JobRuntime
  end
end
```

```
# activerecord/lib/active_record/railtie.rb

initializer "active_record.log_runtime" do
  require "active_record/railties/controller_runtime"
  ActiveSupport.on_load(:action_controller) do
    include ActiveRecord::Railties::ControllerRuntime
  end

require "active_record/railties/job_runtime"
ActiveSupport.on_load(:active_job) do
  include ActiveRecord::Railties::JobRuntime
end
end
```



```
# activerecord/lib/active_record/railtie.rb

initializer "active_record.log_runtime" do
  require "active_record/railties/controller_runtime"
  ActiveSupport.on_load(:action_controller) do
    include ActiveRecord::Railties::ControllerRuntime
  end

  require "active_record/railties/job_runtime"
  ActiveSupport.on_load(:active_job) do
    include ActiveRecord::Railties::JobRuntime
  end
end
```



```
# activejob/lib/active_job/railtie.rb

initializer "active_model.deprecator",
  before: :load_environment_config do |app|

  app.deprecators[:active_model] =
    ActiveModel.deprecator
end
```

```
# activejob/lib/active_job/railtie.rb
```

```
initializer "active_model.deprecator",  
  before: :load_environment_config do |app|
```

```
  app.deprecators[:active_model] =  
    ActiveModel.deprecator  
end
```



- Railties are the core of the framework

- Railties are the core of the framework
- Railties control load order and when hooks should be run

- Railties are the core of the framework
- Railties control load order and when hooks should be run
- Enables components to work together without adding dependencies

Architecture & Patterns

Agnostic interfaces



```
if connection.is_a?(PostgresqlAdapter)
# ...
elsif connection.is_a?(Mysql2Adapter)
# ...
elsif connection.is_a?(TrilogyAdapter)
# ...
elsif connection.is_a?(Sqlite3Adapter)
# ...
else
# ...
end
```

```
if connection.is_a?(PostgresqlAdapter)
# ...
elsif connection.is_a?(Mysql2Adapter)
# ...
elsif connection.is_a?(TrilogyAdapter)
# ...
elsif connection.is_a?(Sqlite3Adapter)
# ...
else
# ...
end
```

```
module ActiveRecord
  module ConnectionAdapters
    class AbstractAdapter
      # define interface
    end
  end
end
```

```
module ActiveRecord
  module ConnectionAdapters
    class AbstractAdapter
      # define interface
    end
  end
end
```

```
module ActiveRecord
  module ConnectionAdapters
    class PostgresqlAdapter < AbstractAdapter
      # inherit or redefine interface
    end
  end
end
```

```
connection.supports_foreign_keys?  
=> true
```

```
class AbstractAdapter
  def supports_foreign_keys?
    false
  end
end
```

```
class AbstractAdapter
  def supports_foreign_keys?
    false
  end
end
```

```
class PostgresqlAdapter < AbstractAdapter
  def supports_foreign_keys?
    true
  end
end
```

```
# activestorage/lib/active_storage/service.rb
```

```
module ActiveStorage
  class Service
    def delete(key)
      raise NotImplementedError
    end
  end
end
```

```
# activestorage/lib/active_storage/service/gcs_service.rb
```

```
class ActiveStorage
  class Service::GCSService < Service
    def delete(key)
      instrument :delete, key: key do
        file_for(key).delete
      rescue Google::Cloud::NotFoundError
        # Ignore files already deleted
      end
    end
  end
end
```

@service.delete(key)

- Consistent interface for all supported libraries

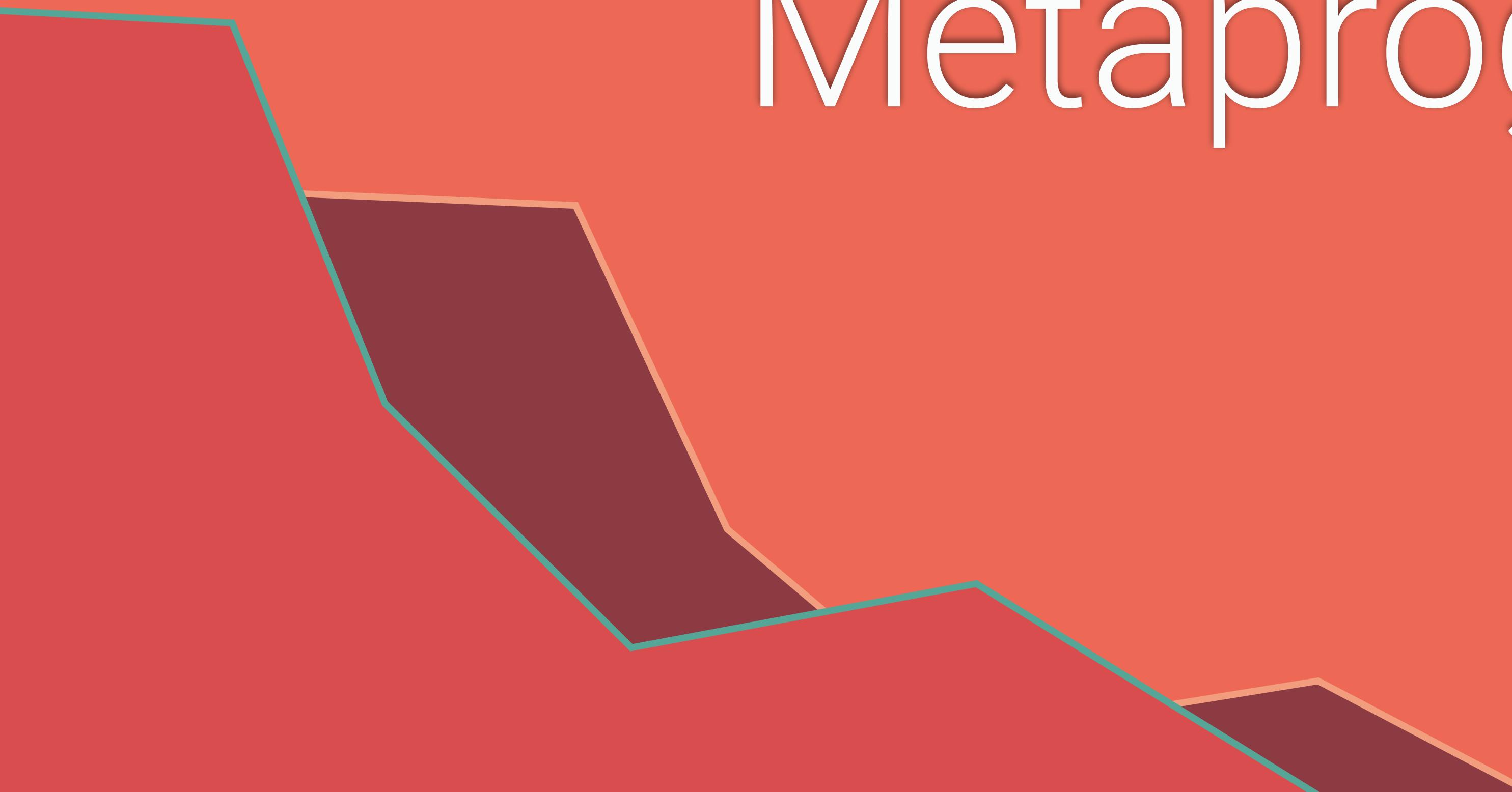
- Consistent interface for all supported libraries
- Simplifies Rails code to avoid using ``is_a?``

- Consistent interface for all supported libraries
- Simplifies Rails code to avoid using ``is_a?``
- Makes it easy for apps to swap out adapters / services

- Consistent interface for all supported libraries
- Simplifies Rails code to avoid using `is_a?`
- Makes it easy for apps to swap out adapters / services
- Lowers the maintenance burden

Architecture & Patterns

Metaprogramming



```
class Post < ApplicationRecord
  has_many :comments
end
```

```
class Comment < ApplicationRecord
  belongs_to :post
end
```

```
post = Post.first  
post.comments  
  
=> [#<Comment:0x000000010e353838...>,  
     #<Comment:0x000000010e3530e0>]
```

```
post = Post.first  
post.method(:comments).source_location
```

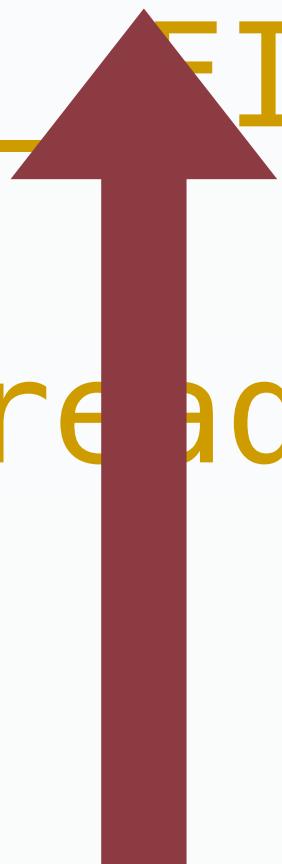
```
post = Post.first
post.method(:comments).source_location
=> ["rails/activerecord/lib/
      active_record/associations/builder/
      association.rb", 103]
```

```
# activerecord/lib/active_record/associations/builder/
association.rb

class ActiveRecord::Associations::Builder
  class Association
    def self.define_readers(mixin, name)
      mixin.class_eval <<-CODE, __FILE__, __LINE__ + 1
      def #{name}
        association(:#{name}).reader
      end
    CODE
    end
  end
end
```

```
# activerecord/lib/active_record/associations/builder/
association.rb

class ActiveRecord::Associations::Builder
  class Association
    def self.define_readers(mixin, name)
      mixin.class_eval <<-CODE, __FILE__, __LINE__ + 1
      def #{name}
        association(:#{name}).reader
      end
    CODE
    end
  end
end
```



```
# activerecord/lib/active_record/associations/builder/
association.rb

class ActiveRecord::Associations::Builder
  class Association
    def self.define_readers(mixin, name)
      mixin.class_eval <<-CODE, __FILE__, __LINE__ + 1
        def #{name}
          association(:#{name}).reader
        end
      CODE
    end
  end
end
```



Post::GeneratedAssociationMethods

```
# activerecord/lib/active_record/associations/builder/  
association.rb
```

```
class ActiveRecord::Associations::Builder
```

```
  class Association
```

```
    def self.define_readers(mixin, name)
```

```
      mixin.class_eval <<-CODE, __FILE__, __LINE__ + 1
```

```
        def #{name}
```

```
          association(:#{name}).reader
```

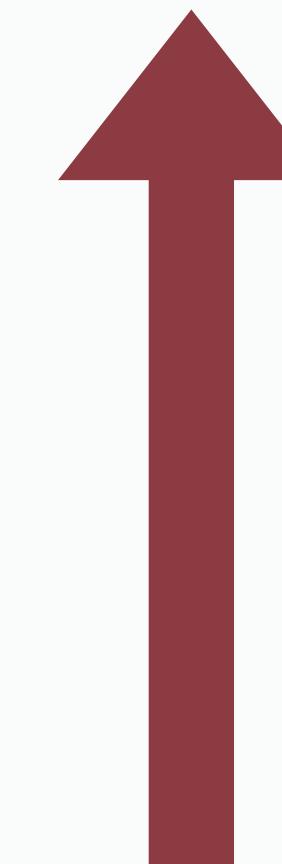
```
        end
```

```
      CODE
```

```
    end
```

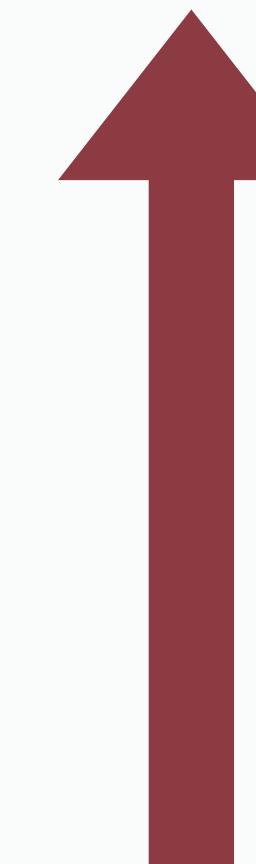
```
  end
```

```
end
```



```
# activerecord/lib/active_record/associations/builder/  
association.rb
```

```
class ActiveRecord::Associations::Builder  
  class Association  
    def self.define_writers(mixin, name)  
      mixin.class_eval <<-CODE, __FILE__, __LINE__ + 1  
      def #{name}=(value)  
        association(:#{name}).writer(value)  
      end  
    end  
  end  
end
```



```
# activerecord/lib/active_record/associations/builder/
association.rb

class ActiveRecord::Associations::Builder
  class Association
    def self.define_writers(mixin, name)
      mixin.class_eval <<-CODE, __FILE__, __LINE__ + 1
        def #{name}=(value)
          association(:#{name}).writer(value)
        end
      CODE
    end
  end
end
```



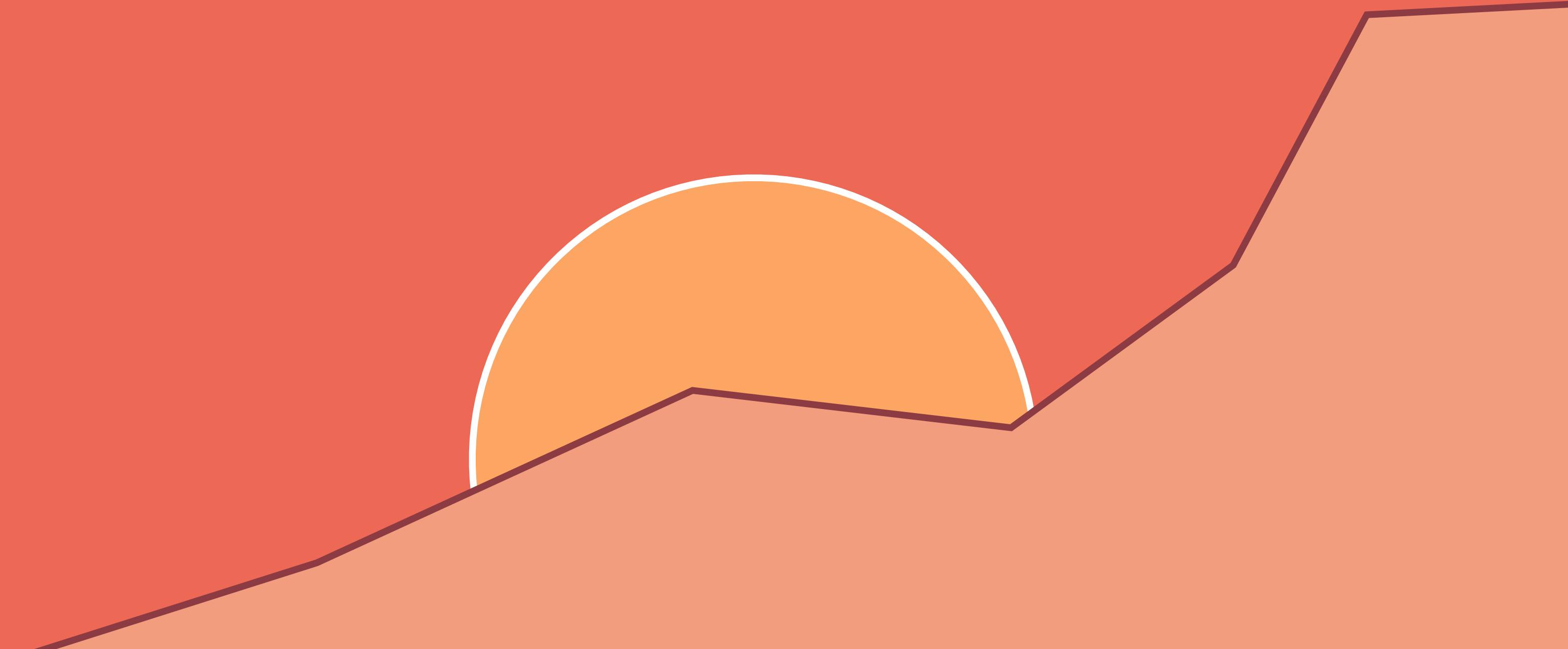
- Powerful tool that enables us to build beautiful, simple APIs

- Powerful tool that enables us to build beautiful, simple APIs
- Hides complexity from your application

- Powerful tool that enables us to build beautiful, simple APIs
- Hides complexity from your application
- Where "Rails Magic" comes from

Maintaining Rails

Why I work on it



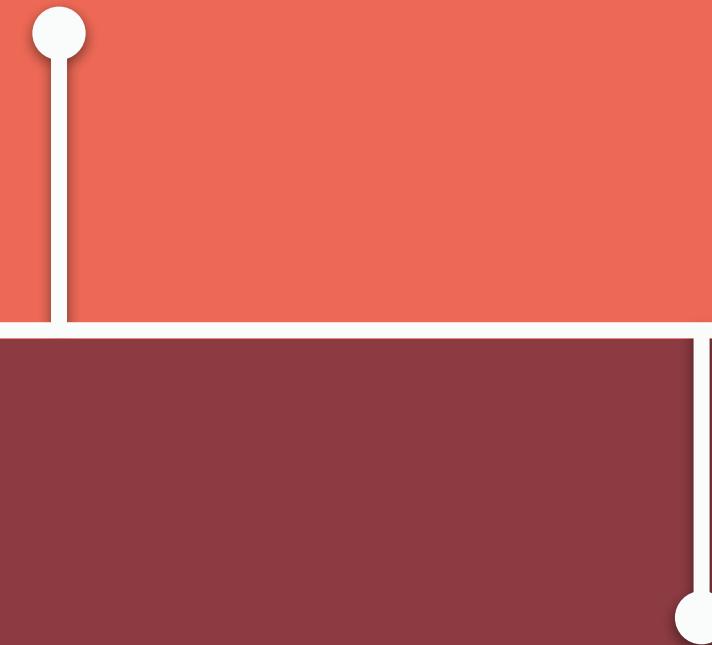
2010

Introduced
to Rails



2010

Introduced
to Rails

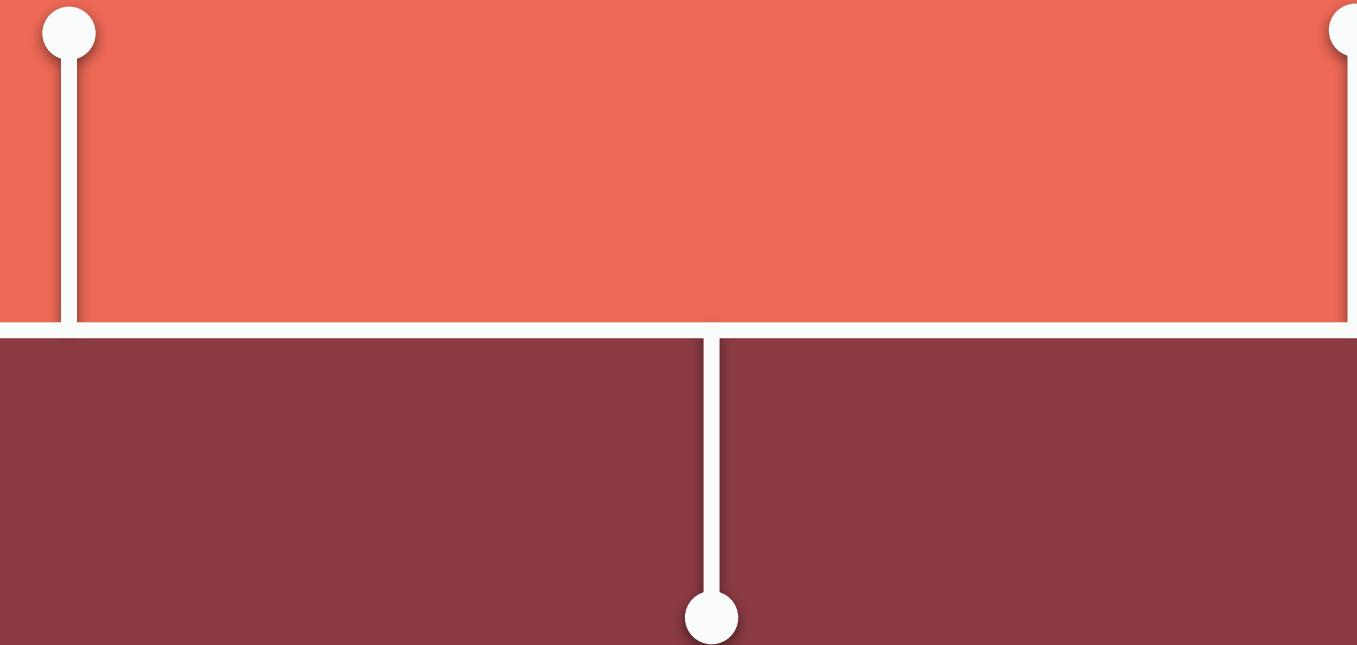


2011

Big Nerd Ranch

2010

Introduced
to Rails



2014

1st conference
1st contribution

2011

Big Nerd Ranch

2010

Introduced
to Rails



2014

1st conference
1st contribution



2011

Big Nerd Ranch



2015

First RailsConf



2010

Introduced
to Rails



2014

1st conference
1st contribution



2017

Join Rails Core



2011

Big Nerd Ranch



2015

First RailsConf



2010

Introduced
to Rails



2014

1st conference
1st contribution



2017

Join Rails Core



2011

Big Nerd Ranch



2015

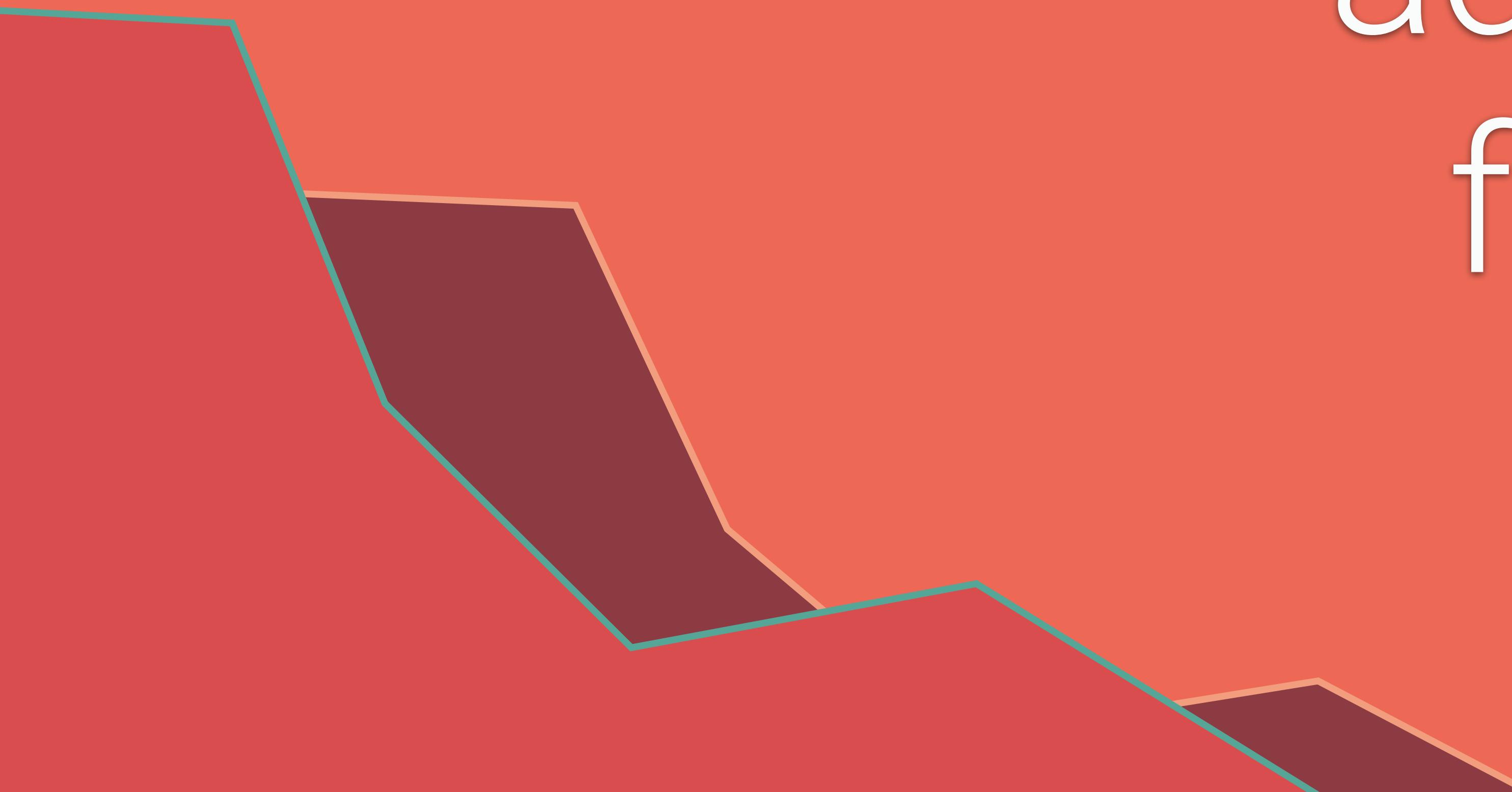
First RailsConf



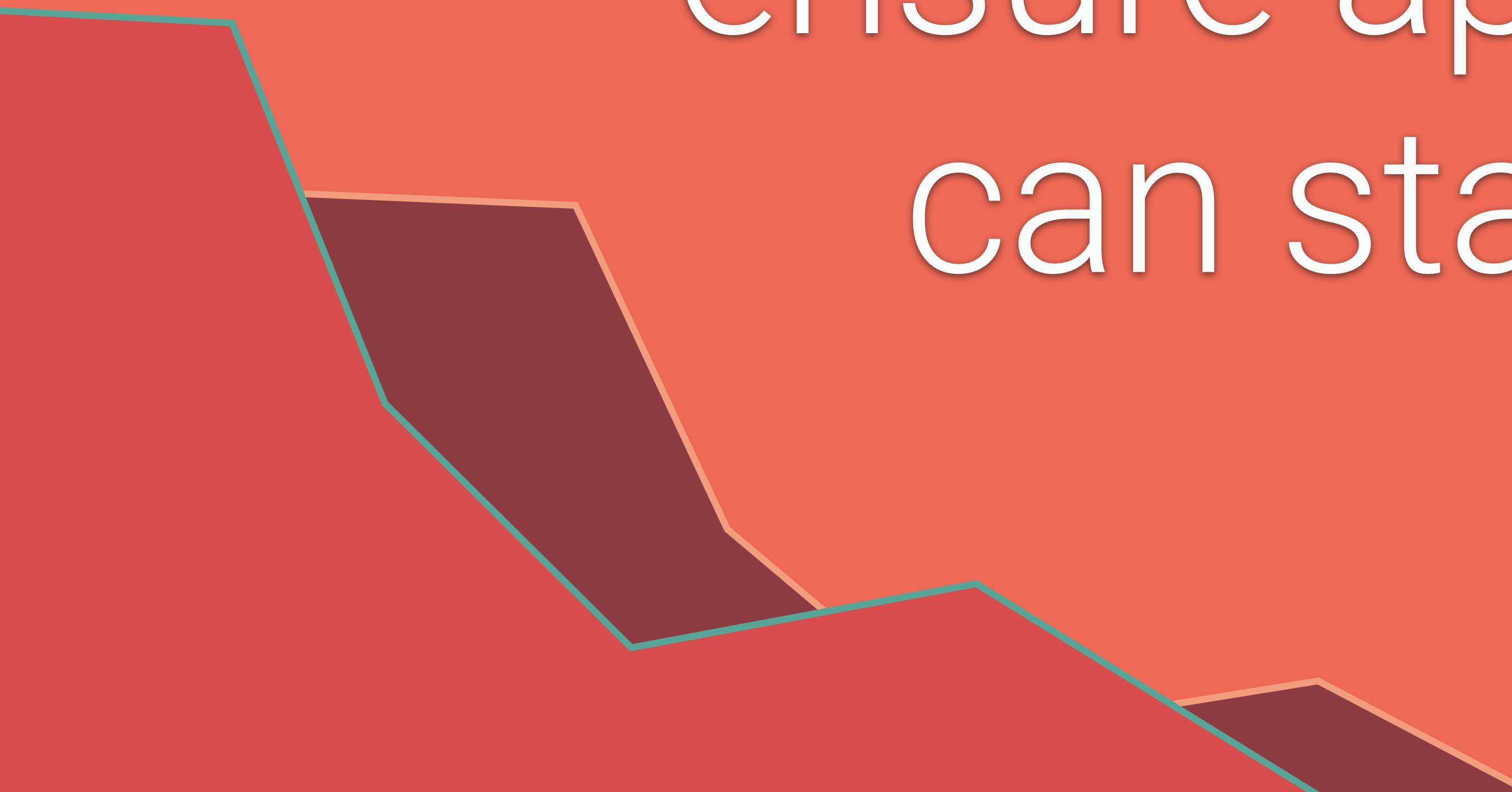
2023

Rails is 20!

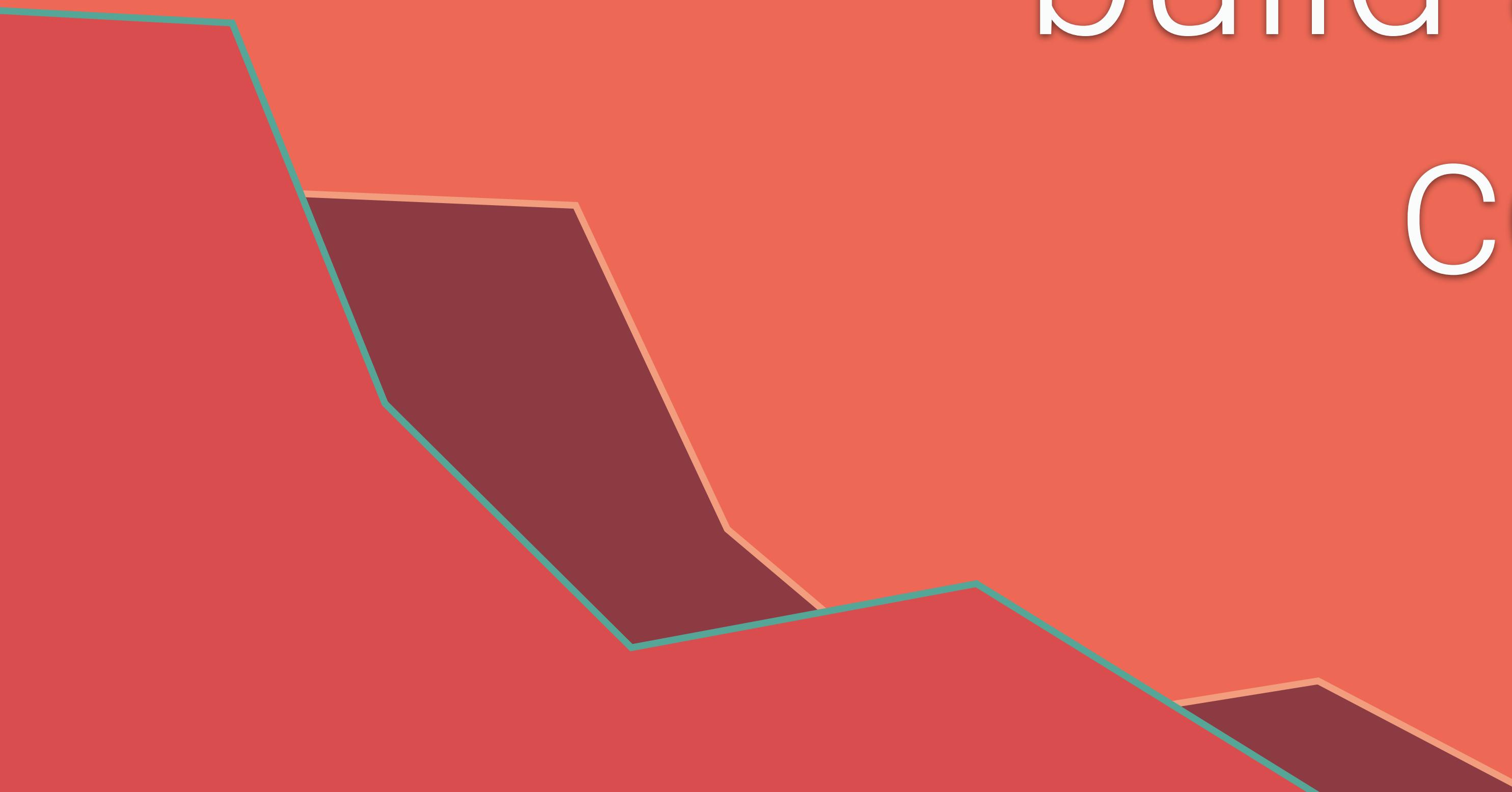




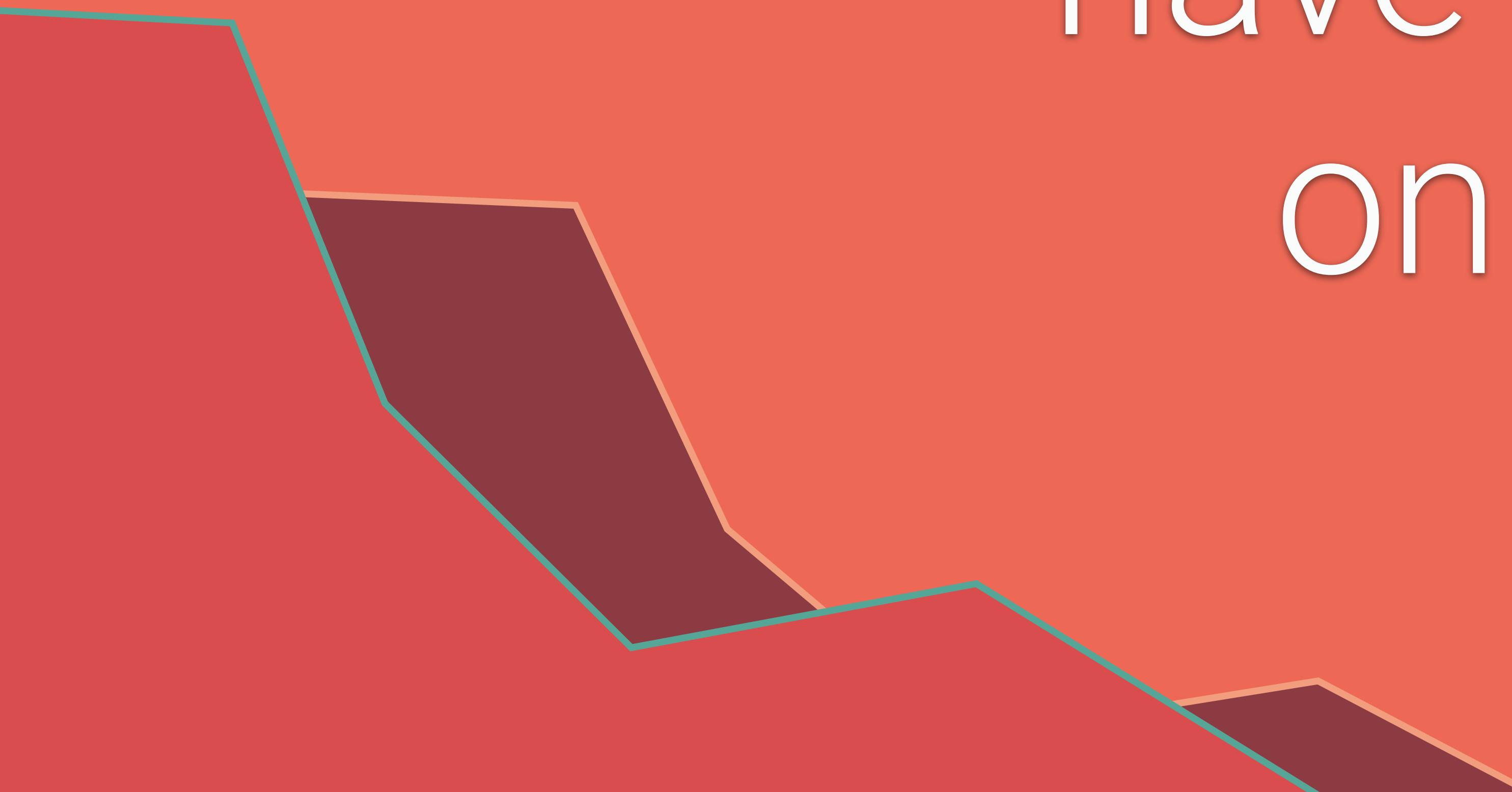
I work on Rails to
advance the
framework



I work on Rails to
ensure applications
can stay on Rails

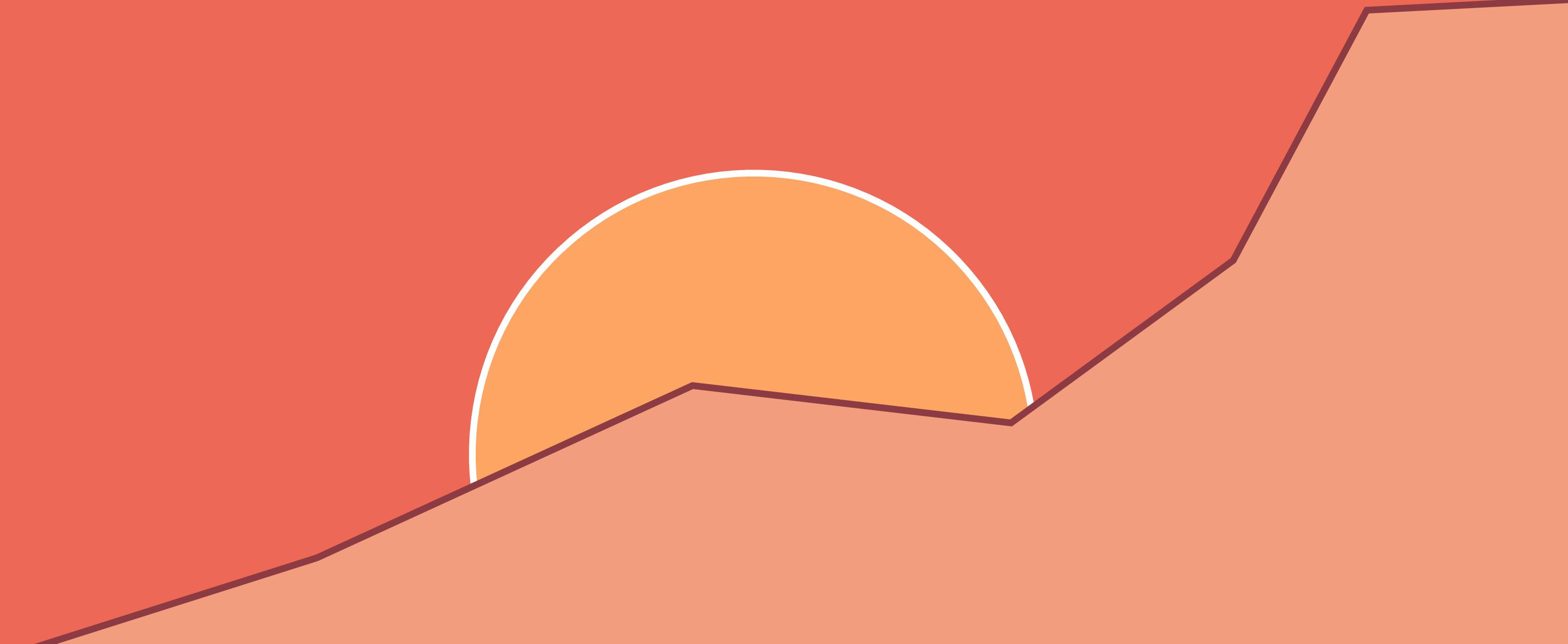


I work on Rails to
build a stronger
community



I work on Rails to
have an impact
on the future

Rails is so much more than just a framework



Rails is
inspiring

Rails is
empowering

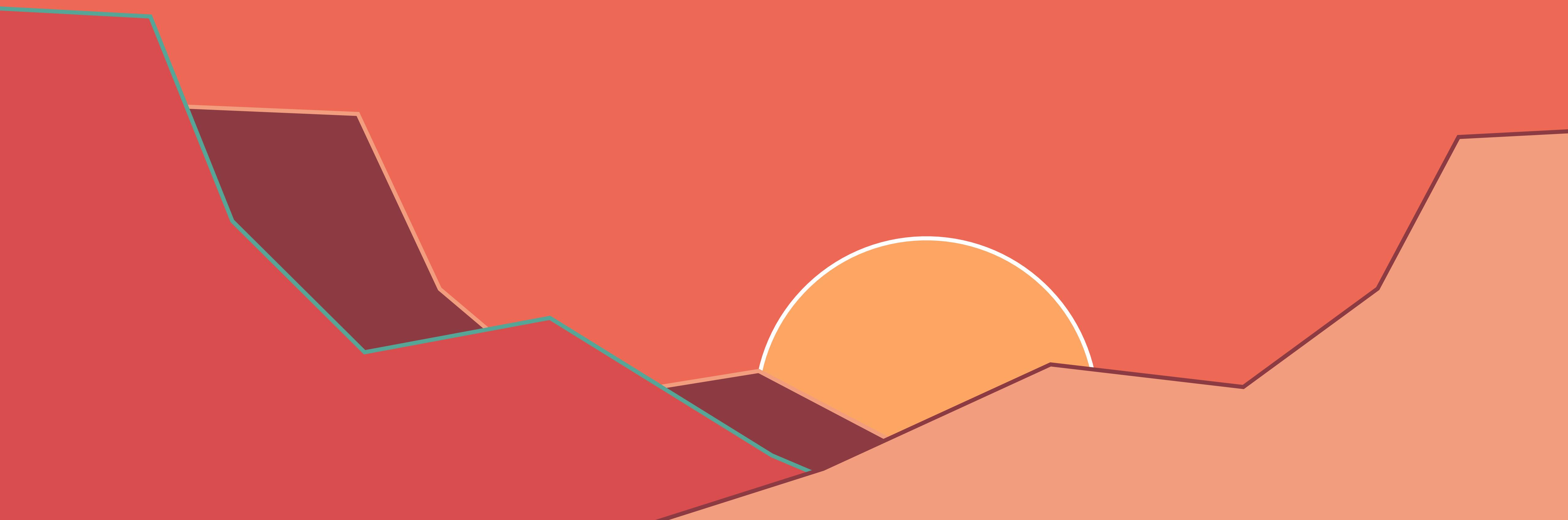
Rails is
imperfect

Rails is
the applications we build

Rails is
the team behind it

Rails is
the community

Rails *is* magic



Thank You!





Eileen M. Uchitelle

eileencodes.com

@eileencodes

Senior Staff Engineer @ Shopify
Rails Core Team