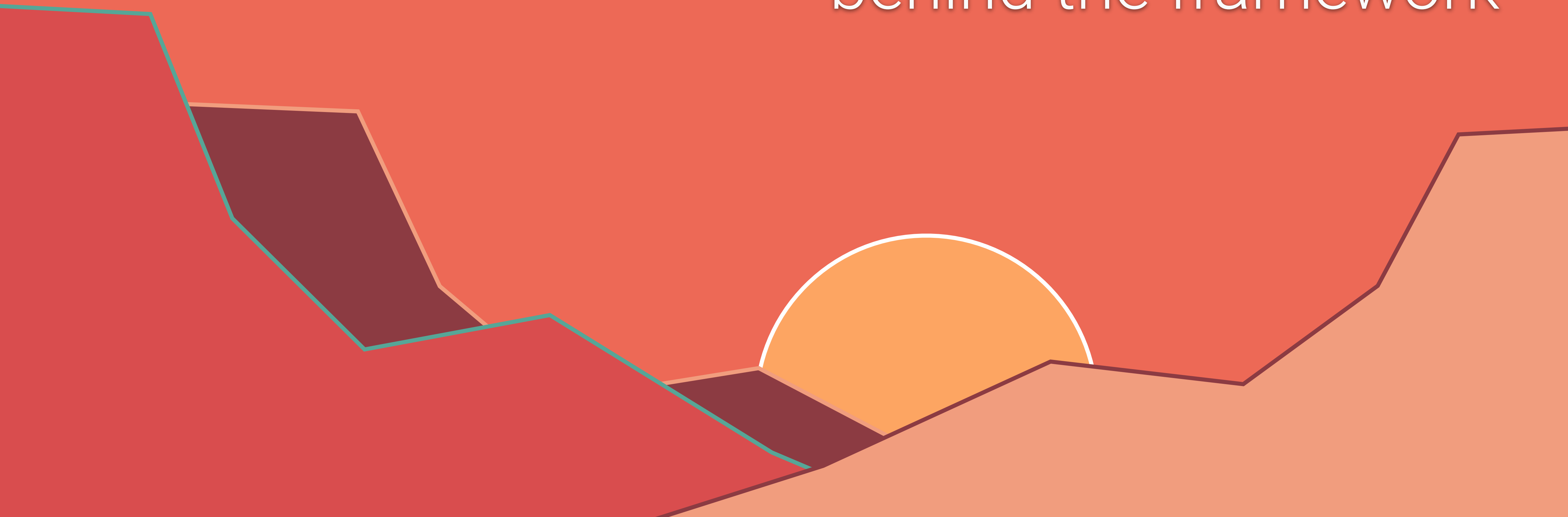


# The Magic of Rails

exploring the principles & techniques  
behind the framework









Hello RailsConf! I'm  
**Eileen M. Uchitelle**

@eileencodes

@eileencodes@ruby.social



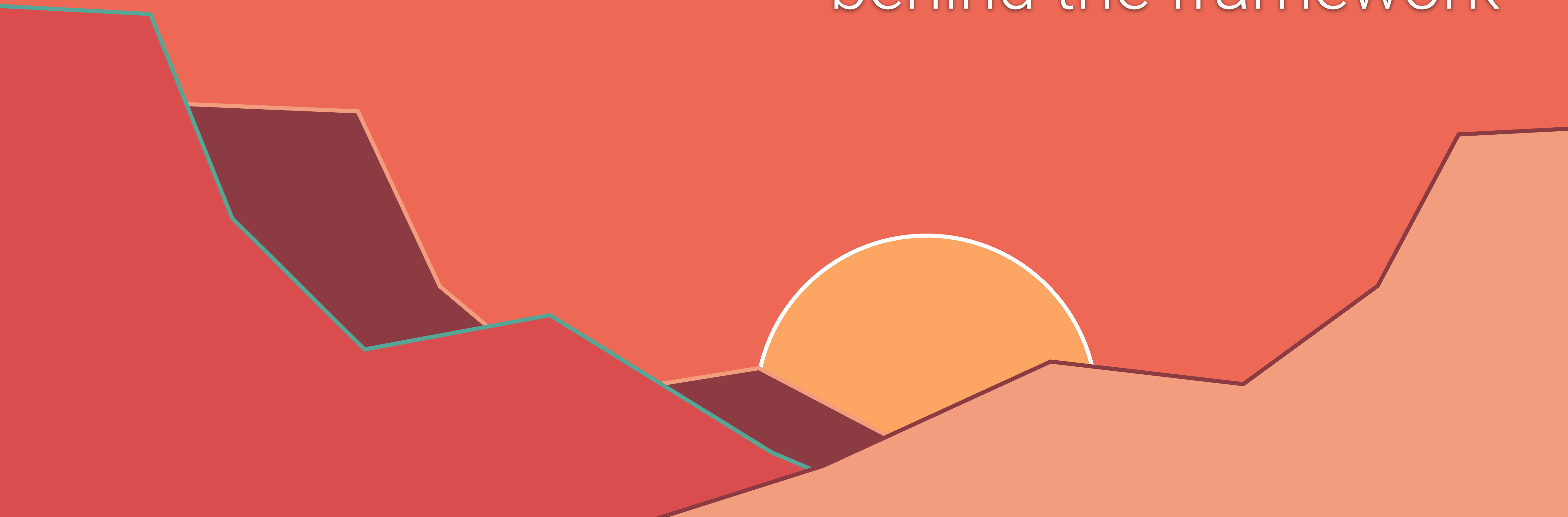
*shopify*



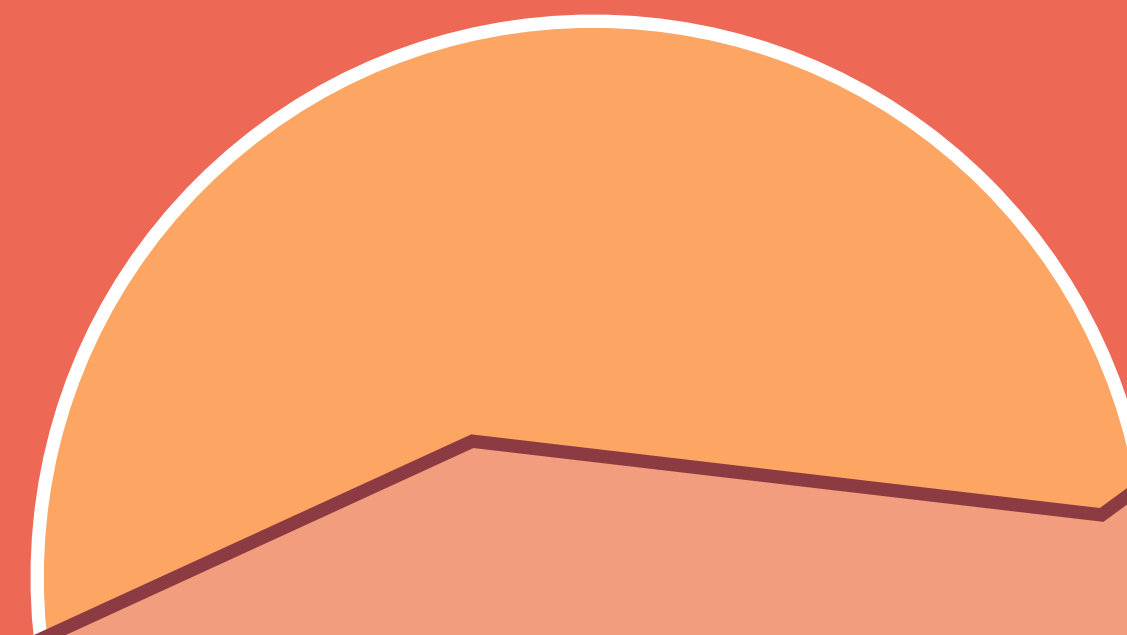


# The Magic of Rails

exploring the principles & techniques  
behind the framework



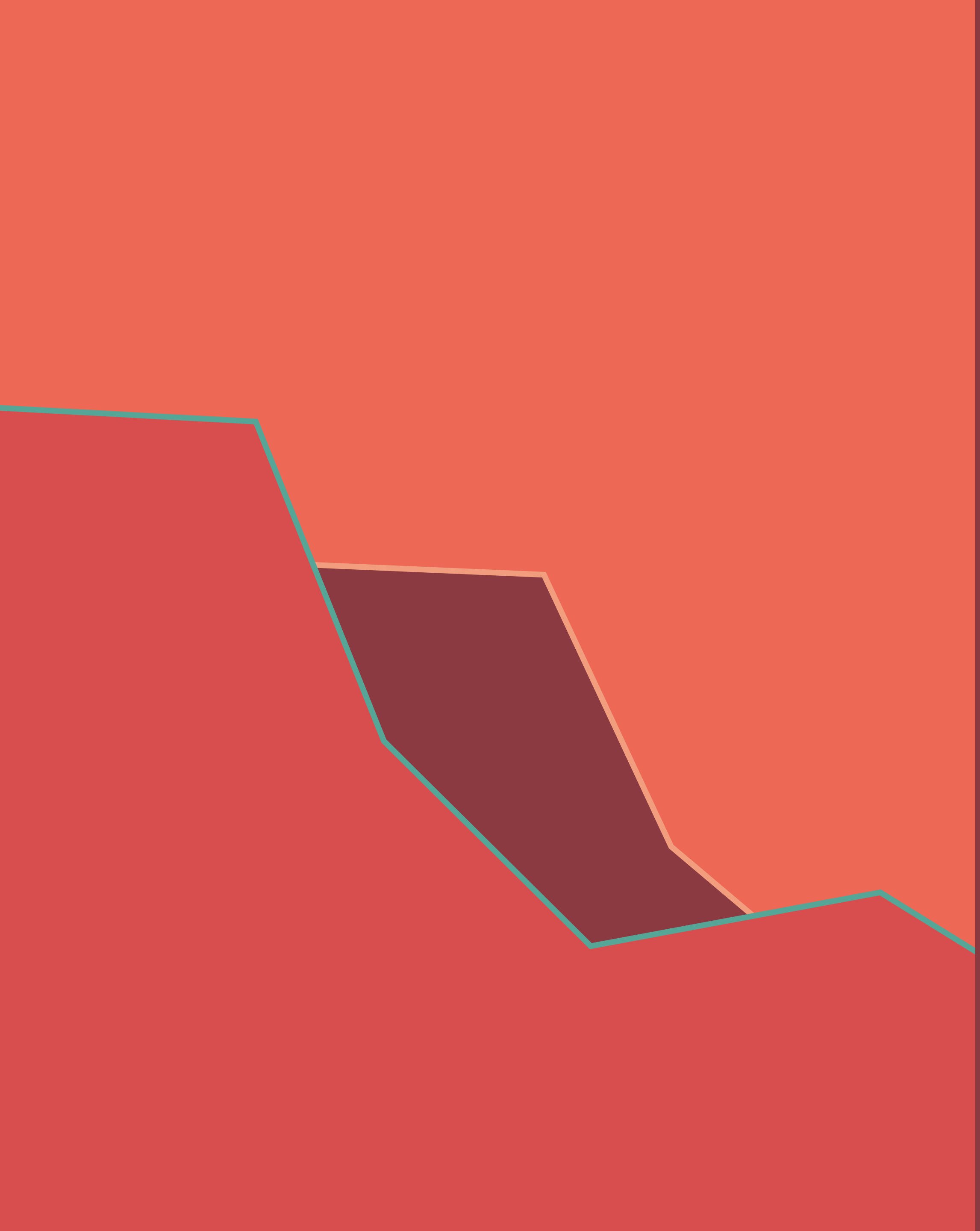
# What is Ruby on Rails?



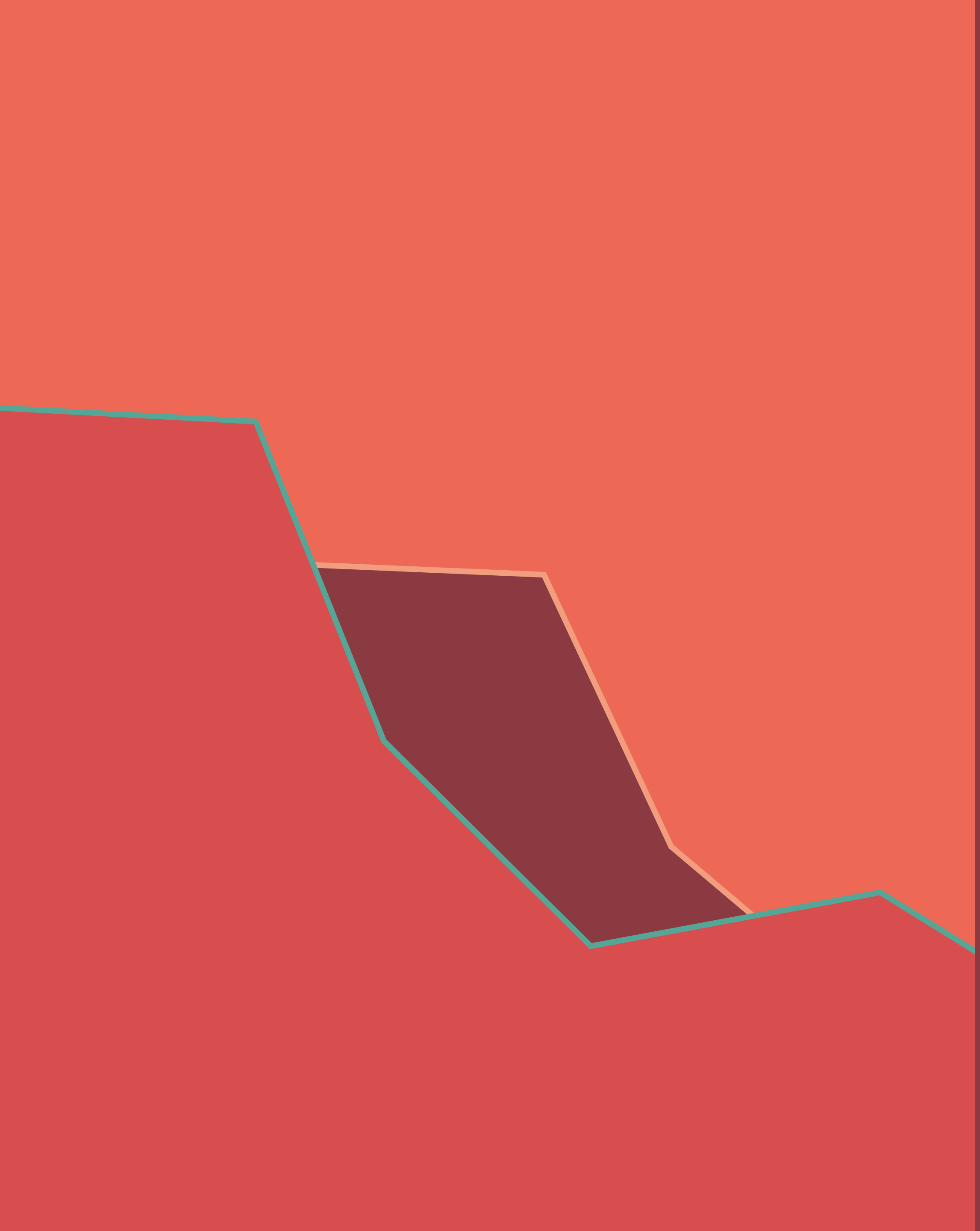
The background features a solid orange-red upper section and a darker red lower section separated by a jagged, hand-drawn line. A dark red, irregular polygon is positioned in the lower-left area, partially overlapping the darker red background.

**Rails is**  
modular, but  
not fractured

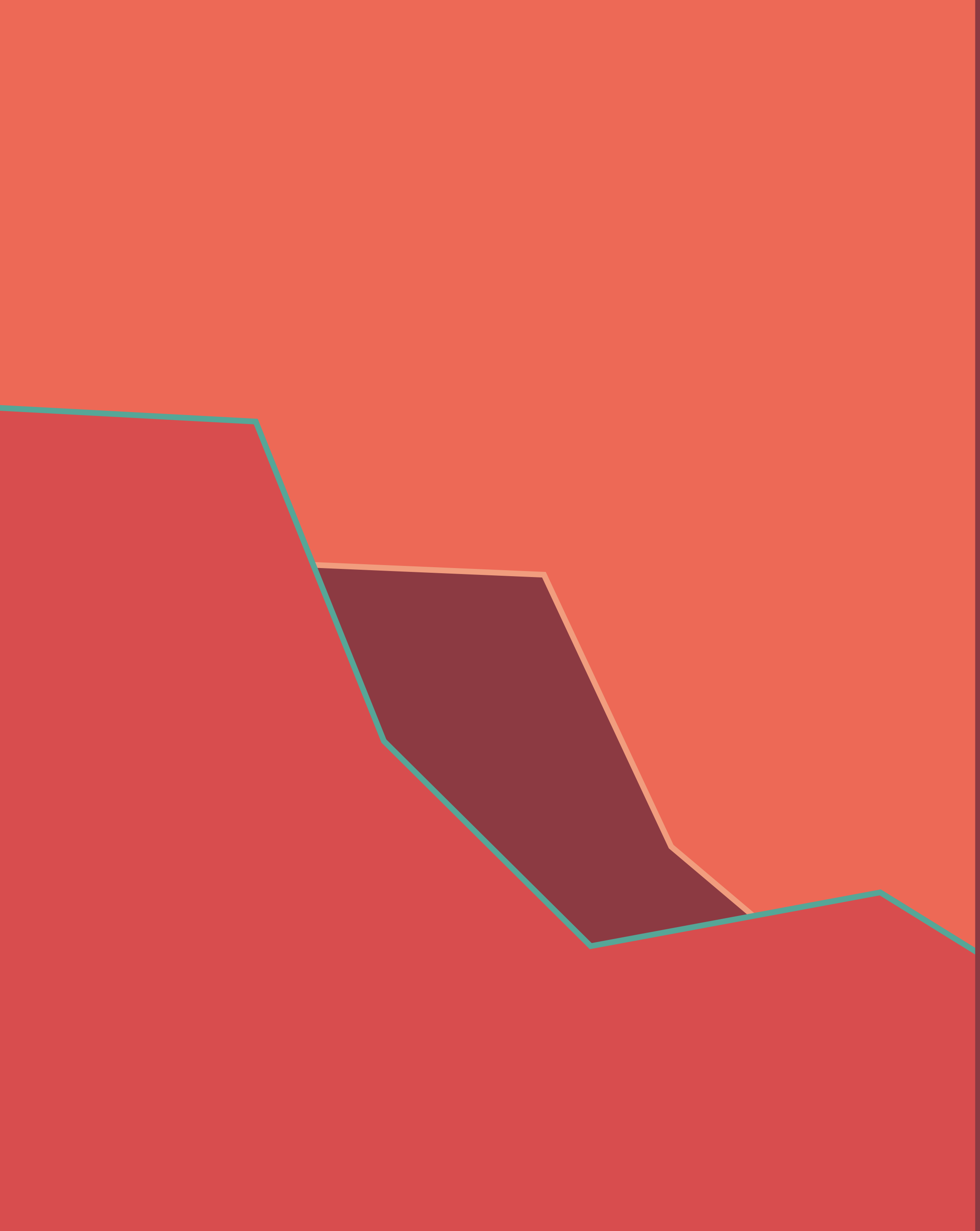


The background features a solid orange-red color. On the left side, there is a large, abstract geometric shape in a darker red. This shape has several facets and is outlined with thin lines in a teal and a light orange color. The overall design is modern and minimalist.

**Rails is**  
designed to  
have agnostic  
interfaces

The background features a solid orange-red color. On the left side, there is a large, abstract geometric shape in a darker red color. This shape has several vertices and edges, creating a jagged, mountain-like profile. A thin, light teal line follows the outer boundary of this dark red shape. The text is positioned on the right side of the image, against the orange-red background.

**Rails is**  
extracted from  
applications

The background features a solid orange-red upper section and a darker red lower section separated by a jagged, hand-drawn line. A dark maroon polygon is positioned in the lower-left area, partially overlapping the two background colors.

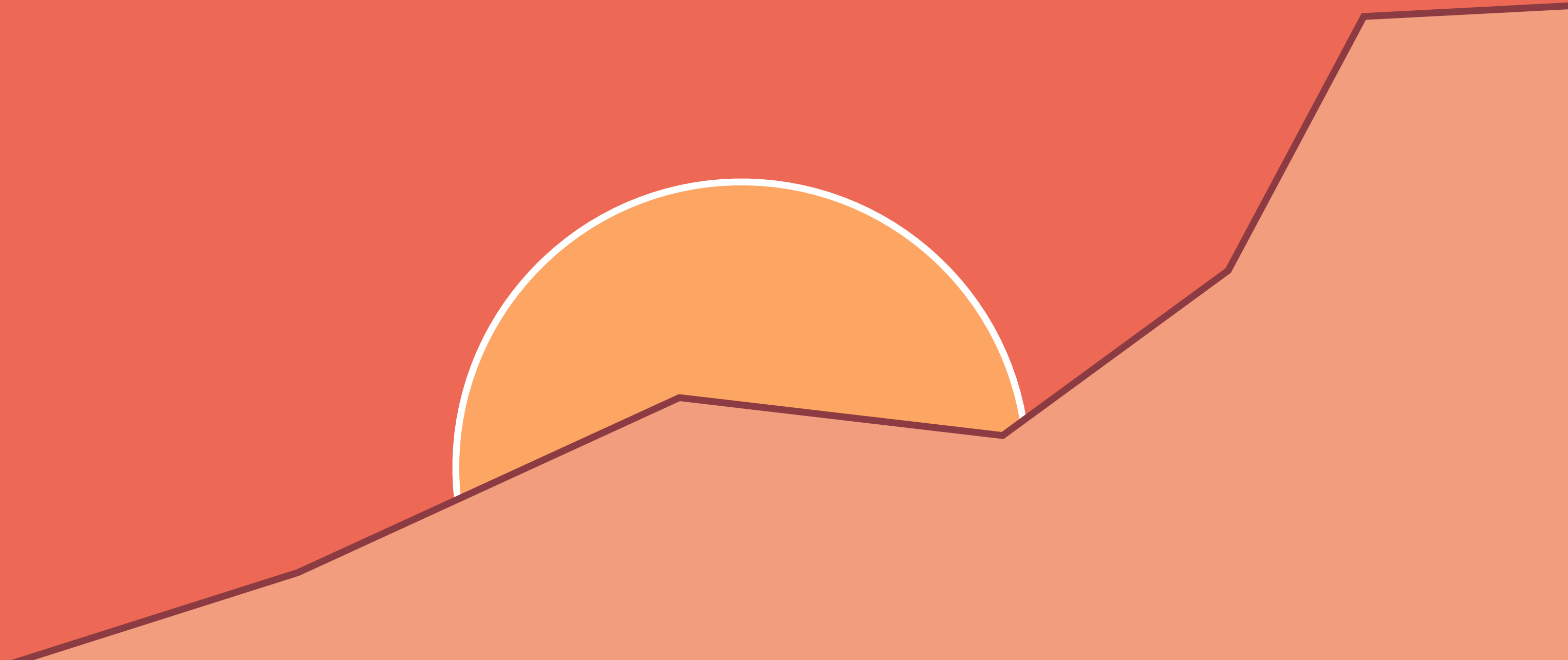
**Rails is**  
made of  
simple and  
aesthetic APIs

The background features a solid orange-red upper section and a darker red lower section separated by a jagged, hand-drawn style line. A dark maroon polygon is positioned in the lower-left area, partially overlapping the two background colors.

**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

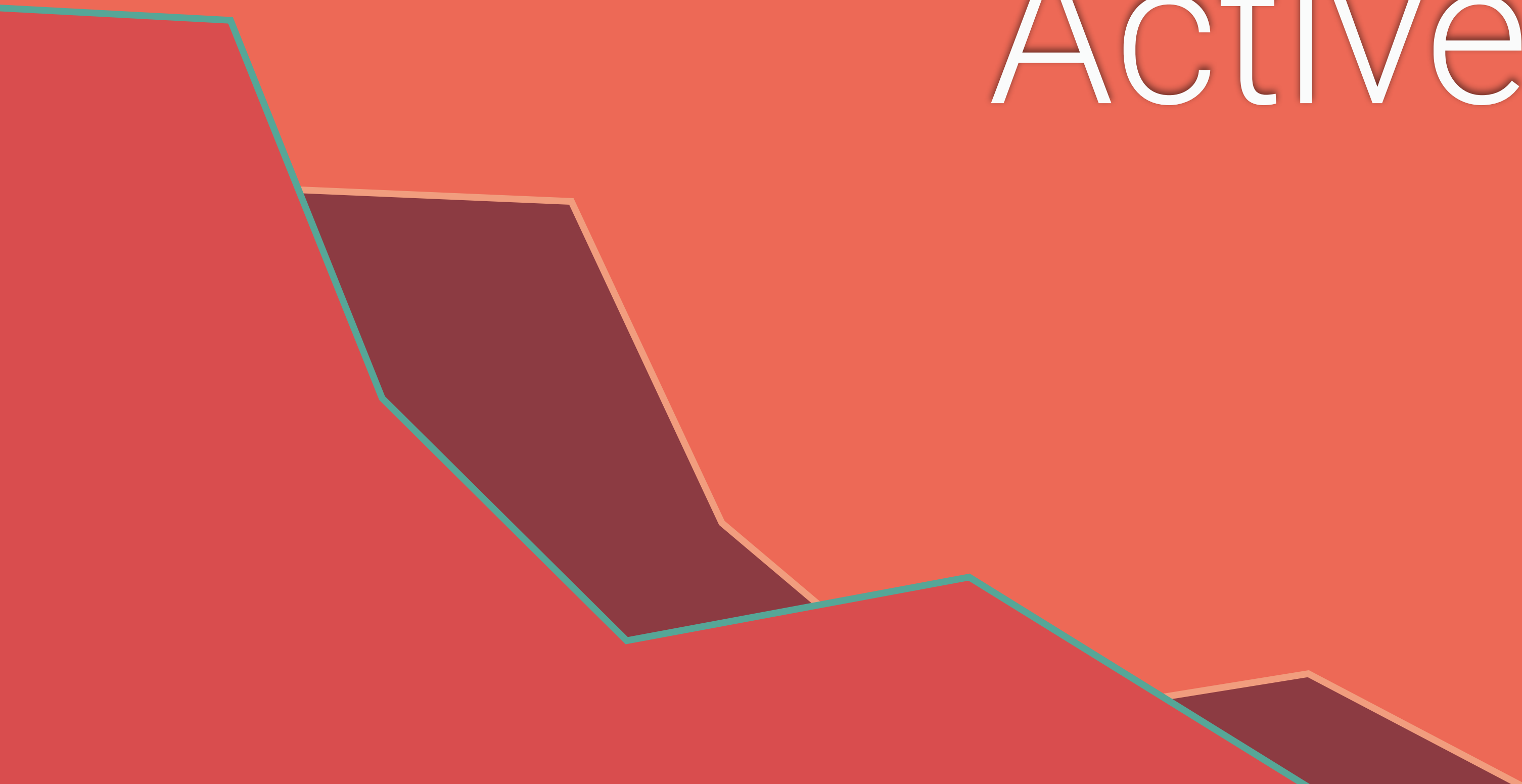
Active Storage

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

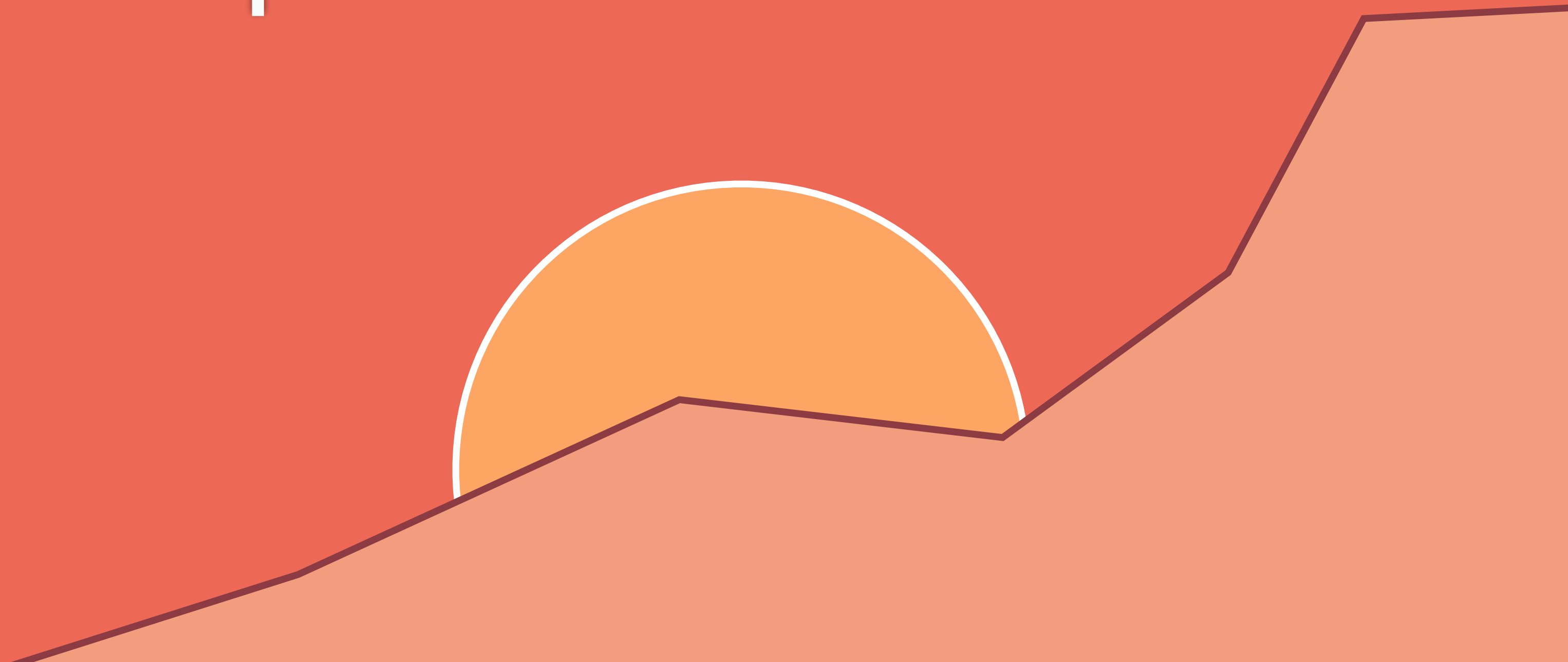
Railties

## 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



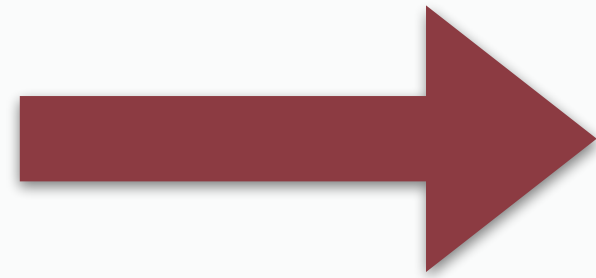


# Application



Application

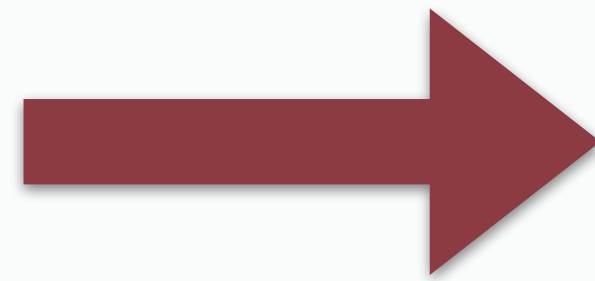
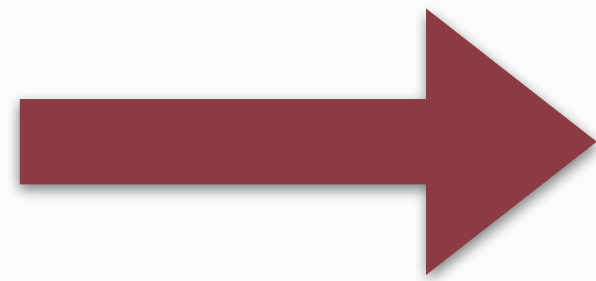
Register hooks



Application

Register hooks

Load components

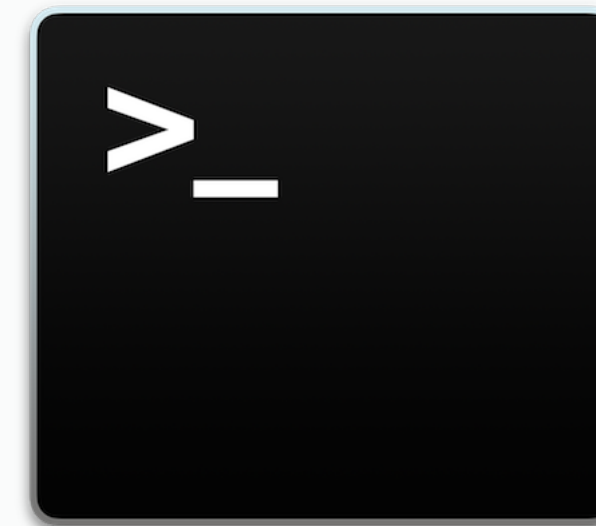
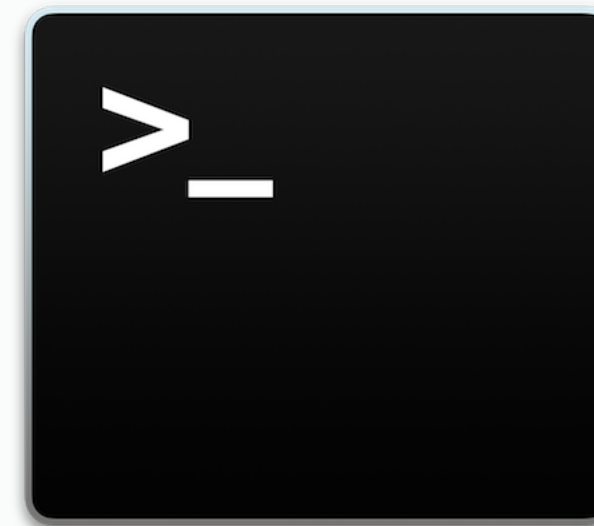
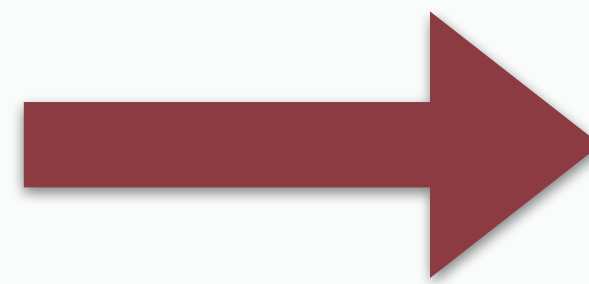


Application

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
```

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

```
initializer "active_job.logger" do  
  ActiveSupport.on_load(:active_job) {  
    self.logger = ::Rails.logger  
  }  
end
```

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

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

```
  app.deprecators[:active_model] =  
    ActiveSupport.deprecator
```

```
end
```

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

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

```
  app.deprecators[:active_model] =  
    ActiveSupport.deprecator
```

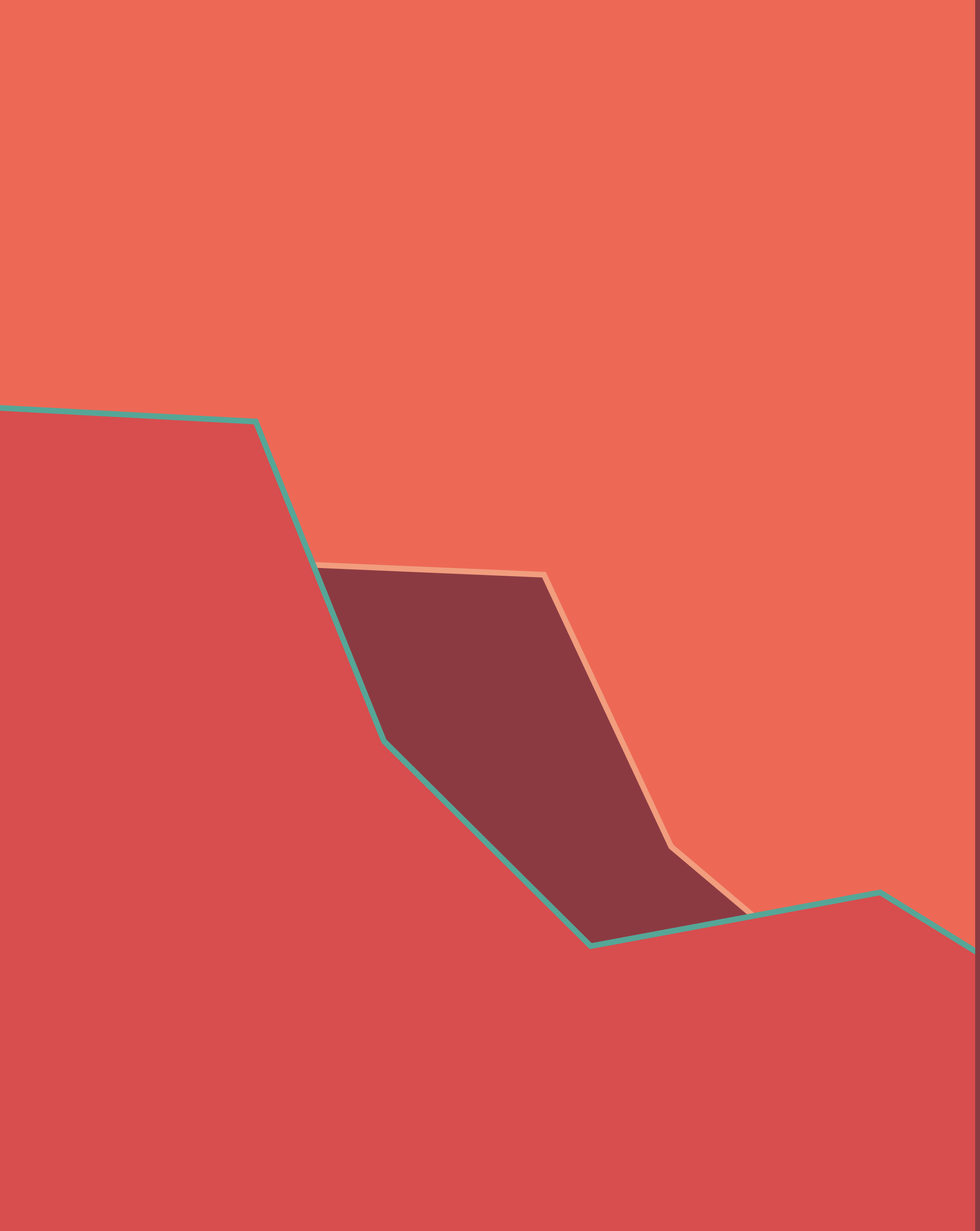
```
end
```



- 
- Railties are the core of the framework

- 
- Raities are the core of the framework
  - Raities 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_?(SQLite3Adapter)
  # ...
else
  # ...
end
```

```
if connection.is_a?(PostgresqlAdapter)
  # ...
elsif connection.is_a?(Mysql2Adapter)
  # ...
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 ActiveSupport
  class Service
    def delete(key)
      raise NotImplementedError
    end
  end
end
```

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

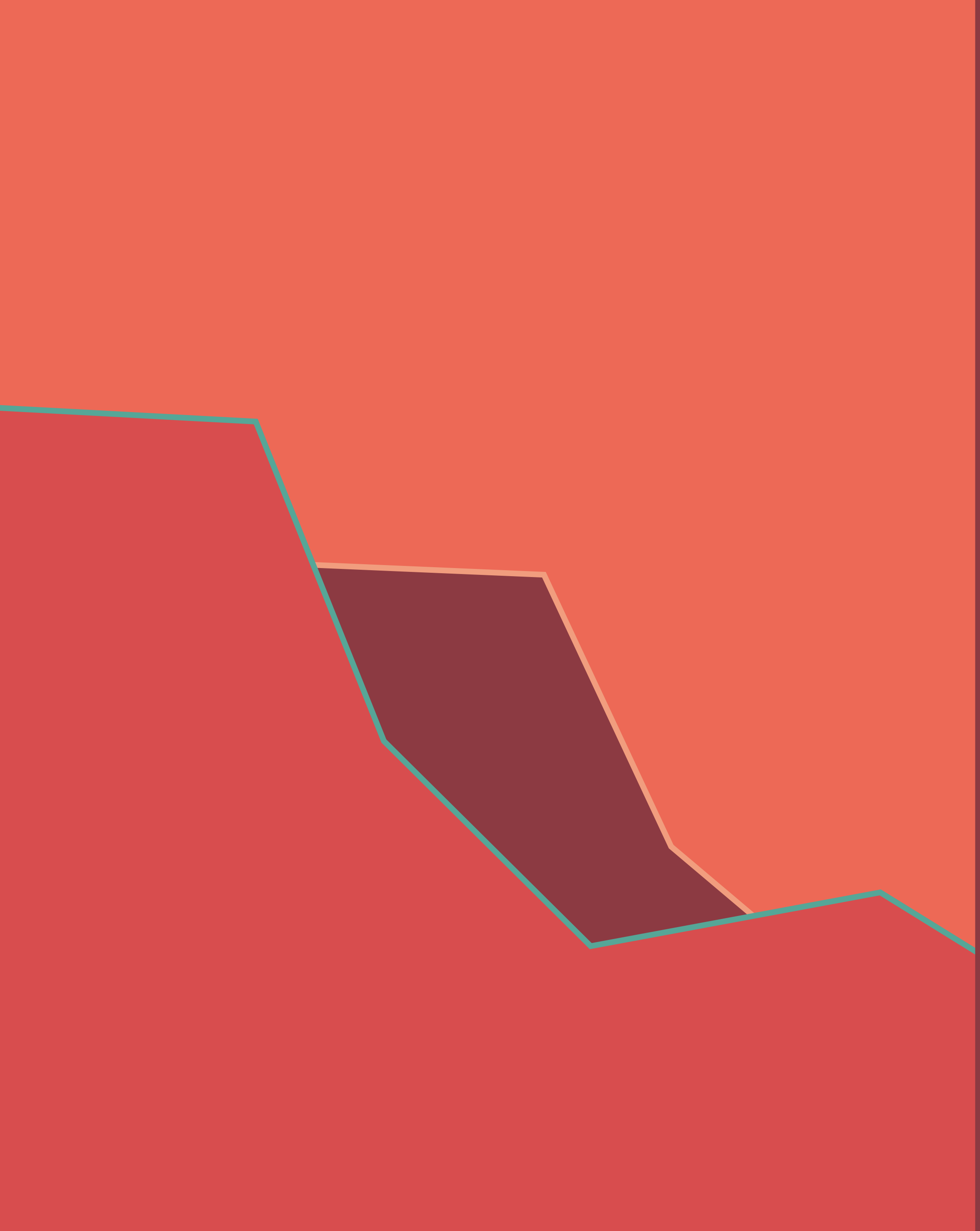
```
class ActiveStorage
  class Service::GCSService < Service
    def delete(key)
      instrument :delete, key: key do
        file_for(key).delete
      rescue Google::Cloud::NotFoundError
        # Ignore files already deleted
      end
    end
  end
end
end
```

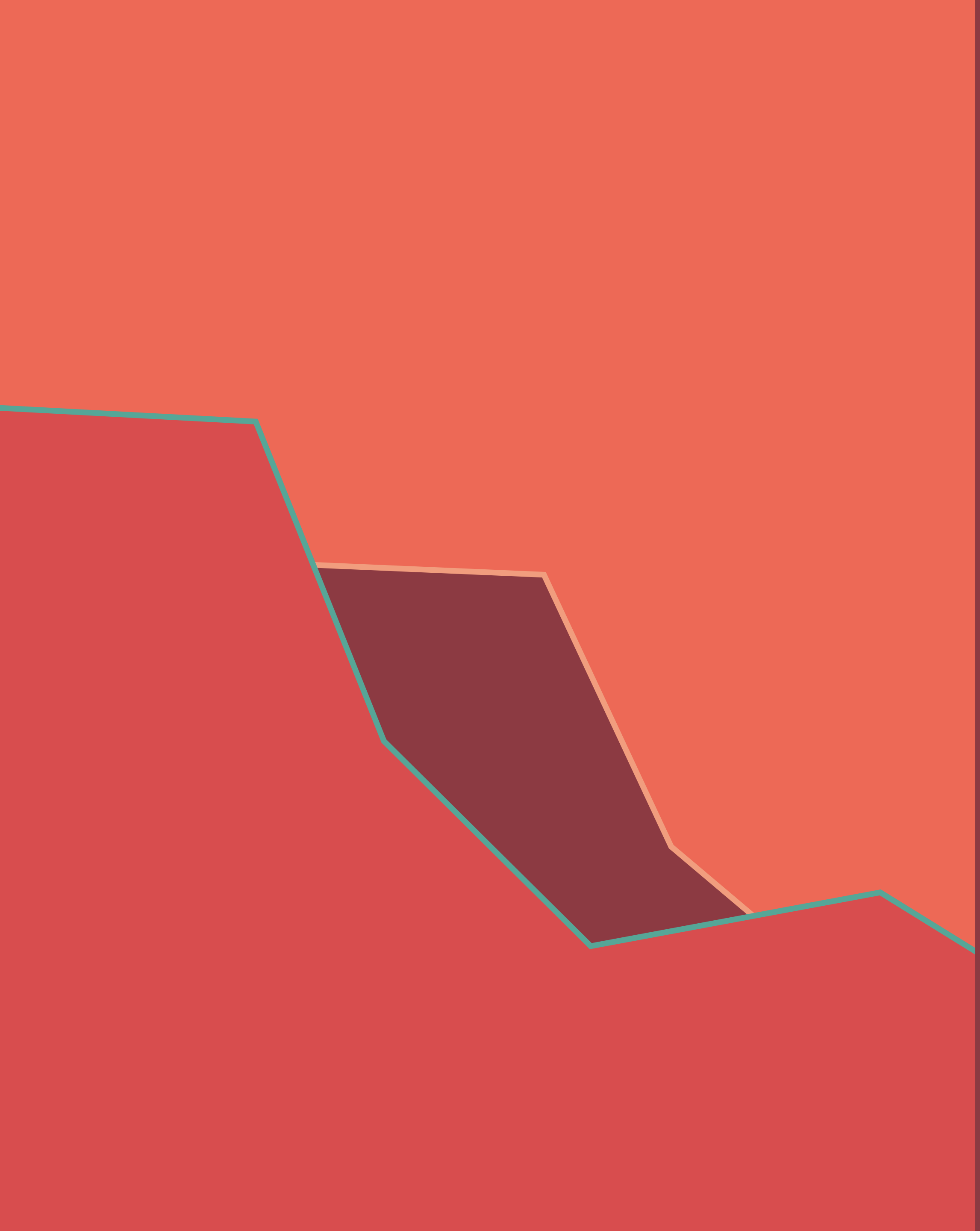
```
@service.delete(key)
```

- 
- 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



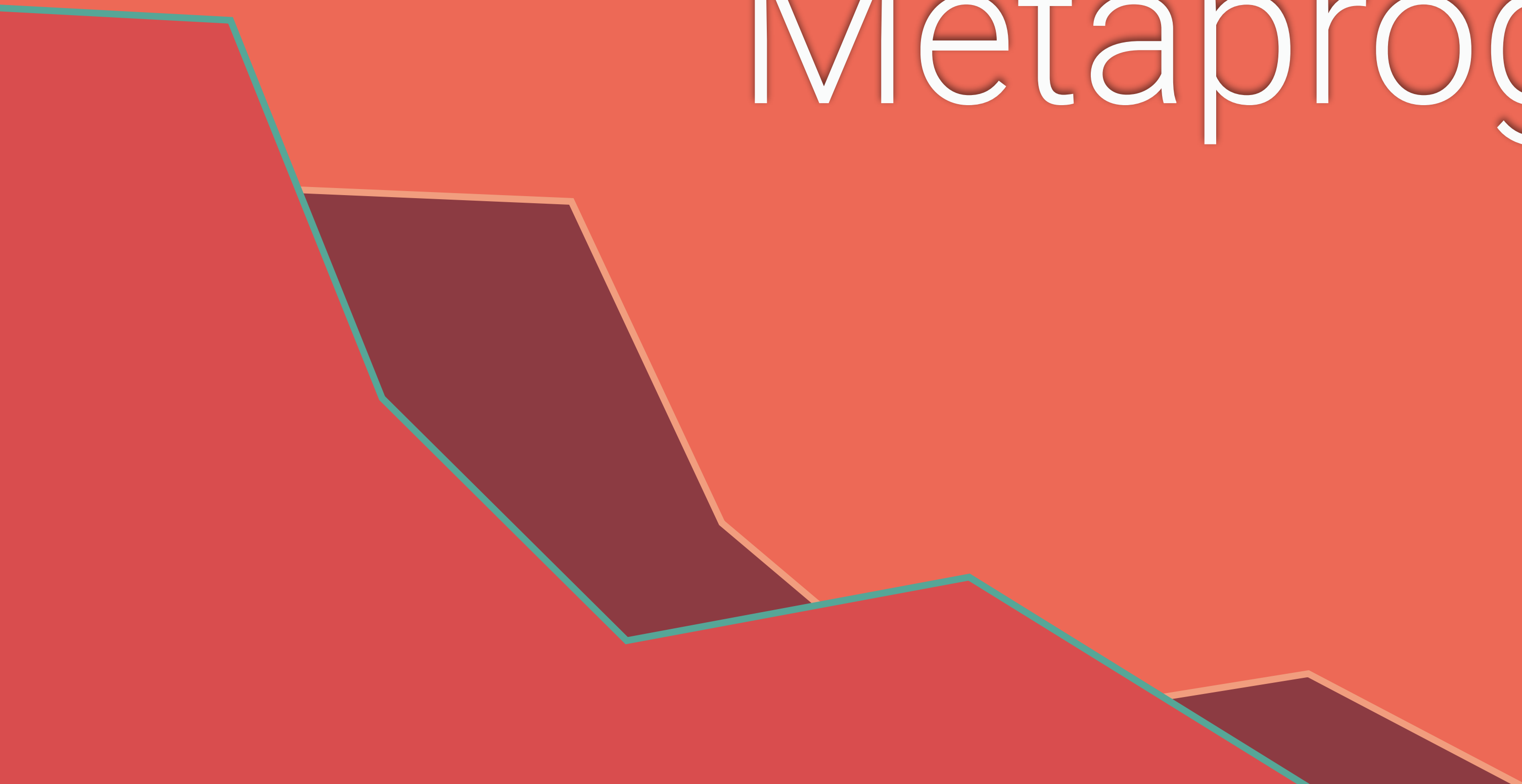
# Migrating Shopify's Core Rails Monolith to Trilogy

Adrianna Chang

Monday, April 24 @ 3pm

# 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  
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  
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  
end
```



Post::GeneratedAssociationMethods

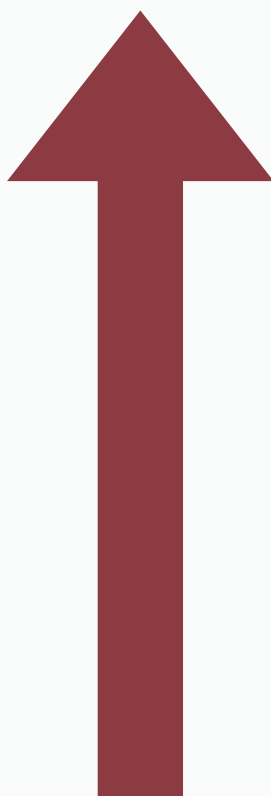
```
# activerecord/lib/active_record/associations/builder/  
association.rb
```

```
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  
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  
end
```

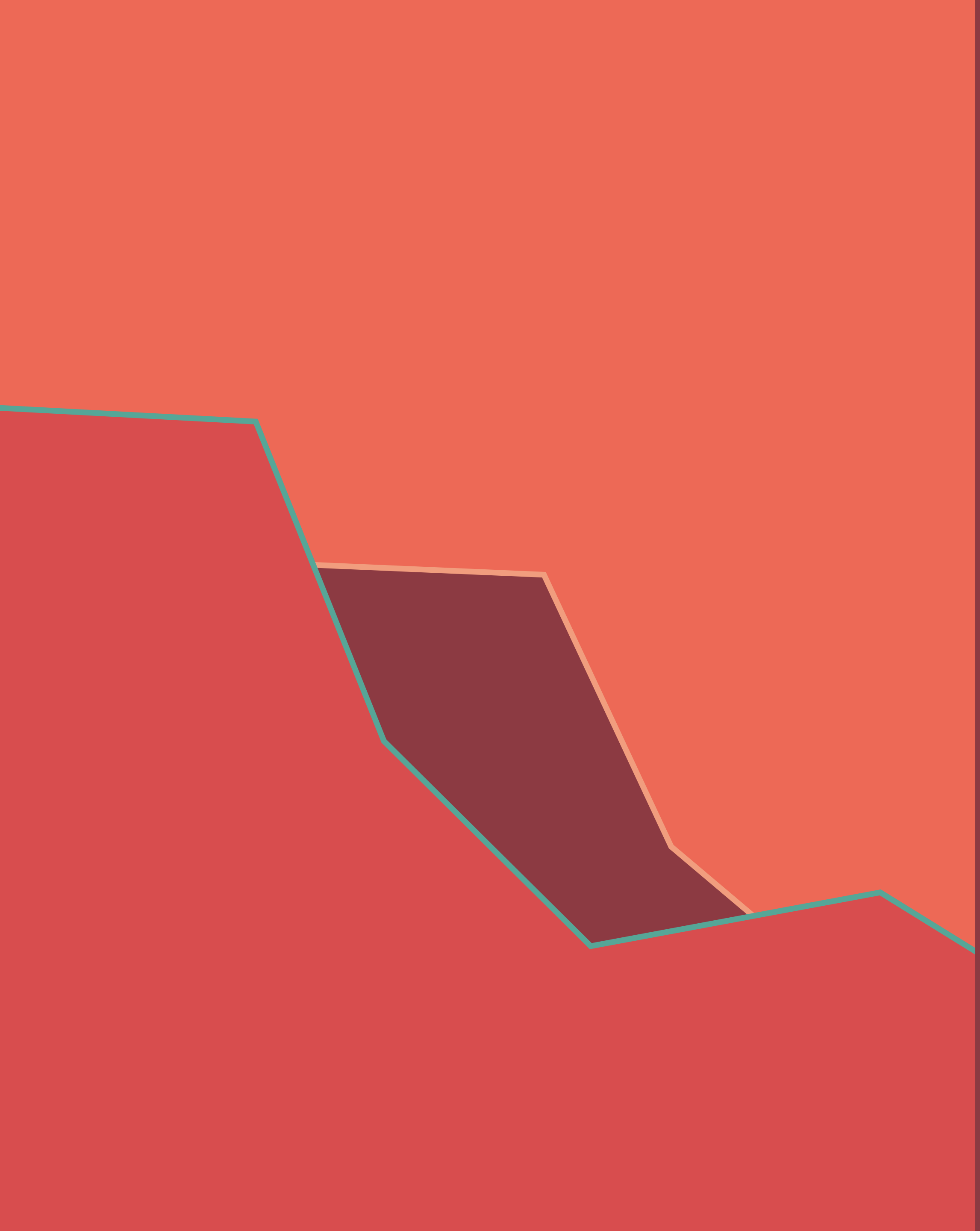


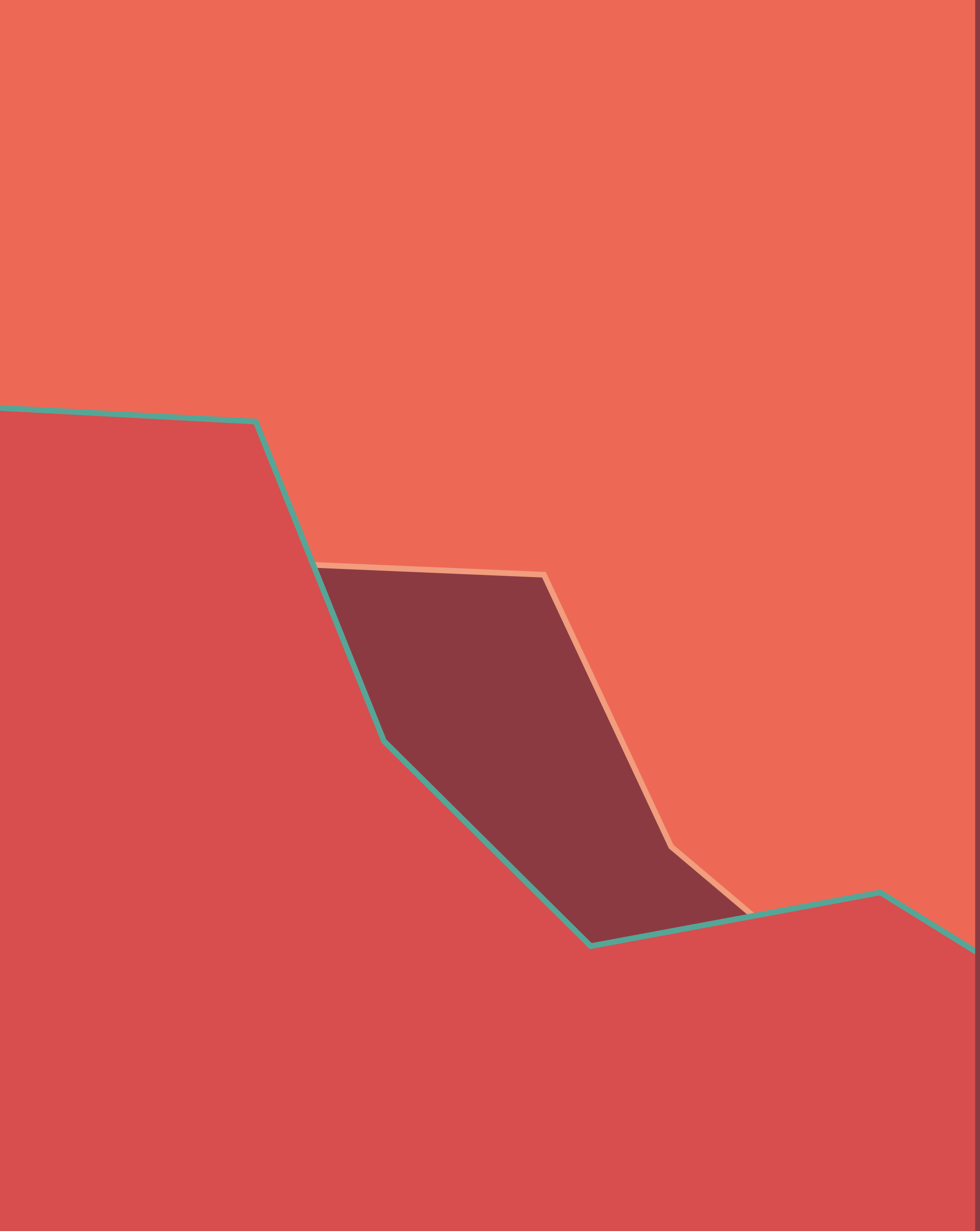
```
# activerecord/lib/active_record/associations/builder/  
association.rb
```

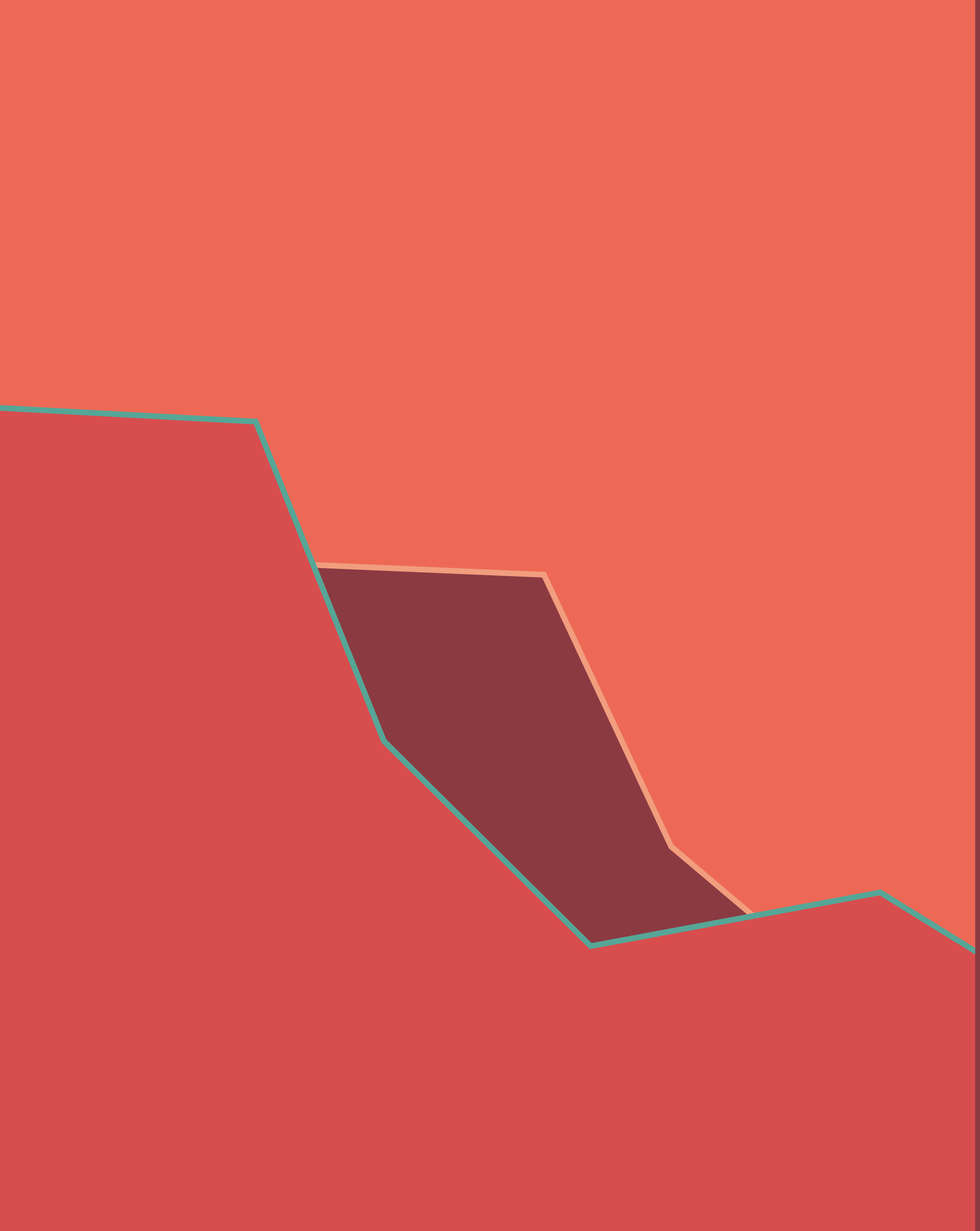
```
class ActiveRecord::Associations::Builder  
  class Association  
    def self.define_writers(mixin, name)  
      mixin.class_eval <<-CODE, __FILE__, __LINE__ + 1  
        def #{name}=(value)  
          association(:#{name}).writer(value)  
        end  
      CODE  
    end  
  end  
end  
end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





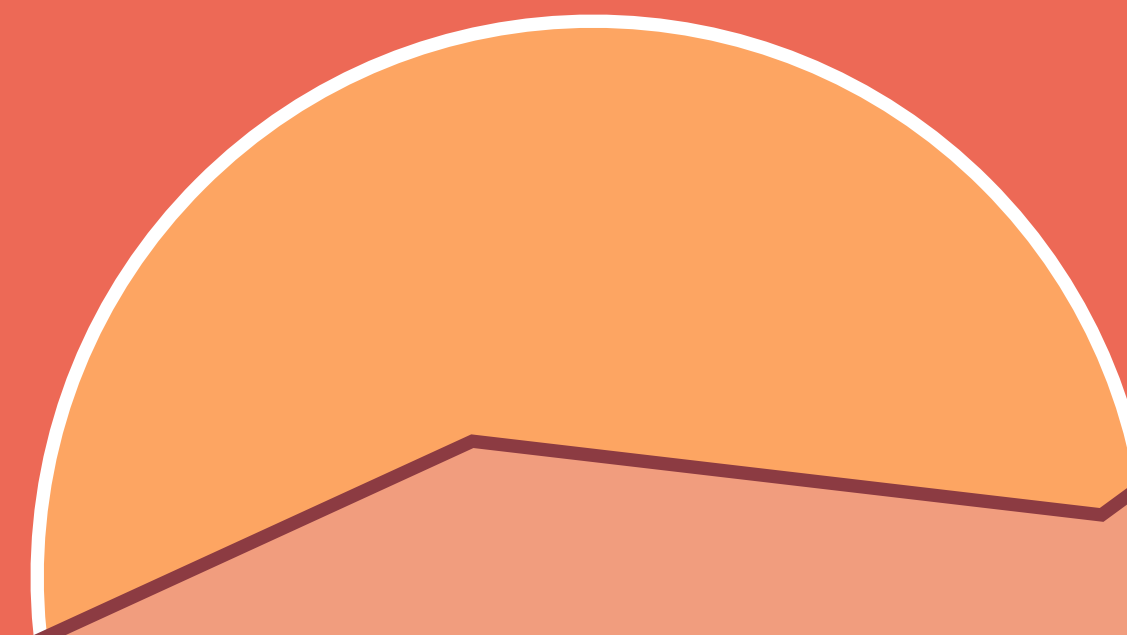
- 
- Powerful tool that enables us to build beautiful, simple APIs

- 
- The left side of the slide features a decorative design with overlapping geometric shapes. A large, solid red shape occupies the bottom-left corner. Above it, a teal-colored shape with a complex, multi-sided polygonal outline is visible. Within this teal shape, there is a smaller, dark maroon-colored polygon. The background of the entire slide is a light, muted teal color.
- Powerful tool that enables us to build beautiful, simple APIs
  - Hides complexity from your application

- 
- The left side of the slide features a decorative design with overlapping geometric shapes. A large, solid red shape occupies the bottom-left corner. Above it, a smaller, dark red shape is partially visible. To the right of these, a light red shape with a thin teal border is positioned. The background of the entire slide is a solid light red color.
- 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

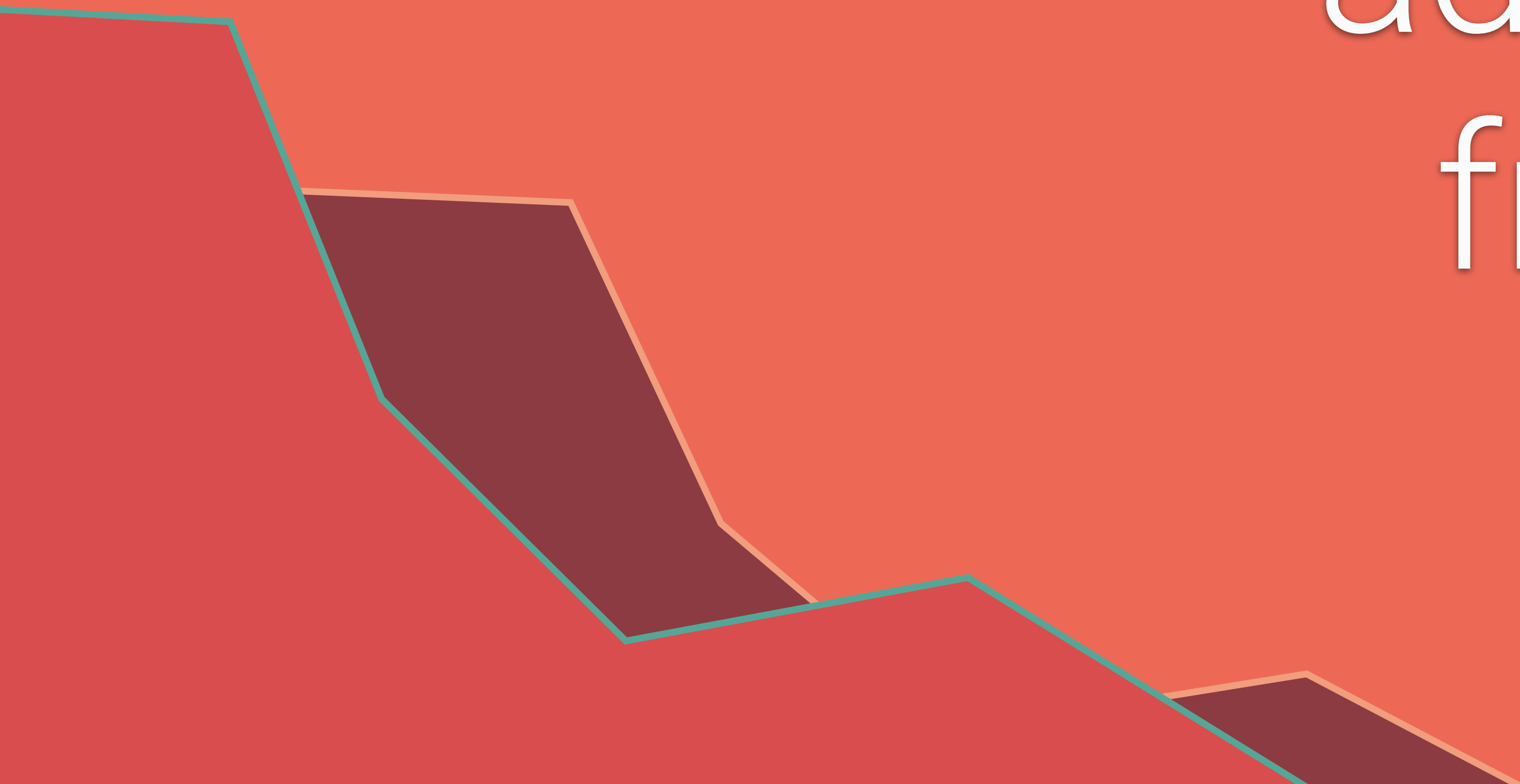
**2015**

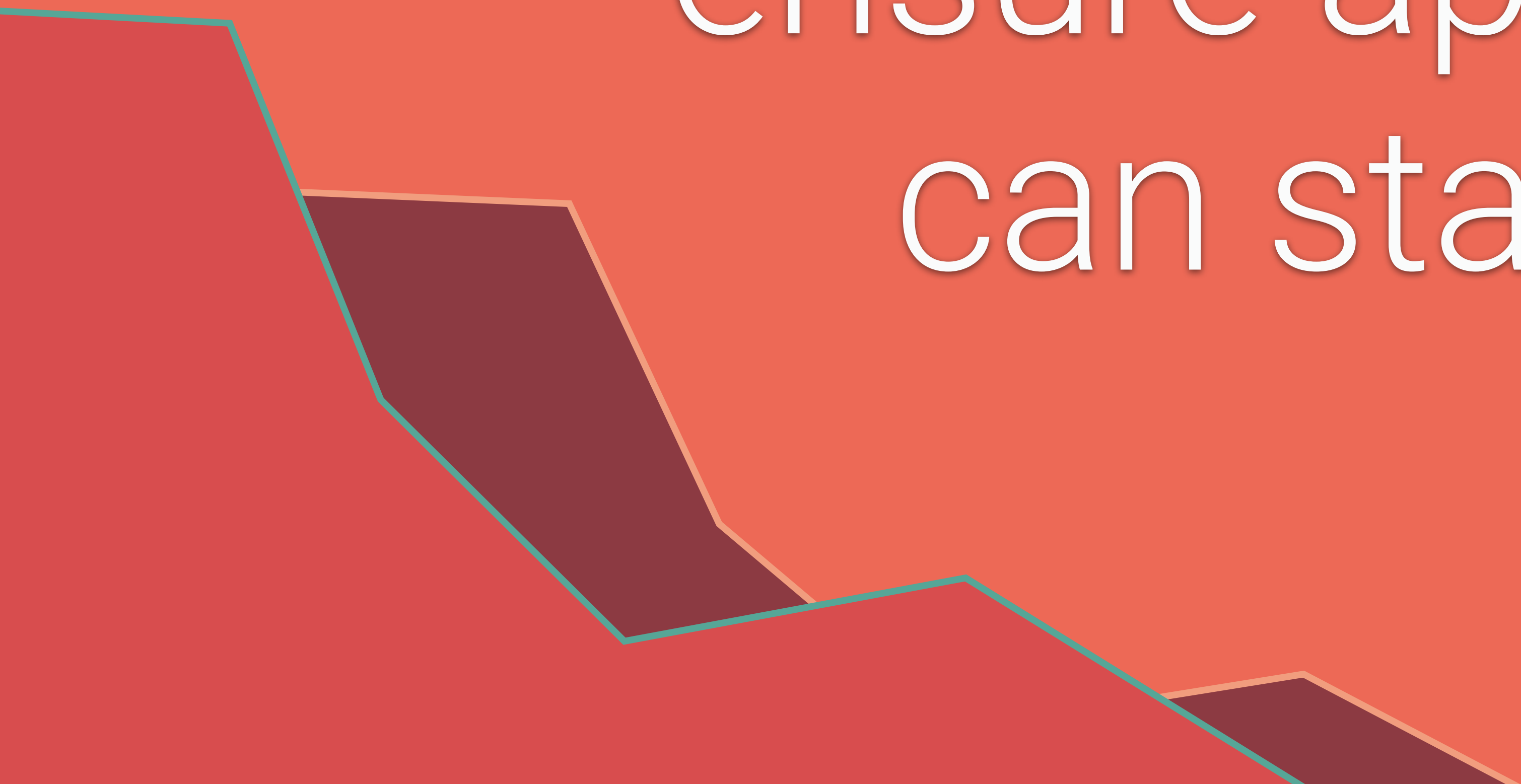
First RailsConf

**2023**

This RailsConf

**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



Abstract geometric shapes in the bottom left corner, consisting of a large red polygon and several smaller, overlapping polygons in shades of red and orange.

**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

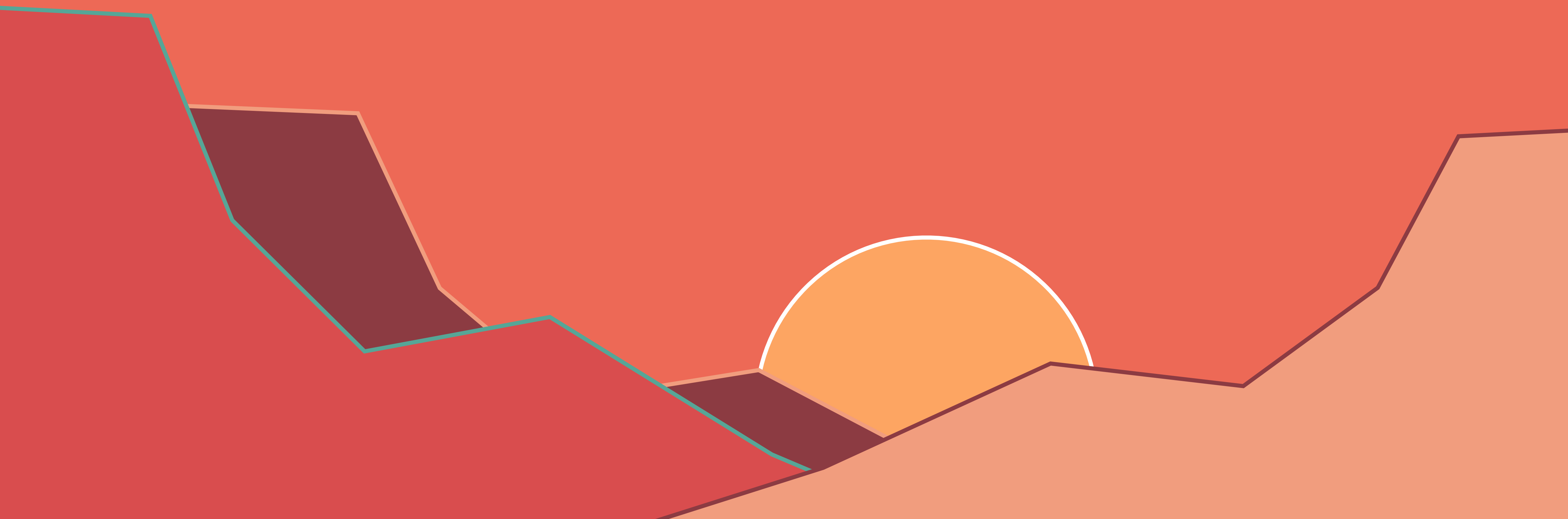
**Rails is**

the applications we build

**Rails is**  
the team behind it

**Rails is**  
the community

**Rails** *is* magic



# Thank You!

