```
$ rails test
$ git add -A
$ git commit -m "Add user microposts"
```

Because so many things can go wrong with the configuration, we'll deploy the app directly from our current topic branch, making sure it's working before merging into `master`. We can do this by including the branch name in the push to Heroku as follows:

```
$ git push heroku user-microposts:master
```

As usual, we then reset the database and reseed the sample data:

```
$ heroku pg:reset DATABASE
$ heroku run rails db:migrate
$ heroku run rails db:seed
```

Because Heroku comes with an installation of ImageMagick, the result is successful image resizing and upload in production, as seen in Figure 13.40.

Exercises

Solutions to the exercises are available to all Rails Tutorial purchasers here.

To see other people's answers and to record your own, subscribe to the Rails Tutorial course or to the Learn Enough All Access Bundle.

1. Upload a large image and confirm directly that the resizing is working in production. Does the resizing work even if the image isn't square?

13.5 Conclusion

With the addition of the Microposts resource, we are nearly finished with our sample application. All that remains is to add a social layer by letting users

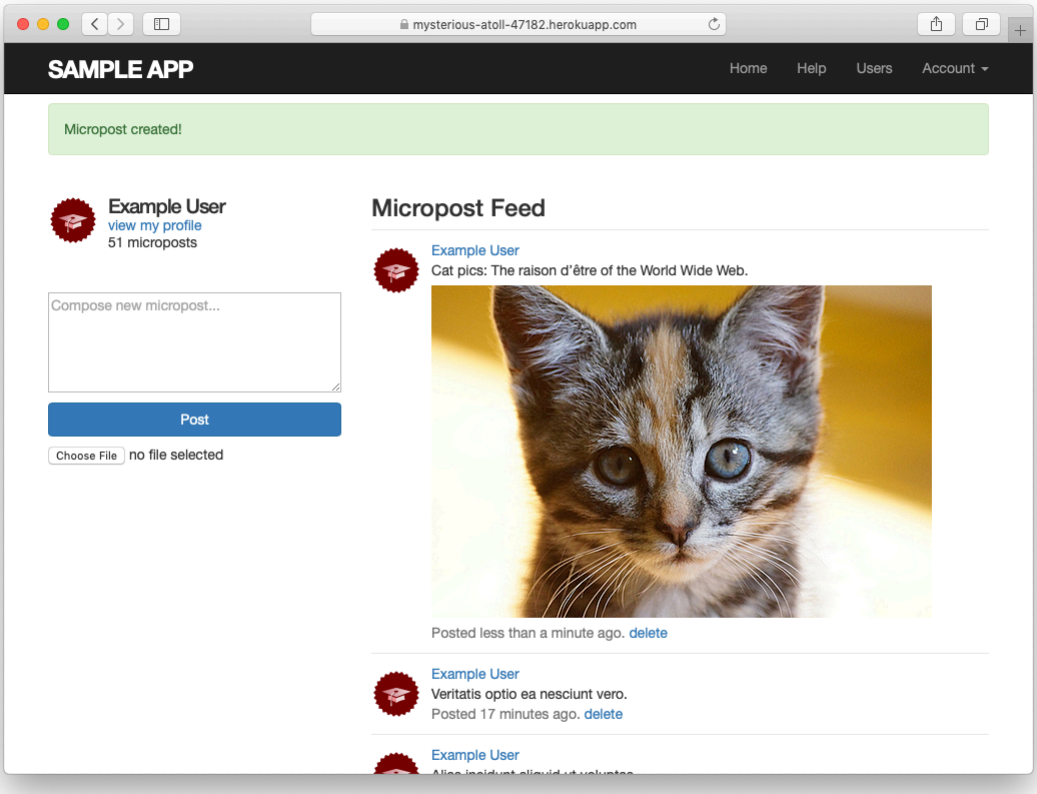

Figure 13.40: An uploaded image in production.

follow each other. We'll learn how to model such user relationships, and see the implications for the microposts feed, in Chapter 14.

If you skipped Section 13.4.4, be sure to commit your changes:

```
$ rails test
$ git add -A
$ git commit -m "Add user microposts"
```

Then merge into **master**:

```
$ git checkout master
$ git merge user-microposts
$ git push
```

And finally deploy to production:

```
$ git push heroku
$ heroku pg:reset DATABASE
$ heroku run rails db:migrate
$ heroku run rails db:seed
```

It's worth noting that this chapter saw the last of the necessary gem installations. For reference, the final **Gemfile** is shown in Listing 13.75.[25]

Listing 13.75: The final **Gemfile** for the sample application.

```
source 'https://rubygems.org'
git_source(:github) { |repo| "https://github.com/#{repo}.git" }

gem 'rails',                      '6.0.1'
gem 'image_processing',           '1.9.3'
gem 'mini_magick',                '4.9.5'
gem 'active_storage_validations', '0.8.2'
gem 'bcrypt',                     '3.1.13'
gem 'faker',                      '2.1.2'
gem 'will_paginate',              '3.1.8'
```

[25] As always, you should use the version numbers listed at gemfiles-6th-ed.railstutorial.org instead of the ones listed here.

```
gem 'bootstrap-will_paginate',    '1.0.0'
gem 'bootstrap-sass',             '3.4.1'
gem 'puma',                       '3.12.1'
gem 'sass-rails',                 '5.1.0'
gem 'webpacker',                  '4.0.7'
gem 'turbolinks',                 '5.2.0'
gem 'jbuilder',                   '2.9.1'
gem 'bootsnap',                   '1.4.4', require: false

group :development, :test do
  gem 'sqlite3', '1.4.1'
  gem 'byebug',  '11.0.1', platforms: [:mri, :mingw, :x64_mingw]
end

group :development do
  gem 'web-console',           '4.0.1'
  gem 'listen',                '3.1.5'
  gem 'spring',                '2.1.0'
  gem 'spring-watcher-listen', '2.0.1'
end

group :test do
  gem 'capybara',                  '3.28.0'
  gem 'selenium-webdriver',        '3.142.4'
  gem 'webdrivers',                '4.1.2'
  gem 'rails-controller-testing', '1.0.4'
  gem 'minitest',                  '5.11.3'
  gem 'minitest-reporters',        '1.3.8'
  gem 'guard',                     '2.15.0'
  gem 'guard-minitest',            '2.4.6'
end

group :production do
  gem 'pg',          '1.1.4'
  gem 'aws-sdk-s3', '1.46.0', require: false
end

# Windows does not include zoneinfo files, so bundle the tzinfo-data gem
gem 'tzinfo-data', platforms: [:mingw, :mswin, :x64_mingw, :jruby]
```

13.5.1 What we learned in this chapter

- Microposts, like Users, are modeled as a resource backed by an Active Record model.
- Rails supports multiple-key indices.

- We can model a user having many microposts using the **`has_many`** and **`belongs_to`** methods in the User and Micropost models, respectively.
- The **`has_many`**/**`belongs_to`** combination gives rise to methods that work through the association.
- The code **`user.microposts.build(...)`** returns a new Micropost object automatically associated with the given user.
- Rails supports default ordering via **`default_scope`**.
- Scopes take anonymous functions as arguments.
- The **`dependent: :destroy`** option causes objects to be destroyed at the same time as associated objects.
- Pagination and object counts can both be performed through associations, leading to automatically efficient code.
- Fixtures support the creation of associations.
- It is possible to pass variables to Rails partials.
- The **`where`** method can be used to perform Active Record selections.
- We can enforce secure operations by always creating and destroying dependent objects through their association.
- We can upload images using Active Storage.