Listing 7.35: The configuration file for the production webserver.
config/puma.rb

```
# Puma configuration file.
max_threads_count = ENV.fetch("RAILS_MAX_THREADS") { 5 }
min_threads_count = ENV.fetch("RAILS_MIN_THREADS") { max_threads_count }
threads min_threads_count, max_threads_count
port        ENV.fetch("PORT") { 3000 }
environment ENV.fetch("RAILS_ENV") { ENV['RACK_ENV'] || "development" }
pidfile ENV.fetch("PIDFILE") { "tmp/pids/server.pid" }
workers ENV.fetch("WEB_CONCURRENCY") { 2 }
preload_app!
plugin :tmp_restart
```

We also need to make a so-called **`Procfile`** to tell Heroku to run a Puma process in production, as shown in Listing 7.36. The **`Procfile`** should be created in your application's root directory (i.e., in the same directory as the **`Gemfile`**).

Listing 7.36: Defining a **`Procfile`** for Puma.
./Procfile

```
web: bundle exec puma -C config/puma.rb
```

7.5.3 Production database configuration

The final step in our production deployment is properly configuring the production database, which (as mentioned briefly in Section 2.3.5) is PostgreSQL. My testing indicates that PostgreSQL actually works on Heroku without any configuration, but the official Heroku documentation recommends explicit configuration nonetheless, so we'll err on the side of caution and include it.

The actual change is easy: all we have to do is update the **`production`** section of the database configuration file, **`config/database.yml`**. The result, which I adapted from the Heroku docs, is shown in Listing 7.37.

Listing 7.37: Configuring the database for production.
config/database.yml

```
# SQLite version 3.x
#   gem install sqlite3
#
#   Ensure the SQLite 3 gem is defined in your Gemfile
#   gem 'sqlite3'
#
default: &default
  adapter: sqlite3
  pool: 5
  timeout: 5000

development:
  <<: *default
  database: db/development.sqlite3

# Warning: The database defined as "test" will be erased and
# re-generated from your development database when you run "rake".
# Do not set this db to the same as development or production.
test:
  <<: *default
  database: db/test.sqlite3

production:
  adapter: postgresql
  encoding: unicode
  # For details on connection pooling, see Rails configuration guide
  # https://guides.rubyonrails.org/configuring.html#database-pooling
  pool: <%= ENV.fetch("RAILS_MAX_THREADS") { 5 } %>
  database: sample_app_production
  username: sample_app
  password: <%= ENV['SAMPLE_APP_DATABASE_PASSWORD'] %>
```

7.5.4 Production deployment

With the production webserver and database configuration completed, we're ready to commit and deploy:[15]

[15] We haven't changed the data model in this chapter, so running the migration at Heroku shouldn't be necessary, but only if you followed the steps in Section 6.4. Because several readers reported having trouble, I've added **`heroku run rails db:migrate`** as a final step just to be safe.