Setup Heroku

deanperry edited this page on 16 Feb 2012 · 29 revisions

Setup Heroku

To setup ror_ecommerce to work on Heroku follow these steps:

Create a Heroku account

Go to Heroku and create an account.

Change compass for production

Gemfile changes

Add the following to the Gemfile:

```
gem 'aws-sdk'

group :production do
  gem  'pg'
end
```

InsertIntoFile

Move the mysql2 gem to development/test or change development database config user to `postgres` :

```
group :test, :development do
  gem 'mysql2', '~> 0.3.10'
end
```

ChangeFileRegex

Remove the `autotest-fsevent` from the Gemfile. Heroku doesn't like is as it's for Mac.

```
if RUBY_PLATFORM =~ /darwin/
  # gem "autotest-fsevent", '~> 0.2.5'
end
```

ChangeFileRegex

Images

In order to use Paperclip on Heroku you need to have an Amazon S3 account. There is excellent documentation on how to setup S3 on Heroku's DevCenter.

Once that is setup you will need to comment out the following in `config/environments/production.rb`

```
PAPERCLIP_STORAGE_OPTS = {  :styles => {:mini => '48x48>',
                                        :small => '100x100>',
                                        :medium => '200x200>',
                                        :product => '320x320>',
                                        :large => '600x600>' },
                            :default_style => :product,
                            :url => "/assets/products/:id/:style/:basename.:extension'
                            :path => ":rails_root/public/assets/products/:id/:style/:b
```

ChangeFileRegex

You will need to create a bucket on Amazon S3. There are several tools to do this. I suggest taking a look at the Firefox plugins for this. Cyberduck is a good free Mac app that supports Amazon S3 as well as Rackspace Cloud Files. You also need to create a `config/s3.yml` config file as instructed by the `aws-s3` gem.

Edit your `app/models/image.rb` from `has_attached_file :photo, PAPERCLIP_STORAGE_OPTS` to:

```
has_attached_file :photo, {
  :styles => {  :mini => '48x48>',
                :small => '100x100>',
                :product => '320x320>',
                :large => '600x600>' },
  :default_style => :product,
  :storage => :s3,
  :s3_credentials => {:access_key_id     => ENV['AWS_ACCESS_KEY_ID'],
                      :secret_access_key => ENV['AWS_SECRET_ACCESS_KEY']},
  :path => ":attachment/:style/:id-:basename.:extension",
  :bucket => 'youbucketname'
} ##  this constant is in /config/environments/*.rb
```

ChangeFileLines

Add the `asset_sync` gem to your Gemfile and generate the config file:

```
rails g asset_sync:install --provider=AWS
```

RootFile with cd referenced from hidden code with variable for directory

This will generate an `asset_sync` config file for Amazon S3/Cloudfront at `config/initializers/asset_sync.rb`

It should look like the following:

```
AssetSync.configure do |config|
  config.fog_provider = 'AWS'
  config.fog_directory = ENV['FOG_DIRECTORY']
  config.aws_access_key_id = ENV['AWS_ACCESS_KEY_ID']
  config.aws_secret_access_key = ENV['AWS_SECRET_ACCESS_KEY']

  # Don't delete files from the store
  # config.existing_remote_files = "keep"
  #
  # Increase upload performance by configuring your region
  # config.fog_region = 'eu-west-1'
  #
  # Automatically replace files with their equivalent gzip compressed version
  # config.gzip_compression = true
  #
  # Use the Rails generated 'manifest.yml' file to produce the list of files to
  # upload instead of searching the assets directory.
  # config.manifest = true
  #
  # Fail silently.  Useful for environments such as Heroku
  # config.fail_silently = true
end
```

ChangeFileLines

Create a new Heroku instance. The stack command is important as the default stack doesn't support Rails 3 properly

```
heroku create --stack cedar
```

RootFile

This will create the Heroku instance and add it's git URL to your git repo. Commit the changes and push it up to your Heroku instance.

```
git add . && git commit . -m 'heroku commit'
git push
git push heroku master
```

RootFile

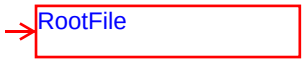
Setup AWS S3

For this step, you will need your Amazon AWS API credentials. You can find them on this page.

Once you have these credentials you can now add then to Heroku:

You need to add your `AWS_ACCESS_KEY_ID` , `AWS_SECRET_ACCESS_KEY` & `FOG_DIRECTORY` details like this: Replace the following examples with your API key and Amazon S3 bucket (FOG_DIRECTORY):

```
heroku config:add AWS_ACCESS_KEY_ID=xxxxxxxxxxx
heroku config:add AWS_SECRET_ACCESS_KEY=xxxxxxxxxxx
heroku config:add FOG_DIRECTORY=youbucketname
```



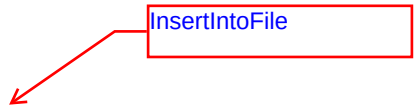
To view what variables are already set on your Heroku instance, run `heroku config` .

Once these details have been set, you can then run the Rails asset compilation and push the assets up to your Amazon AWS bucket by using this command:

```
heroku run rake assets:precompile --trace
```

Add the following to `config/aws.rb`

```
require 'aws'
# log requests using the default rails logger
AWS.config(:logger => Rails.logger)
# load credentials from a file
if File.exists?(File.dirname(__FILE__)+"/../aws.yml")
   config_path = File.expand_path(File.dirname(__FILE__)+"/../aws.yml")
else
   config_path = nil
end
if config_path
  AWS.config(YAML.load(File.read(config_path)))
else
   AWS::S3::Base.establish_connection!(
     :access_key_id     => ENV['AWS_ACCESS_KEY_ID'],
     :secret_access_key => ENV['AWS_SECRET_ACCESS_KEY']
   )
end
```



Lastly be sure to edit `order_item.rb` `order_items_in_cart` because Postgres doesn't like MySQL syntax. This is because Heroku uses Postgres and not MySQL.

```
def self.order_items_in_cart(order_id)
  find(:all, :joins => {:variant => :product },
          :conditions => { :order_items => { :order_id => order_id}},
          :select => "order_items.id, order_items.order_id, order_items.shipping_ra
                      products.shipping_category_id,
                                  count(*) as quantity,
                                  products.shipping_category_id as shipping_cate
                                  SUM(order_items.price) as sum_price,
                                  SUM(order_items.total) as sum_total",
          :group => "order_items.id,
                     products.shipping_category_id,
                     order_items.order_id,
                     order_items.shipping_rate_id,
                     order_items.state,
                     order_items.tax_rate_id,
                     order_items.price,
                     order_items.total, order_items.variant_id")
end
```

Look at Heroku's documentation

I find Heroku's documentation to be excellent. The navigation is average though. Luckily google is great and I search heroku's docs by doing google searches.

You can push your development database to your Heroku instance by running the command `heroku db:push`. Before you can run this command you need to have the `taps` gem added to your Gemfile. You can read more about this command here.

If all else fails

If you are experiencing problems or need help, create an issue and we would be happy to help :)

Even if it is an Heroku issue I'd love to document the information in this readme for other people to see.