1. Scale images

Tool: <https://developers.google.com/speed/pagespeed/insights>

<https://github.com/jtescher/carrierwave-imageoptimizer>

2. Compress content

**Tool**: Online Compression Test (gzip) <http://www.whatsmyip.org/http-compression-test/>

Since requests to Cedar apps are made directly to the application server – not proxied through an HTTP server like nginx – any compression of responses must be done within your application.

http://stackoverflow.com/a/7792966/238880

Cedar doesn't use Nginx, so you have to gzip assets yourself with Rack::Deflater, like so :

# config.ru

require ::File.expand\_path('../config/environment', \_\_FILE\_\_)

**use Rack::Deflater**

run YourApp::Application

Also set static cache control in production.rb

# Cache static assets for atleast 1 year

**config.static\_cache\_control = "public, max-age=31536000"**

3. HTTP Caching

Use Rack::Cache and memcache

<https://devcenter.heroku.com/articles/rack-cache-memcached-rails31>

**a)** Add ‘dalli’ gem to Gemfile and bundle install

**b)** In production.rb

config.cache\_store = :dalli\_store

**c)** Once you’ve configured your application to use memcache we can not configure Rack::Cache. In production.rb, add the following:

config.action\_dispatch.rack\_cache = {

:metastore => Dalli::Client.new,

:entitystore => 'file:tmp/cache/entity',

:allow\_reload => false

}

If not specified, Dalli::Client.new automatically retrieves the Memcache server location from the MEMCACHE\_SERVERS environment variable. Otherwise it will default to localhost and default port.

**d)** Serve Static Assets:

To allow your application to properly serve, invalidate and refresh static assets several config settings must be updated in config/environments/production.rb. Allow Rails to serve assets with the serve\_static\_assets setting.

**config.serve\_static\_assets = true**

Additionally, specify how long an item should stay cached by setting the Cache-Control headers. Without a Cache-Control header static files will not be stored by Rack::Cache

**config.static\_cache\_control = "public, max-age=31536000"**

These settings tell Rack::Cache to store static elements for a very long time.

The Cache-Control header and, in general HTTP Caching, can be applied to dynamic content as well. To properly invalidate modified files Rails updates a hash digest in the file name. Enable this approach with the config.assets.digest setting.

**config.assets.digest = true**

You also want to confirm that caching is turned on in production.

**config.action\_controller.perform\_caching = true**

**e)** Provision memcache addon on heroku

$ heroku addons:add memcache

**f)** Deploy the app on heroku and use heroku logs command to view the log output

$ heroku logs --ps web -t

You should see **cache** entries in your production log-stream. Tailing **miss, store** tokens indicate that the item was not found in the cache but has been saved for the next request.

cache: [GET /assets/application-95bd4fe1de99c1cd91ec8e6f348a44bd.css] **miss, store**

cache: [GET /assets/application-95fca227f3857c8ac9e7ba4ffed80386.js] miss, store

cache: [GET /assets/rails-782b548cc1ba7f898cdad2d9eb8420d2.png] miss, store

**fresh** indicates item was found in cache and will be served from cache.

cache: [GET /assets/application-95bd4fe1de99c1cd91ec8e6f348a44bd.css] fresh

cache: [GET /assets/application-95fca227f3857c8ac9e7ba4ffed80386.js] fresh

cache: [GET /assets/rails-782b548cc1ba7f898cdad2d9eb8420d2.png] fresh

Your Rails 3.1+ application is now configured to cache static assets using Memcached, freeing up dynos to perform dynamic application requests.

$ heroku logs --ps web -t

**g)** If a setting is not configured properly, you might see **miss** in your logs instead of **store** or **fresh**.

cache: [GET /assets/application-95bd4fe1de99c1cd91ec8e6f348a44bd.css] **miss**

cache: [GET /assets/application-95fca227f3857c8ac9e7ba4ffed80386.js] miss

cache: [GET /assets/rails-782b548cc1ba7f898cdad2d9eb8420d2.png] miss

**When this happens ensure that the Cache-Control header exists by using curl to inspect asset response headers.**

$ curl -I 'http://rack-cache-demo.herokuapp.com/assets/shipit-72351bb81da0eca408d9bd8342f1972.jpg'

HTTP/1.1 200 OK

Age: 632

**Cache-Control: public, max-age=31536000**

Content-length: 70522

Etag: "72351bb81da0eca408d9bd8342f1b972"

Last-Modified: Sun, 25 Mar 2012 01:51:21 GMT

**X-Rack-Cache: fresh**

The response headers should contain **Cache-Control** with the value specific in the **config.static\_cache\_control** setting i.e.: public, max-age=31536000. Also confirm that you are seeing the **X-Rack-Cache** header indicating the status of your asset (fresh/store/miss). If you see unexpected results please check your production configuration settings.

**h)** If you modify a file and your server continues to serve the old file check that you committed it to your Git repository before deploying. You can check to see if it exists in your compiled code by using heroku run bash and listing the contents of the public/assets directory. This directory should contain the hashed asset file names.

$ **heroku run bash**

Running bash attached to terminal... up, run.1

$ **ls public/assets**

application-95bd4fe1de99c1cd91ec8e6f348a44bd.css application.css manifest.yml

application-95bd4fe1de99c1cd91ec8e6f348a44bd.css.gz application.css.gz rails-782b548cc1ba7f898cdad2d9eb8420d2.png

application-95fca227f3857c8ac9e7ba4ffed80386.js application.js rails.png

application-95fca227f3857c8ac9e7ba4ffed80386.js.gz application.js.gz

Also confirm that the file is listed in Rails’ **manifest.yml**.

rails.png: rails-782b548cc1ba7f898cdad2d9eb8420d2.png

application.js: application-95fca227f3857c8ac9e7ba4ffed80386.js

application.css: application-95bd4fe1de99c1cd91ec8e6f348a44bd.css

If the file you’re looking for does not show up try running bundle exec rake assets:precompile RAILS\_ENV=production locally and ensure that it is in your own public/assets directory.

4. Smush Images

at [www.smushit.com](http://www.smushit.com)