Node.js Express with Nginx Reverse Proxy and Cache

May 21, 2016 10:45 am Chanh Mac OS X

As a web developer, have you ever asked yourself whether to use a period or a plus sign to concatenate two strings? The former is used by PHP and the latter by Javascript. If you switch between the frontend Javascript and the backend PHP languages often, you'll find yourself asking this and other syntax questions.

Node.js eliminates that problem by supporting the use of Javascript in the backend. It is a Javascript runtime that replaces the PHP backend. However, using Javascript for both frontend and backend is not the best reason to migrate to Node.js. You will want to use Node.js because of the Node Package Manager (NPM). NPM makes finding, sharing, and reusing Javascript code packages a breeze. When implementing a backend function, the first step to take is to search for an NPM package that provides that function already.

Node.js Express is a simple but powerful Node.js web application framework. We'll use it as the backend to serve web pages. While Express can serve static files, Nginx is much faster at that task and provides many other benefits.

Using Nginx as a reverse proxy (browsers query Nginx which then calls Express) and cache for Express provides the following benefits:

- Nginx is built as a high performance server with many optimizations. It can handle many concurrent connections and can perform load balancing. Nginx can serve (or cache and serve) static files like HTML, javascript, images, and CSS more efficiently than Node.js.
- As the point of entry, Nginx provides better security because it is an older, proven web server solution. Nginx supports both the older SSL/TLS and newer HTTP/2 protocols.
- Nginx can be used for port 80 and 433 binding to avoid having to run Node.js as a root user, which is bad practice and possible vulnerability. (Under Unix, the first 1024 ports require root privileges to bind to.)

Below are the steps I took to setup Node.js and configure a Nginx reverse proxy and cache. Though I'm doing the steps below on Mac OS X 10 Yosemite, the Node.js code and Nginx configuration should be applicable to other platforms.

Install Node.js

We will install Node.js using MacPorts. Launch the Terminal app and run these commands:

```
# Node.js uses python scripts; python 2.7 is included with Mac OS X since v10.8
python --version

# Install Node.js
sudo port install nodejs
node -v

# Install Node Package Manager
sudo port install npm
npm -v

# Create a project directory
mkdir myproject
cd myproject
mkdir static

# Create a NPM project package.json file
```

```
npm init

# Install Express package
npm install express --save
```

The “npm init” command will prompt for the project name, version, and description in order to generate a “package.json” file. You can accept the defaults and edit the “package.json” file later.

The “npm install express –save” command will download the Express package to a node_modules subdirectory. The optional “–save” flag will add the Express package and its version as a dependency in the “package.json” file. (The alternative “–save-dev” flag will save the package as a development dependency instead.)

The benefit of the above is that you can give the “package.json” file to another developer and they can run the “npm install” command to install the specific versions of all dependent packages (as listed in the “package.json”).

Express Server

Create a a file named “server.js” with the following content:

```
// Require Express package
var express = require('express');

// Create an Express app
var app = express();

// Serve static files from static dir
app.use(express.static('static'));

// Handle get on root / request
app.get('/', function (req, res) {
res.send('Hello World!');
});

// Bind to port 8080
var server = app.listen(8080, function () {
var port = server.address().port;
console.log('Listening on port %s', port);
});
```

To test, do the following:

1. Put an image file, say “earth.gif”, under the static subdirectory.
2. In a Terminal window, launch the server with this command:

```
node server.js
```

3. Browse to http://localhost:8080/ and you will see a “Hello World!” response.
4. Browse to http://localhost:8080/earth.gif to see the static image file.
5. In the Terminal window running Node.js, press Ctrl-C to quit.

Instead of using the “node server.js” command, you can use the “npm start” command. If you look inside the “package.json” file, you will see that there exists a script target named “start” which runs the “node server.js” command.

Install Nginx

Install Nginx with this command:

```
sudo port install nginx
```

The Nginx configuration file is located at “/opt/local/etc/nginx/nginx.conf”. The default document root is set to “share/nginx/html”, which maps to the “/opt/local/share/nginx/html” directory.

You can start, reload, and stop Nginx using these commands:

```
# Start Nginx
sudo port load nginx

# Reload the config (actually restarts)
sudo nginx -s reload

# Stop Nginx
sudo port unload nginx

# Check to see if Nginx is running
ps -e | grep nginx
```

While Nginx is running, browse to http://localhost/ to see the Nginx welcome page.

If you need to troubleshoot, the Nginx error log file is located at "/opt/local/var/log/nginx/error.log".

Nginx Reverse Proxy

When Nginx proxies a request to Node.js, it will optimized the request headers it receives from the client:

- Nginx gets rid of any empty headers from the proxied request.
- Nginx considers any header names with underscores as invalid. It will remove them. If you wish to preserve these headers, set the Nginx "underscores_in_headers" directive to "on".
- The "Host" header is re-written to the value defined by the $proxy_host variable. This will be the IP address or hostname and port number of the upstream, as defined by the "proxy_pass" directive.
- The "Connection" header is changed to "close" to indicate to the upstream server that this connection will be closed once the original request is responded to.

Configure the Nginx reverse proxy by running "sudo nano /opt/local/etc/nginx/nginx.conf" and making the following changes to the root location:

```
server {
    # ...

    # Helpful headers to pass to Node.js
    proxy_set_header Host $host;
    proxy_set_header X-Forwarded-Proto $scheme; #http pr https
    proxy_set_header X-Real-IP $remote_addr; #client IP address
    # List of IP addresses client has been proxied through until now
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;

    location / {
        #root   share/nginx/html;
        #index  index.html index.htm;

        proxy_pass http://localhost:8080;
    }
```

Reload the Nginx config with the "sudo nginx -s reload" command. Browsing to http://localhost/ will now show the Node.js "Hello World!" message, instead of the Nginx welcome page.

By default, the "Host" header is set to "$proxy_host", which is the upstream hostname or IP address and port defined in the "proxy_pass" definition. In the above, we have overridden it to be "$host" which is set to, in order of preference: the hostname from the request line itself, the optional "Host" header from the client request, or the server name matching the request. This is a best practice for Nginx and ensures that the "Host" header passed to the proxied server is as accurate as possible.

Note: The headers sent by the client are available in Nginx as variables. The variables start with an "$http_" prefix, followed by the header name in lowercase, and with any dashes replaced by underscores. So the client "Host" header is available in Nginx as "$http_host".

If you wish to reverse proxy a non-root path location, use this Nginx location configuration:

```
location /proxy/ {
    # Ending slash prevents passing /proxy/ path to Node.js
    proxy_pass http://localhost:8080/; # Ending slash required!
}
```

Browse to http://localhost/proxy/ to test.

Nginx Serves Static Files

If Nginx has access to the Node.js project directory (for example, "/Users/myuser/myproject"), it is best to configure Nginx to serve the static files directly.

Run "sudo nano /opt/local/etc/nginx/nginx.conf" and add the following "/static/" location:

```
location /static/ {
    root /Users/myuser/myproject; # Node.js project location
    expires 30d; # Cache-Control: client cache valid for 30 days
}
```

Browse to http://localhost/static/earth.gif to test.

The "expires 30d" command adds a "Cache-Control" response header telling the browser client to only cache the static resource for 30 days maximum. Install Curl and dump the response header to see the "Cache-Control" value:

```
# Install Curl
sudo port install curl
```

```
# Run Curl to dump response headers
curl -X GET -I http://localhost/static/earth.gif
# Cache-Control: max-age=2592000
```

Unfortunately, the above solution requires the URL to contain the "/static/" path. One workaround to avoid the "/static/" path is to have Nginx serve static files with specific extensions like so:

```
# location ~* means to use case-insensitive match
location ~* ^.+\.(jpg|jpeg|gif|png|ico|css|bmp|js|html|htm)$ {
root /Users/chvuong/nodejs/static;
expires 30d;
}
```

Because Node.js returns "Cache-Control" with an immediate expiration, we can verify that Nginx is serving the static file by doing the following:

```
# Get static image file from Nginx
curl -X GET -I http://localhost/earth.gif
# Cache-Control: max-age=2592000

# Get static image file directly from Node.js
curl -X GET -I http://localhost:8080/earth.gif
# Cache-Control: public, max-age=0
```

If we use Node.js to provide APIs only, we can configure Nginx to return static files only if they exist. Because API paths don't usually correspond to existing directories and/or static files, Nginx will pass the API calls to Node.js. Here's the Nginx configuration to use:

```
location / {
root /Users/chvuong/nodejs/static;
expires 30d;

# Return static file if exist; otherwise pass to Node.js
try_files $uri @nodejs;
}

location @nodejs {
proxy_pass http://localhost:8080;
}
```

Please remove or comment out the previous "location ~* ^.+\.(jpg|jpeg|gif|png|ico|css|bmp|js|html|htm)$" block because it will interfere with the above configuration.

Test using the Curl commands above to get the "earth.gif" static file. You can also edit "server.js" to remove or comment out the line below to prevent Node.js from serving static files:

```
// Serve static files from static dir
//app.use(express.static('static'));
```

Nginx Cache

If Nginx does not have access to the static files (for example, Nginx is running on another server), you can configure Nginx to cache the static files returned by Node.js. If the browser client requests a static file which is already cached, Nginx will return it without having to request that file from Node.js again.

Run "sudo nano /opt/local/etc/nginx/nginx.conf" and add the following cache configuration:

```
http {
  # ...

# Cache for static files
proxy_cache_path /tmp/nginx-cache levels=1:2 keys_zone=staticcache:8m max_size=100m inactive=60m use_temp_path=off;
# keyzone size 8MB, cache size 100MB, inactive delete 60min
proxy_cache_key "$scheme$request_method$host$request_uri";
proxy_cache_valid 200 302 60m; # cache successful responses for 60 minutes
proxy_cache_valid 404 1m; # expire 404 responses 1 minute

server {
  # ...

location / {
proxy_pass http://localhost:8080;
}
```

```
# Only cache static files; don't cache the dynamic API response!
location ~* ^.+\.(jpg|jpeg|gif|png|ico|css|bmp|js|html|htm)$ {
proxy_cache staticcache; # Use "staticcache" we defined above
proxy_cache_bypass $http_cache_control; # Support client "Cache-Control: no-cache" directive
add_header X-Proxy-Cache $upstream_cache_status; # Hit or Miss

# Nginx cache to ignore Node.js default "Cache-Control: public, max-age=0"
# and don't pass it on to clients
proxy_ignore_headers Cache-Control;
proxy_hide_header Cache-Control;
expires 60m; # "Cache-Control: max-age=3600" tells client to cache for 60 minutes

proxy_pass http://localhost:8080;
}
```

Caveats:

- Double-check that “server.js” is configured to serve static files:

```
// Serve static files from static dir
app.use(express.static('static'));
```

- Nginx will create the “/tmp/nginx-cache” directory because “/tmp” allows write access to everyone. If you change the cache directory location, please make sure to manually create the necessary directories. You will see an error in the Nginx error log if Nginx can’t create or access the cache directory.

You can test the cache by using Curl and taking note of the “X-Proxy-Cache” response header:

```
# First try will result in cache miss
curl -X GET -I http://localhost/earth.gif
# X-Proxy-Cache: MISS

# Second try will result in cache hit
curl -X GET -I http://localhost/earth.gif
# X-Proxy-Cache: HIT
```

If you wish to cache a non-root path location, use this Nginx location configuration:

```
location /proxy/ {
proxy_pass http://localhost:8080/; # Ending slash required!
}

# Match string changed to capture path after /proxy/
location ~* ^/proxy/(.+\.(jpg|jpeg|gif|png|ico|css|bmp|js|html|htm))$ {
proxy_cache staticcache;
proxy_cache_bypass $http_cache_control;
add_header X-Proxy-Cache $upstream_cache_status;

proxy_ignore_headers Cache-Control;
proxy_hide_header Cache-Control;
expires 60m;

# Pass captured path string without /proxy/ to Node.js
proxy_pass http://localhost:8080/$1;
}
```

Nginx Load Balancing

Once you have Nginx reverse proxy working, load balancing is very simple. Here’s the Nginx load balancing configuration:

```
http {
  # ...

# Node.js servers for load balancing
upstream nodejs-backend {
least_conn; # Give new connection to backend with least active connections
localhost:8080;
nodejs2.example.com; # default port 80
nodejs2.example.com:8080;
}

server {
  # ...
```

```
location / {
# Pass to the upstream name, instead of the specific nodejs hostname
proxy_pass http://nodejs-backend;
}
```

I only tested the above with one Node.js server in the upstream definition (because I only have one server). It worked for one server and it should work for more.

Most info derived from:

- Tutorial: Get started with Node.js on Mac (Beginner)
- Node.js – Express Framework
- Serving static files in Express
- Understanding Nginx HTTP Proxying, Load Balancing, Buffering, and Caching
- Nginx Caching
- A Guide to Caching with NGINX
- Quick-Start: Regex Cheat Sheet
- nginx as cache proxy not caching anything

Leave a Comment

Name

Mail (will not be published)

Website (optional)

Your comment

Submit Comment

You can use these tags: <a href="" title=""> <abbr title=""> <acronym title=""> <b> <blockquote cite=""> <cite> <code> <del datetime=""> <i> <q cite=""> <s> <strike>

Please note: Comment moderation is enabled and may delay your comment. There is no need to resubmit your comment.

Categories

- Audio Visual
- Hardware
- Internet
- Linux
- Mac OS X
- Mobile Devices
- Windows
- Windows Development

Mac OS X

- Node.js Express with Nginx Reverse Proxy and Cache
- Setup LAMP (Linux, Apache, MySQL, PHP) on Mac OS X 10.10 Yosemite
- Windows 8.1 Boot Camp on 2015 Macbook Pro Retina 13 Inch
- Revert Mac OS X Yosemite Core Storage Back to Mac OS Extended HFS+
- Bootable USB Flash Drive to Install Mac OS X 10.10 Yosemite
- Update to Latest Subversion Using MacPorts
- Make Mac Screen Lock Secure and Convenient
- Sharing Applications Between Mac OS X and Windows

Customize My Own Mac OS X 10.8 Mountain Leopard

Setup Mac OS X 10.8 Mountain Lion, Windows 7 Boot Camp, and Shared FAT32 Partition

Recent Posts

- Outdoor Home Video Surveillance With CleverLoop
- Clone a Big Hard Drive to a Smaller One
- Node.js Express with Nginx Reverse Proxy and Cache
- Free SSL Certificate from Let's Encrypt for Nginx
- Nginx with PHP and MySQL on Windows 7
- Setup LAMP (Linux, Apache, MySQL, PHP) on Mac OS X 10.10 Yosemite
- Automate Remote Backup of WordPress Database
- Rotate Video Without Black Bars
- Clone a Hard Drive Using Clonezilla Live
- Create Bootable USB Flash Drive With DiskPart Command-Line Utility

Archives

► 2016 (7)
► 2015 (10)
► 2014 (12)
► 2013 (5)
► 2012 (15)
► 2011 (21)
► 2010 (11)
► 2009 (9)
► 2008 (13)
► 2007 (4)
► 2006 (4)
► 2004 (1)
► 2003 (2)
► 2002 (5)
► 2001 (2)
► 1999 (1)
► 1998 (2)
► 1995 (1)

Modified theme based on design by Opisy GG Sennik with RW-WPthemes.