Running Node.js in Production

@DaveChubbuck
https://github.com/davidchubbs

IT WORKED FINE IN TEST

OPS PROBLEM NOW

Options

- PaaS, like Heroku
- Forever
- Upstart
- pm2
- Passenger
- Others?

If interested in learning more about pm2:

https://github.com/Unitech/pm2

PHUSION PASSENGER

EASY DEPLOYMENT FOR **RUBY, PYTHON, NODE.JS** AND **METEOR**
KEEP YOUR USERS HAPPY AND SAVE YOUR BUSINESS TIME AND MONEY

DOWNLOAD

TRUSTED BY LEADING BRANDS AROUND THE WORLD

PIXAR · The New York Times · airbnb · OAKLEY · symantec.
JUNIPER NETWORKS · 37signals · salesforce.com · WellCare · SponsorPay
NBCUniversal · AMERICAN EXPRESS · HITACHI Inspire the Next · DI Digitally Imported

and currently over **350,000** other sites!

Why Passenger? 4 minute video clip:

http://vimeo.com/phusionnl/review/84945384/73fe7432ee

Nginx 101

```
# simple directive
config value;

# block directive
server {
    . . .
}
```

```
# block directives can have simple & block directives inside { }

# Configuration files (files containing directives) can be split up into
separate files.
# /etc/nginx/nginx.conf       - main config file
# /etc/nginx/sites-available/ - directory of custom config files
# /etc/nginx/sites-enabled/   - directory of links to /sites-available/
```

Nginx 101

```
http {       # /etc/nginx/nginx.conf
    access_log   /var/log/nginx/access.log;
    . . .
}

server {     # /etc/nginx/sites-enabled/default.conf
    listen       80 default_server;
    server_name  domain.com;
    # overwrite http block access_log
    access_log   /var/log/nginx/access.global.log;
    . . .
}

server {     # /etc/nginx/sites-enabled/sub.domain.conf
    listen       80;
    server_name sub.domain.com;
    . . .
}
```

Nginx

```
http {       # /etc/nginx/nginx.conf
    access_log   /var/log/nginx/access.log;
    . . .
}

server {     # /etc/nginx/sites-enabled/default.conf
    listen       80 default_server;
    server_name  domain.com;
    # overwrite http block access_log
    access_log   /var/log/nginx/access.global.log;
    . . .
}

server {     # /etc/nginx/sites-enabled/sub.domain.conf
    listen       80;
    server_name sub.domain.com;
    . . .
}
```

```
GET /contact HTTP/1.1
User-Agent: curl/7.30.0
Host: domain.com
Accept: */*

GET / HTTP/1.1
User-Agent: curl/7.30.0
Host: 146.148.47.47
Accept: */*

GET / HTTP/1.1
User-Agent: curl/7.30.0
Host: sub.domain.com
Accept: */*
```

One major benefit of using Passenger is that it's built into Nginx, meaning your Nginx config files == Passenger config files. When Nginx starts up, Passenger starts up. Plus this allows Passenger to add Nginx sugar.

Forever Nginx Setup Example

```
upstream name {
    server localhost:3000;
    server localhost:3001;
}

server {
    listen      80;
    server_name domain.com;
    location / {
        proxy_pass          http://name;
        proxy_redirect      off;
        proxy_set_header    x-real-ip  $remote_addr;
        . . . etc.
    }
    location /static/ {
        root  /path/to/static;
    }
}
```

Passenger Nginx Setup Example

```
server {
    listen      80;
    server_name domain.com;
    passenger_enabled  on;
    passenger_app_root /home/nodejs/app-name;
    root               /home/nodejs/app-name/public;
}
```

```
# Passenger starts Node instances for you, manages their ports, and
balances the load across those instances.
# Passenger can cluster your app without any app-code modification.
# Passenger will serve files statically if in `root`, else, will have
Node instance handle request.
# Owner of server.js/app.js file becomes owner of Node process.
```

passenger-status

```
root@node-apps-1:~# passenger-status
Version : 4.0.48
Date    : 2014-08-27 17:47:50 +0000
Instance: 4325
----------- General information -----------
Max pool size : 4
Processes     : 4
Requests in top-level queue : 0

----------- Application groups -----------
/home/nodejs/admin-app#default:
  App root: /home/nodejs/admin-app
  Requests in queue: 0
  * PID: 4350    Sessions: 0       Processed: 10      Uptime: 40s
    CPU: 0%      Memory  : 13M     Last used: 0s ago
  * PID: 4363    Sessions: 0       Processed: 0       Uptime: 40s
    CPU: 0%      Memory  : 12M     Last used: 40s ago
  * PID: 4376    Sessions: 0       Processed: 0       Uptime: 39s
    CPU: 0%      Memory  : 12M     Last used: 39s ago
  * PID: 4389    Sessions: 0       Processed: 0       Uptime: 39s
    CPU: 0%      Memory  : 12M     Last used: 39s ago
```

passenger-memory-stats

```
--------- Nginx processes ----------
PID    PPID  VMSize    Private  Name
------------------------------------
4325  1     112.0 MB  0.2 MB   nginx: master process /usr/sbin/nginx
4326  4325  112.4 MB  0.5 MB   nginx: worker process
4327  4325  112.4 MB  0.5 MB   nginx: worker process
4329  4325  112.5 MB  0.7 MB   nginx: worker process
4330  4325  112.4 MB  0.5 MB   nginx: worker process
### Processes: 5
### Total private dirty RSS: 2.29 MB


---- Passenger processes -----
PID    VMSize    Private  Name
------------------------------
4307  25.5 MB   0.3 MB   PassengerWatchdog
4310  172.8 MB  1.1 MB   PassengerHelperAgent
4316  145.0 MB  0.9 MB   PassengerLoggingAgent
4350  592.1 MB  14.1 MB  Passenger NodeApp: /home/nodejs/admin-app
4363  592.1 MB  12.9 MB  Passenger NodeApp: /home/nodejs/admin-app
4376  656.1 MB  13.0 MB  Passenger NodeApp: /home/nodejs/admin-app
4389  592.1 MB  12.9 MB  Passenger NodeApp: /home/nodejs/admin-app
### Processes: 7
### Total private dirty RSS: 55.26 MB
```

Load Balancing

Because Passenger can spin up its own instances, *each application can have a fluid number of instances*. If an app is experiencing heavy load, another instance is created; if load lightens, instances are removed. Consequentially, hosting multiple apps on the same server can save you money by apps being able to share your server's resource buffer (like extra CPU cores) instead of paying for each app to have its own server and buffer.

- Configurations allow you to set min and max instances per app, and min and max instances per server.
- Apps can be set to allow idling (0 instances) if no requests occur for *n* seconds.
- Passenger can cache instantiation process for faster instance creation.

```
server {   # domain.com
    . . .
    passenger_min_instances  1;    # default: 1; 0 = can idle if timeout
    passenger_max_instances  8;    # default: 0; 0 = no max
}

server {   # admin.domain.com
    . . .
    passenger_min_instances  0;
    passenger_max_instances  2;
}
```

```
http {
    . . .
    passenger_max_pool_size           20; # default: 6
    passenger_max_instances_per_app   10; # default: 0; 0 = no max
    passenger_pool_idle_time          0;  # default: 300; 0 = no idling
    passenger_max_preloader_idle_time 0;  # default: 300; 0 = no idling
    . . .
    passenger_rolling_restarts         on;# enterprise; default: off
    passenger_resist_deployment_errors on;# enterprise; default: off
}
```

```
# App Setup
passenger_nodejs  /path/to/node;   # run separate versions of Node
passenger_app_env development;     # NODE_ENV; default: production
passenger_startup_file server.js;  # default: app.js
passenger_app_type node;           # required if ^

# Memory Leaks
passenger_max_requests  0;         # restart after n requests
passenger_memory_limit  0;         # enterprise; restart after n MB

# Request Handling
passenger_max_request_queue_size 100;             # default: 100
passenger_request_queue_overflow_status_code 503; # default: 503
passenger_sticky_sessions        off;             # default: off
passenger_sticky_sessions_cookie_name _passenger_route;

# Logging
passenger_log_level 1;
passenger_debug_log_file /log/path;   # http scope only

# most of these can be in http, server, location, or if Nginx blocks
```

Restarting Server

```
passenger-config restart-app /app/root/dir    # never rolling restart
passenger-config restart-app /app/root/dir --rolling-restart

# or if only allowing FTP access, touch tmp/restart.txt file
# ^ will be rolling if rolling is enabled
```

Downsides

- Enterprise licensing. pm2 is free.
- Logging can’t be split up by application, though logs do contain PID. (meaning `passenger_debug_log_file` can only be set in `http` block)
  - Example Log Line: `App <PID> stdout: <message>`

- Installation: https://www.phusionpassenger.com/download
- Passenger + Nginx Tutorial: https://github.com/phusion/passenger/wiki/Phusion-Passenger%3A-Node.js-tutorial
- Passenger + Nginx API Docs: https://www.phusionpassenger.com/documentation/Users%20guide%20Nginx.html