Node.js + Nginx - What now?

Ask Question

I've set up Node.js and Nginx on my server. Now I want to use it, but, before I start there are 2 questions:

1. How should they work together? How should I handle the requests?
2. There are 2 concepts for a Node.js server, which one is better:

   a. Create a separate HTTP server for each website that needs it. Then load all JavaScript code at the start of the program, so the code is interpreted once.

   b. Create one single Node.js server which handles all Node.js requests. This reads the requested files and evals their contents. So the files are interpreted on each request, but the server logic is much simpler.

It's not clear for me how to use Node.js correctly.

node.js nginx concept

edited Apr 17 '15 at 23:55
Igor Antun
103 1 2 14

asked Feb 15 '11 at 20:49
Van Coding
10.9k 15 66 113

11 Answers

Nginx works as a front end server, which in this case proxies the requests to a node.js server. Therefore you need to setup an nginx config file for node.

This is what I have done in my Ubuntu box:

Create the file `yourdomain.com` at `/etc/nginx/sites-available/` :

```
vim /etc/nginx/sites-available/yourdomain.com
```

In it you should have something like:

```
# the IP(s) on which your node server is running. I chose port 3000.
upstream app_yourdomain {
    server 127.0.0.1:3000;
    keepalive 8;
}

# the nginx server instance
server {
    listen 80;
    listen [::]:80;
    server_name yourdomain.com www.yourdomain.com;
    access_log /var/log/nginx/yourdomain.com.log;

    # pass the request to the node.js server with the correct headers
    # and much more can be added, see nginx config options
    location / {
      proxy_set_header X-Real-IP $remote_addr;
      proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
      proxy_set_header Host $http_host;
      proxy_set_header X-NginX-Proxy true;

      proxy_pass http://app_yourdomain/;
      proxy_redirect off;
    }
 }
```

If you want nginx (>= 1.3.13) to handle websocket requests as well, add the following lines in the `location /` section:

```
proxy_http_version 1.1;
proxy_set_header Upgrade $http_upgrade;
proxy_set_header Connection "upgrade";
```

Once you have this setup you must enable the site defined in the config file above:

```
cd /etc/nginx/sites-enabled/
ln -s /etc/nginx/sites-available/yourdomain.com yourdomain.com
```

Create your node server app at `/var/www/yourdomain/app.js` and run it at `localhost:3000`

```
var http = require('http');

http.createServer(function (req, res) {
    res.writeHead(200, {'Content-Type': 'text/plain'});
    res.end('Hello World\n');
}).listen(3000, "127.0.0.1");
console.log('Server running at http://127.0.0.1:3000/');
```

Test for syntax mistakes:

```
nginx -t
```

Restart nginx:

```
sudo /etc/init.d/nginx restart
```

Lastly start the node server:

```
cd /var/www/yourdomain/ && node app.js
```

Now you should see "Hello World" at yourdomain.com

One last note with regards to starting the node server: you should use some kind of monitoring system for the node daemon. There is an awesome tutorial on node with upstart and monit.

edited Mar 10 at 22:08
RAnders00
2,825 3 22 49

answered Feb 16 '11 at 10:20
Joao Da Silva
11.9k 1 10 7

9 Thanks for the post, will nginx cache node.js responses for the server above, or rerun them each time. – William Jul 19 '11 at 19:31

68 Is there any reason why you can't just do `location / { proxy_pass http://127.0.0.1:3000; }` ? Why do you need the whole `upstream` config bit? – Robin Winslow Nov 20 '11 at 23:59

16 +1, Very straightforward and simple answer to a common question; great for people who want to set up virtual hosts using node and nginx. The only thing I think you missed is a qualitative answer to why nginx-in-front-of-node is best for serving multiple vhosts (asker's second question). – Paul d'Aoust Mar 6 '12 at 17:52

29 @Robin Winslow in case you want to add more servers for servers for load balancing. – Joao Da Silva Aug 16 '12 at 10:53

64 It should be noted that this (very helpful) answer refers to one flavor of nginx that, by default, comes with `sites-enabled` and `sites-available` directories inside `/etc/nginx` . If your version came without these two directories, it likely has a single `conf.d` directory instead. In that case, following these instructions would have no effect, UNLESS you modify the `include` statement inside the file `nginx.conf` to point to `sites-enabled` instead of the default `conf.d` . Hope that makes sense. It should become self explanatory once you see the said `include` statement inside `nginx.conf` . – meetamit Oct 18 '12 at 11:35

You can also setup multiple domain with nginx, forwarding to multiple node.js processes.

For example to achieve these:

- domain1.com -> to Node.js process running locally http://127.0.0.1:4000
- domain2.com -> to Node.js process running locally http://127.0.0.1:5000

/etc/nginx/sites-enabled/domain1

```
server {
    listen 80;
    listen [::]:80;
    server_name domain1.com;
    access_log /var/log/nginx/domain1.access.log;
    location / {
        proxy_pass    http://127.0.0.1:4000/;
    }
}
```

In /etc/nginx/sites-enabled/domain2

```
server {
    listen 80;
    listen [::]:80;
    server_name domain2.com;
    access_log /var/log/nginx/domain2.access.log;
    location / {
        proxy_pass    http://127.0.0.1:5000/;
    }
}
```

edited Mar 10 at 22:09
RAnders00
2,825 3 22 49

answered May 2 '12 at 11:38
250R
21.1k 6 27 23

4 I am using your method of proxy_pass, but for some reason `http://example.com` gets automatically `302` 'd to `http://www.example.com` . Why is that? – Kristian Nov 13 '15 at 8:00

Do you have Cloudflare or something similar? The configuration above shouldn't redirect at all. – ozzieisaacs Jun 18 at 21:46

You can also have different urls for apps in one server configuration:

- yourdomain.com/app1/* -> to Node.js process running locally http://127.0.0.1:3000
- yourdomain.com/app2/* -> to Node.js process running locally http://127.0.0.1:4000

In **/etc/nginx/sites-enabled/yourdomain**:

```
server {
    listen 80;
    listen [::]:80;
    server_name yourdomain.com;

    location ^~ /app1/{
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header Host $http_host;
        proxy_set_header X-NginX-Proxy true;
        proxy_pass    http://127.0.0.1:3000/;
    }

    location ^~ /app2/{
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header Host $http_host;
        proxy_set_header X-NginX-Proxy true;
        proxy_pass    http://127.0.0.1:4000/;
    }
}
```

Restart nginx:

```
sudo service nginx restart
```

Starting applications.

node app1.js

```
var http = require('http');
http.createServer(function (req, res) {
    res.writeHead(200, {'Content-Type': 'text/plain'});
    res.end('Hello from app1!\n');
}).listen(3000, "127.0.0.1");
console.log('Server running at http://127.0.0.1:3000/');
```

node app2.js

```
var http = require('http');
http.createServer(function (req, res) {
    res.writeHead(200, {'Content-Type': 'text/plain'});
    res.end('Hello from app2!\n');
}).listen(4000, "127.0.0.1");
console.log('Server running at http://127.0.0.1:4000/');
```

edited Mar 10 at 22:10
RAnders00
2,825 3 22 49

answered Mar 31 '15 at 17:31
0x8BADF00D
3,652 1 25 29

nginx is not free right? – Asiri Liyana Arachchi Sep 7 '16 at 8:37

3 Open source community version is free but they have version with other features which are not free. nginx.com/products/feature-matrix – 0x8BADF00D Sep 7 '16 at 19:34

Sorry for my ignorance. What is the purpose, benefits of serving it this way? do you have any example or case of use? Thanks in advance. – Mauro Aguilar Jun 29 '17 at 21:28

2 @MauroAguilar If you need 2 node.js app on one server you can serve them using

suggested way (using different ports). In my cases it were two different test apps. – 0x8BADF00D Jun 29 '17 at 21:34

1 @MauroAguilar, you can run them in single one and there is no benefit if it could be part of one project and has same purpose. But if you need to run 2 different projects with different purposes and with different configurations on one server then you have benefit to use this config. – 0x8BADF00D Jun 29 '17 at 21:54

I proxy independent Node Express applications through Nginx.

Thus new applications can be easily mounted and I can also run other stuff on the same server at different locations.

Here are more details on my setup with Nginx configuration example:

> **Deploy multiple Node applications on one web server in subfolders with Nginx**
>
> Things get tricky with Node when you need to move your application from from localhost to the internet.
>
> There is no common approach for Node deployment.
>
> Google can find tons of articles on this topic, but I was struggling to find the proper solution for the setup I need.
>
> Basically, I have a web server and I want Node applications to be mounted to subfolders (i.e. http://myhost/demo/pet-project/) without introducing any configuration dependency to the application code.
>
> At the same time I want other stuff like blog to run on the same web server.
>
> Sounds simple huh? Apparently not.
>
> In many examples on the web Node applications either run on port 80 or proxied by Nginx to the root.
>
> Even though both approaches are valid for certain use cases, they do not meet my simple yet a little bit exotic criteria.
>
> That is why I created my own Nginx configuration and here is an extract:

```
upstream pet_project {
  server localhost:3000;
}

server {
  listen 80;
  listen [::]:80;
  server_name frontend;

  location /demo/pet-project {
    alias /opt/demo/pet-project/public/;
    try_files $uri $uri/ @pet-project;
  }

  location @pet-project {
    rewrite /demo/pet-project(.*) $1 break;

    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header Host $proxy_host;
    proxy_set_header X-NginX-Proxy true;

    proxy_pass http://pet_project;
    proxy_redirect http://pet_project/ /demo/pet-project/;
  }
}
```

From this example you can notice that I mount my Pet Project Node application running on port 3000 to http://myhost/demo/pet-project.

First Nginx checks if whether the requested resource is a static file available at */opt/demo/pet-project/public/* and if so it serves it as is that is highly efficient, so we do not need to have a redundant layer like Connect static middleware.

Then all other requests are overwritten and proxied to *Pet Project Node* application, so the Node application does not need to know where it is actually mounted and thus can be moved anywhere purely by configuration.

proxy_redirect is a must to handle Location header properly. This is extremely important if you use *res.redirect()* in your Node application.

You can easily replicate this setup for multiple Node applications running on different ports and add more location handlers for other purposes.

From: http://skovalyov.blogspot.dk/2012/07/deploy-multiple-node-applications-on.html

edited Mar 10 at 22:10
RAnders00
2,825 3 22 49

answered Jul 13 '12 at 7:53
skovalyov
1,409 1 11 12

1 Why and how you should do it in subdomains instead: skovalyov.blogspot.dk/2012/10/… – skovalyov Oct 23 '12 at 9:57

Link only answer … can you please summarize the relevant parts in your answer in case you blog is gone? – kaiser Mar 14 '16 at 14:59

1 @kaiser updated with the content of the article. – skovalyov Mar 15 '16 at 21:10

Node.js with Nginx configuration.

```
$ sudo nano /etc/nginx/sites-available/subdomain.your_domain.com
```

add the following configuration so that Nginx acting as a proxy redirect to port 3000 traffic from the server when we come from “subdomain.your_domain.com”

```
upstream subdomain.your_domain.com {
  server 127.0.0.1:3000;
}
server {
  listen 80;
  listen [::]:80;
  server_name subdomain.your_domain.com;
  access_log /var/log/nginx/subdomain.your_domain.access.log;
  error_log /var/log/nginx/subdomain.your_domain.error.log debug;
  location / {
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarder-For $proxy_add_x_forwarded_for;
    proxy_set_header Host $http_host;
    proxy_set_header X-NginX-Proxy true;
    proxy_pass http://subdomain.your_domain.com;
    proxy_redirect off;
  }
}
```

edited Mar 10 at 22:11
RAnders00
2,825 3 22 49

answered Mar 17 '14 at 12:24
aquadir
81 1 1

answering your question 2:

I would use option `b` simply because it consumes much less resources. with option 'a', every client will cause the server to consume a lot of memory, loading all the files you need (even though i like php, this is one of the problems with it). With option 'b' you can load your libraries (reusable code) and share them among all client requests.

But be ware that if you have multiple cores you should tweak node.js to use all of them.

answered Jan 3 '12 at 4:51
hugo_leonardo
4,578 5 26 47

2 Follow this advice if resources are your most important issue (unlikely). There are different compromises between (a) and (b). Option (a) is probably better if you wish to the sites to be more independent e.g. site restart or maintenance, db

connections, code base, library dependencies, moving sites between servers, etc. – robocat Feb 8 '16 at 0:56

You could also use node.js to generate static files into a directory served by nginx. Of course, some dynamic parts of your site could be served by node, and some by nginx (static).

Having some of them served by nginx increases your performance..

answered Jul 5 '13 at 14:17

code ninja
6,394 6 36 64

We can easily setup a Nodejs app by Nginx acting as a reverse proxy.
The following configuration assumes the NodeJS application is running on 127.0.0.1:8080,

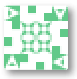
```
server{
   server_name domain.com sub.domain.com; # multiple domains

   location /{
    proxy_pass http://127.0.0.1:8080;
    proxy_set_header Host $host;
    proxy_pass_request_headers on;
   }

   location /static/{
     alias /absolute/path/to/static/files; # nginx will handle js/css
   }
 }
```

in above setup your Nodejs app will,

- get `HTTP_HOST` header where you can apply domain specific logic to serve the response. '
- Your Application must be managed by a process manager like pm2 or supervisor for handling situations/reusing sockets or resources etc.
- Setup an error reporting service for getting production errors like sentry or rollbar

NOTE: you can setup logic for handing domain specific request routes, create a middleware for expressjs application

answered Jan 7 '17 at 18:54


I Am Batman
2,302 1 13 30

I made a repository in Github which you can clone, vagrant-node-nginxboilerplate

basically the node.js app at `/var/www/nodeapp` is

```
var http = require('http');
http.createServer(function (req, res) {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end('Hello World\n');
}).listen(4570, '127.0.0.1');

console.log('Node Server running at 127.0.0.1:4570/');
```

and the nginx config at `/etc/nginx/sites-available/` is

```
server {
        listen 80 default_server;
        listen [::]:80 default_server;

        root /var/www/nodeapp;
        index index.html index.htm;

        server_name localhost;

        location / {
          proxy_pass http://127.0.0.1:4570;
          proxy_http_version 1.1;
          proxy_set_header Upgrade $http_upgrade;
          proxy_set_header Connection 'upgrade';
          proxy_set_header Host $host;
          proxy_cache_bypass $http_upgrade;
        }
}
```

edited Mar 10 at 22:11
RAnders00
2,825 3 22 49

answered Mar 18 '15 at 10:24
steven iseki
7,180 8 54 82

Nginx can act as a reverse proxy server which works just like a project manager. When it gets a request it analyses it and forwards the request to upstream(project members) or handles itself. Nginx has two ways of handling a request based on how its configured.

- serve the request
- forward the request to another server

```
server{
 server_name mydomain.com sub.mydomain.com;

 location /{
  proxy_pass http://127.0.0.1:8000;
  proxy_set_header Host $host;
```

```
         proxy_pass_request_headers on;
        }

        location /static/{
          alias /my/static/files/path;
        }

    }
```

Server the request

> With this configuration when the request url is `mydomain.com/static/myjs.js` it returns the `myjs.js` file in `/my/static/files/path` folder. When you configure nginx to serve static files it handles the request itself.

forward the request to another server

> When the request url is `mydomain.com/dothis` nginx will forwards the request to http://127.0.0.1:8000. The service which is running on the localhost 8000 port will receive the request and returns the response to nginx and nginx returns the response to the client.

When you run node.js server on the port 8000 nginx will forward the request to node.js. Write node.js logic and handle the request. That's it you have your nodejs server running behind the nginx server.

If you wish to run any other services other than nodejs just run another service like Django, flask, php on different ports and config it in nginx.

answered Sep 27 '17 at 8:07

Vkreddy Komatireddy
101 1 5

You can run nodejs using pm2 if you want to manage each microservice means and run it. Node will be running in a port right just configure that port in nginx(/etc/nginx/sites-enabled/domain.com)

```
server{
    listen 80;
    server_name domain.com www.domain.com;

  location / {
     return 403;
  }
    location /url {
        proxy_pass http://localhost:51967/info;
    }
}
```

Check whether localhost is running or not by using ping.

And

```
Create one single Node.js server which handles all Node.js requests. This rea
requested files and evals their contents. So the files are interpreted on eac
but the server logic is much simpler.
```

This is best and as you said easier too

answered Jul 6 at 12:01

gokul kandasamy
17 7

protected by Tushar Gupta Jul 16 '15 at 7:02

Thank you for your interest in this question. Because it has attracted low-quality or spam answers that had to be removed, posting an answer now requires 10 reputation on this site (the association bonus does not count).

Would you like to answer one of these unanswered questions instead?