

# How to deploy Express.js app to the web using node.js and digital ocean

by Chris Skov

**Step 1:** Create droplet with Node.js quick start option

**Step 2:** Add domain name to your droplet

Pictured below is a screenshot of the desired setup.

The screenshot shows the DigitalOcean DNS management interface. At the top, there are tabs for different record types: A, AAAA, CNAME, MX, TXT, NS, SRV, and CAA. The 'A' tab is selected. Below the tabs, there is a text input field for the hostname, a dropdown menu for 'WILL DIRECT TO', and a text input field for 'TTL (SECONDS)'. The hostname field contains 'greaterevil.goinghandev.com', the dropdown menu is set to 'Select resource or enter custom IP', and the TTL field is set to '3600'. A 'Create Record' button is visible to the right. Below this form, there is a section titled 'DNS records' which contains a table with the following data:

Type	Hostname	Value	TTL (seconds)
A	greaterevil.goinghandev.com	directs to 134.122.64.191	3600

**Step 3:** Setup non-root user on droplet and grant the user admin privileges.

Guide can be found here

<https://www.digitalocean.com/community/tutorials/initial-server-setup-with-ubuntu-18-04>

**Step 4:** create a folder for all your projects

```
mkdir [folder name]
```

navigate into folder

```
cd [folder name]
```

**Step 5:** clone your project into this folder.

```
sudo git clone [github path for project]
```

**Step 6:** navigate into the root of the project and install the necessary dependencies.

```
cd [project name]
```

```
sudo npm install
```

**Step 7:** grant admin privileges to project. This can be done from any directory.

```
sudo chmod -R a+rw [name of your project]
```

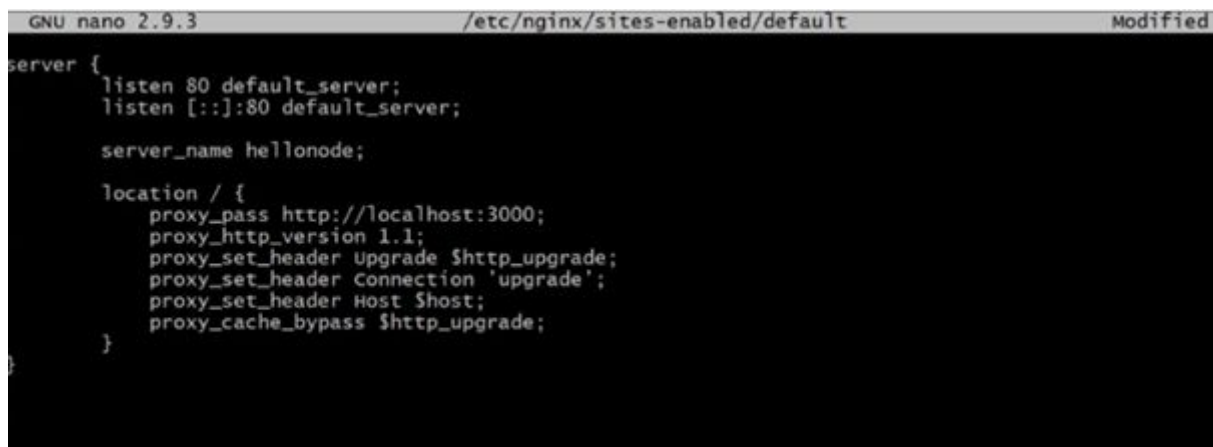
**Step 8:** Create .env file and paste in your mongodb connection string.

sudo nano .env

**Step 9:** Allow access to the site from the web/Set up server name

sudo nano /etc/nginx/sites-enabled/default.

Pictured below is a screenshot of the nano terminal.



```
GNU nano 2.9.3 /etc/nginx/sites-enabled/default Modified
server {
    listen 80 default_server;
    listen [::]:80 default_server;

    server_name hellonode;

    location / {
        proxy_pass http://localhost:3000;
        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;
    }
}
```

Replace “hellonode”, on line 5 with the name you created in step 2.

Replace the port on line 8 with the port you have selected in your .env file.

**Step 10:** Make sure nginx configuration is ok.

sudo nginx -t

**Step 11:** Restart nginx

sudo systemctl restart nginx

**Step 12:** navigate into the root of your project and start the server to check everything works.

cd [folder]/[your project name]

npm start

**Step 12:** Build your project.

Navigate into the root of your project.

npm run build.

**Step 13:** Start the process manager and name your pro

In the root of your project

```
pm2 start ./build/app.js --name [your desired name]
```

**Step 14:** Configure the process manager to restart the server on crashes or other events that would prevent the server from running.

```
pm2 startup
```

copy the line with the start up command

Pictured below is a screenshot of a video by Lars The Knowledge Giver aka Lars Mortensen.

```
sudo env PATH=$PATH:/usr/local/bin /usr/local/lib/node_modules/pm2/bin/pm2 startup systemd -u lam --hp /home/lam
```

Run the command.

**Step 15:** Reboot the droplet

```
sudo reboot
```

**Step 16:** Set up continuous integration

In the root of your project

```
pm2 start ./build/app.js --watch --ignore-watch="node_modules"
```

Replace app.js with the name of your file if relevant.

**Step 17:** Set up SSL

Go to certbot. <https://certbot.eff.org/lets-encrypt/ubuntuionic-nginx>

Follow the steps provided.

When presented with the option pictured below it is vital that you select option two (2) to ensure the security of your site.

```
Please choose whether or not to redirect HTTP traffic to HTTPS, removing HTTP access.
-----
1: No redirect - Make no further changes to the webserver configuration.
2: Redirect - Make all requests redirect to secure HTTPS access. Choose this for
new sites, or if you're confident your site works on HTTPS. You can undo this
change by editing your web server's configuration.
-----
Select the appropriate number [1-2] then [enter] (press 'c' to cancel): |
```

**Step 18:** Restart nginx. This can be done from any directory.

```
sudo systemctl restart nginx
```

Congratulations! You're all set!