Como fazer o build de aplicações Node em um Servidor LINUX!

Instalando o NGINX
apt install -y nginx

Instalação do UFW
apt install ufw

ufw app list
nano /etc/default/ufw

Permitindo conexões SSH e Liberando porta ssh
ufw allow ssh
ufw allow 22

Liberando o nginx em http e https
ufw allow 'Nginx HTTP'
ufw allow 'Nginx HTTPS'

Habilitando UFW
ufw enable
ufw status

Instalando NODE.JS
apt update

apt install –y nodejs npm

node -v && npm –v

ou

Agora vamos fazer as configurações do Node.js, para isto acesse o **Github do Node.js,** procure as instruções para instalação segundo seu SO. No meu caso é o Ubuntu então vou seguir os passos abaixo:

```
# Using Ubuntu
curl -sL https://deb.nodesource.com/setup_13.x | sudo -E bash -
sudo apt-get install -y nodejs

# Using Debian, as root
curl -sL https://deb.nodesource.com/setup_13.x | bash -
apt-get install -y nodejs
```

Configurando Aplicação
Adicione a aplicação NODE dentro do diretório /var/www. Para isto entre na pasta www utilizando o comando:
 # cd /var/www

Faça o clone da aplicação, utilizando:
 # git clone https://github.com/usuario/api-conect.git

(Caso você não esteja utilizando o git, crie um zip de sua aplicação e jogue dentro da pasta /var/www do servidor).
Feito isto, entre na pasta que você clonou, e faca a instalação das dependências:

#npm install

Utilizando banco Externo

Caso você esteja utilizando um banco externo, ele provavelmente já estará configurado. Então precisamos apenas fazer as configurações em nossa aplicação para que ele identifique os serviços.

Configurando Variáveis de ambiente no Backend

Estamos quase prontos para rodar nossa aplicação, mas antes precisamos fazer as configurações das variáveis de ambiente.
Sua aplicação provavelmente terá um arquivo **.env** ou **.env.example** para realizar o as configurações globais da aplicação.
obs: caso você possua um arquivo **.env.example** faca uma copia para um arquivo **.env** para que ele seja reconhecido pela aplicação node.
Abra o arquivo **.env** e realize as configurações que o arquivo pede, no meu caso, como tenho um banco relacional, o Redis e alguns URL para configurar, preciso realizar a configuração. Para isto abra o arquivo como digitando o comando:

Exibindo arquivos ocultos:
 # ls -la

Criando o arquivo .env e acresente a seguinte linha no arquivo:
DATABASE_URL=postgres://{db_username}:{db_password}@{host}:{port}/{db_name}
DATABASE_URL=postgres://postgres:root@localhost:5432/ecommerce

 # nano .env

Depois START sua aplicação:
 # npm start

Caso apresente o seguinte erro:
sh: 1: nodemon: Permisson denied

Rode o comando para instalar o nodemon globalmente:
 # npm install -g nodemon

Rode novamente o START:
 # npm start

Configurando PM2

Não podemos utilizar o Node para executar nossa aplicação pois ao finalizar a sessão SSH obtida pelo terminal nossa aplicação será parada. Então uma alternativa é utilizar um serviço que rode nossos arquivos em segundo plano.

Para instalar o **PM2** rode o comando:

```
# npm install -g pm2
```

O comando **-G** adiciona nossa dependência em modo global na aplicação, o que facilita na execução.

Agora vamos adicionar nosso arquivo a fila de execução do **PM2.** No meu caso eu possuo dois arquivos que precisam ser adicionados na fila, pois estou utilizando o Redis para processamento de filas de trabalho. Para isto eu rodo o comando:

```
# pm2 start server.js –name API Laser
```

Para garantir que seu serviço está rodando digite o comando:

```
# pm2 list
```

MICROSSERVIÇO

```
# pm2 list
# pm2 stop
# pm2 restart server
# pm2 delete
```

Painel de controle do terminal

O PM2 oferece uma maneira simples de monitorar o uso de recursos de seu aplicativo. Você pode monitorar a memória e a CPU de maneira fácil e direta de seu terminal com:

```
# pm2 monit
```

Adicionando PM2 na Daemon (na inicialização do sistema)

Caso nosso servidor sofra um restart, nossa aplicação precisa subir automaticamente, para isto precisamos adicionar o PM2 como **daemon**.

Rode o comando:

```
# pm2 startup systemd
```

Ele irá te retornar um comando para que você execute conforme seu sistema operacional.
Execute este comando.
Agora com o **PM2** configurado como **daemon** digite o comando abaixo para adicionar os arquivos que estão rodando para reinicialização automática:

```
# pm2 save
```

Caso deseje remover:

```
# pm2 unstartup systemd
```

Configurando o DNS

Quer testar sua aplicação por dentro do servidor? Você pode usar o utilitário cURL no no terminal, como abaixo.

 # curl http://localhost:4000/api/orders

Ele vai fazer um GET naquela URL informada e vai imprimir o HTTP response no console mesmo. Mas mesmo funcionando localmente, você deve ter notado que a sua aplicação ainda não é acessível pela Internet. Esse é o próximo passo.

Obs.: Faça as configurações no portal da REGISTRObr onde o domínio está registrado

Configurando Proxy Reverso

Uma boa prática a se fazer e configurar um proxy reverso em nossa aplicação para que a url do nosso **backend** fique: **IP:PORT.**

Instalação do NGINX

a Instalação do **NGINX** é bem simples pois não precisamos fazer nenhuma configuração a mais dentro da nossa aplicação, precisamos apenas rodar o comando:

```
# apt install nginx
```

Após a instalação do **NGINX** você pode testar se funcionou entrando no IP da sua aplicação. Ele irá te mostrar uma tela de boas-vindas ao **NGINX.**

Realizando proxy Reverso (1ª opção)

Algo que precisamos ter antes de realizar o proxy reverso é um domínio ou subdomínio para configurarmos as rodas.

Após isto navegue até as configurações gerais no **NGINX:**

```
# cd /etc/nginx
```

Você irá se deparar com varias pastas, mas a que nos importa é a **sites-available** e a **sites-enabled.**

Vamos entrar na pasta **/sites-available** e criar um arquivo como o comando:

```
# touch backend
```

Após o arquivo criado, você pode copiar as configurações que vou deixar abaixo e mudar apenas a porta que está rodando para a porta da sua aplicação e o server_name para o domínio:

```
server
{
        server_name seudominio.com www.seudominio.com;
        location / {
          proxy_pass http://127.0.0.1:4000;
          proxy_http_version 1.1;
          proxy_set_header Upgrade $http_upgrade;
```

```
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_cache_bypass $http_upgrade;
      }
    }
```

Feito isto, precisamos criar uma referência deste arquivo que acabamos de criar para a pasta /sites-enabled. Para isto na pasta /etc/nginx digite o comando PWD e ele te dará o caminho total da aplicação. feito isto pegue este caminho e digite o comando:

 # ln -s <caminho-da-aplicacao>/sites-available/backend <caminho-da-aplicacao>/sites-enabled

Pronto! agora reinicie o serviço do NGINX e seu proxy estará configurado!

 # services nginx restart

Realizando proxy Reverso (2ª opção)
Configurando o SSL

Para gerar e instalar um certificado emitido pela Let's Encrypt vamos utilizar um utilitário chamado Certbot. Use o comando abaixo para obtê-lo uma vez conectado via SSH no seu droplet, lembrando que ele é um Linux Ubuntu.

 # apt-get install certbot

Depois de instalado o Certbot, vamos instalar o plugin para Nginx, visando facilitar nosso trabalho de configuração.

 # apt-get install python3-certbot-nginx

A instalação automatizada vai procurar uma configuração de Nginx que referencie o domínio para o qual você irá instalar o certificado. Sendo assim, vamos abrir o arquivo de configuração do Nginx para alterar a variável server_name.

 # /etc/nginx/sites-available/default

Procure pela linha que diz:

server_name _;

E substitua por (colocando o seu domínio nas variações necessárias):

server_name seudominio.com www.seudominio.com;

A configuração em **/etc/nginx/sites-available/default** ficará assim:

server_name 192.168.250.200; ou **seudominio.com www.seudominio.com;**

```
        location / {
#       root /var/www/api-conect;
#       location /var/www/api-conect/server.js {
          # First attempt to serve request as file, then
```

```
        # as directory, then fall back to displaying a 404.
        ### try_files $uri $uri/ =404; #Estava exibindo o erro 404 na rota:
        "192.168.250.200/orders_db" ao comentar o err>

        proxy_pass http://localhost:4000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_cache_bypass $http_upgrade;
    }
```

Obs.: Se **o** *Nginx retornar 404 para API Nodejs comente a linha* ***try_files $uri $uri/ =404****; Depois do arquivo editado e salvo, reinicie o Ngnix.*

For mutiple proxies:

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

(...)

  # other proxy
  location /1/ {
    proxy_pass http://localhost:3001/;
  }

  # other proxy
  location /2/ {
    proxy_pass http://localhost:3002/;
  }

  location / {
    try_files $uri $uri/ =404;
  }
}
```

sudo systemctl reload nginx

Agora, para rodar o Certbot a fim de gerar e instalar o certificado no Nginx, rode o seguinte comando (incluí duas variações do domínio).

 # certbot --nginx -d seudominio.com -d www.seudominio.com

Durante a instalação, será solicitado o seu email e se concorda com os termos da Let’s Encrypt. tem uma pergunta se quer compartilhar o seu email com a fundação e a última, que é bem

importante, diz sobre redirecionamento de tráfego HTTP para HTTPS, o que eu recomendo que você aceite, para ninguém conseguir utilizar sua aplicação sem segurança na rede.

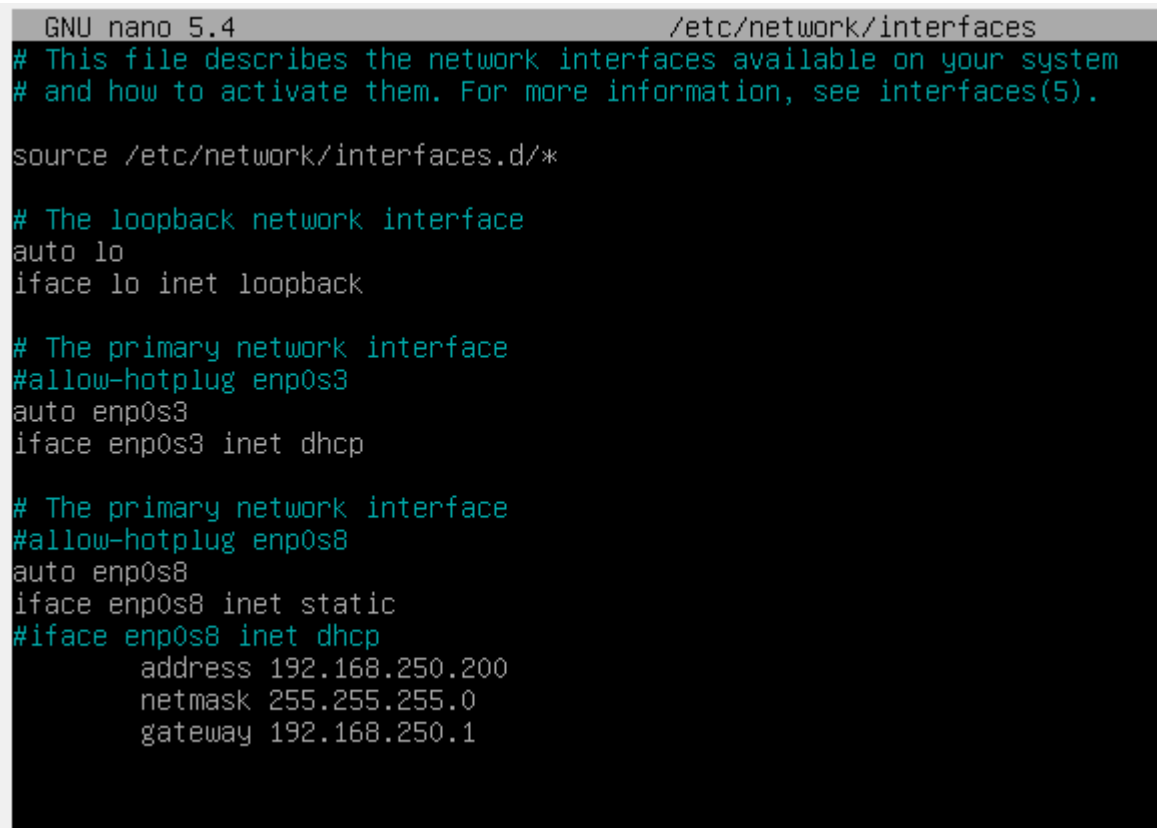
Problema de cacher:

https://coderedirect.com/questions/585310/how-to-setup-routes-with-express-and-nginx

https://notes.kindrazki.dev/como-fazer-o-build-de-aplica%C3%A7%C3%B5es-node-em-um-servidor-linux-c9cd862ff906
https://www.digitalocean.com/community/tutorials/how-to-install-node-js-on-debian-9-pt
https://www.digitalocean.com/community/tutorials/how-to-set-up-a-node-js-application-for-production-on-ubuntu-18-04-pt
https://www.youtube.com/watch?v=RmzH-Xq0DYA
https://pm2.keymetrics.io/
https://pm2.keymetrics.io/docs/usage/process-management/
https://www.digitalocean.com/community/tutorials/how-to-set-up-a-firewall-with-ufw-on-debian-10
https://www.luiztools.com.br/post/deploy-de-aplicacao-node-js-na-digital-ocean-2/
https://stackoverflow.com/questions/62638744/nginx-unexpectedly-returns-404-for-nodejs-api
https://github.com/gabrielwillemann/fast-help/blob/master/nginx/sites-avaible-examples.md#for-mutiple-proxies

VirtualBox: duas interfaces de rede, NAT(enp0s3) e somente host(enp0s3), em um Debian

Para fazer o teste de Deploy da API



https://qastack.com.br/unix/37122/virtualbox-two-network-interfaces-nat-and-host-only-ones-in-a-debian-guest-on