### ****Deploy a Node.js App on Ubuntu with SSL (Let's Encrypt) & Nginx (Port 3000)****

In this guide, we will:

1. Set up a **Node.js** application on **port 3000**.
2. Install and configure **Nginx** as a reverse proxy.
3. Secure the site with **Let's Encrypt SSL**.

## **Step 1: Install Node.js & PM2**

SSH into your Ubuntu server and install Node.js:

sudo apt update && sudo apt upgrade -y

curl -fsSL https://deb.nodesource.com/setup\_18.x | sudo -E bash -

sudo apt install -y nodejs

Verify installation:

node -v

npm -v

Install **PM2** to manage the Node.js process:

npm install -g pm2

## **Step 2: Create a Sample Node.js App**

Create a new project folder:

mkdir /var/www/myapp && cd /var/www/myapp

Create a simple server.js file:

const express = require("express");

const app = express();

app.get("/", (req, res) => {

res.send("Hello, Node.js with SSL!");

});

app.listen(3000, () => {

console.log("Server running on port 3000");

});

Install dependencies:

npm init -y

npm install express

Start the app using PM2:

pm2 start server.js --name myapp

pm2 startup

pm2 save

## **Step 3: Install & Configure Nginx as a Reverse Proxy**

Install Nginx:

sudo apt install nginx -y

Create a new Nginx configuration file:

sudo nano /etc/nginx/sites-available/myapp

Add the following content:

server {

listen 80;

server\_name example.com;

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;

}

}

Save and exit (CTRL+X, then Y, then ENTER).

Enable the configuration:

sudo ln -s /etc/nginx/sites-available/myapp /etc/nginx/sites-enabled/

sudo nginx -t

sudo systemctl restart nginx

## **Step 4: Secure with Let's Encrypt SSL**

Install **Certbot**:

sudo apt install certbot python3-certbot-nginx -y

Get an SSL certificate:

sudo certbot --nginx -d example.com -d www.example.com

This will:

* Automatically configure SSL for Nginx.
* Set up a **renewal cron job**.

To test renewal:

sudo certbot renew --dry-run

## **Step 5: Enable Firewall & Restart Services**

Allow necessary ports:

sudo ufw allow 'Nginx Full'

sudo ufw enable

Restart services:

sudo systemctl restart nginx

pm2 restart myapp

## **Step 6: Access Your App**

Visit your domain https://example.com. 🚀

Would you like to configure **auto-deployment** from GitHub as well? 😊

Your current Nginx configuration listens on **HTTP (port 80)** but does not force **HTTPS**. Below is the correct **HTTPS configuration** for your Node.js app running on **port 3000**.

**1. Correct Nginx Configuration with HTTPS**

Edit the Nginx configuration file:

sudo nano /etc/nginx/sites-available/myapp

Replace its content with:

server {

listen 80;

server\_name example.com www.example.com;

return 301 https://$host$request\_uri; # Redirect HTTP to HTTPS

}

server {

listen 443 ssl;

server\_name example.com www.example.com;

ssl\_certificate /etc/letsencrypt/live/example.com/fullchain.pem;

ssl\_certificate\_key /etc/letsencrypt/live/example.com/privkey.pem;

ssl\_protocols TLSv1.2 TLSv1.3;

ssl\_ciphers HIGH:!aNULL:!MD5;

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;

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

proxy\_set\_header X-Forwarded-Proto $scheme;

}

}

🔹 **Explanation**:

* The **first server block** redirects all HTTP traffic (port 80) to **HTTPS**.
* The **second server block** listens on **port 443** and applies SSL.
* Uses **Let’s Encrypt SSL certificate paths**.
* Properly **forwards headers** to the Node.js app.

**2. Apply the Configuration**

Save the file (CTRL+X, then Y, then ENTER).

Test Nginx:

sudo nginx -t

If no errors, restart Nginx:

sudo systemctl restart nginx

**3. Obtain & Renew SSL Certificate Automatically**

If you haven't obtained an SSL certificate yet, run:

sudo certbot --nginx -d example.com -d www.example.com

To check renewal:

sudo certbot renew --dry-run

This ensures automatic renewal every 90 days.

**4. Final Steps**

* **Allow HTTPS in Firewall**:
* sudo ufw allow 'Nginx Full'
* **Restart services**:
* sudo systemctl restart nginx
* pm2 restart myapp

Now, your app should be accessible at **https://example.com** with automatic SSL and HTTP redirection. 🚀

Would you like help with **auto-deploying updates from GitHub**? 😊

After running the command:

sudo certbot certonly --manual --preferred-challenges dns -d digi.cfaiteam.com -d api.cfadmin.cfaiteam.com -d auth.cfadmin.cfaiteam.com

Certbot will prompt you to add **TXT records** to your domain’s **DNS settings** for verification.

### ****Next Steps:****

#### **1. Add TXT Records in DNS**

Certbot will display verification challenges for each domain, like this:

Please deploy a DNS TXT record under the name:

\_acme-challenge.digi.cfaiteam.com

with the following value:

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

It will show similar records for:

* \_acme-challenge.api.cfadmin.cfaiteam.com
* \_acme-challenge.auth.cfadmin.cfaiteam.com

✅ **Go to your domain provider's DNS settings (e.g., Network Solutions, GoDaddy, Cloudflare, etc.)**

* **Create a TXT record** for each domain with the corresponding **values** Certbot provided.
* Example:
* Name: \_acme-challenge.digi.cfaiteam.com
* Type: TXT
* Value: XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

#### **2. Wait for DNS Propagation**

* It may take **a few minutes to hours** for DNS changes to propagate.
* You can check if the TXT records are set using:
* nslookup -q=TXT \_acme-challenge.digi.cfaiteam.com

#### **3. Continue Certbot Process**

Once you’ve added the TXT records, go back to your terminal **(Certbot will still be waiting)**.  
Press **ENTER** to let Certbot verify the records.

If successful, you’ll see:

Congratulations! Your certificate has been saved at:

/etc/letsencrypt/live/digi.cfaiteam.com/fullchain.pem

And similar paths for the other domains.

#### **4. Configure Nginx for SSL**

Now, update your **Nginx configuration** to use the SSL certificates.

Edit your site's Nginx config:

sudo nano /etc/nginx/sites-available/digi.cfaiteam.com

Replace it with:

server {

listen 80;

server\_name digi.cfaiteam.com api.cfadmin.cfaiteam.com auth.cfadmin.cfaiteam.com;

return 301 https://$host$request\_uri; # Redirect HTTP to HTTPS

}

server {

listen 443 ssl;

server\_name digi.cfaiteam.com api.cfadmin.cfaiteam.com auth.cfadmin.cfaiteam.com;

ssl\_certificate /etc/letsencrypt/live/digi.cfaiteam.com/fullchain.pem;

ssl\_certificate\_key /etc/letsencrypt/live/digi.cfaiteam.com/privkey.pem;

ssl\_protocols TLSv1.2 TLSv1.3;

ssl\_ciphers HIGH:!aNULL:!MD5;

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;

proxy\_set\_header X-Real-IP $remote\_addr;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

proxy\_set\_header X-Forwarded-Proto $scheme;

}

}

Save & exit (CTRL+X, then Y, then ENTER).

#### **5. Restart Nginx**

sudo nginx -t # Check for errors

sudo systemctl restart nginx

#### **6. Auto-Renewal Setup**

Since you used **manual DNS verification**, automatic renewal is **not enabled by default**. You must renew it manually every 3 months:

sudo certbot certonly --manual --preferred-challenges dns -d digi.cfaiteam.com -d api.cfadmin.cfaiteam.com -d auth.cfadmin.cfaiteam.com

### 🎉 ****Final Test****

Visit **https://digi.cfaiteam.com**, **https://api.cfadmin.cfaiteam.com**, and **https://auth.cfadmin.cfaiteam.com** in your browser to verify SSL is working.

Would you like an **auto-renewal script** for DNS challenges? 😊

To renew an SSL certificate created using the ACME challenge (e.g., Let's Encrypt with Certbot) on an Ubuntu server, follow these steps:

**Step 1: Check Current Certificate Expiry**

Run the following command to check when your SSL certificate expires:

sudo openssl x509 -noout -dates -in /etc/letsencrypt/live/yourdomain.com/fullchain.pem

**Step 2: Manually Renew the Certificate**

Run the renewal command:

sudo certbot renew --preferred-challenges dns

If you used **HTTP challenge**, run:

sudo certbot renew --preferred-challenges http

**Step 3: Restart Web Server**

After renewal, restart your web server to apply changes:

sudo systemctl restart nginx  # For Nginx  
# OR  
sudo systemctl restart apache2  # For Apache

**Step 4: Automate Renewal (Optional)**

Let's Encrypt certificates are valid for 90 days, so you can automate renewal using a cron job:

sudo crontab -e

Add this line at the end:

0 3 \* \* \* /usr/bin/certbot renew --quiet && systemctl restart nginx

This will attempt renewal every day at 3 AM and restart Nginx.

**Step 5: Verify the New Certificate**

After renewal, check the new expiry date:

sudo openssl x509 -noout -dates -in /etc/letsencrypt/live/yourdomain.com/fullchain.pem

Let me know if you need further assistance! 🚀