Installation Guide

- Installation Guide
  - Installation Guide for SSH, SCP, and FTP on Windows and macOS
  - Windows 11 Guide (As of September 2025)
    - Step 1: Enable Built-in SSH and SCP (CLI)
      - Step 2: Set Up FTP (CLI with Built-in curl)
    - Step 3: Install GUI Tools (Recommended for Beginners)
      - Install WinSCP (for SSH/SCP GUI)
      - Install FileZilla (for FTP GUI)
    - Testing on Windows
  - macOS Guide (Sequoia 15 or Later, As of September 2025)
    - Step 1: Access Built-in SSH and SCP (CLI)
    - Step 2: Set Up FTP (CLI with Built-in curl)
    - Step 3: Install GUI Tools (Recommended for Beginners)
      - Install Cyberduck (for SSH/SCP GUI)
      - Install FileZilla (for FTP GUI)
    - Testing on macOS
- Installing NGINX, Configuring a Self-Signed SSL Certificate, and Testing on Ubuntu
  - Basic Linux Command Line
    - Navigation and File Operations
    - File Content Operations
    - System and Process Management
    - Permissions and Ownership
    - Network and System Status
    - Tips for Beginners
  - Using Nano Text Editor
  - Part 1: Installing NGINX
    - Step 1: Update Package Index
    - Step 2: Install NGINX
    - Step 3: Verify and Start NGINX
    - Step 4: Test Basic Installation
  - Part 2: Installing OpenSSL (for Certificates)
  - Part 3: Generating a Self-Signed SSL Certificate
    - Step 1: Create SSL Directory
    - Step 2: Generate Certificate
  - Part 4: Configuring NGINX for HTTPS
    - Step 1: Edit Configuration
    - Step 2: Test and Reload
    - Step 3: Firewall (if using UFW)
  - Part 5: Creating Test Content
  - Part 6: Testing the Setup
    - Step 1: Check Listening Ports
    - Step 2: Test SSL with OpenSSL
    - Step 3: Browser Test

    - Part 7: Troubleshooting
        - Logs
    - Part 8: Security and Best Practices
    - Conclusion
- Installing NVM and Node.js on Ubuntu
    - Step 1: Install Dependencies
    - Step 2: Install NVM
    - Step 3: Install Node.js Using NVM
    - Step 4: Install a Sample Node.js App (For Testing)
    - Step 5: Troubleshooting NVM/Node.js
- Configuring NGINX as a Reverse Proxy on Ubuntu
    - Understanding NGINX Config Files
    - Part 1: Basic Reverse Proxy Setup (HTTP on Port 80)
    - Part 2: Secure Reverse Proxy (HTTPS on Port 443 with HTTP Redirect)
    - Part 3: Advanced Options and Troubleshooting
    - Conclusion

Installation Guide for SSH, SCP, and FTP on Windows and macOS

We'll cover **both command-line (CLI)** options (text-based, no mouse needed after setup) and **graphical user interface (GUI)** options (point-and-click apps).

Prerequisites:

- A stable internet connection.
- Administrator access on your computer (you'll be prompted for your password).
- We'll use free, official tools only.

Safety Note: Only use these for legitimate purposes (e.g., accessing your own servers). Never share credentials.

Windows 11 Guide (As of September 2025)

Windows 11 has some tools built-in, but we need to enable them. If you're on an older version like Windows 10, the steps are nearly identical.

Step 1: Enable Built-in SSH and SCP (CLI)

SSH and SCP are bundled together in "OpenSSH Client," a free tool from Microsoft that's already on your PC but disabled by default.

1. **Open Settings**:
    - Click the **Start button** (Windows icon in the bottom-left corner of your screen). It looks like a Windows logo.
    - Type "Settings" (without quotes) in the search bar that appears.
    - Click the **Settings** app (gear icon) that shows up.
2. **Navigate to Optional Features**:

   - In the Settings window, click **Apps** on the left sidebar (it has a colorful square icon).
   - Click **Optional features** (under "Apps & features").

3. **Add OpenSSH Client**:
   - Click the **View features** button (top-right).
   - In the search box at the top, type "OpenSSH Client".
   - Check the box next to **OpenSSH Client** (it should appear in the list).
   - Click **Next** at the bottom.
   - Click **Install** (it may take 1-2 minutes; your PC might ask for your password—enter it).

4. **Verify Installation**:
   - Press the **Windows key** (on your keyboard, between Ctrl and Alt) + **R** to open the Run dialog.
   - Type `powershell` and press Enter. A blue window (PowerShell) opens—this is your command line.
   - Type `ssh` and press Enter.
   - If it says something like "usage: ssh [-46AaCfGgKkMNnqsTtVvXxYy]" or "command not found" doesn't appear, it's installed! (Ignore the error about no arguments; that's normal.)
   - Close PowerShell by typing `exit` and pressing Enter.

Troubleshooting: If it fails, restart your PC and retry. If still stuck, search "OpenSSH Windows 11" in your browser for video help.

Step 2: Set Up FTP (CLI with Built-in curl)

curl is a simple command-line tool for FTP (and more) that's already built into Windows 11—no install needed.

1. **Verify curl**:
   - Open PowerShell again (Windows key + R, type `powershell`, Enter).
   - Type `curl --version` and press Enter.
   - It should show version info (e.g., "curl 8.x"). If not, update Windows: Go to Settings > Windows Update > Check for updates.

2. **Basic Usage Example** (Test Later):
   - To download a file via FTP: `curl -u username:password ftp://server.com/path/file.txt -o downloadedfile.txt`
   - We'll test in the "Testing" section below. (Replace placeholders with real details from your server.)

Step 3: Install GUI Tools (Recommended for Beginners)

For easier point-and-click use:

- **WinSCP** for SSH/SCP (free, secure file manager).
- **FileZilla** for FTP (free, supports FTP and secure variants).

Install WinSCP (for SSH/SCP GUI)

1. **Download**:
    - Open your web browser (e.g., Edge or Chrome—click its icon on the taskbar).
    - Go to `https://winscp.net` (type it in the address bar and press Enter).
    - Click the big **Download WinSCP** button (green or blue).
2. **Install**:
    - The file (e.g., "WinSCP-6.x-Setup.exe") downloads to your Downloads folder.
    - Open File Explorer (folder icon on taskbar).
    - Click **Downloads** on the left.
    - Double-click the WinSCP file.
    - If prompted by User Account Control (popup), click **Yes** and enter your password.
    - In the installer: Click **Next** > Accept license > **Typical** installation > **Install** > **Finish**.
3. **Launch and Basic Use**:
    - Search for "WinSCP" in Start menu and open it.
    - It shows login screens—enter server details (host, username, password) for SSH/SCP connections.
    - Drag files between panels for transfers.

Install FileZilla (for FTP GUI)

1. **Download**:
    - In your browser, go to `https://filezilla-project.org`.
    - Click **Download FileZilla Client** (not Server).
2. **Install**:
    - Downloads to Downloads folder—open File Explorer > Downloads.
    - Double-click the .exe file (e.g., "FileZilla_3.x_win64-setup.exe").
    - Click **Yes** on UAC popup.
    - Installer: **I Agree** > **Next** > Install components (default) > **Next** > **Install** > **Finish** (uncheck "View README" if you want).
3. **Launch and Basic Use**:
    - Search "FileZilla" in Start and open.
    - Top fields: Host (server address), Username, Password, Port (21 for FTP).
    - Click **Quickconnect**—files appear in bottom panels for drag-and-drop.

Troubleshooting: If antivirus blocks download, allow it temporarily. Restart if needed.

Testing on Windows

1. **SSH/SCP Test** (CLI): In PowerShell, type `ssh user@yourserver.com` (replace with real details). It should prompt for password. For SCP: `scp localfile.txt user@server:/path/`.
2. **FTP Test** (CLI): `curl ftp://example.com/test.txt -o test.txt` (use a public FTP like test.com if available).
3. **GUI Test**: Open WinSCP/FileZilla, connect to a test server (e.g., demo.test.rebex.net for SFTP).

---

macOS Guide (Sequoia 15 or Later, As of September 2025)

macOS (your Mac's operating system) has SSH and SCP built-in forever—no install needed! We'll just show you how to access them. For FTP, curl is also built-in.

Step 1: Access Built-in SSH and SCP (CLI)

1. **Open Terminal** (Your Command Line):
    - Click the **Apple menu** (icon, top-left corner).
    - Hover over **System Settings** (or System Preferences on older macOS).
    - No—wait, for Terminal: Press **Command (⌘) + Space** to open Spotlight search.
    - Type "Terminal" and press Enter. A black window opens.
2. **Verify Installation**:
    - In Terminal, type `ssh` and press Enter.
    - It shows usage info (like "OpenSSH_9.x"). Success!
    - For SCP: Type `scp`—same thing.

Note: If you're on macOS Sequoia (15), SSH works fine, but firewall might block incoming connections—check System Settings > Network > Firewall > Options, and allow "sshd-session" if needed.

Step 2: Set Up FTP (CLI with Built-in curl)

curl is pre-installed.

1. **Verify curl**:
    - In Terminal, type `curl --version` and press Enter.
    - Shows version (e.g., "curl 8.x").
2. **Basic Usage Example**:
    - Download: `curl -u username:password ftp://server.com/path/file.txt -o downloadedfile.txt`

Step 3: Install GUI Tools (Recommended for Beginners)

- **Cyberduck** for SSH/SCP (free, simple drag-and-drop). Or use built-in Finder.
- **FileZilla** for FTP.

Install Cyberduck (for SSH/SCP GUI)

1. **Download**:
    - Open Safari (or your browser—blue compass icon).
    - Go to `https://cyberduck.io`.
    - Click **Download** (for macOS).

2. **Install**:
    - Downloads a .zip or .dmg to Downloads (open Finder > Downloads).
    - Double-click the .dmg file.
    - Drag the **Cyberduck** icon to your **Applications** folder (in the dmg window).
    - Eject the dmg (right-click > Eject).
    - Open Applications (in Finder sidebar), double-click Cyberduck.
3. **Basic Use**:
    - Click **Open Connection**.
    - Choose SFTP (for SSH/SCP), enter server, username, password.
    - Bookmark for reuse; drag files to transfer.

(Alternative: macOS Finder supports SSH—Go > Connect to Server > `sftp://server.com`.)

Install FileZilla (for FTP GUI)

1. **Download**:
    - In browser, go to `https://filezilla-project.org`.
    - Click **Download FileZilla Client**.
2. **Install**:
    - Downloads .dmg to Downloads.
    - Double-click .dmg.
    - Drag **FileZilla** icon to **Applications** folder.
    - Eject dmg.
3. **Launch and Basic Use**:
    - In Applications or Spotlight, open FileZilla.
    - Enter Host, Username, Password, Port 21.
    - **Quickconnect**—drag files between local/remote panes.

Troubleshooting: If Gatekeeper blocks, right-click app > Open. Update macOS if prompted.

Testing on macOS

1. **SSH/SCP Test** (CLI): In Terminal, `ssh user@yourserver.com`. For SCP: `scp localfile.txt user@server:/path/`.
2. **FTP Test** (CLI): `curl ftp://example.com/test.txt -o test.txt`.
3. **GUI Test**: Open Cyberduck/FileZilla, connect to demo server (e.g., test.rebex.net).

Installing NGINX, Configuring a Self-Signed SSL Certificate, and Testing on Ubuntu

Installation Guide on a server like `project-1-12.eduhk.hk`. Your project domain should be `project-1-XX.eduhk.hk`

Basic Linux Command Line

Before starting with NGINX installation, here are essential Linux commands you'll need:

Navigation and File Operations

- **Show current directory**: `pwd` (print working directory)
- **List files and folders**: `ls` (basic) or `ls -la` (detailed with hidden files)
- **Change directory**: `cd /path/to/directory` or `cd ~` (home directory)
- **Create directory**: `mkdir directory_name`
- **Remove files**: `rm filename` or `rm -rf directory_name` (recursive/force)
- **Copy files**: `cp source destination`
- **Move/rename**: `mv old_name new_name`

File Content Operations

- **View file content**: `cat filename` (entire file) or `less filename` (paginated)
- **Edit files**: `nano filename` (beginner-friendly) or `vim filename` (advanced)
- **Create/edit file**: `touch filename` (create empty) or `echo "content" > filename`
- **Search in files**: `grep "search_term" filename`

System and Process Management

- **Check running processes**: `ps aux` or `top` (interactive)
- **Kill process**: `kill process_id` or `killall process_name`
- **Check disk usage**: `df -h` (disk space) or `du -sh directory` (directory size)
- **Check system info**: `uname -a` (system info) or `whoami` (current user)

Permissions and Ownership

- **Change permissions**: `chmod 755 filename` (read/write/execute permissions)
- **Change ownership**: `chown user:group filename`
- **Run as administrator**: `sudo command` (requires admin privileges)

Network and System Status

- **Check network**: `ping google.com` or `wget http://example.com`
- **Check listening ports**: `netstat -tuln` or `ss -tuln`
- **Check system services**: `systemctl status service_name`

Tips for Beginners

- Use **Tab** to auto-complete commands and file names
- Use **Up/Down arrows** to navigate command history
- Use `Ctrl + C` to cancel/interrupt running commands
- Use `man command_name` to view manual/help for any command

- Always be careful with `rm` and `sudo` commands

Using Nano Text Editor

`nano` is a simple terminal-based editor used for editing NGINX configs and files. Key shortcuts (especially for the ones mentioned):

- **Open a file**: `sudo nano /path/to/file` (e.g., `sudo nano /etc/nginx/sites-available/default`).
- **Remove all lines (clear content)**: Press `Ctrl + K` repeatedly to cut (delete) lines one by one, or hold it to clear faster.
- **Paste content**: Right-click in the terminal to paste (or use `Ctrl + U` to uncut/paste previously cut text).
- **Save and exit**: Press `Ctrl + X`, then `Y` (yes) to confirm saving changes.
- **Other useful shortcuts**:
  - `Ctrl + O`: Save without exiting.
  - `Ctrl + W`: Search.
  - `Ctrl + G`: View all shortcuts.

Always test configs after editing (e.g., `sudo nginx -t`).

Part 1: Installing NGINX

Step 1: Update Package Index

```
sudo apt update
```

Step 2: Install NGINX

```
sudo apt install nginx
```

Step 3: Verify and Start NGINX

```
sudo systemctl status nginx
sudo systemctl start nginx  # If not running
sudo systemctl enable nginx  # Start on boot
```

Step 4: Test Basic Installation

Find your server IP:

```
ip addr show | grep inet  # Shows all IPs; look for e.g., 192.168.56.182 under
ens33
# Or shorter: hostname -I
# Public IP (if needed): curl ifconfig.me
```

Visit `http://project-1-12.eduhk.hk` (e.g., `http://192.168.56.182`) in a browser. You should see the NGINX welcome page.

Part 2: Installing OpenSSL (for Certificates)

```
sudo apt install openssl
```

Part 3: Generating a Self-Signed SSL Certificate

Step 1: Create SSL Directory

```
sudo mkdir /etc/nginx/ssl
```

Step 2: Generate Certificate

```
sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout
/etc/nginx/ssl/nginx.key -out /etc/nginx/ssl/nginx.crt
```

During prompts (Distinguished Name fields):

Important:

- **Common Name (e.g., server FQDN) []**: `project-1-12.eduhk.hk` (use your domain/FQDN; verify with `ping project-1-12.eduhk.hk`).

This creates a 1-year valid, 2048-bit RSA self-signed cert. Secure files:

```
sudo chmod 600 /etc/nginx/ssl/nginx.key
sudo chmod 644 /etc/nginx/ssl/nginx.crt
```

Part 4: Configuring NGINX for HTTPS

Step 1: Edit Configuration

```
sudo nano /etc/nginx/sites-available/default
```

Use `Ctrl + K` to clear existing content if needed, paste the new config (right-click to paste), then `Ctrl + X` > `Y` to save/exit.

Replace with:

```
server {
    listen 80;
    listen [::]:80;
    server_name project-1-12.eduhk.hk;

    # Redirect HTTP to HTTPS
    return 301 https://$host$request_uri;
}

server {
    listen 443 ssl;
    listen [::]:443 ssl;
    server_name project-1-12.eduhk.hk;

    ssl_certificate /etc/nginx/ssl/nginx.crt;
    ssl_certificate_key /etc/nginx/ssl/nginx.key;

    root /var/www/html;  # Default root; not /var/www/root
    index index.html index.htm index.nginx-debian.html;

    location / {
        try_files $uri $uri/ /index.html;
    }
}
```

- Replace `project-1-12.eduhk.hk` with your domain/IP if needed.
- Root is `/var/www/html` by default (verify: `cat /etc/nginx/sites-available/default | grep root`).

Step 2: Test and Reload

```
sudo nginx -t  # Check syntax
sudo systemctl reload nginx
```

Step 3: Firewall (if using UFW)

```
sudo ufw allow 'Nginx Full'  # Allows 80 and 443
sudo ufw status
```

Part 5: Creating Test Content

Default root: `/var/www/html` (not `/var/www/root`—that's non-standard).

```
ls -l /var/www/html  # Check contents
echo "<h1>Hello from project-1-12.eduhk.hk with HTTPS!</h1>" | sudo tee
```

```
/var/www/html/index.html
sudo chown www-data:www-data /var/www/html/index.html
sudo systemctl reload nginx
```

Part 6: Testing the Setup

Step 1: Check Listening Ports

Install `netstat` if missing:

```
sudo apt install net-tools
```

Then:

```
sudo netstat -tuln | grep ':80\|:443'
# Alternative (no install): sudo ss -tuln | grep ':80\|:443'
```

Expected:

```
tcp 0 0 0.0.0.0:80  0.0.0.0:* LISTEN
tcp 0 0 0.0.0.0:443 0.0.0.0:* LISTEN
tcp6 0 0 :::80 :::* LISTEN
tcp6 0 0 :::443 :::* LISTEN
```

Step 2: Test SSL with OpenSSL

Basic connection:

```
openssl s_client -connect 192.168.56.182:443  # Use your IP
```

Full HTTP test:

```
echo -e "GET / HTTP/1.1\r\nHost: project-1-12.eduhk.hk\r\nConnection:
close\r\n\r\n" | openssl s_client -connect 192.168.56.182:443 -servername project-
1-12.eduhk.hk
```

Expected: `HTTP/1.1 200 OK` with your `<h1>` content. Self-signed warning (`verify error:num=18`) is normal.

Step 3: Browser Test

- Visit `https://project-1-12.eduhk.hk` or `https://192.168.56.182`.

- Accept "not private" warning (self-signed).
- Should show your test page.

Part 7: Troubleshooting

Issue	Possible Cause	Solution
`netstat: command not found`	Not installed	`sudo apt install net-tools` or use `ss`.
No HTTP response in OpenSSL	Empty `/var/www/html` or bad config	Create `index.html`; check `sudo nginx -t`.
400 Bad Request	Incomplete request in basic `openssl s_client`	Use full GET command; normal for partial requests.
Port not listening	Config error or NGINX down	`sudo systemctl status nginx`; reload.
Browser mismatch warning	CN doesn't match access method	Use domain in browser; regenerate cert if using IP.
Certificate fields wrong (e.g., C=AU)	Default prompts	Regenerate with correct inputs (e.g., HK).

Logs

```
sudo tail -f /var/log/nginx/error.log  # Errors
sudo tail -f /var/log/nginx/access.log  # Access
sudo journalctl -u nginx  # System logs
```

Part 8: Security and Best Practices

- **Self-Signed Limits**: Warnings in browsers; use Let's Encrypt for production.
- **Updates**: `sudo apt update && sudo apt upgrade`.
- **Backup**: Copy `/etc/nginx/ssl/*` securely.
- **Trust Cert Locally** (testing): `sudo cp /etc/nginx/ssl/nginx.crt /usr/local/share/ca-certificates/nginx.crt && sudo update-ca-certificates`.

Conclusion

This covers installing NGINX, SSL setup, testing, and tools like `nano` and `netstat`. Your setup on `project-1-12.eduhk.hk` (IP: 192.168.56.182) should now serve HTTPS content. For advanced topics, see NGINX Docs.

Installing NVM and Node.js on Ubuntu

This section assumes you are continuing on your Ubuntu server (e.g., `project-1-12.eduhk.hk`) after setting up NGINX and SSL as described earlier. NVM (Node Version Manager) allows you to install and manage multiple versions of Node.js easily. Node.js is a JavaScript runtime for building server-side applications, often used for web apps that listen on ports like 3000.

Prerequisites:

- You have SSH access to your Ubuntu server (as set up in the earlier SSH guide).
- Run all commands as a non-root user with sudo privileges (or switch to root if preferred, but NVM works best with a regular user).
- A stable internet connection on the server.

Safety Note: Only install from official sources. Keep your server updated with `sudo apt update && sudo apt upgrade`.

Step 1: Install Dependencies

Node.js and NVM require some basic packages. Install them if not already present:

```
sudo apt update
sudo apt install curl build-essential -y
```

- `curl`: For downloading the NVM installer.
- `build-essential`: Includes compilers needed for native Node.js modules.

Step 2: Install NVM

NVM is installed per-user. Run the following as your regular user (not sudo):

```
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.40.0/install.sh | bash
```

- This downloads and runs the latest NVM installer script (as of September 2025; check nvm-sh/nvm on GitHub for the exact version if needed).
- It adds NVM to your `~/.bashrc` or `~/.zshrc` (depending on your shell).

Close and reopen your terminal (or SSH session), or source the profile:

```
source ~/.bashrc  # Or ~/.zshrc if using Zsh
```

Verify NVM:

```
nvm --version
```

It should output something like "0.40.0".

Troubleshooting: If "command not found," ensure the source command ran correctly or add `export NVM_DIR="$HOME/.nvm"` and `[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh"` to your `~/.bashrc` manually with `nano ~/.bashrc`, then source it.

Step 3: Install Node.js Using NVM

Install the latest LTS (Long-Term Support) version of Node.js (recommended for stability; as of September 2025, this is likely Node 22.x or higher—check with `nvm ls-remote` for options).

```
nvm install --lts
```

- This installs Node.js and npm (Node Package Manager).
- To install a specific version: `nvm install 20` (e.g., for Node 20.x).

Set it as default:

```
nvm use --lts
nvm alias default lts
```

Verify:

```
node --version  # e.g., v22.9.0
npm --version   # e.g., 10.8.3
```

Step 4: Install a Sample Node.js App (For Testing)

To test, create a simple Express.js app that listens on port 3000.

1. Create a project directory:

```
mkdir ~/my-node-app
cd ~/my-node-app
```

2. Initialize and install Express:

```
npm init -y
npm install express
```

3. Create `app.js` with Nano:

```
nano app.js
```

Paste this content:

```
const express = require('express');
const app = express();
const port = 3000;

app.get('/', (req, res) => {
  res.send('<h1>Hello from Node.js on Ubuntu!</h1>');
});

app.listen(port, () => {
  console.log(`App running on http://localhost:${port}`);
});
```

Save and exit (`Ctrl + X`, `Y`).

4. Run the app (for testing; use a process manager like PM2 in production):

```
node app.js
```

- Visit `http://project-1-12.eduhk.hk:3000` in a browser (replace with your domain/IP). If NGINX isn't proxying yet, it should show the hello message (allow port 3000 in firewall if needed: `sudo ufw allow 3000`).

Stop with `Ctrl + C`. For production, install PM2:

```
npm install -g pm2
pm2 start app.js
pm2 startup  # For auto-start on boot
```

Step 5: Troubleshooting NVM/Node.js

Issue	Possible Cause	Solution
`nvm: command not found`	Profile not sourced	Run `source ~/.bashrc` or restart session.
Installation fails	No internet or missing deps	Check connection; rerun `sudo apt install curl build-essential`.
Port in use	Another app on 3000	`sudo ss -tuln | grep 3000`; kill process if needed.
npm permissions	Global install issues	Avoid sudo with npm; use NVM's user install.

Logs: Check `pm2 logs` or console output.

Best Practices: Keep Node.js updated (`nvm install --lts --reinstall-packages-from=current`). Use environment variables for configs.

Configuring NGINX as a Reverse Proxy on Ubuntu

A reverse proxy forwards client requests to a backend server (e.g., your Node.js app on port 3000) and returns responses. This allows NGINX to handle traffic on ports 80 (HTTP) and 443 (HTTPS), improving security, load balancing, and performance. We'll proxy to `localhost:3000` (assuming your Node.js app runs there).

Prerequisites:

- NGINX installed and basic HTTPS configured (from earlier sections).
- Your Node.js app running on port 3000 (e.g., via PM2).
- Understand: Clients connect to NGINX (80/443) → NGINX proxies to app (3000) → Response back.

Understanding NGINX Config Files

NGINX configs are modular for easy management:

- **/etc/nginx/nginx.conf**: The main global config file. It includes directives for worker processes, events, and HTTP settings. Importantly, it includes all files from `/etc/nginx/sites-enabled/*` (via `include /etc/nginx/sites-enabled/*;` in the `http` block). Do not edit this directly unless tweaking global settings—focus on site-specific files.

- **/etc/nginx/sites-available/**: Directory for site configuration files. Create or edit files here (e.g., `default` or `my-site`). These are not active until linked.

- **/etc/nginx/sites-enabled/**: Directory with symbolic links (symlinks) to files in `sites-available`. NGINX only loads configs from here. To enable a site: `sudo ln -s /etc/nginx/sites-available/my-site /etc/nginx/sites-enabled/`. To disable: Remove the symlink (`sudo rm /etc/nginx/sites-enabled/my-site`).

Relationship:

- `nginx.conf` → includes `sites-enabled/*` → symlinks to `sites-available/*`.
- This setup allows you to create/test configs in `sites-available` without affecting live sites, then enable them via symlinks.
- After changes: Always test with `sudo nginx -t`, then reload `sudo systemctl reload nginx`.
- Other files: `/etc/nginx/conf.d/` for additional snippets (not used here).

Tips: Use `sudo nano` for editing. Backup originals: `sudo cp /etc/nginx/sites-available/default /etc/nginx/sites-available/default.bak`.

Part 1: Basic Reverse Proxy Setup (HTTP on Port 80)

We'll modify the existing `default` config to proxy to port 3000.

1. Edit the config:

```
sudo nano /etc/nginx/sites-available/default
```

Clear existing content if needed (Ctrl + K repeatedly), and replace with:

```
server {
    listen 80;
    listen [::]:80;
    server_name project-1-12.eduhk.hk;  # Replace with your domain/IP

    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_pass`: Forwards requests to your app.
- Headers: Enable WebSockets if needed (e.g., for real-time apps).
- Save and exit.

2. Test and reload:

```
sudo nginx -t
sudo systemctl reload nginx
```

3. Ensure symlink (if not already):

```
sudo ln -s /etc/nginx/sites-available/default /etc/nginx/sites-enabled/
sudo nginx -t
sudo systemctl reload nginx
```

4. Test: Visit `http://project-1-12.eduhk.hk`—it should show your Node.js app's content (e.g., "

   # Hello from Node.js

   "). Check logs if issues: `sudo tail -f /var/log/nginx/error.log`.

Part 2: Secure Reverse Proxy (HTTPS on Port 443 with HTTP Redirect)

Extend for HTTPS using your self-signed cert.

1. Edit the config again:

```
sudo nano /etc/nginx/sites-available/default
```

Replace with full config:

```
server {
    listen 80;
    listen [::]:80;
    server_name project-1-12.eduhk.hk;

    # Redirect all HTTP to HTTPS
    return 301 https://$server_name$request_uri;
}

server {
    listen 443 ssl;
    listen [::]:443 ssl;
    server_name project-1-12.eduhk.hk;

    ssl_certificate /etc/nginx/ssl/nginx.crt;      # From earlier setup
    ssl_certificate_key /etc/nginx/ssl/nginx.key;

    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;
    }
}
```

- This redirects HTTP to HTTPS and proxies HTTPS to 3000.
- Save, test (`sudo nginx -t`), reload (`sudo systemctl reload nginx`).

2. Firewall: Ensure ports 80/443 are open:

```
sudo ufw allow 'Nginx Full'
sudo ufw status
```

3. Test: Visit `https://project-1-12.eduhk.hk`—accept self-signed warning, see app content. HTTP should redirect automatically.

Part 3: Advanced Options and Troubleshooting

- **Multiple Sites**: Create new files in `sites-available` (e.g., `sudo nano /etc/nginx/sites-available/my-app`), add server blocks, then symlink to `sites-enabled`.
- **Custom Root/Static Files**: If serving static files alongside proxy, add `root /var/www/html;` and `try_files $uri @proxy;` with `location @proxy { proxy_pass ...; }`.
- **Logging**: Add `access_log /var/log/nginx/my-app.access.log;` and `error_log /var/log/nginx/my-app.error.log;` inside server blocks.

Issue	Possible Cause	Solution
502 Bad Gateway	App not running on 3000	Check `pm2 status` or `sudo ss -tuln | grep 3000`; restart app.
Config syntax error	Typo in file	`sudo nginx -t` shows line; fix and reload.
No redirect	Missing return 301	Verify HTTP server block.
Cert issues	Path wrong	Check `ssl_certificate` paths; regenerate if needed.
Symlink missing	File not enabled	`ls /etc/nginx/sites-enabled/`; add symlink.

Best Practices: Use Let's Encrypt for real certs (`sudo apt install certbot python3-certbot-nginx; sudo certbot --nginx`). Monitor with `sudo systemctl status nginx`. For production, add rate limiting and security headers in `nginx.conf`.

Conclusion

Your NGINX is now proxying to your Node.js app on port 3000, with HTTPS enabled. This setup uses the modular `sites-available` and `sites-enabled` structure for easy management. For more, see NGINX Proxy Docs.