

Home > Blogs

# Host a MERN Stack App on a VPS

By Binamra Lamsal - 2 years ago

Web Development

Share post on:



HOSTINGER

## Make sure these steps before proceeding

- Before pushing to github, do not forget to use environment variables where it's needed. Like for frontend, you might need to setup environment variable for API\_URL. Like for backend, you might need to setup environment variable for database url.
- Make sure to include .gitignore with node\_modules inside it so that it gets ignored by github.
- In backend, make sure that you have **start** script which starts the backend dev server using **node \_\_\_\_\_.js** file. You can use **dev** script for development server using **nodemon**. Please don't use nodemon in production.

<iframe width="560" height="315"  
src="https://www.youtube.com/embed/rm8AhGGYEVA?si=x8K33p9bgexXUK2X"

```
title="YouTube video player" frameborder="0" allow="accelerometer; autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-picture; web-share" allowfullscreen></iframe>
```

## Buy Hostinger VPS Hosting

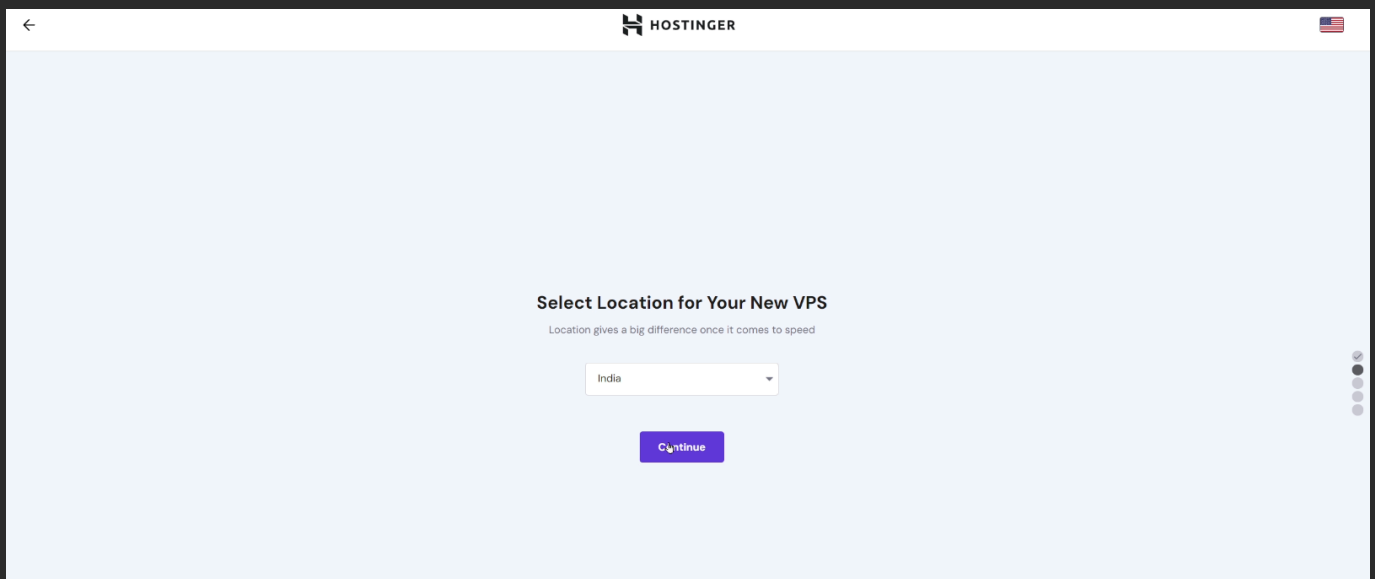
Click here to Buy India Best Hosting 🖱️ [Http://Www.Hostinger.Com/Thapa7](http://Www.Hostinger.Com/Thapa7)

For Thapa Family ✨ Use the discount code **THAPA7** to get Extra 10% OFF!

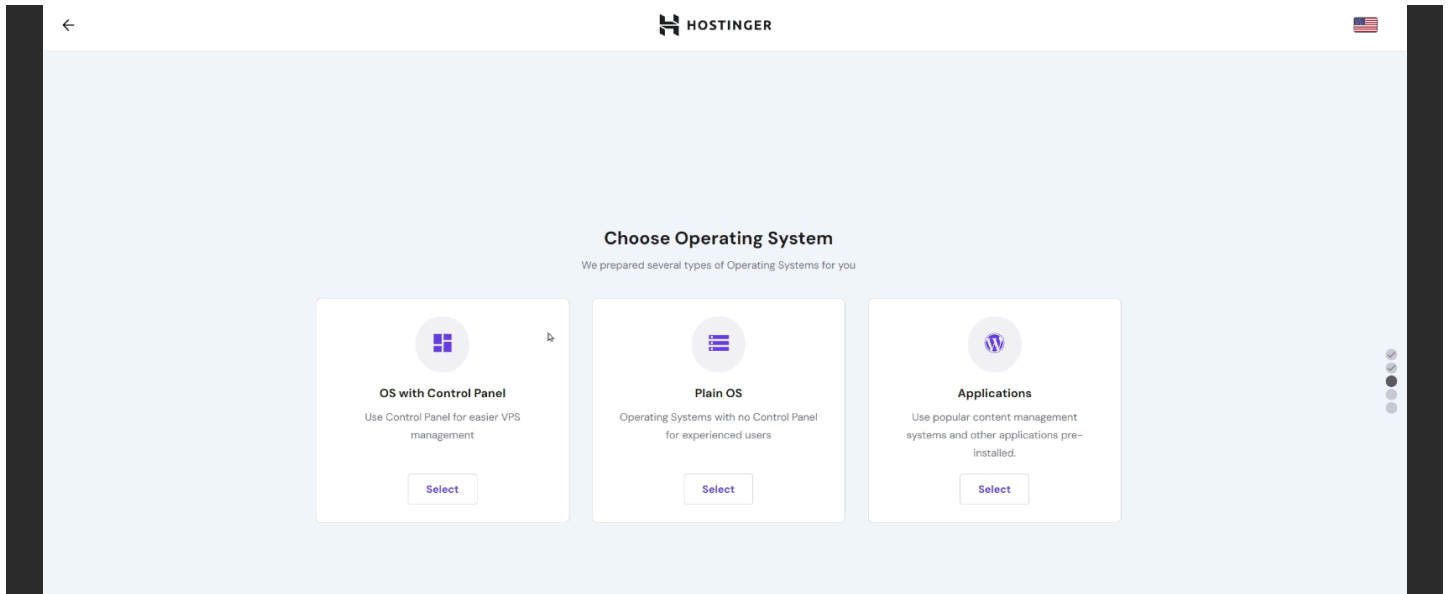
After Successful Payment, We will redirect to Homepage, we need to simply click on Start.

Now, then we need to select location

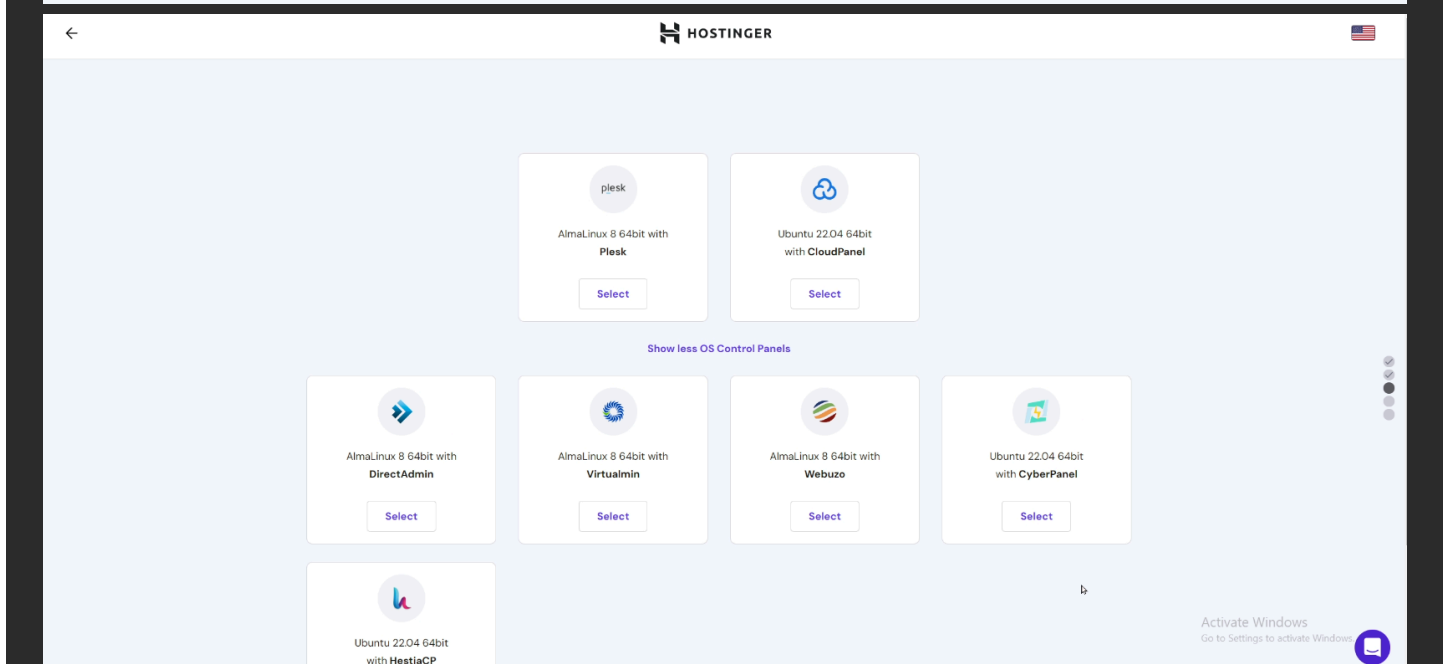
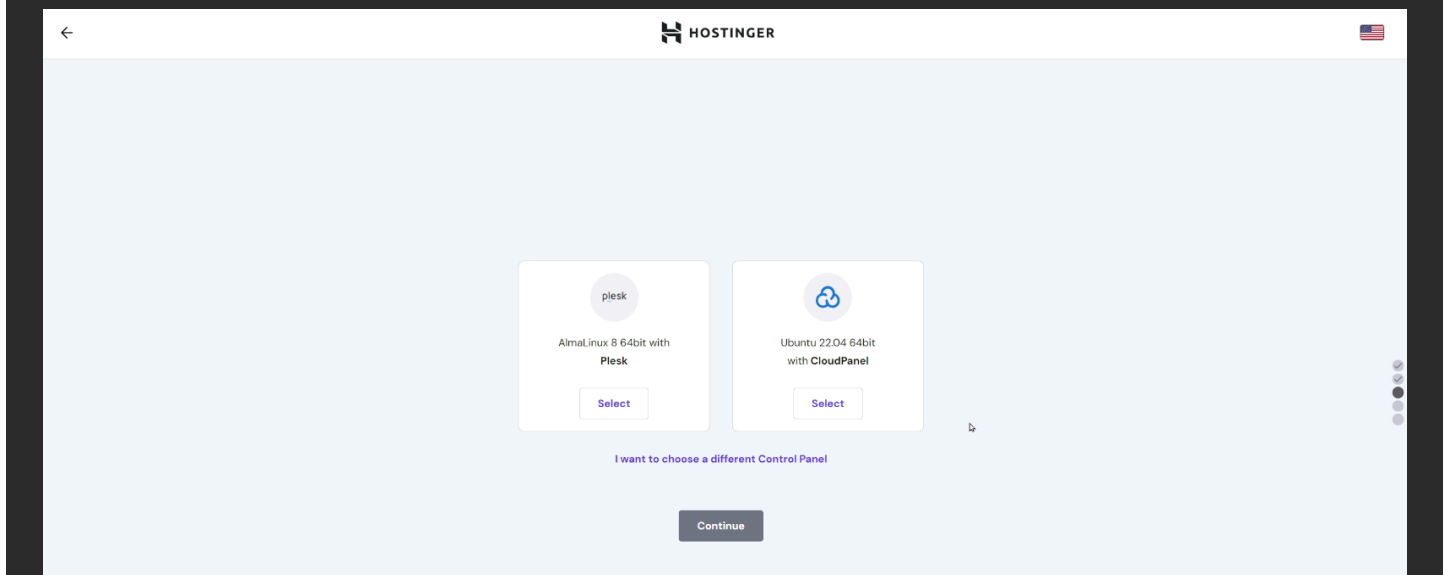
## Initial Setup

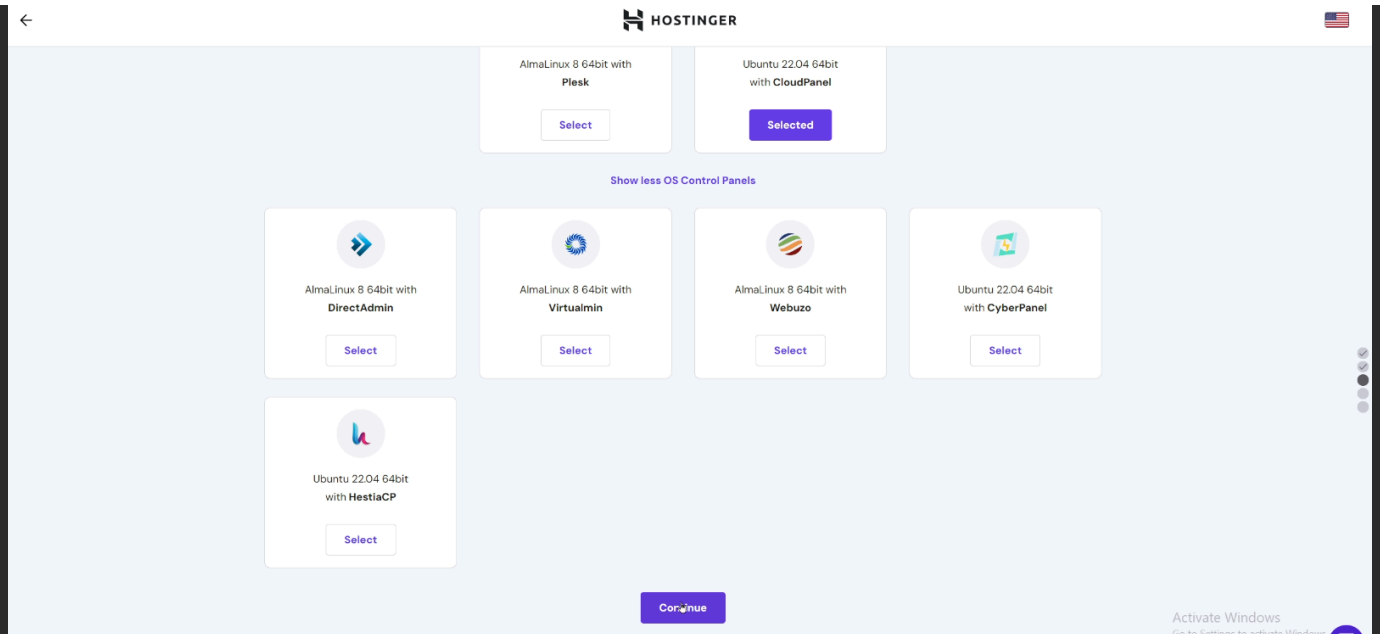


We need to choose Operating system, We will choose OS with control Panel and click on select

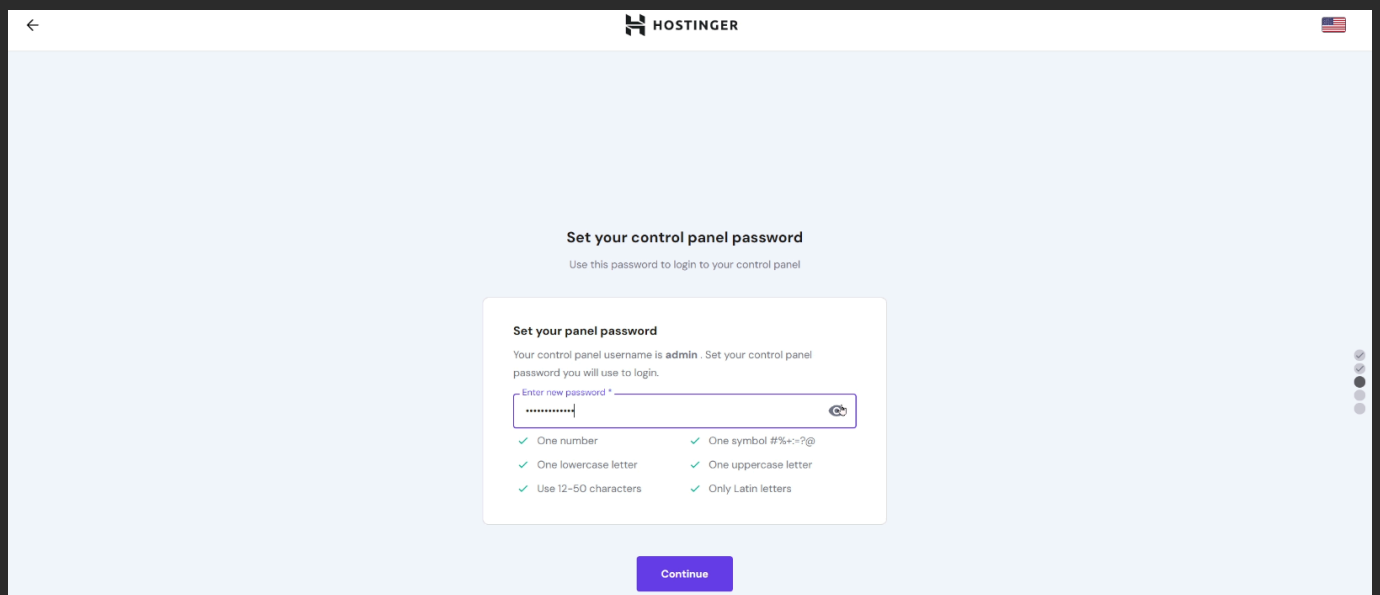
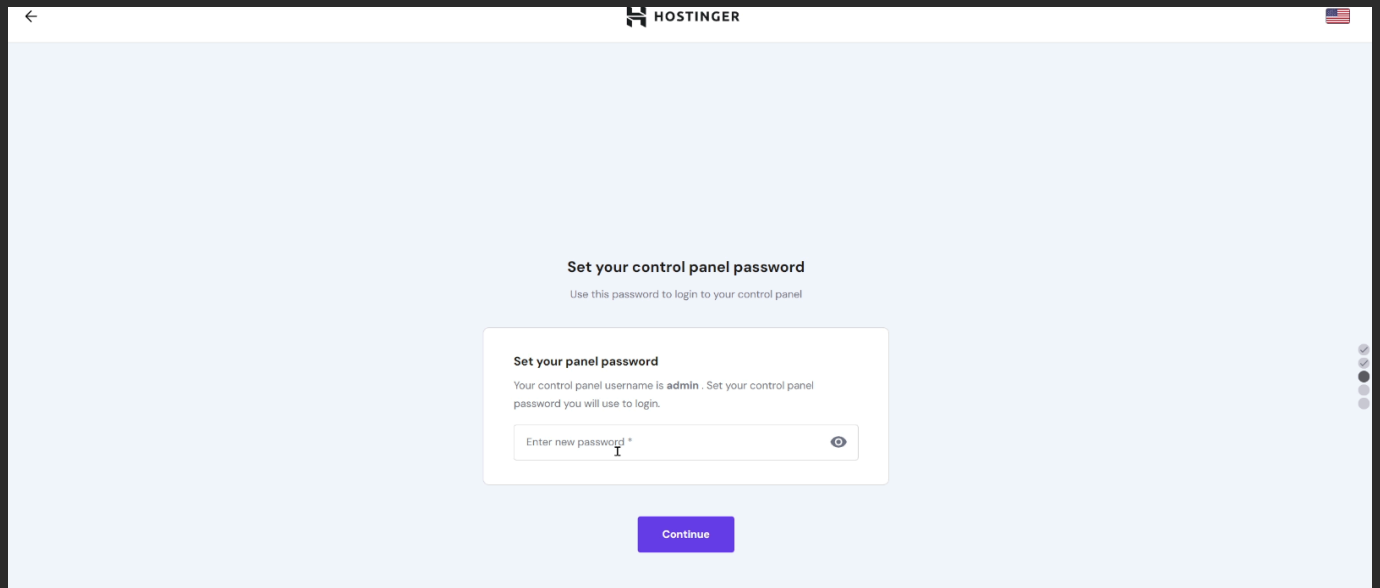


We need to choose the OS for our VPS and we will select Ubuntu 22.04 64bit





Now, we need to set our Panel Password, By default the panel username is `admin`



Now we need to set the password and ssh key for our VPS Server, Remember you can connect to your VPS using SSH. Open a terminal on your local machine and use the following command, replacing `<your-username>` and `<your-vps-ip>` with your actual VPS details:

The screenshot shows the Hostinger VPS setup interface. At the top, it says "You're Doing Great! Almost There..." with a subtext "Set password and SSH key for your new server". There are three main sections: 1. "Your VPS hostname" with a text box containing "srv416761hstgr.cloud". 2. "Set secure root password" with a text box for "Enter root password" and a toggle for visibility. 3. "Add SSH key (optional)" with a large text box for "SSH key (optional)" and a "Name" field below it. A "Save & Continue" button is at the bottom. A "Activate Windows" watermark is visible in the bottom right corner.

We just need to review our Information and that's it.

The screenshot shows the Hostinger VPS setup interface at the "Let's Finish Setting Up" stage. It says "Review your information and finish the set up!". A summary box displays: "VPS Location: India", "Operating System: Ubuntu 22.04 64bit with CloudPanel", and "Hostname: srv416761hstgr.cloud". A large blue button with a white circle icon is at the bottom.

Our VPS is getting ready to process

srv416761.hstgr.cloud

Creating process started

25% Initiating setup

We will go with VPS Dashboard

**Well Done, You Are Ready!**

Head to the VPS dashboard to manage your server, or buy a domain to make your website easily accessible

**VPS dashboard**

Manage your VPS from our custom made dashboard

[Manage server](#)**Buy a Domain**

Purchase a new domain name for your VPS

[Buy](#)

This is it, we are in our VPS Dashboard finally

The screenshot shows the Hostinger VPS Dashboard for the server **srv416761.hstgr.cloud**. The top navigation bar includes links for Home, Hosting, Emails, Domains, **VPS**, and Billing. The left sidebar contains a menu with options like Overview, AI Assistant BETA, Settings, Server Usage, Snapshots & Backups, Latest Actions, Operating System, Firewall, Help, and Refer to A Friend. The main content area displays the VPS information table and management options.

VPS information	SSH access	Panel access	Plan details
IP address	62.72.59.218		
Status	Running		
VPS uptime	-		
Current OS	Ubuntu 22.04 64bit with CloudPanel		
Location	India		
Node	in-mum-pve-node24.hostinger.io		

Buttons: [Reboot VPS](#), [Stop VPS](#)

**Manage VPS**

**Plan details**

Your current plan is KVM 2. Upgrade for bigger resources!

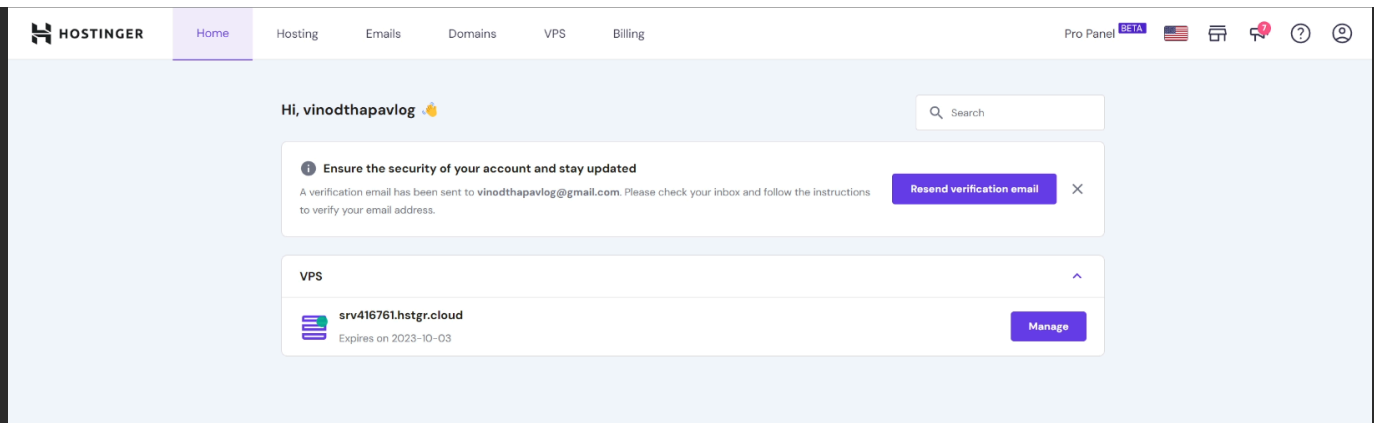
[Upgrade](#)

**Expiration date**

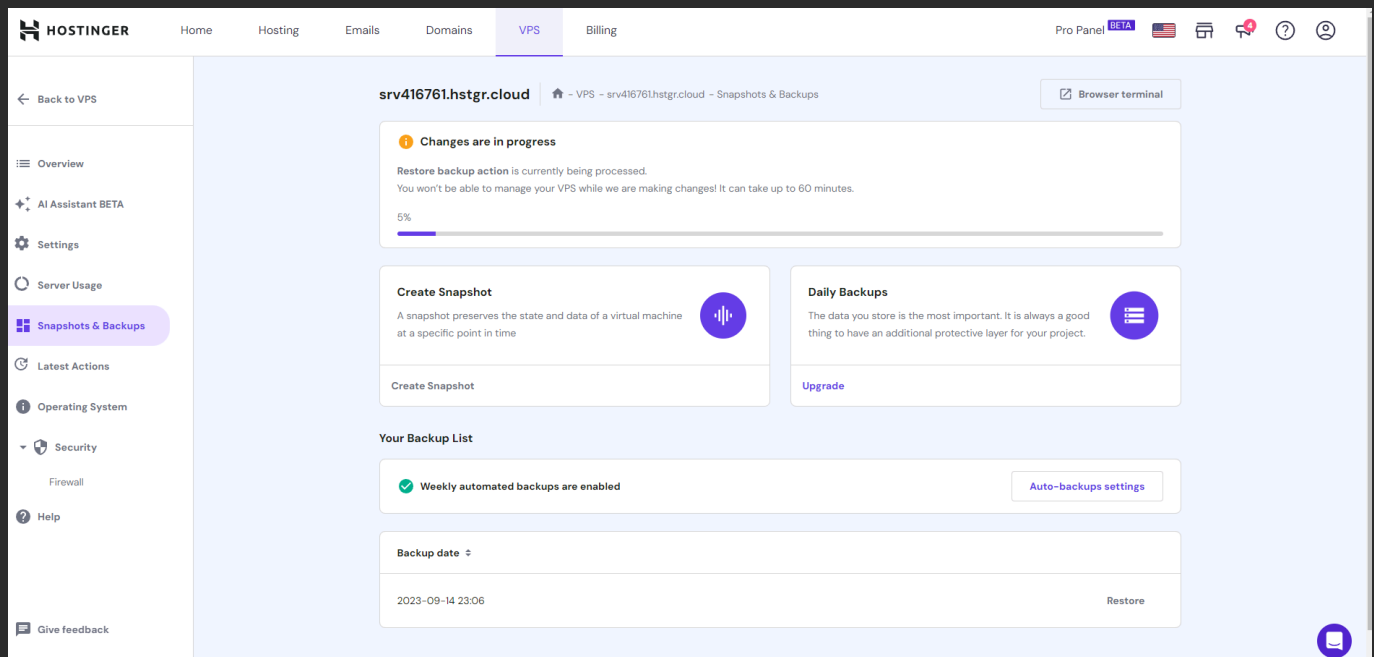
Your VPS will expire on 2023-10-03

[Renew](#)

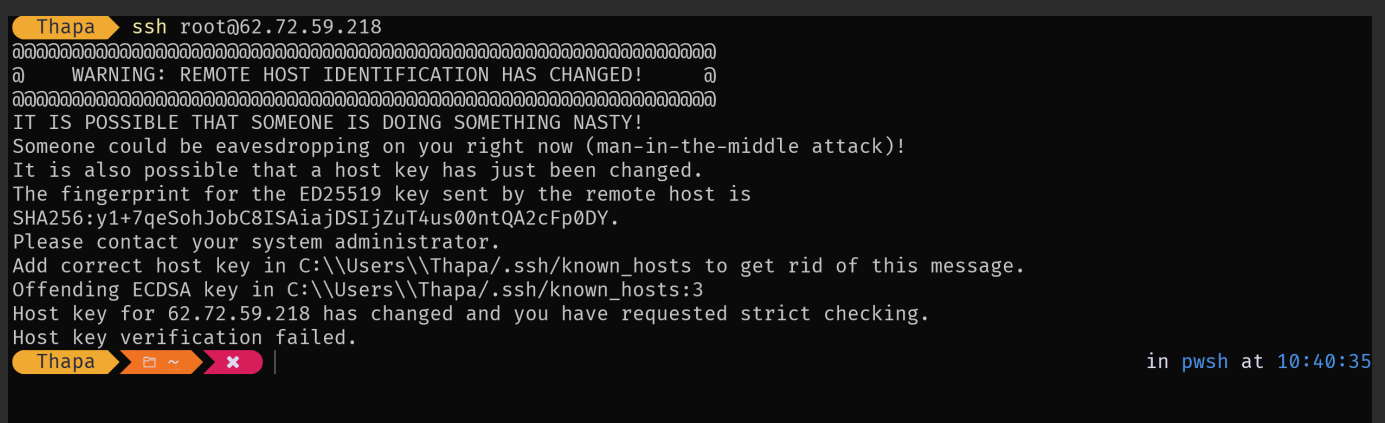
Go to Home, which is on Navbar, just verify your email for future updates and reference.



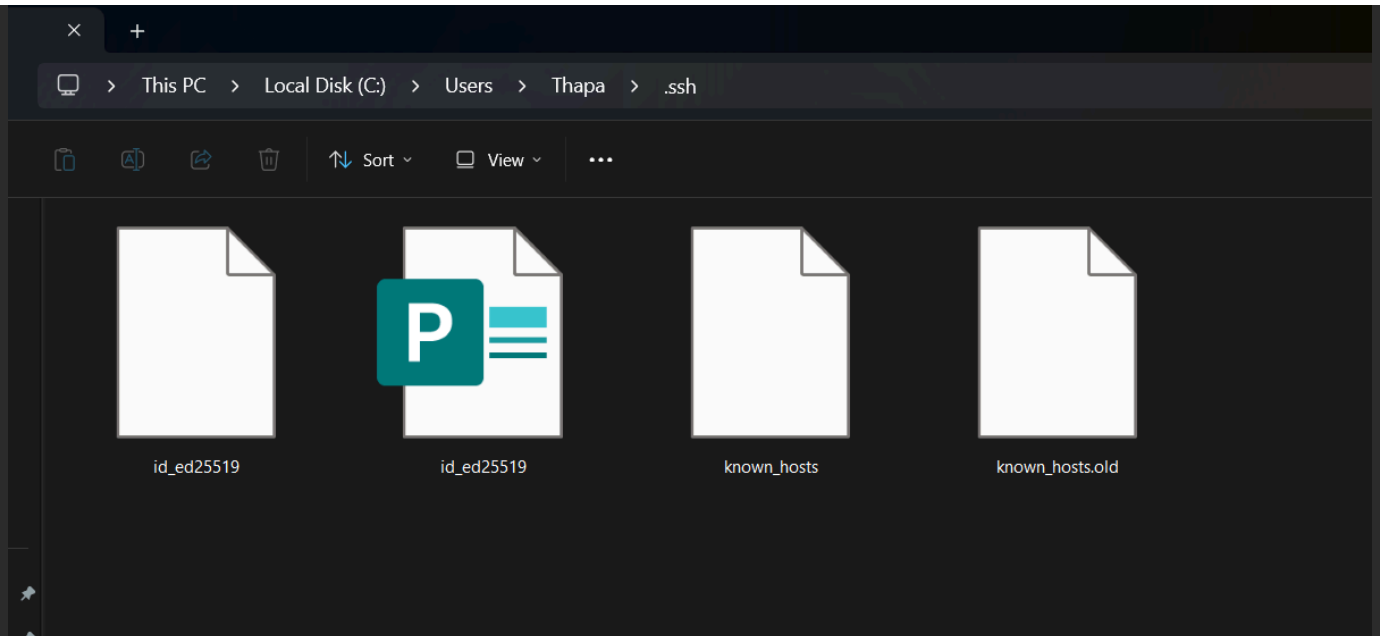
But for me, I need to restore to show all things frm start



Also, changed the ssh password too and then try to login



Since we changed the password and I have already added my ip address as trusted host. I need to remove old list.



Finally now, we can login

```
Thapa ssh root@62.72.59.218
The authenticity of host '62.72.59.218 (62.72.59.218)' can't be established.
ED25519 key fingerprint is SHA256:y1+7qeSohJobC8ISAiajDSIjZuT4us00ntQA2cFp0DY.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? y
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added '62.72.59.218' (ED25519) to the list of known hosts.
root@62.72.59.218's password:

#####
###           Welcome to CloudPanel           ###
#####

* Website:      https://www.cloudpanel.io
* Documentation: https://www.cloudpanel.io/docs/v2/
* Best Practices: https://www.cloudpanel.io/docs/v2/best-practices/
* CloudPanel:   https://62.72.59.218:8443
* CloudPanel CLI: clpctl

Last login: Thu Sep 14 13:30:00 2023 from 113.199.230.33
root@srv416761:~#
```

## Steps to Host MERN App

### Access Your VPS:

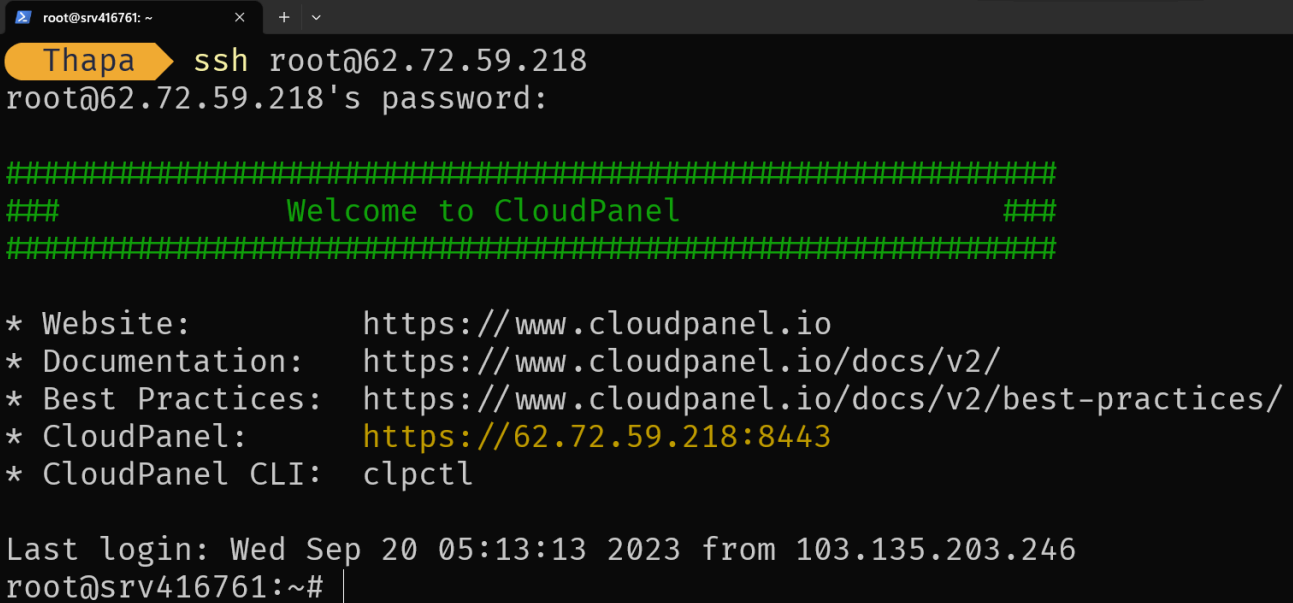
- Log in to your Hostinger account and access your VPS dashboard.

### Connect to Your VPS:



You can connect to your VPS using SSH. Open a terminal on your local machine and use the following command, replacing `<your-username>` and `<your-vps-ip>` with your actual VPS details:

```
ssh <your-username>@<your-vps-ip>
```



A terminal window titled 'root@srv416761: ~' shows an SSH session. The user 'Thapa' runs 'ssh root@62.72.59.218'. The terminal displays the password prompt, a green welcome message from CloudPanel, a list of links for website, documentation, best practices, and the CloudPanel CLI, and the last login information.

```
Thapa ssh root@62.72.59.218
root@62.72.59.218's password:

#####
###           Welcome to CloudPanel           ###
#####

* Website:           https://www.cloudpanel.io
* Documentation:     https://www.cloudpanel.io/docs/v2/
* Best Practices:    https://www.cloudpanel.io/docs/v2/best-practices/
* CloudPanel:        https://62.72.59.218:8443
* CloudPanel CLI:    clpctl

Last login: Wed Sep 20 05:13:13 2023 from 103.135.203.246
root@srv416761:~# |
```

## Update and Upgrade:

Before installing any software, it's a good practice to update your VPS packages:

```
sudo apt update
sudo apt upgrade
```

### sudo:

- **sudo** stands for "Superuser Do" or "Substitute User and Do." It is a command that allows users with the appropriate privileges to execute commands as the superuser (administrator) or another user, based on their configuration. This is often used for performing administrative tasks that require elevated permissions.

## apt:

- **apt** stands for "Advanced Package Tool." It is a command-line package management tool used in Debian-based Linux distributions, including Ubuntu. **apt** allows you to interact with the system's package manager to install, update, remove, or manage software packages.

So, when you run **sudo apt update**, you are telling the system to refresh the package repository information, and when you run **sudo apt upgrade**, you are telling the system to upgrade your installed packages to their latest versions.

## Pending kernel upgrade

Newer kernel available

The currently running kernel version is 5.15.0-1040-kvm which is not the expected kernel version 5.15.0-1042-kvm.

Restarting the system to load the new kernel will not be handled automatically, so you should consider rebooting.

<Ok>

## Daemons using outdated libraries

Which services should be restarted?

```
[*] chrony.service
[*] clp-agent.service
[*] clp-nginx.service
[*] clp-php-fpm.service
[*] cron.service
[ ] dbus.service
[*] irqbalance.service
[*] memcached.service
[*] multipathd.service
[*] mysql.service
[ ] networkd-dispatcher.service
[*] nginx.service
[*] packagekit.service
[*] php7.1-fpm.service
[*] php7.2-fpm.service
[*] php7.3-fpm.service
[*] php7.4-fpm.service
[*] php8.0-fpm.service
[*] php8.1-fpm.service
[*] php8.2-fpm.service
[*] polkit.service
```

<Ok>

<Cancel>

## Install Node.js using NVM

We are going to install using **NVM** (Node Version Manager)

Go to NVM's repo.

## About

nvm is a version manager for [node.js](#), designed to be installed per-user, and invoked per-shell. `nvm` works on any POSIX-compliant shell (sh, dash, ksh, zsh, bash), in particular on these platforms: unix, macOS, and [windows WSL](#).

## Installing and Updating

### Install & Update Script

To **install** or **update** nvm, you should run the [install script](#). To do that, you may either download and run the script manually, or use the following cURL or Wget command:

```
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.5/install.sh | bash
```

```
wget -qO- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.5/install.sh | bash
```

The name "curl" stands for "Client for URLs." `curl` is a command-line tool and library for transferring data with URLs. `curl` allows you to send HTTP requests to web servers. You can use it to retrieve web pages, download files, send POST requests, and more.

This command downloads and runs the NVM installation script. It will add NVM to your shell's configuration files (e.g., `.bashrc` or `.zshrc`). We need to reload our shell configuration to use NVM since we installed right now.

If you're using Bash, you can try running:

```
// Since our VPS uses bash shell
source ~/.bashrc

// If your VPS uses zsh shell
source ~/.zshrc
```

After running, run `nvm --version` if you're not getting the expected output of the NVM version, it's possible that the `nvm` command is not recognized in your current shell session. This can happen if the shell session hasn't properly loaded the NVM script. Reconnect to your VPS might solve this issue.

If it's still not working, you can manually run this command:

## Install & Update Script

To **install** or **update** nvm, you should run the [install script](#). To do that, you may either download and run the script manually, or use the following cURL or Wget command:

```
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.4/install.sh | bash
```

```
wget -qO- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.4/install.sh | bash
```

Running either of the above commands downloads a script and runs it. The script clones the nvm repository to `~/.nvm`, and attempts to add the source lines from the snippet below to the correct profile file (`~/.bash_profile`, `~/.zshrc`, `~/.profile`, or `~/.bashrc`).

```
export NVM_DIR="$([ -z "${XDG_CONFIG_HOME-}" ] && printf %s "${HOME}/.nvm" || printf %s "${XDG_CONFIG_HOME}/nvm")
[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh" # This loads nvm
```

```
Last login: Wed Sep 20 05:17:39 2023 from 103.135.203.246
root@srv416761:~# nvm --version
0.39.5
```

Once NVM is installed, you can install the desired version of Node.js. For example, to install the latest LTS (Long Term Support) version of Node.js, you can run:

```
nvm install <node-version>

// Example:
// nvm install 18
// v16 is dead, so please do not use it.

// or
nvm install --lts

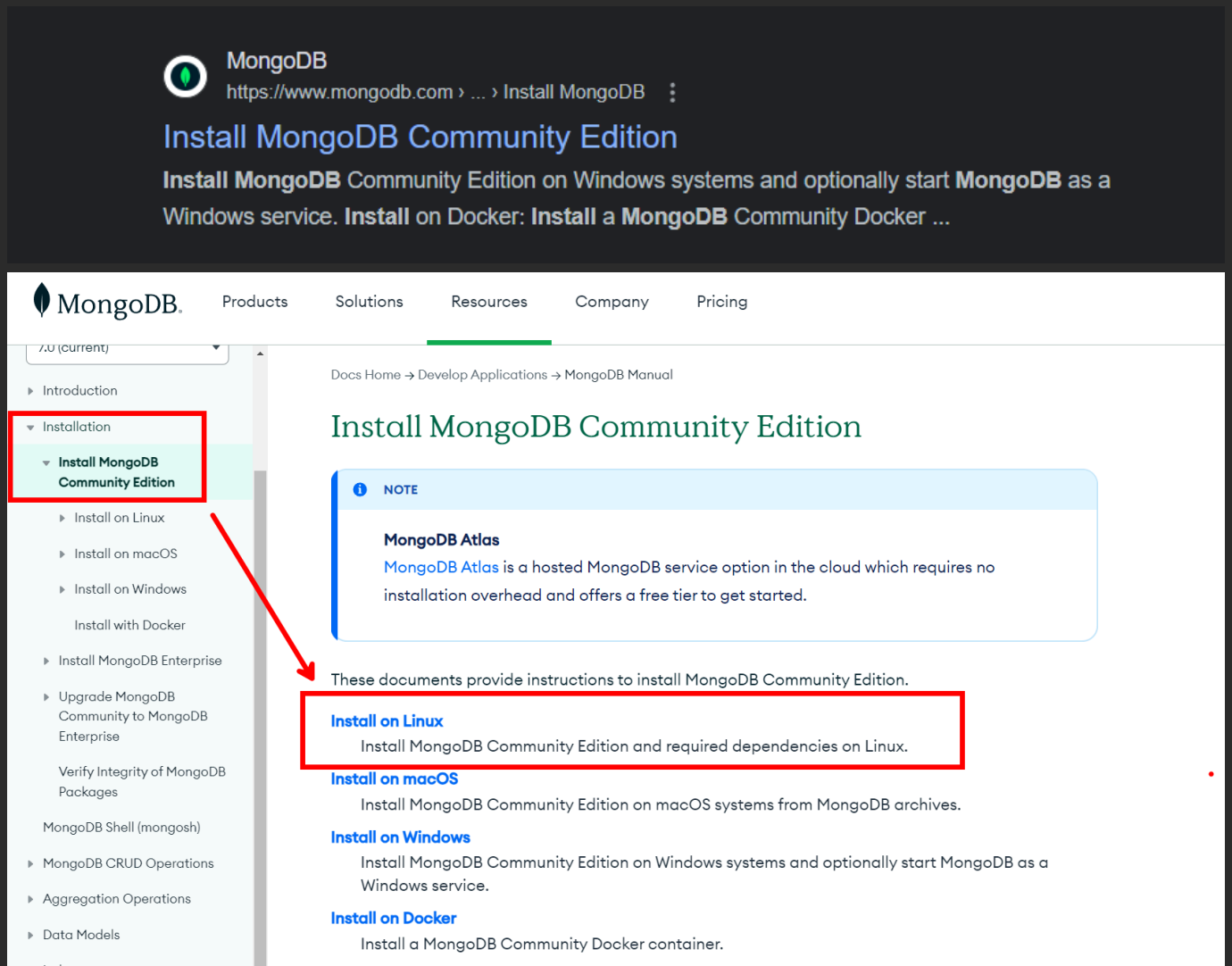
// It is recommended to use LTS version for deployment.
```

Now just to check node.js install or not run the `node -v`

```
node -v

// Output: v18.18.0
```

# Install MongoDB:



MongoDB  
https://www.mongodb.com > ... > Install MongoDB

## Install MongoDB Community Edition

Install MongoDB Community Edition on Windows systems and optionally start MongoDB as a Windows service. [Install on Docker: Install a MongoDB Community Docker ...](#)

Products Solutions Resources Company Pricing

7.0 (current)

- Introduction
- Installation
  - Install MongoDB Community Edition**
  - Install on Linux
  - Install on macOS
  - Install on Windows
  - Install with Docker
- Install MongoDB Enterprise
- Upgrade MongoDB Community to MongoDB Enterprise
- Verify Integrity of MongoDB Packages
- MongoDB Shell (mongosh)
- MongoDB CRUD Operations
- Aggregation Operations
- Data Models

Docs Home → Develop Applications → MongoDB Manual

## Install MongoDB Community Edition

**NOTE**

**MongoDB Atlas**  
MongoDB Atlas is a hosted MongoDB service option in the cloud which requires no installation overhead and offers a free tier to get started.

These documents provide instructions to install MongoDB Community Edition.

- Install on Linux**  
Install MongoDB Community Edition and required dependencies on Linux.
- Install on macOS**  
Install MongoDB Community Edition on macOS systems from MongoDB archives.
- Install on Windows**  
Install MongoDB Community Edition on Windows systems and optionally start MongoDB as a Windows service.
- Install on Docker**  
Install a MongoDB Community Docker container.

Choose "Install on Linux" then choose your Linux distribution that your VPS is using. We are using Ubuntu VPS.

## Recommended

For the best installation experience, MongoDB provides packages for popular Linux distributions. These packages are the preferred way to run MongoDB. The following guides detail the installation process for these systems:

### Install on Red Hat

Install MongoDB Community Edition on Red Hat Enterprise and related Linux systems using `.rpm` packages.

### Install on Ubuntu

Install MongoDB Community Edition on Ubuntu Linux systems using `.deb` packages.

### Install on Debian

Install MongoDB Community Edition on Debian systems using `.deb` packages.

### Install on SUSE

Install MongoDB Community Edition on SUSE Linux systems using `.rpm` packages.

### Install on Amazon

Install MongoDB Community Edition on Amazon Linux AMI systems using `.rpm` packages.

There you will see some steps in "Install MongoDB Community Edition" section, follow that. For Ubuntu, there are 4 steps.

Packages mentioned in Step 1 is installed by default in our VPS, so we can skip that.

The name "curl" stands for "Client for URLs." **curl** is a command-line tool and library for transferring data with URLs.

# Install MongoDB Community Edition

Follow these steps to install MongoDB Community Edition using the `apt` package manager.

## 1 Import the public key used by the package management system

From a terminal, install `gnupg` and `curl` if they are not already available:

```
sudo apt-get install gnupg curl
```



To import the MongoDB public GPG key from <https://pgp.mongodb.com/server-7.0.asc>, run the following command:

```
curl -fsSL https://pgp.mongodb.com/server-7.0.asc | \  
sudo gpg -o /usr/share/keyrings/mongodb-server-7.0.gpg \  
--dearmor
```



## 2 Create a list file for MongoDB

Create the list file `/etc/apt/sources.list.d/mongodb-org-7.0.list` for your version of Ubuntu.



## 2 Create a list file for MongoDB.

Create the list file `/etc/apt/sources.list.d/mongodb-org-6.0.list` for your version of Ubuntu.

Click on the appropriate tab for your version of Ubuntu. If you are unsure of what Ubuntu version the host is running, open a terminal or shell on the host and execute `lsb_release -dc`.

**Ubuntu 22.04 (Jammy)**

Ubuntu 20.04 (Focal)

Ubuntu 18.04 (Bionic)

Ubuntu 16.04 (Xenial)

Create the `/etc/apt/sources.list.d/mongodb-org-6.0.list` file for **Ubuntu 22.04 (Jammy)**

```
echo "deb [ arch=amd64,arm64 signed-by=/usr/share/keyrings/mongodb-server-6.0.g
```

## 3 Reload local package database.

Issue the following command to reload the local package database:

```
sudo apt-get update
```

## 4 Install the MongoDB packages.

You can install either the latest stable version of MongoDB or a specific version of MongoDB.

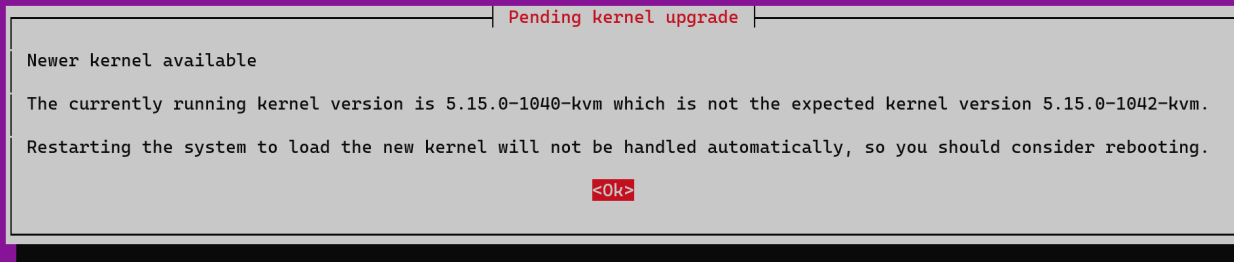
**Install the latest version of MongoDB.** Install a specific release of MongoDB.

To install the latest stable version, issue the following

```
sudo apt-get install -y mongodb-org
```

This command installs MongoDB and its components.

Whenever this type of prompt open press Enter. Also the last 4th number command to install mongodb-org takes times. So keep patience.



**Extra Tips:** If you are wondering Jammy ("Jammy" is a codename for a version of the Ubuntu operating system.) "Jammy" corresponds to Ubuntu 22.04, which is expected to be released in April 2022. Each Ubuntu release also has a version number that includes the year and month of the expected release, so "Ubuntu 22.04" means it's scheduled for release in April 2022".

## Start and Enable MongoDB:

Once MongoDB is installed, start and enable the MongoDB service:

```
sudo systemctl start mongod

// If the above command is not working:

sudo systemctl enable mongod
```

## Check the Status:

You can check the status of the MongoDB service to ensure it's running without issues:

```
sudo systemctl status mongod
```

```
root@srv416761:~# sudo systemctl status mongod
○ mongod.service – MongoDB Database Server
   Loaded: loaded (/lib/systemd/system/mongod.service; disabled; vendor preset: enabled)
   Active: inactive (dead)
     Docs: https://docs.mongodb.org/manual
root@srv416761:~# |
```

for the first time, it will show inactive because we haven't start our MongoDB.

```
root@srv416761:~# sudo systemctl start mongod
root@srv416761:~# sudo systemctl enable mongod
Created symlink /etc/systemd/system/multi-user.target.wants/mongod.service → /lib/systemd/system/mongod.service.
root@srv416761:~# sudo systemctl status mongod
● mongod.service - MongoDB Database Server
   Loaded: loaded (/lib/systemd/system/mongod.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2023-09-20 06:12:26 UTC; 10s ago
     Docs: https://docs.mongodb.org/manual
   Main PID: 16860 (mongod)
    Memory: 73.9M
       CPU: 378ms
    CGroup: /system.slice/mongod.service
            └─16860 /usr/bin/mongod --config /etc/mongod.conf

Sep 20 06:12:26 srv416761 systemd[1]: Started MongoDB Database Server.
```

Check Your Ubuntu Version: `lsb_release -a`

```
root@srv416761:~# lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 22.04.3 LTS
Release:        22.04
Codename:       jammy
root@srv416761:~# |
```

## Install Git:

- You might need Git to clone your MERN application repository. Install it using:

```
sudo apt install git
```

- After installing git, if your github repository is private then you might want to install **GitHub CLI** too. We need for Ubuntu. So...



Thapa  
Technical

[Home](#)

[About](#)

[Source  
Code](#)

[Blogs](#)

[Courses](#)

[Contact](#)

[Login](#)

[Signup](#)

## Linux & BSD

**gh** is available via:

- [our Debian and RPM repositories](#);
- community-maintained repositories in various Linux distros;
- OS-agnostic package managers such as [Homebrew](#), [Conda](#), and [Spack](#); and
- [our releases page](#) as precompiled binaries.

For more information, see [Linux & BSD installation](#).

### Debian, Ubuntu Linux, Raspberry Pi OS (apt)

Install:

```
type -p curl >/dev/null || (sudo apt update && sudo apt install curl -y)
curl -fsSL https://cli.github.com/packages/githubcli-archive-keyring.gpg | sudo dd of=/usr/share/keyrings/githubcli-archive-keyr
&& sudo chmod go+r /usr/share/keyrings/githubcli-archive-keyring.gpg \
&& echo "deb [arch=$(dpkg --print-architecture) signed-by=/usr/share/keyrings/githubcli-archive-keyring.gpg] https://cli.github
&& sudo apt update \
&& sudo apt install gh -y
```

#### Note

We were recently forced to change our GPG signing key. If you've previously downloaded the `githubcli-archive-keyring.gpg` file, you should re-download it again per above instructions. If you are using a keyserver to download the key, the ID of the new key is

`23F3D4EA75716059`.

Upgrade:

```
sudo apt update
sudo apt install gh
```

After GitHub CLI is installed, use this command follow the steps mentioned by it to automatically setup everything required for authentication:

```
gh auth login
```

```
root@srv416761:~# gh auth login
? What account do you want to log into? GitHub.com
? What is your preferred protocol for Git operations? HTTPS
? Authenticate Git with your GitHub credentials? Yes
? How would you like to authenticate GitHub CLI? Login with a web browser

! First copy your one-time code: 4DE8-379D
- Press Enter to open github.com in your browser... |
```

```
root@srv416761:~# gh auth login
? What account do you want to log into? GitHub.com
? What is your preferred protocol for Git operations? HTTPS
? Authenticate Git with your GitHub credentials? Yes
? How would you like to authenticate GitHub CLI? Login with a web browser

! First copy your one-time code: 0343-E853
- Press Enter to open github.com in your browser...
! Failed opening a web browser at https://github.com/login/device
exec: "xdg-open,x-www-browser,www-browser,wslview": executable file not found in $PATH
Please try entering the URL in your browser manually
✓ Authentication complete. Press Enter to continue...

- gh config set -h github.com git_protocol https
✓ Configured git protocol
✓ Logged in as thapatechnical
root@srv416761:~# |
```



## Device Activation

Enter the code displayed on your device

0 3 4 3 - E 8 5 3

Continue

GitHub staff will never ask you to enter your code on this page.

## Clone Your Repository (MERN Application):

Use Git to clone your MERN application's repository into a directory on your VPS.

```
git clone <github-repo>

cd <your-project-folder>
```

## Setup backend

Now, let's start with backend. Go inside your backend or server folder and install all dependencies

```
npm install
yarn install
pnpm install
bun install

// But we are using pnpm in our project, so first we need to instal pnpm and then insta
npm i -g pnpm
pnpm i
```

```
ls
// Output:
' client package.json package-lock.json pnpm-lock.yaml server

cd server
```

## Setup environment variables

```
nano .env // Nano is a shell based text editor which comes by default on linux

// We have an environment variable named DB_URL so let's include it.
MONGODB_URI=mongodb://127.0.0.1/vps-test-db
JWT_SECRET=oHSLF&#WsA#WpJ!QCWLz7rCL5^sHgD62Qe8tgb3&SSscRu$sFi*L9MSSgnBM2*!$pyXS&XwHyK8f
```

```
PORT=8000
```

```
// Then press Ctrl + O and enter to save it.
```

```
// The press Ctrl + X to exit out of nano.
```

```
// Let's use "cat" command to see if everything is fine.
```

```
// The cat command in Linux (short for "concatenate") is primarily used to display the  
cat .env
```

```
// This shows content of .env file
```

## Start backend server

```
npm run start
```

```
yarn start
```

```
pnpm start // we will use pnpm bcz we use pnpm to instal all the packages
```

```
bun run start
```

```
root@srv416761:~/TypeScript-Simple-Mern-Project/server# pnpm start
```

```
> TypeScript Project Server@1.0.0 start /root/TypeScript-Simple-Mern-Project/server  
> ts-node server.ts
```

```
=====  
===== ENV: development =====  
🚀 App listening on the port 8000  
=====
```

The problem with this approach is that if an error occurs then it doesn't restart itself and your site will be down.

To fix this issue, we can use pm2.

pm2 (Process Manager 2) is a popular process manager for Node.js applications. It is used to manage and monitor Node.js applications, ensuring they run reliably in production environments.

pm2 simplifies the process of managing Node.js applications by allowing you to start, stop, restart, and monitor multiple Node.js processes with a single command.

```
npm i -g pm2
pm2 start npm --name "test_server" -- start
pm2 ls
pm2 logs test_server

// -- start (there is a space, Keep in mind)
// To test our backend
curl http://localhost:8000
// Note: In our backend we have a route for homepage so we are using it
```

Here's what each part of the command does:

- **pm2 start npm**: This tells **pm2** to use **npm** as the interpreter.
- **-name "test\_server"**: This assigns the name "test\_server" to the application.
- **-- start**: This specifies that you want to run the "start" script defined in your **package.json** file.

```
pm2 -v
// 5.3.0
```

Here we can clearly see our backend is now live and status is online.



```
root@srv416761:~/TypeScript-Simple-Mern-Project/server# pm2 -v
5.3.0
root@srv416761:~/TypeScript-Simple-Mern-Project/server# pm2 start npm --name "test_server" -- start
[PM2] Starting /root/.nvm/versions/node/v18.18.0/bin/npm in fork_mode (1 instance)
[PM2] Done.
```

id	name	mode	u	status	cpu	memory
0	test_server	fork	0	online	0%	48.4mb

```
root@srv416761:~/TypeScript-Simple-Mern-Project/server# pm2 ls
```

id	name	mode	u	status	cpu	memory
0	test_server	fork	0	online	0%	65.8mb

```
root@srv416761:~/TypeScript-Simple-Mern-Project/server# pm2 logs test_server
[TAILING] Tailing last 15 lines for [test_server] process (change the value with --lines option)
/root/.pm2/logs/test-server-error.log last 15 lines:
/root/.pm2/logs/test-server-out.log last 15 lines:
0|test_ser |
0|test_ser | > TypeScript Project Server@1.0.0 start
0|test_ser | > ts-node server.ts
0|test_ser |
0|test_ser | =====
0|test_ser | ===== ENV: development =====
0|test_ser | 🚀 App listening on the port 8000
0|test_ser | =====
```

Activate Wind  
Go to Settings to e

## Frontend Setup

Go to frontend or client folder. Install all the dependencies and build the project:

```
pnpm i
pnpm build
npm run build
bun run build

// setup environment var
nano .env
// We already have some content in our repo so Edit the vite api url to
http://localhost:8000
// Note: To make it easier, please setup your project so that it uses API URL environme

// as that's our server
// Ctrl + O and Enter to save
// Ctrl + X to exit
```

Run this command to preview our project:

```
pnpm preview --host
```

```
root@srv416761:~/TypeScript-Simple-Mern-Project/client# pnpm preview --host
> client@0.0.0 preview /root/TypeScript-Simple-Mern-Project/client
> vite preview "--host"

→ Local:    http://localhost:4173/
→ Network:  http://62.72.59.218:4173/
→ press h to show help
```

Now, Here we can't see our website page, because of firewall. To make our PORT publicly accessible, we need to follow some steps.

Use this command to see status of publicly accessible ports:

```
sudo ufw status
```

```
// The command sudo ufw status is used to check the status of the Uncomplicated Firewall
```

Use this command to allow your port:

```
sudo ufw allow 4173
```

```

22/tcp (v6)          ALLOW      Anywhere (v6)
30/tcp (v6)          ALLOW      Anywhere (v6)
443 (v6)             ALLOW      Anywhere (v6)
3433:8443/tcp (v6)   ALLOW      Anywhere (v6)
Nginx Full (v6)      ALLOW      Anywhere (v6)

root@srv416761:~/TypeScript-Simple-Mern-Project/client# sudo ufw allow 4173
Rule added
Rule added (v6)
root@srv416761:~/TypeScript-Simple-Mern-Project/client# sudo ufw status
Status: active

```

To	Action	From
--	---	---
22/tcp	ALLOW	Anywhere
30/tcp	ALLOW	Anywhere
443	ALLOW	Anywhere
3433:8443/tcp	ALLOW	Anywhere
Nginx Full	ALLOW	Anywhere
4173	ALLOW	Anywhere
22/tcp (v6)	ALLOW	Anywhere (v6)
30/tcp (v6)	ALLOW	Anywhere (v6)
443 (v6)	ALLOW	Anywhere (v6)
3433:8443/tcp (v6)	ALLOW	Anywhere (v6)
Nginx Full (v6)	ALLOW	Anywhere (v6)
4173 (v6)	ALLOW	Anywhere (v6)

Run the command again `pnpm preview --host` to check is it publicly available or not.

```
pnpm preview --host
```

```

root@srv416761:~/TypeScript-Simple-Mern-Project/client# pnpm preview --host

> client@0.0.0 preview /root/TypeScript-Simple-Mern-Project/client
> vite preview "--host"

→ Local:    http://localhost:4173/
→ Network:  http://62.72.59.218:4173/
→ press h to show help

```

Here, you can see the URL. You can open it on your browser. If you can't see it. You can go to:

```

http://<your-vps-ip>:<your-frontend-port>    // this is the syntax of the url that we ge

// Example

http://62.72.59.218:4173

```

This works but vite preview is not made for production. It's recommended to use something like **nginx** to serve the assets.

```
sudo ufw deny 4173 // We are just denying the port that we made public earlier.
```

Nginx is a popular web server and reverse proxy server that is commonly used in VPS (Virtual Private Server) and server environments

You can install nginx using this command:

```
sudo apt install nginx

// It is installed by default on hostinger vps
```

```
ls

cd /etc/nginx

cd sites-enabled/

ls
//default.conf

cat default.conf
// we dont need default.conf file

rm default.conf

cd /etc/nginx/sites-available
nano 62.72.59.218.conf

// It is common convention to name the file with domain name
// or ip address based on what you are using
```

```
server {
    listen 80;
    root <project-folder>;
```

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

- <project-folder>
  - Go to your frontend project where it is, make sure your frontend project is built using **npm build**. Since, we are using vite, it's inside "dist". It should be inside "build" for create-react-app
  - Our folder: `/root/TypeScript-Simple-Mern-Project/client/dist`
- We are using that location context to allow all URLs to render index.html because we are using client side react using react-router-dom for routing.

This configuration sets up an Nginx server to listen on port 80 (HTTP) and serve static files from a specific director.

1. **listen 80;** This specifies that the server should listen on port 80, which is the default port for HTTP traffic.
2. **root /root/TypeScript-Simple-Mern-Project/client/dist;** This sets the document root for this server block. It tells Nginx to serve files from the `/root/TypeScript-Simple-Mern-Project/client/dist` directory when handling requests to this server block. This is typically used for serving static files like HTML, CSS, JavaScript, and images.
3. **location / { ... }** This is a location block that defines how Nginx should handle requests to the root directory ("/"). It contains the following directives:
  - **try\_files \$uri \$uri/ =404;** This directive tells Nginx how to handle requests. It tries to find a matching file in the specified directory (**\$uri**) and, if not found, it appends a trailing slash (**\$uri/**) and, if still not found, it returns a 404 error (**=404**). Essentially, it serves existing files as-is and sends 404 errors for everything else.

Use this command to check if nginx is working properly:

```
sudo nginx -t  
  
// nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
```

```
// nginx: configuration file /etc/nginx/nginx.conf test is successful
```

```
// go to /etc/nginx/sites-enabled
```

```
ln -s ../sites-available/62.72.59.218.conf .
```

```
// It creates symbolic link (also known as a symlink or soft link) of our configuration
```

```
systemctl restart nginx
```

```
// to restart the Nginx web server on a Linux system.
```

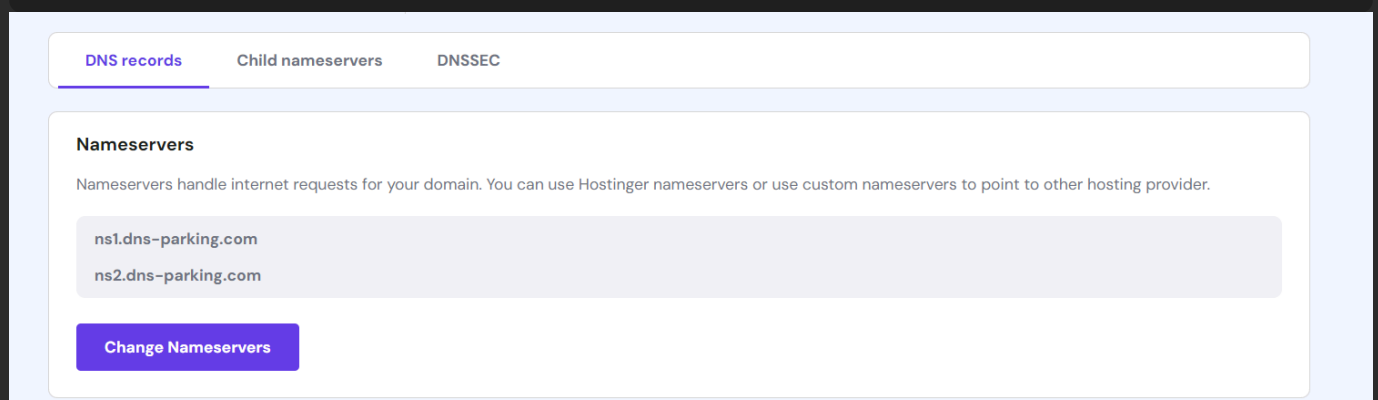
```
// By executing systemctl restart nginx, you ensure that any changes you've made to Ngi
```

Now if you go to <http://<your-ip-address>> then you will see your website. It works right? But let's see how you can connect it with a proper domain.

Go to your domain DNS settings and If the domain is not from hostinger then change the nameservers to this two.

ns1.dns-parking.com

ns2.dns-parking.com



and Add a new A record:

- Name: @
- Points to: your VPS IP address
- TTL: default
- Name: www

- Points to: your domain name (e.g., domain.tld)
- TTL: default

**Manage DNS records**

These records define how your domain behaves. Common uses include pointing your domain at web servers or configuring email delivery for your domain.

Type*	Name*	Points to*	TTL*	Add Record
A	@	62.72.59.218	14400	

If you are going to use subdomain then use <subdomain-name> and "www.<subdomain-name>" as name.

```
// now rename our config to match our domain.
// go to /etc/nginx/sites-available
// cp 62.72.59.218.conf mywebsite.com.conf
cp 62.72.59.218.conf thapatechnical.online.conf

// now let's edit this configuration to reflect our changes.
// nano mywebsite.com.conf
nano thapatechnical.online.conf
```

We are copying our previous configuration named "62.72.59.218.conf" and giving it a new name `thapatechnical.online.conf`

```
server {
    listen 80;
    server_name <domain> www.<domain>;
    root <project-root>;

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

- <domain>
  - Enter your website domain here.
  - For us, it's: server\_name thapatechnical.online www.thapatechnical.online

- <project-root>
  - Enter where your frontend built project is.
  - For us, it's: /root/TypeScript-Simple-Mern-Project/client/dist

```
// go to /etc/nginx/sites-enabled  
  
ln -s ../sites-available/mywebsite.com.conf  
ln -s ../sites-available/thapatechnical.online.conf  
  
// It creates symlink of our configuration file in current dir.  
  
systemctl restart nginx
```

Now if you go to mywebsite.com then you will see your website.

## Backend domain setup

- Name: @
- Points to: your VPS IP address
- TTL: default
- Name: www
- Points to: your domain name (e.g., domain.tld)
- TTL: default

For backend: api.thapatechnical.online. Our frontend is running on thapatechnical.online and we need to run backend on either server that's why we are going to create the subdomain of it api.thapatechnical.online

```
server {  
    listen 80;  
    server_name <api.domain> www.<api-domain>;  
  
    location / {  
        proxy_pass http://localhost:<port>;  
        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;
    }
}
```

- <api-domain>
  - Put your domain for backend API.
  - For us, it's: server\_name api.thapatechnical.online  
www.api.thapatechnical.online
- <port>
  - Enter the port where your **Localhost** backend is running. Remember when we started backend using pm2
  - For us, it's: 8000

```
go to /etc/nginx/sites-enabled
// ln -s ../sites-available/mywebsite.com.conf
ln -s ../sites-available/api.thapatechnical.online.conf
// It creates symlink of our configuration file in current dir.
systemctl restart nginx
```

```
root@srv416761:/etc/nginx/sites-available# nano api.thapatechnical.online.conf
root@srv416761:/etc/nginx/sites-available# cd ..
root@srv416761:/etc/nginx# cd sites-enabled/
root@srv416761:/etc/nginx/sites-enabled# ln -s ../sites-available/api.thapatechnical.online.conf
root@srv416761:/etc/nginx/sites-enabled# ls
62.72.59.218.conf  api.thapatechnical.online.conf  thapatechnical.online.conf
```

now, if you go to <http://api.mywebsite.com> you will see your api.

Congrats, but let's update our api from client. Go to client

```
cd
// Go to your frontend location

nano .env

// Update the api url to new url: http://api.mywebsite.com
```

```
// Update the api url to new url: http://api.thapatechnical.online  
pnpm build
```

Now, your website is live.

## SSL Setup

We are going to use Let's Encrypt SSL certificate using certbot. To install certbot:

```
sudo apt-get install certbot python3-certbot-nginx  
  
// To install certificate to your domain you can use this command:  
sudo certbot --nginx -d mywebsite.com -d www.mywebsite.com  
sudo certbot --nginx -d thapatechnical.online -d www.thapatechnical.online  
// you can also use "sudo certbot --nginx -d mywebsite.com" but since we have "www"  
// too we are passing another argument with another website.  
  
// When you do it for first time, it will ask for email, enter any email.  
// Then it will tell to agree to conditions or whatever, choose "Y"  
// At last, it will tell if you want to share your email, choose "n"  
  
// Now, let's do for our api one  
sudo certbot --nginx -d api.mywebsite.com -d www.api.mywebsite.com  
sudo certbot --nginx -d api.thapatechnical.online -d www.api.thapatechnical.online  
  
// Congrats, SSL is successfully installed on your website
```

Note: After installing ssl on your api, you would need to change api url from client because we had used "http" at that time. So, go to your client, update your API URL with https and build the project.

```
root@srv416761:~/TypeScript-Simple-Mern-Project# dir
' client package.json package-lock.json pnpm-lock.yaml server
root@srv416761:~/TypeScript-Simple-Mern-Project# cd client/
root@srv416761:~/TypeScript-Simple-Mern-Project/client# nano .env
root@srv416761:~/TypeScript-Simple-Mern-Project/client# pnpm build

> client@0.0.0 build /root/TypeScript-Simple-Mern-Project/client
> tsc && vite build

vite v4.4.5 building for production...
✓ 47 modules transformed.
dist/index.html 0.40 kB | gzip: 0.27 kB
dist/assets/index-b7363cf9.js 169.84 kB | gzip: 54.64 kB
✓ built in 2.49s
root@srv416761:~/TypeScript-Simple-Mern-Project/client# |
```

**Thapa Technical**[Home](#)[About](#)[Services](#)[News](#)[Blogs](#)[Videos](#)[Contact](#)

© Copyright 2025, All rights reserved!

[Disclaimer](#)[Terms And  
Conditions](#)[Privacy  
Policy](#)[Refund  
Policies](#)