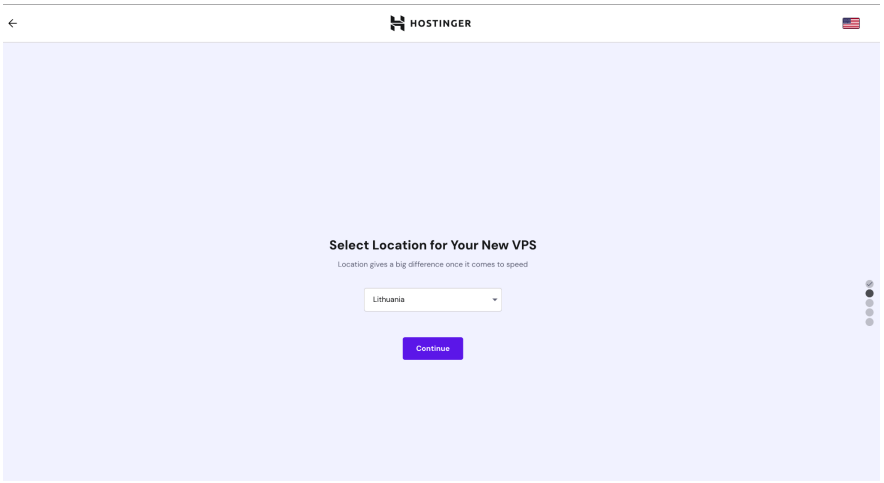
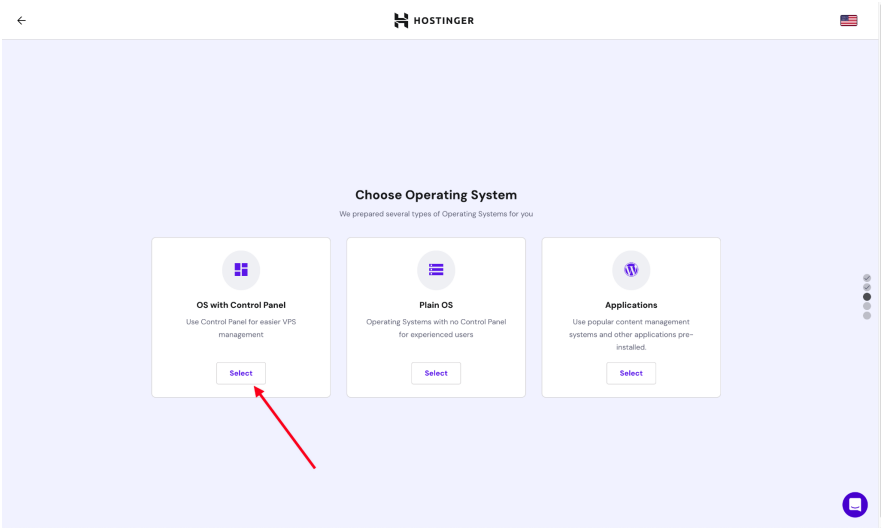
How to Deploy a Next.js Application to a VPS

1. Sign up to Hostinger and obtain a VPS server, Use code **SONNY** for **10% off**: https://www.hostinger.com/sonny

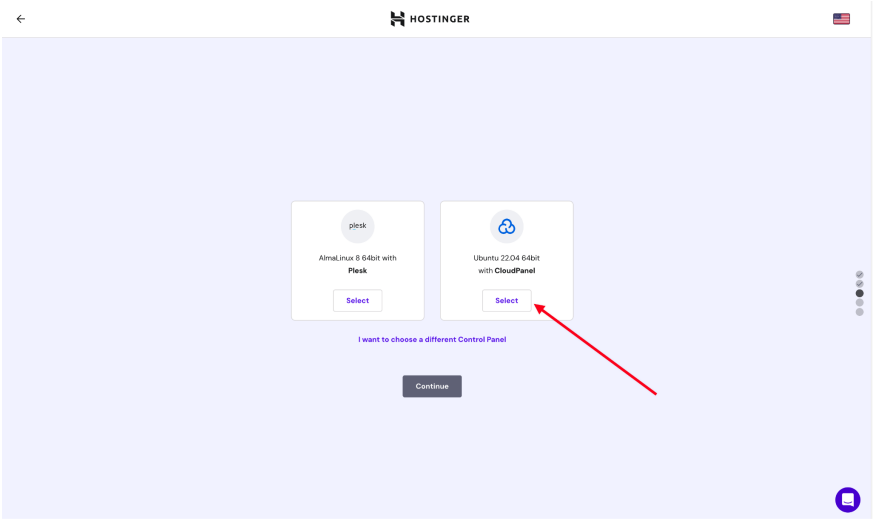
    a. Select a location for the VPS



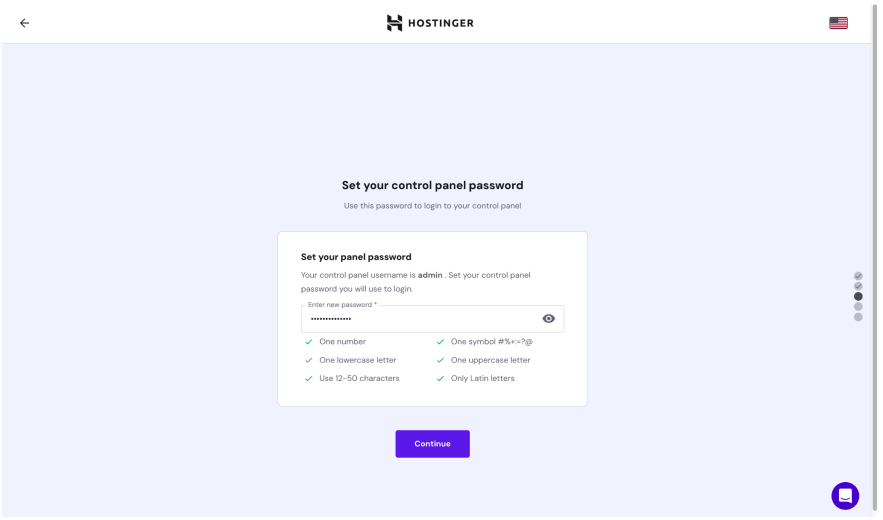
    b. Select the option for OS with control Panel



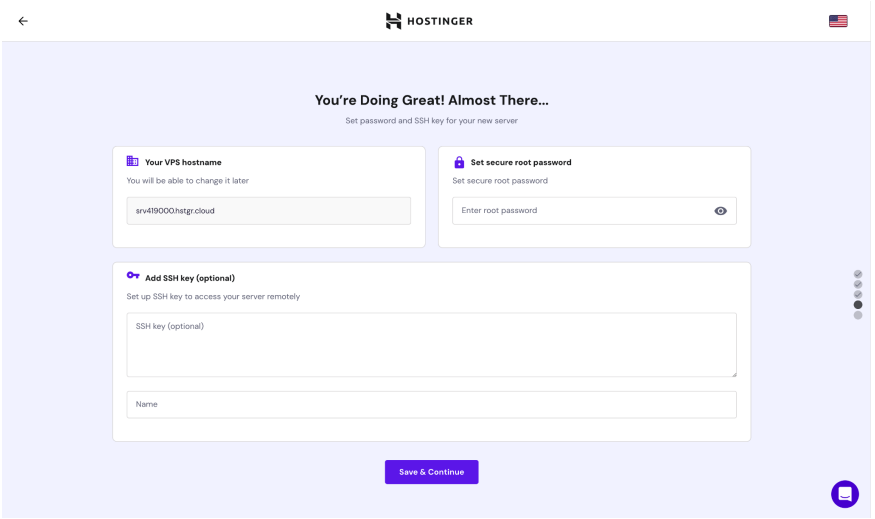
c. Select CloudPanel with Ubuntu



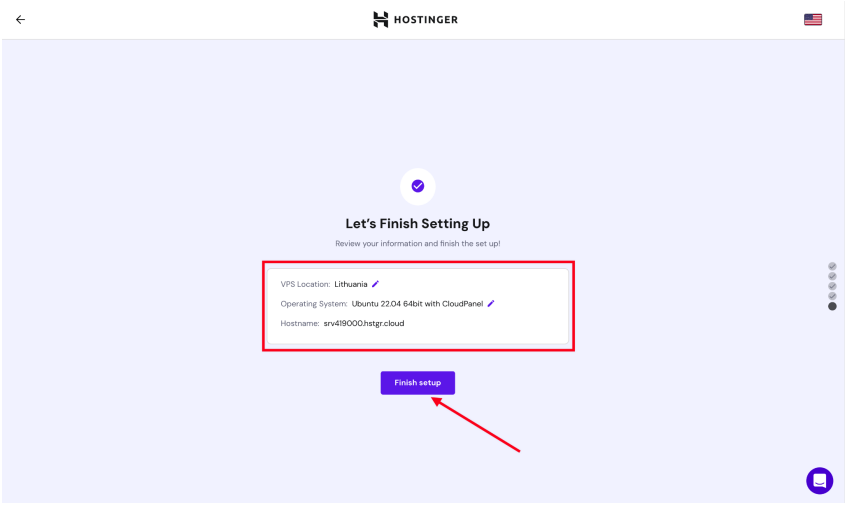
d. Set a Panel Password (SAVE this password for later, it's very important)



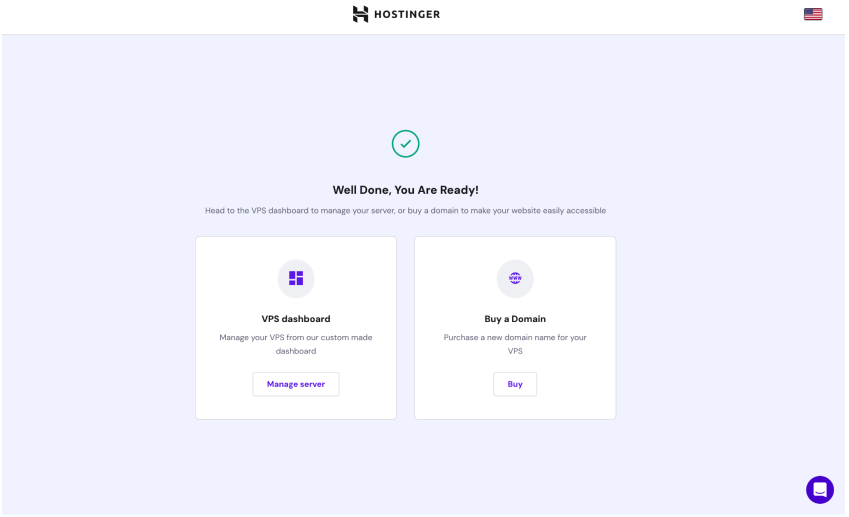
e. Set a secure root password (used for SSH steps later on) and then Next.



f. Click Finish Setup

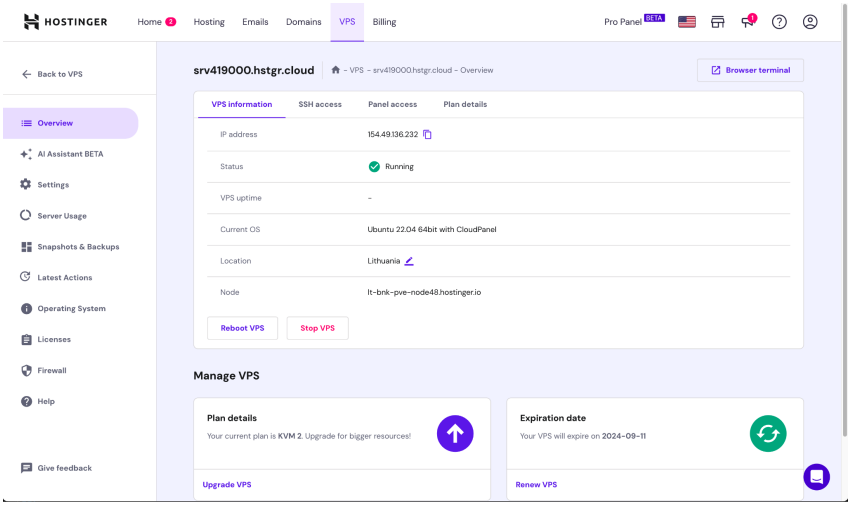


g. Wait for the Server to be initialised and set up - then click Manage Server!



h. Copy the IP address in the VPS dashboard

i.



2. Install **Ubuntu 22.04 64bit** template on your VPS (Above steps).

3. Connect to it through **SSH (You can do this through the regular terminal however I recommend Warp Terminal https://www.warp.dev/)**:

Replace your_server_ip with the IP address located in your dashboard

```
Unset
ssh root@your_server_ip
```

Shell Command Explained:

`ssh`: This is the command to initiate an SSH connection. SSH is a secure protocol for connecting to and managing remote servers.

`root`: This is the username you want to use when connecting to the remote server. In this case, it's set to "root." The "root" user typically has administrative privileges on a Unix/Linux system, which means it can perform system-level tasks and make changes that other users may not be allowed to do.

`@`: This symbol separates the username from the hostname (or IP address) of the remote server.

`your_server_ip`: This is a placeholder for the actual IP address of the server you want to connect to. In the command provided, it instructs you to replace "your_server_ip" with the actual IP address of the server where you want to establish an SSH connection. The IP address is the unique identifier of the server on the internet or within a local network.

So, when you run this command by replacing "your_server_ip" with the real IP address of your server, it will establish an SSH connection to that server with the "root" user, provided that you have the necessary credentials (such as the password or SSH key) for authentication.

4. Remove Apache2 and install reverse proxy and node.js server:
    a. You may get a message asking you to wait for 60 seconds, in this time you must wait for the terminal to pass 60 seconds - this is due to a deprecated package being used in the process.

Unset

```
sudo apt-get purge apache2*

curl -sL https://deb.nodesource.com/setup_20.x -o nodesource_setup.sh

sudo bash nodesource_setup.sh

sudo apt update && sudo apt upgrade -y
```

Shell Command Explained:

`sudo apt-get purge apache2*`
This command is used to remove (purge) any packages related to Apache HTTP Server (`apache2`). The asterisk (*) is a wildcard character that matches any package name starting with "apache2." This step ensures that Apache is not installed or is completely removed from the system.

`curl -sL https://deb.nodesource.com/setup_20.x -o nodesource_setup.sh`
This command uses `curl` to download a script from the specified URL (`https://deb.nodesource.com/setup_20.x`) and saves it as a file named `nodesource_setup.sh` in the current directory. The script is provided by NodeSource, a repository for Node.js packages. It's used to add Node.js repository information to the system.

`sudo bash nodesource_setup.sh`
This command executes the `nodesource_setup.sh` script with elevated privileges (sudo). The script adds the Node.js repository to the system's package sources, making Node.js packages available for installation.

`sudo apt update && sudo apt upgrade -y`
These two commands are combined using `&&` to update the package lists (`sudo apt update`) and upgrade the installed packages (`sudo apt upgrade -y`). The `-y` flag automatically confirms the upgrade without requiring user interaction.

5. Create a new Nginx configuration file for your Next.js application:

```
Unset
sudo nano /etc/nginx/sites-available/nextjs.conf
```

Shell Command Explained:

`sudo`: This command executes the subsequent command with superuser (root) privileges. This is typically required because modifying Nginx configuration files often requires administrative access.

`nano`: This is a text editor, and it is being used to create and edit the Nginx configuration file. The `nano` command will open a text editor in the terminal, allowing you to create and edit the file interactively.

`/etc/nginx/sites-available/nextjs.conf`: This part of the command specifies the path and filename for the Nginx configuration file you are creating. Here's what it means:

`/etc/nginx/`: This is the directory where Nginx stores its configuration files on many Linux systems. It's the standard location for Nginx configuration files.

`sites-available/`: This is a subdirectory where you typically store configuration files for different websites or web applications that you want to configure with Nginx.

`nextjs.conf`: This is the name you are giving to your Next.js application's Nginx configuration file. You can choose any name you like, but it's common to use a descriptive name related to the application you're configuring.

So, when you run this command, it will open the `nano` text editor with an empty file located at `/etc/nginx/sites-available/nextjs.conf`, allowing you to create or edit the Nginx configuration specifically for your Next.js application. You would typically add configuration directives to this file to define how Nginx should handle requests for your Next.js application, such as setting up the server block, specifying the domain or IP address, and defining proxy rules if needed.

6. Paste the following configuration, replacing your_server_ip with your domain name or VPS IP address:

Unset

```
server {
 listen 80;
 server_name your_server_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;
 }
}
```

Shell Command Explained:

`server { ... }`: This block of configuration defines an Nginx server block, which specifies how Nginx should handle requests for a specific domain name or IP address.

`listen 80;`: This directive tells Nginx to listen on port 80, which is the default HTTP port.

`server_name your_server_ip;`: Here, `server_name` specifies the domain name or IP address for which this server block should handle requests. The configuration instructs you to replace `your_server_ip` with your actual domain name or VPS IP address.

`location / { ... }`: This block defines how Nginx should handle requests to the root path ("/"). It sets up a reverse proxy to pass requests to a local service running on port 3000.

`proxy_pass http://localhost:3000;`: This directive specifies that incoming requests to this location should be proxied to the service running on `http://localhost:3000`. This is commonly used when you have a web application or API running on a different port, such as a Node.js application.

`proxy_http_version 1.1;`: It sets the HTTP version for the proxy communication.

`proxy_set_header Upgrade $http_upgrade;`, `proxy_set_header Connection 'upgrade';`, `proxy_set_header Host $host;`: These directives are used to pass certain headers from the client request to the proxied server. They are often used to handle WebSockets and maintain host information.

`proxy_cache_bypass $http_upgrade;`: This directive is typically used to bypass caching for requests that involve upgrades, such as WebSocket connections.

In summary, this configuration sets up an Nginx server block to handle requests to a specific domain or IP address, forwarding those requests to a service running on `localhost:3000`. It's a common configuration for hosting web applications and APIs behind Nginx, providing a reverse proxy and handling various headers for proper communication. You should replace `your_server_ip` with the actual domain name or IP address you want to use for your application.

Save and close the file.

7. Create a symbolic link to enable the configuration (This is a crucial step):

Unset

```
sudo ln -s /etc/nginx/sites-available/nextjs.conf
/etc/nginx/sites-enabled/

rm /etc/nginx/sites-enabled/default.conf
```

Shell Command Explained:

`sudo ln -s /etc/nginx/sites-available/nextjs.conf /etc/nginx/sites-enabled/`

`sudo`: This command is used to execute the following command with superuser (root) privileges.

`ln -s`: This command is used to create a symbolic link.

`/etc/nginx/sites-available/nextjs.conf`: This is the source file for the symbolic link. It points to the Nginx configuration file for your Next.js application, which you likely created or modified earlier.

`/etc/nginx/sites-enabled/`: This is the destination directory where the symbolic link is being created. In Nginx, this directory typically contains symbolic links to the configuration files that should be active and used by Nginx.

So, this command creates a symbolic link named "nextjs.conf" in the `/etc/nginx/sites-enabled/` directory that points to your Next.js application's configuration file in `/etc/nginx/sites-available/`.

`rm /etc/nginx/sites-enabled/default.conf`

`rm`: This is the command to remove (delete) a file.

`/etc/nginx/sites-enabled/default.conf`: This command deletes the default Nginx site configuration file. On some Linux distributions, there's a default configuration file that might interfere with your custom configurations. Removing it ensures that only your custom site configuration (in this case, "nextjs.conf") is used.

In summary, these commands create a symbolic link to enable your custom Nginx configuration for your Next.js application and remove the default Nginx configuration, which might conflict with your custom setup. This is a common sequence of steps when configuring Nginx to host web applications or websites.

8. Restart Nginx:

```
Unset
sudo service nginx restart
```

Shell Command Explained:

`sudo`: This command is used to execute the following command with superuser (root) privileges. Restarting Nginx often requires administrative privileges because it involves managing system services.

`service nginx restart`: This part of the command instructs the system's service management system to restart the Nginx service. Here's what it does:

`service`: This command is used to interact with system services on many Linux distributions.

`nginx`: This specifies the name of the service you want to manage, in this case, the Nginx web server.

`restart`: This is the action you want to perform on the Nginx service. It means stopping the Nginx service (if it's running) and then starting it again.

So, when you run this command with superuser privileges, it will gracefully stop and then start the Nginx web server. Restarting Nginx is often necessary when you make changes to its configuration files to apply the new settings without completely stopping and starting the server.

9. Create a new Next.js application and follow the prompts:

```
Unset
mkdir /var/www/nextjs

cd /var/www/nextjs
```

Shell Command Explained:

`mkdir /var/www/nextjs`

`mkdir`: This is the "make directory" command, used to create a new directory.

`/var/www/nextjs`: This is the path where the new directory will be created. In this case, it's creating a directory named "nextjs" within the "/var/www" directory. This path is often used for hosting web applications.

So, this command creates a new directory named "nextjs" at the location "/var/www/nextjs."

`cd /var/www/nextjs`

`cd`: This is the "change directory" command, used to navigate to a different directory.

`/var/www/nextjs`: This is the directory path you want to navigate to. By running this command, you change your current working directory to "/var/www/nextjs."

In summary, these commands are preparing the environment for creating and working with a Next.js application. The first command creates a directory to host the application, and the second command navigates into that directory so that you can start working on your Next.js project within the "/var/www/nextjs" directory.

10. Install GitHub agent and login to GitHub (optional)

Unset

```
apt install gh
gh auth login
```

Shell Command Explained:

`apt install gh`

`apt`: This is the package management command on Debian-based Linux distributions like Ubuntu. It is used to install, upgrade, or remove software packages.

`install`: This is the subcommand that tells `apt` to install a package.

`gh`: This is the package name for the GitHub CLI tool.

So, this command installs the GitHub CLI (`gh`) on your system using the `apt` package manager.

`gh auth login`

`gh`: This is the GitHub CLI tool you installed in the previous step.

`auth login`: This is a subcommand provided by `gh` that initiates the GitHub authentication process, allowing you to log in to your GitHub account.

When you run this command, it will prompt you to authenticate with your GitHub account. The authentication process may involve opening a web browser, signing in to your GitHub account, and granting permissions to the GitHub CLI tool. Once authenticated, the CLI tool will store an authentication token locally, allowing you to interact with your GitHub repositories and perform various GitHub-related tasks from the command line.

In summary, these commands install the GitHub CLI tool on your system and then use it to log in to your GitHub account, enabling you to work with GitHub repositories and perform actions via the command line interface.

Follow the prompts, and when asked, copy the one-time code and open a browser http://github.com/login/device

11. Install node version manager

```
Unset
apt install npm
npm install -g n
n latest
```

Shell Command Explained:

`apt install npm`

`apt`: This is the package management command on Debian-based Linux distributions like Ubuntu. It is used to install, upgrade, or remove software packages.

`install`: This is the subcommand that tells `apt` to install a package.

`npm`: This is the Node.js package manager, which is used to manage Node.js packages and modules.

So, this command installs the Node Package Manager (npm) on your system.

`npm install -g n`

`npm`: This is the Node Package Manager.

`install -g n`: This command uses `npm` to install the "n" package globally ("-g"). The "n" package is a popular Node.js version manager.

After running this command, you'll have the "n" version manager installed on your system, which allows you to manage different versions of Node.js.

`n latest`

`n`: This is the Node.js version manager you installed in the previous step.

`latest`: This is a subcommand for "n" that instructs it to install the latest stable version of Node.js.

When you run this command, "n" will download and install the most recent stable release of Node.js. This ensures that you have an up-to-date version of Node.js on your system.

In summary, these commands install the Node Package Manager (npm) for managing Node.js packages, then install the "n" package globally, and finally use "n" to install the latest stable version of Node.js. This setup allows you to easily switch between different versions of Node.js on your Linux system as needed.

12. Initialise a new next.js app:

```
Unset
npx create-next-app@latest mynewapp
```

Shell Command Explained:

`npx`: `npx` is a package runner tool that comes with npm (Node Package Manager). It allows you to run packages that are not globally installed on your system directly from the npm registry. In this case, you're using it to run the `create-next-app` package.

`create-next-app@latest`: This part of the command specifies the package you want to run using `npx`. `create-next-app` is the package name, and `@latest` ensures that the latest version of this package is used.

`mynewapp`: This is the argument you provide to `create-next-app`. It is the name of the new Next.js application you are creating. You can replace "mynewapp" with your preferred name for your application.

When you run this command, it does the following:

It uses `npx` to fetch the latest version of the `create-next-app` package from the npm registry (if it's not already installed locally).

It runs `create-next-app` with the specified application name ("mynewapp" in this case).

The generator will then scaffold a new Next.js application in a directory named "mynewapp" (or whatever name you specified) with the default project structure, including necessary files and folders to get you started with Next.js development.

This command is a convenient way to initialize a new Next.js project without needing to install global packages or manage dependencies manually. It ensures that you are using the latest version of the `create-next-app` package and sets up a basic Next.js project for you to start building upon.

13. You can run the dev environment of the app:

```
Unset
npm run dev
```

Your Next.js application is now deployed and accessible at your domain name or VPS IP address. To keep your application running in the background and automatically restart on crashes or server reboots, you should use a process manager like PM2

Shell Command Explained:

`npm run dev`
This command is using npm (Node Package Manager) to run a script named "dev" defined in the "scripts" section of the application's package.json file.

`npm`: This is the Node Package Manager, a tool used for managing JavaScript packages and running scripts defined in a package.json file.

`run`: This is a command used with npm to execute a script defined in the package.json file.

`dev`: This is the name of the script you want to run. In this context, it typically starts the development environment of the application.

When you run `npm run dev`, it executes the specified script, which often involves starting a development server, compiling code, and performing other tasks required to run the application in a development environment.

The behaviour of this command can vary depending on how the "dev" script is configured in the package.json file of the project. In a Next.js project, for example, running `npm run dev` usually starts a local development server that serves the Next.js application, enables hot reloading, and provides tools for debugging and development.

In summary, `npm run dev` is a convenient way to start the development environment for a Node.js application, allowing developers to work on and test their code as they build and develop the application.

14. If you want to build your app in production mode

```
Unset
npm build
npm start
```

Shell Command Explained:

`npm build`
This command is using npm (Node Package Manager) to run a script named "build" defined in the "scripts" section of the application's package.json file.

`npm`: This is the Node Package Manager, a tool used for managing JavaScript packages and running scripts defined in a package.json file.

`build`: This is the name of the script you want to run. In the context of most Node.js applications, running the "build" script typically involves tasks such as compiling source code, optimizing assets, and preparing the application for production deployment.

`npm start`
This command is also using npm to run a script named "start" defined in the package.json file.

`start`: This is the name of the script you want to run when starting the application in production mode. In many Node.js applications, running "start" typically involves starting the application server with optimized production settings.

When you run `npm build`, it executes the "build" script to prepare your application for production. This may involve tasks like minifying JavaScript and CSS, bundling assets, and creating production-ready build artifacts.

After successfully running `npm build`, you can then use `npm start` to start your application in production mode. This command usually starts a server that serves the optimized and built version of your application, making it available to users in a production environment.

15. To install PM2 globally on your VPS:

```
Unset
cd ~

sudo npm install -g pm2
```

Shell Command Explained:

`cd ~`
This command is used to change the current directory to the user's home directory (`~` is a shorthand representation for the home directory).

`cd`: This is the "change directory" command.

`~`: This symbol represents the user's home directory.

Changing the directory to the home directory ensures that the subsequent command is executed from the user's home directory.

`sudo npm install -g pm2`

`sudo`: This command is used to execute the following command with superuser (root) privileges. Installing software globally on a server often requires administrative access.

`npm install -g pm2`: This part of the command uses npm (Node Package Manager) to install the PM2 package globally ("-g" flag). Here's what it does:

`npm`: This is the Node Package Manager, a tool for managing JavaScript packages and modules.

`install`: This is the command used to install a package.

`-g`: This flag stands for "global," indicating that PM2 should be installed globally on the system, making it available as a command-line tool for all users.

`pm2`: This is the package name for PM2, which is a process manager for Node.js applications. PM2 is used to manage and monitor Node.js processes, making it particularly useful for running Node.js applications in production environments.

So, when you run these commands on your VPS, it navigates to the home directory and then uses `sudo` to install PM2 globally on the server. Once installed, you can use the `pm2`

command to manage Node.js applications, start and stop processes, and monitor their performance. PM2 is a popular choice for deploying and managing Node.js applications in a production environment.

16. Navigate to the Next.js application directory (if not already there):

```
Unset
cd /var/www/nextjs
```

Shell Command Explained:

`cd`: This is the "change directory" command in Unix-like operating systems, including Linux. It's used to navigate between directories.

`/var/www/nextjs`: This is the directory path you want to navigate to. In this case, you are specifying the absolute path to the directory where your Next.js application is stored. Absolute paths start from the root directory ("/") and provide the full path to the target directory.

So, when you run this command, it changes the current working directory to "/var/www/nextjs," assuming that this is where your Next.js application is located. This is useful because you can then run other commands or perform tasks within the context of your application's directory. It's important to navigate to the correct directory before performing actions related to your application to ensure that the commands operate on the right files and folders.

17. Start the Next.js application using PM2:

```
Unset
pm2 start npm --name "nextjs" -- start
```

This command will start the Next.js application with the name “nextjs” using the npm start command. PM2 will automatically restart the application if it crashes or if the server reboots.

Shell Command Explained:

```
pm2 start npm --name "nextjs" -- start
```

`pm2`: This is the PM2 command-line utility used to manage and run Node.js processes.

`start`: This is a subcommand of PM2, indicating that you want to start a new Node.js process.

`npm`: This part specifies that you want to run the `npm` command. In this context, PM2 is acting as a process manager for the `npm` command, which is often used to start Node.js applications.

`--name "nextjs"`: This option sets a name for the PM2 process. In this case, it's named "nextjs." Naming the process is useful for managing and identifying it later.

`-- start`: This part of the command specifies the arguments that should be passed to the `npm` command. In this case, it's telling `npm` to run the "start" script defined in your application's package.json file. The "start" script is typically used to start your Next.js application.

When you run this command, PM2 will start your Next.js application with the name "nextjs." PM2 will monitor the application and automatically restart it if it crashes or if the server reboots. It provides a robust way to manage and keep Node.js applications running in production environments, ensuring high availability and stability.

18. To ensure PM2 starts on boot, run:

Unset

```
pm2 startup
```

This command will generate a script that you can copy and paste into your terminal to enable PM2 to start on boot.

Shell Command Explained:

`pm2 startup`
This is the PM2 command used to configure PM2 to start automatically on system boot.

When you run `pm2 startup`, PM2 will generate a script that you can copy and paste into your terminal. This script is specific to your system's init system (e.g., systemd, Upstart, SysV, etc.) and is designed to configure the init system to start PM2 on system boot.

The generated script typically contains commands that add PM2 as a startup service, ensuring that PM2 and the applications managed by PM2 are automatically started when your server restarts or boots up. The exact content of the script may vary depending on your system's init system, so it's important to use the script generated by `pm2 startup` for your specific environment.

By running this command and following the instructions provided, you can ensure that PM2 manages your Node.js applications reliably, even after server reboots or system crashes, making it suitable for production deployments.

19. Save the current PM2 processes:

Unset

```
pm2 save
```

Shell Command Explained:

`pm2 save`
This is a PM2 command that instructs PM2 to save the current state of running processes, their configurations, and startup settings into a file.

When you run `pm2 save`, PM2 will create or update a JSON file (usually named `pm2.json` or `processes.json`) that contains information about your currently

managed processes, including details like their names, script paths, and configuration settings. This file is typically stored in the PM2 configuration directory.

Saving the current state of PM2 processes is useful for a few reasons:

Restoring State: You can use the saved configuration to restore your PM2 processes exactly as they were after a server reboot or application update.

Managing Startup: It helps configure PM2 to start the specified processes automatically when the server restarts.

Sharing Configurations: You can share the configuration file with other team members or across multiple servers to ensure consistent process management.

In summary, `pm2 save` is a command to save the current state of PM2-managed processes and configurations, allowing you to easily restore them or ensure they start automatically on server boot.