

## IndusSearch Engine Documentation

---

Indus Search Engine, developed by Endurasolution, is built on Next.js, Typesense, and Node.js. This documentation provides all the necessary configurations for a more stable installation.

### Requirements

- Two or Three Server (if typesense and site install same server then Two server need)
- Ubuntu latest or above 20 version need
- Typesense latest version installed
- Optional (try to use best latency server and 1Gbps port speed)
- Server minimum 8gb ram, 4cpu, 80gb ssd

### Installation

First We need to install Typesense on ubuntu server

To install **Typesense** on an Ubuntu server, you can use the official Typesense apt repository. Here are the commands to run in your terminal.

#### Step 1: Add the Typesense apt repository

First, add the **GPG key** and the repository to your system's software sources.

```
curl -O https://download.typesense.com/typesense-api/typesense-api-key-2023.pub && \
sudo mv typesense-api-key-2023.pub /usr/share/keyrings/typesense-api-key-2023.pub && \
echo "deb [signed-by=/usr/share/keyrings/typesense-api-key-2023.pub]
https://download.typesense.com/typesense-api/ubuntu $(lsb_release -sc) main" | sudo tee
/etc/apt/sources.list.d/typesense.list
```

---

#### Step 2: Update your package list

Next, update your package list to include the newly added repository.

```
sudo apt-get update
```

---

### Step 3: Install Typesense

Now, install the **Typesense server** package.

```
sudo apt-get install typesense-server
```

---

### Step 4: Configure and start Typesense

Typesense is automatically started and runs as a service after installation. The configuration file is located at `/etc/typesense/typesense-server.conf`. You can edit this file to change settings like the **API key** or data directory.

You can check the status of the Typesense service with this command:

```
sudo systemctl status typesense-server
```

If you need to start, stop, or restart the service, use the following commands:

```
sudo systemctl start typesense-server  
sudo systemctl stop typesense-server  
sudo systemctl restart typesense-server
```

When Typesense is installed on Ubuntu via the DEB package, the **admin API key** is automatically generated and can be found in the configuration file.

### Finding the API Key

To view the contents of the configuration file, use the following command in your terminal:

```
sudo cat /etc/typesense/typesense-server.conf
```

Inside this file, you'll find a line that looks like this:

```
api-key = <your_api_key_here>
```

The string of characters after `api-key =` is the **admin API key**. This key has full administrative permissions, so it's important to keep it secure. The Typesense documentation recommends using this key only to create more specific, scoped keys for your applications and not to use it directly in a production environment.

## Next Nginx Install with reverse proxy to this typesense

Here is how you can install **Nginx** and configure it as a **reverse proxy** for Typesense on an Ubuntu server.

### Step 1: Install Nginx

First, update your package index and then install Nginx using `apt`.

```
sudo apt update  
sudo apt install nginx
```

After the installation is complete, Nginx will automatically start running as a service. You can check its status to confirm it's active.

```
sudo systemctl status nginx
```

---

### Step 2: Configure the Nginx Reverse Proxy

You need to create a new Nginx server block configuration file to act as the reverse proxy for Typesense.

**Navigate to the sites-available directory:**

```
cd /etc/nginx/sites-available/
```

**Create a new configuration file.** You can name it something descriptive like `typesense.conf`.

```
sudo nano typesense.conf
```

1.

**Add the reverse proxy configuration.** Paste the following code into the file. Be sure to replace `your_domain_or_ip` with your server's domain name or public IP address. Typesense, by default, runs on port `8108`.

```
server {
    listen 80;
    server_name your_domain_or_ip;

    location / {
        proxy_pass http://localhost:8108;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}
```

2. This configuration tells Nginx to:

- **listen** on port 80 for HTTP requests.
- **server\_name** identifies the domain or IP it should respond to.
- **location /** directs all incoming requests to the root of your domain.
- **proxy\_pass http://localhost:8108;** is the most important part, as it forwards those requests to the Typesense server running on the local machine on port 8108.
- **proxy\_set\_header** directives ensure the original client's IP and other important information are passed along to Typesense.

3. **Save and close the file.** In `nano`, press `Ctrl + X`, then `Y`, and then `Enter`.

---

## Step 3: Enable the Configuration

To activate your new configuration, you need to create a symbolic link from the `sites-available` directory to the `sites-enabled` directory.

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

---

## Step 4: Test and Restart Nginx

Before restarting the service, it's a good practice to test the Nginx configuration for syntax errors.

```
sudo nginx -t
```

If the output shows **syntax is ok** and **test is successful**, you can safely restart Nginx to apply the changes.

```
sudo systemctl restart nginx
```

Now, all HTTP requests to your server's domain or IP address on port 80 will be routed to your Typesense instance running on port 8108.

## SSL Adding

Adding a free SSL certificate from Let's Encrypt to your Nginx reverse proxy is a standard and highly recommended process. The tool to automate this is **Certbot**.

Here's a step-by-step guide to get it done on your Ubuntu server.

### Prerequisites

- You have a domain name registered (e.g., **example.com**).
- Your domain's DNS records are pointing to the public IP address of your server.
- You have Nginx installed and configured with a server block for your domain on port 80 (as you did in the previous step).

## Step 1: Install Certbot and the Nginx plugin

The **certbot** package and its Nginx plugin are available in Ubuntu's default repositories.

```
sudo apt update
sudo apt install certbot python3-certbot-nginx
```

---

## Step 2: Ensure Nginx configuration is correct

Certbot needs to be able to find the `server_name` directive in your Nginx configuration file to correctly configure the SSL certificate.

Open your Nginx configuration file for Typesense:

```
sudo nano /etc/nginx/sites-available/typesense.conf
```

Make sure the `server_name` line is correctly set to your domain name. It should look like this:

```
server_name example.com www.example.com;
```

If you made any changes, save the file (**Ctrl + X**, then **Y**, then **Enter**) and test your Nginx configuration syntax.

```
sudo nginx -t
```

If the test is successful, reload Nginx to apply the changes.

```
sudo systemctl reload nginx
```

---

## Step 3: Allow HTTPS traffic through the firewall

If you have a firewall like UFW enabled, you need to allow HTTPS traffic on port 443. Nginx registers a few profiles with UFW during installation.

**Check your current UFW status:**

```
Bash
sudo ufw status
```

1. You'll likely see **Nginx HTTP** is allowed.

**Allow the Nginx Full profile** (which includes both HTTP and HTTPS traffic) and remove the old rule.

```
sudo ufw allow 'Nginx Full'  
sudo ufw delete allow 'Nginx HTTP'
```

2. Your status should now show **Nginx Full** is allowed.

---

## Step 4: Obtain and Install the SSL Certificate

Now you can run Certbot with the Nginx plugin to automatically obtain and install the certificate.

Bash

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

- **--nginx**: This flag tells Certbot to use the Nginx plugin to modify the configuration file.
- **-d example.com -d www.example.com**: This specifies the domains you want the certificate for.

When you run this command for the first time, Certbot will prompt you for a few things:

- An **email address** for urgent renewal notices.
- To **agree to the Terms of Service**.
- Whether to **redirect HTTP traffic to HTTPS**. It is highly recommended to choose the option to redirect all traffic.

Certbot will then automatically handle the domain verification, obtain the certificate, modify your Nginx configuration file to use HTTPS, and set up automatic renewal.

## Step 5: Verify automatic renewal

Certbot automatically creates a systemd timer or cron job to renew your certificates before they expire. You can test this process with a "dry-run" to ensure it's working correctly.

Bash

```
sudo certbot renew --dry-run
```

If the dry run completes without errors, your certificates will renew automatically in the background.

After these steps, your Typesense API will be accessible securely over HTTPS.

## Now the Typesense is installed ,Next is Main Website install

To set up your Next.js project with Typesense and manage your environment variables, you'll need to follow a few steps, including installing Node.js, creating the Next.js app, and configuring the `.env.example` file.

---

### Step 1: Install Node.js

First, ensure you have **Node.js** installed on your server, as it's a prerequisite for running Next.js and npm. Using a version manager like **NVM (Node Version Manager)** is recommended as it allows you to easily switch between different Node.js versions.

To install NVM on Ubuntu, run this command:

```
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.7/install.sh | bash
```

After the installation is complete, close and reopen your terminal, or run `source ~/.bashrc` to activate NVM.

Now, you can install the latest stable version of Node.js and npm:

```
nvm install --lts  
nvm use --lts
```

---

### Step 2: Upload a Next.js App Script

Next, upload the app source code to server any folder is best option example i upload the source to folder search then go to the folder using command

```
cd /search
```

---

### Step 3: Install the Package



npm install

---

## Step 4: Configure the `.env` file

The `.env.example` file you provided is a template for your project's configuration. You need to create a new file named `.env` and populate it with your specific details. **Never commit the `.env` file to version control (Git)**, as it contains sensitive information.

**Edit the file the file:**

```
nano .env.example .env
```

**Edit the `.env` file:** Open the `.env` file and replace the placeholder values with your actual configuration details.

```
# .env
```

```
# Typesense Configuration
```

```
TYPESENSE_HOST=your-typesense-domain.com # Use the domain you set up with Nginx and Certbot
```

```
TYPESENSE_PORT=443 # Use port 443 for HTTPS
```

```
TYPESENSE_PROTOCOL=https # Use https for a secure connection
```

```
TYPESENSE_API_KEY=your_generated_api_key_from_typesense_server_conf
```

```
# YouTube API (for video search)
```

```
YOUTUBE_API_KEY=your-youtube-api-key
```

```
# Next.js Configuration
```

```
NEXT_PUBLIC_APP_URL=https://your-domain.com # Change this to your live domain URL
```

```
# SMTP Email Configuration
```

```
SMTP_HOST=smtp.gmail.com
```

```
SMTP_PORT=587
```

```
SMTP_SECURE=auto
```

```
SMTP_USER=your-email@gmail.com
```

```
SMTP_PASS=your-app-password # Use an app password, not your regular email password
```

```
SMTP_FROM=noreply@indussearch.com
```

```
# Razorpay Payment Configuration
RAZORPAY_KEY_ID=your-razorpay-key-id
RAZORPAY_KEY_SECRET=your-razorpay-key-secret
```

```
ALLOW_ADMIN_REGISTER=true
```

1. Next.js automatically exposes variables prefixed with `NEXT_PUBLIC_` to the client-side, while keeping other variables secure on the server. The `NEXT_PUBLIC_APP_URL` variable is correctly set up for this.

---

## Step 5: Start the Next.js Development Server

After setting up the `.env` file, you can start your Next.js development server to begin building your application.

```
npm run dev
```

This will demo run the app on server ip with port

If server ip is 192.25.35.2 then port is 3000

Then test with <http://192.25.35.2:3000>

Configuring **Nginx** for a Next.js application is a crucial step for production deployment. The most common setup is to use Nginx as a reverse proxy, which forwards requests to your running Next.js application, typically on `localhost:3000`. This also allows you to handle SSL (HTTPS), load balancing, and serve static assets more efficiently.

## Step 1: Build your Next.js application

Before you can run the application, you need to build it for production. Navigate to your project directory and run the following command:

```
npm run build
```

This command creates a `.next` folder with the optimized production build.

## Step 2: Start your Next.js server

Next.js needs to be running in the background for Nginx to proxy requests to it. A tool like **PM2** is highly recommended to manage the Node.js process and ensure it stays online, restarting automatically if it crashes.

### Install PM2 globally:

```
sudo npm install -g pm2
```

1.

### Start your Next.js application with PM2:

```
pm2 start npm --name "indus-search" -- start
```

2. Replace "**indus-search**" with a name for your application. This command will start the application and keep it running in the background. You can check its status with **pm2 status**.

### Ensure PM2 starts on reboot:

```
pm2 startup  
pm2 save
```

---

## Step 3: Configure Nginx as a Reverse Proxy

You'll create a new Nginx server block configuration for your Next.js application.

### Create a new configuration file:

```
sudo nano /etc/nginx/sites-available/indus-search.conf
```

1. You can replace **indus-search.conf** with your domain name.

**Add the configuration.** Paste the following code into the file. Be sure to replace **your\_domain.com** with your actual domain.

```
Nginx  
server {  
    listen 80;  
    server_name your_domain.com www.your_domain.com;
```

```

location / {
    proxy_pass http://localhost:3000;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection 'upgrade';
    proxy_set_header Host $host;
    proxy_cache_bypass $http_upgrade;
}

# Optimized static file serving
location /_next/static/ {
    alias /path/to/your/nextjs-app/.next/static/;
    expires 1y;
    access_log off;
}
}

```

2. **Important:** You must replace `/path/to/your/nextjs-app/` with the actual absolute path to your Next.js project directory on the server.
3. **Save and close the file** (`Ctrl + X, Y, Enter`).

---

## Step 4: Enable the Configuration

Create a symbolic link to enable your new configuration file.

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

---

## Step 5: Test and Restart Nginx

Finally, test the Nginx configuration for syntax errors and restart the service to apply the changes.

```

sudo nginx -t
sudo systemctl restart nginx

```

Your Next.js app will now be running and configured to communicate with your Typesense instance via your secure Nginx reverse proxy.

Cerbot also same ssl install for the domain also. Use that same command add own search engine domain.

## Web Crawler Install

Note:- Make sure the web crawler need a fresh separate server because it take more memory,

Then install the Nodejs using

To install NVM on Ubuntu, run this command:

```
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.7/install.sh | bash
```

After the installation is complete, close and reopen your terminal, or run `source ~/.bashrc` to activate NVM.

Now, you can install the latest stable version of Node.js and npm:

```
nvm install --lts  
nvm use --lts
```

Then upload the web bot script file then run this command

```
npm install
```

Then try to run the [app.js](#) for the webcrawler ,and [image.js](#) for image crawling,and [imagenon.js](#) is same image crawl but need to install tensorflow

To make the [app.js](#) and [image.js](#) run in background use pm2

(before run install pm2 using “sudo npm install -g pm2”)

Example if pm2 installed then run

```
pm2 start app.js like pm2 start image.js
```

Also make sure .env add the typesense details correctly.

And RESPECT\_ROBOTS must be true , follow all urls robots.txt.

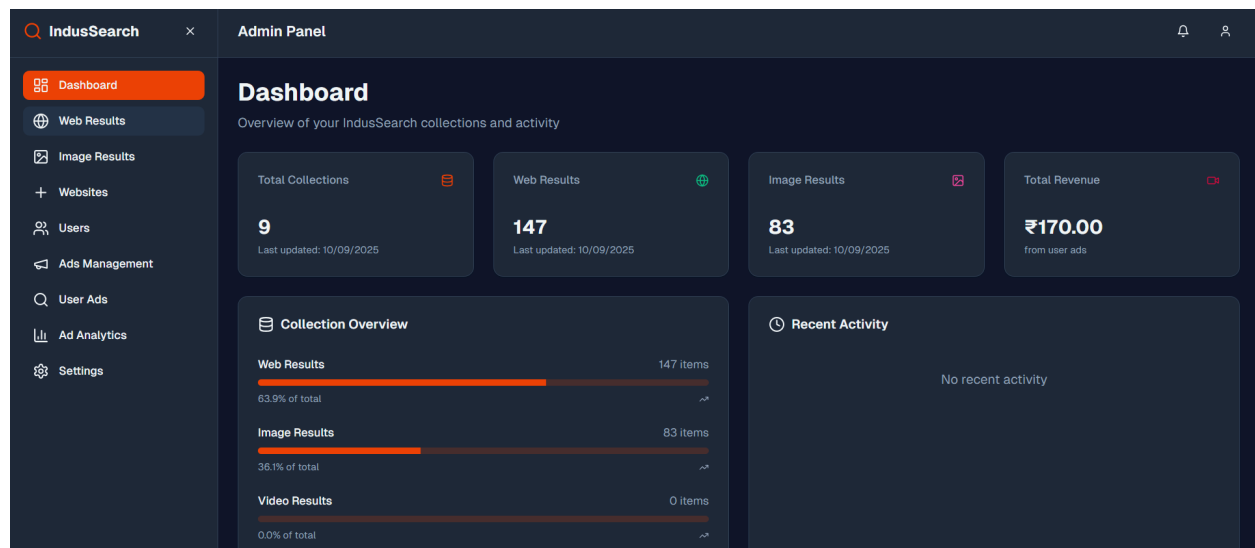
Thats all done .

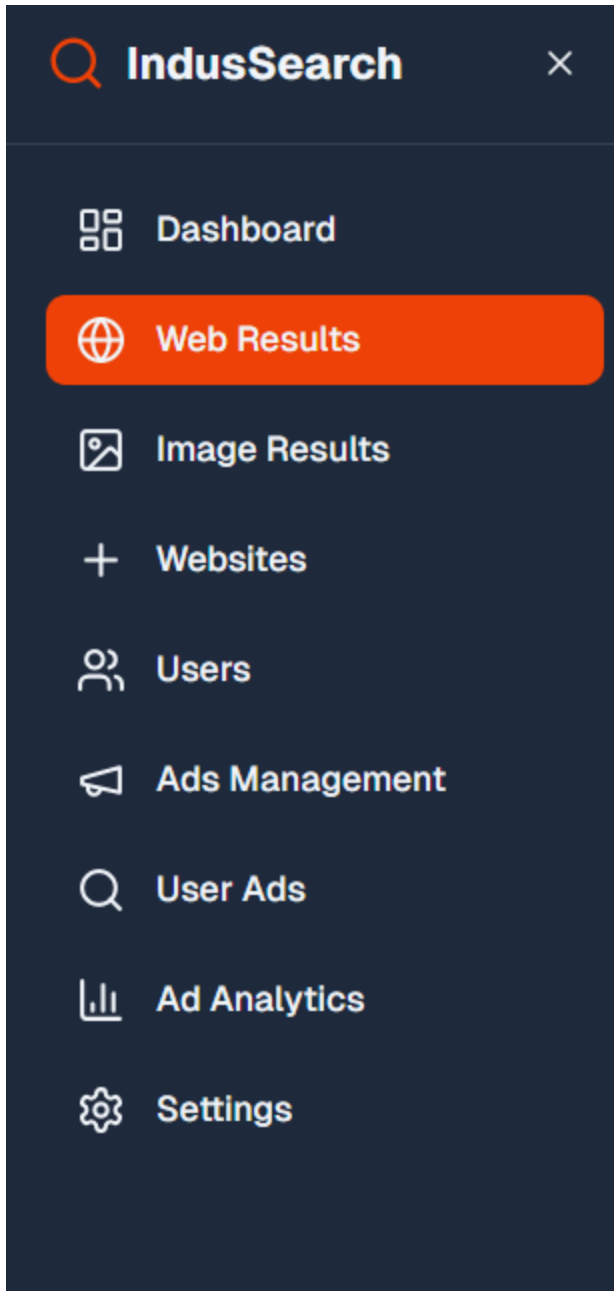
## Configure Website details

Log in to search engine url like (<https://example.com/webadmin/auth>)

Make sure ALLOW\_ADMIN\_REGISTER should be true then first time you can see a register option to create admin username and password then change the ALLOW\_ADMIN\_REGISTER to false.

After success then go to login then login to see the dashboard, here all status can see.





Web Result :- Manage all Websites Results

Image Results :- Manage all image Results

Website:- Add websites to crawler

User:-Manage all users

AdsManager:- Manage admin created ads

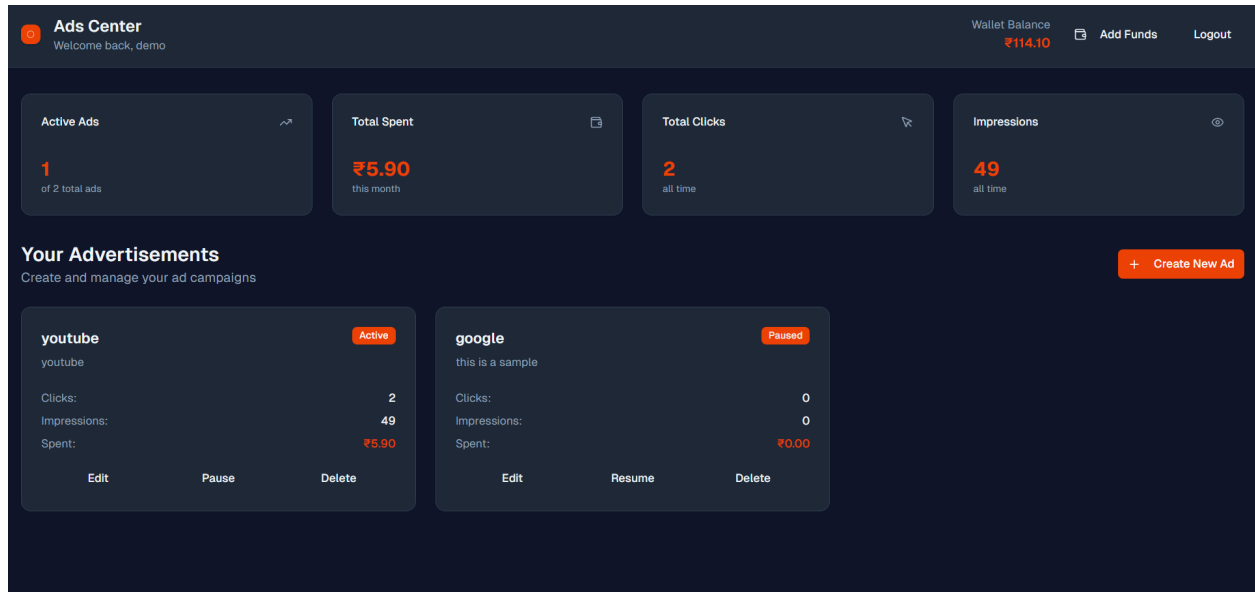
User Ads:- Manage user created ads

Ad analytics:- get all Analytics of ads

Setting:- All website details and color ,logo,seo

# User Portal

=====



User ads center able to create user ads , topup wallet using Razorpay.  
Also user can add there websites for crawler.

## Disclaimer

If any copyright issue face while working the site we are not responsible.

Endurasolution is not responsible for any copyright issues arising while using the platform. All users must ensure compliance with legal terms and robots.txt rules.