Laravel deploy with VPS Setup

Point your domain to VPS (DNS setup)

- Go to the site from which the client purchased the domain and log in
- From the domain list, find the domain and go to its DNS settings
- In the Host Record section, add two ARecords for host ‘@’ and ‘api’. Both of their value will be the IP of the VPS. (You will get the VPS IP from where the client purchased the VPS), and the TTL will be automatic or 60
- Also, update the CNAME Record for ‘www’ where the value will be the domain name, and the TTL will be automatic or 60
- Then check if the domain and api subdomain become available or not on https://www.whatsmydns.net’

Deploy on VPS

Step 1: Go to the Terminal of your Computer and log in to VPS

```
ssh root@ip
```

Enter the VPS root password. (You will find it where the VPS is)

Step 2: Update the system

```
sudo apt update

sudo apt upgrade
```

Step 3: Enable and configure UFW (Uncomplicated Firewall) and check the port status

```
sudo ufw allow 22/tcp

sudo ufw enable

sudo ufw status
```

Step 4: Installing NVM

```
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.40.3/install.sh | bash
```

- Restart. Stop the terminal and log in again to the VPS.
- Check the version nvm -v

Step 5: Installing Node

```
nvm install node
npm -v
node -v
```

Step 6: Installing and Configuring PHP

- Update package lists and check available PHP versions

```
sudo apt update

apt search php | grep "^php[0-9]"
```

- Install latest

```
sudo apt install -y php php-fpm php-mysql php-xml php-curl php-mbstring php-
```

Or if you want to install any specific version

```
sudo apt install -y php8.4 php8.4-fpm php8.4-mysql php8.4-xml php8.4-curl ph
```

- Check directly for PHP 8.4

```
sudo systemctl status php8.4-fpm
```

If not enabled then

- Start PHP 8.4 FPM service

```
sudo systemctl start php8.4-fpm
```

- Enable PHP 8.4 FPM to start on boot

```
sudo systemctl enable php8.4-fpm
```

- Check status

```
sudo systemctl status php8.4-fpm

php -v
```

Step 7: Installing and Configuring MYSQL Database

- Install mysql

```
sudo apt install -y mysql-server
```

- Start and enable MySQL

```
sudo systemctl start mysql
sudo systemctl enable mysql
```

- secure installation

```
sudo mysql_secure_installation
```

- Give validate password yes and give strong password
- **Remove anonymous users?** → Type y and press Enter
- **Disallow root login remotely?** → Type n and press Enter
- **Remove the test database and access to it?** → Type y and press Enter
- **Reload privilege tables now?** → Type y and press Enter

Setting up the username and password:

- Connect to MySQL as root (no password needed initially)

```
sudo mysql
```

- Once in the MySQL prompt, run these commands:

```
ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY 'Yo
```

- Flush and Exit

```
FLUSH PRIVILEGES;
EXIT;
```

- Test MySQL Connection

```
mysql -u root -p
```

Enter password:

Step 8: Installing Composer

- Download the Composer installer

```
curl -sS https://getcomposer.org/installer -o composer-setup.php
```

- Install Composer globally

```
sudo php composer-setup.php --install-dir=/usr/local/bin --filename=composer
```

- Remove installer

```
rm composer-setup.php
```

- Verify installation

```
composer --version
```

Step 8: Installing PM2

- Install PM2 globally

```
npm install -g pm2
```

Step 9: Configure Firewall

```
sudo ufw allow 80
sudo ufw allow 443

sudo ufw reload
sudo ufw status
```

Step 10: Create Web Directories

- check first

```
cd /var/www
```

- create

```
mkdir  /var/www
```

Step 11: Github Connect

```
ls -al ~/.ssh
```

```
ssh-keygen -t rsa -b 4096 -C "github@smtech24.com"
```

- Enter the default or give specific passphrase

```
cat ~/.ssh/id_rsa.pub
```

- Save the SSH-RA, then configure it in sm technology GitHub, ask the leader to do that or someone who has access

```
ssh -T git@github.com
```

```
git clone <github repo's ssh key"
```

Step 12: Allow the backend port

```
ls
```

```
cd backend-project-name
```

```
ls
```

```
sudo ufw allow 8000
```

Step 13: Connect Database

```
mysql -u root -p
```

- Create database

```
CREATE DATABASE database_name
```

- Create a dedicated database user

```
CREATE USER 'database_user'@'localhost' IDENTIFIED BY 'SecurePass123!';
```

- Grant privileges

```
GRANT ALL PRIVILEGES ON markustayev_db.* TO 'database_user'@'localhost';
```

- Flush privileges

```
FLUSH PRIVILEGES;
```

```
EXIT;
```

Step 14: Edit and configure .env

```
nano .env
```

Step 15: Run all required commands to run the project

- composer update or install if required

```
composer install

composer update
```

- Migrate the database

```
php artisan migrate
```

- Link the Storage

```
php artisan storage:link
```

- Run the seeder if needed

```
php artisan db:seed
```

- If you have any scheduler

Go to crontab and select 1, which is nano and easiest

```
crontab -e
```

In the cron tab, add this line and replace the project-name and save

```
* * * * * cd /var/www/project-name && /usr/bin/php artisan schedule:run >> /dev
```

- pm2 start

here 85.31.235.77:8000 is vps_ip:backend_port allowed before

```
pm2 start php --name php-server -- -S ip_number:8000 -t public
```

```
pm2 status
```

```
pm2 save
```

```
pm2 startup
```

- Give these necessary permissions

```
sudo chown -R www-data:www-data /var/www/project-backend-name/storage
```

```
sudo chmod -R 775 /var/www/project-backend-name/storage /var/www/project
```

You can check locally

```
php artisan serve --host=0.0.0.0 --port=8000
```

You can now check whether the backend is running on 85.31.235.77:8000 (vps_ip:backend_port) or not

Step 16: Nginx Setup

- Check if already running or not

```
sudo systemctl status nginx
```

- install

```
sudo apt install nginx
```

```
nginx -v
```

- start and enable

```
sudo systemctl start nginx
```

```
sudo systemctl enable nginx
```

- port allow

```
sudo ufw allow 80/tcp
```

```
sudo ufw allow 443/tcp
```

```
sudo ufw reload
```

```
cd /etc/nginx
```

```
ls
```

conf.d fastcgi_params koi-win modules-available nginx.conf
scgi_params sites-enabled uwsgi_params fastcgi.conf koi-utf
mime.types modules-enabled proxy_params sites-available snippets win-
utf

```
cd conf.d
```

```
ls
```

```
nano api.mglobalranking.com.conf
```

```
server {
    server_name api.mglobalranking.com;

    root /var/www/backend-project-name/public;
    index index.php index.html;

    location / {
        try_files $uri $uri/ /index.php?$query_string;
    }

    location ~ \.php$ {
        include snippets/fastcgi-php.conf;
        fastcgi_pass unix:/run/php/php8.4-fpm.sock; # Adjust to your PHP version
        fastcgi_param SCRIPT_FILENAME $document_root$fastcgi_script_name;
        include fastcgi_params;
    }

    location ~ /\.ht {
        deny all;
    }
    client_max_body_size 1500M;


}
server {
    if ($host = api.myglobalrankings.com) {
        return 301 https://$host$request_uri;
    } # managed by Certbot


    listen 80;
    server_name api.myglobalrankings.com;
    return 404; # managed by Certbot


}
```

Secure with HTTPS (Optional, Recommended)

1st go to the backend project folder

```
sudo apt update
```

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

```
sudo certbot --nginx -d backend_url
```

Set test renewal manually

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

Now, check your backend URL