# Деплой веб-сайта на VDS: Next.js, Docker, Django

При создании веб-приложения конечной целью является публикация его в интернете. Но это может показаться сложным, особенно если вы раньше никогда этим не занимались. Целью данного руководства как раз и будет рассказать об этапах публикации сайта на VDS-сервер (с последующим обеспечением его защиты), с применением технологий контейнеризации.

В этой статье будут рассматриваться несколько технологий, и если вы столкнетесь с незнакомыми понятиями, не стоит волноваться. По мере прочтения у вас сложится пазл в голове, и в дальнейшем вы сами сможете **устанавливать сайт на VDS**. Итогом станет полноценный сайт с доменным именем и SSL-сертификатом.

# План действий

В этой статье примером будет выступать полноценный веб-сайт — Frontend (клиентская часть) на Next.js, Backend (серверная часть) на Django (python), контейнеризация через Docker.

Для этого мы выполним следующие шаги:

* Создадим пример Frontend (NextJS) и Backend (Django) сайтов;
* Настроим Docker-контейнеры;
* Настроим Nginx (сервер) и LetsEncrypt (SSL-сертификат).

# Необходимые требования

Начнём с главного:

1. Аккаунт в Timeweb Cloud — [зарегистрироваться можно по этой ссылке](https://timeweb.cloud/my/projects);
2. VDS — облачный сервер с IPv4, [создать можно по этой ссылке](https://timeweb.cloud/my/servers/create);
3. [Доменное имя](https://timeweb.cloud/my/domains)

Рекомендуем прочитать данную статью полностью, перед тем как приступать к развертыванию. Это даст вам понять, что вас ждет и как лучше поступить в вашем конкретном случае.

В случае отсутствия доменного имени, вы можете приобрести бесплатный технический домен (доменное имя задается 3 уровня, например: .tw1.su).

О том, как его получить, вы можете прочитать в [нашей документации](https://timeweb.cloud/docs/domains/technical-domain).

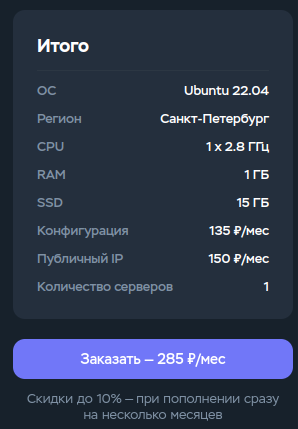
Продолжим наше руководство по тому, **как опубликовать сайт на VDS-сервере**.

# Настройка сервера

Купить сервер можно по [этой ссылке](https://timeweb.cloud/my/servers/create).

Внизу вы видите самый дешевый комплект — 285 рублей в месяц за стандартный VDS с публичным IPv4. ОС желательно выбрать Ubuntu, так как именно для нее было спроектировано наше руководство.

Публичный IPv4 требуется, если ваша сеть не поддерживает IPv6-подключения.



После вам нужно будет подключиться по протоколу SSH к вашему серверу.

SSH — это сетевой протокол для удалённого управления операционной системой с помощью командной строки и передачи данных в зашифрованном виде.

Если у вас Linux, то вы можете просто ввести команду:

ssh root@<ip>

Как вы видите, по умолчанию мы подключаемся к root. Это является брешью в безопасности (злоумышленник может получить доступ ко всей системе), поэтому позднее мы ее исправим.

А для Windows вы можете установить [PuTTY](https://www.putty.org/) — клиент SSH. Но не обязательно использовать PuTTY, с версии Windows 10 1809 SSH-клиент доступен через PowerShell.

## Установка нужного комплекта программ

Нам нужно будет установить:

1. Docker — это платформа контейнеризации, с помощью которой можно автоматизировать и отделить приложения от основной системы.
2. Docker-compose — это надстройка над докером, которая позволяет запускать несколько контейнеров одновременно и маршрутизировать потоки данных между ними.
3. CertBot — это клиент для установки SSL-сертификата от Let’s Encrypt. Let’s Encrypt — это центр сертификации, который создает по умолчанию сертификаты сроком действия в 90 дней.
4. Nginx — веб-сервер.

sudo apt install nginx certbot python3-certbot-nginx  
sudo snap install docker

Docker мы будем устанавливать через Ubuntu Snaps, потому что благодаря ему автоматически будут установлены все нужные зависимости для работы Docker-compose.

Ubuntu Snaps - это универсальный формат пакетов, который не зависит от конкретного Linux-дистрибутива.

## Настройка безопасности

Итак, вы решили **опубликовать сайт на VDS сервере**. Но стоит подумать о защите сервера, иначе злоумышленники могут получить несанкционированный доступ к системе.

### Новый пользователь

Как сказано выше, одна из проблем — через ssh мы сразу подключаемся к суперпользователю (root), то есть удаленно администрируем сервер без ограничений. А этим может воспользоваться злоумышленник.

Поэтому мы заведем другого пользователя и отключим доступ к ssh суперпользователю.

Для этого существует команда useradd:

useradd [options] <username>

После нужно задать пароль новому пользователю и добавить его в группу sudo:

passwd <username>  
usermod -aG sudo <username>

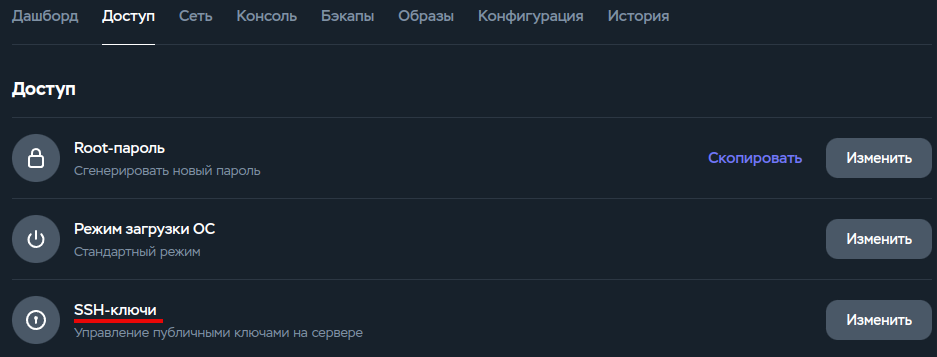
Группа sudo нужна для того, чтобы мы могли работать с системой.

## Аутентификация через ключ

Чтобы мы могли подключаться к серверу через ключ, а не через пароли, нам нужно его сгенерировать. Иначе останется возможность утечки пароля администратора:

ssh-keygen -t rsa

Затем скопируйте публичный ключ (~/.ssh/id\_rsa.pub) и вставьте его в секцию SSH-ключей сервера на Timeweb Cloud.



Также давайте настроим SSH-ключ для пользователя.

Есть разные программы, реализующие подключение по SSH, такие, как lsh и Dropbear, но самой популярной является OpenSSH. Установка клиента OpenSSH на Ubuntu:

sudo apt install openssh

Запуск демона SSH (sshd) на сервере под Ubuntu:

sudo systemctl start ssh  
sudo systemctl enable ssh

После, вы должны создать на сервере директорию SSH в домашней директории юзера (будучи из-под аккаунта администратора) и добавить публичный ключ, который мы сгенерировали ранее, в файл authorized\_keys. А также изменить права:

mkdir -p /home/<username>/.ssh && touch /home/<username>/.ssh/authorized\_keys  
chmod 700 /home/<username>/.ssh && chmod 600 /home/<username>/.ssh/authorized\_keys  
chown -R <username>:<username> /home/<username>/  
chown -R <username>:<username> /home/<username>/.ssh

После отредактируем файл /etc/ssh/sshd\_config:

PermitRootLogin no # изменить с yes на no  
PasswordAuthentication no # изменить с yes на no

Эти параметры запретят входить через root и аутентификацию по паролю.

## Настройка брандмауэра

Брандмауэр (Firewall), грубо говоря, блокирует часть портов для защиты от посторонних подключений. В Ubuntu идет встроенный ufw, поэтому настроим его:

sudo ufw allow ssh # разрешаем ssh  
sudo ufw allow 'Nginx Full' # разрешаем Nginx  
sudo ufw enable # запускаем

## Fail2Ban

[Fail2Ban](https://www.fail2ban.org/wiki/index.php/Main_Page) — это утилита для защиты от брутфорса (bruteforce — грубая сила, взлом методом перебора паролей). Она анализирует логи и попытки входа, и может блокировать IP адреса в зависимости от правил. Например, после 5 неудачных попыток в течение 10 минут, блокируем IP адрес на 2 часа.

Установка:

sudo apt install fail2ban  
systemctl start fail2ban # запускаем  
systemctl enable fail2ban # включаем автозагрузку

И редактируем конфиг /etc/fail2ban/jail.conf:

[DEFAULT]  
ignorecommand =  
bantime = 120m  
findtime = 10m  
maxretry = 9

Если в течение 10 минут было 9 попыток войти, мы блокируем IP на 120 минут.

Нам надо перезапустить сервис fail2ban, так как мы отредактировали конфигурацию. Сделать это можно через systemctl (утилита для управления системой инициализации systemd):

sudo systemctl restart fail2ban

# Контейнеризация

Основная часть **установки сайта на VDS-сервер** — это настройка контейнеризации через Docker.

Docker — это платформа для создания контейнеров с открытым исходным кодом.

Контейнеры Docker — это стандартизированные, изолированные и портативные пакеты программного обеспечения, которые включают в себя всё необходимое для запуска приложения, включая код, среду выполнения, системные инструменты, библиотеки и настройки.

Для этого туториала был создан [репозиторий](https://github.com/alexeev-prog/website-deploy-tutorial), который содержит два каталога: frontend и backend.

Директория frontend содержит код для Next.js-приложения, когда как backend отвечает за серверную часть (Django).

Чтобы начать, склонируете репозиторий-шаблон:

git clone https://github.com/alexeev-prog/website-deploy-tutorial

И измените в backend/backend/settings.py константу ALLOWED\_HOSTS:

ALLOWED\_HOSTS = [  
 "109.68.212.254", 'hardtobecoder.tw1.su', 'localhost', '0.0.0.0'  
]

109.68.212.254 замените на IP-адрес сервера, hardtobecoder.tw1.su на ваш домен.

## Создание Docker-контейнера для Frontend (клиентская часть)

В директории frontend есть файл Dockerfile, который отвечает за установку модулей и запуск приложения:

FROM node:20-alpine  
  
WORKDIR /usr/app  
COPY ./ /usr/app/  
  
COPY package.json /usr/app/  
RUN npm install --legacy-peer-deps  
COPY . /usr/app  
  
RUN npm run build  
  
CMD ["npm", "run", "start"]

Он задействует NodeJS и пакетный менеджер npm для установки и запуска.

Также, есть файл frontend/.Dockerignore, который копирует frontend/.gitignore.

## Создание Docker-контейнера для Backend (серверная часть)

Теперь рассмотрим контейнер для серверной части на Django. Аналогично с Frontend (клиентская часть), есть Dockerfile, который выглядит следующим образом:

FROM python:3.9  
ENV PYTHONBUFFERED 1  
ENV PYTHONDONTWRITEBYTECODE 1  
  
copy requirements.txt .  
  
RUN python -m pip install -r requirements.txt  
  
RUN mkdir app  
WORKDIR /app  
COPY . /app  
  
CMD gunicorn --workers 3 -b 0.0.0.0:8000 backend.wsgi

Данный Docker-контейнер будет запускать gunicorn — WSGI сервер. Позднее мы подключим к нему Nginx как обратный прокси-сервер.

И абсолютно также существует файл backend/.dockerignore.

## Создание Docker-compose

Для того, чтобы наши Docker-контейнеры работали вместе, есть файл Docker-compose.yml:

version: '3.8'  
  
services:  
 frontend:  
 build:  
 context: ./frontend  
 dockerfile: Dockerfile  
 ports:  
 — "3000:3000"  
 depends\_on:  
 — backend  
  
 backend:  
 build:  
 context: ./backend  
 dockerfile: Dockerfile  
 ports:  
 — "8000:8000"

Docker-compose — этой скрипт на python, который позволяет запускать несколько Docker-контейнеров и настраивать их совместную работу.

# Настройка Nginx

Nginx — это веб-сервер. При обслуживании Django-приложения Nginx выступает в качестве обратного прокси-сервера: отвечает за обработку входящих запросов и их переадресацию.

В директории nginx есть два файла — default и create.sh. default является файлом конфигурации нашего сервера:

server {  
 listen 80;  
 server\_name hardtobecoder.tw1.su;  
  
 location / {  
 proxy\_pass http://localhost:8000;  
 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;  
 }  
}

Замените hardtobecoder.tw1.su на ваш домен, а 8000 на нужный вам порт (в зависимости от того, какой вы указали в Dockerfile Backend’а или Frontend’а).

После этого посмотрим файл create.sh — баш скрипт для установки и настройки прокси-сервера Nginx и загрузки SSL-сертификата:

sudo rm -rf /etc/nginx/sites-enabled/default # удаляем старую конфигурацию  
sudo cp default -r /etc/nginx/sites-available/ # копируем конфигурацию  
sudo ln -s /etc/nginx/sites-available/default /etc/nginx/sites-enabled # создаем ссылку в каталог включенных сайтов  
sudo certbot --nginx -d hardtobecoder.tw1.su # генерируем SSL-сертификат  
sudo nginx -t # проверяем конфигурацию  
sudo nginx -s reload # перезагрузка Nginx  
sudo systemctl restart Nginx # перезагрузка сервиса Nginx

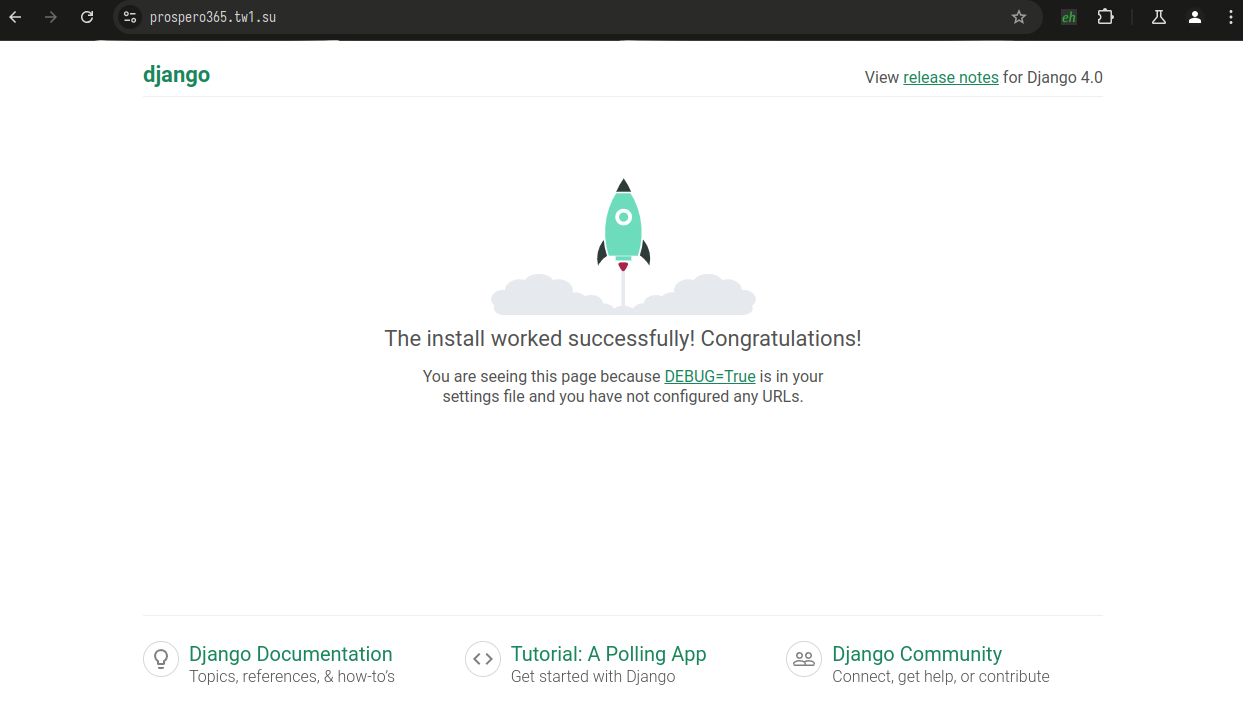
Не забудьте изменить hardtobecoder.tw1.su на ваш домен.

# Запуск

Если вы все настроили правильно, перейдите в директорию с файлом Docker-compose.yml и запустите следующие команды:

sudo docker-compose build  
sudo docker-compose up -d  
cd nginx  
./create.sh

После того как вы перейдете на адрес сайта, вы увидите что ваше приложение доступно через HTTPS благодаря домену и SSL-сертификату!



Эти команды соберут контейнеры, запустят в режиме демона, и в конце будет включен и настроен Nginx.

Сайт успешно опубликован! Вы теперь знаете, **как поставить сайт на VDS**: как настроить сервер, Docker-контейнеры, запускать веб-приложения на python или js. Если вам показалось это руководство полезным, поделитесь им с друзьями или коллегами. Если у вас возникли проблемы, перепроверьте ваши действия или напишите в поддержку Timeweb Cloud.