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

Многие из вас, скорее всего, создавали веб-приложения. И вы хотели поделиться им - с другом, с заказчиком или просто похвастаться. Но для этого надо опубликовать сайт, и с этим могут возникнуть проблемы. “Монолитный” запуск для production-проектов не подходит, поэтому разработчики изобрели Docker-контейнеры для разделения сервисов. Именно поэтому в этой статье я расскажу о том, как их настраивать и соединять их работу, а также как настроить и обезопасить сам VDS.

В этом руководстве 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 сможете сами **устанавливать сайт на VDS**. В конце вы сможете опубликовать сайт на своем домене с 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, база данных sqlite3.

Для этого мы выполним следующие шаги:

* Настроим Frontend (клиент) и Backend (сервер) части;
* Настроим docker-контейнеры;
* Настроим nginx (сервер) и letsencrypt (ssl-сертификаты).

# Необходимые требования

Начнем 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.

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.

# Доменное имя

Вы можете [купить домен](https://timeweb.cloud/my/domains) или воспользоваться бесплатным техническим.

Технический домен — это бесплатный тестовый домен, который позволит вам работать с сервером без регистрации нового домена или технического переноса уже существующего.

Вы можете самостоятельно добавлять технические домены в своей панели управления. Для этого:

1. Перейдите в раздел «Домены».
2. Кликните «Добавить домен».
3. Укажите домен в одной из бесплатных технических зон:

* .tw1.su
* .tw1.ru
* .webtm.ru
* .twc1.net

1. Выберите сервер, к которому нужно привязать домен.
2. Нажмите «Добавить».

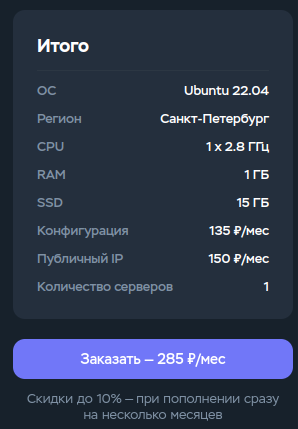
Для технического домена невозможно:

1. Изменить настройки DNS, в том числе указать стороннюю A-запись и перенаправить его на другой хостинг.
2. Создать почтовый ящик.
3. Заказать SSL-сертификат.

# 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 будет Ubuntu 22.04, все команды будут именно для нее.



После вам нужно будет подключиться по SSH.

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, 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 и прокси сервер.

sudo apt install nginx  
sudo snap install docker

Docker мы будем устаналивать через Ubuntu Snaps, т.к. у нас уже автоматически будут установлены все нужные зависимости для работы docker-compose.

## Настройка безопасности

Итак, вы решили **опубликовать сайт на VDS сервере**. Но стоит подумать о защите сервера, иначе злоумышленники могут подключиться к нему.

### Новый пользователь

Как сказано выше, одна из проблем - то что 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>

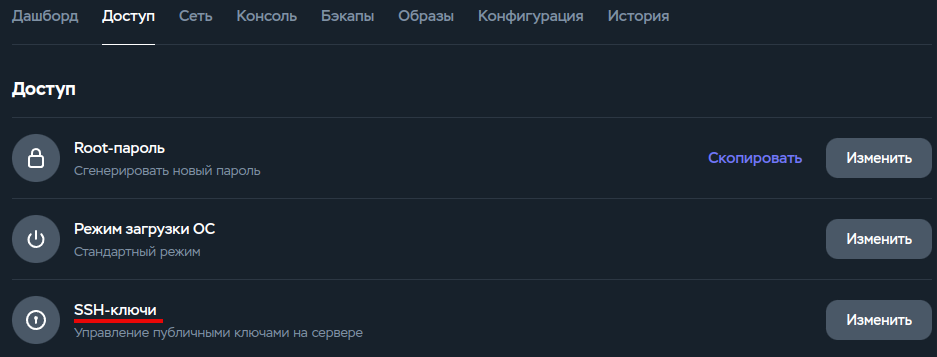
Группа sudo нужна для того, чтобы мы могли работать с системой.

## 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 для реализации протокола 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>/.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

## Смена порта SSH

Злоумышленник изначально будет пробовать 22 порт - порт сервиса ssh по умолчанию. Чтобы не допустить взлома, надо сменить его.

Поэтому в /etc/ssh/sshd\_config поменяйте следующее значение:

Port 9009 # поменяйте с 22 на любой свободный

И перезапустим ssh:

sudo systemctl restart ssh

# 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", 'prospero365.tw1.su', 'localhost', '0.0.0.0'  
]

109.68.212.254 замените на IPv4 адрес сервера, prospero365.tw1.su на ваш домен.

## Создание докер-контейнера для Frontend (клиентская часть)а

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 докер-контейнера для Backend (серверная часть)а

Теперь создадим контейнер для нашей серверной части на Django. Аналогично с Frontend (клиентская часть)ом, 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. После 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.

# NGINX

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

Установка:

sudo apt install nginx

В директории nginx создадим два файла - default и create.sh. default будет файлом конфигурации нашего сервера:

server {  
 listen 80;  
 server\_name prospero365.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 prospero365.tw1.su на ваш домен, а 8000 на нужный вам порт (frontend или backend).

После этого создадим файл create.sh:

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 prospero365.tw1.su # генерируем ssl-сертификат  
sudo nginx -t # проверяем конфигурацию  
sudo nginx -s reload # перезагрузка nginx  
sudo systemctl restart nginx # перезагрузка сервиса nginx

Этот скрипт будет пересоздавать конфигурацию сайта, включая создания ssl-сертификата. Не забудьте изменить prospero365.tw1.su на ваш домен.

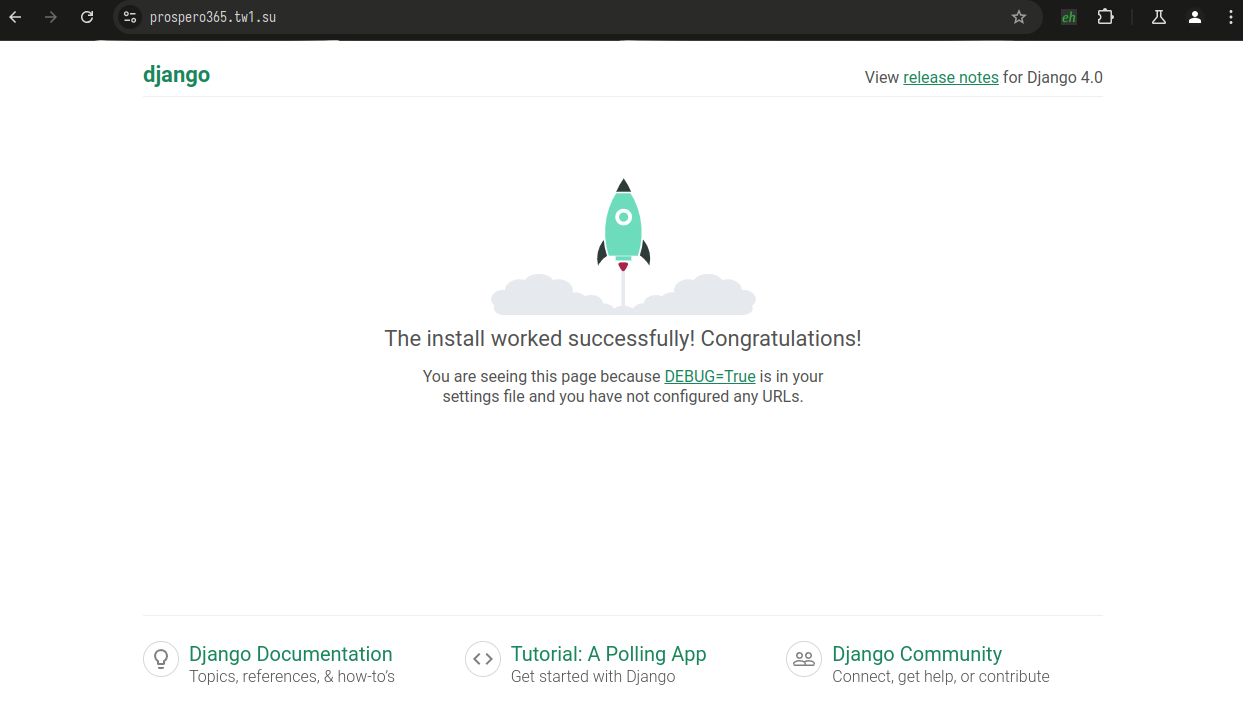
Команда certbot --nginx -d генерирует и вставляет автоматически в наш конфиг ssl-сертификаты.

# Запуск

После того, как вы все настроили, перейдите в директорию с файлом docker-compose.yml и запустите следующие команды:

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

После того, как вы перейдете на адрес сайта, вы увидете что ваше приложение доступно через https!



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

Поздравляю вас! 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 другом.