

# Шаг 1 — Настройка конфигурации веб-сервера

Создать директорию проекта для настройки WordPress с именем `wordpress` и перейти в эту директорию:

```
1. mkdir wordpress && cd wordpress
2.
```

Создать директорию для файла конфигурации:

```
1. mkdir nginx-conf
2.
```

Открыть файл с помощью редактора (nano):

```
1. nano nginx-conf/nginx.conf
2.
```

В этом файле добавить серверный блок с директивами для имени сервера и корневой директории документов, а также блок расположения для направления запросов сертификатов от клиента Certbot, обработки PHP и запросов статичных активов.

Добавить в файл следующий код. Обязательно заменитм `example.com` на свое доменное имя.

```
~/wordpress/nginx-conf/nginx.conf

server {

    listen 80;

    listen [::]:80;


    server_name example.com www.example.com;


    index index.php index.html index.htm;


    root /var/www/html;


    location ~ /.well-known/acme-challenge {

        allow all;
```

```
        root /var/www/html;

    }

    location / {

        try_files $uri $uri/ /index.php$is_args$args;

    }


    location ~ /\.php$ {

        try_files $uri =404;

        fastcgi_split_path_info ^(.+\.php) (/.+)$;

        fastcgi_pass wordpress:9000;

        fastcgi_index index.php;

        include fastcgi_params;

        fastcgi_param SCRIPT_FILENAME
$document_root$fastcgi_script_name;

        fastcgi_param PATH_INFO $fastcgi_path_info;

    }


    location ~ /\.ht{

        deny all;

    }


    location = /favicon.ico {

        log_not_found off; access_log off;

    }

    location = /robots.txt {
```

```

        log_not_found off; access_log off; allow all;

    }

    location ~* \.(css|gif|ico|jpeg|jpg|js|png)$ {

        expires max;

        log_not_found off;

    }

}

```

Сохранить и закрыть файл после завершения редактирования. Если `nano`, нажать `CTRL+X`, `Y`, затем `ENTER`.

После настройки конфигурации Nginx можно перейти к созданию переменных среды для передачи в контейнеры приложения и базы данных во время исполнения.

## Шаг 2 — Настройка переменных среды

В главной директории проекта `~/wordpress`, открыть файл с именем `.env`:

```

1. nano .env
2.

```

Добавить в файл следующие имена и значения переменных: Обязательно предоставить здесь **свои собственные значения** для каждой переменной:

```

~/wordpress/.env
MYSQL_ROOT_PASSWORD=your_root_password

MYSQL_USER=your_wordpress_database_user

MYSQL_PASSWORD=your_wordpress_database_password

```

Сохранить и закрыть файл после завершения редактирования.

Использовать Git для контроля версий, [инициализировать текущую рабочую директорию в качестве репозитория](#) с помощью `git init`:

```

1. git init
2.

```

Затем открыть файл `.gitignore`:

```

1. nano .gitignore

```

```
2.
```

Добавить `.env` в файл:

```
~/wordpress/.gitignore
```

```
.env
```

Сохранить и закрыть файл после завершения редактирования.

Добавить `.env` в файл `.dockerignore`

Открыть файл:

```
1. nano .dockerignore
2.
```

Добавить `.env` в файл:

```
~/wordpress/.dockerignore
```

```
.env
```

Добавить файлы и директории, связанные с разработкой вашего приложения:

```
~/wordpress/.dockerignore
```

```
.env
```

```
.git
```

```
docker-compose.yml
```

```
.dockerignore
```

Сохранить файл и закрыть его после завершения.

## Шаг 3 — Определение служб с помощью Docker Compose

Открыть файл `docker-compose.yml`:

```
1. nano docker-compose.yml
2.
```

Добавить следующий код для определения версии файла Compose и базы данных `db`:

```
~/wordpress/docker-compose.yml
```

```
version: '3'
```

```
services:
```

```

db:

  image: mysql:8.0

  container_name: db

  restart: unless-stopped

  env_file: .env

  environment:

    - MYSQL_DATABASE=wordpress

  volumes:

    - dbdata:/var/lib/mysql

  command: '--default-authentication-plugin=mysql_native_password'

  networks:

    - app-network

```

Затем под определением службы `db` добавить определение для нашей службы приложения `wordpress`:

~/wordpress/docker-compose.yml

```

...

wordpress:

  depends_on:

    - db

  image: wordpress:5.1.1-fpm-alpine

  container_name: wordpress

  restart: unless-stopped

  env_file: .env

  environment:

    - WORDPRESS_DB_HOST=db:3306

    - WORDPRESS_DB_USER=$MYSQL_USER

```

```
- WORDPRESS_DB_PASSWORD=$MYSQL_PASSWORD

- WORDPRESS_DB_NAME=wordpress

volumes:

- wordpress:/var/www/html

networks:

- app-network
```

Далее под определением службы приложения `wordpress` добавить следующее определение для службы Nginx `webserver`:

```
~/wordpress/docker-compose.yml

...

webserver:

  depends_on:

    - wordpress

  image: nginx:1.15.12-alpine

  container_name: webserver

  restart: unless-stopped

  ports:

    - "80:80"

  volumes:

    - wordpress:/var/www/html

    - ./nginx-conf:/etc/nginx/conf.d

    - certbot-etc:/etc/letsencrypt

  networks:

    - app-network
```

Под определением `webserver` добавить последнее определение для службы `certbot`. Обязательно заменить адрес электронной почты и доменные имена, на свои собственные:

~/wordpress/docker-compose.yml

```
certbot:

  depends_on:

    - webserver

  image: certbot/certbot

  container_name: certbot

  volumes:

    - certbot-etc:/etc/letsencrypt

    - wordpress:/var/www/html

  command: certonly --webroot --webroot-path=/var/www/html --email
sammy@example.com --agree-tos --no-eff-email --staging -d example.com -d
www.example.com
```

Под определением службы `certbot` добавить определения сети и тома:

~/wordpress/docker-compose.yml

```
...

volumes:

  certbot-etc:

  wordpress:

  dbdata:


networks:

  app-network:

    driver: bridge
```

Итоговый файл `docker-compose.yml` будет выглядеть примерно так:

~/wordpress/docker-compose.yml

```
version: '3'
```

```
services:

  db:

    image: mysql:8.0

    container_name: db

    restart: unless-stopped

    env_file: .env

    environment:

      - MYSQL_DATABASE=wordpress

    volumes:

      - dbdata:/var/lib/mysql

    command: '--default-authentication-plugin=mysql_native_password'

    networks:

      - app-network

  wordpress:

    depends_on:

      - db

    image: wordpress:5.1.1-fpm-alpine

    container_name: wordpress

    restart: unless-stopped

    env_file: .env

    environment:

      - WORDPRESS_DB_HOST=db:3306

      - WORDPRESS_DB_USER=$MYSQL_USER
```



```
- WORDPRESS_DB_PASSWORD=$MYSQL_PASSWORD

- WORDPRESS_DB_NAME=wordpress

volumes:

- wordpress:/var/www/html

networks:

- app-network

webserver:

  depends_on:

    - wordpress

  image: nginx:1.15.12-alpine

  container_name: webserver

  restart: unless-stopped

  ports:

    - "80:80"

  volumes:

    - wordpress:/var/www/html

    - ./nginx-conf:/etc/nginx/conf.d

    - certbot-etc:/etc/letsencrypt

  networks:

    - app-network

certbot:

  depends_on:

    - webserver
```

```

image: certbot/certbot

container_name: certbot

volumes:

  - certbot-etc:/etc/letsencrypt

  - wordpress:/var/www/html

  command: certonly --webroot --webroot-path=/var/www/html --email
sammy@example.com --agree-tos --no-eff-email --staging -d example.com -d
www.example.com

volumes:

  certbot-etc:

  wordpress:

  dbdata:

networks:

  app-network:

    driver: bridge

```

Сохранить и закрыть файл после завершения редактирования.

После добавления определений службы запустить контейнеры и протестировать запросы сертификата.

## Шаг 4 — Получение сертификатов SSL и учетных данных

Создать контейнеры с помощью команды `docker-compose up` и флага `-d`, которые будут запускать контейнеры `db`, `wordpress` и `webserver` в фоновом режиме:

1. `docker-compose up -d`
- 2.

Вывод, подтверждающий, что службы были успешно созданы:

Output

```
Creating db ... done

Creating wordpress ... done

Creating webserver ... done

Creating certbot ... done
```

С помощью `docker-compose ps` проверить статус служб:

```
1. docker-compose ps
2.
```

Все успешно, службы `db`, `wordpress` и `webserver` - статус `Up`, а работа контейнера `certbot` завершена с сообщением о статусе `0`:

Открыть `docker-compose.yml`:

```
1. nano docker-compose.yml
2.
```

В разделе файла с определением службы `certbot` заменить флаг `--staging` в параметрах `command` на флаг `--force-renewal`

```
~/wordpress/docker-compose.yml

...

certbot:

  depends_on:

    - webserver

  image: certbot/certbot

  container_name: certbot

  volumes:

    - certbot-etc:/etc/letsencrypt

    - certbot-var:/var/lib/letsencrypt

    - wordpress:/var/www/html

  command: certonly --webroot --webroot-path=/var/www/html --email
sammy@example.com --agree-tos --no-eff-email --force-renewal -d
example.com -d www.example.com

...
```

Для воссоздания контейнера `certbot`:

```
1. docker-compose up --force-recreate --no-deps certbot
2.
```

Вывод, указывающий, что запрос сертификата выполнен успешно.

## Шаг 5 — Изменение конфигурации веб-сервера и определения службы

Остановить работу `webserver`:

```
1. docker-compose stop webserver
2.
```

Получить [рекомендуемые параметры безопасности Nginx](#) от Certbot с помощью `curl`:

```
1. curl -sSL https://raw.githubusercontent.com/certbot/certbot/master/certbot-nginx/certbot_nginx/_internal/tls_configs/options-ssl-nginx.conf
2.
```

Затем удалить ранее созданный файл конфигурации Nginx:

```
1. rm nginx-conf/nginx.conf
2.
```

Открыть другую версию файла:

```
1. nano nginx-conf/nginx.conf
2.
```

Добавить следующий код в файл для перенаправления HTTP на HTTPS и добавления учетных данных, протоколов и заголовков безопасности SSL. Обязательно замените `example.com` на свое доменное имя:

```
~/wordpress/nginx-conf/nginx.conf

server {

    listen 80;

    listen [::]:80;


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

```
        location ~ /\.well-known/acme-challenge {

            allow all;

            root /var/www/html;

        }

        location / {

            rewrite ^ https://$host$request_uri? permanent;

        }

    }

server {

    listen 443 ssl http2;

    listen [::]:443 ssl http2;

    server_name example.com www.example.com;


    index index.php index.html index.htm;


    root /var/www/html;


    server_tokens off;


    ssl_certificate /etc/letsencrypt/live/example.com/fullchain.pem;

    ssl_certificate_key /etc/letsencrypt/live/example.com/privkey.pem;


    include /etc/nginx/conf.d/options-ssl-nginx.conf;
```

```
add_header X-Frame-Options "SAMEORIGIN" always;

add_header X-XSS-Protection "1; mode=block" always;

add_header X-Content-Type-Options "nosniff" always;

add_header Referrer-Policy "no-referrer-when-downgrade" always;

add_header Content-Security-Policy "default-src * data: 'unsafe-
eval' 'unsafe-inline'" always;

# add_header Strict-Transport-Security "max-age=31536000;
includeSubDomains; preload" always;

# enable strict transport security only if you understand the
implications


location / {

    try_files $uri $uri/ /index.php$is_args$args;

}


location ~ /\.php$ {

    try_files $uri =404;

    fastcgi_split_path_info ^(.+\.(php))(/.+)$;

    fastcgi_pass wordpress:9000;

    fastcgi_index index.php;

    include fastcgi_params;

    fastcgi_param SCRIPT_FILENAME
$document_root$fastcgi_script_name;

    fastcgi_param PATH_INFO $fastcgi_path_info;

}
```

```

        location ~ /\.ht {

            deny all;

        }

        location = /favicon.ico {

            log_not_found off; access_log off;

        }

        location = /robots.txt {

            log_not_found off; access_log off; allow all;

        }

        location ~* \.(css|gif|ico|jpeg|jpg|js|png)$ {

            expires max;

            log_not_found off;

        }

    }
}

```

Открыть файл `docker-compose.yml`:

1. `nano docker-compose.yml`
- 2.

В определении службы `webserver` добавить следующее распределение порта:

```

~/wordpress/docker-compose.yml
...

webserver:

    depends_on:

        - wordpress

    image: nginx:1.15.12-alpine

```

```
container_name: webserver

restart: unless-stopped

ports:

  - "80:80"

  - "443:443"

volumes:

  - wordpress:/var/www/html

  - ./nginx-conf:/etc/nginx/conf.d

  - certbot-etc:/etc/letsencrypt

networks:

  - app-network
```

Сохранить и закрыть файл после завершения редактирования.

Повторно создать службу `webserver`:

```
1. docker-compose up -d --force-recreate --no-deps webserver
2.
```

Проверить службы с помощью команды:

```
1. docker-compose ps
2.
```

Результат - службы `db`, `wordpress` и `webserver` запущены:

После запуска контейнеров, завершить процесс установки WordPress через веб-интерфейс.

## Шаг 6 — Завершение установки через веб-интерфейс

В браузере перейти на домен сервера.

```
https://example.com
```

После нажатия **Continue** (Продолжить) перейти на главную страницу настройки, выбрать имя сайта и пользователя.

Ввести адрес электронной почты и указать





## Welcome

Welcome to the famous five-minute WordPress installation process! Just fill in the information below and you'll be on your way to using the most extendable and powerful personal publishing platform in the world.

## Information needed

Please provide the following information. Don't worry, you can always change these settings later.

**Site Title**

Sammy's Blog

**Username**

sammy

Username can have only alphanumeric characters, spaces, underscores, hyphens, periods, and the @ symbol.

**Password**

GFMyRD&IL^k4yTmv@L

Hide

Strong

**Important:** You will need this password to log in. Please store it in a secure location.

**Your Email**

sammy@example.com

Double-check your email address before continuing.

**Search Engine  
Visibility**

☐ Discourage search engines from indexing this site

It is up to search engines to honor this request.

Install WordPress

После нажатия **Install WordPress** (Установить Wordpress) внизу страницы на экране появится запрос выполнения входа:



## Success!

WordPress has been installed. Thank you, and enjoy!

**Username**

sammy

**Password**

*Your chosen password.*

Log In

После входа - доступ к панели управления WordPress:

