[New! Premium CPU-Optimized Droplets are now available. Learn more ->](https://www.digitalocean.com/blog/introducing-premium-cpu-optimized-droplets" \t ")

**Шаг 1 — Настройка конфигурации веб-сервера**

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

1. mkdir wordpress && cd wordpress

Создать директорию для файла конфигурации:

1. mkdir nginx-conf

Открыть файл с помощью редактора (nano):

1. nano nginx-conf/nginx.conf

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 заменитm 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, нажатьCTRL+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, откроыть файл с именем .env:

1. nano .env

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](https://www.digitalocean.com/community/tutorials/how-to-use-git-a-reference-guide#set-up-and-initialization) с помощью git init:

1. git init

Затем открыть файл .gitignore:

1. nano .gitignore

Добавьть .env в файл:

~/wordpress/.gitignore

.env

Сохранить и закрыть файл после завершения редактирования.

Добавить .env в файл .dockerignore

Открыть файл:

1. nano .dockerignore

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

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](https://docs.docker.com/compose/reference/up/) и флага -d, которые будут запускать контейнеры db​​​, wordpress и webserver в фоновом режиме:

1. docker-compose up -d

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](https://docs.docker.com/compose/reference/ps/) проверить статус служб:

1. docker-compose ps

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

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

Вывод, указывающий, что запрос сертификата выполнен успешно.ot | again. To non-interactively renew \*all\* of your certificates, run

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

Остановить работу webserver :

1. docker-compose stop webserver

Получить [рекомендуемые параметры безопасности Nginx](https://github.com/certbot/certbot/blob/master/certbot-nginx/certbot_nginx/tls_configs/options-ssl-nginx.conf) от Certbot с помощью curl:

1. curl -sSLo nginx-conf/options-ssl-nginx.conf https://raw.githubusercontent.com/certbot/certbot/master/certbot-nginx/certbot\_nginx/\_internal/tls\_configs/options-ssl-nginx.conf

Затем удалить ранее созданный файл конфигурации Nginx:

1. rm nginx-conf/nginx.conf

Открыть другую версию файла:

1. nano nginx-conf/nginx.conf

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

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

Проверить службы с помощью команды:

1. docker-compose ps

Результат - службы db, wordpress и webserver запущены:

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

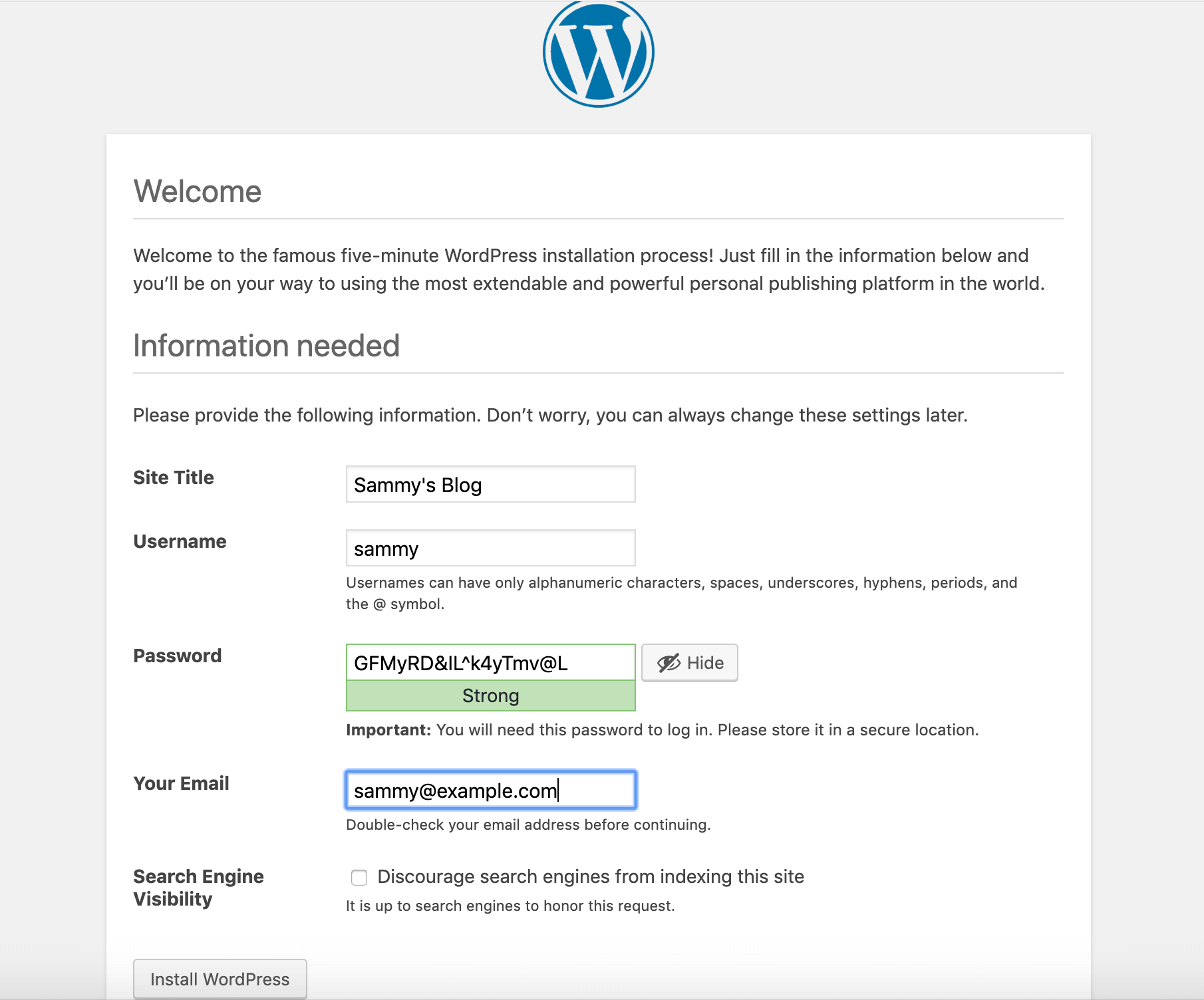
**Шаг 6 — Завершение установки через веб-интерфейс**

В браузере перейти на домен сервера.

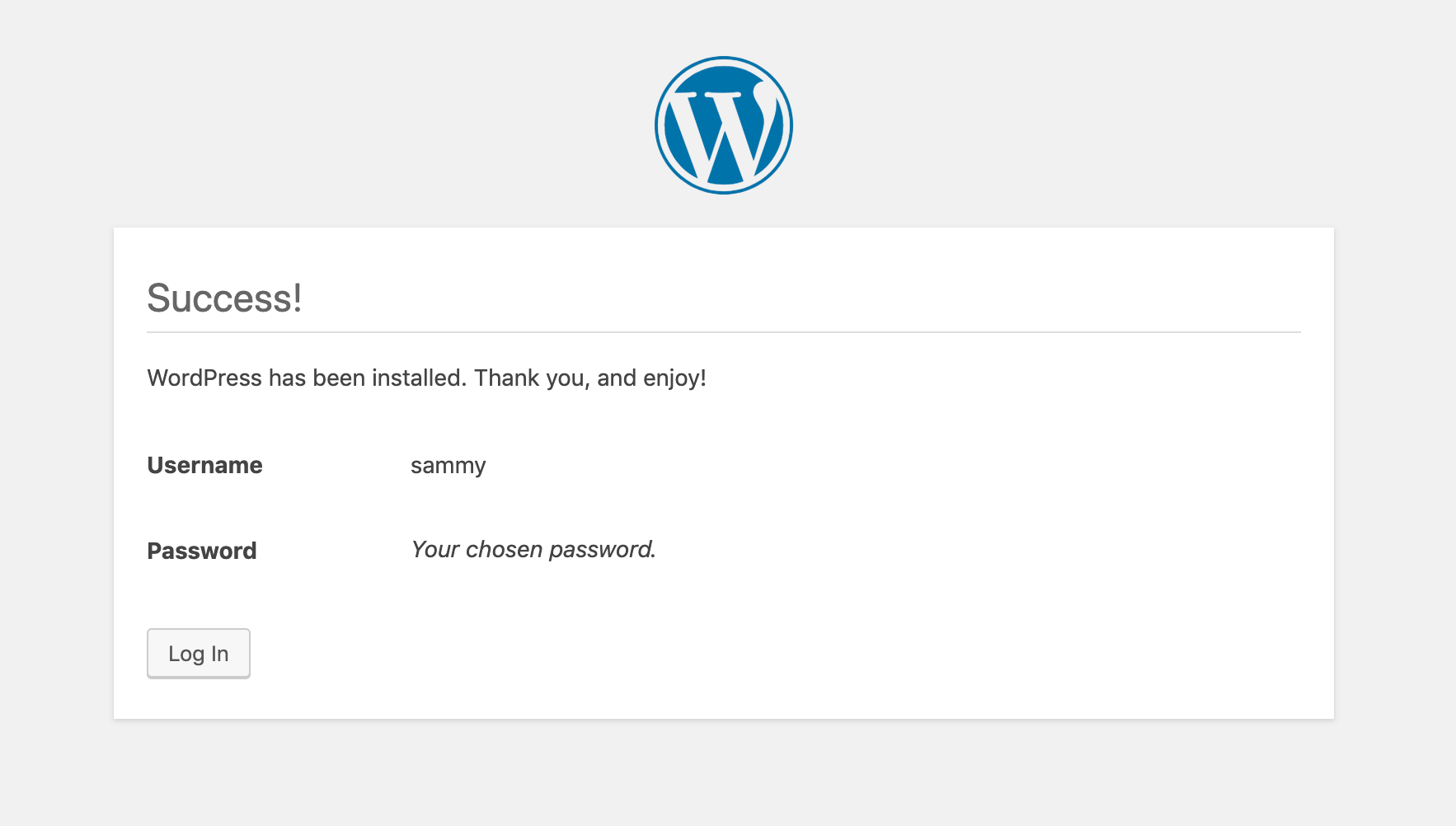
https://example.com

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

Ввести адрес электронной почты и указать



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



После входа - доступ к панели управления WordPress:

