How To Deploy NGINX Reverse Proxy On Docker

Posted January 6, 2021 NGINX DOCKER REVERSE PROXY

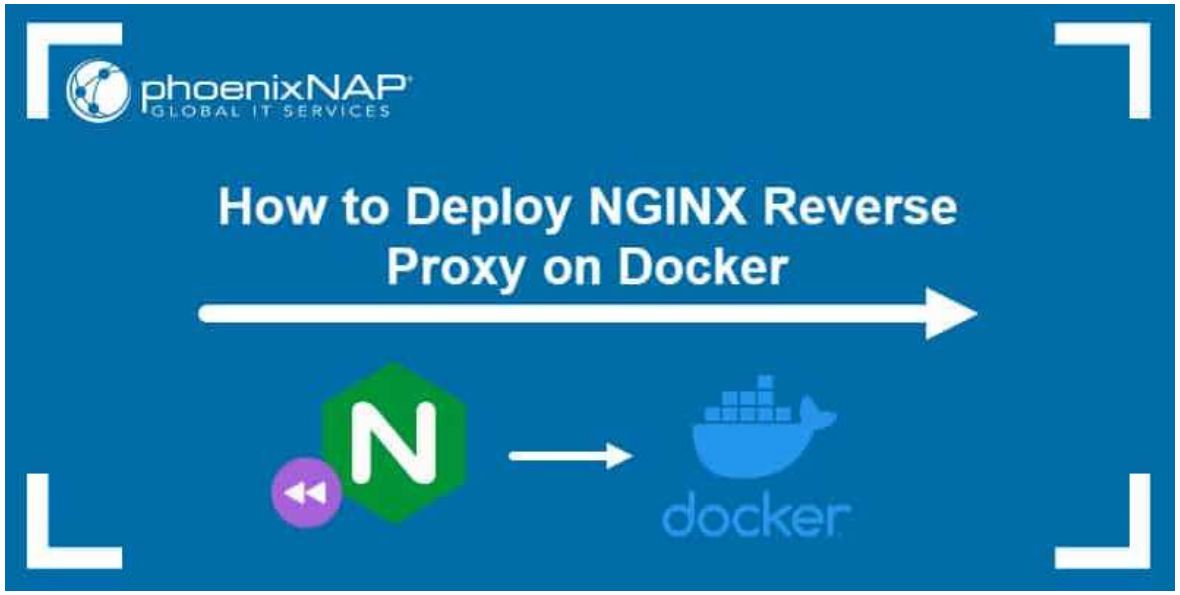
Contents

Introduction

A reverse proxy intercepts incoming requests and directs them to the appropriate server. Not only does this speed up performance, it also strengthens server security.

The easiest way to set up and manage reverse proxies is to use Nginx and Docker. This guide sets up two sample web services inside Docker containers and a Nginx reverse proxy for those services.

In this tutorial, you will learn how to set up a reverse proxy on Docker for two sample web servers.



Prerequisites

- A user account with **sudo** privileges
- A command line/terminal window (**Ctrl-Alt-T**)
- Docker installed on your system
- Docker Compose installed on your system
- A registered domain name with an SSL Certificate associated with it

Why Do You Need Reverse Proxy?

While the standard proxy server protects the clients, the reverse proxy **protects the server**.

A reverse proxy is a type of proxy set up behind the private network's firewall. Its primary role is to intercept traffic and direct a request to the appropriate server on the backend.

The main reasons for using a reverse proxy are to improve **security** and **performance**. If you have services running on multiple pods, you can redirect all requests coming into the network to go through a number of specified ports.

Additionally, reverse proxies can also handle SSL encryption, caching, and compressing data.

Step 1: Create a Sample Web Service

1. Start by creating a new directory for the first sample web service. In this tutorial, we create a directory **`example1`**, but you can use a name of your choice.

```
mkdir example1
```

2. Move into that directory:

```
cd example1
```

3. Create a docker-compose YAML configuration file for the first container to define the service. Use a text editor of your choice (in this example, we use Nano).

```
sudo nano docker-compose.yml
```

4. Then, add the following content into the .yaml file:

```
version: '2'
services:
 app:
  image: nginx
  volumes:
   - .:/usr/share/nginx/html
  ports:
   - "80"
```

The docker-compose file specifies this is an **app** service that uses the **nginx** image. It mounts the root of **exampe1** from the Docker host to **/usr/share/nginx/html/**. Finally, the configuration exposes the service on port **80**.

The **docker-compose.yml** file varies depending on the service you want to run. The configuration above is specific to the web service we create for this tutorial.

Note: Since this is a sample service, we did not include the version number of the nginx image. However, in a production environment, make sure to specify the version as well.

4. Save and exit the file.

5. While inside the same **example1** directory, create an index file for the web service:

```
sudo nano index.html
```

6. Add a few simple lines to appear on the sample web page:

```
<!DOCTYPE html>
<html>
<head>
<title>Website 1</title>
</head>
<body>
<h1>Hello! This is the first sample website.</h1>
</body>
</html>
```

7. Save and close the file.

8. Build the newly created service using the `docker-compose` command:

```
sudo docker-compose build
```

If you are using a prebuilt image, as in this example, the output responds with **app uses an image, skipping**.

9. Next, start the container with:

```
sudo docker-compose up -d
```

Step 2: Create a Second Sample Web Service

Create a second sample web service by following the same process.

Make sure to return to the home directory if you are still in **example1**. To do so, run `cd` in the terminal window.

1. Create a new directory where you will store the docker-compose and index file for the second website. We will name this directory **example2**.

```
mkdir example2
```

2. Move into **example2** by running:

```
cd example2
```

3. Create a docker-compose file:

```
sudo nano docker-compose.yml
```

4. Define the service using the same specifications as in Step 1. Add the following content to the file:

```
version: '2'
services:
 app:
  image: nginx
  volumes:
   - .:/usr/share/nginx/html
```

```
  ports:
   - "80"
```

5. Save and exit the file.

6. Then, create an index file for the second web service by running:

```
sudo nano index.html
```

7. Add content to the file:

```
<!DOCTYPE html>
<html>
<head>
<title>Website 2</title>
</head>
<body>
<h1>Hello! This is the second sample website.</h1>
</body>
</html>
```

8. Save the changes and exit the file.

9. Build the second service and start the container by running the commands:

```
sudo docker-compose build
```

```
sudo docker-compose up -d
```

Step 3: List Containers

To verify the containers for both services are up and running, list all containers with the command:

```
docker ps -a
```

You should see containers for both web services listed in the output.

Step 4: Set up Reverse Proxy

Next, you need to set up and configure a reverse proxy container. This requires creating multiple files and subdirectories, which should all be stored inside the **proxy** directory.

Therefore, the first step is to create and navigate into the **proxy** directory. To do so, run the commands:

```
mkdir proxy
```

```
cd proxy
```

Configure the Dockerfile

1. Once inside the **proxy** directory, create a Dockerfile for a new custom image:

```
sudo nano Dockerfile
```

2. The file should contain the following:

```
FROM nginx
```

```
COPY ./default.conf /etc/nginx/conf.d/default.conf

COPY ./backend-not-found.html /var/www/html/backend-not-found.html

COPY ./includes/ /etc/nginx/includes/

COPY ./ssl/ /etc/ssl/certs/nginx/
```

The Dockerfile is based on the nginx image. It also copies a number of files from the local machine:

- the default configuration for the proxy service
- an HTML error response
- proxy and SSL configurations and certificates

3. Save and exit the Dockerfile.

Configure the backend-not-found File

Create an index file for a **not found** response:

```
sudo nano backend-not-found.html
```

Add the content:

```
<html>
<head><title>Proxy Backend Not Found</title></head>
<body>
<h2>Proxy Backend Not Found</h2>
</body>
</html>
```

3. Save the changes and close the file.

Configure the default.conf File

1. Create the **default.conf** file inside the proxy directory:

```
sudo nano default.conf
```

2. Add the content:

```
# web service1 config.
server {
listen 80;
listen 443 ssl http2;
server_name example1.test;
```

```
# Path for SSL config/key/certificate
ssl_certificate /etc/ssl/certs/nginx/example1.crt;
ssl_certificate_key /etc/ssl/certs/nginx/example1.key;
include /etc/nginx/includes/ssl.conf;

location / {
include /etc/nginx/includes/proxy.conf;
proxy_pass http://example_app_1;
}

access_log off;
error_log /var/log/nginx/error.log error;
}

# web service2 config.
server {
listen 80;
listen 443 ssl http2;
server_name example2.test;

# Path for SSL config/key/certificate
ssl_certificate /etc/ssl/certs/nginx/example2.crt;
ssl_certificate_key /etc/ssl/certs/nginx/example2.key;
```

```
include /etc/nginx/includes/ssl.conf;

location / {
include /etc/nginx/includes/proxy.conf;
proxy_pass http://example2_app_1;
}

access_log off;
error_log /var/log/nginx/error.log error;
}

# Default
server {
listen 80 default_server;

server_name _;
root /var/www/html;

charset UTF-8;

error_page 404 /backend-not-found.html;
location = /backend-not-found.html {
allow all;
}
```

```
location / {
return 404;
}

access_log off;
log_not_found off;
error_log /var/log/nginx/error.log error;
}
```

The configuration consists of two web services – **example1.test** and **example2.test**. Both server components listen to port 80 and direct Nginx to the appropriate SSL certificate.

Configure the docker-compose.yml File

1. Create a new **docker-compose.yml** file for the proxy service:

```
sudo nano docker-compose.yml
```

2. Copy and paste the following content into the file:

```
version: '2'
services:
proxy:
```

```
build: ./
networks:
- example1
- example2
ports:
- 80:80
- 443:443

networks:
example1:
external:
name: example1_default
example2:
external:
name: example2_default
```

In this file, you connect the proxy and the external networks (the web services **example1** and **example2**). Also, ports 80/443 of the proxy service are bound to ports 80/443 of the Docker host.

Generate Keys and Certificates

1. Start by creating a subdirectory (**ssl**) inside the **proxy** folder:

```
mkdir ssl
```

2. Navigate into the **ssl** subdirectory with:

```
cd ssl
```

3. Next, run each command listed below to create the required files:

```
touch example1.crt
```

```
touch example1.key
```

```
touch example2.crt
```

```
touch example2.key
```

4. Then, use OpenSSL to generate keys and certificates for your web services. For the first web service (**example1**), run the command:

```
sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout example1.key -out example
1.crt
```

The command generates a 2048 bit RSA private key and stores it into the **example1.key** file.

You will also be asked to provide some information that is incorporated into the certificate request. You can leave some of the fields blank.

5. Repeat the process to generate keys and certificates for the second web service (**example2**). Type in the following command in the terminal window:

```
sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout example2.key -out example
2.crt
```

This generates a 2048 bit RSA private key for **example2** and saves it in the **example2.key** file.

Edit the Proxy and SSL Configuration

1. Exit out of the **ssl** subdirectory and back into **proxy**. To do so, use the command:

```
cd ..
```

2. Then, create a new subdirectory under the name **includes**:

```
mkdir includes
```

3. Navigate into **includes** and create the files **proxy.conf** and **ssl.conf**:

```
cd includes
```

```
touch proxy.conf
```

```
touch ssl.conf
```

4. Next, open the **proxy.conf** file:

```
sudo nano proxy.conf
```

5. Add the following configuration:

```
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;
proxy_buffering off;
proxy_request_buffering off;
proxy_http_version 1.1;
proxy_intercept_errors on;
```

6. Save and exit **proxy.conf**.

7. Open the ssl.conf file:

```
sudo nano ssl.conf
```

8. Paste the following lines in the file:

```
ssl_session_timeout 1d;
ssl_session_cache shared:SSL:50m;
ssl_session_tickets off;

ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
ssl_ciphers 'ECDHE-ECDSA-CHACHA20-POLY1305:ECDHE-RSA-CHACHA20-POLY1305:ECDHE-
```

```
ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES256-GCM-
SHA384:ECDHE-RSA-AES256-GCM-SHA384:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-
GCM-SHA384:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA256:ECDHE-ECDSA-
AES128-SHAECDHE-RSA-AES256-SHA384:ECDHE-RSA-AES128-SHA:ECDHE-ECDSA-AES256-
SHA384:ECDHE-ECDSA-AES256-SHA:ECDHE-RSA-AES256-SHA:DHE-RSA-AES128-SHA256:
DHE-RSA-AES128-SHA:DHE-RSA-AES256-SHA256:DHE-RSA-AES256-SHA:ECDHE-ECDSA-
DES-CBC3-SHA:ECDHE-RSA-DES-CBC3-SHA:EDH-RSA-DES-CBC3-SHA:AES128-GCM-SHA256:
AES256-GCM-SHA384:AES128-SHA256:AES256-SHA256:AES128-SHA:AES256-SHA:DES-
CBC3-SHA:!DSS';
ssl_prefer_server_ciphers on;
```

Edit the hosts File

1. Return to the **proxy** directory:

```
cd
```

2. Open the **/etc/hosts** file:

```
sudo nano etc/hosts
```

3. Add the following lines to map the hostnames of the web services to the private IP address of the Docker hosts:

```
10.0.2.15 example1.test
10.0.2.15 example2.test
```

4. Save the changes and close the file.

Note: If you don't know your host's IP address, refer to How to Find Or Check Your IP Address In Linux.

Step 5: Start Reverse Proxy

1. With the configuration files in place, use the docker-compose command to build the container:

```
sudo docker-compose build
```

2. Then, run the container:

```
sudo docker-compose up -d
```

3. Verify you know have three containers – two web services and one reverse proxy container:

```
sudo docker ps -a
```

Step 6: Check Whether Reverse Proxy is Working

Use the `curl` command to check whether the web services and the reverse proxy are working properly. Each domain should respond with the appropriate output.

1. Run the following command to check the first web service:

```
sudo curl example1.test
```

The output responds with the HTML created for **example1.test**.

2. Then, verify the second web service is working with the reverse proxy with:

```
sudo curl example2.test
```

The output responds with the **example2.test** HTML.

Conclusion

This tutorial showed you how to use Docker to set up two sample web services and an Nginx reverse proxy for them.

If you want to learn more about Nginx reverse proxy outside the Docker environment, take a look at How To Set Up & Use NGINX As A Reverse Proxy.

Next you should also read

SysAdmin, Databases, MySQL

MySQL Data Types

March 4, 2021

Data types are important to understand as a name and a data type define

DevOps And Development, Virtualization

How to Fix “helm has no deployed releases” Error

March 4, 2021

Web Servers, SysAdmin

A Comprehensive Guide to Using Snap Packages on Ubuntu

March 4, 2021

Snap allows you to install apps to your system in a

SysAdmin, Bare Metal Servers

How to Use the usermod Command in Linux

March 4, 2021

The usermod command modifies user account

each column in a database table....

READ MORE

When upgrading from a failed deployment, Helm will produce a "helm has no deployed releases" error. This...

READ MORE

contained manner, keeping their dependencies within the...

READ MORE

details: username, password, home directory location, shell, and...

READ MORE

Author

Sofija Simic

Sofija Simic is an aspiring Technical Writer at phoenixNAP. Alongside her educational background in teaching and writing, she has had a lifelong passion for information technology. She is committed to unscrambling confusing IT concepts and streamlining intricate software installations.

How to Fix “helm has no deployed releases“ Error

A Comprehensive Guide to Using Snap Packages on Ubuntu

BACKUP AND RECOVERY

BARE METAL SERVERS

WEB SERVERS

NETWORKING

DATABASES

RFP TEMPLATE

CAREERS

CLOUD SERVICES

SOLUTIONS

LOCATIONS

Live Chat Get a Quote Support | 1-855-330-1509 Sales | 1-877-588-5918

Privacy Policy GDPR Sitemap