

My Wordpress-Docker

Paul Huygen

31st July 2023

Contents

<i>1</i>	<i>Introduction</i>	<i>1</i>
1.1	Tasks to be performed	2
<i>2</i>	<i>Preliminaries</i>	<i>2</i>
<i>3</i>	<i>Construct the docker image</i>	<i>3</i>
<i>4</i>	<i>Connect to the back-up of the original source</i>	<i>5</i>
<i>5</i>	<i>Install the software</i>	<i>5</i>
5.1	Install the Mysql database	5
5.2	Restore Apache	7
5.3	restore files from the backup2l repo	7
<i>6</i>	<i>Run the Docker image</i>	<i>8</i>
<i>7</i>	<i>Indexes</i>	<i>8</i>
7.1	Filenames	8
7.2	Macro's	8
7.3	Variables	8

Abstract

This document generates a Docker image that contains te Wordpress website of CLTL.nl. It is derived from a back-up fom the actual website.

1 Introduction

This document constructs a restoration of a Wordpress website from a back-up. A Docker image with the restoration of the site is made. It serves the following purposes:

1. Provide proof that the back-up is complete. In other words, it is guaranteed that, if a disaster happens, the website can be restored.
2. Describe how it works. After a while, the knowledge on how to use the software instruments for the restoration becomes rusty. Hopefully this document provides clear instructions.
3. The software on the original site has not been updated for a long time. When a Docker with a duplicate exists, this can serve as a template to test upgrading.

1.1 Tasks to be performed

First we have to do the following:

1. Construct a solid back-up procedure.
2. Construct a back-up procedure for the server that runs the website. The information on the back-up is the basis for the restored image.
3. Construct a Docker image with the correct version of the operating system on it. The original server uses a very old version of Ubuntu linux: 14.04.5 LTS, *Trusty Tahr*.
4. Install the database and restore the WordPress part on it.
5. Install the Apache web-server.
6. Restore the WordPress site. This is of a very old version too.
7. Test whether it works.

When this works, we will bother about upgrading the operating system and Wordpress to the latest version.

Basically it works as follows: A script `doit` translates the Nuweb sources. The Nuweb sources produce a Dockerfile and an installation script. The Dockerfile is used to generate a Docker image based on the correct version of Ubuntu. In the image the installation script has to be started manually, because several operations require manual interventions that I don't know how to surpass.

The installation script installs Mysql and loads the Wordpress database in it.

This is the script `doit`. It has four parts:

1. Preliminaries: Make sure that everything that is needed is available.
2. Generate the scripts from the Nuweb source.
3. Build the Docker image.

```
"doit" 2a≡
    #!/bin/bash
    # doit -- generate the image
    <preliminary checks 2b, ... >
    make sources
    <make scripts executable 8b>
    <build the Docker image 4c>
    ◇
```

2 Preliminaries

In order to be able to run the CLTL website, the Docker image needs several resources, e.g. a dump of the orinal Wordpress database. Put these resources in a “transfer directory” and copy that directory into the new image.

```
<preliminary checks 2b> ≡
    mkdir -p transferdir
    ◇
```

Fragment defined by 2b, 3ac, 6a.

Fragment referenced in 2a.

Some of the resources are secret, e.g. passwords. Therefore they cannot be shared on e.g. Github. Write the secrets in a file **secrets** and put it in the transfer directory.

```

< preliminary checks 3a > ≡
    if
        [ ! -e "transferdir/secrets" ]
    then
        cp ../my_secrets/secrets transferdir/ 2>/dev/null
        if
            [ $? .gt. 0 ]
        then
            echo "File with secrets not present"
            exit 1
        fi
    fi
◇

```

Fragment defined by 2b, 3ac, 6a.

Fragment referenced in 2a.

To help you, generate a "template" script, that looks like the real script but with fake information in it.

```

"secret_template" 3b≡
    < password stuff 6b >
◇

```

If all is well, a directory with a partial back-up of the original cltl website server has been mounted on **/home/paul/mnt/b2l**. Let's check that out.

```

< preliminary checks 3c > ≡
    if
        [ ! -e "/home/paul/mnt/b2l/root" ]
    then
        echo "back-up directory b2l has not been mounted."
        exit 1
    fi
◇

```

Fragment defined by 2b, 3ac, 6a.

Fragment referenced in 2a.

3 Construct the docker image

The following rudimentary Dockerfile generates an image for an Ubuntu 14.04 server. After you run the image, you can contact it via the terminal. When you stop it, all modifications are lost.

```
"Dockerfile" 4a≡
  FROM ubuntu:14.04
  EXPOSE 80
  < copy stuff to the image 4b, ... >
  < "run" commands in Dockerfile 5b, ... >
  CMD ["/bin/bash"]
  ◇
```

```
< copy stuff to the image 4b > ≡
  COPY transferdir /root/transferdir/
  ◇
```

Fragment defined by 4be.
Fragment referenced in 4a.

```
< build the Docker image 4c > ≡
  docker build -t ubuntu_docker .
  ◇
```

Fragment referenced in 2a.

To restore the Wordpress-site on the image, run a script with instructions. Load the secret information into this script.

```
"restore" 4d≡
  #!/bin/bash
  source /root/transferdir/secrets
  < restore instructions 5a, ... >
  ◇
```

```
< copy stuff to the image 4e > ≡
  COPY --chmod=775 restore /root/restore
  ◇
```

Fragment defined by 4be.
Fragment referenced in 4a.

We would like to eventually install debian packages. So, let us first prepare for that. I could not yet find a way to do this automatically, without manual intervention. So, after generation of the image, the user has to run it, get access to it and start the **restore** script manually.

When we are at it, let us also install packages that we know we will need later on: sshfs, mysql, php, apache2.

```

⟨ restore instructions 5a ⟩ ≡
    apt-get update
    apt-get upgrade
    apt-get install sshfs apache2 mysql-common mysql-server php5-mysql
    ◇

```

Fragment defined by 5ad, 7abc, 8c.

Fragment referenced in 4d.

We need to supply secret stuff, e.g. passwords. Do this in a file that is not shared in Github. What follows is a template.

```

⟨ “run” commands in Dockerfile 5b ⟩ ≡
    ◇

```

Fragment defined by 5bc.

Fragment referenced in 4a.

4 Connect to the back-up of the original source

5 Install the software

Install the software to run the [cltl](#) website. As far as I am aware now, the website needs 1) Wordpress; 2) Mysql database and 3) Apache. Much of the software can be obtained from the Ubuntu repository. So, update **apt** to enable it to install packages.

We can not perform the installation automatically yet. Too much elements require manual intervention, e.g. to fill in passwords.

```

⟨ “run” commands in Dockerfile 5c ⟩ ≡
    ◇

```

Fragment defined by 5bc.

Fragment referenced in 4a.

5.1 Install the Mysql database

Install the Debian packages for Mysql and load it with the database that has been back-upped from the original source. Unfortunately I do not yet know how to install the packages without the need for intervention by a human operator. You have to provide a root password. Please note this password in a safe place.

Install and start mysql:

```

⟨ restore instructions 5d ⟩ ≡
    apt-get install mysql-common mysql-client mysql-server
    service mysql start
    ◇

```

Fragment defined by 5ad, 7abc, 8c.

Fragment referenced in 4d.

Install the back-up of the Mysql database. This back-up is in a file `wordpress_db.sql` that has been generated by `mysqldump`. It should be located in a “sister” directory `wpbak`.

1. Put this back-up file in the transfer directory to have it imported into the image.
2. Generate the Wordpress user in Mysql.
3. Restore the database in Mysql.

Put the back-up file into the transfer directory. If it cannot be found, stop further actions.

```
< preliminary checks 6a > ≡
if
  [ ! -e "transferdir/wordpress_db.sql" ]
then
  cp ../wpbak/wordpress_db.sql transferdir/ 2>/dev/null
  if
    [ $? .gt. 0 ]
  then
    echo "Mysql dumpfile wordpress_db.sql not present"
    exit 1
  fi
fi
◇
```

Fragment defined by [2b](#), [3ac](#), [6a](#).

Fragment referenced in [2a](#).

Generate the Wordpress user with a password. During the installation you had to fill in a main Mysql password by hand. Hopefully you still remember this password now. Hopefully the password of the Wordpress-user has been written in the secret file.

```
< password stuff 6b > ≡
MYSQL_ROOT_PASSWORD="root_password"
WP_MYSQL_PASSWORD="mysql_password"
◇
```

Fragment referenced in [3b](#).

If all is OK, we are ready to generate the Wordpress user and allow it to handle the Wordpress database.

restore instructions 7a ≡

```
echo
echo Generate the Wordpress database user and grant privileges.
echo
MYSQL_ROOT_USER="root"
MYSQL_WP_USERNAME="wordpress_usr"
DATABASE_NAME="wordpress_db"
mysql -u $MYSQL_ROOT_USER -p$MYSQL_ROOT_PASSWORD <<EOF
CREATE USER '$MYSQL_WP_USERNAME'@'localhost' IDENTIFIED BY '$WP_MYSQL_PASSWORD';
GRANT ALL PRIVILEGES ON $DATABASE_NAME.* TO '$MYSQL_WP_USERNAME'@'localhost';
FLUSH PRIVILEGES;
EOF
◇
```

Fragment defined by 5ad, 7abc, 8c.

Fragment referenced in 4d.

Defines: DATABASE_NAME 7b, MYSQL_ROOT_USER Never used, MYSQL_WP_USERNAME 7b.

Next, create the database as Wordpress-user and restore it from the backup.

restore instructions 7b ≡

```
echo
echo Restore $DATABASE_NAME from the back-up
echo
mysql -u $MYSQL_WP_USERNAME -p$WP_MYSQL_PASSWORD -e "create database $DATABASE_NAME"
mysql -u $MYSQL_WP_USERNAME -p$WP_MYSQL_PASSWORD $DATABASE_NAME < transferdir/wordpress_db.sql
◇
```

Fragment defined by 5ad, 7abc, 8c.

Fragment referenced in 4d.

Uses: DATABASE_NAME 7a, MYSQL_WP_USERNAME 7a.

5.2 Restore Apache

To restore apache, we will install the Ubuntu Apache package and then restore the Apache config files in /etc.

restore instructions 7c ≡

```
apt-get install apache2
◇
```

Fragment defined by 5ad, 7abc, 8c.

Fragment referenced in 4d.

5.3 restore files from the backup2l repo

To access the b2l repo backup2l needs a configuration file.

6 Run the Docker image

```
"run_the_image" 8a≡
    #!/bin/bash
    # run_the_image -- start a container with the cltl image
    docker run -it --mount type=bind,src=/home/paul/mnt/b2l,target=/backup ubuntu_docker
    ◇
```

```
<make scripts executable 8b> ≡
    chmod 775 run_the_image
    ◇
```

Fragment referenced in [2a](#).

Make sure that the mount-point in the docker image exists.

```
<restore instructions 8c> ≡
    mkdir -p /backup
    ◇
```

Fragment defined by [5ad](#), [7abc](#), [8c](#).

Fragment referenced in [4d](#).

7 Indexes

7.1 Filenames

"Dockerfile" Defined by [4a](#).

"doit" Defined by [2a](#).

"restore" Defined by [4d](#).

"run_the_image" Defined by [8a](#).

"secret_template" Defined by [3b](#).

7.2 Macro's

<build the Docker image [4c](#)> Referenced in [2a](#).

<copy stuff to the image [4be](#)> Referenced in [4a](#).

<make scripts executable [8b](#)> Referenced in [2a](#).

<password stuff [6b](#)> Referenced in [3b](#).

<preliminary checks [2b](#), [3ac](#), [6a](#)> Referenced in [2a](#).

<restore instructions [5ad](#), [7abc](#), [8c](#)> Referenced in [4d](#).

<"run" commands in Dockerfile [5bc](#)> Referenced in [4a](#).

7.3 Variables

DATABASE_NAME: [7a](#), [7b](#).

MYSQL_ROOT_USER: [7a](#).

MYSQL_WP_USERNAME: [7a](#), [7b](#).