

My Wordpress-Docker

Paul Huygen

10th June 2023

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 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**.
2. Install the database and restore the WordPress part on it.
3. Install the Apache web-server.
4. Restore the WordPress site. This is of a very old version too.
5. Test whether it works.

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

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 sur-pass.

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
  ⟨ build the Docker image 3c ⟩
  ◇
```

2 Preliminaries

In order to be able to run the CLTL website, the Docker image need 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 2bc, 8c.
Fragment referenced in 2a.

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

```
⟨ preliminary checks 2c ⟩ ≡
  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 2bc, 8c.
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" 2d≡
  ⟨ password stuff 6c ⟩
  ◇
```

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" 3a≡
  FROM ubuntu:14.04
  EXPOSE 80
  < copy stuff to the image 3b, ... >
  < "run" commands in Dockerfile 3g, ... >
  CMD ["/bin/bash"]
◇
```

```
< copy stuff to the image 3b > ≡
  COPY transferdir /root/transferdir/
◇
```

Fragment defined by 3be, 4c, 6a.
Fragment referenced in 3a.

```
< build the Docker image 3c > ≡
  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" 3d≡
  #!/bin/bash
  source /root/transferdir/secrets
  < restore instructions 3f, ... >
◇
```

```
< copy stuff to the image 3e > ≡
  COPY --chmod=775 restore /root/restore
◇
```

Fragment defined by 3be, 4c, 6a.
Fragment referenced in 3a.

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.

```
< restore instructions 3f > ≡
  apt-get update
  apt-get upgrade
◇
```

Fragment defined by 3f, 4a, 5b, 6bd, 7ad, 8ab.
Fragment referenced in 3d.

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 3g > ≡
◇
```

Fragment defined by 3g, 5a.
Fragment referenced in 3a.

4 Connect to the back-up of the original source

We used `backup2l` to back-up the server, so we need this program in our image to restore things:

```
< restore instructions 4a > ≡
    apt-get install backup2l
◇
```

Fragment defined by 3f, 4a, 5b, 6bd, 7ad, 8ab.

Fragment referenced in 3d.

Defines: `backup2l` 5b.

Mount the directory of the back-up on directory `/backup`. We assume that the back-up files that `backup2l` has made are available on directory `/home/paul/mnt/b2l` on the Docker host. The `docker run` instruction contains a mount option that connects this directory to the local `/backup` directory.

The program `backup2l` needs a configuration-file that tells it where the back-up is located. Generate such a config-file. To be safe I filled in all the variables from the original config-file, although most of them are propably not needed.

```
"cltl_bak.conf" 4b≡
    FOR_VERSION=1.5
    VOLNAME="all"
    SRCLIST=(/etc /root /home /var/mail /usr/local /srv)
    SKIPCOND=(-false)
    BACKUP_DIR="/backup"
    MAX_LEVEL=3
    MAX_PER_LEVEL=8
    MAX_FULL=2
    GENERATIONS=1
    CREATE_CHECK_FILE=1
    PRE_BACKUP ()
    { # Nothing to do
    }

    # This user-defined bash function is executed after a backup is made
    POST_BACKUP ()
    {
        # Nothing to do
    }
    AUTORUN=0
    SIZE_UNITS="G"
◇
```

File defined by 4b, 5c.

```
< copy stuff to the image 4c > ≡
    COPY cltl_bak.conf /root/
◇
```

Fragment defined by 3be, 4c, 6a.

Fragment referenced in 3a.

5 Install the software

Install the software to run the `cltl` website.

\langle “run” commands in Dockerfile 5a $\rangle \equiv$

◇

Fragment defined by 3g, 5a.

Fragment referenced in 3a.

6 Connect to the back-up of the original source

We used `backup2l` to back-up the server, so we need this program in our image to restore things:

\langle restore instructions 5b $\rangle \equiv$

```
apt-get install backup2l
```

◇

Fragment defined by 3f, 4a, 5b, 6bd, 7ad, 8ab.

Fragment referenced in 3d.

Defines: `backup2l` 4a.

Mount the directory of the back-up on directory `/backup`. We assume that the back-up files that `backup2l` has made are available on directory `/home/paul/mnt/b2l` on the Docker host. The `docker run` instruction contains a mount option that connects this directory to the local `/backup` directory.

The program `backup2l` needs a configuration-file that tells it where the back-up is located. Generate such a config-file. To be safe I filled in all the variables from the original config-file, although most of them are propably not needed.

```
"cltl_bak.conf" 5c≡
FOR_VERSION=1.5
VOLNAME="all"
SRCLIST=(/etc /root /home /var/mail /usr/local /srv)
SKIPCOND=(-false)
BACKUP_DIR="/backup"
MAX_LEVEL=3
MAX_PER_LEVEL=8
MAX_FULL=2
GENERATIONS=1
CREATE_CHECK_FILE=1
PRE_BACKUP ()
{ # Nothing to do
}

# This user-defined bash function is executed after a backup is made
POST_BACKUP ()
{
    # Nothing to do
}
AUTORUN=0
SIZE_UNITS="G"
◇
```

File defined by 4b, 5c.

```

< copy stuff to the image 6a > ≡
COPY cltl_bak.conf /root/
◇

```

Fragment defined by 3be, 4c, 6a.
 Fragment referenced in 3a.

7 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.

7.1 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 6b > ≡
apt-get install mysql-common mysql-client mysql-server
service mysql start
◇

```

Fragment defined by 3f, 4a, 5b, 6bd, 7ad, 8ab.
 Fragment referenced in 3d.
 Uses: `mysql-client` 8b, `mysql-common` 8b, `mysql-server` 8b.

```

< password stuff 6c > ≡
MYSQL_ROOT_PASSWORD="root_password"
# WP_USERNAME="wordpress_usr"
WP_MYSQL_PASSWORD="password"
◇

```

Fragment referenced in 2d.

```

< restore instructions 6d > ≡
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 $DATABASE_NAME.* TO '$WP_USERNAME'@'localhost';
FLUSH PRIVILEGES;
EOF
◇

```

Fragment defined by 3f, 4a, 5b, 6bd, 7ad, 8ab.
 Fragment referenced in 3d.

```

⟨ restore instructions 7a ⟩ ≡
    mysql -u $MYSQL_WP_USERNAME -p -e "create database" $DATABASE_NAME
    mysql -u $MYSQL_WP_USERNAME -p $DATABASE_NAME < transferdir/wordpress_db.sql
    ◇

```

Fragment defined by 3f, 4a, 5b, 6bd, 7ad, 8ab.
 Fragment referenced in 3d.

7.2 restore files from the backup2l repo

To access the b2l repo backup2l needs a configuration file

8 Run the Docker image

```

"run_the_image" 7b ≡
    #!/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 macros executable 7c ⟩ ≡
    chmod 775 run_the_image
    ◇

```

Fragment never referenced.

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

```

⟨ restore instructions 7d ⟩ ≡
    mkdir -p /backup
    ◇

```

Fragment defined by 3f, 4a, 5b, 6bd, 7ad, 8ab.
 Fragment referenced in 3d.

9 Install packages

9.1 Mysql

The cltlserver contains the following packages with the string “mysql” in its name:

- libcrypt-mysql-perl
- libdbd-mysql-perl
- libmysqlclient-dev
- libmysqlclient18:amd64
- mysql-client-5.5
- mysql-client-core-5.5
- mysql-common
- mysql-server
- mysql-server-5.5
- mysql-server-core-5.5
- php5-mysql

Probably we only need to install `mysql-common`, `mysql-server` and `mysql-client`.

`< restore instructions 8a > ≡`
`◇`

Fragment defined by [3f](#), [4a](#), [5b](#), [6bd](#), [7ad](#), [8ab](#).
 Fragment referenced in [3d](#).

`< restore instructions 8b > ≡`
`apt-get install mysql-common mysql-server mysql-client`
`◇`

Fragment defined by [3f](#), [4a](#), [5b](#), [6bd](#), [7ad](#), [8ab](#).
 Fragment referenced in [3d](#).
 Defines: `mysql-client` [6b](#), `mysql-common` [6b](#), `mysql-server` [6b](#).

Generate the Wordpress database from a back-up dump named `(wordpress_db.sql)`. Transfer this dumpfile to the image and load Mysql with its content.

`< preliminary checks 8c > ≡`
`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 [2bc](#), [8c](#).
 Fragment referenced in [2a](#).

10 Indexes

10.1 Filenames

`"ctl1_bak.conf"` Defined by [4b](#), [5c](#).
`"Dockerfile"` Defined by [3a](#).
`"doit"` Defined by [2a](#).
`"restore"` Defined by [3d](#).
`"run_the_image"` Defined by [7b](#).
`"secret_template"` Defined by [2d](#).

10.2 Macro's

`< build the Docker image 3c >` Referenced in [2a](#).
`< copy stuff to the image 3be, 4c, 6a >` Referenced in [3a](#).
`< make macros executable 7c >` Not referenced.
`< password stuff 6c >` Referenced in [2d](#).
`< preliminary checks 2bc, 8c >` Referenced in [2a](#).
`< restore instructions 3f, 4a, 5b, 6bd, 7ad, 8ab >` Referenced in [3d](#).
`< "run" commands in Dockerfile 3g, 5a >` Referenced in [3a](#).

10.3 Variables

backup2l: [4a](#), [5b](#).

mysql-client: [6b](#), [8b](#).

mysql-common: [6b](#), [8b](#).

mysql-server: [6b](#), [8b](#).