

DOCKER AND WSL 2 IN EDUCATIONAL USE



ICT Project
Degree Programme in Business Information Technology
2019
Iiro Vainio, Joni Komulainen

Author Iiro Vainio, Joni Komulainen
Year 2019

Subject Docker and WSL 2 in educational use

Supervisor Tero Keso

ABSTRACT

Häme University of Applied Sciences has been using virtual machines provided by vCommander on its Linux focused courses. It would be cheaper and, in some cases, easier to run the Linux environment directly on students' own devices. In this documentation we are researching WSL 2 as a platform as an option for vCommander's virtual machines and how suitable it would be in BYOD compatible education along with Docker.

In chapter six we are going through how to prepare Windows device for Docker on WSL 2. The chapter seven focuses on how to install MariaDB, WordPress, phpMyAdmin, Confluence and Nextcloud inside Docker containers using WSL 2 platform, how to harden Apache and MySQL and how to backup the containers. We are also discussing why Fail2ban is not working in this environment.

Our aim in this project was to create a guide on how to get these services working and secured on Docker using WSL 2. The whole Docker Compose file for launching all the services at the same time is in the end of the document (Docker-compose.yml, appendix 3), but it will work as a whole only with Docker Desktop 2.1.6.1 or higher. With earlier versions, go along with the instructions.

During this project we managed to get almost everything working correctly except for Fail2ban due to Docker Desktop's networking. We also left messaging servers out. In the end we came to conclusion that Docker on WSL 2 platform is a good option when considering BYOD compatible education, but the reliability of this environment is not on that level yet.

Keywords Docker, WSL 2, BYOD, Windows, Linux

Pages 41 pages including appendices 6 pages

CONTENTS

VOCABULARY	1
1 INTRODUCTION	1
2 PROJECT GOALS	2
3 BACKGROUND INFORMATION	2
4 METHODS, TOOLS AND TECHNIQUES	3
5 THEORY AND EXISTING KNOWLEDGE.....	3
5.1 What are WSL and WSL 2.....	3
5.2 What is Docker	4
6 PREPARING THE DEVICE	4
6.1 WSL and WSL 2.....	5
6.2 Docker Desktop WSL 2 Backend	5
6.3 Windows Terminal	6
7 WEB AND MESSAGING SERVERS -COURSE.....	6
7.1 WordPress, MariaDB and phpMyAdmin.....	9
7.2 Backup	10
7.3 SSH keys and SCP.....	11
7.4 Nextcloud	12
7.5 Confluence	14
7.6 Fail2ban	21
7.7 Hardening Apache	24
7.8 Hardening MySQL.....	26
8 RESULTS AND ANALYSIS OF THE PROJECT.....	28
9 RISK ANALYSIS.....	29
10 RECOMMENDATIONS	29
SOURCES	31
APPENDIX.....	35

VOCABULARY

Backup: The data is copied and transferred elsewhere so it can be restored later in case of data loss.

Brute-force attack: An attack, where the attacker is trying to make as much traffic as possible to the servers slowing the servers down and preventing other users' access to the service.

BYOD: Bring your own device

Confluence: A content collaboration software by Atlassian that allows teams to collaborate and share knowledge easily.

Container: Allows developer to package up an application with all the parts it needs and deploy it as one package.

Docker: A tool designed to make it easier to create, deploy and run applications by using containers

Docker Desktop: A software for Windows and MacOS providing a development environment for building, shipping and using dockerized applications.

Hardening: Reducing software's vulnerability and making it more secure.

Fail2ban: A software for preventing brute-force attacks to the services.

Linux and Windows: Operating systems.

MariaDB: Database management system based on MySQL.

MySQL: An open-source system for database management

Nextcloud: A software for hosting personal cloud storage services. Comparable to Dropbox.

PhpMyAdmin: Free software intended to deal with administration of MySQL and MariaDB

Postfix: A free mail server for routing and delivering e-mail.

SCP: Secure copy protocol. It allows transferring files from device to another securely.

SSH key: Secure Shell. Network protocol for operating network services safely over an unsecured network. Key will authenticate the remote computer and allow it to authenticate the user without using password.

Ubuntu: One of many Linux distributions or distros.

Virtual machine (VM): A virtual environment to run operating systems and software. Virtual machines work as regular computers so the experience for the user is similar.

Windows Insider: Open software testing program by Microsoft that allows users to download pre-release builds of Windows.

WordPress: Software for building a website.

Windows Subsystem for Linux: Microsoft's solutions to run Linux binary executables on Windows 10 natively.

1 INTRODUCTION

The focus of the project is on testing new technologies and applying them to the Häme University of Applied Sciences (HAMK) BYOD environment. We will research possibilities for HAMK to use WSL 2 and Docker in some of the courses. The vision of this project is to create a guide or definition how Docker works in educational use. We are researching if Docker and WSL 2 are practical with BYOD. Research will be done with latest versions of Windows (Insider Preview), WSL 2 and Docker. In addition to the guide, our objective is answer to the following questions:

- Is WSL 2 well-functioning and stable enough as a platform?
- How Docker works on WSL 2 platform?
- Is Docker functional in WLAN networks located in HAMK (eduroam and HAMKvisitor)?
- Is Docker suitable for educational use?

This project's client is HAMK and our contact person is Tero Keso. Project authors will be two students Iiro Vainio and Joni Komulainen. In this project we are going to use Windows 10 versions from 1909 to 19033. During the project Docker Desktop replaced Tech Preview with Backend and the required Windows version is constantly changing. At the time of writing, the first stable Windows release including WSL 2 will be version 2004 (Ghacks.net, 2019)

2 PROJECT GOALS

First, we will look at the contents of Web and Messaging Servers -course and then we will create a plan for who will implement what parts of the course. In the beginning we must install WSL 2 and Docker Desktop to our computers. Then we start going through the course assignments.

The goal of the project is to create a comprehensive and detailed manual about using WSL 2 and Docker Desktop together, which could be used in addition to vCommander in one or several courses on students' own devices. We will be working with Web and Messaging Servers -course materials and trying to get all the parts done with Docker Desktop and WSL 2 systems.

Our client's main questions are whether WSL 2 is well-functioning and stable enough as a platform, how Docker works on WSL 2 platform, is Docker functional in WLAN networks located in HAMK (eduroam and HAMKvisitor) and is Docker functional in educational use?

3 BACKGROUND INFORMATION

HAMK has been using virtual machines provided by vCommander, which is a hybrid cloud management solution developed by Embotics (Finances Online, n.d.). Our client wants a good alternative option to vCommander in one or more courses on students' own devices. They have been working with WSL 2 and Docker before, but never at the same time using both elements. We need to learn more on how to use Docker.

None of us has worked with Docker Desktop and WSL 2 before, so we need to learn to use them and how we are going to use all needed elements together. This is actual research with these elements together. The latest stable Windows 10 version released during the project is 1909.

Docker, Docker Desktop and WSL 2 all have official guides and documentations. Information about these elements can also be found from forums on the Internet. However, there seems to be no documentation about how to use these elements together and if there will be compatibility issues.

4 METHODS, TOOLS AND TECHNIQUES

In this project we are using laptops running Windows 10 Insider Preview versions from 1909 to 19033, Docker Desktop Edge versions from 2.1.4.0 to 2.1.6.1, WSL 2, Ubuntu 18.04 as a default distro on WSL 2 and Windows Terminal (preview). Inside Linux we are using Nano text editor. It seems that the first stable Windows release including WSL 2 support will be 2004 (Ghacks.net, 2019).

At the time of writing, Windows 10 version 1909 is the latest stable Windows release. We are running insider versions from fast update ring. WSL 2 hasn't been officially released yet and based on the information we have so far, it will be released in the beginning of 2020. WSL 2 support in Docker Desktop has not been implemented on the stable release yet, but it's available on Edge releases and the required Windows version is constantly changing.

In this project we are going to install WordPress, MariaDB, phpMyAdmin, Nextcloud and Confluence. We are also hardening Apache and MariaDB, backing up containers and transferring backups to server using SCP and SSH keys.

We were working also with Postfix and Fail2ban, but they are not currently working with this combination because of how Docker Desktop routes network.

There are some information and documentation about Docker Desktop and WSL 2 available, but in our case, we are going to combine these elements and that is something no one has yet documented. In problem solving, we will be using official documentations and forum conversations and try the suggested solutions and modify them to fit into our environment if needed.

5 THEORY AND EXISTING KNOWLEDGE

In this project we are running Windows Insider Preview versions from 1909 to 19033. Platforms that we are using in this project are WSL 2, Docker Desktop Edge from 2.1.5.0 to 2.1.6.1 and Ubuntu 18.04.

5.1 What are WSL and WSL 2?

WSL stands for Windows Subsystem for Linux. WSL is Microsoft's solution to run Linux binary executables on Windows 10 natively. In practice this

means that you can run Linux environment inside your Windows 10 machine and run some Linux programs and scripts inside this environment. (Ubuntu Wiki, n.d.)

Microsoft didn't include any Linux kernel code in original WSL. In WSL 2 this has however changed. They did changes to the architecture and WSL 2 is more like a very light weight virtual machine than just a compatibility layer so now all Linux programs should be able to run on WSL 2 system. (Ubuntu Wiki, n.d.) According to Microsoft, WSL 2's backend redesign has significantly increased speed when compared to WSL in certain tasks and the filesystem read and write speeds can be even twenty times faster (Microsoft, 2019a).

5.2 What is Docker?

Docker is a software that can run another software in small environments called containers, that packs the software and all its needed dependencies. By using Docker, you can move these containers easily from device to another and start these programs quickly. This can also solve the "it works on my machine" -problem, which is common in IT, because you are basically moving the whole machine, including the program, instead of moving just the program itself.

When using virtual machines, you are virtualizing the hardware to run OS on top of it. When using Docker, you are virtualizing the OS. So, Docker containers are more light weight to run and move around than virtual machines. (Docker, n.d.g)

Docker Desktop is a tool to run Docker on Windows and Apple machines (Armstrong, 2018).

6 PREPARING THE DEVICE

This chapter contains a guide on what you will need to run Docker Desktop and WSL 2 on your own machine. You must be running Windows 10 version 19018 or higher. WSL is mandatory to enable WSL 2 feature for Windows. You must install Docker Desktop WSL 2 Backend (previously Tech Preview). After that you install Ubuntu and Windows Terminal from Microsoft Store. Make the installations in following order. (Docker, n.d.c)

6.1 WSL and WSL 2

WSL is required to enable WSL 2. According to Microsoft's official instruction, enable WSL by running the following command in PowerShell as an administrator:

```
Enable-WindowsOptionalFeature -Online -FeatureName Microsoft-Windows-Subsystem-Linux
```

After running the command, open Microsoft store, search for WSL and install Ubuntu 18.04. After installation, launch Ubuntu 18.04 and create username and password. (Microsoft, 2018)

Make sure that WSL and Virtual Machine Platform are enabled by running the following commands in PowerShell as an administrator and reboot machine after installation.

```
dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /all /norestart
```

```
dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart
```

Check the name of the distro by running command:

```
wsl -l
```

Set your Linux distribution to use WSL 2 by running command. This can take a while, so be patient.

```
wsl --set-version <Distro> 2
```

Make sure that previous command did change the version from 1 to 2 by running command:

```
wsl -l -v
```

WSL 2 can be made your default architecture by running the following command and all the distros installed after this command will be running WSL 2 by default.

```
wsl --set-default-version 2
```

(Microsoft, 2019b)

6.2 Docker Desktop WSL 2 Backend

Download Docker Desktop WSL 2 Backend from Docker's website and install it by running the installer. Set the Docker Desktop to use Linux containers. After installation, enable WSL 2 based engine by clicking settings

from Docker Desktop menu, go to general tab and select “*Enable the experimental WSL 2 based engine*”.

On the same tab deselect option “*Start Docker Desktop when you log in*”. At the time of writing, automatic start causes Docker to misbehave and, in some cases, show crash reports. Docker will still start automatically during the boot but when the device is rebooted everything should be working correctly.

Make sure you are using WSL 2 by running the following command in PowerShell:

```
docker context use wsl
```

In Docker settings on *General* tab, select “*Enable the experimental WSL 2 based engine*” and click “*Enable and restart*”. Then go to *Resources* tab, click on *WSL integration* and turn on *Ubuntu-18.04*. And then again click *Enable and restart*. (Docker, n.d.c)

6.3 Windows Terminal

Windows Terminal is not a necessity, but it’s a handy tool since it can run both Windows and Linux commands in separate tabs at the same time.

Windows Terminal can be installed from Microsoft Store (Microsoft, n.d.). You can use your default Subsystem Linux from the Terminal by running command:

```
wsl
```

7 WEB AND MESSAGING SERVERS -COURSE

Web and Messaging Servers is one of the second year’s courses in Degree Programme in Business Information Technology. The goal of this course is to learn how to plan and put into practice a server environment, maintain it and install services and products in a controlled manner. The following guided installations and configurations were a part of this course and these exercises are being used in testing our environment.

7.1 WordPress, MariaDB and phpMyAdmin using Docker Desktop 2.1.6.0 or lower

WordPress, MariaDB and phpMyAdmin were installed by using a single Docker Compose script. Login to the Subsystem Linux and check if directory */opt/data* already exists. If it doesn’t, create it. Then create directory

wordpress, and file *docker-compose.yml* inside the directory with the following contents. Our compose file will pull and start containers including WordPress and MariaDB and link these two containers together so WordPress can use MariaDB as its database. The script will also set the password for the MySQL root user and link the container's port 80 to the host machine's port 8080:

```
wordpress:
  image: wordpress
  links:
    - wordpress_db:mysql
  ports:
    - 8080:80
wordpress_db:
  image: mariadb
  volumes:
    - /opt/data
  environment:
    MYSQL_ROOT_PASSWORD: root007
```

(DigitalOcean, 2015; Docker, n.d.a)

Go to the *wordpress* directory and run the script with command `docker-compose up -d` and go to *localhost:8080* with your web browser. If you can see WordPress installation page (see PHOTO 1 below), install WordPress and after installation copy the following contents to the *docker-compose.yml* file and run it again to install phpMyAdmin. Note: use same password as earlier in this same file.

If you see a database error (see PHOTO 2 below), copy the following contents to the *docker-compose.yml* file and run it again to install phpMyAdmin, go to address *localhost:8181*, login and create user and database for WordPress and then go back to *localhost:8080* to complete WordPress installation. Note: use same password as earlier in this same file

```
phpmyadmin:
  image: phpmyadmin/phpmyadmin
  links:
    - wordpress_db:mysql
  ports:
    - 8181:80
  environment:
    MYSQL_USERNAME: root
    MYSQL_ROOT_PASSWORD: root007
    PMA_HOST: mysql
    PMA_PORT: 3306
```

(DigitalOcean, 2015; Sergiu, 2017)

The script needs to be run like this in two pieces at the first time when creating the containers. Otherwise something will break, and you won't be able to install WordPress and login to phpMyAdmin. The cause of this

problem is unknown. When the containers already exist, they can be started by running `docker-compose.yml`.

At the time of writing, the results of the script have been inconsistent and sometimes WordPress requires phpMyAdmin first to create database and user for WordPress. Sometimes WordPress will go directly to its installation page and installing phpMyAdmin before completing WordPress installation will break WordPress.

The original tutorial for this installation (DigitalOcean, 2015) used an outdated and unofficial version of phpMyAdmin which had bugs that have been fixed in later versions. We changed the script to use the official up to date phpMyAdmin image. We had to add lines `PMA_HOST` and `PMA_PORT` to the script to make it work. According to Stack Overflow user Sergiu's reply (Sergiu, 2017), phpMyAdmin will search for database from localhost instead of the MariaDB container without these variables.

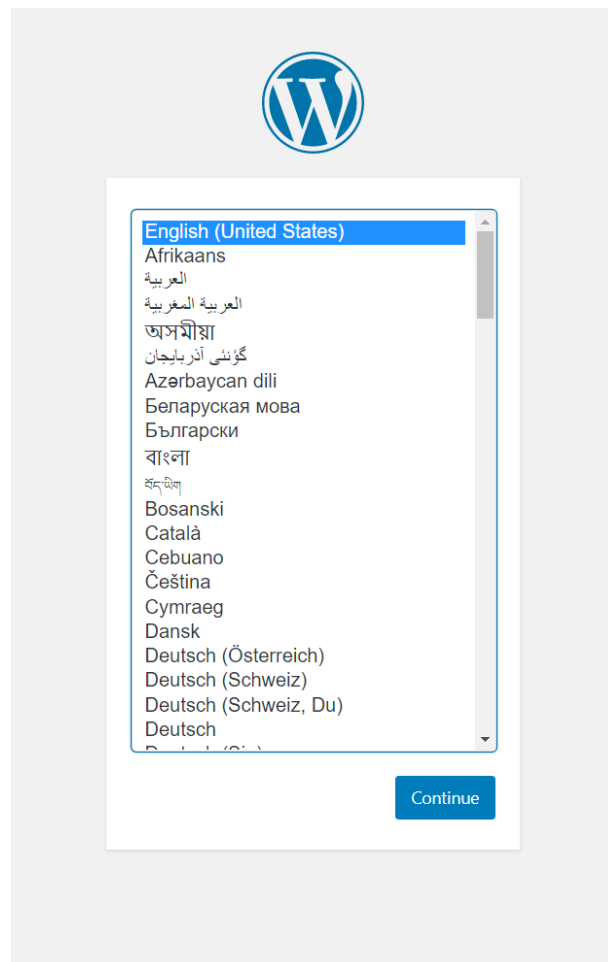


Photo 1. WordPress installation page after running `docker-compose.yml`

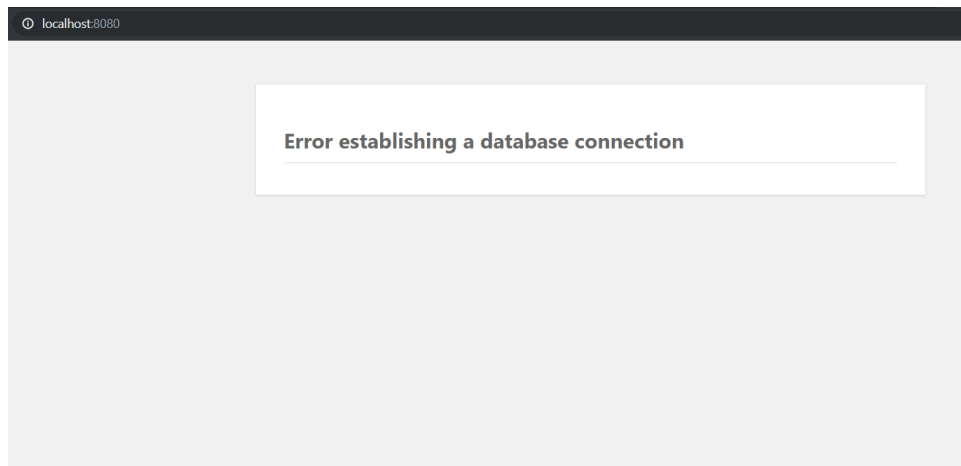


Photo 2. Database error after running *docker-compose.yml*

7.2 WordPress, MariaDB and phpMyAdmin using Docker Desktop 2.1.6.1 or higher

WordPress, MariaDB and phpMyAdmin were installed by using a single Docker Compose script. Login to the Subsystem Linux and check if directory */opt/data* already exists. If it doesn't, create it. Then create directory *wordpress*, and file *docker-compose.yml* inside the directory with the following contents. Our compose file will pull and start containers including WordPress and MariaDB and link these two containers together so WordPress can use MariaDB as its database. The script will also set the password for the MySQL root user and link the container's port 80 to the host machine's port 8080. We are also adding PhpMyAdmin to control the database through graphical environment in our browser at port 8181:

```
wordpress:
  image: wordpress
  links:
    - wordpress_db:mysql
  ports:
    - 8080:80
wordpress_db:
  image: mariadb
  volumes:
    - /opt/data
  environment:
    MYSQL_ROOT_PASSWORD: root007
phpmyadmin:
  image: phpmyadmin/phpmyadmin
  links:
    - wordpress_db:mysql
  ports:
    - 8181:80
  environment:
    MYSQL_USERNAME: root
    MYSQL_ROOT_PASSWORD: root007
    PMA_HOST: mysql
    PMA_PORT: 3306
```

(DigitalOcean, 2015; Sergiu, 2017; Docker, n.d.a)

To get everything working properly, go to *localhost:8080* and you will see “This page isn’t working” message. The containers haven’t fully started yet and the linking is still in progress. Wait till you see a database error (see PHOTO 3). Then go to *localhost:8181* and login. After you are in, create database and user for WordPress and then continue WordPress installation at *localhost:8080*.

The original tutorial for this installation (DigitalOcean, 2015) used an outdated and unofficial version of phpMyAdmin which had bugs that have been fixed in later versions. We changed the script to use the official up to date phpMyAdmin image. We had to add lines PMA_HOST and PMA_PORT to the script to make it work. According to Stack Overflow user Sergiu's reply (Sergiu, 2017), phpMyAdmin will search for database from localhost instead of the MariaDB container without these variables.

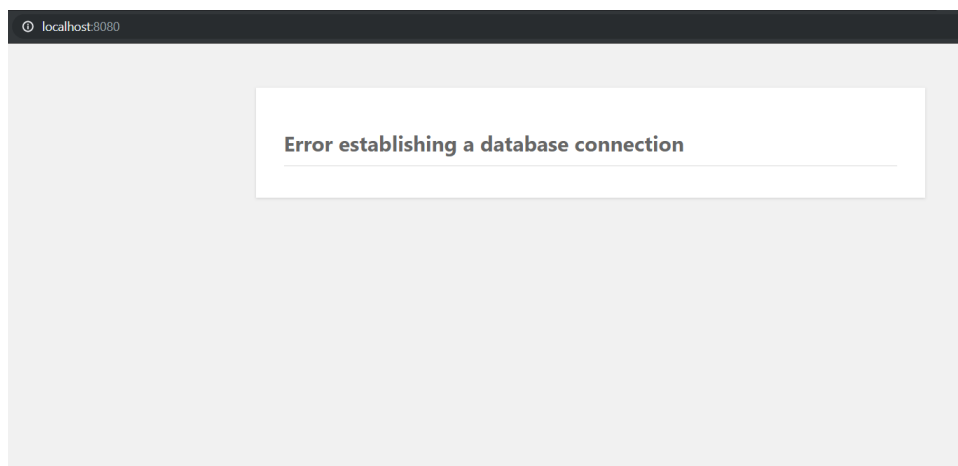


Photo 3. Database error after running *docker-compose.yml*

7.3 Backup

Making backups of your database when it’s located inside a container differs from the way it would be done on a “physical server”. Originally in the course we would have made `mysqldump`, compress it into `bz2`-file and move it to another server by using `SCP` pull or push commands. In this case we are going to create a Docker image of the container containing our database, compress it into a `tar`-file and then move it to another server using `SCP` command. Docker containers save their data to volumes, so we need

to make a backup of the volume as well. This process was done a little differently without Docker Desktop and the steps can be seen in appendix 2, but in theory this guide should work as well.

First, we need to create a Docker image of the container storing our database (wordpress_wordpress_db_1) using its ID with this command:

```
docker commit -p container-ID backup-name
```

Next, we will create a tar-file that contains our docker image of our database. This tar-file will be moved to another server later.

```
docker save -o backup-name.tar backup-name
```

(Mathews, 2016)

Now we need to store the container's data volume in a tar-file as well. We diverged from the original instructions here, because the volumes are located differently. In *docker-compose.yml* we mounted */opt/data* to our database container. Run the following command to compress this folder into a tar-file:

```
sudo tar -czvf name-of-archive.tar /opt/data
```

7.4 SSH keys and SCP

To use SCP in script easily, we will be generating SSH key pair to login to another server without inserting password.

There are some issues with SSH on Subsystem Linux, so we are going to use SSH directly on Windows to transfer the backups to another server. To install SSH on your Windows device, go to Settings > Apps > Apps and features > Manage optional features. Search for *OpenSSH Server* and *OpenSSH Client* and install them both.

Next you need to configure SSH by running these commands in PowerShell or Windows Terminal as an administrator:

```
Start-Service sshd

# OPTIONAL but recommended:
Set-Service -Name sshd -StartupType 'Automatic'

# Confirm the Firewall rule is configured. It should
be created automatically by setup.
Get-NetFirewallRule -Name *ssh*

# There should be a firewall rule named "OpenSSH-
Server-In-TCP", which should be enabled
```

```
# If the firewall does not exist, create one
New-NetFirewallRule -Name sshd -DisplayName 'OpenSSH
Server (sshd)' -Enabled True -Direction Inbound -Pro-
tocol TCP -Action Allow -LocalPort 22
```

Make sure that SSH is installed and enabled on your server as well. (Microsoft, 2019c)

Create SSH keys on your Windows device by running next command in a regular PowerShell or Windows Terminal:

```
ssh-keygen
```

Hit enter on all the questions. Now you should be able to copy your public key (id_rsa.pub) to the server using SCP. Remember to connect your device to eduroam or HAMKvisitor to have access to vCommander's virtual machines if you are using them as servers.

```
scp C:\Users\username\.ssh/id_rsa.pub username@server-
ip:/home/username/.ssh/authorized_keys
```

Now you should be able to use SSH from your Windows machine to your Ubuntu server without using password. (Grande, 2019)

Next, we need to move or copy the backup files from our Subsystem Linux to our Windows to SCP them to the server. We can do it with command:

```
cp /path/to/the/backup-file.tar /mnt/c/Us-
ers/username/backup-folder/backup-file.tar
```

Or if you want to do it using graphical tools, open File Explorer on your Windows machine and write the following path to the address field:

```
\\wsl$
```

There you can see your distros and browse their folders and files.

The last step is to push the backup files to the server using SCP:

```
scp -r C:\Users\username\folder-containing-backups
username@server-
ip:/path/where/you/want/to/store/backups
```

7.5 Nextcloud

We are going to use *docker-compose.yml* file to install Nextcloud. Add the following lines to the existing *docker-compose.yml* at wordpress directory:

```
nextcloud:
  image: nextcloud
  ports:
    - 8282:80
```



```
links:
  - wordpress_db:mysql
volumes:
  - nextcloud:/var/www/html
```

Save the file, run `docker-compose up -d` and go to `localhost:8282` with your browser. There you will see an installation wizard. (Nextcloud, n.d.)

Before continuing installation in browser, go inside your Nextcloud container:

```
docker exec -it wordpress_nextcloud_1 /bin/bash
```

Inside the container create folder `/var/nextcloud` and give it right permissions:

```
mkdir /var/nextcloud
chown www-data:www-data /var/nextcloud
```

Now you can exit the container and continue in the browser.

Give username and password for a new Nextcloud admin account, choose MySQL/MariaDB as your database and fill in the information needed. With database user and database password use the same username and password created earlier when installing WordPress, MariaDB and phpMyAdmin (See PHOTO 4). Our host differs from the original instructions (Keso, 2019) and we need to use the service name of our database container, as GitHub user Isidorelsou mentioned (Isidorelsou, 2018).

```
Data folder: /var/nextcloud
Database user: root
Database password: root007
Database name: nextcloud
Host: mysql
```

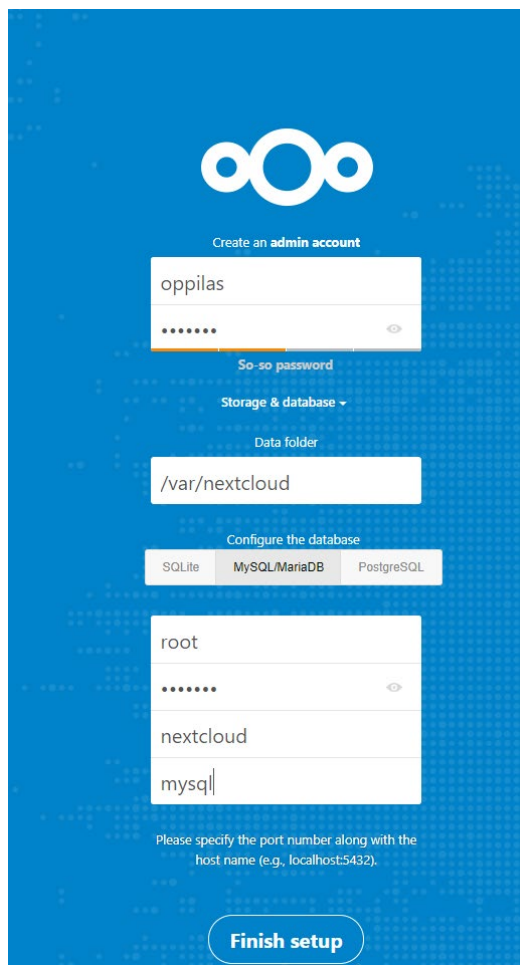


Photo 4. Nextcloud installation wizard in browser

Finish setup and in a while, you should be on the front page of Nextcloud. Please wait patiently, because this may take time to complete. (Keso, 2019)

7.6 Confluence

To install Confluence, run the following command in your WSL console:

```
docker run -v /data/your-confluence-home:/var/atlassian/application-data/confluence --name="confluence" -d -p 8090:8090 -p 8091:8091 atlassian/confluence-server
```

Now you can continue installation in your browser at *localhost:8090*. (Atlassian, n.d.a)

On the first "Get apps" screen just press next without selecting any (See PHOTO 5).

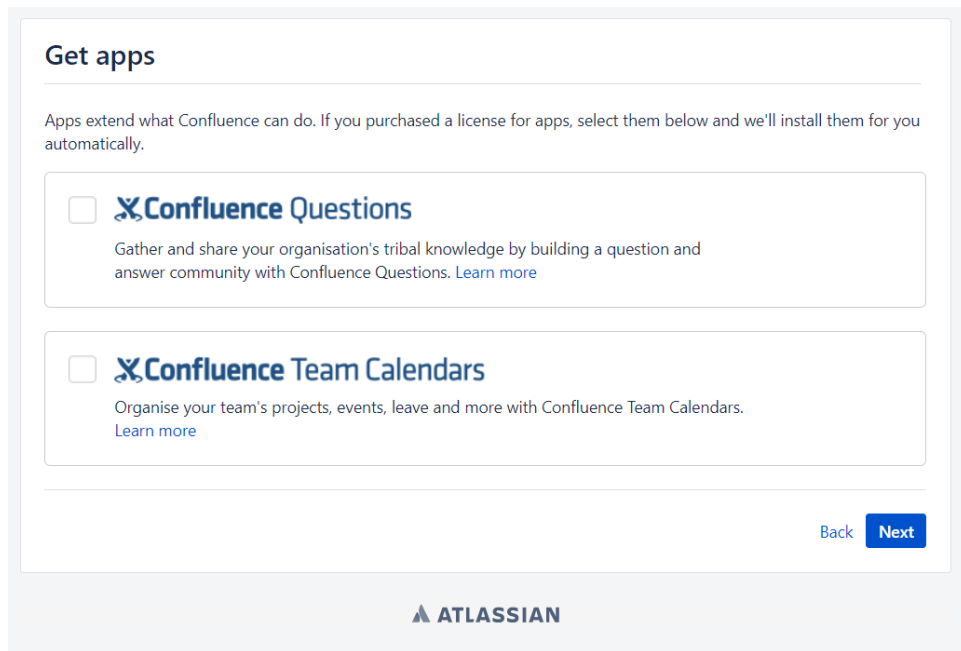


Photo 5. Confluence “Get apps” screen

On the next screen copy your server ID somewhere (e.g. Notepad) and press option “Get an evaluation license” (PHOTO 6).

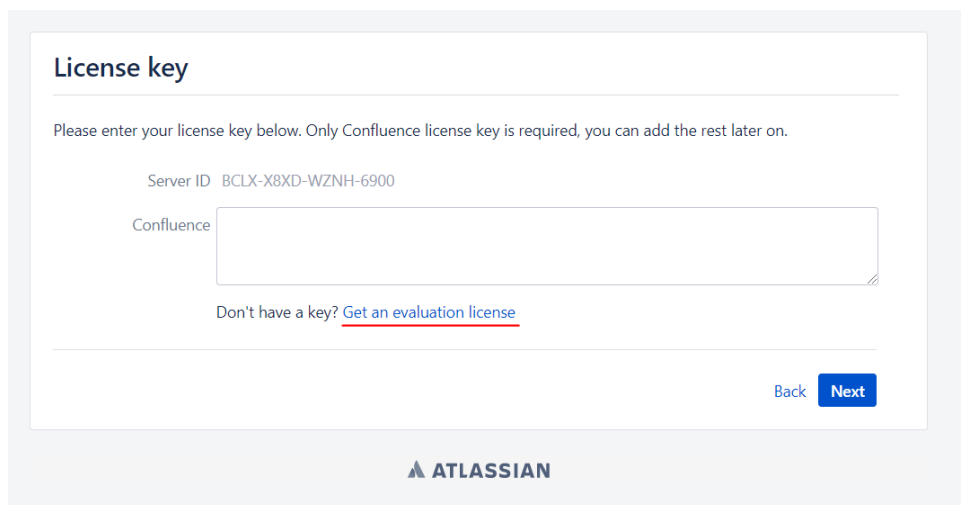


Photo 6. Press option “Get an evaluation license”

Sign up with your school’s email account. On page *New trial license*, select *Confluence (Server)*, make sure that the server ID matches your server’s ID and press *Generate License* (PHOTO 7).

New Trial License

Product:

License type:

Confluence (Server)	Confluence (Data Center)
<ul style="list-style-type: none"> • Manage the entire application on your own servers or virtual machines. • Deployable to a single server. 	Everything with server plus: <ul style="list-style-type: none"> • Active-active clustering for true high availability and uninterrupted access. • High performance under high load and at peak times • Disaster recovery.
<input checked="" type="checkbox"/>	<input type="checkbox"/>

Organization:

Your instance is: up and running not installed yet

Server ID:

Please note we only provide trial support for 90 days per product.

By clicking "Generate License" below, you agree to the Atlassian [Software License Agreement](#) and [Privacy Policy](#).

Photo 7. Select server, check the server ID and press *Generate License*

Confirm the installation of the license to localhost by pressing *Yes* and then click *Next*. On the next screen select *My own database* and continue. (Keso, 2019)

Now we need to configure Confluence to use our MariaDB database. From the dropdown menu, select MySQL and download the MySQL driver (PHOTO 8).

Set up your database

Where should Confluence store its data? [Learn more about connecting Confluence to a database](#)

Database type:

Confluence needs a driver to connect to MySQL. You'll need to:

1. Download the [MySQL driver](#)
2. Drop the .jar file in /opt/atlassian/confluence/confluence/WEB-INF/lib
3. Restart Confluence and continue the setup process.

[Back](#)

ATLASSIAN

Photo 8. Select MySQL and download MySQL driver

Extract the downloaded file and copy `mysql-connector-java.x.y.z.jar` file to the confluence container and then restart the container by running these following commands from the WSL terminal:

```
docker cp /mnt/c/users/username/path-to/mysql-connector-java.x.y.z.jar confluence:/opt/atlassian/confluence/lib
```

```
docker restart confluence
```

Now, go to phpMyAdmin at `localhost:8181` and create database `confluence` and set encoding to `utf8_bin`

After creating a new database, go to your WSL console and check the IP of your database container:

```
docker inspect name-of-the-container
```

Go back to your Confluence setup page at `localhost:8090` and you should be able to fill your database settings (see PHOTO 9):

```
Database type: MySQL
Setup type: simple
Hostname: IP of your database container
Port: 3306
Database name: confluence
Username: root
Password: root007
```

Then hit *Test connection*.

Set up your database

Where should Confluence store its data? [Learn more about connecting Confluence to a database](#)

Database type: MySQL

Setup type: Simple By connection string
Add additional parameters using the database url

Hostname*: 172.17.0.2
Hostname or IP address of your database server

Port*: 3306
TCP port number for your database server

Database name*: confluence

Username*: root

Password:

Test connection

Back Next

ATLASSIAN

Photo 9. Confluence database information

You will get an error about incorrect isolation level (PHOTO 10).

Set up your database

Incorrect isolation level X
Your database must use 'READ-COMMITTED' as the default isolation level. [Learn more](#)

Where should Confluence store its data? [Learn more about connecting Confluence to a database](#)

Photo 10. Isolation level error

This can be fixed by going through these following steps. Go inside your database container:

```
docker exec -it wordpress_wordpress_db_1 /bin/bash
```

Inside the container install Nano text editor:

```
apt update
apt install nano
```

Open *my.cnf* file in Nano text editor:

```
nano /etc/mysql/my.cnf
```

Add the following line into [mysqld] section (PHOTO 11):

```
transaction-isolation=READ-COMMITTED
```

```
[mysqld]
#
# fix for incorrect isolation level error
transaction-isolation=READ-COMMITTED
#
# * Basic Settings
#
#user           = mysql
pid-file        = /var/run/mysqld/mysqld.pid
socket          = /var/run/mysqld/mysqld.sock
port           = 3306
basedir         = /usr
datadir         = /var/lib/mysql
tmpdir          = /tmp
```

Photo 11. [mysqld] section and added line in *my.cnf*

(Atlassian, n.d.b)

Exit the container and reboot it:

```
docker-compose stop
```

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

Now go back to your browser and click *Test connection* again and you should see a message about successful database connection. You can now click *Next* (PHOTO 12).

Set up your database

Where should Confluence store its data? [Learn more about connecting Confluence to a database](#)

Database type: MySQL

Setup type: Simple By connection string
Add additional parameters using the database url

Hostname*: 172.17.0.2
Hostname or IP address of your database server

Port*: 3306
TCP port number for your database server

Database name*: confluence

Username*: root

Password:

Test connection Success! Database connected successfully.

Back Next

ATLASSIAN

Photo 12. Database successfully connected

On the next page load an example site and after that choose *Manage users and groups within Confluence*.

Create a system administrator account and click *Next* (see PHOTO 13).

Username: admin
Name: *Your name*
Email: oppilas@localhost
Password: root007
Confirm: root007

Configure System Administrator Account

Please configure the system administrator account for this Confluence installation.

Configure Account

Username*

Name*

Email*

Password*

Confirm*

[Next](#)

▲ ATlassian

Photo 13. Configuring system admin account

Now you can start using Confluence by clicking *Start*. (Keso, 2019)

Next time when you are starting Confluence, you can do it by running the following command in WSL terminal:

```
docker start confluence
```

7.7 Fail2ban

Fail2ban on WSL with Docker Desktop is currently not working because of how Docker Desktop routes the network traffic. We tested Fail2ban without Docker Desktop and managed to get it to block our WSL's IP and prevent login attempts from the browser (for instructions without Docker Desktop, see appendix 1).

When using Docker Desktop, the path for the Docker logs is not provided and the exact path is unknown. After we got Fail2ban to read those logs, Fail2ban managed to block the IP of the WordPress container's gateway, but it won't prevent new login attempts. The reason for this behaviour is that Docker Desktop writes that IP to the log instead of our host's or device's IP and we didn't manage to change that.

We also noticed that the entries in the logs are little different from each other (see photos 14-17 below), so our filter files need to be different.

```
172.26.192.1 - - [03/Dec/2019:12:10:31 +0000] "POST /wp-login.php HTTP/1.1" 200 2148 "http://172.26.206.126:8080/wp-login.php"
"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.108 Safari/537.36"
```

Photo 14. Failed WordPress login without Docker Desktop

```
172.18.0.1 - - [03/Dec/2019:12:14:34 +0000] "POST /wp-login.php HTTP/1.1" 200 2672 "http://172.20.45.101:8080/wp-login.php"
"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.108 Safari/537.36"
172.18.0.1 - - [03/Dec/2019:12:14:34 +0000] "GET /favicon.ico HTTP/1.1" 200 228 "http://172.20.45.101:8080/wp-login.php"
"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.108 Safari/537.36"
```

Photo 15. Failed WordPress login using Docker Desktop

```
172.26.192.1 - - [03/Dec/2019:12:13:17 +0000] "POST /wp-login.php HTTP/1.1" 302 1153 "http://172.26.206.126:8080/wp-login.php?
loggedout=true" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.108 Safari/
537.36"
172.26.192.1 - - [03/Dec/2019:12:13:17 +0000] "GET /wp-admin/ HTTP/1.1" 200 15779 "http://172.26.206.126:8080/wp-login.php?log
gedout=true" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.108 Safari/537
.36"
```

Photo 16. Successful WordPress login without Docker Desktop

```
172.18.0.1 - - [03/Dec/2019:12:16:24 +0000] "POST /wp-login.php HTTP/1.1" 302 1152 "http://172.20.45.101:8080/wp-login.php?
loggedout=true&wp_lang=en_US" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/78.0.3904.108 Safari/537.36"
172.18.0.1 - - [03/Dec/2019:12:16:24 +0000] "GET /wp-admin/ HTTP/1.1" 200 15728 "http://172.20.45.101:8080/wp-login.php?
loggedout=true&wp_lang=en_US" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/78.0.3904.108 Safari/537.36"
172.18.0.1 - - [03/Dec/2019:12:16:24 +0000] "GET /favicon.ico HTTP/1.1" 200 228 "http://172.20.45.101:8080/wp-admin/" "Mozilla/5.0
(Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/78.0.3904.108 Safari/537.36"
```

Photo 17. Successful WordPress login using Docker Desktop

To recreate the result using Docker Desktop, we are going to install Fail2ban directly on Subsystem Linux. Go to WSL Terminal and run the following commands:

```
sudo apt update
sudo apt install fail2ban
```

(Bousetta, 2017)

To access Docker's logs, we need to create a separate file for them that we can pass for Fail2ban. Run the following command:

```
/home/username/path/to/Your-json.log
```

Now we need to write the contents of the Docker log file to the file we just created in real time. This can be done with the following command provided by Stack Overflow user Juanmirocks (Juanmirocks, 2018). The command needs to be run every time when starting WSL or Docker.

```
docker logs -f wordpress_wordpress_1 &&
/home/username/path/to/Your-json.log &
```

After this, we need to configure Fail2ban to block brute force attacks to our WordPress container. Open fail2ban configuration file in Nano text editor:

```
sudo nano /etc/fail2ban/jail.conf
```

Add the following lines to the file and edit bantime, findtime and maxretry if needed:

```
[wplogin]

enabled          =          true
port             =          http,https
filter          =          wplogin
logpath         = /home/username/path/to/Your-json.log
banaction       =          docker-action
maxretry        =          3
findtime        =          60
bantime = 60
```

Next, we will create *wplogin.conf* file:

```
sudo nano /etc/fail2ban/filter.d/wplogin.conf
```

(Bousetta, 2017)

And we will add the following lines to the file:

```
[Definition]
failregex = ^<HOST> .* "POST .*wp-login.php
ignoreregex = ^<HOST> .* "GET .*favicon.ico
```

(Duško, 2018)

And then we will create a third file, *docker-action.conf*

```
sudo nano /etc/fail2ban/action.d/docker-action.conf
```

And add the following lines to the file:

```
[Definition]

actionstart = iptables -N f2b-wplogin
iptables -A f2b-wplogin -j RETURN
iptables -I FORWARD -p tcp -m multiport --dports 80 -
j f2b-wplogin

actionstop = iptables -D FORWARD -p tcp -m multiport -
-dports 80 -j f2b-wplogin
iptables -F f2b-wplogin
iptables -X f2b-wplogin

actioncheck = iptables -n -L FORWARD | grep -q 'f2b-
wplogin[ \t]'
```

```
actionban = iptables -I f2b-wplogin 1 -s <ip> -j DROP

actionunban = iptables -D f2b-wplogin -s <ip> -j DROP
```

Now we need to restart Fail2ban:

```
sudo service fail2ban restart
sudo fail2ban-client reload
```

(Bousetta, 2017)

You will probably get an error message about missing log file, which you can fix by creating an empty log file:

```
sudo nano /var/log/auth.log
```

Now restart fail2ban again and it should start with no problems.

When trying to log in with false credentials three times, Fail2ban should block an IP which you can check by running:

```
sudo fail2ban-client status wplogin
```

(Bousetta, 2017)

7.8 Hardening Apache

To harden Apache, we are installing ModSecurity and ModEvasive. These need to be installed inside the container that contains our Apache installation. Apache is in our wordpress container. So, go inside that container:

```
docker exec -it wordpress_wordpress_1 /bin/bash
```

First, let's install ModSecurity by running the following commands:

```
apt update
apt install libapache2-mod-security2
```

You can make sure that the installation was successful by running this command:

```
apachectl -M | grep security
```

It should give you the following output:

```
security2_module (shared)
```

Next, we are going to copy default configuration file to *modsecurity.conf*

```
cp /etc/modsecurity/modsecurity.conf-recommended
/etc/modsecurity/modsecurity.conf
```

Now we need to open the copied file in the text editor and change one line:

```
apt install nano
```

```
nano /etc/modsecurity/modsecurity.conf
```

In this file, change line `SecRuleEngine DetectionOnly` to `SecRuleEngine On` (HostAdvice, 2018b)

Some of the instructions tell to copy rules from GitHub, but it will break WordPress, so we are not going to do that. We are going to move forward to ModEvasive instead. To install ModEvasive, run the following command:

```
apt install libapache2-mod-evasive
```

You can make sure that the installation was successful by running this command:

```
apachectl -M | grep evasive
```

It should give you the following output:

```
evasive20_module (shared)
```

To configure ModEvasive, we need to edit its configuration file in text editor:

```
nano /etc/apache2/mods-enabled/evasive.conf
```

Remove the `#`-symbol from the beginning of each line.

The last thing is to create directory for ModEvasive's logs. So, let's create it and fix its permissions with the following commands:

```
mkdir /var/log/mod_evasive
chown -R www-data:www-data /var/log/mod_evasive
```

(HostAdvice, 2018b)

Now both mods are installed and configured, but these changes will be lost when rebooting the container, so make an image of this running container with command:

```
docker commit -p container-ID backup-name
```

Replace the `wordpress` image from your `docker-compose.yml` with the image you just created (see PHOTO 18), so Docker will start WordPress from the modified image instead of the original one without these mods.

```
wordpress:
  #image: wordpress
  image: wordpress-apache-hardened
  links:
    - wordpress_db:mysql
```

Photo 18. The original wordpress image replaced with the modified one at `docker-compose.yml`

To test these mods, stop the containers and start them again. When you go to `http://localhost:8080/?exec=/bin/bash` in your browser, you will get an error telling that you don't have permission to do that. This is because of ModSecurity.

7.9 Hardening MySQL

MySQL hardening is recommended in production environment. Go inside your database container:

```
docker exec -it mariadb /bin/bash
```

In the container run:

```
mysql_secure_installation
```

(MariaDB, 2011.)

And then answer the questions. We must answer a little bit differently than original installation guide tells us:

```
Switch to unix_socket authentication [Y/n]: n
Change the root password? [Y/n]: n
Remove anonymous users? [Y/n]: y
Disallow root login remotely? [Y/n]: n
Remove test database and access to it? [Y/n]: y
Reload privilege tables now? [Y/n]: y
```

Now you can exit database container by running command `exit`. After exiting container, you must save your settings. Run the following command, it will create and name new image of your changes in the container. And then you must replace your `docker-compose.yml` database image with the new image (see PHOTO 19).

```
docker commit mariadb mysql_secure_installation
```

```

GNU nano 2.9.3                                     docker-compose.yml
wordpress:
  image: wordpress
  container_name: wordpress
  hostname: wordpress
  links:
    - wordpress_db:mysql
  ports:
    - 8080:80
wordpress_db:
  image: mariadb ← mysql_secure_installation
  container_name: mariadb
  hostname: mariadb
  environment:
    MYSQL_ROOT_PASSWORD: root007
phpmyadmin:

```

Photo 19. Replace old database image with the new image

If you answer “y” to “Disallow root login remotely?” your database and phpMyAdmin will give you punch of errors (PHOTO 20).

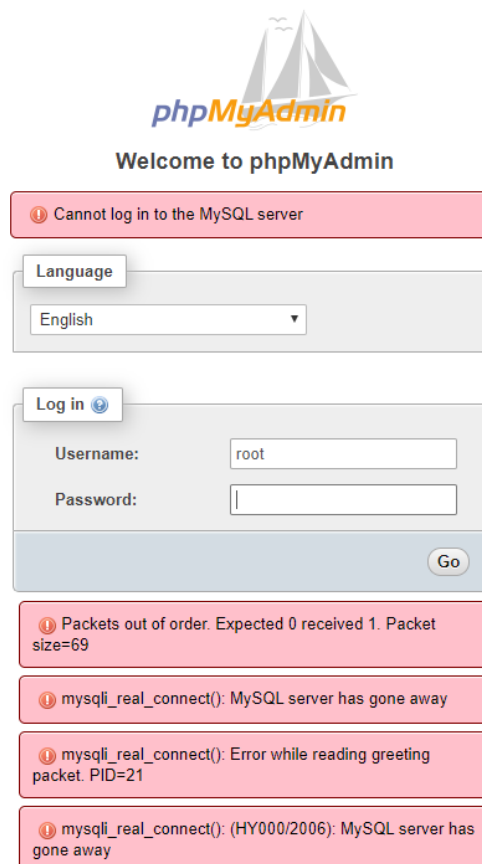


Photo 20. Errors after disallowing root login remotely

8 RESULTS AND ANALYSIS OF THE PROJECT

Using Docker and WSL 2 is a great alternative in BYOD compatible education. Most of the students aren't using Linux on their laptops and they would have a couple of options besides Docker and WSL 2. First one would be using virtual machines provided by the school. This is the method being used currently in HAMK but HAMK's firewall is causing problems for security reasons. Second option would be to run virtual machines locally on students' device or dual boot Linux along the existing OS, but virtual machines are not a great experience on all machines and dual booting requires disk space and, in some cases, the SSD would need to be changed to a bigger one. And last of the options is to buy a whole new device just for Linux. Docker and WSL 2 are a lightweight option that can run on any Windows laptop, which is the most common case. Docker is the easiest and cheapest way for both, school and students, to complete courses requiring Linux. It allows the student to move the containers easily from device to another and it doesn't matter which OS the device itself is running.

We found out that our environment is working surprisingly well when considering that all parts of our environment were still in beta and had never been tested together before. Tasks that are somehow related to networking can be little problematic now though and Docker Desktop's network routing leaves still a lot to be desired. HAMK's own networks, eduroam and HAMKvisitor, caused no problems during the testing since everything ran on our computers locally requiring network connection basically only for downloads.

Most of the Docker containers, guides and tutorials worked well on WSL 2 platform requiring just some minor tweaks. We managed to get most of the *Web and Messaging Servers* course tasks done. Only Fail2ban and Postfix were not successful, but before starting, it was known that they were going to be the most challenging parts. The results were confirmed by doing a fresh install of Windows 10 and all the required components mentioned in this report and re-doing all the steps.

Docker on WSL 2 platform is working, and it is a good option when talking about BYOD, but it has its flaws. Docker Desktop doesn't always boot correctly on first try and the whole device needs to be rebooted for some reason to make Docker working. The networking could be a problem in certain tasks and using Docker Desktop can cause gimmick workarounds to make things work, like in Fail2ban where we couldn't find the direct path to Docker's logs, so we had to read the logs to another file and pass that for Fail2ban.

In educational use this environment is slightly different than a virtual machine in vCommander. In most cases things are needed to be done differently and sometimes it was harder to make it work inside a container than

in vCommander and vice versa. If Docker Desktop on WSL 2 would be used in some courses, the challenge level would depend much on the instructions given and techniques used. For example, getting everything running by using Docker Compose is easier than running the containers without Docker Compose. So, the teachers should decide themselves on how much they will give “complete answers” to the students and if using this environment would be too hard or easy.

So, our answer for the question if WSL 2 and Docker Desktop are suitable for educational use is yes but maybe not yet. However, updates are coming in steady pace and the environment is improving and becoming more reliable.

9 RISK ANALYSIS

The risk analysis was realistic, and we faced some of the problems we expected. In the beginning of the project we were worried about bugs, compatibility issues and updates that could break some functionalities. In the end, we didn't end up running into really major issues. The biggest issues were related to Docker Desktop's way of handling networking and that's the reason why Fail2ban and Postfix didn't succeed.

On one of our laptops we had issues with Docker Desktop not installing itself correctly. We noticed the problem after four weeks of work and did a clean install for the laptop and after that everything was working like it was supposed to.

All our laptops had issues with starting Docker Desktop during the boot and right after it. To prevent this, we deactivated Docker Desktop's automatic start on Windows settings and rebooted our devices right after booting them.

10 RECOMMENDATIONS

The next step would be to go through more of the courses like this to find out the correct ways of making things work. Some of the teachers would need to investigate this and make or modify their courses to match this platform and these tools.

We would also consider whether to keep Docker Desktop or install Docker directly on WSL since in our experiments Docker Desktop made some difficulties and its usefulness was minor giving us no significant benefits whatsoever.

SOURCES

Armstrong, J. (2018). Get to Know Docker Desktop. Blog publication September 17, 2018. Retrieved December 5, 2019, from <https://www.docker.com/blog/get-to-know-docker-desktop/>

Atlassian. (n.d.a). atlassian/confluence-server. Retrieved November 14, 2019, from <https://hub.docker.com/r/atlassian/confluence-server/>

Atlassian. (n.d.b). Database Setup For MySQL. Retrieved November 14, 2019, from <https://confluence.atlassian.com/doc/database-setup-for-mysql-128747.html>

Bousetta, W. (2017). Install and configure Fail2ban with Docker. Blog publication November 21, 2017. Retrieved November 11, 2019, from <https://www.the-lazy-dev.com/en/install-fail2ban-with-docker/>

DigitalOcean. (2015). How To Install WordPress and PhpMyAdmin with Docker Compose on Ubuntu 14.04. Retrieved November 5, 2019, from <https://www.digitalocean.com/community/tutorials/how-to-install-wordpress-and-phpmyadmin-with-docker-compose-on-ubuntu-14-04>

Docker. (n.d.a). Compose file version 1 reference. Retrieved December 10, 2019, from <https://docs.docker.com/compose/compose-file/compose-file-v1/>

Docker. (n.d.b). Docker Desktop for Windows Edge Release notes. Retrieved December 9, 2019, from <https://docs.docker.com/docker-for-windows/edge-release-notes/>

Docker. (n.d.c). Docker Desktop WSL 2 backend. Retrieved November 4, 2019, from <https://docs.docker.com/docker-for-windows/wsl-tech-preview/>

Docker. (n.d.d). Get Docker Engine - Community for Ubuntu. Retrieved November 4, 2019, from <https://docs.docker.com/install/linux/docker-ce/ubuntu/>

Docker. (n.d.e). Install Docker Compose. Retrieved November 4, 2019, from <https://docs.docker.com/compose/install/>

Docker. (n.d.f). Post-installation steps for Linux. Retrieved November 4, 2019, from <https://docs.docker.com/install/linux/linux-postinstall/>

Docker. (n.d.g). What is a Container? Retrieved December 5, 2019, from <https://www.docker.com/resources/what-container>

Duško, V. (2018). Using Fail2ban to block WordPress login attacks. Blog publication March 31, 2018. Retrieved December 3, 2019, from <https://www.prado.lt/using-fail2ban-to-block-wordpress-login-attacks>

Finances Online. (n.d.). vCommander review. Retrieved December 4, 2019, from <https://reviews.financesonline.com/p/vcommander/>

Ghacks.net. (2019). Here is what is new in Windows 10 version 2004. Retrieved December 10, 2019, from <https://www.ghacks.net/2019/12/08/here-is-what-is-new-in-windows-10-version-2004/>

Grande, M. (2019). Key-based Authentication for OpenSSH on Windows. Blog publication 23 May, 2019. Retrieved November 11, 2019, from <https://www.concurrency.com/blog/may-2019/key-based-authentication-for-openssh-on-windows>

HostAdvice. (2018a). How to Secure Apache Web Server with ModEvasive on Ubuntu 18.04 VPS. Retrieved November 13, 2019, from <https://hostadvice.com/how-to/how-to-secure-apache-web-server-with-modevasive-on-ubuntu-18-04-vps/>

HostAdvice. (2018b). How to Setup ModSecurity for Apache on Ubuntu 18.04. Retrieved November 12, 2019, from <https://hostadvice.com/how-to/how-to-setup-modsecurity-for-apache-on-ubuntu-18-04/>

Isidorelsou. (June 23, 2018). Re: Can't set up Nextcloud / SQLSTATE[HY000] [1045] Access denied #374 [Forum post]. Retrieved November 25, 2019, from <https://github.com/nextcloud/docker/issues/374#issuecomment-399693362>

Juanmirocks. (2018, January 19). Re: How to redirect docker container logs to a single file? [Forum post]. Retrieved December 3, 2019, from <https://stackoverflow.com/a/48345336>

Keso, T. (2019). Web ja postipalvelimet. Course in *Liiketoimintaa tukevat verkkopalvelut* module in Häme University of Applied Sciences.

MariaDB. (2011). Mysql_secure_installation. Updated July, 2019. Retrieved November 19, 2019, from https://mariadb.com/kb/en/library/mysql_secure_installation/

Mathews, R. (2016). Docker backup – Easy steps to backup and restore your containers. Blog publication October 15, 2016. Retrieved November 6, 2019, from <https://bobcares.com/blog/docker-backup/>

Microsoft. (2019a). About WSL 2. Updated May 30, 2019. Retrieved December 5, 2019, from <https://docs.microsoft.com/en-us/windows/wsl/wsl2-about>

Microsoft. (2019b). Installation Instructions for WSL 2. Updated May 30, 2019. Retrieved November 4, 2019, from <https://docs.microsoft.com/en-us/windows/wsl/wsl2-install>

Microsoft. (2019c). Installation of OpenSSH For Windows Server 2019 and Windows 10. Updated September 27, 2019. Retrieved November 7, 2019, from https://docs.microsoft.com/en-us/windows-server/administration/openssh/openssh_install_firstuse

Microsoft. (n.d.). microsoft/terminal. Retrieved November 4, 2019, from <https://github.com/microsoft/terminal>

Microsoft. (2018). Windows Subsystem for Linux Installation Guide for Windows 10. Updated July 23, 2018. Retrieved November 4, 2019, from <https://docs.microsoft.com/en-us/windows/wsl/install-win10>

Nextcloud. (n.d.). Nextcloud. Retrieved November 25, 2019, from https://hub.docker.com/_/nextcloud/

Sergiu (2017, September 26). Re: Docker connect phpmyadmin to my MySQL Server [Forum post]. Retrieved November 11, 2019, from <https://stackoverflow.com/a/46433889>

Ubuntu Wiki. (n.d.). WSL. Updated November 21, 2019. Retrieved December 5, 2019, from <https://wiki.ubuntu.com/WSL>

APPENDIX

Appendix 1

FAIL2BAN ON WINDOWS SUBSYSTEM LINUX WITHOUT DOCKER DESKTOP

To install Docker on subsystem Linux without using Docker desktop, go to your WSL terminal and follow Docker's official instructions for Ubuntu (Docker, n.d.d; Docker, n.d.e; see also Docker, n.d.f).

We are going to install Fail2ban directly on Subsystem Linux instead of a Docker container. It probably would work inside a container as well, but we found it easier to run in directly on the host machine. In this project we will be protecting WordPress against brute force attacks.

The installation itself is as simple as running these two commands inside WSL Terminal:

```
sudo apt update
sudo apt install fail2ban
```

After the installation, we need to configure Fail2ban to block brute force attacks to our wordpress container. Open Fail2ban configuration file in Nano text editor:

```
sudo nano /etc/fail2ban/jail.conf
```

Add the following lines to the file and edit `bantime`, `findtime` and `maxretry` if needed:

```
[wplogin]
enabled = true
port = http,https
filter = wplogin
logpath = /var/lib/docker/containers/*/*-json.log
banaction = docker-action
maxretry = 3
findtime = 60
bantime = 60
```

Next, we will create `wplogin.conf` file:

```
sudo nano /etc/fail2ban/filter.d/wplogin.conf
```

And we will add the following lines to the file:

```
[Definition]
failregex = {"log": "<HOST> -. *POST. *wp-login.php. *
ignoreregex =
```

And then we will create a third file, *docker-action.conf*

```
sudo nano /etc/fail2ban/action.d/docker-action.conf
```

And add the following lines to the file:

[Definition]

```
actionstart = iptables -N f2b-wplogin
iptables -A f2b-wplogin -j RETURN
iptables -I FORWARD -p tcp -m multiport --dports 80 -
j f2b-wplogin
```

```
actionstop = iptables -D FORWARD -p tcp -m multiport -
-dports 80 -j f2b-wplogin
iptables -F f2b-wplogin
iptables -X f2b-wplogin
```

```
actioncheck = iptables -n -L FORWARD | grep -q 'f2b-
wplogin[ \t]'
```

```
actionban = iptables -I f2b-wplogin 1 -s <ip> -j DROP
```

```
actionunban = iptables -D f2b-wplogin -s <ip> -j DROP
```

Now we need to restart Fail2ban:

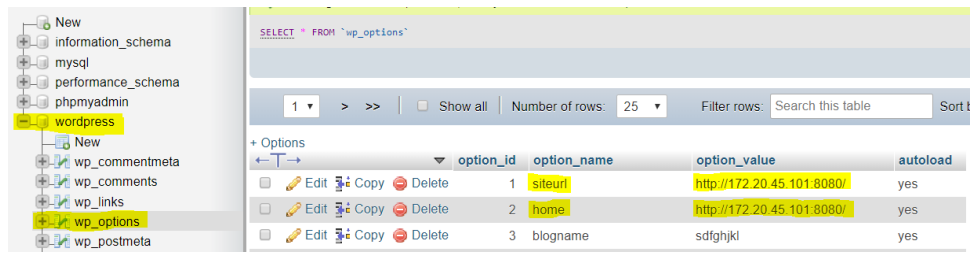
```
sudo service fail2ban restart
sudo fail2ban-client reload
```

You will probably get an error message about missing log file, which you can fix by creating an empty log file:

```
sudo nano /var/log/auth.log
```

Now restart Fail2ban again and it should start with no problems. (Bousetta, W., 2017)

Now Docker is writing Docker0 NIC's IP to the log file and Fail2ban will block that IP. Banning the NIC however does nothing useful. To get Docker writing WSL's IP to the log file, we need to go to phpMyAdmin and change WordPress' site URL and home from *http://localhost:8080* to *http://your-wsl-ip:8080* (PHOTO 21). After this change, when you go to *http://your-wsl-ip:8080* with your browser and login three times using false credentials, Fail2ban will ban your WSL's IP and prevent following attempts for one minute.



SELECT * FROM `wp_options`

1 > >> Show all Number of rows: 25 Filter rows: Search this table Sort t

	option_id	option_name	option_value	autoload
<input type="checkbox"/>	1	siteurl	http://172.20.45.101:8080/	yes
<input type="checkbox"/>	2	home	http://172.20.45.101:8080/	yes
<input type="checkbox"/>	3	blogname	sdfghjkl	yes

Photo 21. Changes to WordPress' siteurl and home in database

BACKUP WITHOUT DOCKER DESKTOP

To install Docker on subsystem Linux without using Docker desktop, go to your WSL terminal and follow Docker's official instructions for Ubuntu (Docker, n.d.d; Docker, n.d.e; see also Docker, n.d.f).

Making backups of your database when it's located inside a container differs from the way it would be done on a "physical server". Originally in the course we would have made mysqldump, compress it into bz2-file and move it to another server by using SCP pull or push commands. In this case we are going to create a Docker image of the container containing our database, compress it into a tar-file and then move it to another server using SCP command. Docker containers save their data to volumes, so we need to make a backup of the volume as well.

First, we need to create a Docker image of the container storing our database (wordpress_wordpress_db_1) using its ID with this command:

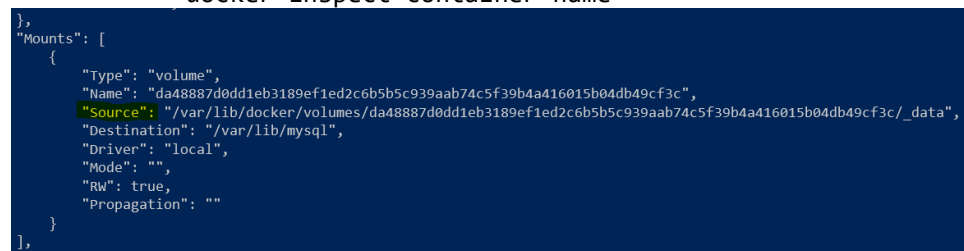
```
docker commit -p container-ID backup-name
```

Next, we will create a tar-file that contains our docker image of our database. This tar-file will be moved to another server later.

```
docker save -o backup-name.tar backup-name
```

Now we need to store the container's data volume in a tar-file as well. Run the following command and take note of the source path under "Mounts" (PHOTO 22):

```
docker inspect container-name
```



```
},
"Mounts": [
  {
    "Type": "volume",
    "Name": "da48887d0dd1eb3189ef1ed2c6b5b5c939aab74c5f39b4a416015b04db49cf3c",
    "Source": "/var/lib/docker/volumes/da48887d0dd1eb3189ef1ed2c6b5b5c939aab74c5f39b4a416015b04db49cf3c/_data",
    "Destination": "/var/lib/mysql",
    "Driver": "local",
    "Mode": "",
    "RW": true,
    "Propagation": ""
  }
],
```

Photo 22. Source path under "Mounts"

Next, we are going to run this command to make the tar-file containing the volume:

```
docker run --rm --volumes-from the-name-of-the-container -v $(pwd):/backup ubuntu tar cvf /backup/name-of-the-backup.tar /mounts/source/path
```

Replace /mounts/source/path with the path you got with the inspect command. (Mathew)

DOCKER-COMPOSE.YML

Here is *docker-compose.yml* which brings up all the pieces together when using Docker Desktop 2.1.6.1 or higher. If running Docker Desktop 2.1.6.0 or lower, install as instructed at chapter seven. First you must create new directory, you can name it whatever you want, example "test". Create *docker-compose.yml* inside your directory and add the following stuff inside the compose. Then you can run compose.

```
mkdir test

sudo nano docker-compose.yml

docker-compose up -d

wordpress:
  image: wordpress
  #container_name: wordpress
  #hostname: wordpress
  links:
    - wordpress_db:mysql
  ports:
    - 8080:80
wordpress_db:
  image: mariadb
  #container_name: mariadb
  #hostname: mariadb
  volumes:
    - /opt/data
  environment:
    MYSQL_ROOT_PASSWORD: root007
phpmyadmin:
  image: phpmyadmin/phpmyadmin
  #container_name: phpmyadmin
  #hostname: phpmyadmin
  links:
    - wordpress_db:mysql
  ports:
    - 8181:80
  environment:
    MYSQL_USERNAME: root
    MYSQL_ROOT_PASSWORD: root007
    PMA_HOST: mysql
    PMA_PORT: 3306
app:
  image: nextcloud
  #container_name: nextcloud
  #hostname: nextcloud
  ports:
    - 8282:80
```

```

links:
  - wordpress_db:mysql
volumes:
  - nextcloud:/var/www/html
service:
  image: atlassian/confluence-server
  #container_name: confluence
  #hostname: confluence
  volumes:
    - confluencedata:/var/atlassian/confluence
  ports:
    - 8090:8090
    - 8091:8091
  links:
    - wordpress_db:mysql

```

(DigitalOcean, 2015; Nextcloud (n.d.); Atlassian (n.d.a))

Now you can see WordPress, phpMyAdmin, Nextcloud and Confluence on your browser. You can continue settings and configurations with instructions above.

Except part of the Confluence mysql connector restart, it must be done differently (page 18). You must exit the container and save the settings by creating new image of the Confluence container. Then replace the old Confluence image with the new “*mysql_connector*” image in your *docker-compose.yml*.

```
docker commit confluence mysql_connector
```

After that, restart your compose and you can continue with the instructions.

```
docker-compose stop
docker-compose up -d
```

In this *docker-compose.yml* has specified container names and hostnames (#-comment sections). They are not mandatory to take with you, but they will simplify your work and your commands will become clearer (PHOTO 23).

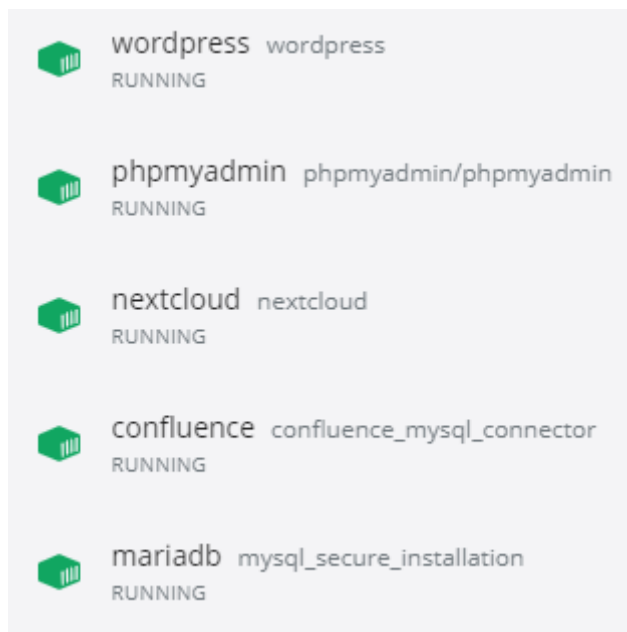


Photo 23. Docker-desktop showing clearly container names when using specified comment sections