<epam>

Containerization. Docker. Lection 3.

Docker Compose

Install Docker Compose

You can run Compose on macOS, Windows, and 64-bit Linux.

Prerequisites

Docker Compose relies on Docker Engine for any meaningful work, so make sure you have Docker Engine installed either locally or remote, depending on your setup.

On desktop systems like Docker Desktop for Mac and Windows, Docker Compose is included as part of those desktop installs.

On Linux systems, first install the Docker Engine for your OS as described earlier.

After that follow instructions (below pages) to install Compose on Linux systems.

To run Compose as a non-root user, see Manage Docker as a non-root user.

Docker Compose

On Linux, you can download the Docker Compose binary from the Compose repository release page on GitHub. Follow the instructions from the link, which involve running the curl command in your terminal to download the binaries. These step-by-step instructions are also included below.

Run this command to download the current stable release of Docker Compose:

```
sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-\
$(uname -m)" -o /usr/local/bin/docker-compose
```

Apply executable permissions to the binary:

```
sudo chmod +x /usr/local/bin/docker-compose
```

Docker Compose

You can use Docker Compose to easily run WordPress in an isolated environment built with Docker containers. This quick-start guide demonstrates how to use Compose to set up and run WordPress. Before starting, make sure you have Compose installed.

Define the project

1. Create an empty project directory.

You can name the directory something easy for you to remember. This directory is the context for your application image. The directory should only contain resources to build that image. This project directory contains a ***docker-compose.yml*** file which is complete in itself for a good starter wordpress project.

2. Change into your project directory.

For example, if you named your directory my_wordpress:

cd my_wordpress/

Docker Compose

Create a **docker-compose.yml** file that starts your WordPress blog and a separate MySQL instance with a volume mount for data persistence:

```
version: '3.3'
services:
   db:
     image: mysql:5.7
     volumes:
     - db_data:/var/lib/mysql
     restart: always
     environment:
       MYSQL_ROOT_PASSWORD: somewordpress
       MYSQL_DATABASE: wordpress
       MYSQL_USER: wordpress
       MYSQL_PASSWORD: wordpress
   wordpress:
     depends_on:
       - db
     image: wordpress:latest
     ports:
       - "8000:80"
     restart: always
     environment:
       WORDPRESS_DB_HOST: db:3306
       WORDPRESS_DB_USER: wordpress
       WORDPRESS_DB_PASSWORD: wordpress
       WORDPRESS_DB_NAME: wordpress
volumes:
    db_data: {}
```

Docker Compose

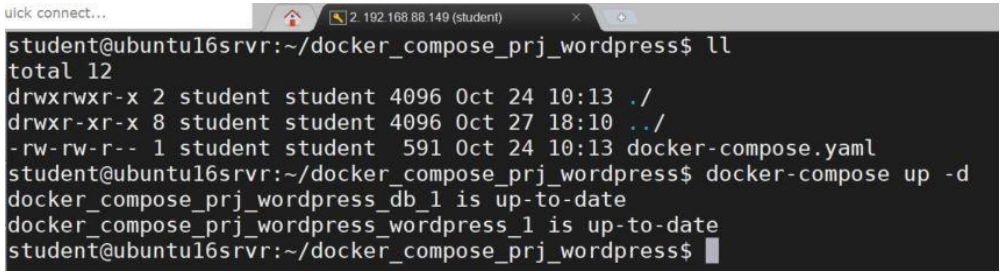
Build the project

In order to buil the project, run following command from your project directory.

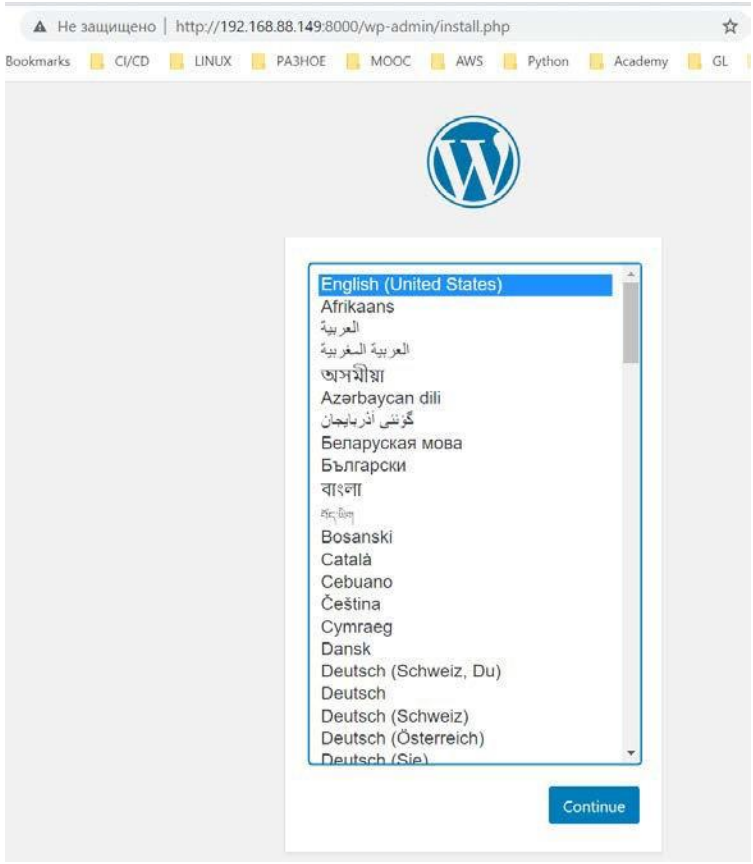
docker-compose *up* -d

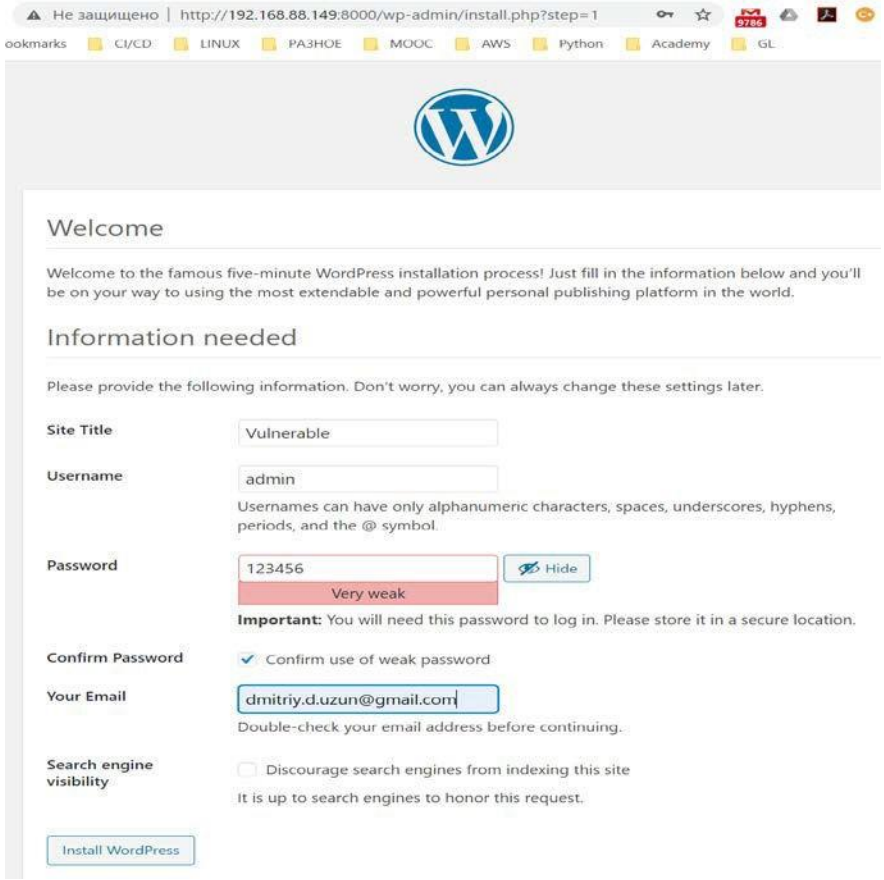
This runs **docker-compose** *up* in detached mode, pulls the needed Docker images, and starts the **wordpress** and **database** containers, as shown in the example below.

```
$ docker-compose up -d
Creating network "my_wordpress_default" with the default driver
Pulling db (mysql:5.7)...
5.7: Pulling from library/mysql
efd26ecc9548: Pull complete
a3ed95caeb02: Pull complete
...
Digest: sha256:34a0aca88e85f2efa5edff1cea77cf5d3147ad93545dbec99cfe705b03c520de
Status: Downloaded newer image for mysql:5.7
Pulling wordpress (wordpress:latest)...
latest: Pulling from library/wordpress
efd26ecc9548: Already exists
a3ed95caeb02: Pull complete
589a9d9a7c64: Pull complete
...
Digest: sha256:ed28506ae44d5def89075fd5c01456610cd6c64006addfe5210b8c675881aff6
Status: Downloaded newer image for wordpress:latest
Creating my_wordpress_db_1
Creating my_wordpress_wordpress_1
```

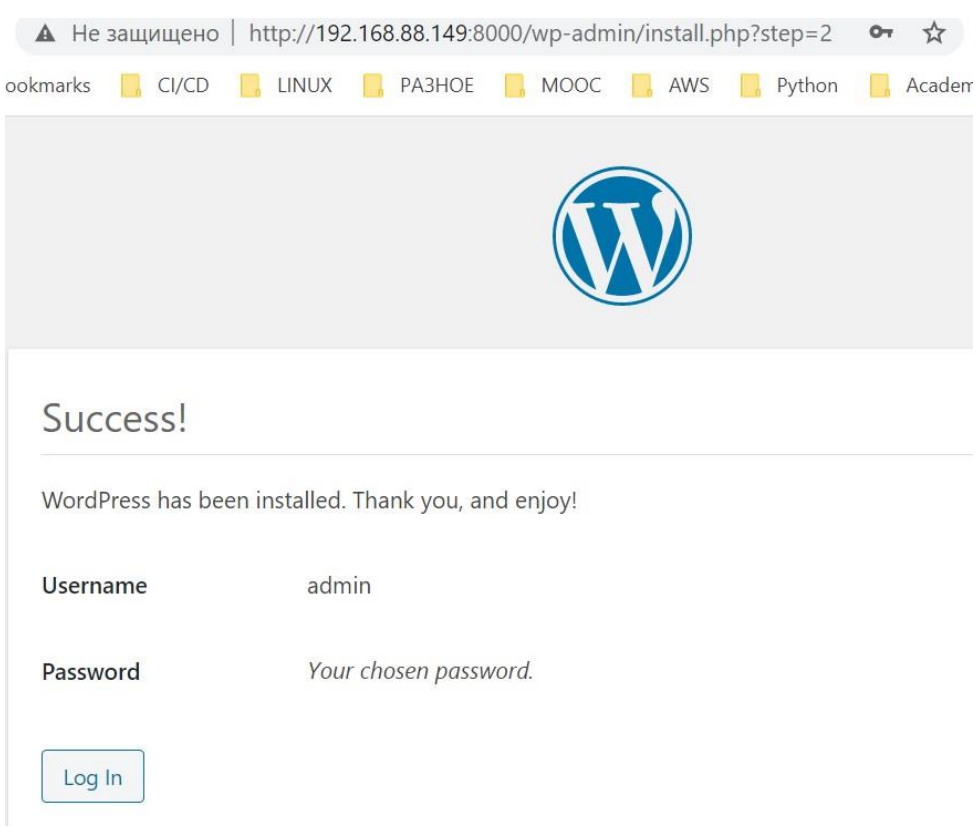


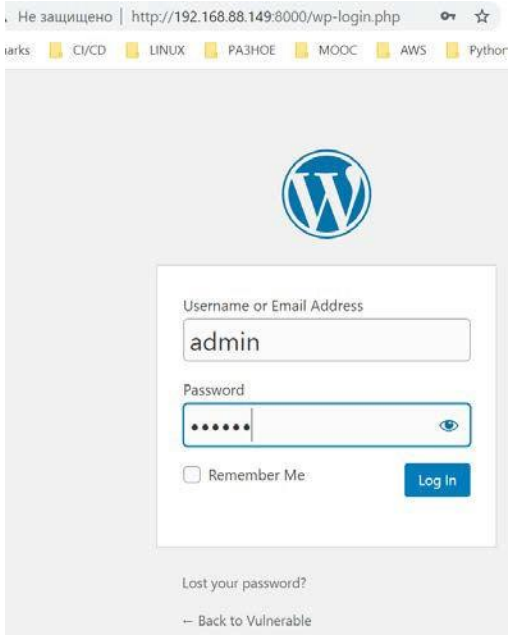
Docker Compose. Wordpress realization



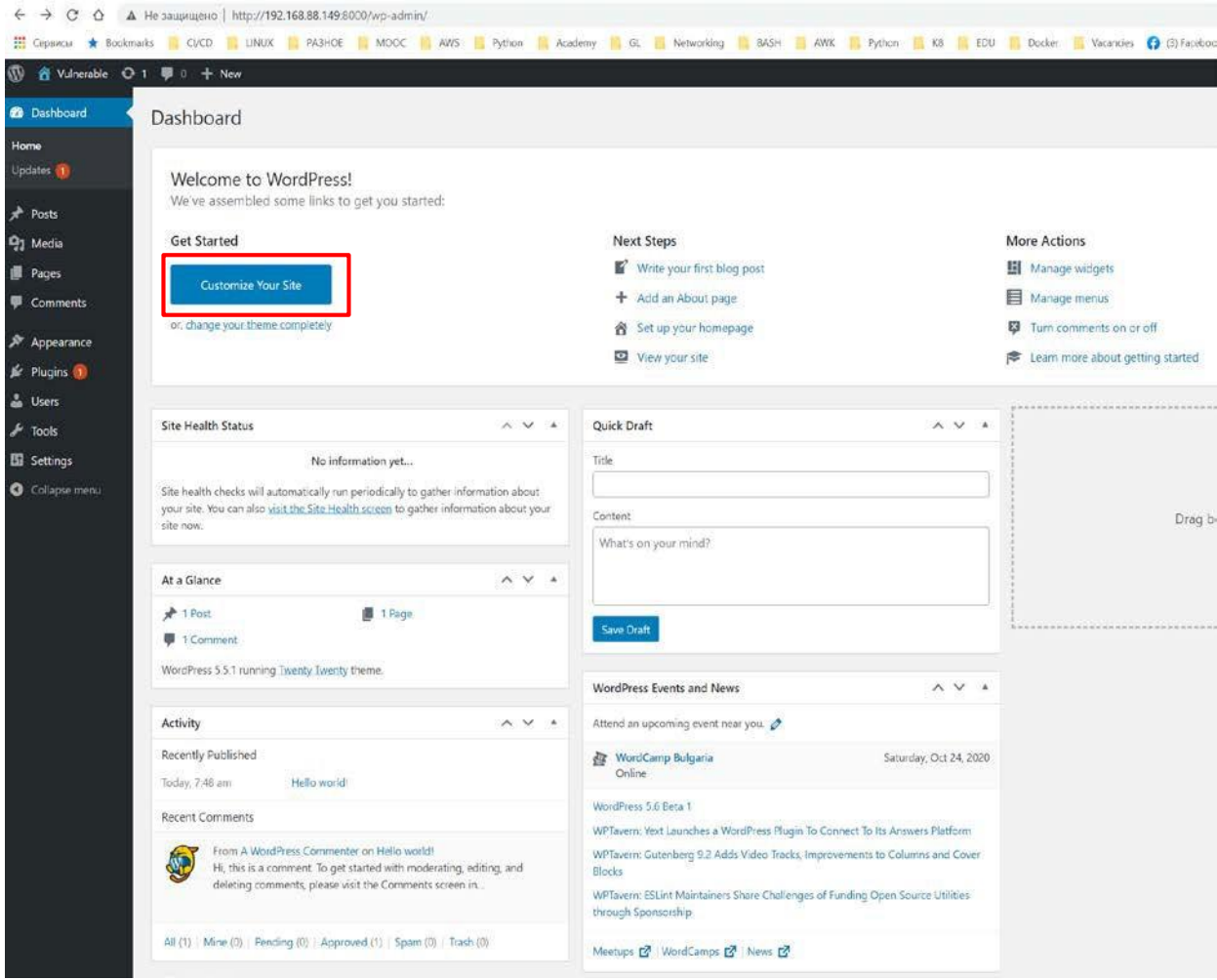


Docker Compose. Wordpress realization

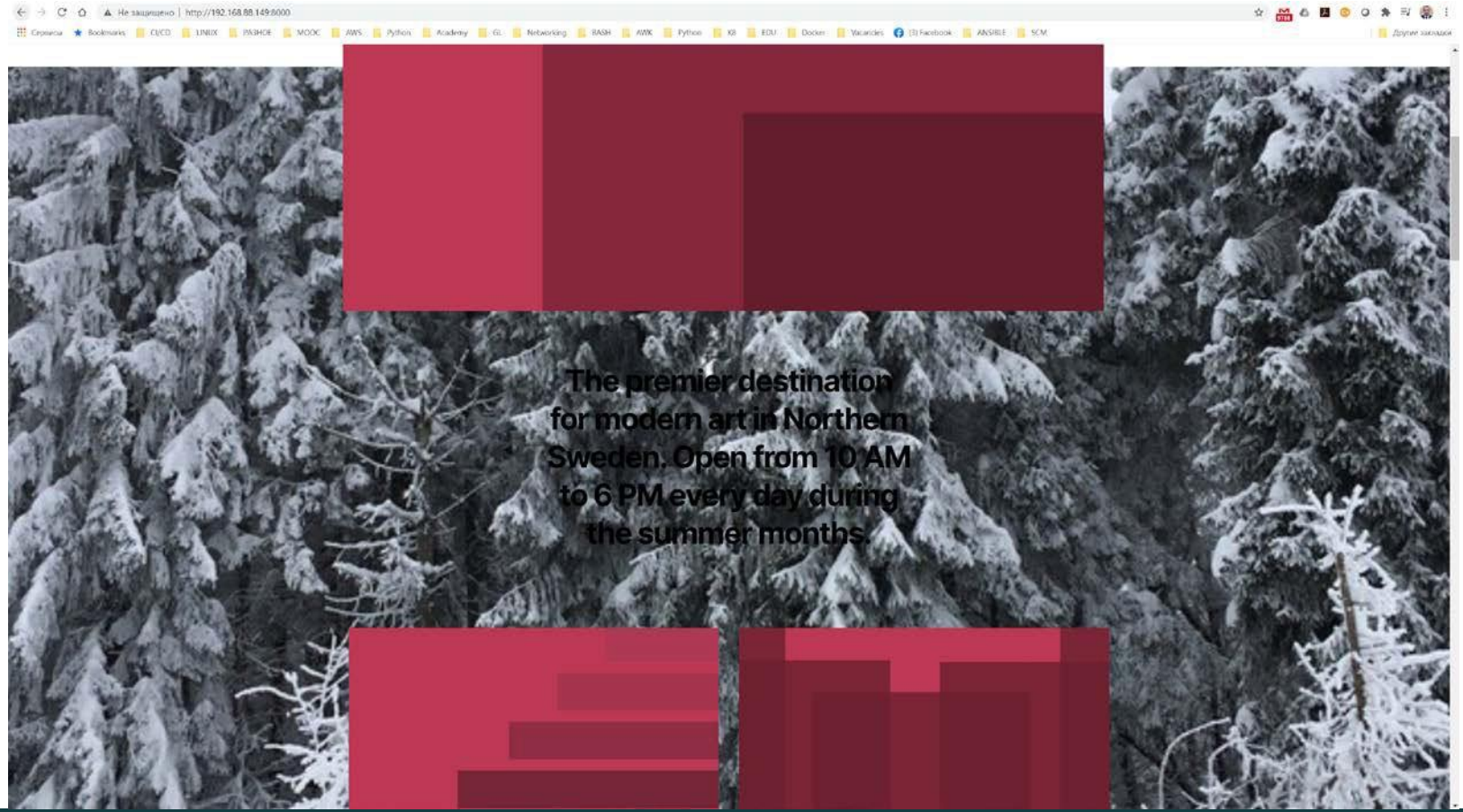




Docker Compose. Wordpress realization



Docker Compose. Wordpress realization



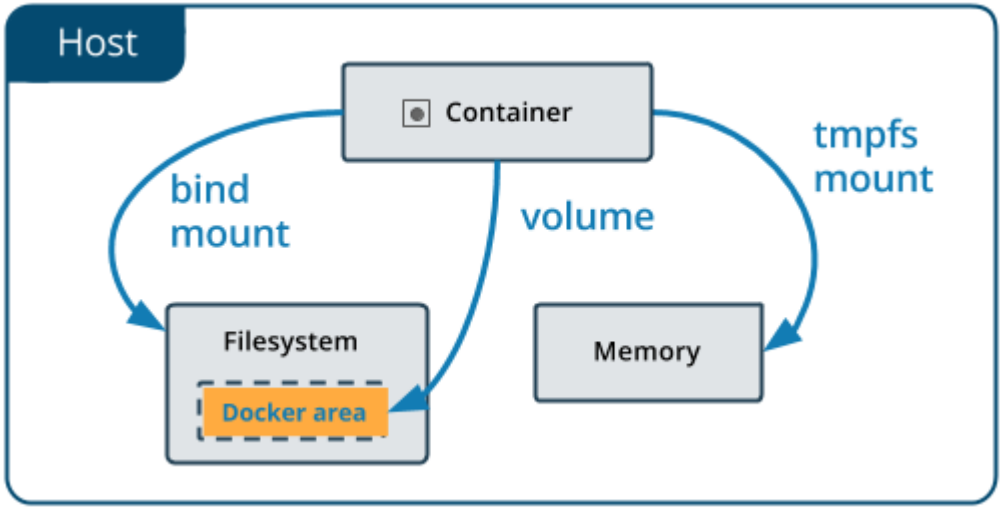
Docker Compose. Use volumes

Volumes are the preferred mechanism for persisting data generated by and used by Docker containers. While bind mounts are dependent on the directory structure and OS of the host machine, volumes are completely managed by Docker. Volumes have several advantages over bind mounts:

-Volumes are easier to back up or migrate than bind mounts.
- You can manage volumes using Docker CLI commands or the Docker API.
- Volumes work on both Linux and Windows containers.
- Volumes can be more safely shared among multiple containers.
-Volume drivers let you store volumes on remote hosts or cloud providers, to encrypt the contents of volumes, or to add other functionality.
- New volumes can have their content pre-populated by a container.
- Volumes on Docker Desktop have much higher performance than bind mounts from Mac and Windows hosts.
-Volumes are often a better choice than persisting data in a container's writable layer, because a volume does not increase the size of the containers using it, and the volume's contents exist outside the lifecycle of a given container.

Docker Compose. Use volumes

If your container generates non-persistent state data, consider using a **tmpfs mount** to avoid storing the data anywhere permanently, and to increase the container's performance by avoiding writing into the container's writable layer.



Docker Compose. Use volumes

Create and manage volumes

You can create and manage volumes outside the scope of any container.

Create a volume:

$ docker volume create my-vol

List volumes:

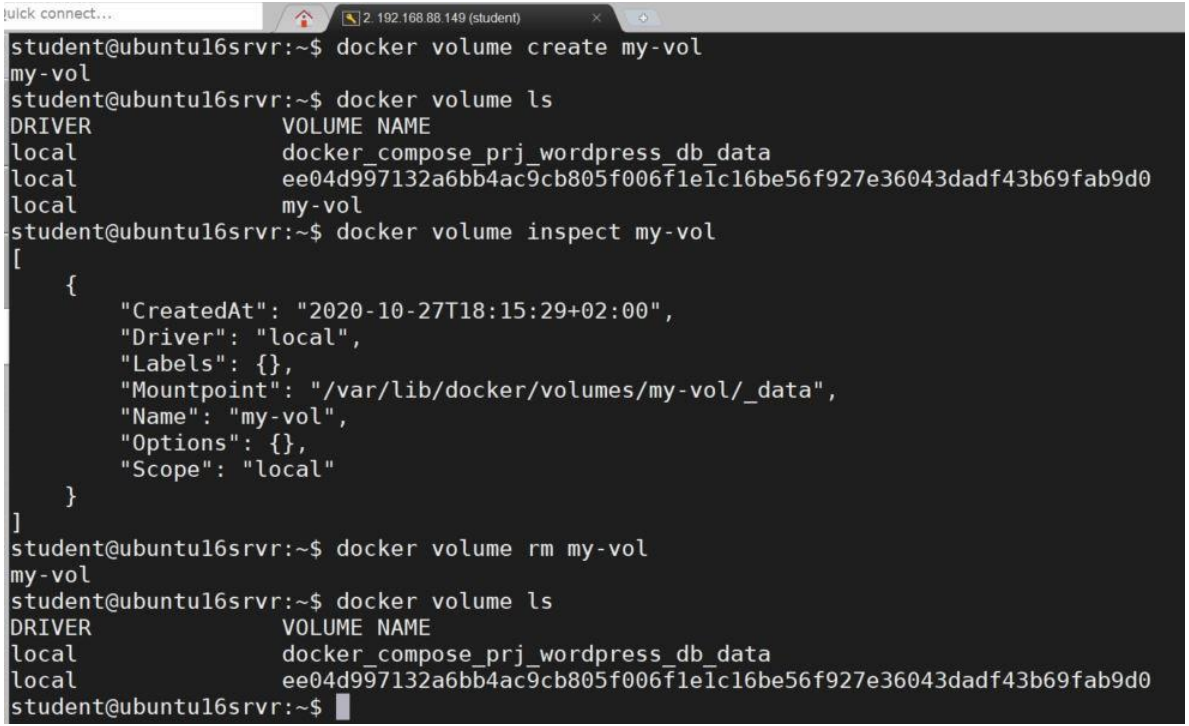
$ docker volume ls

Inspect a volume:

$ docker volume inspect my-vol

Remove a volume:

$ docker volume rm my-vol



Q&A


Thank you!