Introduction to Docker

Lampert, Marcus
lampert@bbaw.de
Berlin-Brandenburgische Akademie der Wissenschaften, Germany

Short Description

Overview

First introduced in 2013, Docker is one of those tools that, not unlike the version control software Git, has become nearly ubiquitous in software development. As a lightweight virtualization software, it elegantly solvescommon problems in software engineering related to developing and deploying applications. Dockercreates independent operating environments, so-called ‘containers’, with which one can deploy applications in a flexible and reproducible manner. A singleDocker command on the command line can launch an entire application, either locally or on a server, with any number and kind of desired frontends, backends, services, databases, etc. This containerized approach to application development has many advantages for software engineering in general, and for the the Digital Humanities specifically, including:

- Ease of setting up an existing project on a new server or personal computer. This is why, for example, open source projects often offer a Docker image, or describe how to set up their application using a standard Dockerfile.
- Continuity between local development and production. Building a project locally on a personal computer with Docker, versus building for production with Docker are often similar, if not identical processes.
- Broad language and version support. A server that runs Docker only needs to have Docker installed in order to support basically every major language and their various versions. If a Docker image exists for a given language and version, then that application can be run. Specifically for the Digital Humanities, where projects have a long duration, this feature could potentially prolong the time for which an applicationcan be supported.

In this workshop, we offer participants a practical introduction to Docker and show by example how it can be integrated into existing and new Digital Humanities projects. The two examples we use are Digital Humanities projects developed at the Berlin-Brandenburg Academy of Sciences and Humanities and are exemplars of two standard application types used in the Digital Humanities: a web application based on an SQL database, and a web application based on an XML NoSQL database. Participants will follow along with the presenter, deploying these applications on their own laptops using Docker. We will also look together at a live Docker application running on a server at the Berlin-Brandenburg Academy of Sciences and Humanities. By the end of the workshop, participants will have a general idea of what Docker is and how to use it, and can begin thinking about how they can use Docker to facilitate their Digital Humanities projects.

Background

The workshop is based on a workshop developed internally at the Berlin-Brandenburg Academy of Sciences and Humanities. At the Academy, we have recognized the advantages that Docker would bring for a number of our projects. This workshop is designed to be practical and accessible to a broad audience. It covers basic concepts and commands that arise in one’s regular work with Docker.

Workshop Structure and Content

The workshop takes four hours, including breaks. It consists of four modules, each module lasting about 45 minutes followed by a 15 minute break:

- Introduction
- Docker
- Docker Compose
- Deployment

In each of these modules, the presenter will explain and demonstrate core concepts on his computer, and then participants will be asked to execute a number of Docker commands on their computers. At least one assistant will assist the presenter in going around the room helping people.

Module 1: Introduction

Topics covered:

- Installing Docker and Docker Compose
- Description of Docker: a software platform for enclosing applications and application components in containers that are encapsulated but share an operating system, i.e. lightweight virtualization
- Advantages of Docker:
  - Reproducible application builds in a reproducible environment.
  - Can run multiple applications on a host, local or a server, without colliding dependencies.
  - Easy to deploy, locally or on a server, an application with multiple parts (frontend, backend, database, etc).
- Disadvantages of Docker:
  - Learning curve
  - Additional work of dockerizing a project
  - Managing Docker on a server, potential security problems.
- The building blocks of Docker: images, containers, volumes, networks, etc.

Module 2: Docker

Objective: learn basic Docker commands.
Topics Covered:

- Anatomy of a docker project
  - ‘Dockerfile’
  - ‘.dockerignore’ (optional)
- Starting the example application without docker
  - ‘cp .env.example .env’
  - ‘php composer install or compose install’
  - ‘php artisan serve’

- Starting applications with docker
  - ‘docker build .’
  - ‘docker run [IMAGE ID]’
  - From a new terminal window: ‘docker ps’
  - ‘docker exec -ti [CONTAINER ID/NAME] bash’
- A few problems:
  - Cannot access the port (see ‘docker network ls’)
  - Image ID is not very descriptive
- Stopping the container:
  - ‘docker ps’
  - ‘docker stop [CONTAINER ID/NAME]’
- Starting the applications with docker, improved
  - ‘docker build -t [NAME] .’
  - ‘docker run -p 8000:8181 [NAME]’
- Starting applications in the background
  - ‘docker run -d -p 8000:8181 [NAME]’
  - ‘docker logs -f [CONTAINER ID/NAME]’
- Starting verses running a container
  - ‘docker container ls’
  - ‘docker start [CONTAINER ID/NAME]’

Module 3: Docker Compose

Objective: learn how a docker-compose.yml file is composed and learn basic Docker Compose commands.
Topics Covered

- Problems with just using Docker
  - Stopping containers is annoying
  - Important things like ports, tags, network, volumes, etc. are only giving at runtime
  - How would you handle a project with multiple containers?
- Docker commands one might often use
  - ‘docker exec -ti [CONTAINER ID/NAME] bash’ - If you have a database in a docker container, for example
  - ‘docker ps’, ‘docker image ls’, ‘docker volume ls’
- Basic Docker Compose commands
  - ‘docker-compose build [services]’
  - ‘docker-compose up [services]’
  - ‘docker-compose up --build [services]’
  - ‘docker-compose down’, ‘docker-compose down -v’
- Common things we do in a docker-compose.yml file
  - expose ports
  - talk with other containers in the same network
  - set environment variables
  - define volumes
- Use cases of Docker Compose
  - Production: deploy with one command
  - Development: start those services you aren't actively working on.

Module 4: Deployment

Objective: Gain a general idea of how to use Docker to deploy an application on a server
Topics Covered:

- One docker-compose.yml for each environment.
  - local.docker-compose.yml
  - production.docker-compose.yml
- Merging Multiple docker-compose.yml files: https://docs.docker.com/compose/extends/
- Vendor docker-compose.yml with a shell script.
  - Define a template.yml with variables:
    environment:
    - BACKEND_URL: https://${BACKEND_URL}:${BACKEND_PORT}
  - For each env define values for the variables. So, env.local looks like:
    BACKEND_URL=localhost
    BACKEND_PORT=1234
  - Have shellscript execute ‘envsubst’ to generate the docker-compose.yml:
    Input: ‘devops/vendor.sh local’
    Output: ‘docker-compose.yml’

Presenter

Marcus Lampert
Wissenschaftlicher Mitarbeiter an der Berlin-Brandenburgischen Akademie der Wissenschaften
lampert@bbaw.de

Marcus works as a software engineer on various Digital Humanities projects at the Berlin-Brandenburg Academy of Sciences and Humanities. He received his PhD 2015 in German Literature from the University of Chicago writing on the content and influence of Johann Gottlieb Fichte’s 1794 *Science of Knowledge*. Prior to joining the Academy, he worked for three years as a software engineer at PricewaterhouseCoopers Deutschland. He enjoys working as a fullstack developer, fiddling on both the frontend and backend, and solving devops challenges. Marcus is driven by the quest to use modern technology to facilitate the sharing of knowledge.

Format of the Workshop and Target Audience

- Language: German, or English depending on the needs of the participants.
- Up to 25 participants. No prior knowledge of Docker required.
- Participants should have an interest in writing and/or deploying websites or other related software applications.
- Basic use of the command line is helpful but not required.
- Experience with Linux systems is helpful but not required.

Participants should have a personal laptop with the following specs:

- admin rights so that they can install software such as Docker and Docker Compose.
- Preferably Mac or Linux as Operating System.
- Ability to connect to the internet.
- Some remaining storage space (50 GB) for using Docker.
- Very old machines might have performance or compatibility issues with Docker.

Technical Equipment

- Large display monitor/screen to connect to presenter’s laptop.

- Power outlets for the participants’ computers.
- Wireless internet connection for all the participants.

Bibliography

Docker Documentation. https://docs.docker.com/ (accessed 1 December 2021).

En.wikipedia.org. Docker (software) - Wikipedia. https://en.wikipedia.org/wiki/Docker_(software) (accessed 1 December 2021).