Analysis of Adopting DevOps Tools for a Homogeneous Production and Development Environment

Henrik Gerdes

hegerdes@uos.de

October 27, 2021

UNIVERSITÄT OSNABRÜCK

Symbic

Table of Contents I

Outline

Goal of the Thesis

- Possibilities to increase efficiency in programming
  - Investigate possible points of improvement
  - Develop an abstract solution concept
  - Apply the concept to a prototype
  - Evaluate and compare the concept to alternative solutions

Outline

Potential Bottlenecks in the development workflow

- Initial Setup
- Testing
- Dependencies
- Configuration

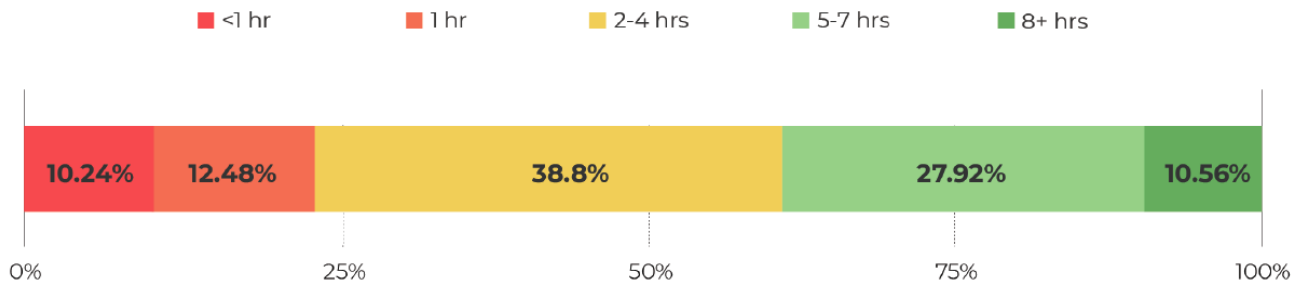


Figure: Hours spent programming
Source: [Inc19]

Initial Setup

- Installing development tools, language frameworks, IDEs, E-Mail, VPN and team-chats
- 69% of developers renew their setup between 1-4 times a year mostly taking about 2-4 hours
- 18% even 5-8 times a year taking up to 18 hours

```
./bootstrap: Bootstrapping from checked-out gnutls sources...
./bootstrap: consider installing git-merge-changelog from gnulib
./bootstrap: getting gnulib files...
./bootstrap: getting translations into po/.reference for gnutls...
./bootstrap: line 702: rsync: command not found
bootstrap failed. Check D:/msys64/media-autobuild_suite-master/build/
ab-suite.bootstrap.log
This is required for other packages, so this script will exit.
  Creating diagnostics file...
```

Figure: Failed bootstrap script

Configuration & Dependencies

- Project A requires NodeJS V12 ↯ Project B requires NodeJS V14
- Development uses Python 3.9 ↯ Production uses Python 3.7
- Development uses PHP 7.3 ↯ Production uses legacy PHP $\leq$ 7.0

- APP A needs a DB connection string with: host, user, password
- APP B listens on port 8080 and requests APP A on port 3000

Configuration & Dependencies

Figure: Netflix's Microservice Death-Star
Source: [DZo19]

Outline

Apply solutions from the server world

- Abstract the underlying hardware and system
- Define and standardize the working environments
- Reduce manual effort and automate processes
- Allow app/test integration on the developer system

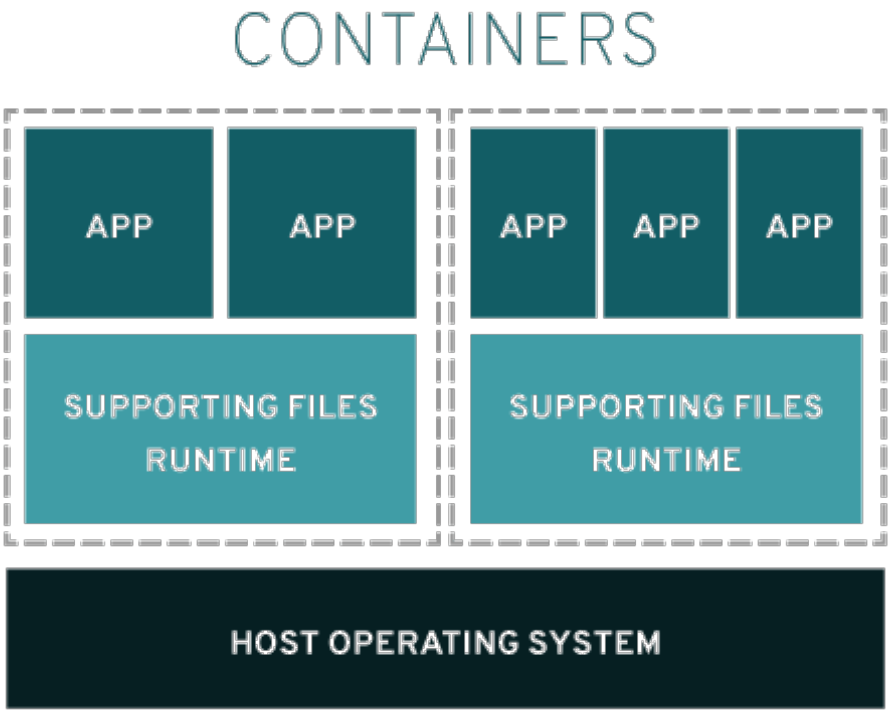


Figure: Container-based virtualization
- Source: [Red21]

- Container-based virtualization
- Less resource needs than a VM; Only the process is isolated
- The app runtime is in a self-contained bundle
- Implemented with Docker

- Docker provides a unified, independent platform
- Closer to the production environment

- Different projects are isolated from each other
- Wider choice of available software

- Orchestrate and test multiple services
- Applying the IaC principle

Architectural concept

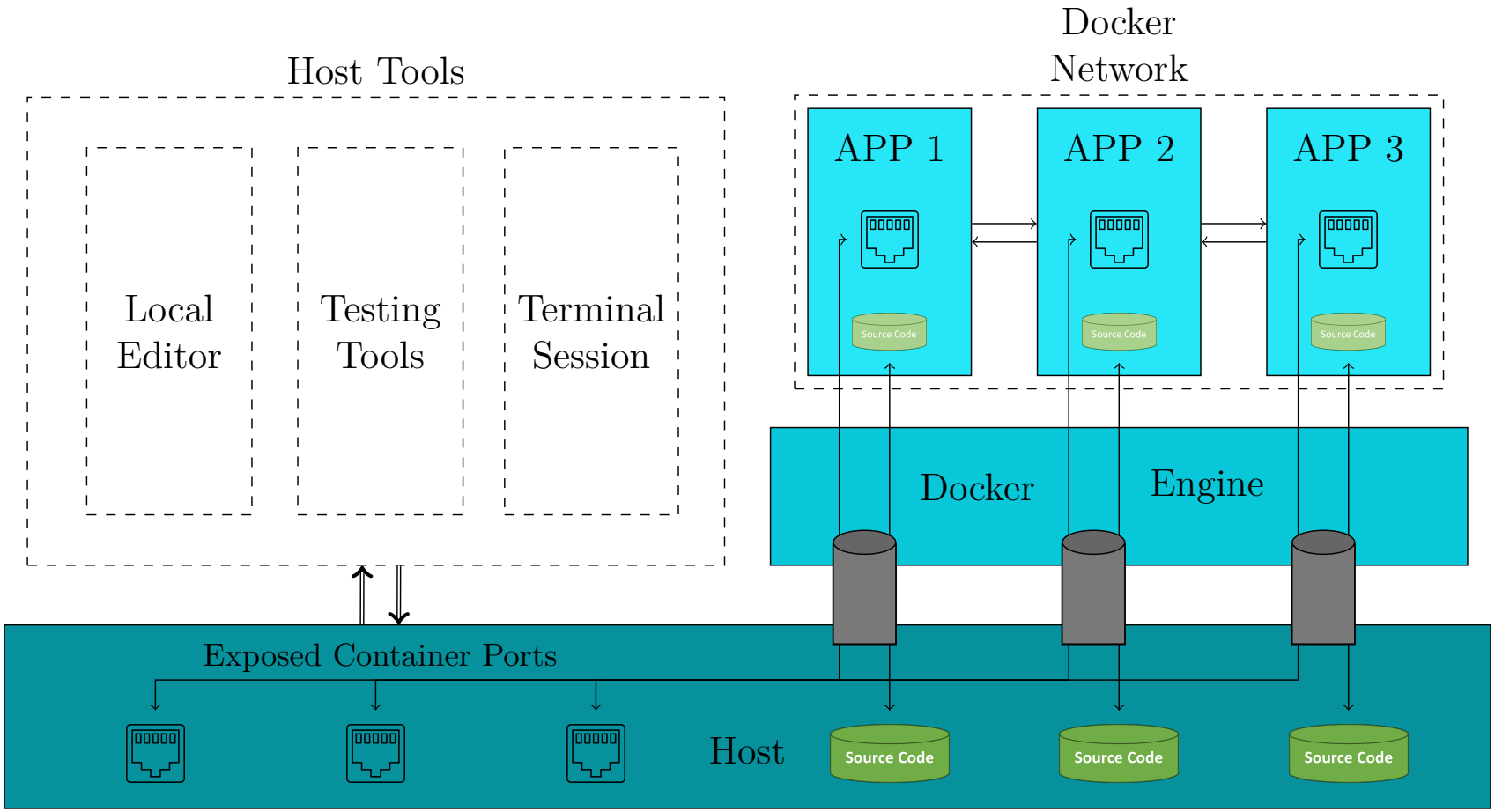
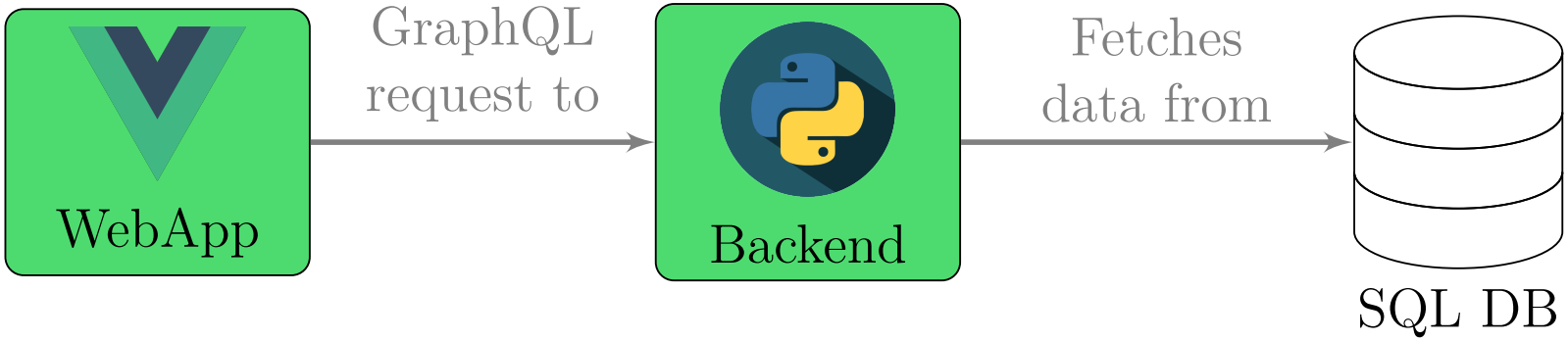
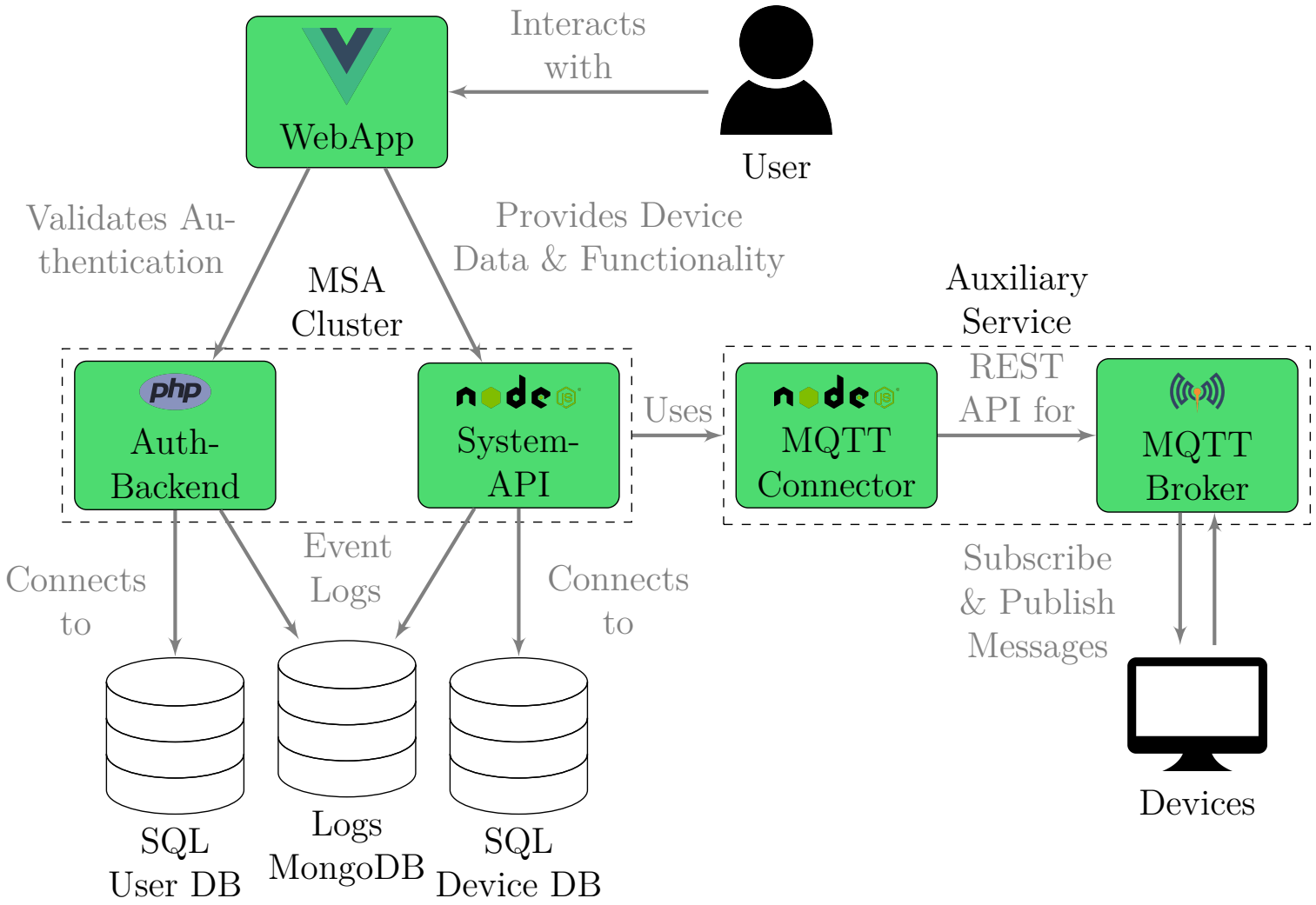


Figure: Development Architecture with DevContainers



DEMO

IoT-Management System by Symbic



Docker-Compose

```yaml
services:
  app:
    image: my-backend-app
    environment:
      - DB_HOST=db
      - DB_PW=yes
      - DB_USER=root
    ports:
      - 8080:8080
    volumes:
      ./app-src:/workspace
  db:
    image: mysql
    environment:
      - MYSQL_ROOT_PASSWORD=yes
    ports:
      - 3306:3306
```

Listing: Exemplary docker-compose.yml file

Outline

Found properties

- Faster initial setup
- Homogeneous environments across all developers
- Larger selection of software
- Greater similarity to the production environment
- Isolated environment to other programs
- More error resistant and faster error recovery

- Container knowledge and infrastructure maintenance required
- Noticeable I/O performance overhead on Windows

Comparison to Alternatives

Codespaces & Gitpod

- Development in a Browser or VSCode
- Depends on external servers and providers
- Computing capacity on demand
- Focuses on one Container

DevContainer

- Development in any editor
- Full control over the code and hosting
- Can be adopted step-by-step
- Hybrid approach allows usage of local tools

Future Potential & Outlook

- Particularly easy to get started with - Suitable for teaching purposes and open source projects
- Allows programming independently of the used device
- Computing capacity can be adjusted flexible and on demand
- Shared applications can be accessed across the team

Questions?

Thank you for your attention!
Any questions?

References I

- DZone. (2019) Microservice Deathstar Created when Configuration Management Shifts to Runtime. Visited on: 2021-10-11. [Online]. Available: https://dzone.com/articles/navigating-the-microservice-deathstar-with-deployh
- GitHub Inc. (2021) GitHub's Engineering Team has moved to Codespaces. Visited on: 2021-08-11. [Online]. Available: https://github.blog/2021-08-11-githubs-engineering-team-moved-codespaces
- A. S. Inc., "Developer Survey - Open Source Runtime Pains 2019," Open Source Languages Company, Tech. Rep., 04 2019, visited on: 2021-08-27. [Online]. Available: https://www.activestate.com/resources/white-papers/developer-survey-2019-open-source-runtime-pains

References II

- K. Indrasiri and P. Siriwardena, *Microservices for the enterprise*. Springer, 2018, ISBN: 978-1-4842-3858-5.
- RedHat Inc. (2021) Containers vs VMs. Visited on: 2021-06-08. [Online]. Available: https://www.redhat.com/en/topics/containers/containers-vs-vms

	Orchestration	Languages	Server/ Self Hosted	Network Support	Package Support	Pricing
Codesandbox.io	×	NodeJS	✓/-	HTTP - accessible from the web	restricted for native packages	Free, 30$, 56$
Stackblitz	×	NodeJS	-/✓	Only WebSockets	no support for native packages	Free, 9$, 39$
Gitpod	within containers	any	✓/✓	Full TCP/UDP	any	Free, 9$, 25$, 39$
CodeSpaces	within containers	any	✓/✓	Full TCP/UDP	any	Free, 4$, 21$
DevContainer	within containers*	any	✓/✓	Full TCP/UDP	any	Free, 5*$, 7*$, 21*$

Table: Comparison of Different Development Solutions -

Dockerfile

```
# Node.js version: 16-bullseye, 14-bullseye, 12-bullseye
ARG VARIANT=16-bullseye
FROM node:${VARIANT}

# Install needed packages, yarn, nvm and other tools
COPY install-scripts/*.sh /tmp/install-scripts/
RUN apt update && bash /tmp/install-scripts/install.sh \
  && apt-get -y install python3 make vim emacs git ssh \
  && npm install -g eslint
```

Listing: NodeJS DevContainer Dockerfile

	Win_RAM	Win_CPU	Win_Docker_RAM	Win_Docker_CPU
nodeJS	108.5 MB	16%	92.2 MB	11%
MariaDB	29.8 MB	<1%	111.9 MB	<1%
Apache	19.9 MB	8%	27.2 MB	<1%
Docker	4078 MB	15%	-	-

Table: CPU and RAM usage of Processes on Windows with and without using Docker

	Linux_RAM	Linux_CPU	Linux_Docker_RAM	Linux_Docker_CPU
nodeJS	65.2 MB	1%	65.0 MB	1%
MariaDB	100.2 MB	<1%	106.0 MB	<1%
Apache	15.7 MB	<1%	10.6 MB	<1%
Docker	185.6 MB	1%	-	-

Table: CPU and RAM usage of Processes on Linux with and without using Docker