

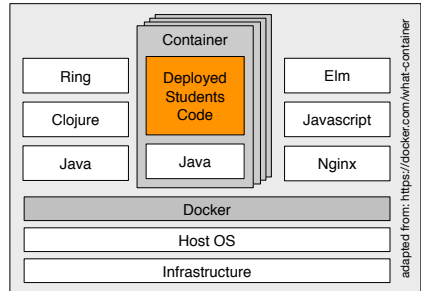
VIRTUALISIERTE ARBEITSUMGEBUNG FÜR DEN TEST VERTEILTER SYSTEME

Microservice-Architektur mit Docker, Clojure & Elm

13. Dezember 2017

Jan-Philipp Willem

Fakultät für Informatik
Hochschule Mannheim



GLIEDERUNG

1. Anforderungen

2. Technologien

3. Inspiration

4. „VAR-Tool“

5. Ausblick

→ <https://github.com/jwillem/var-tool>

ANFORDERUNGEN

ANFORDERUNGEN (1/2)

- Dozent beschreibt ein Experiment im Kontext von verteilten Systemen
- Feste Anzahl an Instanzen, auf denen Lösungen von Programmieraufgaben ausgeführt werden sollen
- Benötigte Technologien können dabei auch definiert werden
- Pro Instanz: Angabe von Name, Ports, standard Command
- Getrennte Netzwerke zwischen Experimenten

ANFORDERUNGEN (2/2)

- Studenten sollen mithilfe der Experimente gewisse Aufgaben lösen
- Upload von Programmpaketen im Frontend
- Angabe von Main-Class und Argumentenliste
- Einzelnes Starten der Instanzen
- Eingabe im Frontend führt zu Ereignis bei stdin von Instanz
- Geschehene Logs (stdout, stderr) in Instanz führen zu Ereignis im Frontend

TECHNOLOGIEN

DOCKER

- Software zum Deployment von Applikationen innerhalb von Containern
- Ähnelt Vorgehensweise von Virtuellen Maschinen
- Im Gegensatz: Mithilfe von Docker-Daemon laufen Prozesse direkt auf Host-OS
- Keine spezifische Hardware-Infrastruktur vorgeschrieben
- Lauffähig auf Linux, Mac-OS, Windows



CLOJURE

- Moderner Lisp-Dialekt auf JVM
- Fokus liegt auf funktionalem Paradigma
- dynamisch typisiert
- Datenstrukturen sind Immutable
- Interessantes Konzept der Nebenläufigkeit: Communicating Sequential Processes (CSP)
- Interoperabilität zu JAVA



ELM

- Rein funktionale Sprache
- ML-artig, ähnlich zu bspw. Haskell
- Kompiliert zu Java-Script
- DSL für HTML, SVG
- Sinnvolle Abstraktionen wie bspw. von Websockets
- Virtual DOM, ähnlich zu React
- Statisch typisiert, dabei sehr hilfreicher Compiler +
- Kein NULL oder undefined
- Core-API nutzt bei unsicheren Werten die Monaden Maybe oder Result +
- The-Elm-Architecture (TEA), Ursprung von Redux in JS +



INSPIRATION

TMUX PANES

```

4 19m
jen-mbp21docker-var jan$ ls -ls
total 64
drwxr-xr-x 13 jan staff 442 May 29 12:27 .
drwxr-xr-x 20 jan staff 608 Jun 5 21:53 ..
-rw-r--r-- 1 jan staff 8196 Jun 19 20:46 .DS_Store
-rw-r--r-- 1 jan staff 678 Jun 19 21:27 .git
-rw-r--r-- 1 jan staff 2076 Apr 10 16:47 .gitignore
-rw-r--r-- 1 jan staff 104 May 26 19:52 .gitmodules
-rw-r--r-- 1 jan staff 1075 Apr 10 16:46 LICENSE
-rw-r--r-- 1 jan staff 25 May 8 22:11 README.md
drwxr-xr-x 12 jan staff 408 Jun 19 21:27 client
-rw-r--r-- 1 jan staff 1271 May 26 19:53 docker-compose.yml
drwxr-xr-x 15 jan staff 518 May 26 19:53 docker-master
drwxr-xr-x 8 jan staff 272 Jun 19 10:41 docs
drwxr-xr-x 10 jan staff 340 May 26 19:33 kafka-websocket
jen-mbp21docker-var jan$
|
| | tokenize-arg-string.js
| | |-- node_modules
| | | |-- camelcase
| | | |-- index.js
| | | |-- license
| | | |-- package.json
| | | |-- readme.md
| | | |-- package.json
| |-- package.json
| |-- public
| |-- css
| | |-- app.css
| | |-- app.css.map
| |-- index.html
| |-- js
| | |-- app.js
| | |-- app.js.map
|
| 1268 directories, 6762 files
jen-mbp21docker-var jan$
|
| ./bootstrap.sh && \
| make install && \
| cd .. && rm -rf kafkacat
| # && AUTO_ADDED_PACKAGES="apt-mark showauto" && \
| # apt-get remove --purge -y $BUILD_PACKAGES $AUTO_ADDED_PACKAGES && \
| # rm -rf /var/lib/apt/lists/* /tmp/* /var/tmp/*
|
| RUN mkdir -p /usr/src/app
| WORKDIR /usr/src/app
|
| # closure app
| # COPY project.clj /usr/src/app/
| # run lein deps
| # copy . /usr/src/app
| # RUN mv "lib/uberjar" sed -n 's/^Created (.+standalone.jar)/lib/' app-standalone.jar
|
| # ENTRYPOINT ["kafkacat"]
| # CMD ["java", "-jar", "app-standalone.jar"]
| CMD ls -ls | kafkacat -b kafka -t logSTOKEN
jen-mbp21docker-var jan$

```

The screenshot shows a web browser window with the URL `localhost:7681`. The browser tabs include "Typing speed test, advance...", "VAR-docker.md", "ts/0922/ttyd: Share your...", "docker run -it --rm ubuntu", and "docker run -it --rm ubuntu".

The main content area displays system statistics:

```

1 [          0.0%] Tasks: 2, 0 thr; 1 running
2 [          0.0%] Load average: 0.00 0.03 0.00
3 [          0.0%] Uptime: 1 day, 02:23:47
4 [          0.7%]
Mem[|||||550M/1.95G]
Swp[          3.37M/3.91G]

```

Below the statistics is a table of running processes:

PID	USER	PRI	NI	VIRT	RES	SHR	S	CPU%	MEM%	TIME+	Command
1	root	20	0	18240	3320	2836	S	0.0	0.2	0:00.05	/bin/bash
420	root	20	0	22992	3364	2792	R	0.0	0.2	0:00.00	htop

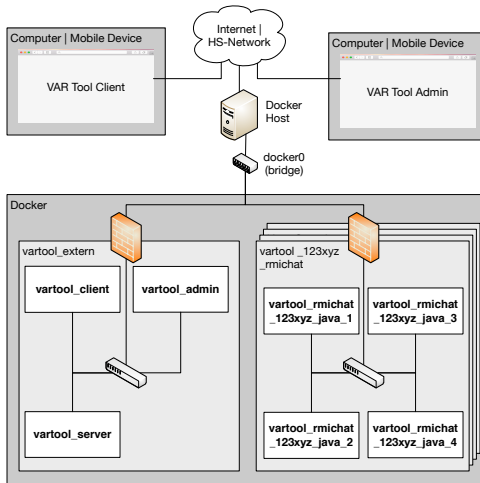
At the bottom of the interface, a command prompt shows the command:

```
$ ttyd docker run -it --rm ubuntu htop
```

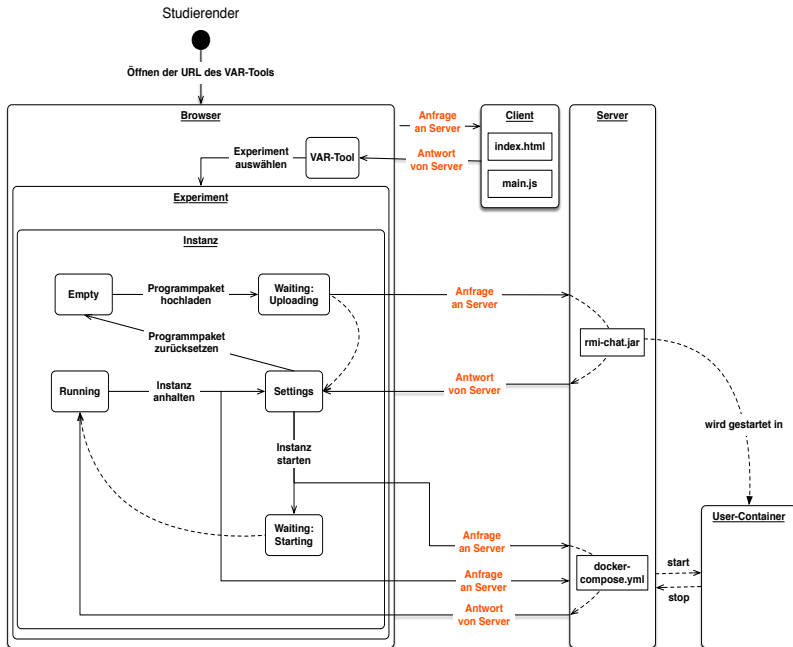
The footer of the interface contains a navigation bar with the following items: F1 Help, F2 Setup, F3 Search, F4 Filter, F5 Tree, F6 SortBy, F7 Nice, F8 Nice, F9 Kill, F10 Quit.

„VAR-TOOL“

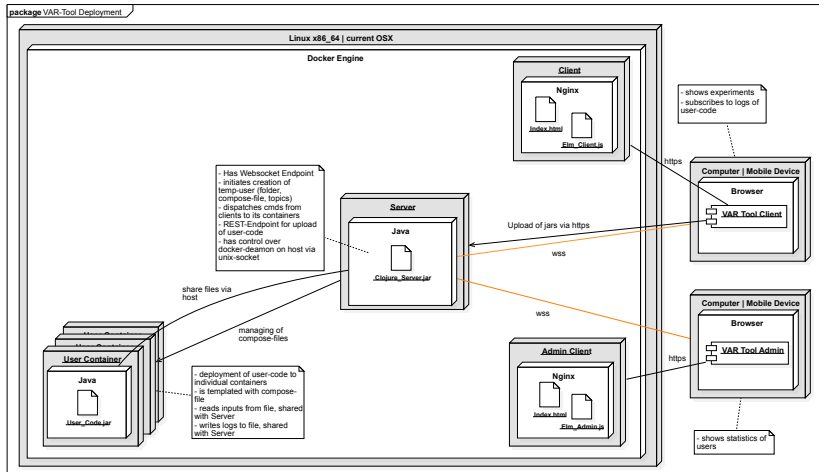
NETWORK



STATE



DEPLOYMENT



AUSBLICK

- Docker in Docker // Binding von Docker-Socket¹
- Sehr spannendes Thema
- Viel Raum für weitere (studentische) Arbeiten
- Erstrebenswerte Aufgabe der Fakultät

¹<https://jpetazzo.github.io/2015/09/03/do-not-use-docker-in-docker-for-ci/>