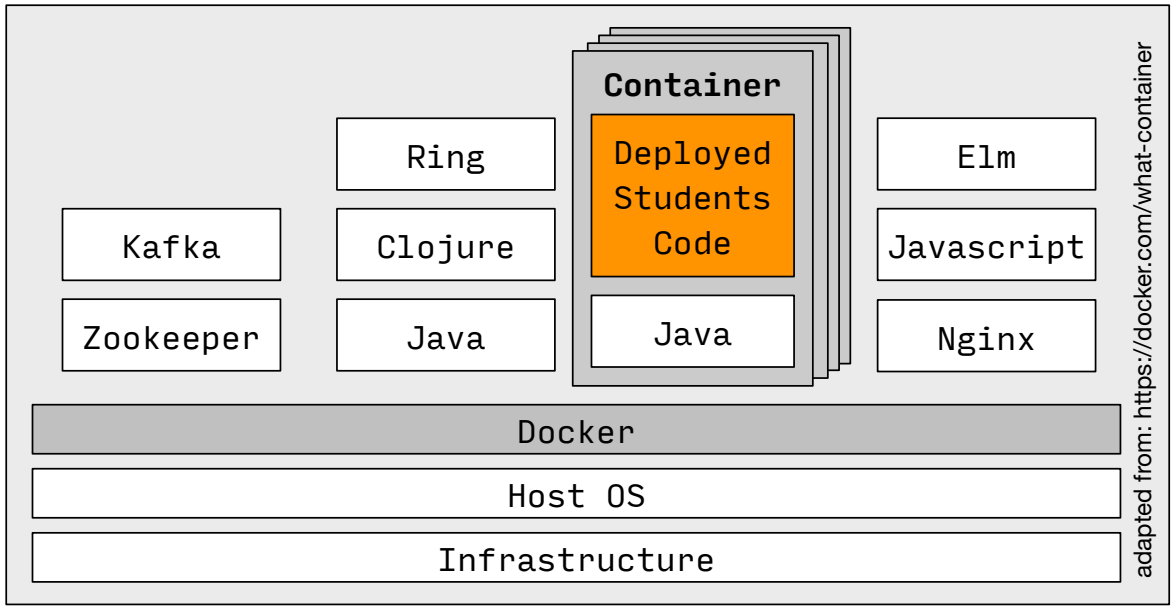
VIRTUALISIERTE ARBEITSUMGEBUNG FÜR DEN TEST VERTEILTER SYSTEME

Microservice-Architektur mit Docker, Kafka, Clojure & Elm

21. Juni 2017

Jan-Philipp Willem

Fakultät für Informatik
Hochschule Mannheim



GLIEDERUNG

1. Kontext
2. Docker
3. Inspiration
4. „VAR-Tool“
5. Ausblick

KONTEXT

BISHERIGER ABLAUF

→ Student soll (verteilte) Aufgaben programmieren
→ Es wird die Benutzung von schwergewichtigen IDEs vorgeschlagen
→ Eclipse, Netbeans bieten Integrationen zu diversen Servern/Technologien
→ Funktionsweise kann in gegebener (Dev-)Umgebung getestet werden

→ Gefahr: keine echte Verteilung
→ Umgebung verschieden zu Production
→ Ergebnis kann sich erheblich unterscheiden (Seiteneffekte,..)
→ Einrichtung der Dev-Umgebung kann dennoch schwierig sein

ANFORDERUNGEN

- → Dozent beschreibt mit generischer Lösung ein Experiment mit fester Anzahl an Instanzen
- → Pro Instanz: Angabe von Name, Ports, Standard-Command
- → Studenten sollen mithilfe der Experimente gewisse Aufgaben lösen
- → Getrennte Netzwerke zwischen Experimenten und nach Außen
- → Upload von Programmpaketen im Webend
- → Angabe von Main-Class und Argumentenliste
- → Einzelnes Starten der Instanzen
- → Ausgabe der dabei geschehenen Logs (stdout, stderror)

ERHOFFTE VORTEILE

→ Dev-Umgebung zum Starten der Instanzen muss nicht zwingend eingerichtet werden

→ Konsistentes UI des Tools zwischen Aufgaben (& Vorlesungen?)

→ Durch Hochladen der Programmpakete pro Instanz wird Verteilung auf Netzwerkebene (hoffentlich) besser wahrgenommen

→ Hostnamen der Instanzen nicht localhost

DOCKER

DESCRIPTION

→ Software zum Deployment von Applikationen innerhalb von Containern

→ Ähnelt Vorgehensweise von Virtuellen Maschienen

→ Im Gegegsatz: Mithilfe von Docker-Deamon laufen Prozesse direkt auf Host-OS

→ Dennoch isoliert (CGroups, Kernel Namespaces, SELinux, AppArmor,..)

→ Keine spezifische Infrastruktur vorgeschrieben (bspw. ESXi)

→ Beste Unterstützung durch Linux-Derivat (Kernel-Tricks)

→ Mittlerweile lauffähige Konzepte auf Osx, Windows

CONTAINER

→ Deskriptive & triviale Beschreibung eines Systems und dessen Bestandteile (Rezept)

→ Instruktionen um Abhähigkeiten zu installieren, Konfigurationen vorzunehmen und Build-Steps auszuführen

→ Kein Transferieren von großen Builds nötig, da Image durch Dockerfile erzeugbar

→ Granulare Sub-Images

→ Trägt implizit zur Dokumentation bei: Jede Änderung an einem Container muss im Rezept ergänzt werden!

DOCKERFILE

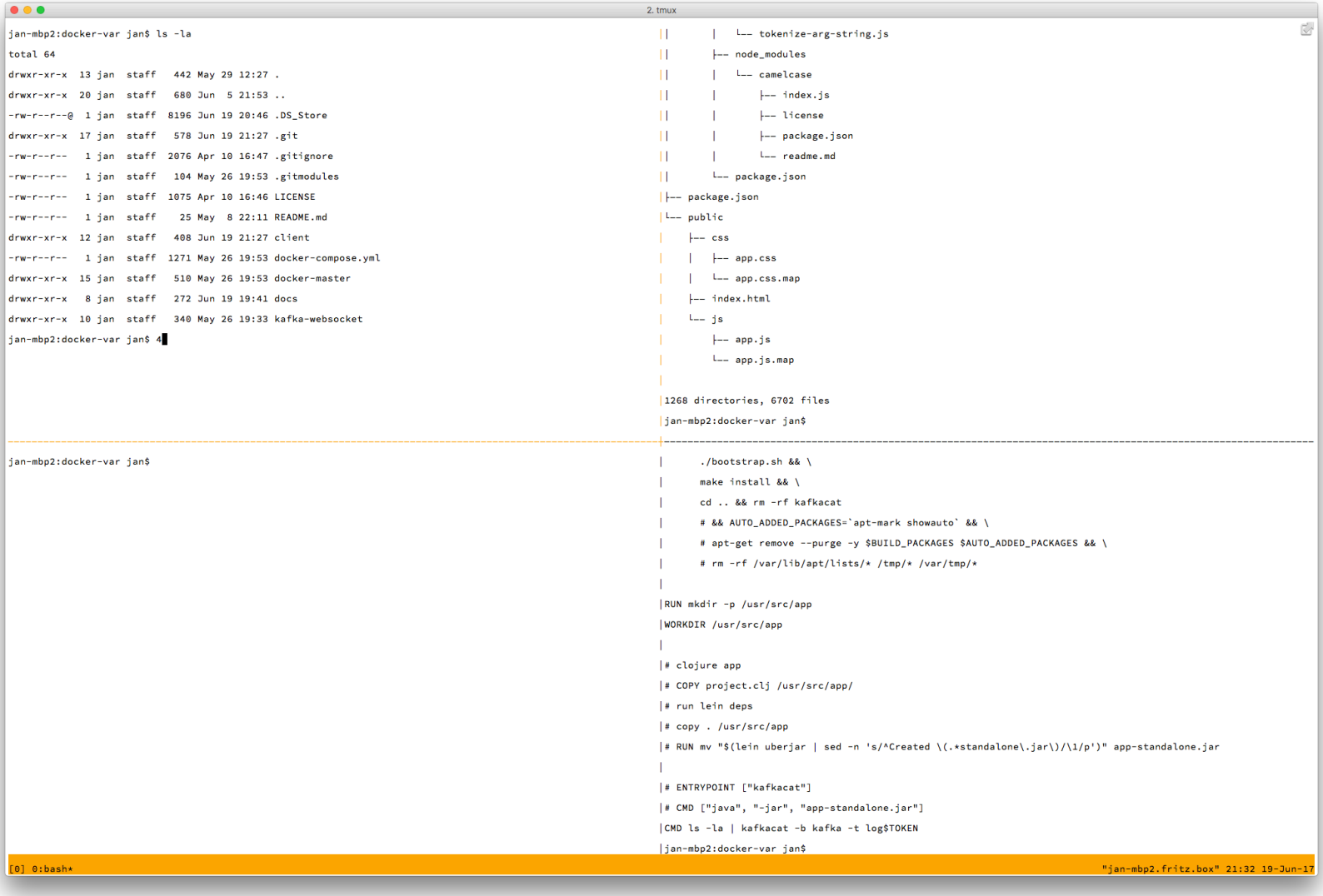
```
FROM clojure

RUN mkdir -p /usr/src/app
WORKDIR /usr/src/app

COPY project.clj /usr/src/app/
RUN lein deps
COPY . /usr/src/app

RUN mv "$(lein ring uberjar | sed -n 's/^Created \(.*standalone\.jar\)/\1/p')" app-standalone.jar

CMD ["java", "-jar", "app-standalone.jar"]
```

DOCKER-COMPOSE

- → Tool zur Unterstützung von Docker-Cli
- → Definition von Services als Bestandteil einer App
- → Ergänzende Angabe von Konfigurations-Parametern an Containern: Volume-Mounts, Ports, Env-Vars, Netzwerke,..
- → Muss sonst bei `docker exec` Command angegeben werden
- → Getrennte Environments für Dev, Test, Prod: `docker-compose[.dev].yml`
- → Zentrale Dokumentation!

INSPIRATION

TMUX PANES


2. tmux
jan-mbp2:docker-var jan$ ls -la
total 64
drwxr-xr-x 13 jan staff   442 May 29 12:27 .
drwxr-xr-x 20 jan staff   680 Jun 5 21:53 ..
-rw-r--r--@ 1 jan staff 8196 Jun 19 20:46 .DS_Store
drwxr-xr-x 17 jan staff   578 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 19:53 .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   510 May 26 19:53 docker-master
drwxr-xr-x   8 jan staff   272 Jun 19 19:41 docs
drwxr-xr-x 10 jan staff   340 May 26 19:33 kafka-websocket
jan-mbp2:docker-var jan$ 4
└── 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, 6702 files
jan-mbp2:docker-var jan$
jan-mbp2:docker-var jan$
./bootstrap.sh && \
make install && \
cd .. && rm -rf kafkacat
&& AUTO_ADDED_PACKAGES=`apt-mark showauto` && \
apt-get remove --purge -y $BUILD_PACKAGES $AUTO_ADDED_PACKAGES && \
rm -rf /var/lib/apt/lists/* /tmp/* /var/tmp/*
RUN mkdir -p /usr/src/app
WORKDIR /usr/src/app
clojure app
COPY project.clj /usr/src/app/
run lein deps
copy . /usr/src/app
RUN mv "$(lein uberjar | sed -n 's/^Created \(.*standalone\.jar\)/\1/p')" app-standalone.jar
ENTRYPOINT ["kafkacat"]
CMD ["java", "-jar", "app-standalone.jar"]
CMD ls -la | kafkacat -b kafka -t log$TOKEN
jan-mbp2:docker-var jan$
[0] 0:bash*
"jan-mbp2.fritz.box" 21:32 19-Jun-17

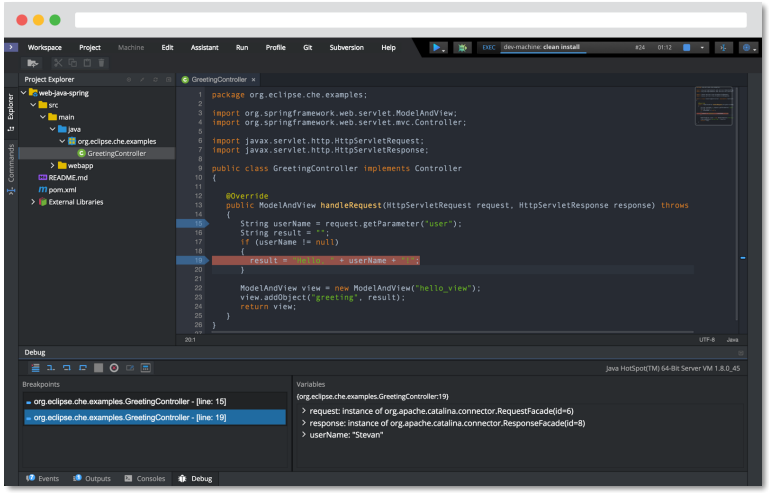
CLOUD-IDES

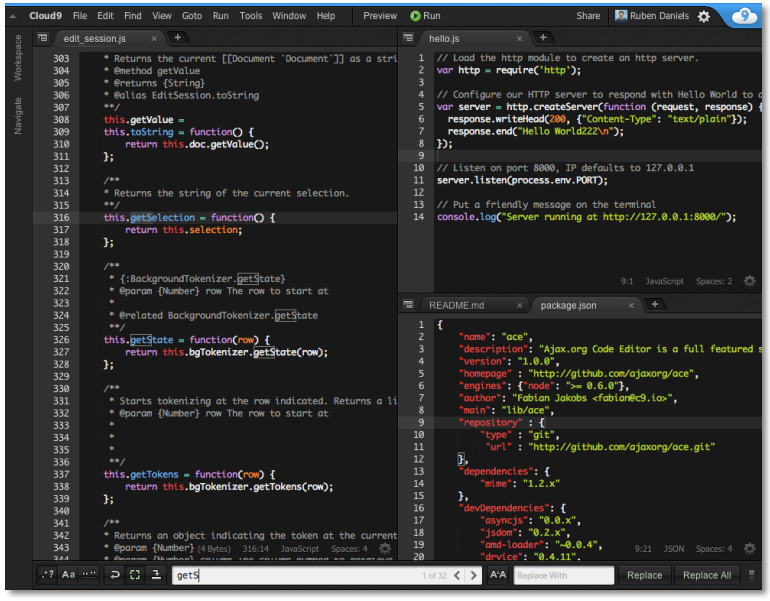
→ Cloud 9

→ Source Lair

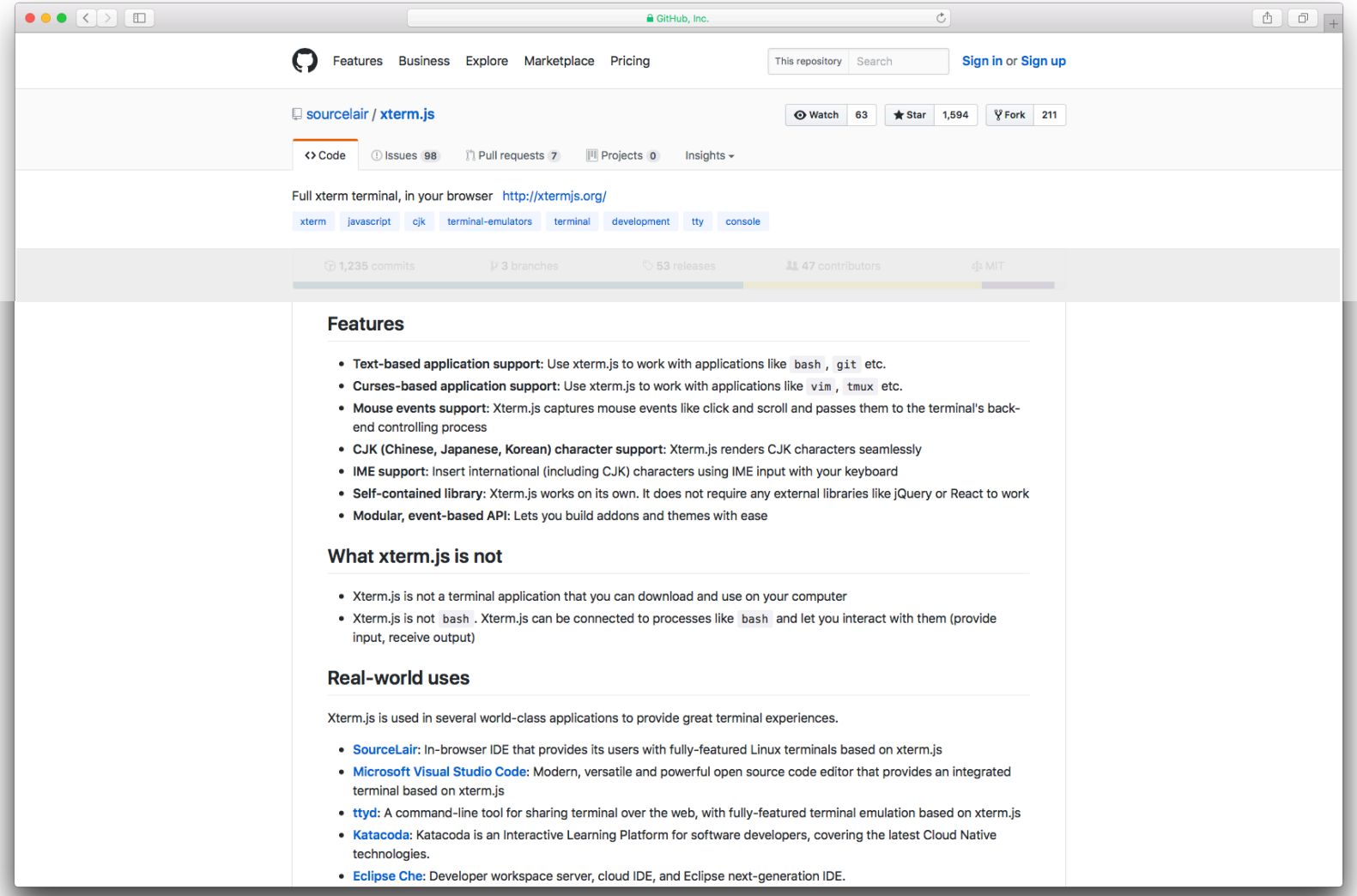
→ Eclipse Che

→ ...

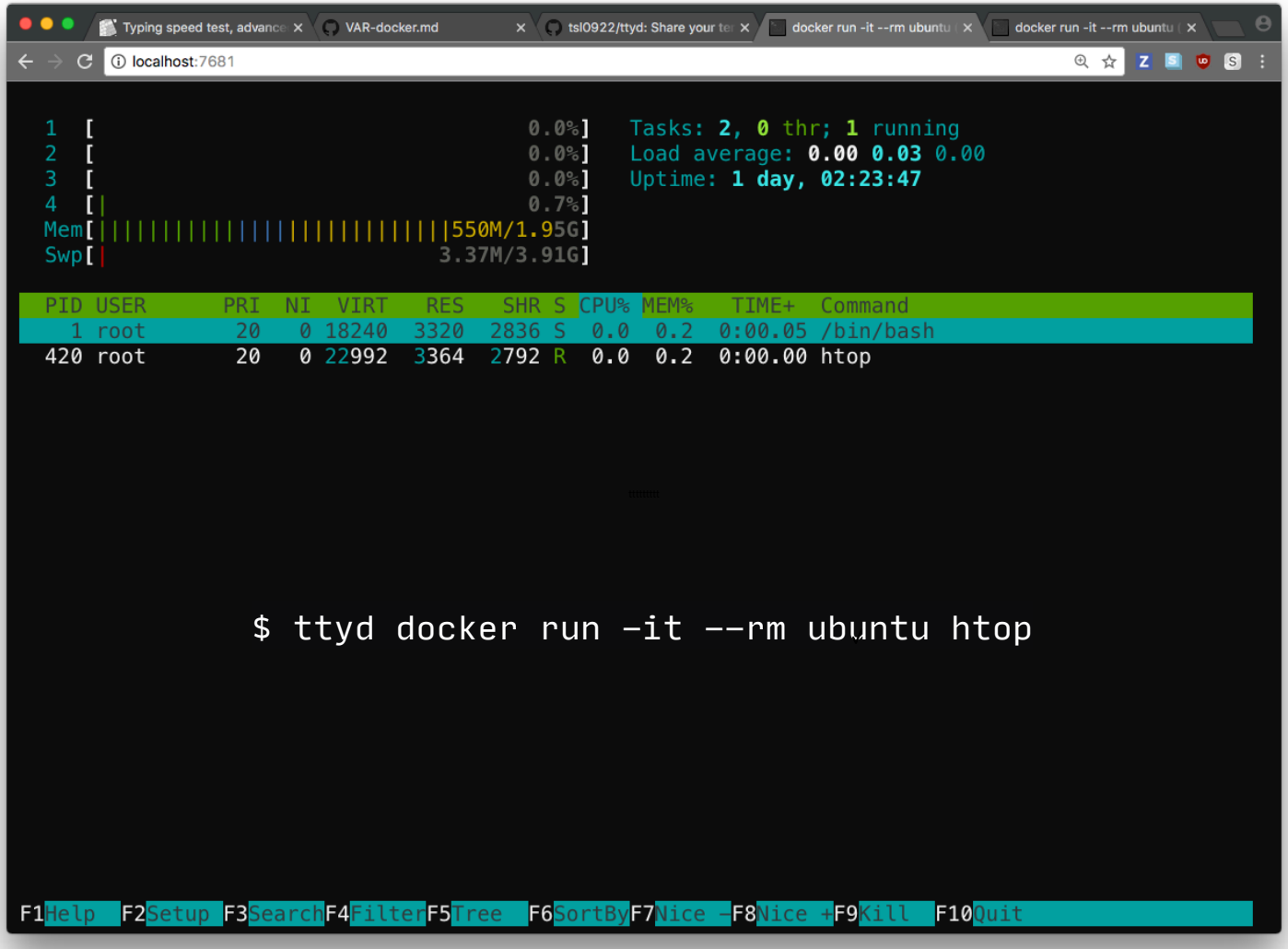




XTERM.JS


GitHub, Inc.
Features Business Explore Marketplace Pricing
This repository Search
Sign in or Sign up
sourcelair / xterm.js
Watch 63 Star 1,594 Fork 211
Code Issues 98 Pull requests 7 Projects 0 Insights
Full xterm terminal, in your browser http://xtermjs.org/
xterm javascript cjk terminal-emulators terminal development tty console
1,235 commits 3 branches 53 releases 47 contributors MIT
Features
Text-based application support: Use xterm.js to work with applications like bash, git etc.
Curses-based application support: Use xterm.js to work with applications like vim, tmux etc.
Mouse events support: Xterm.js captures mouse events like click and scroll and passes them to the terminal's back-end controlling process
CJK (Chinese, Japanese, Korean) character support: Xterm.js renders CJK characters seamlessly
IME support: Insert international (including CJK) characters using IME input with your keyboard
Self-contained library: Xterm.js works on its own. It does not require any external libraries like jQuery or React to work
Modular, event-based API: Lets you build addons and themes with ease
What xterm.js is not
Xterm.js is not a terminal application that you can download and use on your computer
Xterm.js is not bash. Xterm.js can be connected to processes like bash and let you interact with them (provide input, receive output)
Real-world uses
Xterm.js is used in several world-class applications to provide great terminal experiences.
SourceLair: In-browser IDE that provides its users with fully-featured Linux terminals based on xterm.js
Microsoft Visual Studio Code: Modern, versatile and powerful open source code editor that provides an integrated terminal based on xterm.js
ttyd: A command-line tool for sharing terminal over the web, with fully-featured terminal emulation based on xterm.js
Katacoda: Katacoda is an Interactive Learning Platform for software developers, covering the latest Cloud Native technologies.
Eclipse Che: Developer workspace server, cloud IDE, and Eclipse next-generation IDE.

TTYD


Typing speed test, advance
VAR-docker.md
tsl0922/ttyd: Share your te
docker run -it --rm ubuntu
docker run -it --rm ubuntu
localhost:7681
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]
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
$ ttyd docker run -it --rm ubuntu htop
F1Help F2Setup F3SearchF4FilterF5Tree F6SortByF7Nice -F8Nice +F9Kill F10Quit

„VAR-TOOL“

Experimente

Java RMI Chat	Excercise	Excercise	Excercise
Excercise	Excercise	Excercise	Excercise

Java RMI Chat

ChatServer

Container-Einstellungen

Dieser Container enthält bereits Dateien.

Ports

1099 — Out

1099
1337 — In

Main Class

ChatServer

Argumente

ChatClient

```
Exception in thread “main” java.io.FileNotFoundException: Could
not locate var_server/core__init.class or var_server/core.clj on
classpath. Please check that namespaces with dashes use
underscores in the Clojure file name., compiling:(var_server/
core_test.clj:1:1)
   at clojure.lang.Compiler.load(Compiler.java:7391)
   at clojure.lang.RT.loadResourceScript(RT.java:372)
   at clojure.lang.RT.loadResourceScript(RT.java:363)
   at clojure.lang.RT.load(RT.java:453)
   at clojure.lang.RT.load(RT.java:419)
   at clojure.core$load$fn__5677.invoke(core.clj:5893)
   at clojure.core$load.invokeStatic(core.clj:5892)
   at clojure.core$load.doInvoke(core.clj:5876)
   at clojure.lang.RestFn.invoke(RestFn.java:408)
   at clojure.core$load_one.invokeStatic(core.clj:5697)
```

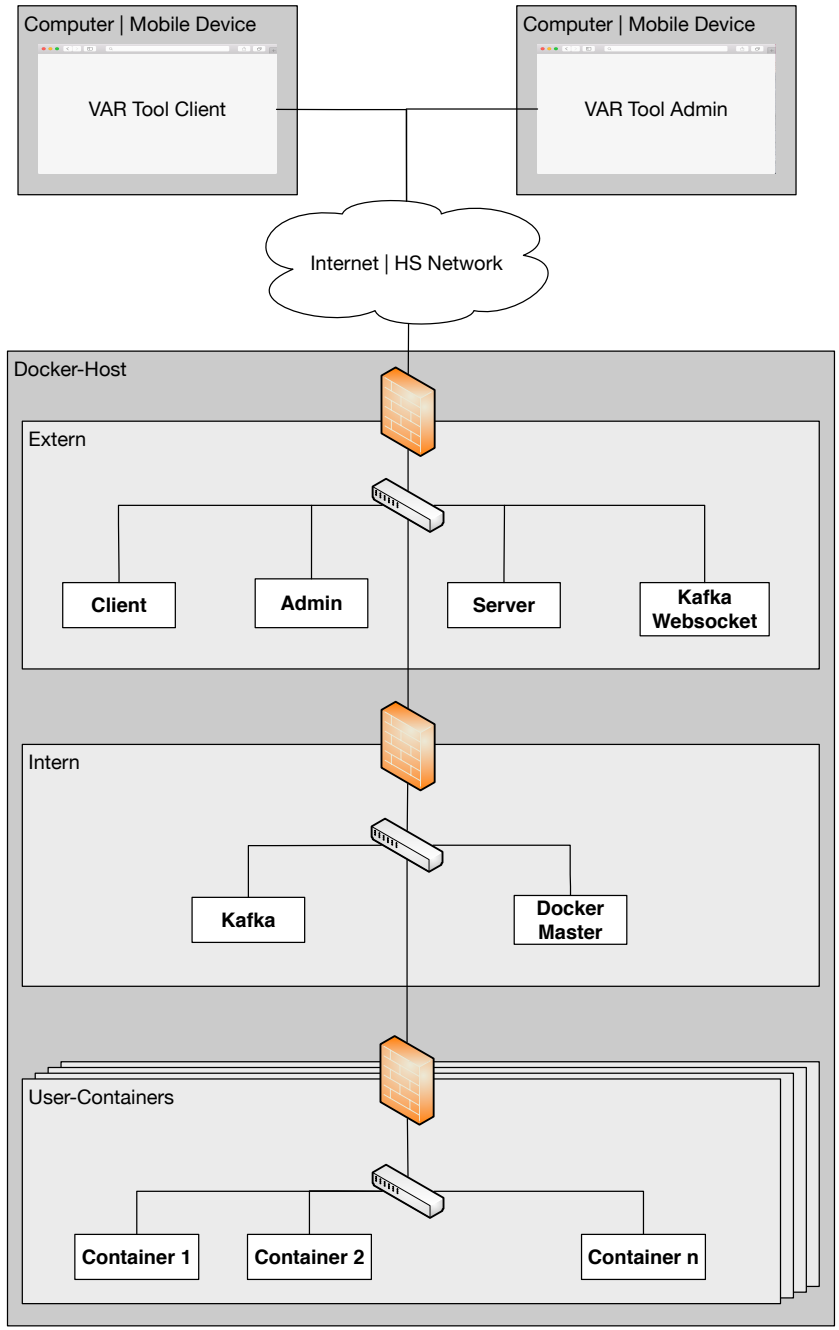
Container 3

Datei hochladen.

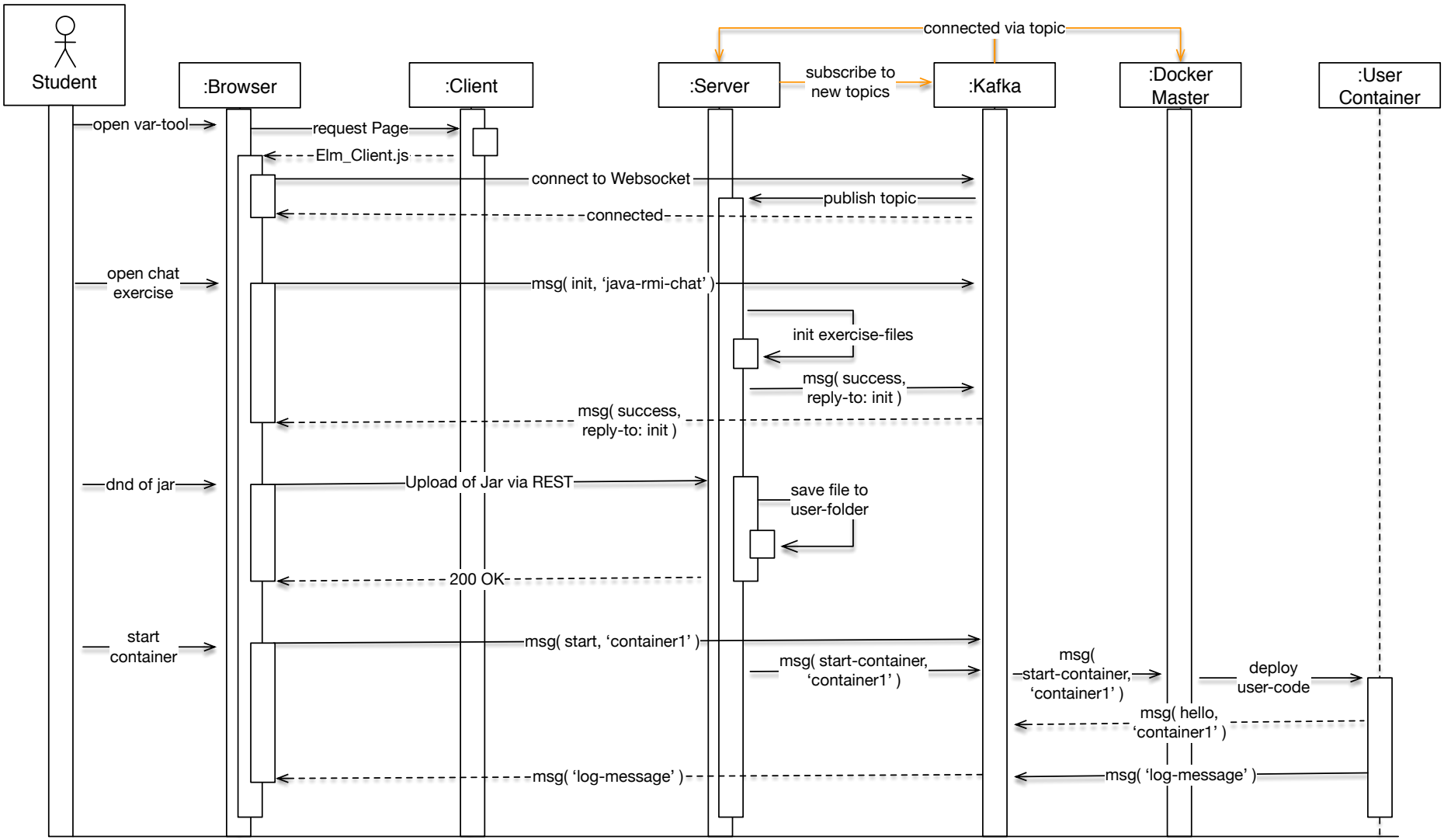
Container 4

Datei wird hochgeladen.

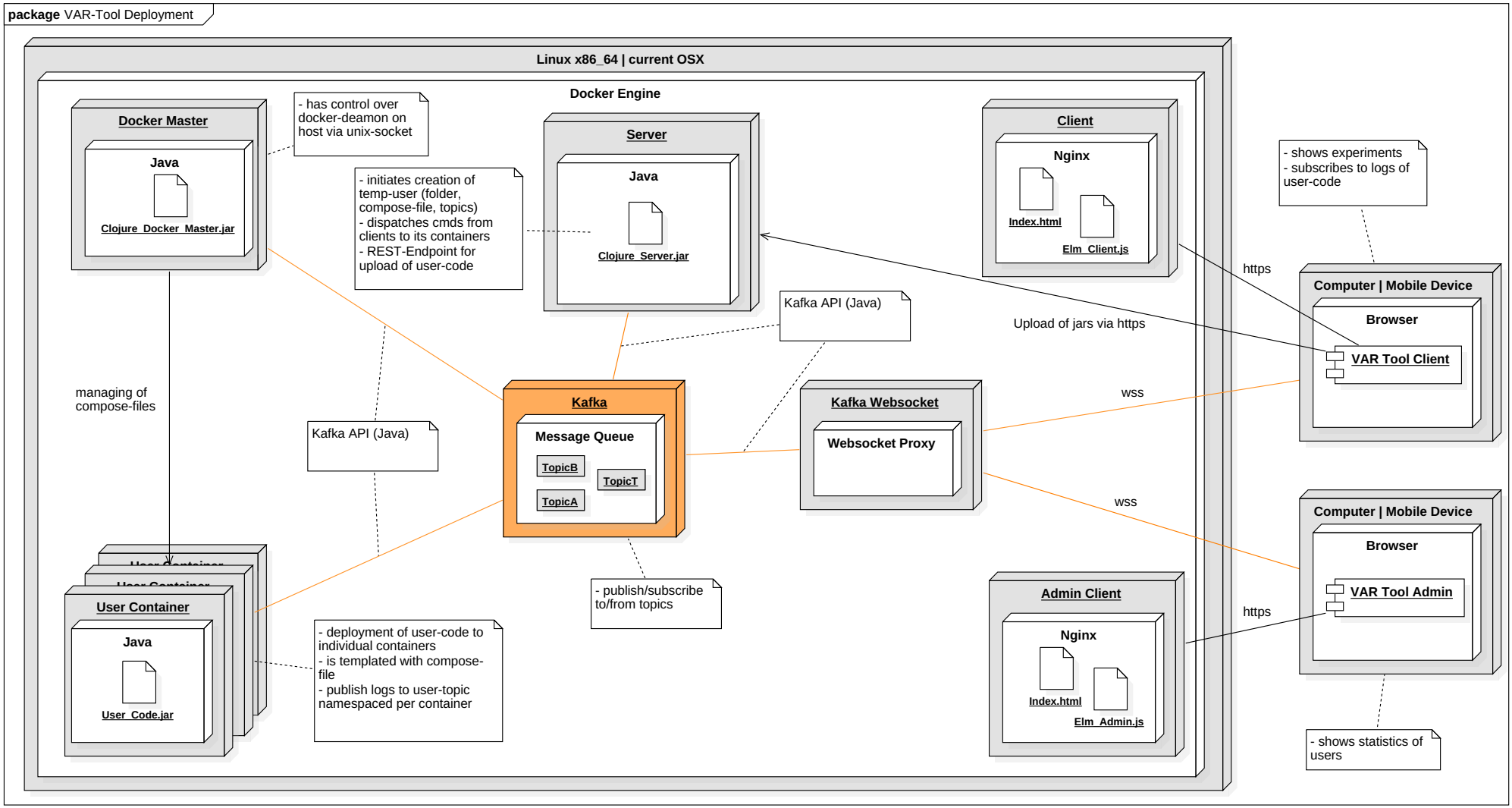
NETWORK


Computer | Mobile Device
VAR Tool Client
Computer | Mobile Device
VAR Tool Admin
Internet | HS Network
Docker-Host
Extern
Client
Admin
Server
Kafka Websocket
Intern
Kafka
Docker Master
User-Containers
Container 1
Container 2
Container n

SEQUENCE


Student
:Browser
:Client
:Server
subscribe to new topics
:Kafka
:Docker Master
:User Container
connected via topic
open var-tool
request Page
Elm_Client.js
connect to Websocket
publish topic
connected
open chat exercise
msg(init, 'java-rmi-chat')
init exercise-files
msg(success, reply-to: init)
msg(success, reply-to: init)
dnd of jar
Upload of Jar via REST
save file to user-folder
200 OK
start container
msg(start, 'container1')
msg(start-container, 'container1')
msg(start-container, 'container1')
deploy user-code
msg(hello, 'container1')
msg('log-message')
msg('log-message')

DEPLOYMENT


package VAR-Tool Deployment
Linux x86_64 | current OSX
Docker Engine
Docker Master
Java
Clojure_Docker_Master.jar
- has control over docker-deamon on host via unix-socket
Server
Java
Clojure_Server.jar
- initiates creation of temp-user (folder, compose-file, topics)
- dispatches cmds from clients to its containers
- REST-Endpoint for upload of user-code
Client
Nginx
Index.html
Elm_Client.js
- shows experiments
- subscribes to logs of user-code
https
Computer | Mobile Device
Browser
VAR Tool Client
Upload of jars via https
Kafka API (Java)
managing of compose-files
Kafka
Message Queue
TopicB
TopicT
TopicA
Kafka API (Java)
Kafka Websocket
Websocket Proxy
wss
wss
Computer | Mobile Device
Browser
VAR Tool Admin
https
- publish/subscribe to/from topics
User Container
Java
User_Code.jar
- deployment of user-code to individual containers
- is templated with compose-file
- publish logs to user-topic namespaced per container
Admin Client
Nginx
Index.html
Elm_Admin.js
- shows statistics of users

COMPOSE-FILE

```
# TODO networks
version: '3'
services:
  server:
    build: ./server
    container_name: server
    ports:
      - “8080:3000"

  client:
    build: ./client
    container_name: client
    ports:
      - “80:80"

  docker-master:
    build: ./docker-master
    container_name: docker-master
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
      - /var/data/users:/var/data/users/
    environment:
      - "TOKEN=ZYX321"
```

```
kafka:
  image: spotify/kafka
  ports:
    - "9092:9092"
    # and zookeeper
    - "2181:2181"
  environment:
    - "ADVERTISED_PORT=9092"

 kafka-websocket:
    build: ./kafka-websocket
    container_name: kafka-websocket
    ports:
      - "7080:7080"
```

AUSBLICK

AUSBLICK & FAZIT

→ EFP: VAR-Tool, Bewertung & Feedback
→ Thesis
→ Veröffentlichung?

→ Sehr spannend
→ Viel Raum für weitere (studentische) Arbeiten
→ Erstrebenswerte Aufgabe der Fakultät

FIN

Danke für die Aufmerksamkeit. – Fragen?