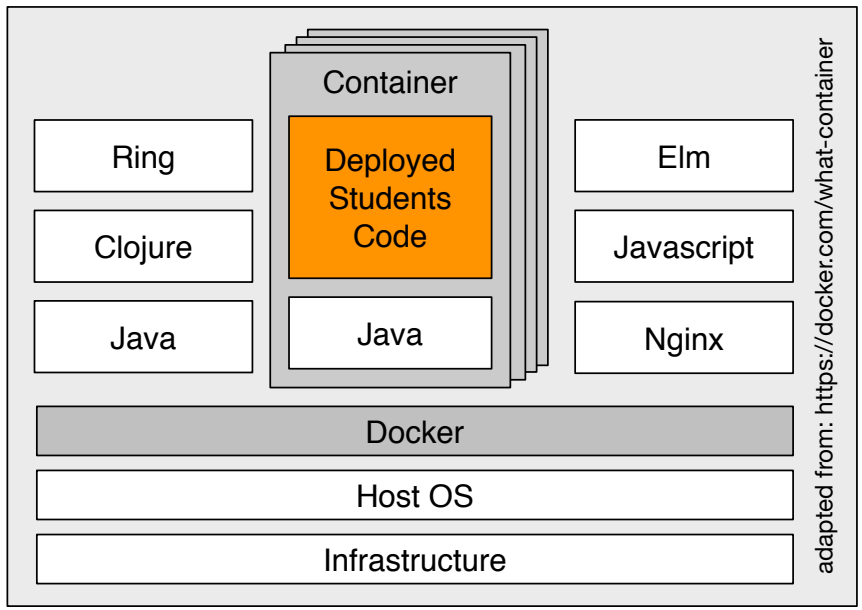
VIRTUALISIERTE ARBEITSUMGEBUNG FÜR DEN TEST VERTEILTER SYSTEME

Microservice-Architektur mit Docker, Clojure & Elm

13. Dezember 2017

> S. 4



Jan-Philipp Willem

Fakultät für Informatik
Hochschule Mannheim

GLIEDERUNG

1. Anforderungen
2. Technologien
3. Inspirationen für ein UI
4. „VAR-Tool“
5. Design-Entscheidungen

→ **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, stderror) 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

→ Docker-Compose: Definition von Services als Bestandteil einer App

> S. 1

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

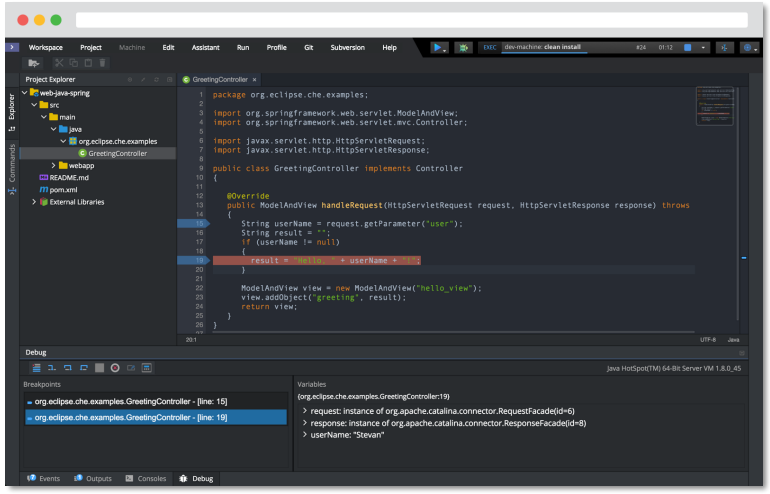
ELM

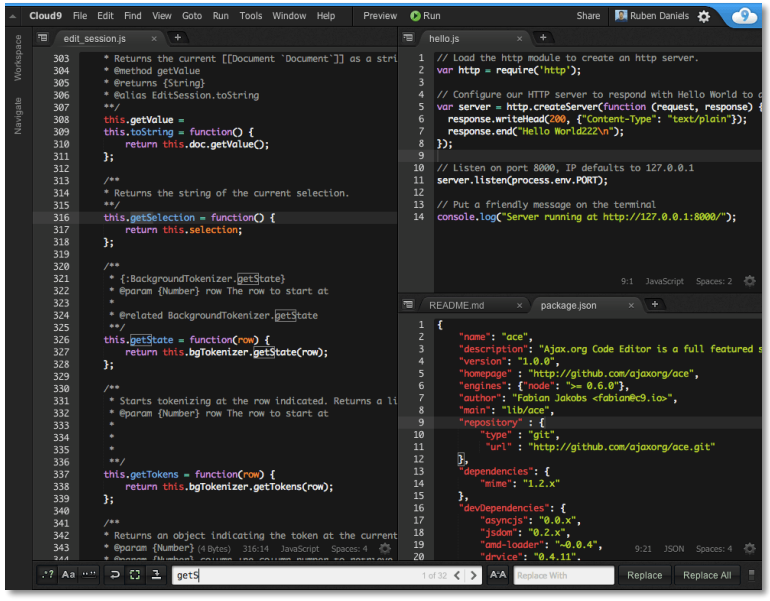
- → Rein funktionale Sprache
- → ML-artig, ähnlich zu bspw. Haskell
- → Kompiliert zu JavaScript & Interop
- → DSL für HTML, SVG
- → Sinnvolle Abstraktionen wie bspw. von Websockets
- → Virtual DOM, ähnlich zu React
- → Statisch typisiert, dabei sehr hilfreicher Compiler +
- → Kein NULL oder undefined,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

INSPIRATIONEN FÜR EIN UI

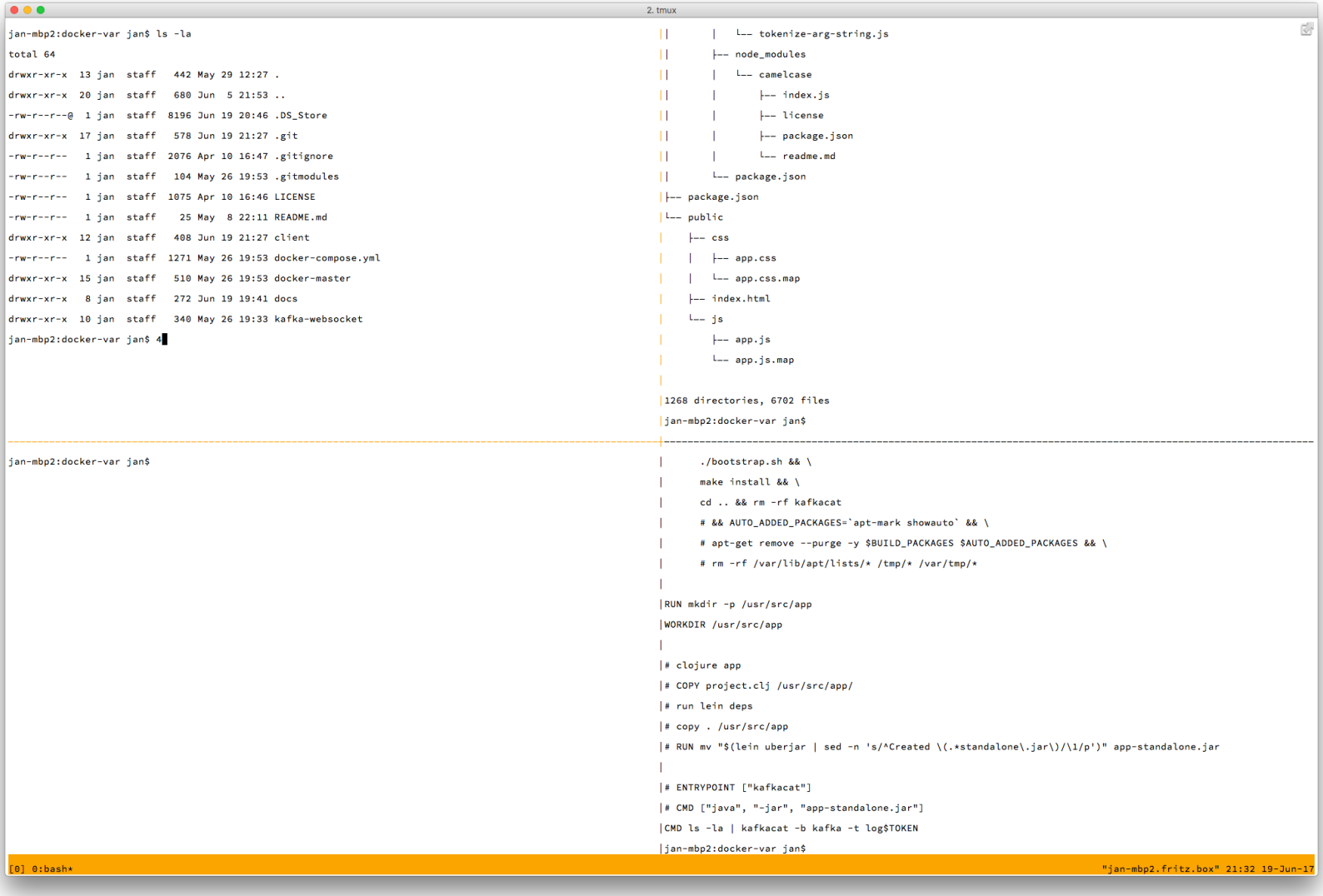
CLOUD-IDES

- → Cloud 9
- → Source Lair
- → Eclipse Che
- → ...

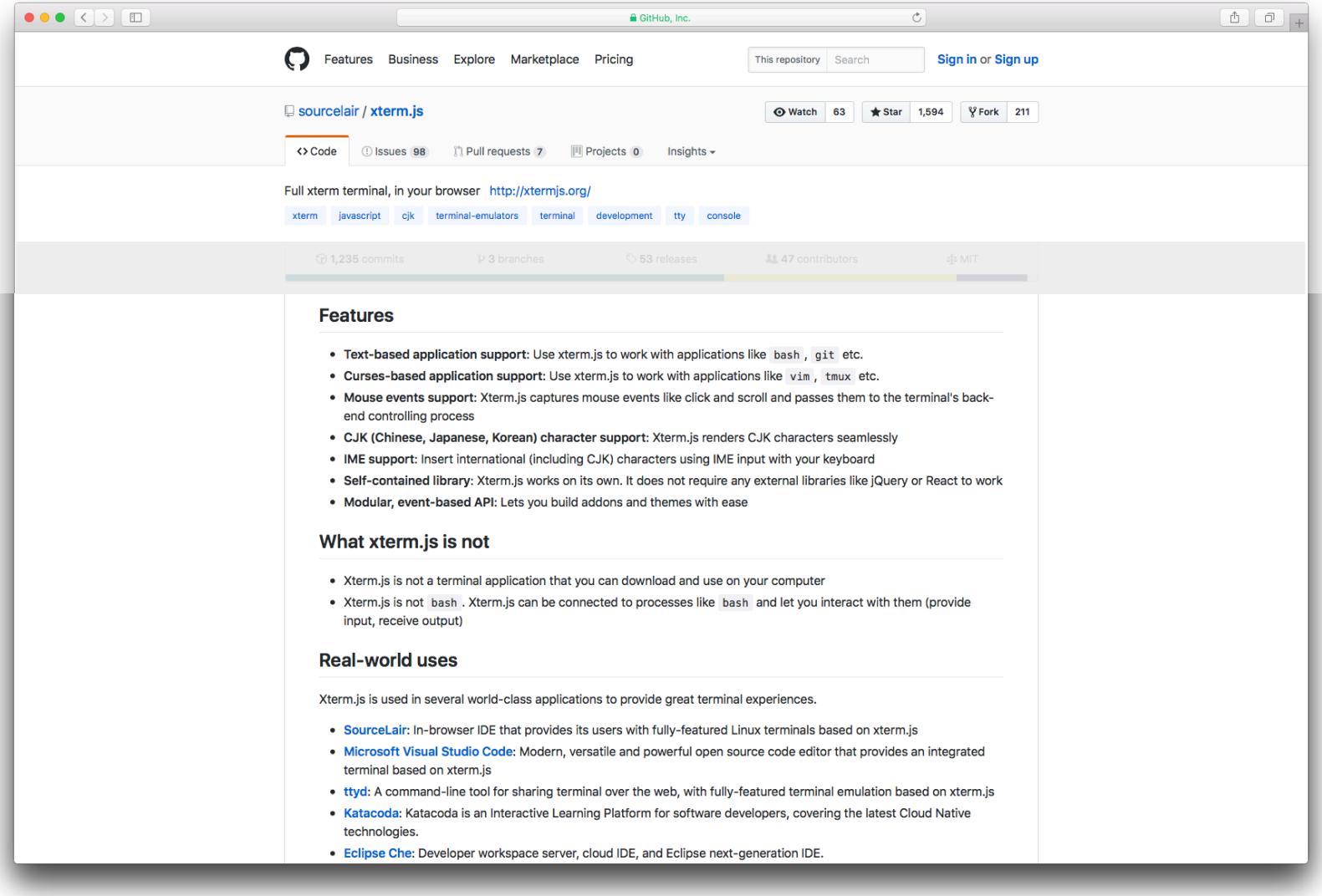




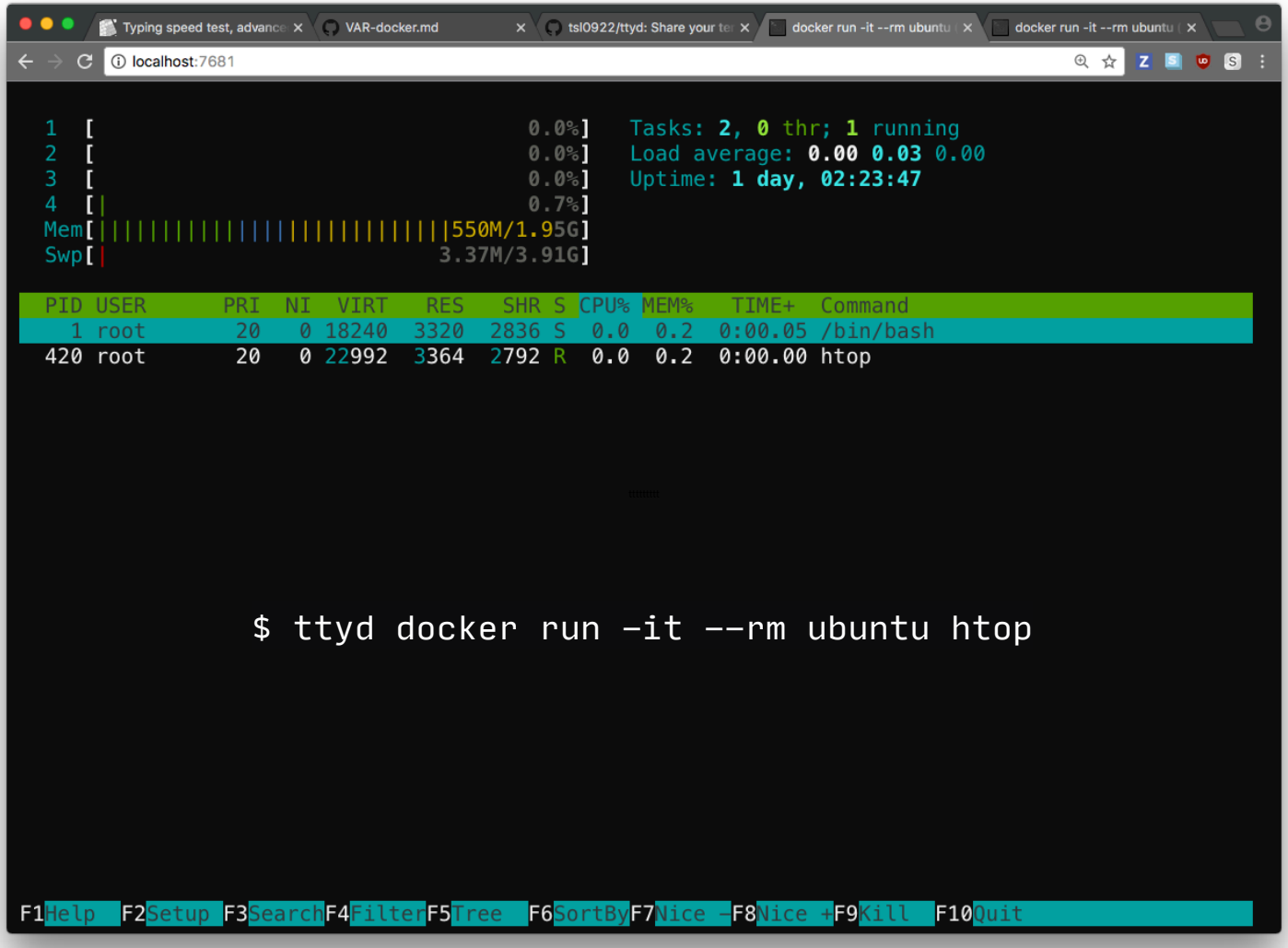
TMUX PANES


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
jan-mbp2:docker-var jan$
[0] 0:bash*
"jan-mbp2.fritz.box" 21:32 19-Jun-17

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

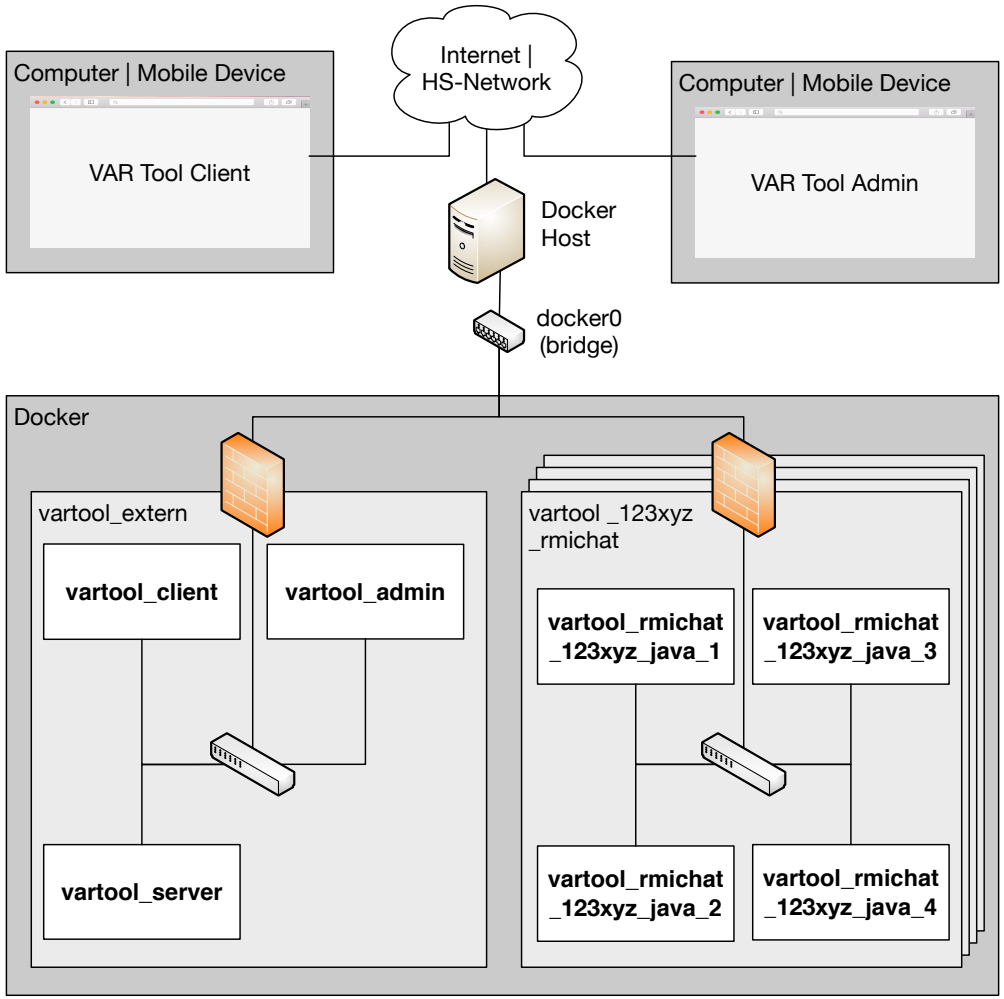

Typing speed test, advance
VAR-docker.md
tsl0922/ttyd: Share your te
docker run -it --rm ubuntu
docker run -it --rm ubuntu
localhost:7681
1 [0.0%] Tasks: 2, 0 thr; 1 running
2 [0.0%] Load average: 0.00 0.03 0.00
3 [0.0%] Uptime: 1 day, 02:23:47
4 [| 0.7%]
Mem[|||||||||||||||||||||||550M/1.95G]
Swp[| 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

PROZESS

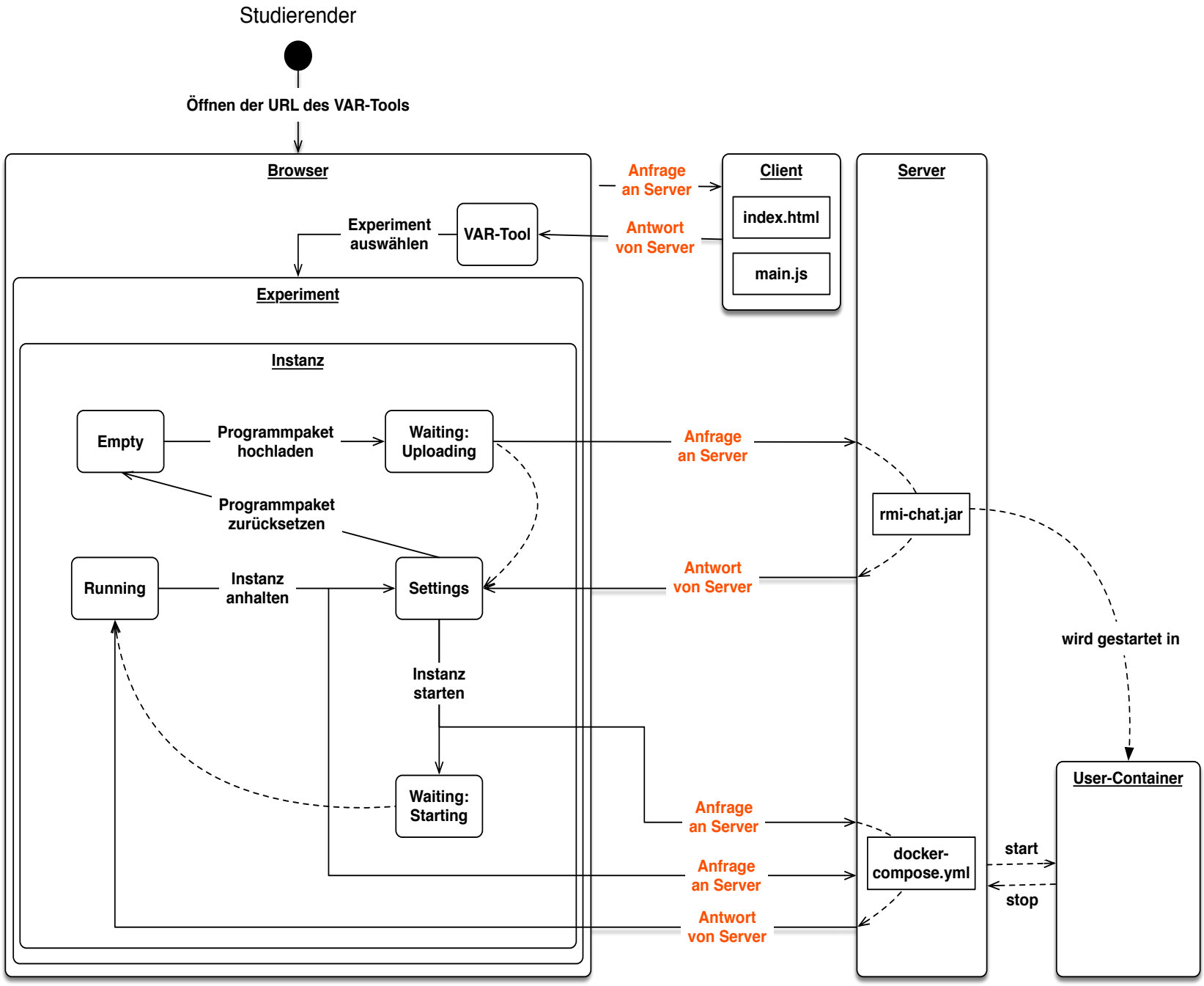
- → Recherche
- → Paper-Scribbles
- → HighFi-Mockups
- → Implementierung als granulare Funktionen in Elm
  - → Components

„VAR-TOOL“

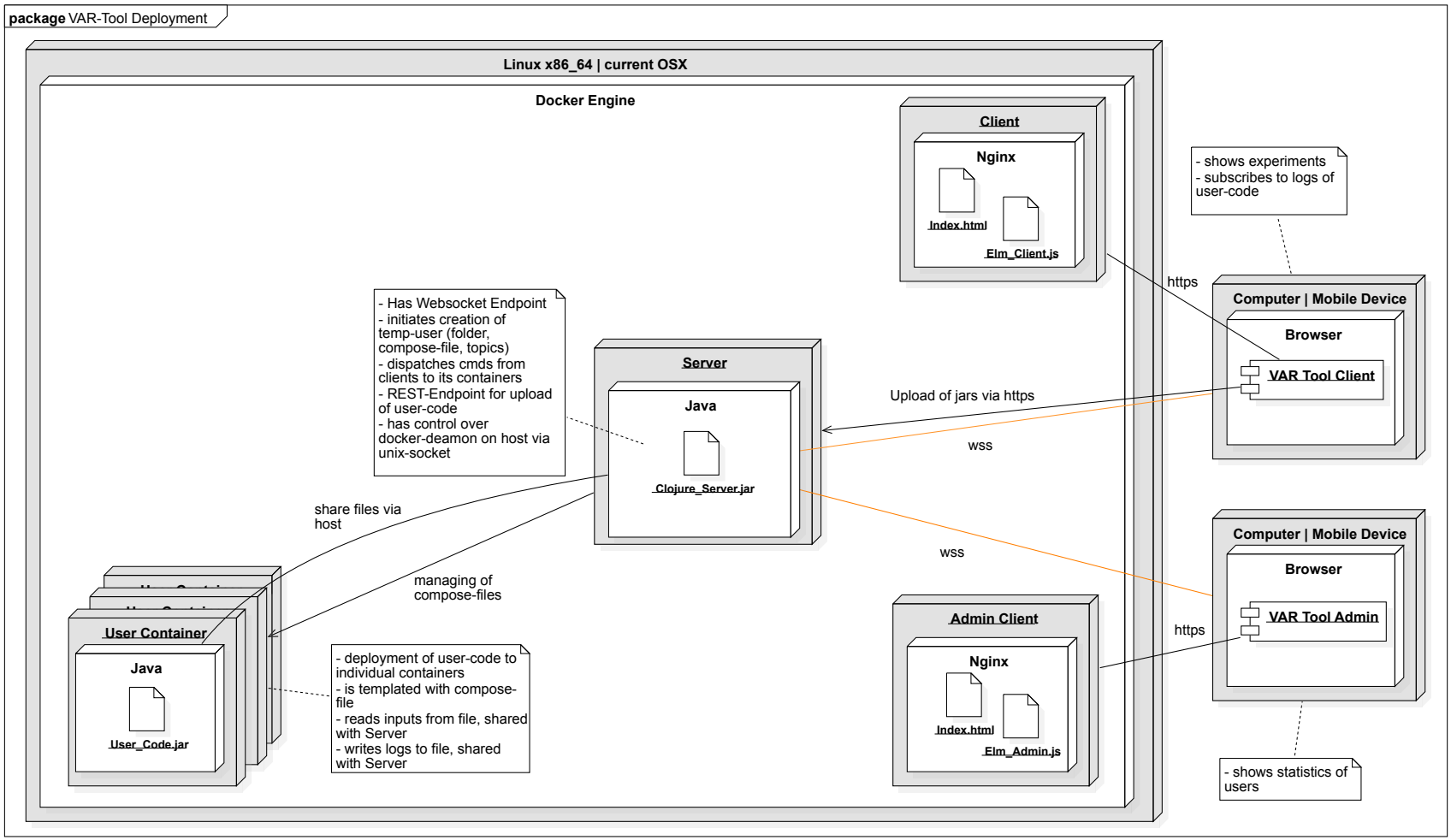
NETWORK


Computer | Mobile Device
VAR Tool Client
Internet |
HS-Network
Computer | Mobile Device
VAR Tool Admin
Docker
Host
docker0
(bridge)
Docker
vartool_extern
vartool_client
vartool_admin
vartool_server
vartool _123xyz
_rmichat
vartool_rmichat
_123xyz_java_1
vartool_rmichat
_123xyz_java_3
vartool_rmichat
_123xyz_java_2
vartool_rmichat
_123xyz_java_4

STATE


Studierender
Öffnen der URL des VAR-Tools
Browser
Experiment auswählen
VAR-Tool
Experiment
Instanz
Empty
Programmpaket hochladen
Waiting: Uploading
Programmpaket zurücksetzen
Running
Instanz anhalten
Settings
Instanz starten
Waiting: Starting
Anfrage an Server
Antwort von Server
Client
index.html
main.js
Server
rmi-chat.jar
wird gestartet in
User-Container
docker-compose.yml
start
stop

DEPLOYMENT


package VAR-Tool Deployment
Linux x86_64 | current OSX
Docker Engine
Client
Nginx
Index.html
Elm_Client.js
- shows experiments
- subscribes to logs of user-code
https
Computer | Mobile Device
Browser
VAR Tool Client
- Has Websocket Endpoint
- initiates creation of temp-user (folder, compose-file, topics)
- dispatches cmds from clients to its containers
- REST-Endpoint for upload of user-code
- has control over docker-deamon on host via unix-socket
Server
Java
Clojure_Server.jar
Upload of jars via https
wss
share files via host
managing of compose-files
wss
Computer | Mobile Device
Browser
VAR Tool Admin
https
User Container
Java
User_Code.jar
- deployment of user-code to individual containers
- is templated with compose-file
- reads inputs from file, shared with Server
- writes logs to file, shared with Server
Admin Client
Nginx
Index.html
Elm_Admin.js
- shows statistics of users

DESIGN-ENTSCHEIDUNGEN

DESIGN-ENTSCHEIDUNG (1/5)

Docker als Infrastruktur-Layer → Entscheidung: gut

- → Vorgabe von Prof. Leuchter
- → Zusätzlich: App in Docker, statt VM
- → Docker in Docker // Binding von Docker-Socket +
- → Problem: Volume-Mount von `data/` in User-Container leer
  - → weil Server startende Instanz kann eigener Mount von Host nicht reaktiv weitergeleitet werden (Kopie wäre möglich)
- → Integration von User-Container und Server über Docker-Daemon
  - → Bestehende Clojure- oder Java-Sdk-Clients ungeeignet
  - → Curl als Shell-Command

DESIGN-ENTSCHEIDUNG (2/5)

Trennung von Front- und Backend → Entscheidung: gut

→ Ermöglicht Austauschbarkeit von jeweiliger Implementierung

→ Parallele Arbeit möglich, ohne den anderen Service zu verletzen

→ Back- & Frontend erfordern eigene Build-Pipelines

→ Development-Container erleichtern die Integration des Toolings beim Entwickeln

DESIGN-ENTSCHEIDUNG (3/5)

Eigene Websocket-Implementierung → Entscheidung: gut

- → Ursprünglich war Apache Kafka als Integrations-Punkt gedacht, mit Websocket-Adapter Verbindung zum Browser
  - → Nur Empfangen, wenig konfigurierbar
- → Nutzen von fertiger Lib für eine TTY-Verbindung im Browser lässt wenig Platz für eigene Implementierung bzw. Use-Case
- → Pro TTY bzw. Instanz würde man ohne Customization einen weiteren WS-Channel nutzen
- → Eigene Lösung ermöglicht die Kommunikation von Server und Client mittels einzelnem Channel
- → Das Protokoll kann selbst definiert werden

```
{ kind: command,
subkind: start-instance,
payload: { experimentId: rmichat,
         instanceId: 1,
         mainClass: var.rmi.chat.ChatClient,

         arguments: Anton }}
```

```
{ kind: message,
subkind: log,
payload: { experimentId: rmichat,
         instanceId: 1,

         log: Hello}}
```

DESIGN-ENTSCHEIDUNG (4/5)

Elm für Frontend → Entscheidung: gut

- → Szenario passend für Single-Page-Application (SPA)
- → reaktiv → Virtual DOM
- → State-Handling mit TEA
- → rein funktional, Immutable
- → Websocket als Task mit resultierender Elm-Msg bei Senden und Empfangen auf WS-Channel
- → Encoding/Decoding von JSON-Strings zu Elm-Types
  - → Typisiertes Message-Protokoll

DESIGN-ENTSCHEIDUNG (5/5)

Clojure für Backend → Entscheidung: mittelmäßig

- → Lisp macht Spaß
- → Ring-Implementierung erwies sich als problematisch :(
  - → CSRF-Token trotz Nutzen von API-Settings
  - → Reihenfolge von Ring-Handlern entscheidend
- → JSON-Encoding & -Decoding war durch Lib sehr einfach
- → Bestehendes JAVA-Ecosystem: bspw. UUID
- → Abstraktion des Websocket-Handlers von Http-Kit war hilfreich
  - → State-Machine
- → Immutable :)

FAZIT & AUSBLICK

→ Viele Probleme, da Fokus auf Full-Stack
→ Integration der Komponenten
→ Üben von Polyglott

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