

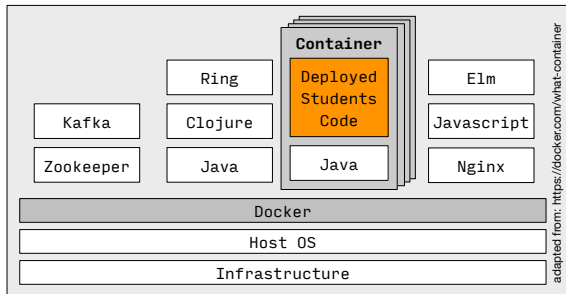
VIRTUALISIERTE ARBEITSUMGEBUNG FÜR DEN TEST VERTEILTER SYSTEME

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, stderr)

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 Maschinen
- Im Gegensatz: 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 Abhängigkeiten 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].yaml`
- Zentrale Dokumentation!

INSPIRATION

TMUX PANES

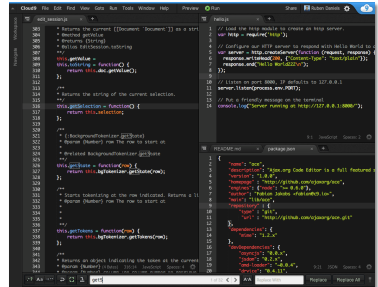
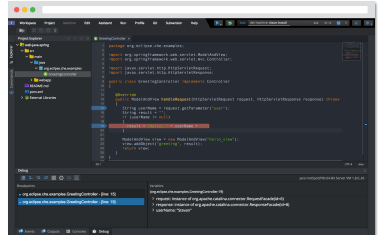
```

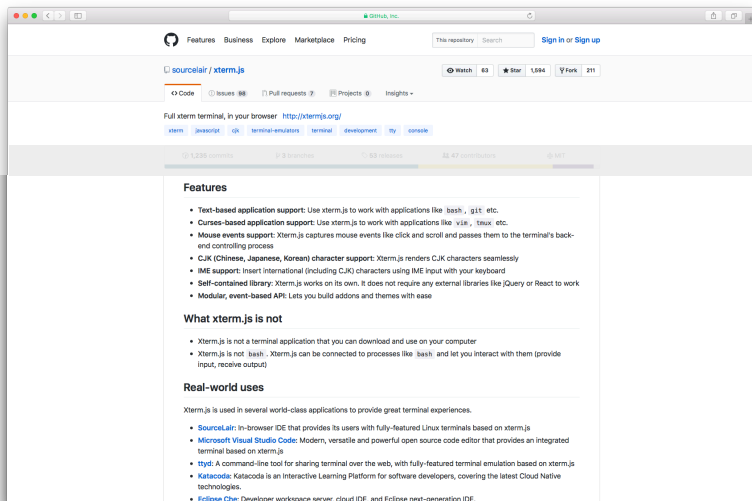
4/28/2020
jen-mbp21docker-var jan$ ls -ls
total 64
drwxr-xr-x 13 jan staff 442 May 29 12:27 .
drwxr-xr-x 20 jan staff 608 Jun 5 21:53 ..
-rw-r--r-- 1 jan staff 8196 Jun 19 20:46 .DS_Store
drwxr-xr-x 17 jan staff 678 Jun 19 21:27 .git
-rw-r--r-- 1 jan staff 2076 Apr 10 16:47 .gitignore
-rw-r--r-- 1 jan 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 $AUTO_ADDED_PACKAGES && \
| # rm -rf /var/lib/apt/lists/* /tmp/* /var/tmp/*
|
| RUN mkdir -p /usr/src/app
| WORKDIR /usr/src/app
|
| # closure app
| # COPY project.clj /usr/src/app/
| # run lein deps
| # copy . /usr/src/app
| # RUN mv "lib/uberjar" sed -n 's/^Created (.*/standalone.jar)/i/p' app-standalone.jar
|
| # ENTRYPOINT ["kafkacat"]
| # CMD ["java", "-jar", "app-standalone.jar"]
| CMD ls -ls | kafkacat -b kafka -t logSTOKEN
jen-mbp21docker-var jan$

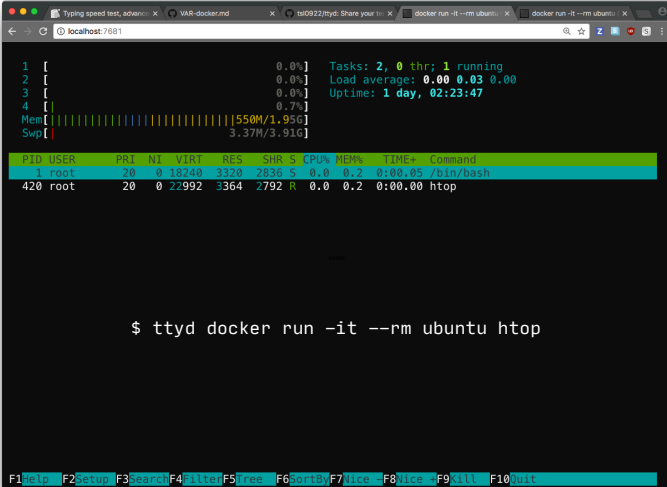
```

CLOUD-IDES

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







The screenshot shows a web browser window with several tabs. The active tab is titled 'localhost:7681'. The browser's address bar shows 'localhost:7681'. The main content area displays a terminal window with a black background and green and red text. The terminal shows the output of the 'htop' command, which displays system statistics and a list of running processes. The statistics section shows: Tasks: 2, 0 thr; 1 running; Load average: 0.00 0.03 0.00; Uptime: 1 day, 02:23:47. The memory section shows: Mem[|||||||||||||||||||||||||||||||||]550M/1.95G; Swp[|]3.37M/3.91G. The process list shows two processes: PID 1, USER root, PRI 20, NI 0, VIRT 18240, RES 3320, SHR 2836, S S, CPU% 0.0, MEM% 0.2, TIME+ 0:00.05, Command /bin/bash; and PID 420, USER root, PRI 20, NI 0, VIRT 22992, RES 3364, SHR 2792, S R, CPU% 0.0, MEM% 0.2, TIME+ 0:00.00, Command htop. At the bottom of the terminal window, there is a prompt '\$ ttyd docker run -it --rm ubuntu htop'. Below the terminal window, there is a footer bar with function key shortcuts: F1 Help, F2 Setup, F3 Search, F4 Filter, F5 Tree, F6 SortBy, F7 Nice, F8 Nice, F9 Kill, F10 Quit.

```
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

F1 Help F2 Setup F3 Search F4 Filter F5 Tree F6 SortBy F7 Nice F8 Nice F9 Kill F10 Quit
```


„VAR-TOOL“

Experimente

Java RMI Chat

Excercise

Excercise

Excercise

Excercise

Excercise

Excercise

Excercise

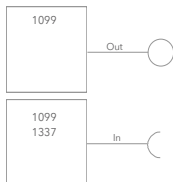
ChatServer



Container-Einstellungen

Dieser Container enthält bereits Dateien. 📁

Ports



Main Class

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



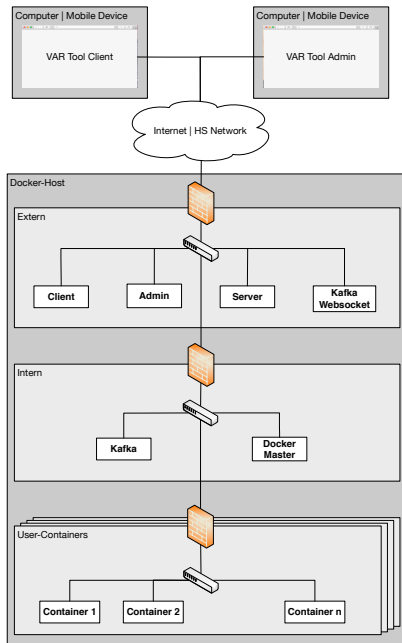
Datei hochladen.

Container 4

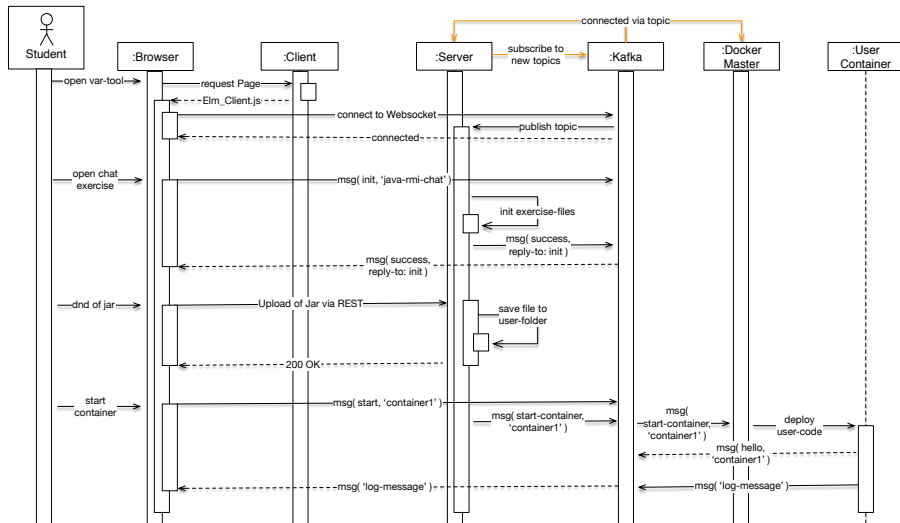


Datei wird hochgeladen.

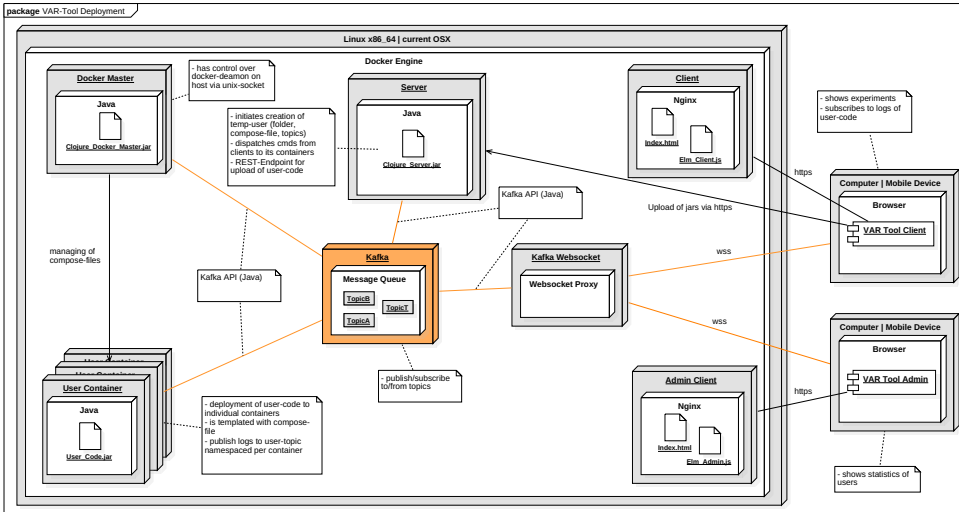
NETWORK



SEQUENCE



DEPLOYMENT



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?