Project Presentation Stage 1

Stefanie Ziltener, Marc Heimgartner, Benjamin Bürgisser, Simon Tännler

Advanced Software Engineering FS 2017, University of Zürich

High-level Overview: Client-Server
^ we will go through each of these boxes

Containerized Microservices deployed through Kubernetes

Dummy Image

- Components embedded in (Docker) Containers

- Containers have (ideally) one single responsibility

Containers as the instantiation of a Microservice
^ We've seen the benefits of Containers in the lecture

Frontent and API

- User inputs term
- Click on register
- Send request to API

Dummy Image

Explain the Architecture by
^ following the Story of a Request through
our Architecture

Timeseries: MongoDB (I)

- Request Handler stores Term in MongoDB

- Persistency guaranteed by GCE [Persistence Disk](#)

Dummy Image

TODO: Img of newly created Term in JSON

Twitter Service

- Gets notified of newly created Terms
- Streaming stops and restarts with the new Term added for <u>tracking</u>
- Arriving Tweets are immediately stored into the Queue

Dummy Image

Restarts are due to Streaming API limitations
^ We're trying to keep this a light as possible
^ b/c we have some limitation which we'll talk about later

Worker Queue

- <u>Redis</u>: in-memory data structure store
- A FIFO queue of Strings (Tweets)
- Load generator
  - API endpoint to add Strings to Queue directly

Dummy Image

Compute Workers

- Running Workers process the Queue:
  - Assign Tweet to Term (filtering)
  - Calculate Sentiment
- Length of the Queue defined the number of Workers
  - Scaled through Kubernetes.

Twitter API does not tell you which term the tweet matched on

Timeseries: MongoDB (II)

- Workers store the calculated Sentiment into MongoDB.

Dummy Image

The amount of data is actually very small
^ TODO: Img of newly created Term in JSON

Displaying Results

- Rest API gets request

- Collect relevant data from MongoDB

- Browser renders data

- Socket gets opened for continues pushes

Dummy Image

Architectural Styles

- Client / Server through Rest API

- Event-Driven notification of Term updates

- Pipes and Filters

- Blackboard: Redis Queue

  - Factory: Twitter Service

  - Worker: Compute Workers

- Highly decoupled

TODO: this feels like it needs more work

Do you even scale?
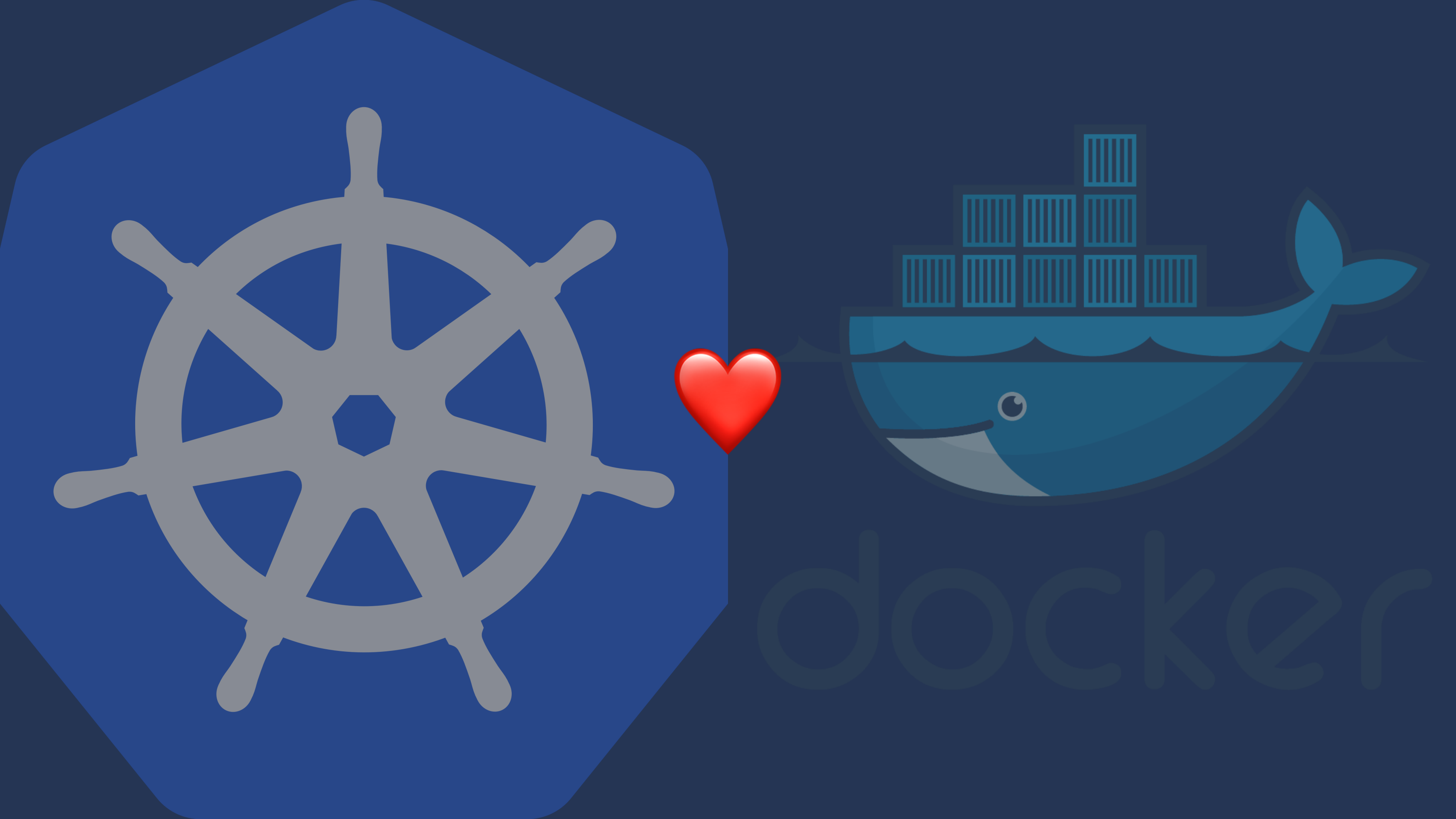
Kubernetes in one slide

@marc chasch ächt du das no mache?

Containers embedded in Pods

What does <u>Kubernetes</u> do for us

- Every component is potentially scalable through Kubernetes

  - Even <u>MongoDB</u>!

- Fault Tolerance:

  - Container recovery through Kubernetes

  - Decoupled design and Microservice

- Elasticity

  - Container scaling through Kubernetes

Concernes

- High lock-in to Kubernetes

- Twitter

  - Only 400 Term, thus no scaling

  - May not match to terms perfectly

- Redis: may become a bottleneck

  - but we highly doubt it

-

TODO: this feels like it needs more work

Technology Zoo - Rational

Cloud Platform: <u>Google Container Engine</u> (GCE)
 * Easy support of <u>Kubernetes</u>

Containerization: <u>Docker</u>
 * Popular Container engine

Container orchestration: <u>Kubernetes</u>
 * Popular Container orchestration

Programming Language: Google Go
 * New Programming language 🎉

Dev Environment

Docker containers using Docker Compose

Demo

Questions?