AssertJ

Debugging in Microservice-Umgebungen

TestNG

Learning Outcomes

Fehlersuche in Anwendungen

- Wie debugge ich eine Anwendung? - Microservices im Speziellen

  Nicht zu verwechseln mit Quality Control

Empfehlungen (Player Development)

- Wie debugge ich (effizient) meinen Player und verbessere seine Robustness?

(Debug-)Tools

Tools

- Testing

  Vermeiden von Problemen im Vorfeld durch gutes Testing

- IDE-Debugger

  Manuelles Setzen von Breaking Points und Ablaufen der Program-Instructions

- (Service-) Logs

  Logs einer einzelnen Instanz einer Anwendung

- Tracing

  Tracing von (fachlichen) Prozess-Zusammenhängen über Servicegrenzen hinweg

- Log Aggregation w/ Tracing

  Zentrale Auswertung und Analyse zusammenhängender Logs

Testing

Warum testet man?

- Prüfen, ob etwas funktioniert

  Sicherstellen, dass ein neues Feature funktioniert.

- Prüfen, wann etwas nicht mehr funktioniert - Regression Tests

  Sicherstellen, dass neue Features (oder Refactorings) keine Nebeneffekte verursachen.

Testing

Application Testing

- Unit Tests

  Test einer einzelnen Unit (Java-Class)
- Intergration Tests

  Test von Dependencies (z.B. Datenbanken)
- Service Tests

  Blackbox Test eines einzelnen Service
- E2E Tests

  Blackbox Test des gesamten Systems
- Production Tests

  Tests während des produktiven Betriebs
- Manual Tests

  Manuelle oder explorative Tests durch einen Anwender

Infrastructure Testing

- Chaos Engineering

  Simulation realer Ausfälle oder Probleme
- Netflix Chaos Monkey

  Abschalten einzelner EC2 Instances in AWS
- Netflix Latency Monkey

  Latency Spikes und Verbindungsabbrüche
- Facebook Storm

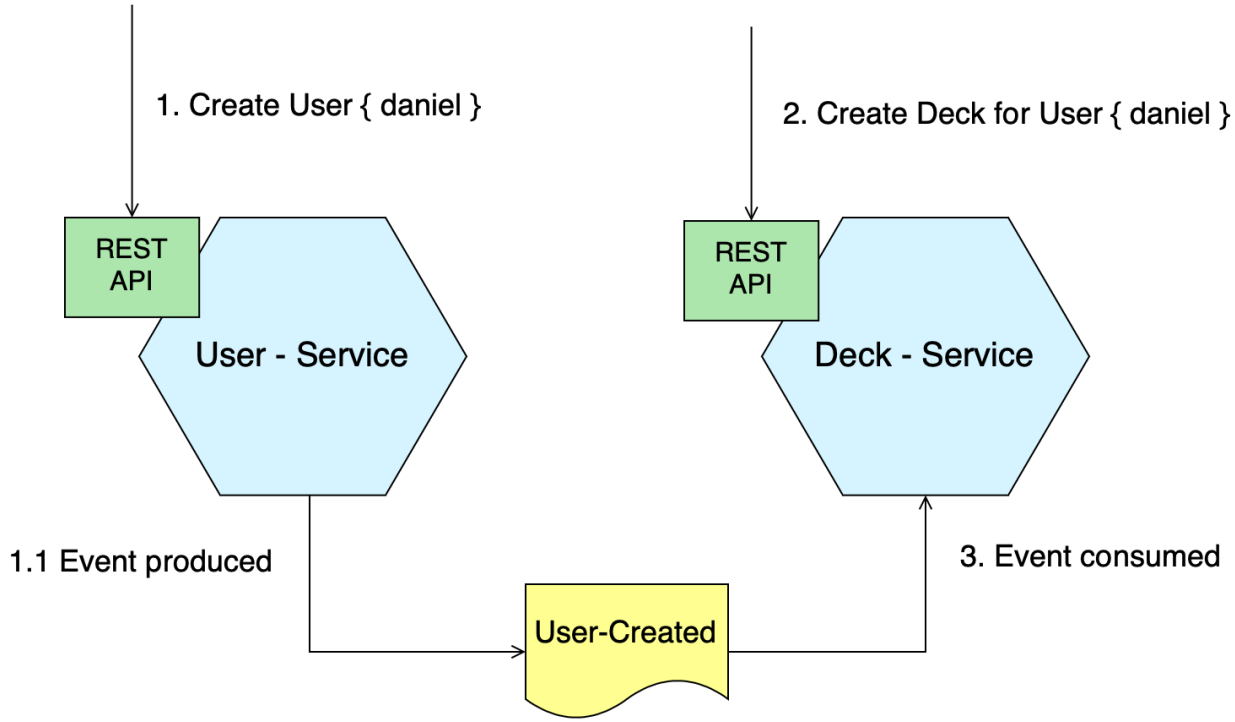
  Ausfall ganzer Data Centers
- AWS Fault Injection Simulator (FIS)

  Service für Chaos Engineering in AWS

Testing

Nachteil

- Man kann nur Szenarien testen, denen man sich im Vorfeld bewusst ist – Bsp. Race Conditions



Abfolge

1. **User Service:** Erstellen eines neuen Users

   1.1 Löst ein User-Created Event aus

2. **Deck Service:** Erstellen eines neuen Decks

   Führt zum Fehler: User not found

3. **Deck Service:** Einlesen des Events

Technology
Arts Sciences
TH Köln

IDE-Debugger

Debugger

- Lokal oder Remote

  In bspw. Intellij oder als Remote Debugger

- Profiler

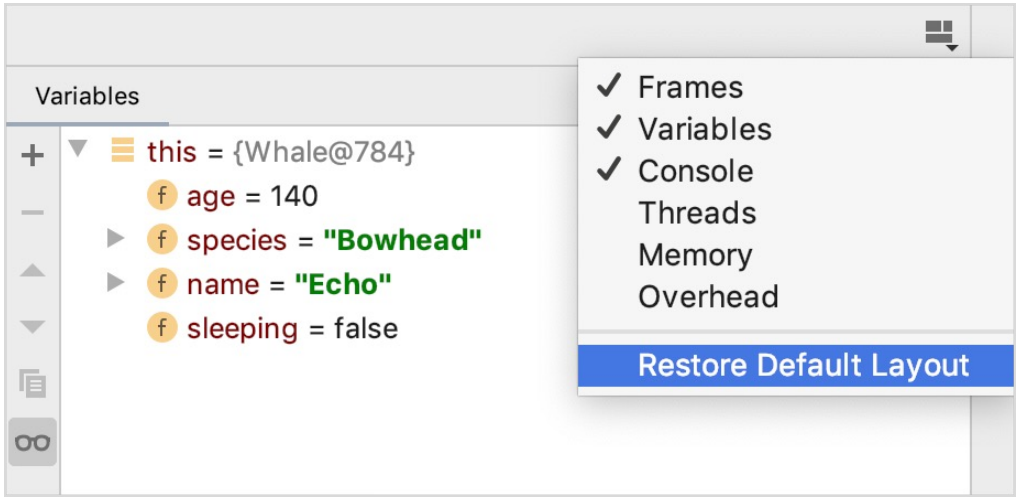
  Auslesen des Memory Heaps

Nachteile

- Setzt testbare Zustände voraus

  Kann eine Live Umgebung oder ein Test sein

```
11
13
14      void print() {
15          System.out.println("Printing to console");
16      }
17  }
```



https://www.jetbrains.com/help/idea/debugging-code.html

Logs

collab_service	Publishing 'clone-card' to 'cmd.decks-cards.0' kafka-topic. CloneCard{payloadDto=CloneCardDto[referencedCardId=414ce74e-2da9-46d8-a22b-79fe5b096b54, targetDeckId=11f70a85-2a12-4516-b4c4-4169188a6bed], AbstractProducerEvent{eventId=59122be4-39eb-48b5-bdf0-82db1a9931bd, transactionId='99785cd835b5f742', correlationId=2dad18e8-0db1-4234-ab81-7a77a01fb495, eventName='clone-card', topic='cmd.decks-cards.0', occurredAt=EventDateTime{dateTime=2022-08-27T19:07:56Z}}}
collab_service	New CollaborationCard created with 1 pending Correlations.
collab_service	Received 'CardCreatedEvent' CardCreated{payload=CardCreatedDto[cardId=414ce74e-2da9-46d8-a22b-79fe5b096b54, deckId=f2c7cdcd-0722-410d-b48d-985cad615dc1, userId=6378c08d-f786-48d8-a49f-504127a50a3e], AbstractConsumerEvent{eventId=63a5e5ab-27a9-47bf-a2f6-00b62419b2e4, transactionId='99785cd835b5f742', correlationId=null, eventName='card-created', occurredAt=2022-08-27T19:07:56Z, receivedAt=2022-08-27T19:07:56Z, topic='cdc.decks-cards.0'}}.
deck_service	Request successful. Responding w/ 201.
deck_service	Publishing 'card-created'-event to 'cdc.decks-cards.0' kafka-topic. CardCreated{payloadDto=CardCreatedDto[cardId=414ce74e-2da9-46d8-a22b-79fe5b096b54, deckId=f2c7cdcd-0722-410d-b48d-985cad615dc1, userId=6378c08d-f786-48d8-a49f-504127a50a3e], AbstractProducerEvent{eventId=63a5e5ab-27a9-47bf-a2f6-00b62419b2e4, transactionId='99785cd835b5f742', correlationId=null, eventName='card-created', topic='cdc.decks-cards.0', occurredAt=2022-08-27T19:07:56Z}}
deck_service	New Default-Card successfully created for 'ruoksnmxpfrj' in Deck 'rzaoqejc'.
deck_service	POST /cards: Create new Card... CardRequestDto[deckId=f2c7cdcd-0722-410d-b48d-985cad615dc1, cardType=default, hint=null, frontView=null, backView=null]

Logs

Levels

- Trace

  Tracing durch Program-Instructions
- Debug

  Ergänzende Debug Ausgaben
- Info

  Prod. Level für wichtige Ereignisse
- Warning

  Kritische Situationen, von denen die Anwendung recovern kann (z.B. Fallback)
- Error

  Kritische Fehler die zum Absturz führen

Schwierigkeiten

- Verteilte Logs

  Zusammenhänge zwischen Services gehen verloren. Jeder Service besitzt eine Vielzahl von Instanzen.
- Multi Threading

  Kann zur Unlesbarkeit führen, wenn mehrere Prozesse eine Datei beschreiben.

- Was sollte man loggen?

Wann logge ich was?

Batch-Processing

Eine periodische stufenweise Verarbeitung großer Datenmengen – Bsp. TV Daten.

- Kontext / State sind die Input-Daten

  Solange man diese mit einer entsprechenden Fehlermeldung hat, können Fehler lokal reproduziert werden.
- Warnings

  Wenn Daten sich nicht verarbeiten lassen oder nicht dem erwarteten Format entsprechen.
- Exceptions

Realtime Event-Driven Applications

Parallele Prozesse mit einem sich laufend ändernden State in Realtime.

- Kontext / State sind Prozess-Trigger (Events) und der Zustand der Domain (Datenbank)

  Wie sah der exakte Zustand zum Zeitpunkt eines Fehlers aus?

Auditing

Bspw. Trailing von Usern (AWS CloudTrail).

Distributed Tracing

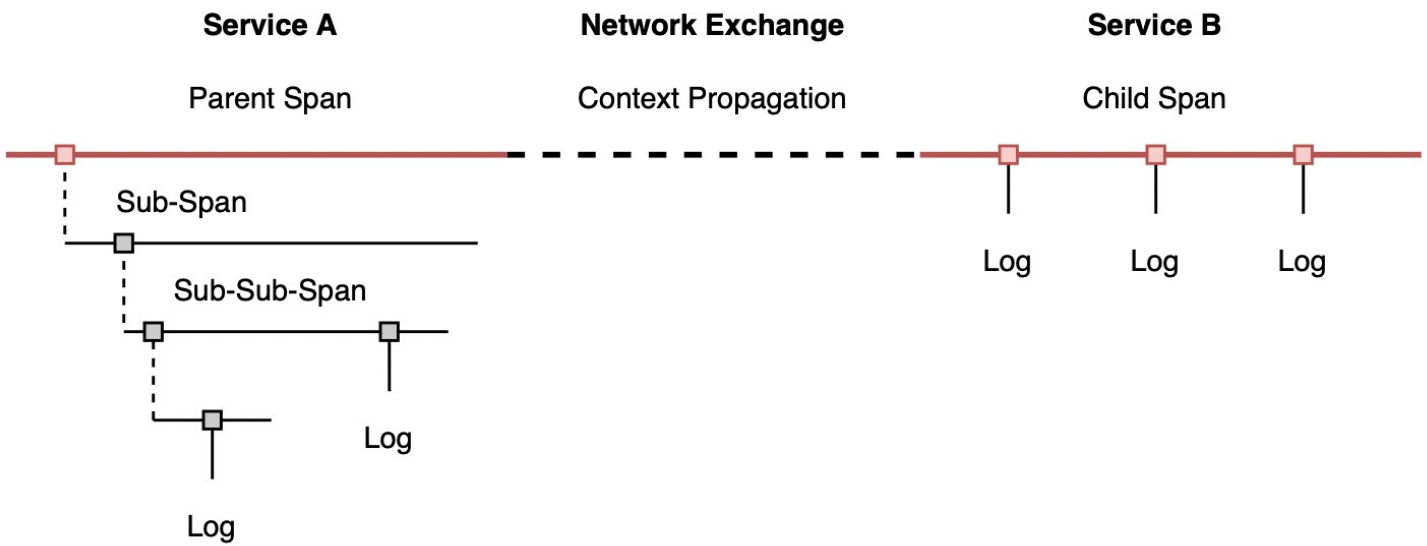
Context Propagation

- Trace-ID

  Eine einheitliche ID für fachliche Zusammenhänge

- Span-ID

  Eine eigene ID für einzelne Abschnitte



Instrumentation

- Blackbox

  Äußerliches Tracing über Endpoints

- Whitebox

  Internes Tracing eines Service

Technology
Arts Sciences
TH Köln

Distributed Tracing

Anwendungsfälle

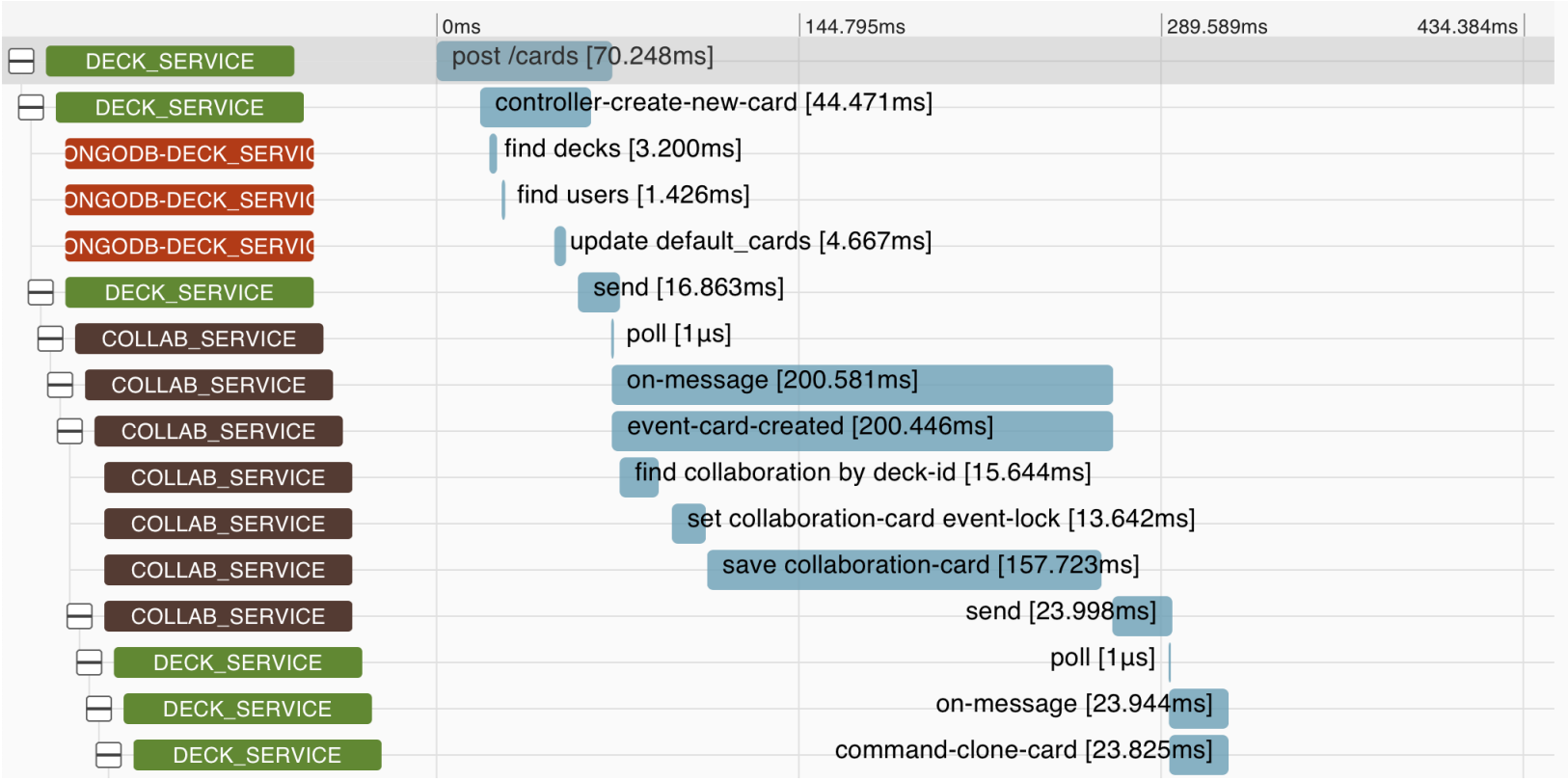
- Debugging

  Fachliche Zusammenhänge oder Performance Bottlenecks

- Monitoring

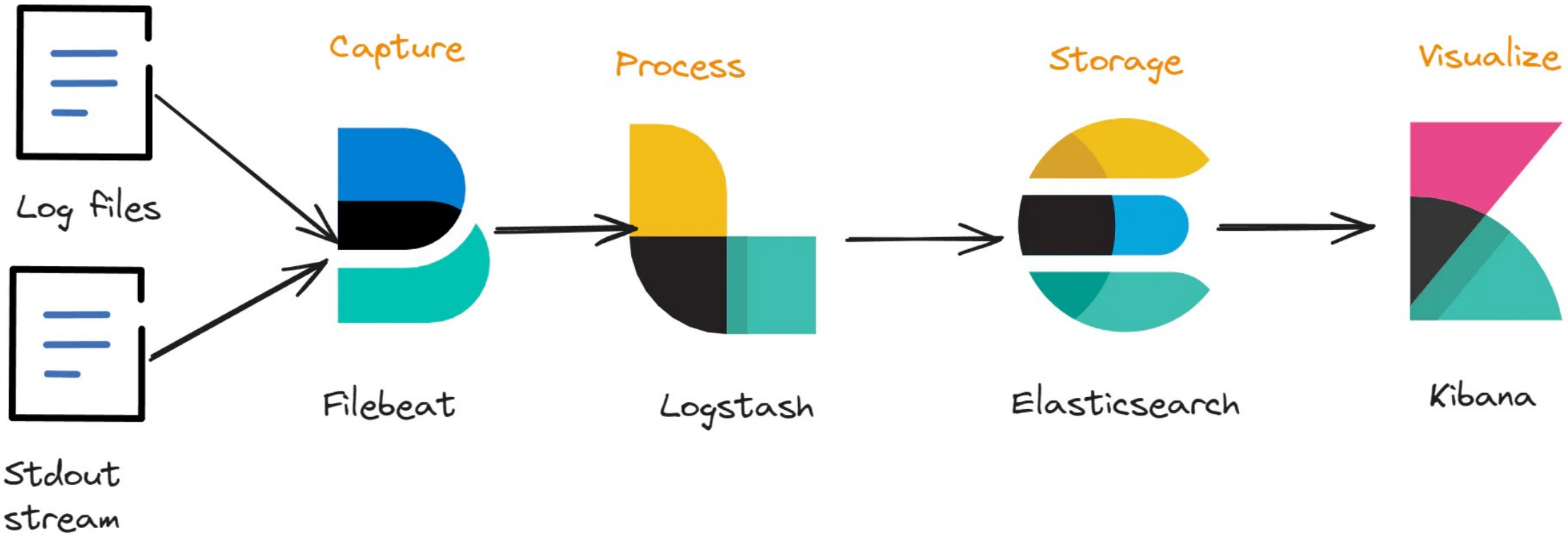
  Response Time oder Ausfälle

- Ergänzt Logs um eine Trace-ID und Span-ID



Technology
Arts Sciences
TH Köln

ELK-Stack

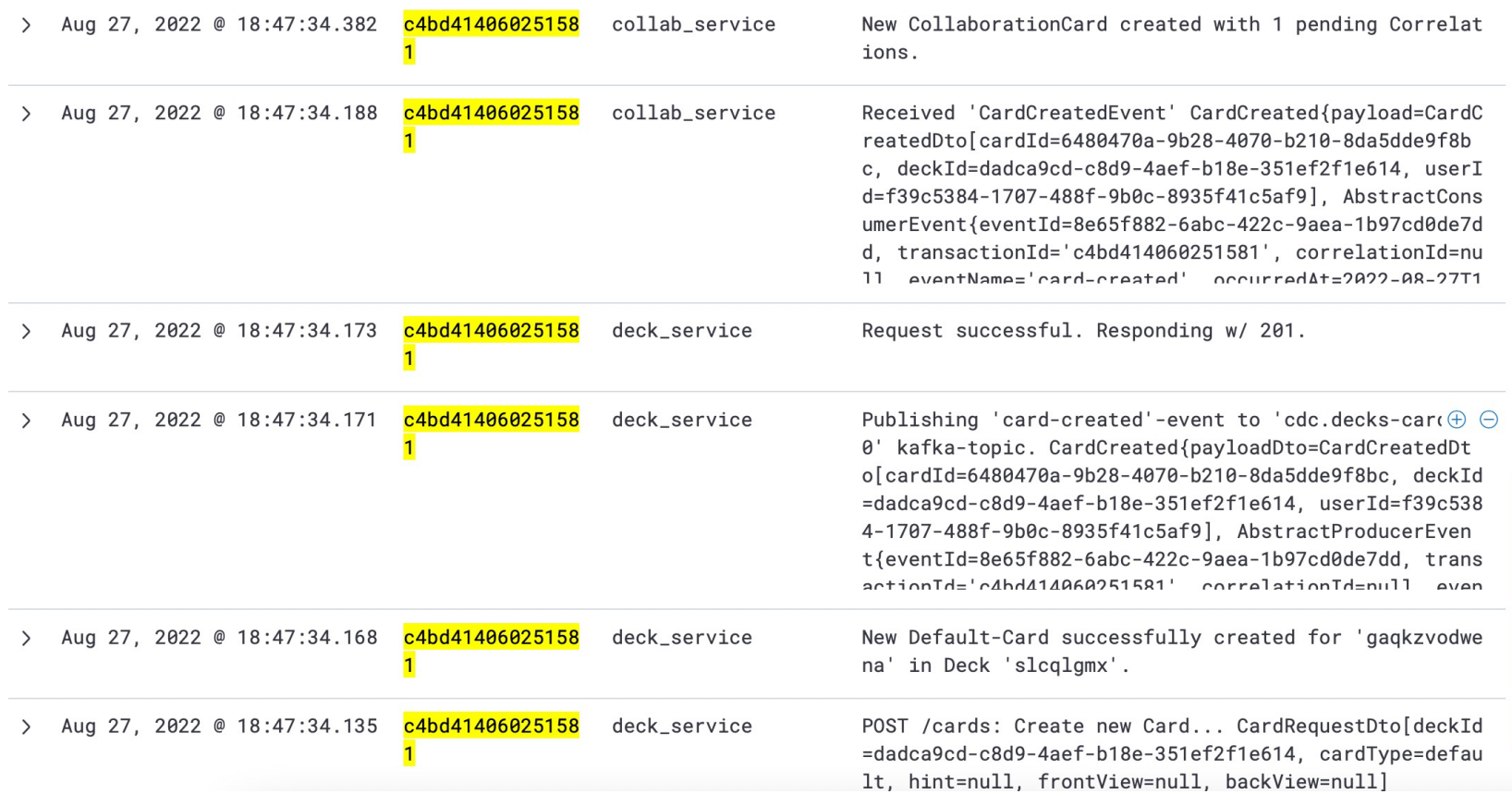

Capture
Process
Storage
Visualize
Log files
Stdout stream
Filebeat
Logstash
Elasticsearch
Kibana

Log Aggregation w/ Tracing

ELK-Stack

- Zentrale Auswertung und Analyse verteilter Logs anhand von IDs

  Ermöglicht das Nachvollziehen von Problemen oder das Einrichten von Monitorings und Alarms über die Frequenz spezieller Messages oder Keywords



Technology
Arts Sciences
TH Köln

Empfehlung
für die Player Entwicklung

Technology
Arts Sciences
TH Köln

Empfehlung - Testing

Was sollte ich testen?

- *Kosten-Nutzen optimiert die wichtigsten Bestandteile, weil Regression-Tests aufgrund der Team- und Anwendungsgröße eine untergeordnete Rolle spielen.*
- *Auf den Happy-Path konzentrieren!*

- Unit Tests komplexer Bereiche

  Command- und Path Algorithms, komplexe Services
- Database Interactions

  Smoke Test und komplexe Custom Queries – Achtung, es macht keinen Sinn Spring generierte Queries umfangreich zu testen!
- Event Consumer

  Smoke Test zum Einlesen von Events – auch hier: keine Libraries testen!
- Web Clients

  Commands mit Mock-Endpoints testen

Empfehlung - Testing

Service- / Smoke Test

- *Grundlegende Funktionen eines Happy Path über Schnittstellen des Service (Events und REST) testen.*

- Ablauf
  - *Zustände in den jeweiligen Stages als Assertions auswerten.*

    - Player anlegen
    - Game beitreten
    - Commandos abschicken
    - Events empfangen und auswerten
    - Commandos abschicken
    - Events empfangen

Empfehlung - Logging

Was sollte ich loggen?

- *Alles, um ein Problem rekonstruieren zu können - während des laufenden Betriebs.*

- Endpoints (Info)

  Events und Commands mit Payload in und out
- Domain Events (Info)

  Erstellen von Commands, Games oder Robots
- Ungewollte Situationen, die nicht zum Absturz führen (Warn)

  Abgelehnte Commands wegen zu langsamem Player
- Exceptions (Error)

  Explizite Fehlermeldungen mit hilfreichen Informationen – Einfangen des Kontextes

Empfehlung - Logging

Zusätzliches Debug Logging

- *Unterstützte Informationen zum Betrieb der Anwendung, auf die nach erfolgreicher Testphase verzichtet werden kann.*

- Database Interactions

  Laden und Speichern von Objekten
- Complex Computations

  Berechnung des nächsten Commands oder Pathing Algorithms
- Interne Map

  Print der kompletten Map
- Status jedes Robots

  Position, Metriken und nächster Command