Continuous Performance Testing

kommt die App noch mit?

Christian Kühn

Software - Entwickler

Gute-Laune-Verbreiter

Interessen:
Security, DevOops, Automation

HIER BIN ICH MENSCH
HIER KAUF ICH EIN

TECH

dmTECH GmbH

100% dm-Tochter

komplette Konzern-IT
Hardware
Software
Betrieb

Was sind eigentlich Tests?

„Unter Testen versteht man den Prozess des Planens, der Vorbereitung und der Messung, mit dem Ziel, die Eigenschaften eines IT-Systems festzustellen und den Unterschied zwischen dem tatsächlichen und dem erforderlichen Zustand aufzuzeigen.“

- Pol, Koomen und Spillner,

Wozu performance-Tests?

bessere Einschätzung der Lastverteilung im produktiv-Betrieb

"Vorhersehen" wie das (Gesamt-) System auf Besonderheiten reagiert

Ausfallsicherheit schaffen

Wozu performance-Tests?

Schwachstellen bzw. Bottlenecks erkennen

Kosten einsparen

Kundenzufriedenheit erhöhen

Performance messen

Anteil fehlerhafter Requests

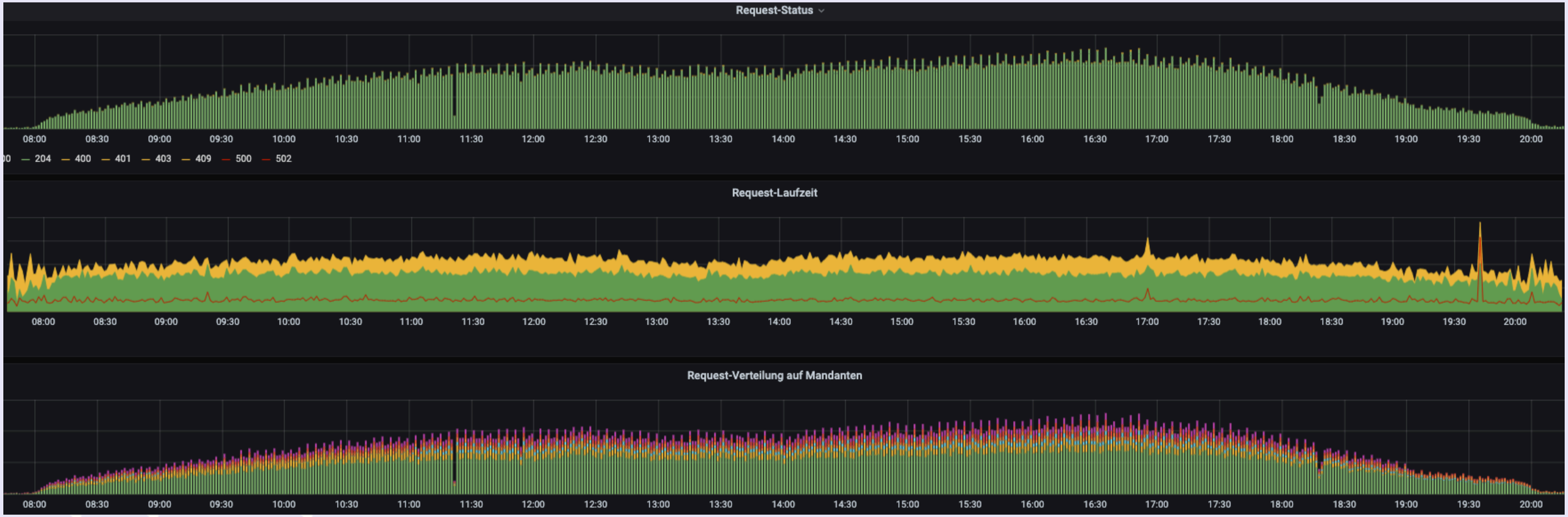
requests werden langsam

app crash

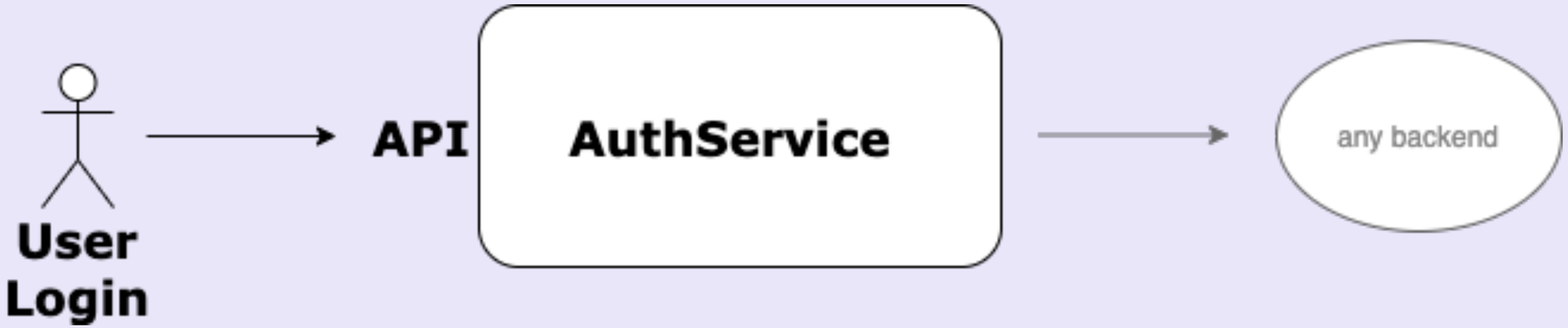
Performance messen

Metriken!!

Metriken - Auswertung - Alerting



Anwendung



load test as code

"Load Test as Code"

Scala - basiert (Akka)

HTTP-Lasttests mit Netty (REST, SOAP)

reporting als Metriken (graphite)

https://gatling.io

Scenario

Beschreibung des Testfalls

```
val successfulAuthentication: ScenarioBuilder = scenario("Successful Authenticate")
        .exec(http("auth")
                .post("/api/authenticate")
                .body(StringBody("""{ "account": "12345", "pass": "hunter2" }""")).asJson
                .check(status.is(200))

val failedAuthentication: ScenarioBuilder = scenario("Failed Authenticate")
        .exec(http("auth")
                .post("/api/authenticate")
                .body(StringBody("""{ "account": "12345", "pass": "passw0rd" }""")).asJson
                .check(status.is(401)))
```

Scenario (2)

Kombination mehrerer Szenarien

```
val authenticateCombined: ScenarioBuilder = scenario("Authenticate combo")
        .randomSwitch(
            85.0 -> exec(successfulAuthentication),
            15.0 -> exec(failedAuthentication))
```

Gatling

dm TECH

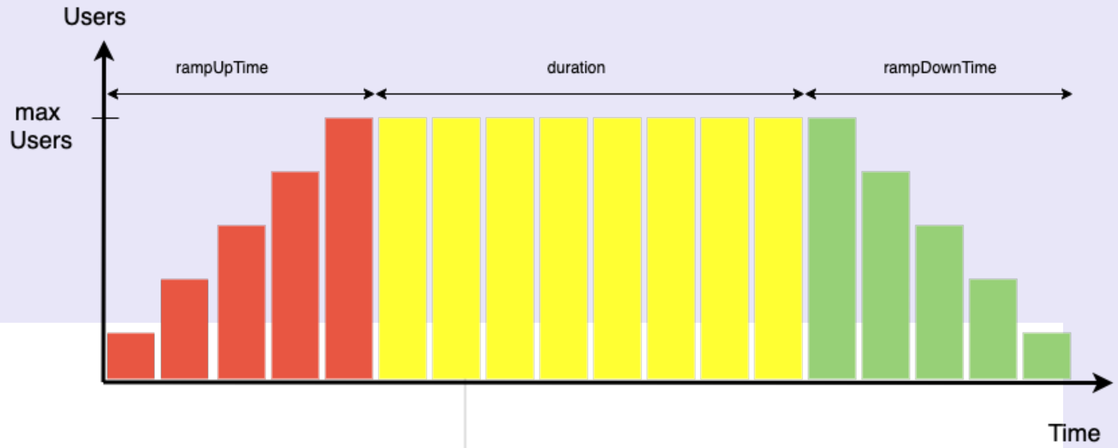
Simulation

Konfiguration des Testfalls

- Anzahl der User / Consumer
- httpConfig
- rampUp / -down
- duration

Simulation (Beispiel)

```
val getHttpConfig: () => HttpProtocolBuilder = () => {
    val httpConfig: HttpProtocolBuilder = http
            .baseUrl("http://localhost:8080")
            .userAgentHeader("Gatling")
            .acceptEncodingHeader("gzip")
}
```




```
setUp(
  authenticateCombined.inject(
    rampUsersPerSec( rate1 = 0) to 30 during (10 seconds), // RAMP UP, 3 additional requests per second)
    constantUsersPerSec( rate = 30) during(30 seconds) randomized, // 20 requests per second for 30 seconds
    rampUsersPerSec( rate1 = 30) to 0 during (10 seconds)) // RAMP Down, 3 less requests per second
).protocols(httpConfig)
  .assertions(
    global.failedRequests.percent.lte( threshold = 0.05), // under 5% wrong answers
    global.responseTime.percentile4.lte( threshold = 1000) // 99% of requests answered quicker than 1000ms
  )
```

Gatling

dm TECH

demo

Gatling

dm TECH

report -> siehe html, grafana

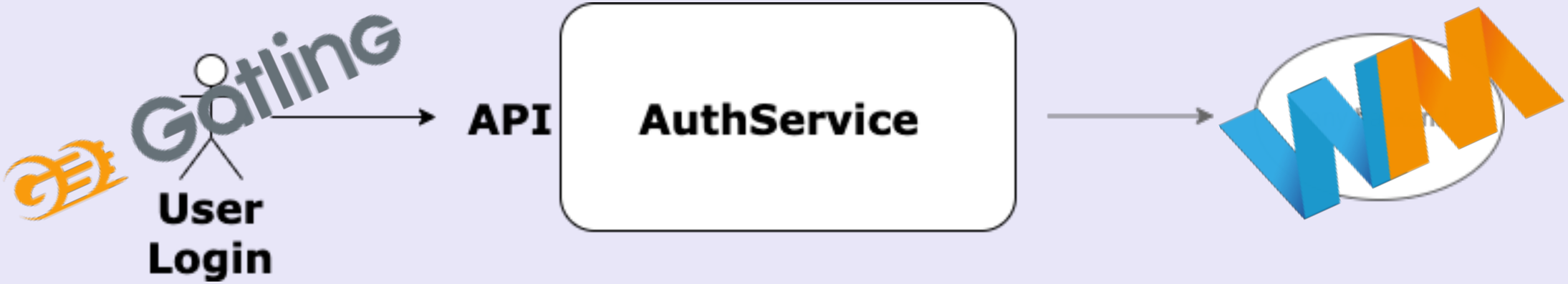
"continuous testing"

Merge-Requests -> immer (?)

"release"-System -> regelmäßig

Prod -> regelmäßig

Anwendung



Wiremock

API - Simulator (HTTP)

vorbereitete Responses
abhängig vom Request

record / playback

JUnit oder Standalone

Http-Server bzw. -Services mocken

Responses abhängig vom Request
z.B. bestimmtes Mapping auf auth-header

Konfiguration in Java oder JSON


```
{
  "request": {
    "url": "/api/authenticate",
    "method": "POST",
    "bodyPatterns": [
      {
        "matchesJsonPath": "$[?(@.pass == 'hunter2')]"
      }
    ]
  },
  "response": {
    "status": 200,
    "body": "{ \"isAuthenticated\": \"true\" }",
    "fixedDelayMilliseconds": 350,
    "headers": {
      "Content-Type": "application/json;charset=UTF-8",
      "Cache-Control": "no-cache, no-store, max-age=0, must-revalidate",
      "Pragma": "no-cache",
      "Expires": "0",
      "Date": "{{now timezone='GMT' format='EEE, d MMM yyyy HH:mm:ss z'}}"
    }
  },
  "metadata": {
    "description": "Nutzer mit richtigem Passwort logt sich ein."
  }
}
```

Consumer testen für Drittanbieter-Services

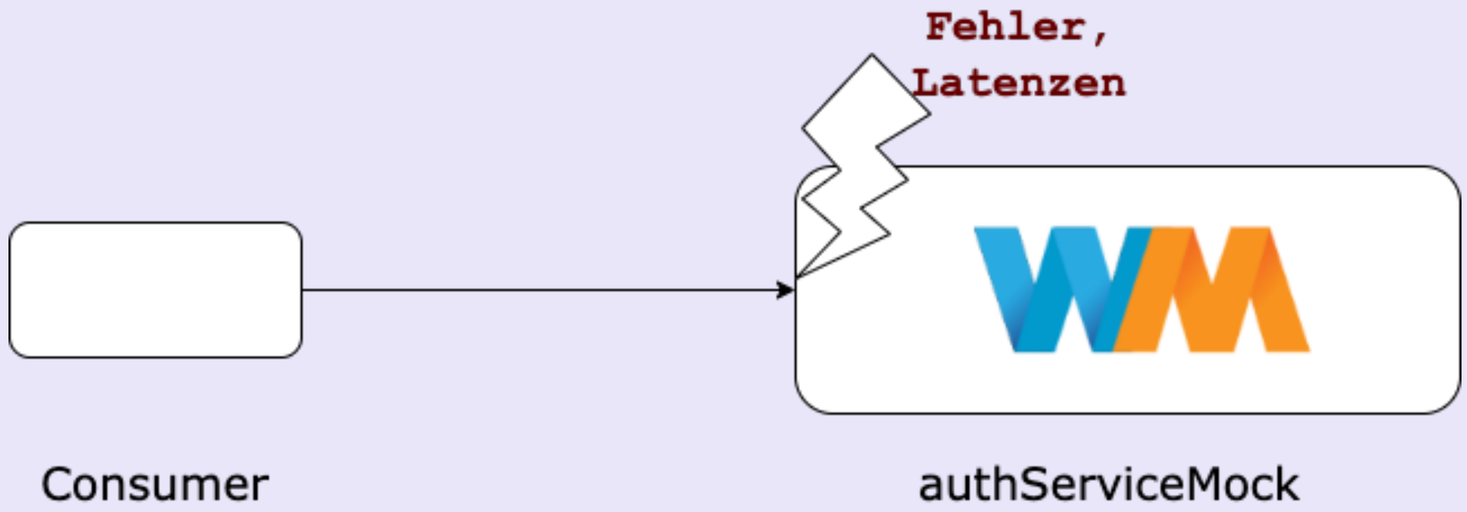
Fehler und Exceptions in Request einbauen

Latenzen simulieren (z.B. für performance-Tests)

verschiedene Zustände pro Scenario möglich

Steuerung und Verifizieren
über "Admin"-API

Sandbox für den eigenen Service anbieten



Danke für euer Interesse! :)

Fragen?

Freue mich über Feedback chris@clowncomputing.de @CYxChris