Testdurchführung

Es sind 4 Durchläufe gemacht worden:

- 5m langer Lauf während der POS Last-Tests (17.07.)
  - 30s RampUp
  - 5m Testlauf
  - 1000 VUs maximal
  - 300 Requests / Sekunde
- 10m langer Lauf auf besserer Hardware für Lastagenten, ohne POS (19.07.)
  - 30s RampUp
  - 10m Testlauf
  - 1000 VUs maximal
  - 300 Requests / Sekunde
- 10m langer Lauf auf besserer Hardware für Lastagenten, reduzierte Last, ohne POS (19.07.)
  - 30s RampUp
  - 10m Testlauf
  - 100 VUs maximal
  - 50 Requests / Sekunde
- 10m langer Lauf auf besserer Hardware für Lastagenten, reduzierte Last, ohne POS, auf TEST (19.07.)
  - 30s RampUp
  - 10m Testlauf
  - 100 VUs maximal
  - 100 Requests / Sekunde
  - auf TEST

Dabei sind alle Tests auf einem GitHub Runner gelaufen, also direkt innerhalb der Infrastruktur der Loyalty Platform.

Bei jedem Test wurden zusätzlich zum K6-Summary-Report auch CSV Dateien erstellt, in denen die relevantesten Daten zur weiteren Auswertung vorhanden sind.

Testergebnisse

Es gibt 4 Testergebnisse

1. Testlauf

- mit gleichzeitig laufenden POS Schnittstellen-Tests

Leider ist der CouponActivation Lasttest mehrfach wegen `Out Of Memory` fehlgeschlagen. Dem Lastagenten standen nur 1GB zur Verfügung. Das war in vorhergehenden Tests nicht aufgefallen. Die

benötigten Resourcen waren ebenso unbekannt.

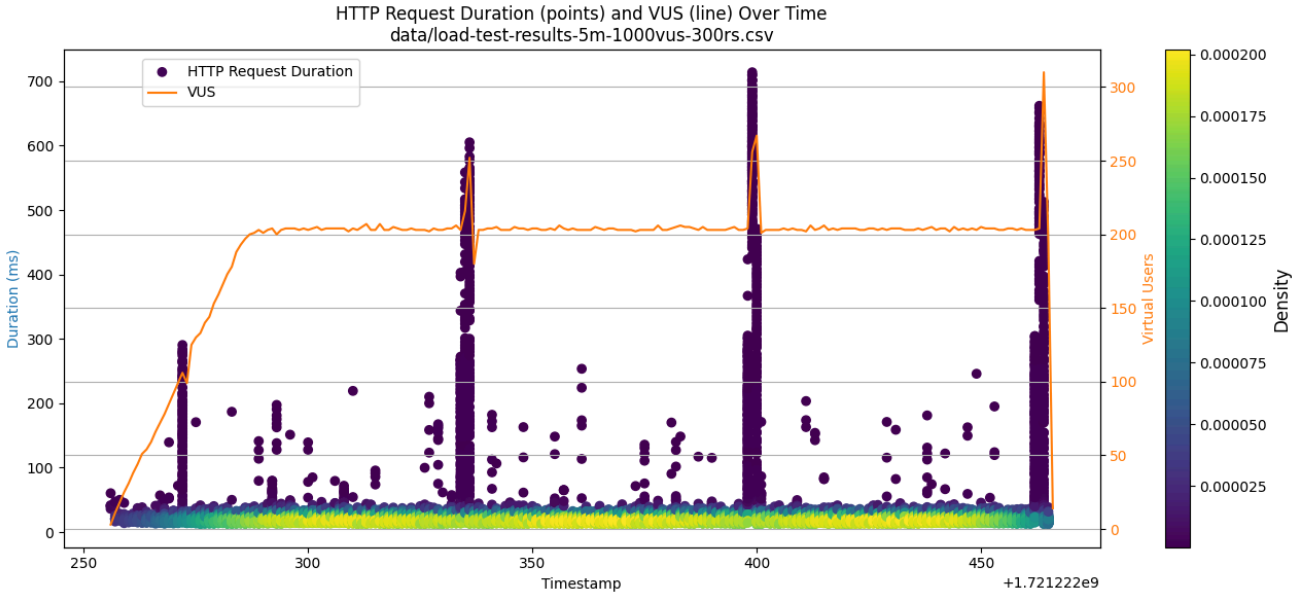
Der Test lief ohne Unterbrechung für ca. 5 Minuten und konnte zumindest kleine Einblicke in das erhalten des Services erlauben. Kernpunkt war hier auf jeden Fall das neue Sizing des Lastagenten.

Ansonsten meldete der K6-Summary-Report folgende Werte:

```
checks.........................: 0.00% ✓ 0         ✗ 0
736     data_received..................: 22 MB 104 kB/s
737     data_sent......................: 53 MB 253 kB/s
738     dropped_iterations.............: 284   1.345747/s
739     http_req_blocked...............: avg=169.72µs min=161ns   med=270ns
max=183.38ms p(90)=361ns   p(95)=400ns
740     http_req_connecting............: avg=17.91µs  min=0s      med=0s
max=11.05ms  p(90)=0s      p(95)=0s
741   ✓ http_req_duration..............: avg=27.04ms  min=11.76ms med=17.91ms
max=713.93ms p(90)=24.36ms p(95)=32.18ms
742       { expected_response:true }...: avg=27.04ms  min=11.76ms med=17.91ms
max=713.93ms p(90)=24.36ms p(95)=32.18ms
743     http_req_failed................: 0.00% ✓ 0         ✗ 38730
744     http_req_receiving.............: avg=53.15µs  min=9.86µs  med=39.15µs
max=147.05ms p(90)=66.64µs p(95)=83.93µs
745     http_req_sending...............: avg=58.12µs  min=24.7µs  med=54.53µs
max=4.53ms   p(90)=72.09µs p(95)=82.33µs
746     http_req_tls_handshaking.......: avg=150.43µs min=0s      med=0s
max=181.43ms p(90)=0s      p(95)=0s
747     http_req_waiting...............: avg=26.93ms  min=11.68ms med=17.81ms
max=713.85ms p(90)=24.24ms p(95)=32.07ms
748     http_reqs......................: 38730 183.52386/s
749     iteration_duration.............: avg=1.02s    min=4.62µs  med=1.01s
max=1.71s    p(90)=1.02s   p(95)=1.03s
750     iterations.....................: 38730 183.52386/s
751     vus............................: 14    min=3       max=310
752     vus_max........................: 319   min=50      max=319
```

Wichtig zu erwähnen:

- Die maximale Antwortzeit ist über den geforderten 500ms (Großteil ist deutlich darunter, avg = 27ms)
- die Request-Rate konnte gehalten, sogar übertroffen werden
- es gab keine HTTP-Fehler

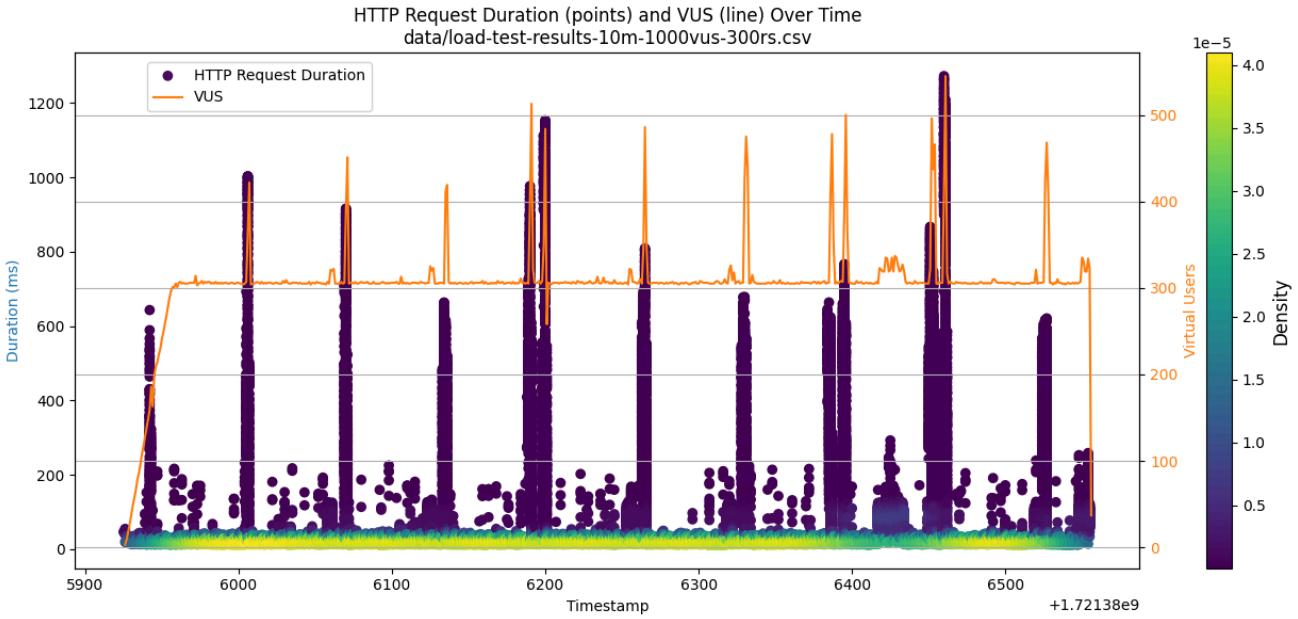


Interessant sind hier die Ausreißer, die wir oben schon in der Summary sehen konnten. Die Ausreißer scheinen sich alle an fixen Zeitpunkten zu sammeln. Der zeitliche Abstand beträgt ungefähr 1 Sekunde, gleichmäßig und kontinuierlich. Hier sollte von den App-Entwicklern tiefer nachgeschaut werden. Dieses Verhalten ist aktuell 100% reproduzierbar, sowohl auf `TEST` als auch auf `PROD`.

2. Testlauf

Das Verhalten ist analog wie zuvor.

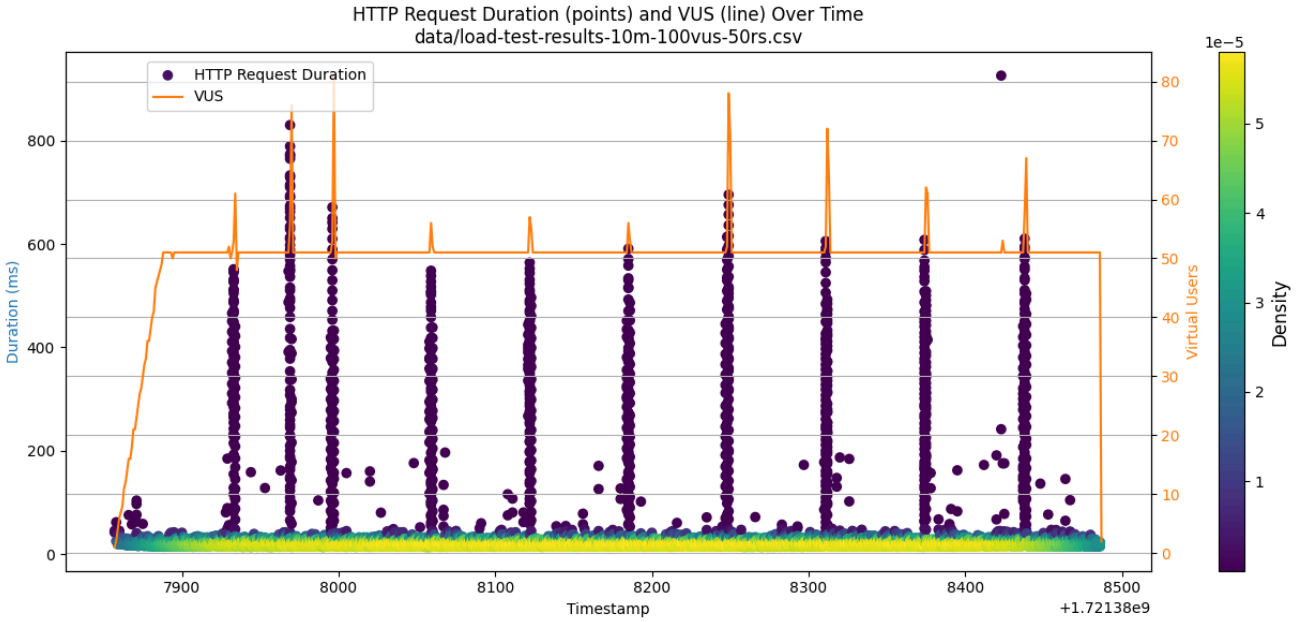
```
checks.........................: 0.00%  ✓ 0          ✗ 0
     data_received..................: 100 MB 159 kB/s
     data_sent......................: 252 MB 399 kB/s
     dropped_iterations.............: 785    1.24387/s
     http_req_blocked...............: avg=70.92µs min=150ns   med=261ns
max=182.61ms p(90)=351ns   p(95)=391ns
     http_req_connecting............: avg=8.14µs  min=0s      med=0s
max=51.66ms  p(90)=0s      p(95)=0s
   ✓ http_req_duration..............: avg=46.07ms min=12.04ms med=18.05ms
max=1.27s    p(90)=74ms    p(95)=158.94ms
       { expected_response:true }...: avg=46.07ms min=12.04ms med=18.05ms
max=1.27s    p(90)=74ms    p(95)=158.94ms
     http_req_failed................: 0.00%  ✓ 0          ✗ 183730
     http_req_receiving.............: avg=54.83µs min=10.47µs med=42.65µs
max=8.58ms   p(90)=75.64µs p(95)=98µs
     http_req_sending...............: avg=61.95µs min=25.63µs med=59.24µs
max=4.96ms   p(90)=75.85µs p(95)=88.33µs
     http_req_tls_handshaking.......: avg=61.98µs min=0s      med=0s
max=178.96ms p(90)=0s      p(95)=0s
     http_req_waiting...............: avg=45.95ms min=11.9ms  med=17.94ms
max=1.27s    p(90)=73.89ms p(95)=158.81ms
     http_reqs......................: 183730 291.128916/s
     iteration_duration.............: avg=1.04s   min=4.16µs  med=1.01s
max=2.27s    p(90)=1.07s   p(95)=1.16s
     iterations.....................: 183730 291.128916/s
     vus............................: 37     min=3        max=545
     vus_max........................: 619    min=3        max=619
```



3. Testlauf

Das Verhalten ist analog wie zuvor.

```
checks.........................: 0.00% ✓ 0        ✗ 0
     data_received..................: 17 MB 26 kB/s
     data_sent......................: 42 MB 67 kB/s
     dropped_iterations.............: 82    0.129948/s
     http_req_blocked...............: avg=56.81µs min=211ns   med=261ns
max=50.77ms  p(90)=431ns   p(95)=531ns
     http_req_connecting............: avg=5.58µs  min=0s      med=0s
max=3.99ms   p(90)=0s      p(95)=0s
   ✓ http_req_duration..............: avg=28.96ms min=13.47ms med=19.02ms
max=925.74ms p(90)=24.84ms p(95)=31.52ms
       { expected_response:true }...: avg=28.96ms min=13.47ms med=19.02ms
max=925.74ms p(90)=24.84ms p(95)=31.52ms
     http_req_failed................: 0.00% ✓ 0        ✗ 30683
     http_req_receiving.............: avg=67.57µs min=12.45µs med=59.88µs
max=9.49ms   p(90)=98.27µs p(95)=113.62µs
     http_req_sending...............: avg=75.3µs  min=33.85µs med=71.12µs
max=5.92ms   p(90)=95.53µs p(95)=104.56µs
     http_req_tls_handshaking.......: avg=50.26µs min=0s      med=0s
max=48.11ms  p(90)=0s      p(95)=0s
     http_req_waiting...............: avg=28.82ms min=13.28ms med=18.87ms
max=925.61ms p(90)=24.7ms  p(95)=31.35ms
     http_reqs......................: 30683 48.62414/s
     iteration_duration.............: avg=1.02s   min=2.88µs  med=1.01s
max=1.92s    p(90)=1.02s   p(95)=1.03s
     iterations.....................: 30683 48.62414/s
     vus............................: 2     min=1      max=81
     vus_max........................: 83    min=1      max=83
```

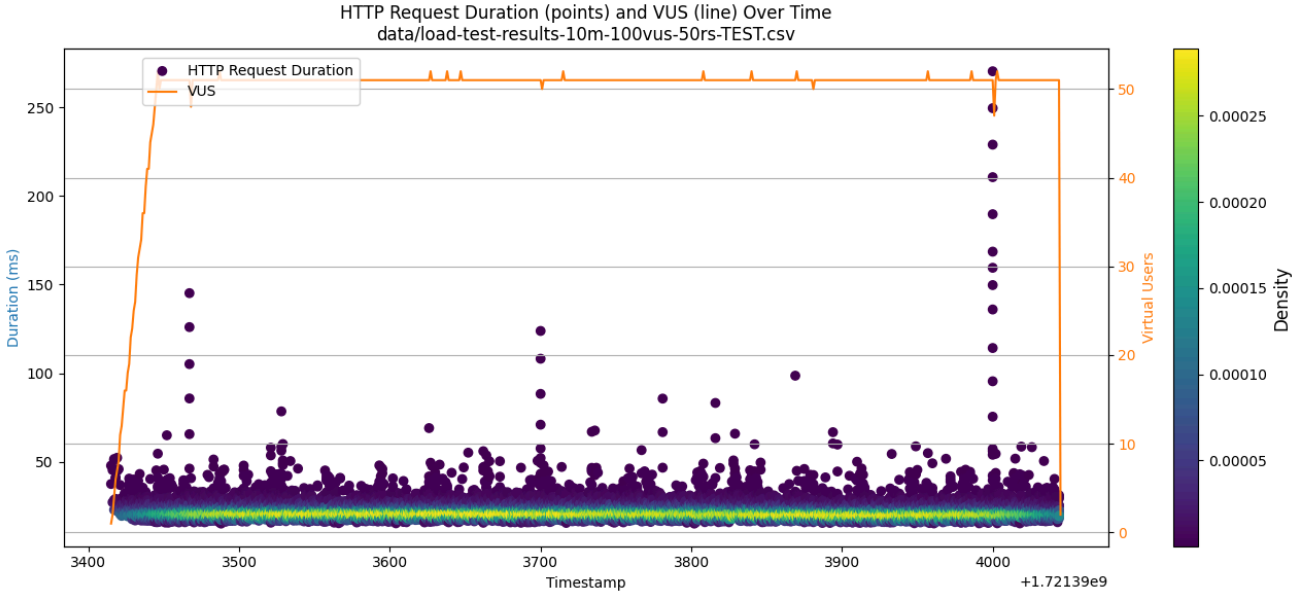


4. Testlauf

Das Verhalten ist sehr ähnlich wie zuvor. Dieses mal wurde die Test-Umgebung unter Last gesetzt, um sie mit den Ergebnissen von PROD zu vergleichen. Die Parameter sind dieselben wie beim **3. Testlauf**. Die Anzahl der Requests / Sekunde ist mit `48.6` identisch. Auch das Muster ist hier erkennbar, aber in deutlich abgeschwächter Form. Das gibt anhaltspunkte auf Unterschiede in der Ausstattung der Umgebungen, möglicherweise lässt sich hier etwas optimieren. Die maximale Antwortzeit ist selbst in den Spitzen nicht über 500ms und erfüllt damit alle Anforderungen.

```
checks.........................: 0.00% ✓ 0        ✗ 0
     data_received..................: 16 MB 26 kB/s
     data_sent......................: 42 MB 67 kB/s
     dropped_iterations.............: 60    0.095083/s
     http_req_blocked...............: avg=45.77µs min=220ns   med=271ns
max=53.7ms   p(90)=441ns    p(95)=531ns
     http_req_connecting............: avg=4.96µs  min=0s      med=0s
max=4.72ms   p(90)=0s       p(95)=0s
   ✓ http_req_duration..............: avg=21.5ms  min=15.07ms med=20.66ms
max=270.45ms p(90)=25.06ms  p(95)=27.57ms
       { expected_response:true }...: avg=21.5ms  min=15.07ms med=20.66ms
max=270.45ms p(90)=25.06ms  p(95)=27.57ms
     http_req_failed................: 0.00% ✓ 0        ✗ 30705
     http_req_receiving.............: avg=67.93µs min=18.21µs med=59.94µs
max=24.53ms  p(90)=96.75µs  p(95)=112.48µs
     http_req_sending...............: avg=76.9µs  min=31.14µs med=72.11µs
max=3.73ms   p(90)=100.07µs p(95)=111.08µs
     http_req_tls_handshaking.......: avg=39.91µs min=0s      med=0s
max=51.5ms   p(90)=0s       p(95)=0s
     http_req_waiting...............: avg=21.35ms min=14.91ms med=20.52ms
max=270.34ms p(90)=24.92ms  p(95)=27.42ms
     http_reqs......................: 30705 48.65881/s
     iteration_duration.............: avg=1.02s   min=4.31µs  med=1.02s
max=1.27s    p(90)=1.02s    p(95)=1.02s
     iterations.....................: 30705 48.65881/s
```

```
vus.............................: 2      min=1      max=52
vus_max.........................: 61     min=1      max=61
```



Allgmeines

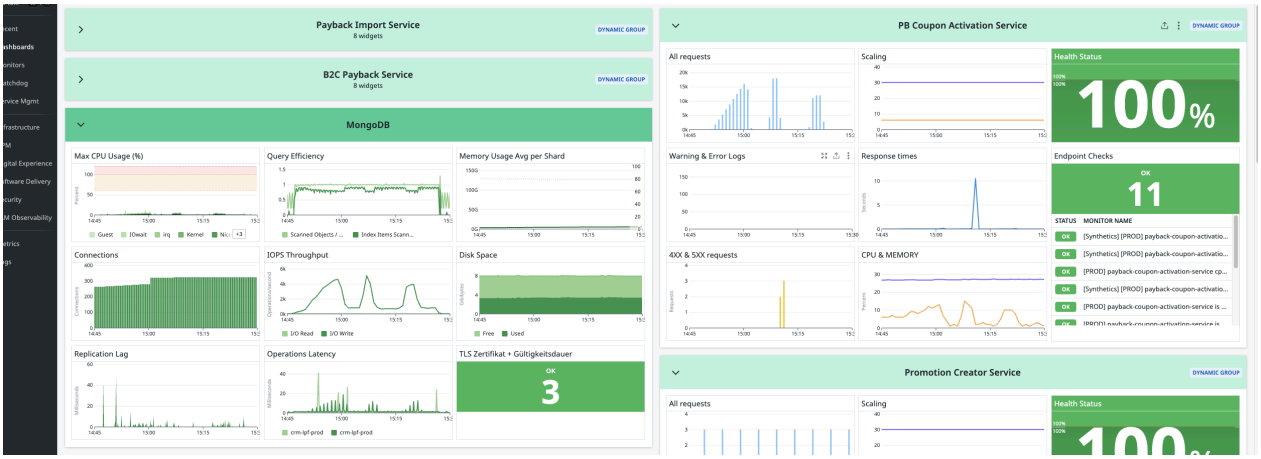
Die gemessenen Metriken aus Datadog können hier nachgesehen werden: https://app.datadoghq.eu/dashboard/mfi-nwu-sus/pacman-dashboard?fromUser=true&refresh_mode=paused&tpl_var_clustername%5B0%5D=crm-lpf-prod&tpl_var_stage%5B0%5D=prod&view=spans&from_ts=1721219850894&to_ts=1721223635448&live
Es gab keinerlei Auffälligkeiten. Die Services waren zu jeder Zeit mit weniger als 20% ausgelastet, die Datenbank war konstant unter 10%.

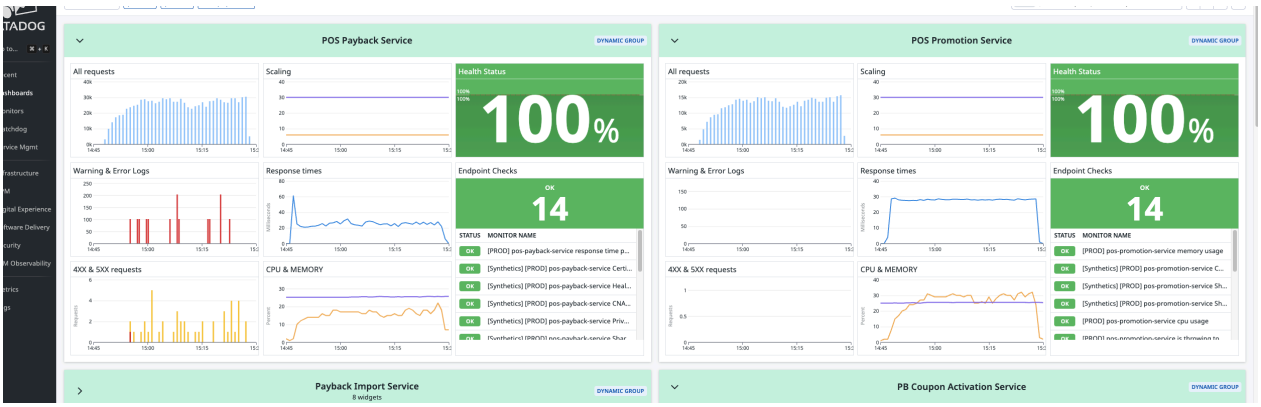
Während des ersten Tests wurde sehr intensiv das Monitoring beobachtet. Es wurden keine Auffälligkeiten entdeckt.

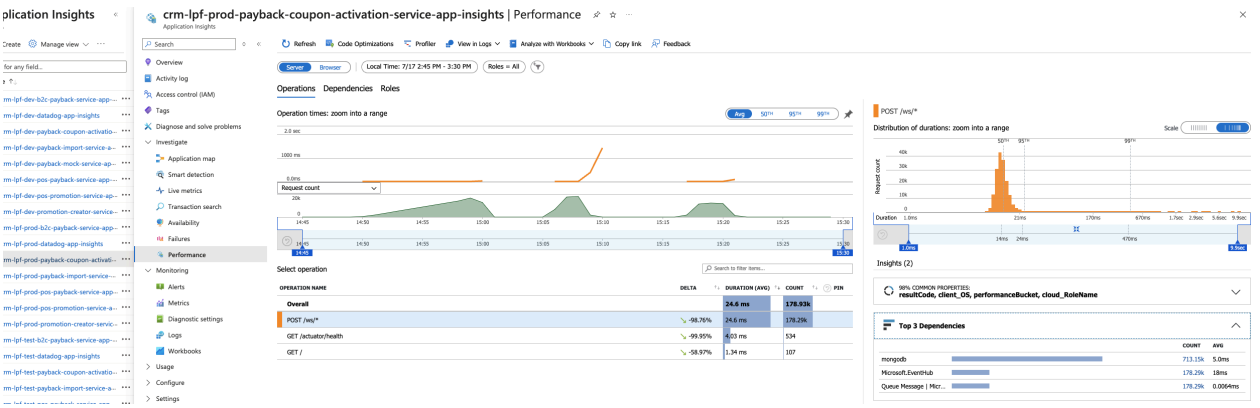
Eine Erkenntnis, die jedoch gemacht werden konnte: Die hohen Antwortzeiten bei den POS Service, aus der Region Minden, konnten in der Loyalty Platform nicht gesehen werden. Auf der anderen Seite sind die hohen Antwortzeiten bei der Coupon Aktivierung aus unserem Netzwerk heraus auch in unseren Metriken zu sehen.

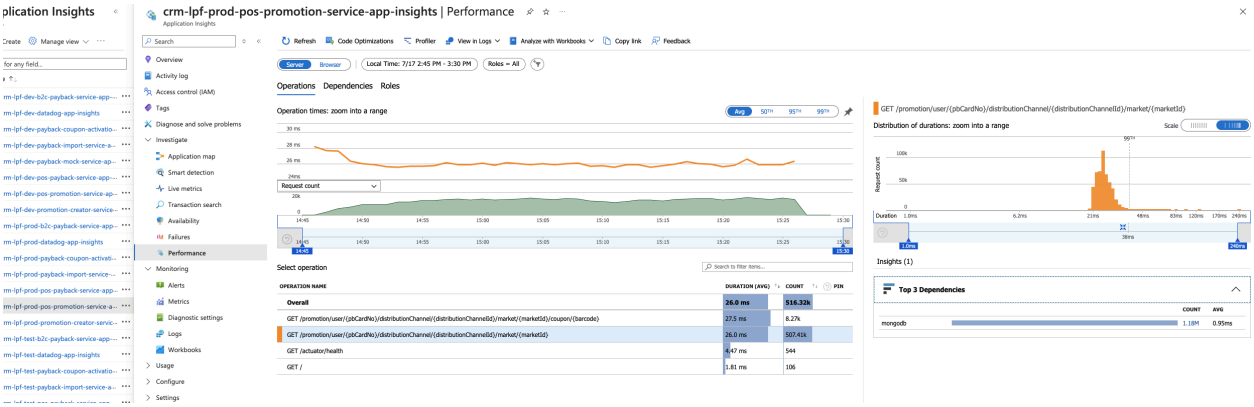
NOTE: Die Timeouts und Ausreißer aus der Region Minden konnten auch bei niedrigen Lasten gesehen werden. Eine Wiederholung am 19.07. mit niedriger Last konnte kein solches Verhalten mehr Feststellen. An diesem Tag gab es einen Crowdstrike Zwischenfall der zum Ausfall vieler PC-Systeme geführt hat. Die Netzwerk-Last könnte als Folge dessen reduziert gewesen sein.

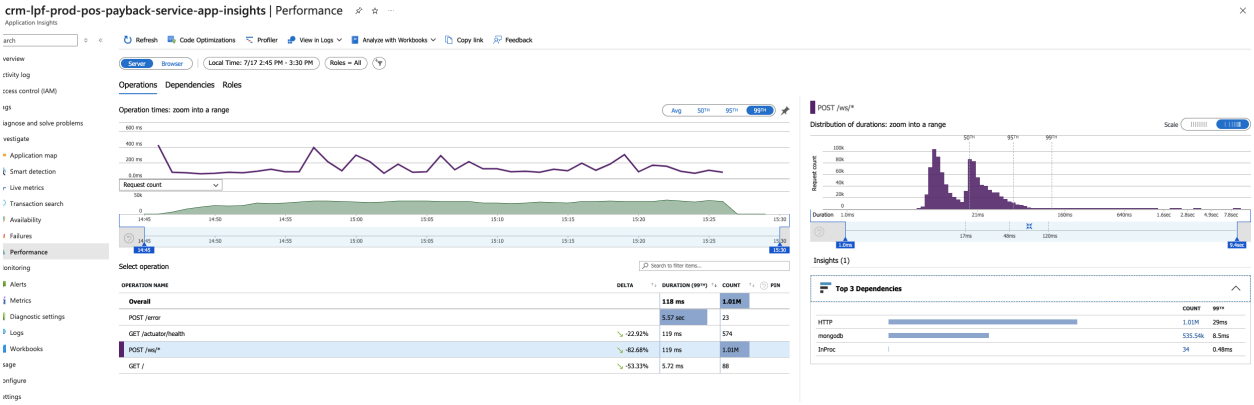
Der Vollständigkeit halber, hier verschiedene Auszüge des Monitorings während des 1. Testlaufs:


Payback Import Service
8 widgets
DYNAMIC GROUP
B2C Payback Service
8 widgets
DYNAMIC GROUP
MongoDB
Max CPU Usage (%)
Query Efficiency
Memory Usage Avg per Shard
Connections
IOPS Throughput
Disk Space
Replication Lag
Operations Latency
TLS Zertifikat + Gültigkeitsdauer
OK
3
PB Coupon Activation Service
DYNAMIC GROUP
All requests
Scaling
Health Status
100%
Warning & Error Logs
Response times
Endpoint Checks
OK
11
4XX & 5XX requests
CPU & MEMORY
Promotion Creator Service
DYNAMIC GROUP
All requests
Scaling
Health Status


POS Payback Service
DYNAMIC GROUP
All requests
Scaling
Health Status
100%
Warning & Error Logs
Response times
Endpoint Checks
OK
14
4XX & 5XX requests
CPU & MEMORY
POS Promotion Service
DYNAMIC GROUP
All requests
Scaling
Health Status
100%
Warning & Error Logs
Response times
Endpoint Checks
OK
14
4XX & 5XX requests
CPU & MEMORY
Payback Import Service
8 widgets
DYNAMIC GROUP
PB Coupon Activation Service
DYNAMIC GROUP


crm-lpf-prod-payback-coupon-activation-service-app-insights | Performance
Application Insights
Refresh
Code Optimizations
Profiler
View in Logs
Analyze with Workbooks
Copy link
Feedback
Overview
Activity log
Access control (IAM)
Tags
Diagnose and solve problems
Investigate
Application map
Smart detection
Live metrics
Transaction search
Availability
Failures
Performance
Monitoring
Alerts
Metrics
Diagnostic settings
Logs
Workbooks
Usage
Configure
Settings
Operations
Dependencies
Roles
Operation times: zoom into a range
Select operation
OPERATION NAME
DELTA
DURATION (AVG)
COUNT
Overall
24.6 ms
178.93k
POST /ws/*
-98.76%
24.6 ms
178.29k
GET /actuator/health
-99.95%
4.03 ms
534
GET /
-58.97%
1.34 ms
107
POST /ws/*
Distribution of durations: zoom into a range
Insights (2)
Top 3 Dependencies
mongodb
Microsoft.EventHub
Queue Message | Micr...


plication Insights
crm-lpf-prod-pos-promotion-service-app-insights | Performance
Application Insights
Refresh
Code Optimizations
Profiler
View in Logs
Analyze with Workbooks
Copy link
Feedback
Overview
Activity log
Access control (IAM)
Tags
Diagnose and solve problems
Investigate
Application map
Smart detection
Live metrics
Transaction search
Availability
Failures
Performance
Monitoring
Alerts
Metrics
Diagnostic settings
Logs
Workbooks
Usage
Configure
Settings
Server
Browser
Local Time: 7/17 2:45 PM - 3:30 PM
Roles = All
Operations
Dependencies
Roles
Operation times: zoom into a range
Avg
50TH
95TH
99TH
Request count
Select operation
OPERATION NAME
DURATION (AVG)
COUNT
P95
Overall
26.0 ms
516.32k
GET /promotion/user/{pbCardNo}/distributionChannel/{distributionChannelId}/market/{marketId}/coupon/{barcode}
27.5 ms
8.27k
GET /promotion/user/{pbCardNo}/distributionChannel/{distributionChannelId}/market/{marketId}
26.0 ms
507.41k
GET /actuator/health
4.47 ms
544
GET /
1.81 ms
106
GET /promotion/user/{pbCardNo}/distributionChannel/{distributionChannelId}/market/{marketId}
Distribution of durations: zoom into a range
Scale
Request count
Duration
Insights (1)
Top 3 Dependencies
COUNT
AVG
mongodb
1.18M
0.95ms


crm-lpf-prod-pos-payback-service-app-insights | Performance
Application Insights
Refresh
Code Optimizations
Profiler
View in Logs
Analyze with Workbooks
Copy link
Feedback
Server
Browser
Local Time: 7/17 2:45 PM - 3:30 PM
Roles = All
Operations
Dependencies
Roles
Operation times: zoom into a range
Avg
50TH
95TH
99TH
Request count
Select operation
OPERATION NAME
DELTA
DURATION (99TH)
COUNT
P95
Overall
118 ms
1.01M
POST /error
5.57 sec
23
GET /actuator/health
-22.92%
119 ms
574
POST /ws/*
-82.68%
119 ms
1.01M
GET /
-53.33%
5.72 ms
88
POST /ws/*
Distribution of durations: zoom into a range
Scale
Request count
Duration
Insights (1)
Top 3 Dependencies
COUNT
99TH
HTTP
1.01M
29ms
mongodb
535.54k
8.5ms
InProc
34
0.48ms