


Om de non functionals te valideren zal gebruikt gaan worden van de tool k6s om microservices te stresstesten. Sinds stresstesten een zwaar proces is, is ervoor gekozen om slechts op 1 microservice tegelijkertijd te stresstesten. In het geval van de kwetter applicatie zal dit zijn de kweetreadservice. In het kader van CQRS is de core functionaliteit opgesplitst in een read en een write microservice. De read microservice zal het gross van de acties voltooien, sinds deze verantwoordelijk is voor het ophalen van de data. In het volgende document zal stap voor stap uitgelegd worden welke testen er zijn uitgevoerd met de bijbehorende resultaten:

Voor de initiële test is het volgende bestand gebruikt:

De test runt:

```
PS C:\Documents\School\Fontys ICT\Semester 6\Advanced software\Individueel\Kwetter-semester-6\K8s\k6s> k6 run LoadTest.js
```



```
.io  
execution: local  
script: LoadTest.js  
output: -  
  
scenarios: (100.00%) 1 scenario, 10 max VUs, 1m30s max duration (incl. graceful stop):  
    * default: Up to 10 looping VUs for 1m0s over 1 stages (gracefulRampDown: 30s, gracefulStop: 30s)  
  
[ ]  
running (0m18.8s), 03/10 VUs, 33 complete and 0 interrupted iterations  
default [=====>-----] 03/10 VUs  0m18.8s/1m00.0s
```

De volgende stap is om de test results online te plaatsen naar k6s cloud:

```
import http from 'k6/http';
import { sleep, check } from 'k6';

export const options = {
  ext: {
    loadimpact: {
      projectID: 3642900,
      // Test runs with the same name groups test runs together
      name: "First cloud based test"
    }
  },
  stages: [
    { duration: '1m', target: 10 }
  ],
};

const API_BASE_URL = ''

export default function () {
  const urlres = http.get('http://localhost:5050/KweetRead/AllKweets')
  sleep(1)
}
```

De output is als volgt en is goed te monitoren:

THRESHOLDS
(0/0)

CHECKS
(0/0)

HTTP
(300/300)

ANALYSIS
Compare metrics

SCRIPT
View executed script

LOGS
Execution logs

FILTERS

Filter expression...

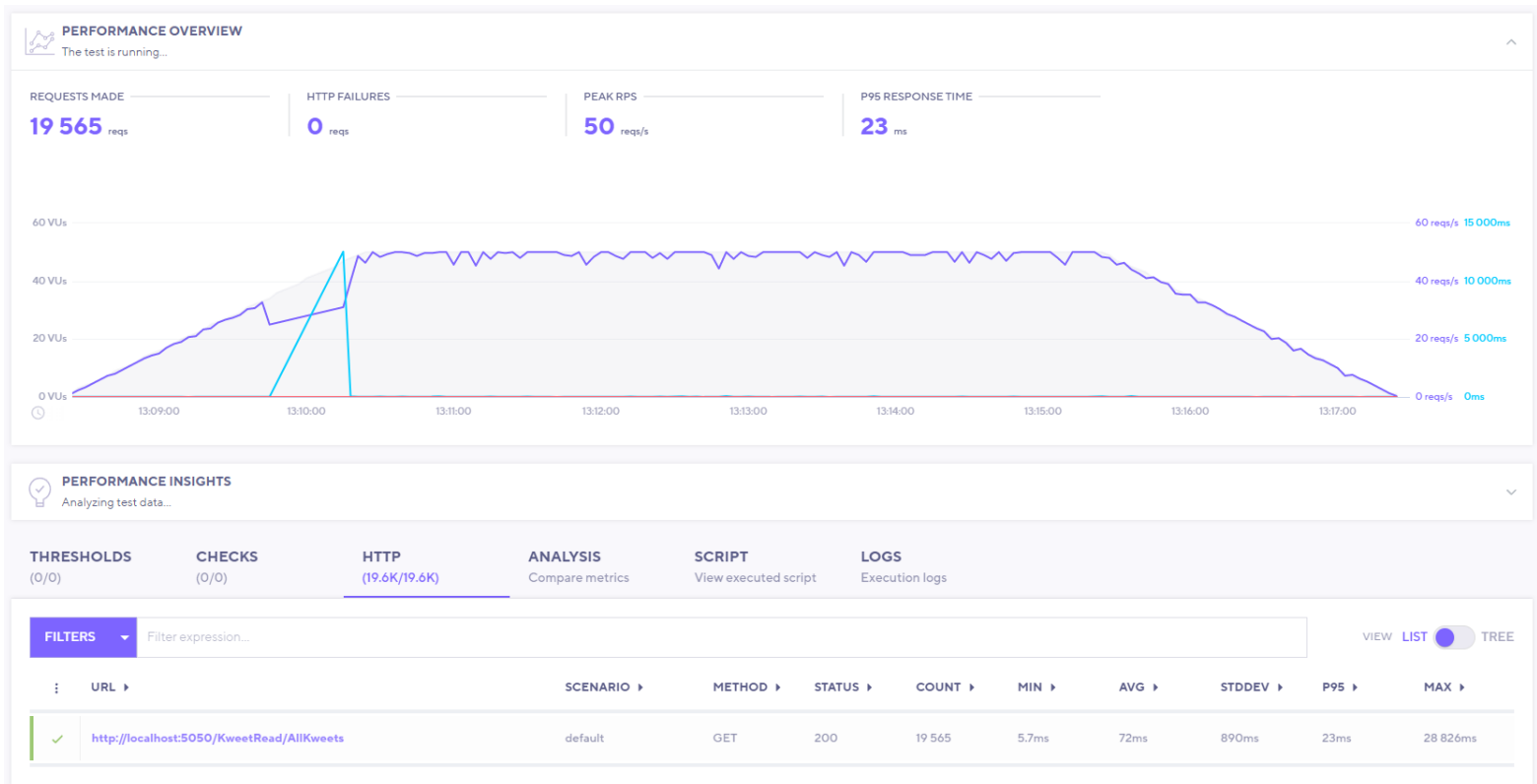
VIEW

LIST

TREE

	URL	SCENARIO	METHOD	STATUS	COUNT	MIN	AVG	STDDEV	P95	MAX
✓	http://localhost:5050/KweetRead/AllKweets	default	GET	200	300	6.1ms	8.9ms	2.5ms	13ms	22ms

Vervolgens gaan we nu dezelfde test doen, alleen dan met 50 gebruikers en over een langere periode:



Naarmate de

De volgende test die uitgevoerd gaat worden is het proberen van het upscalen van pods. Hiervoor is een autoscale aangebracht:

kubectl autoscale deployment kweetreadservice-deployment --cpu-percent=50 --min=1 --max=4

```
PS C:\Users\Nickv> kubectl get hpa
NAME                                REFERENCE                                TARGETS      MINPODS  MAXPODS  REPLICAS  AGE
kweetreadservice-deployment         Deployment/kweetreadservice-deployment   <unknown>/50%  1         4         0         4s
```

```

K6S > JS StressTest.js > default
1  import http from 'k6/http';
2  import { sleep, check } from 'k6';
3
4  export const options = {
5    ext: {
6      loadimpact: {
7        projectID: 3642900,
8        // Test runs with the same name groups test runs together
9        name: "Load balancing Test"
10     }
11   },
12   stages: [
13     { duration: '2m', target: 250},
14     { duration: '6m', target: 250},
15     { duration: '4m', target: 0}
16   ],
17 };
18
19 export default function () {
20
21   http.batch([
22     ['GET', 'http://localhost:5050/KweetRead/AllKweets'],
23     ['GET', 'http://localhost:5050/KweetRead/ReactionsByKweet?kweetId=1476cc1f-bbee-431e-93aa-7d9898aaed78']
24   ])
25   sleep(1)
26 }

```

De eerste test was nadat het ongeveer 250 virtual users bereikte afgekappt door de kweetreadservice pod. Dit heeft te maken met dat de pod out of memory was. In de nieuwe pogingen wordt een request toegevoegd aan de configuratie. Dit zou ervoor moeten zorgen dat vroegtijdig een nieuwe pod horizontaal wordt gescaled.

Test V2

In de nieuwe test is er een geupdate horizontal pod autoscaler toegevoegd, daarnaast is ook de metrics server gedownload. Deze voedt informatie over hoeveel % van de pods cpu en memory wordt gebruikt. Dit is nodig voor de hpa om te bepalen wanneer er een nieuwe pod bij moet komen.

NAME	REFERENCE	TARGETS	MINPODS	MAXPODS	REPLICAS	AGE
hpa-kweetreadservice	Deployment/kweetreadservice-deployment	<unknown>/60%, <unknown>/60%	1	5	0	40s
hpa-kweetreadservice	Deployment/kweetreadservice-deployment	46%/60%, 0%/60%	1	5	1	60s
hpa-kweetreadservice	Deployment/kweetreadservice-deployment	46%/60%, 0%/60%	1	5	1	2m
hpa-kweetreadservice	Deployment/kweetreadservice-deployment	72%/60%, 174%/60%	1	5	1	3m
hpa-kweetreadservice	Deployment/kweetreadservice-deployment	64%/60%, 66%/60%	1	5	3	4m
hpa-kweetreadservice	Deployment/kweetreadservice-deployment	68%/60%, 27%/60%	1	5	4	5m
hpa-kweetreadservice	Deployment/kweetreadservice-deployment	63%/60%, 40%/60%	1	5	5	6m
hpa-kweetreadservice	Deployment/kweetreadservice-deployment	34%/60%, 0%/60%	1	5	5	7m

NAMESPACE	NAME	PF	READY	RESTARTS	STATUS	CPU	MEM	%CPU/R	%CPU/L	%MEM/R	%MEM/L	IP	NODE	AGE
default	kweetreadservice-deployment-7f478f9cf6-crd24	●	1/1	2	Running	1	50	0	0	39	19	10.1.0.183	docker-desktop	52m
default	kweetreadservice-deployment-7f478f9cf6-g59wn	●	1/1	0	Running	2	55	0	0	43	21	10.1.0.193	docker-desktop	10m
default	kweetreadservice-deployment-7f478f9cf6-j4swg	●	1/1	0	Running	2	51	0	0	40	20	10.1.0.191	docker-desktop	11m
default	kweetreadservice-deployment-7f478f9cf6-wd2pg	●	1/1	0	Running	1	52	0	0	41	20	10.1.0.192	docker-desktop	11m

Zoals te zien tijdens de load test, worden de pods automatisch ge upscaled naar maximaal 5 replicas. Deze test ging nog niet helemaal goed, sinds ik deze service moest port forwarden op mijn pc omdat ngingx het niet deed. In de volgende test zal nginx gebruikt gaan worden en zal een volledige loop van de test afgemaakt worden:

```
PS C:\Users\Nickv> kubectl get hpa --watch
```

NAME	REFERENCE	TARGETS	MINPODS	MAXPODS	REPLICAS	AGE
hpa-kweetreadservice	Deployment/kweetreadservice-deployment	60%/70%, 0%/70%	1	5	4	16m
hpa-mongodb	Deployment/mongodb	54%/50%, 2%/50%	1	5	2	4m49s
hpa-kweetreadservice	Deployment/kweetreadservice-deployment	60%/70%, 0%/70%	1	5	4	17m
hpa-mongodb	Deployment/mongodb	54%/50%, 1%/50%	1	5	2	5m
hpa-kweetreadservice	Deployment/kweetreadservice-deployment	59%/70%, 0%/70%	1	5	4	18m
hpa-mongodb	Deployment/mongodb	56%/50%, 1%/50%	1	5	2	6m
hpa-kweetreadservice	Deployment/kweetreadservice-deployment	66%/70%, 19%/70%	1	5	4	19m
hpa-mongodb	Deployment/mongodb	57%/50%, 8%/50%	1	5	3	7m
hpa-kweetreadservice	Deployment/kweetreadservice-deployment	62%/70%, 61%/70%	1	5	4	20m
hpa-mongodb	Deployment/mongodb	57%/50%, 17%/50%	1	5	4	8m
hpa-kweetreadservice	Deployment/kweetreadservice-deployment	63%/70%, 93%/70%	1	5	4	21m
hpa-mongodb	Deployment/mongodb	58%/50%, 18%/50%	1	5	5	9m
hpa-kweetreadservice	Deployment/kweetreadservice-deployment	63%/70%, 87%/70%	1	5	5	22m
hpa-mongodb	Deployment/mongodb	59%/50%, 18%/50%	1	5	5	10m
hpa-kweetreadservice	Deployment/kweetreadservice-deployment	62%/70%, 80%/70%	1	5	5	23m
hpa-mongodb	Deployment/mongodb	60%/50%, 19%/50%	1	5	5	11m
hpa-kweetreadservice	Deployment/kweetreadservice-deployment	62%/70%, 80%/70%	1	5	5	24m
hpa-mongodb	Deployment/mongodb	61%/50%, 18%/50%	1	5	5	12m
hpa-kweetreadservice	Deployment/kweetreadservice-deployment	62%/70%, 72%/70%	1	5	5	25m
hpa-mongodb	Deployment/mongodb	62%/50%, 16%/50%	1	5	5	13m
hpa-kweetreadservice	Deployment/kweetreadservice-deployment	62%/70%, 48%/70%	1	5	5	26m
hpa-mongodb	Deployment/mongodb	62%/50%, 11%/50%	1	5	5	14m
hpa-kweetreadservice	Deployment/kweetreadservice-deployment	62%/70%, 30%/70%	1	5	5	27m
hpa-mongodb	Deployment/mongodb	63%/50%, 7%/50%	1	5	5	15m
hpa-kweetreadservice	Deployment/kweetreadservice-deployment	63%/70%, 13%/70%	1	5	5	28m
hpa-mongodb	Deployment/mongodb	64%/50%, 3%/50%	1	5	5	16m
hpa-kweetreadservice	Deployment/kweetreadservice-deployment	62%/70%, 0%/70%	1	5	5	29m
hpa-mongodb	Deployment/mongodb	64%/50%, 2%/50%	1	5	5	17m

NAMESPACE↑	NAME	PF	READY	RESTARTS	STATUS	IP	NODE	AGE
default	kweetreadservice-deployment-7f478f9cf6-5755n	•	1/1	0	Running	10.1.1.233	docker-desktop	21m
default	kweetreadservice-deployment-7f478f9cf6-dr8kb	•	1/1	0	Running	10.1.1.227	docker-desktop	39m
default	kweetreadservice-deployment-7f478f9cf6-f7djs	•	1/1	0	Running	10.1.1.232	docker-desktop	24m
default	kweetreadservice-deployment-7f478f9cf6-fwgbd	•	1/1	0	Running	10.1.1.234	docker-desktop	18m
default	kweetreadservice-deployment-7f478f9cf6-tfzdd	•	1/1	0	Running	10.1.1.240	docker-desktop	8m55s
default	mongodb-844bb6f844-4sn6m	•	1/1	0	Running	10.1.1.236	docker-desktop	14m
default	mongodb-844bb6f844-n7ldl	•	1/1	0	Running	10.1.1.238	docker-desktop	10m
default	mongodb-844bb6f844-nmm7d	•	1/1	0	Running	10.1.1.235	docker-desktop	15m
default	mongodb-844bb6f844-sxg64	•	1/1	0	Running	10.1.1.237	docker-desktop	11m
default	mongodb-844bb6f844-x2bk2	•	1/1	0	Running	10.1.1.239	docker-desktop	9m52s

PS C:\Users\Wickv\Documents\School\Semester 6\individueel\Kwetter-semester-6\K8s\k6s> k6 run StressTest.js



execution: local
script: StressTest.js
output: -

scenarios: (100.00%) 1 scenario, 300 max VUs, 10m30s max duration (incl. graceful stop):
* default: Up to 300 looping VUs for 10m0s over 3 stages (gracefulRampDown: 30s, gracefulStop: 30s)

```

data_received.....: 41 MB  68 kB/s
data_sent.....: 30 MB  50 kB/s
http_req_blocked.....: avg=7.72µs  min=0s    med=0s    max=12.37ms p(90)=0s    p(95)=0s
http_req_connecting.....: avg=2.74µs  min=0s    med=0s    max=12.37ms p(90)=0s    p(95)=0s
http_req_duration.....: avg=21.16ms min=2.52ms med=10.39ms max=2.43s  p(90)=39.45ms p(95)=69.24ms
{ expected_response:true }...: avg=21.16ms min=2.52ms med=10.39ms max=2.43s  p(90)=39.45ms p(95)=69.24ms
http_req_failed.....: 0.00% ✓ 1 ✗ 245185
http_req_receiving.....: avg=409.66µs min=0s    med=0s    max=240.8ms p(90)=585.79µs p(95)=1.62ms
http_req_sending.....: avg=16.22µs min=0s    med=0s    max=21.94ms p(90)=0s    p(95)=0s
http_req_tls_handshaking.....: avg=0s    min=0s    med=0s    max=0s    p(90)=0s    p(95)=0s
iterations.....: 122593 204.026636/s
vus.....: 1 min=1 max=300
vus_max.....: 300 min=300 max=300

```

running (10m00.9s), 000/300 VUs, 122593 complete and 0 interrupted iterations

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL



execution: local
script: SpikeTest.js
output: -

scenarios: (100.00%) 1 scenario, 1000 max VUs, 5m0s max duration (incl. graceful stop):
* default: Up to 1000 looping VUs for 4m30s over 6 stages (gracefulRampDown: 30s, gracefulStop: 30s)

data_received.....	16 MB	61 kB/s					
data_sent.....	12 MB	44 kB/s					
http_req_blocked.....	avg=35.4µs	min=0s	med=0s	max=33.59ms	p(90)=0s	p(95)=0s	
http_req_connecting.....	avg=28.37µs	min=0s	med=0s	max=33.59ms	p(90)=0s	p(95)=0s	
http_req_tls_handshaking.....	avg=0s	min=0s	med=0s	max=0s	p(90)=0s	p(95)=0s	
http_req_waiting.....	avg=1.29s	min=2.62ms	med=875.74ms	max=8.25s	p(90)=3s	p(95)=3.37s	
http_reqs.....	98356	363.103723/s					
iteration_duration.....	avg=2.71s	min=1s	med=2.66s	max=9.26s	p(90)=4.19s	p(95)=4.81s	
iterations.....	49178	181.551861/s					
vus.....	3	min=2	max=1000				
vus_max.....	1000	min=1000	max=1000				

running (4m30.9s), 0000/1000 VUs, 49178 complete and 0 interrupted iterations

default ✓ [=====] 0000/1000 VUs 4m30s

PS C:\Users\Nickv\Documents\School\Semester 6\individueel\Kwetter-semester-6\K8s\k6s> |