Test avec k6

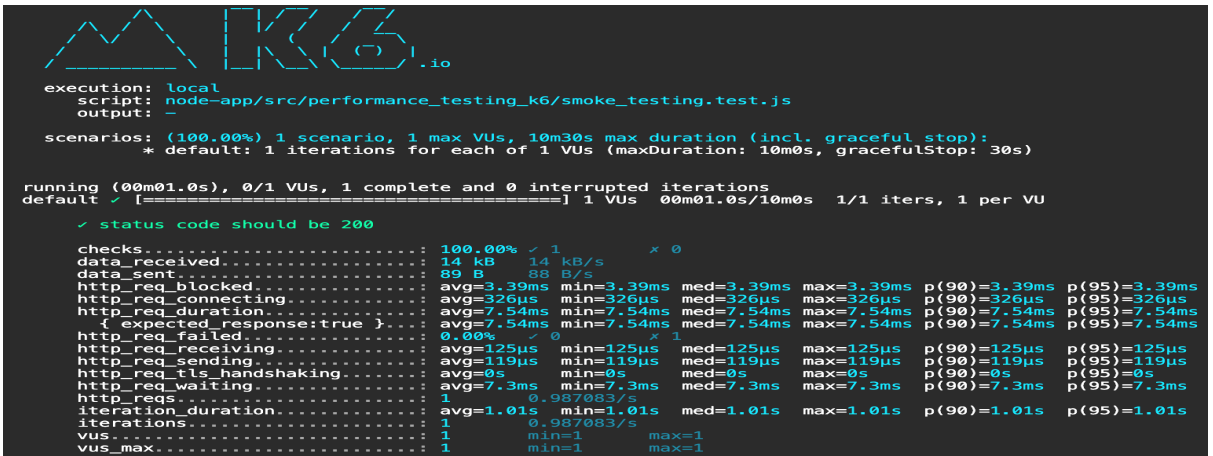
Les différents type de tests :

- **Smoke testing** - vérifiez que votre système ne génère pas d'erreurs sous une charge minimale.
- **Load testing -** accédez aux performances de votre système pendant les charges normales et de pointe
- **Stress testing** - vérifiez la fiabilité et la stabilité de votre système sous une charge lourde ou dans des conditions extrêmes
- **Spike testing** - accédez aux performances de votre système en cas de forte augmentation du trafic
- **Soak testing** - vérifiez la fiabilité et la stabilité de votre système sur une longue période de temps

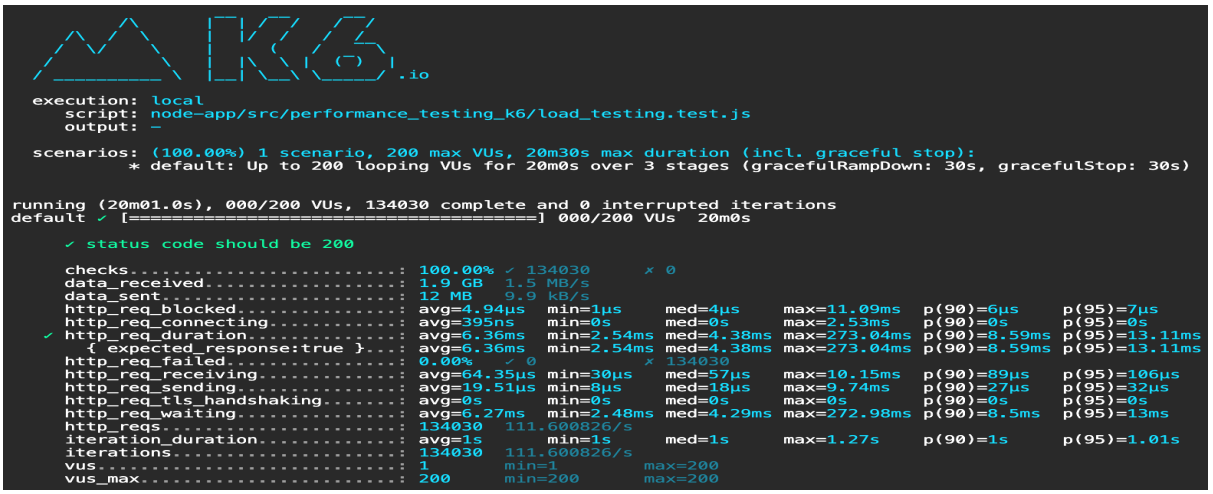
Comprendre les résultats d'un test k6 :

- `vus` - nombre d'utilisateurs virtuels actifs
- `vus_max` - utilisateurs virtuels maximum alloués pour le test
- `iterations` - nombre agrégé d'appels de la fonction
- `iteration_duration` - le temps total nécessaire pour exécuter la fonction
- `dropped_iterations` - nombre d'entre eux `iterations` n'ont pas pu être démarrés
- `data_received` - quantité de données reçues
- `data_sent` - quantité de données envoyées
- `checks` - taux de contrôles réussis
- `http_reqs` - total des requêtes générées par k6
- `http_req_blocked` - temps passé à attendre une connexion TCP gratuite avant de lancer la requête
- `http_req_connecting` - temps passé à établir une connexion TCP
- `http_req_tls_handshaking` - temps passé sur la négociation TLS
- `http_req_sending` - temps consacré à l'envoi de données
- `http_req_waiting` - temps passé à attendre une réponse de l'hôte distant
- `http_req_receiving` - temps consacré à la réception des données
- `http_req_failed` - Le nombre de requêtes échouées selon la méthode *setResponseCallback()*
- `http_req_duration` - durée totale de la demande. Il est calculé en fonction de `http_req_sending` + `http_req_waiting` + `http_req_receiving.`

`Résultat du test smoke testing`

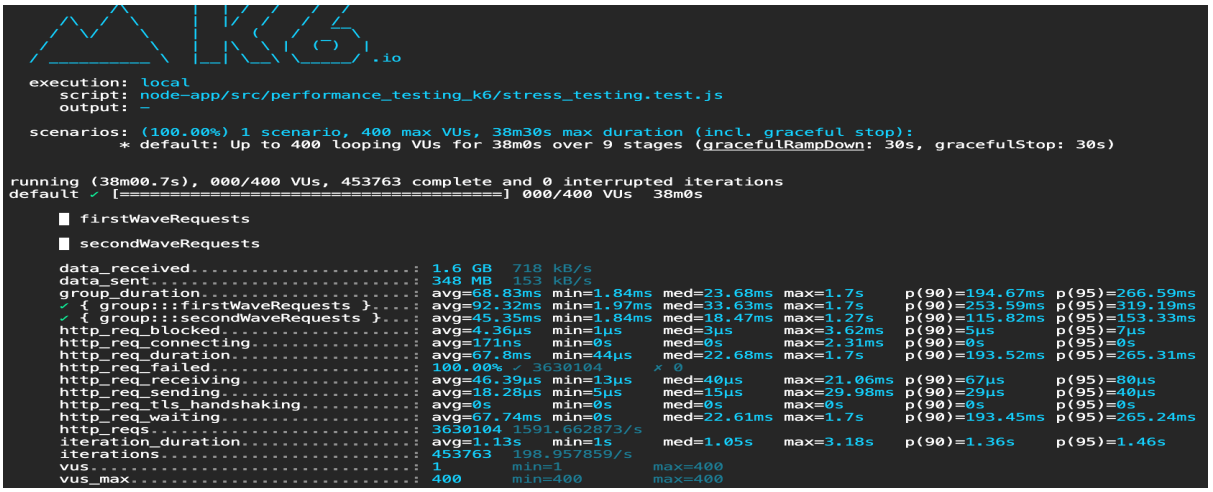


Résultat du test load testing



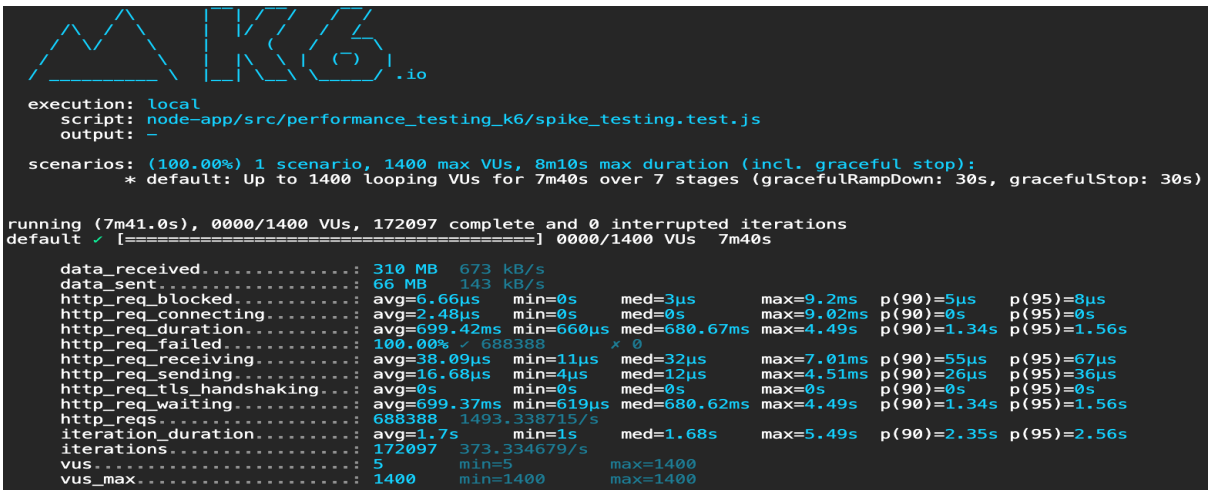
```
  execution: local
     script: node-app/src/performance_testing_k6/load_testing.test.js
     output: -

  scenarios: (100.00%) 1 scenario, 200 max VUs, 20m30s max duration (incl. graceful stop):
           * default: Up to 200 looping VUs for 20m0s over 3 stages (gracefulRampDown: 30s, gracefulStop: 30s)


running (20m01.0s), 000/200 VUs, 134030 complete and 0 interrupted iterations
default ✓ [======================================] 000/200 VUs  20m0s

     ✓ status code should be 200

     checks.........................: 100.00% ✓ 134030      ✗ 0
     data_received..................: 1.9 GB  1.5 MB/s
     data_sent......................: 12 MB   9.9 kB/s
     http_req_blocked...............: avg=4.94µs   min=1µs     med=4µs     max=11.09ms  p(90)=6µs     p(95)=7µs
     http_req_connecting............: avg=395ns    min=0s      med=0s      max=2.53ms   p(90)=0s      p(95)=0s
   ✓ http_req_duration..............: avg=6.36ms   min=2.54ms  med=4.38ms  max=273.04ms p(90)=8.59ms  p(95)=13.11ms
     { expected_response:true }...: avg=6.36ms   min=2.54ms  med=4.38ms  max=273.04ms p(90)=8.59ms  p(95)=13.11ms
     http_req_failed................: 0.00%   ✓ 0           ✗ 134030
     http_req_receiving.............: avg=64.35µs  min=30µs    med=57µs    max=10.15ms  p(90)=89µs    p(95)=106µs
     http_req_sending...............: avg=19.51µs  min=8µs     med=18µs    max=9.74ms   p(90)=27µs    p(95)=32µs
     http_req_tls_handshaking.......: avg=0s       min=0s      med=0s      max=0s       p(90)=0s      p(95)=0s
     http_req_waiting...............: avg=6.27ms   min=2.48ms  med=4.29ms  max=272.98ms p(90)=8.5ms   p(95)=13ms
     http_reqs......................: 134030  111.600826/s
     iteration_duration.............: avg=1s       min=1s      med=1s      max=1.27s    p(90)=1s      p(95)=1.01s
     iterations.....................: 134030  111.600826/s
     vus............................: 1       min=1         max=200
     vus_max........................: 200     min=200       max=200
```

Résultat du stress testing



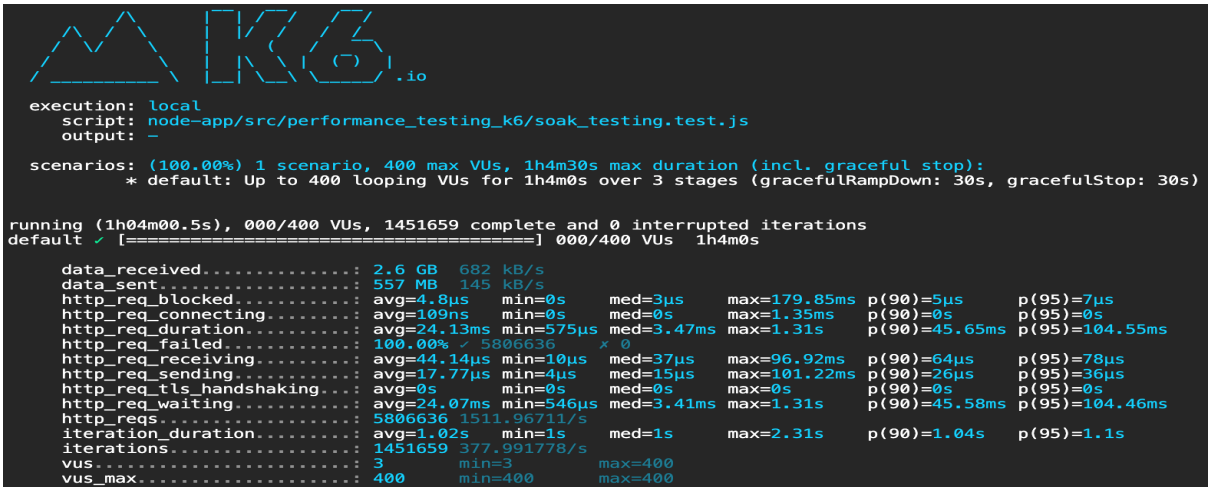
```
  execution: local
     script: node-app/src/performance_testing_k6/stress_testing.test.js
     output: -

  scenarios: (100.00%) 1 scenario, 400 max VUs, 38m30s max duration (incl. graceful stop):
           * default: Up to 400 looping VUs for 38m0s over 9 stages (gracefulRampDown: 30s, gracefulStop: 30s)


running (38m00.7s), 000/400 VUs, 453763 complete and 0 interrupted iterations
default ✓ [======================================] 000/400 VUs  38m0s

     ✓ firstWaveRequests

     ✓ secondWaveRequests

     data_received..................: 1.6 GB  718 kB/s
     data_sent......................: 348 MB  153 kB/s
     group_duration.................: avg=68.83ms min=1.84ms med=23.68ms max=1.7s    p(90)=194.67ms p(95)=266.59ms
     ✓ { group:::firstWaveRequests }...: avg=92.32ms min=1.97ms med=33.63ms max=1.7s    p(90)=253.59ms p(95)=319.19ms
     ✓ { group:::secondWaveRequests }...: avg=45.35ms min=1.84ms med=18.47ms max=1.27s   p(90)=115.82ms p(95)=153.33ms
     http_req_blocked...............: avg=4.36µs   min=1µs     med=3µs     max=3.62ms   p(90)=5µs     p(95)=7µs
     http_req_connecting............: avg=171ns    min=0s      med=0s      max=2.31ms   p(90)=0s      p(95)=0s
     http_req_duration..............: avg=67.8ms   min=44µs    med=22.68ms max=1.7s     p(90)=193.52ms p(95)=265.31ms
     http_req_failed................: 100.00% ✓ 3630104     ✗ 0
     http_req_receiving.............: avg=46.39µs  min=13µs    med=40µs    max=21.06ms  p(90)=67µs    p(95)=80µs
     http_req_sending...............: avg=18.28µs  min=5µs     med=15µs    max=29.98ms  p(90)=29µs    p(95)=40µs
     http_req_tls_handshaking.......: avg=0s       min=0s      med=0s      max=0s       p(90)=0s      p(95)=0s
     http_req_waiting...............: avg=67.74ms  min=0s      med=22.61ms max=1.7s     p(90)=193.45ms p(95)=265.24ms
     http_reqs......................: 3630104 1591.662873/s
     iteration_duration.............: avg=1.13s    min=1s      med=1.05s   max=3.18s    p(90)=1.36s   p(95)=1.46s
     iterations.....................: 453763  198.957859/s
     vus............................: 1       min=1         max=400
     vus_max........................: 400     min=400       max=400
```

Résultat du spike testing



```
  execution: local
     script: node-app/src/performance_testing_k6/spike_testing.test.js
     output: -

  scenarios: (100.00%) 1 scenario, 1400 max VUs, 8m10s max duration (incl. graceful stop):
           * default: Up to 1400 looping VUs for 7m40s over 7 stages (gracefulRampDown: 30s, gracefulStop: 30s)


running (7m41.0s), 0000/1400 VUs, 172097 complete and 0 interrupted iterations
default ✓ [======================================] 0000/1400 VUs  7m40s

     data_received..............: 310 MB  673 kB/s
     data_sent..................: 66 MB   143 kB/s
     http_req_blocked...........: avg=6.66µs   min=0s     med=3µs     max=9.2ms   p(90)=5µs     p(95)=8µs
     http_req_connecting........: avg=2.48µs   min=0s     med=0s      max=9.02ms  p(90)=0s      p(95)=0s
     http_req_duration..........: avg=699.42ms min=660µs med=680.67ms max=4.49s   p(90)=1.34s   p(95)=1.56s
     http_req_failed............: 100.00% ✓ 688388      ✗ 0
     http_req_receiving.........: avg=38.09µs  min=11µs  med=32µs    max=7.01ms  p(90)=55µs    p(95)=67µs
     http_req_sending...........: avg=16.68µs  min=4µs   med=12µs    max=4.51ms  p(90)=26µs    p(95)=36µs
     http_req_tls_handshaking...: avg=0s       min=0s    med=0s      max=0s      p(90)=0s      p(95)=0s
     http_req_waiting...........: avg=699.37ms min=619µs med=680.62ms max=4.49s   p(90)=1.34s   p(95)=1.56s
     http_reqs..................: 688388  1493.338715/s
     iteration_duration.........: avg=1.7s     min=1s     med=1.68s   max=5.49s   p(90)=2.35s   p(95)=2.56s
     iterations.................: 172097  373.334679/s
     vus........................: 5       min=5         max=1400
     vus_max....................: 1400    min=1400      max=1400
```

Résultat du soak testing



```
  execution: local
     script: node-app/src/performance_testing_k6/soak_testing.test.js
     output: -

  scenarios: (100.00%) 1 scenario, 400 max VUs, 1h4m30s max duration (incl. graceful stop):
           * default: Up to 400 looping VUs for 1h4m0s over 3 stages (gracefulRampDown: 30s, gracefulStop: 30s)


running (1h04m00.5s), 000/400 VUs, 1451659 complete and 0 interrupted iterations
default ✓ [======================================] 000/400 VUs  1h4m0s

     data_received..............: 2.6 GB  682 kB/s
     data_sent..................: 557 MB  145 kB/s
     http_req_blocked...........: avg=4.8µs    min=0s     med=3µs     max=179.85ms p(90)=5µs     p(95)=7µs
     http_req_connecting........: avg=109ns    min=0s     med=0s      max=1.35ms   p(90)=0s      p(95)=0s
     http_req_duration..........: avg=24.13ms min=575µs med=3.47ms max=1.31s    p(90)=45.65ms p(95)=104.55ms
     http_req_failed............: 100.00% ✓ 5806636     ✗ 0
     http_req_receiving.........: avg=44.14µs min=10µs   med=37µs    max=96.92ms  p(90)=64µs    p(95)=78µs
     http_req_sending...........: avg=17.77µs min=4µs    med=15µs    max=101.22ms p(90)=26µs    p(95)=36µs
     http_req_tls_handshaking...: avg=0s       min=0s     med=0s      max=0s       p(90)=0s      p(95)=0s
     http_req_waiting...........: avg=24.07ms min=546µs med=3.41ms max=1.31s    p(90)=45.58ms p(95)=104.46ms
     http_reqs..................: 5806636 1511.96711/s
     iteration_duration.........: avg=1.02s    min=1s     med=1s      max=2.31s    p(90)=1.04s   p(95)=1.1s
     iterations.................: 1451659 377.991778/s
     vus........................: 3       min=3         max=400
     vus_max....................: 400     min=400       max=400
```

Source :

https://k6.io/docs/javascript-api/k6-http/setresponsecallback-callback/

https://k6.io/docs/using-k6/thresholds/

https://k6.io/docs/test-types/smoke-testing/

https://k6.io/docs/test-types/load-testing/

https://k6.io/docs/test-types/stress-testing/

https://k6.io/docs/test-types/stress-testing/#spike-testing

https://k6.io/docs/test-types/soak-testing/