2CSDE80 Software Testing and Quality Assurance

Lab-8 Task

Submitted by: Labdhi Sheth 18BCE101

Aim: To demonstrate a load testing experiment.

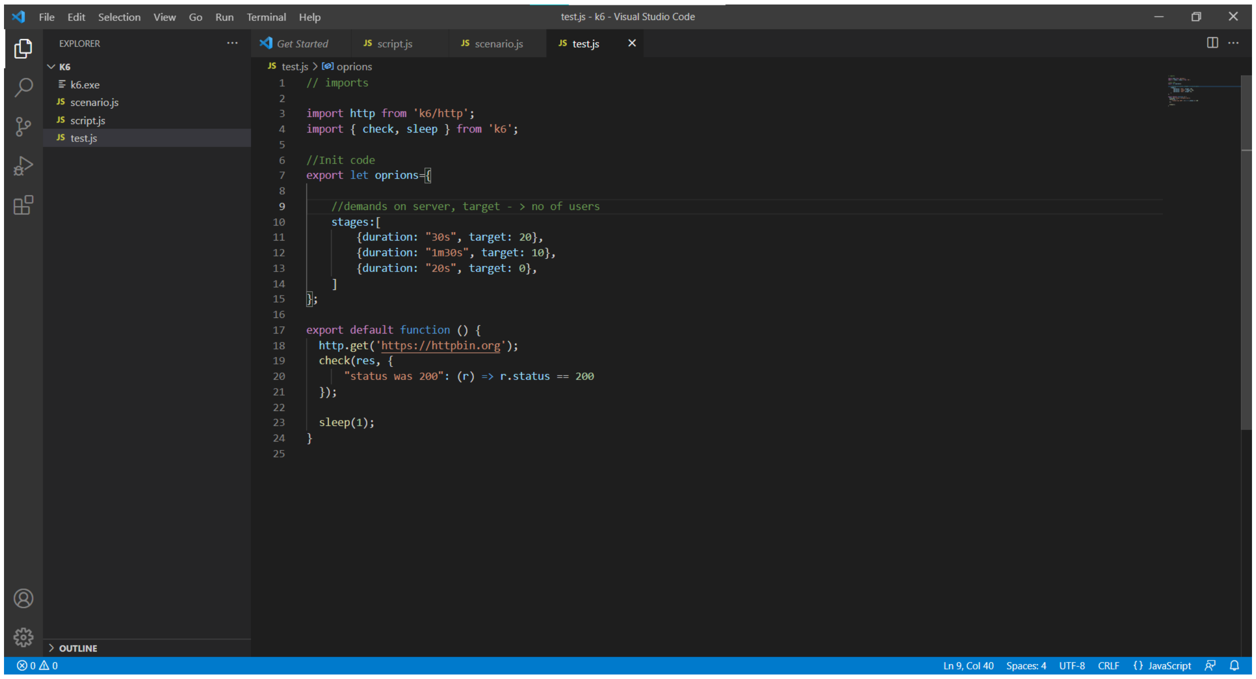
Tasks:

- **Understand the K6 tool.**
- **Understand the various performance metric of the K6 tool by running the mentioned script.**
- **Create different scripts for non-built metrics of the K6 tool.**

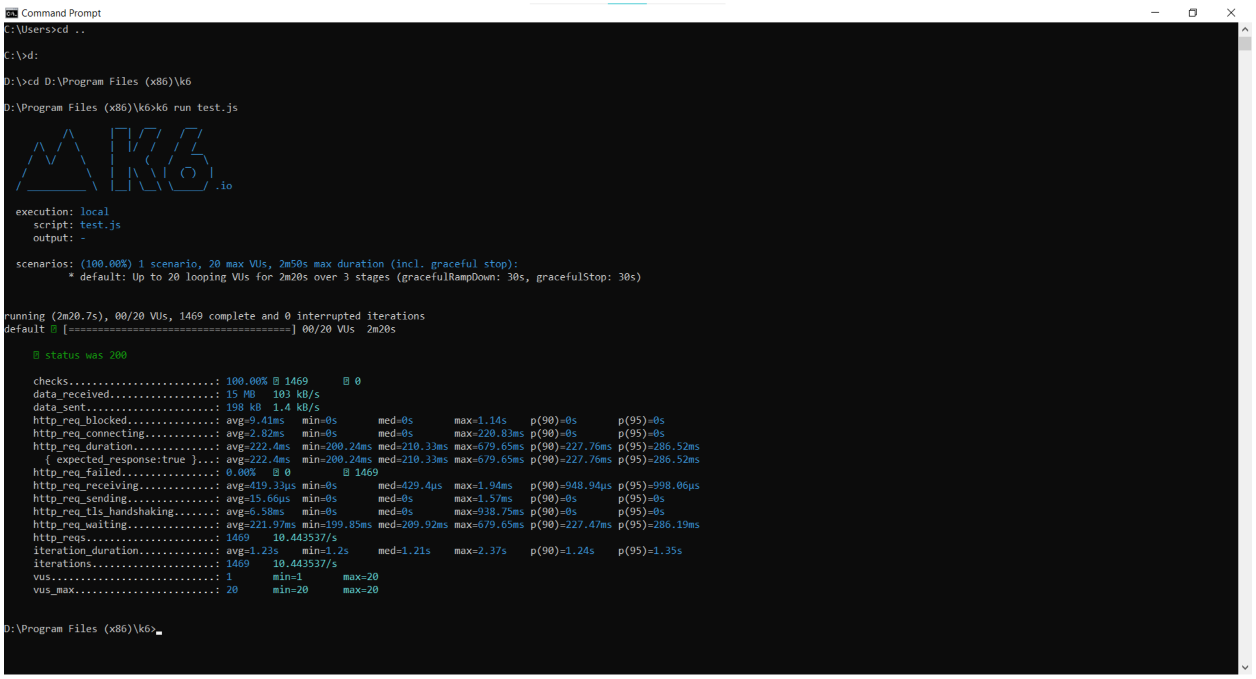
Methodology:

Load testing generally refers to the practice of modeling the expected usage of a software program by simulating multiple users accessing the program concurrently. It ensures that your application can perform as expected in production. It identifies where and when your application breaks, so you can fix the issue before shipping to production. It gives confidence in the system & its reliability and performance. It helps identify the bottlenecks in the system under heavy user stress scenarios before they happen in a production environment.

1. test.js


File Edit Selection View Go Run Terminal Help
test.js - k6 - Visual Studio Code
EXPLORER
K6
k6.exe
scenario.js
script.js
test.js
Get Started
script.js
scenario.js
test.js
test.js > oprions
OUTLINE
Ln 9, Col 40 Spaces: 4 UTF-8 CRLF {} JavaScript

```javascript
// imports

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

//Init code
export let oprions={

    //demands on server, target - > no of users
    stages:[
        {duration: "30s", target: 20},
        {duration: "1m30s", target: 10},
        {duration: "20s", target: 0},
    ]
};

export default function () {
  http.get('https://httpbin.org');
  check(res, {
      "status was 200": (r) => r.status == 200
  });

  sleep(1);
}
```



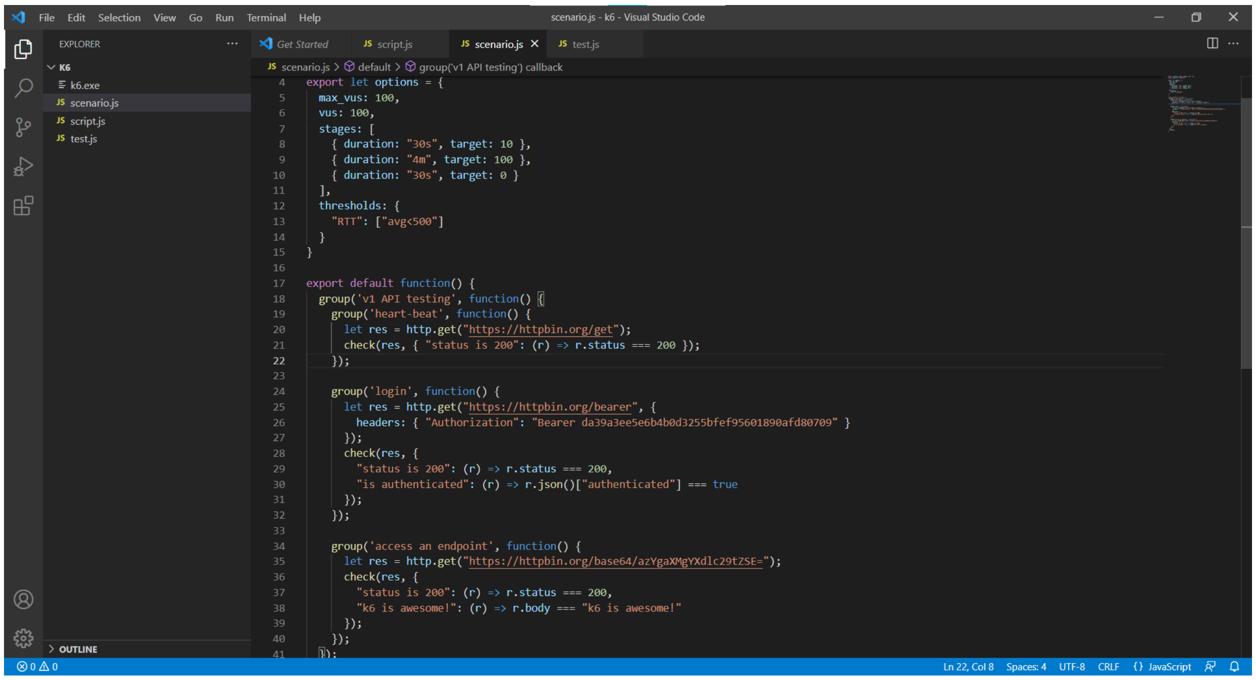
```
C:\Users>cd ..

C:\>d:

D:\>cd D:\Program Files (x86)\k6

D:\Program Files (x86)\k6>k6 run test.js

          /\      |‾‾| /‾‾/   /‾‾/
     /\  /  \     |  |/  /   /  /
    /  \/    \    |     (   /   ‾‾\
   /          \   |  |\  \ |  (‾)  |
  / __________ \  |__| \__\ \_____/ .io

  execution: local
     script: test.js
     output: -

  scenarios: (100.00%) 1 scenario, 20 max VUs, 2m50s max duration (incl. graceful stop):
           * default: Up to 20 looping VUs for 2m20s over 3 stages (gracefulRampDown: 30s, gracefulStop: 30s)


running (2m20.7s), 00/20 VUs, 1469 complete and 0 interrupted iterations
default ? [====================================] 00/20 VUs  2m20s

     ? status was 200

     checks.........................: 100.00% ? 1469      ? 0
     data_received..................: 15 MB   103 kB/s
     data_sent......................: 198 kB  1.4 kB/s
     http_req_blocked...............: avg=9.41ms   min=0s       med=0s       max=1.14s    p(90)=0s       p(95)=0s
     http_req_connecting............: avg=2.82ms   min=0s       med=0s       max=220.83ms p(90)=0s       p(95)=0s
     http_req_duration..............: avg=222.4ms  min=200.24ms med=210.33ms max=679.65ms p(90)=227.76ms p(95)=286.52ms
       { expected_response:true }...: avg=222.4ms  min=200.24ms med=210.33ms max=679.65ms p(90)=227.76ms p(95)=286.52ms
     http_req_failed................: 0.00%   ? 0         ? 1469
     http_req_receiving.............: avg=419.33µs min=0s       med=429.4µs  max=1.94ms   p(90)=948.94µs p(95)=998.06µs
     http_req_sending...............: avg=15.66µs  min=0s       med=0s       max=1.57ms   p(90)=0s       p(95)=0s
     http_req_tls_handshaking.......: avg=6.58ms   min=0s       med=0s       max=938.75ms p(90)=0s       p(95)=0s
     http_req_waiting...............: avg=221.97ms min=199.85ms med=209.92ms max=679.65ms p(90)=227.47ms p(95)=286.19ms
     http_reqs......................: 1469    10.443537/s
     iteration_duration.............: avg=1.23s    min=1.2s     med=1.21s    max=2.37s    p(90)=1.24s    p(95)=1.35s
     iterations.....................: 1469    10.443537/s
     vus............................: 1       min=1      max=20
     vus_max........................: 20      min=20     max=20


D:\Program Files (x86)\k6>
```

Let the scenario be:

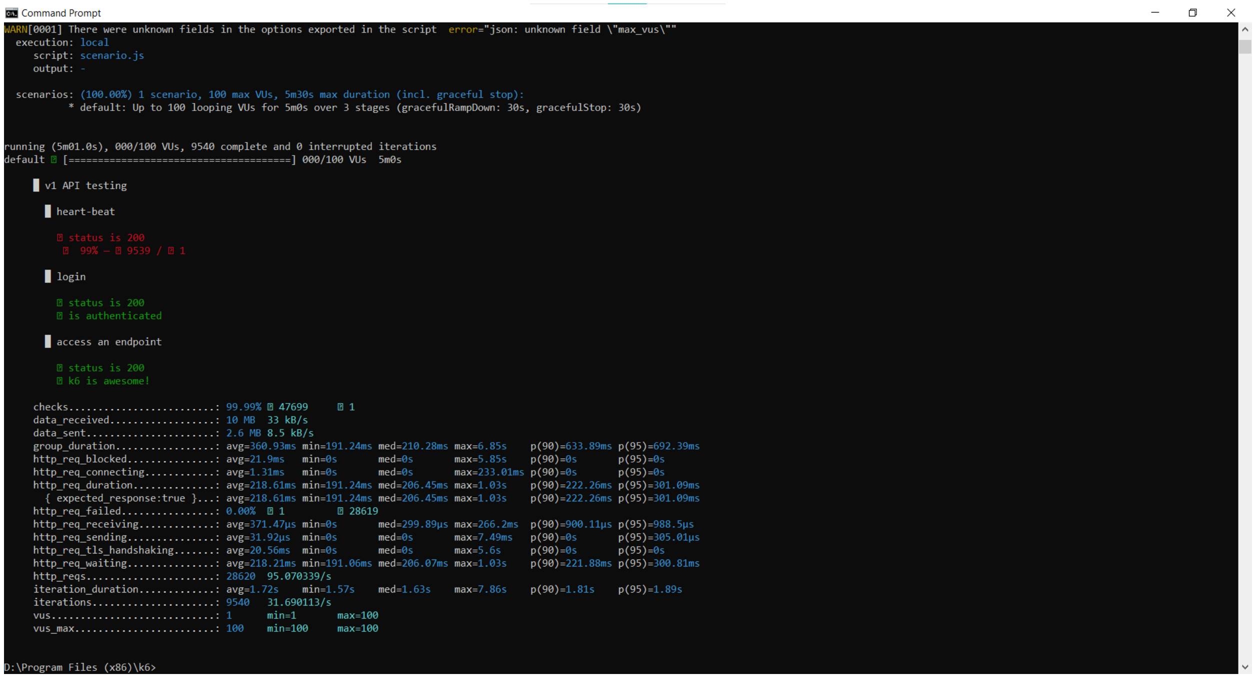
1. A user checks of the API is up, hence checking the heartbeat of our API, and then tries to send further messages.
2. Then they try to create a token to be able to access other parts of our API or the private areas.

3. Finally, they try to access an endpoint of our API.

This scenario is to be tested by 100 individual users, starting with 10 users, ramping up to 100 and gradually down to 0, and to keep the response time of all requests below 500ms.


File Edit Selection View Go Run Terminal Help
scenario.js - k6 - Visual Studio Code
EXPLORER
K6
k6.exe
scenario.js
script.js
test.js
OUTLINE
Get Started
script.js
scenario.js
test.js
scenario.js > default > group('v1 API testing') callback
Ln 22, Col 8 Spaces: 4 UTF-8 CRLF {} JavaScript

```
export let options = {
  max_vus: 100,
  vus: 100,
  stages: [
    { duration: "30s", target: 10 },
    { duration: "4m", target: 100 },
    { duration: "30s", target: 0 }
  ],
  thresholds: {
    "RTT": ["avg<500"]
  }
}

export default function() {
  group('v1 API testing', function() {
    group('heart-beat', function() {
      let res = http.get("https://httpbin.org/get");
      check(res, { "status is 200": (r) => r.status === 200 });
    });

    group('login', function() {
      let res = http.get("https://httpbin.org/bearer", {
        headers: { "Authorization": "Bearer da39a3ee5e6b4b0d3255bfef95601890afd80709" }
      });
      check(res, {
        "status is 200": (r) => r.status === 200,
        "is authenticated": (r) => r.json()["authenticated"] === true
      });
    });

    group('access an endpoint', function() {
      let res = http.get("https://httpbin.org/base64/azYgaXMgYXdlc29tZSE=");
      check(res, {
        "status is 200": (r) => r.status === 200,
        "k6 is awesome!": (r) => r.body === "k6 is awesome!"
      });
    });
  });
```



```
WARN[0001] There were unknown fields in the options exported in the script  error="json: unknown field \"max_vus\""
  execution: local
     script: scenario.js
     output: -

  scenarios: (100.00%) 1 scenario, 100 max VUs, 5m30s max duration (incl. graceful stop):
           * default: Up to 100 looping VUs for 5m0s over 3 stages (gracefulRampDown: 30s, gracefulStop: 30s)


running (5m01.0s), 000/100 VUs, 9540 complete and 0 interrupted iterations
default ▯ [======================================] 000/100 VUs  5m0s

     █ v1 API testing

       █ heart-beat

         ▯ status is 200
          ▯  99% — ▯ 9539 / ▯ 1

       █ login

         ▯ status is 200
         ▯ is authenticated

       █ access an endpoint

         ▯ status is 200
         ▯ k6 is awesome!

     checks.........................: 99.99% ▯ 47699     ▯ 1
     data_received..................: 10 MB  33 kB/s
     data_sent......................: 2.6 MB 8.5 kB/s
     group_duration.................: avg=360.93ms min=191.24ms med=210.28ms max=6.85s    p(90)=633.89ms p(95)=692.39ms
     http_req_blocked...............: avg=21.9ms   min=0s       med=0s       max=5.85s    p(90)=0s       p(95)=0s
     http_req_connecting............: avg=1.31ms   min=0s       med=0s       max=233.01ms p(90)=0s       p(95)=0s
     http_req_duration..............: avg=218.61ms min=191.24ms med=206.45ms max=1.03s    p(90)=222.26ms p(95)=301.09ms
       { expected_response:true }...: avg=218.61ms min=191.24ms med=206.45ms max=1.03s    p(90)=222.26ms p(95)=301.09ms
     http_req_failed................: 0.00%  ▯ 1         ▯ 28619
     http_req_receiving.............: avg=371.47µs min=0s       med=299.89µs max=266.2ms  p(90)=900.11µs p(95)=988.5µs
     http_req_sending...............: avg=31.92µs  min=0s       med=0s       max=7.49ms   p(90)=0s       p(95)=305.01µs
     http_req_tls_handshaking.......: avg=20.56ms  min=0s       med=0s       max=5.6s     p(90)=0s       p(95)=0s
     http_req_waiting...............: avg=218.21ms min=191.06ms med=206.07ms max=1.03s    p(90)=221.88ms p(95)=300.81ms
     http_reqs......................: 28620  95.070339/s
     iteration_duration.............: avg=1.72s    min=1.57s    med=1.63s    max=7.86s    p(90)=1.81s    p(95)=1.89s
     iterations.....................: 9540   31.690113/s
     vus............................: 1      min=1       max=100
     vus_max........................: 100    min=100     max=100


D:\Program Files (x86)\k6>
```

All the checks in the tests are passed, specifically 56213 checks in 33741 requests. The amount of data sent and received, vus, iterations, and some other metrics are also shown. The very important part is the metrics that start with HTTP. They signify the average, minimum, maximum, p(90), and p(95) of the amount of time each request or group of requests has taken to complete. Since we have defined the average threshold to be 500ms. Since the average time of all requests hasn't taken that long, the test is passed.

Conclusion:

K6 tool is robust and well-documented Javascript APIs for test scripting. It has deep customization through multiple configuration options and Scenarios. Parameterization through environment variables Websockets support and lifecycle hooks for customizing setup and teardown.