Grafana k6

Canton Linux Enthusiasts

G. Saxena (@gsvolt)

Thursday, February 24, 2022

Section 1

Introduction

Introduction

- G, gsvolt
- Co-Founder, Akron Makerspace
  - Makerspace:
    - A place in which people with shared interests, can gather to work on projects while sharing ideas, equipment, and knowledge.
- Owner, Whiskey Bits LLC
  - Custom Software
- Fond of k6

Section 2

Performance Testing

Performance

- The capabilities of (how good, fast or efficient) of a machine when observed under particular conditions

Testing

- Take measures to check the quality, performance or reliability of something before putting it into widespread use

Performance Testing

- Act of measuring, validating or checking a systems capabilities (stability, speed, efficiency) when observed under particular conditions
- Branch of testing whose primary concern is verifying how well a system works instead of if it works

Focus 1/3: Time

- Measure responsiveness of a solution to given events. How fast will it react.
- Can include qualitative aspects of a user's experience like responsiveness during normal operations.

Focus 2/3: Efficiency

- Impact after a given event
- These can be computer resource consumption like: CPU, RAM and measurable metrics like:
  - temperature, connections, data transfers etc).

Focus 3/3: Scenario

- “when observed under specific conditions”
- Eg: Real User Monitoring, instrumentation, load-tests, single-user tests, no-user tests

Why Performance Testing?

- To make sure your software responds fast
- Test resource consumption - ensure resources are properly used for multiple users
- Ensure availability (SRE) of your software
- Allow early detection of problems in software

Section 3

k6 OSS

k6 OSS

- 

- “Modern load testing tool built for developer happiness”
- “Democratize load testing”

Existing Load Testing Tools

- Apache JMeter
- Gatling
- Micro Focus Load Runner
- And others...

History

- 2008: Formerly known as Load Impact
- 2011: Load Impact 2.0
- 2015: Load Impact 3.0
- 2017: k6 released to the public
- 2020: Active development on xk6 begins (Extensions/Integrations for k6)
- 2021: Acquired by Grafana Labs

Few myths of Load Testing

- Load tests always break applications
- Repeatedly send the same requests to API
- Expensive to do load testing
- Only load test in production

Why k6?

- Developer and tester friendly
  - Javascript scripting language to write tests
  - Familiar to users of Cypress, Selenium WebDriver, Puppeteer, Playwright
  - Visual Studio Code Extension
  - Many integrations

Why k6?

- Performant
  - Written in Go with performance in mind. Keyword: *concurrent*
- Upto ~40K users on one load generator (one k6 instance at runtime)
  - This is important because with ~30K VU's, we can get 300k requests per second or 6-12M requests per minute
  - Each VU is expected to use atleast 1-5 MB RAM
- Locally we can aim to run about 1K VUs using about 1-5 GB RAM in a typical load test

Why k6?

- Easy to Scale
  - Run tests with one load generator locally, or use the same script on the cloud
  - Can be used in tandem with tools like kubernetes to mimick Google scale load tests

Why k6?

- Integrates with tools like CircleCI, Github Actions, Gitlab, Jenkins, Team City, Azure Pipelines
- k6 OSS and k6 Cloud (SaaS, paid, value added offering) - similar to Grafana

Why k6? (Light Reading)

- Executive Solution Sheet
- Whitepaper: Modern Load Testing for Engineering Teams
- Stewardship
- Buy vs Build

Load Testing Manifesto

Manifesto

- Simple testing is better than no testing
- Load testing should be goal oriented
- Load testing by developers
- Developer experience is super important
  - Local environment
  - Everything as code
  - Automation
- Load test in a pre-production environment

Installation: k6 on Debian

```
$ sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 \
    --recv-keys C5AD17C747E3415A3642D57D77C6C491D6AC1D69
$ echo "deb https://dl.k6.io/deb stable main" | \
    sudo tee /etc/apt/sources.list.d/k6.list
$ sudo apt-get update
$ sudo apt-get install k6
```

Other platforms...

```
$ k6 version
k6 v0.36.0 (2022-01-24T09:50:03+0000/ff3f8df, go1.17.6, linux/amd64)
```

k6 CLI

```
k6 [command]

Available Commands:
- archive     Create an archive
- cloud       Run a test on the cloud
- convert     Convert a HAR file to a k6 script
- help        Help about any command
- inspect     Inspect a script or archive
- login       Authenticate with a service
- pause       Pause a running test
- resume      Resume a paused test
- run         Start a load test
- scale       Scale a running test
- stats       Show test metrics
- status      Show test status
- version     Show application version
```

Test Life Cycle

```
// test.js
// 1. init code
// - Can import modules

export function setup() {
  // 2. setup code - called once per test, after 1.
}

export default function (data) {
  // 3. VU (Virtual User) code - Mandatory (hard)
  // Runs repeatedly - locally, in the cloud as well as in clustered mode
  // - Cannot load anything from local filesystem or modules
}

export function teardown(data) {
  // 4. teardown code - called once per test, after 3.
}
```

Section 4

k6 Demo

Pre-demo

Launch the api in its own terminal:

```
cd api
make run
```

01_test.js

For this and other tests that follow, run one of these commands in a different terminal:

```
cd test
make run_01
```

or

```
cd test
k6 run 01_test.js
```

Review Results in Terminal

- Only 1 VU was utilized
- Report includes statistics like:
  - data_received, data_sent
  - http_reqs
- Very simple/silly test: Only uses 1 VU
- Next, lets run `make run_01_load` to run the test for 30s with 10 VUs

01_test.js - Part2

```
cd test
make run_01_load
```

or

```
cd test
k6 run --vus 10 --duration 30s 01_test.js
```

Review Results in Terminal

- Slighly better imitation of a load test
- Immediately see valuable feedback
- Lets learn/discuss/review:
  - Metric Types (similar to Prometheus) and,
  - Built-in metrics
- Next, lets do two modifications:
  - Add a local filesystem module for our tests so that we can share constants like url to use
  - Roll our command-line arguments into an options object in the script and increase the number of VUs to 100

02_test.js

```
cd test
make run_02
```

Review Results in Terminal

- Getting a bit better at loading the system
- ~160MB data transfer by approx. 350K requests
- We've removed command line arguments
- Still not real world though
- Next, lets do these modifications:
  - Add stages in our options to make the test more real-world
  - Add sleep method to allow for some wait time
  - Add check method to check response from api
  - Add a non-existing route/page in addition to /albums to see how the load test reports its results

03_test.js

```
cd test
make run_03
```

Review Results in Terminal

- Few checks failed and were reported in the results
- Hmm.. failures.. wonder what the exit code is for this run in the terminal, so lets check `echo $?`
  - Outcome: 0
- Ideologically, k6 always returns 0 - to use k6 in CI/CD pipelines, we need to add thresholds to explicitly mark a failure

04_test.js

```
cd test
make run_04
```

Review Results in Terminal

- This time we see this:
  - ERRO[0011] some thresholds have failed
- Also, `echo $?` results in a non-zero return value which can be safely used in CI/CD

Section 5

k6 - Deeper Dive

Test Types

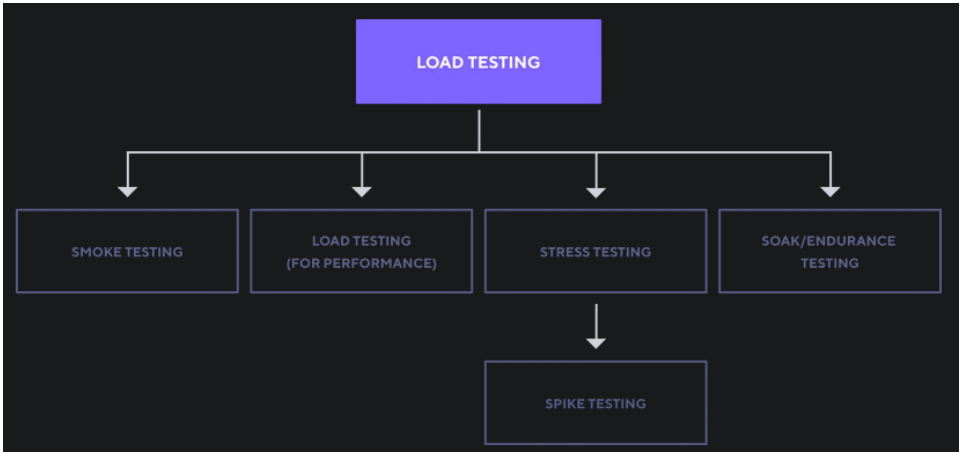

Figure 1: Test Types

Smoke Test

- Small load test using which we can verify test script doesn't have issues and no errors are thrown under minimal load
- More details...

Load Test

- Performance of a system in terms of concurrent users or requests per second
- Defines tests that represent typical load and peak load
- Using thresholds you define tests for both cases
- Recommended to always provide ramp-up time in your scripts
- Advice: “Start Small”
- If your load tests are failing you’ve managed to stress your system
- More details. . .

Stress Test

- Attempt to find extreme conditions your system can handle, gradually loading the system
- “Beyond breaking point”
- Eg: Black Friday/Cyber Monday
- More details...

Spike Test

- Immediately (as quickly as possible) load the entire system - no ramp-up times
- Surge
- Objective is to figure out if/how your system recovers from a spike
- More details...

Soak Test

- Reliability and performance of a system over an extended period of time
- Find issues related to memory leaks, lost requests, race-conditions, disk storage, and external requests
- Typically a way to verify how well your database is doing
- Recommended to start with 80% of your systems capacity, and extend duration to hours
- This test may yield excess cost - so be mindful of that in a cloud environment!
- More details...

Advanced Features

- HAR Converter
- Browser Recorder
- Examples & Tutorials
- Postman <-> k6
- Typescript <-> k6

Section 6

xk6

Why xk6?

- **Ex**tensibility
- Step1: Choose extensions
- Step2: Download a k6 binary with those extensions
- Step3: Use the downloaded version of k6 for your tests that can leverage the extensions you selected

Protocols supported by k6

- HTTP/1.1
- HTTP/2
- WebSockets
- gRPC
- Examples of extensions:
  - xk6-datadog: Query datadog metrics
  - xk6-sql: Load test database systems
  - And others...

Custom Extensions

- One rule: Name needs to begin with “xk6-” in a public github repository
- You can create your own extensions

Integrations

- Integrations

Section 7

k6 Cloud

Why k6 Cloud ?

- Scalability
- Automation
- collaboration
- Rich Insights
- Integrations

Test Builder

- Test Builder

k6 OSS vs k6 Cloud

- Comparing k6 OSS and k6 Cloud

Short Example

- Free Tier offers 50 concurrent users after a sign-up

```
$ export K6_TOKEN=secret_token
$ k6 login cloud --token $K6_TOKEN
$ k6 cloud test.js
```

Pricing

- Developer
- Team
- Pro
- Custom & Enterprise
- More Details. . .

k6 Cloud Docs

- Docs

Section 8

k6 Pros vs Cons

k6 Pros vs Cons

Pros

- Not declarative (glad they went away from yaml/json in favor of JS)
- JS itself isn't executed, but converted to Go
- Very extensible
- Excellent performance
- Self-hosted or SaaS
- Results export

Cons

- None that I have witnessed yet :)

Section 9

References/Resources

Github

- Awesome k6
- k6
- xk6
- Other repos. . .

Other

- Slack
- Youtube
- Forum
- StackOverflow Tag

Section 10

Questions/Feedback