Perfect question — you should be able to **explain your k6 test script clearly and confidently**. Here’s a structured way to do it:

**1️⃣ Purpose of the test**

“I wanted to measure the performance and scalability of my URL shortener under high load, and also show the impact of caching versus direct database access.”

**2️⃣ What the script does**

* **Loads all short URLs** from a file (short\_urls.txt) using SharedArray so they can be shared across virtual users efficiently.
* **Simulates virtual users** (vus) performing GET requests to random short URLs, mimicking real user traffic.
* **Disables automatic redirects** (redirects: 0) so that the 302 status and Location header can be verified.
* **Performs checks**:
  + status is 302 → the short URL correctly redirects.
  + has Location header → ensures redirect target exists.
* **Records metrics**:
  + http\_req\_duration → response time per request.
  + http\_req\_failed → failed request rate.
* **Sleep between requests** (sleep(0.1)) → prevents overwhelming your laptop while still testing concurrency.

**3️⃣ Load stages**

stages: [

{ duration: '1m', target: 100 }, // ramp-up to 100 users

{ duration: '3m', target: 100 }, // sustain load

{ duration: '1m', target: 0 }, // ramp-down

]

* Gradually increases users to see how the system behaves under increasing load.
* Maintains steady load to measure stability.
* Gradually reduces to observe recovery.

**4️⃣ How it tests caching impact**

* You run **two versions** of the backend:
  1. With Redis caching enabled
  2. Without caching (direct database hits)
* Compare metrics: average response time, 95th percentile, throughput.
* Demonstrates how caching reduces latency and improves scalability.

**5️⃣ Metrics and thresholds**

* http\_req\_duration p(95)<500ms → ensures 95% of requests are fast.
* http\_req\_failed rate<0.01 → ensures almost no failures.
* You can report: average, median, max response times, throughput (req/sec), % of successful requests.

✅ **Tip for interview:**  
Explain it like you’re showing **how a real user interacts** with the system, how **Redis caching optimizes repeated requests**, and how k6 simulates **high concurrency without breaking your laptop**. Then back it up with the numbers you got.

If you want, I can **write a ready-to-read explanation** you can literally say in an interview in under 60 seconds — concise but impressive.

Do you want me to do that?

**You said:**

yes

**ChatGPT said:**

Here’s a concise, interview-ready explanation you can confidently deliver in under a minute:

\*\*“To test my URL shortener, I used k6 to simulate real user traffic. I first loaded all short URLs from a file and had virtual users randomly request them. I disabled automatic redirects so I could validate that each request correctly returns a 302 status and has a Location header. I set stages to gradually ramp up users, sustain load, and ramp down, mimicking real-world traffic patterns.

I also ran tests with and without Redis caching to show its effect on performance. Metrics I monitored include response time, request success rate, and throughput. With caching, 95% of requests completed under 83ms, while without caching, it was around 146ms, clearly demonstrating the performance improvement.

This approach let me safely stress-test hundreds of thousands of requests and generate concrete data to show my system can handle high concurrency efficiently.”\*\*

If you want, I can also **shorten it even further to 30–40 seconds**, perfect for an interviewer who doesn’t want a long explanation