Engineer

Prueba de carga

Prueba que mide un sistema con tráfico simulado

Qué es

Una prueba de carga dirige una cantidad planificada de tráfico simulado a un sistema y mide cómo se comporta mientras lo soporta: cuánto tardan las respuestas, cuántas peticiones fallan, qué tasa de procesamiento sostiene y qué recursos gastan los servidores para ello. El tráfico procede de un script que representa a muchos usuarios a la vez; la herramienta k6 llama usuario virtual a cada uno. Realizar pruebas de carga es la actividad; una prueba de carga es una ejecución concreta, con un perfil, una duración y criterios de aprobación.

El término se usa tanto en sentido amplio como restringido. La documentación de k6, publicada por Grafana Labs, clasifica las pruebas por la carga que aplican. Una prueba de humo ejecuta unos pocos usuarios virtuales para comprobar que el script funciona. Una prueba de carga media simula un día típico en producción. Una prueba de estrés supera esa media; una prueba de resistencia mantiene la carga media durante horas; una prueba de picos aplica una ráfaga repentina y muy intensa durante unos minutos; y una prueba del punto de ruptura sigue aumentando la carga hasta que algo cede. Como «prueba de carga» puede referirse a cualquier prueba que simule tráfico, k6 denomina prueba de carga media a la variante restringida y señala que algunos equipos hablan de prueba de un día de actividad o prueba de volumen. Su guía es franca sobre el vocabulario: «ni siquiera existe consenso sobre los nombres de estos tipos de prueba». Las categorías también son relativas: la carga que somete a estrés a una aplicación es un día corriente para otra.

Una prueba de carga no es una prueba funcional. No pregunta si el proceso de pago suma correctamente la cesta, sino si sigue respondiendo y con qué rapidez mientras lo utiliza un volumen realista de usuarios. Ambas se necesitan: un script que envía peticiones incorrectas a gran velocidad solo mide lo deprisa que el sistema dice que no.

Utilice el concepto en una decisión

Aplique «¿Qué es una prueba de carga?» a una decisión real. Podemos ayudarle a relacionar el concepto con el sistema que está construyendo o comprando. Un equipo que lleva una idea del boceto a un producto al que sus clientes pueden acceder, sin pasarla entre tres proveedores.

Explore nuestros servicios Póngase en contacto →

Compruebe los supuestos detrás del término

Utilice la explicación y los trabajos enlazados para formular preguntas sobre su proyecto. Podemos examinar los supuestos con usted antes de elegir una herramienta. Ingresos multiplicados por 10 con mapas GIS — 10x crecimiento de ingresos impulsado desde el producto cartográfico

Ingresos multiplicados por 10 con mapas GIS Póngase en contacto →

Por qué importa

Todo sistema tiene un nivel de tráfico a partir del cual deja de dar abasto, y la única pregunta es quién lo encuentra primero. Locust, una herramienta de código abierto para pruebas de carga cuyas pruebas son Python puro, ilustra ese comportamiento en su guía de inicio. A medida que la herramienta añade usuarios, llega un punto, alrededor de 20 usuarios en su ejemplo, en el que los tiempos de respuesta crecen tan deprisa que las peticiones por segundo dejan de aumentar aunque sigan llegando más usuarios. La documentación llama al sistema objetivo sobrecargado o saturado. Ese punto de inflexión es la capacidad real del sistema tal como está construido, diga lo que diga el diagrama de arquitectura, y resulta mucho más barato encontrarlo en un ensayo que ante los clientes.

La rampa hacia ese punto de inflexión ya aporta información por sí sola. La guía de k6 sobre pruebas de carga media dice que el primer resultado aparece mientras la carga sigue aumentando: si el tiempo de respuesta empeora al crecer el tráfico y si el sistema falla antes incluso de alcanzar la carga completa. La meseta muestra después si el rendimiento y el uso de recursos se mantienen estables, ya que algunos sistemas se vuelven erráticos precisamente en ese tramo.

Las pruebas de carga también convierten el rendimiento en un criterio obligatorio de aprobación, en vez de una cuestión de opiniones. k6 permite que un script declare umbrales, criterios de aprobación o fallo para sus métricas, como que falle menos del 1 % de las peticiones o que el 95 % de las respuestas llegue en un máximo de 200 ms. La documentación de k6 señala que quienes realizan pruebas suelen usar los umbrales para dejar escritos sus objetivos de nivel de servicio. Si se cumplen todos, k6 termina con el código 0; si falla cualquiera, termina con un código distinto de cero, de modo que un flujo de integración puede rechazar una versión que haya ralentizado el servicio sin que nadie tenga que leer una gráfica.

Cómo se aplica en la práctica

Piense en una tienda que se prepara para una oferta. La mayor parte del tráfico durante la oferta corresponde a personas que no conseguirán el artículo, así que la respuesta que debe resistir es «agotado», y ese es el camino que hay que ensayar primero. La siguiente prueba está escrita en k6. Inicia peticiones a una tasa programada en vez de usar un número fijo de usuarios, sube hasta 2.000 peticiones por segundo en tres minutos, mantiene esa carga durante veinte minutos y vuelve a bajarla. La rampa sigue las indicaciones de k6 para pruebas de carga media: la subida ocupa del 5 al 15 % de la prueba, la meseta dura al menos cinco veces lo que la subida y la bajada aproximadamente lo mismo que esta. Después vendría una prueba de picos, que k6 recomienda para la venta de entradas, los lanzamientos de productos y las ofertas de temporada, casi sin rampa; esta primera demuestra que la respuesta de agotado resiste a un ritmo estable.

```
import http from 'k6/http';

export const options = {
  scenarios: {
    sold_out: {
      executor: 'ramping-arrival-rate',
      startRate: 0,
      timeUnit: '1s',
      preAllocatedVUs: 400,
      stages: [
        { target: 2000, duration: '3m' },
        { target: 2000, duration: '20m' },
        { target: 0, duration: '3m' },
      ],
    },
  },
  thresholds: {
    http_req_failed: ['rate<0.01'],
    http_req_duration: ['p(95)<200'],
    dropped_iterations: ['count<1'],
  },
};

export default function () {
  http.get('https://shop.example/sale/items/4711');
}
```

Ejecútela con `k6 run sold-out.js`. El número de usuarios virtuales es una estimación para la que la documentación de k6 da una fórmula: la mediana del tiempo de iteración multiplicada por la tasa, más un margen. Con 2.000 peticiones por segundo y alrededor de una décima de segundo cada una, hay 200 usuarios virtuales ocupados a la vez, y 400 dejan margen. Si las respuestas se ralentizan, cada iteración dura más, el grupo se agota y k6 registra en `dropped_iterations` cada iteración que se queda sin un usuario virtual libre. Por eso está el tercer umbral: una iteración descartada significa que la prueba no envió la carga que afirma, y k6 señala que el propio sistema puede estar ralentizándose cuando el generador descarta iteraciones. Los otros dos umbrales hacen fallar la ejecución si falla el 1 % o más de las peticiones o si el percentil 95 es de 200 ms o más.

El escenario procede de la nota de este sitio sobre vender la última entrada exactamente una vez. Con un millón de compradores y diez mil unidades, señala, 990.000 peticiones tienen que fallar, por lo que el rechazo debe ser la respuesta más barata que dé el sistema. Su sección sobre demostrar el diseño incluye pruebas de carga de los rechazos: el camino de agotado tiene que seguir siendo rápido cuando representa el 99 % del tráfico, y el borde de la red tiene que absorberlo. Una prueba que solo ensaya una compra correcta mide el 1 % que menos importa para la capacidad.

Dónde surgen los problemas

La prueba espera al sistema que está midiendo. En lo que k6 denomina modelo cerrado, cada usuario virtual solo empieza la siguiente iteración cuando termina la anterior. Si el servidor se ralentiza, los usuarios virtuales también se ralentizan, cae la tasa de llegada y el sistema recibe una prueba más suave que la prevista, un efecto que, según k6, parte de la bibliografía llama omisión coordinada. Los visitantes reales no se esperan unos a otros. Si la pregunta es si un sistema puede soportar una tasa determinada de peticiones, use un ejecutor de modelo abierto, como `constant-arrival-rate` o `ramping-arrival-rate`, que inicia las iteraciones según lo programado sea cual sea el tiempo de respuesta.

El generador de carga es el cuello de botella. Si la máquina que ejecuta la prueba se queda sin usuarios virtuales, CPU o memoria, los resultados describen al generador, no al sistema objetivo. k6 desaconseja `maxVUs` en la mayoría de los casos porque añadir usuarios virtuales durante la prueba consume CPU y memoria y puede sobrecargar el generador y distorsionar los resultados.

Dimensione `preAllocatedVUs` de antemano con unos pocos ensayos locales y vigile `dropped_iterations`. Cuando una máquina no basta, Locust distribuye la carga entre varias con un proceso maestro y muchos trabajadores.

La primera ejecución es la mayor. La documentación de k6 cuenta que miles de sus usuarios han realizado pruebas de carga que hicieron caer rápidamente sus aplicaciones o sus entornos de preproducción. Empiece con una prueba de humo de unos pocos usuarios virtuales para comprobar el script y el entorno; después ejecute una de carga media y solo entonces una de estrés. k6 califica esa secuencia de esencial. Los problemas que aparecen con poco volumen se encuentran mucho más fácilmente antes de que la carga sea alta.

La carga sube y baja como una montaña rusa. Un patrón que sube y baja varias veces desperdicia recursos y dificulta saber qué cambio causó cada síntoma. k6 recomienda una dirección cada vez: subir, mantener una meseta y bajar. Mantenga también el escenario, los datos y el entorno iguales entre ejecuciones para que dos resultados puedan compararse.

Nadie decidió qué significa aprobar. Una ejecución que termina con una página de gráficas y sin criterios se interpreta de manera distinta según quién la mire, normalmente con generosidad. Escriba los umbrales antes de la ejecución a partir del objetivo que debe cumplir el servicio y deje que decida el código de salida. En ejecuciones largas, k6 puede abortar en cuanto se incumple un umbral con `abortOnFail`, y `delayAbortEval` evita que se detenga antes de que haya suficientes muestras para una evaluación significativa.

Un tipo de prueba se toma como respuesta completa. Una prueba de carga media dice poco de una oferta que llega en segundos, y una de picos dice poco de un fallo que solo aparece tras horas de uso constante. k6 expresa claramente que ningún tipo de prueba elimina todo el riesgo y que el perfil de riesgo decide la combinación: un sistema con riesgos derivados de un uso prolongado necesita primero pruebas de resistencia; uno con riesgos derivados de un uso intensivo necesita pruebas de estrés. Un pico también puede activar procesos que nunca se ejecutan en un día corriente, por lo que el script debe incluirlos.

Lecturas adicionales

- Tipos de pruebas de carga — la descripción de k6 de las pruebas de humo, carga media, estrés, resistencia, picos y punto de ruptura, con la carga y la duración típicas de cada una.

- Pruebas de carga media — cómo dar forma a la subida, la meseta y la bajada, y qué interpretar de cada fase.
- Umbrales — los criterios de aprobación y fallo de k6, su sintaxis y cómo un umbral incumplido determina el código de salida.
- Modelos abiertos y cerrados — por qué una prueba cuyos usuarios esperan las respuestas infravalora la lentitud de un sistema.
- Asignación de usuarios virtuales para una tasa de llegada — cuántos usuarios virtuales necesita una prueba de modelo abierto y qué significan las iteraciones descartadas.
- ¿Qué es Locust? — la explicación de la propia documentación de Locust sobre su herramienta de pruebas de carga en Python y su modo distribuido.

Glosario

Elija según sus restricciones reales

Describa el flujo de trabajo, las interfaces existentes y los usuarios que necesitan el cambio. Podemos definir una entrega comprobable y un traspaso que permita a su equipo mantener lo entregado.

Vea cuánto cuesta Póngase en contacto →

Haga práctica la terminología

Pregunte cómo se aplica «¿Qué es una prueba de carga?» a su organización. Basta un ejemplo concreto de su sistema actual para iniciar la conversación.

Explore nuestros servicios

welcome@engineer.company +45 50 10 61 84 Copenhague, Dinamarca

Llévese esta página

Tarjeta de contacto

Creado con esmero y cariño por D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Derechos de autor © 2025 – actualidad · Engineer ApS · Todos los derechos reservados.

Impreso desde https://engineer.company/es/glossary/load-test/