**Engineer**

# ingeniería del caos

experimentar con fallos controlados en un sistema en marcha

## Qué es

La ingeniería del caos consiste en introducir deliberadamente fallos del mundo real en un sistema en funcionamiento, bajo control, para averiguar si los soporta. Principles of Chaos Engineering, el breve documento al que se remite la mayoría de quienes la practican, la define como «la disciplina de experimentar con un sistema para generar confianza» en que puede resistir condiciones turbulentas en producción. La palabra importante es experimentar. Un experimento de caos no contrasta una función con una especificación; plantea una pregunta sobre el sistema completo y acepta la respuesta que obtenga.

Su justificación comienza en los sistemas distribuidos. Cada servicio puede funcionar correctamente y, aun así, las interacciones entre ellos pueden producir resultados que nadie predijo; los sucesos reales poco frecuentes, como un servidor que falla o un enlace de red que se corta, lo empeoran. Los Principles enumeran las debilidades que se ocultan ahí: ajustes de comportamiento alternativo incorrectos cuando un servicio no está disponible, tiempos de espera mal elegidos que multiplican los reintentos, dependencias desbordadas por quienes las llaman y un componente que arrastra a varios más al fallar. Nada de eso aparece al probar cada parte por separado.

Según los Principles, un experimento consta de cuatro pasos:

- Definir el estado estable como una salida medible del sistema que indique un comportamiento normal, como la tasa de procesamiento, la tasa de errores o un percentil de latencia.
- Formular la hipótesis de que el estado estable se mantendrá tanto en un grupo de control como en uno experimental.
- Introducir variables que reflejen sucesos reales: servidores que fallan, discos que se averían o conexiones de red que se cortan.

- Comparar las salidas medidas de ambos grupos y buscar pruebas que contradigan la predicción.

  Cuanto más difícil sea alterar el estado estable, más confianza puede tener el equipo en el sistema. Si se refuta la hipótesis, el equipo habrá encontrado una debilidad cuando todavía es pequeña y hay personas observando, en lugar de durante una caída nocturna.

La herramienta más conocida es Chaos Monkey de Netflix, que termina aleatoriamente instancias de máquinas virtuales y contenedores en producción. Su README la presenta como ejemplo de herramienta que sigue los Principles, y su propósito declarado es conseguir que los ingenieros construyan servicios resistentes a los fallos de instancias.

Utilice el concepto en una decisión

Aplique «¿Qué es la ingeniería del caos?» a una decisión real. Podemos ayudarle a relacionar el concepto con el sistema que está construyendo o comprando. Monitorización y alertas que avisan cuando un fallo aún es pequeño, y cifras que muestran con qué frecuencia se mantiene así.

**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. Primer panel de observabilidad de la empresa — la primera visión de la empresa sobre sus propios sistemas

**Primer panel de observabilidad de la empresa** Póngase en contacto →

## Por qué importa

La redundancia, la conmutación por error y los mecanismos alternativos son promesas sobre lo que ocurre cuando algo se rompe, y una promesa que nunca se ha probado es una hipótesis. Un sistema de reserva que nunca ha recibido tráfico, una política de reintentos que nunca se ha encontrado con una dependencia lenta o una alerta que nunca se ha disparado: todos pueden funcionar, pero averiguarlo con el primer fallo real sale caro. La ingeniería del caos traslada ese descubrimiento a un momento que elige el equipo.

También pone a prueba el sistema, no solo las partes. Las pruebas unitarias y de integración comprueban comportamientos que alguien ya había previsto. Los Principles distinguen directamente ambas cosas: los experimentos de caos verifican que el sistema funciona, en lugar de intentar validar cómo funciona. Por eso se centran en la salida que notaría un usuario, como la tasa de errores y la latencia, no en los detalles internos de un componente.

La evidencia se encuentra en producción. Los sistemas se comportan de forma distinta según su entorno y sus patrones de tráfico, y los Principles sostienen que tomar muestras del tráfico real es la única forma fiable de captar la ruta de las solicitudes. El libro Site Reliability Engineering de Google llega a la misma conclusión por el camino contrario: la carga sintética puede no mostrar dónde fallan realmente los componentes, por lo que, una vez que un servicio dispone de protección contra la sobrecarga, propone ejercitar fallos en una parte limitada del servicio en funcionamiento, con capacidad de reserva preparada por si las protecciones no funcionan y alguien debe conmutar manualmente.

Por último, cambia los incentivos. El README de Chaos Monkey lo expresa con claridad: exponer a los ingenieros a fallos con más frecuencia les da motivos para construir servicios resistentes. Un equipo cuyas instancias pueden terminarse cualquier día laborable lo tiene en cuenta desde el principio, en lugar de descubrir esa carencia cuando una máquina muere sin avisar.

## Cómo se aplica en la práctica

Un experimento de caos se deja por escrito antes de ejecutarlo, como un plan de pruebas. Este corresponde a una tienda en línea inventada y todos los números se han creado para el ejemplo:

```
Experiment:   the payment provider slows down
Steady state: checkout success rate and p99 checkout latency, per 5 min
Hypothesis:   with 300 ms added to calls to the payment provider, the
              experimental group's success rate stays within 0.2 points
              of the control group's
Groups:       2% of checkout traffic in each, assigned at random
Variable:     300 ms of latency on the payment client, nothing else
Abort if:     success rate gap above 1 point for 2 minutes in a row
Window:       30 min, working hours, on-call engineer told in advance
Result:       hypothesis disproved; retries tripled the calls to the
              provider and p99 latency passed the 1 s timeout
Follow-up:    a retry budget per request; rerun the experiment after
```

Aquí, refutar la hipótesis es el resultado útil. La debilidad se encuentra en un 2 % del tráfico durante unos minutos, con una condición de cancelación ya acordada, y la acción posterior es un cambio de ingeniería, no una disculpa.

La propia variable suele introducirse con herramientas corrientes. En Linux, la disciplina de colas netem emula propiedades de redes reales, incluidos retraso, pérdida, duplicación y corrupción. Por ejemplo:

```
# experiment 1: add 100 ms of delay to every packet leaving eth0
tc qdisc add dev eth0 root netem delay 100ms
tc qdisc del dev eth0 root

# experiment 2: drop one packet in a thousand, at random
tc qdisc add dev eth0 root netem loss 0.1%
tc qdisc del dev eth0 root
```

La línea `del` es la importante y debe formar parte de un paso de limpieza que se ejecute tanto si el experimento termina bien como si falla o se interrumpe a la mitad. Netem también admite un `seed` para poder reproducir las pérdidas aleatorias, y su manual advierte que, para obtener resultados de rendimiento TCP realistas, netem debe situarse en la entrada del host receptor.

En cuanto a máquinas completas, Chaos Monkey muestra cómo son unos valores predeterminados prudentes. De fábrica, `enabled` es false y `leashed` es true, lo que significa que las terminaciones solo se simulan y registran. Cuando se activa, trabaja con grupos de instancias, por aplicación, stack o clúster, con un tiempo medio entre terminaciones de dos días por grupo y un mínimo de un día, solo en días laborables, entre las 09:00 y las 15:00 en su zona horaria configurada. Se pueden excluir cuentas, regiones y stacks, y su configuración tiene un lugar para un comprobador que indique si hay una caída en curso; las implementaciones incluidas no hacen nada, por lo que corresponde al operador proporcionar esa comprobación. La lista de pruebas de fallos en producción del libro SRE es igual de concreta: reducir el número de tareas deprisa o despacio, perder de golpe la capacidad equivalente a un clúster y hacer que los backends no respondan, incluidos los no críticos cuyo silencio aún puede bloquear un frontend.

En este sitio, el caso práctico sobre cómo se demostró que el bloqueo de intrusiones funciona en cada ejecución, de septiembre de 2026, no es un experimento de caos en sentido pleno, pero tiene la misma estructura. Cada ejecución de endurecimiento bloquea una dirección del bloque que los estándares reservan para documentación, lee la regla resultante directamente del núcleo en lugar de confiar en el estado que comunica el servicio y retira el bloqueo en un bloque de limpieza que se ejecuta tanto si la comprobación termina bien como si falla. Como la dirección de prueba no pertenece a nadie, un bloqueo residual que sobreviva a una ejecución fallida no puede dejar fuera a una

red real, y la página explica el coste: escribir una regla en producción en cada ejecución es una intervención real en un sistema activo, aceptada porque ya se había demostrado que la alternativa era peor. La implementación de copias de seguridad y pruebas de restauración aplica esa idea a la recuperación en su diseño: un flujo de trabajo mensual restaura un archivo conocido en una ubicación temporal y compara sus bytes. El registro de implementación fechado el 2026-10-27 indica que los temporizadores estaban desactivados sin un repositorio configurado y no documenta ninguna restauración completada en el entorno operativo. El flujo de trabajo existe en código; un ejercicio de recuperación satisfactorio sigue siendo una entrega aparte.

## Dónde surgen los problemas

**Romper cosas sin una hipótesis.** Terminar un servidor aleatorio y observar lo que ocurre es una demostración, no un experimento. Escriba primero el estado estable, la hipótesis y la condición de cancelación. Si el equipo no puede ver su estado estable en un panel en tiempo real, eso es lo primero que debe corregir, y el experimento puede esperar.

**Observar los detalles internos en lugar de la salida.** El uso de CPU de un host o la longitud de una cola dicen poco sobre si los usuarios lo han notado. Los Principles piden una salida medible del sistema completo, como la tasa de procesamiento, las tasas de errores y los percentiles de latencia, porque de eso trata la hipótesis.

**Un alcance de daños sin límites.** Los Principles aceptan cierto impacto negativo a corto plazo, pero hacen responsable al ingeniero de contener sus efectos. Empiece con una instancia o un pequeño porcentaje del tráfico, mantenga un interruptor de cancelación que funcione sin la herramienta que causó el daño y nunca ejecute un experimento mientras ya hay un incidente en curso.

**No salir nunca de preproducción.** Un entorno de preproducción es un primer paso sensato y un mal último paso, porque su tráfico, sus datos y los sistemas vecinos no son los de producción. Pase a una pequeña parte del tráfico real cuando el experimento haya funcionado correctamente antes de pasar a producción, en lugar de dar por definitivos los resultados de preproducción.

**Ejecutarlo una sola vez.** Un sistema que superó un experimento en marzo ha cambiado muchas veces desde entonces. Los Principles consideran que los experimentos manuales exigen mucho trabajo y acaban siendo insostenibles, y

piden automatizarlos para que se ejecuten continuamente y una regresión en la resistencia aparezca la misma semana en que se introduce.

**Limitarse siempre a terminar máquinas.** Los fallos que detienen un proceso de forma clara son el caso fácil, porque algún componente los detecta. Los Principles también enumeran fallos de software como respuestas malformadas y sucesos que no son fallos, como un pico de tráfico o un evento de escalado, y el libro SRE añade un backend que acepta solicitudes y nunca responde, lo que aún puede perjudicar a un frontend cuando las solicitudes tienen plazos de espera largos. Priorice las variables según su impacto probable o su frecuencia.

**Hallazgos que no llevan a ninguna parte.** Una hipótesis refutada es una debilidad con nombre. Asígnele responsable y prioridad en el mismo sistema de seguimiento que todo lo demás y vuelva a ejecutar el experimento una vez publicada la corrección para confirmar que ahora se mantiene el estado estable.

## Lecturas adicionales

- Principios de la ingeniería del caos — la definición, los cuatro pasos de un experimento y los cinco principios avanzados, actualizados por última vez en marzo de 2019.
- Chaos Monkey — la documentación de Netflix para su herramienta de terminación de instancias, que requiere Spinnaker.
- Chaos Monkey: configurar el comportamiento mediante Spinnaker — frecuencia de terminación, agrupación y excepciones, con sus valores predeterminados.
- Chaos Monkey: formato del archivo de configuración — todos los ajustes y sus valores predeterminados, incluidos el modo leashed y el horario de funcionamiento.
- Abordar los fallos en cascada — el capítulo 22 del libro Site Reliability Engineering de Google, incluidas las pruebas hasta el fallo y las pruebas de fallos en producción.
- tc-netem(8) — el manual de Linux del emulador de red utilizado para introducir retraso, pérdida y corrupción.

Glosario

Elija según sus restricciones reales

Describa la ruta de despliegue, los riesgos operativos y quiénes se encargan de las guardias. Podemos definir el alcance de un cambio con comprobaciones, una vía de recuperación y responsabilidades claras.

**Vea cuánto cuesta** Póngase en contacto →

# Haga práctica la terminología

Pregunte cómo se aplica «¿Qué es la ingeniería del caos?» 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/chaos-engineering/