# Chaos Engineering

## Arquitectura del Software

## 2020/2021

Esther González García de Vega – UO269763

Adrián Álvarez Rodríguez – UO265336

# Índice

¿Qué es?....3
Origen....3
¿Cómo funciona?....4
Beneficios....5
Modelos de perturbación....5
Bibliografía....6

## ¿Qué es?

Es un proceso de ingeniería que permite identificar fallos antes de que se produzcan.

Es una solución propuesta para prevenir calamidades, es decir, se entiende que el sistema va a fallar en un momento dado y se proponen distintas soluciones para ello.

Es un método nuevo, ya que hasta ahora se realizaba una batería de pruebas al final del desarrollo. Aquí los fallos se hacen a posta para así descubrir errores en la infraestructura, lo cual es un cambio bastante grande.

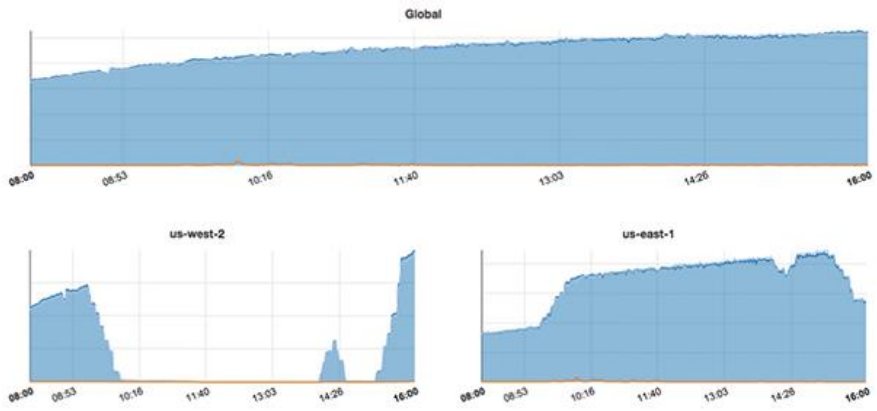
"La ingeniería del caos nació por necesidad como un medio para apuntar y abordar específicamente las vulnerabilidades en los modelos de redes distribuidas a gran escala. Adoptarlo puede requerir un cambio cultural: desde proteger sus aplicaciones a toda costa hasta permitir un pequeño peligro en su vida."

## Origen

Hace unos años (20 de septiembre de 2015), el servicio DynamoDb de AWS (Amazon Web Service) experimentó un problema de disponibilidad en una región, lo que hizo que fallaran más de 20 servicios adicionales de AWS.

Netflix experimentó, entonces, un problema de disponibilidad, pero gracias al uso de Chaos Kong la plataforma estuvo preparada para el incidente.

Este ejercicio de Chaos Kong se realiza de forma regular, lo que nos da la confianza de que, si una región se cae, todavía podremos servir a nuestros clientes.




Ejercicio de Chaos Kong en curso

En la imagen superior podemos observar el tráfico evacuado ese día de la región oeste.

## ¿Cómo funciona?

La mejor manera de probar el sistema a gran escala es seguir los siguientes pasos:

1. Definir un estado estable medible que represente circunstancias normales de uso.

Hay que definir un estado estable de la aplicación al principio del experimento, ya que este estado puede variar según el propósito de la aplicación y el caso de uso. También hay que definir las métricas para evaluar el rendimiento y decidir los resultados aceptables.

Para entenderlo mejor usaremos un ejemplo:

*Se esperaría que todos los compradores de una página web llenen sus carritos y continúen al proceso de pago sin incidentes.*

*Las métricas usadas para evaluar el resultado pueden ser indicadores de clave de rendimiento (KPI), como la tasa de abandono del carrito de compra, o también garantías de latencia y acuerdo del nivel del servicio (SLA) como el porcentaje de tiempo de actividad.*

*Entonces, evaluando estos datos durante un periodo de tiempo específico se obtendría el rendimiento de un estado estable.*

2. Desarrollar una hipótesis de comportamiento que deberán seguir los grupos de control y los de desafío.

Se debe hacer una suposición fundamental sobre el resultado esperado utilizando conjuntos de datos y otra información. Estos datos deben respaldar su hipótesis y depender de resultados medibles. Esto obliga al ingeniero del software a concentrarse en verificar que el sistema funciona, en vez de profundizar en cómo o por qué funciona.

*Siguiendo con el ejemplo de antes, los ingenieros pueden usar datos del proceso de pago que trabaja a niveles óptimos y plantear la hipótesis de que una aplicación permitirá al cliente cargar su carrito desde la página del producto, pasar a la página de pago, tabular el pedido, estimar una fecha de entrega y generar una factura sin fallos, redireccionamientos u otros errores.*

3. En un grupo de desafío, introducir factores de estrés de red realistas (fallos de servidor, de red o conexiones cortadas).

Para ello hay una serie de factores que pueden simularse o introducirse de otro modo en el proceso. Estos factores deben simular problemas reales que pueden ocurrir cuando la aplicación está en uso y, además, priorizarlos por prioridad de que ocurran.

Los problemas pueden ser varios y van desde fallos en el servidor hasta fallos relacionados con el crecimiento de tráfico repentino.

*Por ejemplo, en el caso visto, un factor de estrés sería: en una compra en línea, si una venta es de temporada lo intentará comprar más gente a la vez.*

4. Intentar invalidar la hipótesis con las diferencias entre los grupos de control y de desafío después de todo el caos.

En este paso final tendremos que comparar nuestros resultados frente a los de la hipótesis original. ¿El sistema funciona como se anticipaba, más allá de las expectativas o peor de lo que se esperaba?

Después de reunir todos los datos relacionados todos los miembros del equipo deben reflexionar sobre los resultados y las mejoras o revisiones necesarias.

Este proceso se puede automatizar mediante herramientas como Metasplout, Nmap y VPN que nos permiten cambiar variables y expandir pruebas de forma exponencial.

## Beneficios

Como vemos, son muchos los beneficios de la ingeniería del caos en la implementación. Algunos de los beneficios **para la empresa** propietaria del servicio:

- Mitigación de riesgos comerciales.
- Fomentar la confianza con el cliente.
- Reducir la carga de trabajo de los equipos de trabajo.
- Menor riesgo de pérdidas de ingresos.
- Menores costes de mantenimiento.
- Ingenieros más felices.

Además, habrá beneficios **para el cliente**, como es disfrutar de una mayor disponibilidad de servicios más confiables y menos propensos a interrupciones.

También hay ventajas **para el equipo tecnológico**, como son: reducir los incidentes de fallos, obtener una visión más profunda de cómo funcionan sus aplicaciones y les conducirá a un mejor diseño y a un tiempo medio de respuesta más rápido.

## Modelos de perturbación

Hay un conjunto de herramientas desarrolladas por Netflix, llamadas Simian Army, para probar la confiabilidad, seguridad o resistencia de su infraestructura de AWS e incluye las siguientes herramientas:

- **Chaos Monkey**: es una herramienta inventada en 2011 por Netflix para probar la resistencia de su infraestructura.
  Su funcionamiento es deshabilitando intencionadamente computadoras en la red de producción de Netflix para comprobar cómo responden los sistemas ante la interrupción.
  El código fue lanzado por Netflix en 2012 bajo una licencia de Apache.
- **Chaos Gorilla**: es un conjunto de herramientas para realizar ingeniería del caos nativa de la nube.
  Introduce retrasos artificiales en la capa de comunicación cliente-servidor para simular la degradación del servicio, así mide si los servicios responden adecuadamente.
- **Chaos Kong**: es el mono más alto de la jerarquía. Hace caerse a una región entera de AWS. Es poco común, pero Chaos Kong simula una respuesta y la recuperación ante ese posible problema.
- **Latency Monkey**: introduce retrasos en la comunicación para simular interrupciones o degradaciones de red.
- **Janitor Monkey**: identifica y elimina los recursos que no se usan para así evitar el desperdicio y el desorden.

# Bibliografía

https://www.infoq.com/articles/chaos-engineering-security-networking/

https://netflixtechblog.com/chaos-engineering-upgraded-878d341f15fa

https://www.genbeta.com/desarrollo/chaos-engineering-resistirian-tus-servidores-nube-ataque-ejercito-monos-locos

https://en.wikipedia.org/wiki/Chaos_engineering

https://github.com/dastergon/awesome-chaos-engineering

https://www.pandasecurity.com/es/mediacenter/seguridad/ingenieria-del-caos-para-que-sirve-introducir-fallos-adrede/

https://www.prored.es/el-ejercito-de-simios-de-netflix/