23-4-2021

Ingeniería del caos

UO247134 - UO270543
FACULTAD INGENIERÍA INFORMATICA OVIEDO

Índice

Introducción

En el mundo del desarrollo web se ha tendido cada vez más la descentralización de la aplicación surgiendo paradigmas como la de los microservicios y delegando toda la carga de trabajo de la red a los servidores como Amazon Web Service.

Sin embargo, esto ha llevado a ser excesivamente confiados y desarrollar unos mantras como:

- Las redes son estables:
  - Nunca habrá un evento que las deje fuera de servicio
- Las redes son seguras:
  - No existen las filtraciones de información ni ataque exitoso contra ellas.
- Las redes son homogéneas:
  - Cualquier servidor web ofrecerá el mismo servicio, sin atender siquiera la geolocalización del mismos.
- La latencia es cero:
  - Cualquier petición hecha se cumplirá al instante o de manera tan rápida que será innecesaria cualquier medida de sincronización.
- El ancho de banda es infinito:
  - El servidor aguantará cualquier cantidad de usuarios conectados a la vez.
- La topología nunca cambia:
  - El cómo funciona el servidor siempre será igual, por lo que no habrá posibilidad de que las prestaciones de este varíen.
- Solo existe un admin y es siempre el mismo:
  - No teniendo que preocuparse de poder eliminar o conceder los permisos a este tipo de usuarios de manera proactiva y despreocupándose de las consecuencias de no tener las cuentas adecuadas para la tarea.
- Los costes de transporte son nulos:
  - Da igual que cantidad de información se mueva de cada vez, esto no provocará ninguna variación en la robustez del servicio.

Por principios como este surge la ingeniería del caos como contra medida a posibles eventualidades en el sistema.

Ingeniería del caos

Netflix en 2010 vio estos fallos en las prácticas de las empresas que usaban los mismos servicios que ellos, y se les ocurrió esta metodología que les permitió pasar sin complicación alguna por la caída de servidores del año 2015 de Amazon Web Service.

Esta práctica consiste en poner sobre cierto estrés a los servidores desde los que se trabaja para verificar si se mantiene estable o por lo contrario surgen fallos a la hora de prestar el servicio.

Ejemplos de esto es el añadir una latencia más alta de lo normal o un tráfico de datos desmesurado.

Principios

Está metodología sigue una serie de principios de manera cíclica, ya que es un proceso que se mantiene a lo largo del tiempo para adaptarse a tiempos actuales.

Definición del estado Base

Primero se selecciona el estado que consideramos como ideal, es decir, el volumen de trabajo esperado en un caso normal y el rendimiento que se espera de la aplicación por el cliente promedio en esas condiciones.

Un estado base podría ser el de selección del capítulo siguiente automático cuando termines el capítulo actual que estás viendo.

Elaboración de hipótesis

Se elabora una hipótesis sobre el caso base, una hipótesis es plantear que el servicio se mantendrá estable bajo unas determinadas condiciones que se nos ocurrieron y que son realistas de cara a su funcionamiento, como puede ser la alteración de la latencia o el tráfico, y con información recabada de antemano acerca de cómo sería el comportamiento esperable de esta situación lo redactamos.

Adición de casos de estrés

Cuando ya tenemos la situación planteada toca forzar el caos, a través de herramientas de prueba se somete el servicio a las condiciones de la hipótesis y se comprueba los efectos de estos sobre ella.

Documentación de los resultados

En caso de haber roto nuestra hipótesis descubriéndose bugs, glitches u algún otro malfuncionamiento del sistema bajo estas condiciones se toma nota y se trabaja para elaborar una serie de medidas que puedan evitar el quiebre del sistema.

Esta metodología se ejecutará de manera cíclica con herramientas de automatización, con el fin de poder captar todos esos errores que en fase de desarrollo o bajo condiciones normales de uso no podrían ser visibles.

Herramientas

Simian Army

La Simian Army es el conjunto de herramientas desarrollado por el departamento técnico de Netflix para poner a prueba la capacidad de respuesta a fallos imprevistos de sus servidores.

Actualmente en el GitHub de Netflix podemos acceder a esta herramienta, aunque solo cuenta con tres de los nueve monos que han desarrollado.

El paquete está compuesto por

- Chaos Monkey*
- Latency Monkey
- Doctor Monkey
- Conformity Monkey*
- Janitor Monkey*
- Security Monkey*
- 10-18 Monkey
- Chaos Gorila
- Chaos Kong

La idea tras su creación reside en que no podemos garantizar que un componente, por mucho que este testeado y analizado, no vaya a fallar. Para Netflix esto era un problema serio, ya que un fallo en

un componente que pudiese provocar una reacción en cadena sería un problema para su idea de estar siempre disponibles para sus clientes.

Basándose en lo anterior, Netflix desarrollo su arquitectura de manera que un componente pudiese fallar sin afectar a la disponibilidad de todo su sistema.

A pesar de ello, comprendieron que no era suficiente con desarrollar esta arquitectura tolerante a fallos, que también debían encontrar una manera de poder evaluar su funcionabilidad de manera continuada, para poder garantizar que su sistema es robusto y está preparado para responder y sobreponerse a imprevistos. Así nace el primero de los monos, el Chaos Monkey

Chaos Monkey

La herramienta debe su nombre a que, según los técnicos de Netflix, el sistema debe ser capaz de anteponerse hasta a la situación más inverosímil y destructiva posible, que un mono con ametralladoras se cuele en los centros de datos, y comience a destruirlo todo.

El Chaos Monkey se encarga de inhabilitar temporal y arbitrariamente servidores, con el único objetivo de garantizar que el cliente pudiera seguir con su consumo sin enterarse de estos errores técnicos.

Las pruebas con el Chaos Monkey permitieron a los ingenieros del departamento técnico pudiesen ver de qué manera se comportaba el sistema, para poder anticipar como lo haría en una situación real, desarrollando así herramientas y mecanismos de recuperación automática de estos componentes, para evitar males mayores en una situación real.

Este mono está disponible en GitHub.

Latency Monkey

Una compañía de series en online como Netflix no tardo en darse cuenta de que no bastaba solo con garantizar que los servidores funcionasen bien para que los clientes no tuviesen problemas al consumir su plataforma.

En la era digital, todo tiene que estar listo al instante, si una página web tarda dos o tres segundos más de lo habitual en enseñarnos que ofrece, nos cansaremos y nos iremos a otra.

Latency Monkey es la herramienta que se encarga de introducir retrasos en las comunicaciones, simulando la degradación de estos sistemas en particular, además también es capaz de simular caídas de red.

Conformity Monkey

Es importante que una instancia o aplicación que se añada a nuestros servidores este diseñada, probada e implementada de manera correcta, es decir, que se sigan una serie de buenas prácticas a la hora de implementar nuevas instancias o servicios.

Muchas veces por falta de tiempo, o porque el ingeniero encargado no está familiarizado con el lenguaje que está utilizando para desarrollar dicho componente, se producen estos fallos.

El Conformity Monkey es la herramienta encargada de determinar si un elemento del servidor no cumple con las reglas que se han establecido por la compañía. Si alguna de estas reglas determina que ese componente no es conforme, el mono se pone en contacto con el desarrollador de la instancia, vía correo electrónico, comentando el error, en que norma ha fallado, y que puede hacer para solucionarlo.

El diseño del mono además facilita la inclusión de nuevas pruebas, o modificar las ya existentes.

Este mono está disponible en GitHub.

Doctor Monkey

La integridad de los servidores es importante para garantizar el buen funcionamiento de los servicios que la compañía ofrece. Es importante que estos servidores estén en buen estado.

El Doctor Monkey es el encargado de realizar verificaciones de estado, mediante la supervisión de métricas de rendimiento, como la carga de la CPU, para detectar instancias en mal estado, para el análisis de la causa raíz y la eventual reparación o retiro de esta.

Janitor Monkey

Para garantizar el buen funcionamiento de las instancias y aplicaciones de los servidores, es importante eliminar las partes que no se utilizan, y que están, o pueden estar, consumiendo recursos de manera innecesaria.

El Janitor Monkey es una herramienta que garantiza que los distintos entornos se están ejecutando libres de elementos superfluos, o basura.

La herramienta identifica y elimina estos elementos no utilizados, evitando así el desperdicio de recursos.

Este mono está disponible GitHub

Security Monkey

Garantizar la seguridad de las comunicaciones, y la seguridad dentro de los servidores y aplicaciones de estos, es importante para evitar ataques, y por tanto, perdidas para la compañía o incluso clientes de esta.

Security Monkey es una herramienta desarrollada para encontrar violaciones de seguridad e identificar vulnerabilidades dentro del sistema. Está preparada, además, para eliminar cualquier elemento ofensivo o de riesgo dentro del sistema.

Para garantizar la seguridad en las comunicaciones revisa los certificados SSL (Los protocolos de seguridad de la capa de transporte)

Y al ser Netflix una compañía de distribución de medios digitales también es necesario comprobar que los contenidos cumplen con el DRM, para evitarle problemas a la compañía

10-18 Monkey

Netflix es una compañía disponible en muchas regiones distintas, por lo que es importante que la plataforma y sus contenidos, estén adaptadas, internacionalizadas y localizadas correctamente.

10-18 Monkey debe su nombre a l10n (Localization) y i18n (Internationalization), y se encarga de detectar problemas dentro de la configuración de los elementos a los que pueden acceder al mismo tiempo clientes de distintas regiones, con diferentes idiomas, conjuntos caracteres, husos horarios...

Chaos Gorila

Netflix pensó que no serviría de nada todo el esfuerzo que estaban empleando en hacer su arquitectura de servidores robusta, si un agente externo, como los centros de datos de AWS, se caen o dejan de estar disponibles.

Amazon Web Services o AWS es una colección de servicios de computación en la nube, que en conjunto forman una plataforma que puede ser utilizada por empresas que quieran utilizar estos servicios.

Chaos Gorila surge como una evolución del Chaos Monkey, pero este simula la caída de una o varias zonas de disponibilidad de los centros de datos de una región de AWS.

Al igual que el Chaos Monkey, estas pruebas ayudan a los ingenieros de Netflix a anticipar y preparar el servidor ante la posibilidad de estas caídas del servicio AWS en algunas zonas. Previniendo así problemas para sus clientes.

Chaos Kong

Es la herramienta más extrema de todas. Los técnicos de Netflix consideraron necesario que debían estar preparados para la caída de, no solo un par de centros de datos de AWS, si no para toda una región, es decir, que se caigan todas las zonas de disponibilidad de Europa, por ejemplo, o que deje de estar disponible la nube de manera temporal.

Chaos Kong simula esta caída masiva de la nube de AWS, y de nuevo preparó los servidores y centros de datos de Netflix para esta situación tan extrema.

En 2015 el servicio DynamoDB de Amazon experimentó un problema de disponibilidad en su región US-EAST-1. Esa inestabilidad provocó que fallaran más de 20 servicios de AWS adicionales que dependen de DynamoDB. Algunos de los sitios y aplicaciones más importantes de Internet no estuvieron disponibles de forma intermitente durante un período de seis a ocho horas ese día.

En cambio, Netflix estaba preparado para este imprevisto gracias a utilizar Chaos Kong para “entrenar” sus centros de datos ante este tipo de situaciones, al ejecutar experimentos de forma regular que simulan una interrupción regional, pudieron identificar a tiempo cualquier debilidad sistémica y solucionarla, por lo que, en lugar de estar tantas horas inactivo, solo estuvieron un corto periodo de tiempo.

Litmus Chaos

LitmusChaos es un conjunto de herramientas para realizar ingeniería de caos nativa de la nube.

Estas herramientas sirven para orquestar el caos en Kubernetes para ayudar a los SRE a encontrar debilidades en sus implementaciones.

Los equipos de SRE son los encargados de utilizar el software para gestionar los sistemas, resolver los problemas y automatizar las tareas operativas.

Estos equipos pueden utilizar Litmus para ejecutar experimentos de caos inicialmente en el entorno de ensayo y, finalmente, en producción para encontrar errores y vulnerabilidades.

Litmus cuenta además con la ChaosHub, un repositorio lleno de experimentos y pruebas que implementar en tus aplicaciones para aplicarles caos.

Automatización

Ejecutar los experimentos manualmente es muy trabajoso e incluso puede llegar a ser insostenible. Automatizar los experimentos es importante para poder ejecutarlos continuamente, reduciendo asi el riesgo de nuevos errores y vulnerabilidades.

Kubernetes es un sistema open source para automatizar el despliegue, de aplicaciones en contenedores.

Utilizar Kubernetes es una buena manera de automatizar el uso de herramientas de ingeniería del caos. LitmusChaos, por ejemplo, es completamente compatible.

En cuanto a la Simian Army, y el conjunto de herramientas del caos de Netflix, surge ChAP (Chaos Automation Platform)

Un análisis automatizado y continuo se conecta al mismo sistema que se usa para hacer análisis.

Antes de ChAP, se podía identificar y corregir una vulnerabilidad, pero luego, después de corregirla, el entorno podía cambiar, o podría ocurrir algo que lo modificase durante la ejecución de la prueba, y esta no pararía, lo que podría llevar a un incidente.

Con ChAP si se detecta un error durante la prueba, esta se parará automáticamente, identificando y reportando el error.

Beneficios

La ingeniería del caos proporciona confianza en nuestros sistemas, para nosotros y para los posibles consumidores de los servicios que proveamos.

Utilizar caos nos garantiza que estaremos preparados ante imprevistos, y que sabremos responder ante ellos.

La ejecución periódica de nuestras pruebas nos ayuda a mejorar sistemáticamente nuestros servidores, ante posibles amenazas o problemas imprevistos.

No es complicado de implementar, ya que las herramientas ya existen y en su gran mayoría son de uso libre.

Son un buen complemento a los clásicos test de código, yendo un poco más lejos que estos, ya que también nos permiten evaluar cosas como la calidad del código, o si se adapta a las normas o guías de estilo de nuestra empresa.