




# Chaos Engineering

Esther González García de Vega, UO269763

Adrián Álvarez Rodríguez, UO265336

# Índice

1. ¿Qué es?
2. Motivación
3. ¿Por qué?
4. Origen
5. ¿Cómo funciona?
6. Beneficios
7. Herramientas
8. Conclusiones
9. Referencias



¿Qué es?

# ¿Qué es?

- Es una disciplina que nos permite identificar fallos antes de que se conviertan en pérdidas de servicio.





# Motivación

# Motivación

- En 2010, Netflix se dio cuenta que la mejor defensa contra fallos inesperados es **fallar a menudo**; causando fallos de forma deliberada, forzamos a que nuestros servicios sean construidos con alta **resiliencia**.



¿Por qué?

# ¿Por qué?

## El fallo es **inevitable**:

- El software tiene bugs
- El hardware falla
- Hay cortes eléctricos
- Los humanos cometemos errores
- Los humanos somos limitados
- La actual aproximación del testing no es suficiente







Origen

# Origen

- 2010: Creación del mono del caos



## Origen

- 2010: Creación del mono del caos
- 2011: Creación del ejército de simios



# Modelos de perturbación



# Origen

- 2010: Creación del mono del caos
- 2011: Creación del ejército de simios
- 2012: el Código del mono del caos fue liberado
- 2014: Netflix crea oficialmente el rol de Chaos Engineer y crean FIT
- 2017: Chaos Automation Platform





Google

 Microsoft  
Azure

**NETFLIX**

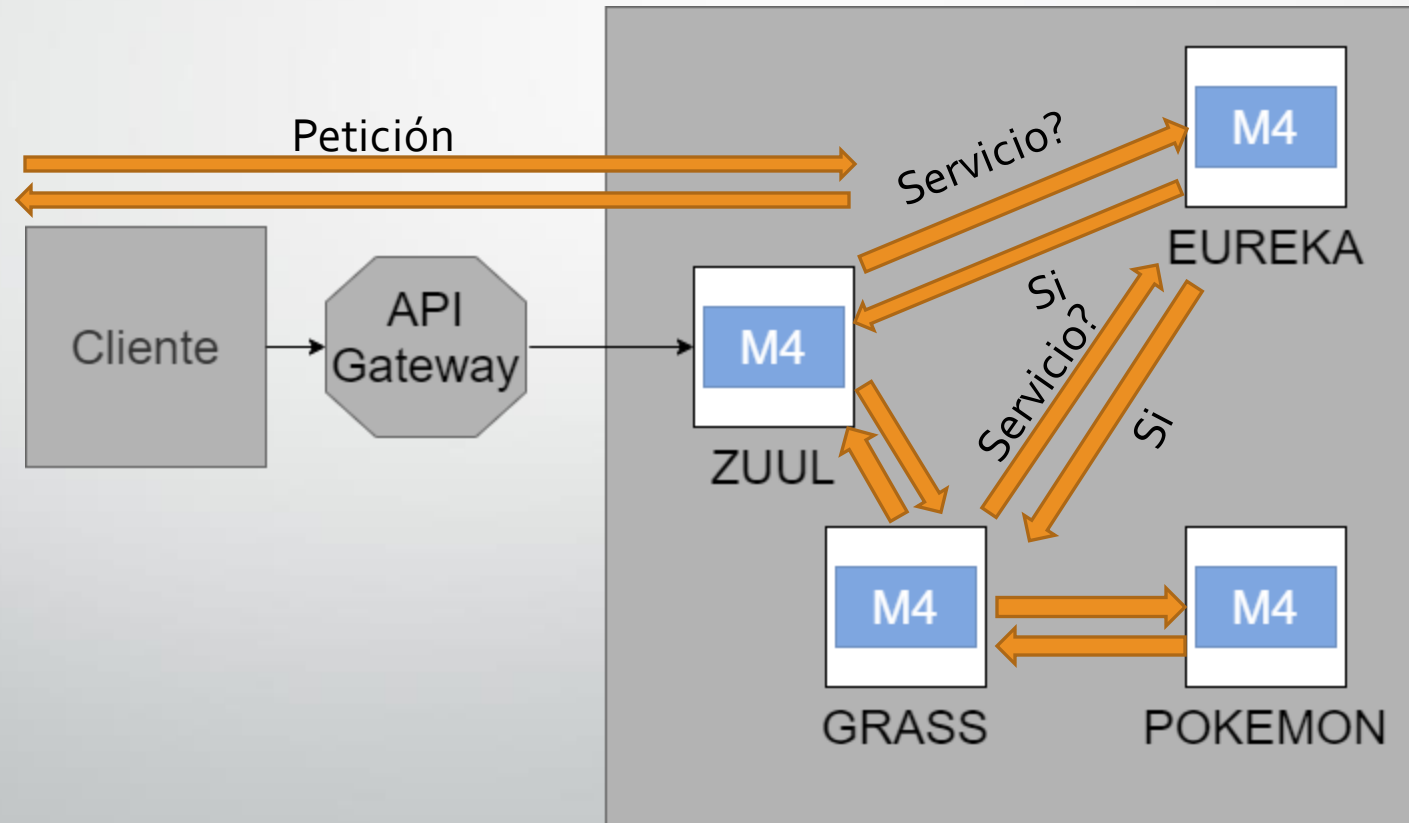
LinkedIn

Uber



¿Cómo funciona?

# Ejemplo





# Primer paso – Definir el estado estable

- Sera nuestro punto de partida para realizar el experimento
- Es la respuesta que esperamos cuando nuestro sistema responda correctamente

## Segundo paso – Hipótesis

- Tiene que ser una hipótesis positiva
- En caso de fallo parcial o total el usuario no notara la incidencia, ni error. Solo percibirá que no encuentra Pokémons
- No realizar el experimento si no tenemos confianza de que nuestro sistema lo soporte
- Tener claro el objetivo del experimento

## Tercer paso – limitar el alcance

- Escoger un entorno que nos garantice una afectación mínima
- Si es posible, limitar el número de usuarios afectados

## Cuarto paso – Identificar las métricas

- Identificar las métricas que se pueden ver afectadas
- Monitorizar el experimento
- En local se imita el tráfico generado en producción

## Quinto paso – Avisar

- Avisar a todas las áreas que se pueden ver afectadas
- A poder ser, realizar el experimento en horario de trabajo

## Sexto paso – Realizar el experimento

- La primera ejecución siempre debe ser manual

## Séptimo paso – Analizar los resultados

- Si es necesario, se repite el experimento, hasta obtener una conclusión

## Octavo paso – Incrementar el alcance

- De local a preproducción
- De preproducción a producción
- De una parte de los usuarios del sistema a todos los usuarios



## Noveno paso – Automatizar

- Ahora que nuestro sistema es **resiliente**, ya podemos automatizar la prueba
- Probar que el experimento sigue pasando cada cierto tiempo
- No siempre es necesario que este en producción, puede bastar con que este en preproducción



# Beneficios

# Beneficios

- ✓ Mitigación de riesgos comerciales
- ✓ Mayor confianza con el cliente
- ✓ Menor riesgo de pérdidas de ingresos
- ✓ Menores costos de mantenimiento
- ✓ Ingenieros más felices

Herramientas



**Gremlin**



# Conclusiones

# Conclusiones

- Mejora la resiliencia de tu sistema
- Mejora la experiencia del cliente
- Mejora la mantenibilidad del sistema
- Evita posibles fallos en producción de un sistema informático
- Equipos mejor entrenados

# Referencias

