# ON DESIGNING AND DEPLOYING

# INTERNET-SCALE SERVICES

## INTRODUCCIÓN

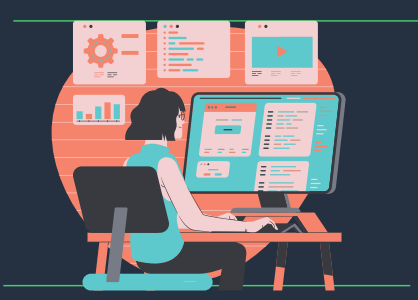
En el presente articulo se habla de las buenas prácticas para el diseño y desarrollo de servicios de operaciones amigables y Sistemas de software centrados en datos a gran escala

## DISEÑO GENERAL DE LA APLICACIÓN

Cuando los sistemas fallan, existe una tendencia natural a mirar primero a las operaciones, ya que ahí es donde el realmente ocurrió el problema

Es normal observar primero las operaciones
Los problemas de operaciones se originan en el diseño y desarrollo

## CONCEPTOS BÁSICOS DE OPERACIONES AMIGABLES



## DISEÑO PARA FALLOS

Todo el servicio debe ser capaz de sobrevivir a la falla sin administración humana e interacción.

## SEGMENTO DE HARDWARE BÁSICO

Todos los componentes de el servicio debe apuntar a un hardware básico

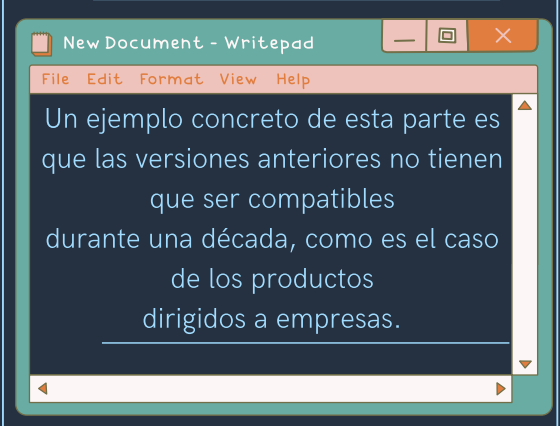
## REDUNDANCIA Y RECUPERACIÓN DE FALLAS

Diseñar un servicio tal que cualquier sistema pueda chocar (o ser derribado para el servicio) en cualquier tiempo.

## MULTIUSUARIO

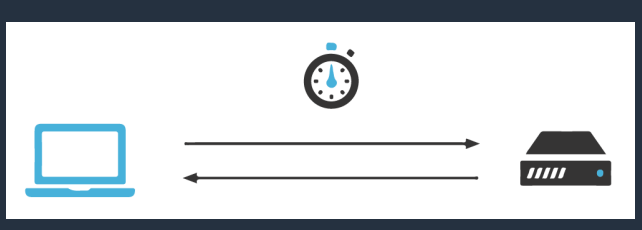
Todo tiene que ser compartido por varios usuarios al mismo tiempo.

## SOFTWARE DE UNA SOLA VERSIÓN




## GESTIÓN AUTOMÁTICA Y APROVISIONAMIENTO

- Ser redundante y reiniciable
- Apoyo de la distribución geográfica
- Instalación y aprovisionamiento automático
- Mantener la implementación simple
- Hacer fallar los servicios regularmente



## GESTIÓN DE LA DEPENDENCIA

- Suponer la latencia
- Aislar las fallas
- Mantener la implementación simple
- Hacer fallar los servicios regularmente

Seminario de Sistemas Distribuidos
De Jesús Moreno - Herrera Godina - Sanchez Torres

# CICLO DE LANZAMIENTO DE PRUEBAS

## CALIDAD

La mayoría de los servicios tienen al menos una laboratorio de pruebas que es tan similar a la producción como (económicamente) posible y todos los buenos equipos de ingeniería utilizan cargas de trabajo de producción para impulsar los sistemas de prueba de manera realista.

## REGLAS

1.El sistema de producción ha de tener suficiente redundancia
2.Los errores deben ser detectados
3.Debe ser posible revertir rápidamente todos los cambios

## MEJORES PRACTICAS

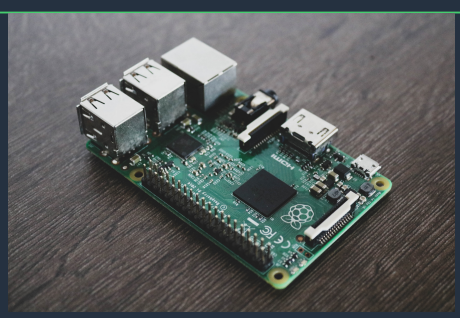
- Usar datos de producción para encontrar problemas.
- Invertir en ingeniería.
- Reversión de la versión de soporte
- Mantener la compatibilidad hacia adelante y hacia atrás.
- Implementación de un solo servidor
- Prueba de estrés por carga

## MEJORES PRACTICAS

- Realizar pruebas de capacidad y rendimiento
- Cree e implemente de manera superficial e iterativa
- Prueba con datos reales.
- Ejecutar pruebas de aceptación a nivel del sistema
- Probar y desarrollar en entornos completos.

## SELECCIÓN Y ESTANDARIZACIÓN DE HARDWARE

- Determinar qué hardware es actualmente el mejo
- Hacer la calificación del hardware y la implementación del
- software una vez que el hardware esté instalado



## PLANIFICACIÓN DE OPERACIONES Y CAPACIDAD

- Responsabilizar al equipo de desarrollo.
- Sólo borrado temporal.
- Seguimiento de la asignación de recursos
- Realice un cambio a la vez

## MEJORES PRACTICAS

- Use solo SKU estándar
- Compre bastidores completos
- Escribir en una abstracción de hardware
- Abstraer la red y el naming

## AUDITORÍA, SUPERVISIÓN Y ALERTA

- En cada cambio hay que tomar un registro de quien y cuándo se hizo.
- Cada alerta debe representar un problema.

## AUDITORÍA, SUPERVISIÓN Y ALERTA

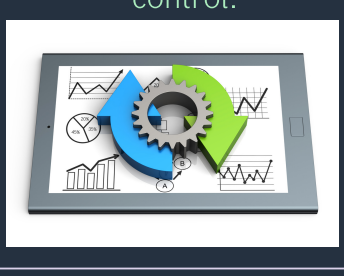
- Para obtener niveles de alerta correctos: 1) relación de tickets de alerta a problema (con un objetivo de casi uno) y 2) número de problemas de salud de los sistemas sin las alertas correspondientes (con un objetivo de casi cero).

## AUDITORÍA Y SUPERVISIÓN Y ALERTA

Puntos importantes a considerar:

- Instrumenta todo
- Los datos son el activo más valioso
- Visión de servicio al cliente.
- Disponer de suficientes datos de producción.

- Todos los errores deben ser procesables.
- Permitir un diagnostico rapido de problemas de producción.
- Exponer la información para su control.



Seminario de Sistemas Distribuidos
De Jesús Moreno - Herrera Godina - Sanchez Torres

## DEGRADACIÓN GRADUALY CONTROL DE ADMISIÓN

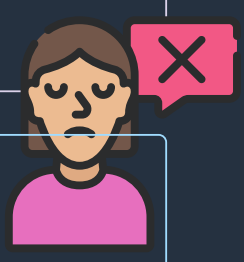
El servicio debe ser capaz de degradarse con elegancia y controlar las admisiones.
En general, un ''big red switch'' es una acción diseñada y probada que se puede tomar cuando el servicio ya no es capaz de cumplir su función.

## BOTÓN ROJO

El concepto de un interruptor rojo grande es mantener el progreso del procesamiento vital mientras se elimina o se retrasa alguna carga de trabajo no crítica.

## CONTROL DE ACCESO

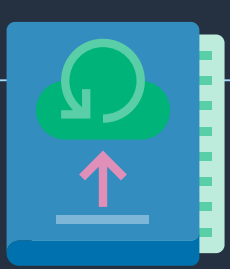
Si la carga actual no puede ser procesada en el sistema, introducir más carga de trabajo en el sistema sólo asegura que una sección mayor de la base de usuarios va a tener una mala experiencia.



## AUDITORÍA, SUPERVISIÓN Y ALERTA

- Para obtener niveles de alerta correctos: 1) relación de tickets de alerta a problema (con un objetivo de casi uno) y 2) número de problemas de salud de los sistemas sin las alertas correspondientes (con un objetivo de casi cero).

Si el sistema falla y se cae, hay que ser capaz de volver a ponerlo en marcha poco a poco asegurándose de que todo está bien.



## PLAN DE COMUNICACIÓN CON LOS CLIENTES Y LA PRENSA

Los sistemas fallan. Las comunicaciones deben estar disponibles a través de múltiples canales

## COBERTURA DE LA PRENSA

Cada tipo de catástrofe debe contar con un plan sobre a quién llamar, cuándo hacerlo y cómo gestionar las comunicaciones.


## AUTOPROVISIONAMIENTO Y AUTOAYUDA DEL CLIENTE

El autoaprovisionamiento del cliente reduce sustancialmente los costes y también aumenta la satisfacción del cliente.

Seminario de Sistemas Distribuidos
De Jesús Moreno - Herrera Godina - Sanchez Torres