Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

Clase 24
Pruebas de sistemas / *Deployment*

IIC3745 – Testing

Rodrigo Saffie

rasaffie@uc.cl

30 de noviembre de 2020

Pruebas de sistemas

- Pruebas de la aplicación en su ambiente de ejecución
- Validan los requisitos no funcionales, como por ejemplo:
  - Rendimiento
  - Seguridad
  - Escalabilidad
  - Resiliencia
  - Portabilidad
- Sirven para asegurar un *Service Level Agreement*
  - Auth0

Métricas

- Una métrica son datos procesados que expresan numéricamente el rendimiento sobre un criterio
  - Ejemplo: *Coverage*
- Sirven para:
  - tener respaldo cuantitativo sobre un criterio
  - comparar la efectividad de distintas estrategias

Métricas en *software*

- Ejemplos de métricas:
  - Tiempos de respuesta
  - Flujo (*throughput*): solicitudes por minuto (o segundo)
  - Uso de RAM / CPU
- New Relic
- Scout

Pruebas de rendimiento

- Se busca probar que el sistema cumpla con los requerimientos de desempeño
  - Por ejemplo, tiempos de respuesta, uso de RAM/CPU/*bandwidth*
- Ejemplos:
  - **Pruebas de carga**: asegurar el comportamiento del sistema bajo ciertas condiciones de uso
  - **Pruebas de estrés**: probar cómo responde el sistema dada una carga mayor para la cual fue diseñado
- Herramientas:
  - Loader
  - BlazeMeter
  - JMeter

Pruebas de escalabilidad

- Pruebas las políticas de escalabilidad para los sistemas bajo situaciones controladas
- Tipos de escalabilidad
  - **Vertical**: reemplazar los componentes por otros con mayor capacidad
  - **Horizontal**: aumentar la cantidad de componentes para que ejecuten el mismo proceso en paralelo

Pruebas de seguridad

- Ataques simulados para detectar fortalezas y debilidades de los sistemas (*white hat hackers*)
- *Bug bounty programs*: recompensas por reportar vulnerabilidades
  - Bugcrowd
  - GitHub
  - Google
- Se deben reportar con extremo cuidado
  - Si se reportan como *bug* funcional se corre el riesgo que otras personas se aprovechen de la vulnerabilidad

Pruebas de resiliencia

- Probar la capacidad de un sistema de gestionar y recuperarse sobre errores inesperados
  - Aún cuando ocurran errores el sistema debería ser capaz de proveer un nivel aceptable de servicio.
- Netflix Chaos Monkey

Pruebas de portabilidad

- Se prueba la capacidad del sistema de ejecutar correctamente en distintos ambientes.
- browserling
- Firebase Test Lab

Políticas de *deployment*

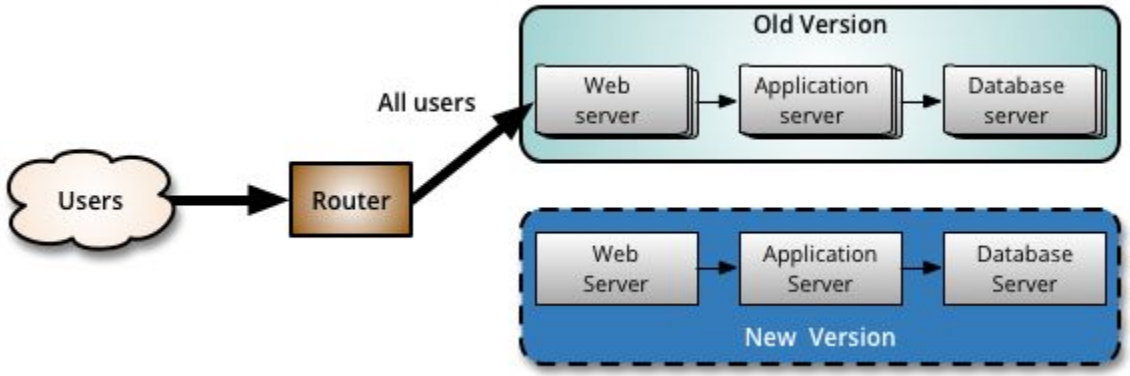
- Son los pasos necesarios para disponibilizar el *software* a sus usuarios reales.
- Los principales atributos a considerar son:
  - *Downtime*
  - Costo en recursos (servidores)
  - Capacidad de *rollback*
- *Six Strategies for Application Deployment*
  - *Recreate*
  - *Blue/Green*
  - *Canary*

Recreate

- Consiste en apagar el servicio para luego realizar la instalación de la nueva versión y volver a disponibilizar.
- Es la manera más simple de realizar *deploy*, pero incurre en *downtime* del servicio, lo que impacta directamente a los usuarios.

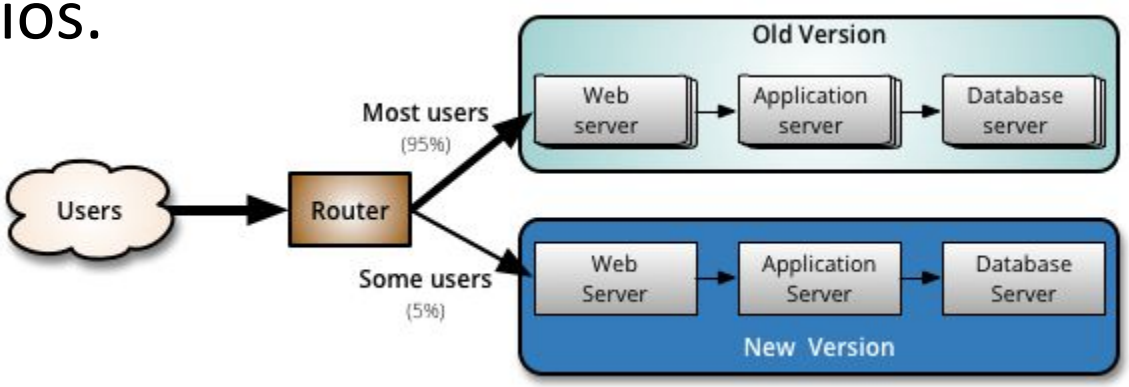
Blue/Green

- Se monta todo un sistema con la nueva versión en paralelo al anterior.
- Una vez que el nuevo sistema está listo para recibir solicitudes, se redirige todo el tráfico a este.
- Si bien esta estrategia tiene mayor costo en recursos, no incurre en *downtime* y facilita la capacidad para realizar *rollback*.



Canary

- Se monta todo un sistema con la nueva versión en paralelo al anterior.
- Una vez que el nuevo sistema está listo para recibir solicitudes, se redirige parte del tráfico a este (con usuarios seleccionados y controlados).
- Similar a *Blue/Green* con la diferencia de que si se detectan problemas estos no impactan a todos los usuarios.



CI / CD

- ***Continuous Integration (CI)***: una práctica de desarrollo de código que requiere que el código se integre de manera constante a un repositorio central.
  - Es más fácil y menos costoso realizar pequeñas integraciones continuamente que viceversa.
- ***Continuous Deployment (CD)***: una práctica de *deployment* en que se realiza *deploy* a producción de manera automática cada vez que se integra una nueva funcionalidad.
  - Depende fuertemente de la automatización y calidad de las pruebas.

Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

Clase 24
Pruebas de sistemas / *Deployment*

IIC3745 – Testing

Rodrigo Saffie

rasaffie@uc.cl

30 de noviembre de 2020