Desarrollo de Software I

Facultad de Ingeniería
Escuela de Ingeniería de Sistemas y Computación

Mauricio Gaona
mauricio.gaona@correounivalle.edu.co
Profesor

2024-II

Desarrollo de Software I

Clase anterior

Taller Conceptos básicos del desarrollo de una aplicación con Django.

Estimación de costos y tiempo requerido para el desarrollo de una aplicación usando metodologías ágiles.

Conceptos básicos sobre el World Wide Web

Tareas:
Investigar que son variables de sesión en una aplicación web.
Investigar la diferencia entre GET y POST.

Tareas:
Investigar que son variables de sesión en una aplicación web.

Las **variables de sesión** en una aplicación web son una forma de almacenar información del usuario durante la interacción con la aplicación. Estas variables permiten que los datos persistan entre las diferentes solicitudes HTTP (que son sin estado) hechas por el usuario durante su sesión.

Características principales de las variables de sesión

1. Persistencia temporal: Los datos de la sesión se mantienen mientras dure la sesión del usuario, que normalmente termina cuando el navegador se cierra o después de un tiempo de inactividad.
2. Almacenamiento del lado del servidor: A diferencia de las cookies, las variables de sesión generalmente se almacenan en el servidor, y solo se almacena una referencia (identificador de sesión) en el navegador del usuario.
3. Seguridad: Como los datos sensibles se guardan en el servidor y no en el cliente (como en el caso de las cookies), las sesiones son más seguras frente a manipulaciones directas por parte del usuario.
4. Uso común: Se utilizan para gestionar la autenticación de usuarios, almacenar preferencias, o mantener el estado de una aplicación entre varias páginas (por ejemplo, los productos en un carrito de compras).

Tareas:
Investigar que son variables de sesión en una aplicación web.

Funcionamiento general

1. El servidor crea una sesión para el usuario cuando este inicia una interacción.

2. El servidor asigna un identificador único para esa sesión.

3. El identificador se almacena en el navegador del usuario en forma de una cookie.

4. Cada vez que el usuario hace una solicitud, el identificador se envía al servidor para acceder a las variables de sesión asociadas.

Este mecanismo es esencial para mantener el contexto de la aplicación web y permite mejorar la experiencia del usuario sin necesidad de solicitar repetidamente la misma información.

Tarea: Como crear una variable de sesión en Django

Tareas:

Investigar la diferencia entre **GET** y **POST**.

La diferencia entre **GET** y **POST** en el contexto de las solicitudes HTTP, que son comúnmente usadas en aplicaciones web, es fundamental para el envío de datos entre el cliente (generalmente un navegador) y el servidor. Ambas son métodos HTTP que indican diferentes tipos de operaciones sobre los datos, y se utilizan dependiendo del objetivo de la solicitud.

GET

1. Función: El método GET se utiliza para **solicitar** datos de un servidor. Es decir, se usa para recuperar información (por ejemplo, una página web o los datos de un formulario).

2. Datos en la URL: Los datos que se envían mediante GET se adjuntan a la URL en forma de **parámetros de consulta** (query strings). Estos parámetros aparecen después de un signo de interrogación en la URL.

Ejemplo: **https://ejemplo.com/buscar?producto=libro&categoria=tecnologia**

1. Visible en el navegador: Dado que los datos forman parte de la URL, son visibles en la barra de direcciones del navegador, lo cual puede no ser adecuado para información sensible.

2. Limitación de tamaño: Los datos enviados a través de una solicitud GET tienen una limitación de tamaño, ya que las URLs tienen un límite de longitud (aproximadamente 2000 caracteres en la mayoría de los navegadores).

3. Idempotente: Las solicitudes GET son idempotentes, lo que significa que realizar la misma solicitud repetidamente no debería tener efectos secundarios. Por ejemplo, acceder a una página varias veces no cambia su estado.

4. Caché: Las solicitudes GET pueden ser **almacenadas en caché** por el navegador, lo que significa que, si se solicita el mismo recurso varias veces, el navegador puede usar la copia almacenada en lugar de hacer una nueva solicitud al servidor.

Tareas:
POST.

POST

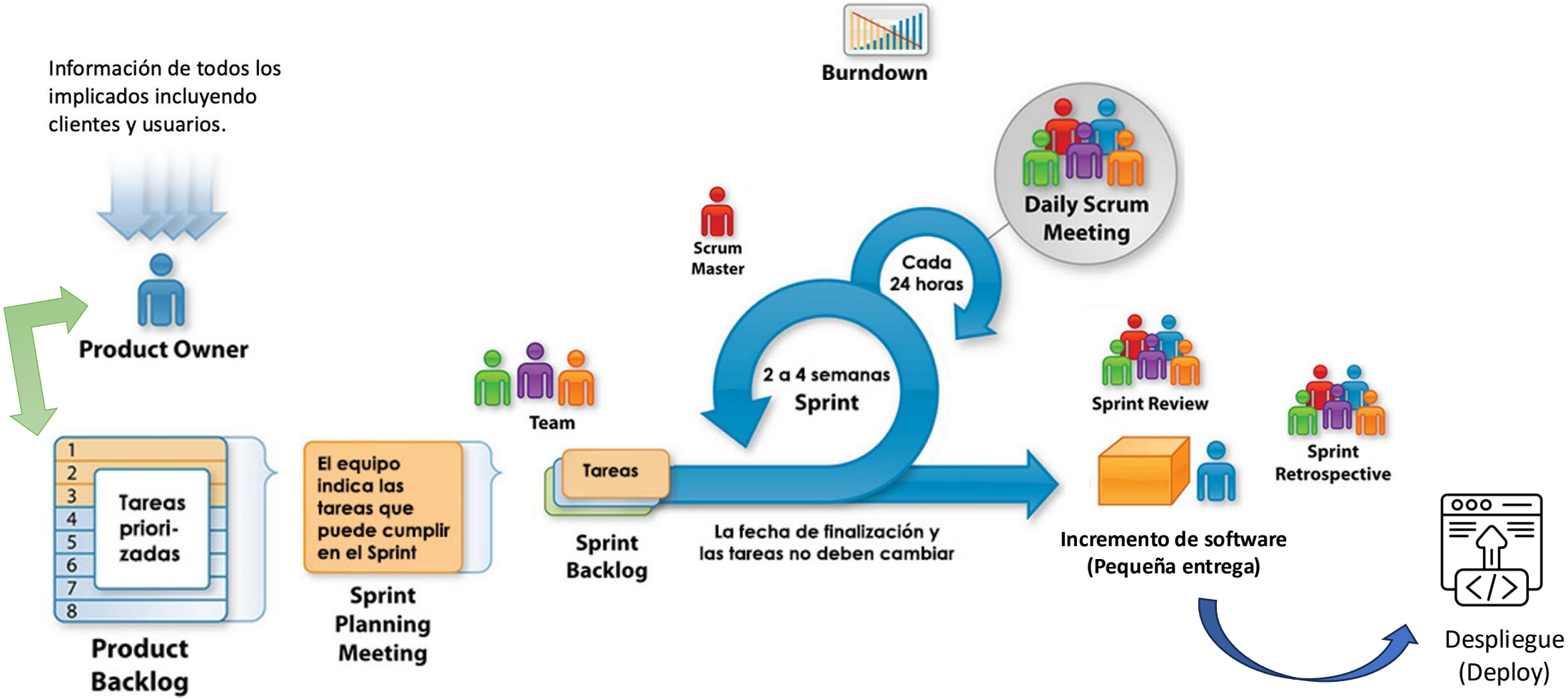
1. Función: El método POST se utiliza para **enviar datos** al servidor, generalmente para procesar y realizar alguna acción (como enviar un formulario de registro, crear una cuenta o enviar comentarios).
2. Datos en el cuerpo de la solicitud: Los datos enviados con POST no se adjuntan a la URL, sino que se envían dentro del **cuerpo de la solicitud**. Esto los hace invisibles en la barra de direcciones.
3. Adecuado para datos sensibles: Debido a que los datos no se muestran en la URL, el método POST es más adecuado para enviar información confidencial, como contraseñas o detalles de tarjetas de crédito.
4. Sin limitación de tamaño: No hay una limitación estricta en cuanto a la cantidad de datos que se pueden enviar en una solicitud POST, aunque el servidor puede tener límites de tamaño configurados.
5. No idempotente: Las solicitudes POST no son idempotentes, lo que significa que realizar la misma solicitud varias veces puede producir diferentes resultados. Por ejemplo, enviar un formulario de compra varias veces puede generar múltiples pedidos.
6. No almacenada en caché: Las solicitudes POST no se almacenan en caché, lo que significa que cada vez que se envían, una nueva solicitud llega al servidor.

Tareas:

GET vs POST.

Característica	GET	POST
Función	Recuperar datos	Enviar datos al servidor
Datos enviados	A través de la URL (query string)	En el cuerpo de la solicitud
Visibilidad	Visible en la barra de direcciones	Invisible para el usuario
Tamaño de datos	Limitado por la longitud de la URL	No tiene limitación significativa
Uso para datos sensibles	No recomendado (datos visibles en la URL)	Recomendado (datos no visibles en la URL)
Idempotente	Sí, puede repetirse sin cambiar el estado	No, puede cambiar el estado del servidor
Caché	Puede ser almacenada en caché	No almacenada en caché
Repetibilidad	Seguro repetir una solicitud	No es seguro repetir (puede generar acciones múltiples)

Metodología ágil Scrum



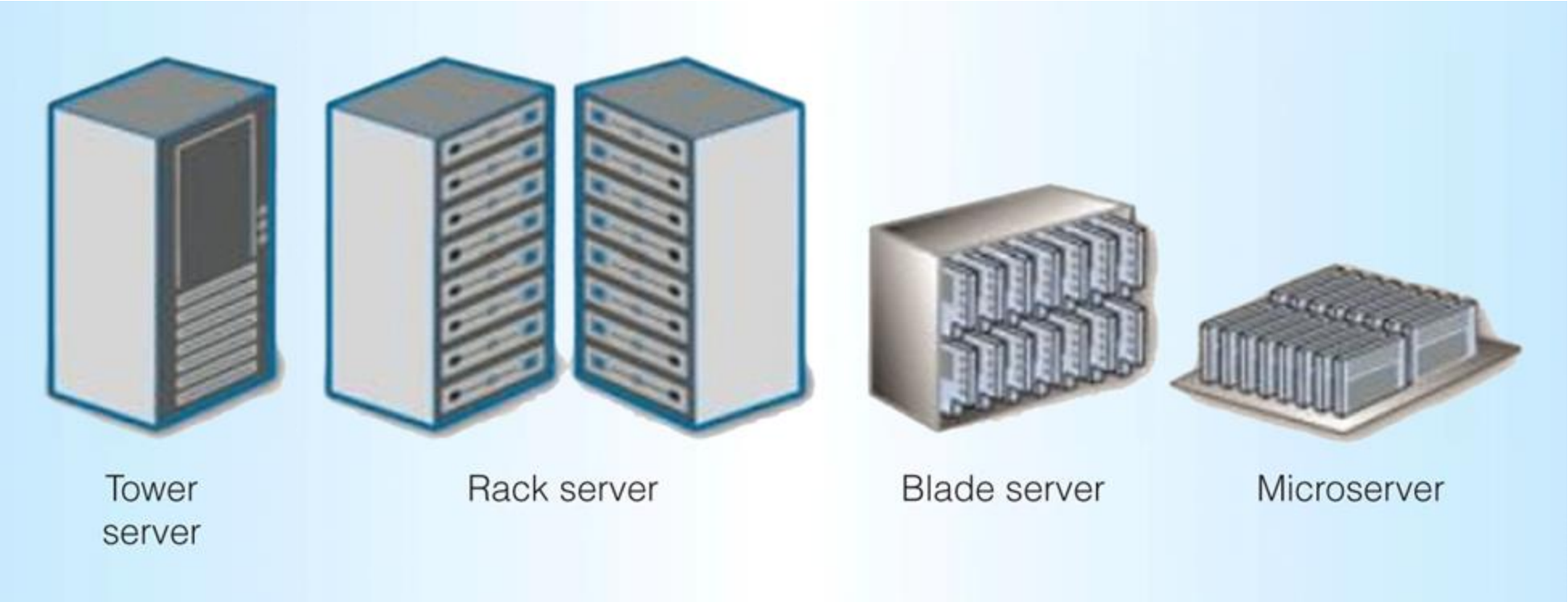
Modelo de Despliegue

Representa la plataforma o el conjunto de servidores requeridos por una aplicación de software para ponerla en funcionamiento para ser usada por los usuarios finales.

Modelo de despliegue

Servidor (Host)

Evolución del hardware para servidores



Un equipo
Data Center local (on-premise)
Data Center externo
Cloud Data Center

Donde colocar los servidores ?

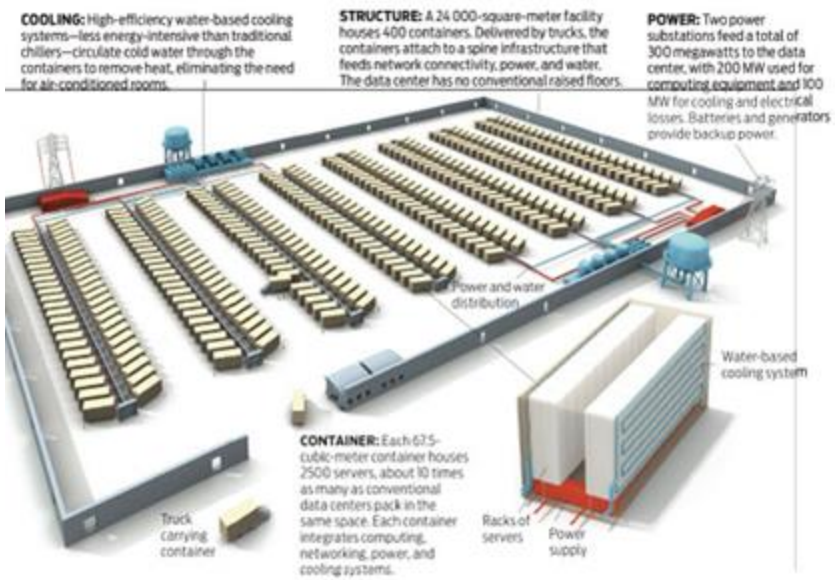
Modelo de despliegue

Data Center

Centro de control de Red (NOC)

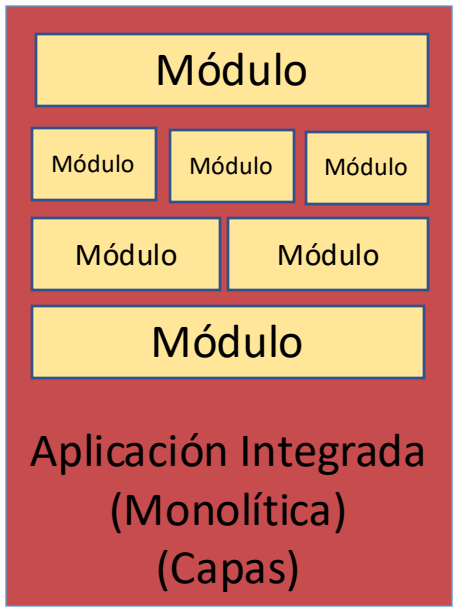
Modelo de despliegue

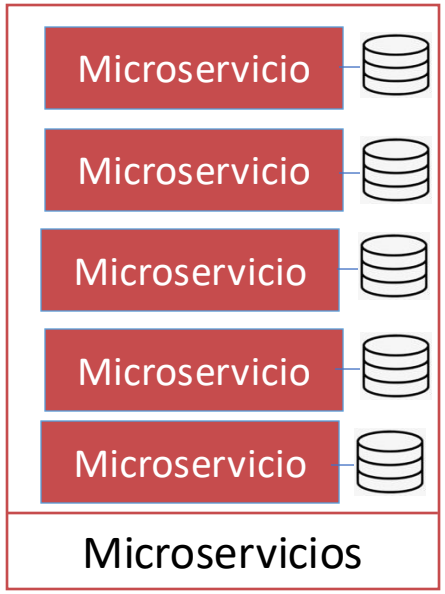
Centro de computación en Nube

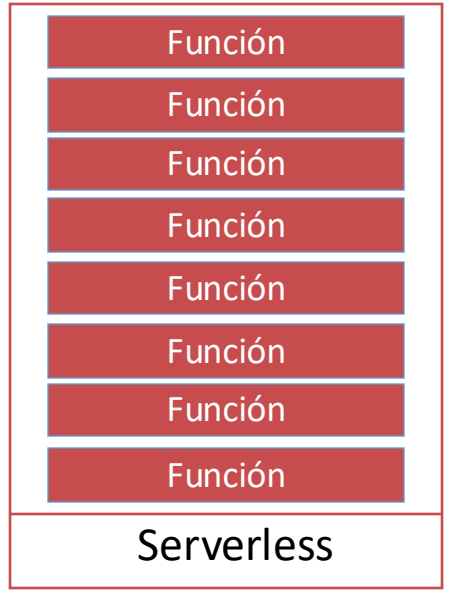


Modelo de despliegue

¿Como construir aplicaciones para para ser desplegadas: Estilo arquitectural







Servidores físicos o virtuales (locales o en la nube)

Plataformas en la nube

Modelo de despliegue

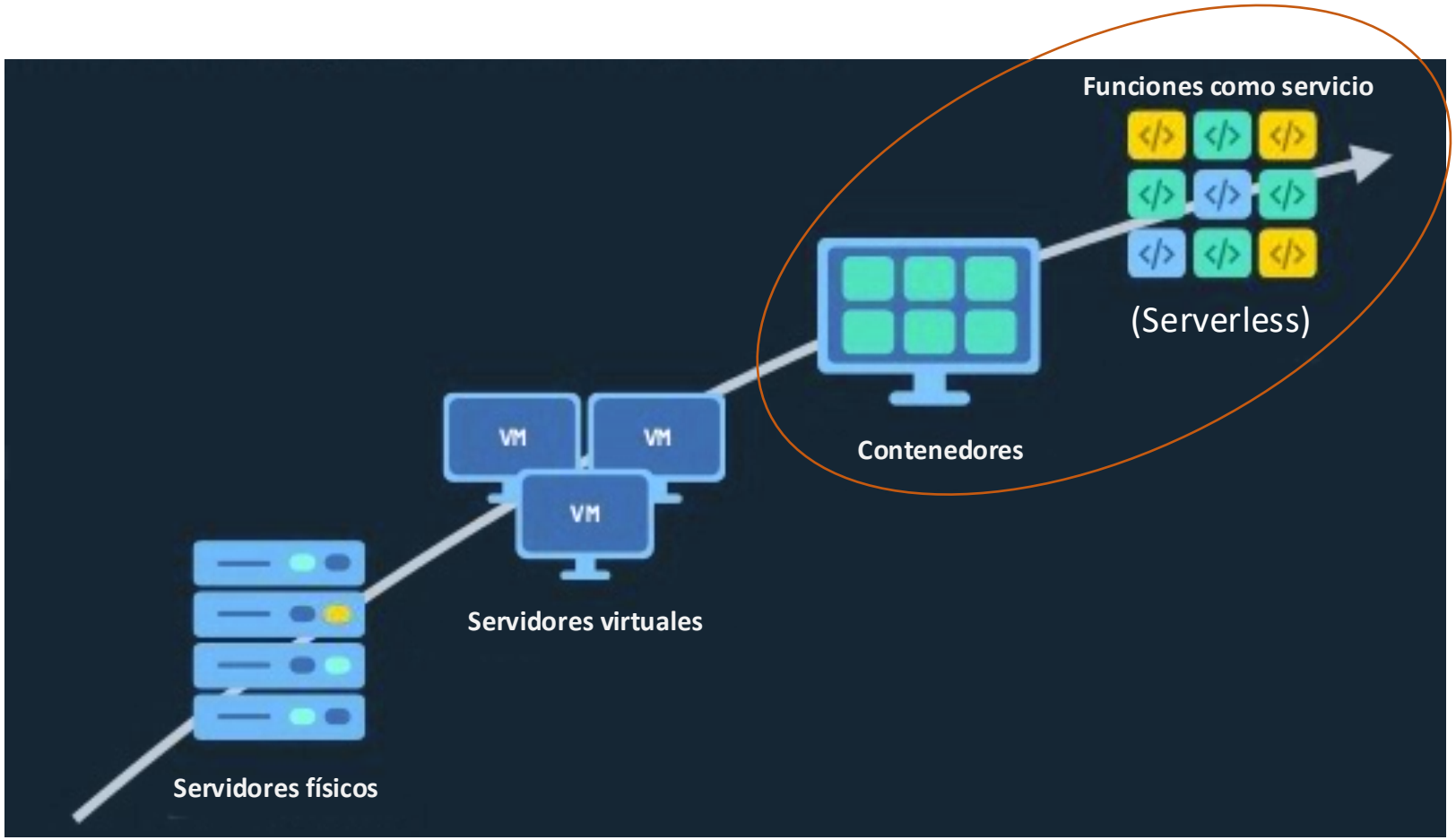
¿Como construir aplicaciones para para ser desplegadas: Estilo arquitectural

Serverless
Es una tecnología en la no se usan servidores (físico o virtual) si no que usa una plataforma en la nube la cual utiliza contenedores temporales y sin estado donde se ejecuta el código de las funciones. Estos contenedores se crean en el momento que se ejecuta la aplicación y luego desaparecen. Además, todas las funciones pueden ejecutarse independientemente por una aplicación.

Esta tecnología se asocia con FaaS que significa Function as a Service, que fue creada en 2014 por hook.io y que luego se ha ido desarrollando mediante proyectos tan importantes como Microsoft Azure Functions, IBM Cloud Functions, Google Cloud Functions y AWS Lambda, Apache OpenWhisk y OpenFaaS (Open Source)

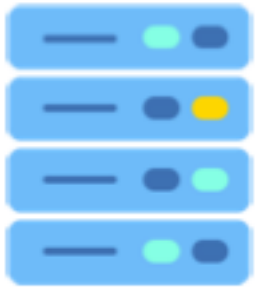
Modelo de despliegue

Evolución de los servidores para despliegue de aplicaciones



Modelo de despliegue

Evolución de los servidores para despliegue de aplicaciones



Servidor físico

- Code
- App Container
- Language Runtime
- Operating System
- Hardware

Máquina virtual

- Code
- App Container
- Language Runtime
- Operating System
- Hardware

Contenedores

- Code
- App Container
- Language Runtime
- Operating System
- Hardware

Serverless

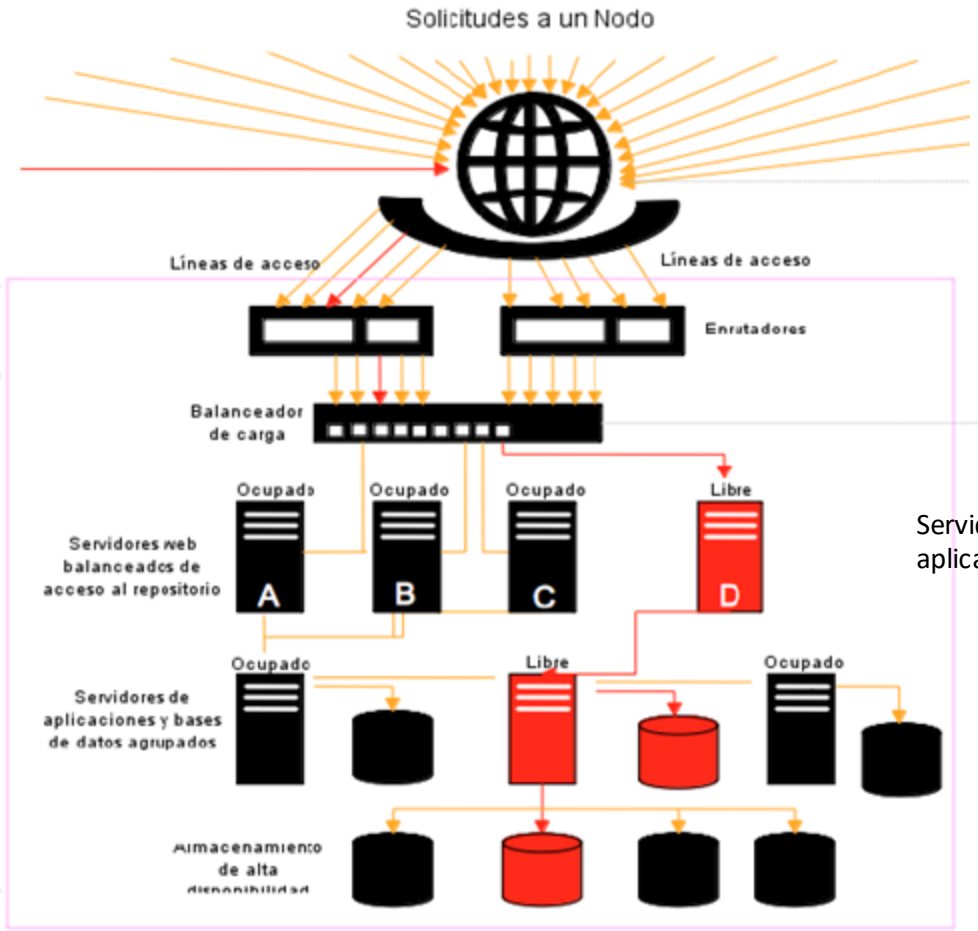
- Code
- App Container
- Language Runtime
- Operating System
- Hardware

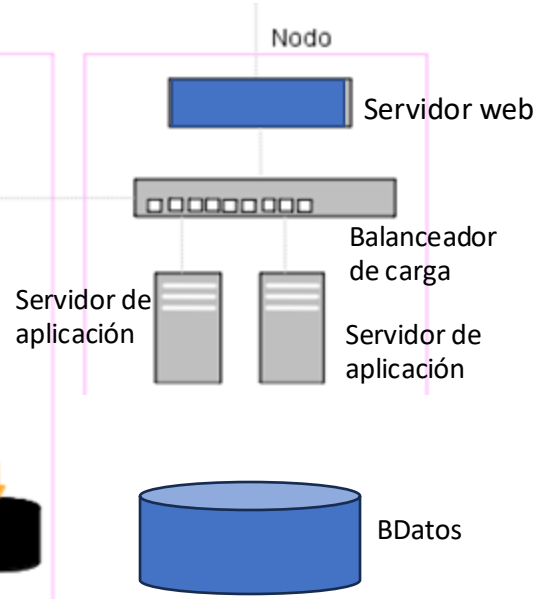
Modelo de despliegue

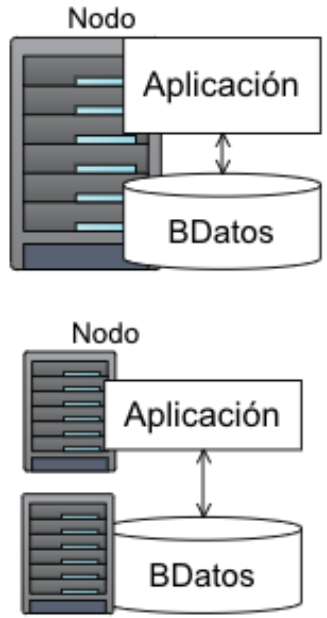
Ambientes para desarrollo, pruebas y despliegue de aplicaciones

Modelo de despliegue

Nodos para despliegue de aplicaciones (Diagrama de despliegue)







Modelo de despliegue

Nodos para despliegue de aplicaciones (Diagrama de despliegue)

Redundancia: Múltiples discos, servidores o incluso centros de datos se utilizan para almacenar copias de los datos.
Replicación: Los datos se copian y sincronizan en tiempo real entre los diferentes dispositivos de almacenamiento.
Failover automático: Si un dispositivo falla, el sistema cambia automáticamente a una réplica funcional sin intervención manual.
Monitoreo: El sistema monitorea constantemente el estado de los dispositivos de almacenamiento para detectar y responder a cualquier fallo.

Modelo de despliegue

Como calcular el tamaño de los servidores de los nodos para despliegue de aplicaciones

(RAM, Procesadores, Espacio de almacenamiento, Ancho de banda)

1. Según los requisitos de la aplicación

• **Cantidad de usuarios**: Estimar cuántos usuarios concurrentes tendrá la aplicación. Esto te permitirá entender la demanda que se generará sobre el servidor.
• **Requerimientos de procesamiento**: Algunas aplicaciones requieren mayor capacidad de cómputo debido a cálculos complejos, como las que implementan inteligencia artificial, procesamiento de imágenes o gestión de grandes volúmenes de datos, uso de videos, etc.
• **Memoria requerida**: Evaluar cuánta memoria RAM necesita la aplicación para funcionar de manera eficiente, considerando los procesos en segundo plano y la multitarea.
• **Almacenamiento**: Determinar cuánto espacio en disco se necesita, no solo para la instalación de la aplicación, sino también para el almacenamiento de datos (bases de datos, logs, archivos generados por usuarios, etc.).

Modelo de despliegue

Como calcular el tamaño de los servidores de los nodos para despliegue de aplicaciones (RAM, Procesadores, Espacio de almacenamiento, Ancho de banda)

2. Dimensionamiento basado en la carga esperada:

• **CPU**: Identificar el número de núcleos o procesadores que el servidor debe tener para soportar las tareas concurrentes y el procesamiento de solicitudes de usuarios.

• **Memoria RAM**: Estimar la cantidad de memoria que se necesitará según el tipo de aplicación (por ejemplo, aplicaciones web pesadas o servicios que hacen uso intensivo de memoria como bases de datos en memoria).

• **Almacenamiento y tipo de disco**: Considerar si necesitas discos HDD o SSD, dependiendo del rendimiento necesario para la escritura y lectura de datos.

Como calcular el tamaño de los servidores de los nodos para despliegue de aplicaciones

(RAM, Procesadores, Espacio de almacenamiento, Ancho de banda)

3. Escalabilidad y redundancia

• **Escalado horizontal o vertical**: Decidir si es más eficiente escalar los servidores aumentando sus recursos (escalado vertical) o añadiendo más servidores (escalado horizontal). En entornos cloud, el escalado horizontal es muy popular ya que permite añadir o quitar servidores según la demanda.

• **Redundancia y balanceo de carga**: Asegurar que el sistema es tolerante a fallos, incorporando servidores de respaldo o replicación, y distribuyendo la carga entre varios servidores mediante un balanceador de carga.

Como calcular el tamaño de los servidores de los nodos para despliegue de aplicaciones

(RAM, Procesadores, Espacio de almacenamiento, Ancho de banda)

Herramientas y monitoreo

• **Monitoreo del uso**: Utilizar herramientas para medir el uso de recursos del servidor una vez que la aplicación esté en producción, y ajustar el tamaño de los servidores según la demanda real.

• **Pruebas de rendimiento**: Realizar pruebas de estrés y carga para determinar si los recursos asignados son suficientes o si es necesario ajustar la configuración.

Modelo de despliegue

Como calcular el tamaño de los servidores de los nodos para despliegue de aplicaciones
(RAM, Procesadores, Espacio de almacenamiento, Ancho de banda)

Ejemplo: Caraterísticas típicas en VM

1 Gigas RAM 1 o 2 Procesadores	2 Gigas RAM 2 Procesadores	4 Gigas RAM 2 Procesadores	8 Gigas RAM 2 Procesadores	16 Gigas RAM 2 Procesadores
4 Gigas RAM 4 Procesadores	8 Gigas RAM 4 Procesadores	16 Gigas RAM 4 Procesadores	32 Gigas RAM 4 Procesadores	64 Gigas RAM 4 Procesadores
8 Gigas RAM 8 Procesadores	16 Gigas RAM 8 Procesadores	32 Gigas RAM 8 Procesadores	64 Gigas RAM 16 Procesadores	128 Gigas RAM 16 Procesadores

Otros aspectos que pueden impactan :
- [Número de aplicaciones en el mismo servidor]
- Velocidad del procesador
- Tipos de discos (SSD)
- Núcleos de GPU

Las pruebas carga de las aplicaciones ayudan a determinan las características de los servidores requeridos

¿Cómo calcular las características de un servidor(es) para una Aplicación?

1. Por tipo de aplicación (Ejemplos típicos)

Tipo de app	RAM	CPU	GPU	Disco
Web estática / API liviana	1–2 GB	1–2 vCPU	No	20–50 GB SSD
Aplicación web con base de datos	4–8 GB	2–4 vCPU	Opcional	50–100 GB SSD
Procesamiento de datos / backend	8–16 GB	4–8 vCPU	Opcional	100+ GB SSD
IA / Machine Learning	16–64 GB	8+ vCPU	Sí (T4, A100)	200+ GB SSD

¿Cómo calcular las características de un servidor(es) para una Aplicación?

2. Número de usuarios simultáneos

Estima cuántos usuarios usarán la aplicación al mismo tiempo.
Más usuarios = más RAM y CPU necesarias. (o más nodos)

Regla general:
• 1 vCPU y 512 MB – 1 GB de RAM por cada 100 usuarios concurrentes (API estándar).
• Si hay procesamiento en tiempo real, aumenta esa cifra.

3. Carga de trabajo (workload)

Pregunta: ¿La app hace cálculos intensivos, consultas a bases de datos, renderiza gráficos?

• **Cálculos intensivos** → necesitas CPU o GPU.
• **Renderizado o inferencia de modelos de IA** → necesitas GPU.
• **Almacenamiento de archivos o logs** → necesitas más disco.

Modelo de despliegue

Despliegue de aplicaciones (**Deploy**)

El despliegue de aplicaciones consiste en copiar los archivos de una aplicación en un servidor o nodo donde funcionará la aplicación. (configurado)

Que archivos copiar:

- Programas ejecutables o scripts
- Librerías requeridas con sus versiones específicas
- Otras librerías o paquetes (Ej: librerías gráficas, etc)
- Archivos de datos requeridos
- Bases de datos y drivers de conexión

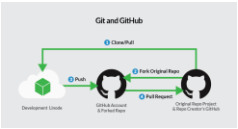
Modelo de despliegue

Despliegue continuo: Despliegue por Release, por Sprint o Hot fix

Métodos para hacer Deploy:

Manual

Instalar desde un repositorio (EJ. Git pull)
git clone https://github.com/github/miapp.git



Automático: **Iac** (Infraestrucrure as a Code)

Infraestructure as a Code (IaaC)
Infraestructura como código (IaC) significa administrar su infraestructura de TI mediante archivos de configuración.

Los procesos de desarrollo ágil intentan que el “Deploy” cada vez sea más automático

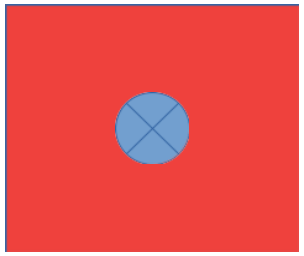
Despliegue de aplicaciones Web: Estrategias para actualizar el servidor

Tipo Stand Alone (Stop-Start)

- Se basa en tener un servidor (de producción) sobre el cual se realizan los cambios, **parar el servidor**, **realizar los cambios** y luego **reiniciar el servidor**.
- Cundo es por primera vez, puede ser un proceso adecuado.
- No se recomienda esta estrategia cuando el servidor esta en producción.
- Al parar el servidor todos los usuarios no tendrán acceso al servicio.

En funcionamiento

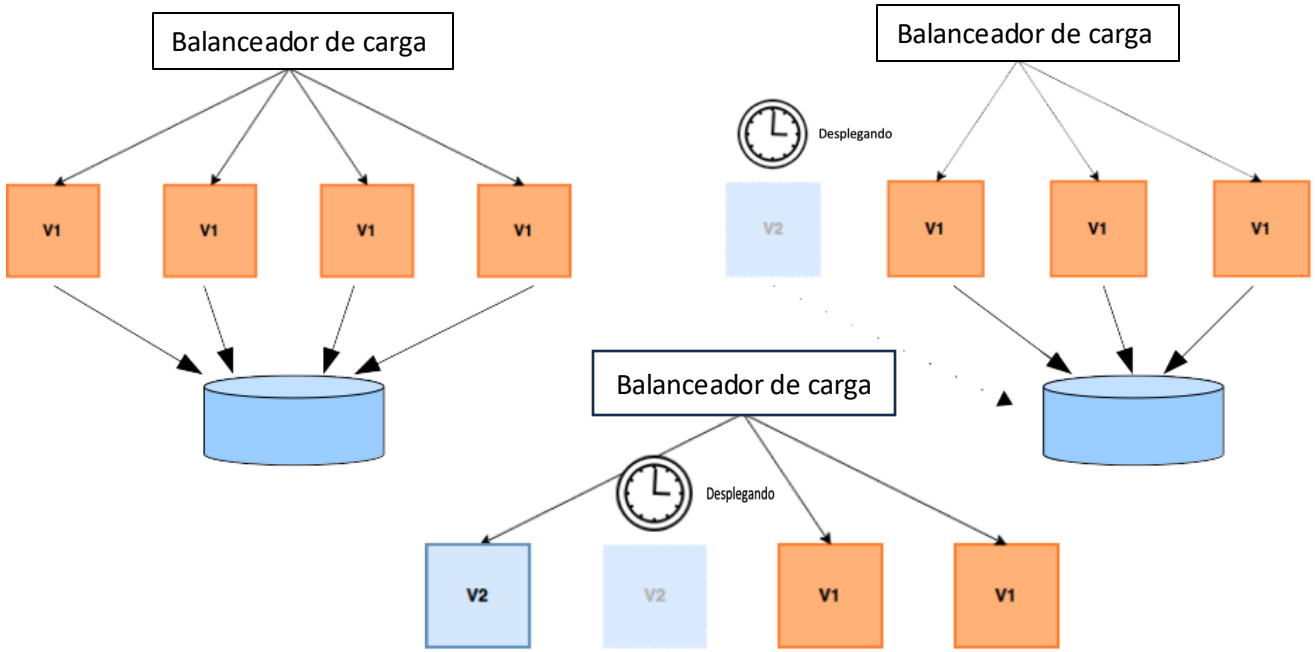
No disponible



En funcionamiento

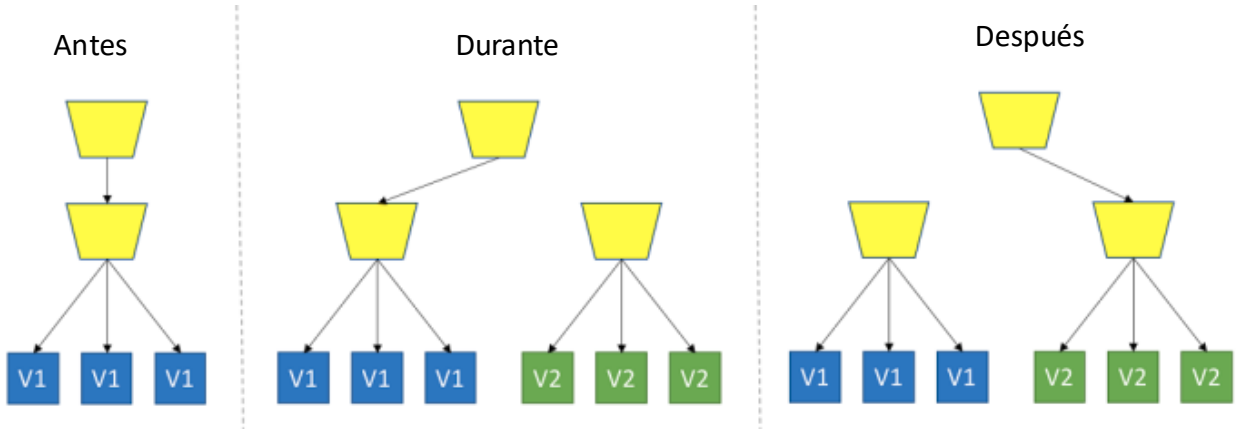
Despliegue de aplicaciones Web: Estrategias para actualizar el servidor

Rolling upgrade: Se basa en tener una capa de balanceo de la aplicación.
Se desconecta un servidor del balanceador, se actualizase y se prueba,
si funciona bien, se conecta al balanceador de nuevo y se repite el proceso hasta actualizar todos los servidores.



Despliegue de aplicaciones Web: Estrategias para actualizar el servidor

Despliegue Blue/Green: Duplicar temporalmente la infraestructura y desplegar todo a la vez

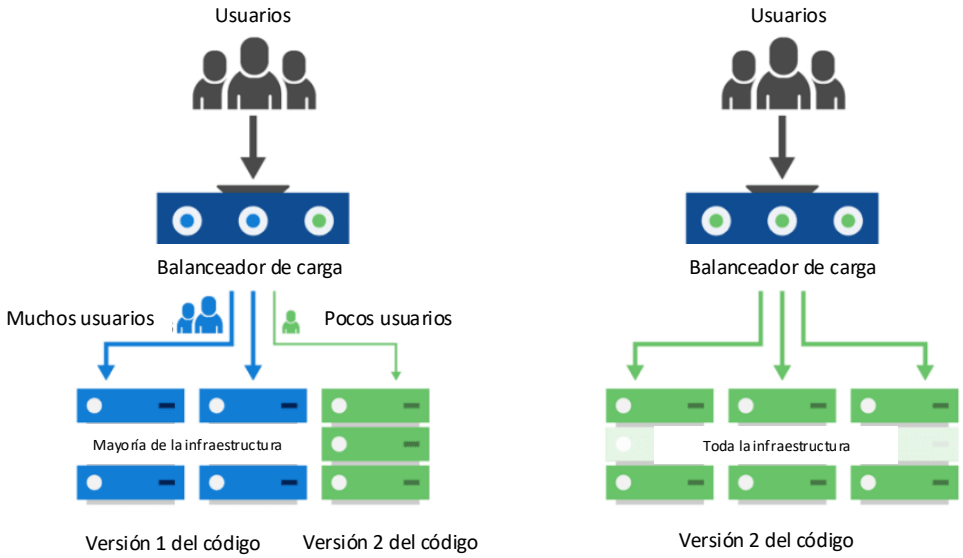


Esta estrategia también se puede aplicar cuando es un solo servidor.
Usar contenedores es muy práctico para este tipo de despliegue.
También le dan nombres como Despliegue A/B o Red-Black.

Despliegue de aplicaciones Web: Estrategias para actualizar el servidor

Canary Deployment: "rolling out releases" Despliegue del canario es un patrón de despliegue que se aplica a un subconjunto de usuarios o servidores.

- La idea principal es primero desplegar los cambios en un conjunto pequeño de servidores, probar con algunos usuarios y luego continuar el despliegue hacia el resto de usuarios y servidores.
- El despliegue "canary" sirve como sistema indicador de alerta temprana, si el despliegue del "canario" falla, el resto de servidores no se ven impactados.



Preguntas ?

Fecha de Exposiciones, Taller y Proyecto

Grupo	Tema	Fecha exposición
	Pruebas unitarias en Python y Django	Abril 4 de 2025
	Estándares de codificación en Python y Django	Abril 4 de 2025
	Tablero Kanban	Abril 4 de 2025
	Taller 2 Django (arquitectura desacoplada)	Abril 11 de 2025
	Chrome DevTools	Abril 25 de 2025
	Git y git flows (incluir git tags)	Abril 25 de 2025
	Web services con Django	Abril 25 de 2025
	Primera entrega del proyecto	**Mayo 2 de 2025**
	WAF (Web Applications Firewall)	Mayo 9 de 2025
	Desarrollo de aplicaciones web usando LLMs (ChatGPT)	Mayo 9 de 2025
	Low code y nocode	Mayo 9 de 2025
	Taller 3 (Despliegue)	Mayo 16 de 2025
	Examen Final	**Mayo 30 de 2025**
	Entrega final del proyecto	**Junio 6 de 2025**

Trabajo en equipos del proyecto del curso

Gracias

Universidad del Valle