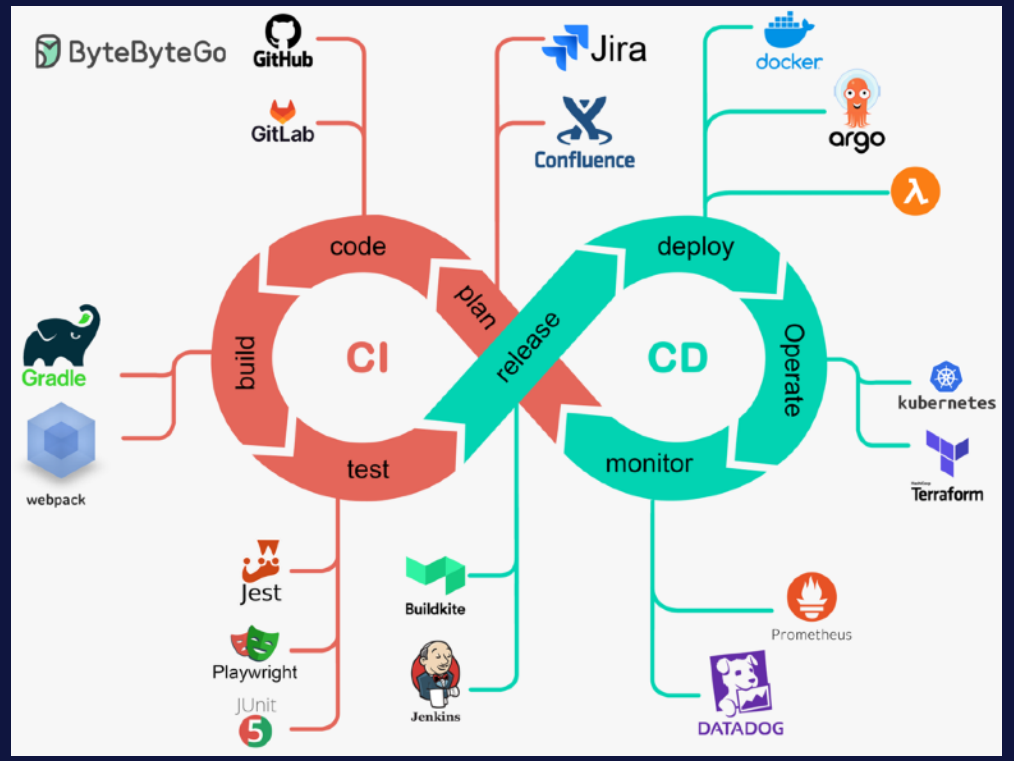
códigofacilito

CI/CD: Conceptos y Ventajas

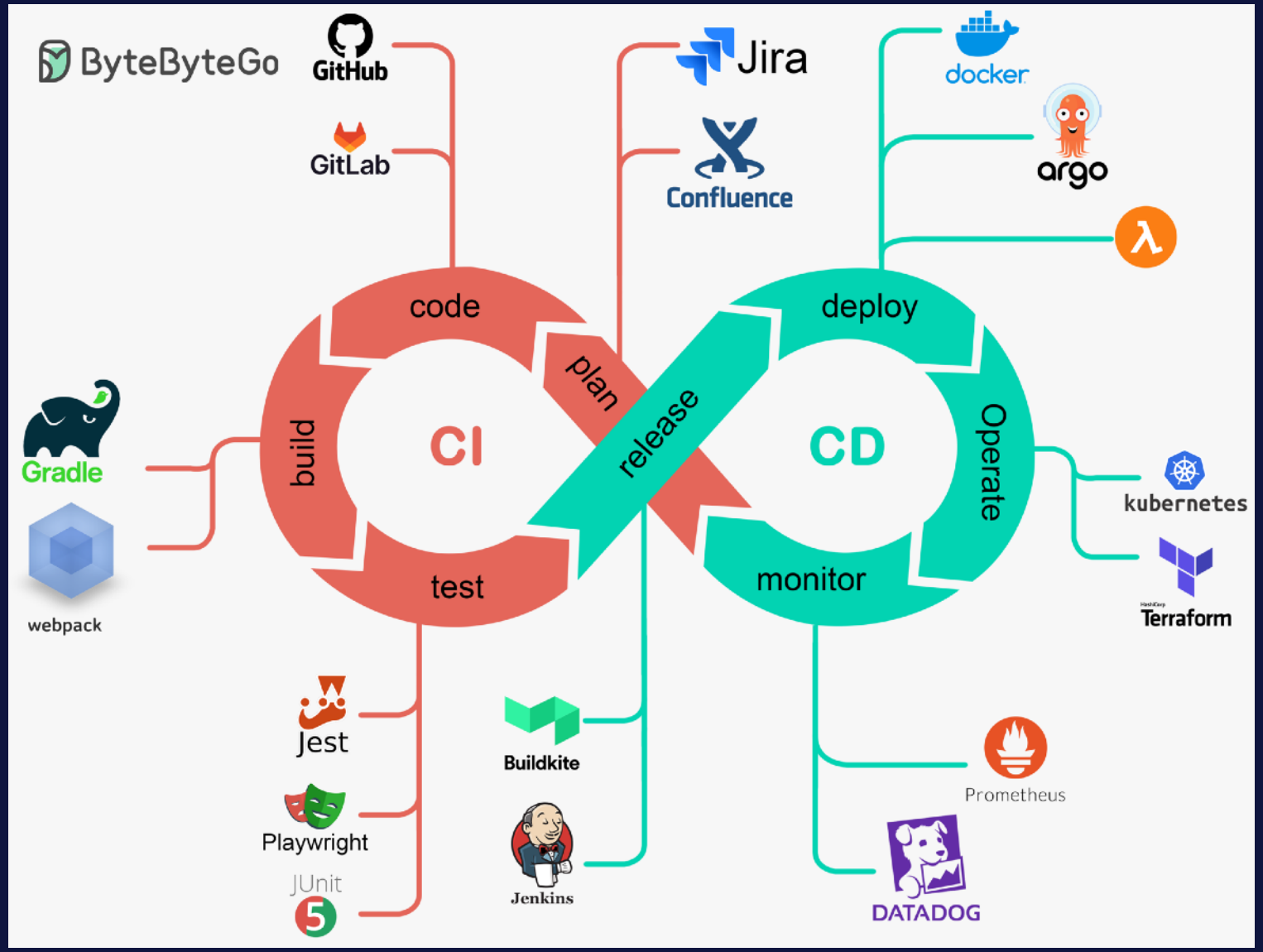
Bootcamp de DevOps

🚀 ¿Qué es CI/CD?

- Automatización del proceso de desarrollo de software
- CI = Integración Continua
- CD = Despliegue Continuo



🚀 ¿Qué es CI/CD?


ByteByteGo
GitHub
GitLab
Jira
Confluence
docker
argo
code
plan
build
CI
release
CD
deploy
Operate
monitor
test
Gradle
webpack
kubernetes
Terraform
Jest
Playwright
JUnit
Buildkite
Jenkins
Prometheus
DATADOG

🚀 ¿Qué es Integración Continua (CI)?

- El sistema automáticamente compila el código y ejecuta todas las pruebas cada vez que un desarrollador realiza cambios
- Los desarrolladores integran su código varias veces al día en un repositorio central, manteniendo el código actualizado
- Se ejecutan pruebas automatizadas que verifican que el código funcione correctamente y cumpla con los requisitos del negocio
- Los problemas se detectan y corrigen inmediatamente, evitando que los errores se acumulen o lleguen a producción

🚀 Beneficios de la Integración Continua (CI)

Evita merge conflicts	Asegura sincronización frecuente del código entre desarrolladores y el repositorio principal.
Reduce el esfuerzo de revisión de código	Minimiza revisiones manuales gracias a la colaboración continua y automatizada.
Acelera el proceso de desarrollo	Mejora la velocidad de desarrollo mediante colaboración eficiente e integración fluida.
Build listo para pruebas	Garantiza una versión compilada y actualizada del software siempre disponible para testing.
Reduce el backlog del proyecto	Disminuye tareas pendientes mediante implementación continua de cambios en el repositorio.

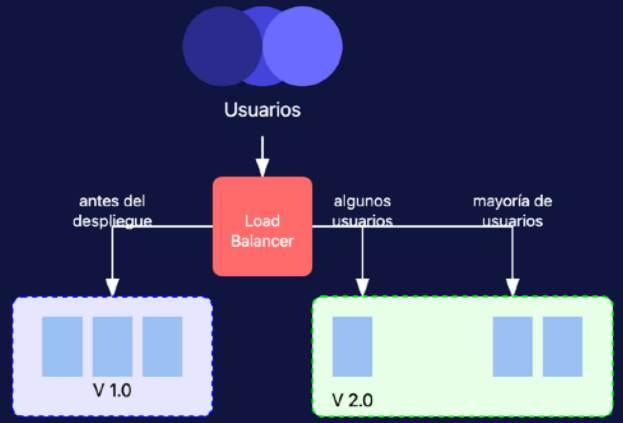
🚀 ¿Qué es Despliegue Continuo (CD)?

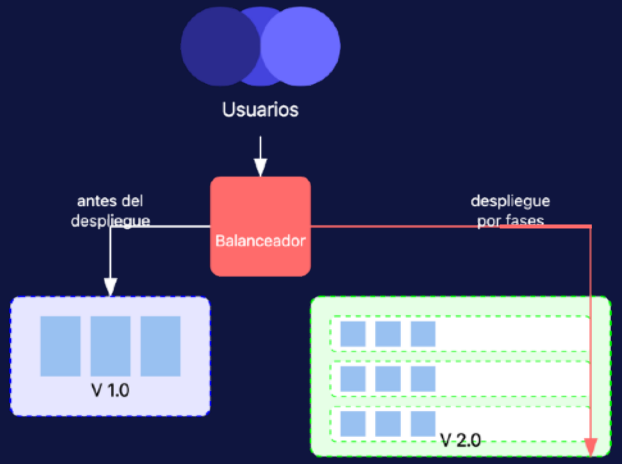
- El código que pasa todas las pruebas automatizadas se despliega automáticamente en producción, eliminando intervenciones manuales
- La entrega continua asegura que el código esté siempre listo para desplegarse, pasando por múltiples ambientes de prueba (QA, Staging, Performance)
- El proceso completo incluye compilación, pruebas, generación de artefactos y entrega a ambientes específicos
- El código se mantiene siempre desplegable, permitiendo entregas rápidas y confiables al ambiente de producción

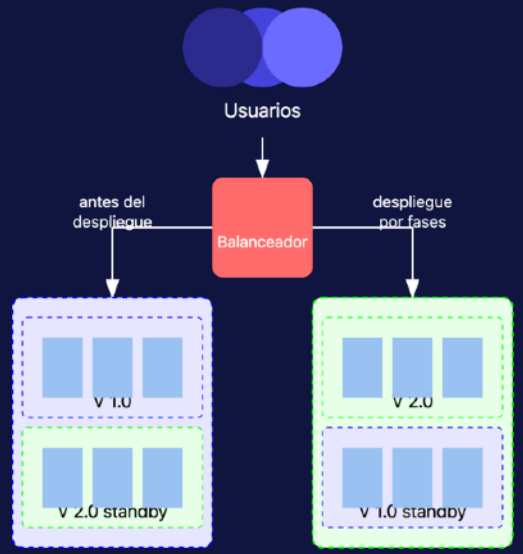
🚀 Beneficios del Despliegue Continuo (CD)

Desarrollo más rápido	Facilita un desarrollo y entrega más rápida de productos mediante la automatización del proceso. Resulta en ciclos de desarrollo más cortos.
Despliegues menos riesgosos	La automatización reduce errores manuales durante el despliegue permitiendo un proceso más suave y una identificación y solución más rápida de problemas.
Mejora continua de calidad	Permite mejorar constantemente la calidad del producto mediante automatización y mejoras incrementales frecuentes.

Estrategias de Despliegue

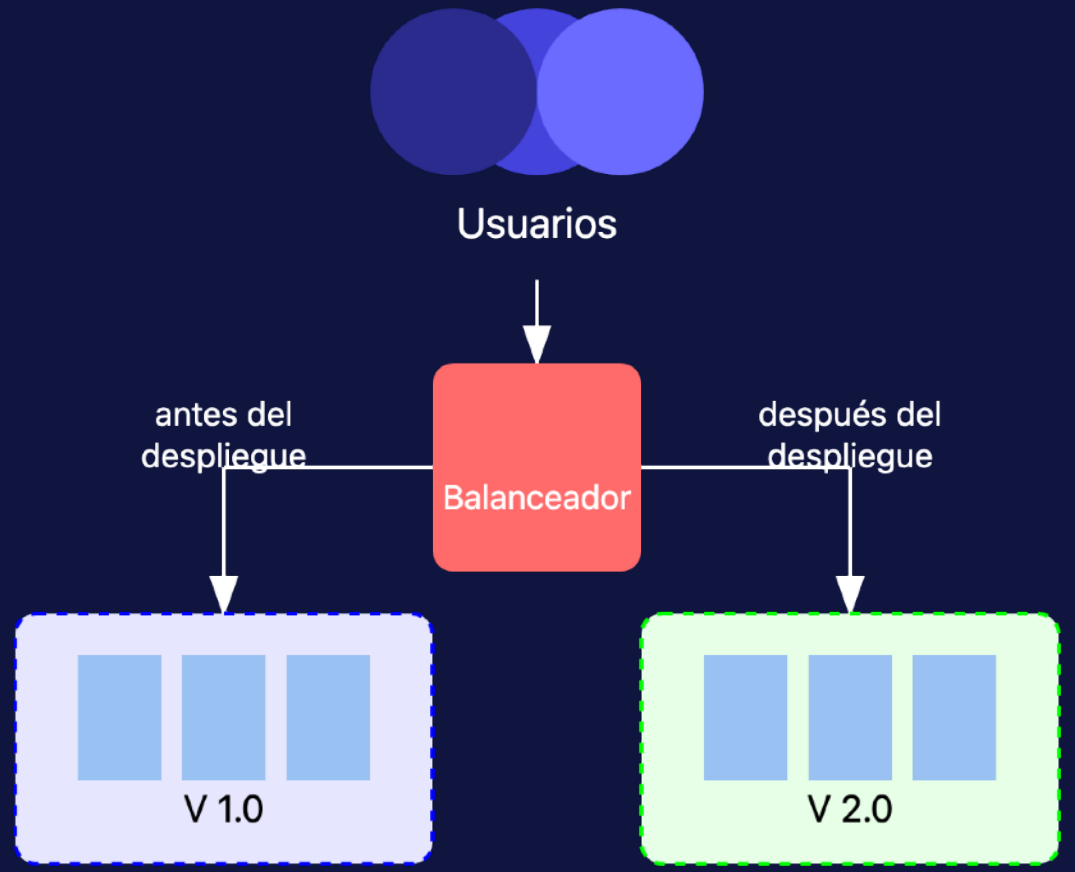




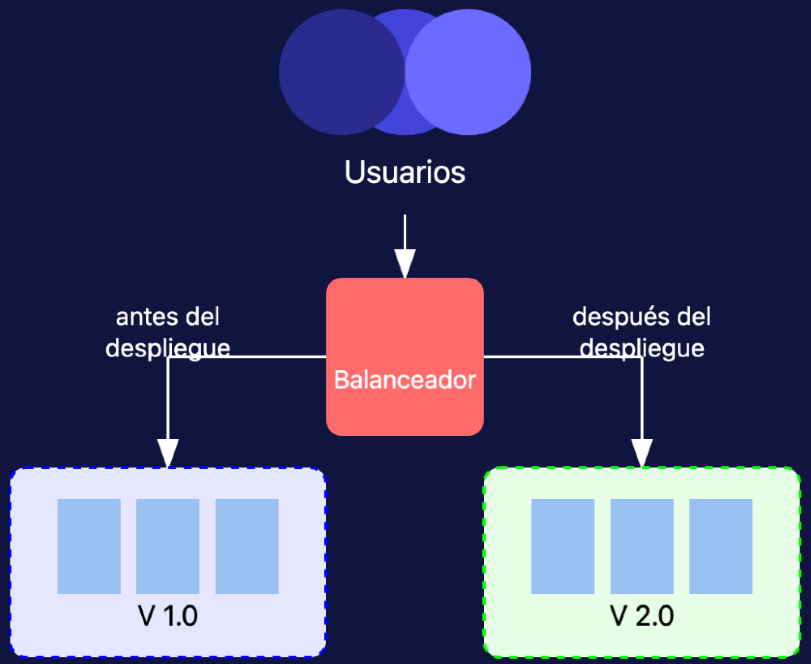


Despliegue Big Bang

- Actualiza todo el sistema de una sola vez, como quitar una bandita rápidamente
- Requiere que el sistema se apague temporalmente durante el cambio
- Si algo sale mal, se puede volver atrás, pero esto puede causar problemas
- A veces es la única opción, especialmente con actualizaciones complejas de bases de datos



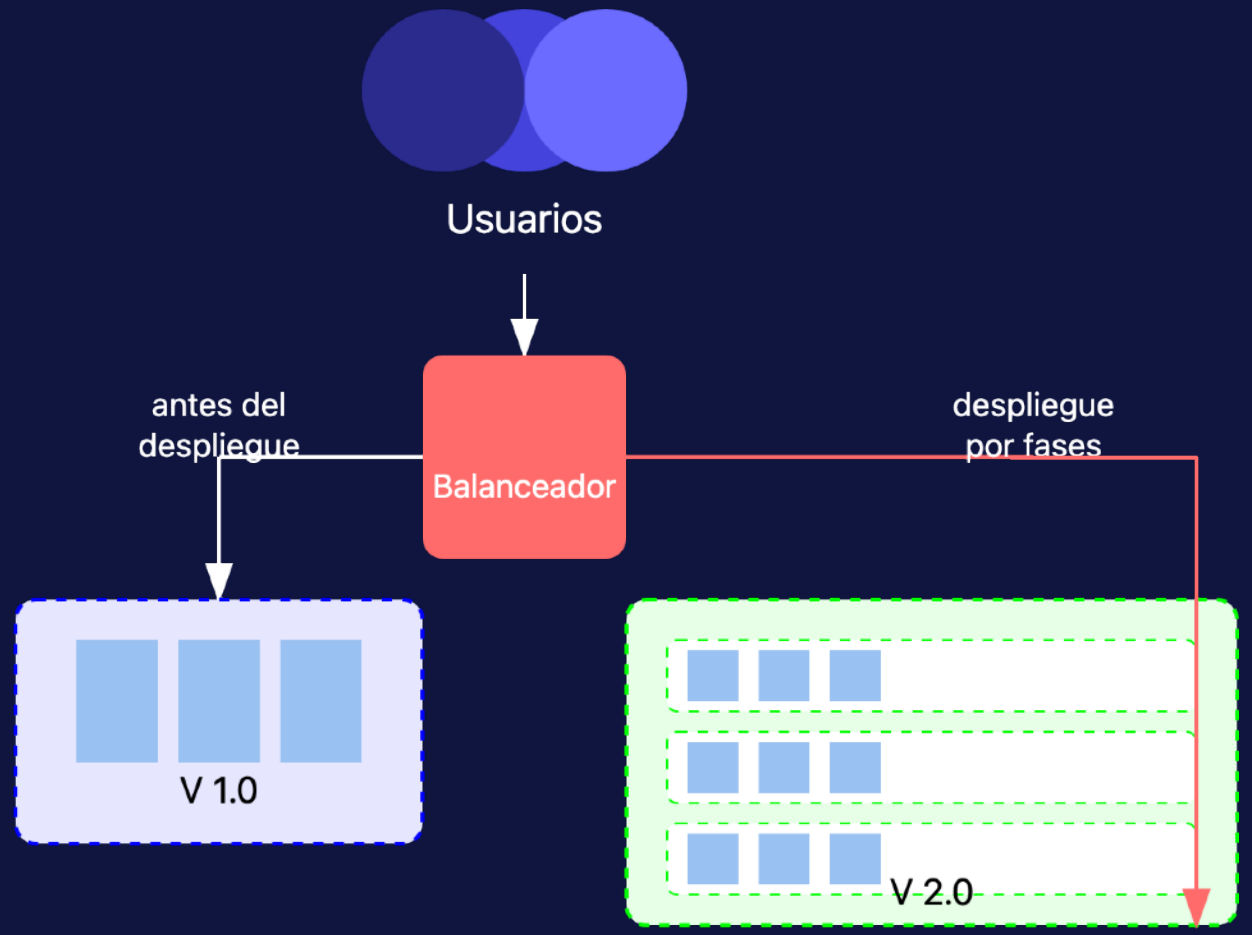
Ejemplo de despliegue Big Bang



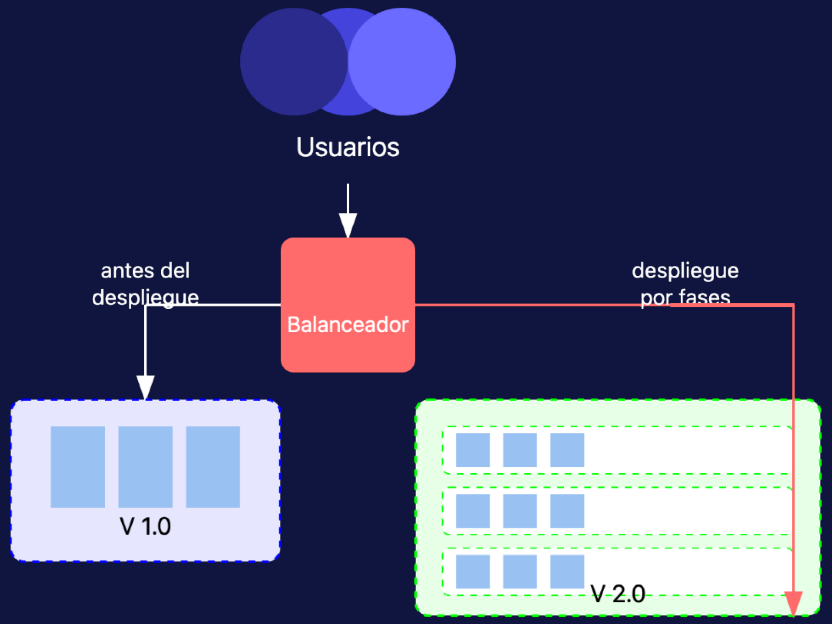

```yaml
jobs:
  deploy-all:
    name: Deploy Everything
    runs-on: ubuntu-latest

    steps:
      # Detener todo
      - name: Stop All Services
        run: |
          echo "Stopping all services..."
          docker-compose down

      # Desplegar todo a la vez
      - name: Deploy All Services
        run: |
          echo "Deploying all services
simultaneously..."
          docker-compose up -d \
            web-app \
            api \
            database \
            cache

      # Comprobación rápida de salud
      - name: Health Check
        run: |
          echo "Checking if everything is up..."
          sleep 30
          docker-compose ps
```

Despliegue Rolling

- Actualiza el sistema por partes: se actualiza un servidor a la vez mientras los demás siguen funcionando
- La principal ventaja es que el servicio nunca se detiene completamente, los usuarios siguen usando el sistema durante la actualización
- Si hay problemas, se detectan temprano y solo afectan a una pequeña parte del sistema
- Es un proceso más lento pero más seguro, aunque no permite elegir qué usuarios reciben la actualización primero



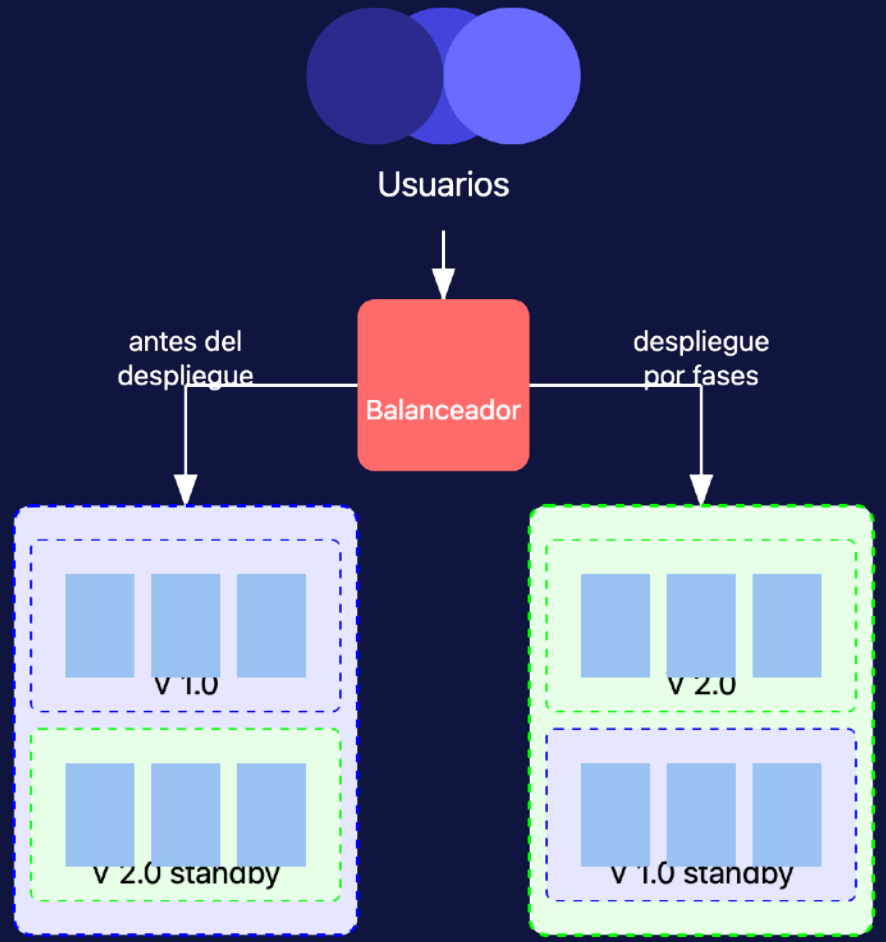
Ejemplo de despliegue Rolling



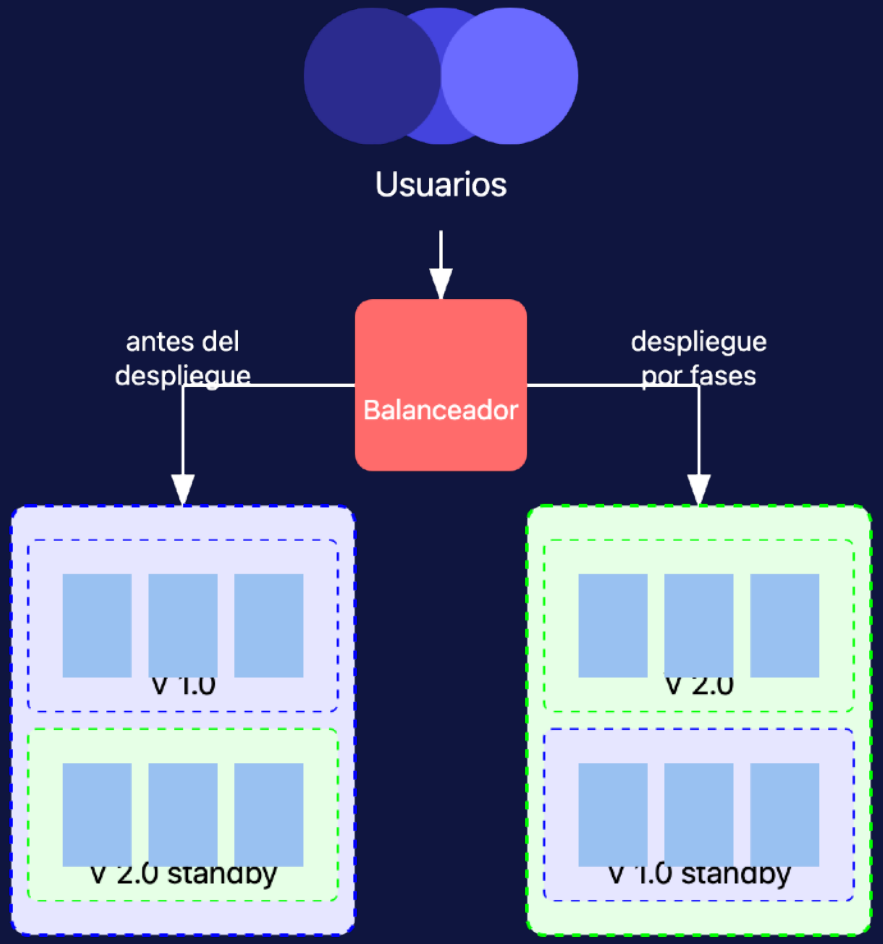

```
    steps:
      # Actualizar la imagen del despliegue
      # Este paso usa `kubectl set image` para actualizar
la imagen del contenedor en el despliegue
      - name: Actualizar la imagen del despliegue
        run: |
          kubectl set image deployment/mi-despliegue \
          mi-contenedor=mi-registro/mi-imagen:$
{{ github.sha }}

      # Verificar el estado del despliegue
      # Asegurarse de que los pods estén en buen estado
después del despliegue
      - name: Verificar el estado del despliegue
        run: |
          kubectl rollout status deployment/mi-despliegue
```

Despliegue Blue Green

- Usa dos ambientes idénticos: uno activo (Blue) que atiende a los usuarios y otro inactivo (Green) donde se prueba la nueva versión
- Cuando la nueva versión está lista y probada, simplemente se redirige el tráfico del ambiente Blue al Green, haciendo el cambio sin interrupciones
- Si hay problemas, es fácil volver atrás cambiando de nuevo al ambiente anterior
- La desventaja principal es que requiere el doble de recursos al mantener dos ambientes idénticos y no permite dirigir la actualización a usuarios específicos



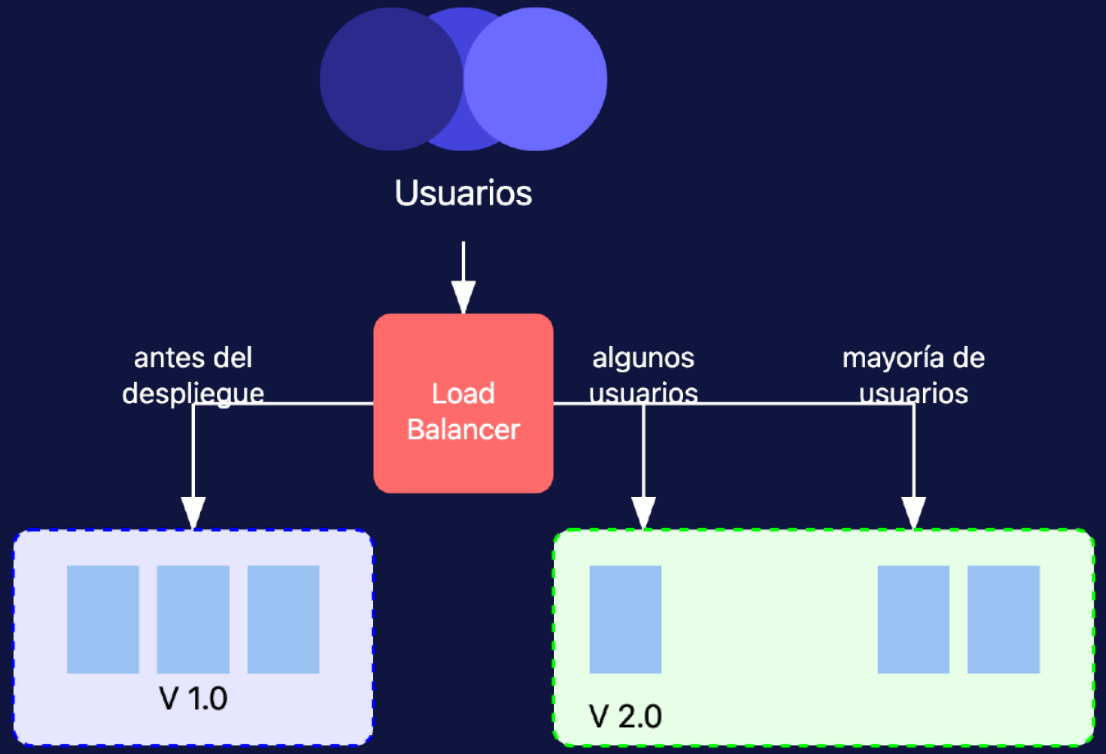
Ejemplo de despliegue Blue Green



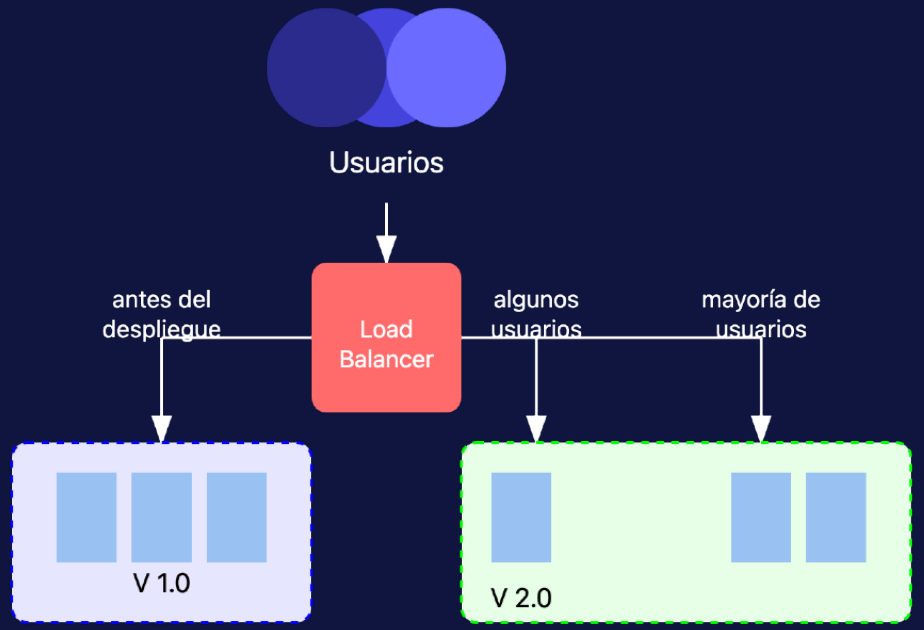

```
      # 1. Desplegar a un entorno Green
      # Copiar los artefactos al entorno Green (servidores o
contenedores)
      - name: Desplegar artefactos al entorno Green
        run: |
          ssh usuario@green-servidor 'deploy-script.sh --
imagen mi-imagen:${{ github.sha }}'

      # 2. Validar el entorno Green
      # Ejecutar pruebas para asegurar que el entorno Green
funciona correctamente
      - name: Validar el entorno Green
        run: |
          curl -f http://green-servidor/health || exit 1

      # 3. Cambiar tráfico al entorno Green
      # Usar un balanceador de carga para redirigir el
tráfico
      - name: Cambiar tráfico al entorno Green
        run: |
          ssh usuario@load-balancer 'update-config.sh --
target green-servidor'

      # 4. (Opcional) Desactivar entorno Blue
      # Una vez validado el entorno Green, desactiva el
entorno Blue
      - name: Desactivar entorno Blue
        run: |
          ssh usuario@blue-servidor 'stop-application.sh'
```

Despliegue Canario

- Prueba la nueva versión con un pequeño grupo de usuarios (los "canarios") antes de lanzarla a todos
- Si todo funciona bien con los canarios, se despliega gradualmente al resto; si hay problemas, solo se afecta a ese pequeño grupo
- Permite seleccionar específicamente quién recibe la actualización (por ejemplo, por ubicación o tipo de dispositivo)
- Es más complejo de implementar y requiere un monitoreo cuidadoso, pero es muy seguro al combinar pruebas reales con riesgo limitado



Ejemplo de despliegue Canario

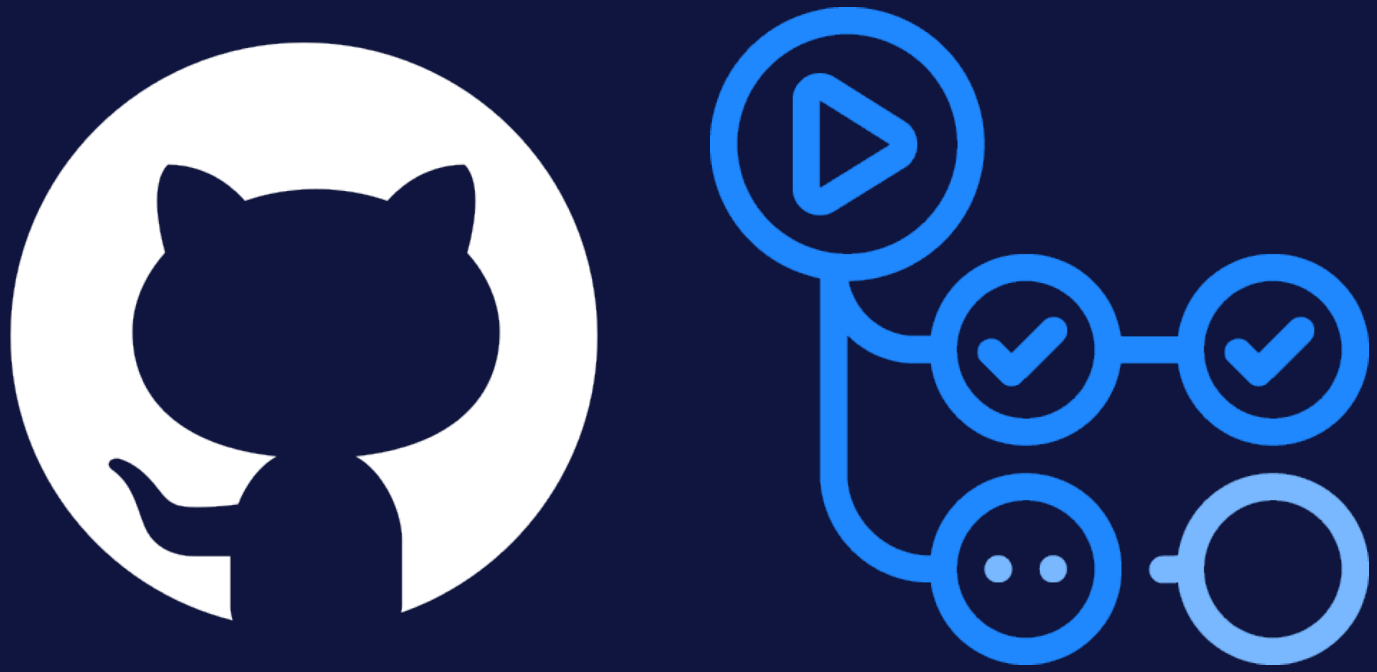



```
    # 1. Desplegar a Canary
     - name: Desplegar a Canary
       run: ssh usuario@canary-servidor "deploy-script.sh --
imagen mi-imagen:${{ github.sha }}"

     # 2. Validar Canary
     - name: Validar Canary
       run: curl -f http://canary-servidor/health

     # 3. Dirigir tráfico parcial al Canary
     - name: Actualizar balanceador de carga
       run: ssh usuario@load-balancer "update-config.sh --
add-canary canary-servidor --weight 10"

     # 4. Aprobar despliegue completo manualmente
     - name: Aprobar despliegue completo
       uses: hmarr/auto-approve-action@v2
       with:
         prompt: "¿Aprobar despliegue completo a
producción?"
         timeout: 60

     # 5. Desplegar a producción
     - name: Desplegar a producción
       if: success()
       run: |
         ssh usuario@prod-servidor "deploy-script.sh --
imagen mi-imagen:${{ github.sha }}"
         ssh usuario@load-balancer "update-config.sh --
remove-canary --target prod-servidor"
```

👋 ArgoCD

https://argo-cd.readthedocs.io/en/stable/

👋 Github Actions



https://github.com/features/actions