Bibliografía:

* Arquitectura:
  + <https://martinfowler.com/bliki/BlueGreenDeployment.html>
  + <https://developer.ibm.com/urbancode/videos/bluegreen-deployments-ibm-urbancode/>
* Feature Toggle vs Branch
  + <https://en.wikipedia.org/wiki/Feature_toggle>
  + <https://www.youtube.com/watch?v=skLJuksCRTw>
  + <https://github.com/launchdarkly/featureflags/wiki/Blue-Green-Deployments>
* Aplicación Práctica
  + <https://github.com/awslabs/ecs-canary-blue-green-deployment>
  + <https://docs.cloudfoundry.org/devguide/deploy-apps/blue-green.html#implementations>
  + <https://www.digitalocean.com/community/tutorials/how-to-use-blue-green-deployments-to-release-software-safely>
  + <https://technologyconversations.com/2016/02/08/blue-green-deployment/>
  + <https://www.youtube.com/watch?v=vIviK2RjL0I>
* DOs & DON’Ts
  + <https://minops.com/blog/2015/02/the-dos-and-donts-of-bluegreen-deployment/>
* Vs. Canary Deployment
  + <https://martinfowler.com/bliki/CanaryRelease.html>
  + <http://blog.christianposta.com/deploy/blue-green-deployments-a-b-testing-and-canary-releases/>

1. Presentación (Tefa)  
   Buenos días, somos el grupo número 10 conformado por: [ronda de nombres]. Trabajaremos sobre el tema “Blue Green Deployment” - El objetivo de la presentación es continuar con lo ya presentado sobre el tema, pero con una perspectiva más práctica.
2. Arquitectura (Tomi)  
   Blue Green Deployment es una estrategia de despliegue de software para reducir tiempos de inactividad de los servicios. Se requiere de dos entornos de producción, blue y green. Sólo uno será activo y brindará servicios en determinado momento.
3. Arquitectura (Tomi)  
   El entorno inactivo espera un release. Los desarrolladores lo despliegan allí y se configura el balanceador de cargas para redirigir el tráfico de usuarios al entorno actualizado. El intercambio de estados hace del original un respaldo en caso de fallas, si no se elimina. El proceso puede repetirse.
4. Feature Toggle (Tomi).  
   Existe una técnica llamada feature toggle, utilizada para despliegue continuo. Se basa en ocultar funcionalidad incompleta, pero incluida en el release. Según Martin Fowler, ésta debiera ser la última alternativa para continuar el trabajo de desarrollo sin interrupciones.
5. Feature Branch (Tomi).  
   Una mejor opción es feature branch, que permite paralelismo sin los inconvenientes del código suelto. Esto se acerca a la filosofía de blue-green deployment, aplicando alta disponibilidad y releases bien definidos. Independiza de los tiempos de despliegue y reduce riesgos.
6. Aplicación Práctica (Ro)  
   Ahora vamos a analizar algunos ejemplos de aplicación práctica. Los usuarios pueden utilizar los servicios Web de Amazon, contratando infraestructura como servicio. Por ejemplo, Ruta 53 como DNS. Step Functions para dirigir el tránsito al entorno correcto.
7. Aplicación Práctica (Ro)  
   Container Service para instanciar cada entorno y un balanceador de carga en cada uno para ser direccionados. Esta infraestructura tiene un retraso de hasta 60 segundos por aplicar DNS. Funciona, pero es un mal ejemplo de aplicación. La siguiente es una mejor implementación.
8. Aplicación Práctica (Pia)

En primera instancia, existe una aplicación en un determinado entorno, azul.

La ruta configurada dirige todo el tráfico allí. Para un nuevo despliegue, se configura una ruta temporal para verificar la funcionalidad de la nueva versión en el otro entorno, verde.

1. Aplicación Práctica (Pia)

Con el visto bueno, se configura la ruta real para balancear el tráfico entre ambos entornos. Este es un periodo de transición momentáneo. Finalmente, para dejar de enviar requests al entorno inicial, se quita el mapeo de su ruta, y se desliga la ruta temporal.

1. Aplicación Práctica (Ro)  
   Como alternativa local, pueden crearse dos containers para delimitar los entornos dentro de un mismo nodo. Un servicio proxy se reconfigura con cada nuevo release para apuntar al entorno deseado. La base de datos debe ser compatible con ambas versiones.
2. Aplicación Práctica (Pia)

También puede asignarse un IP flotante en el data center, cada entorno como droplet/servidor individual. Con un despliegue sólo es necesario cambiar el droplet al cual apunta el IP para redireccionar el tráfico al entorno activo.

1. DOs and DON’Ts (Alex)

- Se presentan a continuación recomendaciones para implementar Blue-Green Deployment.

- Se debe Mantener el mismo balanceador de cargas para los entornos.Si se recibe grandes cantidades de datos, es probable que el nuevo balanceador no escale suficientemente rápido, perdiéndose paquetes.

1. DOs and DONa’Tos (Alex)

-Configurar las aplicaciones para que indiquen cuando están listas para procesar tráfico una vez cargadas en un entorno.Un inicio temprano resultaría en pérdida de paquetes.

-Establecido un nuevo entorno como activo, no eliminar la instancia restante automáticamente. Utilizar un timeout para posibilitar la finalización de peticiones prolongadas.

1. DOs and DON’Ts (Alex)  
   -No hacer el cambio de versiones modificando solo el DNS. De lo contrario, ambos entornos deberán estar activos hasta resolver la migración de clientes.

-NO Administrar todos los recursos con una única herramienta, combinar ventajas de aquellas especializadas para recursos susceptibles a cambios y otra para los no susceptibles.

1. Vs. Canary Deployment (Mark)  
   En comparación con Canary Deployment, ambas estrategias permiten introducir cambios en la funcionalidad mediante un despliegue seguro y rápido en infraestructura no ruteada. Esto se debe a que reducen la posibilidad de imprevistos en el ámbito de producción, con la disponibilidad de un rollback.
2. Vs. Canary Deployment (Mark)  
   Además, ambas facilitan la automatización del despliegue a través de herramientas locales o de infraestructura como servicio. Proveen flexibilidad para adaptarse a diversos contextos y control para seguir el proceso. Ambos deben preocuparse por compatibilidad de versiones con su base de datos.
3. Vs. Canary Deployment (Mark)  
   Su diferencia radica en el objetivo y su implementación. Blue-Green Deployment es binario: el cambio se introduce para todos los usuarios y el tráfico tiene una única vía. No existe una transición gradual.
4. Vs. Canary Deployment (Mark)  
   En cambio, Canary Deployment coloca a un número creciente de usuarios en una nueva versión, obteniendo un feedback sin impactar sobre la totalidad de los clientes. El tráfico es compartido entre versiones hasta su integración completa.
5. Conclusión (Tefa)  
   En conclusión, podemos decir que implementar la metodología blue-green deployment parece bastante tedioso. Sin embargo, es requerido en contextos que requieren de alta disponibilidad y un casi nulo tiempo de inactividad. El esfuerzo, más que nada, se centra en la estructuración inicial de las herramientas.
6. Conclusión (Tefa)  
   Existen alternativas pagas y gratuitas, que sólo requieren un poco de conocimiento sobre su configuración. Como resultado, se simplifica enormemente el despliegue, que es altamente sensible a errores inesperados durante el proceso, además de asegurar un usuario contento. Gracias