Engineer

Déploiement bleu-vert

◐ basculer le trafic réel entre deux environnements identiques

Ce que c’est

Le déploiement bleu-vert est une méthode de publication logicielle utilisant deux environnements de production aussi semblables que possible. L’un, appelé ici bleu, traite toutes les requêtes. La nouvelle version est installée sur l’autre, vert, puis testée sans qu’aucun trafic client ne l’atteigne. Lorsqu’elle est prête, on modifie le composant placé devant les deux pour que toute nouvelle requête aille vers le vert. Le bleu continue à fonctionner sans être modifié. Si le vert se comporte mal, la même modification en sens inverse remet le bleu en service.

Les couleurs n’ont aucune signification en elles-mêmes, et c’est leur intérêt. Des noms comme « ancien » et « nouveau » ou « principal » et « secours » changent de sens à chaque publication. Deux couleurs se contentent d’alterner : après la prochaine bascule, le vert est en service et le bleu au repos. Martin Fowler a décrit la technique sur son bliki en 2010 et attribue son nom, avec une certaine réserve, à Daniel Terhorst-North et Jez Humble. L’orthographe anglaise habituelle comporte un trait d’union, mais les outils ont leurs conventions : Argo Rollouts, contrôleur Kubernetes qui ajoute ce type de stratégie de déploiement, l’appelle `blueGreen` dans sa configuration.

Tout point unique qui détermine la destination des requêtes peut assurer la bascule. La version de Fowler modifie un routeur. Il peut tout aussi bien s’agir de la cible d’un répartiteur de charge, du serveur amont d’un mandataire inverse ou, dans Kubernetes, d’un Service dont le sélecteur d’étiquettes choisit les Pods recevant son trafic. Les deux environnements ne doivent pas nécessairement être deux centres de données : Fowler considère comme suffisants du matériel distinct, des machines virtuelles distinctes ou deux zones d’un même système avec leurs propres adresses IP.

Deux techniques voisines éclairent la différence. Une mise à jour progressive remplace quelques instances à la fois : anciennes et nouvelles versions répondent donc côte à côte jusqu'à la disparition de la dernière ancienne instance. Revenir en arrière revient à refaire cette progression dans l'autre sens. Une publication canari envoie d'abord une petite part du trafic vers la nouvelle version, puis l'élargit par étapes. Le déploiement bleu-vert déplace tout d'un coup, mais conserve une copie complète et active de la version précédente. Le retour se fait donc en une étape, de même ampleur que l'aller.

Utilisez le concept pour prendre une décision

Utilisez « Qu'est-ce que le déploiement bleu-vert ? » pour éclairer une décision réelle. Nous pouvons vous aider à relier ce concept au système que vous construisez ou achetez. Des changements testés, publiés et réversibles, pour livrer aux clients un mardi après-midi plutôt qu'un vendredi soir tard.

Découvrir nos prestations Contactez-nous →

Vérifiez les hypothèses derrière le terme

Appuyez-vous sur l'explication et les réalisations liées pour formuler les questions de votre projet. Nous pouvons examiner les hypothèses avec vous avant le choix d'un outil. Déploiement SaaS GIS automatisé : GitHub Actions — déploiement, données et reporting automatisés

Déploiement SaaS GIS automatisé : GitHub Actions Contactez-nous →

Pourquoi c'est important

Une publication devient un événement court et réversible plutôt qu'une opération longue à sens unique. Les étapes lentes — copie des fichiers, démarrage des processus, préchauffage des caches — se déroulent sur le vert pendant que le bleu continue à servir les clients, qui ne les attendent donc jamais. La bascule elle-même est rapide, tout comme son annulation : rien n'est reconstruit ni réinstallé, puisque la version précédente ne s'est jamais arrêtée. Cela change la manière dont une équipe envisage une publication un après-midi ordinaire et la rapidité avec laquelle elle peut agir si quelque chose semble anormal.

Le vert peut aussi être vérifié dans des conditions très proches de la production avant qu'un client le voie : même réseau, même configuration, même base de données. Un test de bon fonctionnement sur le vert démontre bien davantage que le même test dans une préproduction différente sur une douzaine de petits points. Argo Rollouts formalise cette approche avec un Service de prévisualisation qui dirige les requêtes vers la nouvelle version avant sa promotion, afin de la tester sans lui envoyer le trafic de production.

L'environnement au repos a une seconde fonction. Une fois la publication stabilisée, le côté qui vient de cesser de recevoir le trafic devient le lieu de préparation de la suivante. Entre-temps, il sert de secours prêt à prendre le relais. Selon Fowler, ce même mécanisme « vous permet de tester votre procédure de reprise après sinistre à chaque publication », puisque chaque bascule est un transfert vers le secours que quelqu'un observe.

Les coûts sont clairs. Deux environnements de production nécessitent la capacité de deux environnements, au moins pendant la publication. Si la couche applicative se duplique facilement, les données, elles, ne se dupliquent pas aussi simplement : le bleu et le vert partagent presque toujours une seule base de données. Chaque version publiée doit donc laisser des données lisibles par les deux versions. Une équipe incapable de le garantir obtient une bascule rapide vers l'avant et un retour arrière défectueux.

À quoi cela ressemble en pratique

Sur Kubernetes, la plus petite version fonctionnelle comprend deux Deployments et un Service. Les Deployments `web-blue` et `web-green` exécutent la même application avec les étiquettes `app: web` et `slot: blue` ou `slot: green` . Le Service atteint par les clients sélectionne un emplacement :

```
apiVersion: v1
kind: Service
metadata:
  name: web
spec:
  selector:
    app: web
    slot: blue
  ports:
    - port: 80
      targetPort: 8080
```

Une publication place la nouvelle image sur le côté au repos, attend qu'il soit prêt, le teste par un second Service qui sélectionne uniquement `slot: green` , puis fait basculer celui qui reçoit le trafic réel :

```
kubectl set image deployment/web-green web=registry.example.com/web:1.5.0
kubectl rollout status deployment/web-green --timeout=5m
# smoke tests against the web-green preview Service go here
```

```
kubectl patch service web -p '{"spec":{"selector":{"slot":"green"}}}'
# the way back is the same command with the other colour
kubectl patch service web -p '{"spec":{"selector":{"slot":"blue"}}}'
```

`kubectl patch` utilise un correctif par fusion stratégique si aucun type n'est précisé. Seule la clé `slot` du sélecteur change donc, et `app: web` reste en place. Le contrôleur de Service recherche continuellement les Pods correspondant au sélecteur et actualise les points de terminaison du Service lorsque cet ensemble change. Les nouvelles connexions arrivent ainsi sur le vert, tandis que le bleu reste prêt avec l'image précédente.

Argo Rollouts automatise la même chorégraphie avec un objet `Rollout` : un `activeService` qu'il dirige vers les nouveaux pods lors de la promotion et un `previewService` facultatif qui les atteint avant. Avec `autoPromotionEnabled: false`, le déploiement se met en pause avant la bascule jusqu'à ce que quelqu'un exécute `kubectl argo rollouts promote web`.

La prestation DevOps et CI/CD promet des changements testés, publiés et réversibles. Le déploiement bleu-vert est une manière de rendre le dernier de ces engagements peu coûteux. Il repose sur deux habitudes illustrées par les déploiements d'une plateforme sur Azure, sans aucun second emplacement de production : les modèles Bicep maintiennent une infrastructure cible identique d'un environnement à l'autre, et le même artefact construit est promu du développement à la préproduction puis à la production, en récupérant les réglages de chaque environnement à l'arrivée. Ce parcours fait avancer une version à travers une succession d'environnements. Le déploiement bleu-vert lui ajoute un jumeau à l'extrémité. Les deux mêmes habitudes — infrastructure identique et artefact unique inchangé — permettent à l'équipe de croire que ce jumeau en est réellement un.

Où les problèmes surviennent

Un changement de schéma transforme le retour en impasse. Le vert démarre, exécute une migration renommant une colonne et traite le trafic sans difficulté. Dix minutes plus tard, un défaut renvoie le trafic vers le bleu, qui demande encore l'ancien nom de colonne et échoue à chaque requête. Le retour arrière a été prévu pour l'application, mais pas pour les données. Le conseil de Fowler de 2015 consiste à séparer les deux : publier un changement de base de données compatible avec les deux versions, puis l'application, et supprimer ce dont seule l'ancienne version avait besoin une fois la nouvelle stabilisée.

L'état écrit sur le vert devient inaccessible après un retour au bleu. Tout ce que le vert conserve pour lui-même — fichiers téléversés sur son disque local, stockage de sessions en mémoire ou cache propre de tâches en attente — manque sur le bleu lorsque le trafic y revient. Les utilisateurs se retrouvent déconnectés ou ne retrouvent plus ce qu'ils viennent d'enregistrer. Conservez l'état partagé dans des stockages utilisés par les deux couleurs. Lorsque le vert doit avoir ses propres données, Fowler suggère d'alimenter les deux environnements avec les transactions ou de démarrer le vert en lecture seule, puis de le passer en lecture-écriture seulement après qu'il a fait ses preuves.

Le retour arrière disparaît lors de la réduction de capacité, avant que le défaut n'apparaisse. Après la bascule, Argo Rollouts réduit le nombre d'instances de l'ancien ReplicaSet, et `scaleDownDelaySeconds` vaut 30 par défaut. Un défaut qui apparaît au bout d'un quart d'heure — une tâche nocturne ou une fuite lente — ne trouve plus de bleu vers lequel revenir. Fixez un délai couvrant le temps habituellement nécessaire à l'apparition des problèmes, et acceptez que le prix à payer soit cette capacité :

```
strategy:
  blueGreen:
    activeService: web
    previewService: web-preview
    autoPromotionEnabled: false
    scaleDownDelaySeconds: 3600
```

La promotion suit l'état prêt, pas le bon fonctionnement. `autoPromotionEnabled` vaut true par défaut : Argo Rollouts bascule donc dès que les nouveaux pods sont prêts. Une sonde de disponibilité indique qu'un processus répond, pas que le passage en caisse fonctionne. Faites une pause avant la promotion, comme dans le bloc ci-dessus, et testez par le Service de prévisualisation. Vous pouvez aussi ajouter une `prePromotionAnalysis`, qui abandonne le déploiement en cas d'échec, et une `postPromotionAnalysis`, qui revient à la version précédente en cas d'échec après la bascule.

Le côté au repos dérive. Entre deux publications, quelqu'un applique une modification urgente de configuration uniquement à l'environnement en service. La version suivante arrive sur le côté au repos, qui conserve encore l'ancien réglage. La bascule censée changer une chose en change alors deux. Construisez les deux emplacements avec les mêmes modèles et la même source de

configuration, afin qu'une modification de l'un soit une modification des deux. Traitez le côté au repos comme le lieu de préparation de la prochaine publication, ainsi que le décrit Fowler.

Pour aller plus loin

- Déploiement bleu-vert — la description de Martin Fowler de 2010, avec sa note de 2015 sur les changements de base de données.
- Argo Rollouts : BlueGreen — Services actif et de prévisualisation, promotion automatique, analyse avant et après la bascule et délai de réduction du nombre d'instances.
- Service Kubernetes — la manière dont un sélecteur choisit les Pods qui reçoivent le trafic d'un Service.
- kubectl patch — correctifs par fusion stratégique, par fusion et JSON ; la commande qui change le sélecteur dans l'exemple ci-dessus.



Glossaire

Choisissez selon vos contraintes réelles

Présentez le parcours de déploiement, les risques opérationnels et les personnes d'astreinte. Nous pouvons définir un changement avec des vérifications, une procédure de reprise et une responsabilité claire.

Consulter les tarifs Contactez-nous →

Rendez la terminologie concrète

Demandez comment « Qu'est-ce que le déploiement bleu-vert ? » s'applique à votre organisation. Un exemple concret tiré de votre système actuel suffit pour engager la discussion.

Découvrir nos prestations

welcome@engineer.company +45 50 10 61 84 Copenhague, Danemark

Emportez cette page avec vous

Fiche de contact

Créé avec soin et amour par D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Droit d'auteur © 2025 – aujourd'hui · Engineer ApS · Tous droits réservés.

Imprimé depuis https://engineer.company/fr/glossary/blue-green-deployment/