3 - Déploiement "Blue/Green" vs "Canary"

3. Déploiement « Blue/Green » vs « Canary »

3.1 Déploiement « Blue / Green »

Le déploiement **« blue/green »** est une technique qui **réduit les temps d'arrêt et les risques** en exécutant **deux environnements de production** identiques appelés **« Blue » et « Green »**.

À tout moment, **un seul des deux environnements peut être accessible.** Dans l'exemple ci-dessous, l'environnement servant l'ensemble du trafic de production est le « Blue » en version N alors que l'environnement « Green » en version N+1 est inactif.

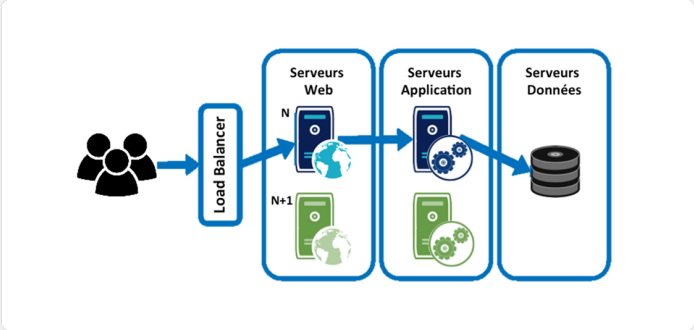


Figure 1 : L'environnement Blue est actuellement utilisé

Lorsque vous **préparez une nouvelle version de votre logiciel**, le **déploiement et la phase finale de test** se déroulent **dans l'environnement qui n'est pas en direct** : dans cet exemple, **Green**. Une fois que vous avez déployé et testé le logiciel en vert, vous changez la configuration du load-balancer pour que toutes les demandes entrantes passent de l'environnement « bleue » au « vert ». Le vert est maintenant exposé directement et le bleu est inactif.

Cette technique permet d'**éliminer les temps d'arrêt dûs au déploiement de l'application**, on parle de **"zéro downtime"**. De plus, le **déploiement "Blue/Green" réduit le risque** : si quelque chose d'**inattendu se produit** avec votre nouvelle version sur l'environnement **"Green"**, vous pouvez **immédiatement revenir à la dernière version** en exposant à nouveau les services sur **"Blue"**.

Le seul **point négatif** de cette approche est qu'elle vous **force à maintenir deux environnements de production en parallèle.**

Donc, dans le cas d'**applications très consommatrices de ressources**, cette **technique** sera **assez coûteuse**.

3.2 Amélioration de l'approche « Blue / Green »

a. Exposition progressive

La technique de l'exposition progressive apporte plus de **souplesse** tout en **réduisant les risques de bug de façon significative.**

Le principe de l'**exposition progressive** est assez **simple** : la **nouvelle version de l'application** est rendue disponible à un **nombre restreint d'utilisateurs**. Plus elle est **stable** et plus on est **satisfait de sa réactivité**, plus elle est **accessible** à un **nombre important** d'utilisateurs, jusqu'à une **expositon** complète à l'**ensemble des utilisateurs**.

Dans la pratique, une fois que l'application est déployée dans l'environnement Green, un petit nombre d'utilisateurs sont redirigés vers cette dernière, tandis que l'environnement Blue continue à recevoir le gros du trafic.
Il faut donc mettre en place des **indicateurs de télémétrie** pour **évaluer la nouvelle application.** Mais également un **Load Balancer** pour **gérer la répartition des utilisateurs entre les deux environnements.**

Au final, en procèdant ainsi vous vous retrouverez toujours avec une **version stable** et **éprouvée** en **production**.

b. Gestion des bases de données

La principale problématique de cette approche vient du **partage de la même base de données** pour les deux environnements. Il va donc se poser la question de la **gestion des évolutions de la base de données.**

Pour résoudre cette problématique, les équipes vont devoir systématiquement **maintenir une matrice de compatibilité** entre **chaque schéma** de la base de données afin que **deux versions de l'application** puissent **tourner simultanément**.

Par conséquent, pour ajouter un nouveau schéma de base de données, il va falloir **refactoriser le schéma** afin de faire fonctionner **2 versions en parallèle**.

3.3 Déploiement de type « Canary »

Les **déploiements** de type **"canary"** sont un **modèle de déploiement applicatif** à un **sous-ensemble d'utilsateurs** et de **serveurs**. L'idée est d'abord de **déployer le changement** sur un **petit sous-ensemble** de serveurs, de **le tester**, puis de **déployer le changement progressivement** sur le **reste** des serveurs.

Le déploiement "canary" sert d'indicateur d'alerte tout en gardant un impact réduit sur les temps d'arrêt : si le déploiement échoue, seule une petite partie des serveurs est affectée.

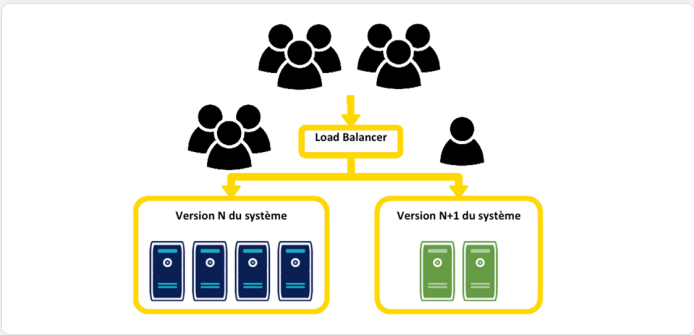


Figure 2 : Déploiement « canary » de la version N+1

Le nom « canary » provient du nom de l'oiseau dans le temps où ils étaient régulièrement utilisés dans les mines de charbon comme **système d'alerte précoce**. Les gaz toxiques comme le monoxyde de carbone, le méthane ou le dioxyde de carbone dans la mine tueraient l'oiseau avant d'affecter les mineurs. Les signes de détresse de l'oiseau indiquaient aux mineurs que les conditions n'étaient pas sécuritaires.

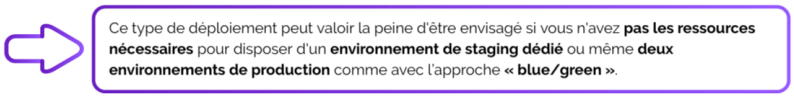
Par **exemple**, imaginez un **environnement** qui a **4 serveurs web.**
Plutôt que de se déployer simplement sur tous les serveurs de l'environnement, un déploiement de type **« canary »** ressemblerait à ceci :

- Déploiement sur un ou plusieurs serveurs
- Monitorer et tester jusqu'à être satisfait
- Déploiement progressif sur les serveurs restants

La phase de test du déploiement des "canaris" peut fonctionner de plusieurs façons.
Vous pouvez **exécuter des tests automatisés,** effectuer des **tests manuels vous-même,** ou même **laisser le serveur en direct et attendre de voir si les utilisateurs finaux rencontrent des problèmes.** En fait, ces trois approches pourraient être utilisées conjointement.

En fonction de la manière dont vous prévoyez de tester, vous pouvez décider de retirer le serveur « canary » de l'équilibreur de charge de production et de n’envoyer le trafic dessus que lors du déploiement à l’ensemble des serveurs.

Les déploiements de type « canary » sont similaires à l'**utilisation d'un environnement de staging** (dernier environnement avant la production). La différence est que les **environnements de staging sont généralement dédiés à la qualité**, en effet un serveur de staging ne devient pas un serveur de production. En revanche, dans un déploiement « canary », **le serveur fait partie de la flotte de production lorsque le déploiement est terminé.**



C’est pour cela que **cette méthode de déploiement** est souvent **mise en place par les géants du web.** On imagine mal Google ou Amazon doubler leur flotte de serveurs de production.

Pour voir la fiche complète et les documents attachés, rendez-vous sur
https://elearning.26academy.com/course/play/5aa2661754470b34d4e474f9