

container

kubernetes

cm

Cours: Stratégies de Déploiement dans Kubernetes

Introduction

- Ce cours explore les différentes stratégies de déploiement dans Kubernetes pour garantir des mises à jour sans interruption et une gestion efficace des applications.

Vue d'ensemble de Kubernetes

- Kubernetes est un orchestrateur de conteneurs open-source qui facilite le déploiement, la mise à l'échelle et la gestion d'applications conteneurisées. Il utilise des concepts tels que les pods, les services et les déploiements.

Chapitre 1: Introduction aux Stratégies de Déploiement

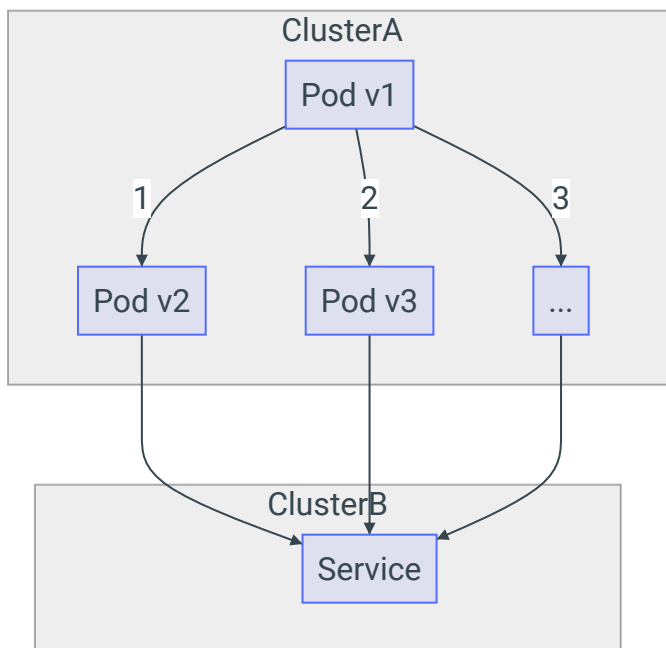
- Les stratégies de déploiement définissent comment les nouvelles versions d'applications sont déployées. Objectifs : réduction des temps d'arrêt, gestion des risques, amélioration de la disponibilité.

Chapitre 2: Rolling Updates

- Les Rolling Updates consistent à mettre à jour les pods progressivement, un par un, pour minimiser l'impact sur la disponibilité de l'application.
- Kubernetes remplace progressivement les anciennes instances de pods par les nouvelles, assurant ainsi une transition en douceur.
- Les Rolling Updates mettent à jour progressivement les pods sans interruption de service.
- Avantages: Réduction des temps d'arrêt, transition en douceur, possibilité de revenir en arrière en cas d'échec.
- Limitations: Nécessite une gestion minutieuse des versions pour éviter des conflits potentiels.
- **Exemple de Code:**

```
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: mon-app
5  spec:
6    replicas: 3
7    template:
8      metadata:
9        labels:
10         app: mon-app
11      spec:
12        containers:
13         - name: mon-conteneur
14           image: mon-image:v2
15    strategy:
16      type: RollingUpdate
17      rollingUpdate:
18        maxSurge: 1
19        maxUnavailable: 1
```

- Explication du code :
- `maxSurge` : Nombre maximum de pods supplémentaires autorisés pendant la mise à jour.
- `maxUnavailable` : Nombre maximum de pods non disponibles pendant la mise à jour.
- **Exemple d'architecture:**



Explication du diagramme :

- a. Le déploiement initial a plusieurs pods de la version 1 (Pod v1).

- b. Lors d'une mise à jour, un nouveau pod de la version 2 (Pod v2) est ajouté tout en conservant les pods existants.
- c. Ce processus se répète progressivement, assurant une transition en douceur.

Chapitre 3: Blue-Green Deployments

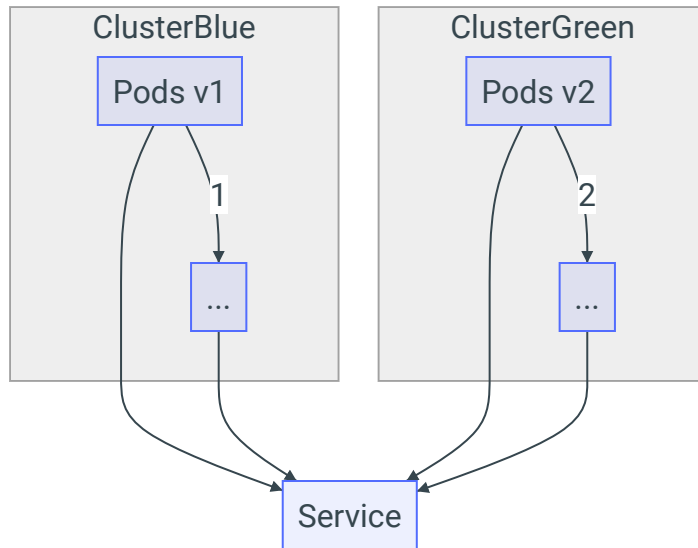
- Avantages: Zéro temps d'arrêt, possibilité de tester la nouvelle version avant le basculement complet.
- Limitations: Requier des ressources pour maintenir deux environnements simultanément.
- Cette stratégie maintient deux environnements (Blue et Green), permettant un basculement rapide entre les versions.
- **Exemple de Code:**

```
1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: mon-app-service
5  spec:
6    selector:
7      app: mon-app
8    ports:
9      - protocol: TCP
10      port: 80
11      targetPort: 8080
12  ---
13  apiVersion: apps/v1
14  kind: Deployment
15  metadata:
16    name: mon-app-blue
17  spec:
18    replicas: 3
19    selector:
20      matchLabels:
21        app: mon-app
22    template:
23      metadata:
24        labels:
25          app: mon-app
26          env: blue
27      spec:
28        containers:
29          - name: mon-conteneur
30            image: mon-image:v1
```

- Explication du code :

Création d'un service et d'un déploiement pour l'environnement Blue. Un fichier similaire doit être créé pour l'environnement Green.

- **Exemple d'architecture:**



Explication du diagramme :

- a. L'environnement Blue est en production avec plusieurs pods de la version 1.
- b. L'environnement Green est préparé avec la version 2.
- c. Le basculement est effectué en redirigeant le service vers l'environnement Green.

Chapitre 4: A/B Testing

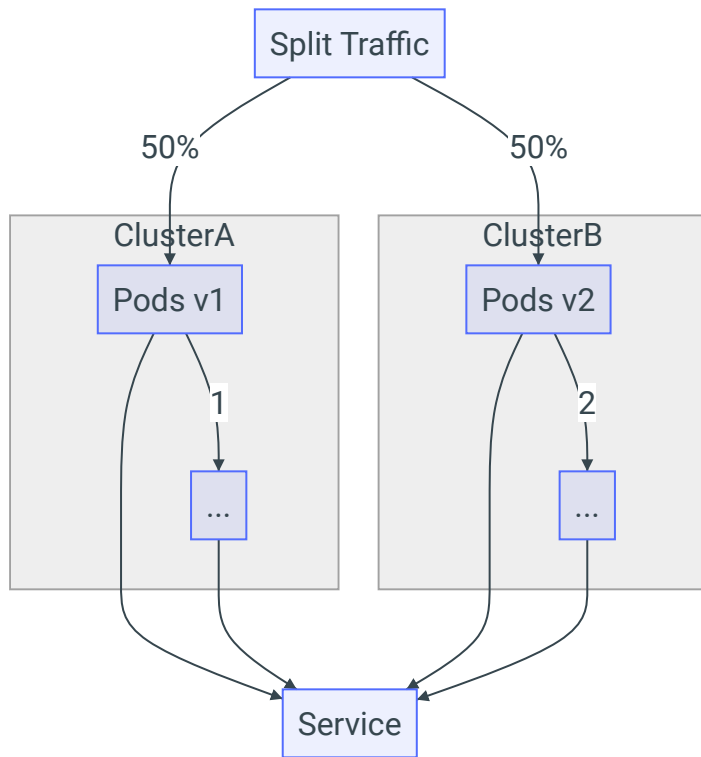
- **Avantages:** Évaluation précise des performances, prise de décision basée sur des données réelles.
- **Limitations:** Nécessite une gestion fine des groupes d'utilisateurs, peut augmenter la complexité.
- Cette stratégie expose différents groupes d'utilisateurs à différentes versions de l'application simultanément.

- **Exemple de Code:**

```
1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: mon-app-service
5  spec:
```

```
6   selector:
7     app: mon-app
8   ports:
9     - protocol: TCP
10       port: 80
11       targetPort: 8080
12
13   ---
14
15   apiVersion: apps/v1
16   kind: Deployment
17   metadata:
18     name: mon-app-a
19   spec:
20     replicas: 3
21     selector:
22       matchLabels:
23         app: mon-app
24         version: a
25     template:
26       metadata:
27         labels:
28           app: mon-app
29           version: a
30     spec:
31       containers:
32         - name: mon-conteneur
33           image: mon-image:version-a
```

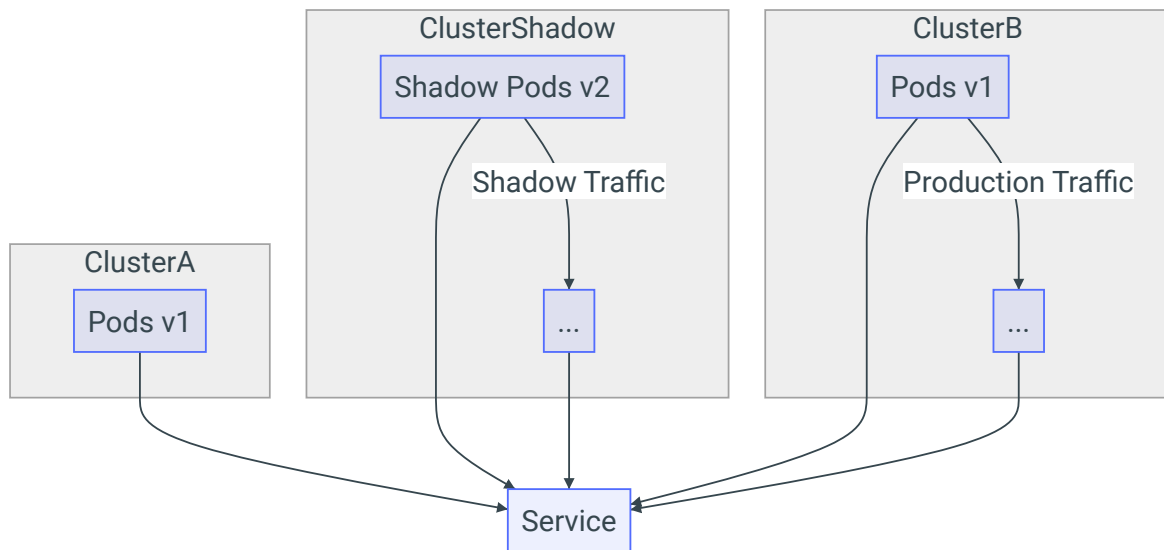
- Explication du code :
- Création d'un service et d'un déploiement pour la version A. Un fichier similaire doit être créé pour la version B.
- **Exemple d'architecture:**

**Explication du diagramme :**

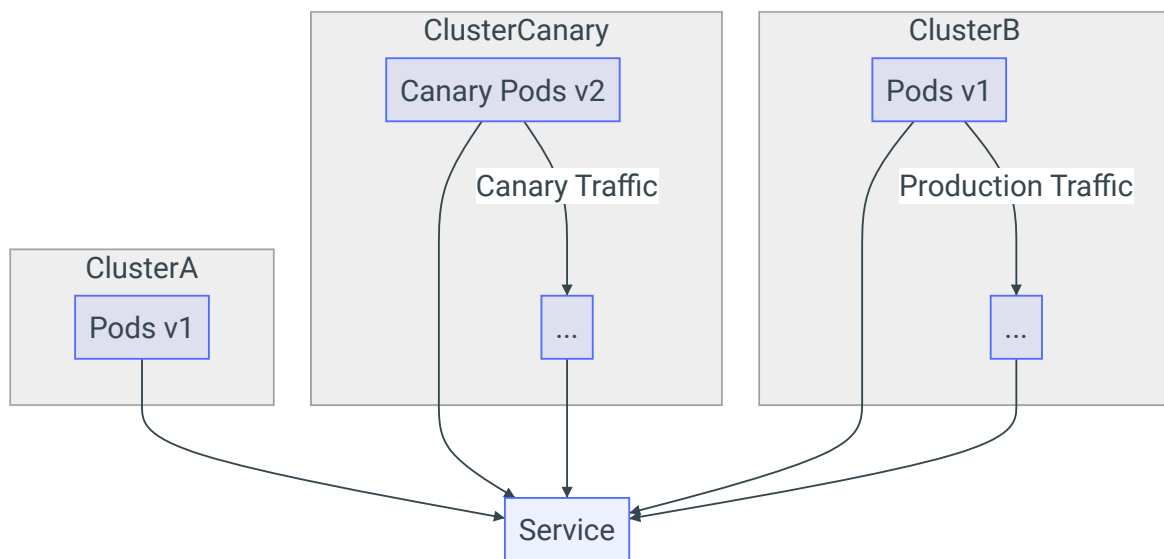
- Deux environnements sont préparés, l'un avec la version A et l'autre avec la version B.
- Le trafic est divisé entre les deux versions (50% chacune) pour l'A/B Testing.

Quelques exemples d'autres stratégies de déploiement

Shadow Deployment

**Explication du diagramme :**

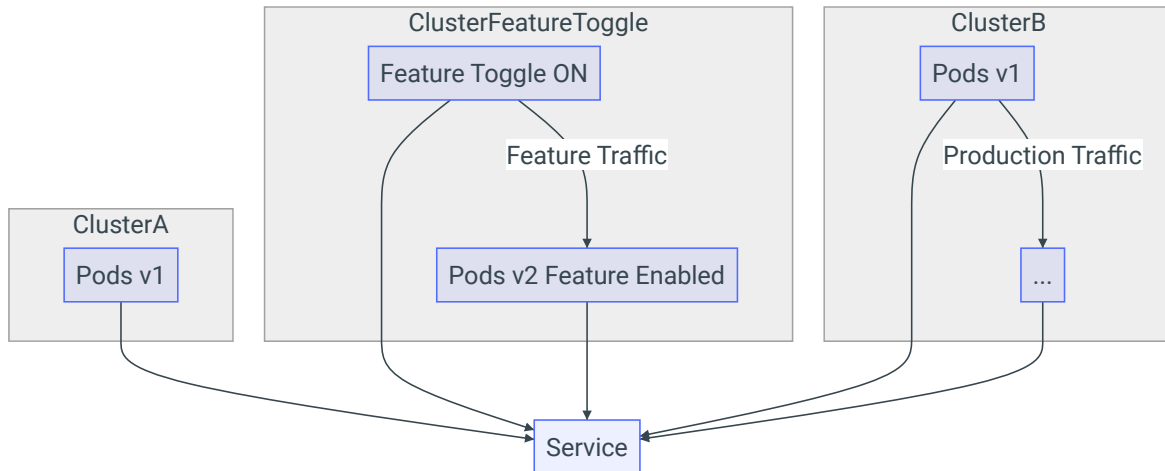
- Les Shadow Pods (B) répliquent le comportement de la version v2, mais le trafic réel est dirigé vers la version v1 (D). Ceci permet de tester la nouvelle version en conditions réelles sans affecter les utilisateurs.

Canary Deployment**Explication du diagramme :**

- Les Canary Pods (B) représentent la version v2 et reçoivent une partie du trafic (Canary Traffic) pour tester la nouvelle version avec un sous-ensemble d'utilisateurs avant un déploiement

complet.

Feature Toggles



Explication du diagramme :

- Le Feature Toggle (B) permet d'activer ou de désactiver une fonctionnalité spécifique. Lorsqu'il est activé, le trafic est dirigé vers la version v2 avec la fonctionnalité activée (C).

Contexte de chaque type de déploiement

1. Rolling Updates:

Exemple d'Application : Une application web de e-commerce.

Pourquoi : Les mises à jour régulières sont essentielles pour corriger des bugs, améliorer les performances et ajouter de nouvelles fonctionnalités. Dans le e-commerce, une stratégie de rolling update permet de minimiser les interruptions de service tout en garantissant que les utilisateurs bénéficient constamment des dernières améliorations.

2. Blue-Green Deployments:

Exemple d'Application : Un système de gestion de contenu (CMS) utilisé pour la publication de contenu en ligne.

Pourquoi : Les déploiements Blue-Green sont particulièrement utiles pour les applications nécessitant une haute disponibilité et une gestion des mises à jour en douceur. Dans le cas d'un CMS, cette stratégie permet de basculer instantanément entre deux environnements, garantissant une transition sans heurts lors de la publication de nouveaux contenus ou de mises à jour.

majeures.

3. A/B Testing:

Exemple d'Application : Une application mobile de réseau social.

Pourquoi : Les applications de réseau social cherchent souvent à améliorer l'engagement des utilisateurs et à tester de nouvelles fonctionnalités. En utilisant la stratégie A/B Testing, l'application peut tester différentes interfaces utilisateur, algorithmes de recommandation, ou méthodes d'affichage de contenu pour évaluer ce qui fonctionne le mieux en fonction du comportement réel des utilisateurs.

4. Shadow Deployment:

Exemple d'Application : Un système de traitement de paiement en ligne.

Pourquoi : Les déploiements de l'ombre sont utiles pour tester de nouvelles versions dans un environnement de production sans affecter les transactions réelles. Dans un système de traitement de paiement, cela peut être crucial pour garantir la stabilité du système tout en introduisant des mises à jour de sécurité ou de nouvelles fonctionnalités.

5. Canary Deployment:

Exemple d'Application : Un service de streaming vidéo en ligne.

Pourquoi : Les services de streaming vidéo cherchent constamment à améliorer la qualité de la diffusion et à introduire de nouvelles fonctionnalités. En utilisant la stratégie de déploiement Canary, l'application peut tester de nouvelles versions avec un groupe restreint d'utilisateurs avant de déployer la mise à jour à l'ensemble de la base d'utilisateurs, minimisant ainsi les risques de problèmes de performance ou d'expérience utilisateur.

6. Feature Toggles:

Exemple d'Application : Un logiciel de gestion de projet.

Pourquoi : Dans un logiciel de gestion de projet, différentes équipes peuvent travailler sur des fonctionnalités distinctes en parallèle. En utilisant des Feature Toggles, ces fonctionnalités peuvent être intégrées au produit principal, mais activées ou désactivées dynamiquement en fonction des besoins du client ou de la planification de la sortie. Cela permet une gestion flexible des fonctionnalités sans nécessiter de déploiements distincts.

Récapitulatif des Points Clés

1. **Importance des Stratégies de Déploiement** : Nous avons compris l'importance cruciale des stratégies de déploiement dans un contexte Kubernetes. Ces stratégies offrent des moyens structurés et intelligents pour introduire de nouvelles fonctionnalités, appliquer des correctifs et améliorer les performances sans causer d'interruptions majeures.
2. **Rolling Updates** : La stratégie de Rolling Updates permet une mise à jour progressive des pods, garantissant une transition en douceur vers une nouvelle version tout en maintenant la disponibilité des services.
3. **Blue-Green Deployments** : Avec les Blue-Green Deployments, nous avons découvert comment maintenir deux environnements distincts, permettant des basculements instantanés entre les versions, minimisant ainsi les risques et assurant une continuité des services.
4. **A/B Testing** : La stratégie A/B Testing a été explorée comme un moyen efficace de tester différentes versions d'application simultanément, en évaluant leur impact réel sur des groupes d'utilisateurs spécifiques.
5. **Stratégies Avancées** : Nous avons abordé des concepts avancés tels que les Shadow Deployments et les Feature Toggles, offrant des approches flexibles pour tester de nouvelles fonctionnalités et introduire des changements en douceur.
6. **Bonnes Pratiques** : Enfin, nous avons exploré les bonnes pratiques en matière de sécurité des déploiements, de gestion des configurations et de mise en œuvre de stratégies de sauvegarde.

QCM

QCM sur les Stratégies de Déploiement dans Kubernetes

Question 1: Quelle est l'objectif principal des stratégies de déploiement dans Kubernetes ?

- a) Minimiser l'utilisation des ressources.
- b) Garantir des mises à jour sans interruption et une gestion efficace des applications.
- c) Automatiser la création de clusters Kubernetes.

Question 2: Quelle stratégie de déploiement est utilisée pour mettre à jour progressivement les pods sans interruption de service ?

- a) Blue-Green Deployments.
- b) Canary Deployment.
- c) Rolling Updates.

Question 3: Dans une stratégie de Rolling Updates, qu'est-ce que `maxSurge` contrôle ?

- a) Le nombre maximum de pods indisponibles pendant la mise à jour.
- b) Le nombre maximum de pods supplémentaires autorisés pendant la mise à jour.
- c) Le pourcentage du trafic dirigé vers la nouvelle version.

Question 4: Quel est l'avantage principal des Blue-Green Deployments ?

- a) Maximiser l'utilisation des ressources.
- b) Permettre un basculement instantané entre deux environnements.
- c) Tester différentes versions d'application simultanément.

Question 5: Pourquoi la stratégie A/B Testing est-elle souvent utilisée dans le développement d'applications ?

- a) Pour minimiser les coûts de déploiement.
- b) Pour tester différentes versions d'application simultanément avec des groupes d'utilisateurs distincts.
- c) Pour automatiser les déploiements Kubernetes.

Question 6: Quelle stratégie permet de tester une nouvelle version dans un environnement de production sans affecter les utilisateurs réels ?

- a) Shadow Deployment.
- b) Canary Deployment.
- c) Feature Toggles.

Question 7: Quel est l'objectif principal des Feature Toggles ?

- a) Activer ou désactiver des fonctionnalités dynamiquement.
- b) Mettre à jour progressivement les pods sans interruption de service.
- c) Basculement instantané entre deux environnements.

Question 8: Pourquoi les Canary Deployments sont-ils utiles dans le contexte des applications de streaming vidéo en ligne ?

- a) Pour tester différentes versions d'application simultanément.
- b) Pour minimiser les interruptions de service lors des mises à jour.
- c) Pour garantir une haute disponibilité du service.

Question 9: Quel type de déploiement est souvent utilisé pour tester une nouvelle fonctionnalité en

production sans déployer une nouvelle version complète de l'application ?

- a) Rolling Updates.
- b) Shadow Deployment.
- c) Blue-Green Deployments.

Question 10: Quelle stratégie est particulièrement adaptée à la gestion de projets avec plusieurs équipes travaillant sur des fonctionnalités distinctes ?

- a) A/B Testing.
- b) Feature Toggles.
- c) Shadow Deployment.

Merci de votre attention

Last update: December 7, 2023 14:05:49