

RED HAT® OPENSIFT Container Platform

Module 1

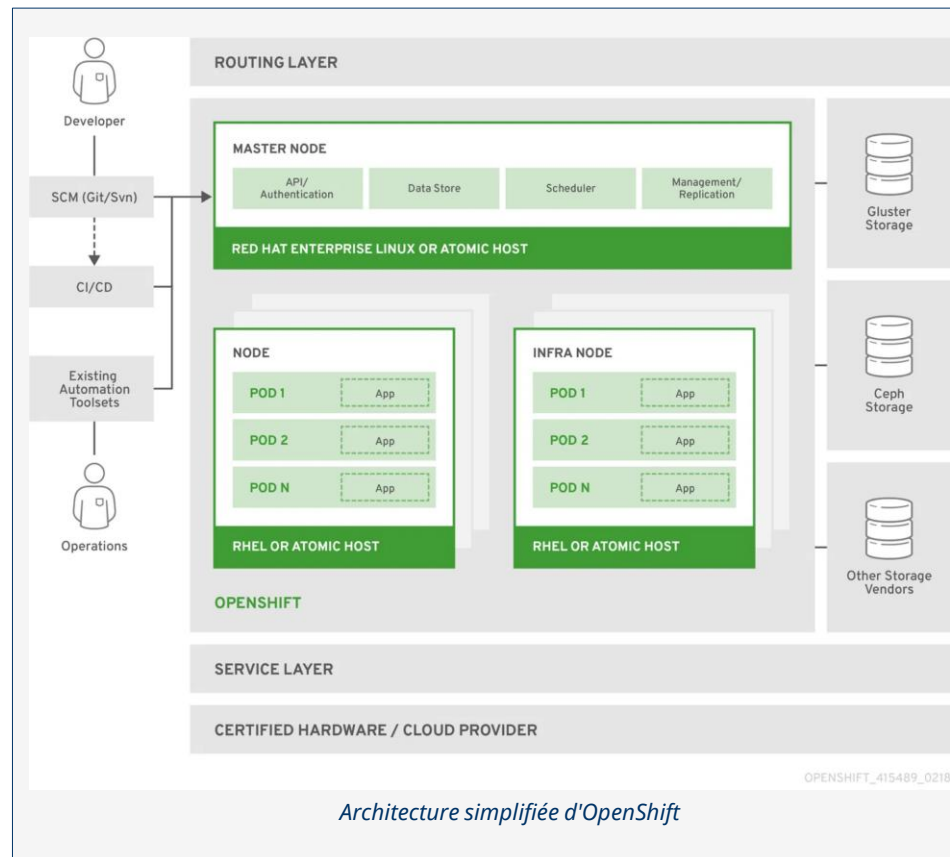
Déploiement et gestion d'applications sur un cluster OpenShift

Construction d'images, déploiement et gestion d'applications

Objectifs du module

À la fin de ce module, vous serez capable de :

- Construire des images** pour vos applications en utilisant les meilleures pratiques
- Déployer des applications** sur un cluster OpenShift en utilisant différentes méthodes
- Gérer des applications** déployées sur OpenShift (surveillance, scaling, configuration)
- Migrer des applications existantes** vers un environnement OpenShift
- Appliquer les bonnes pratiques** pour des déploiements sécurisés et efficaces



Aperçu du module

Ce module couvre les trois aspects fondamentaux du déploiement d'applications sur OpenShift :

🔧 1. Construction d'images pour une application

Création d'images de conteneurs optimisées pour vos applications en utilisant Dockerfile, Docker et Podman. Application des bonnes pratiques pour des images légères et sécurisées.

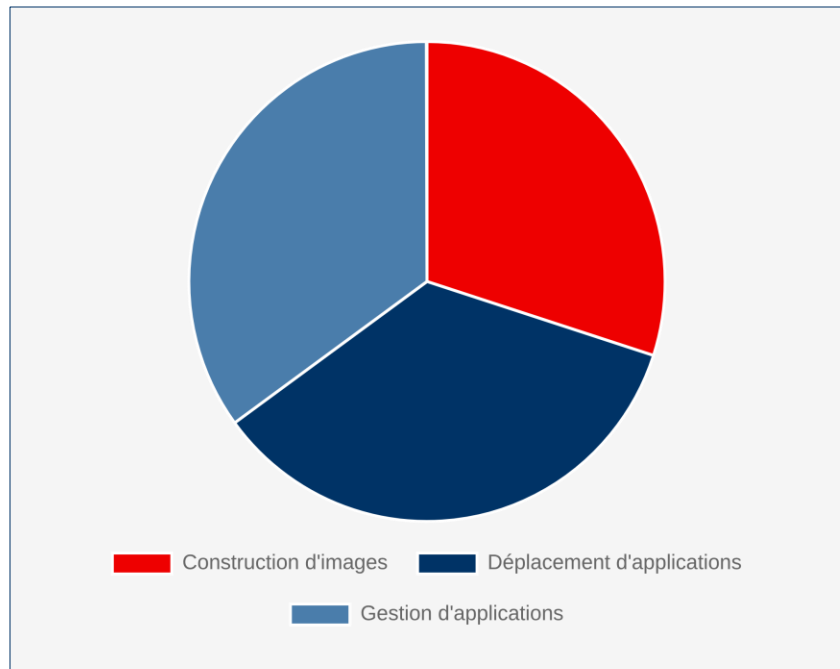
🚀 2. Déplacement d'une application vers OpenShift

Méthodes de déploiement sur OpenShift, utilisation de la commande **oc new-app**, déploiement depuis des images existantes et stratégies de migration.

⚙️ 3. Gestion d'une application sur OpenShift

Surveillance et scaling d'applications, gestion des configurations et secrets, stratégies de déploiement et meilleures pratiques pour la maintenance.


Répartition du temps par section





Introduction à la construction d'images


La construction d'images de conteneurs est une étape fondamentale pour déployer des applications sur OpenShift. Une image bien conçue garantit performance, sécurité et facilité de maintenance.


Concepts clés

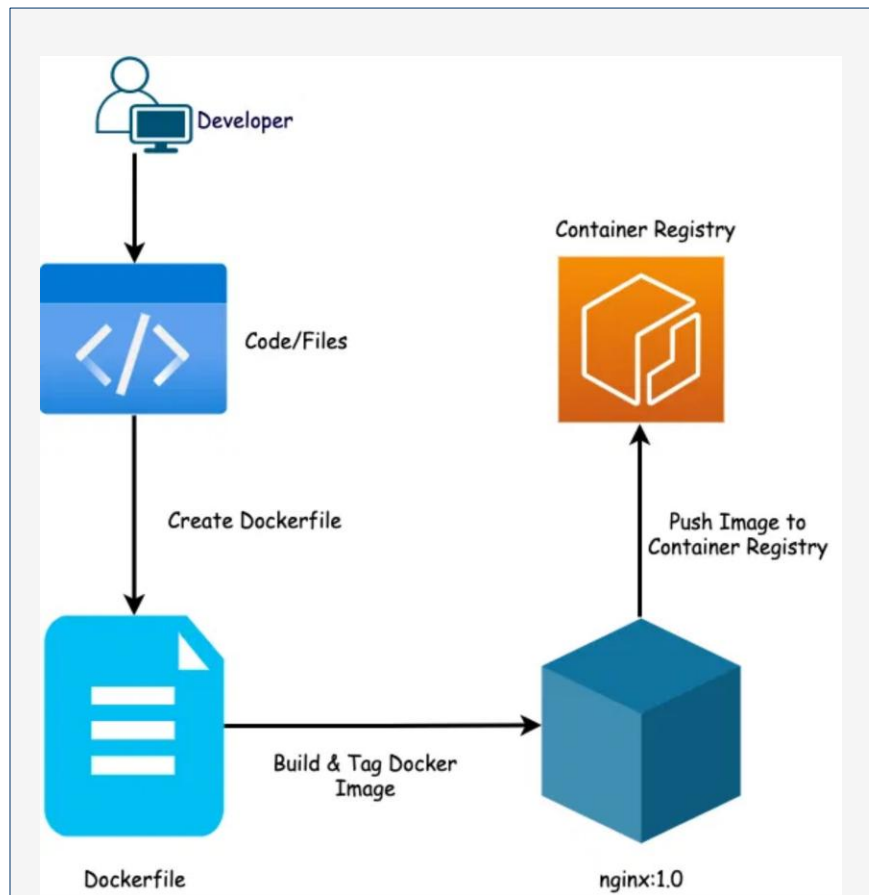
 **Image de conteneur** : Package autonome contenant l'application, ses dépendances et la configuration nécessaire pour l'exécuter

 **Dockerfile** : Fichier texte contenant les instructions pour construire une image

 **Couches (layers)** : Chaque instruction dans un Dockerfile crée une couche immuable, permettant la réutilisation et l'optimisation

 **Image de base** : Image parent sur laquelle votre image est construite (ex: ubuntu, alpine, ubi8)

 **Registre** : Service de stockage et de distribution d'images de conteneurs (ex: Docker Hub, Quay.io, registre interne OpenShift)








Processus de construction d'une image de conteneur

Dockerfile et bonnes pratiques

Un **Dockerfile** est un fichier texte contenant une série d'instructions pour construire une image Docker. Voici les bonnes pratiques essentielles :

✓ Bonnes pratiques

-  **Utiliser des images de base officielles** et minimales (alpine, slim)
-  **Combiner les commandes RUN** pour réduire le nombre de couches
-  **Nettoyer les caches** après installation des paquets
-  **Éviter d'utiliser root** en créant un utilisateur dédié
-  **Utiliser des tags spécifiques** pour les images de base, jamais latest

Construction de l'image

Construire l'image avec un tag

```
docker build -t mon-app:1.0
```

Alternative avec Podman

```
podman build -t mon-app:1.0
```

Exemple de Dockerfile optimisé

Utiliser une image de base officielle avec tag spécifique

```
FROM node: 16-alpine
```

Définir des variables d'environnement

```
ENV NODE_ENV= production
```

```
ENV APP_HOME= /app
```

Créer un utilisateur non-root

```
RUN addgroup -S appgroup && adduser -S appuser -G appgroup
```

Définir le répertoire de travail

```
WORKDIR ${APP_HOME}
```

Copier les fichiers de dépendances

```
COPY package*.json ./
```

Installer les dépendances en une seule couche

```
RUN npm ci --only=production && npm cache clean --force
```

Copier le code source de l'application

```
COPY --chown=appuser:appgroup . .
```

Changer vers l'utilisateur non-root

```
USER appuser
```

Exposer le port et définir la commande de démarrage

```
EXPOSE 8080
```

```
CMD ["node", "server.js"]
```

Construction d'images avec Docker et Podman

Deux outils principaux sont disponibles pour construire des images de conteneurs à partir d'un Dockerfile :

Docker

Outil historique et le plus répandu pour la construction et l'exécution de conteneurs.

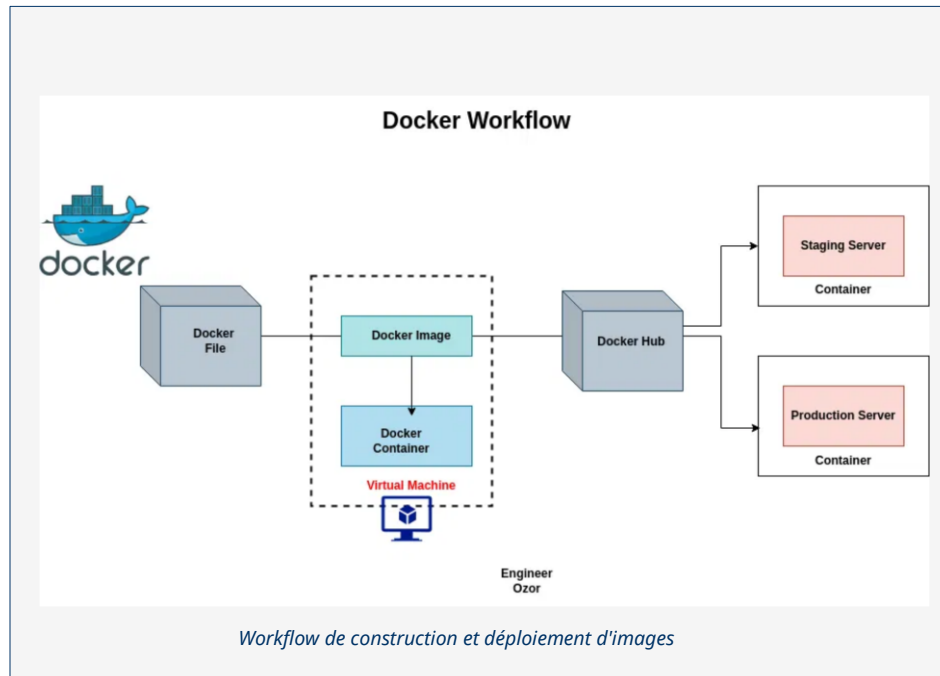
```
# Construction d'une image
docker build -t mon-app:latest .
```

Podman

Alternative sans démon, compatible avec Docker, privilégiée dans l'écosystème Red Hat.

```
# Construction d'une image
podman build -t mon-app:latest .
```

Caractéristique	Docker	Podman
Architecture	Basé sur un démon	Sans démon (daemonless)
Privilèges root	Nécessaires	Non nécessaires (rootless)
Intégration OpenShift	Bonne	Excellente

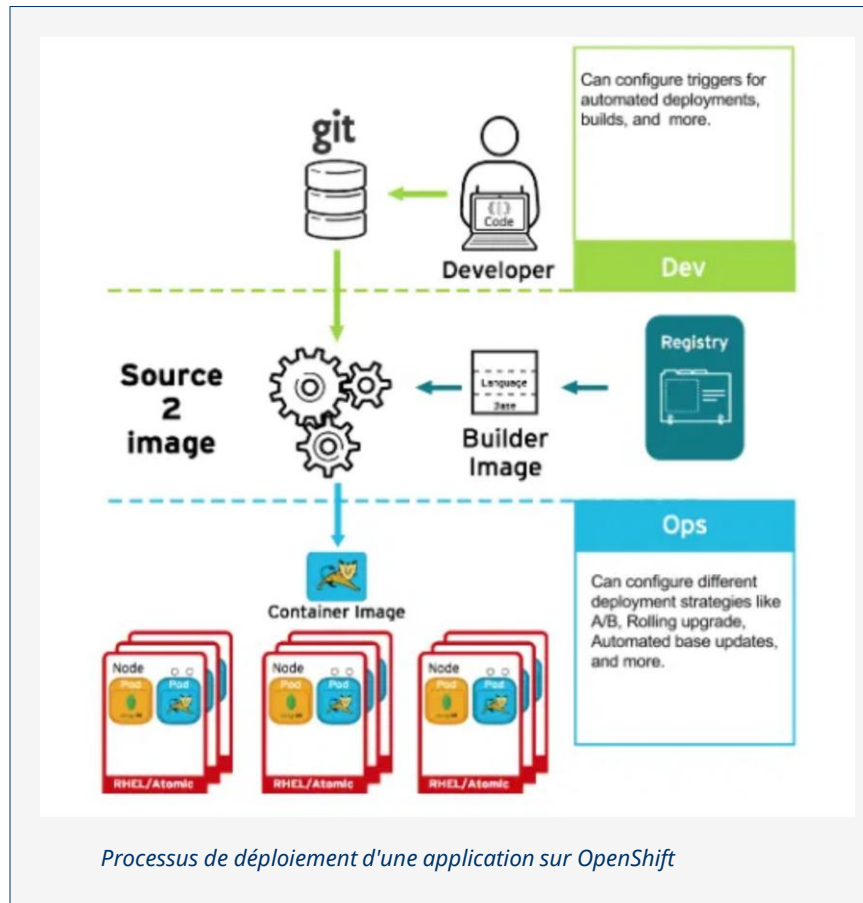


Introduction au déplacement d'applications vers OpenShift

Le déplacement d'applications existantes vers OpenShift permet de bénéficier de l'orchestration de conteneurs, de l'autoscaling et d'une gestion simplifiée du cycle de vie des applications.

Concepts clés

- Une **application conteneurisée** est composée d'un ou plusieurs conteneurs qui fonctionnent ensemble
- Le **déploiement sur OpenShift** peut se faire à partir de code source, d'images existantes ou de templates
- Les **services et routes** permettent d'exposer les applications à l'intérieur et à l'extérieur du cluster
- Les **stratégies de déploiement** définissent comment les mises à jour sont appliquées (Rolling, Recreate, etc.)
- La **persistance des données** est gérée via des persistants (PV) et des claims (PVC)



Méthodes de déploiement sur OpenShift

OpenShift offre plusieurs méthodes pour déployer vos applications, chacune adaptée à différents cas d'usage et niveaux d'expertise.

> Interface en ligne de commande (CLI) Recommandé

Utilisation de la commande **oc new-app** pour déployer depuis différentes sources (Git, images, templates).

🖥 Console Web OpenShift

Interface graphique intuitive pour créer et gérer des applications via un navigateur web.

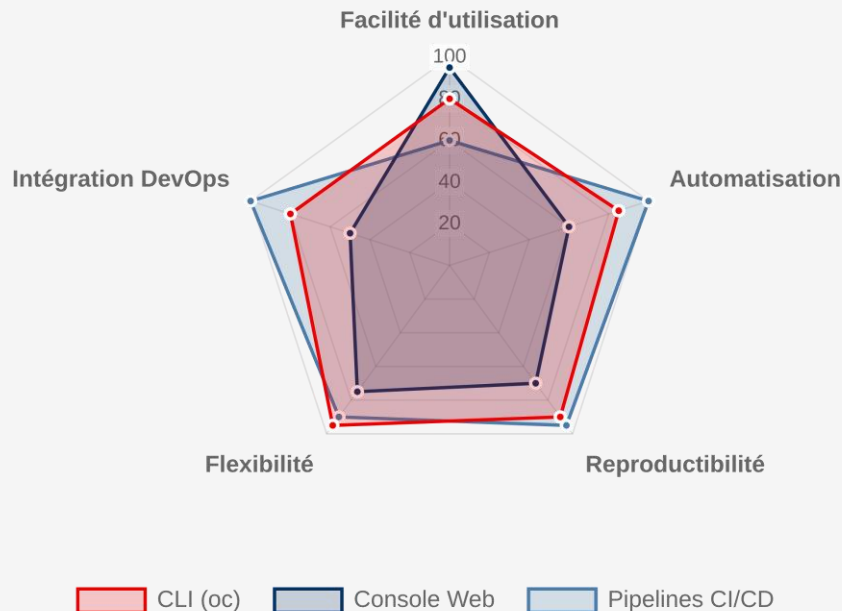
🔗 Pipelines CI/CD Avancé

Intégration avec Jenkins, Tekton ou GitLab CI pour automatiser le déploiement dans un flux DevOps.

📄 Fichiers YAML déclaratifs Avancé

Définition des ressources OpenShift (DeploymentConfig, Service, Route) dans des fichiers YAML pour un déploiement reproductible.

Comparaison des méthodes de déploiement



Utilisation de la commande oc new-app

La commande **oc new-app** est l'outil principal pour déployer des applications sur OpenShift. Elle peut créer automatiquement tous les objets nécessaires (BuildConfig, DeploymentConfig, Service, etc.) à partir de différentes sources.

🔗 Sources de déploiement

Dépôt Git : Déploie directement depuis un dépôt de code source

Image de conteneur : Déploie à partir d'une image existante dans un registre

Template : Utilise un modèle prédéfini pour déployer une application

Option	Description
--name	Nom à assigner aux objets créés
--namespace	Projet où déployer l'application
--context-dir	Sous-répertoire contenant l'application dans un dépôt Git
--strategy	Stratégie de build (docker, source, pipeline)
--env	Variables d'environnement à définir

Déploiement depuis un dépôt Git

```
# Détection automatique du langage et du framework
oc new-app https://github.com/sclorg/nodejs-ex
ex
# Spécifier une branche
oc new-app https://github.com/sclorg/nodejs-ex#dev
```

Déploiement depuis une image

```
# Depuis Docker Hub
oc new-app mysql:5.7
# Depuis un registre privé
oc new-app --docker-image=registry.example.com/myapp:latest
```

Déploiement avec options avancées

```
# Avec variables d'environnement
oc new-app mysql:5.7 --env=MYSQL_ROOT_PASSWORD=secret \
  --env=MYSQL_DATABASE=mydb
# Avec un nom personnalisé et des labels
oc new-app https://github.com/sclorg/nodejs-ex
  --name=frontend --labels=app=myapp,tier=frontend
```

Déploiement depuis des images existantes

Le déploiement d'applications à partir d'images de conteneurs existantes est une méthode rapide et fiable pour déployer des applications sur OpenShift.

1 Identifier l'image source

Sélectionner une image depuis un registre public (Docker Hub, Quay.io) ou privé (registre interne OpenShift).

2 Déployer l'image avec `oc new-app`

Utiliser la commande `oc new-app` en spécifiant l'image et les paramètres nécessaires.

3 Configurer l'application

Ajouter des variables d'environnement, des volumes persistants et d'autres configurations spécifiques.

4 Exposer l'application

Créer une route pour rendre l'application accessible depuis l'extérieur du cluster.

Bonnes pratiques

Toujours spécifier une version précise de l'image (tag) plutôt que **latest**
Utiliser des images de confiance provenant de sources vérifiées
Configurer des limites de ressources (CPU, mémoire) pour chaque déploiement

Exposer l'application

Créer une route pour exposer le service

`oc expose service mon-site-web`

Obtenir l'URL de la route

`oc get route mon-site-web`

Déploiement depuis Docker Hub

Déployer une image depuis Docker Hub

`oc new-app nginx:1.21 --name= mon-site-web`

Ajouter des variables d'environnement

`oc set env deployment/ mon-site-web \`

`NGINX_HOST= www.exemple.com`

`NGINX_PORT= 8080`

Déploiement depuis un registre privé

Créer un secret pour l'authentification

`oc create secret docker-registry mon-registre-secret \`

`--docker-server= registry.exemple.com`

`--docker-username= utilisateur`

`--docker-password= motdepasse`

Lier le secret au compte de service

`oc secrets link default mon-registre-secret --for=pull`

Déployer l'image privée

`oc new-app registry.exemple.com/mon-app:1.0`

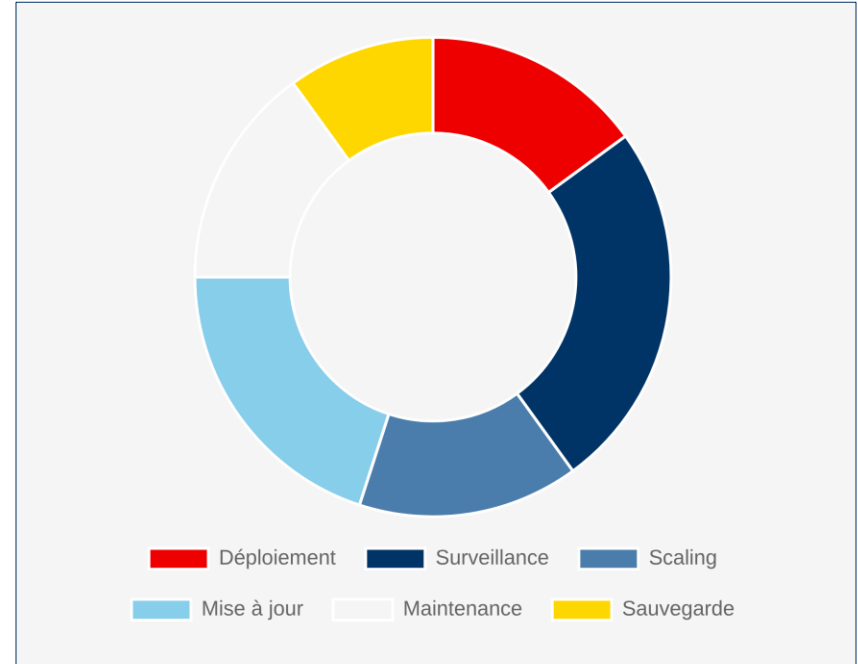
Introduction à la gestion d'applications

Une fois votre application déployée sur OpenShift, vous devez la gérer efficacement pour garantir sa disponibilité, ses performances et sa sécurité. La gestion d'applications est un aspect essentiel du cycle de vie des applications sur OpenShift.

Aspects clés de la gestion d'applications

-  **Surveillance et monitoring** : Suivi des métriques de performance, de santé et d'utilisation des ressources
-  **Scaling** : Ajustement automatique ou manuel du nombre de pods en fonction de la charge
-  **Configuration** : Gestion des variables d'environnement, ConfigMaps et Secrets
-  **Mises à jour et rollbacks** : Déploiement de Nouvelles versions et retour à des versions antérieures si nécessaire
-  **Sécurité** : Gestion des accès, des politiques de sécurité et des vulnérabilités
-  **Sauvegarde et restauration** : Protection des données et plans de reprise après sinistre

Cycle de vie de gestion d'une application



Surveillance et scaling d'applications

La surveillance et le scaling sont essentiels pour maintenir la disponibilité et les performances de vos applications sur OpenShift, en s'adaptant dynamiquement aux variations de charge.

📈 Surveillance des applications

Sondes de santé : Liveness et Readiness probes pour vérifier l'état des conteneurs

--- **Métriques** : Utilisation CPU, mémoire, réseau et stockage via Prometheus

Logs : Centralisation et analyse des journaux d'applications

🔧 Scaling d'applications

--- **Scaling horizontal** : Ajout ou suppression de pods (réplicas)

— **Scaling vertical** : Ajustement des ressources (CPU, mémoire) allouées aux pods

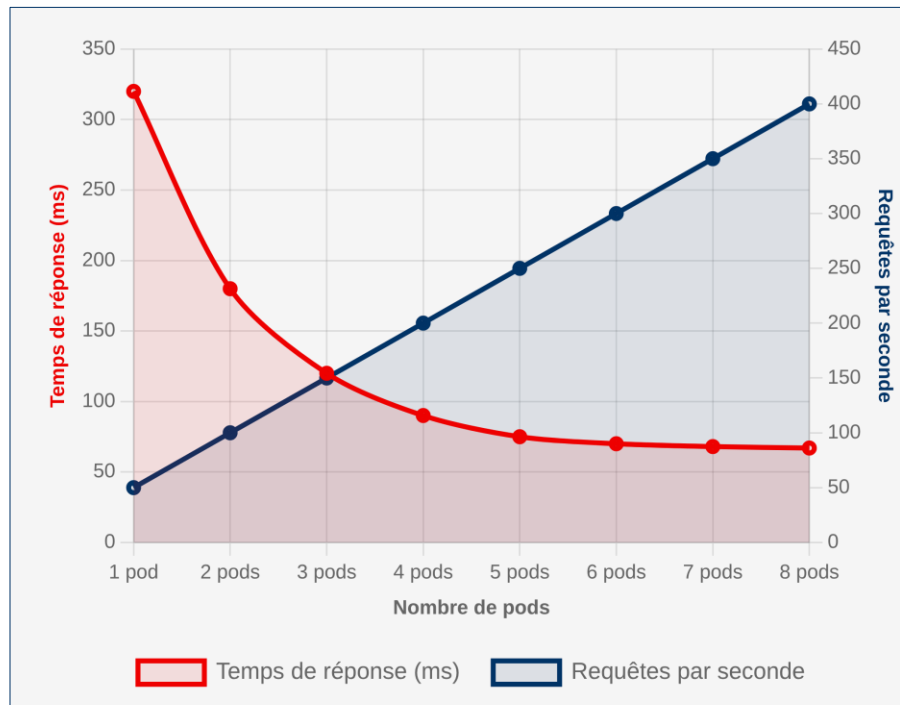
Autoscaling : Ajustement automatique basé sur l'utilisation des ressources ou des métriques personnalisées

Scaling manuel

```
oc scale deployment mon-app --replicas=5
```

Configuration d'un autoscaler

```
oc autoscale deployment mon-app --min=2 --max=10 --cpu-percent=80
```



Impact du scaling sur les performances

Gestion des configurations et secrets

La séparation du code et de la configuration est une bonne pratique essentielle pour les applications cloud-native. OpenShift fournit des mécanismes dédiés pour gérer les configurations et les données sensibles.



ConfigMaps

Stockent des données de configuration non-sensibles sous forme de paires clé-valeur, accessibles comme variables d'environnement, fichiers montés ou arguments de ligne de commande.



Secrets

Similaires aux ConfigMaps mais conçus pour stocker des données sensibles (mots de passe, tokens, etc.) avec des protections.

[en français et en anglais](#)

Caractéristique	ConfigMap	Secret
Type de données	Non-sensibles	Sensibles
Encodage	Text brut	Base64
Stockage	etcd (non-chiffré)	etcd (potentiellement chiffré)
Visibilité	Visible dans les logs	Masqué dans les logs

Gestion des configurations et secrets

La séparation du code et de la configuration est une bonne pratique essentielle pour les applications cloud-native. OpenShift fournit des mécanismes dédiés pour gérer les configurations et les données sensibles.

Création d'un ConfigMap

Depuis des valeurs littérales

```
oc create configmap app-config --from-literal=APP_ENV=production \
  --from-literal=APP_DEBUG=false
```

Depuis un fichier

```
oc create configmap app-config --from-file=config.json
```

Utilisation dans un déploiement

Monter un ConfigMap comme volume

```
oc set volume dc mon-app --add --type=configmap --configmap-name= app-config /
--mount-path=/etc/config
```

Utiliser un Secret comme variables d'environnement

```
oc set env dc mon-app --from=secret db-credentials
```

Création d'un Secret

Depuis des valeurs littérales

```
oc create secret generic db-credentials --from-literal= username \
  --from-literal=password=s3cr3t
```

Depuis des fichiers

```
oc create secret generic tls-certs --from-file=tls.crt \
  --from-file=tls.key
```

Stratégies de déploiement

OpenShift propose plusieurs stratégies de déploiement qui déterminent comment les nouvelles versions de votre application sont déployées et comment les versions précédentes sont remplacées.



Rolling

Par défaut

Déploie progressivement les nouveaux pods tout en supprimant les anciens. Garantit une disponibilité continue de l'application pendant la mise à jour.



Recreate

Simple

Supprime tous les pods existants avant de déployer les nouveaux. Provoque une interruption de service mais garantit un environnement propre pour la nouvelle version.



Blue/Green

Avancé

Maintient deux environnements identiques (bleu et vert) et bascule le trafic de l'un à l'autre. Permet un rollback instantané en cas de problème.

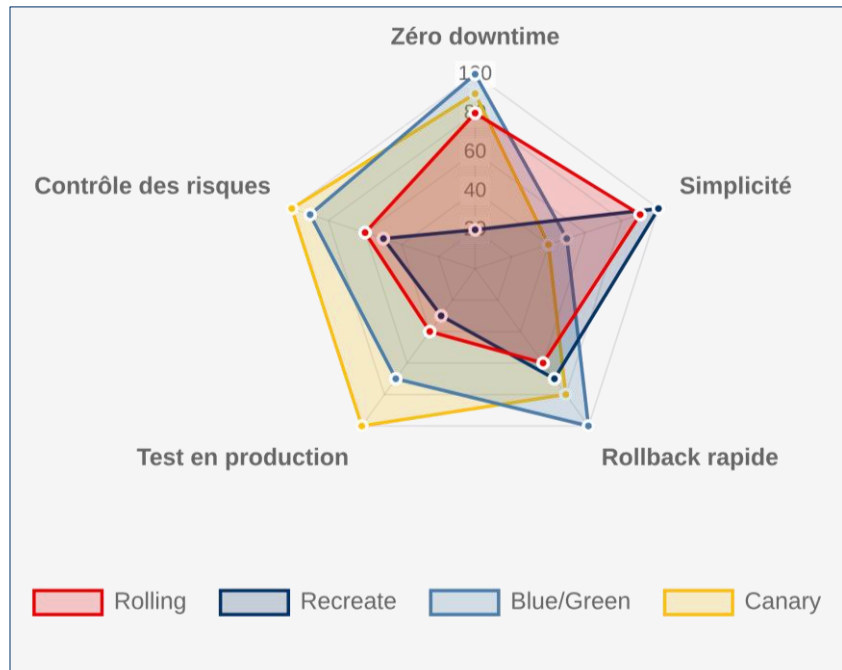


Canary

Avancé

Dirige un petit pourcentage du trafic vers la nouvelle version pour la tester en production avant un déploiement complet.

Comparaison des stratégies de déploiement



Résumé et prochaines étapes

Dans ce module, nous avons exploré les fondamentaux du déploiement et de la gestion d'applications sur OpenShift. Voici les points clés à retenir :

✓ Points clés

La **construction d'images** est la base du déploiement d'applications conteneurisées, avec des bonnes pratiques essentielles pour la sécurité et les performances

OpenShift offre **plusieurs méthodes de déploiement** adaptées à différents cas d'usage, avec la commande **oc new-app** comme outil principal

La **gestion efficace des applications** implique la surveillance, le scaling, la configuration et les stratégies de déploiement

Les **bonnes pratiques** de sécurité et d'optimisation sont essentielles pour des déploiements robustes et fiables

Prochains modules

- Module 2 : Développement d'applications pour OpenShift
- Module 3 : Configuration avancée et personnalisation
- Module 4 : Intégration CI/CD avec OpenShift

Compétences acquises dans ce module

