DÉPLOYER UNE APPLICATION DEPUIS LE DASHBOARD

Interface graphique et étapes clés

Étapes de déploiement

1 Accéder au dashboard Kubernetes

Connectez-vous au dashboard via l'URL fournie ou avec `kubectl proxy` puis accédez à

http://localhost:8001/api/v1/namespaces/kubernetes-dashboard/services/https:kubernetes-dashboard:/proxy/ .

2 Créer un nouveau déploiement

Sélectionnez "Déploiements" puis cliquez sur "Créer". Choisissez entre un formulaire ou un éditeur YAML.

3 Configurer l'application

Remplissez les champs requis : nom, image, réplicas, labels, variables d'environnement et ressources.

4 Déployer et surveiller

Cliquez sur "Déployer" et suivez l'état du déploiement. Vérifiez que les pods sont en état "Running".

Conseils pratiques

Utilisez des labels cohérents pour faciliter la gestion des ressources.

Définissez des limites de ressources pour éviter la surcharge du cluster.

Pour les environnements de production, privilégiez les fichiers YAML versionnés.

CRÉER UN DÉPLOIEMENT À PARTIR D'UN FICHIER YAML

Structure et commandes essentielles

Structure d'un fichier YAML de déploiement

```
apiVersion : apps/v1  kind : Deployment  metadata : name : nginx-deployment  labels : app : nginx  spec : replicas : 3 selector : matchLabels  : app : nginx  template : metadata : labels : app : nginx  spec : containers : - name : nginx  image : nginx:1.14.2  ports : - containerPort  : 80
```

Les sections clés d'un fichier YAML de déploiement Kubernetes :

- **apiVersion/kind** : Définit le type de ressource et sa version API
- **metadata** : Nom et labels pour identifier le déploiement
- **spec.replicas** : Nombre de pods à maintenir
- **spec.selector** : Comment identifier les pods gérés par ce déploiement
- **spec.template** : Modèle pour créer les nouveaux pods

Commandes pour appliquer un fichier YAML

```
$ kubectl apply -f deployment.yaml deployment.apps/nginx-deployment created
```

```
$ kubectl get deployments NAME READY UP-TO-DATE AVAILABLE AGE nginx-deployment 3/3 3 3 45s
```

```
$ kubectl describe deployment nginx-deployment Name: nginx-deployment Namespace: default CreationTimestamp: Wed, Sep 29, 2025 10:30:00 Labels: app=nginx Annotations: deployment.kubernetes.io/revision: 1 Selector: app=nginx Replicas: 3 desired | 3 updated | 3 total | 3 available StrategyType: RollingUpdate ...
```

Bonnes pratiques

- **Versionnez vos fichiers YAML** dans un système de contrôle de version comme Git
- **Utilisez des labels cohérents** pour organiser et sélectionner vos ressources
- **Spécifiez toujours une version d'image** précise plutôt que d'utiliser le tag"latest"
- **Définissez des limites de ressources** pour chaque conteneur (CPU etmémoire)
- **Utilisez kubectl diff**appliquer

EXPOSER UN SERVICE EN UTILISANT NODEPORT

Configuration et accès

Qu'est-ce qu'un Service NodePort ?

Un Service NodePort expose une application sur un port statique sur chaque nœud du cluster. Il permet d'accéder à l'application depuis l'extérieur du cluster en utilisant l'adresse IP de n'importe quel nœud et le port attribué.

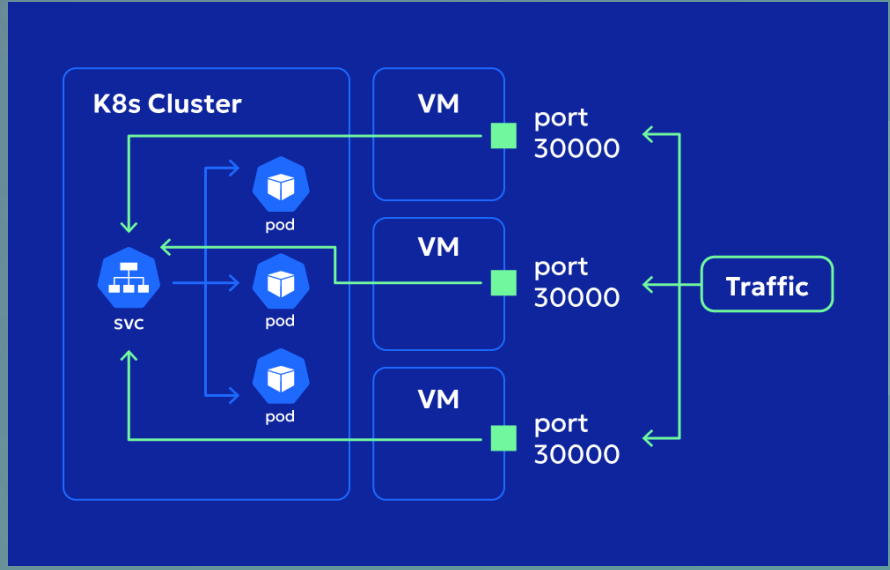
```
apiVersion : v1  kind : Service  metadata : name : my-nodeport-service  spec : selector : app : my-app  type : NodePort  ports : - port : 80  # Port du Service  targetPort : 8080  # Port du  Pod  nodePort : 30007  # Port exposé sur le nœud (optionnel)
```

Plage de ports

Les ports NodePort sont attribués dans la plage 30000-32767 par défaut. Si non spécifié, Kubernetes attribue automatiquement un port disponible dans cette plage.

Considérations de sécurité

L'exposition via NodePort ouvre un port sur tous les nœuds. Assurez-vous de configurer correctement les règles de pare-feu pour limiter l'accès si nécessaire.



Commandes utiles

```
# Créer un Service NodePort  kubectl expose deployment my-deployment --type=NodePort --port=80  kubectl get service my-nodeport-service  # Obtenir l'URL d'accès (avec Minikube)  minikube service my-nodeport-service --url
# Accéder au Service depuis l'extérieur  curl http://<NODE-IP>:  <NODE-PORT>

# Vérifier les détails du Service
```

ACCÉDER À UNE APPLICATION DEPUIS LE MONDE EXTÉRIEUR

Méthodes pour exposer vos services Kubernetes

Méthodes d'exposition

NodePort

Expose le service sur un port statique sur chaque nœud du cluster. Accessible via `<NodeIP>:<NodePort>`. Limité à la plage de ports 30000-32767 par défaut.

LoadBalancer

Crée un équilibreur de charge externe dans l'infrastructure cloud. Fournit une adresse IP externe dédiée qui route le trafic vers le service.

Ingress

Ressource API qui gère l'accès externe aux services, généralement HTTP. Permet le routage basé sur l'URL, la terminaison TLS et l'hébergement virtuel.

Port-forwarding

Solution temporaire pour le développement et le débogage. Transfère les connexions d'un port local à un port sur un pod.

```
kubectl port-forward pod/<pod-name> <local-port>:<pod-port>
```

Comparaison des méthodes

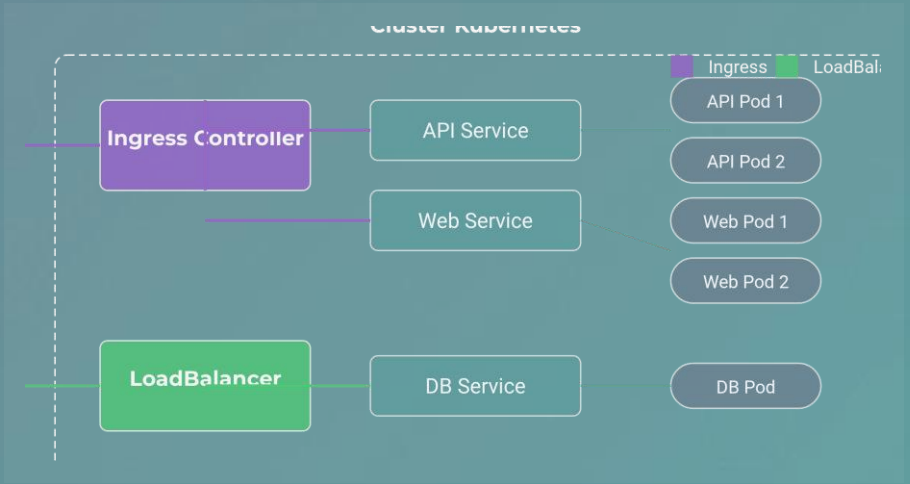
Méthode	Cas d'utilisation	Complexité
NodePort	Environnements de test, démos	Faible
LoadBalancer	Applications de production dans le cloud	Moyenne
Ingress	Applications web avec plusieurs services	Élevée
Port-forwarding	Développement et débogage	Très faible

INGRESS ET LOADBALANCER

Comparaison entre les ressources pour l'exposition des services

Comparaison des approches

Caractéristique	LoadBalancer	Ingress
Niveau	Service	Application (L7)
Routage	IP + Port	Nom d'hôte, chemin
SSL/TLS	Non	Oui
Coût	1 par service	1 pour plusieurs services
Configuration	Simple	Plus complexe
Cas d'usage	Service unique	Multiples services



INGRESS ET LOADBALANCER

Comparaison entre les ressources pour l'exposition des services

Exemple de LoadBalancer

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app: my-app
  ports:
  - port: 80
    targetPort: 8080
  type: LoadBalancer
```

Exemple d'Ingress

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: my-ingress
spec:
  rules:
  - host: app.example.com
    http:
      paths:
      - path: /api
        pathType: Prefix
        backend:
          service:
            name: api-service
            port:
              number: 80
      - path: /
        pathType: Prefix
        backend:
          service:
            name: web-service
            port:
              number: 80
```

STRATÉGIE DE MISE À JOUR, ROLLING UPDATE

Mécanismes de mise à jour progressive des applications

Types de stratégies de déploiement

Rolling Update (par défaut)

Remplace progressivement les anciens pods par de nouveaux. Garantit la disponibilité continue de l'application pendant la mise à jour. Kubernetes gère automatiquement le trafic vers les pods disponibles.

Recreate

Supprime tous les pods existants avant de créer les nouveaux. Provoque une interruption de service mais garantit que seule la nouvelle version est exécutée. Utile pour les applications qui ne supportent pas plusieurs versions simultanées.

Blue/Green (avec services)

Déploie la nouvelle version (green) à côté de l'ancienne (blue). Bascule le trafic d'un coup vers la nouvelle version en modifiant le sélecteur du Service. Permet un rollback rapide en cas de problème.

Canary (avec Ingress ou Service Mesh)

Dirige un petit pourcentage du trafic vers la nouvelle version pour la tester en production. Augmente progressivement le trafic si tout fonctionne correctement. Nécessite généralement des outils supplémentaires comme Istio.

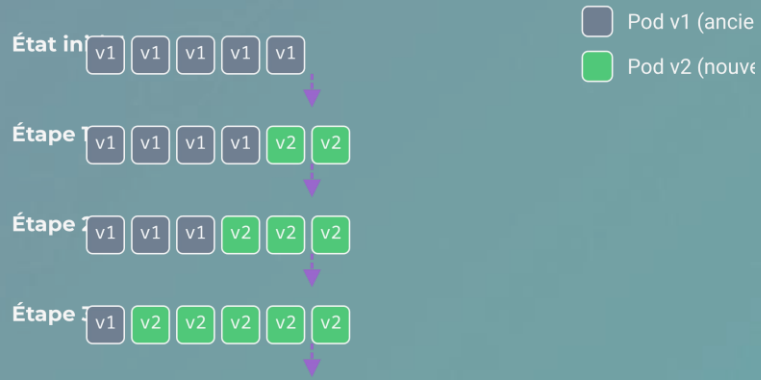
STRATÉGIE DE MISE À JOUR, ROLLING UPDATE

Mécanismes de mise à jour progressive des applications

Configuration du Rolling Update

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 5
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 1
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.16.1
```

Processus de Rolling Update



Bonnes pratiques

- **Configurez des readiness probes** pour s'assurer que les nouveaux pods sont prêts avant de recevoir du trafic
- **Ajustez maxSurge et maxUnavailable** selon les besoins de disponibilité et les ressources du cluster
- **Utilisez des versions d'image spécifiques** plutôt que le tag "latest" pour contrôler précisément les mises à jour