Introduction

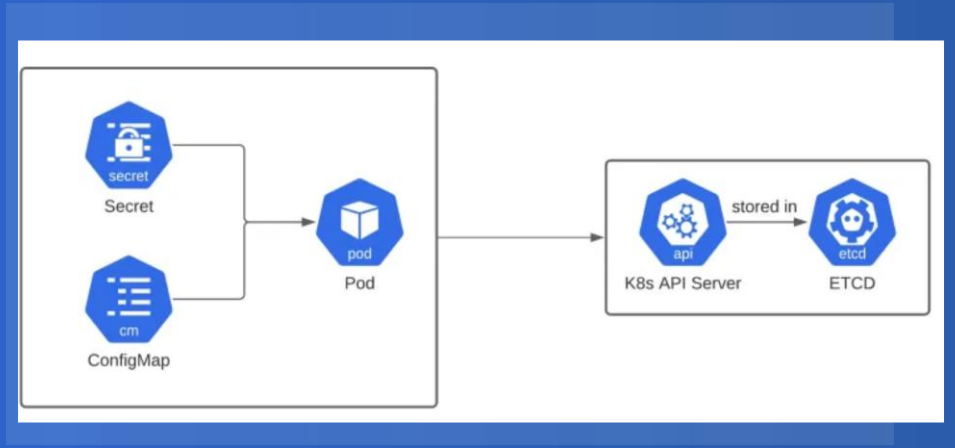
Dans un environnement Kubernetes, la**gestion de la configuration** et des **données sensibles** est essentielle pour maintenir des applications flexibles, sécurisées et faciles à deployer.

Les applications modernes nécessitent une séparation claire entre :

- Le code de l'application
- La configuration spécifique à l'environnement
- Les données sensibles (mots de passe, jetons, clés)

Kubernetes propose deux ressources principales pour répondre à ces besoins :

- **ConfigMaps** : pour les données de configuration non sensibles
- **Secrets** : pour les données confidentielles



ConfigMaps : Définition et Objectif

Une **ConfigMap** est un objet API Kubernetes utilisé pour stocker des données **non confidentielles** sous forme de paires clé-valeur.

Les ConfigMaps permettent de :

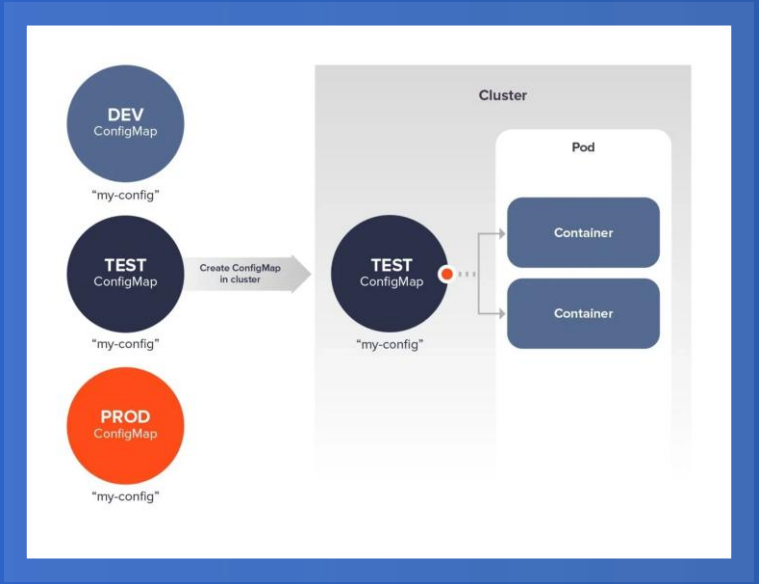
- Découpler la configuration de l'image du conteneur
- Rendre les applications portables entre environnements

Les données peuvent être consommées comme :

- Variables d'environnement
- Arguments de ligne de commande
- Fichiers dans un volume monté

Points importants

- Limite de taille : 1 MiB maximum
- Ne fournit pas de chiffrement ou de secret

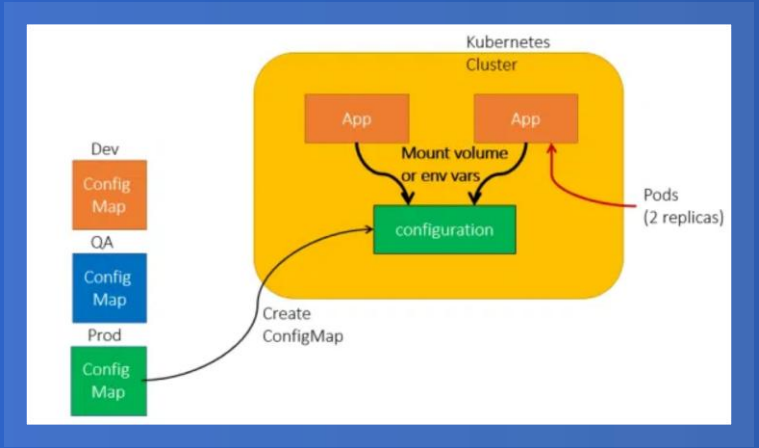


Création et Utilisation des ConfigMaps

Création d'une ConfigMap

Plusieurs méthodes sont disponibles :

```
# Via un fichier YAML
kubectl apply -f configmap.yaml

# À partir de valeurs littérales
kubectl create configmap app-config \
  --from-literal=APP_ENV=production \
  --from-literal=APP_DEBUG=false

# À partir de fichiers
kubectl create configmap app-config \
  --from-file=config.properties
```



Utilisation dans un Pod

Les ConfigMaps peuvent être utilisées de trois façons principales :

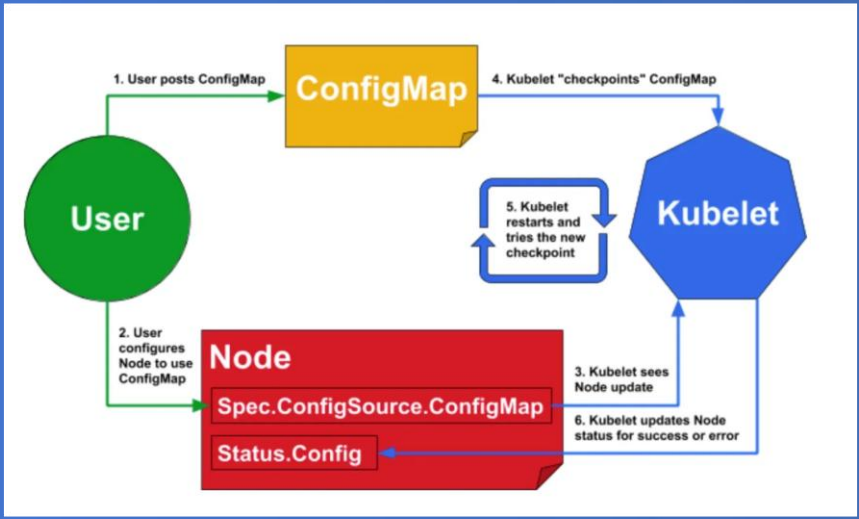
- Variables d'environnement (**env**)
- Arguments de commande (**command**)
- Fichiers dans un volume (**volumeMounts**)

ConfigMaps : Exemples Pratiques

Exemple : Configuration d'application

Utilisation d'une ConfigMap pour configurer une application web :

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: webapp-config
data:
  # Paramètres simples
  APP_ENV: "production"
  LOG_LEVEL: "info"

  # Fichier de configuration
  config.json: |
    {
      "database": "mysql",
      "host": "db-service"
    }
```



Configuration multi-environnement

ConfigMaps distinctes pour différents environnements

Secrets : Définition et Objectif

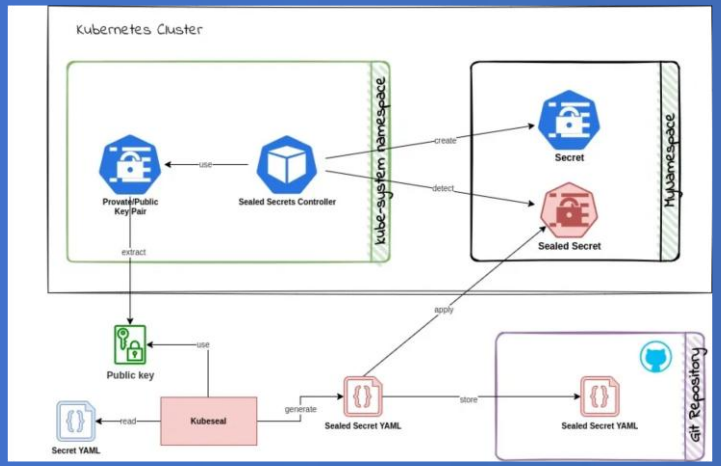
Un **Secret** est un objet qui contient une petite quantité de données sensibles telles qu'un mot de passe, un jeton ou une clé.

Les Secrets permettent de :

- Stocker des informations confidentielles séparément du code
- Réduire le risque d'exposition des données sensibles
- Centraliser la gestion des informations d'identification

Similaires aux ConfigMaps, mais spécifiquement conçus pour les données confidentielles.

⚠ **Attention** : Par défaut, les Secrets sont stockés non chiffrés dans etcd (la base de données de Kubernetes).



Création et Utilisation des Secrets

Création d'un Secret

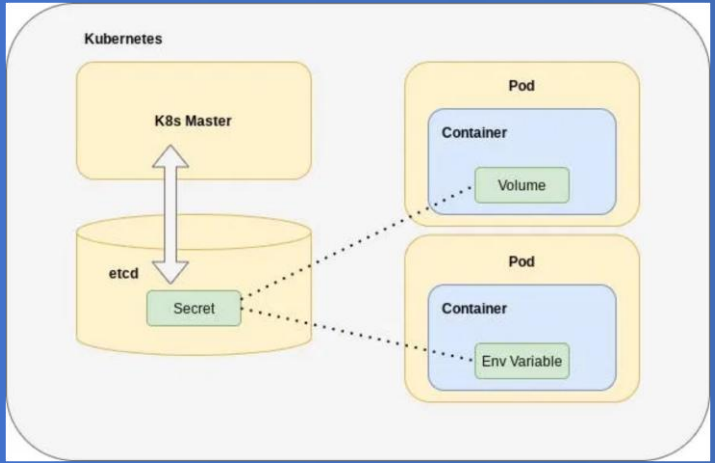
```
# Via un fichier YAML
kubectl apply -f secret.yaml

# À partir de valeurs littérales
kubectl create secret generic db-credentials \
  --from-literal=username=admin \
  --from-literal=password=s3cr3t
```

⚠ Les valeurs sont encodées en Base64, mais **pas chiffrées** !

Utilisation dans un Pod

Les Secrets peuvent être utilisés comme :

- Variables d'environnement
- Fichiers dans un volume
- Identifiants pour les registres d'images (**imagePullSecrets**)



Sécurité des Secrets

⚠ **Par défaut, les Secrets Kubernetes sont stockés non chiffrés dans etcd !**

Mesures de sécurité recommandées

- **Chiffrement au repos** : Activer le chiffrement des données dans etcd
- **RBAC** : Configurer des règles d'accès avec privilèges minimaux
- **Restriction d'accès** : Limiter l'accès aux Secrets à des conteneurs spécifiques
- **Solutions externes** : Utiliser des gestionnaires de secrets externes (HashiCorp Vault, AWS Secrets Manager)

Bonnes pratiques

- Rotation régulière des Secrets
- Audit des accès aux Secrets
- Ne jamais exposer les Secrets dans les logs

ConfigMaps vs Secrets

Caractéristique	ConfigMaps	Secrets
Objectif	Données de configuration non sensibles	Données sensibles (mots de passe, tokens, clés)
Stockage	Non chiffré dans etcd	Non chiffré par défaut (encodage Base64)
Limite de taille	1 MiB	1 MiB
Utilisation	Variables d'environnement, fichiers, arguments	Variables d'environnement, fichiers, imagePullSecrets
Visibilité	Visible en clair dans les commandes kubectl	Masqué par défaut dans les commandes kubectl

Quand utiliser ConfigMaps

- Fichiers de configuration d'application
- Variables d'environnement non sensibles
- Configuration spécifique à l'environnement
- Paramètres de journalisation

Quand utiliser Secrets

- Identifiants de base de données
- Clés API et tokens OAuth
- Certificats TLS et clés privées
- Identifiants pour les registres d'images

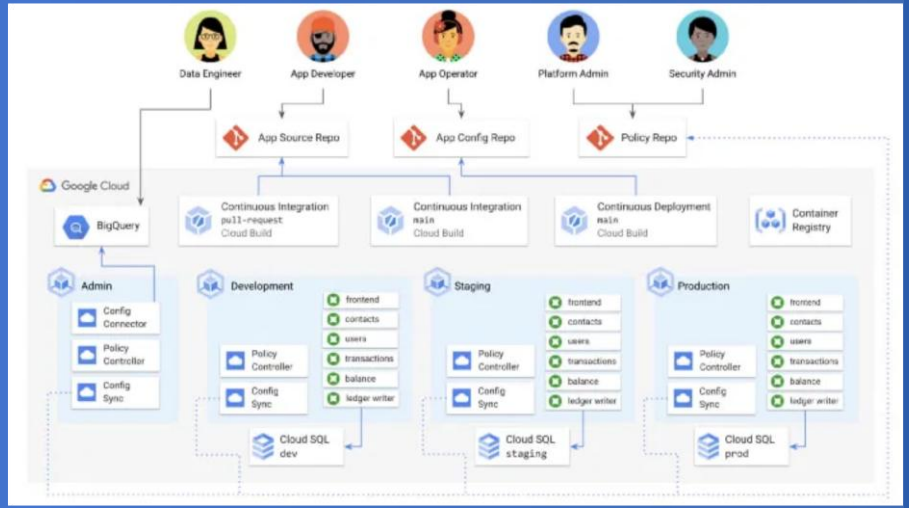
Bonnes Pratiques

Pour les ConfigMaps

- Utiliser des ConfigMaps **immuables** pour éviter les modifications accidentelles
- Organiser les ConfigMaps par **composant** ou **fonction**
- Utiliser des **préfixes** cohérents pour les noms de clés

Pour les Secrets

- Mettre en place une**rotation régulière** des Secrets
- Utiliser des**solutions externes** pour lesenvironnements de production
- Limiter l'accès aux Secrets avec des**règles RBAC** strictes



Conclusion et Questions

Points clés à retenir

- Les **ConfigMaps** sont pour les données de configuration non sensibles
- Les **Secrets** sont pour les onnées confidentielles
- Les deux permettent de découpler la configuration du code
- Les Secrets ne sont pas chiffrés par défaut
- Utilisez des solutions externes pour une sécurité renforcée
- Appliquez les bonnes pratiques de sécurité

Questions ?

N'hésitez pas à poser vos questions