PARTIE 3 : GitOps avec ArgoCD

Durée : 45 minutes

Objectifs

Dans cette troisième partie, vous allez implémenter le GitOps avec ArgoCD pour automatiser le déploiement continu de l'application CloudShop depuis Git.

Compétences évaluées : - Installation et configuration d'ArgoCD - Création d'Applications ArgoCD - Configuration des sync policies (auto-sync, prune, self-heal) - Gestion du cycle de vie (rollback, sync manual) - Intégration Git comme source unique de vérité

Travail à Réaliser

Tâche 3.1 : Installation d'ArgoCD (4 points)

Méthode 1 : Installation via Manifests (Recommandée)

```
# 1. Créer le namespace argocd
kubectl create namespace argocd

# 2. Installer ArgoCD
kubectl apply -n argocd -f
https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml

# 3. Attendre que tous les pods soient prêts
kubectl wait --for=condition=ready pod --all -n argocd --timeout=300s

# 4. Vérifier l'installation
kubectl get pods -n argocd
kubectl get svc -n argocd
```

Méthode 2 : Installation via Helm (Alternative)

```
# 1. Ajouter le repo Helm
helm repo add argo https://argoproj.github.io/argo-helm
helm repo update

# 2. Installer ArgoCD
helm install argocd argo/argo-cd \
```

```
  --namespace argocd \
  --create-namespace \
  --set server.service.type=LoadBalancer

# 3. Vérifier
kubectl get pods -n argocd
```

Accès à l'interface ArgoCD

```
# Méthode 1 : Port-forward (dev local)
kubectl port-forward svc/argocd-server -n argocd 8080:443

# Méthode 2 : NodePort (Kind/Minikube)
kubectl patch svc argocd-server -n argocd -p '{"spec": {"type": "NodePort"}}'

# Récupérer le mot de passe admin initial
kubectl -n argocd get secret argocd-initial-admin-secret -o jsonpath="{.data.password}" |
base64 -d && echo

# Accéder à l'UI
# URL : https://localhost:8080
# User : admin
# Password : (mot de passe récupéré ci-dessus)
```

Installation du CLI ArgoCD (Optionnel mais recommandé)

```
# macOS
brew install argocd

# Linux
curl -sSL -o argocd-linux-amd64
https://github.com/argoproj/argo-cd/releases/latest/download/argocd-linux-amd64
sudo install -m 555 argocd-linux-amd64 /usr/local/bin/argocd
rm argocd-linux-amd64

# Login via CLI
argocd login localhost:8080 --insecure --username admin --password <password>
```

Tâche 3.2 : Préparation du Repository Git (2 points)

ArgoCD nécessite un repository Git contenant les manifests Kubernetes.

Option A : Repository Public (Plus Simple)

```
# 1. Créer un repo GitHub public : cloudshop-k8s
# 2. Pusher les manifests k8s/

git init
git add k8s/
git commit -m "Initial Kubernetes manifests"
git remote add origin https://github.com/<votre-user>/cloudshop-k8s.git
git push -u origin main
```

Option B : Repository Privé

```
# 1. Créer un Personal Access Token sur GitHub
# Settings > Developer settings > Personal access tokens > Generate new token
# Permissions : repo (full control)

# 2. Ajouter le repo dans ArgoCD
argocd repo add https://github.com/<user>/cloudshop-k8s.git \
  --username <github-username> \
  --password <github-token>

# Ou via l'UI : Settings > Repositories > Connect Repo
```

Structure du Repository

```
cloudshop-k8s/
 base/                # Manifests de base
   namespaces/
   configs/
   deployments/
   services/
   statefulsets/
   ingress/
 overlays/            # Overlays Kustomize (optionnel)
   dev/
   staging/
   prod/
 README.md
```

Tâche 3.3 : Création des Applications ArgoCD (6 points)

Créez les Applications ArgoCD pour déployer les différents composants.

Application 1 : Infrastructure (Namespace, ConfigMaps, Secrets)

Fichier : argocd/apps/infrastructure.yaml

```yaml
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: cloudshop-infrastructure
  namespace: argocd
  finalizers:
  - resources-finalizer.argocd.argoproj.io
spec:
  project: default

  source:
    repoURL: https://github.com/<votre-user>/cloudshop-k8s.git
    targetRevision: HEAD
    path: base
    directory:
      include: '{namespaces,configs}/**/*.yaml'

  destination:
    server: https://kubernetes.default.svc
    namespace: cloudshop-prod

  syncPolicy:
    automated:
      prune: true
      selfHeal: true
      allowEmpty: false
    syncOptions:
    - CreateNamespace=true
    retry:
      limit: 5
      backoff:
        duration: 5s
        factor: 2
        maxDuration: 3m
```

Application 2 : Database (StatefulSet PostgreSQL)

Fichier : argocd/apps/database.yaml

```yaml
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: cloudshop-database
  namespace: argocd
  finalizers:
  - resources-finalizer.argocd.argoproj.io
spec:
  project: default

  source:
    repoURL: https://github.com/<votre-user>/cloudshop-k8s.git
    targetRevision: HEAD
    path: base/statefulsets

  destination:
    server: https://kubernetes.default.svc
    namespace: cloudshop-prod

  syncPolicy:
    automated:
      prune: true
      selfHeal: true
    syncOptions:
    - CreateNamespace=false
    retry:
      limit: 5
      backoff:
        duration: 5s
        factor: 2
        maxDuration: 3m

  # Health check personnalisé pour StatefulSet
  ignoreDifferences:
  - group: apps
    kind: StatefulSet
    jsonPointers:
    - /spec/volumeClaimTemplates
```

Application 3 : Backend Services

Fichier : argocd/apps/backend.yaml

```yaml
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: cloudshop-backend
  namespace: argocd
  finalizers:
  - resources-finalizer.argocd.argoproj.io
spec:
  project: default

  source:
    repoURL: https://github.com/<votre-user>/cloudshop-k8s.git
    targetRevision: HEAD
    path: base
    directory:
      include: '{deployments,services}/**/*.yaml'
      exclude: 'deployments/frontend.yaml'

  destination:
    server: https://kubernetes.default.svc
    namespace: cloudshop-prod

  syncPolicy:
    automated:
      prune: true
      selfHeal: true
    syncOptions:
    - CreateNamespace=false
    retry:
      limit: 5
      backoff:
        duration: 5s
        factor: 2
        maxDuration: 3m
```

Application 4 : Frontend + Ingress

Fichier : argocd/apps/frontend.yaml

```yaml
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: cloudshop-frontend
  namespace: argocd
```

```
  finalizers:
  - resources-finalizer.argocd.argoproj.io
spec:
  project: default

  source:
    repoURL: https://github.com/<votre-user>/cloudshop-k8s.git
    targetRevision: HEAD
    path: base
    directory:
      include: '{deployments/frontend.yaml,ingress}/**/*.yaml'

  destination:
    server: https://kubernetes.default.svc
    namespace: cloudshop-prod

  syncPolicy:
    automated:
      prune: true
      selfHeal: true
    syncOptions:
    - CreateNamespace=false
    retry:
      limit: 5
      backoff:
        duration: 5s
        factor: 2
        maxDuration: 3m
```

Application App-of-Apps (Optionnel - Avancé)

Fichier : argocd/apps/app-of-apps.yaml

```
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: cloudshop-platform
  namespace: argocd
  finalizers:
  - resources-finalizer.argocd.argoproj.io
spec:
  project: default

  source:
```

```
  repoURL: https://github.com/<votre-user>/cloudshop-k8s.git
  targetRevision: HEAD
  path: argocd/apps

 destination:
  server: https://kubernetes.default.svc
  namespace: argocd

 syncPolicy:
  automated:
   prune: true
   selfHeal: true
```

Tâche 3.4 : Déploiement et Synchronisation (3 points)

Déployer les Applications

```
# Méthode 1 : Via kubectl
kubectl apply -f argocd/apps/infrastructure.yaml
kubectl apply -f argocd/apps/database.yaml
kubectl apply -f argocd/apps/backend.yaml
kubectl apply -f argocd/apps/frontend.yaml

# Méthode 2 : Via ArgoCD CLI
argocd app create cloudshop-infrastructure \
  --repo https://github.com/<user>/cloudshop-k8s.git \
  --path base \
  --dest-server https://kubernetes.default.svc \
  --dest-namespace cloudshop-prod \
  --sync-policy automated \
  --auto-prune \
  --self-heal

# Vérifier les applications
argocd app list
kubectl get applications -n argocd
```

Synchroniser manuellement (si nécessaire)

```
# Via CLI
argocd app sync cloudshop-infrastructure
argocd app sync cloudshop-database
argocd app sync cloudshop-backend
```

```
argocd app sync cloudshop-frontend

# Voir le status
argocd app get cloudshop-backend

# Via UI
# Cliquer sur "Sync" pour chaque application
```

Tâche 3.5 : Test de Rollback et Self-Heal (2 points)

Test 1 : Rollback après un changement

```
# 1. Modifier un Deployment (ex: changer l'image)
git clone https://github.com/<user>/cloudshop-k8s.git
cd cloudshop-k8s
# Editer base/deployments/frontend.yaml : changer le tag de l'image
sed -i 's/frontend:latest/frontend:v2.0.0/' base/deployments/frontend.yaml
git add base/deployments/frontend.yaml
git commit -m "Update frontend to v2.0.0"
git push

# 2. ArgoCD détecte le changement et sync automatiquement
# Attendre quelques secondes
argocd app get cloudshop-frontend

# 3. Rollback vers la version précédente
argocd app rollback cloudshop-frontend

# Ou via l'UI : Application > History > Rollback to previous version
```

Test 2 : Self-Heal

```
# 1. Supprimer manuellement un Deployment
kubectl delete deployment frontend -n cloudshop-prod

# 2. ArgoCD détecte la différence et recrée le Deployment automatiquement
# (attendre 30-60 secondes)

# 3. Vérifier que le Deployment est recréé
kubectl get deployment frontend -n cloudshop-prod
argocd app get cloudshop-frontend
```

Test 3 : Sync Failure et Retry

```
# 1. Introduire une erreur volontaire dans un manifest
# Ex: image inexistante, secret manquant, etc.

# 2. Push le changement
git add .
git commit -m "Test sync failure"
git push

# 3. Observer le comportement d'ArgoCD
argocd app get cloudshop-backend
# Status devrait être "OutOfSync" ou "SyncFailed"

# 4. Corriger l'erreur et push
git revert HEAD
git push

# 5. ArgoCD retry automatiquement
```

Validation

Checklist de Validation

```
# 1. ArgoCD installé et accessible
kubectl get pods -n argocd
# Tous les pods doivent être Running

# 2. Accès à l'UI ArgoCD
# Ouvrir https://localhost:8080 dans le navigateur

# 3. Applications créées
kubectl get applications -n argocd
# Doit lister : cloudshop-infrastructure, cloudshop-database, cloudshop-backend,
cloudshop-frontend

# 4. Applications synchronisées
argocd app list
# Status : Synced, Healthy

# 5. Ressources déployées
kubectl get all -n cloudshop-prod
```

```
# Tous les pods Running, Services créés, Ingress configuré

# 6. Test auto-sync
# Modifier un fichier dans Git, push, vérifier que ArgoCD sync automatiquement

# 7. Test self-heal
kubectl delete pod -l app=frontend -n cloudshop-prod
# Pod doit être recréé automatiquement

# 8. Health checks
argocd app get cloudshop-backend
# Health Status : Healthy pour toutes les ressources
```

Commandes de Debugging

```
# Voir les logs d'une application
argocd app logs cloudshop-backend

# Voir l'historique des syncs
argocd app history cloudshop-backend

# Comparer l'état Git vs Cluster
argocd app diff cloudshop-backend

# Forcer une resync complète
argocd app sync cloudshop-backend --force

# Voir les events ArgoCD
kubectl get events -n argocd --sort-by='.lastTimestamp'

# Logs du serveur ArgoCD
kubectl logs -n argocd -l app.kubernetes.io/name=argocd-server --tail=100
```

Concepts Clés

GitOps Principles

1. **Déclaratif** : L'état désiré est décrit en Git (YAML)
2. **Versionné** : Tout changement est tracé dans Git
3. **Pull-based** : ArgoCD tire depuis Git (pas de push)
4. **Convergence** : ArgoCD réconcilie en continu l'état réel vs désiré

Sync Policies

- **Auto-sync** : Synchronisation automatique lors d'un changement Git
- **Prune** : Suppression automatique des ressources retirées de Git
- **Self-heal** : Réconciliation automatique si une ressource est modifiée manuellement

Health Checks

ArgoCD évalue la santé des ressources : - **Deployment** : Replicas ready - **StatefulSet** : Pods ready - **Service** : Endpoints disponibles - **Ingress** : Backend services healthy

---

Erreur : Application OutOfSync

```
# Voir les différences
argocd app diff <app-name>

# Forcer la synchronisation
argocd app sync <app-name> --force

# Vérifier les logs
argocd app logs <app-name>
```

Erreur : Repository inaccessible

```
# Vérifier les credentials
argocd repo list

# Re-ajouter le repo
argocd repo add https://github.com/<user>/repo.git \
  --username <user> \
  --password <token>
```

Erreur : Application Degraded

```
# Voir les ressources en erreur
argocd app get <app-name>

# Voir les events Kubernetes
kubectl get events -n cloudshop-prod --sort-by='.lastTimestamp'
```

```
# Voir les logs des pods
kubectl logs -l app=<app-name> -n cloudshop-prod
```

Auto-sync ne fonctionne pas

```
# Vérifier la config de l'application
kubectl get application <app-name> -n argocd -o yaml | grep -A 5 syncPolicy

# Vérifier que ArgoCD détecte les changements
argocd app get <app-name> --refresh

# Vérifier les logs du repo-server
kubectl logs -n argocd -l app.kubernetes.io/name=argocd-repo-server
```

Ressources

- ArgoCD Documentation
- GitOps Principles
- ArgoCD Best Practices

Une fois cette partie terminée, passez à PARTIE4_OBSERVABILITE.md