Lab 2 - Gestion du Stockage et Surveillance PVC PostgreSQL

Objectif

Mettre en place une infrastructure de stockage robuste pour PostgreSQL avec surveillance proactive de la saturation du PVC, en réponse à l'alerte critique du document de passation.

Contexte Critique

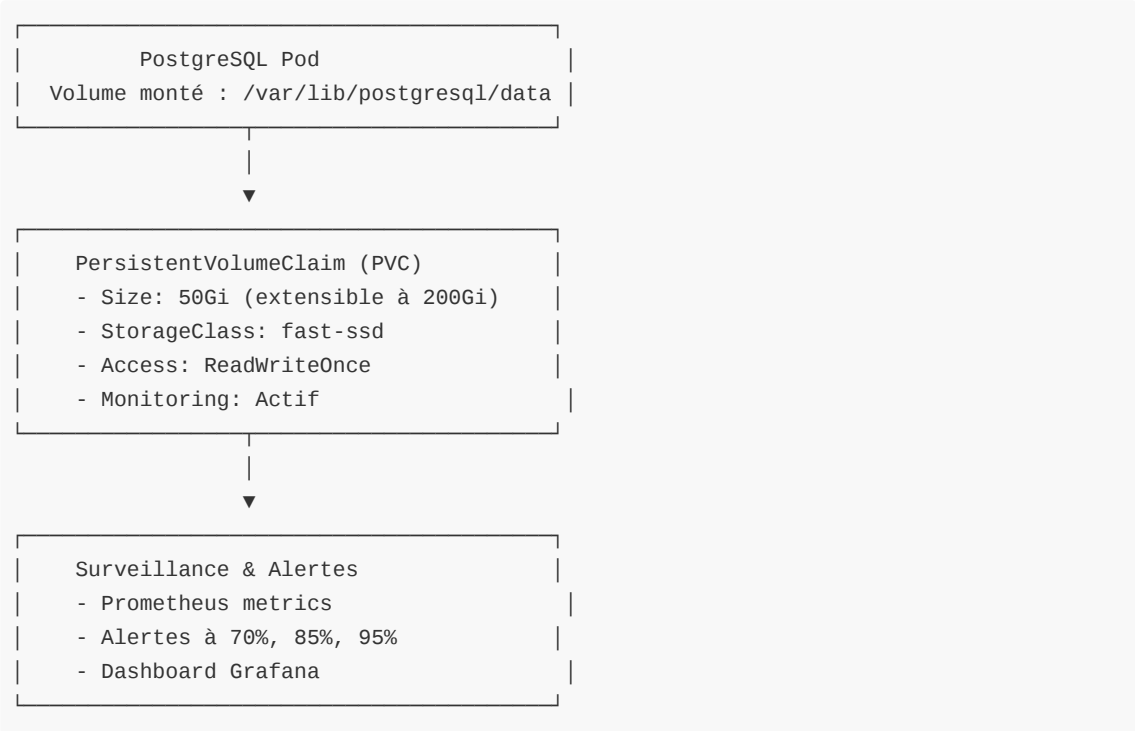
> ***ALERTE DU DOCUMENT** : "Il est crucial de surveiller la saturation du PVC de la base de données PostgreSQL de Random en risque d'avoir une interruption des services"*

Vous devez créer une infrastructure qui prévient automatiquement les problèmes de saturation du stockage.

Prérequis

- Cluster Kubernetes fonctionnel
- Namespace random-db créé (Lab 1)
- kubectl et helm installés
- Accès à un système de stockage (local-path, NFS, ou cloud provider)
- Prometheus Operator installé (pour la surveillance)

Architecture de Stockage Cible



Exercice 1 : Création du StorageClass (20 min)

Créer un StorageClass optimisé pour les bases de données avec les caractéristiques suivantes :

- Nom : `fast-ssd-expandable`
- Permettre l'expansion automatique des volumes
- Politique de récupération : Retain (ne pas supprimer les données)
- Mode de binding : WaitForFirstConsumer (pour optimiser le placement)
- Type de disque : SSD/haute performance

Adapter selon votre environnement :

- Cluster local (k3s/minikube) : utiliser le provisioner local
- AWS : utiliser EBS avec type gp3
- GCP : utiliser pd-ssd
- Azure : utiliser managed-premium

Livrables :

- Fichier `storageclass.yaml`
- Vérification avec `kubectl get storageclass`
- Documentation justifiant les choix techniques

Exercice 2 : Déploiement PostgreSQL avec StatefulSet (30 min)

Déployer PostgreSQL 15 dans le namespace random-db avec :

Configuration PVC :

- Taille initiale : 50Gi
- StorageClass : fast-ssd-expandable
- AccessMode : ReadWriteOnce

Configuration PostgreSQL :

- Image : postgres:15-alpine
- Port : 5432
- Variables d'environnement via ConfigMap et Secret
- PGDATA : /var/lib/postgresql/data/pgdata

Ressources :

- Requests : 1 CPU, 2Gi RAM
- Limits : 2 CPU, 4Gi RAM

Health checks :

- LivenessProbe : pg_isready avec délai initial 30s
- ReadinessProbe : pg_isready avec délai initial 5s

Service :

- Type : Headless (ClusterIP: None)
- Port : 5432

Livrables :

- Fichier `postgres-statefulset.yaml` complet
- Vérification du déploiement

- Test de connexion à la base

Exercice 3 : Surveillance du PVC - Métriques Prometheus (40 min)

Mettre en place une surveillance complète de l'utilisation du PVC.

Partie A : Exporter les Métriques (20 min)

Créer un sidecar container dans le pod PostgreSQL qui :

- Lit régulièrement l'utilisation du filesystem `/var/lib/postgresql/data`
- Expose les métriques au format Prometheus sur le port 9090
- Publie les métriques :
  - `pvc_capacity_bytes` : Capacité totale
  - `pvc_used_bytes` : Espace utilisé
  - `pvc_available_bytes` : Espace disponible
  - `pvc_usage_percent` : Pourcentage d'utilisation

Indice : Utiliser un script shell avec `df` et un serveur HTTP simple (netcat, busybox, ou python)

Livrables :

- ConfigMap contenant le script de monitoring
- Modification du StatefulSet pour ajouter le sidecar
- ServiceMonitor pour Prometheus

Partie B : Configuration Prometheus (20 min)

Configurer Prometheus pour scraper les métriques :

- Créer un ServiceMonitor dans le namespace random-db
- Intervalle de scraping : 15 secondes
- Labels appropriés pour l'identification

Livrables :

- Fichier `servicemonitor.yaml`
- Vérification des targets dans Prometheus UI
- Capture d'écran des métriques dans Prometheus

Exercice 4 : Alertes PrometheusRule (30 min)

Créer des règles d'alerte pour prévenir la saturation du PVC.

Alertes à configurer :

1. **PVCUsageWarning** (70% utilisé)
   - Sévérité : warning
   - Message : "PVC PostgreSQL à 70% de capacité"
   - Durée : 5 minutes
2. **PVCUsageCritical** (85% utilisé)
   - Sévérité : critical
   - Message : "PVC PostgreSQL à 85% - Action urgente requise"

    - Durée : 2 minutes
3. **PVCUsageEmergency** (95% utilisé)
    - Sévérité : emergency
    - Message : "PVC PostgreSQL à 95% - Risque imminent d'interruption"
    - Durée : 1 minute
4. **PVCGrowthRate** (projection saturation < 7 jours)
    - Sévérité : warning
    - Calculer le taux de croissance et alerter si saturation prévue

Livrables :

- Fichier `prometheusrule.yaml`
- Test des alertes avec simulation de saturation
- Configuration Alertmanager (routes, receivers)

Exercice 5 : Dashboard Grafana (25 min)

Créer un dashboard Grafana dédié au monitoring du stockage PostgreSQL.

Panels requis :

1. Gauge : Pourcentage d'utilisation du PVC (avec seuils colorés)
2. Graph : Évolution de l'utilisation sur 7 jours
3. Graph : Taux de croissance quotidien
4. Stat : Espace disponible restant
5. Graph : Projection de saturation (extrapolation linéaire)
6. Table : Liste des plus gros fichiers/tables PostgreSQL
7. Stat : Temps estimé avant saturation

Variables de dashboard :

- Namespace
- Pod PostgreSQL
- Intervalle de temps

Livrables :

- Fichier JSON du dashboard Grafana
- ConfigMap pour provisionner automatiquement le dashboard
- Capture d'écran du dashboard fonctionnel

Exercice 6 : Procédure d'Extension du PVC (20 min)

Documenter et tester la procédure d'extension du PVC en cas de saturation imminente.

Étapes à documenter :

1. Vérifier que le StorageClass supporte l'expansion
2. Éditer le PVC pour augmenter la taille
3. Vérifier l'extension côté Kubernetes
4. Vérifier l'extension dans le pod PostgreSQL
5. Documenter les éventuels redémarrages nécessaires

Test pratique :

- Étendre le PVC de 50Gi à 60Gi
- Documenter le temps d'expansion
- Vérifier qu'aucune interruption de service n'a lieu

Livrables :

- Procédure documentée étape par étape
- Script bash automatisant l'extension
- Logs de test de l'extension

Exercice 7 : Stratégie de Backup (30 min)

Mettre en place une stratégie de backup pour protéger les données.

Méthode 1 : Snapshots de PV

- Créer un VolumeSnapshotClass
- Configurer des snapshots automatiques quotidiens
- Tester la restauration depuis un snapshot

Méthode 2 : pg_dump

- Créer un CronJob exécutant pg_dump
- Stocker les dumps dans un PVC dédié ou S3
- Rotation des backups (conserver 7 jours)

Exigences :

- Backups quotidiens à 2h du matin
- Rétention : 7 jours pour dumps, 3 snapshots
- Test de restauration mensuel

Livrables :

- Fichiers YAML pour les deux méthodes
- Documentation de restauration
- Script de test de restauration

Exercice 8 : Runbook - Gestion de Crise (20 min)

Créer un runbook pour réagir rapidement en cas de saturation.

Sections du runbook :

1. Détection du problème (alertes, symptômes)
2. Diagnostic rapide (commandes de vérification)
3. Actions d'urgence (libération d'espace immédiate)
4. Extension du PVC (procédure détaillée)
5. Nettoyage PostgreSQL (VACUUM, suppression logs)
6. Post-mortem (analyse des causes)
7. Prévention (ajustements de configuration)

Livrables :

- Document Markdown complet du runbook
- Script `emergency-cleanup.sh`
- Checklist imprimable pour NOC

Validation Finale

Checklist :

- [ ] StorageClass configuré avec expansion activée
- [ ] PostgreSQL déployé avec StatefulSet et PVC
- [ ] Métriques exposées et scrapées par Prometheus
- [ ] Alertes configurées et testées
- [ ] Dashboard Grafana fonctionnel
- [ ] Procédure d'extension documentée et testée
- [ ] Stratégie de backup opérationnelle
- [ ] Runbook complet et accessible

Test de Charge : Simuler une montée en charge du stockage :

- Remplir progressivement le PVC avec des données test
- Vérifier le déclenchement des alertes aux bons seuils
- Tester la procédure d'extension sous charge
- Valider la restauration depuis un backup

QCM - Évaluation des Connaissances

Question 1

Pourquoi utiliser un StatefulSet plutôt qu'un Deployment pour PostgreSQL ?

- A) Pour avoir un nom de pod stable et prévisible
- B) Parce que c'est plus rapide
- C) Les Deployments ne supportent pas les PVC
- D) C'est une obligation Kubernetes

Question 2

Que signifie `allowVolumeExpansion: true` dans un StorageClass ?

- A) Le volume sera automatiquement étendu quand il est plein
- B) Il est possible d'étendre manuellement le PVC sans recréer le pod
- C) Le volume peut être monté sur plusieurs pods
- D) Le volume est créé plus rapidement

Question 3

Quelle est la politique de récupération (reclaimPolicy) la plus sûre pour des données critiques ?

- A) Delete - pour libérer l'espace automatiquement
- B) Retain - pour conserver les données même après suppression du PVC
- C) Recycle - pour réutiliser le volume
- D) Aucune différence, c'est pareil

Question 4

Pourquoi utiliser `volumeBindingMode: WaitForFirstConsumer` ?

- A) Pour créer le volume dans la même zone de disponibilité que le pod

- B) Pour accélérer la création du volume
- C) Pour économiser des ressources
- D) C'est obligatoire pour les StatefulSets

Question 5

Quelle commande permet d'étendre un PVC de 50Gi à 100Gi ?

- A) kubectl scale pvc --size=100Gi
- B) kubectl patch pvc -p '{"spec":{"resources":{"requests":{"storage":"100Gi"}}}}'
- C) kubectl resize pvc 100Gi
- D) kubectl expand pvc --to=100Gi

Question 6

Dans Prometheus, quelle fonction permet de calculer le taux de croissance d'une métrique ?

- A) increase()
- B) rate()
- C) predict_linear()
- D) growth_rate()

Question 7

Pourquoi utiliser un sidecar container pour exposer les métriques du PVC ?

- A) C'est plus rapide que node-exporter
- B) Pour avoir des métriques spécifiques au pod sans modifier l'image PostgreSQL
- C) C'est obligatoire pour Prometheus
- D) Pour économiser des ressources

Question 8

Quelle est la différence entre un VolumeSnapshot et un pg_dump ?

- A) VolumeSnapshot est un snapshot du volume entier, pg_dump un export logique SQL
- B) Aucune, c'est la même chose
- C) VolumeSnapshot est pour Kubernetes, pg_dump pour Docker
- D) pg_dump est obsolète

Question 9

Selon le document de passation, quel est le risque principal à surveiller ?

- A) La saturation CPU du pod PostgreSQL
- B) Le nombre de connexions simultanées
- C) La saturation du PVC de la base de données
- D) La latence réseau

Question 10

Dans un ServiceMonitor, quel paramètre définit la fréquence de collecte des métriques ?

- A) scrapeTimeout
- B) scrapeInterval

- C) interval
- D) frequency

Question 11

Pourquoi est-il important d'avoir des alertes à plusieurs seuils (70%, 85%, 95%) ?

- A) Pour faire joli dans Grafana
- B) Pour avoir des niveaux d'urgence progressifs et le temps de réagir
- C) C'est obligatoire pour Prometheus
- D) Pour envoyer plus d'emails

Question 12

Quelle commande permet de vérifier l'utilisation réelle du PVC depuis le pod ?

- A) kubectl get pvc
- B) df -h (dans le pod)
- C) kubectl describe pvc
- D) kubectl top pvc

Question 13

Dans un CronJob de backup, quelle expression cron correspond à "tous les jours à 2h du matin" ?

- A) 0 2 * * *
- B) * 2 * * *
- C) 2 0 * * *
- D) 0 0 2 * *

Question 14

Pourquoi utiliser un service Headless (ClusterIP: None) pour un StatefulSet ?

- A) Pour économiser des adresses IP
- B) Pour avoir un DNS stable pour chaque pod du StatefulSet
- C) C'est obligatoire pour les StatefulSets
- D) Pour des raisons de sécurité

Question 15

Que se passe-t-il si on tente d'étendre un PVC dont le StorageClass ne supporte pas l'expansion ?

- A) L'extension est silencieusement ignorée
- B) Une erreur est retournée et le PVC reste inchangé
- C) Le pod est redémarré automatiquement
- D) Le PVC est supprimé et recréé

Réponses

1. A | 2. B | 3. B | 4. A | 5. B | 6. C | 7. B | 8. A | 9. C | 10. B | 11. B | 12. B | 13. A | 14. B | 15. B