# Rapport d'Analyse sur la Lenteur d'IntelliJ IDEA en Environnement IKS avec Stockage par Blocs VPC à 5 IOPS/Go

## Introduction et Contexte de l'Analyse

Le présent rapport est une évaluation technique approfondie visant à diagnostiquer la cause fondamentale de la lenteur observée lors de l'exécution d'environnements de développement intégrés (IDE) de type IntelliJ IDEA au sein d'un cluster IBM Kubernetes Service (IKS). La configuration spécifique analysée utilise un volume Persistent Volume Claim (PVC) de 40 Go basé sur le service IBM Virtual Private Cloud (VPC) Block Storage, provisionné avec le profil à paliers (tiered profile) de 5 IOPS par gigaoctet (5 IOPS/Go).

L'analyse démontre que le goulot d'étranglement principal est d'origine I/O, spécifiquement le plafond extrêmement bas d'opérations d'entrée/sortie par seconde (IOPS) imposé par le profil de stockage sélectionné. Les exigences d'un IDE moderne, axées sur l'indexation intensive, sont fondamentalement incompatibles avec les caractéristiques de performance offertes par le stockage *tiered* de faible densité. Ce rapport détaille cette incompatibilité, quantifie la dégradation de performance et propose des solutions stratégiques fondées sur l'adoption de profils de performance définie (SDP ou Custom) d'IBM VPC.

## Chapitre 1: Profil d'Exigences d'E/S de l'Environnement de Développement IntelliJ IDEA

Pour comprendre la source de la lenteur, il est essentiel d'analyser le comportement d'un IDE tel qu'IntelliJ, en particulier ses exigences en matière de performances de stockage. Les IDE

de cette catégorie ne sont pas de simples éditeurs de texte; ce sont des systèmes complexes qui dépendent fortement de l'accès rapide et aléatoire aux métadonnées du projet.

## 1.1 La Nature I/O-Intensive de l'Indexation IntelliJ IDEA

Les fonctionnalités centrales qui définissent IntelliJ IDEA—telles que la complétion de code intelligente, les inspections en temps réel, la recherche d'usages, la navigation rapide et le refactoring—reposent toutes sur un processus continu et intensif appelé **indexation**.[1]

L'indexation construit une carte virtuelle complète du code source du projet. Ce processus doit s'exécuter au démarrage du projet, lors du basculement entre branches de code, après le chargement de plugins ou suite à des mises à jour externes majeures de fichiers (comme après une compilation ou une génération de code).[1] Contrairement aux opérations de lecture/écriture séquentielles qui bénéficient d'un débit élevé, l'indexation se caractérise par une multitude d'opérations d'E/S aléatoires de petite taille. Ces opérations consistent en la lecture et l'écriture de milliers, voire de millions, de petits fichiers de métadonnées et d'index qui doivent être accédés et mis à jour très rapidement.

La performance de l'IDE est directement liée à la capacité du système de stockage à gérer ces E/S aléatoires avec une latence minimale. Si le stockage ne parvient pas à répondre aux requêtes d'indexation assez rapidement, l'application entière peut sembler se figer ou devenir léthargique, provoquant des délais importants lors des changements de fichiers ou de l'exécution de processus en arrière-plan.[2]

## 1.2 Priorité à l'IOPS sur le Débit

Le dysfonctionnement perçu par l'utilisateur s'explique par la nature fondamentale des charges de travail de l'IDE : elles sont limitées par le nombre d'opérations (IOPS) plutôt que par le volume total de données transférées (débit).

L'IOPS mesure le nombre d'opérations de lecture ou d'écriture qu'un périphérique de stockage peut exécuter par seconde. Le débit, quant à lui, mesure la quantité de données (en Mo/s ou Mb/s) transférée par seconde.[3] Pour des charges de travail impliquant des requêtes d'E/S aléatoires de petite taille (typiquement 4 KB à 16 KB pour les systèmes de fichiers Linux ou les bases de données d'indexation), le facteur limitant est invariablement l'IOPS.[4]

Étant donné que l'indexation est une tâche hautement synchrone, elle progresse requête par

requête. Si le stockage ne peut traiter qu'un nombre limité de requêtes par seconde, le temps total requis pour indexer un projet se prolonge considérablement. Même si le débit théorique maximum du volume est élevé, il reste inaccessible si le nombre d'opérations (IOPS) est trop faible. En effet, un débit élevé est principalement pertinent pour des E/S séquentielles ou très volumineuses (256 KB à 1 MB).[4] La lenteur observée n'est pas un manque de "tuyau" (débit), mais un manque de "guichets de péage" (IOPS) pour traiter la multitude de petites transactions.

### 1.3 Facteurs Secondaires d'Optimisation du Contexte Kubernetes

Bien que le stockage soit la cause première, l'optimisation des performances de l'IDE dans un conteneur Kubernetes nécessite également de considérer les ressources internes du Pod.

D'une part, l'IDE est une application Java gourmande en ressources. Des problèmes de performance peuvent résulter d'une allocation de mémoire insuffisante pour la machine virtuelle Java (JVM), forçant des cycles constants de *Garbage Collection* (GC), qui consomment des ressources CPU et introduisent des micro-gels, simulant ou aggravant la lenteur du stockage.[7] L'augmentation de la taille du tas (heap size) via l'option

-Xmx est une solution courante.[7] D'autre part, il est crucial, pour réduire la charge d'E/S inutile, d'exclure de l'indexation les dossiers contenant les sorties de compilation ou les fichiers générés dynamiquement, car ils sont sources d'E/S superflues.[1]

Enfin, il convient de souligner que, pour le développement distant (comme dans un conteneur IKS), JetBrains met en garde contre l'utilisation de systèmes de fichiers réseau, tels que le Network File System (NFS) ou SMB, pour le stockage de code principal, insistant sur la nécessité d'un stockage par blocs à faible latence pour respecter la sémantique POSIX et les exigences de performance.[8]

## Chapitre 2: Évaluation Critique de la Configuration Actuelle IBM IKS/VPC

La configuration actuelle de l'utilisateur repose sur le VPC Block Storage d'IBM, un système de stockage par blocs virtualisé et distribué sur le réseau, offrant divers profils de performance. Le choix du profil *tiered* à 5 IOPS/Go est la source directe de la dégradation de l'expérience

utilisateur.

## 2.1 Le Diagnostic Quantifié: Le Plafond Inacceptable de 200 IOPS

Le service IBM VPC Block Storage propose des profils de performance dont les IOPS sont soit fixes et proportionnelles à la taille du volume (profils *tiered*), soit définies explicitement par l'utilisateur (profils *custom* ou SDP).[9]

L'utilisateur a provisionné un volume de 40 Go avec le profil 5iops-tier.[9] La performance maximale garantie (le plafond) est donc calculée comme suit:

Ce plafond de 200 IOPS représente la capacité maximale du stockage à traiter les opérations de lecture/écriture aléatoires par seconde. Dans le contexte d'un IDE intensif comme IntelliJ, qui peut générer des milliers d'opérations d'E/S aléatoires en rafale pendant l'indexation, 200 IOPS est une contrainte intolérable. Ce niveau de performance est comparable à celui d'un disque dur mécanique (HDD) standard sous forte sollicitation.[10]

## 2.2 La Disconnectivité entre Débit Théorique et Débit Effectif

Il est crucial de noter que le profil 5iops-tier peut théoriquement supporter un débit maximal allant jusqu'à 768 MBps.[9] Cependant, ce chiffre est trompeur dans le cas d'une charge de travail I/O-intensive dominée par de petites E/S aléatoires.

En utilisant la formule reliant IOPS, taille moyenne d'E/S, et débit, il est possible de calculer le débit effectif réel lors des phases d'indexation. Si l'on suppose une taille d'E/S aléatoire moyenne de 4 KB (typique pour les workloads Linux et les opérations de métadonnées) [5]:

$$ \text{Débit Effectif} = 200 \text{ IOPS} \times 4 \text{ KB/Opération} / 1024 \approx \mathbf{0.78 \text{ MBps}} $$

Le calcul révèle que, malgré un potentiel théorique de 768 MBps, le système de stockage est effectivement limité à un débit inférieur à 1 MBps pour les opérations critiques de l'IDE. Cette limitation drastique du débit effectif est la manifestation directe des IOPS insuffisantes et constitue la cause principale de la lenteur observée. L'augmentation du débit maximal (par exemple, en passant à un profil 10iops-tier, qui offre 1024 MBps [9]) ne ferait que doubler les IOPS à 400, améliorant marginalement le débit effectif à 1.56 MBps, ce qui resterait largement

insuffisant pour une expérience interactive.

### 2.3 Latence Inhérente au Block Storage Réseau

Le VPC Block Storage est, par définition, un stockage réseau. Le concept même de stockage persistant en cloud (comme les Persistent Disks ou Amazon EBS) implique intrinsèquement une latence plus élevée par rapport aux disques physiques locaux (comme les SSD ou les NVMe).[4]

Les solutions de stockage par blocs réseau opèrent généralement avec une latence de l'ordre de quelques millisecondes (ms) [11], ce qui est nécessaire pour les charges de travail d'entreprise ou les bases de données.[12] Cependant, les SSD NVMe locaux offrent des latences de l'ordre de la microseconde (

), soit des centaines de fois plus rapides.[13]

Dans le cas présent, le faible nombre d'opérations (200 IOPS) est combiné à la latence du réseau pour chaque requête. Ce double handicap crée un effet de goulot d'étranglement cumulatif et perceptible. Le développeur, habitué à la performance réactive d'un NVMe local, subit une dégradation de l'accès aux données par un facteur de performance estimé entre 50 et 100 fois par rapport à un SSD standard.[10] La lenteur observée n'est pas seulement un problème de capacité, mais un effondrement de la performance interactive due à l'incapacité du volume à gérer simultanément un grand nombre de requêtes à faible latence.

## Chapitre 3: Analyse Comparative des Options de Stockage

Pour mieux contextualiser le manque de performance actuel, il est impératif de comparer les 200 IOPS provisionnées avec les références du marché et les alternatives de stockage en environnement Kubernetes.

### 3.1 Performance de Référence du Stockage Local (SSD vs. NVMe)

L'expérience d'un développeur est historiquement ancrée dans la performance des disques locaux.

- **Disques Durs (HDD):** Ils servent de référence historique. Les HDD offrent généralement autour de 100 IOPS pour les opérations aléatoires et sont désormais relégués aux sauvegardes ou aux données d'archivage non critiques.[10] La configuration actuelle à 200 IOPS place le stockage de l'IDE dangereusement proche de cette classe de performance.
- **SSD SATA (Stockage Solide):** L'adoption des SSD a marqué un changement radical, offrant une augmentation des IOPS de l'ordre de 2 à 5 magnitudes par rapport aux HDD.[10] Les SSD standard délivrent couramment entre 10 000 et 50 000 IOPS aléatoires et des latences de l'ordre de quelques centaines de microsecondes ( ).[13] Ce niveau de performance est considéré comme le strict minimum pour maintenir une expérience d'IDE fluide et réactive.
- **NVMe Local (Non-Volatile Memory Express):** Ces disques, utilisant l'interface PCIe, représentent le summum de la performance en stockage local.[15] Les NVMe modernes, tels que ceux utilisés dans les datacenters ou les stations de travail haut de gamme, peuvent atteindre des performances spectaculaires, dépassant 500 000 IOPS, voire 900 000 IOPS en lecture aléatoire, avec des latences critiques mesurées à moins de 100 .[13] Cette performance est essentielle pour les workloads hautement transactionnels et pour minimiser le temps d'attente pendant les processus comme l'indexation.

## 3.2 Définition et Inadéquation de NFS (Network File System)

La requête de l'utilisateur mentionnait également le protocole NFS. Il est crucial de clarifier ce qu'est le NFS dans le contexte Kubernetes et pourquoi il n'est pas une solution de rechange performante pour un IDE.

NFS est un protocole de système de fichiers réseau qui, dans Kubernetes, peut être utilisé pour provisionner un PersistentVolume (PV).[17] L'avantage principal de NFS est sa capacité à supporter le mode d'accès

ReadWriteMany, permettant à plusieurs Pods de monter et d'écrire simultanément sur le même volume, une fonctionnalité non disponible avec le Block Storage classique (qui est ReadWriteOnce).[18]

Cependant, NFS présente plusieurs inconvénients majeurs en matière de performance et de fiabilité pour un usage critique d'IDE :

1. **Fiabilité et Point de Défaillance Unique (SPOF):** Les implémentations NFS reposent souvent sur un serveur unique pour le stockage. Si ce serveur tombe en panne, tous les clients (Pods) perdent l'accès, créant un SPOF.[20]

2. **Dégradation sous Charge Aléatoire:** Bien que NFS puisse être rapide sur un réseau local pour des transferts séquentiels, il est connu pour subir une dégradation de performance significative lors du traitement de volumes élevés de données ou d'un grand nombre de connexions concurrentes, en particulier lors d'opérations aléatoires.[20] Cela est souvent dû à la surcharge du protocole RPC et à la gestion des tables d'emplacements RPC (*RPC slot tables*) [21], qui ajoutent de la latence.
3. **Incompatibilité Logicielle Avérée:** Comme mentionné précédemment, la documentation JetBrains pour le développement distant exclut spécifiquement les systèmes de fichiers réseau tels que NFS ou SMB, car ils ne fournissent pas les garanties de faible latence et la cohérence de la sémantique POSIX requises pour un fonctionnement optimal de l'IDE.[8]

Par conséquent, si le stockage par blocs actuel est trop lent, migrer vers NFS n'est pas une solution viable pour améliorer la réactivité de l'IDE; cela risque au contraire d'introduire une latence encore plus élevée et imprévisible.

## 3.3 Synthèse Comparative des Performances de Stockage

Le tableau suivant récapitule la performance relative des différentes technologies de stockage, démontrant l'écart monumental entre la configuration actuelle et les exigences minimales pour un environnement de développement professionnel.

Comparaison des Profils de Stockage I/O pour Applications IDE

| Type de Stockage | Configuration Clé | IOPS Aléatoires (4K) | Latence Typique | Débit Effectif @ 4KB | Pertinence pour IntelliJ/IKS |
|---|---|---|---|---|---|
| HDD Standard | Disque mécanique | | ms | MBps | Cause de lenteur garantie [10] |
| **VPC Block (Actuel)** | **40 GB @ 5 IOPS/GB** | **200 (Plafond)** [9] | **ms (Réseau)** [4] | **MBps** | **Cause Racine de la Lenteur** |

| | | | | | |
|---|---|---|---|---|---|
| VPC Block (Cible Recommandée) | SDP/Custom @ 5,000 IOPS | | ms | MBps | Minimum pour l'interactivité |
| SSD SATA Local | Interface SATA III | | | MBps | Référence de performance acceptable [10] |
| NVMe Local (PCIe) | PCIe Gen 4/5 | | | MBps | Performance Optimale [13] |
| NFS Réseau | Dépend du serveur/réseau | Variable (Souvent faible) | Élevée, Incohérente [20] | Variable | Inapproprié (Anti-pattern pour IDE) [8] |

L'analyse de ce tableau confirme que le volume Block Storage provisionné est nettement plus proche d'un HDD que d'un SSD en termes de performance d'E/S aléatoires, rendant l'expérience de développement insoutenable.

# Chapitre 4: Recommandations Stratégiques et Migration vers la Performance Définie

La résolution du problème nécessite un changement fondamental dans le modèle de provisionnement du stockage par blocs, en déplaçant l'approche de la densité (IOPS/Go) vers la performance garantie (IOPS fixes).

## 4.1 Migration Impérative vers les Profils SDP ou Custom d'IBM VPC

Les profils *tiered* (3iops, 5iops, 10iops) d'IBM lient strictement les IOPS à la taille du volume,

les rendant inadaptés aux charges de travail I/O-intensive qui nécessitent une petite capacité (40 Go) mais une performance transactionnelle élevée. Même si l'utilisateur augmentait la taille du volume à 100 Go sur le profil 10iops-tier, cela ne fournirait que 1000 IOPS, ce qui reste très loin des performances SSD minimales souhaitées.

La solution consiste à adopter les profils qui permettent de découpler les IOPS du volume : le profil **Custom** ou le profil **SDP (Software Defined Performance)**.[9]

1. **Le Profil Custom:** Ce profil permet de provisionner explicitement une valeur d'IOPS dans une plage allant de 1 à 48 000 IOPS.[9] Il garantit une performance constante indépendante de la taille du volume (dans la limite de la taille maximale supportée).
2. **Le Profil SDP:** Le profil SDP est conçu pour des exigences de très haute performance et de faible latence, permettant un provisionnement allant jusqu'à 64 000 IOPS.[9] C'est le choix technique optimal pour simuler la réactivité d'un SSD local dans un environnement de stockage par blocs réseau.

**Stratégie de Provisionnement Recommandée:**

Il est recommandé de configurer un nouveau PVC utilisant le profil **SDP** (ou Custom) et de demander un provisionnement minimal de **5 000 IOPS**. Un niveau optimal, visant à minimiser la latence de l'indexation, se situerait entre **8 000 et 10 000 IOPS**. Cette augmentation massive des IOPS (passant de 200 à 5 000 ou plus) permettra de traiter la densité de petites E/S aléatoires requises par IntelliJ, restaurant ainsi la productivité du développeur.

Bien que le profil SDP/Custom puisse engendrer des coûts de stockage plus élevés que le profil 5 IOPS/Go, cette dépense est largement justifiée par l'amélioration de la productivité du développeur, dont le temps perdu à attendre les processus d'indexation dépasse rapidement l'économie réalisée sur un stockage sous-performant.

## 4.2 Mesures d'Atténuation et d'Optimisation du Pod

Le changement de profil de stockage est la mesure la plus critique, mais elle doit être complétée par des ajustements au niveau du Pod et de l'IDE pour assurer une performance maximale.

- **Optimisation de l'Allocation Mémoire de la JVM:** L'image du conteneur exécutant IntelliJ doit être configurée pour augmenter l'allocation maximale de mémoire heap (-Xmx). Si la valeur par défaut est trop basse (par exemple 750 Mo), la JVM passe un temps excessif en *Garbage Collection*, ce qui rend l'IDE non réactif et monopolise le CPU.[7] Une allocation de 4 Go à 8 Go de RAM est souvent nécessaire pour les projets de taille moyenne à grande.

- **Gestion des Ressources Kubernetes:** Le Pod doit spécifier des requests et des limits adéquates pour le CPU et la mémoire.[22] Si les ressources CPU sont limitées ( *throttled*) par le Kubelet, même avec un stockage rapide, les processus intensifs comme l'indexation ou la compilation seront ralentis. Il est recommandé de garantir une réserve de 4 vCPU minimum et 8 Go de RAM pour le développement à distance.[8]
- **Exclusion des Fichiers Générés:** S'assurer que les dossiers de sortie de compilation (ex: target/, build/) sont exclus de l'indexation de l'IDE. Cela réduit considérablement la charge d'E/S aléatoire générée par les opérations de construction de projet.[1]

### 4.3 Alternative Architecturelle: Utilisation de Disques Locaux NVMe (Instance Store)

L'alternative offrant la latence la plus faible est l'utilisation de disques SSD NVMe locaux (ou *instance store*) si les nœuds IKS sous-jacents les supportent. Ces disques sont directement attachés au matériel physique et offrent les performances mentionnées dans la section 3.1 (latence en microsecondes).

Kubernetes permet de gérer ce type de stockage via des Local Persistent Volumes. Cependant, il faut noter que ces disques sont éphémères; les données stockées dessus sont perdues si le nœud de travail est redémarré ou re-provisionné.[23] Cette solution n'est donc viable que si l'état de l'IDE (code source et index) est considéré comme non critique, ou si un mécanisme de synchronisation externe robuste est mis en place pour sauvegarder le travail fréquemment. Pour un environnement de développement stable et persistant, le Block Storage réseau à performance définie (SDP/Custom) reste le choix le plus approprié et le plus gérable.

## Conclusions et Recommandations

L'analyse technique confirme que la lenteur de l'IDE IntelliJ dans l'environnement IKS est due à un **goulot d'étranglement sévère au niveau des IOPS** du stockage par blocs VPC, provisionné au profil 5iops-tier de 40 Go.

La performance actuelle de 200 IOPS est catastrophiquement insuffisante pour la charge de travail d'E/S aléatoires générée par l'indexation de l'IDE, limitant le débit effectif à moins de 1 MBps pour ces opérations critiques. Ce niveau est comparable à celui d'un disque dur

mécanique en saturation.

Les recommandations pour résoudre ce dysfonctionnement sont les suivantes :

1. **Migration du Profil de Stockage (Priorité Absolue):** Migrer immédiatement le Persistent Volume Claim vers un profil de performance définie IBM VPC, soit **Custom**, soit **SDP**.
2. **Provisionnement d'IOPS:** Provisionner le nouveau volume à un niveau d'IOPS minimum de **5 000**, et idéalement de 8 000 à 10 000 IOPS, pour aligner la performance sur celle d'un SSD standard et garantir une expérience utilisateur interactive et réactive.
3. **Exclusion et Optimisation:** Configurer l'IDE pour exclure les dossiers de compilation de l'indexation et ajuster les options de la JVM dans l'image du conteneur pour augmenter l'allocation de mémoire (-Xmx), réduisant ainsi la pression sur le CPU et la mémoire.
4. **Exclusion de NFS:** Éviter l'utilisation de NFS pour le stockage du répertoire de travail principal de l'IDE, car ce protocole est explicitement non recommandé pour les charges de travail exigeantes en latence aléatoire.

### Sources des citations


1. Indexing | IntelliJ IDEA Documentation - JetBrains, consulté le octobre 4, 2025, https://www.jetbrains.com/help/idea/indexing.html
2. Performance Issues Running IntelliJ IDEA on Low-Spec AMD Laptops, consulté le octobre 4, 2025, https://intellij-support.jetbrains.com/hc/en-us/community/posts/24664493797650-Performance-Issues-Running-IntelliJ-IDEA-on-Low-Spec-AMD-Laptops
3. Throughput and IOPS: how much is enough? - ntorga's, consulté le octobre 4, 2025, https://ntorga.com/throughput-and-iops-how-much-is-enough/
4. Persistent Disk performance overview | Compute Engine - Google Cloud, consulté le octobre 4, 2025, https://cloud.google.com/compute/docs/disks/performance
5. Block Storage capacity and performance - IBM Cloud Docs, consulté le octobre 4, 2025, https://cloud.ibm.com/docs/vpc?topic=vpc-capacity-performance
6. IOPS versus Throughput - storage - Stack Overflow, consulté le octobre 4, 2025, https://stackoverflow.com/questions/15759571/iops-versus-throughput
7. Any tools for benchmarking IntelliJ itself? : r/IntelliJIDEA - Reddit, consulté le octobre 4, 2025, https://www.reddit.com/r/IntelliJIDEA/comments/eclj9l/any_tools_for_benchmarking_intellij_itself/
8. System requirements for remote development | IntelliJ IDEA Documentation - JetBrains, consulté le octobre 4, 2025, https://www.jetbrains.com/help/idea/prerequisites.html
9. Block Storage for VPC profiles - IBM Cloud Docs, consulté le octobre 4, 2025, https://cloud.ibm.com/docs/vpc?topic=vpc-block-storage-profiles
10. Latency of SSDs versus HDDs - Super User, consulté le octobre 4, 2025, https://superuser.com/questions/1414662/latency-of-ssds-versus-hdds

11. Choisir son stockage pour les charges de travail d'IA et de ML dans Google Cloud, consulté le octobre 4, 2025, https://cloud.google.com/architecture/ai-ml/storage-for-ai-ml?hl=fr
12. Qu'est-ce que le stockage en bloc ? | OVHcloud, consulté le octobre 4, 2025, https://www.ovhcloud.com/fr/learn/what-is-block-storage/
13. SSDs: Understanding IOPS vs. Latency (in Random read/write) : r/hardware - Reddit, consulté le octobre 4, 2025, https://www.reddit.com/r/hardware/comments/ss8awp/ssds_understanding_iops_vs_latency_in_random/
14. Choose an Azure storage service - Azure Architecture Center | Microsoft Learn, consulté le octobre 4, 2025, https://learn.microsoft.com/en-us/azure/architecture/guide/technology-choices/storage-options
15. NVMe vs. M.2: What's the difference? - IBM, consulté le octobre 4, 2025, https://www.ibm.com/think/topics/nvme-vs-m2
16. Comprendre la technologie SSD : NVMe, SATA, M.2 - Kingston Technology, consulté le octobre 4, 2025, https://www.kingston.com/fr/ssd/what-is-nvme-ssd-technology
17. Persistent Volumes - Kubernetes, consulté le octobre 4, 2025, https://kubernetes.io/docs/concepts/storage/persistent-volumes/
18. NFS in kubernetes - Persistent volume type - YouTube, consulté le octobre 4, 2025, https://www.youtube.com/watch?v=kKiHJrq_kmo
19. Block Storage vs. File Storage for Kubernetes: Does Using an NFS Server on Top of Block Storage Address the ReadOnce Limitation? - Reddit, consulté le octobre 4, 2025, https://www.reddit.com/r/kubernetes/comments/1fz6hv5/block_storage_vs_file_storage_for_kubernetes_does/
20. How reliable is NFS for using in Kubernetes? - DevOps Stack Exchange, consulté le octobre 4, 2025, https://devops.stackexchange.com/questions/17004/how-reliable-is-nfs-for-using-in-kubernetes
21. NFS in NetApp ONTAP Best practice and implementation guide, consulté le octobre 4, 2025, https://www.netapp.com/media/10720-tr-4067.pdf
22. Resource Management for Pods and Containers - Kubernetes, consulté le octobre 4, 2025, https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/
23. Choosing an AWS storage service, consulté le octobre 4, 2025, https://docs.aws.amazon.com/decision-guides/latest/storage-on-aws-how-to-choose/choosing-aws-storage-service.html