



Veille Technologique Quotidienne

Technique

11 septembre 2026 – Fréquence : Quotidienne (du Lundi au Vendredi)

Synthèse & Sujet du Jour

GitOps pour le LLMOps en 2026 : Déploiement Déclaratif de LLM avec OpenShift GitOps (ArgoCD), KServe et vLLM

Guide complet pour implémenter une approche GitOps sur vos modèles d'IA : déploiement déclaratif d'InferenceServices, gestion des adaptateurs LoRA et Canary Releases sur Red Hat OpenShift AI.

1. Périmètre

Famille de veille : 2. Métiers, Compétences & Technologies.

Fréquence d'exécution : Quotidienne (du Lundi au Vendredi à 08h30).

Diffusion : [Radar Web Aperta](#) — [LinkedIn Aperta Scientia](#) — [X @ApertaScientia](#).

2. Sujet du Jour — Article Publié sur le Radar Web

GitOps pour le LLMOps en 2026 : Déploiement Déclaratif de LLM avec OpenShift GitOps (ArgoCD), KServe et vLLM

L'adoption massive des modèles de fondation et des architectures d'inférence distribuée en entreprise pose un défi opérationnel majeur : **comment déployer, versionner et promouvoir des modèles d'IA en production avec la même rigueur, traçabilité et automatisation que le code applicatif ?**

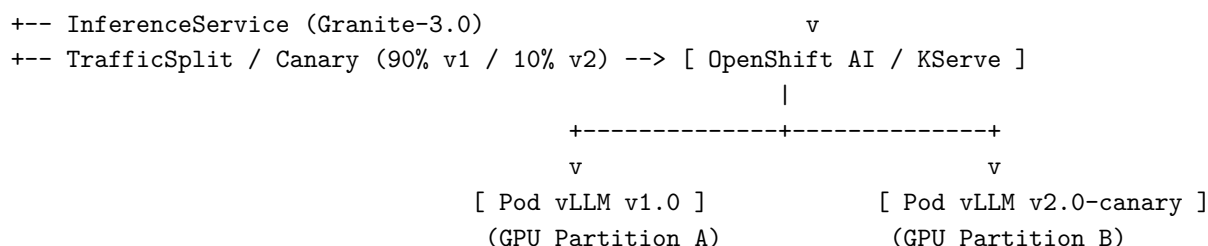
Trop souvent, le cycle de vie des modèles d'IA souffre d'actions manuelles en console, de scripts ad hoc pour télécharger des poids de dizaines de gigaoctets, et d'un manque d'observabilité sur les versions d'infrastructures d'inférence actives.

En 2026, la convergence entre le **Platform Engineering cloud-native** et le **LLMOps** impose une approche : **le GitOps pour l'IA**. En combinant **OpenShift GitOps (ArgoCD)**, **KServe** et le runtime d'inférence **vLLM** sur **Red Hat OpenShift AI (RHOAI)**, les équipes d'infrastructure déclarent l'état cible de leurs modèles d'IA dans des dépôts Git audités, automatisant les déploiements, les *Canary Releases* et les *rollbacks* instantanés.

1. Les Fondamentaux du GitOps appliqué aux Workloads d'IA

Le paradigme GitOps applique les principes d'infrastructure déclarative (IaC) et de réconciliation continue à la couche d'inférence et d'entraînement IA :

```
[ Dépôt Git : manifests/ ] --(Sync Declarative)--> [ OpenShift GitOps (ArgoCD) ]
+++ ServingRuntime (vLLM)                          | (Reconciliation Loop)
```



1. **Tout est déclaré dans Git** : La configuration du moteur d'inférence (taille de contexte, *tensor parallelism*, quantification AWQ/FP8), la source des poids (registre OCI, bucket S3 souverain ou Hugging Face sécurisé), les quotas GPU et les politiques d'autoscaling.
2. **Immutabilité des versions de modèles** : Chaque promotion d'une nouvelle version de modèle (ex: passage de `granite-3.0-8b-instruct-v1` à `granite-3.0-8b-instruct-v2-lora`) fait l'objet d'une Pull Request documentée et audité.
3. **Réconciliation automatique & Anti-Drift** : Le contrôleur ArgoCD détecte immédiatement toute divergence manuelle sur le cluster OpenShift et force le retour à l'état désiré.

2. Spécification Déclarative d'un InferenceService KServe avec vLLM

Sous OpenShift AI, un modèle LLM est exposé sous la forme d'une ressource personnalisée (CRD) `InferenceService`. Voici la déclaration GitOps standard pour déployer le modèle souverain **IBM Granite 3.0 8B** accéléré par le runtime `vLLM` :

```
apiVersion: serving.kserve.io/v1beta1
kind: InferenceService
metadata:
  name: granite-3-8b-instruct
  namespace: generative-ai-prod
  annotations:
    openshift.io/display-name: "Granite 3.0 8B Instruct - Production"
    serving.kserve.io/deploymentMode: "Serverless"
    sidecar.istio.io/inject: "true"
spec:
  predictor:
    maxReplicas: 4
    minReplicas: 1
    scaleTarget: 15 # Concurrence de requêtes par réplique avant scale-out
    scaleMetric: concurrency
    model:
      modelFormat:
        name: vLLM
      runtime: vllm-runtime-v0-8
      storageUri: s3://models-registry/granite-3.0-8b-instruct/
      resources:
        limits:
          cpu: "8"
          memory: 32Gi
          nvidia.com/gpu: "1"
        requests:
```

```
cpu: "4"
memory: 16Gi
nvidia.com/gpu: "1"
env:
  - name: MAX_MODEL_LEN
    value: "8192"
  - name: GPU_MEMORY_UTILIZATION
    value: "0.90"
  - name: KV_CACHE_DTYPE
    value: "fp8"
```

3. Stratégie de Déploiement Progressif : Canary Release & Traffic Splitting

Le déploiement d'un nouveau modèle de langage en production comporte des risques d'incompatibilité de prompt, de régression de latence (*Time To First Token*) ou de dégradation du débit de génération.

Grâce à l'intégration native d'OpenShift Serverless (Knative Serving) et Istio dans KServe, il est possible de découper le trafic de manière granulaire et déclarative via GitOps :

```
apiVersion: serving.knative.dev/v1
kind: Service
metadata:
  name: granite-serving-router
  namespace: generative-ai-prod
spec:
  template:
    metadata:
      annotations:
        autoscaling.knative.dev/target: "10"
  traffic:
    - tag: current
      revisionName: granite-3-8b-instruct-00001 # Version stable v1
      percent: 90
    - tag: candidate
      revisionName: granite-3-8b-instruct-00002 # Version candidate v2 (LoRA fine-tuné)
      percent: 10
```

Workflow d'orchestration GitOps :

- Étape 1 (Canary 10%)** : La PR GitOps est fusionnée sur la branche **staging-prod**. ArgoCD déploie la révision candidate et achemine 10% des requêtes réelles.
- Étape 2 (Validation Automated SLO)** : Les métriques Prometheus / OpenTelemetry (P99 latency, rate d'erreur 5xx, hallucinations via TrustyAI) sont monitorées pendant 15 minutes.
- Étape 3 (Promotion 100% ou Rollback)** : Si les SLOs sont respectés, un job automatisé ou l'opérateur de promotion met à jour le manifest Git à 100% sur la révision candidate. En cas d'anomalie, un simple `git revert` rétablit instantanément 100% du trafic sur la révision précédente sans interruption de service.



4. Gestion Déclarative des Adaptateurs Multi-LoRA

Plutôt que de dupliquer des pods de serving de 8B ou 70B pour chaque cas d'usage métier, l'architecture moderne exploite les capacités **Multi-LoRA de vLLM** orchestrées par GitOps :

- Le modèle de base (*base model*) reste chargé une seule fois dans la VRAM du GPU.
- Les adaptateurs LoRA légers (quelques dizaines de mégaoctets) pour le service juridique, le support client ou la génération de code sont versionnés dans Git comme des sous-ressources déclaratives.
- Le routage dynamique vers le bon adaptateur s'effectue directement dans le corps de la requête d'inférence (`model: "granite-3.0-8b/support-client"`), garantissant une densité d'usage maximale des GPUs d'entreprise.

5. Tableau Comparatif : Déploiement Traditionnel vs GitOps LLMOps

6. Maîtriser le GitOps et l'Architecture Plateforme IA chez Aperta Scientia

L'automatisation du cycle de vie des plateformes d'IA d'entreprise nécessite une expertise pointue à l'intersection des opérations cloud-native et des architectures de serving de modèles :

- **Cursus AS200 — DevOps & Platform Engineering (3 mois)** : Le cursus de référence pour maîtriser l'automatisation Ansible, l'administration avancée OpenShift, la CI/CD Tekton et le déploiement déclaratif avec **OpenShift GitOps (ArgoCD)**.
- **Cursus AS300 — AI Platform Engineer (3 mois)** : La formation intensive dédiée à l'orchestration de **Red Hat OpenShift AI (RHOAI)**, KServe, vLLM, KubeRay, la gouvernance TrustyAI et la préparation à la certification officielle **Red Hat Certified Specialist in AI/ML (EX267)**.

→ [__HREF_0__](#) | [__HREF_1__](#) | [__HREF_2__](#)

7. Documentation & Références Techniques

- **OpenShift GitOps Documentation** : [__HREF_0__](#)
- **KServe Declarative Inference Architecture** : [__HREF_0__](#)
- **vLLM Multi-LoRA Serving** : [__HREF_0__](#)
- **Red Hat OpenShift AI Serving Guide** : [__HREF_0__](#)

3. Impact sur les Cursus Aperta Scientia

Cursus AS200 — DevOps Platform Engineer

Cette veille alimente directement les modules d'ingénierie de plateforme et de sécurisation avancée de la chaîne de livraison logicielle :

- **Construction & CI/CD (D0288)** : Conception de pipelines déclaratifs Tekton, gestion des builds conteneurisés en environnement sans privilèges (*rootless*) et automatisation via Pipelines-as-Code.



- **Exploitation & Durcissement (D0380)** : Gestion multi-cluster, observabilité de la chaîne de livraison et politiques de conformité à l'échelle.
- **Sécurisation Conteneurs & Plateforme (D0430)** : Implémentation du contrôle d'admission (Sigstore Policy Controller, RHACS), signature keyless d'artefacts (RHTAS), génération de SBOM et analyse continue d'exploitabilité (TPA / VEX).
- **Certifications préparées** : Préparation opérationnelle aux examens **EX188** (Containers), **EX280** (OpenShift Administration) et **EX288** (OpenShift Development).

Cursus AS300 — AI Platform Engineer

La démarche de sécurisation de la supply chain logicielle est transposée à l'infrastructure d'inférence et aux pipelines MLOps :

- **Traçabilité des dépendances GPU & ML** : Audit de provenance et validation SLSA des images de base contenant les runtimes CUDA, ROCm et PyTorch.
- **Signature & Contrôle des Modèles** : Signature cryptographique des poids de modèles et des runtimes de serving vLLM / KServe avant déploiement en cluster.
- **Gouvernance & SBOM IA** : Inventaire automatisé des licences et détection proactive des vulnérabilités sur la pile MLOps (KubeFlow Pipelines, Ray, Triton).

4. Sources

- **Red Hat Developer Blog** — [Lien](#)
- **OpenShift Documentation** — [Lien](#)
- **Kubernetes Community Documentation** — [Lien](#)
- **CNCF Cloud Native Landscape** — [Lien](#)
- **vLLM Project Documentation** — [Lien](#)
- **Sigstore Documentation** — [Lien](#)
- **SLSA Framework** — [Lien](#)

Contacts & Communauté :

- **Direction des Formations** : learning@apertascientia.io
- **Référent Handicap & Bien-être (Care)** : care@apertascientia.io
- **X (Twitter)** : [@ApertaScientia](https://twitter.com/ApertaScientia)
- **Communauté Discord** : discord.gg/eDJzR5Vk85