



Daily Technology Watch

Technology

11 September 2026 – Cadence: Daily (Monday to Friday)

Summary & Topic of the Day

GitOps for LLMOps in 2026: Declarative LLM Deployment with OpenShift GitOps (ArgoCD), KServe, and vLLM

Comprehensive guide to implementing GitOps for enterprise AI models: declarative InferenceServices, LoRA adapter management, and canary rollouts on Red Hat OpenShift AI.

1. Scope

Watch family: 2. Jobs, Skills & Technologies.

Run cadence: Daily (Monday to Friday at 08:30).

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

2. Topic of the Day — Article Published on the Web Radar

GitOps for LLMOps in 2026: Declarative LLM Deployment with OpenShift GitOps (ArgoCD), KServe, and vLLM

The widespread enterprise adoption of foundation models and distributed inference architectures introduces a major operational challenge: **how do you deploy, version, and promote AI models in production with the same rigor, auditability, and automation as application code?**

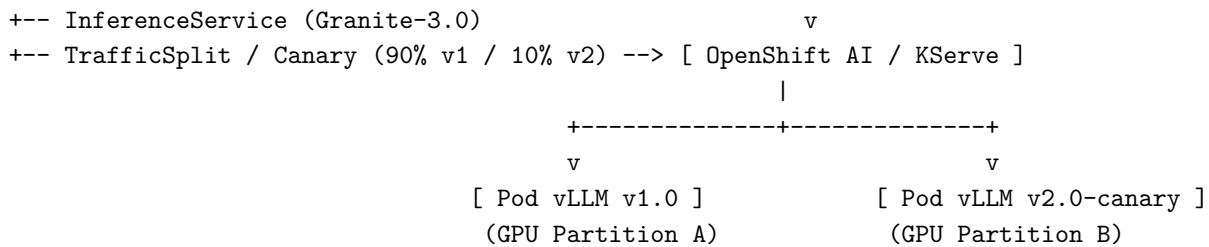
Too often, the AI model lifecycle relies on manual console clicks, ad hoc scripts downloading multi-gigabyte weights, and a lack of real-time visibility into active inference infrastructure revisions.

In 2026, the convergence between **cloud-native Platform Engineering** and **LLMOps** establishes an essential standard: **GitOps for AI**. By combining **OpenShift GitOps (ArgoCD)**, **KServe**, and the **vLLM** high-performance inference engine on **Red Hat OpenShift AI (RHOAI)**, infrastructure teams declare the target state of their AI models in audited Git repositories, enabling automated deployments, zero-downtime *Canary Releases*, and instant rollbacks.

1. Fundamentals of GitOps for AI Workloads

The GitOps paradigm extends declarative Infrastructure as Code (IaC) and continuous reconciliation principles to the AI inference and training layers:

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



1. **Everything is Declared in Git:** Inference engine parameters (context length, tensor parallelism, AWQ/FP8 quantization), model weight sources (OCI registry, sovereign S3 bucket, or authenticated Hugging Face endpoint), GPU resource allocations, and autoscaling policies.
2. **Model Version Immutability:** Every promotion of a new model version (e.g., upgrading from `granite-3.0-8b-instruct-v1` to `granite-3.0-8b-instruct-v2-lora`) is managed via an audited, peer-reviewed Pull Request.
3. **Automated Reconciliation & Drift Detection:** The ArgoCD controller immediately detects out-of-band cluster modifications and enforces the desired Git state automatically.

2. Declarative KServe InferenceService Specification with vLLM

Under OpenShift AI, an LLM is modeled as a declarative `InferenceService` Custom Resource Definition (CRD). Here is the standard production manifest for deploying the sovereign **IBM Granite 3.0 8B** model accelerated by `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 # Target concurrency per replica before scaling 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. Progressive Delivery: Declarative Canary Releases & Traffic Splitting

Deploying a newly fine-tuned language model directly to 100% of production traffic carries severe risks of prompt regressions, Time-To-First-Token (TTFT) latency spikes, or token throughput degradation.

Leveraging OpenShift Serverless (Knative Serving) and Istio integrated within KServe, teams can declare progressive traffic splitting directly 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 # Production baseline v1
      percent: 90
    - tag: candidate
      revisionName: granite-3-8b-instruct-00002 # Candidate revision v2 (Fine-tuned LoRA)
      percent: 10
```

GitOps Orchestration Workflow:

- Phase 1 (10% Canary Routing):** The PR merges into the `staging-prod` branch. ArgoCD synchronizes the candidate revision and routes 10% of live incoming user traffic.
- Phase 2 (Automated SLO Verification):** Prometheus and OpenTelemetry metrics (P99 latency, 5xx error rate, TrustyAI hallucination scores) are continuously evaluated during a 15-minute verification window.
- Phase 3 (100% Promotion or Instant Rollback):** Upon SLO confirmation, the declarative manifest is updated to 100% candidate traffic. If regressions are detected, a single `git revert` restores 100% traffic to the baseline revision within seconds with zero downtime.



4. Declarative Management of Multi-LoRA Adapters

Instead of maintaining dedicated 8B or 70B model replicas for every specialized department, modern enterprise architectures leverage **vLLM Multi-LoRA capabilities** orchestrated via GitOps:

- The heavy base foundation model is loaded once into GPU VRAM.
- Lightweight task-specific LoRA adapters (dozens of megabytes each) for legal parsing, customer support, or code generation are tracked as declarative sub-resources in Git.
- Dynamic request routing targets specific adapters on-the-fly (`model: "granite-3.0-8b/customer-support"`), maximizing enterprise GPU utilization density and slashing infrastructure overhead.

5. Architectural Comparison: Traditional AI vs GitOps LLMOps

6. Master Cloud-Native GitOps and AI Platforms at Aperta Scientia

Automating enterprise AI platform lifecycles requires specialized expertise spanning both cloud-native infrastructure engineering and cutting-edge model serving runtimes:

- **AS200 — DevOps & Platform Engineering (3 months):** The premier program mastering Ansible automation, advanced OpenShift administration, Tekton CI/CD pipelines, and declarative GitOps with **OpenShift GitOps (ArgoCD)**.
- **AS300 — AI Platform Engineer (3 months):** The intensive advanced curriculum dedicated to architecting **Red Hat OpenShift AI (RHOAI)**, KServe, vLLM, KubeRay, TrustyAI governance, and preparing for the official **Red Hat Certified Specialist in AI/ML (EX267)** certification.

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

7. Documentation & Technical References

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

3. Impact on Aperta Scientia Curricula

AS200 Track — DevOps Platform Engineer

This watch directly feeds into our platform engineering and software supply chain security modules:

- **Builds & CI/CD (D0288) :** Designing declarative Tekton pipelines, managing rootless container builds, and automating workflows via Pipelines-as-Code.



- **Operations & Hardening (D0380)** : Multi-cluster management, delivery pipeline observability, and enterprise-wide compliance policies.
- **Container & Platform Security (D0430)** : Implementing admission enforcement (Sigstore Policy Controller, RHACS), keyless artifact signing (RHTAS), SBOM generation, and automated exploitability triage (TPA / VEX).
- **Target Certifications** : Practical preparation for **EX188** (Containers), **EX280** (OpenShift Administration), and **EX288** (OpenShift Development).

AS300 Track — AI Platform Engineer

Supply chain security principles are extended to inference infrastructure and MLOps pipelines:

- **GPU & ML Dependency Traceability** : Provenance auditing and SLSA verification for base images embedding CUDA, ROCm, and PyTorch runtimes.
- **Model Signing & Admission Control** : Cryptographic signing of model weights and vLLM / KServe serving runtimes before cluster deployment.
- **AI Governance & SBOM** : Automated license inventory and continuous vulnerability scanning across the entire MLOps stack (KubeFlow Pipelines, Ray, Triton).

4. Sources

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

Contacts & Community:

- Academic Programs: learning@apertascientia.io
- Disability Officer & Student Care: care@apertascientia.io
- X (Twitter): [@ApertaScientia](https://twitter.com/ApertaScientia)
- Discord Community: discord.gg/eDJzR5Vk85