Task 4 – Model Research and Evaluation

Data Bytes – Project Echo - Engine Team

**1. Executive Summary**

* Project Echo’s Engine Team needs a centralised experiment-tracking and model-registry solution to replace its manual TensorFlow save/load workflow, automate version control, and improve CI/CD integration and team collaboration under on-prem security constraints.
* Eight tools (MLflow, DVC, Guild AI, Metaflow, W&B, Comet, Neptune.ai, Azure ML) were evaluated against feature set, usability, integration, performance, cost, community support and maintenance.
* MLflow and DVC scored highest: MLflow for its end-to-end tracking, native registry and UI; DVC for its Git-native data/pipeline versioning and lightweight CLI.  
  Recommend adopting MLflow as the primary platform, with DVC as a complementary Git-centric tool for dataset and pipeline management.

**2. Context & Requirements**

**2.1 Problem Space**

Echo Engine’s existing codebase relies extensively on **TensorFlow** (with limited **PyTorch** usage) and employs direct file-based saving/loading for models. It lacks any dedicated experiment tracking or registry tool, as indicated by the absence of references to MLflow, Weights & Biases, or similar frameworks. Consequently, **version control** for model iterations and **reproducibility** of experiments is largely manual and decentralised. This setup constrains collaboration, hinders easy rollback to previous model states, and complicates performance comparisons across different training runs.

**2.2 Echo Engine’s Operational Needs**

1. **Model Lifecycle Management**  
   The code demonstrates repeated model.save(...) and tf.keras.models.load\_model(...) calls with various naming conventions. A registry should automate this process, storing model artefact metadata (e.g., date, hyperparameters, performance metrics) for each version.
2. **CI/CD Integration**  
   Although Docker and docker-compose are prevalent, there is no mention of automated pipelines (Airflow, Prefect, etc.). Any chosen tool should easily plug into containerised workflows and support future expansions into CI/CD for model deployment and validation.
3. **Collaboration Features**  
   Multiple environment YAMLs hint at distributed teams or varied dev setups. A registry must enable shared experiment logs, ensuring consistent references to model versions without requiring manual naming or file handling.
4. **On-Prem vs Cloud**  
   Hardcoded credentials (mongodb://root:root\_password@...) and concerns about exposing data externally suggest an **on-prem or private cloud** preference for compliance and data governance.
5. **Security/Compliance**  
   JWT-based user auth indicates role-based access requirements for model data. The solution must respect user roles, handle sensitive data, and potentially integrate with existing authentication layers.

**2.3 Evaluation Criteria**

1. **Feature Set** – Must support experiment logs (hyperparams, metrics, artefacts), model registry, lineage tracking.
2. **Usability** – Clear UI/CLI for data scientists, devops, and researchers.
3. **Integration Capability** – Python SDK, Docker-based workflows, minimal friction with TF and Torch.
4. **Performance/Scalability** – Efficient handling of large model files and frequent experiment logs.
5. **Cost** – Transparent licensing (open-source vs enterprise).
6. **Open-Source/Community Support** – Active development, frequent releases, large user base.
7. **Maintenance/Updates** – Ease of upgrades, available support channels, straightforward installations and patches.

**3. Tool Shortlist**

Below are **eight** representative tools to consider for model registry and experiment tracking. They include both open-source and commercial options, providing a broad view of maturity and ecosystem fit:

1. **MLflow (Open-Source)**
   * **Purpose & Fit**: End-to-end platform for experiment tracking, model packaging, and deployment. Integrates seamlessly with Python ML libraries and supports both TensorFlow and PyTorch out of the box.
   * **Maturity**: Backed by Databricks with a large OSS community; frequent releases and strong plugin ecosystem.
2. **DVC (Open-Source)**
   * **Purpose & Fit**: Think “Git for data” with experiment versioning. Strong synergy with code version control (Git) and can handle large files.
   * **Maturity**: Actively developed; widely adopted for dataset and ML pipeline versioning. Less emphasis on UI-based experiment tracking but excellent CLI.
3. **Guild AI (Open-Source)**
   * **Purpose & Fit**: Simple CLI-based experiment tracker that automatically captures run metadata, hyperparameters, logs, and model output. Integrates with any Python ML code.
   * **Maturity**: Lightweight, community-driven; lacks some advanced registry features but excels at quick experiment capture.
4. **Metaflow (Open-Source)**
   * **Purpose & Fit**: Developed by Netflix, it focuses on pipeline orchestration with experiment tracking. Emphasis on reproducible data science at scale.
   * **Maturity**: Large developer community, stable enterprise features. More pipeline-oriented than pure model registry.
5. **Weights & Biases (Commercial)**
   * **Purpose & Fit**: Cloud-based SaaS for experiment tracking, hyperparam sweeps, data versioning, and deep analytics. Strong TensorFlow/PyTorch integration.
   * **Maturity**: Rapid growth, robust UI, real-time collaboration features. Free tier available for small teams, but enterprise features cost.
6. **Comet (Commercial)**
   * **Purpose & Fit**: Cloud-based experiment tracking and model registry with real-time metric visualisations, code diffs, and collaboration.
   * **Maturity**: Well-established user base, varied pricing tiers. Good developer experience with Python SDK.
7. **Neptune.ai (Commercial)**
   * **Purpose & Fit**: Comprehensive experiment management, model registry, and metadata store. Strong project organisation features for large teams.
   * **Maturity**: Competitive with W&B. Offers on-prem or cloud hosting for compliance.
8. **Azure ML (Commercial)**
   * **Purpose & Fit**: Full-service ML platform by Microsoft. Provides experiment tracking, model registry, and deployment tooling tightly integrated with Azure cloud.
   * **Maturity**: Enterprise-grade, but heavily tied to Azure ecosystems. Potential licensing/cost overhead if you’re not already in Azure stack.

These eight candidates collectively represent the primary categories (pure open-source, hybrid OSS-commercial, and fully managed enterprise) to compare and contrast in **feature sets, cost, integration, and maintainability**.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Tool** | **Core Tracking (Params/**  **Metrics/**  **Artifacts)** | **Model Registry Support** | **UI/UX Quality** | **API/SDK Flexibility** | **Team Collab**  **oration** | **CI/CD Integration** | **Deployment Tracking** | **Cost/Licensing** | **Customisability** |
| **MLflow** | Full suite with automatic logging | Native registry with versioning/staging | Functional UI | Python-centric | Basic | Git integration | MLflow Models deployment | Open-Source | Plugin ecosystem |
| **DVC** | Metrics/params via CLI | Studio registry with lifecycle management | CLI-first + Studio | CLI/Python | Git-based | GitHub Actions compatible | Pipeline-based | Open-Source | Extension system |
| **Guild AI** | Auto-captured metadata | No native registry | Basic dashboards | CLI/R API | Limited | Limited | None | Open-Source | Minimal |
| **Metaflow** | Artifact checkpointing | Pipeline-oriented tracking | AWS Console UI | Python SDK | Team namespaces | AWS Step Functions | Cloud deployment | Open-Source | Checkpoint plugins |
| **W&B** | Real-time metric streaming | Model versioning with stages | Premium dashboard | Python/JS | Advanced sharing | Custom webhooks | Cloud endpoints | Freemium SaaS | Custom reports |
| **Neptune** | Rich metadata tracking | Lightweight registry with tags | Organized UI | Python/API | Project sharing | API triggers | Deployment monitoring | Paid/On-prem | Flexible metadata |
| **Azure ML** | Native Azure integration | Enterprise registry service | Azure Portal UI | Python/.NET | AD integration | Azure DevOps | Azure deployment | Enterprise SaaS | Azure extensions |

**4. Feature Comparison Matrix**

**Key Insights by Tool**

1. **MLflow**   
   **Pros**: Full lifecycle management, strong OSS community, model staging  
   **Cons**: Requires self-hosting for team features
2. **DVC**   
   **Pros**: Git-native data versioning, pipeline management  
   **Cons**: No built-in experiment comparison UI
3. **Guild AI**   
   **Pros**: Zero-config experiment capture  
   **Cons**: No model registry, basic visualization
4. **Metaflow**   
   **Pros**: Netflix-proven at scale, resumeable flows  
   **Cons**: AWS-centric, limited registry
5. **Weights & Biases**   
   **Pros**: Real-time collab, model hyperlinking  
   **Cons**: Cost escalates with usage
6. **Neptune**   
   **Pros**: Metadata flexibility, compliance-ready  
   **Cons**: Registry as secondary feature
7. **Azure ML**   
   **Pros**: End-to-end Azure integration  
   **Cons**: Vendor lock-in, complex pricing

**Shortlist Selection**

MLflow

DVC

**5. Deep Dive: Top 2 Candidates**

Ragul.

**6. Code Implementation Examples**

Ragul.

**7. Recommendation & Justification**

Cannot complete. Pending Ragul.

**References**

MLflow.org, n.d. *Concepts - MLflow*. Available from: <https://mlflow.org/docs/1.23.1/concepts.html> [9 April 2025].

DVC.org, n.d. *Model Registry | Data Version Control*. Available from: <https://dvc.org/doc/use-cases/model-registry> [7 April 2025].

GuildAI.github.io, n.d. *Introduction to Guild AI for R*. Available from: <https://guildai.github.io/guildai-r/articles/guildai.html> [7 April 2025].

Metaflow.org, n.d. *Checkpointing Progress - Metaflow Docs*. Available from: <https://docs.metaflow.org/scaling/checkpointing> [7 April 2025].

CanvasBusinessModel.com, n.d. *How Does Weights & Biases Work?*. Available from: <https://canvasbusinessmodel.com/blogs/how-it-works/how-does-weights-and-biases-work> [7 April 2025].

Neptune.ai, n.d. *ML Model Registry: The Ultimate Guide*. Available from: <https://neptune.ai/blog/ml-model-registry> [5 April 2025].

MicrosoftLearn, n.d. *Register and work with models - Azure Machine Learning*. Available from: <https://learn.microsoft.com/en-us/azure/machine-learning/how-to-manage-models> [5 April 2025].