Distributed ML Model Provenance System in Cloud-Native Environments

- **Project proposer:** Marcela Melara (marcela.melara@intel.com)
- **Open source?** Yes
- **Mentors:** Marcela Melara / Marcin Spoczynski (marcin.spoczynski@intel.com)
- **Additional Advisors:** Anjo Vahldiek-Oberwagner (anjo.lucas.vahldiek-oberwagner@intel.com)

Preferred past experience:

- **Required (for most or all team members):** Linux, Kubernetes basics, Python or Rust
- **Required (at least one team member):** Rust, Go, Containers/Docker
- **Nice to have:** ML pipelines (Kubeflow, MLflow), distributed systems, database administration (PostgreSQL/MongoDB), service meshes (Istio)

Project background:

In real-world ML workflows, multiple stages (data ingestion, preprocessing, training, fine-tuning, evaluation) run as separate services across distributed infrastructure. Each stage produces artifacts that must be tracked and linked to establish a model's provenance. The Atlas CLI project (github.com/IntelLabs/atlas-cli) provides a foundation for ML lifecycle provenance using C2PA manifests, OpenSSF Model Signing (OMS) specification, cryptographic signing, and transparency logs. However, it currently operates as a standalone CLI tool lacking native cloud integration, distributed coordination, and automated pipeline instrumentation.

Project description:

This project aims to extend Atlas into a cloud-native distributed provenance system. We envision the following project steps.

1. Design a demonstration ML lifecycle with multiple stages: (a) data ingestion service that pulls training data, (b) preprocessing service for data transformation, (c) fine-tuning service that takes a base model and produces a tuned model.
2. Implement each stage as a service using Kubeflow Pipelines. Leverage Kubernetes native service discovery and DNS for inter-component communication. Services share artifacts via PVCs or MinIO/S3.
3. Implement a sidecar container pattern for Atlas CLI provenance collection at each stage of the demo ML lifecycle. The sidecar monitors a shared filesystem volume, detects new artifacts (models, datasets), and automatically generates C2PA/OMS manifests using Atlas CLI. The sidecar automatically captures and stores provenance when new outputs appear.
4. Deploy the demo ML lifecycle services, including Atlas CLI sidecars, in a Kubernetes cluster following security best practices (e.g., authentication, TLS communications using a service mesh). Create Helm charts or Kustomize manifests for Atlas components. Configure persistent database storage for manifests and keys. Collect provenance at each service and test coordination between the services.

5. Create APIs for querying provenance across the demo services.

Stretch goals:

1. **Stretch goal 1:** Implement a provenance verification service that can validate the complete lineage of any artifact, checking cryptographic signatures and manifest integrity across all linked assets.
2. **Stretch goal 2:** Write Atlas CLI testing script for automated pipeline configuration, deployment and execution.
3. **Stretch goal 3:** Enable use of confidential computing technologies via frameworks such as CoCo (confidential containers)
4. **Stretch goal 4:** Multi-cloud deployment. Extend the system to work across multiple Kubernetes clusters (e.g., AWS EKS + GCP GKE). Implement cross-cluster provenance synchronization using federated transparency logs.

What team members will learn:

- Kubernetes deployment patterns and security best practices (sidecars, operators, Helm, authentication, service meshes)
- Distributed systems and inter-service communication design and testing
- Database design for graph-based data tracking
- Programming in Rust (extending Atlas CLI) and Go (Kubernetes tooling)
- Cloud-native development and multi-cloud architecture
- ML pipeline orchestration and lifecycle management
- Supply chain security standards and tooling, including C2PA, in-toto, Sigstore and OpenSSF Model Signing (OMS)