AIRFLOW SUMMIT

Scaling ML Infrastructure:

Scaling ML Infrastructure: Lessons from Building Distributed Systems

3.0

Ashok Prakash,
Senior Principal Engineer, Oracle Cloud & Health

New Infrastructure Bottleneck

AIRFLOW SUMMIT

The Core Problem: The Compute Demand

- **The Shift:** We moved from training small CV/NLP models to training and fine-tuning LLMs and large models, requiring vast, contiguous GPU resources (Current Generation GPU Accelerators).
- **The Cost:** GPUs are the most expensive resource in the data center. Idle time means wasted capital.
- **The Time:** In our early days, launching a multi-node GPU experiment took days of manual configuration. This is unacceptable for scaling R&D velocity.
- **My Experience:** I've spent the last few years building the Kubernetes control plane to solve this exact bottleneck for thousands of nodes.

Navigating the AI Scale Challenge

AIRFLOW SUMMIT

The Foundation: Kubernetes and GitOps for Scalable Infrastructure.

The Logic Plane: Why Airflow is the natural orchestrator for MLOps.

Scaling the Pipeline: Technical Deep Dive: Dynamic GPU Allocation and CI/CD.

Airflow for RoI: Orchestrating Observability and Fleet Management.

Key Takeaways and Guidelines.

AIRFLOW SUMMIT

The Foundation: The Kubernetes "Paved Path"

Challenge: Bare-Metal Complexity vs. Developer Velocity

Legacy State: Bare-Metal	The New Goal: Paved Path (Self-Service)
Days to provision a cluster.	**Minutes** to launch an workload.
Manual and configuration.	Self-service (Kubernetes).
Configuration drift is common.	100% Declarative state (GitOps).
Low utilization (≤50%).	Elastic, High utilization (≥80%) via smart scheduling.

Pillar 1: Kubernetes for GPU Abstraction

AIRFLOW SUMMIT

Kubernetes is a great option for control plane compute scaling.

- **Resource Isolation:** Provides guaranteed via Pod limits and requests. Critical for multi-tenant environments.
- **Decoupling Compute:** Abstracts the physical hardware (High-Performance GPU Accelerators) behind a consistent API. The ML team shouldn't care about the OS or drivers.
- **Dynamic Scaling:** Native support for Cluster Autoscaler (CAS) to scale GPU node pools based on pending Pod demand.
- **Advanced Scheduling:** Leveraging specialized GPU Device Plugin/Operator and features like Hardware Partitioning Feature for fractional GPU sharing on inference workloads.

Pillar 2:
Version Control as the Single Source of Truth

AIRFLOW SUMMIT

If it's not in VC, it doesn't exist.

- **IaC (Leading IaC Tool):** Manages the cloud infrastructure layer (VPCs, Network, Load Balancers, Clusters).
- **Config Mgmt (GitOps Controller):** Manages the application and configuration layer within K8s (Operators, CNIs, Logging/Monitoring agents, ML deployments).
- **Eliminating Drift:** Ensures that all cluster changes flow through a codified, reviewed process. Our goal was 100% declarative state, which radically reduced reliability issues.
- **Infra Workflows:** These workflows (create cluster, upgrade node pools, patch network) are too critical for a general-purpose scheduler. They need a system designed for declarative state management.

Pillar 3: Maximizing GPU Return on Investment

AIRFLOW SUMMIT

We must 'sweat the asset', GPUs cannot sit idle.

- **Automated Provisioning:** Built custom pipelines to monitor hardware availability and provision fleets of thousands of nodes across multiple regions automatically.
- **Dynamic Right-Sizing:** Enforcing strict resource requests (70-80% of peak) and using limit ranges to prevent over-reserving, thus improving bin-packing efficiency.
- **Spot/Preemptible Usage:** Orchestrating workloads to use high-risk, low-cost Spot instances whenever possible, using Airflow *upstream* to manage the data staging before the K8s job.
- **Triage and Repair:** Utilizing sophisticated structured logging and automated healing processes to automatically detect and repair unhealthy GPU nodes, minimizing downtime.

Airflow: The MLOps Orchestration Core

Airflow: The Logic Plane for MLOps

AIRFLOW SUMMIT

General-purpose orchestrator is essential for AI pipelines.

- MLOps is a set of heterogeneous tasks: Data, Compute, Model Registry, Serving Layer. No single tool does it all well.
- **Airflow's Role:** It acts as the "glue" and the central nervous system, translating complex, distributed dependencies into simple, readable DAGs.
- **Python Native:** Using Python to define workflows is a massive win for ML teams, who can use the same language for their DAGs and their core logic.

The Three Pillars of Airflow in MLOps

AIRFLOW SUMMIT

Pillar	Description	Airflow Feature	Suggestion
Data Agnosticism	Orchestrates ETL/ELT regardless of source (Cloud Storage, Data Warehouse, etc.)	Providers/Hooks	Great fit
Scalable Compute	Manages the dynamic allocation of isolated, resource-heavy jobs.	KubernetesExecutor	Good Fit with Combination of Kubernetes
Reliability	Ensures pipeline governance, error handling, and automated retries.	SLA, Branching, Callbacks	Great fit for telemetry, resilience

Seamless Integration with the Compute Plane

AIRFLOW SUMMIT

Airflow as the Commander of Kubernetes Resources

- **K8sExecutor:** This is the game-changer for ML. It allows every task to run in its own Kubernetes Pod, isolated and disposable.
- **Task Isolation:** The K8s Pod can request specific requirements, a particular GPU type, specialized PyTorch container image, high memory.
- **Resource Efficiency:** Airflow's scheduler manages the execution *logic* while Kubernetes manages the *compute* resources, leading to maximum utilization of expensive GPUs.

Airflow Orchestrating Data Preparation

Training only starts when data is perfect.

- Data Dependencies: Use Airflow to define the complex sequence: Ingestion → Validation → Feature Engineering → Feature Store Load.
- Data Quality Gates: Integrate data quality checks (DQ tools) as mandatory tasks in the DAG. Use Branching Operators to halt the pipeline if DQ checks fail.
- Example: A daily DAG that cleanses petabytes of time-series data using a Distributed Computing Framework (triggered via a K8sPodOperator) before firing the model training step.

Airflow: MLOps Continuous Training (CT)

AIRFLOW SUMMIT

Automating the Model Lifecycle on New Data

- **Scheduled Retraining:** Airflow ensures your models are kept fresh by running the full pipeline on a defined schedule (e.g., daily, weekly).
- **Event-Driven Triggers:** Leveraging sensors or external calls to trigger a retraining DAG instantly when:
    - Significant new data is ingested.
    - Production metrics detect model drift.
- **Reproducibility:** The DAG acts as the central definition, ensuring that the same set of preprocessing, training parameters, and evaluation steps are executed every time.

Airflow & The Kubernetes Executor: Logic vs. Compute

How to use Airflow for resource-intensive jobs.

- **The Problem with Workers:** Running a 10-hour GPU job directly on an Airflow worker ties up resources and is hard to isolate.
- **The Solution: Kubernetes Executor:** The DAG defines the logic, but each task is executed in a dynamically launched Kubernetes Pod.
- **Benefits:**
  1. **Isolation:** Each task gets its own container image and dependencies.
  2. **Resource Customization:** The K8s Pod can request specific GPU types or quantities.
  3. **Scalability:** K8s handles the scheduling and cluster autoscaling automatically based on Airflow's needs.

Scaling Compute and Data with Airflow

Technical Deep Dive: Dynamic Task Mapping (DTM)

AIRFLOW SUMMIT

Scaling Training and Experimentation within a Single DAG

- **The Need:** We need to run the same training task across of hyperparameter combinations or 50 regional datasets.
- **Airflow DTM:** Allows a task to dynamically generate a list of inputs (e.g., 100 JSON parameter files) that the downstream task will map and execute in parallel.
- **Efficiency:** Instead of writing 100 lines for 100 tasks, you define **one** mapped task, which runs N parallel K8s Pods, each requesting specific GPU resources.
- **Use Case:** Mass parallel hyperparameter tuning using K8sExecutor → faster time-to-best-model.

Technical Deep Dive: Data Sharing via XCom

AIRFLOW SUMMIT

Passing Metadata and Artifacts in a Scaled Pipeline

- **The Challenge:** How does the "Train" task tell the "Evaluate" task where the artifact is stored?
- **Airflow** XCom **(Cross-Communication):** Ideal for passing small metadata objects between tasks, like:
  - S3 URI of the trained model.
  - Model Version ID.
  - Training Metrics Summary.
- **Best Practice:** Do **not** pass large data files. Use XCom to pass *references* (URIs, UUIDs) to data/models stored in dedicated, versioned systems (e.g., Cloud Storage, Model Registry).

Technical Deep Dive: TaskFlow API and Pythonic ML

AIRFLOW SUMMIT

Simplifying Workflow Authoring

- **The Goal:** Make ML engineers feel at home.
- **Airflow** TaskFlow: Use Python functions decorated with `@task` to define tasks, eliminating boilerplate operators.
- **Simplicity:** Native Python type hints and functional composition make DAGs look more like clean Python code, boosting ML developer experience.
- **Integration:** Easily wrap existing PyTorch or TensorFlow or Agentic AI workflow scripts into TaskFlow functions for inclusion in the DAG.

Airflow for Infrastructure Actions: Setup and Teardown

AIRFLOW SUMMIT

Managing Ephemeral Compute Environments

- **The Requirement:** Certain jobs (like multi-node distributed training) require temporary, dedicated infrastructure (e.g., a Ray cluster or a large Spark cluster).
- **Setup/Teardown Tasks:** Airflow allows defining tasks that create (Setup) or destroy (Teardown) external resources.
- **Example Flow:**
    1. **Setup Task:** Use IaC or Cloud Operator to provision a temporary, high-capacity GPU cluster.
    2. **Training Task:** Use the cluster for ML training.
    3. **Teardown Task:** Execute automatically to destroy the cluster, ensuring no compute resources are wasted.

Airflow and Beyond: AI Native Features

AIRFLOW SUMMIT

Looking Forward: New Capabilities for MLOps

- **Sleek & Better Usability:** Modernized interface enhances ML pipeline visibility and debugging
- **Smarter Backfills:** More efficient reprocessing of historical data, critical for retraining models on large, updated datasets.
- **Native** AI/ML **Support:** Increased integration with LLM-adjacent features, streamlining pipelines for Generative AI use cases (e.g., RAG pipelines).

Airflow for Production & Infrastructure RoI

Airflow: The Production CI/CD Gate

AIRFLOW SUMMIT

Ensuring only the best models reach the Paved Path.

- **The Flow:** The is the Continuous Deployment controller, acting after training is complete.
- **Validation Tasks:** Run final evaluation against a live validation set. Calculate SLOs (e.g., f1-score, latency P95).
- **Deployment Gate:** Use Branching Logic to decide the path:
  - **Success:** Model meets SLO→ Trigger Declarative Deployment Tool to update the K8s Serving Deployment.
  - **Failure:** Model fails SLO→ Send alert to team, stop deployment, and potentially rollback to the last known good model.

Best Practice: Modular DAGs for Team Scale

AIRFLOW SUMMIT

Scaling the organization requires separating logic from orchestration.

- **Reusability:** Separate core logic (Python functions/scripts) from the DAG definition.
- **Task Groups:** Use Task Groups to organize complex pipeline stages (ETL, TRAIN, DEPLOY) into collapsible, manageable units within the UI.
- **Collaboration:** Allows data scientists to focus on ML code while ML Engineers focus on DAG structure and environment management.

Best Practice: Security and Credentials Management

AIRFLOW SUMMIT

Airflow as the Secure Intermediary

- **Avoid Hardcoding:** Never store credentials or sensitive configuration in the file.
- **Airflow Connections:** Use the built-in Connections Manager for database/cloud access details.
- **Secrets Backends:** Integrate Airflow with external Secrets Management systems (e.g., Vault or Cloud Secret Stores) for production environments, keeping credentials out of the Airflow database entirely.

The Reliability Imperative: Observability

AIRFLOW SUMMIT

Integrating Pipeline Health into the Infrastructure Stack

- **Logging and Metrics:** Airflow's native logging integrates with external logging solutions, giving us a unified view of pipeline execution alongside GPU host metrics.
- **Airflow's UI as the Dashboard:** The DAG View provides real-time status, logs, and duration tracking for all ML jobs.
- **Automatic Retries:** ML jobs can be flaky. Airflow's built-in retry logic provides resilience against transient failures (e.g., network timeout, brief resource unavailability).

Scaling the Human Factor with Airflow

AIRFLOW SUMMIT

The Self-Service Workflow

- **Developer Experience:** The Paved Path allows ML teams to launch GPU jobs in minutes. Airflow provides the organized, visual workflow to manage those jobs afterward.
- **Standardization:** Airflow enforces a standard structure (DAGs) for all pipeline logic, making it easier for new engineers to onboard and understand existing production flows.
- **Impact:** This standardization supports the massive scale of the engineering organization, turning complex ML operations into a repeatable, auditable flow.

The Paved Path to Production Flow

AIRFLOW SUMMIT

A Full Lifecycle Example

1. **Airflow Task 1 (Prep):** Triggers Job to run ETL using a Spark image. Passes Data URI via XCom.
2. **Airflow Task 2 (Train):** Uses KubernetesExecutor to launch 100 parallel Pods (via DTM). Each Pod requests one GPU.
3. **Airflow Task 3 (Validate):** Retrieves metrics from XCom and runs Branching Operator.
4. **Airflow Task 4 (Deploy):** If metrics pass, triggers Declarative Deployment Tool to deploy the model service to the K8s cluster.

Three Key Technical Guidelines

AIRFLOW SUMMIT

1. KubernetesExecutor**:** Use the KubernetesExecutor exclusively for ML tasks to leverage dynamic GPU allocation and eliminate worker strain.
2. **Maximize** DTM**:** Use Dynamic Task Mapping for all forms of parallel ML work (hyperparameter sweeps, multi-region inference, batch prediction).
3. XCom **for Metadata, Not Data:** Only pass small configuration and artifact URIs between tasks; keep large data in a high-performance external store.

Conclusion: Airflow is the Future of MLOps Scale

AIRFLOW SUMMIT

- Airflow is far more than a simple scheduler, it is the ideal **Logic Orchestrator** for complex, distributed, and heterogeneous ML pipelines.
- By embracing the KubernetesExecutor and features like Dynamic Task Mapping, Airflow directly solves the scaling challenges posed by modern GPU-accelerated AI.
- The path to scaling AI infrastructure is paved by K8s and orchestrated by Airflow.

Questions?

https://www.linkedin.com/in/ashok-prakash/