Streamlining AI Pipeline Across NetApp Storage Platforms

NetApp artificial intelligence solutions

NetApp
September 25, 2026

Table of Contents

Streamlining AI Pipeline Across NetApp Storage Platforms

Streamlining AI Pipeline Across NetApp Storage Platforms (StorageGRID, ONTAP, and E-Series)

Karthikeyan Nagalingam, NetApp

This validated architecture presents a governed, storage-tier-aware AI data pipeline orchestrated by Apache Airflow. The solution ingests raw tabular and text data from NetApp StorageGRID, prepares deterministic datasets and manifests on an ONTAP NAS bucket, and uses NetApp XCP to move prepared data to either NetApp ONTAP S3 or LustreFS according to runtime configuration. It then trains and incrementally fine-tunes scikit-learn models, generates predictions, checkpoints successful training outputs, and archives models, metrics, manifests, and prediction evidence to StorageGRID. The architecture provides reproducible execution, configurable storage placement, failure guardrails, checkpoint-based recovery, and auditable lineage across the AI lifecycle. It is intended for architects, data engineers, ML engineers, and infrastructure teams who need a repeatable deployment and validation pattern for AI pipelines across NetApp storage tiers.

Table of Contents

1. Executive Summary

Karthikeyan Nagalingam, NetApp

1.1 Purpose of this Document

This NetApp Validated Architecture (NVA) documents a production-grade AI data pipeline that ingests raw data, prepares it for machine learning, moves it at high speed between storage tiers using NetApp XCP, trains and incrementally fine-tunes models, generates predictions, and archives results for governance and reproducibility. It is intended to guide architects and infrastructure teams through design rationale, deployment, configuration, and operations.

1.2 Audience

Solution architects, storage/infrastructure engineers, data platform engineers, ML engineers, and IT decision-makers evaluating NetApp storage and data mobility technology for AI/ML workloads.

1.3 Business Challenge Overview

AI teams need data that lands in object storage to be reliably transformed and then delivered to whichever compute-storage tier best serves training — elastic S3 for general workloads, or high-performance parallel filesystems (LustreFS) for I/O-intensive training. Without a governed data mobility and orchestration layer, teams resort to fragile, single-purpose scripts lacking retries, checkpointing, storage flexibility, and audit trails.

Enterprise AI programs are stalling — not because of model or compute limitations, but because legacy storage was not designed for AI. Moving massive datasets, keeping GPUs fed during training, managing checkpoint overhead, and controlling storage costs consume more effort than the AI work itself. Single-tier architectures force a performance-versus-cost trade-off that compounds as models and datasets grow.

1.4 NetApp Solution Summary

The solution orchestrates ingestion, preparation, data mobility, training, fine-tuning, inference, and archival as an Apache Airflow DAG (Directed Acyclic Graph). StorageGRID anchors the raw-data and long-term archival object tiers. Data preparation writes run-stamped datasets and manifests to an ONTAP NAS bucket, exposed by NFS as the XCP source. NetApp XCP then moves the prepared data to a runtime-selectable training destination — NetApp ONTAP S3 or LustreFS — with no code changes required to switch tiers.

> In this design, the ONTAP NAS bucket is used primarily from an XCP supportability and validation perspective, because it provides a consistent NFS source for the mobility workflow. In real production environments, the same active data can also be served directly over high-performance RDMA-capable paths, so the ONTAP NAS layer should be viewed as an operationally convenient and supportable staging pattern rather than the only runtime access method.

This pipeline is a concrete implementation of the broader **NetApp AI Storage Architecture** — a purpose-built, three-tier storage framework that aligns performance and cost to each AI workload phase:

- **E-Series (LustreFS):** Ultra-high-throughput parallel storage for GPU-intensive model training and checkpointing.
- **ONTAP (AFF/AFX):** High-performance active storage for training, fine-tuning, inference, vector/RAG workloads, and selected data preparation use cases, with FAS and NAS bucket support.
- **StorageGRID:** Scalable S3-compatible object storage for data ingestion, governance, and long-term retention.

Data movement across tiers is fully automated via Apache Airflow and NetApp XCP, eliminating manual operations from the critical path entirely.

The NetApp storage architecture helps enterprises support distributed data preparation, model training, fine-tuning, inferencing, and lifecycle governance without forcing all data into a single centralized lake. By reducing unnecessary data movement, it is a natural fit for hybrid AI workflows.

1.5 Why NetApp

Dimension	Conventional Approach	NetApp AI Storage Architecture
GPU Productivity	Storage bottlenecks idle expensive compute	GPUDirect Storage over RDMA helps keep GPU clusters fully utilized
Data Mobility	Manual scripts delay pipelines	Automated inter-tier movement via Airflow + XCP
Cost Control	All data on high-cost flash	FabricPool auto-demotes cold data to low-cost object storage
Multi-Tenancy	Shared namespaces risk data leakage	Dedicated ONTAP SVMs enforce strict tenant isolation
Workload Breadth	Separate stacks for training vs. inference	Single unified architecture spanning the full AI lifecycle

1.6 The Business Case

GPU compute is typically the largest capital expense in an AI infrastructure budget. Every hour a cluster sits idle waiting for data is direct, measurable financial loss. NetApp ensures the right data is on the right tier at the right time — automatically — delivering higher AI throughput, lower TCO, and an architecture that scales from POC to enterprise production without redesign.

1.7 Key Benefits at a Glance

Benefit	Description
Unified orchestration	Single DAG spans ingest → archive
Storage-tier flexibility	S3 or LustreFS selected per run via configuration
Reduced retraining cost	Incremental fine-tuning via `partial_fit`
Faster recovery	Checkpoint-based training resume
Full lineage	Manifests, effective-config logs, archival records
No storage lock-in	Multi-protocol: NFS, S3, Lustre
GPU utilization	Tiered storage keeps compute fed, avoiding idle-GPU cost
Cost-optimized retention	FabricPool auto-tiers cold data to object storage

2. Solution Overview, Business Value and Customer Benefits

Karthikeyan Nagalingam, NetApp

2.1 Solution Description

The pipeline is implemented as the Airflow DAG `example_ai_pipeline_sklearn`, composed of the tasks `ingestion`, `data_prep`, `xcp_preflight_source`, `xcp_copy_prep_to_training`, `model_training`, `fine_tuning`, `inferencing`, `checkpoint_model_training`, `route_manual_archive_stage`, and

`manual_archive`.

2.2 Use Case and Problem Statement

Customers must train and periodically refresh tabular regression and text classification models using raw data sourced from StorageGRID. The pipeline prepares that data in an ONTAP NAS bucket, then uses NetApp XCP to deliver it to either ONTAP S3 or LustreFS for training, based on the per-run destination selection.

2.3 Target Customer Profile and Industry Applicability

Financial services (risk/fraud models), manufacturing (predictive maintenance), healthcare (classification/triage), retail (demand forecasting) — any enterprise requiring governed, repeatable ML training on NetApp infrastructure.

2.4 Solution Scope and Boundaries

In scope: ingestion gating, data preparation (Python/Spark), XCP data mobility (S3/LustreFS), scikit-learn training and incremental fine-tuning, inference, checkpointing, manual archival. **Out of scope:** real-time model serving APIs, streaming ingestion, GPU-based deep learning frameworks.

2.5 Assumptions and Prerequisites

- StorageGRID S3-compatible endpoint reachable from Airflow for raw-data access and archival.
- ONTAP NAS bucket reachable from Airflow for prepared-data writes and exposed through NFS to the XCP host.
- NetApp ONTAP S3-compatible endpoint reachable from Airflow and XCP when ONTAP S3 is selected as the training destination.
- NetApp XCP installed and reachable via SSH (Airflow connection `ssh_default`).
- LustreFS client mounted on the XCP host when LustreFS is selected.
- Airflow 2.x with `boto3`, `scikit-learn`; optional PySpark + Delta/Iceberg packages.

2.6 Business Drivers

Reduce time-to-model, lower infrastructure cost via incremental training, eliminate one-off scripting risk.

2.7 Value Proposition Summary

StorageGRID, ONTAP NAS, ONTAP S3, LustreFS, and XCP combined with Airflow orchestration deliver a storage-tier-aware, observable, resilient AI pipeline.

2.8 Stakeholder Benefits

Stakeholder	Benefit
Data Engineering	Declarative, parameter-driven prep; automatic file discovery; Spark/Delta/Iceberg support
Data Science/ML	Incremental fine-tuning; consistent artifact source across stages; reproducible splits

Stakeholder	Benefit
IT Operations	Runtime storage selection; guard/effective-config logs; checkpoint-based recovery
Compliance/Governance	Manifests and archival records provide auditable lineage; configurable retention stage

2.9 TCO Considerations

Incremental fine-tuning and checkpoint resume reduce recurring compute cost; storage-tier flexibility avoids over-provisioning high-performance storage for all workloads.

2.10 ROI Considerations

Faster model refresh cycles, reduced manual maintenance, reduced recovery time per failure event.

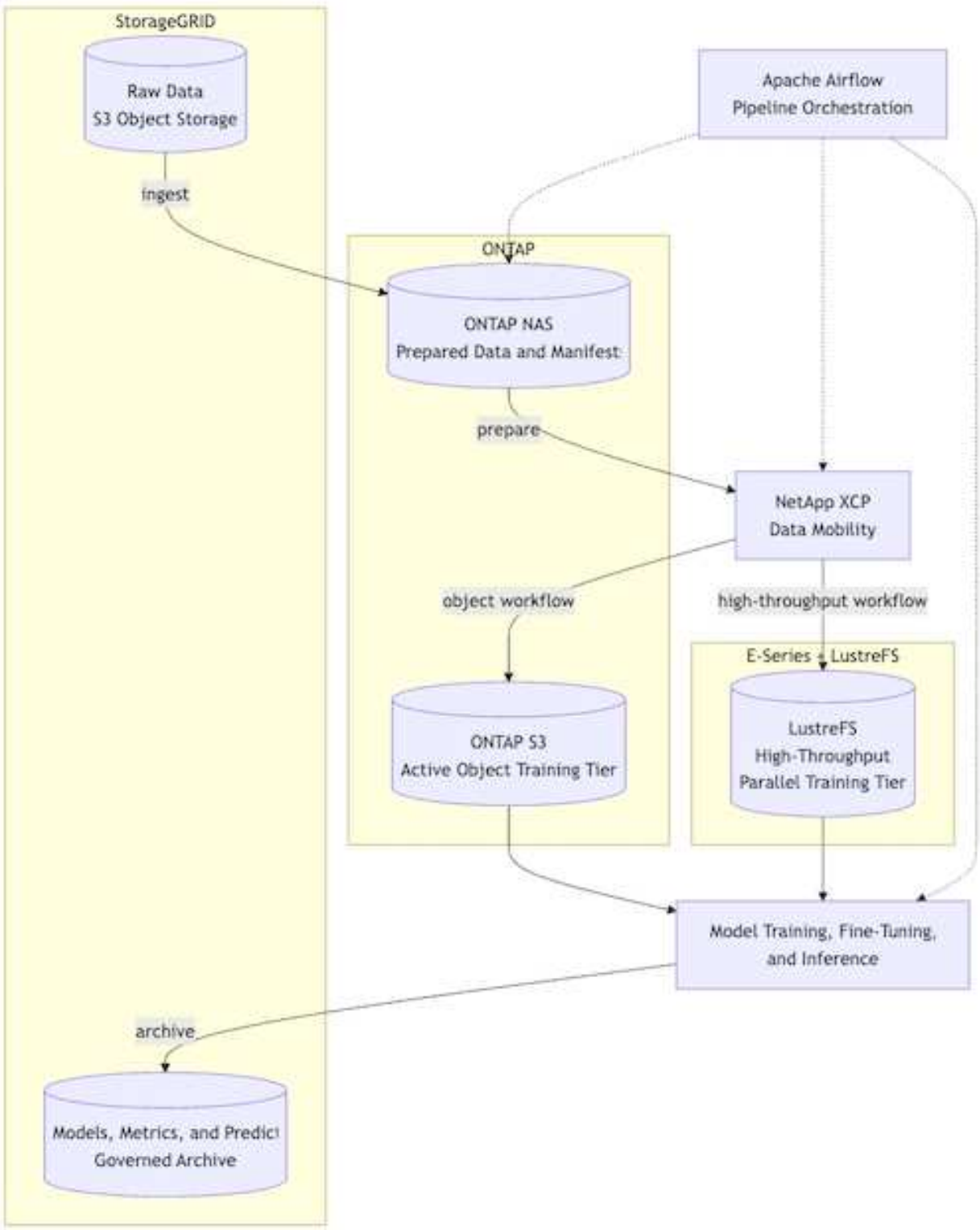
3. Architecture Overview

Karthikeyan Nagalingam, NetApp

This section presents the logical workflow, component interactions, physical deployment, and data lineage for the validated AI pipeline. Together, the diagrams show how Apache Airflow orchestrates data movement from StorageGRID through ONTAP NAS, optionally uses NetApp XCP to place data on ONTAP S3 or LustreFS for model execution, and returns governed results to StorageGRID archival storage.

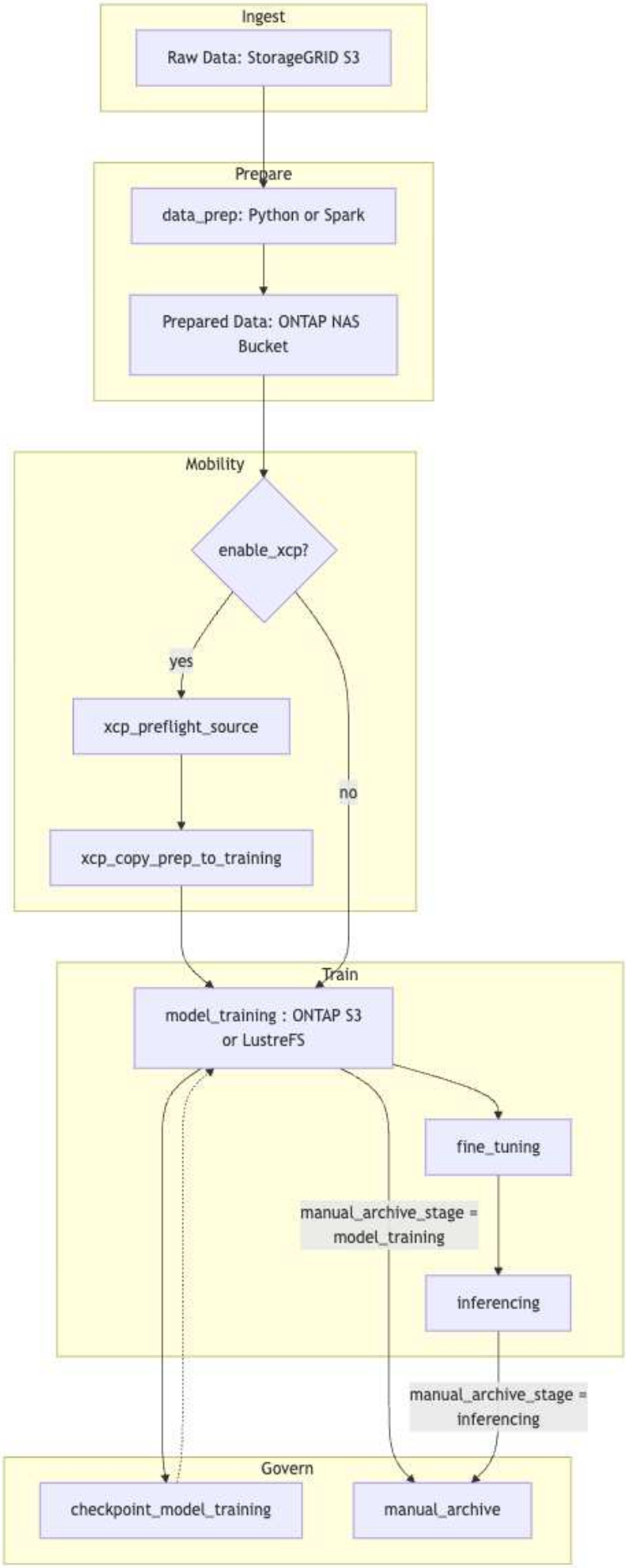
3.1 High-Level Three-Tier AI Storage Architecture

The NetApp AI storage architecture places data on the storage tier that best matches each lifecycle phase. StorageGRID is used for scalable, cost-optimized raw-data ingestion and governed archival. ONTAP NAS and ONTAP S3 are used as the active preparation, data mobility staging, and object-based training workflows. E-Series with LustreFS is used when high-throughput, parallel training I/O is required. Apache Airflow orchestrates the lifecycle, while NetApp XCP moves prepared data from the ONTAP NAS tier to the selected active training tier.



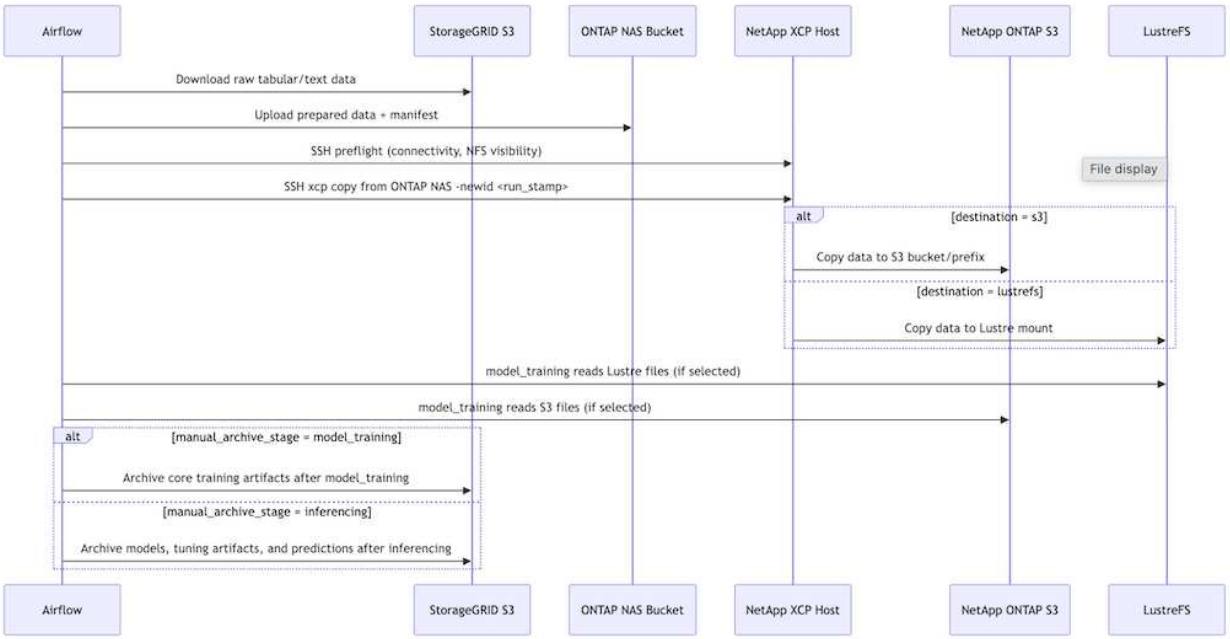
3.2 High-Level Solution Architecture

This workflow diagram shows the Airflow-controlled lifecycle from raw StorageGRID data through preparation, optional XCP mobility, model execution, checkpointing, and archival. `enable_xcp` determines whether prepared data is copied from the ONTAP NAS bucket to ONTAP S3 or LustreFS before training. The archive branches are independent: `manual_archive_stage=model_training` archives the baseline immediately after training, while `manual_archive_stage=inferencing` archives the complete outcome after predictions are generated.


Ingest
Raw Data: StorageGRID S3
Prepare
data_prep: Python or Spark
Prepared Data: ONTAP NAS Bucket
Mobility
enable_xcp?
yes
no
xcp_preflight_source
xcp_copy_prep_to_training
Train
model_training : ONTAP S3 or LustreFS
fine_tuning
manual_archive_stage = model_training
inferencing
manual_archive_stage = inferencing
Govern
checkpoint_model_training
manual_archive

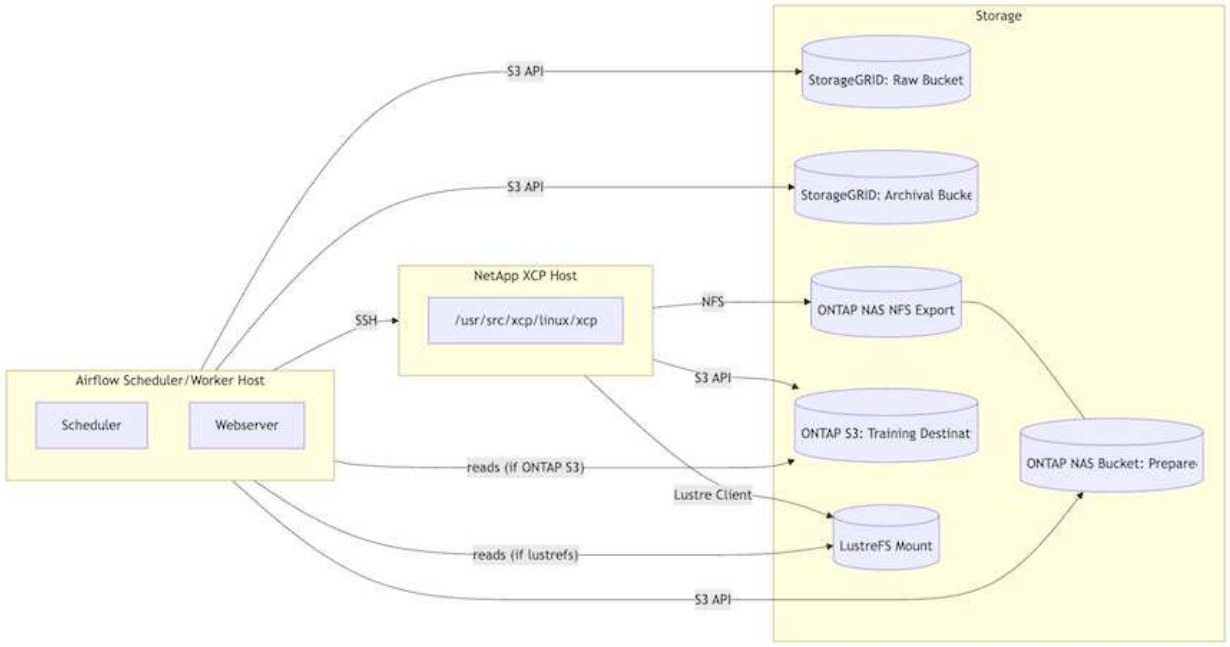
3.3 Component Interaction Diagram

This sequence diagram identifies which platform component performs each exchange. Airflow reads raw objects from StorageGRID and writes prepared data to the ONTAP NAS bucket; it then invokes XCP over SSH. XCP reads the ONTAP NAS NFS export and writes to the configured training tier. The final conditional block shows that StorageGRID archival content depends on the selected manual archive stage.



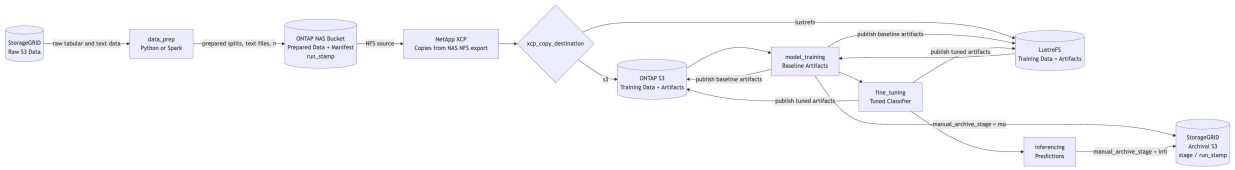
3.4 Physical / Deployment Architecture

This deployment diagram shows the required hosts, storage endpoints, and network interfaces. The Airflow host uses S3 APIs for StorageGRID, the ONTAP NAS bucket, and the selected ONTAP S3 destination, and uses SSH to control the XCP host. The XCP host accesses the prepared-data source through NFS and writes to either ONTAP S3 or the LustreFS mount; when LustreFS is selected, the Airflow host must also mount it to read training data and materialize artifacts.



3.5 Data Flow Overview

This compact flow summarizes data and artifact provenance. Each run receives a unique `run_stamp`, which keeps prepared data, XCP output, model artifacts, predictions, and archived objects traceable to the same execution. XCP destination selection governs the training-data and model-artifact tier, while `manual_archive_stage` governs whether StorageGRID receives the baseline training package or the full inference package.



4. NetApp, Third-Party and Open Source Technology Components

Karthikeyan Nagalingam, NetApp

Component	Role in Solution
NetApp StorageGRID	S3-compatible raw-data ingestion and archival object storage tiers
NetApp ONTAP NAS bucket / NFS	Prepared-data tier; NFS source export validated by XCP preflight before data movement
NetApp ONTAP S3	Optional XCP training-data and model-artifact destination
NetApp XCP	High-throughput data mobility engine; copies to S3 or LustreFS
NetApp FabricPool	Optional tiering for ONTAP NAS prepared data and archive-adjacent data to optimize cost
LustreFS (NetApp-integrated)	High-performance parallel filesystem for training tier
NetApp Console (optional)	Extended monitoring/observability for ONTAP and XCP resources
ONTAP Snapshot/Backup	Point-in-time protection for prepared data in the ONTAP NAS tier, independent of pipeline

4.1 NetApp XCP — Core Value

XCP is the mobility layer that decouples data location from compute location. A single configuration switch (`xcp_copy_destination`) determines whether prepared data and model artifacts land in NetApp ONTAP S3 or a LustreFS high-performance tier, with `model_training`, `fine_tuning`, and `inferencing` automatically reading from and writing to the same destination.

4.2 Third-Party and Open Source Technology Components

Component	Purpose
Apache Airflow	DAG orchestration, retries, XCom task communication, scheduling
Apache Spark	Optional distributed transformation engine for `data_prep`
Delta Lake / Apache Iceberg	Optional lakehouse table formats for prepared tabular data
scikit-learn	`LinearRegression` (tabular) and `SGDClassifier` (text, incremental)
Airbyte / Apache NiFi	Optional ingestion connectors ahead of `data_prep`
boto3	AWS S3-compatible SDK used for StorageGRID, ONTAP NAS bucket, and ONTAP S3 interactions

5. Solution Design and Storage Architecture Detail

Karthikeyan Nagalingam, NetApp

5.1 Design Principles

Configuration over code changes; explicit stage boundaries with typed contracts; consistent artifact routing; fail-fast validation with actionable errors; storage-tier neutrality.

5.2 Stage Design Summary

Stage	Design Summary
Ingestion	`ingestion_tool` selects `s3_direct`, `none`, `airbyte`, or `nifi`; returns normalized metadata
Data Preparation	Discovers/accepts raw StorageGRID tabular keys; normalizes Airbyte/plain CSV; produces deterministic train/val/infer splits; writes a run-stamped prepared-data prefix and manifest to the ONTAP NAS bucket
Data Mobility (XCP)	Copies prepared data from the ONTAP NAS NFS export; `xcp_copy_destination` drives the XCP target and training read path
Model Training	Trains locally or from XCP destination (S3/LustreFS); multi-format (CSV/Parquet/JSON) discovery
Fine-Tuning	Materializes base model from XCP destination; `partial_fit` with expanded class set; republishes tuned model
Inference	Materializes tuned artifacts from same XCP destination; configurable split/text scope
Checkpoint/Resume	`write_stage_checkpoint` persists completion record; `_small_resume_guard` validates/reuses on subsequent runs

Stage	Design Summary
Archival	`model_training` archives core trained models immediately; `inferencing` waits for predictions and archives the full result set; optional XCP-input inclusion and raw-folder cleanup

5.3 Configuration-Driven Philosophy

All non-secret storage, engine, and behavioral selections are `dag_run.conf` parameters — switching between S3 and LustreFS, or Python and Spark, requires no code modification, while Airflow-managed secret storage resolves the required credentials.

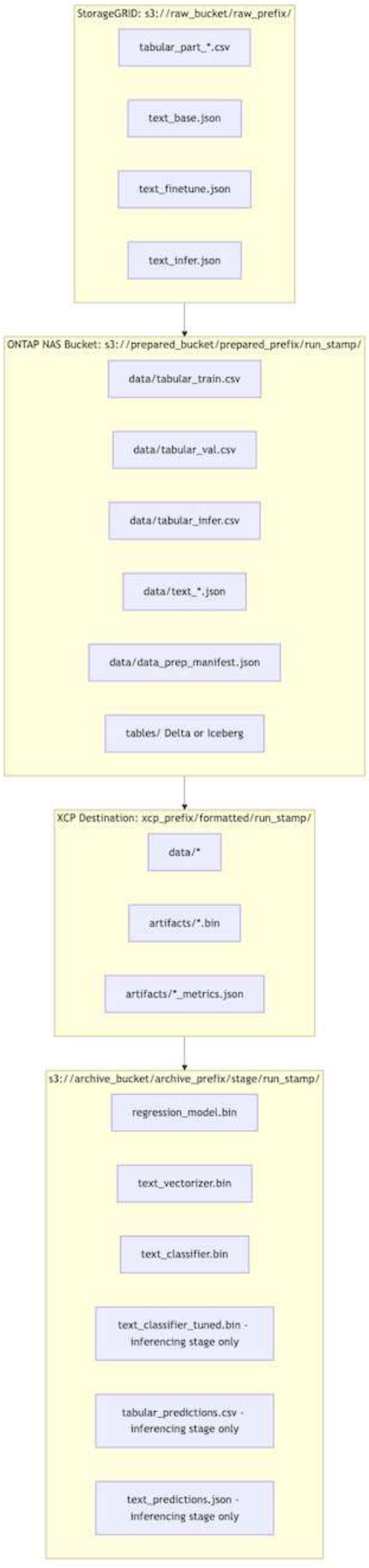
5.4 Storage Architecture Detail

The storage architecture separates the AI lifecycle into purpose-built tiers: StorageGRID holds raw and archival objects, the ONTAP NAS bucket holds prepared and run-stamped datasets, and ONTAP S3 or LustreFS provides the selected active training tier. This separation keeps source data, prepared data, model artifacts, and archived evidence independently manageable while preserving lineage through the shared `run_stamp`.

5.4.1 Storage Layout Diagrams

Tier	Logical Storage Path	Contents / Artifacts Stored
StorageGRID Raw Tier	`s3://raw_bucket/raw_prefix/`	Tabular CSV parts (`tabular_part_*.csv`), text inputs (`text_base.json`, `text_finetune.json`, `text_infer.json`)
ONTAP NAS Prepared Tier	`s3://prepared_bucket/prepared_prefix/run_stamp/`	Formatted splits (`data/tabular_*.csv`), text files, `data_prep_manifest.json`, Lakehouse tables (`tables/` Delta/Iceberg)
Active Training Tier (XCP Dest)	`<xcp_prefix>/formatted/run_stamp/`	Copied data splits, trained model binaries (`artifacts/*.bin`), evaluation metrics (`artifacts/*_metrics.json`)
StorageGRID Archival Tier	`s3://archive_bucket/archive_prefix/stage/run_stamp/`	Stage-scoped baseline or full model artifacts, prediction evidence (`tabular_predictions.csv`, `text_predictions.json`)
Inter-Tier Lineage & Flow	Raw → Prepared → Active Tier → Archival	End-to-end data movement and artifact lineage linked across all storage tiers via the shared `run_stamp`

This diagram maps the logical bucket and prefix layout for a single pipeline run across all storage tiers:


StorageGRID: s3://raw_bucket/raw_prefix/
tabular_part_*.csv
text_base.json
text_finetune.json
text_infer.json
ONTAP NAS Bucket: s3://prepared_bucket/prepared_prefix/run_stamp/
data/tabular_train.csv
data/tabular_val.csv
data/tabular_infer.csv
data/text_*.json
data/data_prep_manifest.json
tables/ Delta or Iceberg
XCP Destination: xcp_prefix/formatted/run_stamp/
data/*
artifacts/*.bin
artifacts/*_metrics.json
s3://archive_bucket/archive_prefix/stage/run_stamp/
regression_model.bin
text_vectorizer.bin
text_classifier.bin
text_classifier_tuned.bin - inferencing stage only
tabular_predictions.csv - inferencing stage only
text_predictions.json - inferencing stage only

5.4.2 Storage Sizing Guidance

Size each tier independently according to its role and retention policy. The ONTAP NAS and selected XCP destination must accommodate concurrent active runs, whereas StorageGRID capacity is driven primarily by raw-data and archival retention. Include temporary working capacity and growth headroom when planning peak pipeline concurrency.

Tier	Sizing Basis
StorageGRID raw bucket	Ingestion volume × retention window
ONTAP NAS prepared-data bucket	Prepared dataset size × number of retained runs
XCP destination (S3 or Lustre)	Peak concurrent training dataset size + model artifact overhead
Archival bucket	Retention policy × archive stage selection (model_training = smaller; inferencing = larger)

5.4.3 Storage Performance Considerations

The XCP destination is selected per run with `xcp_copy_destination`, allowing storage performance to match the workload instead of forcing every workload onto one tier. Select LustreFS for high-row-count datasets, many concurrent readers, or I/O-intensive training. Select ONTAP S3 for cost-conscious, elastic-access workloads whose performance requirements do not demand a parallel filesystem. In either case, XCP keeps data and model artifacts aligned with the selected training tier.

6. Data Mobility with NetApp XCP

Karthikeyan Nagalingam, NetApp

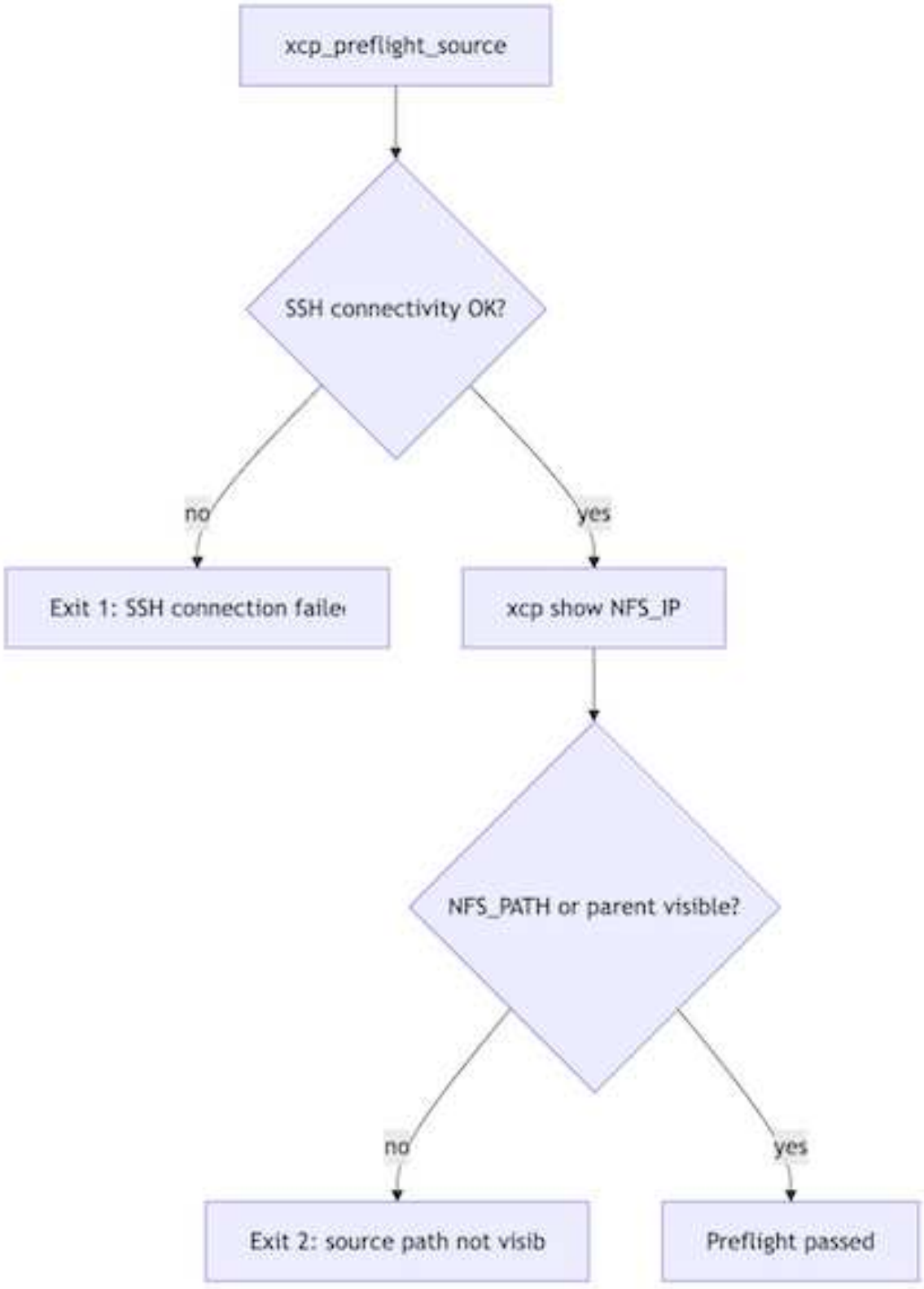
NetApp XCP provides the controlled transfer path between the ONTAP NAS prepared-data tier and the active training tier. Airflow performs the preflight and invokes the copy operation through the XCP host; `xcp_copy_destination` selects ONTAP S3 or LustreFS for each run. The same selected destination is used by model training, fine-tuning, and inference to keep data and artifacts on a consistent tier.

6.1 Why NetApp XCP

XCP is a parallel, multi-threaded copy engine that supports the NFS source, ONTAP S3, and LustreFS destinations used by this architecture. Its `-newid <run_stamp>` option associates each transfer with the pipeline execution, supporting traceability across prepared data, training input, and model-artifact locations.

6.2 XCP Preflight Validation Flow

The preflight task runs before an XCP copy. It verifies that Airflow can reach the XCP host through SSH and that the configured ONTAP NAS NFS source path is visible to XCP. The explicit exit conditions stop the pipeline before a copy begins when connectivity or source visibility is incorrect.



6.3 XCP Copy to S3 — Command Template

Use this path when `xcp_copy_destination=s3`. XCP reads the prepared, run-stamped data from the ONTAP NAS NFS export and writes it to the configured ONTAP S3 bucket and prefix. The named XCP S3 profile supplies destination credentials on the XCP host; model stages then discover both data and artifacts

through `xcp_s3_training_prefix`.

```
xcp copy -newid <run_stamp> \
  -s3.profile <xcp_s3_profile> \
  -s3.endpoint <xcp_s3_endpoint> \
  <xcp_nfs_source_ip>:<xcp_nfs_source_path> \
  s3://<xcp_s3_bucket>/<xcp_s3_dest_prefix>
```

Example: copy the prepared ONTAP NAS dataset to the ONTAP S3 training bucket

```
/usr/src/xcp/linux/xcp copy -newid 20260901_120000 \
    -s3.profile ontaps3 \
    -s3.endpoint http://10.63.150.161 \
    10.63.150.159:/prepnasbucket/example_ai_pipeline \
    s3://trainingbucket/example_ai_pipeline
```

This example uses `20260901_120000` as the XCP job identity. The training task discovers the copied files under the configured `xcp_s3_training_prefix`, such as `example_ai_pipeline/formatted/20260901_120000/data/`.

6.4 XCP Copy to LustreFS — Command Template

Use this path when `xcp_copy_destination=lustrefs`. XCP copies prepared data from the mounted ONTAP NAS source to the mounted LustreFS destination. Both mounts must be available on the XCP host, and LustreFS must also be mounted on the Airflow worker that performs model training, fine-tuning, and inference.

```
findmnt -T <xcp_lustrefs_source_path> -o FSTYPE,TARGET -n | grep -E
'^nfs[0-9]* '  # source mount type
findmnt -T <xcp_lustrefs_dest_path> -o FSTYPE,TARGET -n | grep -E '^lustre
'   # destination mount type
xcp copy -newid <run_stamp> \
  file://<xcp_lustrefs_source_path> \
  file://<xcp_lustrefs_dest_path>
```

Example: copy the prepared ONTAP NAS dataset to the LustreFS training mount

```
findmnt -T /prepnasbucket/example_ai_pipeline -o FSTYPE,TARGET -n | grep
-E '^nfs[0-9]* '
findmnt -T /mnt/lustre/client -o FSTYPE,TARGET -n | grep -E '^lustre '

/usr/src/xcp/linux/xcp copy -newid 20260901_120000 \
    file:///prepnasbucket/example_ai_pipeline \
```

```
    file:///mnt/lustre/client
```

The two `findmnt` commands verify that the source path is mounted through NFS and the destination path is mounted through LustreFS on the XCP host before the copy starts. With `xcp_lustrefs_training_subpath=example_ai_pipeline`, training reads the run-stamped data under `/mnt/lustre/client/example_ai_pipeline/formatted/20260901_120000/data/`.

6.5 XCP Destination Decision Flow

This decision diagram shows how one runtime parameter determines the copy target and the training read path. Select `s3` for the ONTAP S3 object destination or `lustrefs` for the parallel filesystem destination. Downstream stages use the matching source automatically, avoiding changes to model code when storage placement changes.

6.6 XCP Failure Handling and Guardrails

These guardrails make configuration and infrastructure failures observable at the earliest practical point. Preflight failures prevent an invalid transfer, filesystem-type checks validate the ONTAP NAS and LustreFS mounts, and S3 credential validation applies only when ONTAP S3 is the selected XCP destination.

Condition	Guard Behavior
Unsupported `xcp_copy_destination` value	Pipeline rejects the value before any data movement
LustreFS mount not present	Filesystem-type validation fails; copy halted before data movement
SSH unreachable	Preflight exits with code 1 before any copy
NFS path not visible	Preflight exits with code 2 before any copy
Missing XCP S3 credentials (S3 mode only)	`model_training`/artifact sync raises explicit `ValueError`

7. Deployment Architecture

Karthikeyan Nagalingam, NetApp

This section defines the infrastructure placement, software dependencies, and network connectivity required to operate the pipeline. The deployment separates Airflow orchestration from the XCP data-mobility host while connecting both components to StorageGRID, ONTAP NAS, and the selected ONTAP S3 or LustreFS training destination.

7.1 Reference Infrastructure Requirements

The following components establish the minimum functional deployment footprint. Size the Airflow host for the selected preparation engine and concurrent task load; place the XCP host where it can access the ONTAP NAS NFS export and the selected destination without unnecessary network hops. LustreFS mode requires compatible mounts on both the XCP host and the Airflow worker.

Component	Requirement
Airflow host	CPU/memory sized to Spark or Python transform load
XCP host	Network access to NFS/Lustre and NetApp ONTAP S3 endpoints
LustreFS clients	Mounted on XCP host and Airflow host when selected as destination

7.1.1 Reference Validation Environment

The following environment was used as a representative single-node validation profile. It is a starting point for functional and performance evaluation, not a production sizing recommendation; customers must size compute, network, storage capacity, and protection according to their dataset volume, concurrency, retention, recovery, and service-level requirements.

Layer	Reference Hardware / Software
Compute and network	One server with 256 GB RAM, 64 CPU cores, and 10 GbE connectivity
ONTAP active storage	One NetApp A800 system with 48 x 1.8 TB SSDs
Object storage	One NetApp StorageGRID SG5864 appliance
High-throughput training storage	One NetApp E2812 system with LustreFS
Platform software	Kubernetes, Apache Airflow, Apache Airbyte, single-node Apache Spark, NetApp XCP, and LustreFS

7.2 Software Version Matrix

This matrix identifies the runtime software used by the validated pipeline. Keep the Airflow and Python environments compatible with the deployed providers and packages. Spark, Delta Lake, and Iceberg are optional and are required only when `data_prep` uses the Spark transformation path or a corresponding table format.

Software	Version/Notes
Apache Airflow	2.x
Python	3.11
boto3	Latest stable
scikit-learn	Latest stable
PySpark (optional)	Compatible with Delta 3.2.0 / Iceberg 1.5.2 runtime packages

Software	Version/Notes
NetApp XCP	Installed on remote host, e.g. `/usr/src/xcp/linux/xcp`

7.3 Network Requirements

These network flows must be permitted by routing, firewall, and name-resolution policies. HTTPS is recommended for every S3-compatible endpoint in production; use the configured custom port where an endpoint does not use the default HTTPS port. Confirm Lustre client connectivity according to the deployed LustreFS implementation.

Flow	Protocol/Port
Airflow → ONTAP S3	HTTPS/HTTP (443/80 or custom)
Airflow → XCP host	SSH (22)
XCP host → NFS source	NFS (2049)
XCP host → LustreFS	Lustre client ports
XCP host → ONTAP S3 (S3 mode)	HTTPS/HTTP

7.4 Deployment Notes

Section / Topic	Guidance / Operational Practice
SSH Connection Setup	Configure Airflow connection `ssh_default` to target the XCP host over port 22 with restricted SSH key permissions (`600`).
Service Lifecycle Management	Use `tools/airflow_ctl.sh` to manage the local Airflow scheduler and webserver lifecycle during deployment and maintenance.
Credential & Secret Storage	Store credentials, API tokens, and access keys in Airflow Connections, Variables, or a secrets backend rather than inline in `dag_run.conf`.
Infrastructure & Task Observability	Monitor scheduler heartbeat and task execution independently so orchestration availability issues are distinguished from DAG failures.

7.5 Installation and Prerequisites

This subsection defines the baseline installation steps required to build, configure, and run the NetApp AI validated pipeline in a fresh environment. It is intended for operators, architects, and engineers setting up the Airflow workspace before running any DAGs.

7.5.1 Prerequisites

Before installing the solution, confirm that the target host meets the following minimum requirements:

- Linux operating system, preferably Ubuntu 22.04/24.04 or RHEL 8+.

- Python 3.11 with `pip`, `venv`, and build tooling available.
- Git installed on the host for source retrieval and version management.
- Apache Airflow 2.x deployed in a dedicated Python virtual environment.
- SSH access from the Airflow host to the NetApp XCP host.
- Network connectivity from the Airflow host to StorageGRID, ONTAP NAS, and the selected ONTAP S3 or LustreFS destination.
- S3-compatible endpoint access for raw data, prepared-data staging, and archival storage.
- Optional: Java runtime and Spark 3.x if the `spark` preparation path is used.
- Optional: Delta Lake and Iceberg runtime packages when `table_format=delta` or `table_format=iceberg` is selected.
- Optional: Lustre client mount when `xcp_copy_destination=lustrefs` is used.

7.5.2 Software Package Acquisition and Repository Access

To obtain the software package, automation DAGs, deployment scripts, and validation artifacts for this solution, contact the NetApp AI & Data Mobility engineering team at ng-data-mobility-in-ai-pipeline@netapp.com.

Once access is granted, download or clone the project source from GitHub into the target working directory:

```
# Example GitHub clone
mkdir -p /opt/netapp-ai
cd /opt/netapp-ai

git clone https://github.com/<your-org>/<your-repo>.git
cd <your-repo>

git checkout main
ls -la
```

If you already cloned the repository under `/opt/netapp-ai/<your-repo>`, continue from that working directory instead of downloading another copy.

7.5.3 Create the Python Environment

Create a dedicated virtual environment and install the base dependencies for Airflow and the pipeline.

```
export REPO_ROOT=/opt/netapp-ai/<your-repo>
cd "$REPO_ROOT"
python3 -m venv "$REPO_ROOT/.venv"
source "$REPO_ROOT/.venv/bin/activate"

python -m pip install --upgrade pip setuptools wheel
pip install "apache-airflow>=2.8,<3.0" boto3 scikit-learn
```

For Spark-based preparation and Lakehouse table formats, install the optional packages as needed:

```
pip install pyspark==3.5.*
pip install delta-spark==3.2.0
pip install apache-iceberg==1.5.2
```

Use the exact package versions supported by your Spark runtime and cluster topology. Do not mix incompatible Spark, Delta, and Iceberg combinations.

7.5.4 Configure Airflow and the Repository

From the cloned repository root, initialize the Airflow metadata database and verify that the DAG files are visible to Airflow.

```
export AIRFLOW_HOME="$REPO_ROOT/.airflow"
export AIRFLOW__CORE__DAGS_FOLDER="$REPO_ROOT"
export AIRFLOW__CORE__LOAD_EXAMPLES=False

airflow db init
airflow dags list | grep -E
"example_ai_pipeline|example_ai_pipeline_sklearn"
```

The workspace includes the Airflow configuration file and DAGs needed for the validated pipeline. If the repository contains a helper script for local lifecycle management, use it to start the scheduler and webserver instead of invoking Airflow manually.

```
# Start the Airflow scheduler and webserver with the deployment's service
manager.
# or, if using the standard commands:
# airflow scheduler
# airflow webserver
```

7.5.5 Required Connectivity and Secret Configuration

Before running the pipeline, confirm that the following resources are reachable from the Airflow host:

- StorageGRID raw-data S3 endpoint.
- ONTAP NAS prepared-data bucket endpoint.
- ONTAP S3 target endpoint when `xcp_copy_destination=s3`.
- LustreFS mount when `xcp_copy_destination=lustrefs`.
- XCP host over SSH using `ssh_default` or the configured SSH connection.

Store credentials in Airflow Connections, Variables, or a secrets backend rather than embedding them in shell history or DAG code. Set the required access keys, endpoints, and profile mappings before triggering a pipeline run.

7.5.6 Verify the Installation

A successful installation is confirmed when the Airflow scheduler and DAGs are running and the example pipeline can be listed and triggered without configuration errors.

```
airflow dags list
airflow tasks list example_ai_pipeline_sklearn
```

If the installation is correct, the DAG should be present in Airflow and ready for a `CONF_JSON`-driven run using the trigger script in the repository root.

```
./trigger_and_wait_ai_pipeline_sklearn.sh
```

At this point, the environment is ready for the configuration examples in Section 8 and the deployment scenarios described in the operational guide.

8. Configuration Reference

Karthikeyan Nagalingam, NetApp

The configuration reference describes the runtime parameters that control data ingestion, data preparation, XCP mobility, training, checkpointing, and archival. The ONTAP NAS bucket is used primarily as the prepared-data and XCP staging layer for supportability and portability. In real-time production use, the same prepared data can also be consumed through direct RDMA-capable access paths when that better matches the workload and latency requirements.

All runtime behavior is supplied through `dag_run.conf`, allowing the same DAG to serve different source systems, transformation engines, storage destinations, recovery policies, and archive scopes. Configure only the parameter groups required for the selected workflow, keep credentials in Airflow Connections or a secrets backend, and use the examples as non-production templates rather than as credential values.

8.1 Ingestion Parameters

Use these settings to select and configure the upstream ingestion mechanism. `s3_direct` uses the StorageGRID raw-data objects configured in the data-preparation group; Airbyte and NiFi settings are needed only when those respective ingestion tools are selected.

Parameter	Default	Required	Description	Example
`ingestion_tool`	`s3_direct`	No	Selects ingestion mode	`"s3_direct"`
`airbyte_api_url`	None	Airbyte only	Airbyte API endpoint	`"http://airbyte:8000"`
`airbyte_connection_id`	None	Airbyte only	Airbyte connection ID	`"connection-123"`

Parameter	Default	Required	Description	Example
airbyte_api_token	None	Optional	Bearer token	"<TOKEN>"
airbyte_timeout_secs	30	No	Airbyte timeout	60
nifi_api_url	None	NiFi only	NiFi API endpoint	"http://nifi:8080"
nifi_process_group_id	None	NiFi only	NiFi process group	"abc123"
nifi_timeout_secs	30	No	NiFi timeout	60

8.2 Data Preparation Parameters

These settings control raw-data access, prepared-data output, input discovery, and transformation behavior. The s3_raw_* parameters refer to StorageGRID, while the code-compatible s3_formatted_* parameters identify the ONTAP NAS bucket where prepared, run-stamped data and manifests are written.

Raw StorageGRID / prepared-data ONTAP NAS bucket:

Configure separate endpoints and credentials when StorageGRID and the ONTAP NAS bucket use different S3-compatible services or access policies. The generic fallback keys apply only when a more specific raw or prepared-data setting is absent.

Parameter	Default	Description	Example
s3_raw_bucket	ai-raw-data	StorageGRID raw input bucket	"bucket1"
s3_raw_prefix	example_ai_pipeline/raw	StorageGRID raw-data prefix	"example_ai_pipeline/raw"
s3_raw_endpoint_url	fallback chain	StorageGRID S3 endpoint	"http://10.63.150.62:10444"
s3_raw_access_key_id	fallback chain	Raw access key	"<KEY>"
s3_raw_secret_access_key	fallback chain	Raw secret key	"<SECRET>"
s3_raw_region	aws_region	Raw region	"us-east-1"
s3_formatted_bucket	ai-formatted-data	ONTAP NAS prepared-data output bucket	"prepnasbucket"
s3_formatted_prefix	example_ai_pipeline/formatted	ONTAP NAS prepared-data prefix	"example_ai_pipeline/formatted"
s3_formatted_endpoint_url	fallback chain	ONTAP NAS bucket S3 endpoint	"http://10.63.150.159"

Parameter	Default	Description	Example
`s3_formatted_access_key_id`	fallback chain	ONTAP NAS bucket access key	`"<KEY>"`
`s3_formatted_secret_access_key`	fallback chain	ONTAP NAS bucket secret key	`"<SECRET>"`

General fallback: `s3_endpoint_url`, `s3_access_key_id`, `s3_secret_access_key`, `s3_session_token`, `s3_verify`, `aws_region`, `aws_access_key_id`, `aws_secret_access_key`, `aws_session_token`.

Input selection:

Use explicit object-key settings to process known CSV inputs; otherwise, the task discovers eligible tabular objects under the configured StorageGRID raw prefix.

Parameter	Default	Description	Example
`s3_tabular_object_keys`	Auto-discovery	Explicit list of CSV keys	`["raw/demo/a.csv"]`
`s3_tabular_object_key`	None	Single explicit CSV key	`"raw/demo/a.csv"`

Transformation:

Select the Python path for lightweight local preparation or Spark for distributed preparation. Sampling and the seed apply reproducibly to the generated training, validation, and inference splits.

Parameter	Default	Description	Example
`transformation_engine`	`python`	`python` or `spark`	`"spark"`
`spark_required`	`false`	Fail instead of fallback	`true`
`table_format`	`none`	`none`/`delta`/`iceberg`	`"delta"`
`table_format_required`	`false`	Fail if format unavailable	`true`
`sample_count`	all rows	Row limit (≤0 = all)	`14400`
`seed`	`7`	Reproducible shuffle seed	`11`

Spark:

These settings apply only to `transformation_engine=spark`. They control Spark process placement, resource allocation, time limits, and optional Delta Lake or Iceberg runtime dependencies.

Parameter	Default	Description	Example
`spark_submit_bin`	`spark-submit`	Spark binary	`"/opt/spark/bin/spark-submit"`

Parameter	Default	Description	Example
`spark_master`	`local[*]`	Spark master	`"local[*]"`
`spark_driver_memory`	`4g`	Driver memory	`"8g"`
`spark_executor_memory`	`4g`	Executor memory	`"8g"`
`spark_timeout_secs`	`900/7200`	Timeout (0 = disabled)	`0`
`spark_delta_packages`	Delta 3.2.0	Delta runtime package	`"io.delta:delta-spark_2.12:3.2.0"`
`spark_iceberg_packages`	Iceberg 1.5.2	Iceberg runtime package	`"org.apache.iceberg:iceberg-spark-runtime-3.5_2.12:1.5.2"`
`spark_iceberg_catalog`	`local`	Iceberg catalog name	`"local"`

8.3 XCP and Training Destination Parameters

Use this group when `enable_xcp=true` to validate the ONTAP NAS NFS source, invoke the XCP host, and choose the active training tier. `xcp_copy_destination` is the controlling switch: it selects either the ONTAP S3-specific or LustreFS-specific settings below, and downstream model stages use the same selected tier.

Parameter	Default	Required	Description	Example
`enable_xcp`	`false`	Yes for XCP	Enables XCP branch	`true`
`xcp_copy_destination`	`s3`	No	`s3` or `lustrefs`	`"lustrefs"`
`xcp_nfs_source_ip`	`10.63.150.159`	XCP preflight	NFS server IP	`"10.63.150.159"`
`xcp_nfs_source_path`	`/prepnasbucket/example_ai_pipeline`	XCP preflight	NFS export path	`"/prepnasbucket/example_ai_pipeline"`
`ssh_user`	`root`	No	XCP host SSH user	`"root"`
`ssh_host`	`10.63.150.178`	No	XCP host address	`"10.63.150.178"`

S3 destination (required when `xcp_copy_destination=s3`):

Provide these settings only for the ONTAP S3 training path. The profile or direct credentials allow XCP and the model artifact synchronization logic to access the selected ONTAP S3 bucket and prefix.

Parameter	Default	Description	Example
xcp_s3_endpoint	default	XCP S3 endpoint	"http://10.63.150.161"
xcp_s3_bucket	None	XCP destination bucket	"trainingbucket"
xcp_s3_dest_prefix	example_ai_pipeline	Destination prefix	"example_ai_pipeline"
xcp_s3_training_prefix	xcp_s3_dest_prefix	Training discovery prefix	"example_ai_pipeline"
xcp_s3_profile	ontaps3	Named credential profile	"ontaps3"
xcp_s3_profiles	None	Profile map	{"ontaps3": {…}}
xcp_s3_access_key_id	None	Direct key fallback	"<KEY>"
xcp_s3_secret_access_key	None	Direct secret fallback	"<SECRET>"
xcp_s3_region	aws_region	XCP S3 region	"us-east-1"

LustreFS destination (required when `xcp_copy_destination=lustrefs`):

Provide these settings only for the LustreFS training path. The source and destination must be mounted and accessible on the XCP host; the destination must also be accessible to the Airflow worker that runs the model stages.

Parameter	Default	Description	Example
xcp_lustrefs_source_path	/prepnasbucket/example_ai_pipeline	XCP copy source	"/prepnasbucket/example_ai_pipeline"
xcp_lustrefs_dest_path	/mnt/lustre/client	LustreFS mount destination	"/mnt/lustre/client"
xcp_lustrefs_training_subpath	xcp_s3_dest_prefix	Training subdir under mount	"example_ai_pipeline"
xcp_lustrefs_newid	data_prep run stamp	XCP job identifier	"20260901_120000"

Text-source behavior:

These options control how the three text datasets are found after XCP mobility. Explicit keys override discovery; local fallback can be disabled to require that text inputs come from the selected XCP destination.

Parameter	Default	Description	Example
xcp_text_allow_local_fallback	true	Allow local text fallback	false
xcp_text_base_key	Auto	Explicit base-text location	"…/text_base.json"

Parameter	Default	Description	Example
xcp_text_finetune_key	Auto	Explicit fine-tune-text location	"…/text_finetune.json"
xcp_text_infer_key	Auto	Explicit inference-text location	"…/text_infer.json"

8.4 Model Training Parameters

These settings select the local or XCP-delivered source, limit training volume, and preserve reproducible data ordering. When XCP is enabled, training reads the selected ONTAP S3 or LustreFS tier and publishes the baseline artifacts back to that same destination.

Parameter	Default	Description	Example
enable_xcp	false	Selects XCP or local source	true
xcp_copy_destination	s3	Selects source tier	"lustrefs"
sample_count	all rows	Training sample limit	14400
seed	7	Reproducible shuffle	11

Produces: regression_model.bin, text_vectorizer.bin, text_classifier.bin, train_metrics.json.

8.5 Fine-Tuning Parameters

Fine-tuning materializes the baseline text artifacts from the selected XCP destination, applies incremental learning to the fine-tuning dataset, and republishes the tuned classifier and its metrics. Keep its XCP settings consistent with model training to maintain artifact provenance.

Parameter	Default	Description	Example
enable_xcp	false	Enables artifact materialize/publish	true
xcp_copy_destination	s3	Selects artifact source/destination	"lustrefs"

Consumes: text_vectorizer.bin, text_classifier.bin, text_finetune.json. Produces: text_classifier_tuned.bin, fine_tune_metrics.json.

8.6 Inference Parameters

These flags control the scope of scoring after the tuned model is materialized from the selected training tier. Enable either option when validation or business requirements call for outputs beyond the default inference split and inference text input.

Parameter	Default	Description	Example
infer_all_splits	false	Score train/val/infer splits	true
infer_all_text	false	Score base/finetune/infer text	true

Consumes: regression_model.bin, text_vectorizer.bin, text_classifier_tuned.bin.
Produces: tabular_predictions.csv, text_predictions.json.

8.7 Checkpoint Parameters

Checkpointing is scoped to model_training. These settings determine whether the training completion record is created, where it is stored, how upload failures are handled, and whether a later run verifies or reuses a valid completed training result.

Parameter	Default	Description	Example
checkpoint_enabled	true	Enables checkpoint file creation	true
checkpoint_store	local	local or formatted_s3	"formatted_s3"
checkpoint_upload_fail_mode	warn	warn or fail	"fail"
training_checkpoint_reuse_mode	off	off/verify_only/resume_if_exists	"resume_if_exists"
checkpoint_reuse_mode	off	Backward-compatible alias	"verify_only"

8.8 Manual Archive Parameters

Use this group to enable StorageGRID archival, identify the destination bucket and credentials, and choose the archival point. The selected manual_archive_stage controls both the archive timing and whether the archive contains only baseline training artifacts or the complete inference result set.

Parameter	Default	Required	Description	Example
manual_archive_enabled	false	No	Enables archival	true
manual_archive_stage	inferencing	No	Archive point and artifact set: model_training for core training artifacts, or inferencing for the full result set	"model_training"
manual_archive_bucket	archivalbucket	Yes when enabled	Archive bucket	"archivalbucket"

Parameter	Default	Required	Description	Example
`manual_archive_prefix`	`ai-models`	No	Archive prefix	`"ai-models"`
`manual_archive_endpoint`	default	Custom only	Archive S3 endpoint	`"http://10.63.150.62:10444"`
`manual_archive_profile`	`sgdlocal`	Recommended	Credential profile	`"sgdlocal"`
`manual_archive_profiles`	None	Optional	Profile map	`{"sgdlocal": {…}}`
`manual_archive_include_xcp_inputs`	`false`	No	Include XCP inputs in archive	`true`
`manual_archive_cleanup_raw`	`false`	No	Remove local raw folder post-upload	`true`

Aliases: `fabricpool_archive_*` for all `manual_archive_*` parameters.

8.8.1 Manual Archive Stage Decision

The `route_manual_archive_stage` branch task reads `manual_archive_stage` once per DAG run. The selected value determines both when archival runs and which artifacts are uploaded to the StorageGRID archival bucket. Archive objects are stored under `s3://<manual_archive_bucket>/<manual_archive_prefix>/<stage>/<run_stamp>/`.

`manual_archive_stage` value	Archive timing	Artifacts archived	Recommended use
`model_training`	Immediately after `model_training`; does not wait for fine-tuning or inference	`regression_model.bin`, `text_vectorizer.bin`, `text_classifier.bin`, `train_metrics.json`	Preserve a reproducible baseline model, minimize archive volume, or retain a checkpoint before later stages
`inferencing` (default)	After `fine_tuning` and `inferencing` complete	All `model_training` artifacts plus `text_classifier_tuned.bin`, `fine_tune_metrics.json`, `tabular_predictions.csv`, and `text_predictions.json`	Retain the complete business outcome and prediction evidence for a model run

For either selection, set `manual_archive_include_xcp_inputs=true` to add locally materialized XCP training inputs when present. Set `manual_archive_enabled=false` to skip the archival task without changing the training and inference workflow.

8.9 Complete Sample Configuration

This catalog provides starting configurations for the principal deployment and test paths. Every example assumes raw data is in StorageGRID and prepared data is written to the ONTAP NAS bucket. Replace bucket names, endpoint addresses, filesystem paths, and profile names with values from the target environment. Do not place production access keys or secrets in `CONF_JSON`; configure them through Airflow Connections, Variables, or a secrets backend.

8.9.1 XCP to LustreFS with Spark and Delta Lake

Use this full-throughput example for distributed preparation and high-performance LustreFS model I/O. It processes all eligible tabular files under `s3_raw_prefix`, writes Delta tables during preparation, persists the training checkpoint to the ONTAP NAS prepared-data bucket, and archives the complete inference result set.

```
{
  "ingestion_tool": "s3_direct",
  "s3_raw_bucket": "bucket1",
  "s3_raw_prefix": "example_ai_pipeline/raw",
  "s3_formatted_bucket": "prepnasbucket",
  "s3_formatted_prefix": "example_ai_pipeline/formatted",

  "enable_xcp": true,
  "xcp_copy_destination": "lustrefs",
  "xcp_nfs_source_ip": "10.63.150.159",
  "xcp_nfs_source_path": "/prepnasbucket/example_ai_pipeline",
  "xcp_lustrefs_source_path": "/prepnasbucket/example_ai_pipeline",
  "xcp_lustrefs_dest_path": "/mnt/lustre/client",
  "xcp_lustrefs_training_subpath": "example_ai_pipeline",
  "xcp_text_allow_local_fallback": false,

  "transformation_engine": "spark",
  "spark_required": true,
  "table_format": "delta",
  "table_format_required": true,
  "spark_driver_memory": "8g",
  "spark_executor_memory": "8g",
  "spark_timeout_secs": 0,
  "sample_count": 14400,
  "seed": 11,

  "infer_all_splits": true,
  "infer_all_text": true,

  "checkpoint_enabled": true,
  "checkpoint_store": "formatted_s3",
  "training_checkpoint_reuse_mode": "off",

  "manual_archive_enabled": true,
```

```
  "manual_archive_stage": "inferencing",
  "manual_archive_bucket": "archivalbucket",
  "manual_archive_prefix": "ai-models",
  "manual_archive_profile": "sgdlocal",
  "manual_archive_cleanup_raw": true
}
```

8.9.2 No XCP with Python Preparation

Use this baseline configuration for a lightweight functional run. Preparation uses the local Python engine and downstream stages use the pipeline's local prepared-data and artifact paths; XCP, Spark, table formats, checkpoint reuse, and archival are disabled.

```
{
    "ingestion_tool": "s3_direct",
    "s3_raw_bucket": "bucket1",
    "s3_raw_prefix": "example_ai_pipeline/raw",
    "s3_formatted_bucket": "prepnasbucket",
    "s3_formatted_prefix": "example_ai_pipeline/formatted",
    "enable_xcp": false,
    "transformation_engine": "python",
    "table_format": "none",
    "sample_count": 200,
    "seed": 11,
    "checkpoint_enabled": false,
    "manual_archive_enabled": false
}
```

8.9.3 Python Preparation with Two Explicit Input Files

Use this focused test when validating the pipeline against two known StorageGRID CSV objects. `s3_tabular_object_keys` disables automatic tabular-file discovery; the text inputs still use their expected locations under the raw prefix.

```
{
    "ingestion_tool": "s3_direct",
    "s3_raw_bucket": "bucket1",
    "s3_raw_prefix": "example_ai_pipeline/raw",
    "s3_tabular_object_keys": [
        "example_ai_pipeline/raw/tabular_part_01.csv",
        "example_ai_pipeline/raw/tabular_part_02.csv"
    ],
    "s3_formatted_bucket": "prepnasbucket",
    "s3_formatted_prefix": "example_ai_pipeline/formatted",
    "enable_xcp": false,
```

```
    "transformation_engine": "python",
    "table_format": "none",
    "sample_count": 200,
    "seed": 11,
    "manual_archive_enabled": false
}
```

8.9.4 Python Preparation with Automatic All-File Discovery

Omit `s3_tabular_object_keys` and `s3_tabular_object_key` to process every eligible tabular object under the StorageGRID raw prefix. This example is suitable for an all-files functional or scale test using the Python preparation engine.

```
{
    "s3_raw_bucket": "bucket1",
    "s3_raw_prefix": "example_ai_pipeline/raw",
    "s3_formatted_bucket": "prepnasbucket",
    "s3_formatted_prefix": "example_ai_pipeline/formatted",
    "enable_xcp": false,
    "transformation_engine": "python",
    "table_format": "none",
    "sample_count": 0,
    "seed": 11,
    "manual_archive_enabled": false
}
```

`sample_count: 0` means that data preparation uses all available rows. Set a positive value for a bounded test run.

8.9.5 XCP to ONTAP S3 with Python Preparation

Use this configuration when prepared data must be copied from the ONTAP NAS NFS export to an ONTAP S3 training bucket. Model training, fine-tuning, and inference materialize and publish artifacts through that same ONTAP S3 destination.

```
{
    "s3_raw_bucket": "bucket1",
    "s3_raw_prefix": "example_ai_pipeline/raw",
    "s3_formatted_bucket": "prepnasbucket",
    "s3_formatted_prefix": "example_ai_pipeline/formatted",
    "enable_xcp": true,
    "xcp_copy_destination": "s3",
    "xcp_nfs_source_ip": "10.63.150.159",
    "xcp_nfs_source_path": "/prepnasbucket/example_ai_pipeline",
    "xcp_s3_endpoint": "https://ontap-s3.example.com",
    "xcp_s3_bucket": "trainingbucket",
```

```
    "xcp_s3_dest_prefix": "example_ai_pipeline",
    "xcp_s3_training_prefix": "example_ai_pipeline",
    "xcp_s3_profile": "ontaps3",
    "transformation_engine": "python",
    "table_format": "none",
    "sample_count": 14400,
    "seed": 11,
    "checkpoint_enabled": true,
    "checkpoint_store": "formatted_s3",
    "manual_archive_enabled": true,
    "manual_archive_stage": "model_training",
    "manual_archive_bucket": "archivalbucket",
    "manual_archive_profile": "storagegrid-archive"
}
```

8.9.6 XCP to ONTAP S3 with Spark and Iceberg

Use this configuration for distributed preparation with Iceberg table output and ONTAP S3 as the active training destination. Set `spark_required` and `table_format_required` to `true` when a run must fail rather than fall back if Spark or Iceberg is unavailable.

```
{
    "s3_raw_bucket": "bucket1",
    "s3_raw_prefix": "example_ai_pipeline/raw",
    "s3_formatted_bucket": "prepnasbucket",
    "s3_formatted_prefix": "example_ai_pipeline/formatted",
    "enable_xcp": true,
    "xcp_copy_destination": "s3",
    "xcp_nfs_source_ip": "10.63.150.159",
    "xcp_nfs_source_path": "/prepnasbucket/example_ai_pipeline",
    "xcp_s3_bucket": "trainingbucket",
    "xcp_s3_dest_prefix": "example_ai_pipeline",
    "xcp_s3_profile": "ontaps3",
    "transformation_engine": "spark",
    "spark_required": true,
    "table_format": "iceberg",
    "table_format_required": true,
    "spark_driver_memory": "8g",
    "spark_executor_memory": "8g",
    "spark_timeout_secs": 0,
    "sample_count": 0,
    "seed": 11,
    "manual_archive_enabled": true,
    "manual_archive_stage": "inferencing",
    "manual_archive_bucket": "archivalbucket",
    "manual_archive_profile": "storagegrid-archive"
```

```
}
```

8.9.7 Scenario Selection Summary

Scenario	XCP destination	Preparation engine	Table format	Input scope	Archive stage
8.9.1	LustreFS	Spark	Delta	All discovered files	`inferencing`
8.9.2	None	Python	None	All discovered files	Disabled
8.9.3	None	Python	None	Two explicit files	Disabled
8.9.4	None	Python	None	All discovered files	Disabled
8.9.5	ONTAP S3	Python	None	All discovered files	`model_training`
8.9.6	ONTAP S3	Spark	Iceberg	All discovered files	`inferencing`

8.9.8 Execution Walkthroughs

Run the examples from the Airflow workspace. The trigger script generates a unique `manual__<UTC timestamp>` run ID, waits for completion, and returns a nonzero exit code for a failed or timed-out run. The following full configurations use the same `CONF_JSON="$(python3 - <<'PY' …)"` pattern as the operational examples. Configure the referenced Airflow-managed connections, XCP profiles, and archive profiles in your secrets backend before running them, and keep `CONF_JSON` limited to non-secret bucket, endpoint, path, and profile identifiers.

The valid table format name is `iceberg`. Do not use `icerberg`, which is not a supported value.

All files, Spark, Iceberg, XCP to ONTAP S3, checkpoint to prepared-data S3, archive after inference:

```
CONF_JSON="$(python3 - <<'PY'
import json

print(json.dumps({
    "ingestion_tool": "s3_direct",
    "enable_xcp": True,
    "xcp_copy_destination": "s3",
    "xcp_text_allow_local_fallback": True,

    "s3_raw_bucket": "bucket1",
    "s3_raw_prefix": "example_ai_pipeline/raw",
    "s3_raw_endpoint_url": "https://storagegrid.example.com",
    "s3_raw_region": "us-east-1",

    "s3_formatted_bucket": "prepnasbucket",
```

```
    "s3_formatted_prefix": "example_ai_pipeline/formatted",
    "s3_formatted_endpoint_url": "https://ontap-nas.example.com",
    "s3_formatted_region": "us-east-1",

    "xcp_nfs_source_ip": "10.63.150.159",
    "xcp_nfs_source_path": "/prepnasbucket/example_ai_pipeline",
    "xcp_s3_endpoint": "https://ontap-s3.example.com",
    "xcp_s3_bucket": "trainingbucket",
    "xcp_s3_dest_prefix": "example_ai_pipeline",
    "xcp_s3_training_prefix": "example_ai_pipeline",
    "xcp_s3_profile": "ontaps3",

    "transformation_engine": "spark",
    "spark_required": True,
    "table_format": "iceberg",
    "table_format_required": True,
    "sample_count": 0,
    "seed": 11,
    "spark_driver_memory": "8g",
    "spark_executor_memory": "8g",
    "spark_timeout_secs": 0,
    "infer_all_splits": True,
    "infer_all_text": True,

    "checkpoint_enabled": True,
    "checkpoint_store": "formatted_s3",
    "manual_archive_enabled": True,
    "manual_archive_stage": "inferencing",
    "manual_archive_bucket": "archivalbucket",
    "manual_archive_prefix": "ai-models",
    "manual_archive_profile": "sgdlocal",
    "manual_archive_endpoint": "https://storagegrid.example.com",
    "manual_archive_cleanup_raw": True,
    "manual_archive_include_xcp_inputs": True,
}, separators=(",", ":")))
PY
)"
CONF_JSON="$CONF_JSON" AIRFLOW_BIN=./.venv/bin/airflow INGESTION_TOOL
=s3_direct \
  ./trigger_and_wait_ai_pipeline_sklearn.sh
```

All files, Spark, Iceberg, XCP to LustreFS, checkpoint to prepared-data S3, archive after inference:

```
CONF_JSON="$(python3 - <<'PY'
import json
```

```
print(json.dumps({
    "ingestion_tool": "s3_direct",
    "enable_xcp": True,
    "xcp_copy_destination": "lustrefs",
    "ssh_user": "root",
    "ssh_host": "10.63.150.178",
    "xcp_text_allow_local_fallback": True,

    "s3_raw_bucket": "bucket1",
    "s3_raw_prefix": "example_ai_pipeline/raw",
    "s3_raw_endpoint_url": "https://storagegrid.example.com",
    "s3_raw_region": "us-east-1",

    "s3_formatted_bucket": "prepnasbucket",
    "s3_formatted_prefix": "example_ai_pipeline/formatted",
    "s3_formatted_endpoint_url": "https://ontap-nas.example.com",
    "s3_formatted_region": "us-east-1",

    "xcp_nfs_source_ip": "10.63.150.159",
    "xcp_nfs_source_path": "/prepnasbucket/example_ai_pipeline",
    "xcp_lustrefs_source_path": "/prepnasbucket/example_ai_pipeline",
    "xcp_lustrefs_dest_path": "/mnt/lustre/client",
    "xcp_lustrefs_training_subpath": "example_ai_pipeline",

    "transformation_engine": "spark",
    "spark_required": True,
    "table_format": "iceberg",
    "table_format_required": True,
    "sample_count": 0,
    "seed": 11,
    "spark_driver_memory": "8g",
    "spark_executor_memory": "8g",
    "spark_timeout_secs": 0,
    "infer_all_splits": True,
    "infer_all_text": True,

    "checkpoint_enabled": True,
    "checkpoint_store": "formatted_s3",
    "manual_archive_enabled": True,
    "manual_archive_stage": "inferencing",
    "manual_archive_bucket": "archivalbucket",
    "manual_archive_prefix": "ai-models",
    "manual_archive_profile": "sgdlocal",
    "manual_archive_endpoint": "https://storagegrid.example.com",
    "manual_archive_cleanup_raw": True,
```

```
    "manual_archive_include_xcp_inputs": True,
}, separators=(",", ":")))
PY
)"
AIRFLOW_BIN=./.venv/bin/airflow INGESTION_TOOL=s3_direct \
  ./trigger_and_wait_ai_pipeline_sklearn.sh
```

For ONTAP S3 mode, pass only the non-secret `xcp_s3_profile` name in `CONF_JSON` and define the corresponding credentials on the XCP host or through Airflow-managed secret storage before the run. For LustreFS mode, the `xcp_s3_profiles` map is intentionally omitted because the training route does not resolve ONTAP S3 credentials. In both examples, define the StorageGRID raw-data, ONTAP NAS prepared-data, and archive credentials through Airflow-managed secret storage rather than inline JSON. Omitting `s3_tabular_object_keys` and setting `sample_count` to `0` requests automatic discovery of all eligible tabular files and use of all available rows.

Python, no XCP, two-file test:

```
CONF_JSON='{"s3_raw_bucket":"bucket1","s3_raw_prefix":"example_ai_pipeline
/raw","s3_tabular_object_keys":["example_ai_pipeline/raw/tabular_part_01.c
sv","example_ai_pipeline/raw/tabular_part_02.csv"],"s3_formatted_bucket":"
prepnasbucket","s3_formatted_prefix":"example_ai_pipeline/formatted","enab
le_xcp":false,"transformation_engine":"python","table_format":"none","samp
le_count":200,"manual_archive_enabled":false}' \
./trigger_and_wait_ai_pipeline_sklearn.sh
```

Spark, Delta Lake, XCP to LustreFS:

```
CONF_JSON='{"enable_xcp":true,"xcp_copy_destination":"lustrefs","xcp_nfs_s
ource_ip":"10.63.150.159","xcp_nfs_source_path":"/prepnasbucket/example_ai
_pipeline","xcp_lustrefs_source_path":"/prepnasbucket/example_ai_pipeline"
,"xcp_lustrefs_dest_path":"/mnt/lustre/client","xcp_lustrefs_training_subp
ath":"example_ai_pipeline","transformation_engine":"spark","spark_required
":true,"table_format":"delta","table_format_required":true,"sample_count":
14400,"manual_archive_enabled":true,"manual_archive_stage":"inferencing"}'
\
./trigger_and_wait_ai_pipeline_sklearn.sh
```

Spark, Iceberg, XCP to ONTAP S3:

```
CONF_JSON='{"enable_xcp":true,"xcp_copy_destination":"s3","xcp_nfs_source_
ip":"10.63.150.159","xcp_nfs_source_path":"/prepnasbucket/example_ai_pipel
ine","xcp_s3_bucket":"trainingbucket","xcp_s3_dest_prefix":"example_ai_pip
eline","xcp_s3_profile":"ontaps3","transformation_engine":"spark","spark_r
equired":true,"table_format":"iceberg","table_format_required":true,"sampl
e_count":0,"manual_archive_enabled":true,"manual_archive_stage":"inferenci
```

```
ng"}' \
./trigger_and_wait_ai_pipeline_sklearn.sh
```

Before an XCP run, verify the configured ONTAP NAS NFS path and destination accessibility from the XCP host. After a successful run, review task logs for `[data_prep] effective_config`, `[xcp_copy] effective_config`, `[model_training] effective_config`, and `[manual_archive]` to confirm the selected engine, input scope, destination, and archive stage.

9. Operational Guide

Karthikeyan Nagalingam, NetApp

This guide covers routine execution, monitoring, incident triage, training recovery, archive verification, capacity tuning, and data protection. Use it with the configuration scenarios in Section 8 to operate the same pipeline consistently across local, ONTAP S3, and LustreFS training modes.

9.1 Execution Walkthrough

Trigger a run through `trigger_and_wait_ai_pipeline_sklearn.sh` with a `CONF_JSON` payload. The script submits one uniquely named manual run, polls until it reaches a terminal state, and prints local log tails for failed tasks when available. Each primary task emits an `effective_config` record at startup so operators can verify the evaluated configuration rather than relying on the submitted payload alone.

9.2 Monitoring and Observability

Monitor DAG state in the Airflow UI or CLI and correlate it with the run ID and `run_stamp`. The guard logs are grep-friendly: `[data_prep] effective_config`, `[model_training] effective_config`, `[xcp_copy] effective_config`, and `[checkpoint] resume_summary`. Review these entries to validate the transformation engine, selected XCP destination, training source, and checkpoint reuse decision for each run.

9.3 Troubleshooting Quick Reference

Use this table to connect common task failures to the first corrective action. Resolve source connectivity and mount problems before retrying a run; retrying without correcting an invalid configuration or unavailable endpoint will reproduce the same failure.

Symptom	Likely Cause	Resolution
`Could not resolve XCP S3 credentials`	Missing/incorrect `xcp_s3_profile` in `xcp_s3_profiles`	Verify profile name and credentials map
`no readable Lustre source directory`	LustreFS not mounted or wrong `xcp_lustrefs_dest_path`	Verify the filesystem type with `findmnt -T /mnt/lustre/client -o FSTYPE,TARGET -n`; correct path

Symptom	Likely Cause	Resolution
`Missing required text dataset files`	Text JSONs absent in raw S3 prefix	Upload `text_base.json`, `text_finetune.json`, `text_infer.json`
Preflight exit code 1	SSH unreachable	Verify `ssh_default` connection and host/user
Preflight exit code 2	NFS path not exported/visible	Verify export with `xcp show <ip>`

9.4 Checkpoint Operations

Checkpointing applies only to `model_training`. Set `training_checkpoint_reuse_mode=resume_if_exists` to reuse a valid completed training result and skip redundant model training, or use `verify_only` to validate checkpoint eligibility without skipping. Keep reuse disabled (`off`) during initial testing or whenever training inputs, configuration, or expected artifacts have changed.

9.5 Manual Archive Operations

Set `manual_archive_enabled=true` and choose `manual_archive_stage` deliberately. Use `model_training` when governance requires the core model baseline as soon as training succeeds; use `inferencing` when the archive must include final predictions. A run with `model_training` selected continues to fine-tuning and inference after the archival branch; it simply does not archive their later outputs. Verify uploads under the run-stamped StorageGRID prefix and review `[manual_archive]` logs for the selected stage, files found, and upload count.

9.6 Scaling Guidance

Tune `sample_count`, `spark_driver_memory`, `spark_executor_memory`, and `spark_timeout_secs` to the input volume and service-level objective. Start with bounded samples to validate data shape and routing, then use `sample_count=0` for all-row runs after StorageGRID, ONTAP NAS, XCP, and the selected training tier have sufficient capacity and throughput.

9.7 Backup and Recovery

Protect the ONTAP NAS prepared-data bucket with ONTAP Snapshot and backup. Apply StorageGRID protection and retention policies to raw and archival buckets; the archival bucket retains history independent of pipeline re-execution. Combine these storage protections with the training checkpoint record to recover the correct run-scoped data and baseline artifacts after a task, host, or site-level interruption.

10. Security Considerations, Validation and Testing

Karthikeyan Nagalingam, NetApp

This combined section addresses the controls required to operate the AI pipeline securely and the validation activities used to prove the design behaves as expected in representative environments. The same storage-tier separation, least-privilege access model, checkpoint logic, and run-scoped lineage used in the pipeline also inform the validation plan and operational guardrails.

10.1 Security Considerations

Security controls should isolate each storage tier and pipeline function while preserving the lineage required for audit. Use separate least-privilege credentials for StorageGRID raw data, ONTAP NAS prepared data, the selected XCP destination, and StorageGRID archival. Encrypt network traffic and store all secrets outside the DAG definition, examples, and shell history.

Area	Recommendation
Credential management	Use Airflow Connections/Variables or a secrets manager; avoid inline plaintext in `dag_run.conf`
Data in transit	Use TLS-enabled S3 endpoints; encrypted NFS/Lustre transport where supported
Access control	Scope S3 credentials per role (StorageGRID raw, ONTAP NAS prepared data, XCP destination, StorageGRID archival) to minimum required permissions
Secrets in configs	Rotate any credentials exposed during testing; treat trigger scripts/shell history as sensitive
Audit trail	Rely on data-prep manifests, checkpoint records, and archival manifests for compliance evidence

10.2 Validation and Testing

Validation confirms that the pipeline's configurable behavior works reliably across its routing, failure-handling, data-mobility, and multi-file discovery paths. Each validation test area was executed in a representative environment using actual Airflow DAG triggers, with log outputs and storage manifests verified for correctness.

10.2.1 Functional Routing Validation

This test area validates end-to-end data and artifact routing when `enable_xcp=true`.

- **Objective:** Verify that data preparation, XCP mobility, model training, fine-tuning, and inference consistently use the selected XCP destination (`s3` or `lustrefs`).
- **Test Procedure:** Triggered DAG runs with `xcp_copy_destination="s3"` and `xcp_copy_destination="lustrefs"`. Inspected XCom outputs, effective-config logs, and storage locations.
- **Validation Outcome:**
  - In `data_prep`, formatted tabular splits and text files were staged to the ONTAP NAS prepared-data tier under `<xcp_prefix>/formatted/<run_stamp>/data/`.
  - In `xcp_copy`, that prepared dataset was transferred to the chosen XCP destination for downstream model stages.
  - In `model_training`, model inputs were loaded from the XCP destination, and baseline artifacts (`regression_model.bin`, `text_vectorizer.bin`, `text_classifier.bin`, `train_metrics.json`) were published back to `<xcp_prefix>/formatted/<run_stamp>/artifacts/`.
  - In `fine_tuning`, base vectorizer/classifier artifacts were materialized from the XCP destination, incrementally tuned, and published back as `text_classifier_tuned.bin` and

`fine_tune_metrics.json`.

- In `inferencing`, the tuned text classifier plus the baseline regression model and vectorizer were materialized from the XCP destination to generate predictions.

10.2.2 Local Fallback and Non-XCP Execution Validation

This test area validates pipeline behavior when XCP data mobility is disabled (`enable_xcp=false`).

- **Objective:** Ensure the pipeline operates seamlessly using direct local prepared-data paths without requiring SSH connectivity, remote XCP hosts, or XCP credential profiles.
- **Test Procedure:** Triggered pipeline runs with `enable_xcp=false` and default S3/local paths.
- **Validation Outcome:**
  - XCP preflight and copy tasks were safely bypassed.
  - `model_training`, `fine_tuning`, and `inferencing` read directly from local prepared-data directories and published artifacts locally.
  - Verified that no XCP S3 credential resolution or SSH preflight errors occurred when XCP was disabled.

10.2.3 Performance and Tier Comparison (ONTAP S3 vs. LustreFS)

This test area evaluates I/O latency, discovery overhead, and training throughput between ONTAP S3 object storage and LustreFS parallel filesystem tiers.

- **Objective:** Quantify performance characteristics of E-Series with LustreFS versus ONTAP AFF/AFX with ONTAP S3 active training tiers.
- **Test Procedure:** Measured file discovery time, training dataset load latency, and artifact publish/materialize duration across identical dataset sizes (14,400+ rows, multi-file text/tabular inputs).
- **Validation Outcome:**
  - **LustreFS Tier:** Demonstrated lower file-discovery overhead and faster random-access read latency during model training, making it ideal for high-file-count, GPU-intensive parallel training workloads.
  - **ONTAP S3 Tier:** Provided high throughput with simpler multi-protocol management, making it optimal for cost-conscious, elastic-access active storage without requiring client-side filesystem mounts.

10.2.4 Failure Recovery and Training Checkpoint Validation

This test area validates the checkpoint resume system scoped to `model_training`.

- **Objective:** Verify that pipeline recovery accurately detects prior successful training runs and skips redundant computation while preserving artifact integrity.
- **Test Procedure:**
  1. Ran pipeline with `checkpoint_enabled=true` and `checkpoint_store="formatted_s3"`.
  2. Simulated task failure or re-execution with `training_checkpoint_reuse_mode="resume_if_exists"`.
  3. Tested `verify_only` and `off` reuse modes.
- **Validation Outcome:**
  - When `resume_if_exists` was set and a valid checkpoint JSON + all core artifacts existed in S3/LustreFS, `model_training` exited immediately with decision `RESUMED_FROM_CHECKPOINT`,

logging `[checkpoint] resume_summary: decision=RESUMED_FROM_CHECKPOINT`.

- Downstream `fine_tuning` and `inferencing` successfully materialized the verified checkpoint artifacts.
- When `verify_only` was set, the guard validated artifact existence without skipping training.

10.2.5 Scalability and Multi-File Dataset Validation

This test area validates pipeline scalability across large, multi-file tabular and text datasets.

- **Objective:** Confirm automatic dataset discovery, Spark transformation engine execution, and Lakehouse table format support (`delta` and `iceberg`).
- **Test Procedure:**
  1. Ran ingestion and preparation across dozens of CSV/Parquet files under the StorageGRID raw prefix (`s3_raw_prefix`).
  2. Tested explicit file selection using `s3_tabular_object_keys` vs automatic all-file discovery (`sample_count=0`).
  3. Executed preparation using PySpark with `table_format="delta"` and `table_format="iceberg"`.
- **Validation Outcome:**
  - Spark distributed preparation successfully ingested, filtered, split, and formatted large multi-part tabular datasets.
  - Automatic discovery accurately identified all valid tabular CSV/Parquet objects while ignoring non-tabular artifacts.
  - Both Delta Lake and Iceberg table formats were correctly created and registered under `tables/`, with downstream model training discovering and reading the formatted splits.

10.2.6 Example Customer Evaluation Workflow, Sizing Assumptions, and Validation Metrics

This workflow helps customers evaluate whether the design fits their AI pipeline maturity, data-sovereignty requirements, and performance targets. Begin with a representative dataset and one pipeline run, then increase data volume, file count, model complexity, and concurrent runs only after each acceptance criterion is met. Record results by `run_stamp` so comparisons between ONTAP S3 and LustreFS use the same input data and configuration.

Evaluation Step	Example Workflow	Sizing Assumption / Decision	Validation Metrics and Acceptance Evidence
1. Establish a baseline	Run the Python preparation path with `enable_xcp=false` on a bounded sample.	Use the reference validation environment in Section 7.1.1 as an initial functional baseline.	Successful DAG completion; input-row and output-split counts; model metrics; task duration; CPU, memory, and local-disk utilization.

Evaluation Step	Example Workflow	Sizing Assumption / Decision	Validation Metrics and Acceptance Evidence
2. Validate data sovereignty	Keep raw data, prepared data, active training data, and archives in approved storage endpoints and regions. Review credentials and retention policies per tier.	Define the allowed locations, access roles, encryption requirements, and retention policy before moving data.	Endpoint, bucket, prefix, and region recorded in the effective configuration and manifests; least-privilege access verification; archive and deletion-policy evidence.
3. Validate data mobility	Enable XCP and copy the same prepared run to ONTAP S3, then repeat to LustreFS.	Ensure 10 GbE connectivity and sufficient destination capacity for the prepared run, artifacts, working space, and concurrent runs.	XCP completion status; bytes and files copied; copy duration and throughput; source/destination file-count and checksum or manifest comparison; preflight success.
4. Validate training-tier fit	Train, fine-tune, and infer against each selected destination using the same datasets and model settings.	Use ONTAP S3 for object-based workflows; use E-Series with LustreFS when parallel training I/O or high file counts justify it.	Dataset discovery and load time; training, fine-tuning, and inference duration; artifact publish/materialize duration; CPU/GPU utilization where applicable; model-quality consistency.
5. Validate scale and recovery	Increase `sample_count` to all rows, increase file count and concurrent runs, then test checkpoint reuse and archival.	Size capacity for retained run-scoped datasets and artifacts, plus growth headroom; size CPU and memory for Spark and concurrent Airflow tasks.	End-to-end elapsed time; successful run rate; queue time; checkpoint resume decision and elapsed time; archive completeness; storage growth per run; recovery time objective (RTO) attainment.

Before starting the evaluation, customers should define quantitative targets for end-to-end runtime, XCP throughput, training-data load time, maximum concurrent runs, recovery time, and storage retention. The measured baseline and each scaled test should be compared with those targets to determine whether the architecture meets the intended workload and operational requirements.

11. NetApp Differentiation and Competitive Positioning, Appendices

Karthikeyan Nagalingam, NetApp

This combined section positions the architecture against common alternatives and provides the shared reference material used by operators and reviewers. The differentiator is not a single storage endpoint; it is the policy-controlled combination of StorageGRID, ONTAP NAS, ONTAP S3, LustreFS, XCP, and Airflow that lets each AI lifecycle phase use an appropriate tier without rebuilding the workflow.

11.1 NetApp Differentiation and Competitive Positioning

This comparison positions the architecture against common alternatives. Its differentiator is not a single storage endpoint; it is the policy-controlled combination of StorageGRID, ONTAP NAS, ONTAP S3, LustreFS, XCP, and Airflow that lets each AI lifecycle phase use an appropriate tier without rebuilding the workflow.

Alternative Approach	Limitation	NetApp Solution Advantage
Cloud-native-only pipelines	Locked to one storage paradigm	Interchangeable S3/LustreFS via configuration
DIY script-based pipelines	No retry, checkpoint, or lineage	Airflow orchestration + checkpoint/resume natively
Single-tier storage architectures	One performance/cost profile for all workloads	Per-run tier selection via `xcp_copy_destination`
Full retraining on every update	High recurring compute cost	Incremental fine-tuning via `partial_fit`

11.2 Appendices

The appendices provide shared terminology and document-control context for readers who operate, review, or adapt this NVA. Keep the revision history current when changing documented behavior, parameters, storage layouts, or validation evidence.

11.2.1 Glossary

These definitions establish consistent meanings for the architecture's storage, orchestration, and lineage terms.

Term	Definition
XCP	NetApp's high-speed parallel data copy/migration tool
ONTAP S3	NetApp's S3-compatible object storage service on ONTAP
LustreFS	High-performance parallel distributed filesystem
DAG	Directed Acyclic Graph — Airflow's workflow definition
XCom	Airflow's mechanism for passing data between tasks
run_stamp	Unique per-execution timestamp identifier used for lineage
Manifest	JSON record describing a stage's inputs, outputs, and configuration

11.2.2 Revision History

Record each published change to preserve the document's review and approval trail.

Version	Date	Author	Change
1.0	2026-09-25	Solution Architecture Team	Initial publication-ready NVA with full parameter tables and diagrams

11.2.3 Contact Us and Software Requests

To request the software package, automation DAGs, deployment scripts, or validation artifacts to deploy and validate this architecture, or to provide feedback and technical inquiries, contact the NetApp AI & Data Mobility team:

- **Email:** ng-data-mobility-in-ai-pipeline@netapp.com
- **Purpose:** Software package requests, deployment guidance, architecture validation support, and technical feedback.

Copyright information

Trademark information