Architecting the Enterprise Data Mesh

A High-Level Design for Implementing a Multi-Environment Platform with Amazon SageMaker Unified Studio

[Data Platform Team Logo]

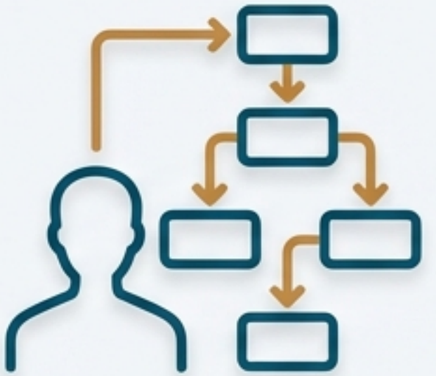
Our Mission: Empower Teams While Ensuring Control

Platform Agility & Innovation

The Data Platform team requires isolated `dev` and `staging` environments to develop and test new platform features (e.g., custom subscription policies, new project templates) without impacting business operations.

Data Sovereignty & Security

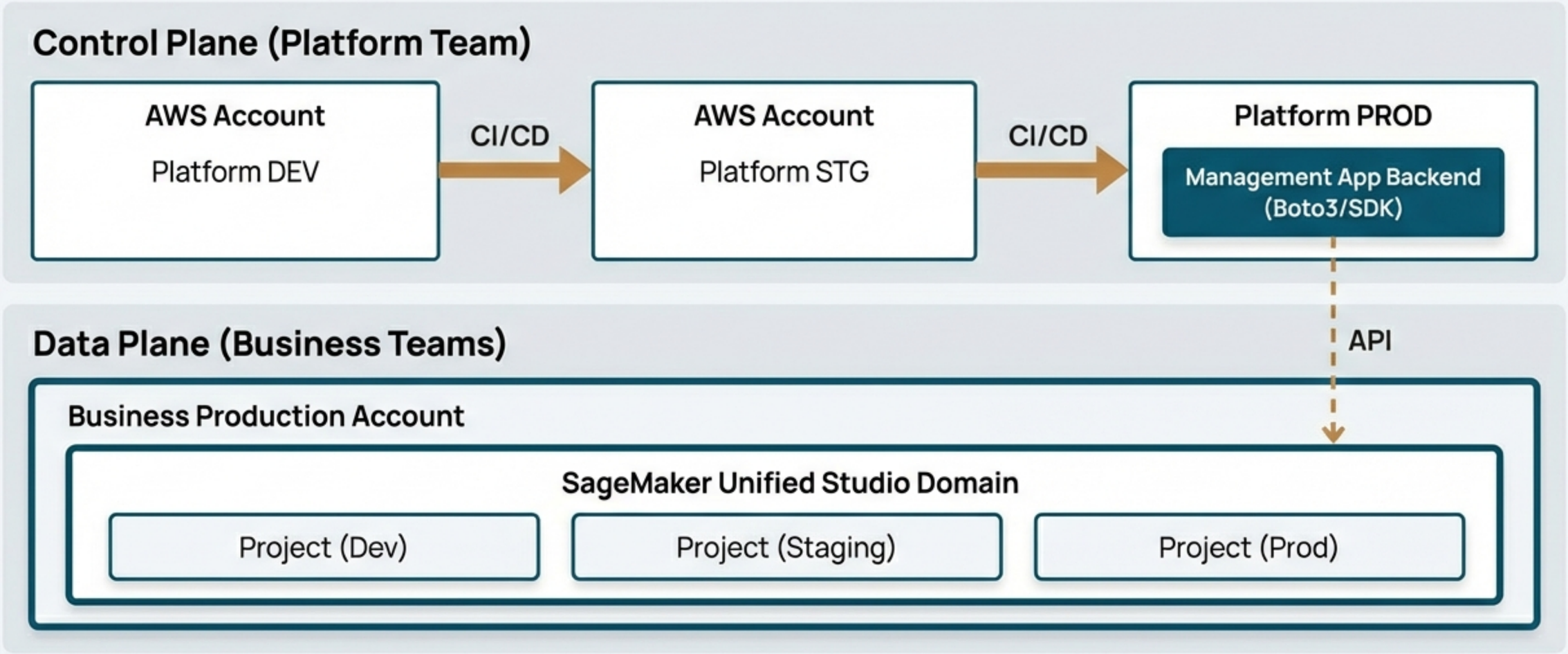
A non-negotiable mandate to keep all business and production data within a single, secure, and audited production AWS account. This simplifies compliance and centralizes data gravity.



Federated Self-Service

The goal is to provide a seamless, automated experience for business teams to discover data assets, create logically separated projects for their work, and manage the entire ML lifecycle via a custom UI-driven application.

The Architectural Blueprint: A Dual-Plane Approach



We separate the platform's lifecycle from the business's data lifecycle for maximum agility and security.

Layer 1: The Control Plane - Building the Factory

Platform Dev Account

Purpose: Rapid experimentation with new platform features and the AWS SDK.

Access: Strictly limited to the Data Platform development team.

Example Use Case: "Testing a new Boto3 script to automate subscription approvals or modifying IAM policies for project roles."

Platform Staging Account

Purpose: End-to-end, pre-production testing of the Management App and its integrations.

Example Use Case: "Validating that a new version of the Management App can correctly create and configure a test SageMaker Project before deploying the app to production."

Platform Prod Account

Purpose: Hosts the stable, production version of the Management App backend. This is the application that interacts with the live Data Plane.

Example Use Case: "The running, trusted backend that serves requests from the business-facing UI to manage projects and subscriptions in the Business Production Account."

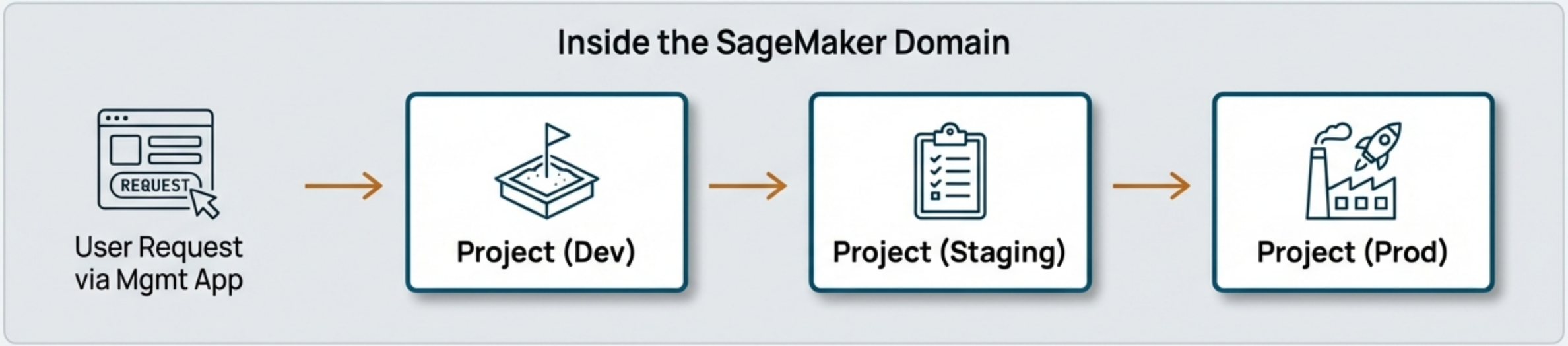
Layer 2: The Data Plane - The Unified Domain

Core Principle: A Single Domain in Production

- **Data Gravity & Security**: Keeps all sensitive business data within one audited account boundary, radically simplifying security posture and compliance overhead.
- **Unified Discovery**: A single domain hosts a single **Amazon SageMaker Catalog**. This enables seamless, cross-project discovery of all published data products, forming the foundation of our data mesh.
- **Centralized Governance**: Simplifies the application of overarching security controls, identity management (via IAM Identity Center), and network policies at the domain level.

> The **SageMaker Domain** is the parent container for all users, projects, and shared assets. It acts as the central hub for our entire Data Mesh, where collaboration and discovery happen.

Layer 3: From Sandbox to Production within a Single Domain



SageMaker Projects as Logical Environments

A Project is a defined boundary for collaboration, permissions, and resources. It provides members with their own isolated data, compute, and source control.

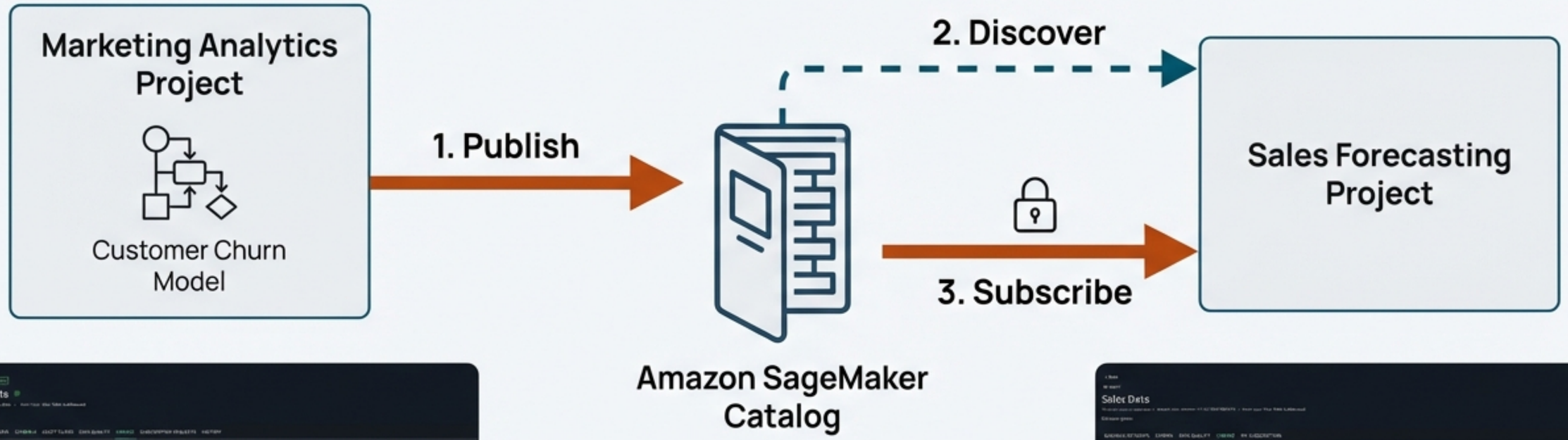
- `**Project (Dev)**": For experimentation, ad-hoc analysis, and new model development.
- `**Project (Staging)**`: For UAT, integration testing, and validation with production-like data before a full rollout.
- `**Project (Prod)**`: For operationalized models, production pipelines, and final reporting assets.

Mechanism for Standardization: Project Profiles

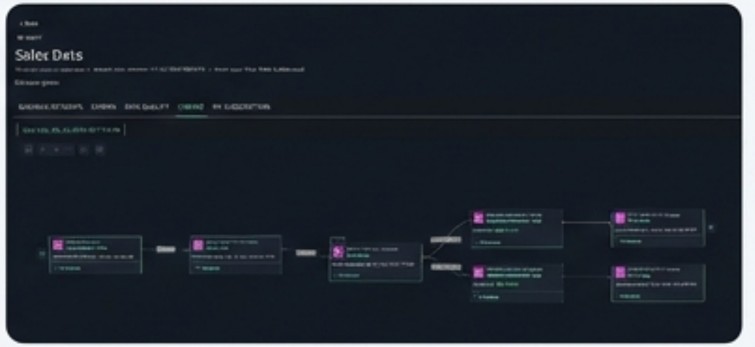
Each project is created from a template called a **Project Profile**, which is configured by the platform administrator.

Profiles control the tools and resources available (e.g., "SQL analytics," "Generative AI application development," "All capabilities"). This ensures consistency and governance for each environment type.

The Data Mesh in Action: Publishing & Subscribing



Producer View: Marketing Analytics



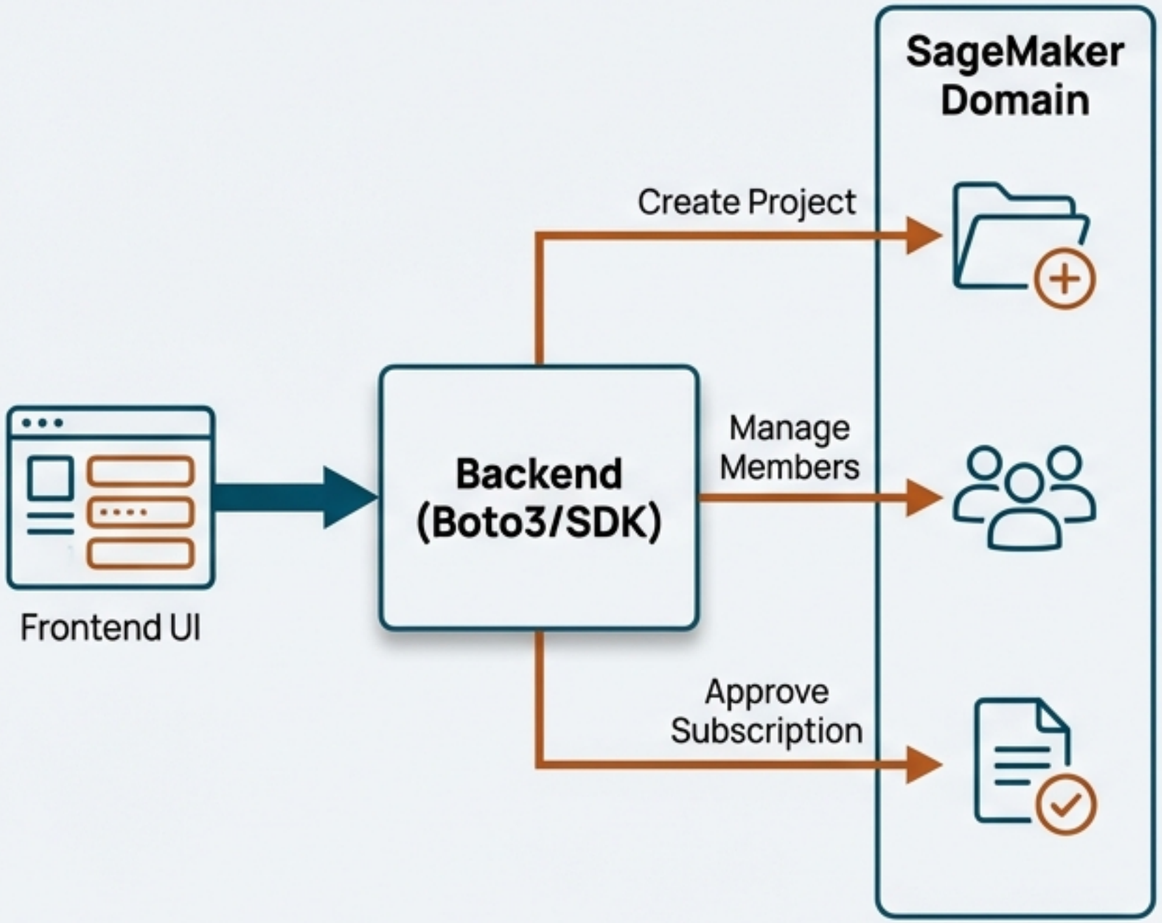
Consumer View: Sales Forecasting

Publishing as a Product: Projects can publish their outputs (models, datasets from Glue, feature groups) to the domain-wide Catalog, making them discoverable data products.

Federated Discovery: The Catalog provides a single, searchable interface for all users in the domain to find available assets, regardless of which project created them.

Governed Subscription: A formal workflow where consumers request access. This approval process will be managed and automated by our custom Management Application.

Layer 4: The Engine of Self-Service - The Management App



Automation Powered by the SDK

The backend leverages the **Amazon SageMaker Unified Studio Library for Python** and Boto3 to programmatically interact with the domain, enabling full automation of user requests.

Key SDK Operations (from `sagemaker_studio` library):

- **Project Management**: `Project(name=...)` to instantiate and manage projects. Access properties like `proj.id`, `proj.project_status`, `proj.iam_role`.
- **Resource Management**: Access project S3 paths (`proj.s3.root`) and manage data source connections (`proj.connection(...)`).
- **Governance**: Programmatically manage members and their roles (Owner, Contributor).

```python
from sagemaker_studio import Project

# Conceptual backend API endpoint
def create_new_project(user_details, proj_name, template_id):
    # Logic to create project via CloudFormation/Service Catalog
    # then instantiate the object to manage it
    proj = Project(name=proj_name, domain_id=OUR_DOMAIN_ID)
    proj.add_member(user_details['id'], role='Owner')
    # ...further configuration...
    return {"status": "success", "project_id": proj.id}
```

Secure by Design: Trusted Identity Propagation

What is Trusted Identity Propagation?

A feature enabled in IAM Identity Center-based domains where a user's identity is securely passed down from SageMaker Studio to underlying AWS services like Athena, Redshift, and Lake Formation.

Why It's Critical for Our Data Mesh

- **Fine-Grained Access Control**: Enables data access policies based on the *end-user's identity and group memberships*, not a generic service role. This is the cornerstone of domain-level data security.
- **Complete Auditability**: All queries in Athena or Redshift are logged against the specific user who ran them, providing a clear and unbroken audit trail.
- **Simplified Governance**: Allows us to manage data permissions centrally in services like Lake Formation, and trust that Studio will enforce them automatically for every user and every query.

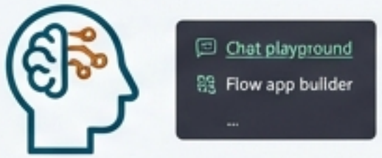
A Unified Workspace for Every Role

Data Exploration & SQL Analytics

- **Tools:** Query Editor, Visual ETL.
- **Use Case:** Data analysts browse the Catalog, run interactive SQL queries against subscribed data, build no-code transformation jobs, and schedule recurring queries.

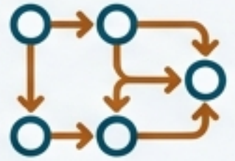
ML Development & Notebooks

- **Tools:** JupyterLab, Code Editor, Managed Spark/Python Kernels.
- **Use Case:** Data scientists pull subscribed data into a notebook, use Spark for large-scale processing, train models, and track experiments with integrated MLflow.



Generative AI Development

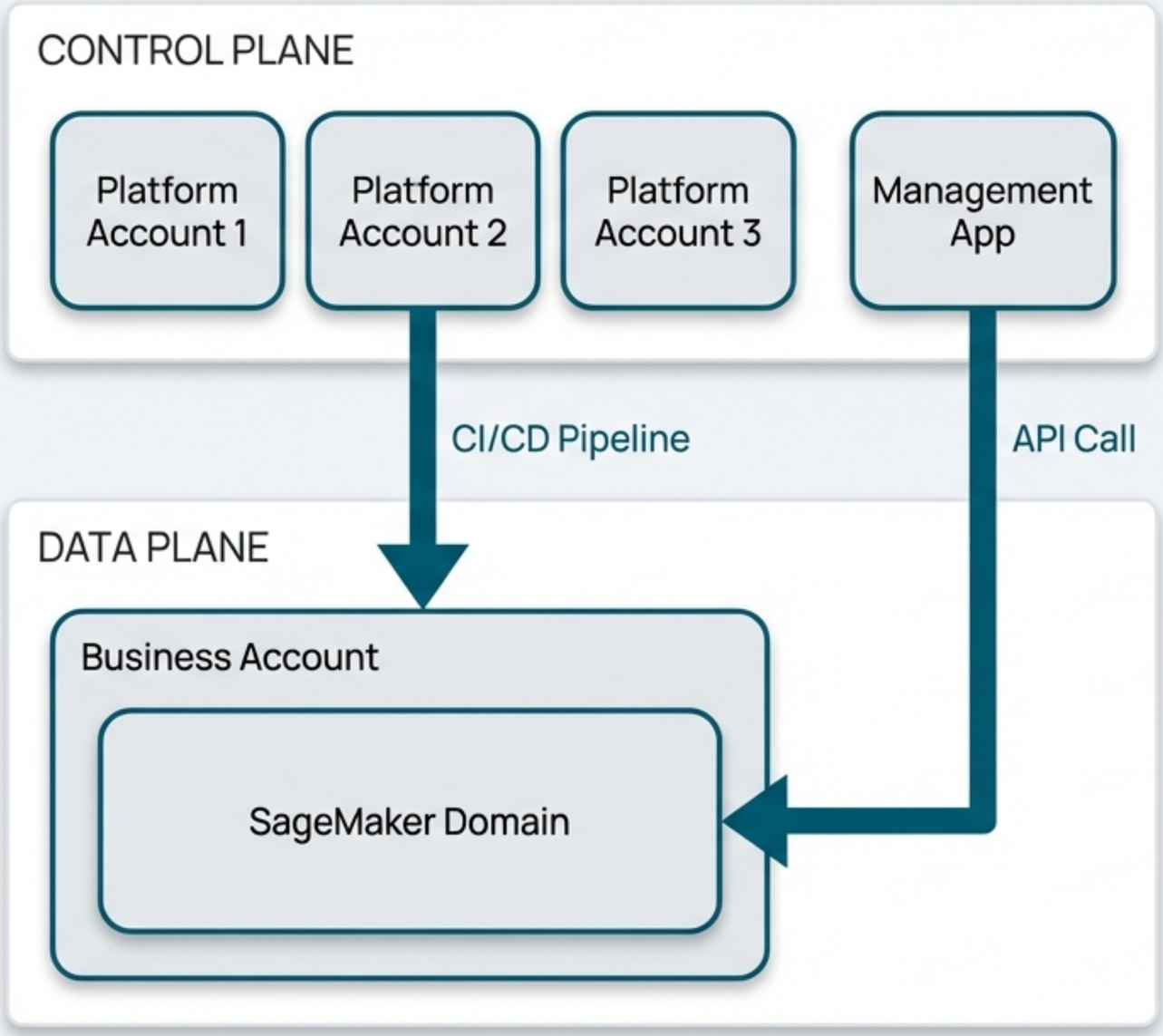
- **Tools:** Bedrock Model Catalog, Chat/Image Playgrounds, App Builder (Chat Agent & Flow Apps).
- **Use Case:** App developers rapidly build and test GenAI applications using foundation models, enhancing them with `Knowledge Bases` for RAG and safeguarding them with `Guardrails`.



Automation & MLOps

- **Tools:** Workflows (Managed Airflow), Pipelines, Training Jobs.
- **Use Case:** MLOps engineers orchestrate multi-step data processing and model training pipelines, defining dependencies and schedules either visually or in code.

Architectural Summary: A Purpose-Built Platform



Key Architectural Pillars

- **Multi-Account Control Plane**: Guarantees platform stability and enables agile, safe development for the platform team.
- **Single Production Data Plane**: Centralizes business data, simplifies governance, and enables powerful, cross-domain discovery.
- **SageMaker Projects**: Provide logically isolated, governed, and templated environments for the full business ML lifecycle.
- **Amazon SageMaker Catalog**: Serves as the central, discoverable registry for all data products, enabling the core data mesh workflow.
- **Custom Management App**: Delivers a seamless, automated, and self-service experience powered by the Studio SDK.
- **Trusted Identity Propagation**: Ensures end-to-end, fine-grained security and a complete audit trail for all data access.

Path to Implementation

Phase 1: Foundational Setup

- Establish AWS Organization and the 3+1 multi-account structure.
- Configure IAM Identity Center as the authentication source.
- Deploy the core SageMaker Domain in the Business Production account.

Phase 2: Management App MVP

- Develop core backend functions (using the Python SDK) for project creation and member management based on predefined Project Profiles.
- Build the initial UI for users to request new projects.

Phase 3: Pilot Project Onboarding

- Onboard a pilot business team to test the dev -> staging -> prod project workflow.
- Establish and document the patterns for publishing and subscribing to the first set of data products in the SageMaker Catalog.

Key Design Decisions for Discussion

? • What is our CI/CD strategy for the Management App itself?
? • What specific configurations are needed for the initial `Dev`, `Staging`, and `Prod` Project Profiles (e.g., instance types, permissions)?
? • What are the data classification standards for our first data products?

Appendix: Key SDK Components (`sagemaker_studio` Library)

Section 1: Core Objects

Project
The primary object for interacting with a project.

```
proj = Project(name="...")
# Properties:
proj.id
proj.name
proj.iam_role
proj.project_status
```

Connection
Object for managing data sources within a project.

```
conn = proj.connection("project.athena")
redshift_client = conn.create_client()
```

Section 2: Utility Modules

sqlutils
A utility for executing SQL against project connections.

```
sqlutils.sql(
    "SELECT * ...",
    connection_name="project.athena"
)
```

sparkutils
A utility for initializing and configuring Spark Connect sessions.

```
spark = sparkutils.init(
    connection_name="my_spark_connection"
)
```

Note: For IAM-based domains, this library is supported only when using Space Distribution Image version 2.11+ or 3.6+ in JupyterLab or Code Editor.

Appendix: Example Project Profile Configurations

Profile	Purpose	Example IAM Policies	Example Tooling
business-dev-profile	Initial data exploration and experimentation.	Read-access to subscribed datasets, write-access to project-specific S3 path, limited compute instance types (e.g., `ml.t3.*`).	Query Editor, JupyterLab, Visual ETL enabled.
business-staging-profile	Pre-production validation and UAT.	Read-access to production-like data, write-access to staging-specific S3 path, ability to create SageMaker Pipelines and Endpoints.	All tools enabled, potentially with larger instance types (ml.m5.*`).
business-prod-profile	Operational workloads and production models.	Highly restricted write permissions, requires approvals for changes, access to production-grade compute (including GPU instances).	Focus on Pipelines, Training Jobs, Inference Endpoints. May have restricted interactive