Getting new ML models in production with ease

October 15, 2024
Ashraf Ahmed
Enterprise Architect, Sophi
aahmed@mathereconomics.com

Agenda

1. What is MLOps?
2. The main challenges of MLOps.
3. Running example: Sophi User Paywall Engine.
4. Data engineering and governance at scale.
5. Continuous model delivery at scale.
6. Experimentation and model control at scale.
7. Key takeaways.

What is MLOps?

- AI systems are **software intensive systems**.
- **Software engineering practices** should also be applied to AI systems.
- **Maintainability**: the simplicity with which you can repair, improve, and comprehend software code. Begins after the product has been delivered.
- **Efficiency:** avoiding wastage such as defects, overproduction, and excessive revisions.
- **Correctness:** when a program or system operates exactly as planned for all of its use cases

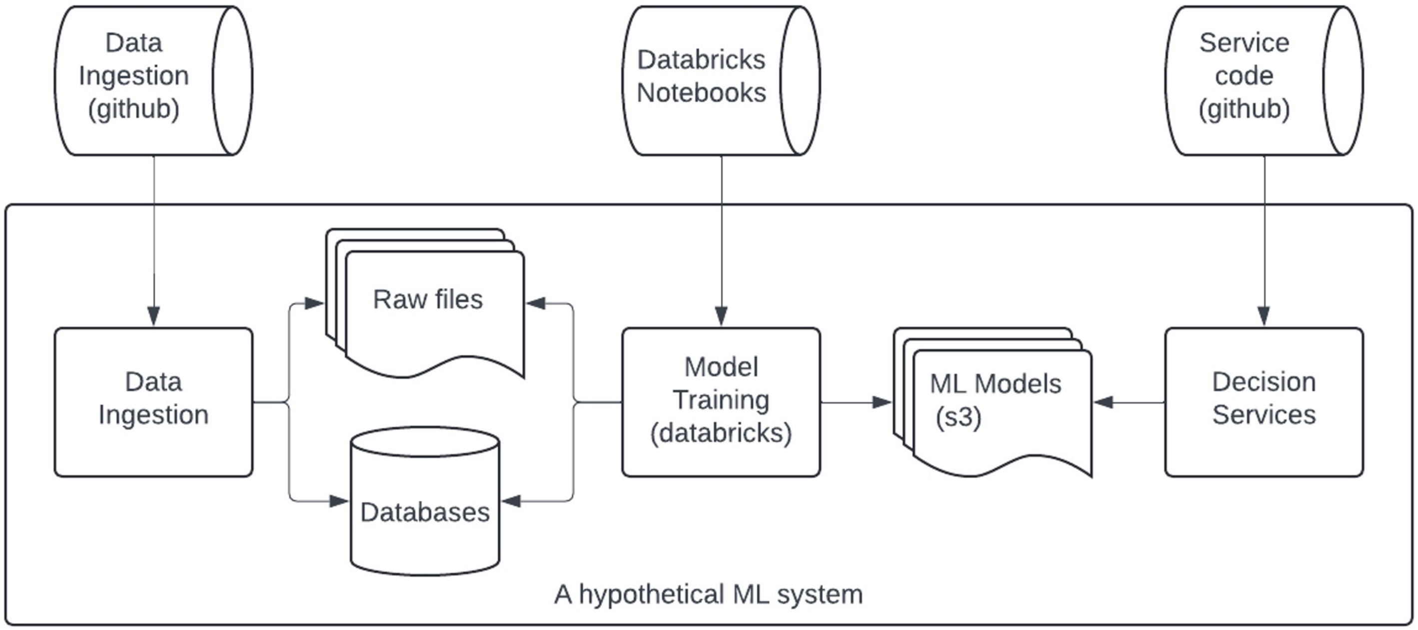
What is MLOps?

- Software engineering practices are more difficult to apply to AI systems.
- The world is constantly changing. **Models will be retrained** otherwise they might drift.
- **AI code depends on data**. Software engineering practices must be extended to data ingestion, processing, and quality control.
- Training code for **early models** might **not be optimized for large amount of data**. Requires considerations of **distributed data processing**.
- **Correctness requires more data**, more **complex models**, and continuous **experimentation**.

What is MLOps?

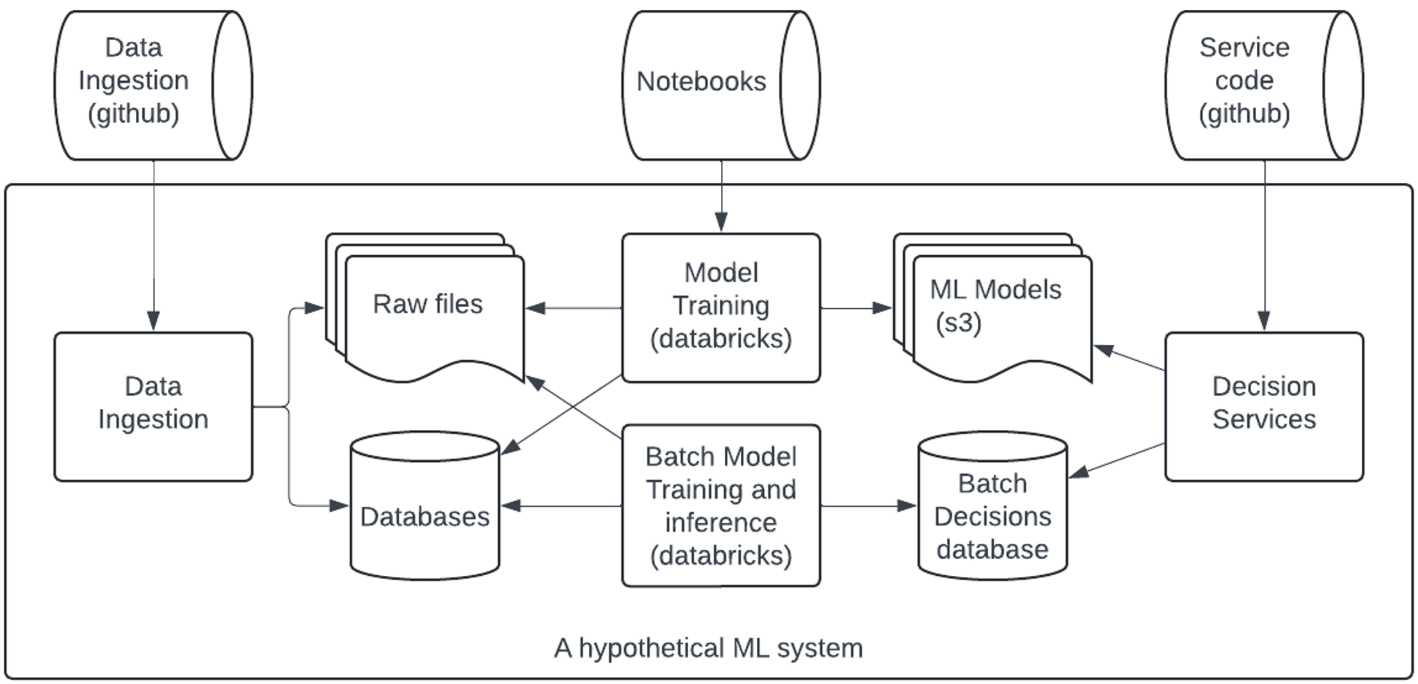
- A **set of practices** for operating AI models in production environments.
- **No agreed upon** set of practices.
- Concerned with Maintainability, Efficiency, and Correctness of **production AI systems.**
- Is impacted by:
    - Data engineering practices and governance.
    - Software Architecture.
    - Continuous model delivery.
    - Model control and experimentation.
    - Infrastructure management: deployment, change control, and monitoring.
- Successful MLOps **requires a successful strategy** for all the above.

The main challenges of MLOps



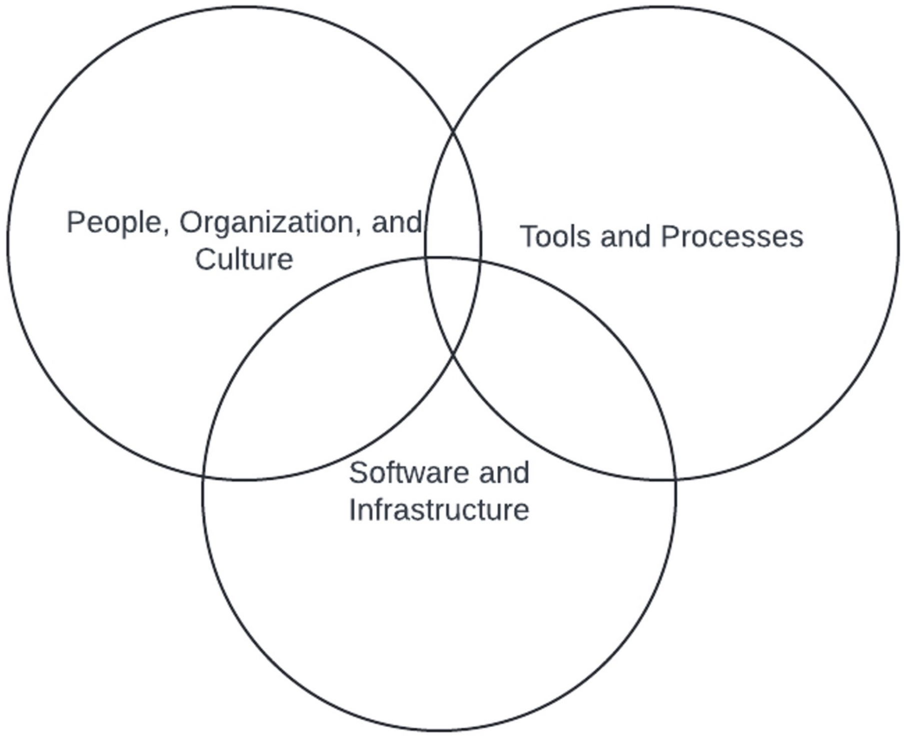
1. No deliberate integration points between engineering and ML code
2. All data processing takes place with ML code.
3. ML training results are in the logs.
4. ML Model performance data is in engineering system logs.
5. Engineering components contain ML dependencies.
6. Bringing new models requires changing non ML components.

The main challenges of MLOps



1. Waste in how data is being modeled.
2. Batch decision system are tightly coupled to data (and services that depend on that data).
3. Data scientists have to deal with how to efficiently and correctly update databases without causing outages.
4. Changing database technology or schema is rather difficult.
5. System is very inflexible for experimentation

The main challenges of MLOps



The main challenges of MLOps: people, organization, and culture

- Solving MLOps problems is not the main focus when building AI systems.
  - The team is typically focused on the main ML/AI problem.
  - Scaling is a secondary problem. You have to first succeed to scale.
- Not all enterprises make it a strategic priority.
  - Organizations typically focus on short term goals (one or two quarters typically).
  - It is harder to quantify value of long term investments.
- Requires more resources than that is needed for solving the main ML problem.
  - Human planning fallacies.
  - Solving the main ML problem requires a different set of skills from scaling MLOps.
- Solving MLOps problems requires a strongly collaborative team.
  - Data engineers and system engineers tend to focus on engineering problems (scale, correctness, change control ...etc).
  - Data Scientists and ML practitioners tend to focus on solving the main ML problem.

The main challenges of MLOps: software and infrastructure

- AI systems are more complex.
  - New components are needed for continuous model improvements (i.e. retraining, tuning, new models ...etc).
  - New components are needed for monitoring data and model quality.
  - Measuring the impact of any single model change is hard to predict ahead of time without extensive large scale testing.
- Current MLOps tools, both open source and end to end systems, are not drop in solutions for many software systems.
  - The problem of ML model tracking is treated separately from how the models are used.
  - No of-the-shelf solutions for running production and alpha models alongside each other.
- Infrastructure demands for AI systems are different from traditional software.

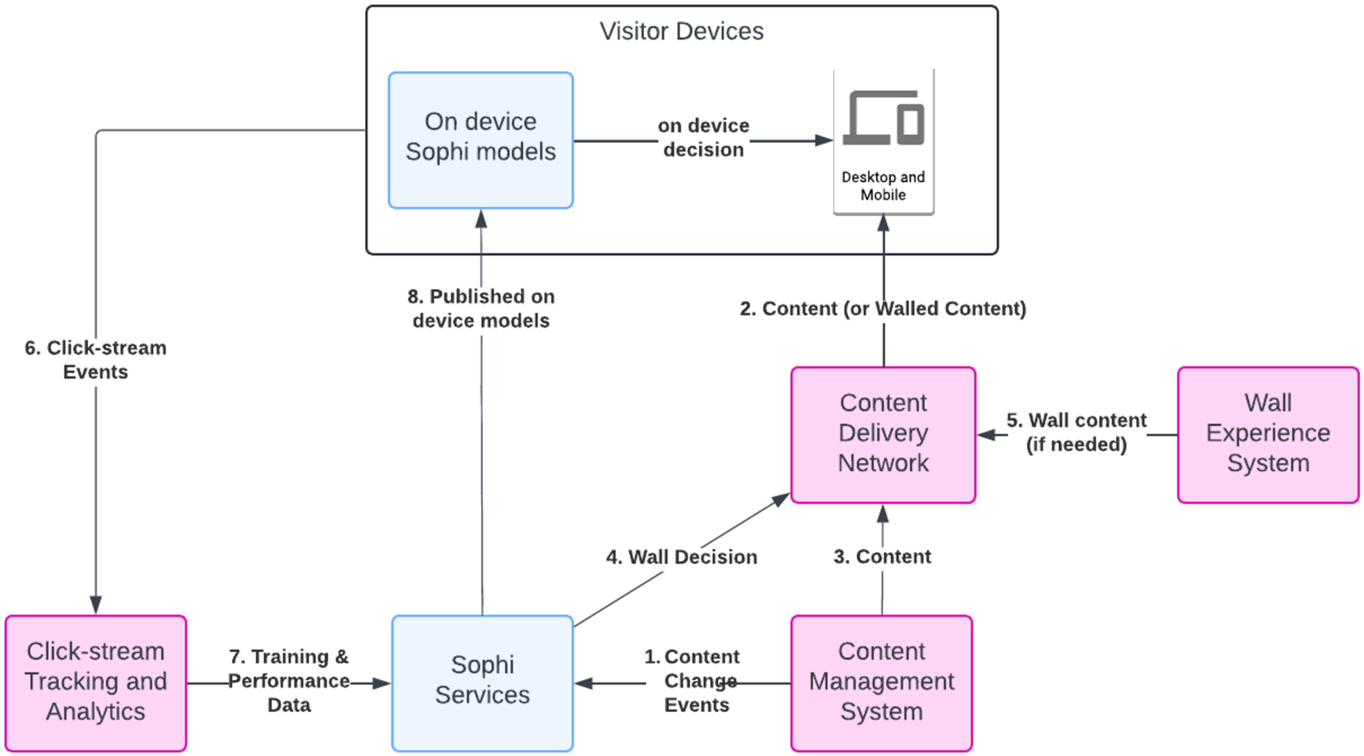
The main challenges of MLOps: tools and processes

- Does not directly benefit from all well established software engineering practices and tools.
  - AI/ML systems quality depends on the data (and how the data is being modeled).
  - Well established software engineering assumes the system is deterministic (i.e. one could create an exhaustive list of requires which can be tested through the development lifecycle).
- Rapid model development and experimentation is still rather new for many small teams due to lack of well established tools and processes.

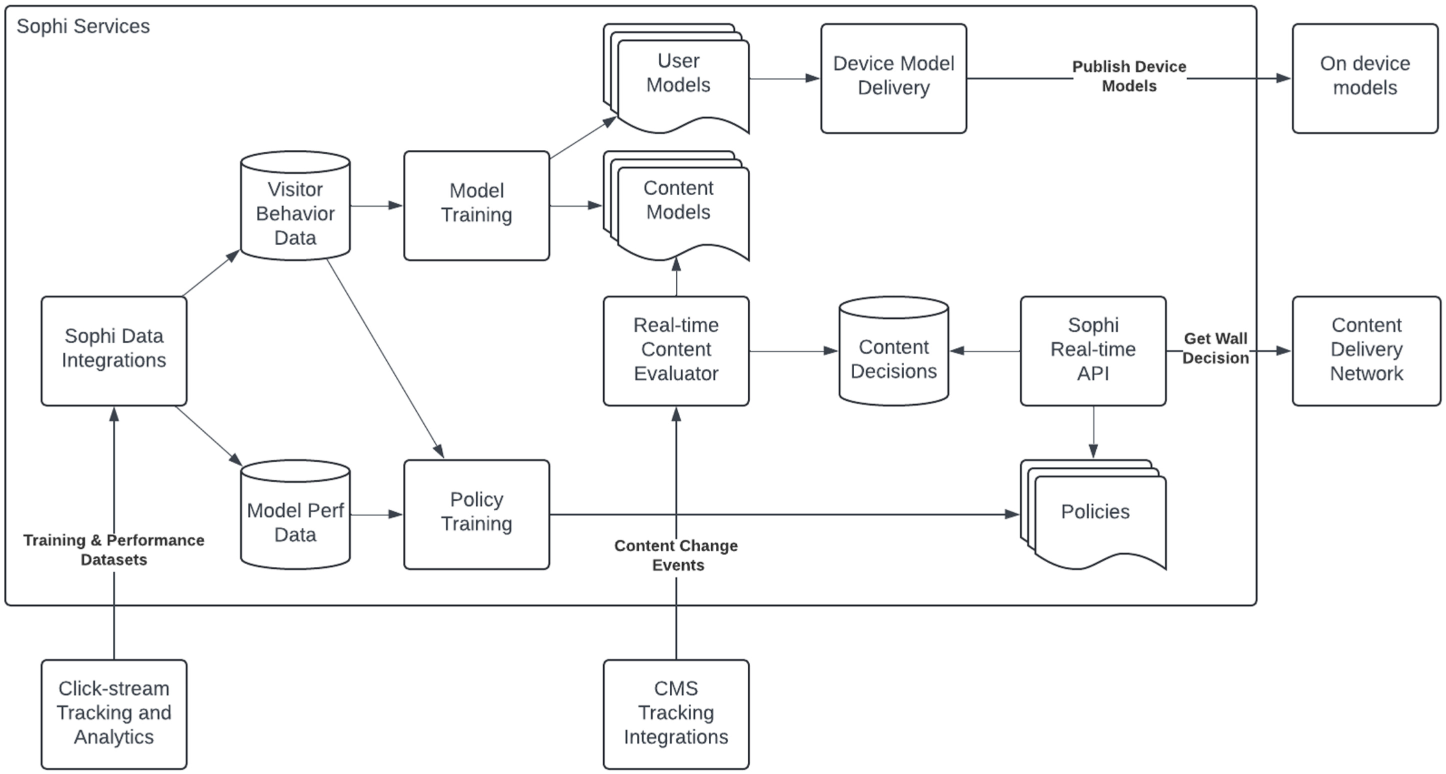
Running example: Sophi User Paywall Engine

- Maximize future revenue by balancing between subscriber and advertisement revenue.
- Choose the best paywall and regwall strategies for both content and users while honoring newsroom constraints such as a particular walling ratio and stop rate.
- Work with different content management systems, click-stream tracking and analytics systems, and wall experience systems.

Running example: Sophi User Paywall Engine



Running example: Sophi User Paywall Engine



Scaling out MLOps

Data and model governance

1. Data catalog
2. Data models
3. Rapid data modeling
4. Model tracking and versioning.
5. Model building vs model operations.

Continuous model delivery

1. Training at scale.
2. Training asset management
3. Integration with engineering components.
4. Delivery without releases.

Experimentation and model control

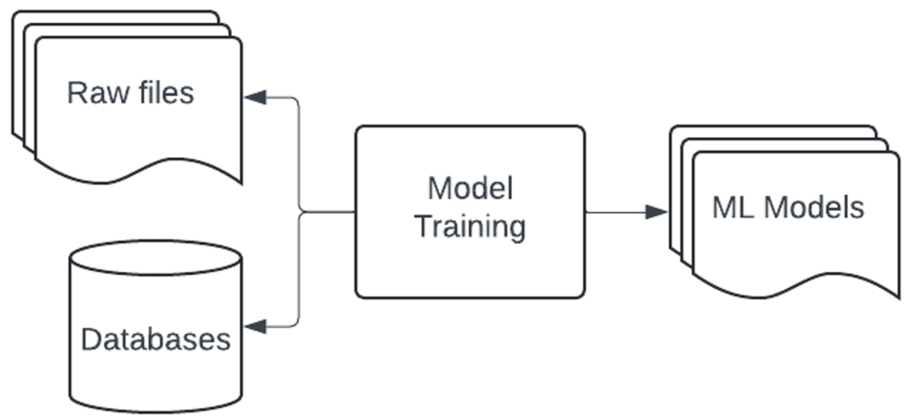
1. Dark mode testing.
2. managing prod and alpha models.

Data engineering and governance at scale: objectives

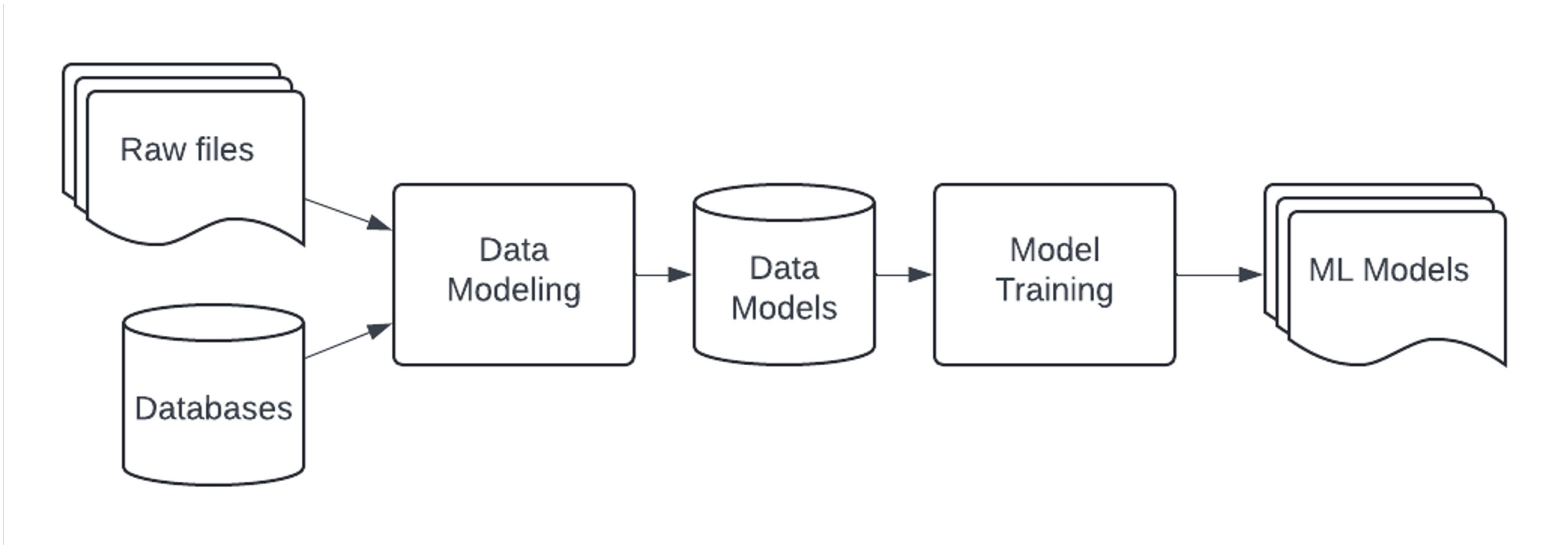
- Data catalog vs data models: definitions vs transformations.
- Rapid data modeling: adding and changing dimensions rapidly.
- Data model tracking and versioning: how to protect downstream systems.

Data engineering and governance at scale: challenges

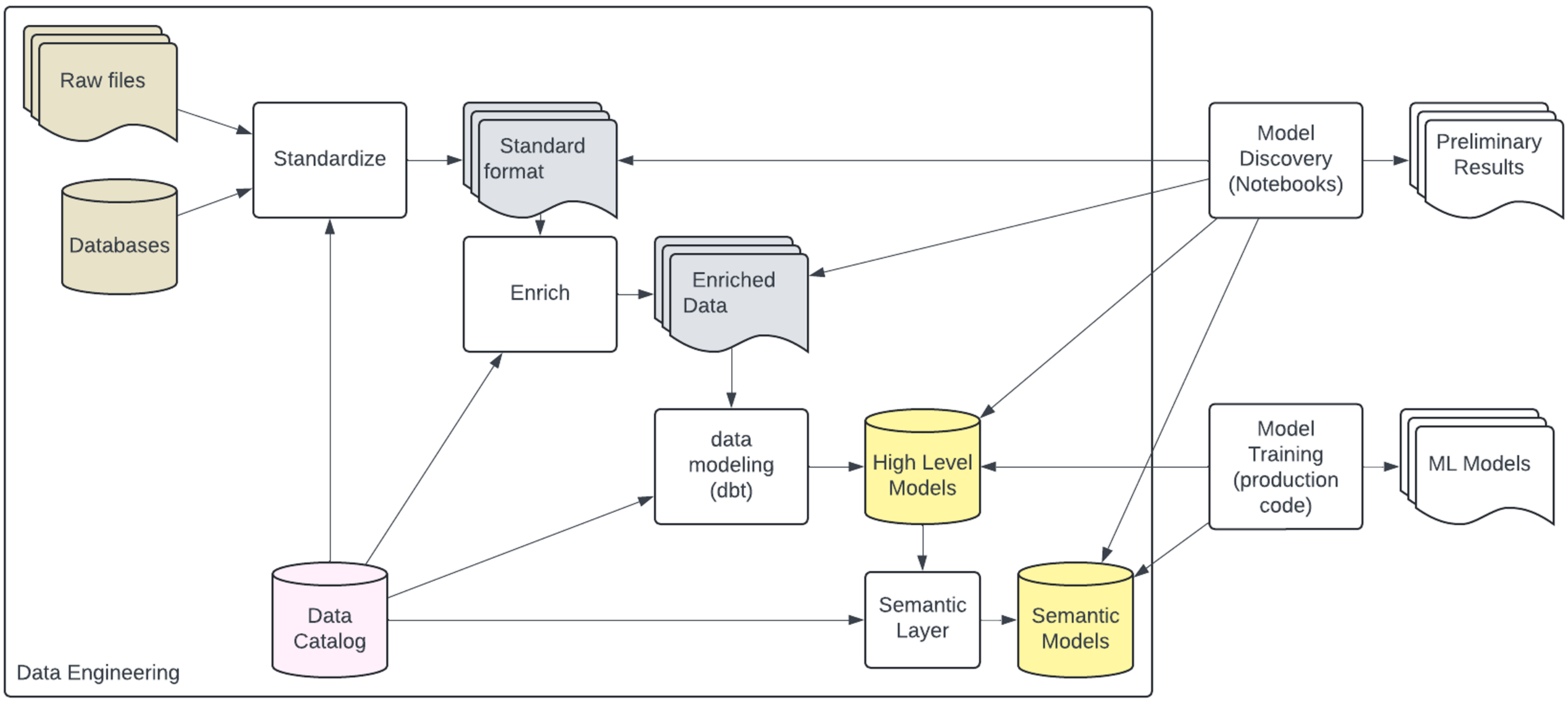
- Model training code is too coupled to data processing.
- Access to data depends on how data is stored.
- Quality of data model depends on data scientist time and understanding of possible defects.
- No formal process for data quality measurement, monitoring, and lineage.
- Data modelling is the responsibility of Data scientists.
- No formal data interface.



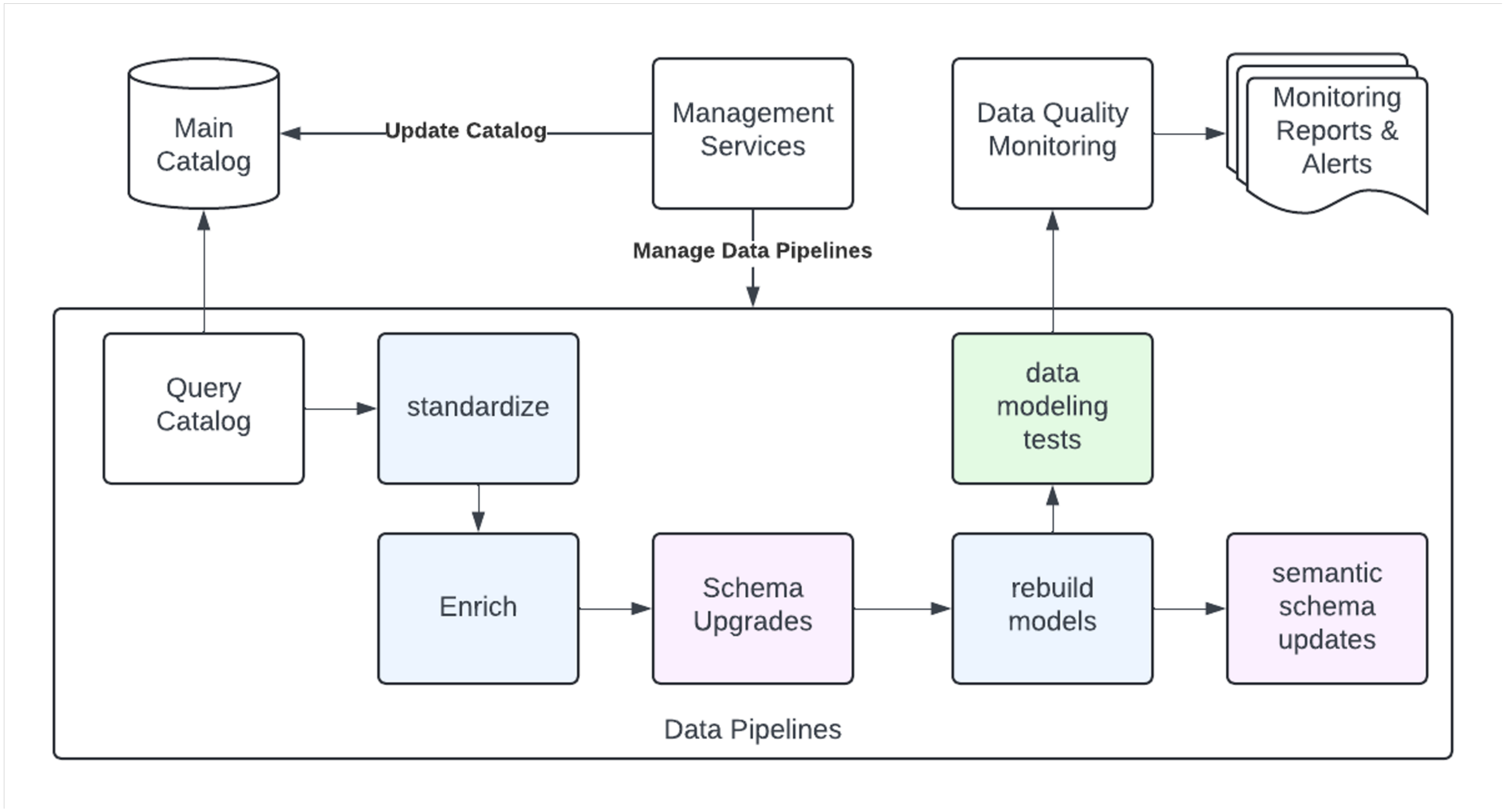
Data engineering and governance at scale: first attempt



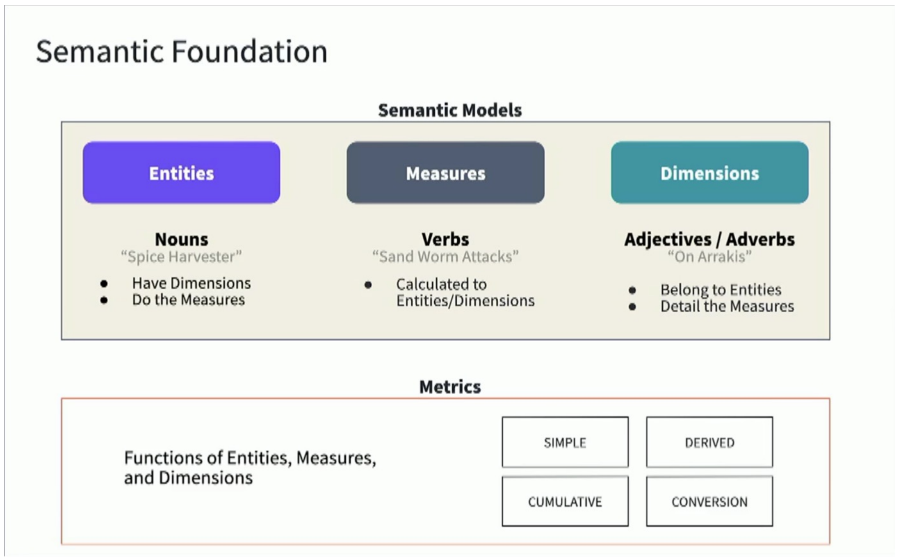
Data engineering and governance at scale



Data engineering and governance at scale



Data engineering and governance at scale: semantic models



DBT Semantic Layer

```yaml
semantic_models:
  - name: transaction # A semantic model with the name Transactions
    model: ref('fact_transactions') # References the dbt model named `fact_transactions`
    description: "Transaction fact table at the transaction level. This table contains or
    defaults:
      agg_time_dimension: transaction_date

    entities: # Entities included in the table are defined here. MetricFlow will use thes
      - name: transaction
        type: primary
        expr: transaction_id
      - name: customer
        type: foreign
        expr: customer_id

    dimensions: # dimensions are qualitative values such as names, dates, or geographica
      - name: transaction_date
        type: time
        type_params:
          time_granularity: day

      - name: transaction_location
        type: categorical
        expr: order_country

    measures: # Measures are columns we perform an aggregation over. Measures are inputs
      - name: transaction_total
        description: "The total value of the transaction."
        agg: sum
```

Data engineering and governance at scale: semantic models

- Semantic Layers enable declarative language based definition of the catalog.
- Semantic layers technologies do not require moving data around. All the processing is on the semantic layer itself.
- Semantic graphs provides a one stop shop for data.
- Most semantic layer technologies support model versioning.
- Data governance first.

Data engineering and governance at scale: key takeaways

- Data models should be powered by a flexible catalog
  - Adding new metrics should be as easy as updating the catalog.
  - Automated catalog updates as part of software releases.
  - Use catalog technologies with atomic updates (such as Apache Nessie for lakehouses).
- Identify different data tiers and govern them accordingly.
  - Different tiers should meet certain data quality standards (if it is in a gold tier, you can trust it).
  - Higher tiers should use relational technology for ease of use.
- Data model governance
  - Version your data models.
  - Create anti corruption layers.

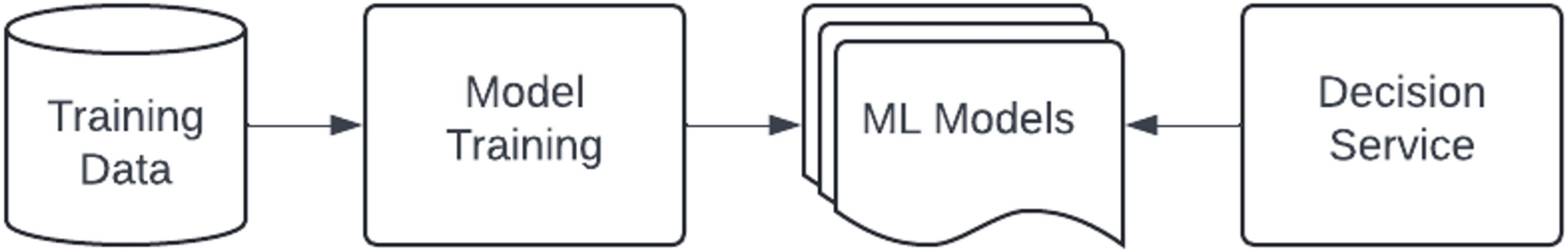
Data engineering and governance at scale: key takeaways

- Data model runtime
  - Favor technology agnostic data models (such as DBT).
  - Favor models that doesn't require moving data around (such as DBT Semantic Layers).
  - Manage the balance between maintenance and efficiency (Snowflake vs Lakehouse)
- Data orchestration
  - Follow well established data handling practices.
  - Use technologies that is appropriate to your team (Airflow vs Mage vs Perfect vs Off-the-shelf ETL tools).
  - Automation orchestration management either through management services or infrastructure-as-code.

Data engineering and governance at scale: objectives

- ML Model training at scale.
- Managing training assets.
- Integration with engineering components.
- ML model tracking: versioning, tagging, and lineage.

Continuous model delivery at scale: first attempt



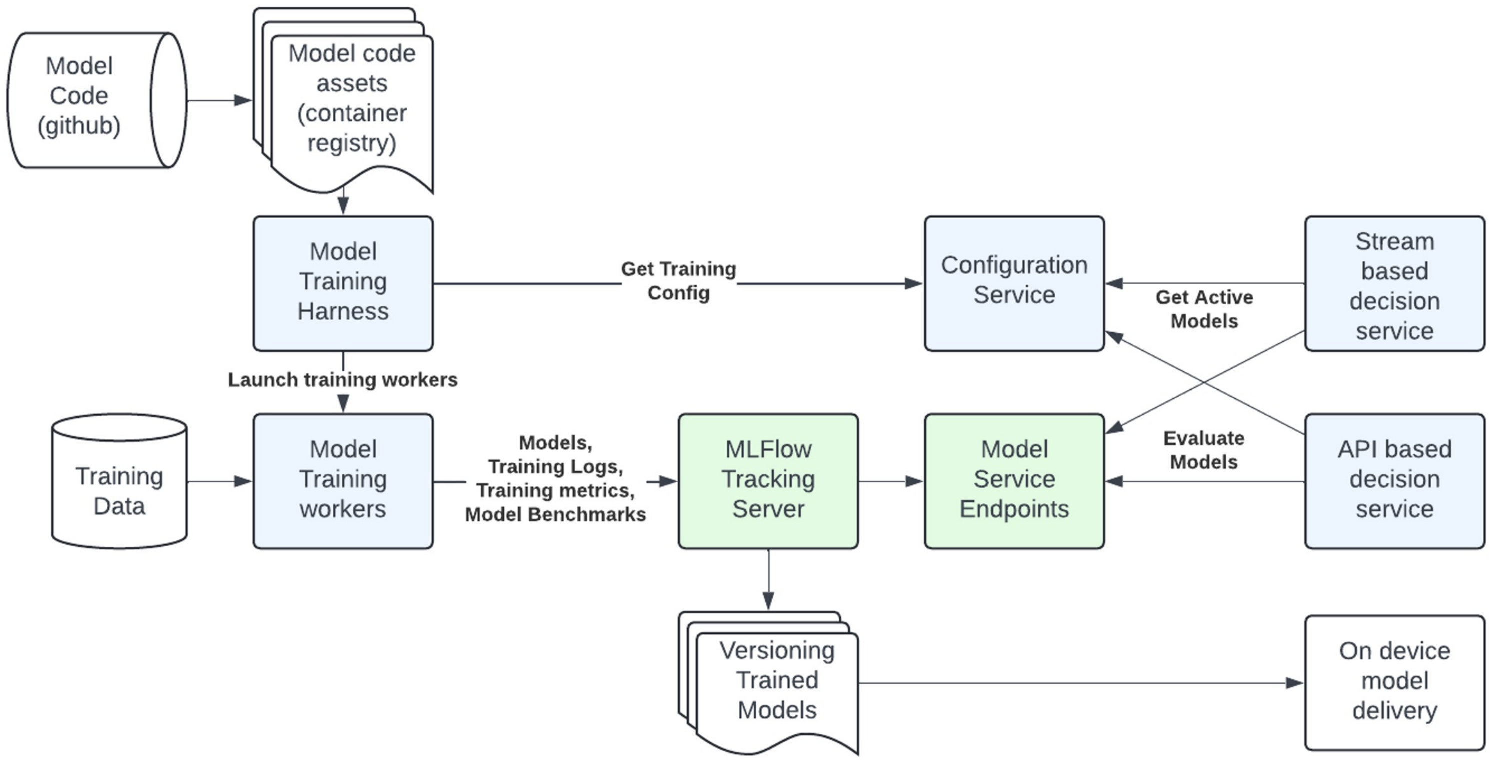
Continuous model delivery at scale: challenges

- Requires deliberate integration with engineering components.
- Real-time model evaluation is different from batch model evaluation.
- Safe model delivery requires model versioning
- Without a specially built model registry, model assets (config, binaries, dataset for training, model benchmarks) might be stored in different systems.

Continuous model delivery at scale: tooling

- Has the most well developed tools (SageMaker, Google Vertex AI, Azure ML, MLFLow, Comet ML ...etc).
- No tools for handling stream based real-time decision making systems.
- Safe delivery requires an easy to use configuration service.
- Continuous model delivery at scale requires a custom built training system (not necessary from scratch) that matches the software architecture.
- An important part of the overall software architecture

Continuous model delivery at scale



Continuous model delivery at scale

Name
..
sql
__init__.py
credentials.example.json
data.py
main.py
model.py
parameters.py

```python
def prepare_data(parameters: dict, snowflake: Snowflake) -> DataFrame:
    """
    Prepare data for training the model. The prepared data will be passed to the train_model function
    :param parameters: Training parameters as passed by the ML automation system from the model confi
    :param snowflake: Snowflake connection object for loading data
    """
```

```python
parameters: Dict[str, Any] = configuration.get_training_parameters(host, model_id)
logger.info(f"Loaded training parameters: {parameters}")
snowflake = Snowflake(configuration.get_secret(parameters.pop("credentials")))
module = parameters.pop("module")  # only needed for module import
path = parameters.pop("path")  # only needed for artifact location
if path is None:
    path = model_id.replace(":", "/")
model_name = f"{model_id}-{host}"
artifact_location = f"s3://{os.environ['MLFLOW_ARTIFACT_BUCKET']}/{path}/{host}"

with start_run(
    run_name=datetime.now().isoformat(timespec="seconds"),
    experiment_id=get_experiment_id(model_name, artifact_location),
) as run:
    log_params({"config_" + inflection.underscore(key): value for key, value in parameters.items()})
    try:
        data = import_module(f"{module}.data").prepare_data(parameters, snowflake)  # prepare training data
        model = import_module(f"{module}.model").train_model(data)  # train model
        metrics = import_module(f"{module}.model").evaluate_model(model, data)  # evaluate model
        logger.info(metrics)
```

Continuous model delivery at scale: key takeaways

- Use a model registry. Many tools exist.
- Integrate the model registry with the rest of your infrastructure. Model registry is not just for data science.
- Manage model training use a centralized a configuration service.
- Use model service endpoints to separate prevent leakage of ML libraries in engineering components.
- Prevent ML systems from direct access to transactional data. All data updates should be through engineered services.
- Establish a clearly defined boundary between ML code and other code.

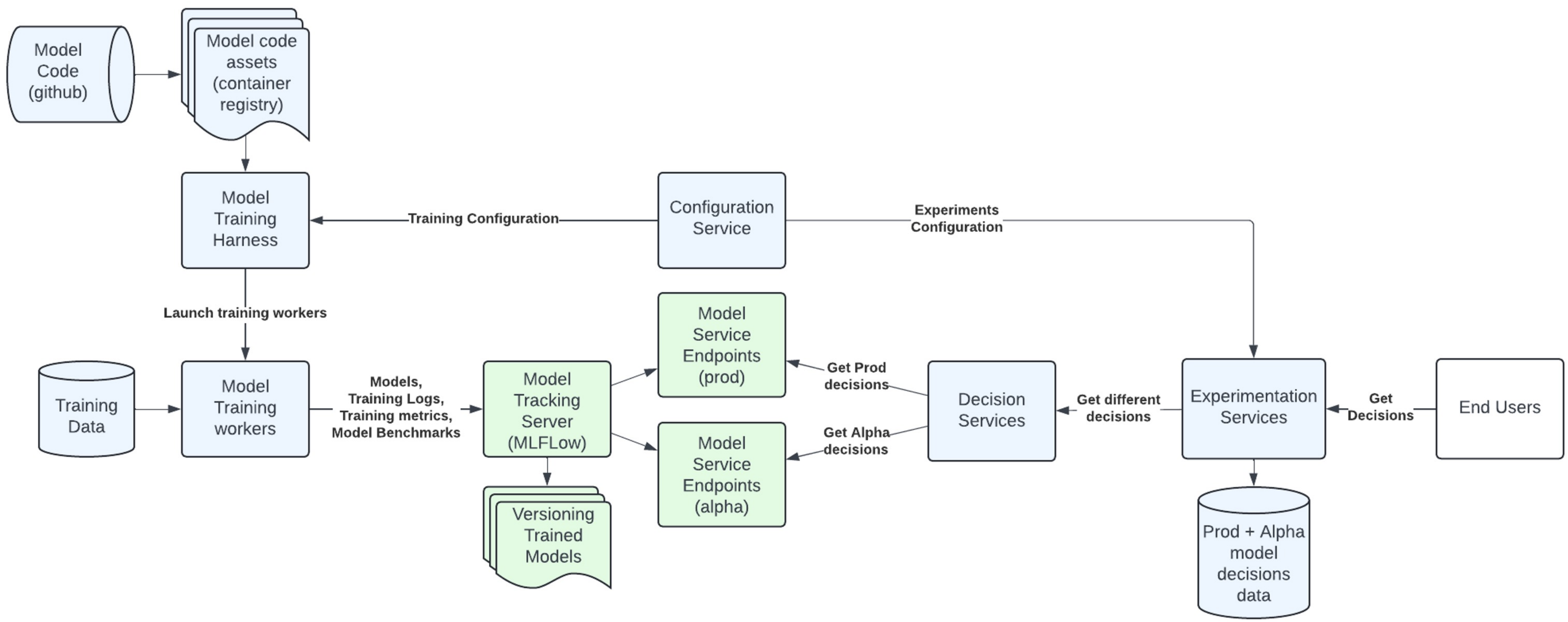
Experimentation and model control at scale: objectives

- Darkmode testing
- Managing alpha and production models.

Experimentation and model control at scale: challenges

- Live experimentation requires changes to the architecture.
- Safe experimentation requires an easy to use configuration system.
- Safe experimentation requires an easy to change policy.
- Assessing experimentation results requires collection of model performance data and integrating it with environmental data (such as the case of visitor behavior in the Paywall case).
- No off the shelf tools for dark mode experimentation.

Experimentation and model control at sale



Experimentation and model control at sale

- Alpha models are delivered similar production (stable) models.
- Experiments are controlled using easy to use configuration services.
- Experiments are managed safely using integrated services.
- Experiments data are collected for later analytics.

Key takeaways

- Simple model delivery and operations require good architecture.
- MLOps is all about extending good software engineering practices to AI systems.
- Current MLOps tools are limited and require many inhouse built systems for full coverage of ML operations.
- Model governance requires a well throughout data engineering practices and tools (Universal model specifications, platform agnostic data modelling, semantic layers and a dynamic catalog).
- A model training harness is essential for ensuring correctness, efficiency, and maintainability

Key takeaways cont.

- Many tools exists for supporting continuous model delivery (SageMaker, Azure ML, Google Vertex AI, MLFlow, comet ML, ...etc).
- Model delivery still requires inhouse components for scaling model training.
- Darkmode experimentation is essential for rapid ML model development.
- Safe experimentation requires a good integrated architecture.