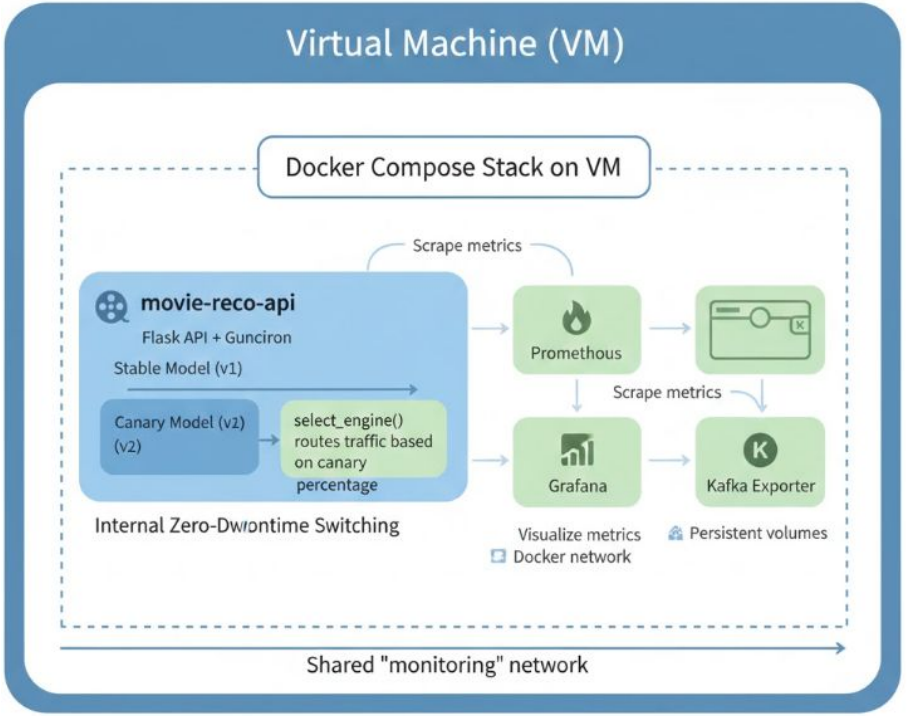
Milestone 3

Eric Qiu, Harry Hong, Jessica Ojo, Samuel Ha

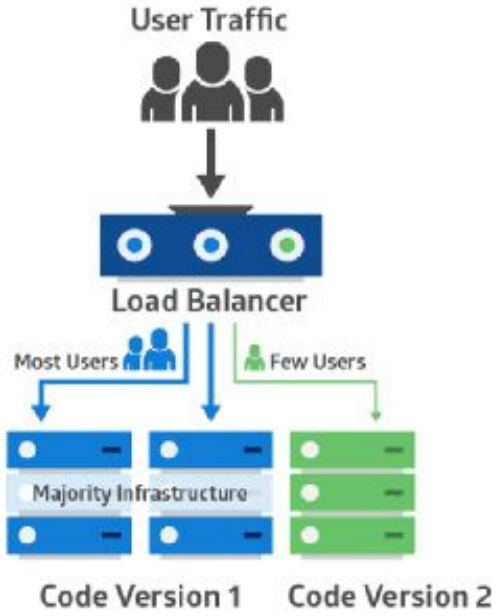
Containerization

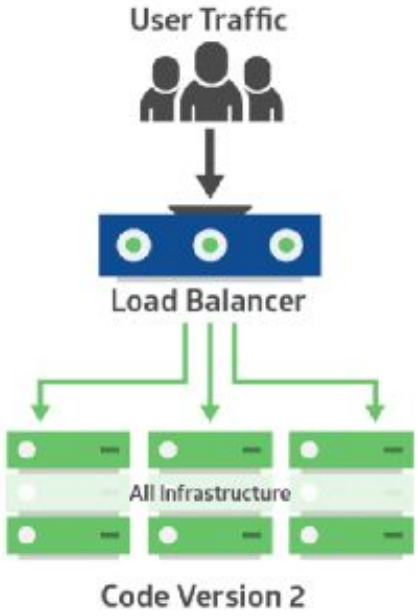
- Single docker image for Inference api
- Docker Compose:
  - API
  - Prometheus
  - Grafana
  - Kafka exporter
- Shared “monitoring” network → metrics exchange internally



Canary Release

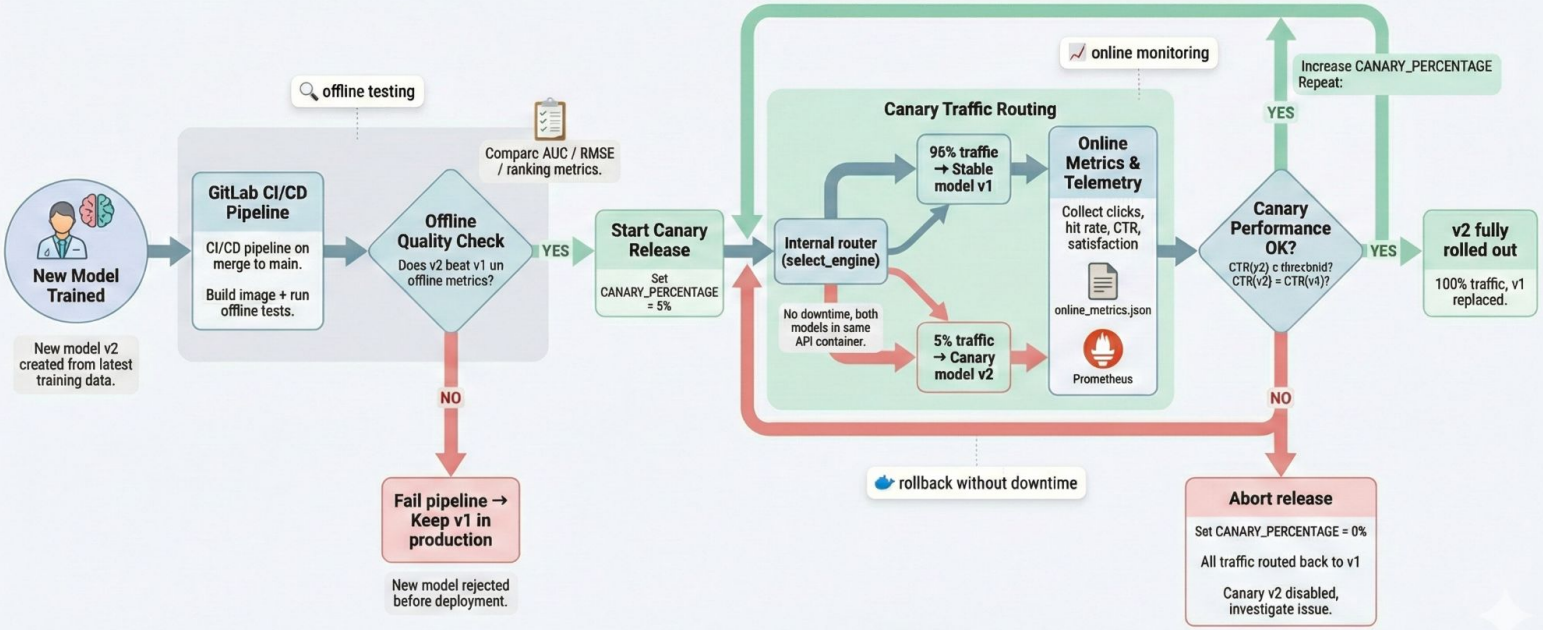
- Deploy both models inside same container
- Selection
- Canary percentage stored in config.json
- Logs model versions
- Fail/Success (Metrics)
  - V1 vs V2





Canary Release Workflow

Utilizing Docker, GitLab CI/CD, and Canary Release Process



Offline Analysis: Feedback Loops

Feedback Loop 1: Relationship between popularity and interactions

- Measure how strongly movie popularity correlated with the number of user interactions.
- Popularity–interaction correlation: 0.24 (Moderate)
- Top 10% of movies receive **~50%** of all interactions
- Risk of popular items receive more visibility
- Can help system quickly converge on popular items

Feedback Loop 2: Genre diversity per user

- Users tend to interact with about **3** genres
- 10th percentile = **0 genres**
- Risk: large amount of users have very narrow genre preference
- Model can keep recommend those genre without exploring new categories

Detect using Training Data


```
"popularity_stats": {
  "n_movies": 21941,
  "corr_popularity_interactions": 0.2441683074559409,
  "top_share_fraction": 0.5009800753904147,
  "top_share_percent": 50.09800753904147,
  "top_k_fraction": 0.1
},
```

```
"genre_diversity": {
  "n_users": 42991,
  "mean_genres": 3.1918308483170894,
  "median_genres": 3.0,
  "p10_genres": 0.0,
  "p90_genres": 7.0
```

Offline Analysis: Fairness Issue

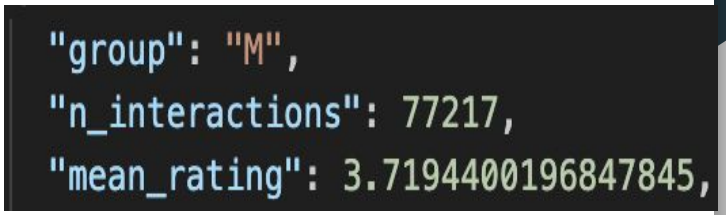
Fairness Issue 1: Training Data Gender Imbalance

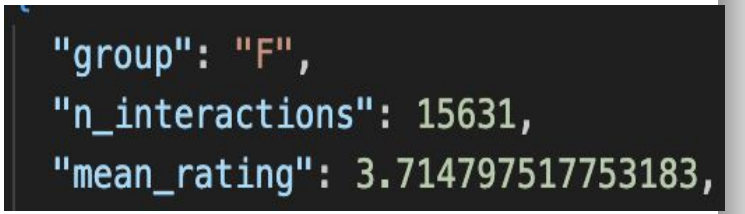
- Male interactions: **77,217** vs. Female: **15,631**
- Prediction quality: higher for males, lower for females.
- Mitigation: Higher training weight for minority group

Fairness issue 2: Popularity Bias Hurting Minority Groups

- If Female users prefer niche genres, popularity bias will cause those to be under-represented
- Female users may see fewer recommendations that match their interests
- Mitigation: exposure constraints such as enforcing minimum exposure for underrepresented genres

Detect using Training Data





Online Analysis: Feedback Loop & Fairness Issue

FeedBack Loop: Click-Through Rate & Hit Rate

- **CTR** measures whether users click (**engagement**).
- **Hit Rate** measures how accurate the recommendations were (**relevance**).
- Both drop when feedback loops occur
- Reason: Popularity and Genre issue can cause stale or narrow recommendations.

Fairness issue: Satisfaction Ratio

- Avg satisfaction rate male / Avg satisfaction rate female
- Ratio = 1 → Perfect fairness
- Ratio < 1 → Male satisfaction < Female Satisfaction
- Ration > 1 → Male satisfaction > Female Satisfaction
- Potential Threshold: less than 0.85 and greater than 1.15
- Provides a simple numerical comparison to identify any gender imbalance

Only Simulated Traffic Used For Testing

Model Retraining

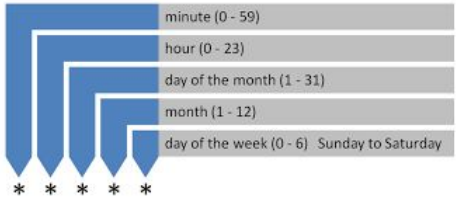
We scheduled **model retraining with CRON.**
DVC is used for data and model versioning

Retraining Rules/Triggers

- Model is updated every 3 days if new data has;
  - < 50 new users info
  - <100 new interactions

We store 3 model versions

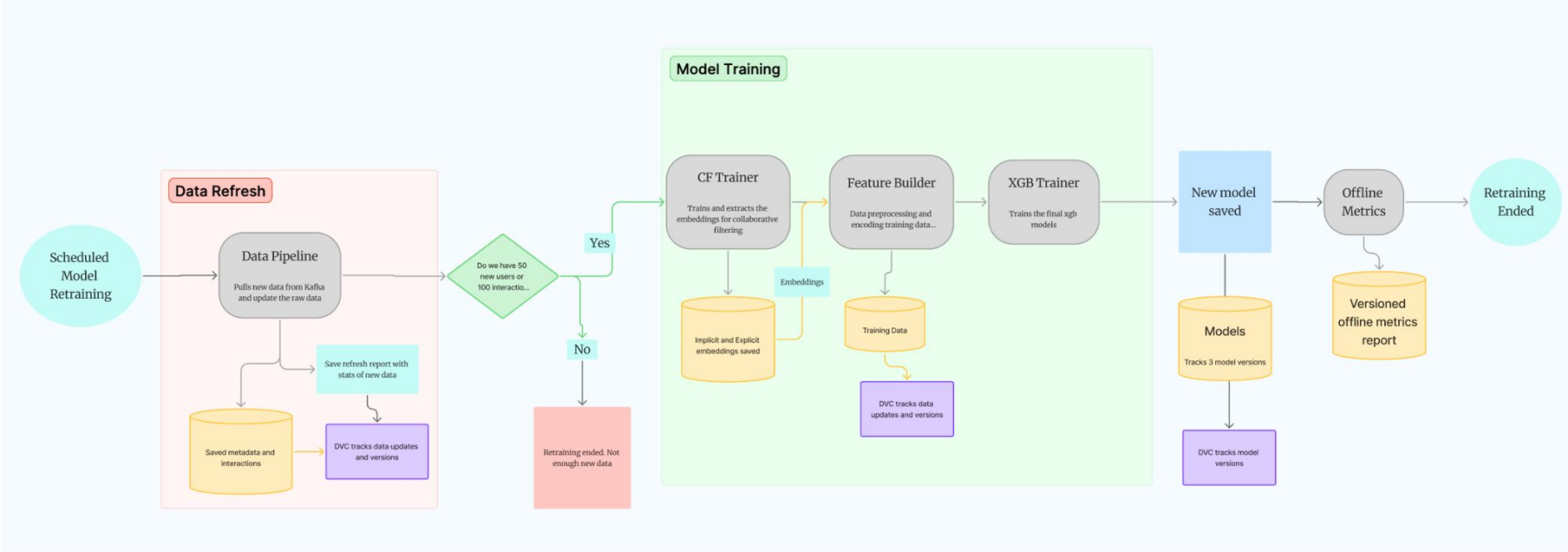
CRON is a time-based job scheduler on Unix-like systems that allows recurring tasks to be executed automatically at specified times or intervals.



DVC by lakeFS

DVC enables Git-based versioning of data and models without committing large files to Git.

Model Retraining Pipeline



Provenance

Prediction provenance is **recorded using lightweight CSV logs** that link each request to a specific model version and training data version, enabling full reproducibility **without database overhead.**

MongoDB —> Over-engineering

Provenance Track:
Pull the logs from canary release

- RequestID
- ModelID+model_version
- Predictions
- Date of Prediction

Model metadata stored separately:

- ModelID
- training method
- data version (DVC reference)
- evaluation metrics
- DVC reference

- Reproducibility
- Debugging
- Performance Tracking
- Separations of concerns

Project Learnings and Reflections

What went well

- Modular set up of the infrastructure
- Detailed tests and monitoring system
- Training algorithm

Major Challenges

- Data limitations
- Cold-start problem
- Updating the input training data and versioning
- Package dependencies, server setup and installations

What can be better

- Data drift aware retraining
- Explore more complex models to improve accuracy
- Address fairness issues and harms

Team Reflections

- Worked in silos - we collaborated and communicated a lot.
- Set up together before we go into the project.
- A lot of extra efforts to learn and implement
- Navigating delays and unexpected circumstances
- Gitlab collaboration significantly improved - merge request reviews.

Questions?