Serving Personalized ML at Scale with Evolving Runtimes

Rayan Daod Nathoo
ML Engineer @ Atinary Technologies

PyData Amsterdam 2026
Friday September 11th, 10:05 - 10:50

ATINARY

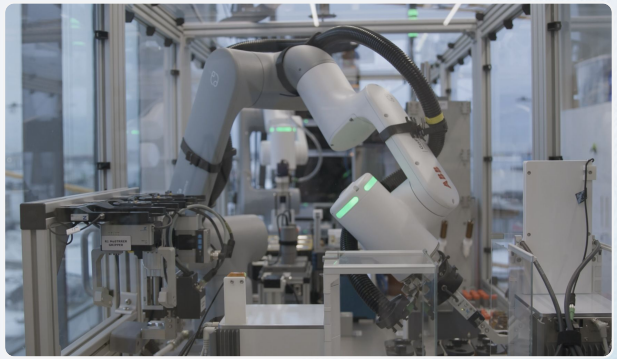
My Intro - Rayan Daod Nathoo

- Machine Learning Engineer in the Research team
- MSc. in Computer Science & ML @ EPFL
- Previous experience in ML Research, Software Engineering
- More information: rayan.daodnathoo.com

Our Company - Atinary Technologies

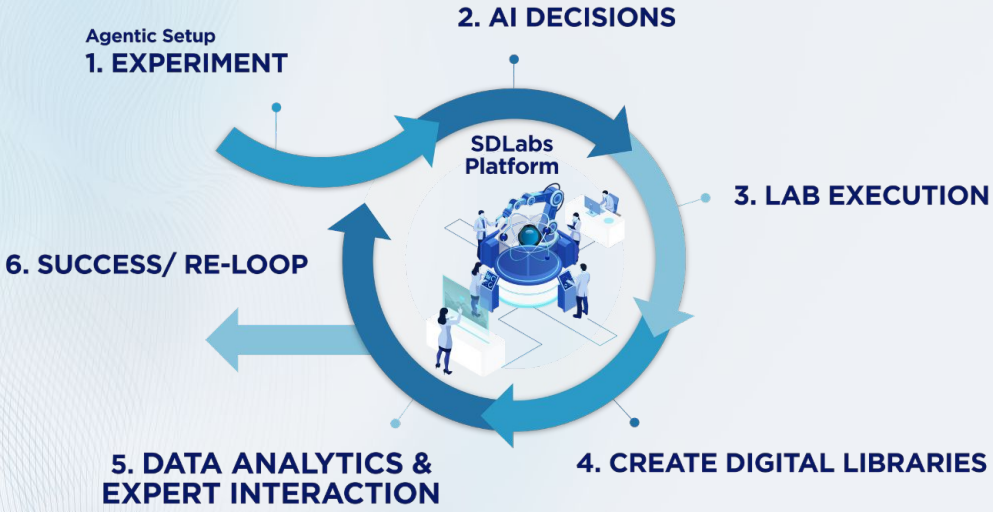
- Swiss-American startup founded in 2019
- Augmenting scientists with Physical AI and Robotics to accelerate R&D
- Multi-disciplinary team - ML, Chemistry, Software, Lab Integration, …

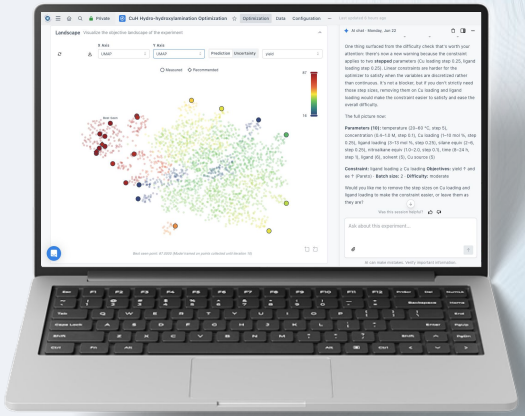
US & Swiss based team



Autonomous Data Factory

Our Software – SDLabs





*Design, Make, Test, Analyze, Learn

Typical Data & Models

base	base_equivalents	ligand	Pd(dba)2_loading	reaction_time	solvent	temperature	imine_yield
K3PO4	2.0	SPhos	5.0	2.0	1,4-dioxane	100.0	75.52
K3PO4	3.5	BrettPhos	1.0	2.0	1,4-dioxane	100.0	100.0
K2CO3	2.0	SPhos	1.0	2.0	DMF	100.0	0.0
K3PO4	2.0	Xantphos	5.0	2.0	DMF	100.0	0.0
Cs2CO3	2.0	SPhos	5.0	2.0	DMF	100.0	0.0
K3PO4	2.0	SPhos	5.0	2.0	DMF	100.0	54.70

Tabular data example

We want to save and serve them!

Numerical features →

Categorical features →

→ Numerical outputs

Model signature

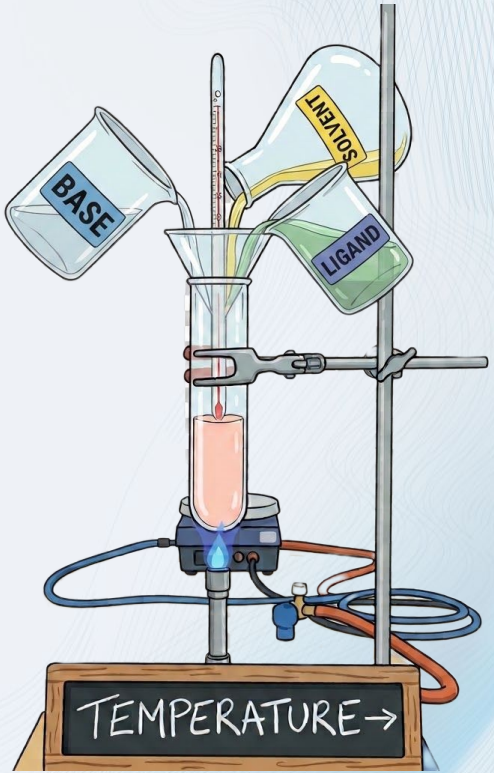
Motivations

Pointwise explainability

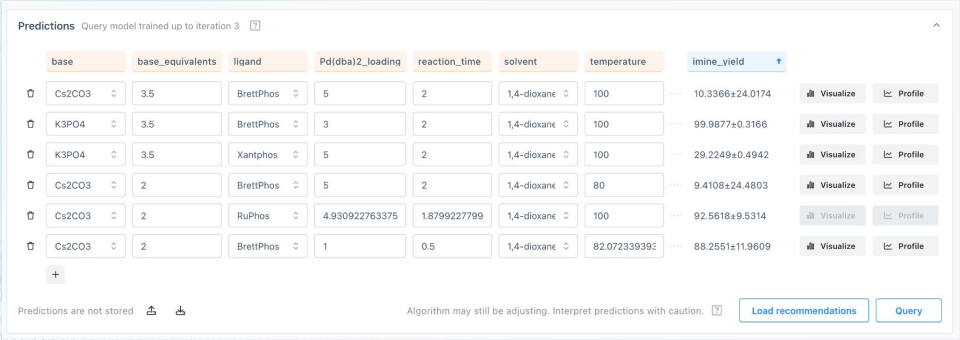
- *What does the model think about that recommendation?*
- *What about that other point I had in mind?*
- *How uncertain is the model?*
- *Is the optimization algorithm currently exploring or exploiting?*

Visualizing models' internal beliefs

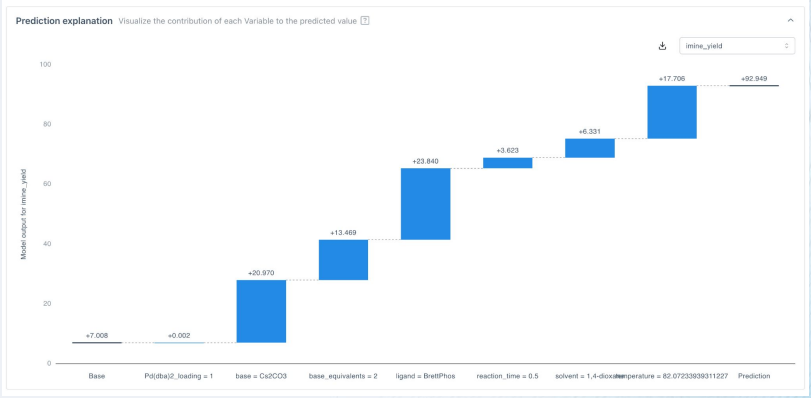
- *Feature importance* (e.g. with Shapley values)
- *Region robustness* (e.g. with predictions around a specific point)
- *Overall model beliefs* (e.g. with predictions on random points sampled uniformly)



Spoiler Alert!

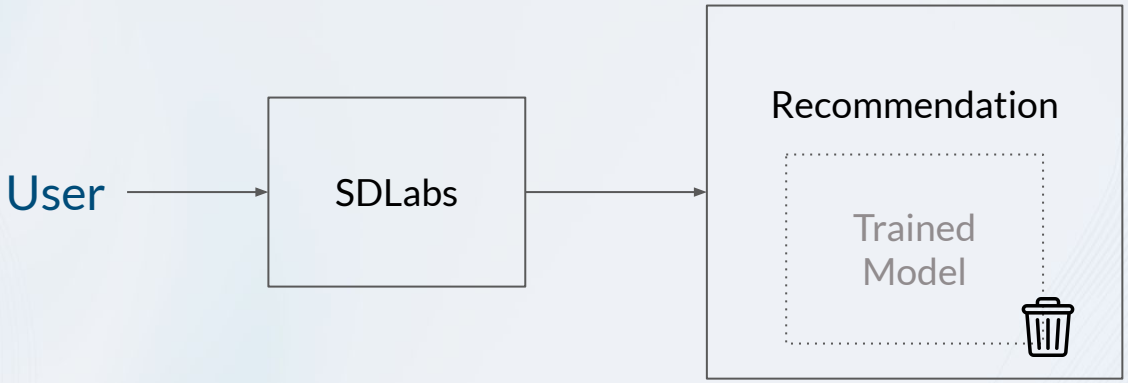


Pointwise predictions in **less than 1 second**



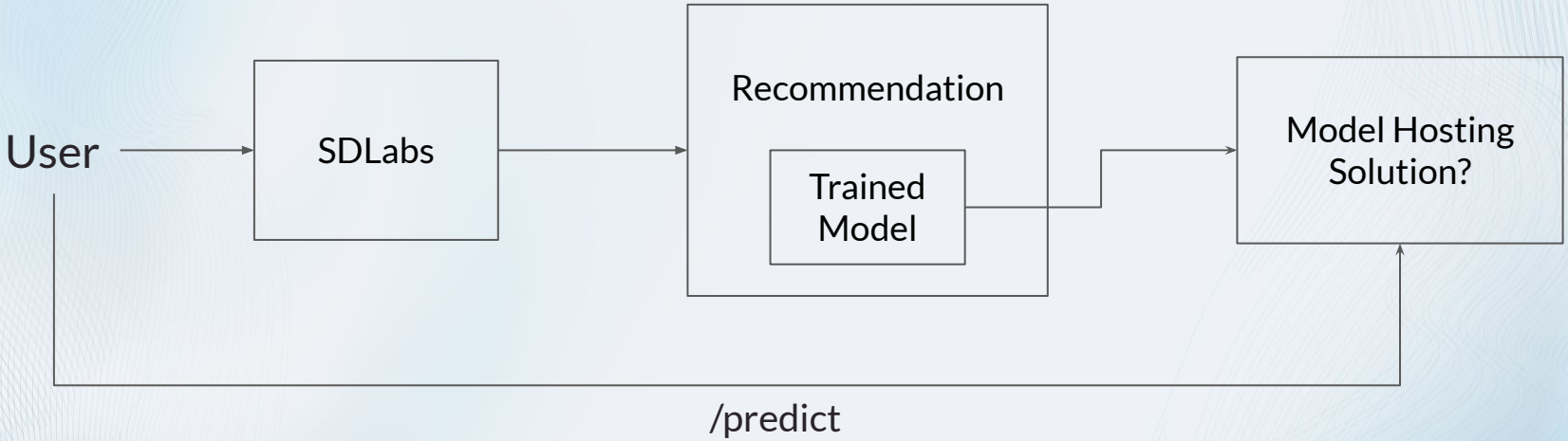
Thousands of predictions in a **few seconds**

We want to go from this...



Old system - discarding the trained models

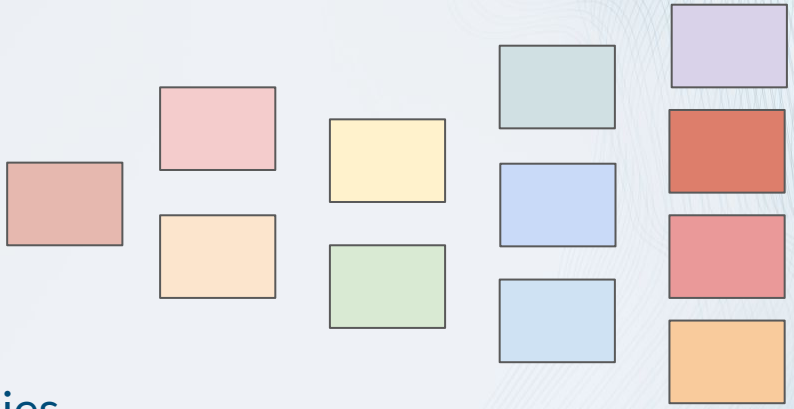
… to this!



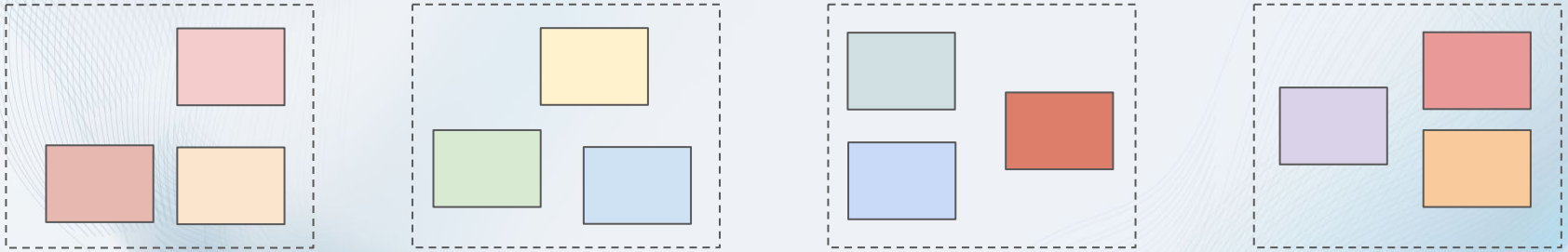
Conceptual goal: serving the trained models

Two Main Challenges

1. **Multiple** *unique* models per user



2. Models can have <u>different dependencies</u>



Technical Wishlist

Multiple *unique* models per user

Low latency pointwise predictions (< 1s)

Batched predictions (up to thousands at a time)

Models trained with **past algorithm versions** should remain available

Easy **adaptability** to **future model architectures** and signatures

Great **developer experience**

This talk is aimed at any team with similar needs!

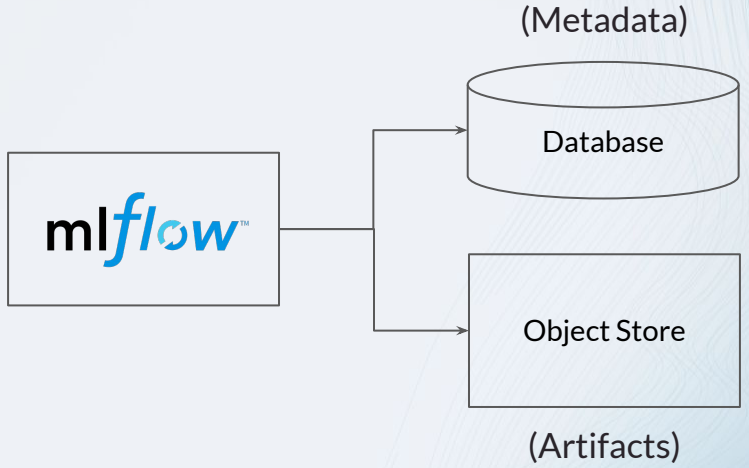
Agenda

1. Hosting a Single Model
2. Challenge 1 - *Multiple unique models per user*
3. Challenge 2 - *Models can have different dependencies*
4. End-to-End Pipelines
5. Local Development
6. Conclusions & Takeaways

Hosting a Single Model

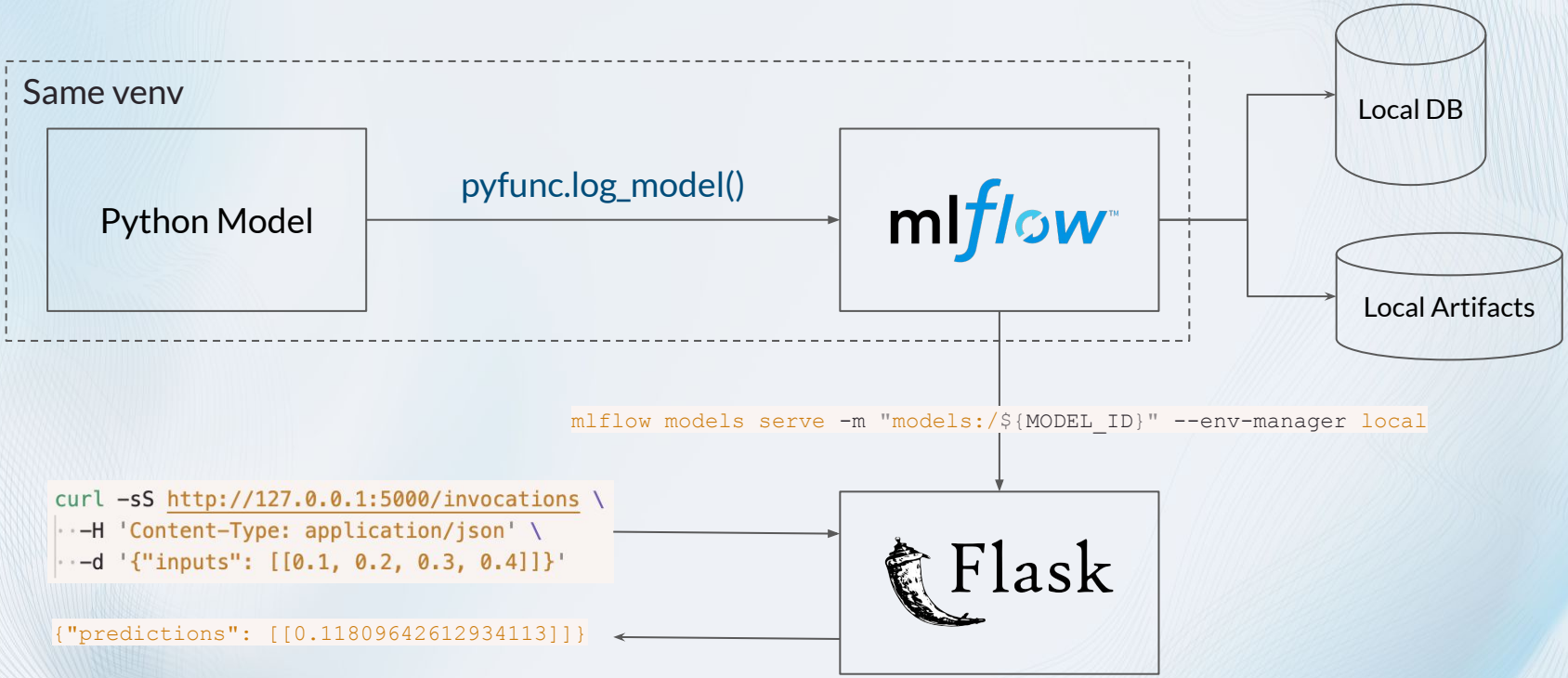
MLflow as Model Hosting Tool

- **Open source** and **actively maintained**
- Used in **bigger projects**
- Well **documented**
- Model **hosting and serving**
- **Dependency management**
- Experiment tracking
- Aligned roadmap



MLflow components

Checkpoint

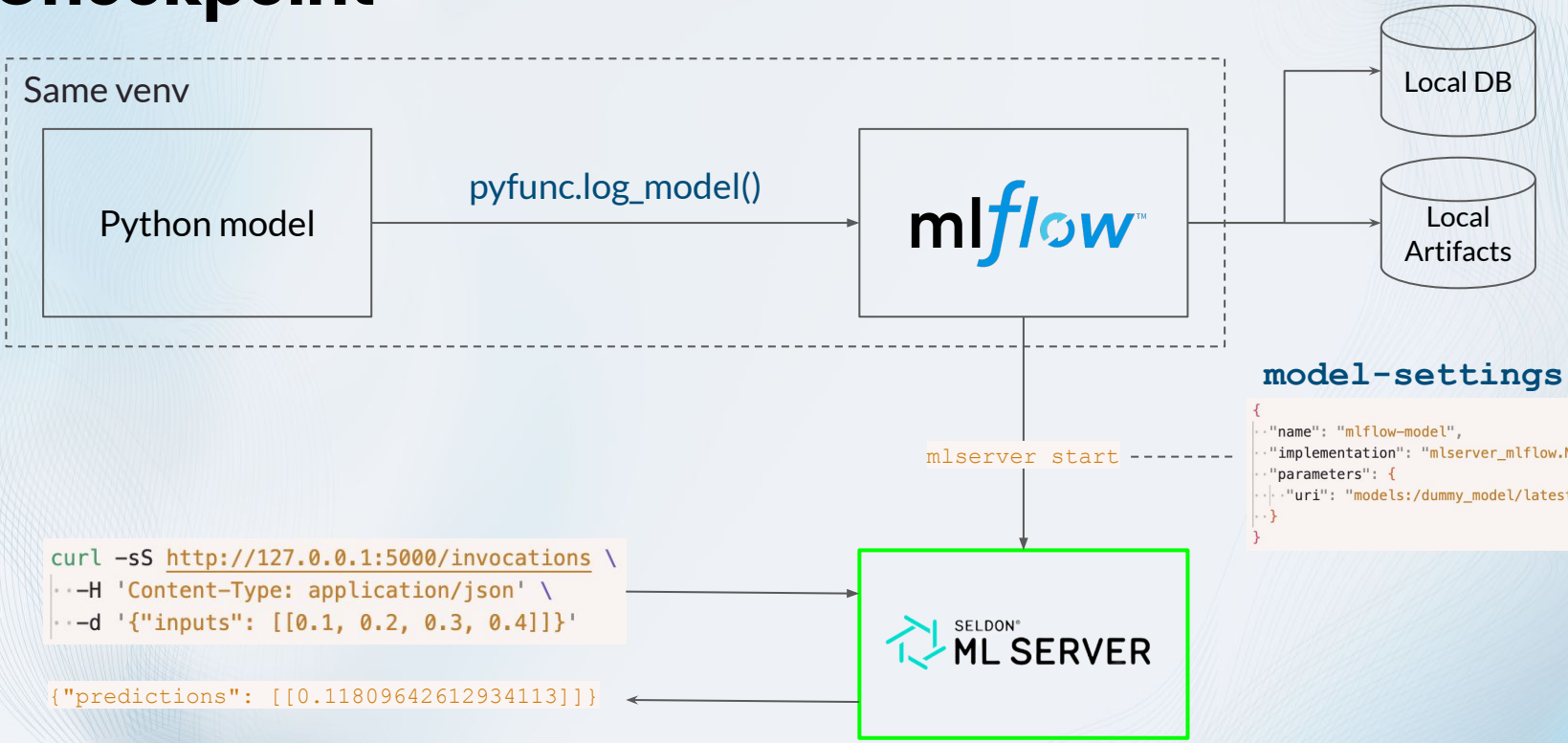


What we have so far

MLServer as Production Serving Engine

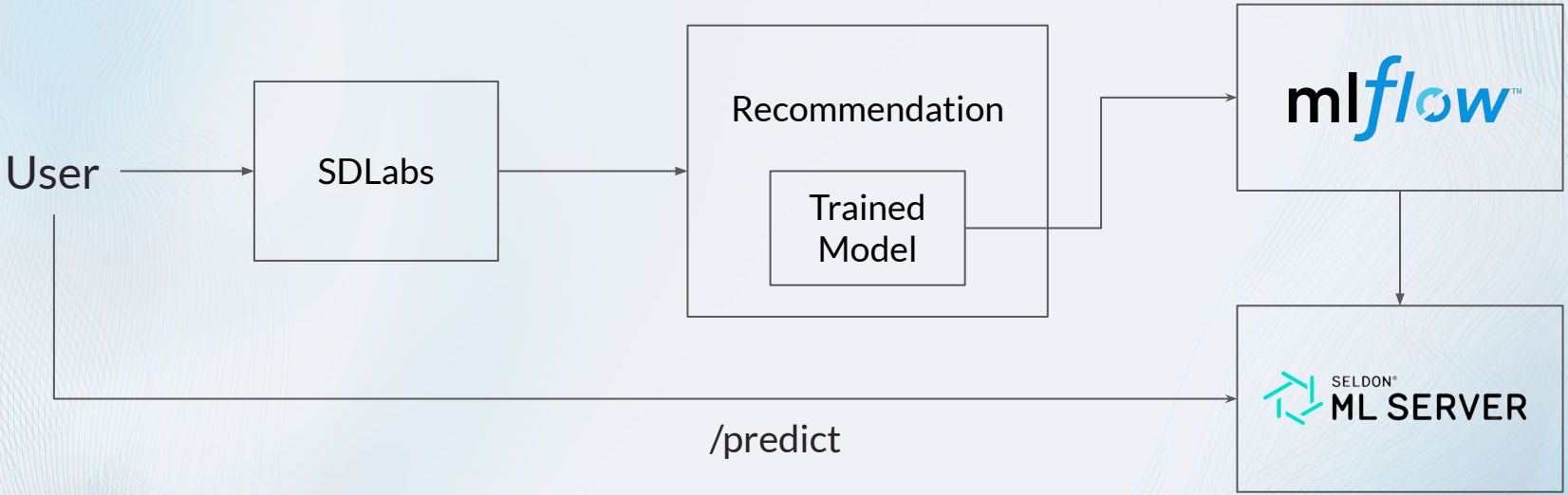
- **Production serving engine** (recommended by MLflow)
- **Asynchronous** requests
- **Parallel workers**
- Core inference server in Seldon Core and KServe
- Adaptive batching

Checkpoint



What we have so far (Flask → MLServer)

Serving a Single Model is Not Enough



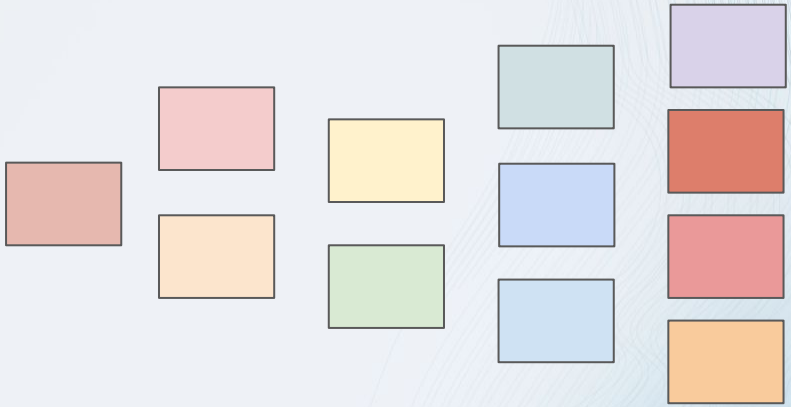
Conceptual goal: serving the trained models

Challenge 1

Multiple *Unique* Models per User

Multiple *Unique* Models per User

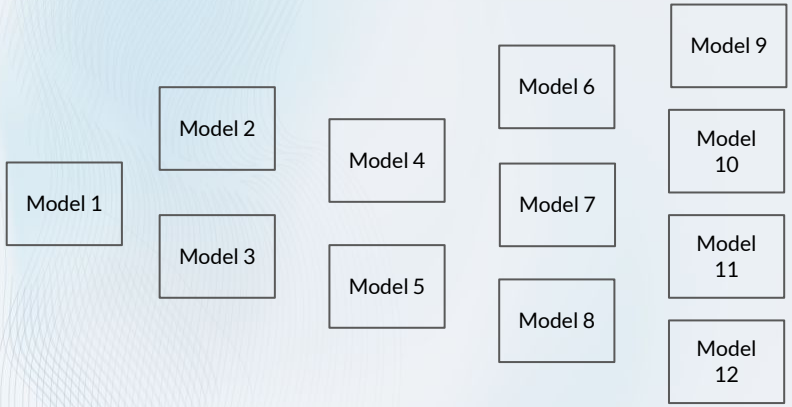
- Many users
- Multiple optimizations per user
- Multiple models per optimization



→ **Increasingly many models to serve!**

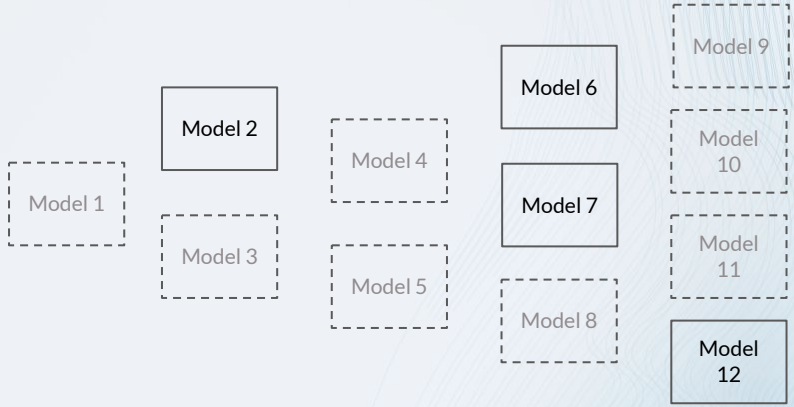
Multiple Models

One distinct server per model?



High infra cost, needs routing, idle time

One server per model with auto-start?



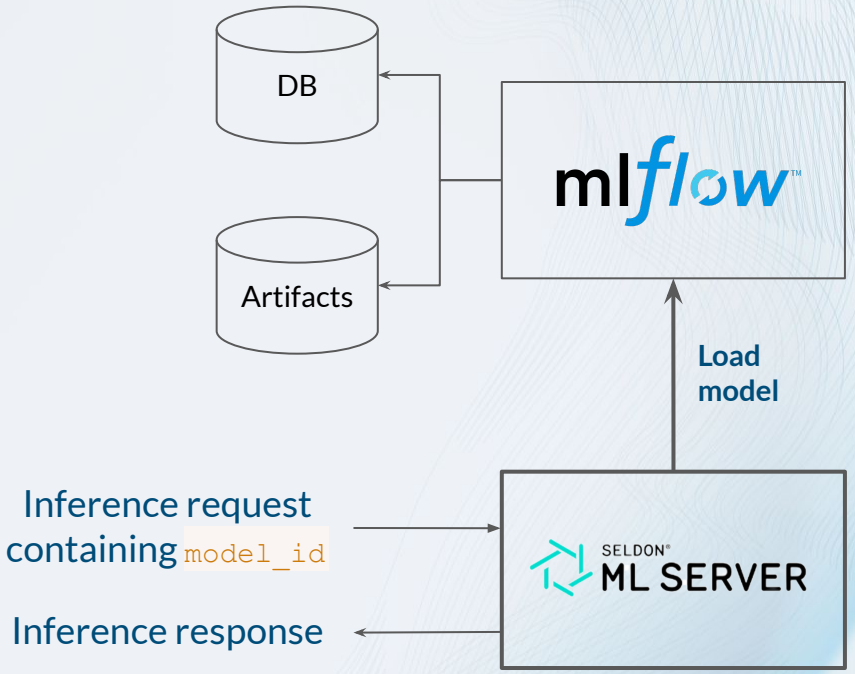
Medium ops cost, needs routing, cold start

🤔 What about a **single server** loading model **before** running prediction?

Multiple Models

Custom runtime in MLServer

- **Include model_id** inside request
- **Load model** at inference time
- **Cache models** for efficiency



Dynamic model loading workflow

Unique Models

Reminder: different number and/or types of inputs/outputs

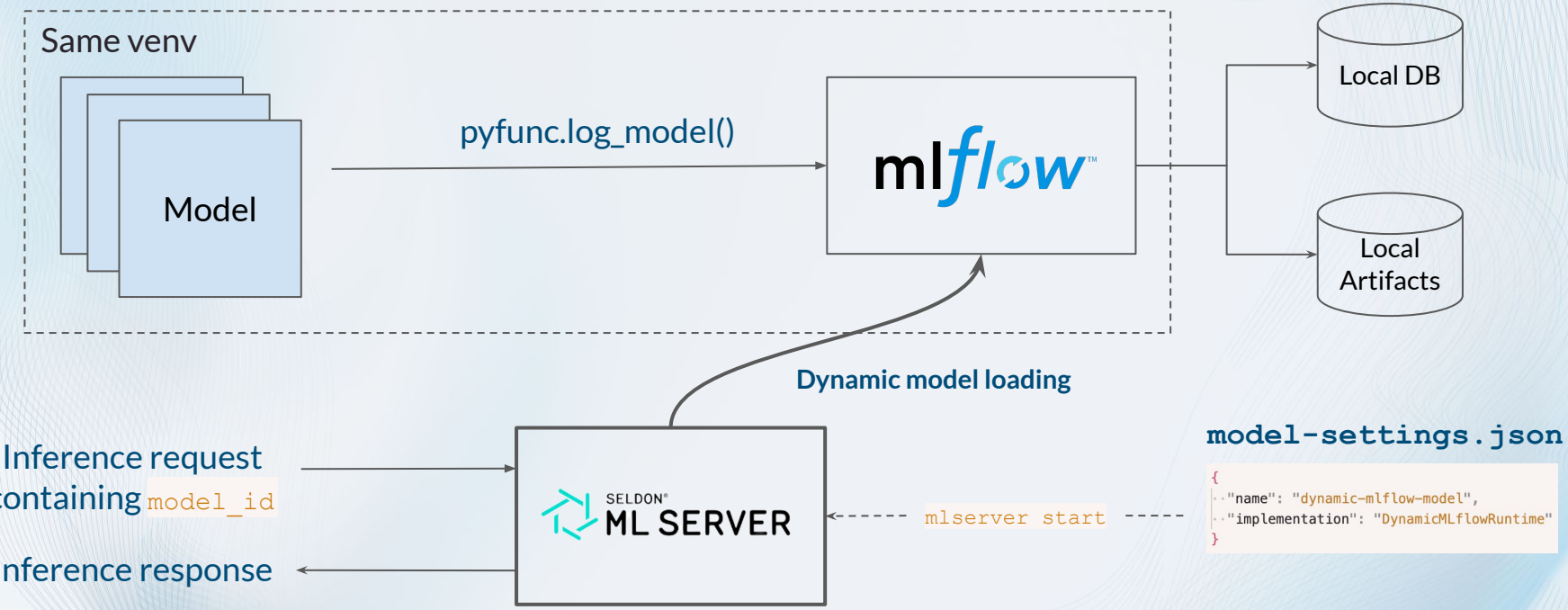
→ **Open Inference Protocol**

Standardized interface for model inference

```
{
  "id": "string", // optional
  "parameters": {}, // optional
  "inputs": [
    {
      "name": "string",
      "shape": [1, 2, 3],
      "datatype": "FP32",
      "parameters": {}, // optional
      "data": [1.0, 2.0, 3.0]
    }
  ],
  "outputs": [ // optional
    {
      "name": "string",
      "parameters": {} // optional
    }
  ]
}
```

Example inference request using the Open Inference Protocol

Checkpoint



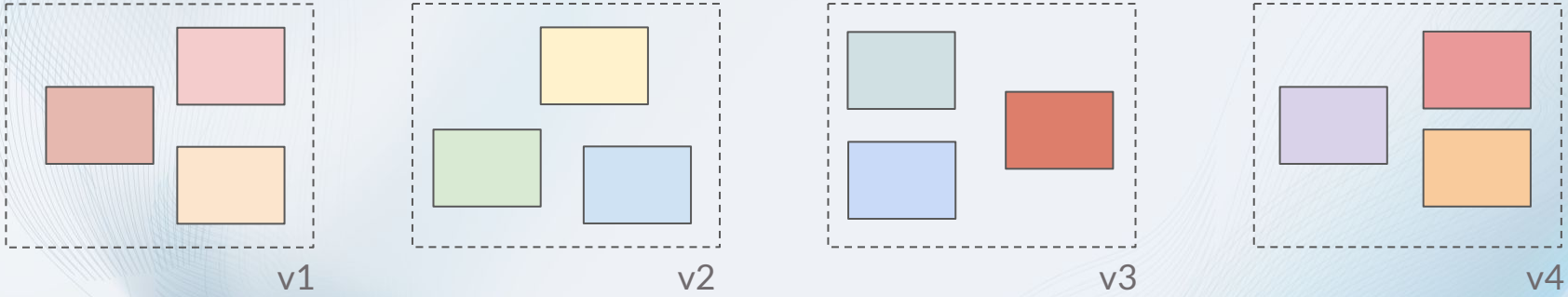
What we have so far (multiple models served!)

Challenge 2

Models can have different dependencies

Evolving Runtimes

- Modeling code changes
- Dependencies are updated
- Dependencies are added



Options Considered

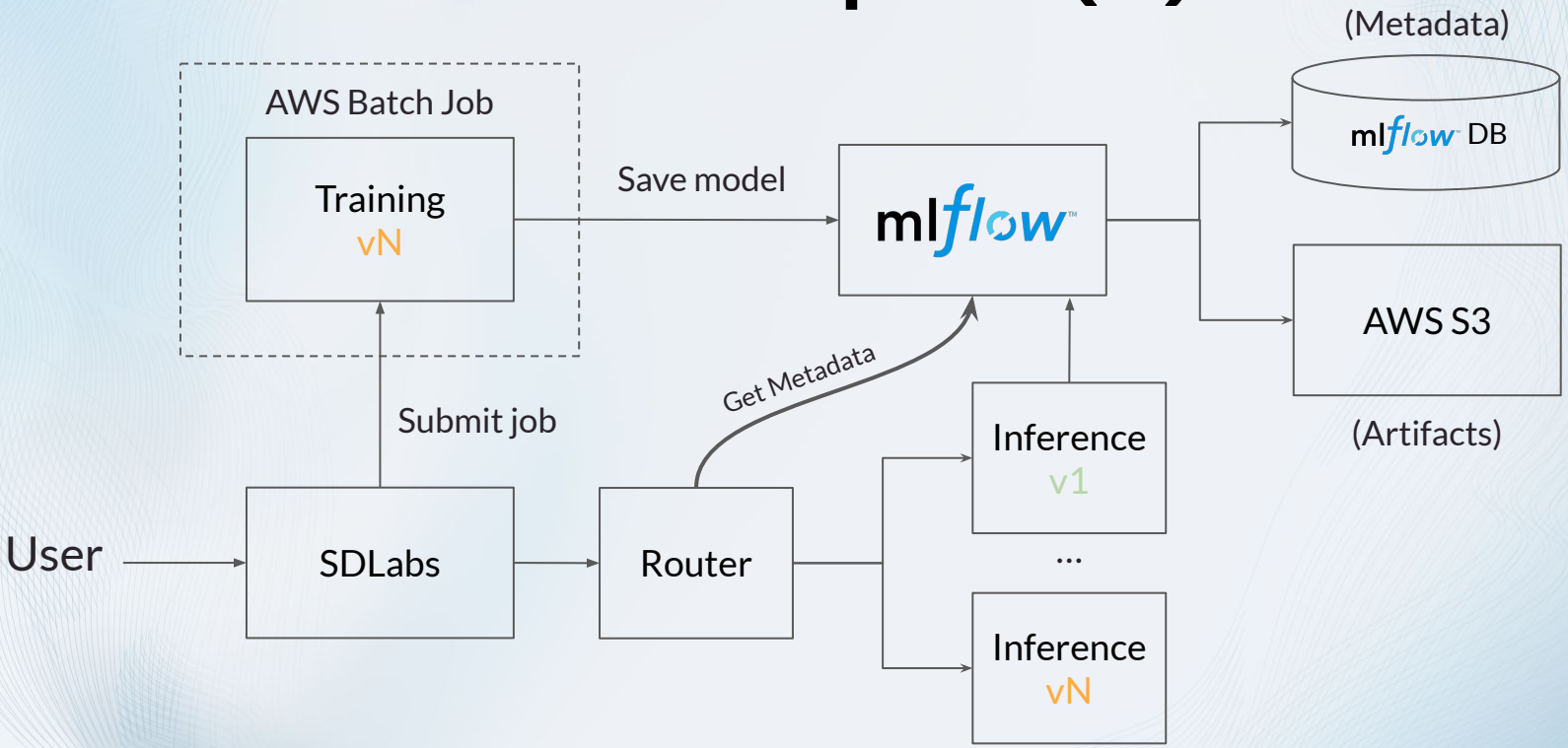
Option	Operational cost	Developer Exp. cost	User Exp. cost
Strict backward compatibility between models	Low	High	None
One inference service per version (always up)	High	High	None
One inference service per version (on request)	Medium	High	Cold start
Create required venv at prediction time	High (custom)	Low	Cold start
Keep N latest versions up 	Medium	Medium	Models expire after some time, then need retraining on latest.

KServe as Infrastructure Serving Engine

- **Production infrastructure** serving engine
- Supports **MLServer** as core inference engine
- Created **Open Inference Protocol**
- **Auto-scaling**
- Inference request **queue**
- **Observability** metric endpoints

End-to-End Production Pipelines

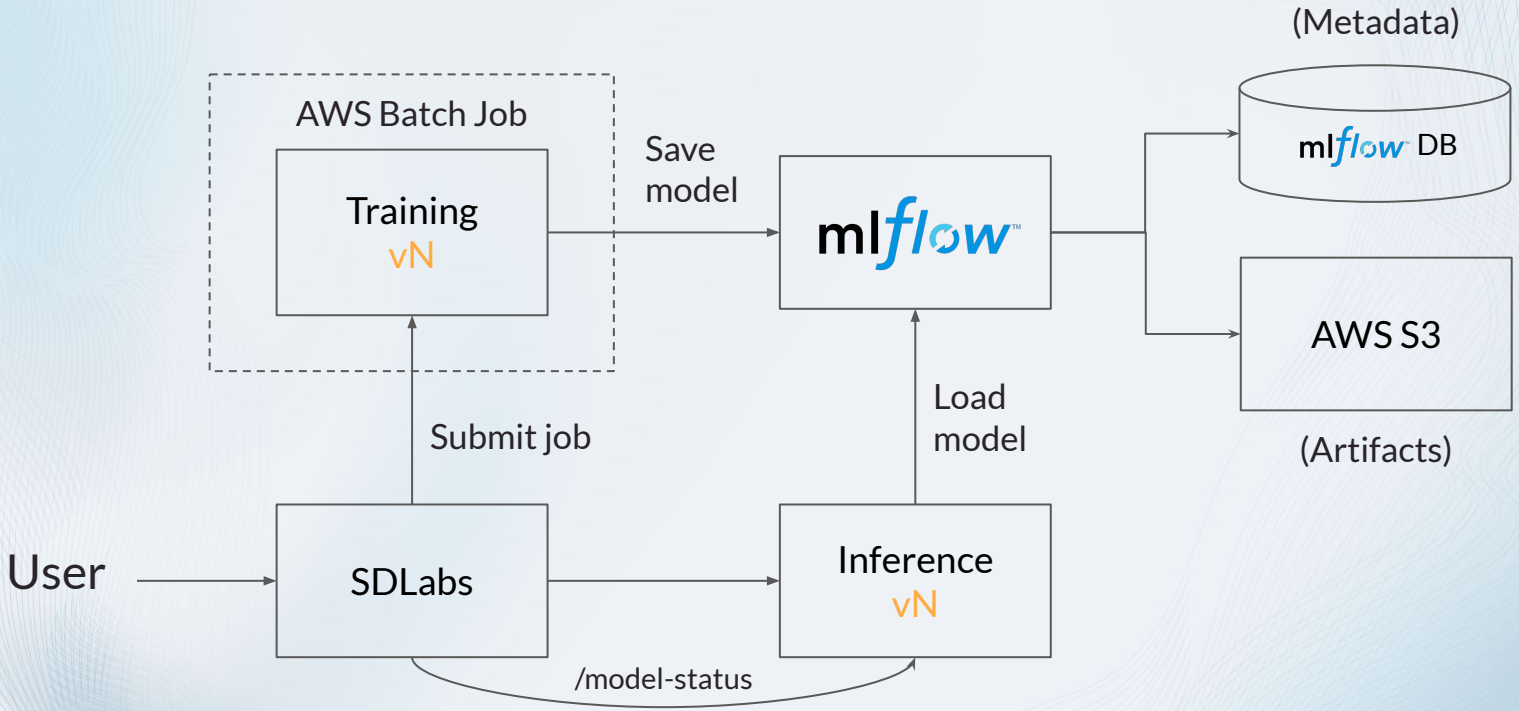
End-to-End Production Pipeline (v1)



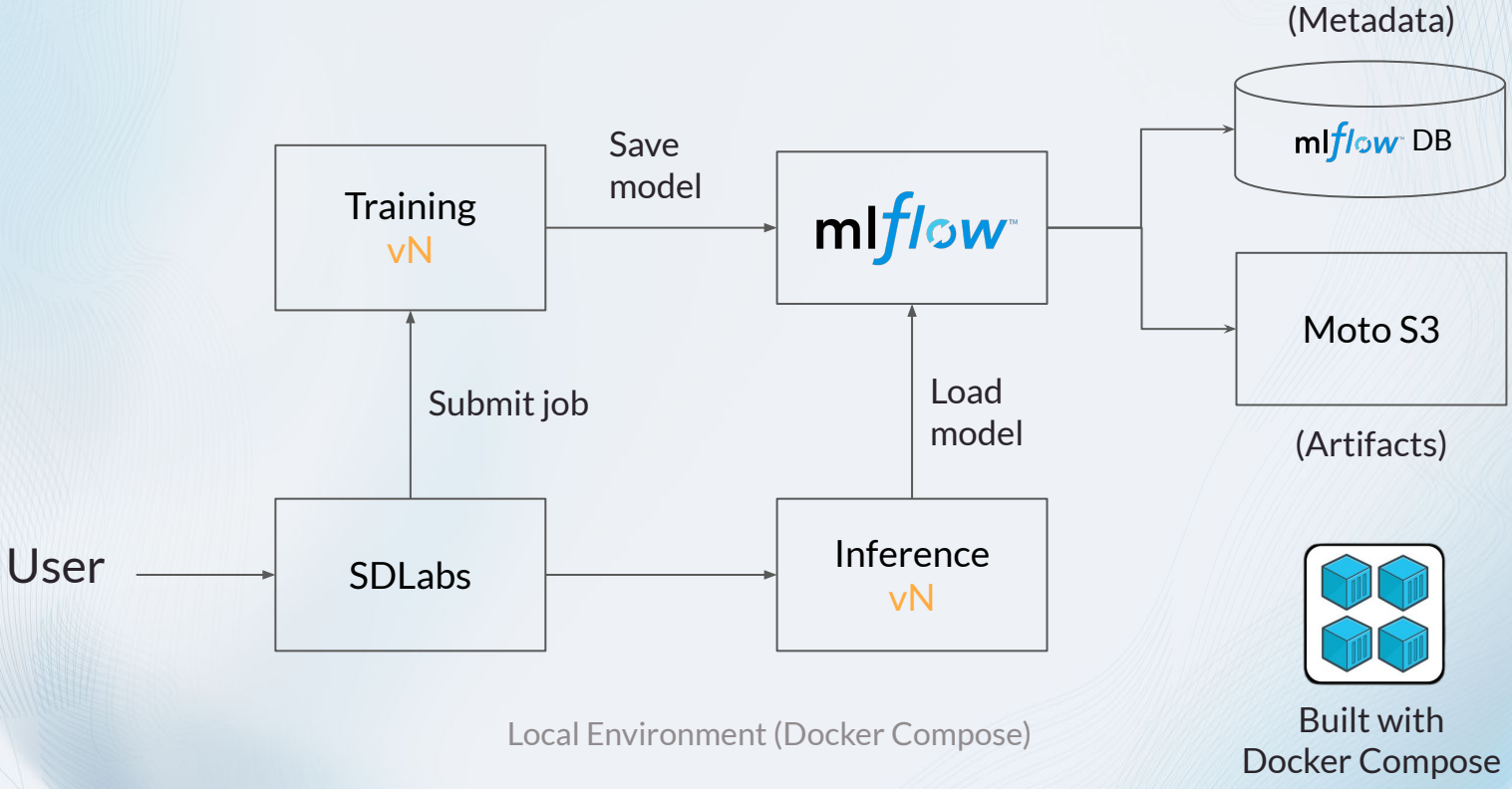
Wishlist Re-Assessment

- Decision to **support only the current model version**
- Leverages **fast retraining**
- Leverages the fact that modeling updates **should only improve** models
- **Eases maintenance** on our end while putting acceptable burden on users

End-to-End Production Pipeline (v2)



Local Development



Checklist

- **Multiple *unique* models** per user
- **Low latency** pointwise predictions (< 1s)
- **Batched predictions** (up to thousands at a time)
- Models trained with **past algorithm versions** should remain available
- Easy **adaptability** to **future model architectures** and signatures
- Great **developer experience**

Checklist

✅ **Multiple *unique* models** per user

✅ **Low latency** pointwise predictions (< 1s)

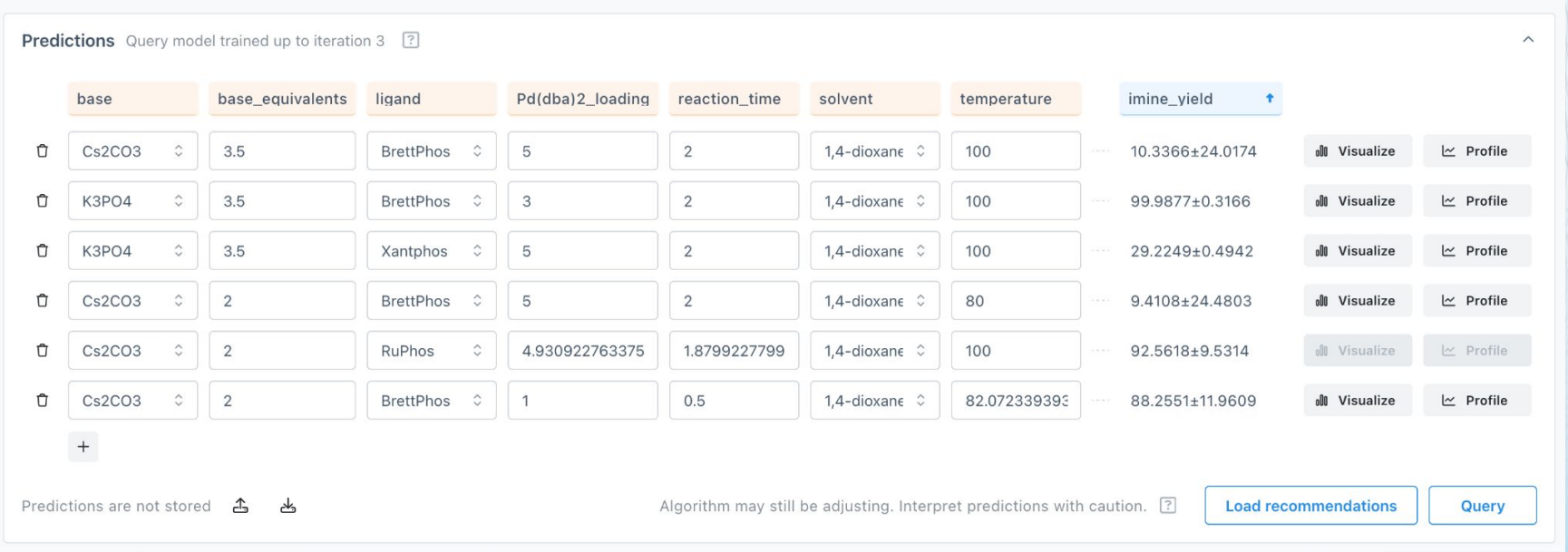
✅ **Batched predictions** (up to thousands at a time)

❌ Models trained with **past algorithm versions** should remain available

✅ Easy **adaptability** to **future model architectures** and signatures

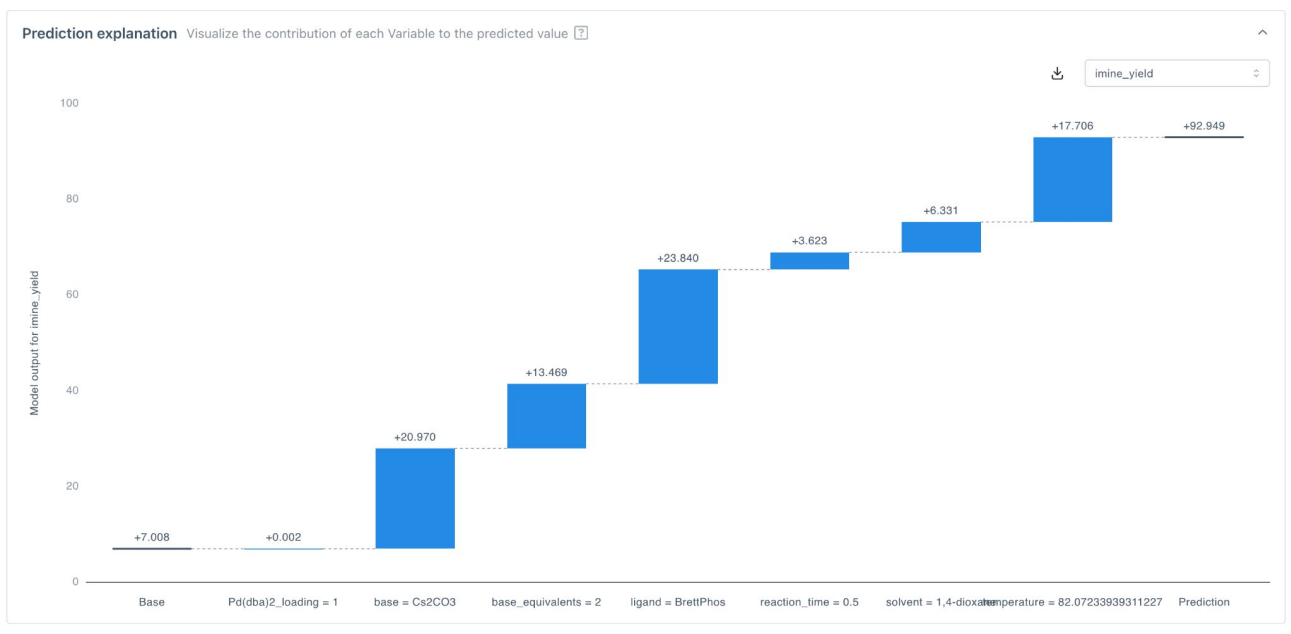
✅ Great **developer experience**

Enabled Pointwise Predictions



SDLabs prediction interface
Predictions generated in **less than 1 second**!

Enabled Visualizations of Model's Beliefs



SDLabs Shapley values plot
(up to tens of thousands of predictions - generated in a **few seconds**!)

Thank you for your attention!

Want to work on **meaningful problems** at the **intersection of AI and scientific discovery?**

Open positions

- Full Stack Developer
- DevOps Engineer

References

- MLflow
- Flask
- MLServer
- KServe
- Open Inference Protocol
- Docker Compose
- Moto