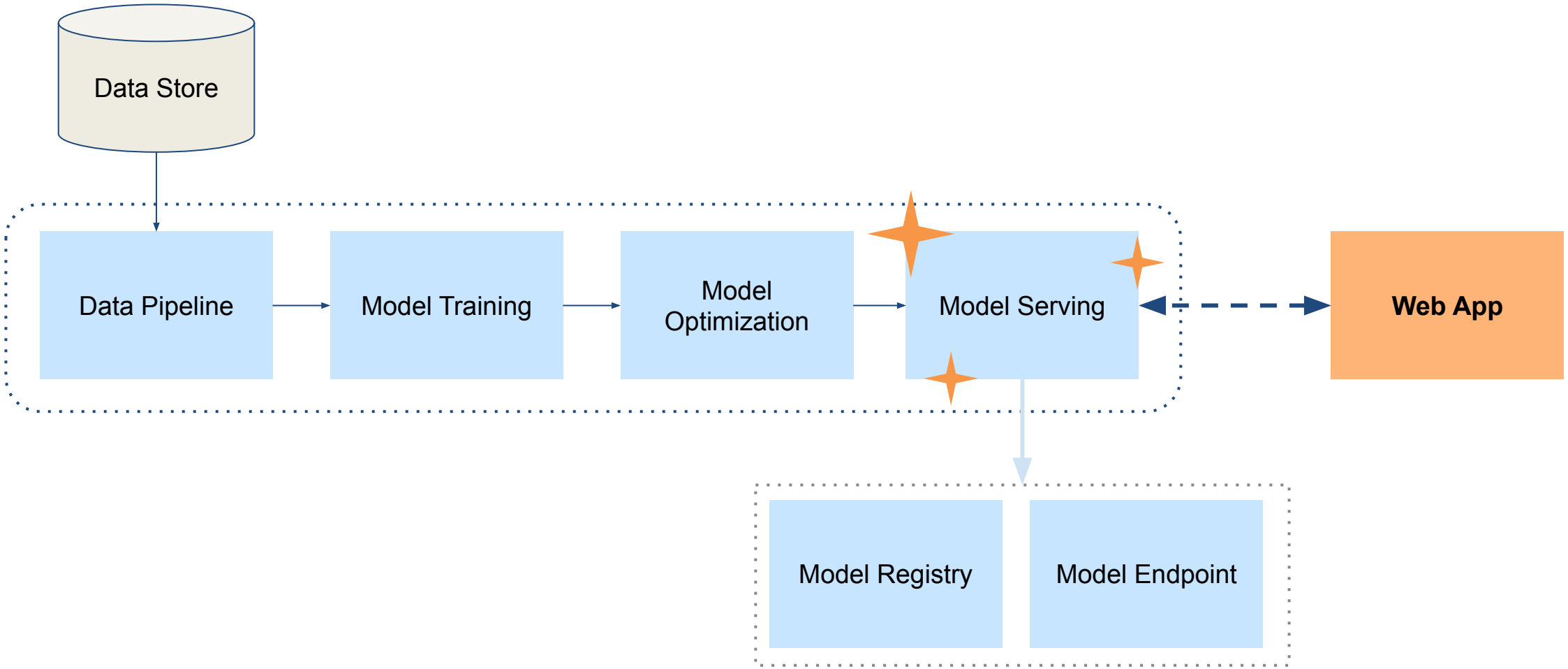
Lecture 10: Model Monitoring

AI-5

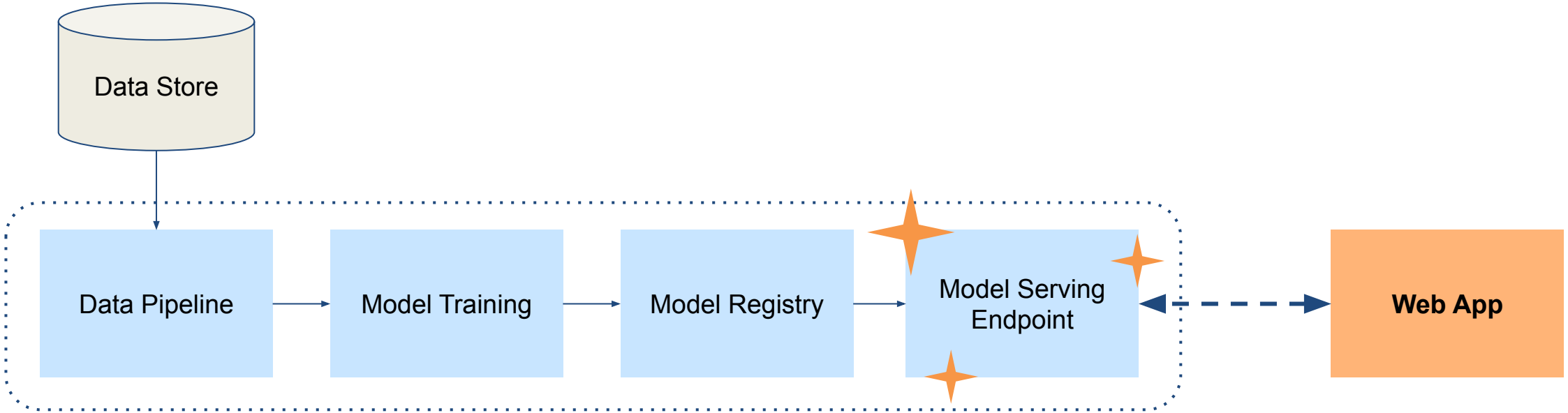
Productionizing AI (MLOps)

Pavlos Protopapas, Shivas Jayaram

The Situation



The Situation



The Situation

We trained and deployed an amazing model. Time to….

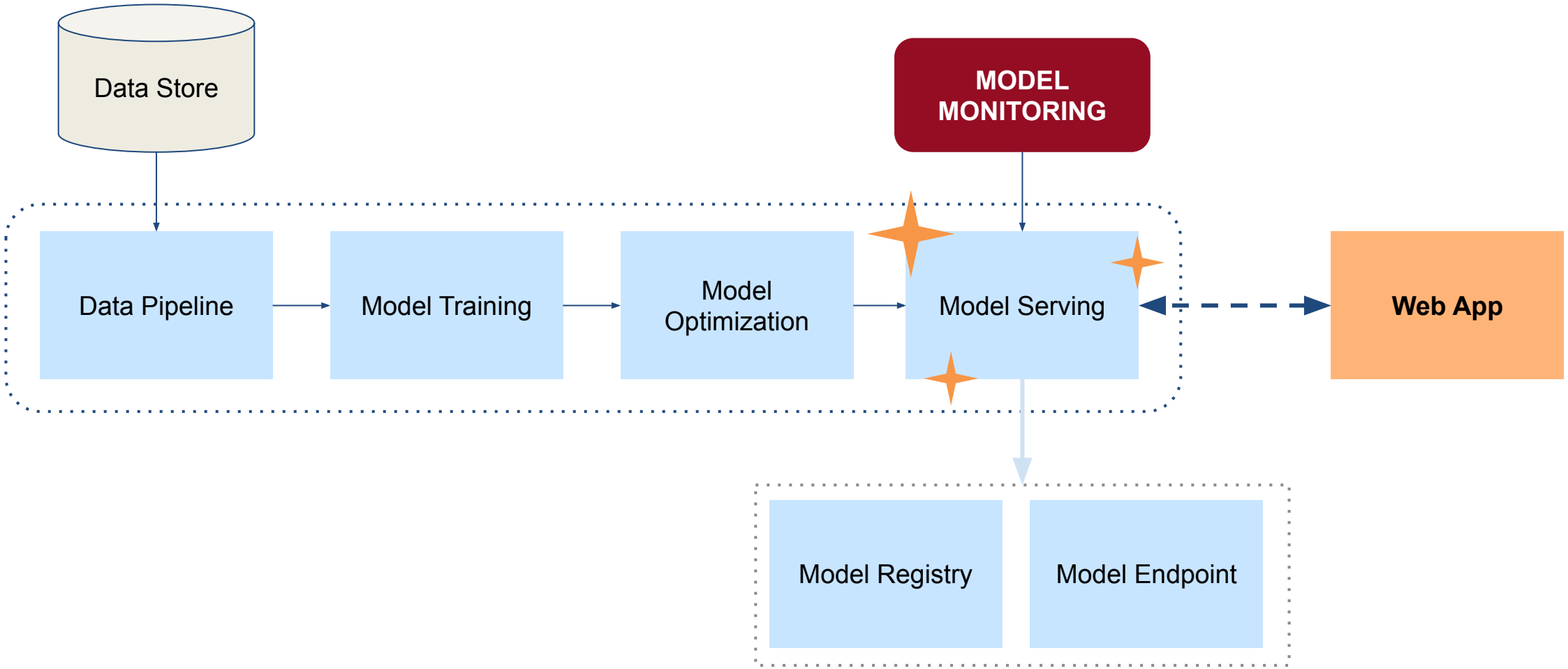
But…

Your boss emails you saying:

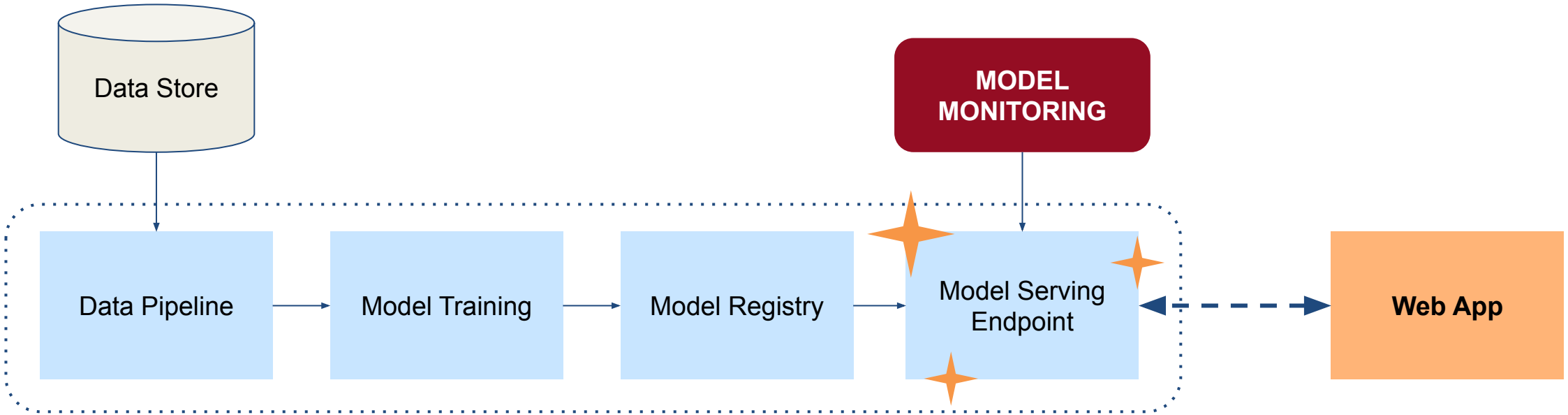
- Your postcard-generating app is encountering massive demand spikes during the holiday season.
- Your volcanic eruption model is receiving abnormal readings from sensors on the mountain's south side.
- Your model for forecasting Mass General's Personal Protective Equipment is not keeping up with the demand from COVID.
- Your loan approval model has been deployed for a month, and performance between demographic groups is degrading while recall remains the same.

But…

The **NEW** Situation



The **NEW** Situation



Two Perspectives of Model Monitoring

Data Scientist

- Monitor for changes in prediction quality
- Continuously assess input data quality
- Check for fairness and prevent bias

MLOps Engineer

- Evaluate API latency and load
- Manage production variants and deployments
- Configure and oversee retraining pipelines

Two Perspectives of Model Monitoring

Data Scientist

- Monitor for changes in prediction quality
- Continuously assess input data quality
- Check for fairness and prevent bias

MLOps Engineer

- Evaluate API latency and load
- Manage production variants and deployments
- Configure and oversee retraining pipelines

THESE CAN OFTEN BE THE SAME TEAM

Preparation Steps for Data Drift and Model Monitoring

Key Components for Monitoring:

Baseline Model

- **Definition**: The original model deployed in production.
- **Best Practice**: Always archive the model artifacts for traceability.
- **Purpose**: Serves as a benchmark for retraining and performance comparison.
- **Tip**: Version your models to easily roll back in case of issues.

Reference Dataset

- **Definition**: A dataset used as a point of reference for monitoring.
- **Purpose**: To compare the model's performance on new data against this baseline.

Additional Considerations for Monitoring

- **Monitoring Tools**: Consider using specialized software for automated monitoring.
- **Alerts**: Set up alerting mechanisms for significant deviations in model performance or data statistics.
- **Retraining Strategy**: Plan how and when the model will be retrained. Will it be manual, scheduled, or triggered by performance metrics?
- **Documentation**: Keep detailed records of all versions of datasets and models, as well as any changes to pre-processing steps, features, and hyperparameters.

Preparation Steps for Data Drift and Model Monitoring

- The baseline or originally deployed **model**
  - Always save a copy of the the model artifacts!
  - We use this for retraining -- more soon

- Baseline or reference **dataset**
  - This will be used to compare what we trained on versus what is happening in production
  - Most of the time this will be the training data, but not always

Production Dataset

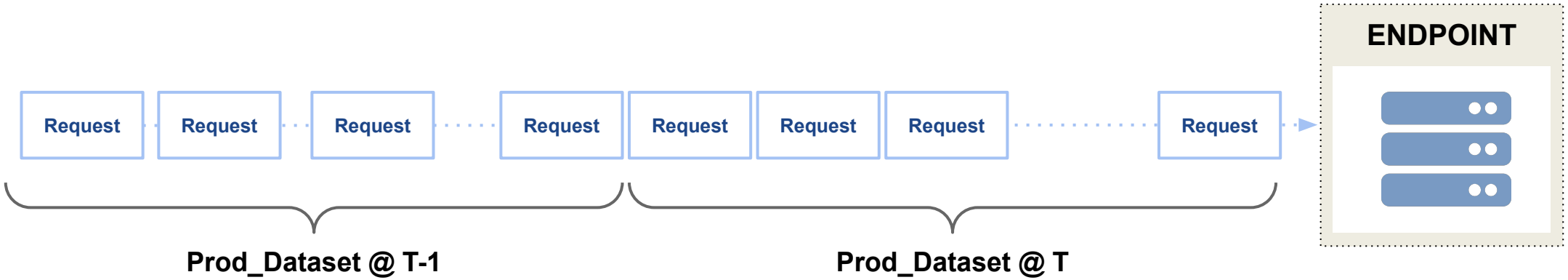
What constitutes the production dataset?

- Incoming traffic is sporadic and **ongoing**
- To track metrics we need to group sequential requests
- This grouping will be considered our production dataset

To do this, we use **windowing**

Production Dataset

Tumbling Window:
Slide = WindowSize

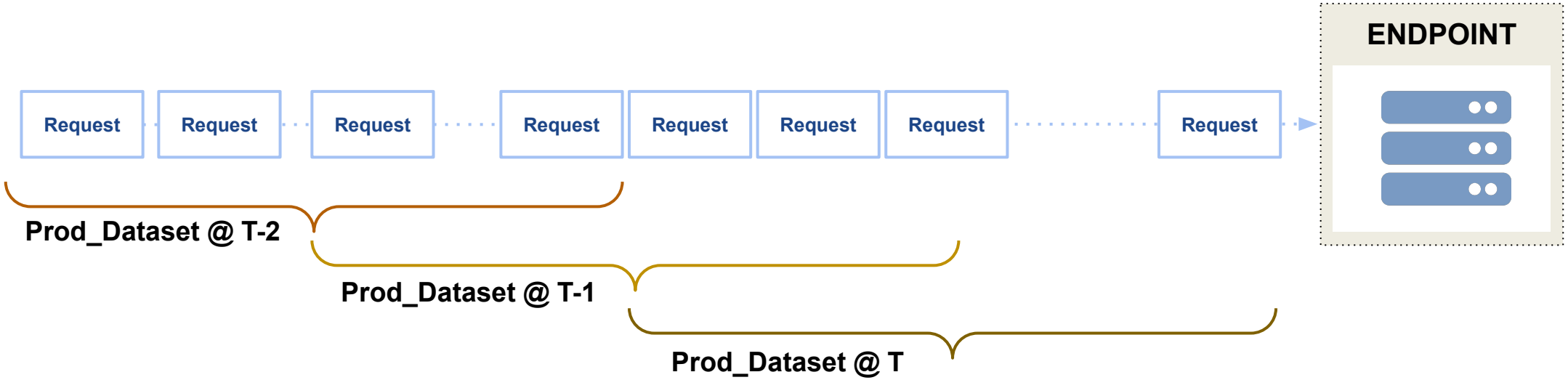


➔ WindowSize: Defined in term of buffer size, or a time period
➔ The windows don't overlap, and always start at the end of the preceding one

Production Dataset

Sliding Window:

Slide < WindowSize



➔ WindowSize: Defined in term of buffer size, or a time period
➔ More common with early detection methods

Evaluating Model: Baseline vs. Production Datasets

We now have our **Baseline** and **Production** datasets.

- **Compare Baseline vs Production:**
  - See if training assumptions hold in production
- **Compare Production @ T w/ Production @ T-1:**
  - Look for gradual shifts happening in production

DS Perspective

Model pre-production:

- 99% Accuracy and "*Everything is Awesome!*"

Model @production:

- 99% Accuracy and "Oh… Why? What? This is horrible!"

Models can and will degrade in production, and it is usually the result of data issues.

Outliers in Production Data: Causes and Implications

Outliers: Data points that deviate significantly from the norm.

The data a model consumes in production can be riddled with outliers which can skew model predictions and affect overall performance.

These outliers can result from various causes, such as sensor problems, malicious actors, and even your inference preprocessing pipeline.

Why do we care? Model Integrity, decision-making

Outliers in Production Data: Causes and Implications

Detecting Outliers:

- Classical methods like column-wise **IQR**
  - Can be challenging with streaming data

- Use **ML** to create anomaly scores for each observation:
  - Isolation Forest
  - Random Cut Forest

Data Drift: Covariate Shift

Covariate Shift:

Refers to the phenomenon when the **distribution of the features - P(X) -** used to train the model are different than the distributions seen in production.

Data Drift: Concept Drift

Concept Drift:

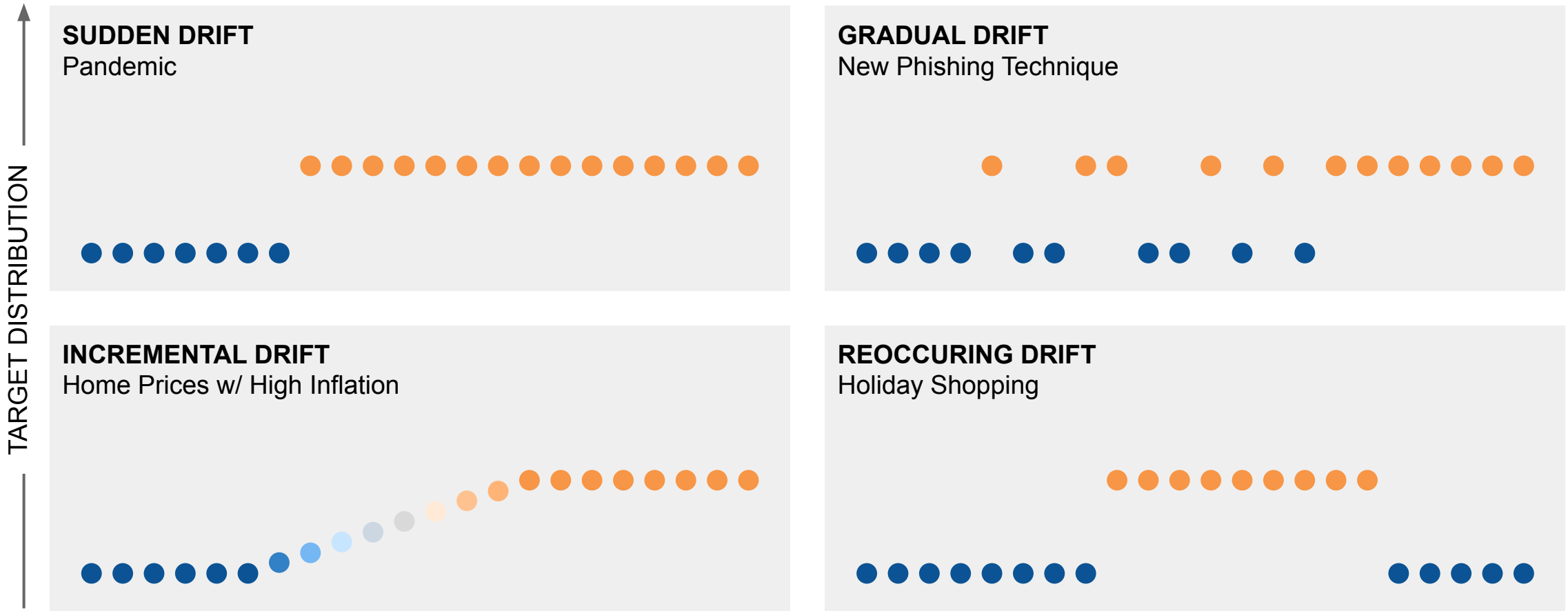
Sometimes called class drift or posterior probability shift, this problem arises when the task the model was designed to perform changes. It is a change in **P(Y|X) or P(Y)**.

Example:

- Facial recognition models suddenly encounter masks due to COVID
- All new phishing techniques are used to get around spam filters

DS Perspective

Concept Drift Patterns:



Data Drift Metrics

Data Drift Metrics:

The problem is a change in two distributions, namely the training $P(Y,X)$ and production $P(Y,X)$. So we test the similarity between these two.

- **Kullback-Leibler Divergence**
- **Jensen-Shannon Divergence**
- **Earth Mover's Distance / Wasserstein**
- **Hellinger Distance**
- **Classifier-Based Distance**

Concept Drift Detection Strategies

Detecting Concept Drift:

- **Performance Monitoring**: Track key performance indicators (KPIs) like accuracy, precision, and recall.
- **Statistical Tests**: Use tests like Chi-Squared or Kolmogorov-Smirnov to compare distributions.
- **Classifier-Based**: Train a model to distinguish between old and new data; high accuracy indicates drift.

Data Drift Metrics

Population Stability Index (Prior Probability Shift)

Characteristic Stability Index (Covariate Shift)

Fairness and Bias

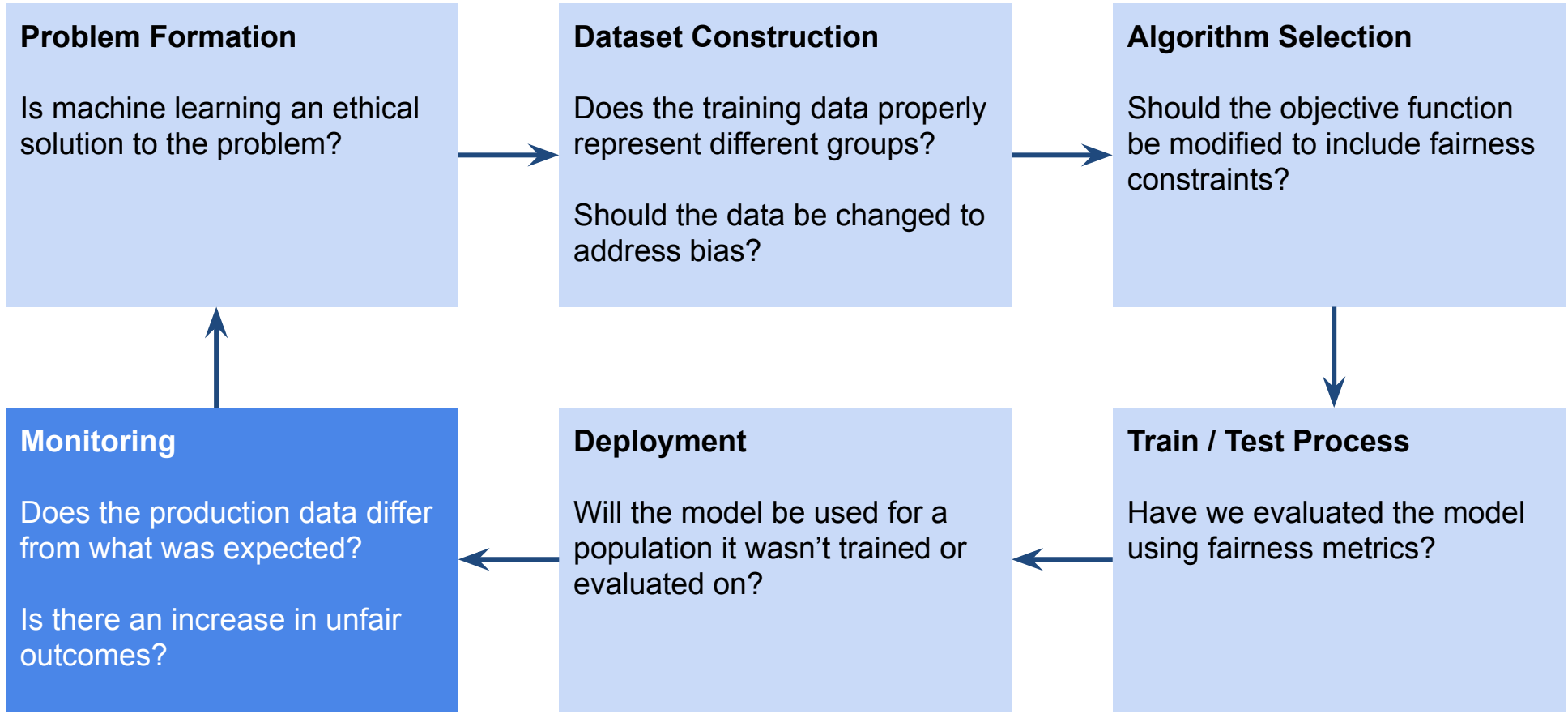
Why Fairness Matters: Unfair models can perpetuate societal biases, causing harm and creating ethical dilemmas.

Monitoring your model for unfair outcomes between facets or subgroups is **important**. To be successful, thought and evaluation must happen during the entire machine learning lifecycle.

So, quick review…

Fairness and Bias

ML Lifecycle:



Fairness and Bias

Fairness Metrics:

This is an active area of research and discussion. New metrics are being proposed frequently. However, some older metrics such as **DPPL** are currently used.

Bias Metrics

DPPL:

Difference in Positive Proportions in Predicted Labels

- Assess disparities in predicted positive outcomes between facets.
- Useful for classification problems

- **DPPL = q'_a - q'_b**
  - $q'_a = n'^{(1)}_a / n_a$: proportion of facit **a** that get positive outcomes of 1
  - $q'_b = n'^{(1)}_b / n_b$: proportion of facit **b** that get positive outcomes of 1

Interpretation of DPPL:

- Positive DPPL: Facet **a** has more predicted positive outcomes (positive bias).
- DPPL near zero: Equal outcomes between facets (demographic parity).
- Negative DPPL: Facet **b** has more predicted positive outcomes (negative bias).

MLOps Perspective

You've discovered that your model and/or data is wonky!

MLOps Perspective

Enter the MLOps Engineer:

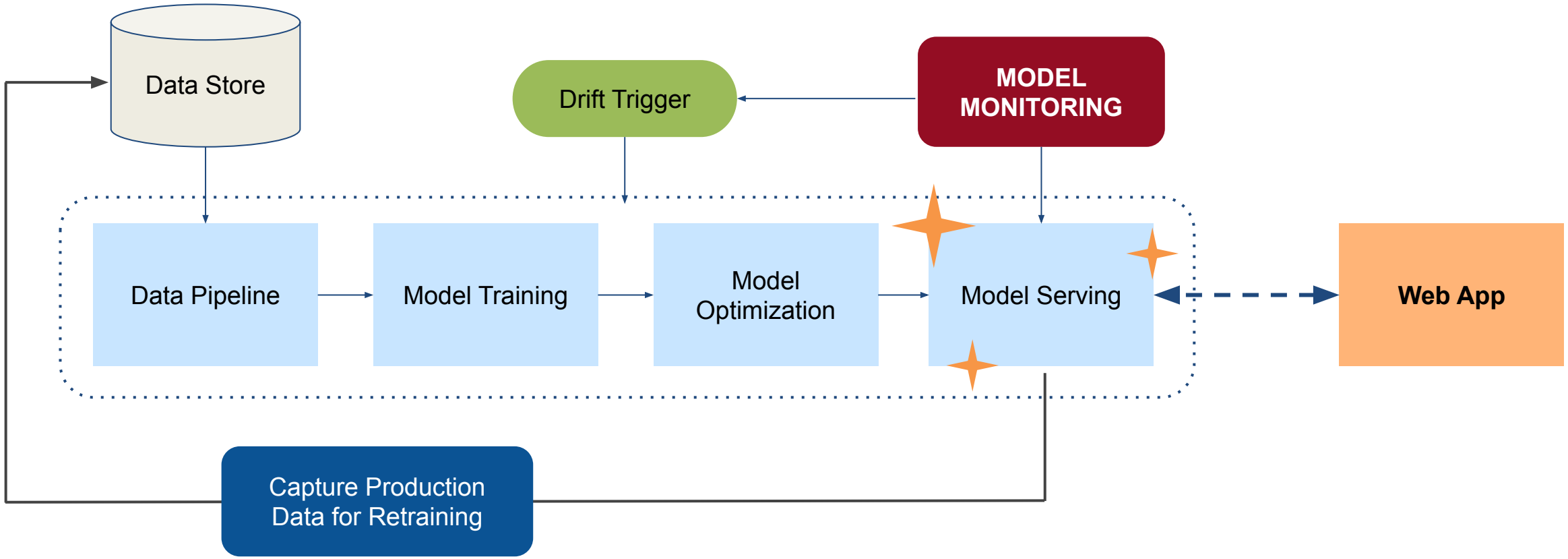
- Handles the **retraining pipeline**
- Monitors model and endpoint **performance**
- Manages new deployments, production variants, and shadow testing

MLOps Perspective

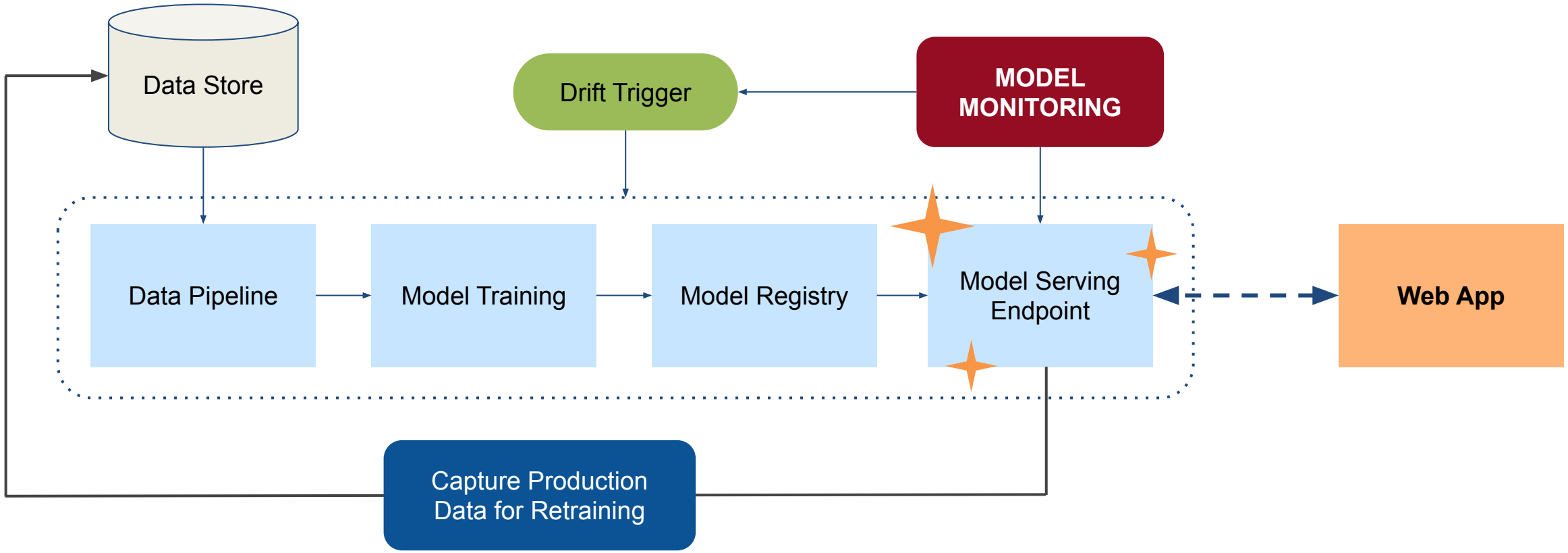
Fix Drift Issues w/ Retraining:

- Periodic Retraining
  - Schedule based on time or number of predictions
  - Easy, but usually inefficient
- Performance Trigger:
  - Use a metric such as accuracy or Prior Probability Shift (PPI) to kick off retraining pipeline

Simplified Retraining Pipeline



Simplified Retraining Pipeline



MLOps Metrics

Beyond the retraining pipeline, MLOps Engineers monitor their own set of metrics related to system performance.

- **GPU/CPU/RAM Utilization**
  Look for bottlenecks or underutilized resources
- **Model Latency**
  The time it takes for the model to make a prediction
- **Overhead Latency**
  Time between when the web app sends a request and receives a response
- **Invocations**
  The number request being received

MLOps Metrics

Handling High Model Latency

- Increase **power** of compute instance hosting endpoint
- Shrink model using quantization or distillation

Handling Bottlenecks or High Load

- **Scale** the endpoint by using more container instances

Coming soon to a lecture near you

Tutorial: Model Monitoring & Data Drift

Steps to perform **Model Monitoring & Data Drift** on mushroom classification models:

- Download best model from WandB.
- Setup model for inference.
- Monitoring Input Data and Predictions on WhyLabs.
- For detailed instructions, please refer to this notebook
  - Model Monitoring & Data Drift.
    (https://colab.research.google.com/drive/1DN99dO7YullZ-6o_W6_bWBz6LwMAoX50?usp=sharing)

THANK YOU