**Data Ingestion**
The first step in model monitoring would be **Data Ingestion**, where the system should log relevant information, such as:

- Input features
- Model predictions
- Metadata
- Timestamps (important for time-series analysis)
- Ground-truth label (if available)

This data should be stored in a centralized storage system, such as a cloud storage bucket or time-series database for subsequent analysis to be performed.

**Detecting Drift**
When it comes to detecting drift, there are different categories to consider:

1. Data Drift
   Data drift occurs when the statistical properties of the input features, such as distribution, mean etc, of the data change significantly over time. This is a good indication of the model encountering data that is not representative of the training data. Some methods that can be used include:

   - The Kolmogorov-Smirnov (K-S) test can be used to compare the similarity in distribution of features between production and training data.
   - Population Stability Index (PSI) can be used to quantify the change in distribution of a feature over time.

   Additionally, model predictions should also be monitored. Changes in prediction distribution (e.g. class imbalance) are also useful in detecting instances where the input features are changing in ways which create unfamiliar conditions for the model, leading to instability.

2. Concept Drift
   Concept drift occurs when there is a change in the relationship between the input features and target variable. This can happen in use cases where trends evolve over time, such as fraud detection (e.g. new fraud tactics, change in customer behavior patterns).

   Detection of concept drift can be achieved using performance metrics, such as precision, recall, F1 score etc. If these metrics show a consistent drop over time, this could be a possible indicator of a change in the relationship between the input features and target variable.

**Visualizations and Alerts**
Effective visualizations, such as dashboards, are useful for tracking model performance by displaying key metrics and distributions over time. They provide a high-level overview of the model’s performance, and makes it easier to spot issues early on.

Furthermore, automated alerts should be configured to notify the relevant teams when drift is detected. This can be done by setting specific thresholds on different metrics. For example, if a precision experiences a drop of 5%, an alert is triggered.

**Retraining**

Once drift is detected, the model needs to be retrained to ensure it remains effective. The process of retraining a model can include:

- Curating new training data from production, which is collected and preprocessed to ensure it is clean and representative.
- Hyperparameter tuning to ensure that the model is tuned to the newest data
- Feature re-engineering to ensure that features selected remain optimal based on the latest data trends.

Lastly, before fully deploying the retrained model, it should be tested in a controlled environment with a small percentage of live traffic, to assess and monitor its performance. This can be done using basic performance metrics, and controlled testing (e.g. A/B testing), and the model can be redeployed once it achieves satisfactory performance on these tests.