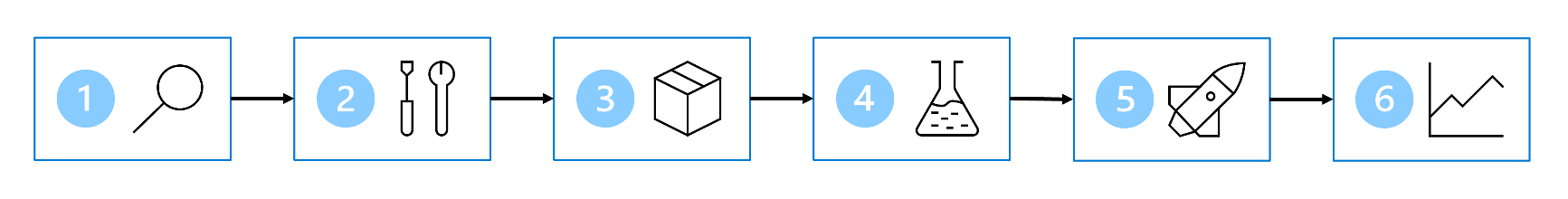
Data is the foundation of machine learning. Both data quantity and data quality will affect the model’s accuracy.

To train a machine learning model, you need access to the data you plan to use. So, before you start to experiment with machine learning models, make sure to identify the data sources and know which tools you want to use to load and transform the data.

Data is the most important input for your machine learning models. You’ll need access to data when training machine learning models, and the trained model needs data as input to generate predictions.

 six steps to plan, train, deploy, and monitor the model:



1. **Define the problem**: Decide on what the model should predict and when it's successful.
2. **Get the data**: Find data sources and get access.
3. **Prepare the data**: Explore the data. Clean and transform the data based on the model's requirements.
4. **Train the model**: Choose an algorithm and hyperparameter values based on trial and error.
5. **Integrate the model**: Deploy the model to an endpoint to generate predictions.
6. **Monitor the model**: Track the model's performance.

* **Define the Problem:**
  + Decide what the model should predict.
  + Determine criteria for success.
* **Get the Data:**
  + Identify where the necessary data is stored (databases, CRM systems, IoT devices).
  + If data is unavailable, collect new data, use publicly available datasets, or purchase curated datasets.
* **Identify the Data Source and Format:**
  + Determine the data's storage location and format (structured, semi-structured, unstructured).
* **Data Formats:**
  + **Tabular/Structured Data:** Data with a defined schema, like Excel or CSV files.
  + **Semi-structured Data:** Data with key-value pairs, such as JSON objects.
  + **Unstructured Data:** Data without a predefined structure, like images, audio, and video files.
* **Prepare the Data:**
  + Explore, clean, and transform the data as per model requirements.
  + Convert data to a suitable format for training (e.g., JSON to tabular).
* **ETL/ELT Process:**
  + Extract, Transform, and Load data into a serving layer for further processing.
* **Train the Model:**
  + Choose an algorithm and set hyperparameters through trial and error.
* **Integrate the Model:**
  + Deploy the model to an endpoint to generate predictions.
* **Monitor the Model:**
  + Track performance and retrain the model if necessary.
* **Example Use Case:**
  + Extract data from IoT sensors as JSON.
  + Convert JSON to tabular format.
  + Aggregate data (e.g., average temperature per hour) for model training.

 **Separate Compute from Storage:**

* Scale compute up or down based on demand.
* Shut down compute when not in use to save costs.
* Ensure data is stored separately to prevent loss and maintain access for other purposes.

 **Store Data for Model Training Workloads:**

* Use cloud data services to store data for flexibility and cost-efficiency.
* Choose the best storage tool based on your data and training service.

 **Common Azure Data Storage Options:**

* **Azure Blob Storage:**
  + Cheapest option for unstructured data (e.g., images, text, JSON).
  + Often used for CSV files.
* **Azure Data Lake Storage (Gen 2):**
  + Advanced version of Azure Blob Storage.
  + Implements hierarchical namespace for easier access control.
  + Suitable for large data with virtually limitless storage capacity.
* **Azure SQL Database:**
  + Stores data as structured tables with defined schema.
  + Ideal for static data that doesn’t change over time.

 **Set Up ETL/ELT Pipelines:**

* Extract, transform, and load data into chosen storage solutions.
* Ensure seamless data serving to Azure Machine Learning, Azure Databricks, or Azure Synapse Analytics for training workflows.

 **Data Ingestion Pipeline:**

* A sequence of tasks to move and transform data.
* Tasks can be triggered manually or scheduled for automation.

 **Creating a Data Ingestion Pipeline with Azure Services:**

* **Azure Synapse Analytics:**
  + Use Synapse Pipelines with an easy-to-use UI or define in JSON format.
  + Copy data using standard connectors.
  + Add transformation tasks using mapping data flow, SQL, Python, or R.
  + Choose from serverless SQL pools, dedicated SQL pools, or Spark pools for compute.
* **Azure Databricks:**
  + Ideal for a code-first approach using SQL, Python, or R.
  + Define pipelines in a notebook and schedule to run.
  + Uses Spark clusters for distributed compute, transforming large amounts of data quickly.
* **Azure Machine Learning:**
  + Provides scalable compute clusters.
  + Create pipelines with Designer or scripts.
  + Typically used for training models but can also perform data extraction, transformation, and storage.
  + May not be as scalable for data transformation as Synapse Analytics or Databricks.

 **Designing a Data Ingestion Solution:**

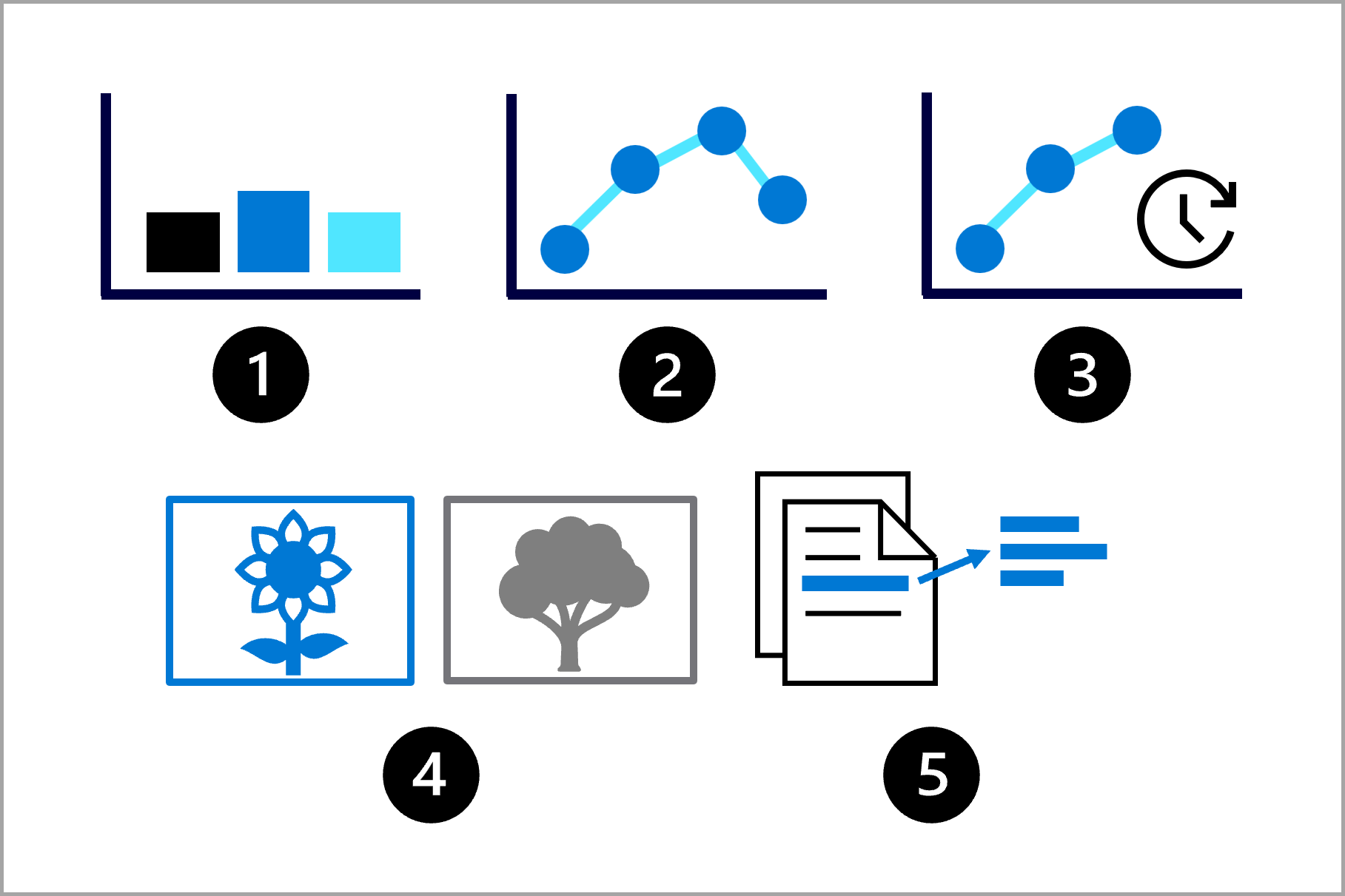
* **Flexibility with Cloud Technologies:**
  + Link services to create a solution that fits your needs.
  + Represent the solution in an architecture diagram.
* **Example Architecture for Data Ingestion:**
  + **Extract** raw data from its source (e.g., CRM system, IoT device).
  + **Copy and transform** the data with Azure Synapse Analytics.
  + **Store** the prepared data in Azure Blob Storage.
  + **Train** the model with Azure Machine Learning.

Starting with the first step, you want to **define the problem** the model will solve by understanding:

* What the model’s output should be.
* What type of machine learning task you’ll use.
* What criteria makes a model successful.

Depending on the data you have and the expected output of the model, you can identify the machine learning task. The task will determine which types of algorithms you can use to **train the model**.

Some common machine learning tasks are:



1. **Classification**: Predict a categorical value.
2. **Regression**: Predict a numerical value.
3. **Time-series forecasting**: Predict future numerical values based on time-series data.
4. **Computer vision**: Classify images or detect objects in images.
5. **Natural language processing** (**NLP**): Extract insights from text.

To evaluate the model, you can calculate performance metrics such as accuracy or precision. The metrics available will also depend on the task your model needs to perform and will help you to decide whether a model is successful in its task.

**Factors to Consider When Choosing a Service:**

* Type of model needed.
* Level of control over model training.
* Time investment in model training.
* Existing services within the organization.
* Preferred programming language.

 **Benefits of Using Azure Services for Training Models:**

* Access to scalable and cost-effective compute.
* Pay only for the compute time used.

 **Commonly Used Azure Services for Training Models:**

* **Azure Machine Learning:**
  + Offers a range of options for training and managing models.
  + Supports UI-based experience with Studio, and code-first experience with Python SDK or CLI.
* **Azure Databricks:**
  + Data analytics platform for data engineering and data science.
  + Uses distributed Spark compute for efficient data processing.
  + Can be integrated with Azure Machine Learning for model training and management.
* **Azure Synapse Analytics:**
  + Analytics service designed for big data ingestion and transformation.
  + Includes machine learning capabilities with Spark pools (MLlib) and integrated Automated Machine Learning.
* **Azure AI Services:**
  + Collection of prebuilt machine learning models for common tasks (e.g., object detection).
  + Models are offered as APIs and can be customized with your own training data.

 **Guidelines for Choosing a Service:**

* **Azure AI Services:**
  + Use for customizable prebuilt models to save time and effort.
* **Azure Synapse Analytics or Azure Databricks:**
  + Use for keeping all data-related projects within the same service.
  + Use for distributed compute with large datasets, working with PySpark.
* **Azure Machine Learning or Azure Databricks:**
  + Use for full control over model training and management.
* **Azure Machine Learning:**
  + Use when Python is the preferred programming language.
  + Use for an intuitive UI to manage the machine learning lifecycle.

**Choosing Compute for Model Training:**

* **CPU vs. GPU:**
  + **CPU:**
    - Suitable for smaller tabular datasets.
    - Cheaper to use.
  + **GPU:**
    - More powerful for unstructured data (images, text).
    - Beneficial for large tabular datasets.
    - Higher cost than CPU.
    - Use libraries like RAPIDs for efficient data preparation and model training with GPUs.

 **General Purpose vs. Memory Optimized:**

* **General Purpose:**
  + Balanced CPU-to-memory ratio.
  + Ideal for testing and development with smaller datasets.
* **Memory Optimized:**
  + High memory-to-CPU ratio.
  + Great for in-memory analytics.
  + Ideal for larger datasets or when working in notebooks.

 **Spark Compute:**

* Offered by services like Azure Synapse Analytics and Azure Databricks.
* Distributes workloads across clusters of driver and worker nodes.
* Reduces processing time by parallel execution.
* Requires code to be written in Spark-friendly languages (Scala, SQL, RSpark, PySpark).
* Must choose between CPU or GPU compute for Spark clusters and specify VM size for nodes.

 **Monitoring Compute Utilization:**

* Iterative process to configure compute resources.
* Monitor training time and compute usage each time you train a model.
* Scale compute up or down based on utilization.
* Consider using GPUs if training takes too long with CPUs.
* Use Spark compute to distribute model training if needed, which may require script rewriting.

**Deploying a Model to an Endpoint:**

* Use endpoints to integrate the model into an application.
* Azure Machine Learning allows deployment to endpoints.

 **Two Deployment Options:**

1. **Real-time Predictions:**
   * For scoring new data as it comes in.
   * Used in applications like mobile apps or websites.
   * Example: Recommending products immediately based on user selection on a website.
2. **Batch Predictions:**
   * For scoring new data in batches and saving results.
   * Used for tasks like forecasting sales.
   * Example: Predicting weekly orange juice sales to ensure supply meets demand.

 **Real-time Predictions Process:**

* Customer selects a product on the website.
* Model recommends other items immediately based on the selection.
* Recommendations are displayed on the website along with the selected product.

 **Batch Predictions Process:**

* Model predicts future values in batches.
* Example: Predicting weekly orange juice sales.
* Predictions are saved and used in reports or other analytics.

**Key Points for Deciding on Real-Time or Batch Deployment**

**Frequency and Timing of Predictions**

* **Real-Time Predictions:**
  + Needed immediately when new data is collected.
  + Example: Customer recommendations on a website.
* **Batch Predictions:**
  + Scheduled or triggered after collecting data over a period of time.
  + Example: Weekly sales forecasts.

**Number of Predictions**

* **Individual Predictions:**
  + Each data point processed separately.
  + Example: Predicting customer churn for each customer individually.
* **Batch Predictions:**
  + Multiple data points processed together.
  + Example: Predicting churn for a group of customers in a batch.

**Compute Power and Costs**

* **Real-Time Deployment:**
  + Requires always-on compute resources (e.g., Azure Container Instance (ACI), Azure Kubernetes Service (AKS)).
  + Higher continuous costs due to the need for immediate availability.
* **Batch Deployment:**
  + Uses compute clusters that can scale up when needed and down to zero when idle.
  + More cost-effective due to scaling down when not in use.

**Decision-Making Factors**

1. **Frequency of Scoring:**
   * Real-time: Immediate predictions needed as data comes in.
   * Batch: Predictions consumed at scheduled times.
2. **Number of Predictions:**
   * Real-time: Individual predictions.
   * Batch: Predictions for large datasets processed in parallel.
3. **Cost Consideration:**
   * Real-time: Higher costs for continuous compute availability.
   * Batch: Cost-effective with compute clusters scaling down when idle.
4. **Compute Power Required:**
   * Simpler models: Less cost and time for predictions.
   * Complex models: More compute power and processing time needed.

**General Guidelines**

* **Choose Real-Time If:**
  + Immediate, individual predictions are essential.
  + Continuous compute costs are justified.
* **Choose Batch If:**
  + Predictions can be scheduled.
  + Cost efficiency is a priority.
  + Slight delay (5-10 minutes) is acceptable for predictions.

Machine Learning operations or MLOps help you to scale your model from a proof of concept or pilot project to production.

**Key Points for Implementing an MLOps Architecture**

**1. Set Up Environments for Development and Production**

* **Development Environment (Dev):**
  + Used for experimenting with model training.
  + Access typically restricted to data scientists.
  + Non-production data is used.
* **Staging/Pre-Production Environment (Pre-Prod):**
  + Used for deploying and testing the best model.
  + A transitional stage before production.
* **Production Environment (Prod):**
  + Final deployment stage where the model is used by end-users.
  + Access typically restricted to machine learning engineers and infrastructure team.

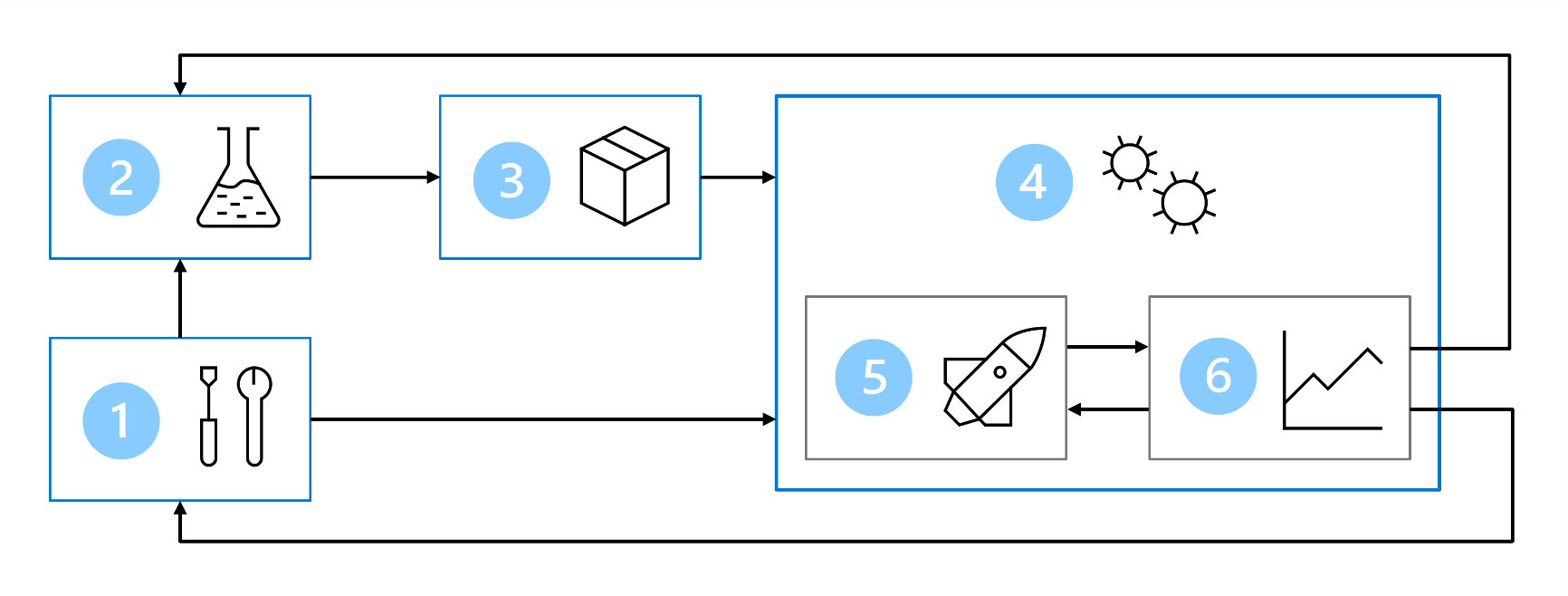
**2. Organize Azure Machine Learning Environments**

* **Multiple Environments:**
  + Separate environments for dev, pre-prod, and prod.
  + Easier access control and resource management.
  + Each environment associated with a separate Azure Machine Learning workspace.
* **Single Environment:**
  + One workspace for both development and production.
  + Smaller Azure footprint, less management overhead.
  + Role-Based Access Control (RBAC) applied to both dev and prod environments, leading to potential access issues.

**3. Design an MLOps Architecture**

* **Setup:**
  + Infrastructure team creates necessary Azure resources (e.g., Azure Machine Learning workspace, Azure Blob storage).
* **Model Development (Inner Loop):**
  + Data scientists explore and process data to train and evaluate models.
  + Focus on model development without worrying about deployment.
* **Continuous Integration:**
  + Packaging and registering the model.
  + Ensuring that the model is reproducible and robust.
* **Model Deployment (Outer Loop):**
  + Machine learning engineers deploy the trained models.
  + Transition from model development to deployment.
* **Continuous Deployment:**
  + Testing the model in a controlled environment.
  + Promoting the model to the production environment.
* **Monitoring:**
  + Monitoring model performance and endpoint availability.
  + Implementing retraining mechanisms based on new data or performance metrics.

MLOps architecture includes the following parts:



1. **Setup**: Create all necessary Azure resources for the solution.
2. **Model development (inner loop)**: Explore and process the data to train and evaluate the model.
3. **Continuous integration**: Package and register the model.
4. **Model deployment (outer loop)**: Deploy the model.
5. **Continuous deployment**: Test the model and promote to production environment.
6. **Monitoring**: Monitor model and endpoint performance.

When you're working with larger teams, you're not expected to be responsible of all parts of the MLOps architecture as a data scientist. To prepare your model for MLOps however, you should think about how to design for monitoring and retraining.

**Designing Monitoring in MLOps**

**1. Monitor the Model**

* **Performance Monitoring:**
  + Use MLflow or similar tools to track model performance metrics during development.
  + Continuously evaluate the model using a subset of new incoming data to ensure it meets performance benchmarks.
* **Metrics to Monitor:**
  + Accuracy, precision, recall, F1 score, etc., depending on the model and use case.
  + Establish benchmarks for each metric to determine when alerts should be triggered.
* **Responsible AI Monitoring:**
  + Check for fairness and bias in predictions.
  + Ensure the model adheres to ethical guidelines and standards.

**2. Monitor the Data**

* **Data Drift:**
  + Track changes in data profiles over time to identify data drift.
  + Retrain models when significant data drift is detected to maintain predictive accuracy.
* **Example of Data Drift:**
  + A model trained on historical automobile data may become less accurate as car manufacturing and engine technologies advance, altering fuel efficiency trends.

**3. Monitor the Infrastructure**

* **Compute Utilization:**
  + Monitor the use of compute resources during both training and deployment phases.
  + Review utilization metrics to optimize compute costs and performance.
* **Scaling:**
  + Scale down compute resources when utilization is low to save costs.
  + Scale up resources when high demand is detected to avoid capacity constraints.

**Designing for Retraining in MLOps**

**1. Determine Retraining Frequency**

* **Scheduled Retraining:**
  + Set up a regular schedule (e.g., weekly, monthly) to retrain the model to ensure it incorporates the latest data and trends.
* **Metrics-Based Retraining:**
  + Monitor the model's performance and data drift continuously. Trigger retraining only when significant changes in performance or data patterns are detected.

**2. Prepare Your Code for Automation**

* **Use Scripts Instead of Notebooks:**
  + Training models with scripts allows for better automation. Scripts can be parameterized to easily adjust training data, hyperparameters, and other configurations.
* **Parameterize Scripts:**
  + Add parameters to your training scripts to allow flexibility in specifying different datasets, training epochs, learning rates, etc.
* **Centralized Code Repository:**
  + Store all project files in a Git-based repository. This setup enables version control, collaboration, and tracking of changes.
  + Implement source control to manage code changes, facilitate collaboration, and maintain a history of modifications.

**3. Automate Retraining Process**

* **Azure Machine Learning Jobs and Pipelines:**
  + Use Azure Machine Learning to create and schedule jobs that run your training scripts.
  + Design and automate end-to-end pipelines that can handle data ingestion, preprocessing, model training, and evaluation.
* **Integration with External Tools:**
  + **Azure DevOps and GitHub Actions:**
    - Utilize Azure DevOps or GitHub Actions to automate your workflows.
    - These tools can trigger Azure Machine Learning pipelines based on specific events, such as new data arrival or changes in the repository.
* **Azure Machine Learning Python SDK and CLI:**
  + For automation tasks, you might prefer using the Azure Machine Learning CLI extension. It's designed for task automation and can be more straightforward when integrated with Azure DevOps or GitHub Actions.
  + The Python SDK can also be used for more granular control over the machine learning workflow within Azure Machine Learning.