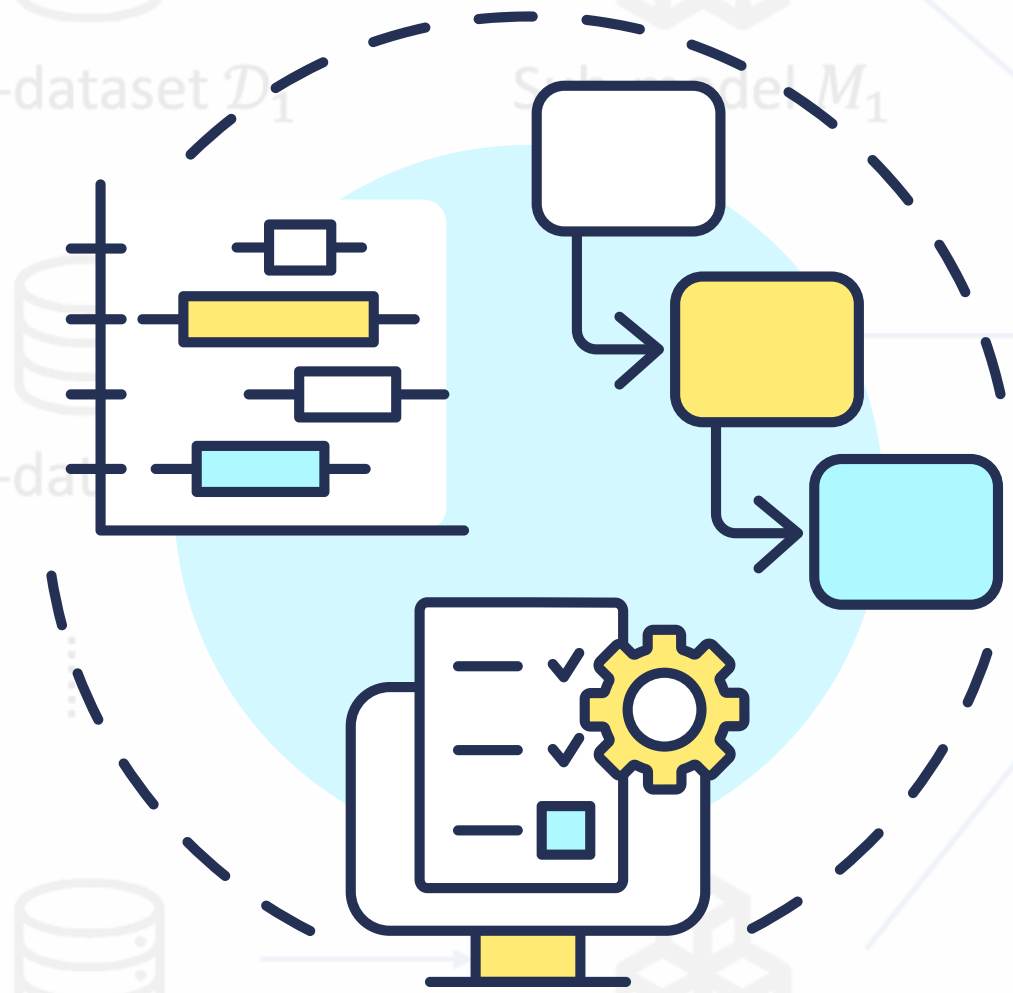
CI/CD For LLM Apps

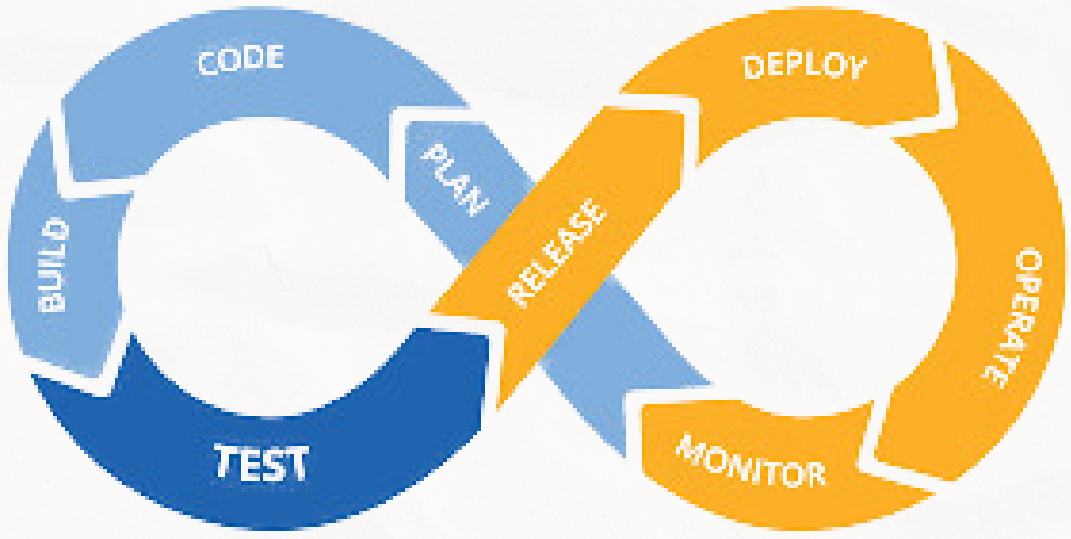
One stop-post to building production grade systems



Bhavishya Pandit

INTRODUCTION

Deploying Large Language Model (LLM) applications efficiently requires a robust Continuous Integration and Continuous Deployment (CI/CD) pipeline. Unlike traditional software, LLM applications involve complex dependencies, large model weights, and performance-sensitive inference, making CI/CD crucial for reliability and scalability.



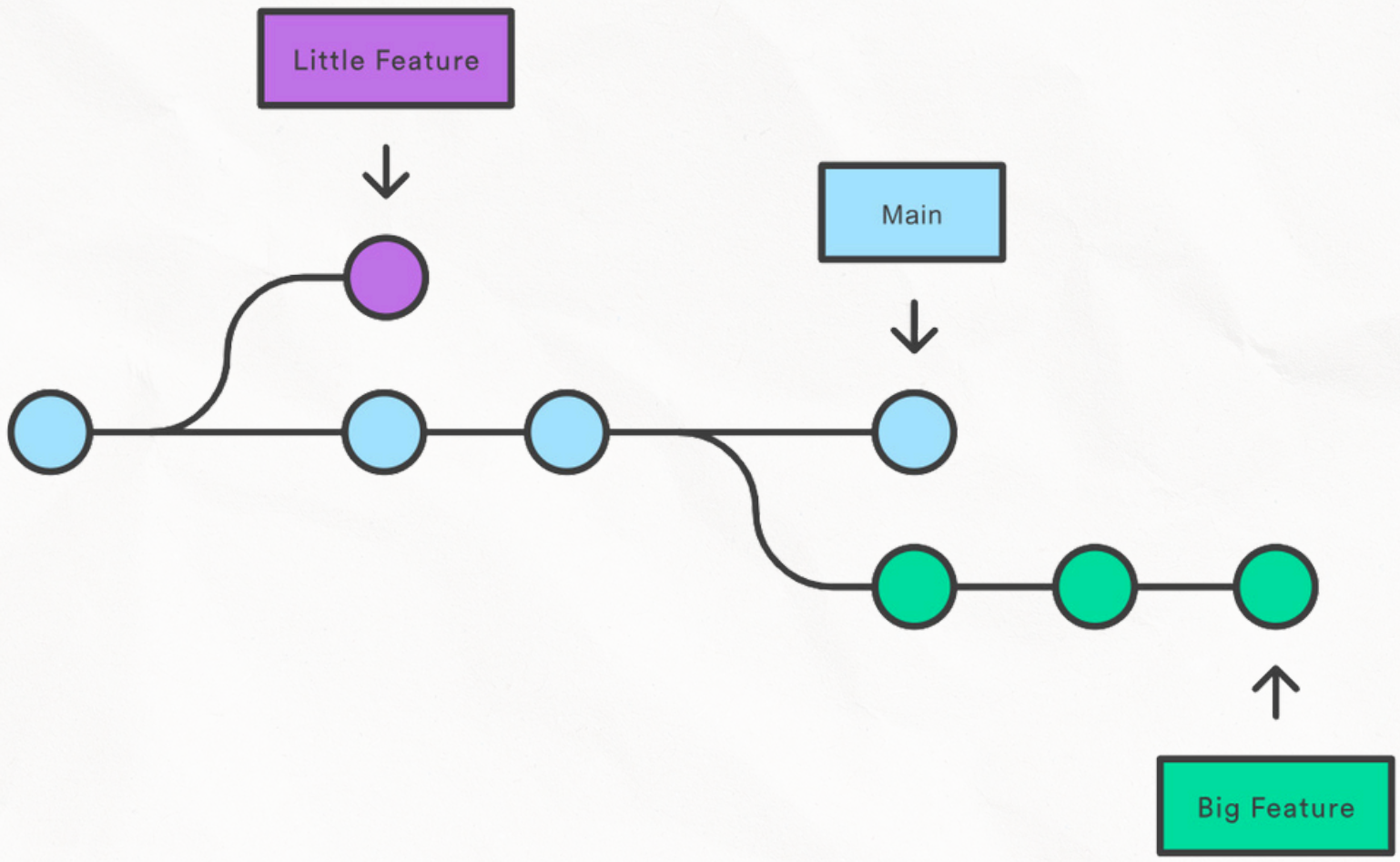
In this guide, we break down the CI/CD process for LLM applications into six key phases:

1. **Version Control and Code Management**
2. **Automated Testing**
3. **Continuous Integration (CI)**
4. **Model Training and Fine-Tuning**
5. **Continuous Deployment (CD)**
6. **Monitoring and Rollback Mechanisms**

VERSION CONTROL & CODE MANAGEMENT

A well-structured version control system ensures that your LLM models, datasets, training scripts, and API logic are properly managed and reproducible across different environments.

- Use **GitHub/GitLab/Bitbucket** to manage repositories.
- Maintain separate branches:
  - **main** → Stable production-ready code
  - **dev** → Experimental features and testing
  - **staging** → Pre-production validation
- Store large model files using **DVC (Data Version Control)** or cloud storage (AWS S3, Google Cloud Storage).
- Use **pre-commit** hooks to enforce coding standards before committing changes.



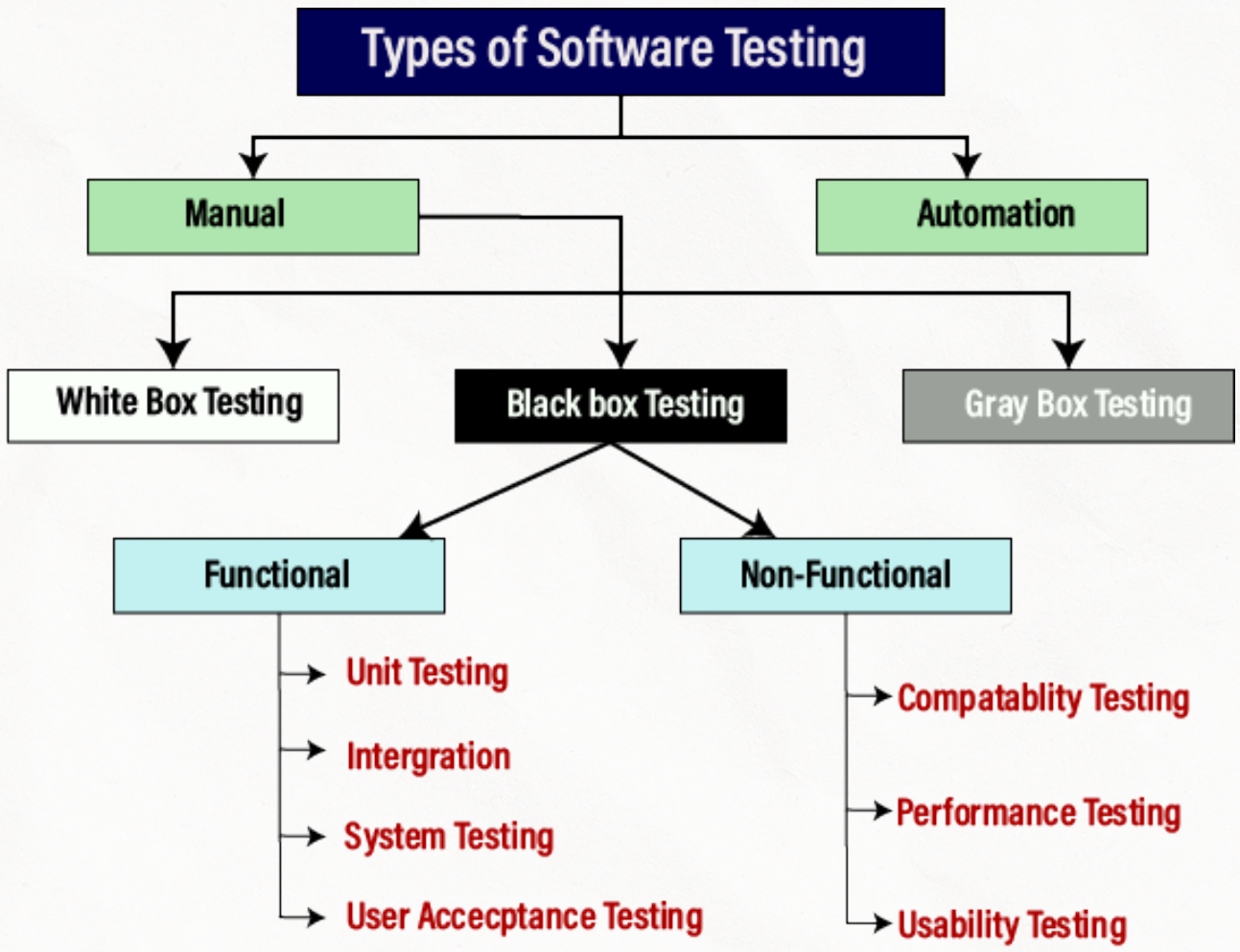
📌 Example: Instead of pushing large .pth or .h5 model files to Git, store them in an object storage service and use version-controlled metadata.

AUTOMATED TESTING

LLM applications require rigorous testing because even minor changes in data processing or prompt handling can lead to unexpected outputs.

Some types of Testing

- **Unit Testing:** Ensures individual functions work as expected (e.g., tokenizer outputs correct token IDs).
- **Integration Testing:** Validates that different modules (e.g., API, database, model) work together.
- **Regression Testing:** Ensures updates don't degrade model performance.
- **Inference Testing:** Compares model responses against a predefined benchmark dataset.



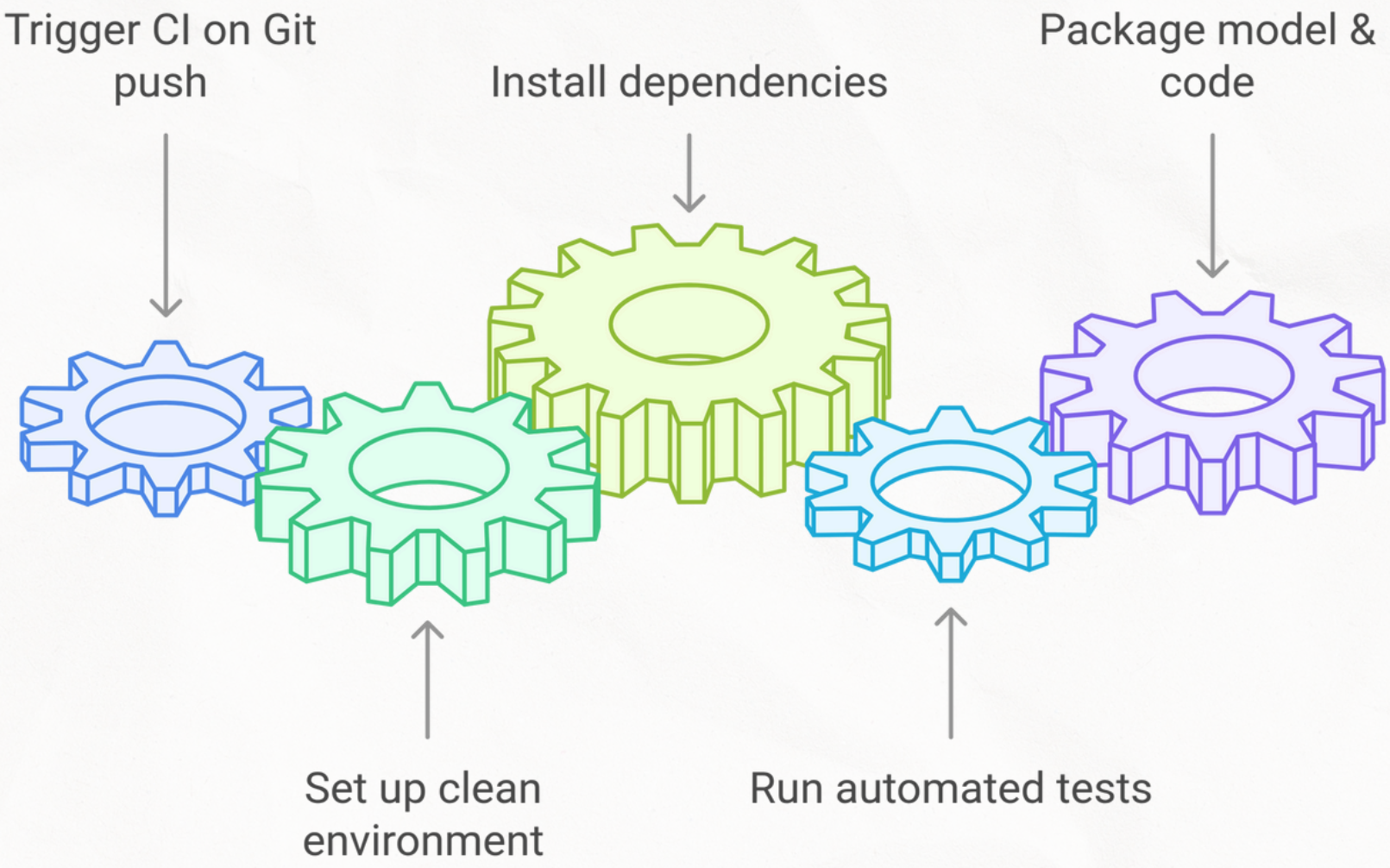
📌**Example:** A test case might check if a summarization model maintains coherence when reducing long documents into a summary.

CONTINUOUS INTEGRATION

CI automates the process of building, testing, and validating changes before merging them into the main codebase.

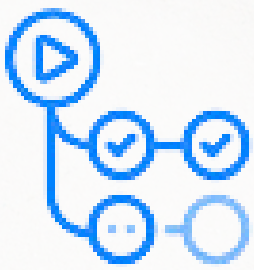
CI Pipeline Workflow

1. Trigger CI on Git push (via GitHub Actions, GitLab CI, or Jenkins).
2. Set up a clean environment using Docker or virtual environments.
3. Install dependencies efficiently, leveraging caching mechanisms.
4. Run automated tests (unit, integration, inference).
5. Package model & code into a container for deployment.



CONTINUOUS INTEGRATION

Tools for CI



Automate testing and packaging

Ensures consistency across environments.

- **Cache Mechanisms** → Speeds up dependency installation using prebuilt images.

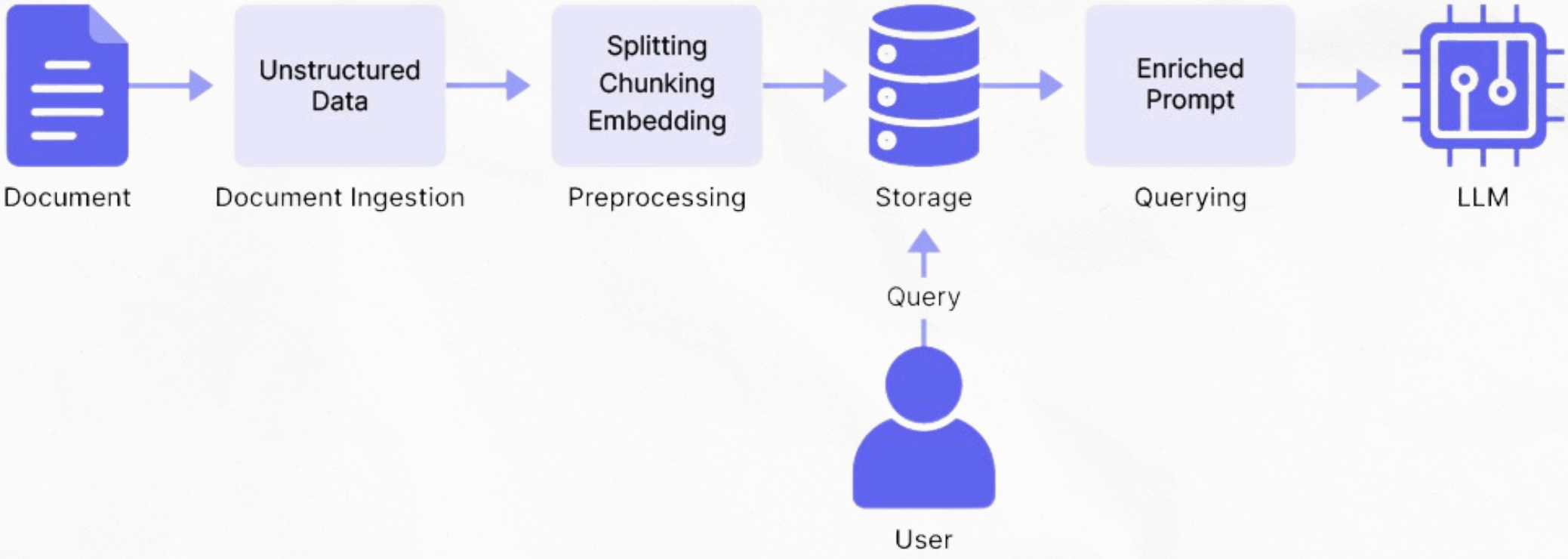
📌 **Example:** A GitHub Action pipeline automatically runs tests and builds a new Docker image whenever a developer pushes changes to the repository.

MODEL TRAINING & FINE-TUNING

LLM applications often require periodic fine-tuning to stay relevant and improve performance on specific tasks. Automating this process prevents manual errors, inconsistencies, and model degradation.

Training Pipeline Workflow

1. **Data Preprocessing:** Clean and format new training data.
2. **Model Training/Fine-Tuning:** Use distributed computing frameworks (e.g., PyTorch Lightning, TensorFlow XLA) for efficiency.
3. **Validation & Benchmarking:** Compare model accuracy against predefined evaluation metrics (e.g., BLEU score for translation models).
4. **Model Versioning:** Save new weights and metadata to a registry.



HIGH LEVEL LLM PIPELINE source: mirascope

MODEL TRAINING & FINE-TUNING

5. Push to Model Hub: Deploy to Hugging Face Model Hub, MLflow, or an internal model registry.

Scaling Training Jobs

- Use **Kubernetes + Kubeflow** for distributed training.
- Leverage cloud-based GPUs/TPUs (**AWS SageMaker, GCP AI Platform**).
- Automate hyperparameter tuning using **Optuna** or Ray Tune.

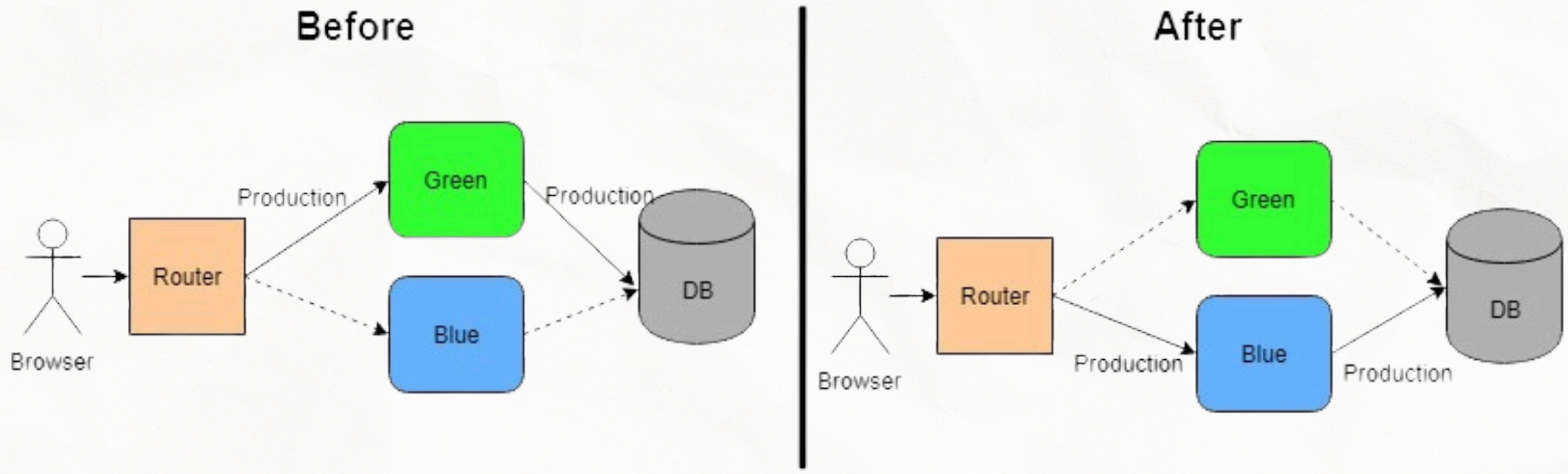
📌 **Example:** A chatbot LLM gets fine-tuned monthly with fresh customer queries, improving responses over time.

CONTINUOUS DEPLOYMENT

CD ensures that new versions of the LLM model reach production safely without downtime or performance degradation.

Deployment Strategies

1. **Blue-Green Deployment:** Run the new and old model versions in parallel, switching traffic gradually.
2. **Canary Deployment:** Release updates to a small subset of users first before full rollout.
3. **Rolling Updates:** Deploy changes in small batches instead of replacing everything at once.



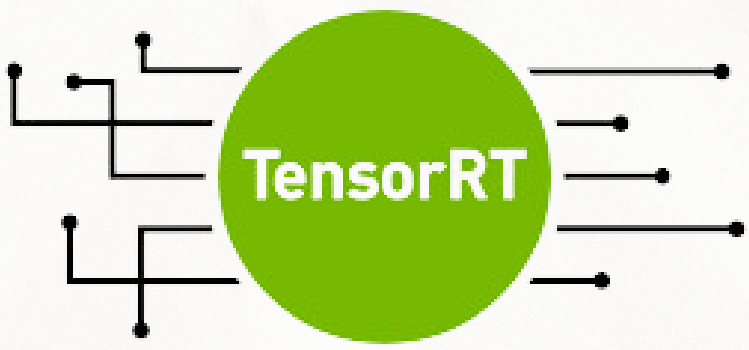
BLUE-GREEN DEPLOYMENT

source: dzone

CONTINUOUS DEPLOYMENT

Deployment Workflow

1. Package Model as an **API** (e.g., FastAPI, Flask, gRPC).
2. Deploy using **Docker** & **Kubernetes**.
3. Expose endpoint via API Gateway (**AWS API Gateway**, **Cloudflare Workers**).
4. Autoscale model inference using **TensorRT, ONNX Runtime**, or **vLLM** for efficient serving.



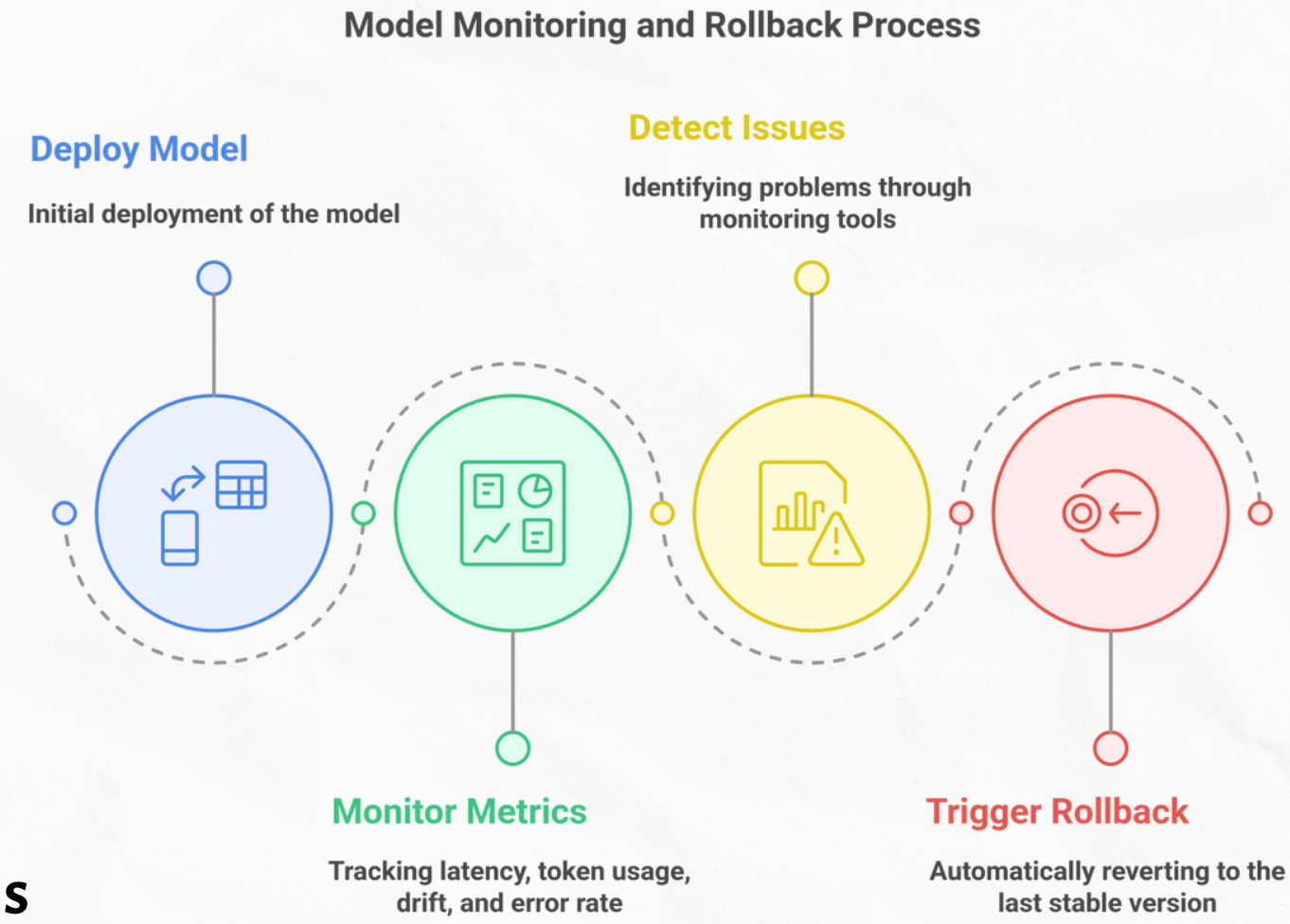
📌 **Example:** A sentiment analysis API updates with a new LLM model version, first tested on 10% of traffic before full deployment.

MONITORING & ROLLBACK MECHANISMS

Post-deployment, it's crucial to track latency, response accuracy, and model drift. A poorly performing model should automatically rollback to the last stable version.

Monitoring Metrics

- **Latency:** Response time per request.
- **Token Usage:** Measures computational cost.
- **Drift Detection:** Compares live data with training data to detect concept drift.
- **Error Rate:** Logs misclassifications or incoherent outputs.



Monitoring Tools

- **Prometheus + Grafana** → Real-time metric visualization.
- **ELK Stack (Elasticsearch, Logstash, Kibana)** → Log aggregation & alerting.
- **OpenTelemetry** → Traces model API performance across microservices.

📌 **Example:** If an LLM chatbot suddenly starts giving nonsensical answers, the pipeline rolls back to the last well-performing model version.

Follow to stay updated on Generative AI

LIKE

COMMENT

REPOST