Automated MLOps Pipeline for Model Training and Deployment

1. **Project Title:** CI/CD/CT Pipeline for Automated ML Model Deployment ([Specific ML Task]) *(Students should replace "[Specific ML Task]" with their chosen example, e.g., "Image Classification", "Sentiment Analysis")*
2. **Project Goal/Objective:** To design and implement a robust, automated MLOps pipeline using cloud-native tools that handles continuous integration (CI), continuous delivery (CD), and continuous training (CT) for a chosen machine learning model. The *pipeline itself* is the primary deliverable.
3. **Problem Statement/Background:** Manually managing the lifecycle of ML models (data validation, training, evaluation, deployment) is inefficient, error-prone, and hinders rapid iteration. MLOps applies DevOps principles to machine learning, enabling automation, reproducibility, and scalability of the ML workflow.
4. **Scope:**
    - **In Scope:** Selecting a standard ML task and dataset (e.g., image classification on CIFAR-10, sentiment analysis on IMDB); setting up source control (e.g., GitHub); configuring a cloud-based CI/CD platform (e.g., GitHub Actions, GitLab CI, Azure DevOps, AWS CodePipeline, Google Cloud Build + Vertex AI Pipelines); automating steps like: data validation, model training, model evaluation (against predefined metrics), model versioning/registration (in a model registry), and potentially triggering deployment to a staging environment upon successful validation/evaluation.
    - **Out of Scope:** Building a highly novel or state-of-the-art ML model (focus is on the pipeline); advanced monitoring or A/B testing features; complex data labeling or feature store integration.
5. **Key Deliverables:**
    - A functional, automated MLOps pipeline triggered by code/data changes (demonstrable via the CI/CD platform).
    - Source code repository containing ML model code, pipeline definition files (e.g., YAML configs), Dockerfiles, scripts.
    - A cloud model registry populated with versioned models generated by the pipeline.
    - (Potentially) A simple deployed staging endpoint serving a model deployed by the pipeline.
    - Final technical report detailing the pipeline architecture, tools used, automation steps, challenges, and demonstration of a pipeline run.
    - Final presentation explaining MLOps concepts and demonstrating the automated pipeline in action.
6. **Potential Datasets/Tasks to Get Started:**
    - **Image Classification:** CIFAR-10, MNIST, Fashion-MNIST.
        - *Links:* Available directly in ML frameworks like TensorFlow Datasets or PyTorch Torchvision.
    - **Sentiment Analysis:** IMDB Movie Reviews, Twitter Sentiment datasets.
        - *Links:* Available in TF Datasets, Hugging Face Datasets.

    - **Tabular Classification:** Iris dataset, Titanic dataset (Kaggle).
        - *Links:* Available in Scikit-learn, Kaggle.
    - **Note:** Choose a relatively small, well-understood dataset/task so the focus remains on building the pipeline infrastructure.
7. **Potential Technical Approach:**
    - **ML Task:** Standard model using Scikit-learn, TensorFlow/Keras, or PyTorch.
    - **Source Control:** GitHub, GitLab, AWS CodeCommit, Azure Repos.
    - **CI/CD Platform:** GitHub Actions (integrates well with GitHub), GitLab CI, Jenkins (self-hosted or cloud), AWS Developer Tools (CodeCommit, CodeBuild, CodePipeline), Google Cloud Build (+ Vertex AI Pipelines for ML steps), Azure DevOps Pipelines.
    - **Key Pipeline Steps:**
        - *Trigger:* Git push to main/specific branch.
        - *Lint/Test:* Code quality checks.
        - *Data Validation (Optional but good):* Check data schema/distribution (e.g., using TFDV or Great Expectations).
        - *Train:* Run training script (likely in a Docker container).
        - *Evaluate:* Run evaluation script, compare metrics against thresholds.
        - *Register:* If evaluation passes, version and save model to a Model Registry (MLflow, Vertex AI, SageMaker, Azure ML).
        - *Deploy (Optional):* Trigger deployment of the registered model to a staging endpoint (e.g., Cloud Run, SageMaker Endpoint, Azure ML Endpoint).
    - **Containerization:** Docker is crucial for creating consistent environments for each pipeline step (training, evaluation, serving).
    - **Infrastructure (Optional):** Infrastructure-as-Code (Terraform, AWS CDK/CloudFormation, Azure ARM/Bicep, Google Cloud Deployment Manager) to define cloud resources.
8. **Deployment Requirements & Tips:**
    - **Requirement:** The primary deliverable is the fully automated MLOps pipeline itself, configured on a chosen cloud CI/CD platform and integrated with source control and a model registry. The pipeline should successfully execute the defined steps (train, evaluate, register) upon code changes. Demonstration involves showing the pipeline run, the resulting registered model, and potentially a deployed staging endpoint updated by the pipeline.
    - **Tips for Students:**
        - **Start Simple:** Build the pipeline incrementally. Get CI working first (e.g., run tests on push). Then add training, then evaluation, etc.
        - **Focus on Automation:** The goal is minimizing manual steps. Use scripting extensively.
        - **Use Cloud Platform Tools:** Leverage the MLOps tools provided by the chosen cloud provider (e.g., Vertex AI Pipelines, SageMaker Pipelines, Azure ML Pipelines) as they integrate well with other cloud services

(storage, registry, endpoints). GitHub Actions is also a strong cross-platform choice.

- **Environment Consistency:** Use Docker containers defined via Dockerfiles for each step (training, evaluation, serving) to ensure environments match. Pin dependency versions in `requirements.txt`.
- **Model Registry is Key:** Use a proper model registry to version models and track lineage (which data/code produced which model). MLflow (can be self-hosted or integrated) or cloud-native registries.
- **Parameterize:** Make scripts and pipeline steps configurable (e.g., learning rate, evaluation thresholds) using configuration files or environment variables.
- **Testing Pipeline Code:** Treat pipeline definition code (e.g., YAML files) like application code – lint it, test it if possible.
- **Resource:** Deep dive into the chosen cloud provider's MLOps documentation and tutorials. Explore tools like MLflow. The `Cloud Computing for Machine Learning` course (especially MLOps parts) is central here.

9. **Potential Challenges:** Steep learning curve for CI/CD/MLOps tools; debugging pipeline failures; managing cloud permissions/credentials across services; ensuring reproducibility; choosing the right level of complexity for the pipeline.
10. **Success Metrics:** Successful automated execution of the end-to-end pipeline; proper model versioning in the registry; demonstration of pipeline trigger and execution; clarity of pipeline definition and documentation.
11. **Team Size / Duration:** 3-4 Students / 14 Weeks