🧠 ML Deployment Workshop Plan: From Training to Model Drift

Duration: 3-4 hours
Audience: Engineers new to ML
Format: Live coding + guided lab (EC2 and Colab friendly)

🔧 Core Philosophy

1. **Start Simple, Build Intuition**: Don't overwhelm with tooling. Use the minimal working pipeline and scale concepts later.
2. **Everything Breaks in Prod**: Emphasize drift, monitoring, and data mismatches after deployment.
3. **Local-to-Cloud Bridge**: Most real ML starts locally but lives on the cloud. Teach reproducibility and deployment as a habit.

🗂️ Workshop Breakdown

📌 Part 1: Setup (15 min)

Objective: Ensure everyone is ready to run notebooks and SSH into EC2

- ✅ Google Colab (or local Python) for training
- ✅ AWS EC2 instance (t2.medium or similar) pre-created (SSH via PEM)
- ✅ Install Python 3.10+, `pip install fastapi uvicorn scikit-learn numpy matplotlib`
- ✅ GitHub repo with starter code

📌 Part 2: Train a Simple Classifier (30 min)

Objective: Basic ML training loop using scikit-learn or PyTorch

- Dataset: CIFAR-10 or MNIST (download via code)
- Model: Simple CNN or sklearn `RandomForestClassifier`
- Metrics: Accuracy on train/test split
- Save model using `joblib` or `torch.save`

📦 Deliverable:

saved_models/
model.joblib

Authored by [Pranjal Joshi](). Feel free to connect: pranjaljoshi [at] live [dot] com

📌 Part 3: Build a FastAPI Inference Server (30 min)

Objective: Serve the model over an API

- Write a FastAPI app:
    - /ping → health check
    - /predict → takes an image (base64 or URL), returns class
- Test locally on Colab (via ngrok or mock request)
- Package as app.py + requirements.txt

    📦 Deliverable:

    fastapi_app/
     app.py
     requirements.txt
     saved_models/model.joblib

📌 Part 4: Deploy to EC2 (30 min)

Objective: Deploy inference server to EC2

- SCP code to EC2
- SSH in, install dependencies
- Run FastAPI with uvicorn app:app --host 0.0.0.0 --port 8000
- Open port 8000 in EC2 security group
- Call prediction endpoint from Colab/local

📌 Part 5: Simulate Model Drift (30 min)

Objective: Show how model performance can degrade post-deployment

- Use CIFAR-10 → Classify CIFAR-100 images (wrong domain)
- Log accuracy drops
- Optionally: Add versioning (model_v1.joblib, model_v2.joblib)
- Teach: Need for retraining pipelines, monitoring

📌 Bonus: Real-World Gotchas (15 min)

Objective: Highlight challenges post-deployment

- 🔁 Inference mismatches (different image preproc)
- 📉 Performance drops with noisy inputs
- 📦 Versioning models & rollback
- 📊 Live metrics tracking (demo with a simple CSV or plot)

Authored by [Pranjal Joshi](). Feel free to connect: pranjaljoshi [at] live [dot] com