Title Slide

- **Title:** Comprehensive Overview of MLflow
- **Subtitle:** Tracking, Serving, Prompting, and Deployment

Slide 1: MLflow Tracking Server Options

- **Default:**
- Command: `mlflow ui`
- Access: `http://localhost:5000`
- Ideal for quick local testing and development

- Stores runs in local file store (`mlruns/` directory)

- **Custom Ports:**

- Command: `mlflow ui --port 1234`
- Access: `http://localhost:1234`
- Useful to avoid port conflicts or run multiple servers

- Ensure to use a browser-safe port (avoid 6000, 6666, etc.)

- **Remote Hosting for Teams:**

- **Dagshub:** Free for small teams, easy integration, GitHub-like interface
- **AWS:**
    - Store artifacts in S3
    - Host server using EC2
    - Track metadata in RDS (PostgreSQL/MySQL)
- **Azure:** Blob storage for model data, Azure ML for workflow automation
- **GCP:** Use GCS for artifacts, BigQuery for tracking, and GKE for scalable deployments

---

Slide 2: MLflow Tracking Features

- **Model Tracking:**
- Automatically or manually track models (via `mlflow.log_model()`)

- Track model versions, source, and associated metrics

- **Code Tracking:**

- Log Git commit hash or manually add as tags

- Ensures reproducibility and version control

- **Data Tracking:**

- Manually log dataset versions/paths

- Integrate with tools like DVC for better tracking

- **Artifact Tracking:**

- Log model files, plots, images, and artifacts

- Supports metadata: `run_id`, `experiment_id`, `owners`, `tags`, `requirements.txt`

- **Metrics:**

- Log scalar metrics like accuracy, loss, F1, etc.

- Visualize them over time across different runs

- **Model Comparisons:**

- UI-based comparison of metrics across runs

- Filter and sort experiments using tags/parameters

- **Parent-Child Run Hierarchy:**

- Create sub-runs to track nested steps (e.g., CV folds)

- **Hyperparameter Tuning:**

- Log each trial using GridSearchCV/RandomizedSearchCV
- Auto-log best parameters and scores

Slide 3: Model Lifecycle and Serving

- **Lifecycle Stages:**
- `None`: Default unassigned state
- `Staging`: Models ready for further evaluation
- `Production`: Deployed and stable version

- `Archived`: No longer in use

- **Promotion/Demotion:**

- Use `mlflow.transition_model_version_stage()` or UI

- Keep history of promotions

- **Model Serving:**

- Serve via REST API: `mlflow models serve -m <model_uri>`
- Integrate with Flask, FastAPI to add UI or auth layers

Slide 4: Prompt Tracking in MLflow

- **Track Prompts via UI or Code:**
- `mlflow.genai.register_prompt()` to log prompt templates

- Track prompt name, template, version, and usage

- **Versioning & Aliases:**

- Use aliases like `production`, `staging`, `beta` for better control

- Roll back to older prompt versions if needed

- **Audit Prompt Evolution:**

- Track who committed what version, with commit messages
- Compare how different versions affect performance

Slide 5: Hugging Face Model Integration

- **Integration with Transformers Library:**
- Log pre-trained or fine-tuned models

- Track tokenizer config and model weights

- **Storage & Versioning:**

- Store HF models in MLflow Registry

- Use metadata to track versions, dataset, performance

- **API Serving:**

- Serve using `mlflow models serve` with HF wrappers
- Wrap in FastAPI for custom inference routes

Slide 6: Gunicorn and Docker Deployment

- **Gunicorn (Linux Only):**
- Run Flask/FastAPI with multiple workers for production
- Command: `gunicorn -w 4 -b 0.0.0.0:8000 app:app`
- **Docker for Windows:**
- Use Docker to containerize app + Gunicorn
- Enables Linux-style deployments on any OS
- **Production Readiness:**
- Add NGINX as reverse proxy, SSL termination, load balancing
- Set environment variables and secrets via Docker/ENV

Slide 7: Legacy Model Logging and Serving

- **Re-logging Old Models:**
- Wrap pre-MLflow models using `mlflow.pyfunc.log_model()`
- Add conda/requirements to serve properly
- **Add Signatures:**
- Define input/output schema manually for API exposure
- Improves model validation and compatibility
- **Serve with MLflow CLI:**
- `mlflow models serve -m <model_uri>`
- Enables quick local API deployment for legacy models