Complete Setup & Deployment Guide

Financial Risk Assessment Model

This guide will walk you through setting up the entire system from scratch.

Prerequisites Checklist

- [ ] Python 3.9+ installed
- [ ] PostgreSQL 12+ installed and running
- [ ] Virtual environment created and activated
- [ ] Git repository cloned

Quick Start (Recommended)

1. Initial Setup (One-time)

```bash
# Navigate to project directory
cd financial_risk_eval

# Make scripts executable
chmod +x scripts/*.sh

# Install Python dependencies
pip install -r requirements.txt

# Install the package
python setup.py develop
```

2. Database Setup

```bash
```

```
# Run database setup script
./scripts/setup_database.sh

# This will:
# - Create the risk_db database
# - Create all schemas (staging, analytics, predictions)
# - Create all tables
```

3. Airflow Setup

bash

```
# Run Airflow setup script (interactive)
./scripts/setup_airflow.sh

# This will:
# - Install Airflow
# - Initialize Airflow database
# - Create admin user (admin/admin)
# - Configure PostgreSQL connection
# - Test your DAG
```

4. Generate Sample Data

bash

```
# Generate 10,000 customers with synthetic data
python scripts/generate_data.py

# This creates:
# - 10,000 customers
# - Credit history for each customer
# - ~50 transactions per customer
# - ~7,000 loan applications
```

5. Start All Services

bash

```
# Start Airflow Scheduler, Webserver, and API
./scripts/start_services.sh

# Services will be available at:
# - Airflow UI: http://localhost:8080 (admin/admin)
# - API: http://localhost:5000
```

6. Trigger the Pipeline

Option A: Via Airflow UI

1. Go to http://localhost:8080
2. Login with admin/admin
3. Find "financial_risk_assessment" DAG
4. Toggle it ON
5. Click "Trigger DAG"

Option B: Via Command Line

```bash
airflow dags trigger financial_risk_assessment
```

Option C: Manual ETL (without Airflow)

```bash
python scripts/run_etl.py
python scripts/train_model.py
```

Detailed Step-by-Step Guide

Phase 1: Environment Setup

1.1 Create Virtual Environment

```bash
```

```
python -m venv venv
source venv/bin/activate  # On Windows: venv\Scripts\activate
```

1.2 Install Dependencies

```bash
pip install -r requirements.txt
```

1.3 Verify Installation

```bash
python -c "import pandas, sklearn, flask, airflow; print('All imports successful')"
```

Phase 2: Database Configuration

2.1 Start PostgreSQL

```bash
# On Linux
sudo systemctl start postgresql
sudo systemctl status postgresql

# On macOS
brew services start postgresql

# On Windows
# Use pgAdmin or Services panel
```

2.2 Create Database

```bash
```

```
# Connect to PostgreSQL
psql -U postgres

# Create database
CREATE DATABASE risk_db;

# Quit psql
\q
```

2.3 Run Schema Script

```bash
psql -U postgres -d risk_db -f sql/schema.sql
```

2.4 Update Database Configuration

Edit `config/database.yaml`:

```yaml
development:
  host: localhost
  port: 5432
  database: risk_db
  username: postgres
  password: YOUR_PASSWORD_HERE  # Change this!
```

Phase 3: Airflow Configuration

3.1 Set Environment Variables

```bash
export AIRFLOW_HOME=$(pwd)/airflow
export DB_PASSWORD=your_postgres_password

# Add to ~/.bashrc for persistence
echo "export AIRFLOW_HOME=$(pwd)/airflow" >> ~/.bashrc
```

3.2 Initialize Airflow

```bash
```

```
airflow db init
```

3.3 Create Admin User

```bash
airflow users create \
    --username admin \
    --firstname Admin \
    --lastname User \
    --role Admin \
    --email admin@example.com \
    --password admin
```

3.4 Add Database Connection

```bash
airflow connections add 'risk_db_connection' \
    --conn-type 'postgres' \
    --conn-login 'postgres' \
    --conn-password 'your_password' \
    --conn-host 'localhost' \
    --conn-port '5432' \
    --conn-schema 'risk_db'
```

3.5 Test DAG

```bash
# Check DAG syntax
python airflow/dags/risk_assessment_dag.py

# List DAGs
airflow dags list | grep financial_risk_assessment

# Test a specific task
airflow tasks test financial_risk_assessment check_data_freshness 2024-01-01
```

Phase 4: Data Generation

4.1 Generate Synthetic Data

```bash
python scripts/generate_data.py
```

4.2 Verify Data

```bash
psql -U postgres -d risk_db

# Check record counts
SELECT 'customers' as table_name, COUNT(*) FROM staging.customers
UNION ALL
SELECT 'credit_history', COUNT(*) FROM staging.credit_history
UNION ALL
SELECT 'transactions', COUNT(*) FROM staging.transactions
UNION ALL
SELECT 'loan_applications', COUNT(*) FROM staging.loan_applications;

# Exit
\q
```

Phase 5: Running the Pipeline

5.1 Start Airflow Services

Terminal 1 - Scheduler:

```bash
airflow scheduler
```

Terminal 2 - Webserver:

```bash
airflow webserver --port 8080
```

5.2 Monitor Pipeline Execution

1. Open http://localhost:8080
2. Login with admin/admin

3. Navigate to "financial_risk_assessment" DAG
4. View execution progress:
   - Green = Success
   - Red = Failed
   - Yellow = Running

5.3 Check Task Logs

```bash
# View logs for specific task
airflow tasks logs financial_risk_assessment run_etl 2024-01-15

# Or view in UI: Click on task → View Logs
```

Phase 6: API Deployment

6.1 Start Flask API

```bash
python api/app.py

# Or in production with Gunicorn
gunicorn -w 4 -b 0.0.0.0:5000 api.app:create_app()
```

6.2 Test API Endpoints

```bash
```

```
# Health check
curl http://localhost:5000/health

# Get risk prediction
curl -X POST http://localhost:5000/api/predict \
  -H "Content-Type: application/json" \
  -d '{
    "customer_id": 123,
    "loan_amount": 50000,
    "loan_term_months": 36
  }'

# Get model metrics
curl http://localhost:5000/api/model-metrics
```

Integration Overview

How Airflow Integrates with Your Code

```
┌─────────────────────────────────────────────────────────────┐
│                    Airflow DAG                              │
│  (airflow/dags/risk_assessment_dag.py)                      │
└─────────────────────────────────────────────────────────────┘
                          │
                          │
                          ├─▶ Task 1: check_data_freshness()
                          │   └─▶ Queries staging schema
                          │
                          ├─▶ Task 2: run_data_quality_checks()
                          │   └─▶ Uses etl/data_quality.py
                          │       └─▶ DataQualityChecker class
                          │
                          ├─▶ Task 3: run_etl_pipeline()
                          │   └─▶ Uses etl/pipeline.py
                          │       └─▶ RiskETLPipeline class
                          │       └─▶ Uses etl/feature_engineering.py
                          │
                          ├─▶ Task 4: train_model()
                          │   └─▶ Uses models/training.py
                          │       └─▶ RiskPredictionModel class
                          │
                          ├─▶ Task 5: generate_batch_predictions()
```

```
│   └─▶ Uses models/predictor.py
│       └─▶ load_model(), predict_risk()
│
└─▶ Task 6: send_summary_report()
    └─▶ Collects XCom data from all tasks
```

Task Dependencies

```
check_data_freshness
        ↓
data_quality_check
        ↓
    run_etl
        ↓
  train_model
        ↓
generate_predictions
        ↓
send_summary_report
```

Data Flow

```
PostgreSQL Staging Schema
        ↓
   ETL Pipeline
        ↓
PostgreSQL Analytics Schema (risk_features)
        ↓
  Model Training
        ↓
Saved Model Files
        ↓
Batch Predictions
        ↓
PostgreSQL Predictions Schema (risk_scores)
        ↓
   Flask API
        ↓
  End Users
```

Common Operations

Daily Operations

```bash
# Check pipeline status
airflow dags list-runs -d financial_risk_assessment

# View recent logs
tail -f logs/etl.log

# Check API health
curl http://localhost:5000/health
```

Retraining Model

```bash
# Trigger full pipeline
airflow dags trigger financial_risk_assessment

# Or just retrain without full ETL
python scripts/train_model.py
```

Deploying New Model

```bash
python scripts/deploy_model.py \
    --model-path models/saved_models/risk_model.pkl \
    --version v1.1.0
```

Stopping Services

```bash
```

```
# Stop all services
./scripts/stop_services.sh

# Or manually
pkill -f "airflow scheduler"
pkill -f "airflow webserver"
pkill -f "api/app.py"
```

Troubleshooting

Issue: Airflow DAG not appearing

Solution:

```bash
# Check for Python errors
python airflow/dags/risk_assessment_dag.py

# Check scheduler logs
tail -f $AIRFLOW_HOME/logs/scheduler/latest/*.log

# Refresh DAGs
airflow dags list-import-errors
```

Issue: Database connection failed

Solution:

```bash
# Test connection
airflow connections test risk_db_connection

# Check PostgreSQL
pg_isready

# Verify credentials in config/database.yaml
```

Issue: Import errors in DAG

Solution:

```bash
# Ensure project is in Python path
export PYTHONPATH="${PYTHONPATH}:$(pwd)"

# Or add to DAG file (already done in updated version):
# sys.path.insert(0, str(project_root))
```

Issue: Model training fails

Solution:

```bash
# Check if analytics.risk_features table has data
psql -U postgres -d risk_db -c "SELECT COUNT(*) FROM analytics.risk_features;"

# If empty, run ETL first
python scripts/run_etl.py
```

Production Deployment

Environment Variables

```bash
export FLASK_ENV=production
export DB_PASSWORD=secure_password
export AIRFLOW__CORE__SQL_ALCHEMY_CONN=postgresql://...
```

Use Production WSGI Server

```bash
# Install gunicorn
pip install gunicorn

# Run API
gunicorn -w 4 -b 0.0.0.0:5000 'api.app:create_app()'
```

Configure Email Alerts

Edit `airflow/airflow.cfg`:

```ini
[email]
email_backend = airflow.utils.email.send_email_smtp

[smtp]
smtp_host = smtp.gmail.com
smtp_user = your_email@gmail.com
smtp_password = your_app_password
smtp_port = 587
smtp_mail_from = your_email@gmail.com
```

Next Steps

1. ✅ System is set up and running
2. 📊 Access Airflow UI to monitor pipelines
3. 🔍 Test API endpoints
4. 📈 View predictions in database
5. 🎨 Connect Tableau to predictions schema
6. 🚀 Deploy to production environment

For questions or issues, check the logs in the `logs/` directory!