ML Volatility Forecasting System

A modular, production-grade MLOps pipeline for short-term market risk prediction.

🌐 Overview

This project predicts **5-day realized volatility (RV_5d)** for the S&P 500 (SPY) using **market, cross-asset, and macro features**.
It evolves in three self-contained levels — from **local Dockerized ML system** to **cloud-native MLOps pipeline** with **drift monitoring, regime discovery**, and optional **RAG-based explainability**.

🚀 Project Roadmap

Level	Description	Tech Focus	Deployment
1	Local, Dockerized MVP — full ML lifecycle with explainability	XGBoost, SHAP, FastAPI, Streamlit	Local / Docker Compose
2	Production pipeline on AWS — daily batch + API + CI/CD	ECS Fargate, EventBridge, API Gateway, Terraform	AWS Cloud
3	Enterprise MLOps — SageMaker Pipelines + Model Registry + Drift Monitoring + Macro + RAG	SageMaker, Model Monitor, OpenSearch / pgvector, Great Expectations	AWS Cloud (Full Production)

🧩 Core Architecture (All Levels)

`data → features → model → inference → api → dashboard`

- **Data contracts** stay identical across levels (`raw.market`, `features_daily`, `predictions_daily`)
- **API contract** (`/predict`) remains stable from start to production
- **Infrastructure** evolves modularly (local → cloud → SageMaker)

- **All services containerized** for reproducibility

🧱 Level 1 — Local MVP (Dockerized)

Goal: Build and run the full ML lifecycle locally.

Stack:

- Python 3.11, Docker Compose
- yfinance, pandas, XGBoost/LightGBM, SHAP, FastAPI, Streamlit
- Parquet + (optional) S3 for storage

Pipeline Steps:

1. `ingest`: fetch SPY, VIX, VIX3M, TLT, HYG
2. `features`: compute ~15 leak-free features
3. `train`: model & SHAP explainability
4. `infer`: daily batch predictions
5. `api`: `/predict` endpoint (local FastAPI)
6. `dashboard`: Streamlit visualizing predictions & SHAP drivers

Key Outputs:

- `features_daily/`, `labels_daily/`, `predictions_daily/` (Parquet)
- `/predict` JSON API
- Local dashboard

Run:

```
docker compose up --build
```

Acceptance:

- One-command local run
- SHAP explainability & RMSE/QLIKE improvement ≥ 1–2 % vs baseline

---

☁️ Level 2 — AWS Serverless Production

Goal: Automate and deploy the pipeline to AWS with CI/CD.

Stack:

- AWS ECR + ECS Fargate / Lambda
- EventBridge + Step Functions
- API Gateway + CloudWatch + Terraform
- GitHub Actions for CI/CD

Daily Workflow:

```
EventBridge → ECS tasks → (ingest → features → infer) → S3 predictions
```

Weekly:

```
EventBridge → ECS task → train → metrics gate → model promotion
```

Key Additions:

- CI/CD builds containers → ECR → Terraform deploy
- CloudWatch alarms, JSON logging
- Optional DynamoDB mirror for fast `/predict`

Acceptance:

- Automated daily predictions by 08:00 local

- `/predict` accessible via API Gateway
- RMSE/QLIKE > Level 1 baseline
- Logs & alarms visible in CloudWatch

🧠 Level 3 — SageMaker + Regimes + Monitoring (+ RAG)

Goal: Production-grade MLOps with drift monitoring & macro context.

Stack:

- SageMaker Pipelines & Model Registry
- SageMaker Model Monitor + Great Expectations
- FRED macro ingestion (Lambda / ECS)
- GMM / HMM regime clustering
- Optional: OpenSearch + pgvector for RAG explainability

Enhancements:

- Add macro & regime features (≈ +20 features)
- SageMaker trains → registers → conditionally deploys
- Drift detection (PSI/KS) triggers retraining
- Optional RAG sidecar: `/daily-brief` & `/qa` with citations

Acceptance:

- Model registry gating (≥ 5 % metric gain)
- Drift & data-quality alerts active
- Macro & regime features in `features_daily`

- Optional RAG returns cited summaries

🧮 Data Contract (Shared Across Levels)

```
s3://<bucket>/
  raw.market/
  curated.market_daily/
  features_daily/
  labels_daily/
  predictions_daily/
```

Schema evolves additively (macro + regime columns). No breaking changes.

🧠 Key Features

- 🧱 Modular architecture (Level 1 → 3)
- 📦 Fully Dockerized microservices
- 📈 Walk-forward validation & SHAP explainability
- ☁️ Serverless AWS orchestration (EventBridge → ECS/Lambda)
- 🧩 SageMaker Pipelines & Model Registry for production MLOps
- ⚙️ Drift & data quality monitoring (PSI, KS, Great Expectations)
- 🧾 Optional RAG explainability with cited macro/market documents

---

📂 Repository Structure

```
ml-risk/
  services/
    ingest/
    features/
    train/
    infer/
```

```
  api/
  dashboard/
libs/
infra/                # Terraform modules
tests/
config/
docker-compose.yml
Makefile
README.md
```

🧰 Make Targets

```
make build        # build all Docker images
make ingest       # run data ingestion
make features     # build features
make train        # train model locally
make infer        # run inference
make api          # start local FastAPI server
make dash         # launch dashboard
make test         # run unit & integration tests
```

🔒 Non-Functional Goals

Attribute	Target
Reproducibility	Deterministic Docker builds, seeded models
Availability	≥ 99 % (L2 + L3)
Latency	`/predict` p95 < 2 s
Cost	Minimal (serverless, spot where possible)
Security	KMS, IAM least-privilege, Secrets Manager
Observability	Structured logs, CloudWatch dashboards, drift alerts

---

📊 Metrics

Metric	Description
RMSE	Error of predicted vs actual RV_5d
QLIKE	Realized variance loss (robust volatility metric)
Brier / AUC	For regime classification (if enabled)
PSI / KS	Feature & target drift monitoring
SHAP importance	Top contributors per prediction

🧾 Resume Highlights

- Built a **modular MLOps system** for **volatility forecasting**, evolving from **Dockerized prototype → AWS pipeline → SageMaker production**.
- Implemented **walk-forward validation**, **SHAP explainability**, and **drift-aware retraining** with **SageMaker Model Registry**.
- Designed **data-first contracts**, **CI/CD infra**, and **serverless orchestration**, ensuring scalability and reliability.
- *(Optional)* Delivered a **RAG explainability sidecar** producing **cited daily risk briefs**.

📖 Next Steps

- Add more macro sources (ISM, CPI revisions)
- Implement model registry rollback CLI
- Extend dashboard with regime visualization
- (Optional) Integrate RAG + OpenSearch for cited summaries