Real-time Lead Scoring: Production-Ready Serving Infrastructure (Salesforce Take-Home)

This repo delivers a **production-quality** reference implementation for a real-time lead scoring system using **FastAPI**, **AWS ECS Fargate** (behind an **ALB**), **Amazon SageMaker** (optional endpoint integration), **Snowflake** connectivity patterns, and **Terraform** for IaC. It is designed to handle **~300 RPS** with **p99 < 1s** under typical payload sizes, and demonstrates best practices for **security, observability, CI/CD, testing, and MLOps**.

> ✅ This project is runnable locally via Docker or `uvicorn` with a light-weight mock model. AWS components are provisioned with Terraform (scoped to a cost-conscious subset by default).

Quick Start (Local)

```
# 1) Create and activate a virtual env (optional)
python3 -m venv .venv && source .venv/bin/activate

# 2) Install dependencies
pip install -r requirements.txt

# 3) Run unit tests
pytest -q

# 4) Start the API
uvicorn app.main:app --host 0.0.0.0 --port 8080 --proxy-headers --
forwarded-allow-ips='*'
# or with Docker:
docker build -t lead-scoring-api:local .
docker run -p 8080:8080 lead-scoring-api:local
```

PROF

Open: `http://localhost:8080/docs` for Swagger.

Load Testing with Locust

We have included a load testing setup using Locust.
This allows you to simulate concurrent requests and validate system performance under load.

Setup

```
cd load_tests
pip install -r requirements.txt
locust -f load_tests/locustfile.py --host http://localhost:8080
```

High-Level Architecture

- **Ingress**: AWS ALB + AWS WAF (rate limiting + managed rules)
- **Compute**: ECS Fargate service running a **FastAPI** app (async) with structured JSON logging
- **Model Inference**:
  - Option A (default demo): In-process mock XGBoost-compatible scorer
  - Option B (prod): Call a **SageMaker real-time endpoint** over private VPC endpoint (recommended for heavy models)
- **Data Lake Writes**: Results are pushed asynchronously to **Kinesis Firehose -> S3** (parquet) with partitioning
- **Features**: 50 features accepted; schema validated with **Pydantic**
- **Observability**:
  - **Prometheus** metrics exposed (`/metrics`) + CloudWatch Container Insights
  - **AWS X-Ray** tracing (enabled via SDK and sidecar/daemon)
  - **SageMaker Model Monitor** (template) for data/quality drift
- **Security**:
  - **Cognito** (JWT) or mTLS in private mesh; Secrets in **AWS Secrets Manager**
  - **Least-privilege IAM**; ECR image scan; CI security checks (Bandit, Trivy, tfsec, Checkov)

See `docs/architecture.md` for the diagram and detailed choices.

Endpoints

- `POST /score` — returns a score 1–5 and latency metadata
- `GET /healthz` — basic liveness
- `GET /readyz` — readiness (includes model warm state)
- `GET /metrics` — Prometheus metrics

Sample request:

PROF

```
{
  "lead_id": "123",
  "features": {
    "f1": 0.12, "f2": 1.0, "f3": 0.0, "...": 0.3
  }
}
```

CI/CD

- **GitHub Actions**
  - Lint (`ruff`), type-check (`mypy`), unit tests (`pytest`, coverage)
  - Security: `bandit` (py), `trivy` (image), `tfsec` + `checkov` (Terraform)
  - Docker build & push to **ECR** (OIDC to AWS)

    - Terraform plan/apply to **staging** on main branch; manual approval for **prod**
- **Blue/Green** deployment on ECS via new task definition & target group weight shift

See `.github/workflows/ci.yml` and `infra/terraform` for details.

Snowflake Integration (Pattern)

This repo shows a lightweight pattern for obtaining features or writing scored outputs to Snowflake using **Snowflake Connector for Python** with **external authentication** (IAM role + Secrets Manager). In local mode the calls are stubbed.

Scope & Cost Decisions

- **Implemented**: Local runnable service, ECS task/service Terraform, ECR, ALB, WAF (basic), CW logs/alarms, Kinesis Firehose to S3 (delivery stream), CI security gates.
- **Documented templates**: SageMaker endpoint + Model Monitor, Cognito integration.
- **Deferred** (explain in `docs/scope_decisions.md`): PrivateLink to Snowflake, end-to-end Lakehouse table creation, full Grafana stack (CloudWatch metrics suffice for demo).

Future Improvements

- Canary deployments with AWS App Mesh
- Online feature store (Feast / SageMaker Feature Store)
- Advanced drift monitors and bias metrics with alerts to Slack (SNS)
