DDx - GCP Deployment Guide

Google Cloud Deployment Instructions

# DDx - Production-Ready Package

This repository contains a **high-level, industry-grade** AI-driven trading system codenamed **DDx**. It is built on:

- **GCP multi-region HPC** for global coverage
- **Kubernetes Federation (K3s + KubeFed)** for cluster orchestration
- **NVIDIA TensorRT** for real-time edge inference
- **Stable-Baselines3** RL with custom Gym environment
- **Autoscaling, self-healing** microservices architecture

## Core Components

1. **`ddx_trading_env.py`**
   - Custom Gym environment that handles market data feed, action space, reward shaping.

2. **`rl_train_master.py`**
   - RL training pipeline (PPO/A2C).
   - Exports final model to ONNX after training.

3. **`edge_inference_bridge.py`**
   - Lightweight Flask/TensorRT server.
   - Loads ONNX model, processes inference requests with minimal latency.

4. **Dockerfiles** (in `docker/`):
   - `Dockerfile.rl_train` builds a container for the RL training environment.
   - `Dockerfile.inference` builds a container for the inference bridge.

5. **Kubernetes Manifests** (in `k8s/`):
   - `ddx-federated-backend.yaml`, `ddx-agents.yaml`, `ddx-redis.yaml`, `ddx-frontend.yaml` for your main system.
   - `ddx-ingress.yaml` for domain/TLS.
   - `ddx-hpa.yaml` for autoscaling.
   - `ddx-cronjob.yaml` for daily or frequent retraining.

6. **Scripts** (in `scripts/`):
   - `deploy.sh`: One-click deployment script for building Docker images and applying K8s manifests.
   - `rollout.sh`: Trigger rolling updates or rollback with the new RL model.

## Deployment Steps

1. **Clone Repo / Extract Tar**
   ```bash
   tar -xvzf ddx_production_package.tar.gz
   cd ddx_production_package
   ```

2. **Build & Push Docker Images**
   - E.g.,
   ```bash
   docker build -t yourregistry.com/ddx-rl-train:latest -f docker/Dockerfile.rl_train .
   docker push yourregistry.com/ddx-rl-train:latest
   ```

3. **Deploy to K8s**
   ```bash
   scripts/deploy.sh
   ```
   This script applies all K8s resources and sets up the environment.

4. **Run RL Training**
   - Wait for the CronJob, or trigger manually:
   ```bash
   kubectl create job --from=cronjob/ddx-rl-training-cron ddx-training-manual
   ```
   After completion, the new ONNX model is saved to shared storage.

5. **Inference Bridge**
   - The system automatically updates the inference containers with the new model (or run `scripts/rollout.sh` if manual).

## Further Customization

- Integrate real-time data feed or fix historical data.
- Adjust reward shaping in `ddx_trading_env.py`.
- Expand the K8s manifests for multi-region federation.
- Add security + encryption for the bridging endpoints.

## Questions

- Post in [Issues] or chat with the system architects for advanced configurations.

Enjoy unstoppable AI-driven trading with DDx!