Qyntara AI

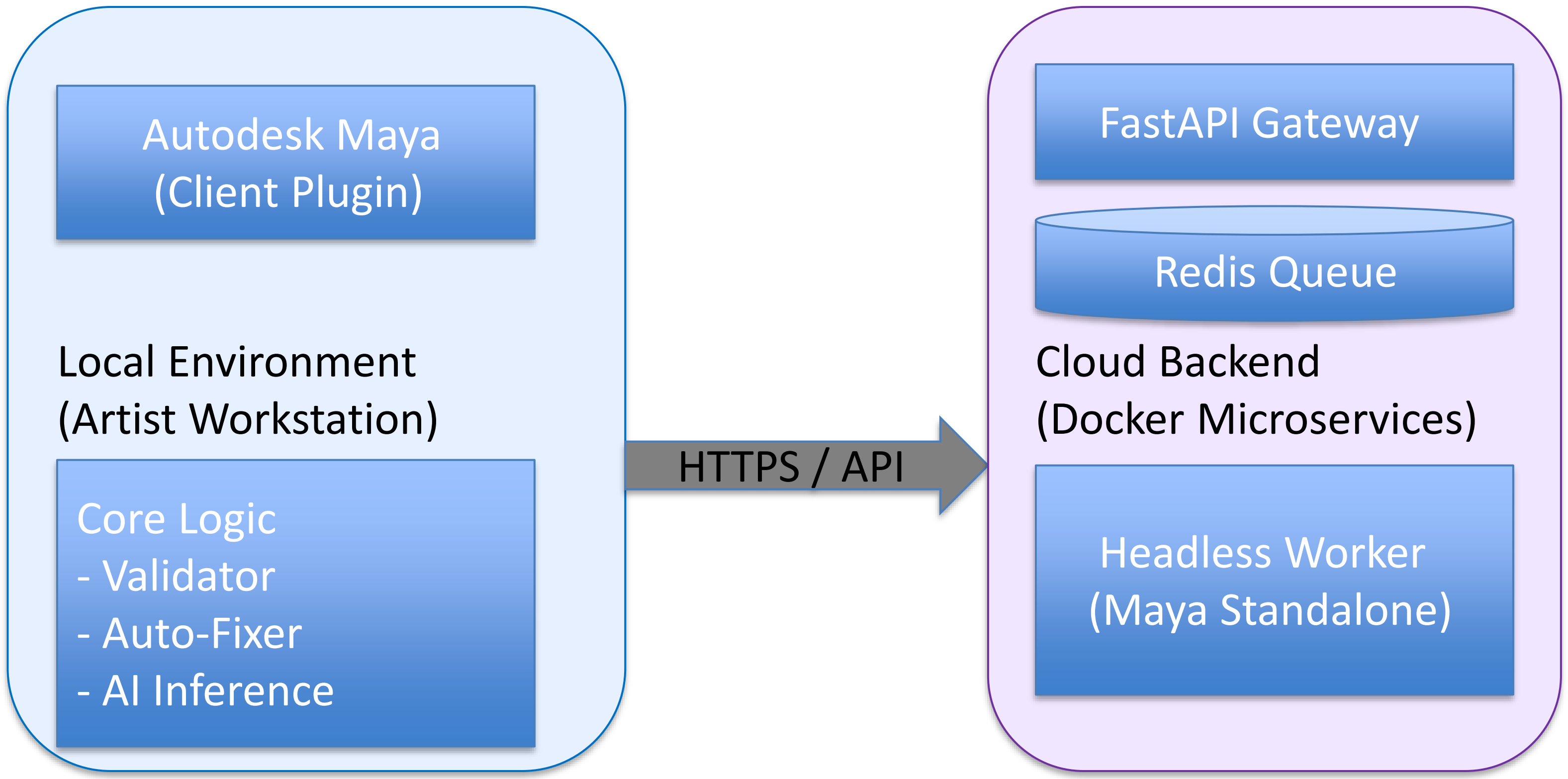
Next-Gen 3D Asset Validation & Auto-Fix System

End-to-End Architecture Overview

Executive Summary

- Hybrid System: Combines Local Maya Plugin with Cloud processing.
- Goal: Ensure Game/VR/VFX assets are technically flawless.
- Key Innovations:
- - 40+ Real-time Geometry Checks (N-gons, Open Edges)
- - Deterministic Auto-Fix Engine
- - Deep Learning Model (MeshAnomalyNet) for subjective quality
- - Dockerized Microservices for scalability

High-Level System Architecture



Client Architecture (Maya Plugin)

- Tech Stack: Python 3, PySide2 (Qt), Maya Python API.
- UI Layer: Decoupled from logic, strictly handles user signals.
- Core Logic: Shared library ('qyntara_ai.core') containing all validation rules.
- Performance: Optimized viewport overlays using OpenMaya API.
- Features:
- - Validation Dashboard
- - Smart Alignment Tools
- - Game Engine Export Presets (Unity/Unreal)

Cloud/Backend Architecture

- Tech Stack: FastAPI, Redis, Docker, Headless Maya.
- API Gateway: Handles file uploads and job validation.
- Message Broker: Redis queues valid jobs for processing.
- Worker Nodes: Scalable containers running Maya Standalone.
- Parity: Reuses the exact same 'Core' rules set as the client.
- Purpose: Allows batch processing of thousands of assets without tying up artist workstations.

AI Model: MeshAnomalyNet

- Architecture Type: PointNet-based Binary Classifier
  - Purpose: Detects non-determinist topology errors (subjective quality).
  - Input: Point Cloud (N=1024 sampled vertex points).
  - Layers:
    - 1. Convolutions (64 -> 128 -> 1024 filters) for feature extraction.
    - 2. Global Max Pooling (aggregates features).
    - 3. Fully Connected Layers (512 -> 256 -> 1) for classification.