Zama Jobs API - Architecture Design

This repository contains the complete architectural design for the Zama Jobs API, a platform for submitting long-running, asynchronous jobs with blockchain confirmation.

Overview

The Zama Jobs API enables developers to submit computational jobs that are processed asynchronously and have their final confirmation recorded on an EVM-compatible blockchain.

Key Features

- **Asynchronous Job Processing**: Submit jobs and poll for status updates
- **Blockchain Confirmation**: Final job results recorded on-chain for immutability
- **Confidential Computing**: Optional FHE (Fully Homomorphic Encryption) for privacy-preserving jobs
- **Node.js Backend**: High-performance async/await architecture with clustering
- **Idempotent Requests**: Safe retry logic with idempotency keys
- **Kong Gateway Integration**: Advanced rate limiting, validation, and metering
- **Private ERC20 Payments**: Encrypted payment processing with on-chain settlement
- **Rate Limiting & Quotas**: Fair usage with per-tenant controls
- **Comprehensive Error Handling**: Clear error codes and messages
- **Security-First Design**: OAuth 2.0 authentication with least-privilege access
- **Usage-Based Metering**: Transparent billing based on actual usage
- **Kubernetes Ready**: Complete deployment manifests with versioning strategy

Repository Structure

```
zama-jobs-api/
├── README.md              # This file - project overview
├── DESIGN.md              # Complete architecture design document
├── openapi.yml            # REST API specification
├── examples/              # Usage examples and samples
│   └── kubernetes-manifests.yaml  # Complete K8s deployment manifests
└── docs/                  # Additional documentation (optional)
```

Documentation

📋 DESIGN.md

Complete architecture design document containing:

- **Architecture Decision Pack**: API governance, platform policies, metering logic
- **Node.js Async Architecture**: High-performance clustering and queue processing
- **FHE Integration**: Confidential computing with Zama Gateway
- **Kong Gateway Integration**: Custom Lua plugins for rate limiting and metering
- **Kubernetes Deployment**: Complete manifests with versioning strategy
- **System Interface & Logic Design**: On-chain smart contract and API handler logic
- **Private ERC20 Integration**: Encrypted payments and billing
- **Reliability & Security Notes**: SLA definitions, error budgets, security measures
- **Reflection Questions**: Design trade-offs and improvement areas

🔌 OpenAPI Specification

Comprehensive REST API specification including:

- **Unified Job Submission**: Single endpoint for regular and FHE jobs with `useFHE` flag
- **Secure FHE Result Retrieval**: Dedicated endpoint for confidential computing results
- Complete endpoint definitions with request/response schemas
- Authentication and security schemes with JWT support
- Error handling with detailed error codes and examples
- Rate limiting headers and Kong Gateway integration examples
- FHE job submission and retrieval workflows
- Interactive API documentation ready for Swagger Editor

⚙️ Kubernetes Manifests

Production-ready Kubernetes deployment including:

- **API Server & Workers**: Node.js applications with proper resource limits
- **Kong Gateway**: Ingress, load balancing, and plugin configuration
- **Database Layer**: PostgreSQL StatefulSet and Redis deployment
- **Monitoring**: Prometheus and Grafana configurations
- **SSL/TLS**: Certificate management and secure communications
- **Versioning**: Support for multiple API versions simultaneously

Quick Start

API Endpoints

Method	Endpoint	Description
`POST`	`/v1/jobs`	Submit a new job (regular or FHE with `useFHE: true`)
`GET`	`/v1/jobs/{jobId}`	Get job status and details
`GET`	`/v1/jobs`	List jobs with filtering and pagination
`POST`	`/v1/jobs/{jobId}/fhe-result`	Retrieve FHE job result via secure re-encryption
`GET`	`/v1/health`	System health check

Example Usage

```
# Submit a regular compute job
curl -X POST https://api.zama.io/v1/jobs \
  -H "Authorization: Bearer YOUR_JWT_TOKEN" \
  -H "Idempotency-Key: $(uuidgen)" \
  -H "Content-Type: application/json" \
  -d '{
    "type": "compute",
    "payload": {
      "algorithm": "prime_factorization",
      "input": {"number": 104729},
      "parameters": {"timeout": 300}
    },
```

```
    "priority": "normal",
    "useFHE": false
  }'

# Submit a confidential FHE job
curl -X POST https://api.zama.io/v1/jobs \
  -H "Authorization: Bearer YOUR_JWT_TOKEN" \
  -H "Idempotency-Key: $(uuidgen)" \
  -H "Content-Type: application/json" \
  -d '{
    "type": "fhe_compute",
    "payload": {
      "algorithm": "encrypted_statistical_analysis",
      "encrypted_data_url": "https://storage.example.com/encrypted_data.enc",
      "client_public_key": "-----BEGIN PUBLIC KEY-----
\nMFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAE...\n-----END PUBLIC KEY-----",
      "parameters": {"fhe_scheme": "tfhe", "security_level": 128}
    },
    "priority": "normal",
    "useFHE": true
  }'

# Check job status
curl -X GET https://api.zama.io/v1/jobs/550e8400-e29b-41d4-a716-446655440000 \
  -H "Authorization: Bearer YOUR_JWT_TOKEN"

# Retrieve FHE job result (for completed FHE jobs)
curl -X POST https://api.zama.io/v1/jobs/550e8400-e29b-41d4-a716-446655440000/fhe-
result \
  -H "Authorization: Bearer YOUR_JWT_TOKEN" \
  -H "Content-Type: application/json" \
  -d '{
    "clientSignature": "0x1234567890abcdef...",
    "retrievalNonce": "retrieval_nonce_12345"
  }'
```

Architecture Highlights

API Governance

- **Versioning**: URL-based versioning (`/v1/`)
- **Idempotency**: 24-hour deduplication window with idempotency keys
- **Error Model**: Machine-readable codes with human-readable messages
- **Rate Limiting**: Token bucket algorithm with per-tenant quotas

Security Design

- **Authentication**: OAuth 2.0 with JWT Bearer tokens
- **Authorization**: Role-based access control (Developer, Team Lead, Admin)
- **Least Privilege**: Scoped permissions with tenant isolation
- **Token Lifecycle**: 15-minute access tokens with refresh token rotation

Blockchain Integration

- **Smart Contract**: `ZamaJobsRegistry` for on-chain confirmations
- **Replay Protection**: Nonce-based with job ID uniqueness
- **Access Control**: Authorized confirmator pattern
- **Gas Optimization**: Efficient confirmation transactions

Reliability Engineering

- **SLA**: 99.9% availability with p95 latency < 500ms
- **Error Budget**: 1h/month with automated alerting
- **Circuit Breakers**: Fail fast patterns for downstream services
- **Monitoring**: Comprehensive metrics and distributed tracing

Technology Stack (Proposed)

- **API Gateway**: Kong Gateway with custom Lua plugins
- **Backend**: Node.js with clustering for high-performance async I/O
- **Queue**: Redis-based job queues (BullMQ) with priority handling
- **Database**: PostgreSQL for job metadata, Redis for caching and sessions
- **Storage**: Object storage for large results and encrypted data
- **Blockchain**: fhEVM with confidential smart contracts
- **FHE**: Zama Gateway integration for confidential computing
- **Payments**: Private ERC20 tokens with encrypted transactions - OPTIONAL
- **Monitoring**: Prometheus + Grafana, OpenTelemetry tracing
- **Authentication**: OAuth 2.0 with JWT Bearer tokens
- **Deployment**: Kubernetes with GitOps and versioning strategy

Usage & Metering

The platform includes comprehensive usage tracking and metering capabilities:

- **Real-time Usage Collection**: Kong-based monitoring of all API interactions
- **Usage Analytics**: Track job submissions, completions, storage usage, and API patterns
- **Configurable Quotas**: Per-tenant limits and rate controls
- **Comprehensive Auditing**: Full audit trail for compliance and monitoring
- **FHE Resource Tracking**: Monitor confidential computing resource usage
- **Blockchain Integration**: Log and track all on-chain confirmation transactions

Development & Testing

Local Development

```
# Clone the repository
git clone https://github.com/your-org/zama-jobs-api.git
cd zama-jobs-api

# Start local development environment
docker-compose up -d

# Run tests
npm test
```

Testing the API

```
# Start with mock data
curl -X POST http://localhost:3000/v1/jobs \
  -H "Content-Type: application/json" \
  -d '{"type": "compute", "payload": {"test": true}}'

# View API documentation
open http://localhost:3000/docs
```

NOTE: This repository contains architectural design documentation for the Zama Jobs API challenge. Implementation is opinionated and focuses on demonstrating architectural decision-making across API governance, security, reliability, and blockchain integration using publicy available information I was able to retrieve regarding the setup which would match a real scenario ie -> Kong Gateway, Asynchronus IO etc