Test 2 Title

Test Author

1 Cloud Marketplace Implementation Plan

This document outlines our phased approach to implementing a Cloud Marketplace backend using a modern serverless architecture. We have designed a two-phase plan to ensure incremental progress and modular development.

1.1 Serverless Architecture Overview

The Cloud Marketplace backend will be implemented using a serverless approach, providing:

- **Cost efficiency**: Pay-per-use model with generous free tiers for development
- **Scalability**: Automatic scaling based on demand
- **Operational simplicity**: No server provisioning or maintenance
- **Modern CI/CD**: Seamless deployment with GitHub Actions

1.2 Technology Stack

Component	Technology	Purpose
Authentication	Auth0	User management, role-based access control
Serverless Functions	Netlify Functions	API endpoints and business logic
Database	FaunaDB	Persistent data storage
Frontend Hosting	GitHub Pages	Static asset delivery
Payment Processing	Stripe	Secure payment handling

1.3 Phase 1: Core Infrastructure & Standard Integrations

Phase 1 focuses on implementing standard cloud services to create a working foundation.

1.3.1 Authentication (Auth0)

- User registration and login flows
- Role-based access control for:

  - `cloud-user`: End users deploying applications
  - `cloud-provider`: Node operators providing infrastructure
  - `cloud-operator`: Dashboard operators managing the marketplace

- JWT validation for secure API access
- Profile management and settings

1.3.2 Database (FaunaDB)

Collections structure: - `users`: Profile data and settings - `deployments`: Application deployment metadata - `provider-requests`: Cloud provider participation requests - `operator-settings`: Dashboard operator configurations - `maintenance-windows`: Scheduled maintenance events

1.3.3 API Endpoints (Netlify Functions)

```
functions/
|-- auth/                  # Authentication-related functions
|  |-- callback.js         # Auth0 callback handler
|  `-- session.js          # Session management
|-- cloud-user/            # Cloud user endpoints
|  |-- deployments.js      # List user's deployments
|  |-- create-deployment.js # Create new deployment
|  `-- deployment-detail.js # Get single deployment details
|-- cloud-provider/        # Cloud provider endpoints
|  |-- register.js         # Register as provider
|  |-- list-nodes.js       # List provider's nodes
|  `-- maintenance.js      # Schedule maintenance
|-- cloud-operator/        # Cloud operator endpoints
|  |-- settings.js         # Dashboard configuration
|  |-- pricing.js          # Update pricing configuration
|  `-- provider-requests.js # Manage provider requests
`-- payments/              # Payment processing
   |-- create-intent.js    # Create payment intent
   |-- methods.js          # Manage payment methods
   `-- webhook.js          # Handle Stripe webhooks
```

1.3.4 Payment Processing (Stripe)

- Payment method registration and storage
- Secure checkout flow integration
- Subscription management for recurring payments
- Webhook handlers for payment events
- Transaction history and receipts

1.3.5 Frontend Integration

- Auth0 authentication flow
- Protected routes for different roles
- API client services

- Stripe Elements for payment forms
- Live data integration with context providers

1.4 Phase 2: Advanced Integrations

After Phase 1 is stable and tested, Phase 2 will integrate specialized functionality.

1.4.1 Wallet Integration

- Currency swap functionality
- Wallet balance monitoring
- Auto-top up configuration
- Transaction history and status tracking

1.4.2 Blockchain & Deployment

- Wallet connection
- Deployment of workloads to nodes
- Resource monitoring and scaling
- Deployment status updates

1.4.3 Provider Alliance System

- Alliance formation workflow
- Node contribution management
- Revenue sharing implementation
- SLA monitoring and enforcement

1.4.4 Advanced Features

- Staking discount tier implementation
- Backup VM configuration
- Advanced node selection criteria
- Low-balance alerts and notifications

1.5 Implementation Milestones

1.5.1 Phase 1 Milestones

1. **Initial Setup & Configuration**
    - Auth0 tenant setup
    - FaunaDB database provisioning
    - Netlify Functions environment configuration
2. **Authentication Implementation**
    - Auth0 integration
    - Role-based access control
    - Protected API middleware
3. **Database Schema & API Development**

- Collection structure creation
- Core CRUD endpoints
- Data validation and error handling

4. **Payment Integration**
    - Stripe integration
    - Checkout flows
    - Webhook handling
5. **Testing & Validation**
    - End-to-end user flow testing
    - Payment processing validation
    - Security assessment

1.5.2 Phase 2 Milestones

1. **Wallet Integration**
    - Wallet connection
    - Currency swap implementation
    - Balance monitoring
2. **Network Integration**
    - Blockchain connectivity
    - Deployment workflows
    - Resource management
3. **Alliance & Collaboration Features**
    - Provider onboarding
    - Alliance formation
    - Revenue sharing
4. **Advanced Marketplace Features**
    - Discount tiers
    - Advanced deployment options
    - Monitoring and notifications

1.6 Free Tier Considerations

The selected technologies offer generous free tiers suitable for development and initial production:

- **Auth0**: 7,000 active users and unlimited logins
- **Netlify Functions**: 125,000 invocations/month, 100 GB bandwidth
- **FaunaDB**: 100K reads, 50K writes, 500K compute operations daily
- **GitHub Pages**: Unlimited for public repositories
- **Stripe**: No monthly fees, pay per transaction (2.9% + $0.30)

1.7 Next Steps

1. Set up Auth0 tenant and configure application settings
2. Create FaunaDB database with initial collections
3. Implement first authentication endpoints with Netlify Functions
4. Update frontend to integrate with Auth0
5. Set up CI/CD pipeline for automated deployment

This phased approach ensures we can progressively build a robust, scalable marketplace while validating core functionality before integrating specialized components.

