Kế Hoạch Migration TheShoeBolt từ Node.js sang Deno

Table of Contents

1. Tóm Tắt Báo Cáo

Báo cáo này trình bày kế hoạch chi tiết để migration dự án TheShoeBolt từ môi trường runtime Node.js hiện tại sang Deno. Dự án là một e-commerce platform phức tạp sử dụng NestJS framework với multi-database architecture và nhiều external integrations.

Kết luận chính: Migration sang Deno là khả thi về mặt kỹ thuật nhưng đòi hỏi effort rất lớn và có nhiều trade-offs. Khuyến nghị cân nhắc kỹ lưỡng ROI trước khi thực hiện.

2. Bước 1: Phân Tích Hiện Trạng

2.1. Kiến Trúc Dự Án Hiện Tại

TheShoeBolt là một NestJS e-commerce platform với:

- **Framework:** NestJS 10.0.0 với TypeScript
- **Databases:** PostgreSQL, MongoDB, Redis, Elasticsearch
- **Authentication:** Clerk + Passport + JWT
- **Message Queue:** RabbitMQ với amqplib
- **Payment:** Stripe integration
- **Email:** Resend service
- **Real-time:** Socket.io WebSocket
- **Security:** Helmet, compression, bcryptjs

2.2. Dependencies Analysis

2.2.1. Core Dependencies (28 packages)

Package	Version	Category	Deno Compatibility
@nestjs/core	^10.0.0	Framework	⚠️ Requires Node.js compatibility
@clerk/backend	^2.3.1	Auth	⚠️ Via npm: import
@elastic/elasticsearch	^8.10.0	Search	⚠️ Via npm: import
typeorm	^0.3.17	ORM	✅ Needs polyfills
mongoose	^7.8.7	ODM	✅ Needs polyfills
socket.io	^4.7.2	WebSocket	⚠️ Via npm: import
amqplib	^0.10.3	Message Queue	⚠️ Via npm: import
stripe	^14.5.0	Payment	⚠️ Via npm: import

2.2.2. Development Dependencies (30 packages)

- TypeScript, Jest, ESLint, Prettier - ✅ Fully compatible
- NestJS CLI - ❌ Not supported natively
- ts-node, tsconfig-paths - ❌ Node.js specific

2.3. Cấu Trúc Module

```
src/
├─── modules/
│    ├─── Infrastructure/clerk/     # Clerk authentication
│    ├─── admin/                    # Admin functionality
│    ├─── auth/                     # Auth logic
│    ├─── chat/                     # Socket.io chat
│    ├─── elasticsearch/            # Search service
│    ├─── emails/                   # Resend integration
│    ├─── payments/                 # Stripe processing
│    ├─── queues/                   # RabbitMQ handling
│    └─── users/                    # User management
├─── config/                      # Environment configs
├─── database/                    # TypeORM setup
└─── common/                      # Shared utilities
```

3. Bước 2: Đánh Giá Tính Tương Thích

3.1. Compatibility Matrix

Component	Native Deno	npm: Support	Effort Level	Alternative
NestJS Framework	❌	⚠️	HIGH	Fresh, Oak, Hono

Component	Native Deno	npm: Support	Effort Level	Alternative
TypeScript	□	□	LOW	Native support
PostgreSQL	□	□	MEDIUM	deno.land/x/postgres
MongoDB	□□	□	HIGH	deno_mongo (limited)
Redis	□	□	MEDIUM	deno.land/x/redis
Elasticsearch	□	□	HIGH	Custom HTTP client
TypeORM	□	□□	VERY HIGH	Custom ORM needed
Mongoose	□	□□	VERY HIGH	Native MongoDB driver
Socket.io	□	□	HIGH	WebSocket API
RabbitMQ	□	□	HIGH	Custom AMQP client
Stripe	□	□	MEDIUM	HTTP API calls
Clerk	□	□	MEDIUM	HTTP API calls

3.2. Thách Thức Chính

3.2.1. 1. Framework Dependencies

- NestJS không có native Deno support
- Decorators và metadata system cần compatibility layer
- Dependency injection container cần adaptation

3.2.2. 2. Database Layer

- TypeORM không tương thích với Deno
- Mongoose cần Node.js polyfills
- Connection pooling và transaction handling khác biệt

3.2.3. 3. External Services

- Tất cả SDK đều cần npm: imports
- Authentication flows cần refactoring
- WebSocket handling khác biệt

4. Bước 3: Kế Hoạch Migration Chi Tiết

4.1. Phase 1: Foundation Setup (4-6 tuần)

4.1.1. Objectives

- Setup Deno development environment
- Create basic project structure
- Implement core utilities

4.1.2. Tasks

1. **Environment Setup**
   - Install Deno 2.0+
   - Configure VS Code với Deno extension
   - Setup import maps và deno.json
2. **Project Structure**
   - Tạo new Deno project structure
   - Convert tsconfig.json sang deno.json
   - Setup import maps cho dependencies
3. **Core Utilities Migration**
   - Convert common utilities
   - Implement logging system
   - Setup configuration management

4.1.3. Deliverables

- Deno project skeleton
- Basic utilities working
- Development environment ready

4.2. Phase 2: Database Layer (6-8 tuần)

4.2.1. Objectives

- Replace TypeORM và Mongoose
- Implement database connections
- Create repository patterns

4.2.2. Tasks

1. **PostgreSQL Migration**

    - Replace TypeORM với deno.land/x/postgres
    - Implement connection pooling
    - Create repository interfaces
2. **MongoDB Migration**
    - Replace Mongoose với native MongoDB driver
    - Implement document schemas
    - Create ODM-like abstractions
3. **Redis & Elasticsearch**
    - Implement Redis client với deno.land/x/redis
    - Create Elasticsearch HTTP client
    - Setup caching layer

4.2.3. Deliverables

- All databases connected
- Repository patterns implemented
- Data access layer working

4.3. Phase 3: Core Services (8-10 tuần)

4.3.1. Objectives

- Migrate business logic
- Implement authentication
- Setup external integrations

4.3.2. Tasks

1. **Authentication System**
    - Replace Clerk SDK với HTTP API calls
    - Implement JWT handling
    - Create auth middleware
2. **Payment Integration**
    - Replace Stripe SDK với HTTP API
    - Implement webhook handling
    - Create payment abstractions
3. **Email & Notifications**
    - Replace Resend SDK với HTTP API

- Implement email templates
- Setup notification system

4.3.3. Deliverables

- Authentication working
- Payment processing functional
- Email system operational

4.4. Phase 4: Advanced Features (6-8 tuần)

4.4.1. Objectives

- Implement real-time features
- Setup message queuing
- Complete API endpoints

4.4.2. Tasks

1. **WebSocket Implementation**
   - Replace Socket.io với native WebSocket
   - Implement chat functionality
   - Setup real-time notifications
2. **Message Queue**
   - Replace RabbitMQ client với HTTP API
   - Implement job processing
   - Setup background tasks
3. **API Completion**
   - Migrate all REST endpoints
   - Implement OpenAPI documentation
   - Setup health checks

4.4.3. Deliverables

- Full API functionality
- Real-time features working
- Production-ready system

5. Bước 4: Risk Assessment & Recommendations

5.1. High-Risk Areas

5.1.1. 1. Performance Impact

- npm: imports có performance overhead
- Database connection handling khác biệt
- Memory usage patterns thay đổi

5.1.2. 2. Ecosystem Maturity

- Nhiều packages chưa có Deno native support
- Community support hạn chế
- Documentation thiếu

5.1.3. 3. Team Learning Curve

- Deno concepts mới (permissions, imports)
- Debugging tools khác biệt
- Deployment strategies mới

5.2. Mitigation Strategies

1. **Gradual Migration**
    - Implement parallel systems
    - A/B testing approach
    - Rollback capabilities
2. **Performance Monitoring**
    - Benchmark critical paths
    - Monitor memory usage
    - Track response times
3. **Team Training**
    - Deno workshops
    - Documentation creation
    - Best practices establishment

5.3. Cost-Benefit Analysis

5.3.1. Costs

- **Development Time:** 24-32 tuần (6-8 tháng)
- **Team Effort:** 2-3 senior developers full-time
- **Risk Factor:** HIGH - nhiều unknowns
- **Maintenance:** Increased complexity initially

5.3.2. Benefits

- **Security:** Deno's permission system
- **Performance:** Potential improvements
- **Modern Runtime:** Latest JavaScript features
- **TypeScript Native:** No compilation step

5.4. Recommendations

5.4.1. Option 1: Full Migration (NOT RECOMMENDED)

- **Pros:** Complete Deno benefits
- **Cons:** Very high risk, long timeline, uncertain ROI
- **Timeline:** 6-8 tháng
- **Risk Level:** VERY HIGH

5.4.2. Option 2: Hybrid Approach (RECOMMENDED)

- Keep Node.js for complex integrations
- Use Deno for new microservices
- Gradual component migration
- **Timeline:** 3-4 tháng cho pilot
- **Risk Level:** MEDIUM

5.4.3. Option 3: Stay with Node.js (ALTERNATIVE)

- Focus on upgrading Node.js version
- Improve current architecture
- Add Deno for specific use cases
- **Timeline:** 1-2 tháng
- **Risk Level:** LOW

6. Kết Luận

Migration TheShoeBolt từ Node.js sang Deno là một dự án phức tạp với nhiều thách thức kỹ thuật. Mặc dù khả thi về mặt kỹ thuật, ROI không rõ ràng và risk level rất cao.

Khuyến nghị cuối cùng: Thực hiện Hybrid Approach với pilot project để đánh giá thực tế trước khi commit full migration.