E-Commerce Platform Technical Documentation

System Architecture

Our e-commerce platform is built on a microservices architecture using Kubernetes for orchestration. The platform consists of several key services:

1. **User Service** - Handles user authentication, registration, and profile management

2. **Product Catalog Service** - Manages product listings, categories, and search functionality

3. **Inventory Service** - Tracks product availability across warehouses

4. **Order Service** - Processes customer orders and manages order lifecycle

5. **Payment Service** - Integrates with payment processors to handle transactions

6. **Shipping Service** - Calculates shipping options and costs

7. **Analytics Service** - Collects and processes user behavior and sales data

Each service is containerized using Docker and deployed to our Kubernetes cluster. Services communicate via REST APIs and asynchronous messaging using Apache Kafka.

Database Architecture

We use a polyglot persistence approach with specialized databases for different services:

- PostgreSQL for transactional data (orders, users, products)

- MongoDB for product catalog (flexible schema for diverse product attributes)

- Redis for caching and session management

- Elasticsearch for full-text search capabilities

- TimescaleDB for time-series analytics data

API Documentation

Authentication

All API endpoints except for public product listings require authentication using JWT tokens.

Endpoint: `/api/v1/auth/login`

Method: POST

Request Body:

```
{
"email": "user@example.com",
"password": "securepassword"
}
```

Response:

```
{
"token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
"expires_at": "2023-12-31T23:59:59Z"
}
```

Product Search

Endpoint: `/api/v1/products/search`

Method: GET

Query Parameters:

- `q`: Search query
- `category`: Category ID (optional)
- `price_min`: Minimum price (optional)
- `price_max`: Maximum price (optional)
- `page`: Page number (default: 1)
- `per_page`: Results per page (default: 20)

Response:

```
{
"total": 150,
"page": 1,
"per_page": 20,
"results": [
{
"id": "prod-12345",
"name": "Wireless Headphones",
"price": 99.99,
"category": "Electronics",
"rating": 4.5,
"image_url": "https://example.com/images/headphones.jpg"
},
...
]
}
```

Deployment Pipeline

We use GitLab CI/CD for our continuous integration and deployment pipeline:

1. Code is pushed to GitLab repository
2. CI pipeline runs unit tests, integration tests, and security scans
3. When tests pass, Docker images are built and pushed to our private registry
4. Staging deployment occurs automatically for the main branch
5. Production deployment requires manual approval

Performance Monitoring

We use Prometheus and Grafana for monitoring system performance:

- Service response times
- Error rates
- CPU and memory usage
- Database query performance
- Message queue lengths

Custom dashboards are available for each service team to monitor their specific metrics.

Security Guidelines

- All API endpoints must use HTTPS
- Passwords must be hashed using bcrypt
- Database credentials are stored in Kubernetes secrets
- JWT tokens expire after 24 hours
- Input validation is required for all API endpoints
- Regular security audits are conducted by third-party consultants

Disaster Recovery

Our disaster recovery plan includes:

- Automated database backups every 6 hours
- Geo-replicated data across multiple AWS regions
- Regular disaster recovery drills
- Automated failover for critical services
- Documentation for manual recovery procedures

Recovery Time Objective (RTO): 4 hours

Recovery Point Objective (RPO): 6 hours