CloudWatch: Kubernetes Resource Monitoring Dashboard PRD

1. Product Overview

1.1 Product Vision

CloudWatch is a comprehensive Kubernetes monitoring solution that provides real-time visibility into cluster resources, application metrics, and system performance through an intuitive dashboard interface.

1.2 Target Users

- DevOps Engineers
- Site Reliability Engineers (SREs)
- System Administrators
- Development Teams

2. Technical Architecture

2.1 Technology Stack

- Frontend : React.js
- Backend :
  - Primary: Go
  - Secondary: Node.js
- Database : MongoDB
- Monitoring Stack :
  - Prometheus
  - Grafana

2.2 System Components

1. Frontend Dashboard

- Real-time metrics visualization
- Interactive charts and graphs
- Customizable dashboards
- Responsive design for various screen sizes

2. Go Backend Service

- Kubernetes API integration
- Metrics collection and processing
- AWS CloudWatch metrics export
- RESTful API endpoints

3. Node.js Backend Service

- User authentication and authorization
- Dashboard configuration management
- Alert management

4. MongoDB Database

- Store user preferences
- Dashboard configurations
- Historical metrics data
- Alert configurations

3. Feature Requirements

3.1 Core Features Cluster Monitoring

- Real-time node status and health
- Pod resource utilization
- Container metrics
- Network statistics
- Storage metrics Resource Visualization
- CPU usage trends
- Memory consumption
- Network I/O
- Storage utilization
- Pod scaling events Alerting System
- Configurable alert thresholds
- Multiple notification channels
- Alert history
- Alert acknowledgment

3.2 Advanced Features Custom Metrics

- Support for custom Prometheus exporters
- User-defined metrics
- Application-specific metrics Analytics
- Historical trend analysis
- Resource optimization recommendations
- Cost analysis
- Performance bottleneck detection

4. User Interface Requirements

4.1 Dashboard Layout

- Modular widget system
- Drag-and-drop customization
- Dark/Light theme support
- Responsive design

4.2 Data Visualization

- Line graphs
- Bar charts
- Heat maps
- Pie charts
- Status indicators

5. Integration Requirements

5.1 Kubernetes Integration

- Support for multiple clusters
- RBAC integration
- Service discovery
- Pod auto-detection

5.2 AWS Integration

- CloudWatch metrics import

- EKS cluster support
- IAM authentication

6. Non-Functional Requirements

6.1 Performance

- Dashboard loading time < 2 seconds
- Metric update interval: 5-60 seconds (configurable)
- Support for up to 1000 concurrent users
- Support monitoring of up to 100 nodes

6.2 Security

- HTTPS encryption
- JWT authentication
- Role-based access control
- API key management

6.3 Availability

- 99.9% uptime
- Automatic failover
- Data backup and recovery

7. Development Phases

Phase 1: Core Infrastructure

- Basic dashboard setup
- Kubernetes API integration
- Essential metrics collection

Phase 2: Enhanced Monitoring

- Advanced metrics
- Custom exporters
- Alert system

Phase 3: Advanced Features

- Analytics
- Recommendations
- Custom dashboards

8. Testing Requirements

8.1 Testing Environments

- Development
- Staging
- Production

8.2 Test Types

- Unit tests
- Integration tests
- End-to-end tests
- Performance tests
- Security tests

9. Deployment

9.1 Infrastructure

- Netlify for frontend hosting
- Containerized backend services
- Managed MongoDB service

9.2 CI/CD

- Automated build pipeline
- Continuous integration
- Automated deployment
- Environment promotion

10. Documentation

10.1 Required Documentation

- API documentation
- User manual
- Installation guide
- Architecture documentation
- Contributing guidelines

11. Success Metrics

11.1 Technical Metrics

- System uptime
- Response time
- Error rates
- API performance

11.2 User Metrics

- User adoption rate
- Feature usage statistics
- User satisfaction scores
- Support ticket volume