🌐 Kimi-Dev-72B Cloud Browser Service

A comprehensive enterprise-grade cloud browser platform with AI-powered code analysis using Kimi-Dev-72B integration.

🎯 Features

🔒 Enterprise Security

- JWT authentication with MFA support
- Role-based access control (Admin/User)
- XSS protection, CSRF prevention, and rate limiting
- Session audit logging and monitoring
- Secure container isolation

🌐 Cloud Browser Service

- Docker-based remote browser sessions
- Real-time browser streaming via VNC
- Multi-browser support (Firefox, Chrome, etc.)
- Session management and lifecycle control
- Resource allocation and monitoring

🤖 AI-Powered Code Analysis

- Kimi-Dev-72B integration for code debugging
- GitHub repository analysis
- Code snippet analysis and suggestions
- Real-time analysis results

- Multiple programming language support

💻 Modern Web Interface

- Responsive React frontend with TypeScript
- Professional UI with TailwindCSS
- Real-time notifications and updates
- Admin dashboard for system management
- User-friendly session management

🏗️ Architecture

```
┌─────────────────┐    ┌─────────────────┐    ┌─────────────────┐
│   Frontend      │    │     Backend     │    │   Browser       │
│   (React)       │◄──►│    (Flask)      │◄──►│  Containers     │
│   Port: 3000    │    │   Port: 5000    │    │  (Docker)       │
└─────────────────┘    └─────────────────┘    └─────────────────┘
                              │
                              ▼
                       ┌─────────────────┐
                       │     Redis       │
                       │  (Sessions)     │
                       │   Port: 6379    │
                       └─────────────────┘
```

🚀 Quick Start

Prerequisites

- Docker and Docker Compose
- 4GB+ RAM

- 10GB+ free disk space

1. Clone and Start

```
# Start all services
bash start.sh

# Or manually with docker-compose
docker-compose up -d
```

2. Access the Application

- **Frontend**: http://localhost:3000
- **Backend API**: http://localhost:5000
- **Health Check**: http://localhost:5000/api/v1/health

3. Login

Admin Account:
- Email: `admin@secure-kimi.local`
- Password: `SecureKimi2024!`

Test User Account:
- Email: `user@example.com`
- Password: `password123`

📁 Project Structure

```
cloud-browser-service/
├── cloud-browser-backend/          # Flask API backend
│   ├── src/
│   │   ├── api/                    # API endpoints
│   │   ├── auth/                   # Authentication & security
│   │   ├── models/                 # Database models
│   │   ├── services/               # Business logic
│   │   └── utils/                  # Helper utilities
│   ├── docker/                     # Browser container configs
│   ├── database/                   # SQLite database
│   └── logs/                       # Application logs
├── cloud-browser-frontend/         # React frontend
│   ├── src/
│   │   ├── components/             # Reusable components
│   │   ├── pages/                  # Application pages
│   │   ├── services/               # API services
│   │   ├── types/                  # TypeScript definitions
│   │   └── utils/                  # Helper utilities
│   └── dist/                       # Built production files
├── docker-compose.yml              # Multi-service orchestration
└── start.sh                        # Quick start script
```

🔧 Development Setup

Backend Development

```
cd cloud-browser-backend

# Create virtual environment
python -m venv venv
source venv/bin/activate  # Linux/Mac
# or venv\\Scripts\\activate  # Windows

# Install dependencies
pip install -r requirements.txt

# Run development server
python src/main.py
```

Frontend Development

```
cd cloud-browser-frontend

# Install dependencies
pnpm install

# Start development server
pnpm run dev

# Build for production
pnpm run build
```

🐳 Docker Services

Core Services

- **backend**: Flask API server
- **frontend**: Nginx-served React app
- **redis**: Session and cache storage

Browser Containers

- **browser-template**: Base template for browser instances
- Dynamically created containers for each session
- VNC access on port 5900
- WebDriver on port 4444

🔐 Security Features

Authentication

- JWT tokens with refresh mechanism
- Password hashing with bcrypt
- 2FA support with TOTP
- Account lockout after failed attempts

Authorization

- Role-based access control
- Admin-only endpoints protection
- Resource ownership validation
- API rate limiting

Container Security

- Isolated browser environments
- Resource limits and quotas
- Network segmentation
- Secure cleanup procedures

📊 Monitoring & Logging

Health Checks

- `/api/v1/health` - Basic service status
- `/api/v1/health/detailed` - Component health
- `/api/v1/health/ready` - Readiness probe
- `/api/v1/health/live` - Liveness probe

Logging

- Structured JSON logging
- Security event logging
- Access request logging
- Container lifecycle logging

Metrics

- Active session count
- Resource utilization
- API response times
- Error rates

🛠️ API Documentation

Authentication Endpoints

```
POST /api/v1/auth/register     # User registration
POST /api/v1/auth/login        # User login
POST /api/v1/auth/logout       # User logout
POST /api/v1/auth/refresh      # Token refresh
GET  /api/v1/auth/profile      # User profile
PUT  /api/v1/auth/profile      # Update profile
```

Session Management

```
GET    /api/v1/sessions        # List user sessions
POST   /api/v1/sessions        # Create new session
GET    /api/v1/sessions/{id}   # Get session details
PUT    /api/v1/sessions/{id}   # Update session
DELETE /api/v1/sessions/{id}   # Delete session
```

Code Analysis

```
POST /api/v1/kimi/analyze/repo     # Analyze repository
POST /api/v1/kimi/analyze/code     # Analyze code snippet
POST /api/v1/kimi/debug           # Debug assistance
GET  /api/v1/kimi/analysis/{id}   # Get analysis results
```

Admin Operations

```
GET    /api/v1/admin/users         # List all users
PUT    /api/v1/admin/users/{id}    # Manage user
GET    /api/v1/admin/sessions      # All sessions
GET    /api/v1/admin/stats         # System statistics
GET    /api/v1/admin/audit         # Audit logs
```

🔧 Configuration

Environment Variables

```
# Backend (.env)
FLASK_ENV=production
SECRET_KEY=your-secret-key
DATABASE_URL=sqlite:///database/app.db
REDIS_URL=redis://redis:6379
DOCKER_HOST=unix:///var/run/docker.sock
KIMI_API_URL=https://api.kimi.ai
KIMI_API_KEY=your-kimi-api-key

# Frontend (.env)
VITE_API_URL=http://localhost:5000/api/v1
VITE_WS_URL=ws://localhost:5000
```

Docker Compose Override

Create `docker-compose.override.yml` for local customizations:

```
version: '3.8'
services:
  backend:
    environment:
      - FLASK_ENV=development
    volumes:
      - ./cloud-browser-backend/src:/app/src
```

🚀 Deployment

Production Deployment

1. Update environment variables
2. Configure reverse proxy (nginx/Apache)
3. Set up SSL certificates
4. Configure monitoring and logging
5. Set up backup procedures

Scaling

- Horizontal scaling with load balancer
- Redis cluster for session storage
- Container orchestration with Kubernetes
- CDN for static assets

🧪 Testing

Backend Tests

```
cd cloud-browser-backend
python -m pytest tests/
```

Frontend Tests

```
cd cloud-browser-frontend
pnpm test
```

Integration Tests

```
# Start services in test mode
docker-compose -f docker-compose.test.yml up
```

📋 Troubleshooting

Common Issues

1. **Port conflicts**: Check if ports 3000, 5000, 6379 are available
2. **Docker permission denied**: Add user to docker group
3. **Container startup fails**: Check logs with `docker-compose logs`
4. **Browser sessions not starting**: Verify Docker socket access

Logs

```
# View all logs
docker-compose logs -f

# View specific service logs
docker-compose logs -f backend
docker-compose logs -f frontend
```

🤝 Contributing

1. Fork the repository
2. Create feature branch (`git checkout -b feature/amazing-feature`)
3. Commit changes (`git commit -m 'Add amazing feature'`)
4. Push to branch (`git push origin feature/amazing-feature`)
5. Open Pull Request

📄 License

This project is licensed under the MIT License - see the LICENSE file for details.

🆘 Support

- 📧 Email: support@kimi-dev.com
- 💬 Discord: Kimi-Dev Community
- 📚 Documentation: docs.kimi-dev.com
- 🐛 Issues: GitHub Issues

Built with ❤️ by the Kimi-Dev team