RBarros Deployment

Complete Docker deployment setup for the RBarros insurance application with Vue.js frontend and Node.js backend.

🏗 Architecture

- **Frontend**: Vue.js application with Nginx
- **Backend**: Node.js/Express API server
- **Database**: MySQL 8.0
- **Reverse Proxy**: Nginx with load balancing
- **CI/CD**: GitHub Actions with secrets integration

🚀 Quick Start

Local Development

1. **Clone with submodules**:

   ```
   git clone --recurse-submodules <repository-url>
   cd rbarros-deployment
   ```

2. **Setup environment**:

   ```
   make setup
   # Edit .env with your local values
   ```

3. **Start services**:

   ```
   # Development mode (with hot reload)
   make dev

   # Production mode
   make prod
   ```

4. **Access the application**:

   - **Application**: http://localhost (via Nginx)
   - **Frontend**: http://localhost:8080 (both environments)
   - **Backend API**: http://localhost:3000
   - **Database**: localhost:3306

Production Deployment

The application automatically deploys via GitHub Actions when you push to the `main` branch.

🐳 Docker Compose Configuration

This project uses a **clean 3-file setup**:

`docker-compose.base.yml` (Base Configuration)

- **Shared configuration** for all environments
- Defines services without environment-specific settings
- No port mappings (defined in override files)

`docker-compose.prod.yml` (Production Overrides)

- **Production-specific settings**
- Port mapping: `8080:80` (Nginx serving static files)
- Used with: `make prod`

`docker-compose.dev.yml` (Development Overrides)

- **Development-specific settings**
- Port mapping: `8080:8080` (Vue CLI dev server)
- Volume mounts for hot reloading
- Used with: `make dev`

Usage:

```
# Production
make prod
# or: docker-compose -f docker-compose.base.yml -f docker-compose.prod.yml up -d

# Development
make dev
# or: docker-compose -f docker-compose.base.yml -f docker-compose.dev.yml up -d
```

Benefits:

- ✅ No port conflicts between environments
- ✅ Both environments use port 8080 consistently
- ✅ Clean separation of concerns
- ✅ Easy to add new environments (staging, testing, etc.)

🔐 GitHub Secrets Configuration

Set these secrets in your GitHub repository settings:

Server Access

- `SERVER_HOST` - Your production server IP/domain
- `SERVER_USERNAME` - SSH username
- `SERVER_PASSWORD` - SSH password

Database Configuration

- `DB_HOST` - Database host (use `database` for Docker MySQL)
- `DB_USER` - Database username
- `DB_PASSWORD` - Database password
- `DB_NAME` - Database name
- `MYSQL_ROOT_PASSWORD` - MySQL root password

Application Secrets

- `SECRET_KEY` - JWT secret key (minimum 32 characters)
- `SECRET_KEY_REFRESH_TOKEN` - Refresh token secret
- `SENDGRID_API_KEY` - SendGrid API key for emails
- `WEBHOOK_SECRET` - GitHub webhook secret

Frontend Configuration

- `VUE_APP_API_URL` - API URL for frontend (e.g., `https://api.yourdomain.com`)

🖥 Production Server Configuration

🚀 Automated Setup (Recommended)

We provide an automated setup script that handles the entire server configuration:

```
# Download and run the setup script
wget https://raw.githubusercontent.com/yourusername/rbarros-
deployment/main/scripts/setup-production-server.sh
chmod +x setup-production-server.sh
./setup-production-server.sh
```

The script will interactively prompt for:

- Repository URL
- Domain name (optional)
- SSL setup with Let's Encrypt
- Database configuration
- And automatically configure everything!

📖 **See scripts/README.md for detailed script documentation.**

📋 Manual Setup (Alternative)

If you prefer manual setup or need to customize the process:

Prerequisites

Your production server needs:

- **Ubuntu 20.04+ / CentOS 8+ / Debian 11+**
- **Minimum 2GB RAM, 2 CPU cores**
- **20GB+ disk space**
- **Root or sudo access**

1. Install Docker & Docker Compose

```
# Update system
sudo apt update && sudo apt upgrade -y

# Install Docker
curl -fsSL https://get.docker.com -o get-docker.sh
sudo sh get-docker.sh

# Add user to docker group
sudo usermod -aG docker $USER

# Install Docker Compose
sudo curl -L
"https://github.com/docker/compose/releases/latest/download/docker-
compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose

# Verify installation
docker --version
docker-compose --version
```

2. Setup Deployment Directory

```
# Create deployment directory
sudo mkdir -p /opt/rbarros-deployment
sudo chown $USER:$USER /opt/rbarros-deployment
cd /opt/rbarros-deployment

# Clone the repository
git clone --recurse-submodules https://github.com/yourusername/rbarros-
deployment.git .

# Make sure submodules are updated
git submodule update --init --recursive
```

3. Configure Firewall

```
# Install UFW (if not installed)
sudo apt install ufw -y

# Allow SSH (important - don't lock yourself out!)
sudo ufw allow ssh

# Allow HTTP and HTTPS
sudo ufw allow 80/tcp
sudo ufw allow 443/tcp

# Allow specific ports for development (optional)
sudo ufw allow 3000/tcp  # Backend API
sudo ufw allow 8080/tcp  # Frontend

# Enable firewall
sudo ufw --force enable

# Check status
sudo ufw status
```

4. Setup SSL Certificates (Optional but Recommended)

Option A: Let's Encrypt (Free)

```
# Install Certbot
sudo apt install certbot -y

# Generate certificates (replace with your domain)
sudo certbot certonly --standalone -d yourdomain.com -d www.yourdomain.com

# Certificates will be in /etc/letsencrypt/live/yourdomain.com/
```

Option B: Manual SSL Setup

```
# Create SSL directory
mkdir -p /opt/rbarros-deployment/nginx/ssl

# Copy your SSL certificates
sudo cp /path/to/your/certificate.crt /opt/rbarros-deployment/nginx/ssl/
sudo cp /path/to/your/private.key /opt/rbarros-deployment/nginx/ssl/
```

5. Configure Nginx for SSL (if using SSL)

Update nginx/nginx.conf to include SSL:

```
# Add this server block for HTTPS
server {
    listen 443 ssl http2;
    server_name yourdomain.com www.yourdomain.com;

    ssl_certificate /etc/nginx/ssl/certificate.crt;
    ssl_private_key /etc/nginx/ssl/private.key;

    # SSL configuration
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers ECDHE-RSA-AES256-GCM-SHA512:DHE-RSA-AES256-GCM-SHA512;
    ssl_prefer_server_ciphers off;

    # Your existing location blocks here...
}

# Redirect HTTP to HTTPS
server {
    listen 80;
    server_name yourdomain.com www.yourdomain.com;
    return 301 https://$server_name$request_uri;
}
```

6. Setup Database (Choose One)

Option A: Use Docker MySQL (Recommended for simplicity)

```
# Database will be created automatically by Docker Compose
# Data persists in Docker volume
```

Option B: External MySQL Database

```
# Install MySQL
sudo apt install mysql-server -y

# Secure installation
sudo mysql_secure_installation

# Create database and user
sudo mysql -u root -p
```

```
CREATE DATABASE rbarros_db CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
CREATE USER 'rbarros_user'@'%' IDENTIFIED BY 'your-secure-password';
GRANT ALL PRIVILEGES ON rbarros_db.* TO 'rbarros_user'@'%';
FLUSH PRIVILEGES;
EXIT;
```

7. Configure GitHub Actions Deployment Path

Update your GitHub Actions workflow (`.github/workflows/workflow.yaml`) with the correct server path:

```
script: |
  cd /opt/rbarros-deployment  # Update this path
  git pull origin main
  # ... rest of the script
```

8. Test Deployment

```
# Manual test deployment
cd /opt/rbarros-deployment

# Set test environment variables
export DB_HOST=database
export DB_USER=rbarros_user
export DB_PASSWORD=your-password
export DB_NAME=rbarros_db
export SECRET_KEY=your-secret-key-here
export SECRET_KEY_REFRESH_TOKEN=your-refresh-secret
export SENDGRID_API_KEY=your-sendgrid-key
export WEBHOOK_SECRET=your-webhook-secret
export MYSQL_ROOT_PASSWORD=your-root-password
export VUE_APP_API_URL=https://yourdomain.com

# Deploy
make deploy-secrets

# Check status
make status
make health
```

9. Setup Automatic Backups (Recommended)

```
# Create backup script
sudo tee /opt/backup-rbarros.sh > /dev/null <<EOF
#!/bin/bash
```

```
cd /opt/rbarros-deployment
make backup
# Move backup to safe location
mv backup-*.sql.gz /opt/backups/
# Keep only last 7 days
find /opt/backups -name "backup-*.sql.gz" -mtime +7 -delete
EOF

# Make executable
sudo chmod +x /opt/backup-rbarros.sh

# Create backup directory
sudo mkdir -p /opt/backups

# Add to crontab (daily backup at 2 AM)
echo "0 2 * * * /opt/backup-rbarros.sh" | sudo crontab -
```

10. Monitoring Setup (Optional)

```
# Install monitoring tools
sudo apt install htop iotop nethogs -y

# Check Docker logs
docker-compose logs -f

# Monitor resources
htop

# Check disk usage
df -h
```

🔧 Server Maintenance

Regular Updates

```
# Update system packages
sudo apt update && sudo apt upgrade -y

# Update Docker images
cd /opt/rbarros-deployment
docker-compose pull
make deploy-secrets

# Clean up old images
docker system prune -f
```

Log Management

```
# View application logs
make logs

# Rotate Docker logs (add to crontab)
echo "0 3 * * * docker system prune -f --filter 'until=24h'" | sudo crontab -
```

Security Updates

```
# Enable automatic security updates
sudo apt install unattended-upgrades -y
sudo dpkg-reconfigure -plow unattended-upgrades
```

🗀 Project Structure

```
rbarros-deployment/
├── rbarros-frontend/          # Vue.js frontend (git submodule)
├── rbarros-backend/           # Node.js backend (git submodule)
│   └── database/init/         # Database initialization scripts
├── nginx/                     # Nginx configuration
├── scripts/                   # Deployment and setup scripts
├── .github/workflows/         # GitHub Actions
├── docker-compose.base.yml    # Base Docker configuration
├── docker-compose.prod.yml    # Production overrides
├── docker-compose.dev.yml     # Development overrides
├── Makefile                   # Easy commands
├── .env                       # Environment variables (local)
├── env.example                # Environment template
└── README.md                  # This file
```

🛠 Available Commands

```
# Setup and basic operations
make setup          # Copy environment template
make build          # Build Docker images
make up             # Start production services (same as 'prod')
make prod           # Start production services
make down           # Stop all services
make restart        # Restart services

# Development
make dev            # Start with development overrides (hot reload)
make logs           # View all logs
make logs-backend   # View backend logs only
make logs-frontend  # View frontend logs only

# Database operations
make backup         # Backup database
make restore BACKUP_FILE=backup.sql  # Restore database

# Maintenance
make clean          # Remove containers and volumes
make health         # Check service health
make status         # Show service status

# Access containers
make shell-backend  # Access backend container
make shell-frontend # Access frontend container
make shell-database # Access database container
```

🔄 CI/CD Workflow

1. **Push to main branch** → Triggers GitHub Actions
2. **SSH to production server** → Pulls latest code
3. **Export GitHub secrets** → As environment variables
4. **Deploy with Docker** → `make deploy-secrets`

🌐 Environment Variables

The Docker setup uses environment variables for configuration:

Local Development (.env file)

```
# Database Configuration
DB_HOST=database
DB_USER=rbarros_user
DB_PASSWORD=your-local-password
DB_NAME=rbarros_db
MYSQL_ROOT_PASSWORD=your-root-password

# Application Secrets
SECRET_KEY=your-jwt-secret-key-minimum-32-characters
SECRET_KEY_REFRESH_TOKEN=your-refresh-token-secret
SENDGRID_API_KEY=your-sendgrid-api-key
WEBHOOK_SECRET=your-webhook-secret

# Frontend Configuration
VUE_APP_API_URL=http://localhost:3000
```

Production (GitHub Secrets)

Environment variables are automatically set from GitHub secrets during deployment. The production configuration requires all environment variables to be properly set - there are no fallback values for security.

🔧 Development Setup

Prerequisites

- Docker & Docker Compose
- Git with submodule support
- Make (optional, for convenience commands)

First Time Setup

```
# Clone with submodules
git clone --recurse-submodules <your-repo-url>
cd rbarros-deployment

# Setup environment
make setup

# Edit .env file with your local values
nano .env

# Start development environment
make dev
```

Development vs Production

Mode	Command	Configuration	Frontend Port	Features
Development	`make dev`	`docker-compose.base.yml` + `docker-compose.dev.yml`	8080	Hot reload, volume mounts, dev tools
Production	`make prod`	`docker-compose.base.yml` + `docker-compose.prod.yml`	8080	Optimized builds, no volume mounts, security hardened

Updating Submodules

```
# Update to latest commits
git submodule update --remote

# Or pull specific submodule
cd rbarros-backend
git pull origin main
```

```
cd ../rbarros-frontend
git pull origin main
```

🐛 Troubleshooting

Common Issues

1. **Port conflicts**:

   ```
   # Check what's using ports
   netstat -tulpn | grep :80
   netstat -tulpn | grep :3000
   ```

2. **Database connection issues**:

   ```
   make logs-database
   make health
   ```

3. **Submodule issues**:

   ```
   git submodule update --init --recursive
   ```

4. **Permission issues**:

   ```
   # Reset everything
   make clean
   make up
   ```

Reset Everything

```
# Nuclear option - removes everything
make clean-all
make up
```

📊 Monitoring

Health Checks

```
# Check all services
make health

# Manual health check
curl http://localhost:3000/health
```

Logs

```
# All services
make logs

# Specific service
make logs-backend
make logs-frontend
make logs-database
```

🔒 Security

- Non-root users in containers
- Environment variable isolation
- Nginx security headers
- Rate limiting on API endpoints
- Secrets management via GitHub

📝 Contributing

1. Make changes to submodules in their respective repositories
2. Update submodule references in this repository
3. Test locally with `make dev`
4. Push to main for automatic deployment

📞 Support

- Check logs: `make logs`
- Health status: `make health`
- Documentation: See `DOCKER_README.md` for detailed Docker info