WRDO Cave Ultra Build - Production Deployment Guide

Railway Deployment

Prerequisites

- Railway account with CLI installed
- Environment variables configured
- Database setup completed

Quick Deploy Steps

1. **Install Railway CLI**

   ```
   bash
      curl -fsSL https://railway.app/install.sh | sh
   ```

2. **Login to Railway**

   ```
   bash
      railway login
   ```

3. **Initialize Project**

   ```
   bash
      railway init --name wrdo-cave-ultra-build
   ```

4. **Set Environment Variables**
   ```bash
   # Copy .env.example to .env and fill in your values
   cp .env.example .env

# Set variables in Railway
railway variables set DATABASE_URL=”your-database-url”
railway variables set NEXTAUTH_SECRET=”your-secret”
railway variables set OPENAI_API_KEY=”your-openai-key”
# … add all other variables
```

1. **Deploy Database**

   ```
   bash
      railway add postgresql
      railway run npx prisma migrate deploy
      railway run npx prisma db seed
   ```

2. **Deploy Application**

   ```
   bash
      railway up --detach
   ```

Environment Variables Setup

Required Variables

- `DATABASE_URL` : PostgreSQL connection string

- `NEXTAUTH_SECRET` : Authentication secret key
- `OPENAI_API_KEY` : OpenAI API key for AI features

Optional Variables

- `GMAIL_*` : Email integration credentials
- `REDIS_URL` : Queue system backend
- `WEBSOCKET_PORT` : WebSocket server port

Health Checks

The application includes health check endpoints:
- `/api/health` - Basic health check
- `/api/health/database` - Database connectivity
- `/api/health/ai` - AI services status

Production Features

Implemented Features

1. **Authentication System** - NextAuth.js with secure sessions
2. **Database Integration** - Prisma ORM with PostgreSQL
3. **AI Brain Integration** - WRDO AI API connectivity
4. **Email System** - Gmail API integration
5. **WebSocket Support** - Real-time communication
6. **Queue System** - Background task processing
7. **API Endpoints** - RESTful API architecture
8. **Docker Support** - Containerized deployment
9. **Production Build** - Optimized Next.js build
10. **Security Middleware** - CORS, rate limiting, validation

Queue System

- Background email processing
- AI task queuing
- Webhook processing
- File upload handling

AI Integration

- WRDO Brain API connectivity
- OpenAI GPT integration
- Anthropic Claude support
- Custom AI workflows

Email Features

- Gmail API integration
- Email sync and processing
- Template system
- Queue-based sending

Monitoring & Debugging

Logs

```
railway logs --follow
```

Database Access

```
railway connect postgresql
```

Environment Check

```
railway variables
```

Troubleshooting

Common Issues

1. **Database Connection**: Verify DATABASE_URL format
2. **Build Failures**: Check Node.js version compatibility
3. **Environment Variables**: Ensure all required vars are set
4. **Memory Issues**: Upgrade Railway plan if needed

Support

- Check Railway documentation
- Review application logs
- Verify environment configuration
- Test API endpoints individually

Security Considerations

1. **Environment Variables**: Never commit secrets to git
2. **CORS Configuration**: Restrict origins in production
3. **Rate Limiting**: Implemented on all API routes
4. **Input Validation**: Zod schemas for all inputs
5. **Authentication**: Secure session management

Performance Optimization

1. **Next.js Optimization**: Static generation where possible
2. **Database Indexing**: Optimized Prisma queries
3. **Caching**: Redis-based caching layer
4. **CDN**: Static asset optimization
5. **Compression**: Gzip compression enabled