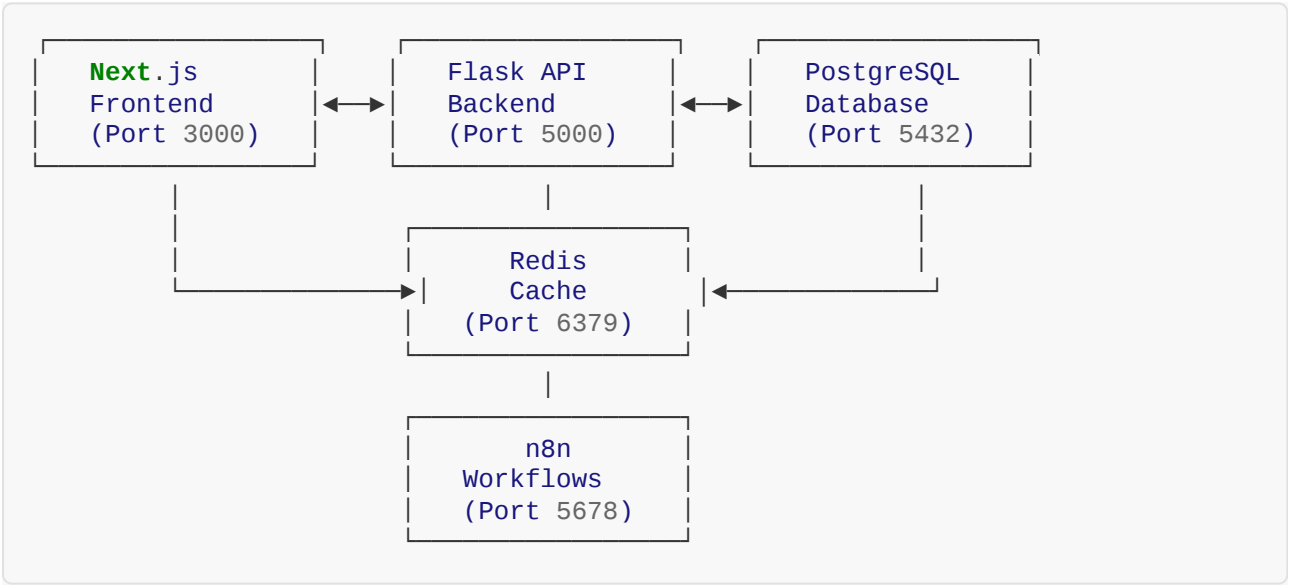
Agent CEO System - Cloud Deployment Guide

Overview

This comprehensive guide covers deploying the Agent CEO system to various free and paid cloud platforms. The system is designed to be cloud-native and can be deployed on multiple platforms with minimal configuration changes.

Architecture Overview

The Agent CEO system consists of the following components:



Free Cloud Platform Options

1. Railway (Recommended for Full Stack)

Pros: - Generous free tier with $5/month credit - Automatic deployments from Git - Built-in PostgreSQL and Redis - Easy environment variable management - Excellent for

full-stack applications

Cons: - Limited to $5/month on free tier - May require credit card for verification

Deployment Steps:

1. **Install Railway CLI:** `bash npm install -g @railway/cli`
2. **Login and Initialize:** `bash railway login railway init`
3. **Deploy using our configuration:** `bash cp deploy/railway.json ./railway.json railway up`
4. **Set Environment Variables:** `bash railway variables set OPENAI_API_KEY=your-key-here railway variables set ANTHROPIC_API_KEY=your-key-here railway variables set POSTGRES_PASSWORD=secure-password railway variables set JWT_SECRET_KEY=your-jwt-secret railway variables set NEXTAUTH_SECRET=your-nextauth-secret`

2. Render (Great for Backend Services)

Pros: - Generous free tier for web services - Automatic SSL certificates - Built-in PostgreSQL free tier - Easy Git-based deployments - Good performance on free tier

Cons: - Services sleep after 15 minutes of inactivity - Limited build minutes on free tier - No Redis on free tier

Deployment Steps:

1. **Push code to GitHub/GitLab**
2. **Connect Repository to Render:**
3. Go to render.com
4. Connect your Git repository
5. Render will detect the `render.yaml` file
6. **Configure Environment Variables:**
7. Set required variables in Render dashboard

8. Use the provided `render.yaml` configuration

3. Vercel (Frontend Only)

Pros: - Excellent for Next.js applications - Global CDN - Automatic deployments - Great performance - Generous free tier

Cons: - Frontend only (need separate backend hosting) - Function execution time limits

Deployment Steps:

1. **Install Vercel CLI:** `bash npm install -g vercel`
2. **Deploy Frontend:** `bash cd frontend vercel --prod`
3. **Configure Environment Variables:**
4. Set variables in Vercel dashboard
5. Point to your backend API URL

4. Heroku (Alternative Option)

Pros: - Well-established platform - Good documentation - Add-ons ecosystem

Cons: - No longer offers free tier - More expensive than alternatives

Deployment Configurations

Environment Variables Setup

Create a `.env` file based on `.env.example`:

```
# Required API Keys
OPENAI_API_KEY=your-openai-api-key
ANTHROPIC_API_KEY=your-anthropic-api-key

# Database
POSTGRES_PASSWORD=secure-password-123

# Security
JWT_SECRET_KEY=your-jwt-secret-key
NEXTAUTH_SECRET=your-nextauth-secret

# Optional: Social Media APIs
TWITTER_API_KEY=your-twitter-api-key
FACEBOOK_ACCESS_TOKEN=your-facebook-token
LINKEDIN_ACCESS_TOKEN=your-linkedin-token
```

Docker Deployment

For platforms supporting Docker:

1. **Build Images:** `bash docker build -t agent-ceo-backend ./backend/agent-ceo-api/ docker build -t agent-ceo-frontend ./frontend/`
2. **Run with Docker Compose:** `bash docker-compose up -d`
3. **Health Check:** `bash curl http://localhost:5000/api/health curl http://localhost:3000/api/health`

Platform-Specific Guides

Railway Deployment (Detailed)

1. **Project Setup:** ```bash # Install Railway CLI npm install -g @railway/cli

Login to Railway railway login

Initialize project railway init agent-ceo-system ```

1. **Service Configuration:** ```bash # Deploy backend railway add --service backend railway deploy --service backend

Deploy frontend railway add --service frontend railway deploy --service frontend

Add PostgreSQL railway add --service postgres

Add Redis railway add --service redis ```

1. **Environment Variables:** ```bash # Set required variables railway variables set OPENAI_API_KEY=sk-... railway variables set ANTHROPIC_API_KEY=sk-ant-... railway variables set POSTGRES_PASSWORD=secure123 railway variables set JWT_SECRET_KEY=your-secret railway variables set NEXTAUTH_SECRET=your-nextauth-secret

Set service URLs railway variables set NEXT_PUBLIC_AGENT_CEO_API_URL=https://your-backend.railway.app railway variables set DATABASE_URL=postgresql://user:pass@host:port/db railway variables set REDIS_URL=redis://host:port ```

1. **Custom Domains (Optional):** `bash railway domain add your-domain.com --service frontend railway domain add api.your-domain.com --service backend`

Render Deployment (Detailed)

1. **Repository Setup:**
2. Push code to GitHub/GitLab
3. Ensure `render.yaml` is in root directory
4. **Service Creation:**
5. Go to Render dashboard
6. Click "New +" → "Blueprint"
7. Connect your repository
8. Render will read `render.yaml` automatically
9. **Database Setup:**
10. PostgreSQL database is created automatically
11. Connection string is provided as environment variable
12. **Environment Variables:** Set in Render dashboard: `OPENAI_API_KEY=your-key ANTHROPIC_API_KEY=your-key JWT_SECRET_KEY=your-secret`

`NEXTAUTH_SECRET=your-secret`

Vercel + Backend Hosting

1. **Frontend on Vercel:** `bash cd frontend vercel --prod`
2. **Backend on Railway/Render:** Deploy backend separately using Railway or Render
3. **Connect Services:** `bash # Set backend URL in Vercel vercel env add NEXT_PUBLIC_AGENT_CEO_API_URL # Enter: https://your-backend-url.com`

Database Setup

PostgreSQL Initialization

The system automatically creates the database schema on first run. The initialization script includes:

- User management tables
- Agent and task tracking
- Strategic insights storage
- Email campaign management
- Data analysis results
- Social media post tracking
- Audit logging

Database Migrations

For schema updates:

```
# Backup current database
./scripts/deploy.sh backup

# Apply migrations (if any)
# Migrations are handled automatically by the Flask app

# Verify schema
psql $DATABASE_URL -c "\dt"
```

Monitoring and Maintenance

Health Checks

The system includes built-in health checks:

```
# Backend health
curl https://your-backend-url.com/api/health

# Frontend health
curl https://your-frontend-url.com/api/health

# Database health
curl https://your-backend-url.com/api/health/database
```

Logging

Logs are available through platform dashboards:

- **Railway:** `railway logs --service backend`
- **Render:** Available in service dashboard
- **Vercel:** Available in function logs

Performance Monitoring

Monitor key metrics: - Response times - Error rates - Database performance - Memory usage - API rate limits

Security Considerations

Environment Variables

Never commit sensitive data:

```
# Add to .gitignore
.env
.env.local
.env.production
```

API Security

- Use strong JWT secrets
- Implement rate limiting
- Validate all inputs
- Use HTTPS in production

Database Security

- Use strong passwords
- Enable SSL connections
- Regular backups
- Monitor access logs

Scaling Considerations

Horizontal Scaling

For high traffic: - Use multiple backend instances - Implement load balancing - Use Redis for session storage - Consider CDN for static assets

Database Scaling

- Use connection pooling

- Implement read replicas
- Consider database sharding
- Monitor query performance

Troubleshooting

Common Issues

1. **Build Failures:**
2. Check Node.js/Python versions
3. Verify dependencies in package.json/requirements.txt
4. Check build logs for specific errors
5. **Database Connection Issues:**
6. Verify DATABASE_URL format
7. Check network connectivity
8. Ensure database is running
9. **API Key Issues:**
10. Verify API keys are set correctly
11. Check API key permissions
12. Monitor API usage limits
13. **CORS Issues:**
14. Update CORS_ORIGINS environment variable
15. Check frontend-backend URL configuration

Debug Commands

```
# Check service status
./scripts/deploy.sh health

# View logs
./scripts/deploy.sh logs

# Test API endpoints
curl -X GET https://your-api-url.com/api/agents
curl -X POST https://your-api-url.com/api/strategic/quick-insights \
  -H "Content-Type: application/json" \
  -d '{"query": "test", "context": "test", "urgency": "normal"}'
```

Cost Optimization

Free Tier Limits

Railway: - $5/month credit - 500 hours execution time - 1GB RAM per service

Render: - 750 hours/month free - 512MB RAM - Services sleep after 15 minutes

Vercel: - 100GB bandwidth - 1000 serverless function invocations - 6000 build minutes

Cost-Effective Strategies

1. **Use Free Tiers Efficiently:**
2. Deploy frontend on Vercel
3. Backend on Railway/Render
4. Use free PostgreSQL tiers
5. **Optimize Resource Usage:**
6. Implement caching
7. Optimize database queries
8. Use efficient Docker images
9. **Monitor Usage:**
10. Set up billing alerts

11. Monitor resource consumption
12. Scale down during low usage

Backup and Recovery

Automated Backups

```
# Create backup script
#!/bin/bash
DATE=$(date +%Y%m%d_%H%M%S)
pg_dump $`DATABASE_URL > backup_`$DATE.sql
```

Recovery Procedures

```
# Restore from backup
./scripts/deploy.sh restore backup_20240101_120000.sql
```

Support and Maintenance

Regular Maintenance Tasks

1. **Weekly:**
2. Check service health
3. Review error logs
4. Monitor API usage
5. **Monthly:**
6. Update dependencies
7. Review security settings
8. Backup database
9. **Quarterly:**
10. Performance optimization

11. Security audit
12. Cost review

Getting Help

1. **Platform Documentation:**
2. Railway: docs.railway.app
3. Render: render.com/docs
4. Vercel: vercel.com/docs
5. **Community Support:**
6. Platform Discord servers
7. Stack Overflow
8. GitHub Issues
9. **Professional Support:**
10. Platform support plans
11. Consulting services
12. Custom development

Conclusion

The Agent CEO system is designed to be cloud-native and can be deployed on various platforms with minimal configuration. Choose the platform that best fits your needs:

- **Railway:** Best for full-stack deployment with minimal setup
- **Render:** Great for backend services with automatic scaling
- **Vercel:** Excellent for frontend with global CDN
- **Combination:** Frontend on Vercel + Backend on Railway/Render

Follow the platform-specific guides and use the provided deployment scripts for a smooth deployment experience.