

Docker Setup Complete

The AI RFP Risk Scanner now has comprehensive Docker support to resolve the GitHub Dependabot error and enable containerized deployment.

Files Created

Core Docker Files

- `Dockerfile` - Production multi-stage build configuration
- `Dockerfile.dev` - Development environment configuration
- `.dockerignore` - Excludes unnecessary files from Docker context
- `docker-compose.yml` - Production orchestration
- `docker-compose.dev.yml` - Development orchestration

GitHub Integration

- `.github/workflows/docker-build.yml` - CI/CD pipeline for Docker builds
- `.github/dependabot.yml` - Dependabot configuration for Docker, npm, and GitHub Actions updates

Management Scripts

- `scripts/docker-start.sh` - Production startup script
- `scripts/docker-dev.sh` - Development startup script
- `scripts/docker-stop.sh` - Stop all containers script
- `scripts/docker-clean.sh` - Cleanup script
- `Makefile` - Convenient commands for all operations

Health & Monitoring

- `healthcheck.js` - Container health check script
- `app/api/health/route.ts` - Health check API endpoint

Documentation & Configuration

- `README.Docker.md` - Comprehensive Docker documentation
- `.env.docker` - Docker environment template
- `.dockerenv` - Container environment indicator

Key Features



Production Ready

- Multi-stage builds for optimal image size
- Standalone Next.js output for better Docker compatibility
- Memory optimizations (4GB heap, garbage collection)
- Health checks and monitoring



Development Friendly

- Hot reload support in development mode
- Volume mounting for live code changes

- Separate development and production configurations
- Easy database reset and management

Security & Best Practices

- Non-root user in containers
- Minimal attack surface with Alpine Linux
- Proper file permissions and ownership
- Security scanning in CI/CD pipeline

Complete Stack

- PostgreSQL database included
- Persistent data volumes
- Network isolation
- Service orchestration with Docker Compose

Quick Commands

```
# Start production environment
make docker-start
./scripts/docker-start.sh

# Start development environment
make docker-dev
./scripts/docker-dev.sh

# Stop all containers
make docker-stop





# Clean up everything
make docker-clean

# View logs
make docker-logs





# Access container shell
make docker-shell
```

Integration Benefits





GitHub Dependabot

-  Docker dependency scanning enabled
-  Automatic security updates for base images
-  Node.js dependency updates in `/app` directory
-  GitHub Actions workflow updates

CI/CD Pipeline

-  Automated Docker builds on push/PR
-  Container Registry publishing (GHCR)
-  Security vulnerability scanning with Trivy
-  Multi-architecture support ready

Production Deployment

-  Optimized multi-stage builds
-  Health checks for monitoring
-  Persistent data volumes
-  Environment-specific configurations

Next Steps

1. **Push to GitHub** - The Dependabot error will be resolved once these files are committed
2. **Configure Secrets** - Update `.env` files with production values
3. **Test Deployment** - Run `make docker-start` to test locally
4. **Production Setup** - Deploy using the Docker Compose configuration

Configuration Notes

Environment Variables

The Docker setup uses environment-specific configurations:

- Development: `.env.development` or `.env.local`
- Production: `.env.production` or container environment variables
- Docker template: `.env.docker` (copy and customize)

Database

- PostgreSQL 15 Alpine for lightweight, secure database
- Persistent volume for data storage
- Automatic initialization and migration support

Memory Management

- Containers configured with 4GB heap size
- Garbage collection enabled for better performance
- Optimized for the application's analysis workload

Troubleshooting

If you encounter issues:

1. **Check logs:** `make docker-logs`
2. **Verify health:** Access `http://localhost:3000/api/health`
3. **Container status:** `docker-compose ps`
4. **Clean restart:** `make docker-clean && make docker-start`

The Docker setup is now complete and ready for use! 🎉