Sentry Environment Configuration Analysis

Date: January 2, 2026
Project: CareLinkAI
Analysis: Sentry Environment Settings

Executive Summary

✅ **Sentry is configured for PRODUCTION environment**

The Sentry integration automatically determines its environment based on `NODE_ENV`, which Next.js sets to `production` during builds and deployments.

Environment Determination Logic

All three Sentry configuration files use the same logic:

```
const ENVIRONMENT = process.env.NODE_ENV || 'production';
```

How NODE_ENV is Set:

1. **Local Development** (`npm run dev`):
   - Next.js automatically sets `NODE_ENV=development`
   - Sentry runs in **development mode**

2. **Production Build** (`npm run build`):
   - Next.js automatically sets `NODE_ENV=production`
   - Sentry runs in **production mode**

3. **Deployed on Render**:
   - Render runs `npm run build` which sets `NODE_ENV=production`
   - Sentry runs in **production mode**

Current Configuration:

`.env` file: Does NOT explicitly set NODE_ENV
Build script: `NODE_OPTIONS='--max-old-space-size=4096' NEXT_TELEMETRY_DISABLED=1 next build`
Default behavior: Falls back to `production` when NODE_ENV is not set

Sample Rates by Environment

Client Config (`sentry.client.config.ts`)

Setting	Production	Development
`tracesSampleRate`	**0.1 (10%)**	1.0 (100%)
`replaysSessionSampleRate`	**0.1 (10%)**	0.1 (10%)
`replaysOnErrorSampleRate`	**1.0 (100%)**	1.0 (100%)
`debug`	**false**	true

Server Config (`sentry.server.config.ts`)

Setting	Production	Development
`tracesSampleRate`	**0.1 (10%)**	1.0 (100%)
`profilesSampleRate`	**0.1 (10%)**	1.0 (100%)
`debug`	**false**	true
`enableTracing`	**true**	true

Edge Config (`sentry.edge.config.ts`)

Setting	Production	Development
`tracesSampleRate`	**0.1 (10%)**	1.0 (100%)
`debug`	**false**	true

Bold values indicate what’s currently active in production.

Configuration Files Analysis

1. Client Configuration (`sentry.client.config.ts`)

DSN Source: `process.env.NEXT_PUBLIC_SENTRY_DSN`
Environment: `process.env.NODE_ENV || 'production'`

Key Features:
- ✅ Browser environment check before initialization
- ✅ Replay integration for session recording
- ✅ Error filtering for connection timeouts
- ✅ Console logging for initialization status

Error Filtering:

```
beforeSend(event, hint) {
  // Filters out ETIMEDOUT and ENETUNREACH errors
  // These don't get sent to Sentry dashboard
}
```

2. Server Configuration (`sentry.server.config.ts`)

DSN Source: `process.env.SENTRY_DSN || process.env.NEXT_PUBLIC_SENTRY_DSN`
Environment: `process.env.NODE_ENV || 'production'`

Key Features:
- ✅ Performance monitoring enabled
- ✅ Profiling support
- ✅ Prisma error filtering in development
- ✅ Connection error filtering

3. Edge Configuration (`sentry.edge.config.ts`)

DSN Source: `process.env.SENTRY_DSN || process.env.NEXT_PUBLIC_SENTRY_DSN`
Environment: `process.env.NODE_ENV || 'production'`

Key Features:
- ✅ Lightweight configuration for edge runtime
- ✅ Duplicate initialization check
- ✅ Basic tracing support

Environment Variables

Currently Set in `.env`:

```
NEXT_PUBLIC_SENTRY_DSN=https://d649b9c85c145427fcf-
b62cecdeaa2d9e@o4510110703216128.ingest.us.sentry.io/4510154442089472
```

NOT Set (uses defaults):

- ❌ `NODE_ENV` - Defaults to `production` in config, but Next.js sets it automatically
- ❌ `SENTRY_ENVIRONMENT` - Not used; relies on NODE_ENV instead

On Render (Production):

Render should have these environment variables:
- ✅ `NEXT_PUBLIC_SENTRY_DSN` - Set for client-side tracking
- ✅ `SENTRY_DSN` - Recommended for server-side tracking (can use same value)
- ⚠️ `NODE_ENV` - Automatically set to `production` by Next.js during build

Current Production Status

What's Running:

🟢 **Environment**: `production`
🟢 **Trace Sampling**: 10% of requests
🟢 **Replay Sampling**: 10% of sessions
🟢 **Error Replay**: 100% of errors
🟢 **Debug Logging**: Disabled
🟢 **Performance Profiling**: 10% of requests

Why Events May Not Be Appearing:

Based on the configuration, Sentry IS properly configured for production. However, the **10% sampling rate** means:

1. Only 10% of traces/transactions are sent to Sentry
2. Only 10% of sessions are recorded
3. 100% of **errors** are still captured

If you're not seeing events, it could be:
- ✅ Network connectivity issues (confirmed from logs)
- ✅ Low sampling rate means most normal traffic isn't sent
- ⚠️ Errors ARE being captured, but network timeouts prevent sending

---

Recommendations

Option 1: Increase Sampling for Testing (Temporary)

Modify the config files to increase sampling in production temporarily:

```
// In sentry.client.config.ts, sentry.server.config.ts, sentry.edge.config.ts
tracesSampleRate: ENVIRONMENT === 'production' ? 0.5 : 1.0,  // 50% instead of 10%
```

Option 2: Add Custom Environment Variable

Create a `SENTRY_ENVIRONMENT` variable for more control:

```
// In all config files
const ENVIRONMENT = process.env.SENTRY_ENVIRONMENT || process.env.NODE_ENV || 'produc-
tion';
```

Then set on Render:

```
SENTRY_ENVIRONMENT=production
```

Option 3: Enable Debug Mode Temporarily

Add to Render environment variables:

```
SENTRY_DEBUG=true
```

Then update configs:

```
debug: process.env.SENTRY_DEBUG === 'true' || ENVIRONMENT === 'development',
```

Option 4: Verify DSN on Render

Ensure Render has BOTH variables set:

```
NEXT_PUBLIC_SENTRY_DSN=https://d649b9c85c145427fcf-
b62cecdeaa2d9e@o4510110703216128.ingest.us.sentry.io/4510154442089472
SENTRY_DSN=https://d649b9c85c145427fcf-
b62cecdeaa2d9e@o4510110703216128.ingest.us.sentry.io/4510154442089472
```

Verification Steps

1. Check Render Environment Variables:

```
# In Render dashboard, verify these are set:
NEXT_PUBLIC_SENTRY_DSN=<your-dsn>
SENTRY_DSN=<your-dsn>
```

2. Check Build Logs:

Look for these messages in Render logs:

```
[Sentry] Client-side initialization successful
[Sentry] Server-side initialization successful
[Sentry] Edge initialization successful
```

3. Test Error Tracking:

Trigger a test error and check if it appears in Sentry dashboard:

```
// In any page/API route
throw new Error('Sentry test error from production');
```

4. Verify Sampling:

With 10% sampling, you'll need significant traffic to see traces. Consider:
- Temporarily increasing to 50-100% for testing
- Or generate multiple test requests to trigger sampling

Troubleshooting

Issue: No events in Sentry dashboard

Possible Causes:
1. ❌ DSN not set on Render (check environment variables)
2. ⚠️ Network timeout preventing events from sending (confirmed issue)

3. ✅ Low sampling rate (only 10% of traffic is sent)
4. ⚠️ No errors actually occurring (normal traffic not sampled enough)

Solutions:
1. Verify DSN is set in Render dashboard
2. Check network connectivity from Render to Sentry
3. Temporarily increase sampling rate for testing
4. Trigger test errors to confirm error tracking works

Issue: Too many events

Solutions:
1. Decrease sampling rates
2. Add more aggressive error filtering in `beforeSend`
3. Enable `ignoreErrors` option for specific error types

Summary

✅ **Sentry IS configured for production environment**
✅ **Environment is determined automatically by NODE_ENV**
✅ **Sampling rates are set conservatively at 10%**
✅ **Error tracking is at 100% (only errors are always sent)**
⚠️ **Network issues may be preventing events from reaching Sentry**
⚠️ **Low sampling means you need significant traffic to see traces**

Bottom Line: Your Sentry setup is correct for production use with conservative sampling. The lack of events is likely due to:
1. Network connectivity issues (ETIMEDOUT errors filtered out)
2. Low traffic volume combined with 10% sampling
3. Errors being filtered before sending

To test if Sentry works, temporarily increase sampling or trigger test errors.