Fast Checklist

How to detect and fix incompatibilities early

By the time most migration bugs reach production, the damage is already done. The goal of this checklist is not perfection. It is an early failure in controlled conditions. Use this before, during, and after migration.

Dependency graph sanity checks

What to do	What this catches
• Generate a full dependency graph for the solution. • Identify packages referenced in more than one version. • Lock shared packages to a single version across all projects.	• Runtime MethodNotFoundException • Transitive dependency conflicts • EF Core design-time failures

If different projects reference different versions of the same package, assume production failure is scheduled, not possible.

Runtime smoke tests, not just builds

What to do	What this catches
• Run basic request flows against the published output. • Exercise background jobs, migrations, and rarely used endpoints. • Test after trimming and single-file publish if enabled.	• Runtime-only binding failures • Reflection and trimming issues • Lazy DI initialization crashes

A green build answers the wrong question. Published artifacts tell the truth.

OS-specific CI pipelines

What to do	What this catches
• Run CI on the same OS as production. • Add Linux pipelines if containers are involved. • Validate file paths, casing, and platform APIs.	• Windows-only API usage • Case-sensitivity bugs • Process execution failures

If production runs on Linux, Windows CI is incomplete by definition.

Configuration validation at startup

What to do	What this catches
• Validate required configuration values during application startup. • Fail fast on missing secrets or invalid settings. • Avoid lazy access to critical configuration.	• First-request crashes • Environment-specific misconfiguration • Hidden reliance on web.config behavior

Configuration errors should stop deployment, not surprise users.

Publish-time trimming validation

What to do	What this catches
• Test serialization paths after publish. • Audit reflection-heavy code. • Disable trimming selectively if required.	• Missing types at runtime • Broken serializers • Silent behavior changes after publication

Trimming optimizes what you explicitly describe. Legacy code rarely does.

EF-specific migration checks

What to do	What this catches
• Verify exact version alignment of all EF Core packages. • Load-test critical queries. • Review lazy loading usage explicitly.	• Migration tooling failures • Performance regressions • EF Core 8 behavior changes

EF issues often pass correctness tests and fail operationally.

Final takeaway

Most .NET migration failures are detectable early if you test the right things at the right time. The checklist is short because the problem space is repetitive. If you apply these checks consistently, migration stops being a gamble and becomes an engineering process. That is the difference between “it compiled” and “it survived production.”