VectraOps

Knowledge Library

CHECKLIST

NET Checklist: Operational Readiness Review for Production Use

Use this practical .NET checklist to verify build quality, security controls, deployment readiness, observability, and rollback safety before production use.

Programming - July 02, 2026

Purpose

The problem with a .NET release is rarely a single broken line of code; it is the gap between a working build and an operable production service. Teams often discover missing configuration, weak secrets handling, incomplete telemetry, or an untested rollback path only after deployment. This checklist is designed to close that gap.

Use it to decide whether a .NET application is ready for production use, what evidence to collect, and who owns each verification step. After working through it, you should be able to confirm whether the release is safe to promote, identify the highest-risk omissions, apply a practical validation workflow, and verify the minimum operational controls before go-live.

If you need a broader context for what the platform includes, review What is Programming in .NET? A Practical Operational View before using this checklist in a deployment review.

How to use this checklist

Work through the phases in order. Each phase includes verifiable checks, the evidence to capture, acceptance criteria, the usual owner, and a review cadence. Treat any failed acceptance criterion as a release blocker unless you explicitly document an approved exception with a compensating control.

Use the checklist during development sign-off, pre-production review, and post-deployment validation. For each item, record the artifact or observation that proves the check was completed, not just the fact that someone said it was done.

Phase 1: Scope and release definition

This phase confirms that everyone is reviewing the same application, the same version, and the same deployment target. Missing scope is a common reason readiness reviews fail late.

Purpose

Confirm the release boundary, operational impact, and ownership before any technical validation begins.

Checklist items

- ? Confirm the application name, service boundary, and release identifier that will be promoted.
- ? Review the target runtime version, SDK version, and deployment environment for compatibility requirements.
- ? Validate the deployment model, such as container, virtual machine, serverless, or hosted service, and document any platform-specific assumptions.
- ? Assign an operational owner, a security reviewer, and a deployment approver for the release.
- ? Document the expected production traffic pattern, dependencies, and any maintenance window or change freeze constraints.

Evidence to capture

- Release manifest or change record
- Version matrix for runtime, SDK, and packages
- Environment target summary
- Ownership and approval record

Acceptance criteria

The release boundary is unambiguous, the target platform is known, and all required reviewers are assigned. Any platform or version dependency that could alter behavior is documented and accepted.

Owner

Release manager or service owner

Review cadence

Per release

Phase 2: Build integrity and dependency control

This phase ensures the application is reproducible and that build inputs are controlled. In .NET systems, unchecked package drift or inconsistent build settings often create differences between test and production behavior.

Purpose

Confirm that the build output is deterministic enough for promotion and that dependencies are known, approved, and traceable.

Checklist items

- ? Confirm the build uses a pinned SDK or documented toolchain version.
- ? Review package references for unnecessary or unapproved dependencies.
- ? Validate that package source trust, restore settings, and lock-file usage match the team?s supply-chain policy.
- ? Test that the application builds cleanly from a fresh checkout or clean build agent.
- ? Document any compiler warnings, analyzers, or code-generation steps that affect the shipped artifact.
- ? Validate that environment-specific values are excluded from source control and injected at deployment time.

Evidence to capture

- Build log from a clean build
- Dependency inventory or package lock file
- Analyzer or warning summary
- Configuration source list

Acceptance criteria

The release builds from controlled inputs, dependencies are accounted for, and no undocumented build-time behavior is required to produce the artifact.

Owner

Build or platform engineer

Review cadence

Each release and after dependency changes

Common mistakes to avoid

A frequent mistake is treating a successful local build as evidence of release readiness. Another is allowing package versions to float without a documented policy, which makes the production artifact harder to reproduce.

Phase 3: Configuration and secrets validation

This phase checks whether the application can start and operate safely with production settings. Many .NET incidents are configuration failures rather than code defects.

Purpose

Confirm that all required configuration values exist, sensitive data is protected, and configuration errors fail safely.

Checklist items

- ? Validate that all required configuration keys are defined for the target environment.

- ? Review default values to ensure they do not enable unsafe behavior in production.
- ? Confirm that secrets are stored outside the repository and rotated according to policy.
- ? Test that missing or malformed configuration causes a controlled startup failure or a clear health-check failure.
- ? Validate that connection strings, API endpoints, and feature flags point to the intended production resources.
- ? Document any configuration overrides used for canary, blue-green, or staged rollout behavior.

Evidence to capture

- Sanitized configuration manifest
- Secret storage reference or policy evidence
- Startup or validation logs
- Feature flag or override inventory

Acceptance criteria

All required settings are present, sensitive values are protected, and the service fails in a predictable way when configuration is invalid. Production endpoints and flags are explicitly confirmed.

Owner

Application engineer with security review

Review cadence

Per release and after any configuration change

Phase 4: Security and authorization checks

This phase validates the access controls and application security boundaries that matter before a service is exposed to production traffic.

Purpose

Confirm that authentication, authorization, and security headers or transport settings are intentional and aligned with policy.

Checklist items

- ? Review authentication flows for the production identity provider, tenant, issuer, or certificate configuration.
- ? Validate authorization rules for privileged operations, administrative endpoints, and service-to-service access.
- ? Test that unauthenticated or under-privileged requests are rejected as expected.
- ? Confirm that transport security requirements such as TLS enforcement and certificate handling are documented and verified.
- ? Review logging to ensure secrets, tokens, and personal or sensitive data are not written to application logs.
- ? Validate that dependency and runtime security scanning results have been reviewed and triaged.

Evidence to capture

- Access control matrix
- Authentication and authorization test results

- Security scan summary and disposition
- Log sample with sensitive data redaction evidence

Acceptance criteria

Access to protected resources is denied unless explicitly allowed, sensitive data is not exposed in logs, and known security findings are either resolved or formally accepted with risk ownership.

Owner

Security engineer or application security reviewer

Review cadence

Per release and after identity or policy changes

Common mistakes to avoid

A common mistake is verifying only the happy path for authenticated users. Another is assuming a framework default is secure without checking the actual production configuration, especially when behavior changes by hosting model or runtime version.

Phase 5: Runtime behavior and failure handling

This phase checks whether the application behaves predictably under normal and abnormal conditions. It is not enough for a service to start; it must fail in ways the operations team can detect and handle.

Purpose

Confirm that startup, shutdown, retries, timeouts, and error handling produce observable and controllable behavior.

Checklist items

- ? Test application startup from the intended deployment artifact and runtime environment.
- ? Validate that readiness and liveness checks reflect actual service health.
- ? Review timeout, retry, and circuit-breaker settings for downstream calls.
- ? Test graceful shutdown behavior, including request draining and resource cleanup.
- ? Confirm that exception handling produces actionable errors without revealing sensitive internals.
- ? Validate that background jobs, scheduled tasks, or hosted services stop and restart safely.

Evidence to capture

- Startup and shutdown logs
- Health-check results
- Downstream failure test notes
- Exception samples or error catalog

Acceptance criteria

The service starts reliably, exposes accurate health signals, degrades predictably when dependencies fail, and stops without corrupting state or leaving abandoned work.

Owner

Application engineer or site reliability engineer

Review cadence

Per release and after dependency or hosting changes

Phase 6: Observability and supportability

Production readiness depends on being able to diagnose issues quickly. Without enough telemetry, teams react slowly and often guess at root cause.

Purpose

Confirm that logs, metrics, traces, and alerts are sufficient for initial triage and incident response.

Checklist items

- ? Validate that application logs include correlation identifiers and meaningful event context.
- ? Review metrics for availability, latency, request volume, and dependency health.
- ? Confirm that distributed tracing or request correlation is enabled where needed.
- ? Test that alerts fire for the right symptom thresholds and route to an active on-call or support channel.

- ? Document runbooks for common failure modes, including startup failure, dependency outage, and elevated error rate.
- ? Validate retention settings and access controls for operational telemetry.

Evidence to capture

- Sample log entries
- Dashboard screenshots or exported metrics definitions
- Alert routing record
- Runbook links or procedures

Acceptance criteria

Operators can determine service health, isolate common faults, and reach the correct response path using the telemetry that exists in production.

Owner

Operations or SRE lead

Review cadence

Per release and quarterly for telemetry quality

Phase 7: Deployment, rollback, and data safety

This phase ensures the release can be promoted safely and reversed if required. Rollback plans are often written but not executable; this phase checks the difference.

Purpose

Confirm deployment repeatability, rollback viability, and data protection during failure scenarios.

Checklist items

- ? Validate that the deployment procedure is documented and repeatable by a second operator.
- ? Test the rollback procedure in a non-production environment or during a controlled rehearsal.
- ? Review database migration steps for forward compatibility and rollback constraints.
- ? Confirm backup, restore, or snapshot procedures are available for stateful components.
- ? Validate that feature flags, configuration toggles, or traffic shifting can reduce blast radius during rollback.
- ? Document the maximum acceptable time to detect, mitigate, and restore service if the release fails.

Evidence to capture

- Deployment runbook
- Rollback rehearsal record
- Migration plan and schema notes
- Backup or restore evidence

Acceptance criteria

The team can deploy, observe, and revert the release using documented steps, and the data recovery strategy is sufficient for the application?s statefulness and risk profile.

Owner

Release engineer with database or platform support

Review cadence

Per release and after any deployment-process change

Common mistakes to avoid

A common mistake is assuming code rollback is enough when schema changes are not reversible. Another is forgetting to rehearse the rollback path with the same permissions and tooling used in production.

Phase 8: Pre-production validation and sign-off

This final phase brings the earlier checks together and confirms the release is ready for controlled production use. It should be the shortest phase if the earlier work was done properly.

Purpose

Confirm that the release passes end-to-end validation and that exceptions, if any, are explicitly accepted.

Checklist items

- ? Test the application in a production-like environment with production-like configuration.
- ? Validate the critical user journey or service transaction end to end.
- ? Review open risks, deferred defects, and exceptions with the approving owner.
- ? Confirm that all required evidence is attached to the release record.
- ? Assign a post-deployment verification owner and a monitoring window.
- ? Document the go/no-go decision and any conditions attached to approval.

Evidence to capture

- Pre-production test results
- Release approval record
- Risk acceptance notes
- Monitoring assignment

Acceptance criteria

The service passes the critical path test, all blocking issues are resolved or approved, and production monitoring responsibilities are assigned before release.

Owner

Service owner or change approver

Review cadence

Per release

Readiness scoring

Use a simple maturity score to decide whether the release is ready, conditionally ready, or blocked. Score each phase from 0 to 3.

- 0 = Not performed
- 1 = Partially performed, evidence incomplete, or acceptance unclear
- 2 = Performed with minor gaps that do not affect core readiness
- 3 = Complete, evidenced, and accepted

Maximum score: 24

Scoring guidance

- 21 to 24: Ready for production use
- 16 to 20: Conditionally ready; release only with documented exceptions and owner approval
- 0 to 15: Not ready; do not promote

A low score in configuration, security, or rollback should be treated as more serious than a low score in documentation quality. If a critical control fails, the release is blocked even when the total score looks acceptable.

Pass/fail decision rules

Use these rules to keep the decision simple and defensible.

- Pass only when all blocking checks in configuration, security, runtime health, observability, and rollback have evidence and acceptance.
- Fail when any production endpoint, secret, authentication rule, or rollback dependency is unverified.

- Fail when the release cannot be rebuilt, redeployed, or monitored using documented procedures.
- Conditional approval is acceptable only when the exception is limited, time-bound, and has an assigned mitigation owner.

Final takeaway

A useful .NET checklist is not just a sign-off form; it is proof that the application can be built, secured, deployed, observed, and recovered with predictable operational control. If the checklist produces unclear evidence or repeated exceptions, treat that as a signal to tighten the release process before the next promotion.