TEMPLATE

NET Production Readiness Review Checklist

A practical .NET operational readiness template to verify build integrity, configuration, security, observability, deployment safety, and rollback preparedness before production use.

.NET Operational Readiness Review Checklist

Resource type: Template Use case: Pre-production, release, and post-deployment validation
Applies to: .NET applications, APIs, background services, containers, and hosted deployments

---

Instructions

Use this checklist during release planning, pre-production review, and go-live validation.
For each item:

- Mark the check as complete only after verifying evidence.
- Record the artifact, log, dashboard, or test result that proves the item was checked.
- Assign an owner and due date where needed.
- Treat any failed acceptance criterion as a release blocker unless a documented exception and compensating control are approved.

Release information

Field	Value
Application / service name	
Release identifier	
Version / build number	
Target environment	
Deployment model	? Container ? VM ? Serverless ? Hosted service ? Other: ____
Planned release date	
Change window / freeze constraints	
Release owner	
Deployment approver	
Security reviewer	
Operations owner	

Evidence log

Checkpoint	Evidence captured	Owner	Status	Notes
Scope and release definition			? Pass ? Fail ? N/A	
Build integrity and dependency control			? Pass ? Fail ? N/A	
Configuration and secrets validation			? Pass ? Fail ? N/A	
Security and authorization checks			? Pass ? Fail ? N/A	

Runtime behavior and failure handling			? Pass ? Fail ? N/A
Observability and diagnostics			? Pass ? Fail ? N/A
Deployment safety and rollback readiness			? Pass ? Fail ? N/A
Data, state, and migration safety			? Pass ? Fail ? N/A
Operational support readiness			? Pass ? Fail ? N/A
Final approval			? Pass ? Fail ? N/A

---

1) Scope and release definition

Purpose: Confirm that everyone is reviewing the same application, version, and deployment target.

Checklist

- [ ] Confirm the application name, service boundary, and release identifier to be promoted.
- [ ] Review the target runtime version, SDK version, and deployment environment for compatibility.
- [ ] Validate the deployment model and document any platform-specific assumptions.
- [ ] Assign an operational owner, security reviewer, and deployment approver.
- [ ] Document expected production traffic, dependencies, maintenance windows, and 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 required reviewers are assigned. Any platform or version dependency that could alter behavior is documented and accepted.

---

2) Build integrity and dependency control

Purpose: Confirm the build output is reproducible and dependencies are controlled.

Checklist

- [ ] Confirm the build uses a pinned SDK or documented toolchain version.
- [ ] Review package references for unnecessary or unapproved dependencies.
- [ ] Validate package source trust, restore settings, and lock-file usage against 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.

Common mistakes to avoid

- Treating a successful local build as evidence of release readiness
- Allowing package versions to float without a documented policy

---

3) Configuration and secrets validation

Purpose: Confirm that the application can start and operate safely with production settings.

Checklist

- [ ] Validate that all required configuration keys are defined for the target environment.
- [ ] Review default values to ensure they do not enable unsafe production behavior.
- [ ] 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 intended production resources.
- [ ] Document 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 predictably when configuration is invalid. Production endpoints and flags are explicitly confirmed.

---

4) Security and authorization checks

Purpose: Confirm authentication, authorization, and security settings are aligned with policy.

Checklist

- [ ] 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 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 logs.
- [ ] Validate that dependency and runtime security scan 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.

Common mistakes to avoid

- Verifying only the happy path for authenticated users
- Assuming a framework default is secure without checking the actual production configuration

---

5) Runtime behavior and failure handling

Purpose: Confirm the application behaves predictably under normal and abnormal conditions.

Checklist

- [ ] 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
- Failure injection or timeout test results
- Background service stop/start validation

Acceptance criteria

The service starts successfully in the target environment, health checks reflect real state, failures are observable, and shutdown behavior is safe.

---

6) Observability and diagnostics

Purpose: Confirm the team can detect, diagnose, and support the service in production.

Checklist

- [ ] Verify structured logging is enabled and includes correlation identifiers where appropriate.
- [ ] Confirm log levels are appropriate for production and do not generate excessive noise.
- [ ] Validate that metrics exist for request rate, latency, error rate, saturation, and dependency health.
- [ ] Confirm distributed tracing or request correlation is available across key boundaries when required.
- [ ] Review dashboards and alerts for meaningful thresholds and actionable notifications.
- [ ] Confirm log retention, metric retention, and trace retention meet operational requirements.

Evidence to capture

- Dashboard links or screenshots
- Sample logs with correlation IDs
- Metric and alert inventory
- Retention policy evidence

Acceptance criteria

Operators can identify failures, trace requests, and receive timely alerts using approved production telemetry.

---

7) Deployment safety and rollback readiness

Purpose: Confirm the release can be deployed, verified, and reversed safely.

Checklist

- [ ] Validate deployment automation or runbooks are current and have been reviewed.
- [ ] Confirm the release supports a safe rollout strategy such as canary, staged, blue-green, or equivalent.
- [ ] Verify rollback steps are documented, tested, and time-bounded.
- [ ] Confirm health gates or post-deploy validation checks exist before full traffic cutover.
- [ ] Verify artifact versioning and image/package immutability for promoted builds.
- [ ] Confirm the team knows the abort criteria and who can stop the rollout.

Evidence to capture

- Deployment runbook or pipeline definition
- Rollback procedure and test results
- Promotion or health-gate checklist
- Artifact version and immutability proof

Acceptance criteria

The deployment path is repeatable, rollback is practical, and the team can halt promotion if health checks or metrics fail.

---

8) Data, state, and migration safety

Purpose: Confirm the release will not corrupt data or leave the application in an unrecoverable state.

Checklist

- [ ] Review database schema changes, migrations, or seed data for compatibility.
- [ ] Validate backward and forward compatibility where rolling deployments are used.
- [ ] Confirm backups, restore points, and recovery procedures are available and current.
- [ ] Test any data migration steps in a non-production environment.
- [ ] Review transaction boundaries, idempotency, and retry behavior for writes.
- [ ] Validate that cache, queue, or file-state changes are safe across rollout and rollback.

Evidence to capture

- Migration plan or schema diff
- Backup and restore evidence
- Dry-run migration result
- Compatibility test results

Acceptance criteria

Data changes are understood, tested, recoverable, and compatible with the selected deployment strategy.

---

9) Operational support readiness

Purpose: Confirm support teams can maintain the service after release.

Checklist

- [ ] Confirm the support model, escalation path, and on-call coverage for the release window.
- [ ] Ensure operational documentation is updated with runbooks, owner contacts, and known issues.
- [ ] Verify that common alerts have corresponding triage steps.
- [ ] Confirm access to logs, dashboards, and deployment tools is available to the right roles.
- [ ] Review any open risks, known defects, or deferred fixes with explicit ownership.
- [ ] Capture final release sign-off from required approvers.

Evidence to capture

- Support and escalation matrix
- Updated runbooks or knowledge base links
- Access verification record
- Final approval record

Acceptance criteria

The service can be supported by the operations team, and ownership for incidents, escalation, and known risks is clear.

Final release decision

Go / No-Go review

Question	Answer	Notes
Is the release scope and version fully defined?	? Yes ? No	
Is the build reproducible and dependency-controlled?	? Yes ? No	
Are configuration and secrets validated for production?	? Yes ? No	
Are security and authorization checks complete?	? Yes ? No	
Are runtime behavior and failure modes acceptable?	? Yes ? No	
Are observability and diagnostics sufficient?	? Yes ? No	
Is deployment safety and rollback ready?	? Yes ? No	
Are data and migration risks controlled?	? Yes ? No	
Is operational support ready?	? Yes ? No	

Decision

- [ ] Go
- [ ] No-Go

- [ ] Go with approved exception(s)

Approvals

Role	Name	Signature / Approval	Date
Release owner			
Operations owner			
Security reviewer			
Deployment approver			

Exception register

Item	Risk	Compensating control	Owner	Expiry date

Post-deployment validation

Complete after deployment and before full release closure.

- [ ] Confirm the deployed version matches the approved release identifier.
- [ ] Verify key user journeys or service endpoints succeed in production.
- [ ] Confirm health checks are green and telemetry is flowing.
- [ ] Review error logs and alerts for regressions.

- [ ] Validate rollback remains available until the release is fully stable.

Post-deploy evidence

- Production verification logs
- Dashboard snapshot or alert status
- Smoke test results
- Closure note

Quick review summary

Category	Status	Notes
Scope and release definition		
Build integrity and dependency control		
Configuration and secrets validation		
Security and authorization checks		
Runtime behavior and failure handling		
Observability and diagnostics		
Deployment safety and rollback readiness		
Data, state, and migration safety		
Operational support readiness		

Reviewer notes

Use this section for risks, follow-ups, exceptions, and operational observations.