VectraOps

Knowledge Library

CHECKLIST

Programming in .NET Production Readiness Checklist

A practical, vendor-neutral checklist for evaluating whether a .NET application is ready for production use, with operational, security, deployment, and observability checks.

Programming in .NET Production Readiness Checklist

Use this checklist to evaluate a .NET workload before production rollout. It is designed for operational, engineering, and security review.

---

1) Workload fit

- [ ] The workload is primarily business logic, service orchestration, integration, automation, API, worker, or internal application code.
- [ ] .NET is being selected for a clear reason, not just familiarity.
- [ ] The team has confirmed that a managed runtime is acceptable for the workload.
- [ ] The application model is defined: API, worker, console, desktop, library, or hybrid.
- [ ] The expected workload profile is documented: latency, throughput, concurrency, and uptime needs.
- [ ] The deployment environment is identified: Linux, Windows, containers, serverless, or VM.

2) Runtime and version control

- [ ] The target .NET runtime version is specified.
- [ ] The version is supported for the intended production window.
- [ ] The application has been tested against the exact runtime version planned for deployment.
- [ ] The runtime model is confirmed: JIT, AOT, or other supported compilation mode.
- [ ] Any framework-dependent or self-contained deployment choice is documented.
- [ ] The base image or host runtime is approved by platform policy.
- [ ] Patch and upgrade cadence is defined.

3) Compatibility checks

- [ ] All required NuGet packages and native dependencies are listed.
- [ ] External services and APIs used by the application are identified.
- [ ] TLS, certificate, and authentication requirements are confirmed.
- [ ] Any OS-specific behavior is documented.
- [ ] Any Linux or Windows file path, permission, or service-account assumptions are verified.
- [ ] Container constraints, if applicable, are tested.
- [ ] Hardware or OS-level dependencies are either removed or explicitly accepted.

4) Build and packaging

- [ ] Build commands are standardized and documented.
- [ ] The build is reproducible from a clean environment.
- [ ] Release artifacts are versioned.
- [ ] The deployment package includes only required files.
- [ ] Environment-specific configuration is externalized.
- [ ] Secrets are not embedded in source code, build scripts, or artifacts.
- [ ] Dependency versions are pinned or governed according to policy.

- [ ] The application can be promoted through environments without code changes.

5) Security review

- [ ] Authentication and authorization requirements are defined.
- [ ] Least-privilege principles are applied to service accounts and identities.
- [ ] Secrets are stored in an approved secrets manager or equivalent secure store.
- [ ] Sensitive data is protected in transit and at rest as required.
- [ ] Input validation is implemented for all external inputs.
- [ ] Dependency scanning is enabled.
- [ ] Vulnerability remediation ownership is assigned.
- [ ] Patch policy for runtime, libraries, and base images is clear.
- [ ] Supply chain controls are in place for package sources.
- [ ] Audit logging requirements are defined and tested.

6) Observability

- [ ] Logging is structured and searchable.
- [ ] Log levels are appropriate for production.
- [ ] Correlation or trace identifiers are supported where needed.
- [ ] Metrics are emitted for latency, errors, throughput, and saturation.
- [ ] Health checks are implemented and validated.
- [ ] Readiness and liveness behavior are defined if the workload is containerized.
- [ ] Alerts are mapped to actionable operational thresholds.
- [ ] Log retention and access controls are defined.

7) Performance and resilience

- [ ] The application has been load tested or benchmarked for expected traffic.
- [ ] Startup time is acceptable for the hosting model.
- [ ] Memory usage is understood under normal and peak load.
- [ ] Threading and async behavior have been reviewed.
- [ ] Retry behavior is bounded and does not amplify failures.
- [ ] Timeouts are configured for outbound calls.
- [ ] Caching, pooling, or batching choices are intentional.
- [ ] Backpressure or queue limits are defined where relevant.
- [ ] Failure modes have been tested, including dependency outage scenarios.

8) Data and state handling

- [ ] Data persistence requirements are documented.
- [ ] Database connections and migrations are controlled.
- [ ] Transaction boundaries are understood.
- [ ] File system usage is approved and tested for the target platform.
- [ ] Temporary files, caches, and buffers have a lifecycle policy.
- [ ] State is not unintentionally stored in process memory alone.
- [ ] Backup and recovery expectations are clear if the application owns state.

9) Deployment and operations

- [ ] The deployment topology is documented.
- [ ] Rollback steps are defined and tested.
- [ ] Release approval criteria are explicit.
- [ ] Configuration changes can be made independently of code changes.
- [ ] Runtime flags or feature toggles are documented if used.
- [ ] Resource requests and limits are set appropriately.

- [ ] Service accounts, permissions, and network access are verified.
- [ ] Operational runbooks exist for startup, failure, and incident response.
- [ ] Ownership and support contacts are named.

10) Testing and validation

- [ ] Unit tests cover critical business logic.
- [ ] Integration tests cover key service boundaries.
- [ ] End-to-end tests cover the main production path.
- [ ] Security-relevant behaviors are tested, not just reviewed.
- [ ] Compatibility testing has been performed in a production-like environment.
- [ ] Configuration validation fails fast when required settings are missing.
- [ ] Negative tests exist for dependency failures and invalid input.

11) Production decision check

- [ ] The workload fit is clear and documented.
- [ ] Runtime version and deployment model are approved.
- [ ] Security and dependency risks are accepted or mitigated.
- [ ] Observability is sufficient for support.
- [ ] Performance is acceptable under expected load.
- [ ] Rollback is available and documented.
- [ ] A named owner is accountable for operation after release.

Final go-live question

Before approving production, ask:

Have we verified that this .NET workload fits the environment, is supportable in production, and can be observed, secured, and rolled back with confidence?

If any answer is unclear, pause the release and close the gap first.