Code Check-In – Process, Policy & Checklist

# 📌 Scope

This document outlines the end-to-end code check-in to production process for a web application using GitHub and Azure Dev Boards, progressing through Development and QA environments, and involving Developers, Tech Leads or any other team members.

**Standardized Git branching strategy: main, feature/\*, hotfix/\*, release/\***

**Note: Below Steps are some processes must and some are optional once project grows we should adopt the process and we must follow process**

# ✅ Developer Code Check-In Checklist

|  |  |
| --- | --- |
| Step | Description |
| ✅ 1.1 | Pull the latest changes from the `main` branch |
| ✅ 1.2 | Create or checkout a feature branch from `main`  **(e.g., Task-1001-Description)** |
| ✅ 1.3 | Link the corresponding Azure Dev Board work item **(e.g., Task-1001)** |
| ✅ 1.4 | Write or update unit tests; ensure test coverage meets the minimum threshold (e.g., >80%) **(Optional ignore)** |
| ✅ 1.5 | Follow coding standards: naming conventions, formatting, logging, and error handling |
| ✅ 1.6 | Run local builds **(Must)** and unit tests; ensure all pass **(Optional ignore)** |
| ✅ 1.7 | Commit with a meaningful message, e.g., **` Task-1001: Description`** |
| ✅ 1.8 | Push the feature branch and raise a Pull Request (PR) to the `Main` branch  Once approval done **do either Merge/ Squash Merge. Keep checking status** |

# 🔍 Pull Request (PR) Review Policy – Tech Lead

|  |  |
| --- | --- |
| Step | Policy |
| 🔍 2.1 | Minimum one reviewer must approve the PR **(Initial: Team Leads or Optional: Dev Team/Manager)** |
| 🔍 2.2 | PR must pass the automated CI pipeline (build, test, lint) **(Optional Once ready)** Respective Team should take care build successful and deployed |
| 🔍 2.3 | Verify implementation aligns with acceptance criteria and work item |
| 🔍 2.4 | Ensure no secrets or hardcoded credentials are included |
| 🔍 2.5 | Review code for functionality, exception handling, performance, and scalability etc. (Some Optional Some must) |
| 🔍 2.6 | Only merge after all approvals; delete the feature branch post-merge (**Optional: Dev can take care this**). |

# 🧪 QA Deployment Process (Optional as of now)

|  |  |
| --- | --- |
| Step | Description |
| 🚀 3.1 | Trigger the CI/CD pipeline to deploy the merged `Main` branch to QA |
| 🚀 3.2 | Tag the QA deployment in Git **(Optional)** |
| 🚀 3.3 | QA team validates the build using manual and/or automated test cases |
| 🚀 3.4 | Log bugs in Azure Dev Boards and address them through new PRs |

# 🚀 Production Deployment Process (Optional as of now)

|  |  |
| --- | --- |
| Step | Description |
| 💪 4.1 | After QA sign-off, create a release\_version branch from `Main` branch |
| 🔒 4.2 | Apply a code freeze and conduct regression testing |
| ✅ 4.3 | Obtain necessary approvals from the Tech Lead, QA, and Product Owner **(Optional)** |
| 🚀 4.4 | Merge the release\_version branch into ` release` |
| 🔀 4.5 | Trigger the production deployment pipeline |
| 📈 4.6 | Monitor logs, Application Insights, and system health post-deployment**(Optional)** |

# 🔁 End-to-End Process Flow

Feature Branch  
 ↓  
Pull Request (Feature → Main)  
 ↓  
Tech Lead/Team Review & CI Checks  
 ↓  
Merged to Main → CI/CD Deploys to QA  
 ↓  
QA Testing → Bug Fixes as Needed  
 ↓  
Release\_Version Branch from Main  
 ↓  
Merge to Release Branch → CI/CD Deploys to Production  
 ↓  
Post-Deployment Verification

# 📋 Best Practices (Option but we should follow day to day process)

* Always **rebase** your feature branch with main before creating a PR
* Commit frequently and push changes regularly
* Keep Pull Requests small, atomic, and easy to review
* Integrate automated tests and linters into the CI/CD pipeline
* Maintain backward compatibility, especially for APIs
* Avoid committing unnecessary files such as node\_modules, build outputs, or temporary configs
* Ensure secrets and credentials are managed via secure key vaults and not stored in code
* Review logs and telemetry to catch post-deployment issues early
* Follow a standardized Git branching strategy: main, feature/\*, hotfix/\*, release/\*