Hein+Fricke

(Code Review and Approval Procedure )

**Purpose:** Internal Audit Compliance & Delivery Process Governance

Hein+Fricke



## Version History

| Creating Person | Version | Date (DD-MM-YYYY) |
|---|---|---|
| Satyakam kaul | 1.0.0 | 24-09-2025 |
| Satyakam kaul | 1.0.1 | 05-10-2025 |
| | | |
| | | |

# 1. Introduction

- **Purpose**: Establish a standardized process for reviewing and approving code changes to maintain quality, catch issues early, and ensure audit traceability using tools like Git.

- **Scope**: Covers all code changes in software projects across development teams, focusing on Git-based repositories.
- **Objectives**: Enforce coding standards, reduce bugs, promote collaboration, and keep detailed records for internal audits.
- **Key Terms**:
    - Git: Version control system for tracking code changes.
    - Branching: Creating separate code lines (e.g., feature branches) in Git.
    - Pull Request (PR): Git mechanism for proposing and reviewing code changes.
    - Reviewer: Assigned team member who evaluates the PR.
- **Standards**: Align with company coding guidelines and audit requirements for traceability.
- **Audience**: Developers, reviewers, team leads, and auditors.
- **How to Use**: Follow this for all code changes; store in Confluence for access.

## 2. Roles and Responsibilities

- **Key Roles**:
    - Developer: Makes code changes, runs tests, creates PRs.
    - Reviewer: Reviews PRs, adds comments, approves or requests changes.
    - Team Lead: Assigns reviewers, resolves issues, ensures process adherence.
- **Developer Tasks**:
    - Use Git for branching and code changes.
    - Run test cases before creating PRs.
    - Act on reviewer comments and update PRs.
- **Reviewer Tasks**:
    - Review code, add comments on issues.
    - Re-review updates until PR is approved.
- **Team Collaboration**:
    - Use Slack #code-reviews for notifications.
    - Link PRs to Jira tickets for tracking.

## 3. Code Review and Approval Process Overview

- **Goals**: Ensure code changes are reviewed thoroughly via Git, with tests run and comments addressed before approval.

- **Core Steps** :
    - Use Git for all version control.
    - Apply branching strategy for code changes.
    - Make code changes and run test cases.
    - Create Pull Request in Git.
    - Assign reviewer(s).
    - Reviewer reviews and add comments.
    - Comments are acted upon by the developer.
    - Review and repeat until PR approval.
    - Merge approved PR.
- **SDLC Fit**: Integrate with Agile sprints (review PRs daily) or Waterfall phases; track in Jira.
- **Change Tracking**: Use Git commits linked to Jira (e.g., “Fix JIRA-123”).
- **Process Flow**:
    - Step 1: Developer branches in Git and makes changes.
    - Step 2: Run tests locally or in CI.
    - Step 3: Create PR, assign reviewer.
    - Step 4: Reviewer adds comments; developer acts on them.
    - Step 5: Repeat reviews until approval; merge PR.
- **Exceptions**: Emergency changes allow quick approval with Team Lead sign-off; log in Jira.
- **Version Control**: Require Git branches like feature/JIRA-123; protect main.
- **Automation**: Use GitHub Actions for auto-tests on PR creation.

## 4. Code Review Planning

- **Review Types**:
    - Feature: For new code changes (e.g., new API).
    - Bugfix: For minor updates.
    - Emergency: For urgent fixes.
- **Scheduling**:
    - Plan reviews in daily standups; set 24-hour turnaround.
    - Use Git branching to isolate changes.
    - Notify via Teams when PR is ready.
- **Review Criteria**:
    - Code changes follow style guides.

    - Test cases run and pass (e.g., >90% coverage).
    - No security issues.
- **Risk Checks**:
    - Verify code changes don't break existing features.
    - Ensure test cases cover edge cases.
- **Mitigation Plans**:
    - Run automated tests before PR.
    - Use Git rebase for clean history.
- **Approvals**:
    - At least 1 reviewer; 2 for critical changes.
    - Log in Git PR.
- **Communication**:
    - Share PR links in Teams.
    - Update Jira with status.
- **PR Description**:
    - Include Jira ID, code changes summary, test results.
- **Dependencies**:
    - Check library versions in Git commits.

## 5. Code Review and Approval Execution

- **Pre-Review Activities:**
    - **Git and Branching**: Use Git; create branches (e.g., git checkout -b feature/JIRA-123).
    - **Code Changes**: Do code changes
    - **Run Test Case**: Execute unit/integration tests local
    - **Commit Changes**: Commit changes (e.g., git commit -m "JIRA-123: Add login")
    - **Create Pull Request**: Push branch, create PR in GitHub/Bitbucket with description.
    - **Assign Reviewer**: Select reviewer(s) in PR tool or Jira.
    - **Checklist**: Confirm tests pass, link Jira, notify on Teams.
- **Review Execution**:
    - **Assign Reviewer**: Team Lead confirms assignment.
    - **Reviewer Reviews and Adds Comments**: Evaluate code, add feedback on standards, tests, and issues.

    - **Comments Are Acted Upon**: Developer updates code, commits, and notifies reviewer.
    - **Review and Repeat**: Re-review updates until all issues resolved and PR approved.
    - **Tools**: GitHub/Bitbucket for reviews; SonarQube for quality.
    - **Checklist**: Verify functionality, style, tests.
    - **Feedback**: Use “Request Changes” or “Approve” in Git tool.
    - **Logging**: Record comments/approvals in Git.
- **Post-Review Activities**:
    - **Approval**: Reviewer approves; merge PR (e.g., squash merge).
    - **Verification**: Run CI tests post-merge.
    - **Notifications**: Post merge in Teams; update Jira.
    - **Documentation**: Update Confluence if needed.

## 11. Document Control and Versioning

- **Owner**: Team Lead; backup Senior Developer.
- **Changelog**: v1.0 (2025-09-22): Initial based on user points.
- **Review**: Every 6 months.
- **Retention**: 3 years in Confluence/AWS S3.
- **Access**: Team leads edit; auditors view.
- **Updates**: Approve via Team Lead/CTO.

## 12. Appendices

- **Templates**: PR template, review checklist.
- **Glossary**: Git, PR, Branching.
- **References**: Coding standards policy.
- **Contacts**: Team Lead: [Name], [Email], Slack @[handle].
- **Sample Logs**: PR approval: “Approved by [Reviewer] at [Date].”
- **Checklists**: Pre-review table (Git, tests, PR).