Code Review Workflows

Evidence, human judgment, and policy-approved automation

Access and cost preflight

Use Enterprise Cloud as the governance baseline.

1. Verify current official GitHub documentation and customer administrator policy.
2. Confirm repository scope, data classification, and review permissions.
3. Use a non-sensitive pull request with explicit acceptance criteria.
4. For metered work, define a customer-owned threshold, escalation route, and stop guard.

Review is a human responsibility

Automation may raise questions. A reviewer decides whether a change is correct, safe, maintainable, and within scope.

```
Issue and acceptance criteria
        ↓
Proposed change and checks
        ↓
Human review
        ↓
Accept, request revision, or implement manually
```

Review rubric

- Correctness against acceptance criteria
- Tests and validation evidence
- Security, dependency, and permission implications
- Scope and maintainability
- Repository conventions and human approval

Conflict-resolution exercise

1. Read both branch intentions.
2. Write the merged behavior and tests before editing.
3. Resolve the conflict manually in the repository sandbox.
4. Run checks and request review.
5. Demonstrate an approved assisted workflow only after checking current documentation and customer policy.

No-access fallback

Use the intentional defects and conflict scenario for a human-only review. Record findings as pull-request comments, complete the fixes manually, and compare with the solution reference.

Agenda and time plan

Time	Topic	Trainer move
0:00	Review workflow and roles	Define accountability
0:08	Summaries and descriptions	Separate draft from evidence
0:18	Manual and automatic review	Show policy controls
0:28	End-to-end PR demo	Apply the rubric
0:43	Suggestions and conflicts	Preserve author intent
0:51	Limits and adoption	Design human gates
0:58	Lab handoff and Q&A	Assign reviewer roles

Transition: “Copilot can increase review coverage; it cannot own the decision.”

Copilot across a PR lifecycle

```
Issue and acceptance criteria
          ↓
Branch and bounded commits
          ↓
Draft summary and PR description
          ↓
Automated checks + Copilot first pass
          ↓
Human review and author response
          ↓
Approval, revision, or close
```

Each stage gives the accountable reviewer evidence.

Commit messages and PR descriptions

Copilot can draft text from the visible change, but authors must verify:

- the user-visible behavior is accurate;
- tests and results are not invented;
- breaking changes and migrations are explicit;
- security and dependency implications are named;
- issue links and acceptance criteria match the work;
- no sensitive details appear in generated prose.

A polished summary can still be wrong.

A reviewable PR template

```
## Problem
What user-visible problem does this solve?

## Change
List the bounded implementation choices.

## Validation
- [ ] Focused tests
- [ ] Existing checks
- [ ] Manual edge case

## Risk and rollback
Name data, dependency, security, and recovery considerations.
```

Requesting Copilot review

For a manual request:

1. Confirm the PR is ready for a first pass.
2. Verify current availability, permissions, and repository policy.
3. Request the Copilot review through the supported PR control.
4. Wait for comments and review them as hypotheses.
5. Resolve, reject with rationale, or implement a verified fix.
6. Request human review with the complete evidence.

Do not use Copilot review to bypass required reviewers.

Automatic review configuration

Automatic review can be configured at repository or organization scope when supported.

Before enabling:

- identify eligible repositories and excluded data;
- define when reviews run and who owns triage;
- preserve branch protection and required human approvals;
- monitor noise, missed findings, and review delay;
- document an off switch and escalation owner.

Trainer cue: Demonstrate navigation, not fixed UI labels that may change.

Quality gates

Gate	Evidence
Ready for first pass	Clear description, focused diff, tests available
Ready for human review	Checks pass; AI comments triaged
Ready for approval	Acceptance criteria met; risks reviewed
Ready for merge	Required approvals and branch protections satisfied

Copilot review comes before human approval. It never replaces it.

Demo PR: intentional defect

```
export function canViewInvoice(user, invoice) {
  return user.role === "admin" || user.id === invoice.ownerId;
}
```

The requirement says suspended users must never view invoices.

```
if (user.status === "suspended") return false;
return user.role === "admin" || user.id === invoice.ownerId;
```

Ask which tests and policy evidence should accompany this suggestion.

Demo narrative: complete lifecycle

1. Open the issue and read acceptance criteria aloud.
2. Inspect the focused branch diff before generated prose.
3. Generate or improve the PR description; correct unsupported claims.
4. Run required checks and request Copilot review.
5. Classify each comment: valid, uncertain, noise, or out of scope.
6. Apply one verified fix and add a regression test.
7. Request human review and record the final decision.

Reviewing suggested fixes

Before applying:

- restate the finding in your own words;
- reproduce the defect or identify the violated requirement;
- inspect the complete patch, not only highlighted lines;
- add or update a test that fails before the fix;
- run broader checks appropriate to the risk;
- retain author ownership of the resulting code.

One-click application saves typing. The reviewer still decides.

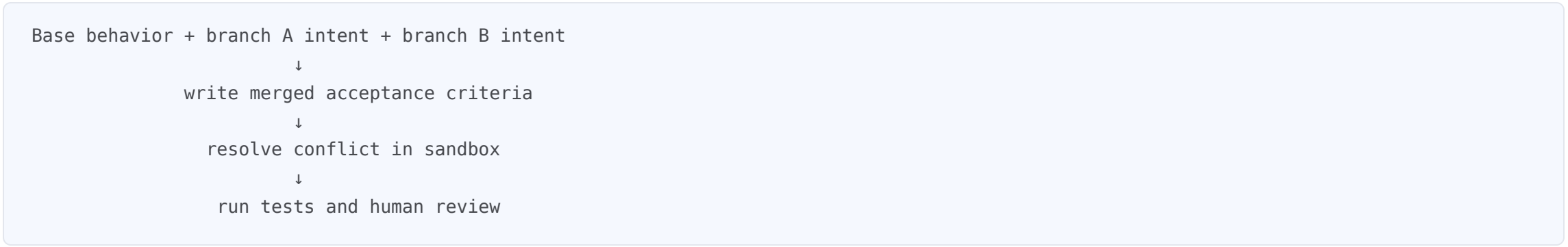
What review may catch

- localized correctness issues;
- missing error handling or tests;
- suspicious patterns visible in the diff;
- inconsistent naming or repository conventions;
- simple maintainability concerns.

What it may miss

- unstated product intent;
- cross-system behavior absent from context;
- runtime, operational, legal, or organizational risk;
- subtle authorization and data-boundary flaws;
- deliberately misleading but plausible changes.

Merge conflict resolution



If an approved cloud-agent workflow is available, compare its proposal with the manual resolution. Do not blindly choose “ours” or “theirs.”

Adoption patterns

Start: Optional first-pass review on a small set of repositories.

Measure: Actionable findings, false positives, review time, rework, and defects that reach users.

Adapt: Tune scope, guidance, ownership, and escalation.

Expand: Only when quality, security, and developer experience meet team criteria.

Keep a manual path for unavailable features and exceptional repositories.

Facilitation prompts

- Which comment needs domain expertise to evaluate?
- What evidence would turn an uncertain comment into a decision?
- Who resolves review noise?
- Which repositories should not use automatic review?
- How will the team detect over-reliance?

Transition: “Now apply the workflow as both author and reviewer.”

Lab handoff

Learners will:

1. create a branch containing intentional quality issues;
2. open a PR and improve its summary;
3. request review and classify every comment;
4. apply only verified suggestions with tests;
5. resolve a prepared merge conflict;
6. add a partner’s human review and reflect on review quality.

Deliverable: A PR with layered evidence and an explicit human decision.

Remember

1. Generated summaries must be verified against the diff.
2. AI review is a first pass before accountable human review.
3. Suggested fixes require reproduction, inspection, and tests.
4. Conflict resolution starts from intended merged behavior.
5. Adoption should be measured and reversible.

Questions and lab readiness

Which review decision must never be delegated without human evidence?

- Who is the final reviewer?
- Which check is required before requesting review?
- Is the conflict scenario ready?