Key Improvements for Git Repository Operations

The optimized rules emphasize a simplified **GitHub Flow**-inspired branching strategy, with clear separation between local editing and GitHub collaboration. This reduces merge conflicts (e.g., avoiding unresolved `<<<<<<<` markers, as defined in Git official docs) through frequent rebasing and squash merges. Verify Gate integrates via GitHub Actions required checks, ensuring automated validation before merges. Evidence retention limits to recent-3 logs to prevent bloat while maintaining auditability.

- **Branching**: main (stable releases), integrate/YYYYMMDD (daily/weekly integration), feat/<descriptive> (feature work) – aligns with 1Part=1Branch principle.
- **Conflict Prevention**: Prefer rebase over merge for linear history; use squash on PR merge (GitHub Docs recommendation).
- **Automation**: Require Fast/Full Verify as status checks on PRs via GitHub Actions.
- **Evidence Policy**: Keep only latest 3 verify reports; archive older ones.

These changes minimize accidents, improve beginner-friendliness, and ensure reproducibility without overcomplicating the SSOT governance.

Enhanced Branching and Workflow Guidelines

Adopt a hybrid of GitHub Flow for simplicity:

- Create short-lived feat branches from integrate/<date>.
- Use PRs to merge into integrate (with squash).
- Periodically merge integrate into main for releases (no-ff for traceability).

This avoids long-lived branches that accumulate conflicts (common pain point in trunk-based development).

Merge Conflict Mitigation Strategies

Git presents conflicts with standard markers:

```
text
<<<<<<< HEAD (yours)
Your changes
=======
Their changes
>>>>>>> branch (theirs)
```

Always resolve by editing files, removing markers entirely, then `git add` and commit.
Prevention tactics:

- Rebase frequently: `git pull --rebase origin integrate/<date>` (resolves conflicts early, creates cleaner history per Atlassian Git tutorials).
- Use GitHub's "Update branch" button in PRs to rebase automatically.
- Enable branch protection: Require linear history (disallow merge commits if preferred) or passing status checks.

Verify Gate Automation

Move Fast Verify to GitHub Actions:

- Trigger on push/PR to feat and integrate branches.
- Fail PR if Verify script returns non-zero.
- Full Verify on merge to integrate/main.
  This enforces checks without manual runs, reducing "forgot to verify" errors (GitHub Docs: branch protection rules).

Evidence and Log Management

- Naming: `YYYYMMDD_HHMMSS_<mode>_<result>_<branch>.md` (e.g., `20260112_001800_Fast_PASS_feat-part10.md`).
- Retention: Keep recent-3 per branch; script to archive/delete older (e.g., `find evidence/ -mtime +14 -delete`).
- Prevents disk bloat while ensuring recent audits are available.

Detailed Operational Enhancements and Rationale

The original SSOT design excels at governance (ADR-first, sources immutability) but lacks concrete Git mechanics, leading to real-world friction in distributed workflows. This deep dive incorporates official Git/GitHub best practices (as of 2026) to close those gaps while preserving core principles like minimal diffs, HumanGate, and Verify/Evidence obligations.

1) Expanded Confusion Points (with Root Causes and Evidence)

- **P0: Conflict Marker Leakage**
  Example: Unresolved `<<<<<<< HEAD` markers committed after sloppy merge.
  Cause: Manual merge without post-resolution check; no automated detection.
  Evidence: Git docs explicitly define marker format and warn that unresolved markers break builds.
- **P0: Branch Proliferation & Stale Integrations**
  Example: Multiple dated integrate branches accumulate, causing cross-branch conflicts.
  Cause: No lifecycle policy for integrate branches (create weekly vs daily?).
  Evidence: GitHub Flow recommends short-lived branches; long-lived ones increase conflict surface.
- **P1: Local-Only Verify Leading to Divergence**
  Example: Local Fast Verify passes, but GitHub CI fails due to environment differences.
  Cause: Verify scripts not executed in shared CI environment.
- **P1: Evidence Bloat vs Loss**
  Example: Thousands of verify logs accumulate, or critical logs deleted accidentally.
  Cause: No explicit retention policy beyond "deletion prohibited."
- **P2: Rebase vs Merge Confusion**
  Example: Team members mix strategies, creating non-linear history that's hard to audit.
  Cause: No explicit preference declared.

2) Refined "Zero-Ambiguity" Operational Flow (Checklist with Commands)

Daily/Feature Workflow (Local → GitHub)

1. Sync base: `git checkout integrate/20260112 && git pull origin integrate/20260112`
2. Create feat branch: `git checkout -b feat/partXX-description`
3. Edit minimally (1Part focus) → commit: `git commit -m "feat: description (ADR-XXXX)"`
4. Run local Fast Verify: `pwsh ./checks/verify_repo.ps1 -Mode Fast` → Save log to evidence/
5. Push: `git push origin feat/partXX-description`
6. Open PR on GitHub: base = integrate/20260112, enable "squash and merge"

PR Review & Merge

1. Reviewer runs/inspects Verify in CI (GitHub Actions).
2. If conflicts: Click "Resolve conflicts" on GitHub or locally rebase: `git pull --rebase origin integrate/20260112` → resolve → force push.
3. Require: 1 approval (HumanGate) + passing checks.
4. Merge via Squash (keeps history clean).

Integration → Release

1. Weekly: Create new integrate/20260119 if needed.
2. On main release: `git checkout main && git merge --no-ff integrate/20260112` → tag → push.
3. Run Full Verify post-merge.

Evidence Cleanup (Monthly Script)

```
PowerShell

# Keep recent-3 per type
Get-ChildItem evidence/verify_reports/ *.md | Sort-Object LastWriteTime -Desc
```

3) Proposed Design Doc Additions (Ready-to-Paste)

```
Markdown

### R-GIT01: Branch Strategy 【MUST】
- main: Protected stable branch; releases only.
- integrate/YYYYMMDD: Integration branch (create weekly on Monday; archive af
- feat/<partXX>-<desc>: Short-lived feature branches (delete after merge).
Root: GitHub Flow[](https://docs.github.com/get-started/quickstart/github-flo

### R-GIT02: Merge Conflict Policy 【MUST NOT】
- Never commit with unresolved conflict markers (<<<<<<<, =======, >>>>>>>).
- Detection: Add to verify_repo.ps1 → `git diff --check` (fails on markers).
- Resolution: Always edit → remove markers → git add → continue.
Root: Git official[](https://git-scm.com/docs/git-merge).

### R-GIT03: Rebase Preference 【SHOULD】
- Keep branches current: `git pull --rebase` before push/PR.
- PR merges: Use squash (linear history).
Root: Atlassian comparison (rebase for clean history).

### R-GIT04: GitHub Actions Integration 【MUST】
- Verify scripts run as required status checks on PRs to integrate/*.
- Branch protection on integrate and main: Require passing checks + 1 review.
Root: GitHub Docs status checks.

### R-GIT05: Evidence Retention 【MUST】
- Keep recent-3 verify reports per branch/mode.
- Auto-archive older logs monthly.
```

4) Verify Gate Deep Improvements

- **Mis-detection Fixes**: Add `--ignore-pattern` for temporary files; include `git diff --check` for whitespace/marker detection.
- **Leak Prevention**: Required CI checks block merges on failure.
- **Naming/Storage**: Timestamp + mode + result + branch for traceability.
- **Retention**: Recent-3 active + git-tagged archives for historical audits.
- **Automation Level**: Fast on push, Full on merge to integrate (via Actions workflow).

5) Expanded Troubleshooting Table

Symptom	Likely Cause	Resolution Steps
`<<<<<<<` markers in committed file	Unresolved merge not cleaned	`git diff --check` → edit file → remove markers → `git add && git commit --amend`
PR shows conflicts	Branch diverged from integrate	GitHub UI: "Update branch" (rebase) or local: `git rebase origin/integrate/...`
Verify fails only in CI	Local env differs from GitHub runner	Standardize runner (ubuntu-latest) → add env vars to workflow
Too many evidence logs	No cleanup	Run retention script → commit cleanup manifest
PR cannot merge (red checks)	Verify Action failed	View Actions log → fix → re-run

These refinements make the SSOT repository dramatically more robust for both solo and team use while staying true to zero-ambiguity and accident-prevention goals.

Key Citations

- Git Merge Documentation: https://git-scm.com/docs/git-merge
- GitHub Flow Guide: https://docs.github.com/get-started/quickstart/github-flow
- GitHub Branch Protection: https://docs.github.com/en/repositories/configuring-branches-and-merges-in-your-repository/managing-protected-branches/about-protected-branches
- Atlassian Merging vs Rebasing: https://www.atlassian.com/git/tutorials/merging-vs-rebasing
- GitHub Actions Status Checks: https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/proposing-changes-to-your-work-with-pull-requests/about-pull-requests (related checks context)