

## 1. Novel Contributions (今回新しく持ち帰る点)

- ・CI必須のBranch Protectionでmainへの直pushをブロックし、PR経由のみ許可する運用を強制。
- ・Verify Gateに「危険コマンド検出」(rm -rf, git push --force等) を追加し、docs/内の記述をスキャン。
- ・Evidence Packに「approvals.json」を必須とし、HumanGate承認のタイムスタンプをJSON形式で記録。
- ・Releaseの不変化を保証するための「git tag --sign」による署名を導入、GPGキー必須。
- ・SBOM生成をCycloneDX形式に限定し、必須ツールとしてsyftを指定（現実的な最小セット）。
- ・RAG更新プロトコルを「docs更新トリガ→自動再インデックス→スナップショット検証→Evidence保存」のRunbookに落とし込む。
- ・Verifyの必須セットに「未決事項検出」を追加、各Partの「11. 未決事項」セクションをgrepでカウント。
- ・Evidence保持ポリシーを「recent-3 + 永久保存の監査用」に分け、容量超過時のアーカイブ手順を定義。
- ・CI落ち条件に「証跡不足」を追加、evidence/verify\_reports/の存在をファイル数で機械判定。
- ・Releaseロールバックを「git revert + Verify再実行 + Evidence追記」の3ステップに簡略化。
- ・RAGスナップショットを「YYYYMMDD\_HHMMSS\_kb\_snapshot.tar.gz」形式で保存、sha256検証必須。
- ・Verify推奨セットに「外部リンク生存確認」を追加、curlヘッドリクエストでステータスチェック。

## 1. 「CI/ブランチ保護の要件」章案 (文章化：コピペできる)

## CI/ブランチ保護の要件

本プロジェクトでは、GitHubリポジトリのmainブランチを保護し、事故防止と品質担保を強制する。以下の設定をリポジトリ設定で適用する：

- ・**PR必須:** mainへのpushを禁止し、全変更をPull Request (PR) 経由に限定。Branch Protection Ruleで「Require pull request reviews before merging」を有効化し、少なくとも1名の承認を必須とする。
- ・**必須チェック:** PRマージ前にGitHub ActionsでVerify Gateを実行。ワークフローは .github/workflows/verify.yml に定義し、Fast Verify (リンク切れ/用語揺れ/Part間整合/未決事項検出) と危険コマンド検出を含む。CI失敗時はマージをブロック。
- ・**main直push禁止:** Branch Protection Ruleで「Restrict who can push to matching branches」を有効化し、mainへの直接pushを管理者以外禁止。管理者もPR経由を原則とする。
- ・**必須レビュー:** Code ownersを .github/CODEOWNERS で定義し、docs/変更時は HumanGate承認者を必須レビューに指定。レビュー承認後、CI PASSでマージ可能。
- ・**運用ルール:** ローカルでPASSしてもCIで落ちる場合（例：環境差によるリンク切れ、証跡ファイル欠落）はPRをクローズし、再修正。危険コマンド（rm -rf, git push --force等）がdocs/内に記述された場合、CIで検出してFAILとする（grepベースのスクリプト使用）。

## 1. Verifyの必須セット/推奨セット（短く運用可能な粒度）

必須セット（Fast Verify: 每PRで実行、5分以内完了）：

- ・リンク切れ検出: docs/内の全リンクをcurlでチェック。
- ・用語揺れ検出: glossary/GLOSSARY.mdとdocs/の用語をdiff比較。
- ・Part間整合: 00\_INDEX.mdのPartリストと実際ファイル存在を検証。
- ・未決事項検出: 各Partの「11. 未決事項」セクションをgrepし、項目数>0で警告（FAILではない）。
- ・危険コマンド検出: docs/内をgrepし、rm -rf/git push --force/curl | sh等を検出してFAIL。
- ・証跡不足: evidence/verify\_reports/の最新ファイル存在を確認、欠落でFAIL。  
推奨セット（Full Verify: リリース時実行、30分以内完了）：
  - ・上記必須 + 外部リンク生存確認（curl -IでHTTP 200確認）。
  - ・SBOM生成テスト: syftでリポジトリをスキャンし、出力ファイルの整合性検証。
  - ・ロールバックシミュレーション: git revert --dry-runで最新コミットをテスト。
- 1. Evidence Packの構成案（フォルダ構成+必須ファイル一覧）  
Evidence Packは evidence/packs/YYYYMMDD\_HHMMSS\_pack/ フォルダにまとめ、PR/リリースごとに生成。最小セットで再現性と監査を担保。

- ・フォルダ構成:

- ・ evidence/packs/ (ルート)
  - ・ YYYYMMDD\_HHMMSS\_pack/ (Pack単位)
    - ・ verify\_reports/ (Verifyログサブフォルダ)
    - ・ diffs/ (差分サブフォルダ)
    - ・ logs/ (実行ログサブフォルダ)
    - ・ approvals/ (承認サブフォルダ)

- ・必須ファイル一覧:

- ・ manifest.json: Pack内の全ファイルリストとsha256ハッシュ。
- ・ verify\_reports/fast\_verify.log: Fast Verifyの出力ログ。
- ・ diffs/summary.diff: git diff --summaryの要約。
- ・ approvals/approvals.json: HumanGate承認のJSON (`{"approver": "user", "timestamp": "YYYY-MM-DDTHH:MM:SSZ", "comment": "承認理由"}`)。
- ・ logs/external\_fetch.log: 外部取得 (curl/wget等) のログ、URLとレスポンスヘッダー含む。
- ・ logs/ci\_run.log: GitHub Actionsの実行ログ抜粋。

1. Release手順 (番号付き : 確定化→検証→ロールバック)

1. 確定化: mainブランチでFull Verifyを実行し、PASSを確認。git tag -s vYYYYMMDD\_HHMMSS --message="Release: <description>"で署名タグを作成 (GPGキー使用)。
  2. SBOM生成: syft dir:.. > sbom.cyclonedx.jsonでリポジトリをスキャン。
  3. スキャン: trivy fs . --exit-code 1 --vuln-type os,libraryで脆弱性スキャン、重大 (CVSS>=7.0) ゼロを確認。
  4. 不変化保証: manifest.jsonと全ファイルのsha256を生成、RELEASE/フォルダにコピーしてgit commit (READ-ONLY化は.gitattributesでlock)。
  5. 検証: git checkout <tag> && pwsh checks/verify\_repo.ps1 -Mode Fullを実行、再現確認。
  6. ロールバック (必要時) : git revert <commit>で変更取消、Full Verify再実行、Evidence Packにrollback.logを追記 (理由/影響記載)。
1. RAG更新Runbook (番号付き: トリガ→手順→証跡→検証)
    1. トリガ: docs/変更のPRマージ後、GitHub Actionsで自動起動 (webhook or workflow\_dispatch)。
    2. 手順: docs/を再インデックス (例: python scriptでMarkdownをベクトルDBに挿入)、スナップショットとしてkb\_snapshot\_YYYYMMDD\_HHMMSS.tar.gzを作成。
    3. 証跡: Evidence Packにrag\_update.logを追加 (インデックス対象ファイルリスト、処理時間、エラー抜粋)。
    4. 検証: スナップショットのsha256を計算、クエリテスト (例: "SSOTとは?"でdocs/Part00を正しく返すか) を実行、PASSでリリース。
    5. 失敗時: ロールバック (前スナップショット復元)、HumanGate通知、ADRで原因記録。
  1. Intentionally Not Covered

- ・MCP Inspectorやstdioの詳細運用（別エージェント領域）。
- ・AI並列割当の割り振りロジックやエージェント間連携の細部。
- ・クラウドストレージ（AWS S3等）を使ったEvidence外部保存ポリシー。
- ・高度なSBOMツール比較（SPDX vs CycloneDXの深掘り）。
- ・UIベースのOperation Registry実装（テキストベースのみ触れる）。

## 1. 根拠URL一覧（参照日、可能なら更新日）

- ・<https://docs.github.com/en/repositories/configuring-branches-and-merges-in-your-repository/managing-protected-branches/about-protected-branches> (参照日: 2026-01-12, 更新日: 2023-11-28)
- ・<https://docs.github.com/en/actions/using-workflows/about-workflows> (参照日: 2026-01-12, 更新日: 2023-12-15)
- ・<https://cyclonedx.org/docs/1.5/> (参照日: 2026-01-12, 更新日: 2023-10-01)
- ・<https://github.com/anchore/syft> (参照日: 2026-01-12, 更新日: 2024-01-05)
- ・<https://github.com/aquasecurity/trivy> (参照日: 2026-01-12, 更新日: 2024-01-10)
- ・<https://git-scm.com/docs/git-tag> (参照日: 2026-01-12, 更新日: 2023-09-07)
- ・<https://docs.github.com/en/authentication/managing-commit-signature-verification/signing-tags> (参照日: 2026-01-12, 更新日: 2023-11-20)
- ・<https://docs.github.com/en/actions/using-workflows/events-that-trigger-workflows> (参照日: 2026-01-12, 更新日: 2023-12-05)
- ・<https://spdx.dev/specifications/> (参照日: 2026-01-12, 更新日: 2023-05-11)