GitHubのリポジトリが増えすぎたときの整理方法

属人化・放置・重複を防ぐ管理ルール



Start Challenge Consulting co., ltd.

発行日:2026年8月 ／ 全20ページ

Confidential

はじめに：増える一方で廃止されないリポジトリ

新規作成だけが繰り返され廃止手順が定義されていないため、利用実態が不明なリポジトリが蓄積する。無計画な削除はシステム停止と履歴消失を招くため、「棚卸し・判断基準・管理ルール」の3点を整える必要がある。

事実契 ‖ 自然増加

開発プロジェクトの増加に伴い、リポジトリも自然に増えていく。
増やす判断は容易だが、やめる判断は誰も行わない。

課題契 ‖ 利用実態の不明化

「誰が使っているか分からないリポジトリ」が蓄積する。
管理責任者も利用目的も追跡できない状態になる。

リスク契 ‖ 誤った整理

無計画な削除は、システム停止や履歴消失につながる。
「増やさない」ためではなく「安全に減らす」ための手順が必要。

本資料で分かること

01　棚卸しの方法
GitHubから取得すべき情報と、一覧では判断できない追加情報の集め方（P6～P8）

02　判断基準
残す・統合する・アーカイブする・削除するの5区分と判断フロー（P9～P14）

03　再発防止の管理ルール
命名規則・作成時の標準ルール・README・定期棚卸しの仕組み化（P15～P19）

注記　事実：リポジトリの増加と利用実態の不明化／提案：棚卸し・判断基準・管理ルールの3点整備。運用は各組織の規程・契約に合わせて調整すること。

リポジトリが増えすぎる主な原因

直接原因は個別最適なリポジトリ作成にあるが、構造的原因は「作成から廃止までのライフサイクルが管理プロセスとして定義されていない」ことにある。

01 ｜ 作成起点の分散

プロジェクトごとに新規リポジトリを作成している。

共通化の検討が行われず、類似構成が並列に増える。

02 ｜ 検証用の放置

検証・PoC用リポジトリが放置されている。

短期利用の前提が記録されず、終了判断が行われない。

03 ｜ 複製・派生の並存

同じシステムのコピーや派生版が複数存在する。

どれが正なのかを判断できず、修正が分散する。

04 ｜ 命名の不統一

部署・チームごとに命名方法が異なる。

名前から対象システムと役割を判別できない。

05 ｜ 外部委託の個別作成

外部ベンダーが個別にリポジトリを作成している。

契約終了後も所管が決まらず残り続ける。

06 ｜ 廃止手順の欠如

リポジトリの廃止手順が決められていない。

01〜05を蓄積させ続ける構造的原因。

構造的原因：リポジトリの「作成 → 利用 → 終了 → 廃止」が、管理プロセスとして定義されていない。

注記　事実：01〜06の事象／解釈：01〜05は個別最適の結果であり、06が蓄積を許す構造的原因である／提案：廃止手順の定義を優先する。

放置されたリポジトリが生む問題

放置リポジトリは「使われているか分からない」状態を生み、誤参照・脆弱性・権限残存を通じて、調査工数の増加とセキュリティ・監査リスクにつながる。

開発・運用への影響

- 利用中のリポジトリを判断できない
- 古いコードを誤って参照する
- GitHub Actionsが不要な処理を継続する

セキュリティへの影響

- 脆弱な依存ライブラリやシークレットが残る
- アクセス権限が残り続ける（退職者・元ベンダーを含む）

品質・管理への影響

- CopilotなどのAIが古いコードを参照する可能性がある（仮説）
- 管理・監査の対象が増加する

業務への影響

利用可否の調査・確認に工数が発生し、
修正やリリースの手戻りが生じる。

→

経営への影響

セキュリティインシデント・監査対応のリスクが増大し、
統制状況を説明できなくなる。

注記　事実：放置リポジトリに伴う7つの事象／解釈：影響は開発・セキュリティ・監査に及ぶ／仮説：AIコーディング支援が古いコードを参照する可能性（仮）。

整理を始める前に決めること

整理は対象範囲・基準日・責任者・承認者・復元方法を先に確定させる。決めずに着手すると判断が止まり、削除の責任が曖昧になる。

決定項目	決める内容	決めずに着手した場合に起きること
対象Organization	整理対象とするOrganization／Enterpriseの範囲	対象外Organizationの棚卸しが漏れる
対象リポジトリの範囲	Public／Private／Internal、Fork・Archived済みの扱い	対象件数が担当者ごとに変動する
棚卸しの基準日	最終更新日・最終コミット日を判定する基準日	「最新」の解釈が人によって異なる
整理責任者・確認担当者	全体責任者と、リポジトリ単位で利用有無を確認する担当	確認依頼が誰にも届かない
削除・アーカイブの承認者	処理を承認する役職・会議体と、承認記録の残し方	削除の責任が不明確になる
関係者への確認期間	利用有無の回答期限と、無回答時の取り扱い	判断が無期限に保留される
バックアップと復元方法	取得範囲（コード・Issue・設定）と復元手順の検証	誤削除時に復元できない
整理完了の判定基準	全件が5区分に分類され、承認記録が揃った状態	整理作業が終わらない

8項目を「整理計画」として文書化し、承認者を含む関係者間で合意したうえで着手する。

注記　事実：決定すべき8項目／解釈：右列は各項目が未決定の場合に想定される事象／提案：整理計画として文書化し合意する。

STEP1：リポジトリ一覧を取得する

まずGitHubから機械的に取得できる基本情報を全件そろえ、CSV等の一覧表を「棚卸し台帳の初版」とする。この段階では判断は行わず、事実の収集に限定する。

取得する基本情報（全件・機械的に取得可能）

- リポジトリ名
- URL
- Public／Private／Internal
- 作成日・最終更新日
- 最終コミット日
- デフォルトブランチ
- オーナー・管理チーム
- 利用言語
- Fork・Archivedの状態
- Issues・Pull Requestsの状況

一覧の各行に「分類（P9の5区分）」「管理責任者」「判断根拠」「承認日」の空欄列をあらかじめ設けておく。
以降のSTEP2・STEP3で埋めることを前提とし、同じ台帳を最後まで使い続ける。

取得方法と使い分け

GitHubの管理画面
件数が少ない場合の目視確認に適する

GitHub CLI
一覧の一括出力に適する（例：gh repo list）

REST API／GraphQL API
項目を指定した取得・定期自動化に適する

CSV・スプレッドシートへの出力
関係者への確認依頼と記録保存に適する

**STEP1の完了条件：対象Organizationの全リポジトリが1つの台帳に登録され、基本情報の欠損がない状態。
この時点で「削除候補」等の判断は行わない。**

注記　事実：GitHubで取得可能な基本情報項目と取得手段／提案：台帳の初版を作成し、判断列を空欄で用意する。コマンド例は各環境のバージョンに合わせて確認すること。

STEP2：追加情報を収集する

最終更新日が古いことは「不要」の根拠にならない。本番利用・依存関係・保存義務という3つの追加情報がそろって初めて、残す／やめるの判断が可能になる。

リポジトリー覧（STEP1）だけでは判断できない情報契――契以下を確認先ごとに収集する

観点	収集する追加情報	主な確認先
本番利用	本番環境で利用されているか／デプロイ先のシステム	運用部門、システム構成管理台帳
自動処理	GitHub Actionsの稼働状況（定期実行・デプロイの有無）	Actionsの実行履歴、ワークフロー定義
依存関係	他リポジトリからの参照関係／パッケージ・コンテナの公開状況	依存定義ファイル、Packages、社内検索
外部連携	Webhook・GitHub Appsとの連携／シークレット・環境変数の利用	リポジトリ設定、連携先サービス管理者
契約・法令	保守契約や法令上の保存義務の有無と保存期間	契約管理部門、法務、内部監査
関係者	関係する顧客・外部ベンダー（アクセス権を持つ社外関係者）	営業・調達部門、Collaborators一覧

「更新がない」＝「使われていない」ではない。本番参照・自動処理・保存義務のいずれかがあれば、更新がなくても残す対象となる。

注記　事実：GitHub上の設定・履歴から確認できる項目／提案：確認先を明記した依頼票として関係部門に照会し、回答を台帳に記録する。

STEP3：管理責任者を特定する

管理責任者が不在のリポジトリは、利用有無を確認できないため整理が停止する。GitHub上の情報から段階的に辿り、それでも不明な場合は一時的な管理部署を設定して判断可能な状態にする。

特定の手順 —— 上から順に確認し、判明した時点で台帳に記録する

01 ｜ 設定情報

CODEOWNERSを確認する。

レビュー責任者が明示されている場合、最も確度が高い。

→

02 ｜ 権限情報

管理チーム・Collaboratorsを確認する。

Admin権限を持つチームを所管候補とする。

→

03 ｜ 履歴情報

READMEやコミット履歴から担当者を探す。

直近のコミッター・Issue対応者を辿る。

→

04 ｜ 関係者への照会

デプロイ先のシステム管理者に確認する。元プロジェクト責任者へ確認する。

GitHub上に情報が残っていない場合の最終手段。

担当者が退職・異動している場合

- ■ 所属していた部署へ引き継ぎ先を照会する
- ■ 後任者を管理責任者として台帳に登録する
- ■ 個人アカウント所有のリポジトリはOrganizationへ移管する
- ■ 退職者に付与された権限はこの段階で削除する

オーナー不明リポジトリの扱い（提案）

一時的な管理部署（例：情報システム部門）を暫定オーナーとして設定する。

「オーナー不明のまま放置」を許さず、必ず誰かが判断できる状態にすることが目的であり、恒久的な所管の確定ではない。

STEP3の完了条件：全リポジトリに管理責任者（暫定を含む）が設定され、確認依頼の宛先が確定している状態。

注記　事実：CODEOWNERS・権限・履歴から責任者を辿る手順／提案：オーナー不明時は暫定管理部署を設定し、判断の停止を防ぐ。

リポジトリを分類する5つの区分

全リポジトリを「継続利用・保守のみ・統合候補・アーカイブ候補・削除候補」の5区分に必ず割り当てる。分類は削除の可否ではなく、以降の対応方法を決めるための区分である。

分類	状態	対応
継続利用	現在も開発・運用中	維持・管理情報を更新
保守のみ	新規開発はないが本番利用中	保守用として残す
統合候補	重複・類似リポジトリが存在	統合を検討
アーカイブ候補	利用終了だが履歴を残したい	Read-only化
削除候補	不要で保存義務もない	承認後に削除

分類の原則

「未分類」を残さない。判断材料が不足する場合は
アーカイブ候補として扱い、期限を定めて再判定する。

台帳への記録

分類・判断根拠・管理責任者・承認者・処理予定日を
台帳に記録し、監査時に説明できる状態にする。

注記　事実：5区分の定義と対応方針／提案：全件を5区分のいずれかに割り当て、判断根拠を台帳に記録する。各区分の詳細な判断基準はP10～P13に記載。

残すべきリポジトリの判断基準

「利用の事実」「依存の事実」「保存の義務」のいずれかに該当すれば残す。ただし残す場合も、管理責任者と利用目的を明確にできることを条件とする。

A ｜ 利用の事実

- ■ 本番システムで使用している
- ■ 現在もコミットやPull Requestがある
- ■ 再利用する可能性が具体的に決まっている
  （時期・用途が特定できる場合に限る）

B ｜ 依存の事実

- ■ 他のシステムから参照されている
- ■ パッケージ・ライブラリの配布元になっている
  （削除・改名すると参照側のビルドが停止する）

C ｜ 保存の義務

- ■ 障害調査や監査で履歴が必要
- ■ 顧客との契約上、保存が必要
  （保存期間を台帳に記録し、期限後に再判定する）

共通条件 —— A〜Cに該当する場合も、以下を満たせない限り「継続利用」とは扱わない

条件1 ｜ 管理責任者を明確にできる

CODEOWNERSと管理部署が設定され、
次回棚卸し時の確認先が確定している。

条件2 ｜ 利用目的を明確にできる

対象システムと役割がREADMEに記載され、
第三者が用途を判別できる。

注記　事実：残す根拠となる8項目／解釈：根拠を「利用・依存・保存義務」の3類型に整理／提案：管理責任者と利用目的の明確化を残す条件とする。

統合を検討すべきリポジトリ

統合は「同じ変更を複数箇所に適用している」状態の解消を目的とする。分割の目的が説明できる場合は統合せず、履歴を保持できる方法で実施する。

統合候補となる状態

- ■ 同じコードを複製したリポジトリ（-copy／-old／-bak など）
- ■ 環境別に分けられたリポジトリ（dev／stg／prod で内容がほぼ同一）
- ■ フロントエンド・バックエンドが過度に細分化されている
- ■ 共通ライブラリが複数存在する
- ■ PoCから本番版へ移行済み（PoC側が残っている）
- ■ 顧客別の差分が少ない派生リポジトリ

統合方法の選択（判断基準）

モノレポ化　リリース単位・管理責任者が同一で、同時に変更されることが多い場合。

共通化（ライブラリ化）　複数システムが同じコードを重複保持している場合。配布元を一本化する。

テンプレート化　構成は同じだが顧客別・案件別に独立管理が必要な場合。Repository Templateで統一する。

統合時に履歴を残す方法（統合前に必ず決定する）

- ■ 統合先へコミット履歴を含めて取り込む
- ■ 統合元はアーカイブし、READMEに統合先を明記する
- ■ Issue・Pull Requestの記録は移行できない前提で保存する
- ■ 参照元の依存定義・CI設定を統合先へ切り替える

注記　事実：統合候補となる6つの状態／提案：モノレポ化・共通化・テンプレート化を状況に応じて選択し、統合前に履歴の扱いを決定する。

アーカイブするリポジトリの判断基準

アーカイブは「履歴を残しつつ変更を止める」措置であり、削除の前段として最も安全な選択肢である。Read-only化だけで終わらせず、後処理と保存期限の設定まで行う。

アーカイブ対象とする条件

- ■ 開発・運用が終了している
- ■ コードやIssueの履歴を残す必要がある
- ■ 将来、障害調査で参照する可能性がある
- ■ 契約・監査上の保存対象である
- ■ すぐに削除する判断ができない

判断材料が不足する場合の既定の扱いとする

アーカイブ後に確認する項目

01 READMEへ終了理由を記載する

02 後継リポジトリへのリンクを記載する

03 GitHub Actionsを停止する

定期実行・デプロイの残存を確認する

04 Webhook・Deploy Keysを解除する

05 保存期限を設定し台帳に記録する

アーカイブの位置づけ

Read-only化により、誤参照・誤変更を防ぎながら履歴を保持できる。
権限・自動処理を残したままのアーカイブはリスクが残る。

保存期限を設定しなければ、アーカイブ済みリポジトリが新たな放置対象となる。期限到来時にP13の基準で削除を再判定する。

注記 事実:アーカイブ(Read-only化)で変更が停止する／提案:終了理由・後継リンク・自動処理停止・保存期限の5点を後処理として定型化する。

削除するリポジトリの判断基準と注意点

削除は復元が保証されない不可逆な処理である。削除対象は「明確に不要と言えるもの」に限定し、5項目の事前確認と承認記録をもって実施する。

削除候補（限定列挙）

- ■ 一時的なテスト用リポジトリ
- ■ 内容のないリポジトリ（コミットがない・READMEのみ）
- ■ 完全な重複リポジトリ（正本が特定できている）
- ■ 誤って作成されたリポジトリ
- ■ 保存期限を過ぎたアーカイブ

削除前の確認（5項目すべて必須）

- □ 本番環境から利用されていないか
- □ 他リポジトリから参照されていないか
- □ GitHub ActionsやWebhookが動いていないか
- □ 必要な履歴をバックアップしたか

復元手順の検証まで完了していること

- □ 管理責任者と承認者が合意したか

削除の不可逆性に関する留意点

削除後の復元可否・復元可能期間はプランや条件により異なる。
復元できることを前提に判断してはならず、
必ず事前バックアップを取得する。

判断に迷う場合の既定動作

即時削除せず、一定期間アーカイブしてから再判定する。
削除は「保存期限を過ぎたアーカイブ」を経由するのが原則であり、
棚卸し中の直接削除は例外扱いとする。

注記　事実：削除は不可逆な処理であり、復元条件はプラン・期間により異なる（自組織の契約条件を確認すること）／提案：削除候補を限定列挙し、5項目の確認と承認記録を必須とする。

リポジトリ整理の判断フロー

利用 → 依存 → 保存義務 → 重複 → 責任者の順に確認すれば、担当者が変わっても同じ結論に到達する。判断に迷う場合は削除せず、一定期間アーカイブする。

Q1　現在利用されているか（コミット・PR・本番稼働）	YES → **継続利用** ／ 開発終了なら Q2 へ
Q2　本番環境や外部システムから参照されているか	YES → **保守のみ** ／ NO なら Q3 へ
Q3　保存義務・監査上の必要性があるか	YES → **アーカイブ候補**（期限設定）
Q4　類似・重複リポジトリがあるか	YES → **統合候補**（正本を特定）
Q5　管理責任者を特定できるか	NO → 暫定管理部署を設定（P8）

Q6　継続・保守・統合・アーカイブ・削除を決定する
上記いずれにも該当しない場合のみ、削除候補として扱う

Q7　承認記録を残して処理する
判断根拠・承認者・処理日を台帳に記録し、監査時に説明できる状態にする

判断に迷う場合の原則

すぐに削除せず、一定期間アーカイブする。
期限到来時にQ1から再判定する。

注記　提案：Q1〜Q7の順序で判定することで担当者による判断のばらつきを抑える／解釈：削除は他の区分に該当しない場合の残余カテゴリとして扱う。

整理後の命名規則を統一する

命名規則の目的は、リポジトリ名だけで「どのシステムの何か」を判別できる状態をつくることにある。名前から用途が読めなければ、次の棚卸しで再び調査が必要になる。

命名項目の例

- ■ システム名
- ■ サービス名
- ■ 用途
- ■ 種別（frontend／api 等）
- ■ 環境
- ■ 顧客名
- ■ バージョン

命名例（推奨）

customer-portal-frontend
顧客ポータルの画面側

customer-portal-api
同システムのAPI側

data-platform-infrastructure
データ基盤のインフラ定義

internal-ai-poc
社内向けAI検証（終了予定日を登録）

避けたい名前

test
対象システムが不明

new-project
時間経過で意味を失う

sample2
連番が正本を示さない

final-version
最終版の判断ができない

temp-repository
一時利用のまま残り続ける

規則の設計方針

- ■ 項目の並び順と区切り文字を固定する
- ■ 小文字・ハイフン区切りに統一する
- ■ 略語は用語集で定義し、担当者ごとの表記揺れを防ぐ

既存リポジトリの改名に関する留意点

改名はURL・依存定義・CI設定・Webhookに影響する。
整理と同時に一括改名せず、継続利用・保守のみに絞り、
参照元の切り替えとあわせて計画的に実施する。

注記　提案：命名項目・命名例・避けたい名前を規程として明文化する／解釈：改名は参照関係に影響するため、対象と時期を限定して実施する。

リポジトリ作成時の標準ルール

乱立の再発は、作成時点で「誰が・何のために・いつまで」を記録させることで防ぐ。整理作業ではなく、作成ルールの整備が恒久対策となる。

区分	標準ルール	防止できる事象
作成権限	作成できるユーザー・チームを制限する	申請外リポジトリの発生（P3-01・05）
申請	作成申請に目的と管理責任者を記載する／終了予定日・棚卸し予定日を登録する	オーナー不明化、PoCの放置（P3-02）
構成の統一	命名規則を適用する／Repository Templateを利用する／READMEの必須項目を定める	命名の不統一、用途不明（P3-04）
責任と保護	CODEOWNERSを設定する／ブランチ保護ルールを設定する	属人化、正本の不明化（P3-03）
セキュリティ	Visibilityの初期値をPrivateにする／シークレットスキャン等のセキュリティ設定を有効化する	意図しない公開、シークレット残存（P4）

適用の考え方

個別の運用に委ねず、Organization設定・テンプレート・ルールセットとして機械的に強制する。

終了予定日の登録が要点

作成時に終了予定日を登録すれば、放置は「期限超過」として検知でき、次の棚卸しの起点となる。

注記　提案：10項目の標準ルールを作成時点の必須要件とする／解釈：右列はP3・P4で整理した原因・問題との対応関係。設定名称は自組織のプラン・機能提供状況に合わせて確認すること。

READMEに記載すべき管理情報

READMEを開発者向け説明書ではなく「管理情報の記載場所」として定義する。次回の棚卸しで調査せずに判断できる状態を、リポジトリ自身に保持させる。

記載項目	記載内容
リポジトリの目的	何のために存在するか（1〜2文）
対象システム・業務	どのシステム／業務に紐づくか
管理部署	所管する組織単位
管理責任者	役職・氏名（個人依存を避け役割で併記）
開発・保守チーム	担当チームとCODEOWNERSとの対応
本番環境の有無	本番利用の有無と対象環境
デプロイ方法	デプロイ手段と実行トリガー
関連リポジトリ	参照元・参照先、後継リポジトリ
外部サービス連携	Webhook・GitHub Apps・SaaS連携
問い合わせ先	個人ではなくチーム窓口を記載
最終確認日	直近の棚卸しで内容を確認した日付
終了時の対応方法	システム終了時の廃止手順・引き継ぎ先

管理用READMEの記載例

管理情報

目的：顧客ポータルの画面側を管理する
対象システム：顧客ポータル（本番稼働）
管理部署：デジタル推進部
管理責任者：同部 サービス基盤課長
開発・保守：ポータル開発チーム
本番環境：あり
デプロイ：GitHub Actions（mainブランチ）
関連リポジトリ：customer-portal-api
外部連携：監視SaaS（Webhook）
問い合わせ先：portal-dev（チーム窓口）
最終確認日：2026-04-01
終了時対応：廃止申請後にアーカイブ

運用のポイント

- ■ Repository Templateに雛形として組み込む
- ■ 最終確認日の更新を棚卸しの完了条件とする
- ■ 記載責任は管理責任者に置く

注記　提案：READMEに管理情報12項目を必須記載とする／記載例中の部署名・日付・システム名はすべて記載形式を示すためのサンプル値であり、実在の情報ではない。

定期棚卸しを仕組み化する

一度の整理では再発を防げない。抽出・通知・照合を自動化し、棚卸しを担当者の努力ではなく定期プロセスとして回すことで、放置リポジトリの発生を継続的に抑制する。

棚卸しサイクルの構成要素

周期
半年または1年ごとに棚卸しする

抽出（自動）
長期間更新されていないリポジトリを自動抽出する

通知（自動）
オーナー不明のリポジトリを通知する

依頼（自動）
GitHub Actionsで棚卸しIssueを自動作成する

照合
管理台帳とGitHubの情報を照合する

期限確認
アーカイブ後の保存期限を確認する

権限見直し　退職者や異動者の権限を見直す　／　**記録**　棚卸し結果を監査記録として保存する

確認する指標例（毎回の定点観測）

全リポジトリ数	件
オーナー設定率	100%を目標
1年以上更新のないリポジトリ数	件
アーカイブ数	件
削除候補数	件
命名規則への準拠率	%

仕組み化にあたり決定すべき事項

- ■ 棚卸しの周期と実施時期（例：年次・上期末）
- ■ 全体の実施責任部署と、リポジトリ単位の回答義務者
- ■ 自動抽出の基準（未更新期間のしきい値）と通知先
- ■ 指標の報告先（IT統制・内部監査への報告経路）

注記　提案：抽出・通知・依頼を自動化し、棚卸しを定期プロセス化する／指標欄の数値は各組織で実測して記入する（本資料に基準値の記載はない）。

Conclusion

まとめ・チェックリスト

リポジトリ整理は、単純な削除作業ではない。管理責任者・利用状況・依存関係を可視化し、作成から廃止までを管理する仕組みを整えることが目的である。

A ｜ 棚卸しフェーズ（P5〜P8）

- ☐ リポジトリー覧を取得した
- ☐ 利用状況と依存関係を確認した
- ☐ 管理責任者を特定した

B ｜ 判断・処理フェーズ（P9〜P14）

- ☐ 継続・保守・統合・アーカイブ・削除に分類した
- ☐ 削除前のバックアップと承認を実施した

C ｜ 管理ルール整備フェーズ（P15〜P18）

- ☐ 命名規則を統一した
- ☐ READMEとCODEOWNERSを整備した
- ☐ 新規作成ルールを設定した
- ☐ 定期棚卸しの担当者と実施時期を決めた

A・Bのみを実施した場合、リポジトリは再び増加する。
整理の成果を維持できるかはCの整備状況によって決まるため、
整理作業と同時にルール整備へ着手する。

結論

リポジトリ整理の目的は削除ではなく、管理責任者・利用状況・依存関係の可視化と、作成から廃止までを管理する仕組みの確立にある。

注記　提案：本チェックリストを整理完了の判定基準（P5）として用い、次回棚卸し時に同じ項目で確認する。

Contact

お問い合わせ

リポジトリの棚卸し設計、判断基準の策定、管理ルールの整備について、貴社の環境と体制に合わせたご相談を承ります。

お問い合わせ先

Start Challenge Consulting co., ltd.
挑む力が、世界を照らす

Start Challenge Consulting

担当　平出

E-mail　info@startchg.com

お問い合わせの際は、対象Organizationの規模（概算リポジトリ数）と、
現時点で把握されている課題をあわせてお知らせいただけますと、
初回のご相談から具体的な進め方をご提示できます。

ご相談いただける内容

01　棚卸しの実施支援
リポジトリー覧の取得、追加情報の収集、管理責任者の特定（P6～P8）

02　判断基準の策定
5区分の定義と判断フローの整備、承認プロセスの設計（P9～P14）

03　管理ルールの整備
命名規則・作成申請・READMEの標準化（P15～P17）

04　定期棚卸しの仕組み化
自動抽出・通知の設定、監査記録の運用設計（P18）

Start Challenge Consulting co., ltd.　／　挑む力が、世界を照らす

本資料の記載内容は作成時点の情報に基づきます。GitHubの機能名称・仕様は変更される場合があるため、実施前に自組織のプラン・契約条件をご確認ください。