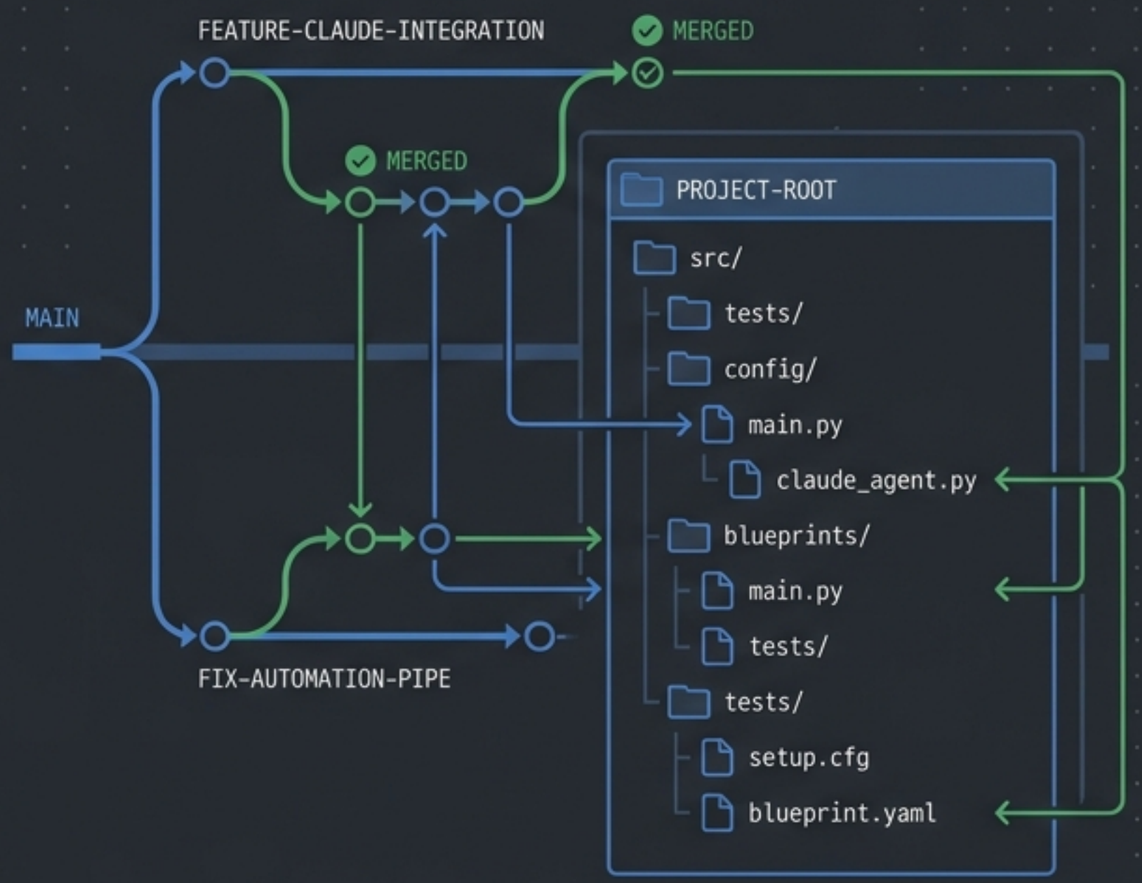
Python × Claude Code: 自動化されたプロジェクト・ブループリント

AIネイティブな開発ワークフローを最速で構築・運用・拡張するためのリファレンスガイド

Version 1.0.0



開発者のための3つの約束

瞬時のセットアップ

テンプレートからの作成後、GitHub Actionsが約１分で自動初期化。設定ファイルを書き換え、すぐに開発可能な状態を構築します。

AIネイティブなワークフロー

プロジェクトルールやセッション記憶を `CLAUDE.md` を通じて統合管理。Claude Codeが構造を理解した上でコーディングを支援します。

組み込み済みの品質保証

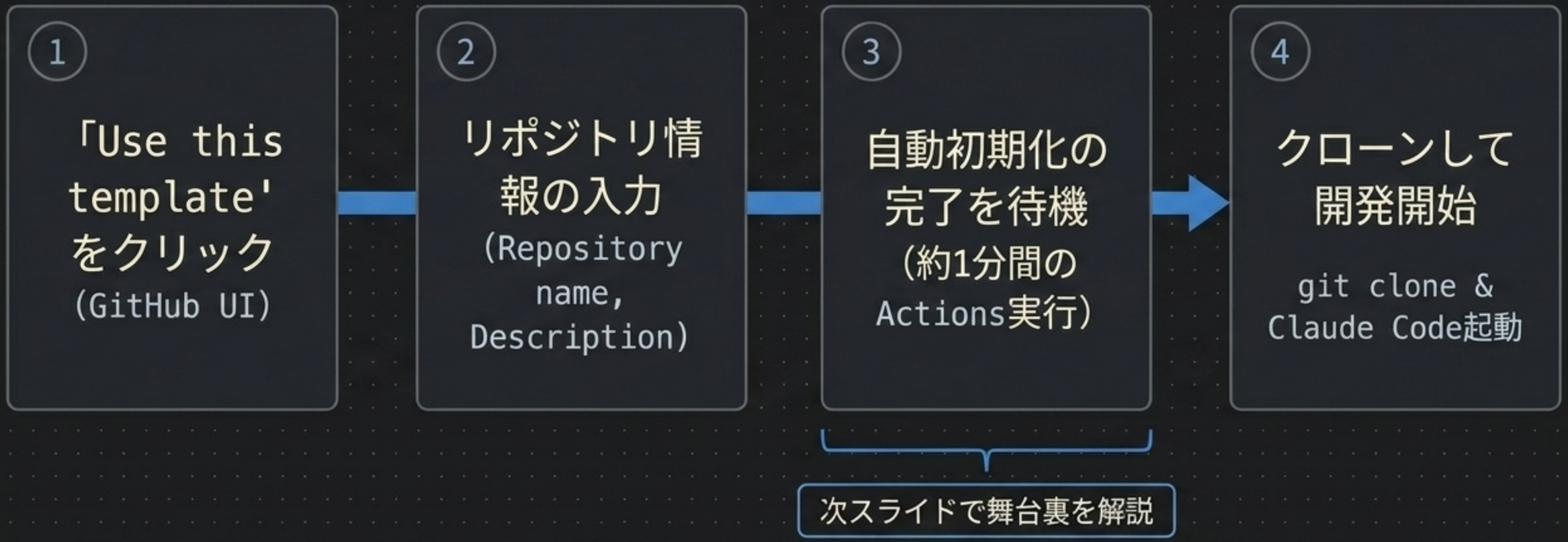
`ruff`、`mypy`、`pytest` による CI/CDと `pre-commit` が標準装備。コードの品質と堅牢性を自動で担保します。

環境の前提条件マトリックス

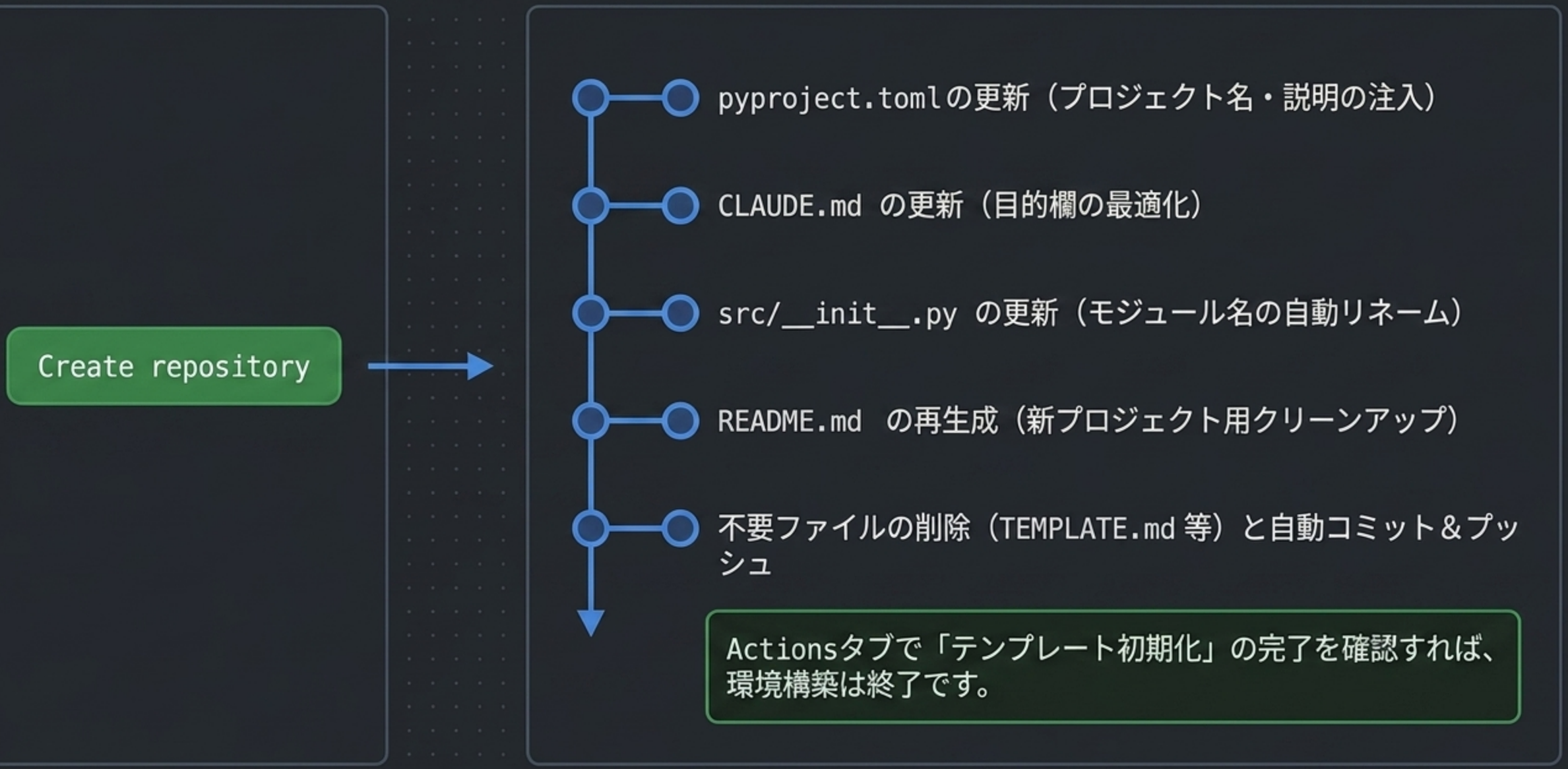
Python	3.12 以上
Git	最新版推奨
Claude Code	CLI インストール済み

これらが揃っていれば、ゼロからAI駆動開発への移行準備は完了です。

プロジェクト始動の4ステップ



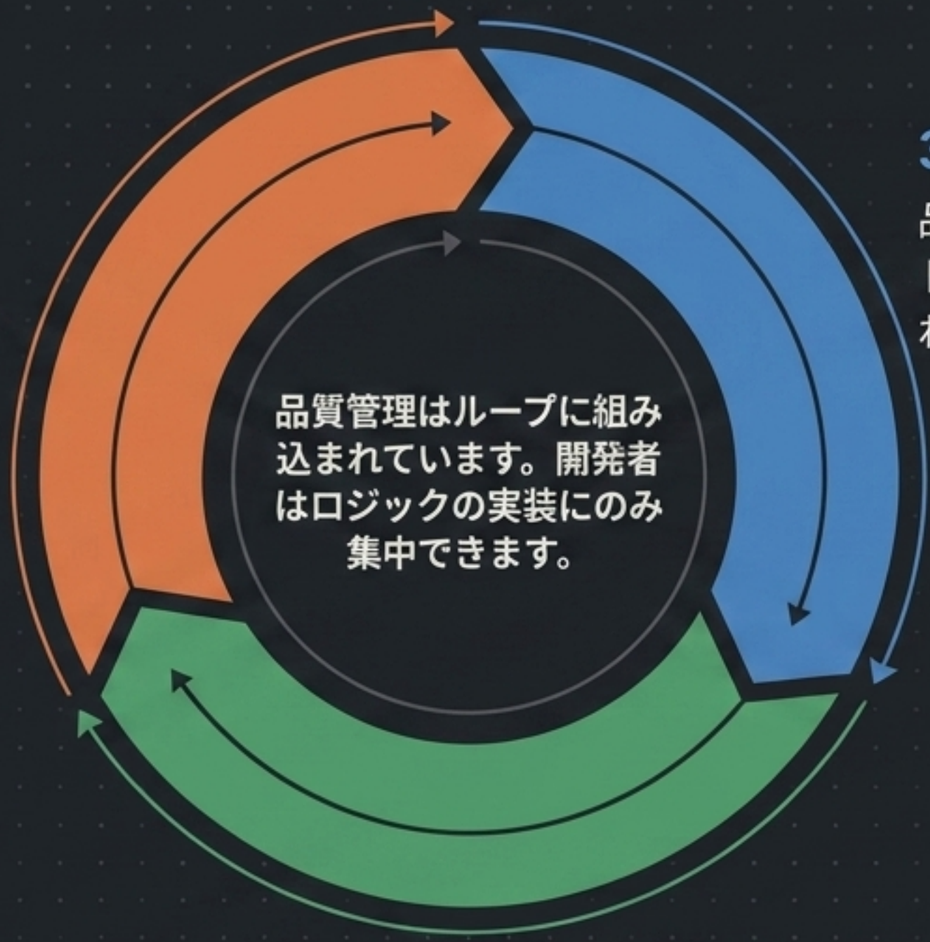
舞台裏の自動化：「魔法の1分間」



デイリー開発ループ

1. コーディング

Claude Codeが CLAUDE.md を読み込み、プロジェクト構造を理解して開発を支援。



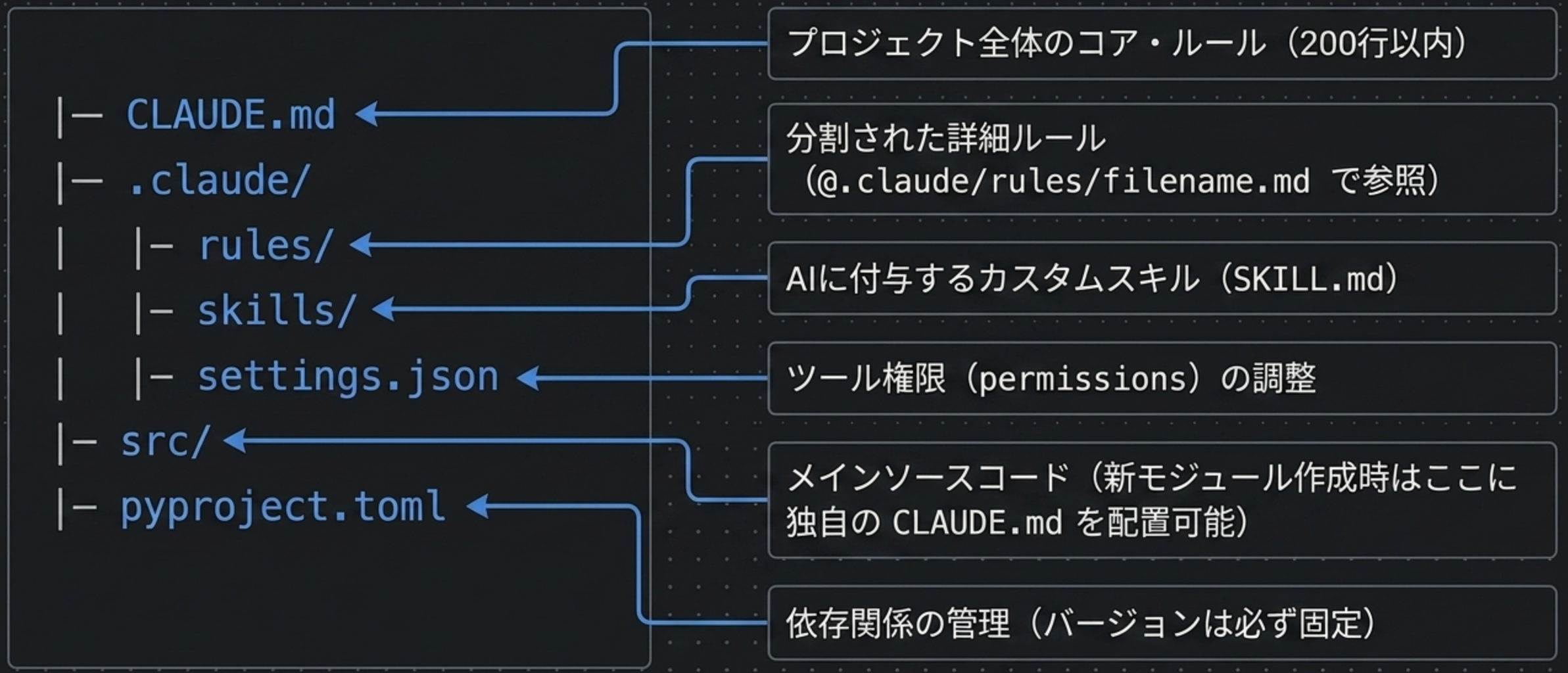
3. コミット完了

品質基準をクリアしたコードのみがリポジトリに保存される。

2. コミット前チェック

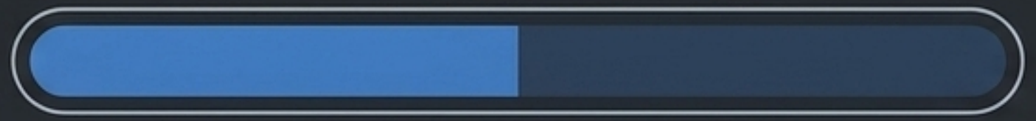
git commit 実行時、pre-commit により ruff（リント・フォーマット）が自動起動。

AI設定アーキテクチャ・マップ



ルール・エンジニアリング：CLAUDE.mdの最適化

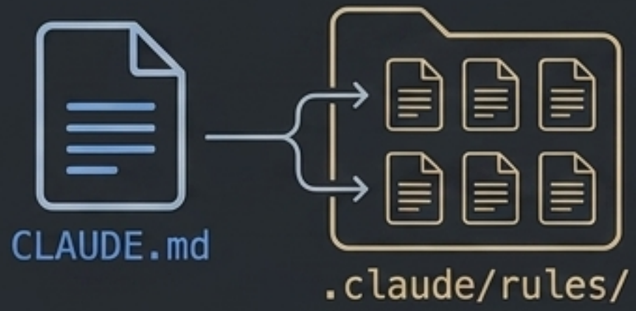
Core Constraints



ルール: プロジェクト固有のルールは **200行以内** に維持。

理由: Claudeのコンテキストを圧迫せず、最も重要な指示に集中させるため。

Delegation Pattern



200行を超える詳細な仕様やニッチなルールは `.claude/rules/` にファイルを分割。

```
@.claude/rules/filename.md
```

必要な場面でファイルパスを使用して動的に参照。

プロジェクト機能とAI能力の拡張

依存関係の追加 (Python)

pyproject.toml

ルール: パッケージのバージョンは必ず固定すること。再現性を担保するため。

スキルと権限の追加 (AI)

.claude/skills/SKILL.md
（カスタムスキルの定義）

.claude/settings.json
（ツール権限 permissions の編集とチューニング）

新規モジュールを追加した場合、そのディレクトリ内に専用の CLAUDE.md を配置することで、Claudeがモジュール固有のルールを学習します。

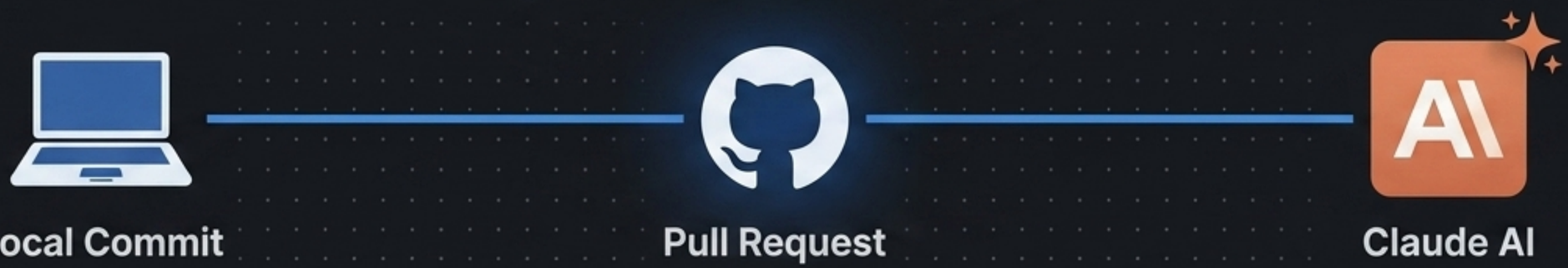
CI/CD 品質保証マトリックス

PushおよびPR時に自動実行される堅牢なテストパイプライン

カテゴリ	ツール	コマンド（自動実行）	目的
リント (Linting)	ruff	`ruff check src/ tests/`	コード品質の検証と静的解析
フォーマット (Formatting)	ruff	`ruff format --check src/ tests/`	一貫したコードスタイルの維持
型チェック (Type Checking)	mypy	`mypy src/`	静的型付けによるバグの未然防止
テスト (Testing)	pytest	`pytest --cov=src`	テスト実行およびカバレッジ測定

GitHub Actions × Claude 統合 (任意設定)

GitHubのIssueやPRで **@claude** とメンションするだけで、
AIが自動でコードレビューや対応を実行。



1. GitHubリポジトリの Settings → Secrets and variables → Actions へアクセス

リポジトリ設定からアクションシークレットに移動します。

2. New repository secret で `ANTHROPIC_API_KEY` を登録

Anthropic APIキーをシークレットとして保存します。

3. `.github/workflows/claude.yml` が自動的にアクティブ化

設定が完了すると、ワークフローが自動的に有効になります。

セッション継続性システム（長期記憶モデル）

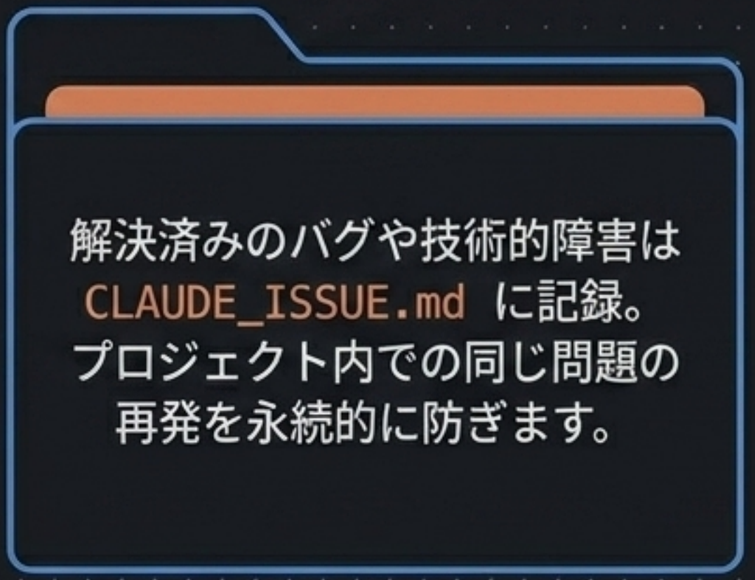
中断時 (Pause)

現在の進捗と残りのTODOを自動で `CLAUDE_PROGRESS.md` に書き出し。

再開時 (Resume)

起動時に `CLAUDE_PROGRESS.md` を読み込み、前回のコンテキストを完全に復元。

エラーの記憶 (Error Memory)



解決済みのバグや技術的障害は `CLAUDE_ISSUE.md` に記録。プロジェクト内での同じ問題の再発を永続的に防ぎます。

トラブルシューティング・ダッシュボード

テンプレート初期化が実行されない

原因: GitHub Actionsが無効、または権限不足。

対処: Settings → Actions → General で「Allow all actions」を有効化し、Actionsタブから手動再実行（Re-run）。

pre-commit が動かない

対処: `pre-commit install` が実行されているか確認。

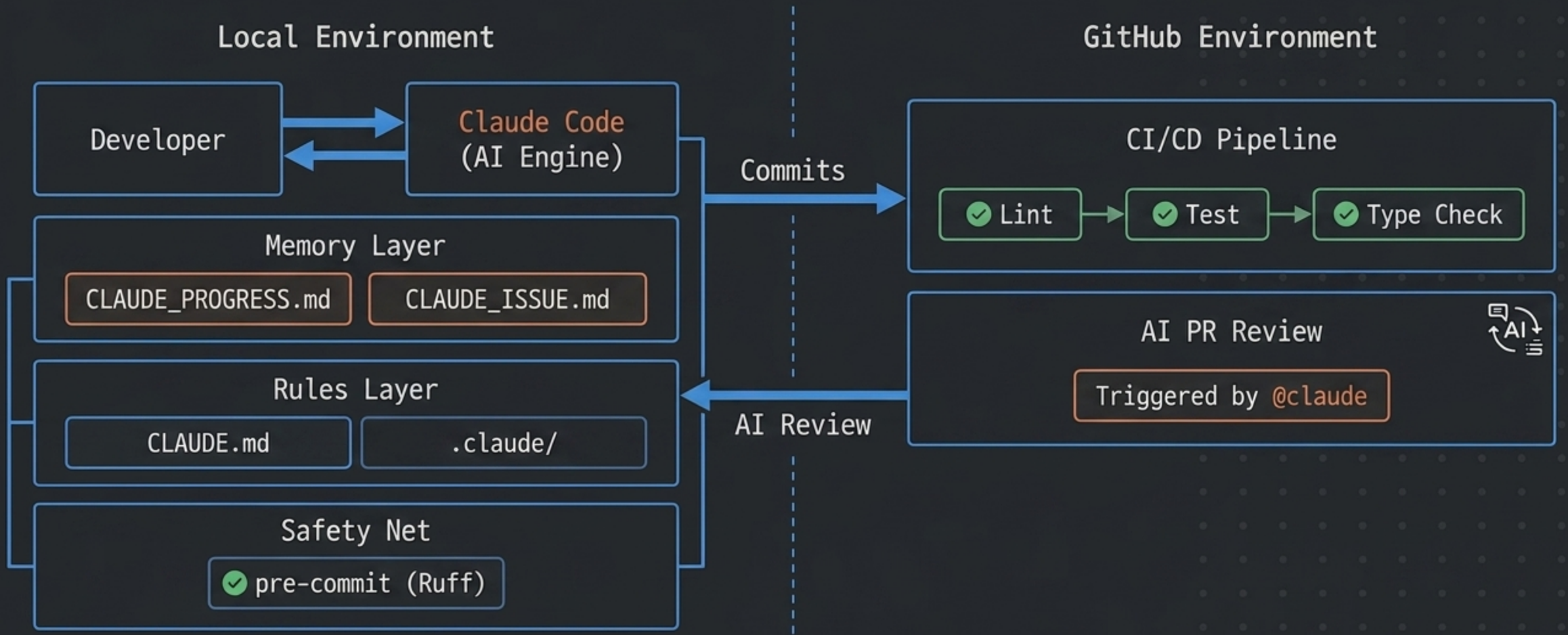
mypy でエラーが出る

対処: 型定義の不足を確認。外部ライブラリの場合は stub の追加を検討。

仮想環境が見つからない

対処: 依存関係のインストールプロセスとパス設定を再確認。

統合エコシステム：完全自動化開発の全体像



単なるコード生成ツールではなく、自己修復とルール遵守を組み込んだ『自律型開発プラットフォーム』として機能します。

次のステップへ

Use this template

テンプレートリポジトリへアクセスし、上のボタンをクリックしてプロジェクトを初期化してください。

あなた専用のAIネイティブな開発環境が、わずか1分で構築されます。

https://github.com/organization/python-claude-template