



# Copilot Agent ハンズオン

Copilot Agents で生産性を飛躍的に向上

Coding Agent & Code Review Agent





## Agenda

- ✓ ハンズオンの概要と目標
- ✓ セットアップ要件
- ✓ GitHub Copilot Code Review Agent の紹介
- ✓ 実習演習（5つの演習）
- ✓ まとめと次のステップ





# ハンズオンの概要と目標

GitHub Copilot Code Review Agent を活用してコードレビューを自動化し、開発ワークフローの効率を向上させることを学びます。

## 達成すること:

- ✓ 自動コードレビューワークフローのセットアップ
- ✓ 品質改善プロセスの実装
- ✓ バグ検出・解決システムの設定

**前提条件:** GitHub Copilot for Business ライセンスが有効化されていること

! [プレースホルダー: Code Review Agent ダッシュボードインターフェース]



# Copilot Coding Agent の利用

サンプルリポジトリをテンプレートとして利用して、ハンズオンを進めます。

- ✓ リポジトリにアクセスします: <https://github.com/yuhattor/terraform-copilot-sample>
- ✓ Use this template ボタンをクリックして、自分のアカウントにコピーします

The screenshot shows the GitHub repository page for `yuhattor / terraform-copilot-sample`. The repository is a public template generated from `yuhattor/copilot-terraform-sample`. It features a navigation bar with tabs for Code, Issues, Pull requests, Actions, Projects, Security, Insights, and Settings. The repository has 0 stars, 0 forks, and 0 watches. A green button labeled "Use this template" is prominently displayed. Below the repository name, there is a section for branches and tags, showing the `main` branch with 1 branch and 0 tags. A search bar and an "Add file" button are also visible. The commit history shows an initial commit by `yuhattor` 13 minutes ago, with a list of files: `DevOps`, `ExampleAzureFunctions`, `Images`, `docs`, and `README.md`. On the right side, there is an "About" section with a description: "No description, website, or topics provided." and a list of links: Readme, Activity, 0 stars, 0 watching, and 0 forks.

yuhattor / terraform-copilot-sample

Public template

generated from [yuhattor/copilot-terraform-sample](#)

Pin Watch 0 Fork 0 Star 0 Use this template

main 1 Branch 0 Tags

Go to file Add file Code

yuhattor Initial commit 9750614 · 13 minutes ago 1 Commit

|                       |                |                |
|-----------------------|----------------|----------------|
| DevOps                | Initial commit | 13 minutes ago |
| ExampleAzureFunctions | Initial commit | 13 minutes ago |
| Images                | Initial commit | 13 minutes ago |
| docs                  | Initial commit | 13 minutes ago |
| README.md             | Initial commit | 13 minutes ago |

About

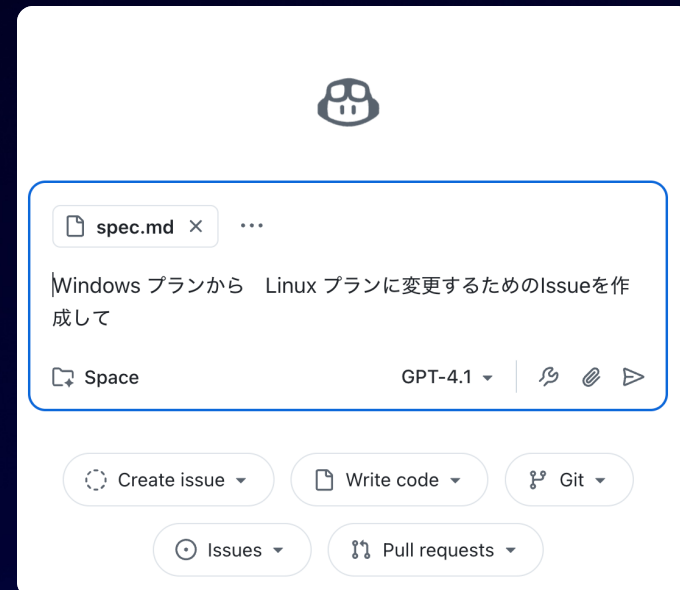
No description, website, or topics provided.

Readme Activity 0 stars 0 watching 0 forks



# Copilot Coding Agent で実装をします

- ✓ Windows プランから Linux プランにコードを編集してみましょう
- ✓ `docs/spec.md` にある Spec ドキュメントを確認して、Issue を作成します
- ✓ コードViewにある GitHub Copilot アイコンをクリックして、Copilot Chat を起動します
- ✓ Copilot と一緒に Issue を作成しましょう Windows プランから Linux プランに変更するための Issueを作成して と Copilot に問いかけましょう
- ✓ Issue を作成したら、GitHub Copilot にアサインします





# Copilot Code Review Agent とは？

## 紹介

GitHub Copilot があなたのコードを解析し、即座に適用可能なアクションブルな提案とともに、インテリジェントなフィードバックを提供します。

## 主要機能

- ✓ 自動セキュリティ脆弱性検出
- ✓ コード品質と保守性の分析
- ✓ パフォーマンス最適化の推奨
- ✓ ベストプラクティスの強制

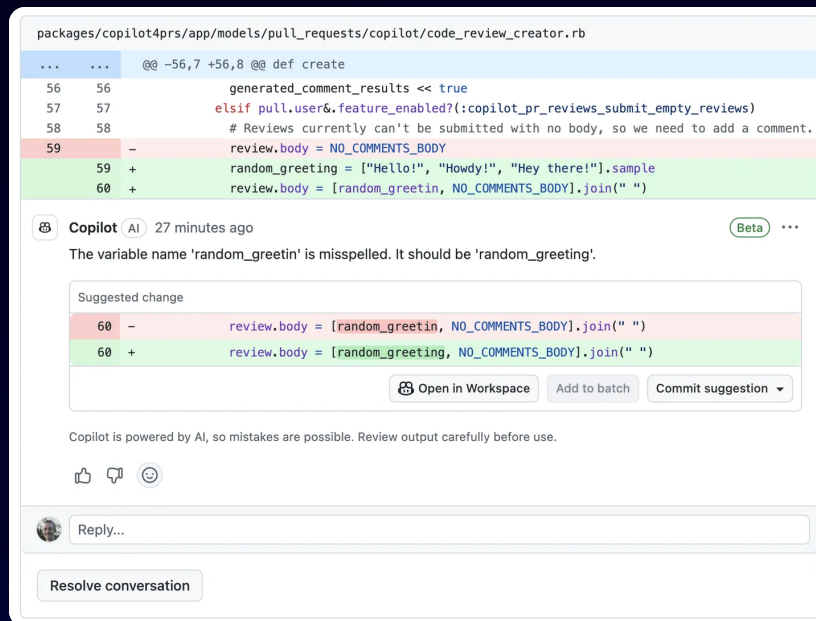
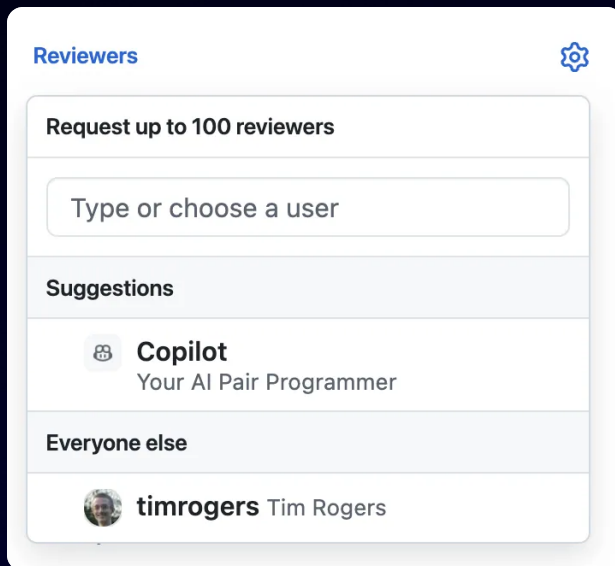
**統合:** GitHub のネイティブ pull request ワークフローとシームレスに連動



# Copilot Code Review の使用方法

## 手動レビュープロセス

1. GitHub.com で pull request を作成または既存のものに移動
2. Reviewers メニューを開き、Copilot を選択
3. 分析完了を待機（通常30秒以内）
4. Copilot の詳細なコメントと提案をレビュー





# Copilot Code Review の使用方法

## レビューの動作

- ✓ "Comment" レビューを提供 ("Approve" や "Request changes" ではない)
- ✓ マージをブロックしたり、必要承認数にカウントされません
- ✓ コメントは人間のレビューアーフィードバックと同様に機能



# Copilot への再レビューリクエスト

## 再レビューが必要な場合

以前にレビューされた pull request に変更をプッシュした後、Copilot は自動的に再分析しません。

## プロセス

1. Reviewers メニューの Copilot 名の隣にあるリフレッシュボタンをクリック
2. 更新された分析を待機
3. 新しいフィードバックを以前の提案と比較



**ベストプラクティス:** 主要なコード変更やセキュリティ修正に対応した後に再レビューをリクエスト



# 演習 1: Fork と手動レビュー

**目標:** サンプルリポジトリで手動 Copilot コードレビューを練習

## 手順

1. コードを含むサンプルリポジトリを作成 または フォーク (<https://github.com/yuhattor/terraform-copilot-sample>)
2. 意図的なコード問題を含む新しいブランチを作成
3. pull request を開き、手動で Copilot レビューをリクエスト
4. フィードバックを分析し、提案された改善を適用

**成功基準:** Copilot がアクションブルな提案とともに少なくとも3つのコード品質問題を特定



## 演習 2: Copilot への再レビューリクエスト

**目標:** 初期フィードバックが不十分な場合の適切な再レビューリクエスト方法を学習

### 手順

1. 特定された問題とその改善方法を分析
2. 分析に基づいて PR にコメントを追加
3. 変更後に GitHub Copilot に再レビューをリクエスト

**成功基準:** Copilot があなたのコメントに対応した更新されたフィードバックを提供



# 自動レビューの有効化

## 設定オプション

- ✓ 個人: あなた自身の pull request に対する自動レビュー
- ✓ Repository レベル: 特定リポジトリのすべての新しい pull request
- ✓ Organization レベル: パターンマッチングによる複数リポジトリ

## メリット

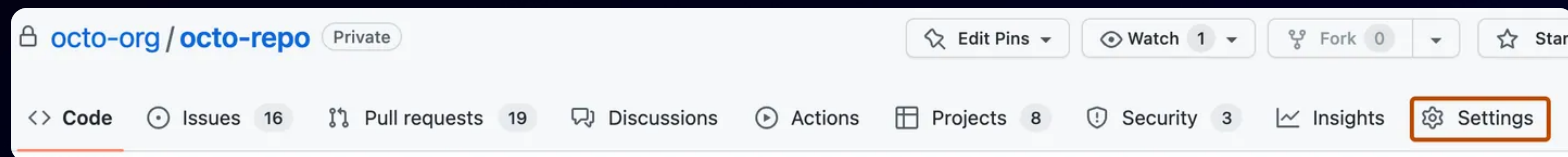
- ✓ 一貫したコード品質の強制
- ✓ すべての pull request への即座のフィードバック
- ✓ 手動レビューのオーバーヘッド削減



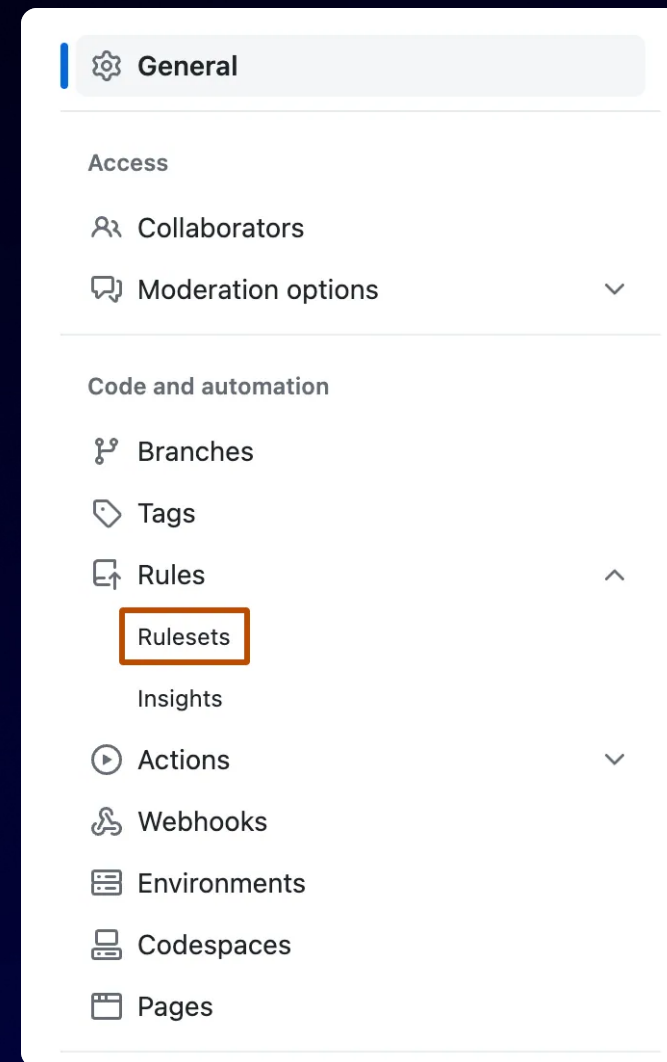
# 自動レビューの設定: 単一リポジトリ (1)

## Branch Ruleset の作成

1. リポジトリの **Settings** タブに移動



2. 左サイドバーで **Rules** → **Rulesets** を選択
3. **New ruleset** → **New branch ruleset** をクリック





# 自動レビューの設定: 単一リポジトリ (2)

## Copilot レビューの有効化

1. ruleset 名を設定し、Enforcement Status を Active に設定
2. Target branches で適切なブランチカバレッジを選択
3. Require pull request before merging を有効化
4. Request pull request review from Copilot をチェック
5. Create をクリックして有効化

☒ **Require a pull request before merging**  
Require all commits be made to a non-target branch and submitted via a pull request before they can be merged.

Hide additional settings ^

**Required approvals**

0 ▾

The number of approving reviews that are required before a pull request can be merged.

☐ **Dismiss stale pull request approvals when new commits are pushed**  
New, reviewable commits pushed will dismiss previous pull request review approvals.

☐ **Require review from specific teams** Private preview  
A collection of reviewers and associated file patterns. Each reviewer has a list of file patterns which determine the files that reviewer is required to review.

☐ **Require review from Code Owners**  
Require an approving review in pull requests that modify files that have a designated code owner.

☐ **Require approval of the most recent reviewable push**  
Whether the most recent reviewable push must be approved by someone other than the person who pushed it.

☐ **Require conversation resolution before merging**  
All conversations on code must be resolved before a pull request can be merged.

☒ **Request pull request review from Copilot** Preview  
Automatically request review from Copilot for new pull requests, if the author has access to Copilot code review.

**Allowed merge methods** Preview

Merge, Squash, Rebase ▾

When merging pull requests, you can allow any combination of merge commits, squashing, or rebasing. At least one option must be enabled.

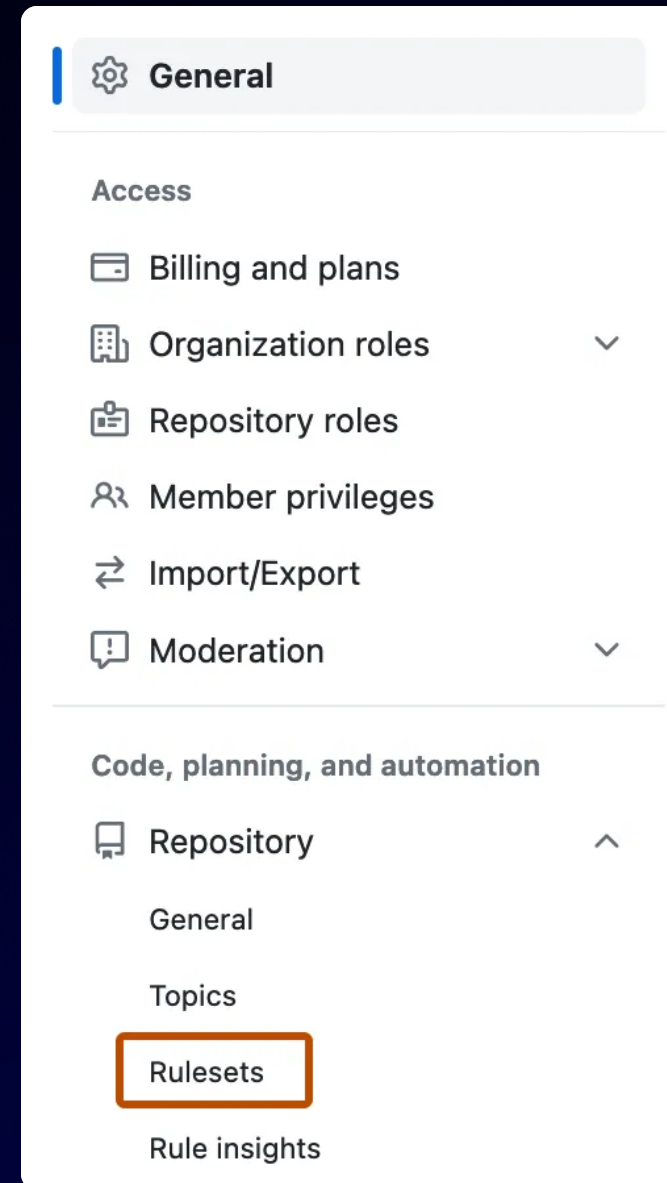


# Organization レベルの設定

## Organization Branch Ruleset の作成

1. プロフィールメニューから [Your organizations](#) にアクセス
2. organization の [Settings](#) に移動
3. "Code, planning, and automation" の下にある [Repository](#) → [Rulesets](#) を選択
4. organization スコープで [New branch ruleset](#) を作成
5. パターンマッチングを使用して [Target repositories](#) を定義
6. ブランチターゲティングと pull request 要件を設定
7. Copilot レビュー要件を有効化

**パターン例:** `*feature` は "feature" で終わるすべてのリポジトリにマッチ





# 演習 3: Repository 自動レビューの有効化

**目標:** リポジトリに対する自動 Copilot レビューを設定

## タスク

1. 管理者アクセス権を持つ3つのリポジトリを特定
2. 各リポジトリで自動 Copilot コードレビューを有効化
3. テスト pull request を作成して設定をテスト
4. 自動レビューが起動することを確認

**代替案:** 権限が不足している場合、ビジネス上の正当化とともにチーム有効化をリクエストする issue を作成



# カスタム指示によるレビューのカスタマイズ

設定ファイル: リポジトリルートの  
`.github/copilot-instructions.md`

## カスタマイズ機能

- ✓ 言語固有のレビュー設定
- ✓ セキュリティチェックリストの強制
- ✓ コードスタイルと可読性基準
- ✓ パフォーマンス最適化の重点領域

– コードレビューを実行する際は、日本語で回答してください。

ファイルスコープ: 指示は Copilot Chat と Code Review Agent の両方に適用



# カスタム指示の例

# Copilot レビュー指示

## セキュリティ重点

- SQL インジェクション脆弱性をスキャン
- 入力サニタイゼーションプラクティスを検証
- 認証と認可をチェック

## コード品質基準

- 一貫した命名規則を強制
- コード重複の機会を特定
- パフォーマンス最適化を提案

## チームの設定

- サイレント失敗よりも明示的エラーハンドリングを優先
- 可能な場合は JavaScript より TypeScript を推奨
- 保守性と可読性に焦点



## 演習 4: カスタムレビューインストラクションの作成

**目標:** コードベースに合わせたレビューガイドラインを実装

**手順:**

1. リポジトリに `.github/copilot-instructions.md` を作成
2. プロジェクトに関連する3-5の具体的なレビュー基準を定義
3. セキュリティ、品質、チーム設定ガイドラインを含める
4. 新しい pull request で指示をテスト
5. Copilot がカスタム基準に従うことを確認

**成功メトリクス:** Copilot レビューが指定されたガイドラインを反映し、プロジェクト固有の問題を捕捉

