

# GitHub Copilotのすべて

Azure DevOpsオンライン Vol.13 Agentic DevOpsって何？

---

2025年6月21日

# 本日のゴール

**(ほぼ) すべての機能の概略を知る**

---

※ 管理機能およびEnterprise限定などの一部機能を除く

## 本日のゴール **2**

本発表内容の学習の仕方を理解する

---

# 本日のゴール

本発表内容の学習コンテンツが自作できるようになる **3**

---



## 自己紹介

- 名前: 中村 充志
- 所属: リコージャパン株式会社
- Microsoft MVP for Development Technologies (2017～)

## 本日の環境

- GitHub Copilot Pro+ (Microsoft MVP特典)
- VS Code + C#

# 発表コンテンツについて

## 学習コンテンツ

---

<https://agreeable-island-0c8e4d900.6.azurestaticapps.net/>

## 発表資料

---

<https://github.com/nuitsjp/all-of-github-copilot>

**たのしい話（機能）の前に楽しくない（お金）の話**



# GitHub Copilot プラン比較

プラン	月額	主な対象	プレミアムリクエスト
Free	無料	個人開発者	50/月
Pro	\$10	個人開発者	300/月
Pro+	\$39	AIパワーユーザー	1,500/月
Business	\$19/ユーザー	チーム・組織	300/ユーザー/月
Enterprise	\$39/ユーザー	大規模組織	1,000/ユーザー/月

一部機能の制限とプレミアムリクエストのリミットの相違

# プレミアムリクエストとは

## 基本概念

---

### リクエストの定義

- Copilotへのすべての依頼
- チャットでの質問、コード生成、拡張機能の利用など

### 課金モデル

- **基本:** プランに含まれる月間許可量
- **追加:** 許可量超過時は **\$0.04/リクエスト**

# モデル別プレミアムリクエスト消費量

モデル	Premium リクエスト
基本モデル（現在はGPT-4.1）	0（有料ユーザー）、1（Free）
Claude Sonnet 4	1
Claude Opus 4	10
o3	1
o4-mini	0.33

# BusinessとEnterpriseの共存は可能か？

## よくある疑問

- ユーザーによって利用頻度が異なり、使い分けたいことがある
- コスト最適化を図りたい

 答え: **共存可能です！**

# ✓ BusinessとEnterpriseの共存可能

## 🔄 混在のメリット

- 同一リポジトリで異なるライセンスユーザーの混在は可能
- ユーザーの役割に応じたライセンス割り当て

## ⚠ 注意点

- ただ現状のライセンス設計にだいぶ無理がある
- 将来変わる可能性はあり

## 重要なポイント

### Organization制限

- OrganizationにはBusinessまたはEnterprise何れかのみ設定可能
  - 例: BusinessレベルのOrganizationでライセンスを割り当てるとBusinessに

### Team単位の管理

- CopilotライセンスはOrganization内のTeamごとに割り当てられる
  - Copilotが有効なユーザーと無効なユーザーが混在可能

### ライセンス優先度




- ライセンスを重複割当した場合、有効ライセンスは優先度によって決定
  - **Enterprise > Business > Pro+ > Pro**
  - おそらくガバナンスを優先するため

# Biz/Entの混在おすすめ設定

## ベストプラクティス

リポジトリ管理Organizationとライセンス管理Organizationを分ける

## 構成例

Organization	ユーザー	用途
 <b>Business License Org</b>	User B1, User B2	ライセンス管理専用
 <b>Enterprise License Org</b>	User E1, User E2	ライセンス管理専用
 <b>Repository Org</b>	全ユーザー	実際の開発作業

# 機能紹介



## 機能①

#	機能名	Free	Pro	Pro+	Business	Enterprise
1	Code completion	✓	✓	✓	✓	✓
2	Copilot Chat	✓	✓	✓	✓	✓
3	Copilot coding agent	✗	✗	✓	✗	✓
4	Copilot in the CLI	✓	✓	✓	✓	✓
5	GitHub Copilot code review	✓	✓	✓	✓	✓
6	GitHub Models	✗	✗	✓	✓	✓
7	Repository and personal custom instructions	✓	✓	✓	✓	✓
8	Organization custom instructions	✗	✗	✗	✓	✓

## 機能②

#	機能名	Free	Pro	Pro+	Business	Enterprise
9	Copilot prompt files	✓	✓	✓	✓	✓
10	Copilot pull request summaries	✗	✓	✓	✓	✓
11	Copilot text completion	✗	✗	✗	✗	✓
12	GitHub Copilot Extensions	✓	✓	✓	✓	✓
13	Copilot Workspace	✗	✓	✓	✓	✓
14	Copilot Spaces	✓	✓	✓	✓	✓
15	Copilot knowledge bases	✗	✗	✗	✗	✓

# ⚠️ 最初にお断り

## 🚫 本日デモできない機能

次の機能は本日お見せできません。

1. 🔒 **制限Preview機能** - 現在募集されていないもの
2. 🏢 **Enterprise限定機能** - 高度な管理・統合機能

これらは簡単な解説のみとなります。

# 1. Code Completion

## リアルタイムコード補完

---

### 概要

- AIによる自動的なコード提案
- 文脈を理解した賢い補完
- 複数行の提案も可能

# デモ: Code Completion

## 実演内容

---

### 1. 基本的なコード補完

- メソッドの自動生成
- パターンの認識

### 2. コメントからのコード生成

```
// 2つの数値を足し算するメソッド
```

### 3. 複雑なロジックの提案

- エラーハンドリング

# 2. Copilot Chat

## 対話型コーディングアシスタント

### 3つのモード

モード	用途	特徴
Ask	質問・説明	コードの理解、技術的な質問
Edit	ファイル編集	複数ファイルの制御された編集
Agent	自律的実行	タスクの自動完了、ツール実行

### コンテキスト指定

#file , #codebase , #selection , @workspace

# デモ: Copilot Chat - Ask Mode

## 実演内容

---

### 1. コードの説明を求める

このCalculatorクラスの機能を説明してください

### 2. ベストプラクティスの質問

C#でのエラーハンドリングのベストプラクティスは？

### 3. コンテキストを使った質問

#file:Calculator.cs このファイルの改善点は？

# デモ: Copilot Chat - Edit Mode

## 実演内容

---

### 1. 複数ファイルの編集

- Calculator.cs と CalculatorTests.cs の同時更新

### 2. リファクタリング

Calculatorクラスをインターフェースに分離して

### 3. 段階的な変更の実行

- 提案の確認
- 選択的な適用



# デモ: Copilot Chat - Agent Mode

## 実演内容

---

### 1. 自律的なタスク実行

新しい機能「平方根計算」を実装して、テストも書いて

### 2. エラーの自動修正

- ビルドエラーの検出と修正

### 3. テストの実行と修正

- 失敗したテストの自動修正

## 3. Copilot Coding Agent

### 自動実装エージェント

---

#### 機能

- GitHub IssueからPRまでの自動実装
- コードの自動生成と検証
- テストの作成と実行

#### ワークフロー

1. Issueの理解と分析
2. 実装計画の作成

## 4. Copilot in the CLI

### コマンドライン支援

---

#### 機能

- コマンドの提案
- コマンドの説明
- エラーの解決支援

#### 使用例

```
gh copilot suggest "ファイルを再帰的に検索"  
gh copilot explain "git rebase -i HEAD~3"
```

# 5. GitHub Copilot Code Review

## AIによるコードレビュー

---

### 機能

- PRの自動レビュー
- 改善提案の生成
- セキュリティ問題の検出

### レビュー内容

- コード品質
- ベストプラクティス

## 6. GitHub Models

### AI言語モデルへのアクセス

---

#### 機能

- 業界をリードする大規模・小規模言語モデルへの直接アクセス
- 様々なAIモデルの試用と比較
- プロトタイピングと実験

#### 用途

- モデルの選定
- APIの事前検証

# 7. Repository and Personal Custom Instructions

## 応答のカスタマイズ

---

### 機能

- 個人の好みに基づく応答調整
- 使用ツール・フレームワークの指定
- プロジェクト固有の要件設定

### 設定例

- コーディング規約の指定
- 優先言語/フレームワーク

## 8. Organization Custom Instructions

### 組織レベルのカスタマイズ

---

#### 機能

- 組織全体での統一された応答設定
- 企業のコーディング規約の適用
- セキュリティポリシーの強制

#### メリット

- 一貫性のあるコード生成
- 組織標準の自動適用

# 9. Copilot Prompt Files

## プロンプトファイルの活用

---

### 機能

- 再利用可能なプロンプトの定義
- プロジェクト固有の設定
- チーム間での共有

### 使用例

- `.github/copilot-prompts.md`
- テンプレートの定義



# 10. Copilot Pull Request Summaries

## PR要約の自動生成

---

### 機能

- PR変更内容の自動要約
- 影響範囲の分析
- レビューア向けの説明生成

### メリット

- レビュー時間の短縮
- 変更内容の明確化

# 11. Copilot Text Completion

## テキスト補完支援

---

### 機能

- PR説明文の迅速・正確な作成支援

### メリット

- 一貫性のある文書作成
- 時間の節約
- 品質の向上

# 12. GitHub Copilot Extensions

## 外部ツールの統合

---

### 機能

- 外部ツールのCopilot Chatへの統合
- サードパーティサービスとの連携
- カスタム拡張の利用

### 対応ツール例

- Docker
- Azure

# 13. Copilot Workspace

## 統合開発環境

---

### 機能

- IssueからPRまでの統合フロー
- コードの自動生成と検証
- レビューと改良の統合

### ワークフロー

1. Issueの理解
2. 実装計画の生成

# 14. Copilot Spaces

## コンテンツ管理・共有

---

### 機能

- 関連コンテンツの整理・集中化
- コンテキストの共有
- チーム間のナレッジ共有

### 用途

- プロジェクトドキュメントの管理
- コードスニペットの共有

# 15. Copilot Knowledge Bases

## 組織知識の統合

---

### 機能

- 組織ドキュメントのコレクション作成・管理
- 内部ナレッジベースとの統合
- カスタムコンテキストの提供

### 用途

- 社内コーディング規約の適用
- ドメイン知識の活用

# 管理者向け機能

## Business/Enterprise プラン

---

### アクセス管理

- メンバーのライセンス管理
- 組織レベルのポリシー設定

### 使用状況分析

- 利用統計の確認
- 効果測定

### セキュリティ

# 学習コンテンツの活用方法

## オンライン教材

---

<https://agreeable-island-0c8e4d900.6.azurestaticapps.net/>

## 学習のポイント

- 正直学習コンテンツとしての精度は微妙
- 発表資料のリポジトリをクローンしてCopilot Chatで適宜質問・修正しながら実行するのがお勧め
- 別に完璧な教材なんて必要がない




## 発表資料

 [https://github.com/nuitsin/all\\_of\\_github\\_copilot](https://github.com/nuitsin/all_of_github_copilot)



# 学習コンテンツの作成のポイント

## 効率的な教材作成フロー

1.  **ソース活用**: 公式ドキュメントなどからプロンプトで学習教材を作成する
2.  **直接参照**: `#fetch` で直接参照させる方法が楽
3.  **ローカル保管**: ただローカルにMarkdownで保管してからプロンプトかけたほうがコンテキスト長の圧迫が少なくてすみそう
  - Chrome拡張（Webpage to Markdownなど）の併用がお勧め

## 発表資料

発表資料を保管しているリポジトリにプロンプトの例があるので参考にどうぞ。

# まとめ

## 本日本日お伝えしたこと

---

### GitHub Copilotの全機能概要

- 15の主要機能
- プラン別の利用可能機能

### 効果的な学習方法

- 段階的アプローチ
- 実践的な練習

### 継続的な活用

# ありがとうございました！

 **GitHub Copilotで開発を加速しましょう！**

---

## リソース

- 学習コンテンツ: <https://agreeable-island-0c8e4d900.6.azurestaticapps.net/>
- 公式ドキュメント: <https://docs.github.com/copilot>
- VS Code ドキュメント: <https://code.visualstudio.com/docs/copilot/>

 **Happy Coding with GitHub Copilot!**

# ありがとうございました！