Kiroともっと仲良くなろう

ーKiroカスタマイズのすすめー

アマゾン ウェブ サービス ジャパン 合同会社
ソリューションアーキテクト
伊藤 一成

自己紹介

伊藤 一成 (Kazunari Ito)

職務：ソリューションアーキテクト
　　主に西日本のお客様を支援させていただいています。

略歴：百貨店情報システム子会社で開発・運用
　　→出向ビジネス
　　→SMB向けワンストップ サービス（企画から運用まで支援）
　　→IoTプラットフォーム構築・運用

好きなAWSサービス：AWS IoT Core 

趣味：サイクリング、テニス、軽登山、ボウリング・・などなど。
　　最近は3Dプリンタも使い始めました。

今日、こんなこと起こりませんでした？

1. **「毎回同じこと説明してる…」**
   コーディングルール、レポート形式、命名規則…毎回伝え直す

2. **「うちのルール知らないじゃん」**
   業務フロー、登録ルール、ワークフロー…進め方の作法を知らない

3. **「便利だけど、定着しない」**
   結局、手作業に戻ってしまう

→ **カスタマイズ**で解決できます

Kiro のカスタマイズ要素

Steering
行動指針・ルール
常時読み込み

Skills
手順書
必要なときだけ参照

MCP
外部ツール接続
常時読み込み

Powers
MCP+ 知識 +Hooks の組み合わせ
自動で起動

Agent
役割定義
ツール・資料・ MCP をまとめて切替

Hooks
自動実行トリガー
特定条件で発動

Prompts
再利用可能な指示テンプレート
使い捨て

Kiro のカスタマイズ要素

Steering
行動指針・ルール
常時読み込み

Skills
手順書
必要なときだけ参照

MCP
外部ツール接続
常時読み込み

Powers
MCP+ 知識 +Hooks の組み合わせ
自動で起動

Agent
役割定義
ツール・資料・ MCP をまとめて切替

Hooks
自動実行トリガー
特定条件で発動

Prompts
再利用可能な指示テンプレート
使い捨て

…わかりにくくないですか？

RPG の冒険者で整理する

Kiro は「あらゆる訓練を受けた冒険者」— 能力はあるが、あなたのギルドのやり方は知らない

Kiro の要素	冒険者で言うと
Steering	人となり（性格・こだわり・ギルドのルール）
Skills	依頼の手順書（特定の任務のときだけ開く）
MCP	常備装備（常に身につけている武器・道具）
Powers	召喚獣（呼べば装備・知識・仕掛けを丸ごと持って現れる）
Agent	ジョブ / クラス（装備・手順書をまとめた役割定義）
Hooks	トラップ / 仕掛け（特定条件で自動発動）
Prompts	スクロール（使うと発動する消耗品）

Kiro のカスタマイズ要素

Steering
行動指針・ルール
常時読み込み
人となり（性格・こだわり・ギルドのルール）

Skills
依頼の手順書
（特定の任務のときだけ開く）

MCP
常備装備
（常に身につけている武器・道具）

Powers
召喚獣
（呼べば装備・知識・仕掛けを丸ごと持って現れる）

Agent
ジョブ / クラス
（装備・手順書をまとめた役割定義）

Hooks
トラップ / 仕掛け
（特定条件で自動発動）

Prompts
スクロール
（使うと発動する消耗品）

…わかった気分になりません？？

まず Steering から始めよう

Steering = 冒険者の**人となり**

性格（Global / Principal）
根本原因を追求する、批判的に考える
どのプロジェクトでも変わらない行動指針

こだわり（Global / 具体ルール）
コーディングルール、命名規則
どのギルドに行っても変えない自分のやり方

ギルドのルール（Workspace）
ワークフロー、レポート形式、登録ルール
このプロジェクト固有のやり方

配置場所は大きく2箇所！
グローバル：　**~/.kiro/steering/**
ワークスペース：**プロジェクトルート/ .kiro/steering/**

~/.kiro/steering/development-principles.md

Markdown

```
# 開発原則
## コミュニケーションとレビュー
### 実質的な議論を重視
- 内容を重視し、深みのない褒め言葉は省く
- 前提を疑い、バイアスを指摘し、反論を提示する
- 必要なら反対意見を恐れず表明する
- 同意する場合も理由と証拠に基づいて説明する
### 批判的思考
- 提案された解決策の前提条件を検証する
- 代替案を検討し、トレードオフを明確にする
- 「なぜこのアプローチなのか」を常に問う

## 修正前の検証
- コード修正前にその修正で正しいか
  客観的に確認してから修正に入る
- 修正が要件を満たすか確認する
- 修正が既存の機能を壊さないか検証

## 根本原因分析
- 同じ修正を繰り返し行わない
- 同じエラーが再発する場合は
  根本原因を特定する
- 「なぜ」を 5 回繰り返す
```

Steering の次は？

Skills（手順書）

長い手順を分離
必要なときだけ読み込み

MCP（装備）

外部ツール接続
常時読み込み

Agent（ジョブ）

装備・手順書をまとめた役割定義
/agent で切替

Powers（召喚獣）

ツール＋知識＋仕掛けを丸ごと召喚
キーワードで動的にアクティブ化

Steering の次は？

Skills（手順書）

長い手順を分離
必要なときだけ読み込み

MCP（装備）

外部ツール接続
常時読み込み

Agent（ジョブ）

装備・手順書をまとめた役割定義
/agent で切替

Powers（召喚獣）

ツール＋知識＋仕掛けを丸ごと召喚
キーワードで動的にアクティブ化

荷物の限界（コンテキストウィンドウ）

MCP = 常備装備。常に荷物を圧迫
Powers = 召喚獣。必要なときだけ現れる

→ 使い分けが重要！

	MCP（常備装備）	Powers（召喚獣）
読み込み	常時	キーワードで動的
コンテキスト	常に消費	召喚中だけ
含まれるもの	ツールだけ	ツール + 知識 + 仕掛け

今日からできること

1. **Steering を 1 つ書いてみる**
   毎回 Kiro に言っていることを .kiro/steering/ に Markdown で書き出す

2. **長い手順は Skills に分離**
   SKILL.md に YAML フロントマター + 手順を書く

3. **MCP / Powers で外部ツールを繋ぐ**
   常時必要なら MCP 、動的に呼びたいなら Powers

最初の一歩は Steering 。それだけで「毎回同じこと言う」がなくなる。

（Kiroに書かせてもOK！）

Thank you!