CLIツールでの開発環境改善まとめ

話す内容はこちら

・いくつかのツールを使った所感
今回話すツールはGeminiCLI Copilot CLIなどのCLI系

・以下の二つを中心に話します
最新の機能ではなく、継続していきたい事
今後活用できそうな工夫

自己紹介

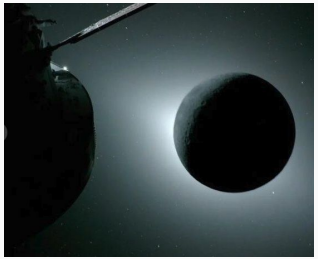
名前:蓬莱寿成

所属:株式会社アイスタイル

X:pawn_4_t

[今月の個人的に興味のあったニュース]

54年ぶりの有人の月周回



継続していきたい事とは？

継続していきたい事とは？

AIの動きが早い

継続していきたい事とは？

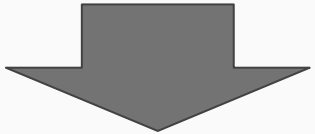
AIの動きが早い

新しい
モデル

新しい
機能

新しい
ツール

動きが早いので構築しても、陳腐化するかもしれない
＝モチベーションが繋がらない



自分がするのが**苦痛になる部分**を代替すれば、
継続する意欲につながる

継続していきたい事とは？

AIの動きが早い

新しい
モデル

新しい
機能

新しい
ツール

動きが早いので構築しても、陳腐化するかもしれない
＝モチベーションが繋がらない

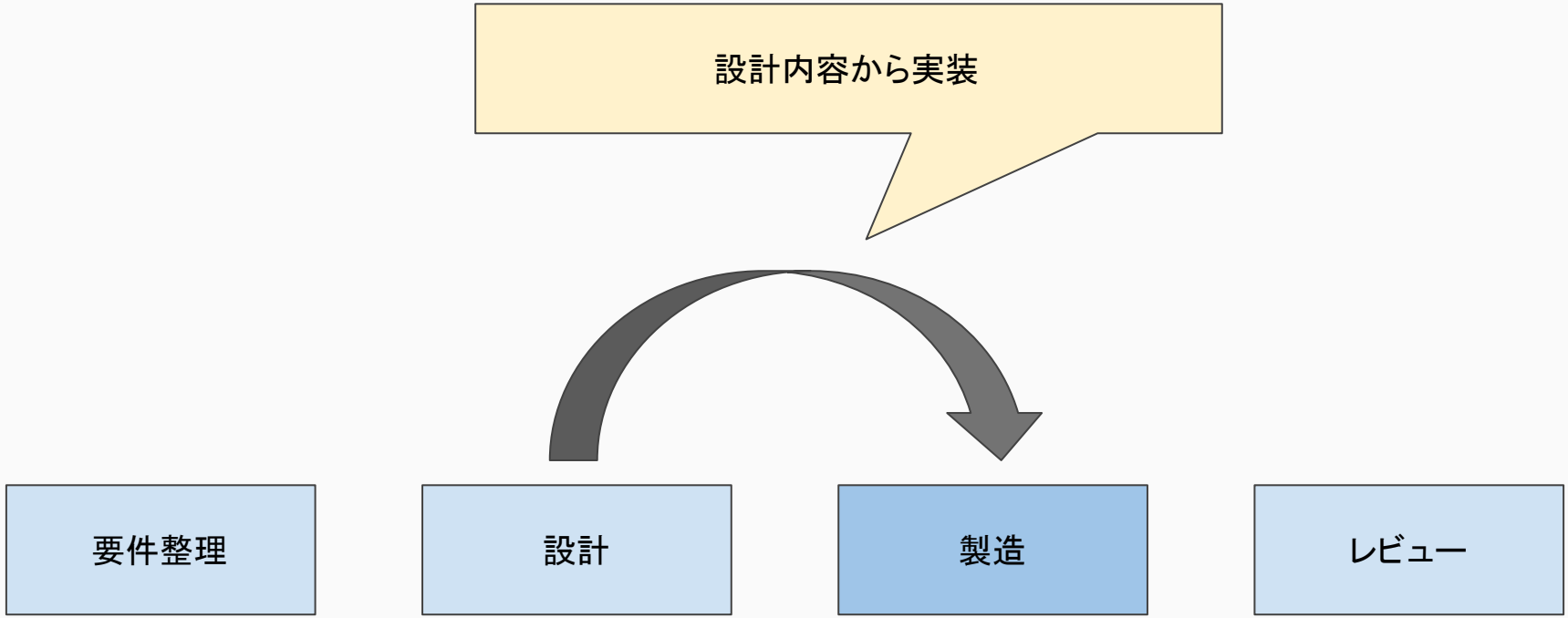
二度同じ事をする

終わった後に新たな
作業が出てくる

自分がするのが**苦痛になる部分**を代替すれば、
継続する意欲につながる

MCPの活用

仕様書の作成や仕様からの実装を作成する
MCPは大量に設定するとトークンを消費するので選別は必要

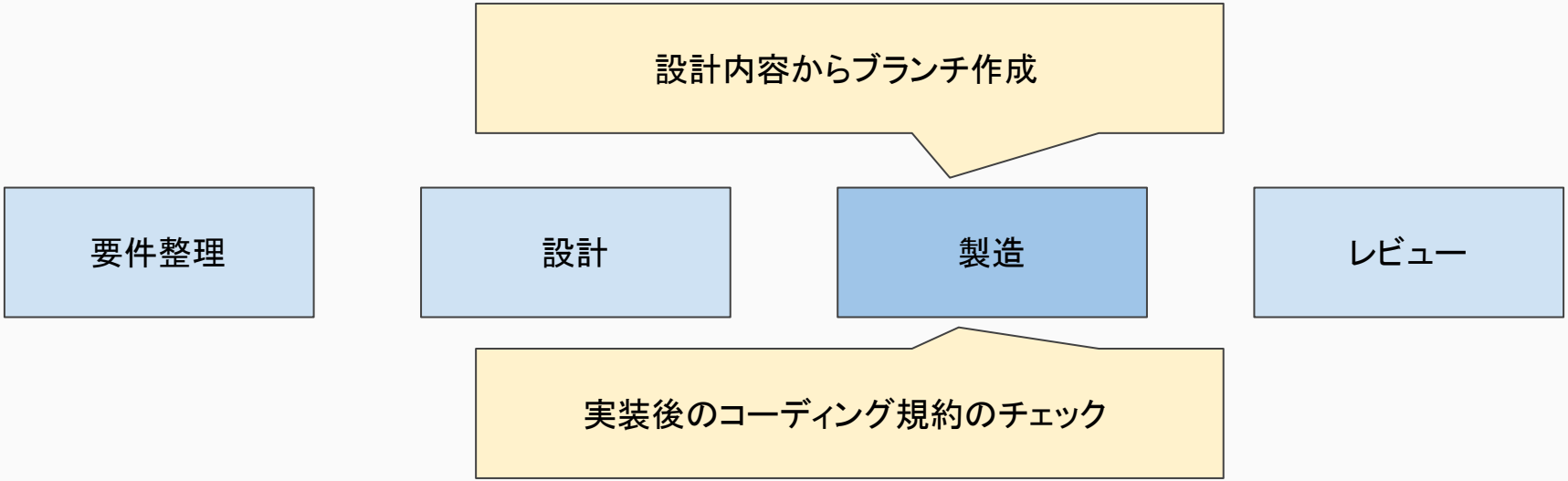


スキルで手順の統一

・スキルとは

同じ手順を繰り返しできるようにする機能

・どのような時に使っているか



スキルで手順の統一

・設計内容からブランチ作成

ブランチ名に実施内容を踏まえた名前や特定の文言を入れるなどのルールがある

例)branch/no-XXXX_(実施内容)

手動でルールに合わせるのは手間＋実施内容から名前を考える時間の省略

スキルで手順の統一

・実装後のコーディング規約のチェック

以前からGithubのプルリク作成時にCopilotでチェックを行っていた

スキルで手順の統一

・実装後のコーディング規約のチェック

以前からGithubのプルリク作成時にCopilotでチェックを行っていた

課題
修正して動作確認してプルリクを作成した時に抜けていた規則が見つかる
→再度修正すると動作確認が必要になる
＝プルリクを作成してからマージまでの時間が長くなった

スキルで手順の統一

・実装後のコーディング規約のチェック

以前からGithubのプルリク作成時にCopilotでチェックを行っていた

課題
　修正して動作確認してプルリクを作成した時に抜けていた規則が見つかる
　→再度修正すると動作確認が必要になる
　＝プルリクを作成してからマージまでの時間が長くなった

対応
　動作確認前にコーディング規約をチェックさせる

スキルで手順の統一

・実装後のコーディング規約のチェック

1)コーディング

2)一旦できた状態でコミット

4)動作確認

5)プッシュ＆プルリクエスト作成

スキルで手順の統一

・実装後のコーディング規約のチェック

1)コーディング

2)一旦できた状態でコミット

3)コーディング規約確認+修正

4)動作確認

5)プッシュ＆プルリクエスト作成

行っていること

1)git fetchでgitの状態を最新化
2)mainブランチと差分のあるファイルの一覧を作成
3)ファイル毎の差分を確認
4)差分内容をコーディング規約を照合
5)規約に違反する部分を修正
どういう**理由**で修正したかをコメントで残す
6)アプリケーションのビルド
7)テストコードが正常に実行できるかを確認

今後したいこと事

今後したい事

安全に性能を**最大**まで引き出す事

今後したい事

安全に性能を**最大**まで引き出す事

・安全とは
少なくとも**自分が監視できる環境** や、たとえ壊れても**影響がない環境** で行う

・最大とは
任せられるタスクは**同時並行** で行う
複数視点でタスクを進めること

今後したい事

安全に性能を**最大**まで引き出す事

・安全とは
少なくとも**自分が監視できる環境** や、たとえ壊れても**影響がない環境** で行う

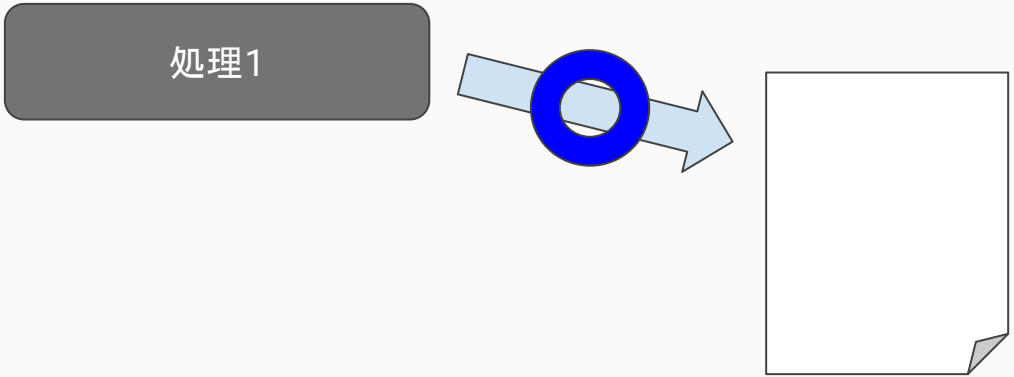
・最大とは
任せられるタスクは**同時並行** で行う
複数視点でタスクを進めること

スキルとコンテナ環境で実現する！！

スキル＋コンテナ環境(同時並行)

・CLI系のツールの課題

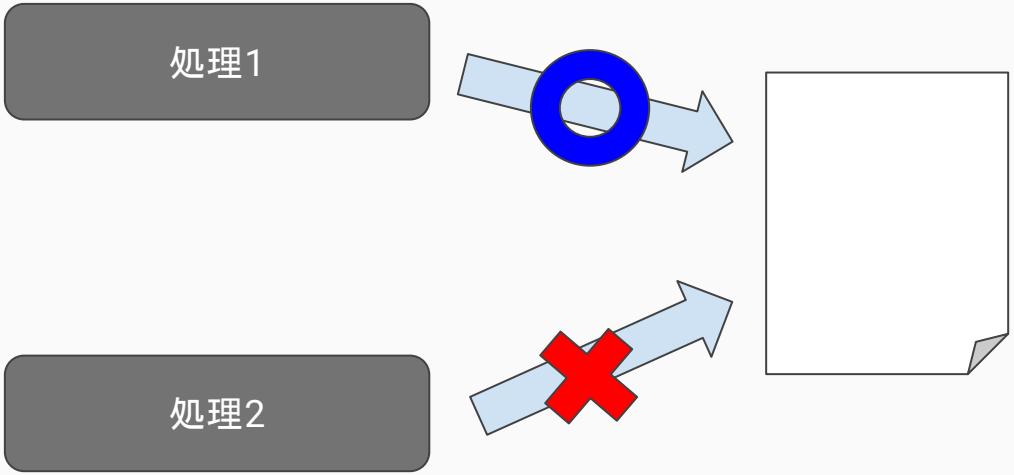
同時に処理をさせることが環境を作らないと難しい
出せる性能を100%を生かせない



スキル＋コンテナ環境(同時並行)

・CLI系のツールの課題

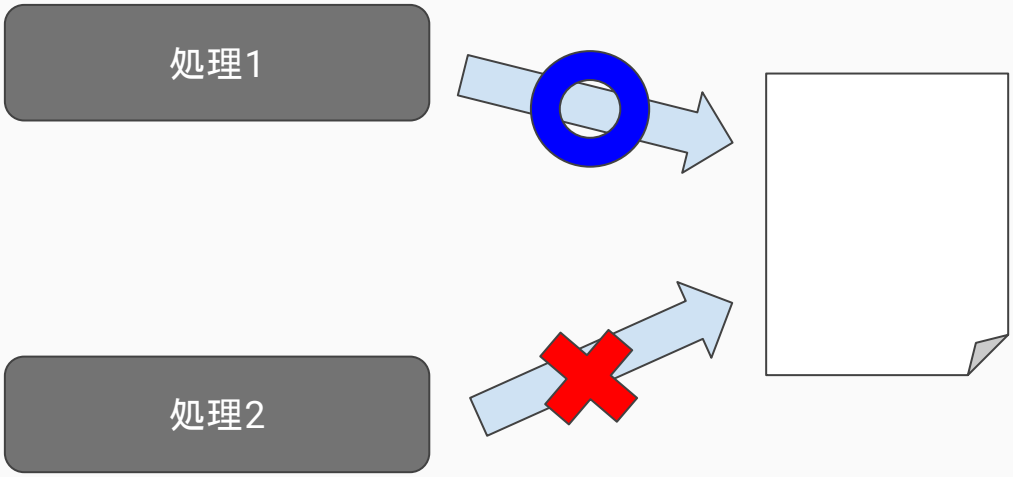
同時に処理をさせることが環境を作らないと難しい
出せる性能を100%を生かせない



スキル＋コンテナ環境(同時並行)

・CLI系のツールの課題

同時に処理をさせることが環境を作らないと難しい
出せる性能を100%を生かせない



本当は処理3つできるがファイルで制限

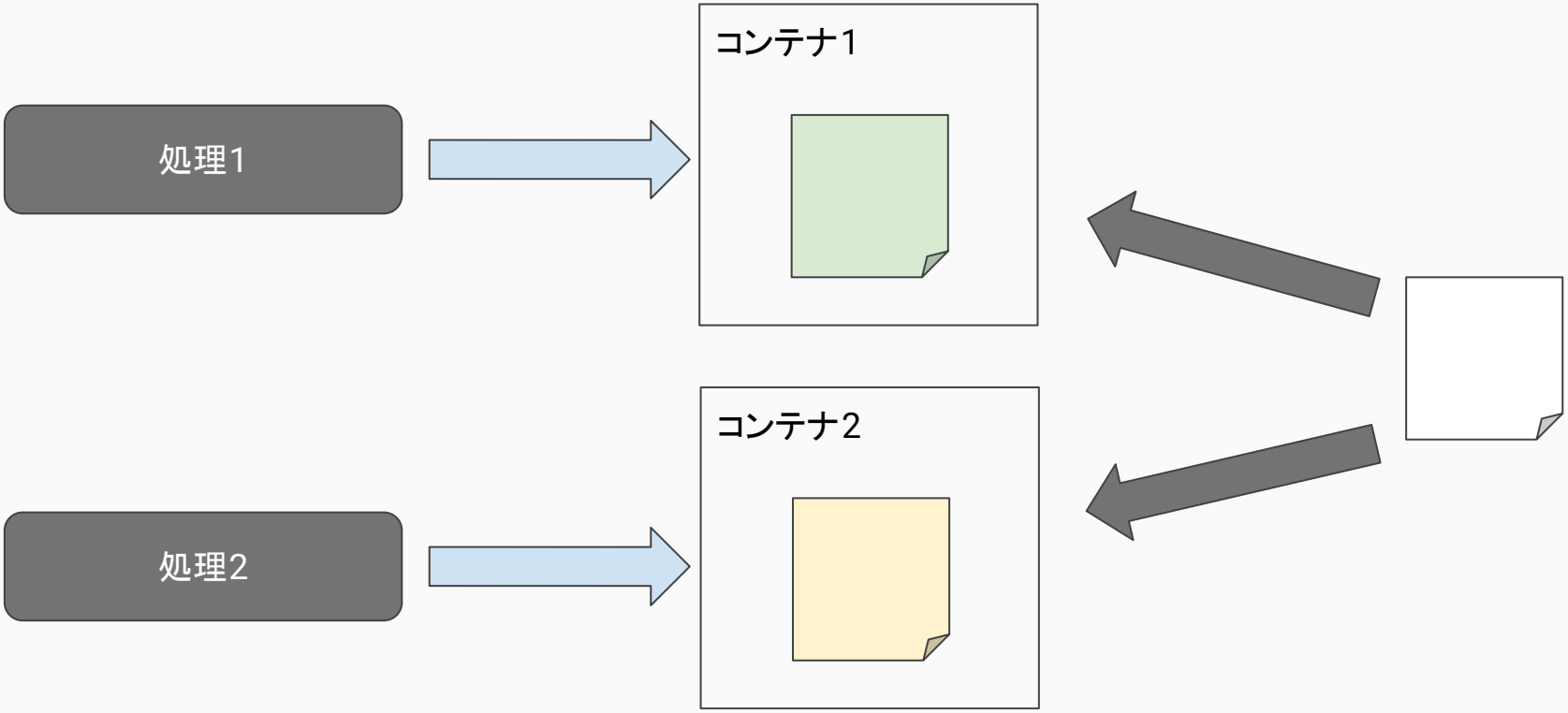
処理1

処理2

処理3

スキル＋コンテナ環境(同時並行)

・CLI系のツールの課題



スキル＋コンテナ環境(同時並行)

・CLI系のツールの課題
スキルで以下の流れで実行
1)gitやDockerが使えることを確認
2)イメージを取得
3)コンテナを作成
4)gitでリモートブランチからソース取得
5)ブランチを作成
6)修正
7)コミット+プッシュ

参考資料

GeminiCLIを複数同時実行！Dockerで作るAI並行開発環境

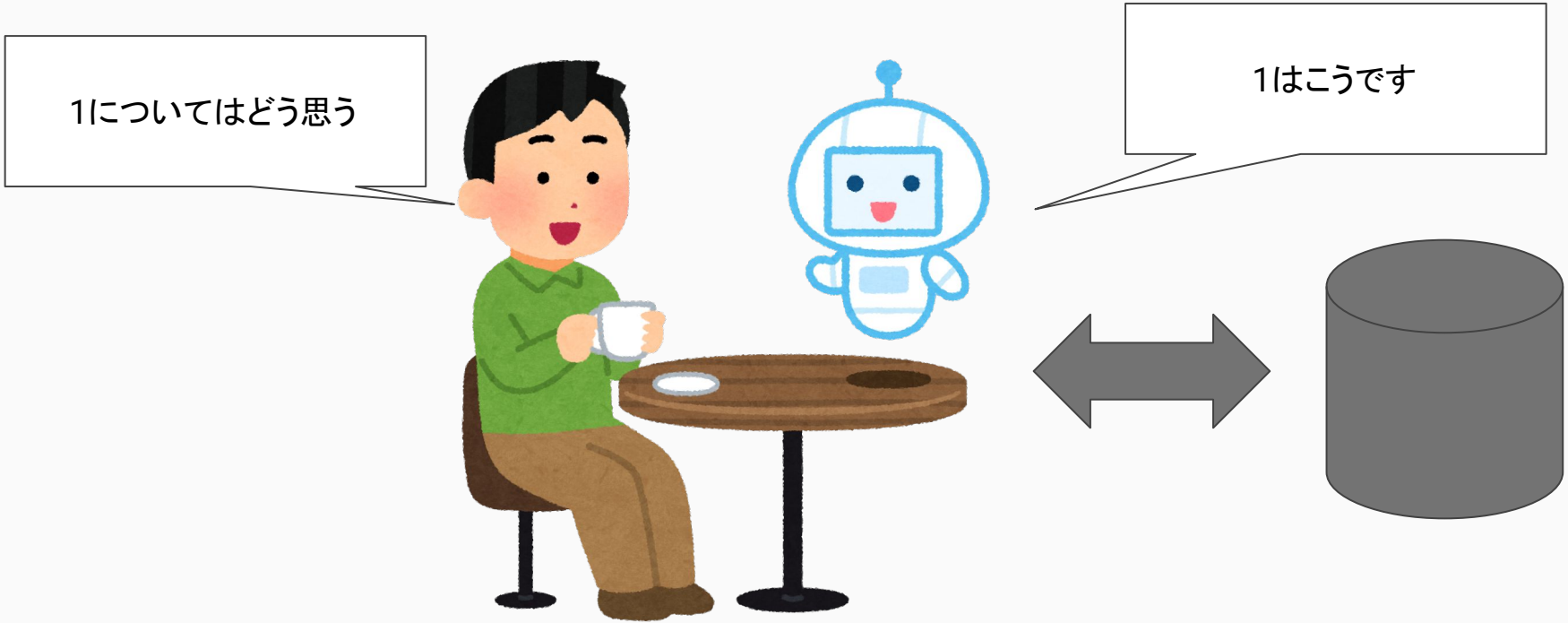
https://qiita.com/ho-rai/items/a5fd9a111a348526ce6d

スキル＋コンテナ環境(複数視点)

・AI系のツールの課題

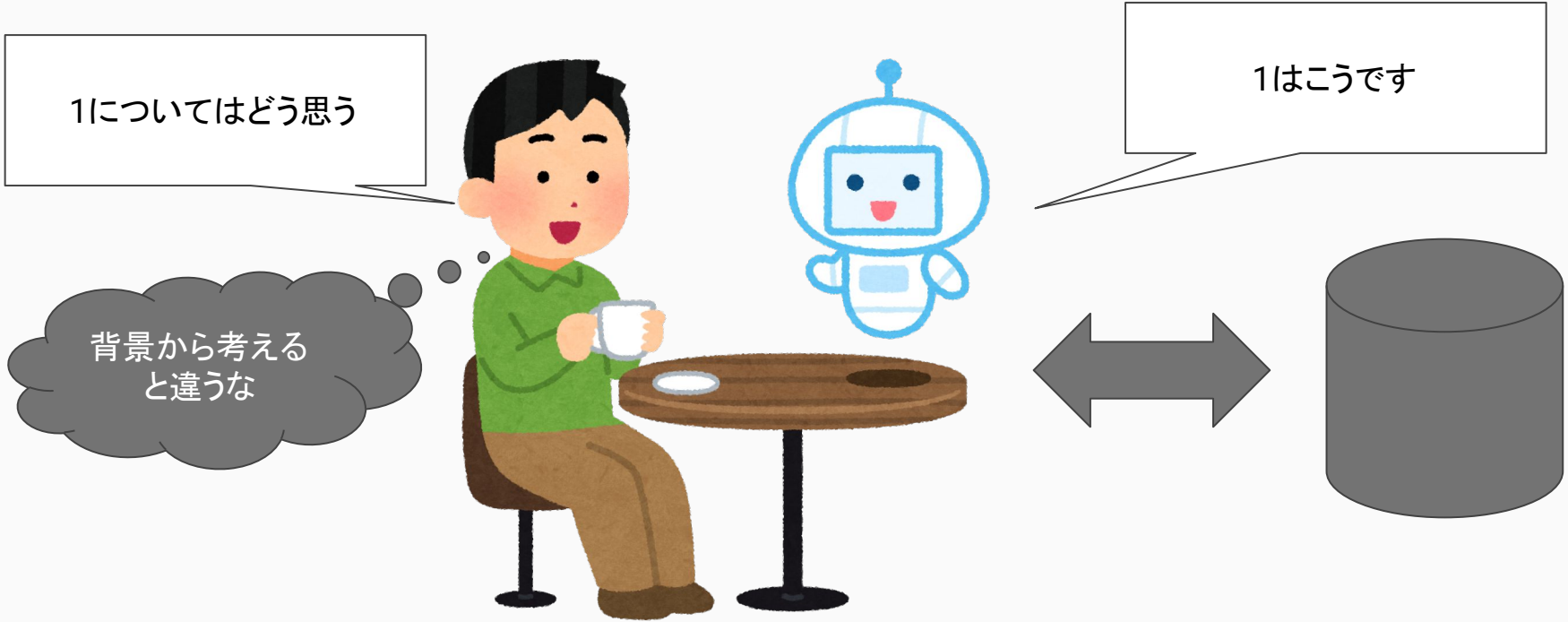
スキル＋コンテナ環境(複数視点)

・AI系のツールの課題



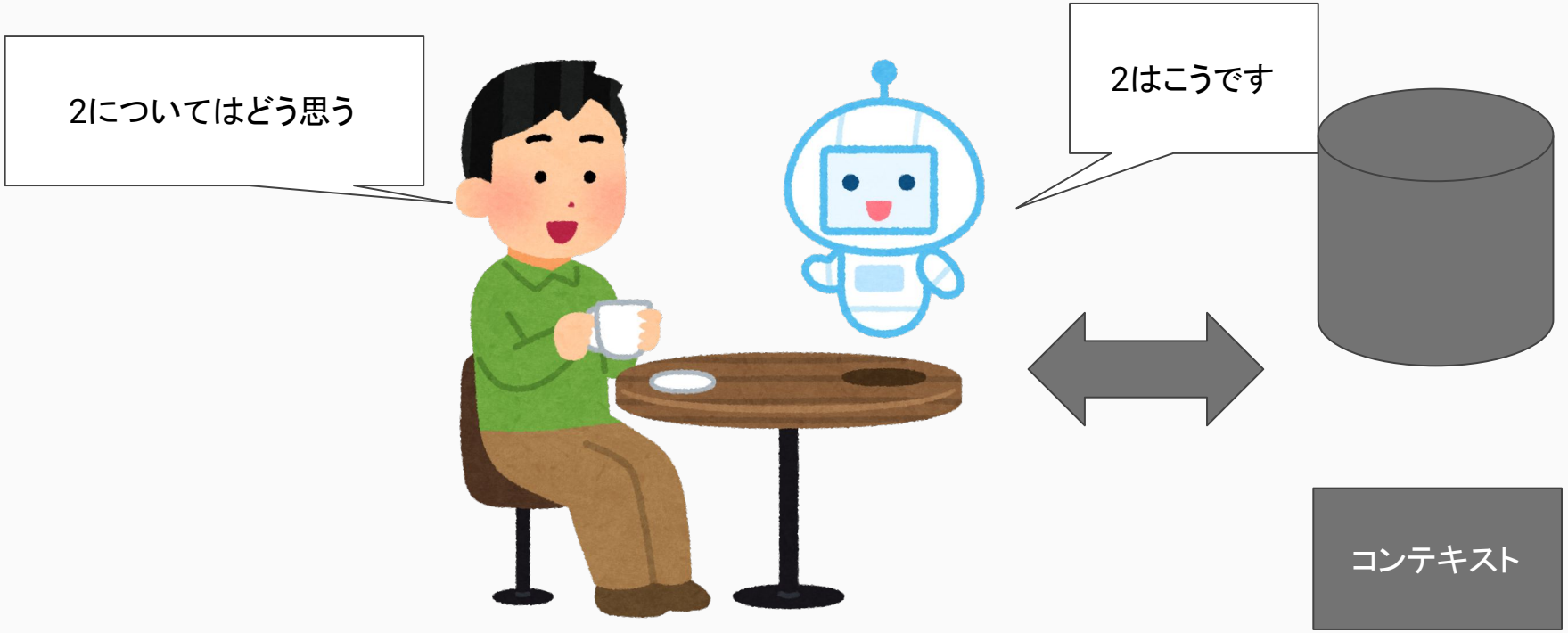
スキル＋コンテナ環境(複数視点)

・AI系のツールの課題



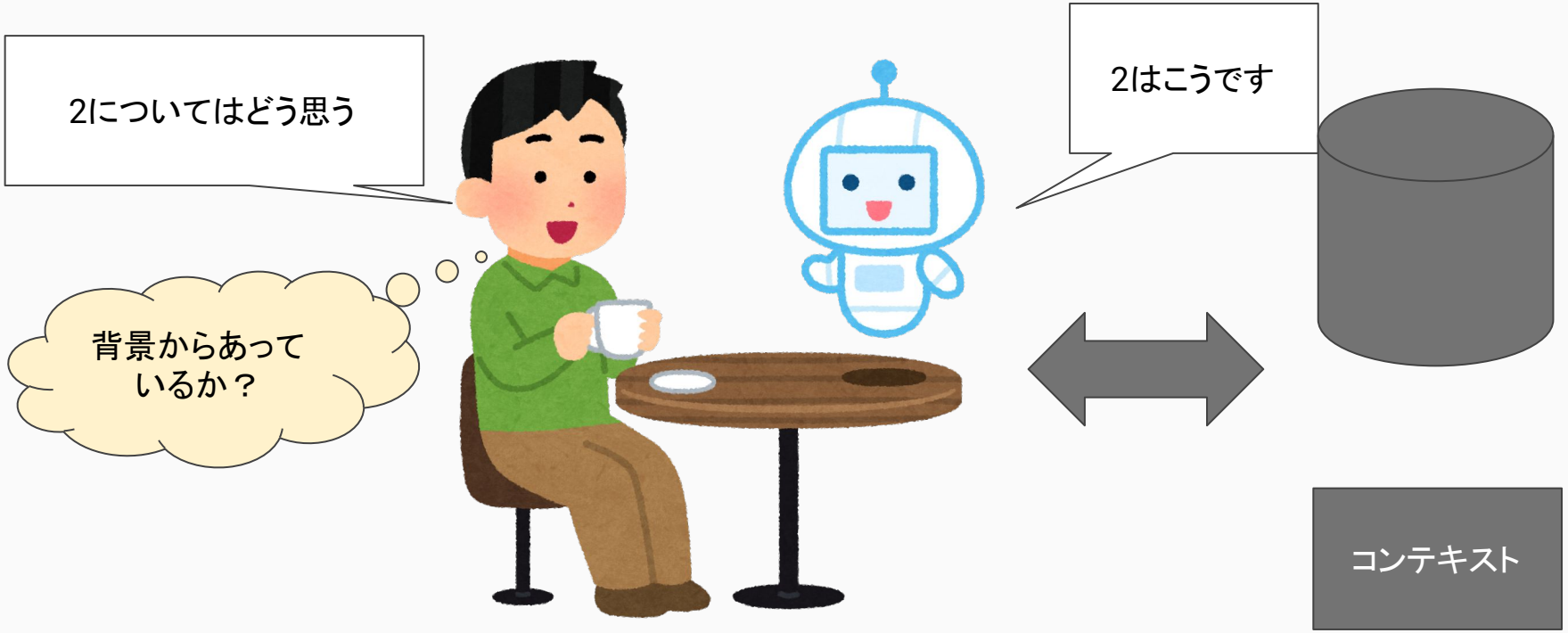
スキル＋コンテナ環境(複数視点)

・AI系のツールの課題



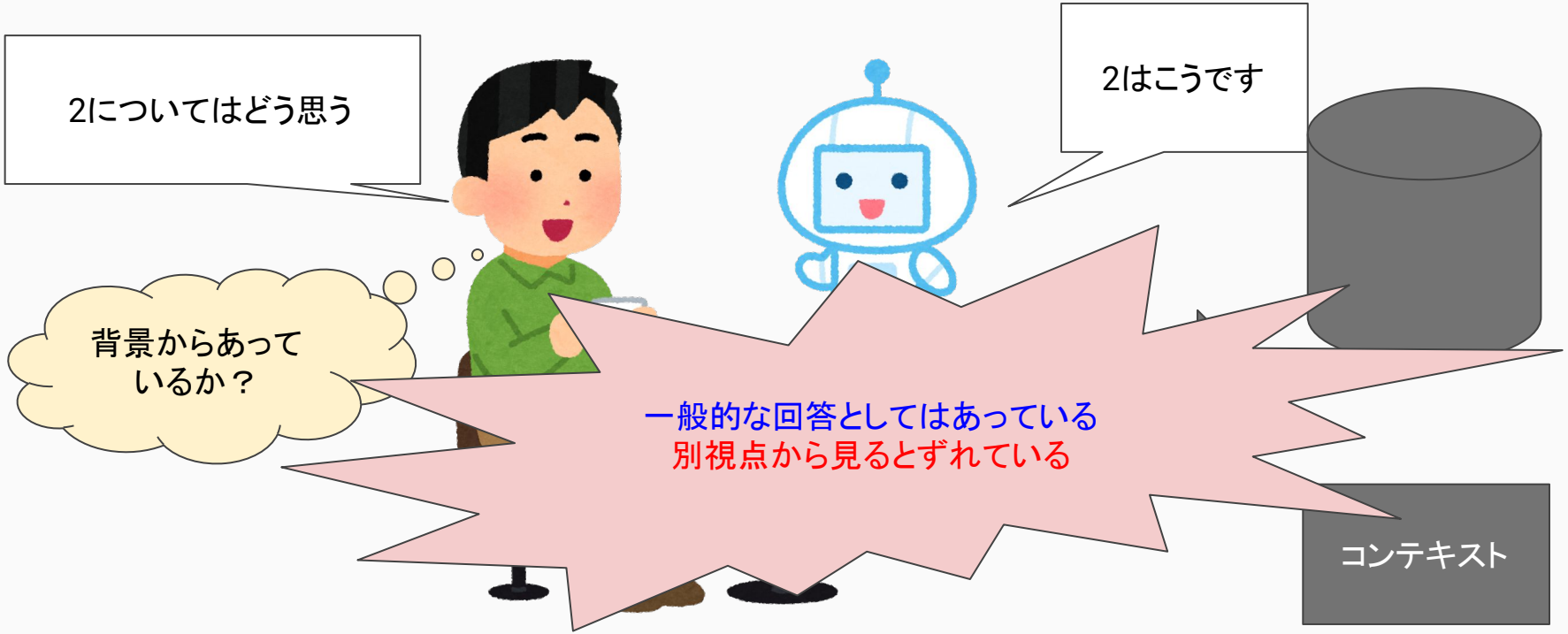
スキル＋コンテナ環境(複数視点)

・AI系のツールの課題



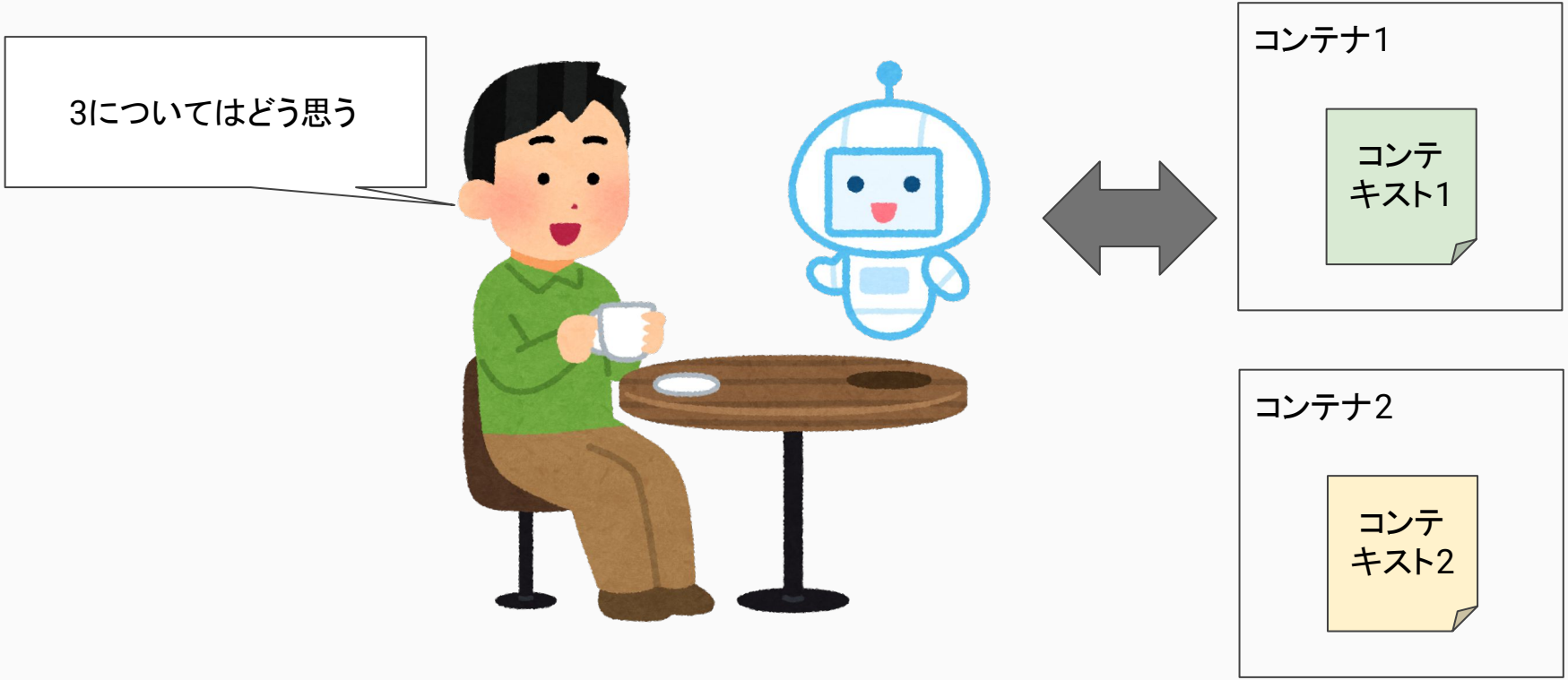
スキル＋コンテナ環境(複数視点)

・AI系のツールの課題



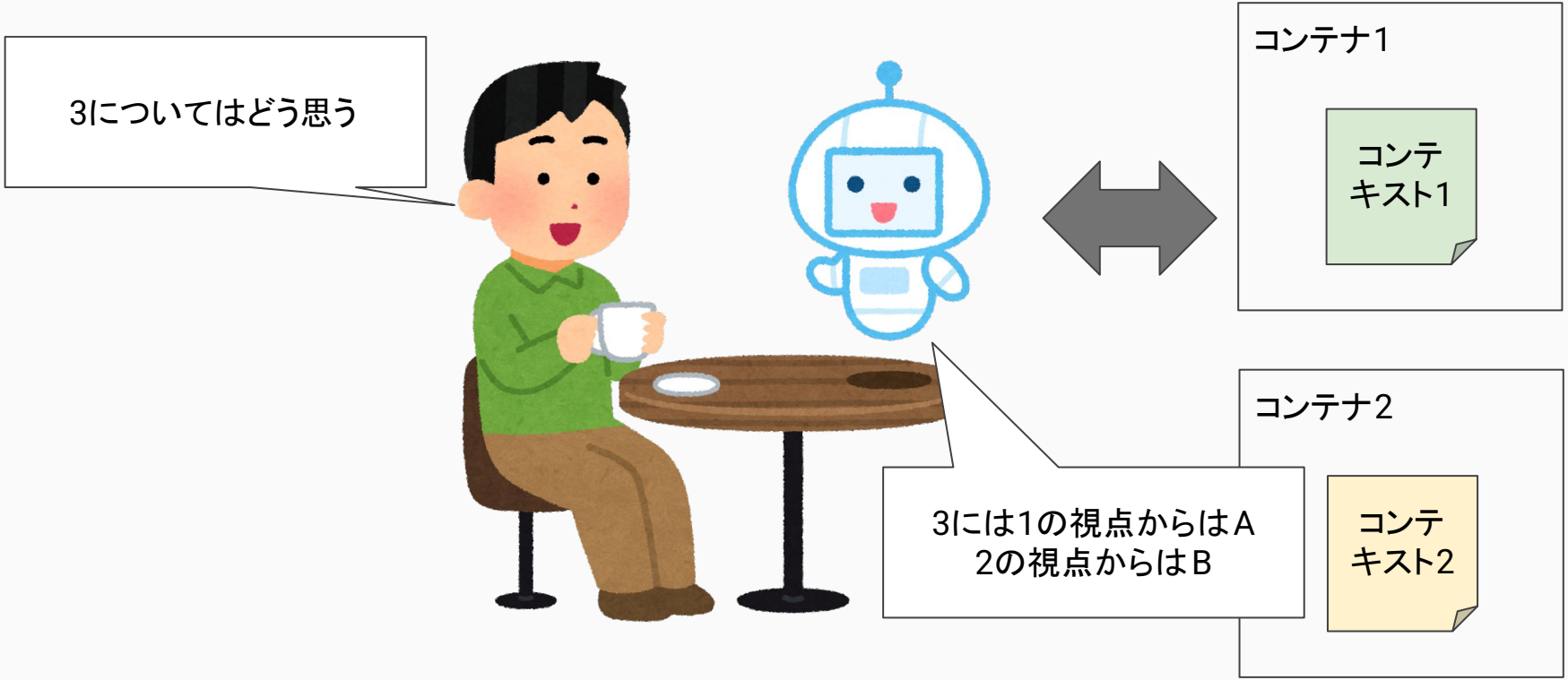
スキル＋コンテナ環境(複数視点)

・AI系のツールの課題



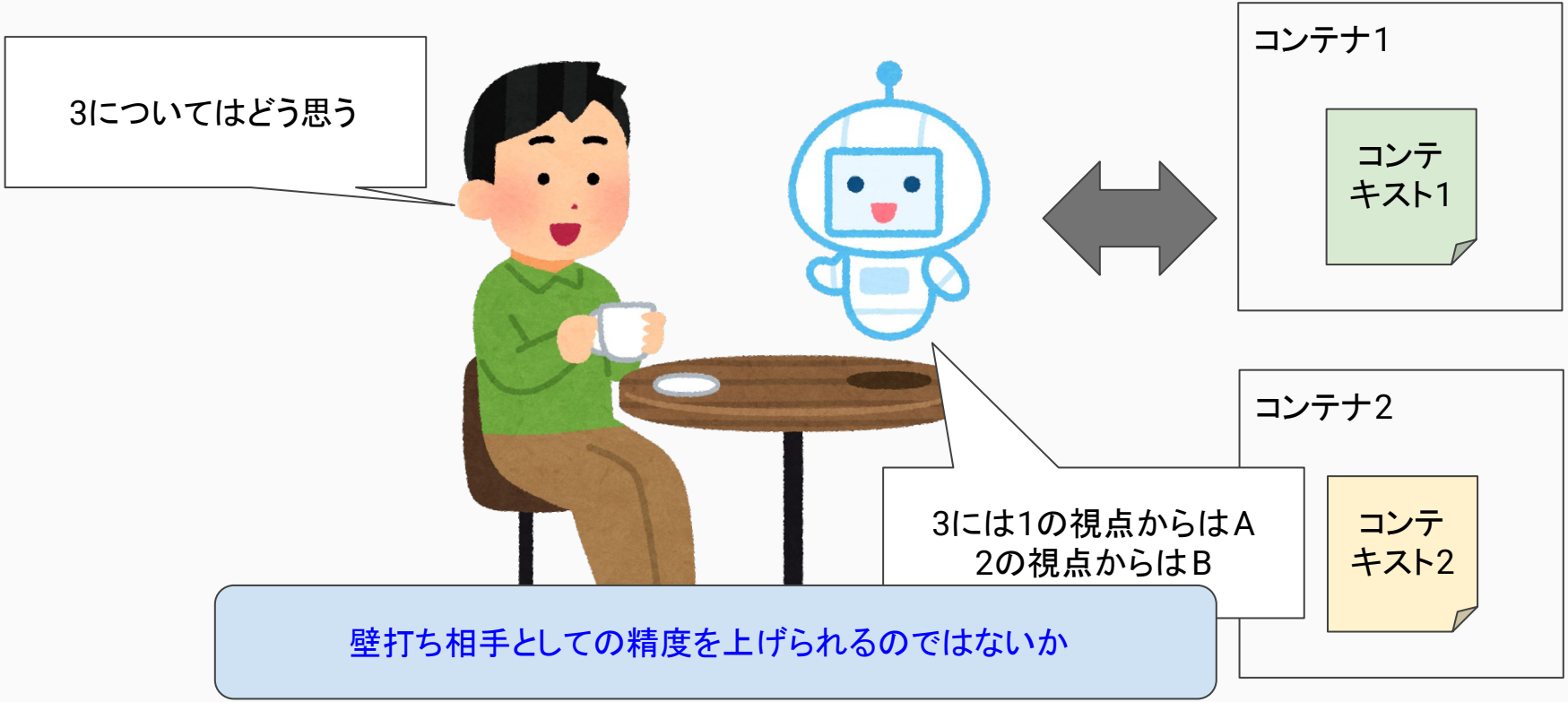
スキル＋コンテナ環境(複数視点)

・AI系のツールの課題



スキル＋コンテナ環境(複数視点)

・AI系のツールの課題



まとめ

まとめ

CLI系のツールの利用方法や自分の考え方を整理

手間がかかる部分を任せるための環境の整理が必要だと感じました

環境を構築することで効率化＋精度を上げる工夫があると感じました

できていない部分はQiitaなどでまとめていきたいと思います

参考資料

GeminiCLIを複数同時実行！Dockerで作るAI並行開発環境

https://qiita.com/ho-rai/items/a5fd9a111a348526ce6d