

クラウド **IDE** とエミュレータを利用した実機レス開発環境

アイコムシステック株式会社 庭野 正義

2020/6/12

背景と目的

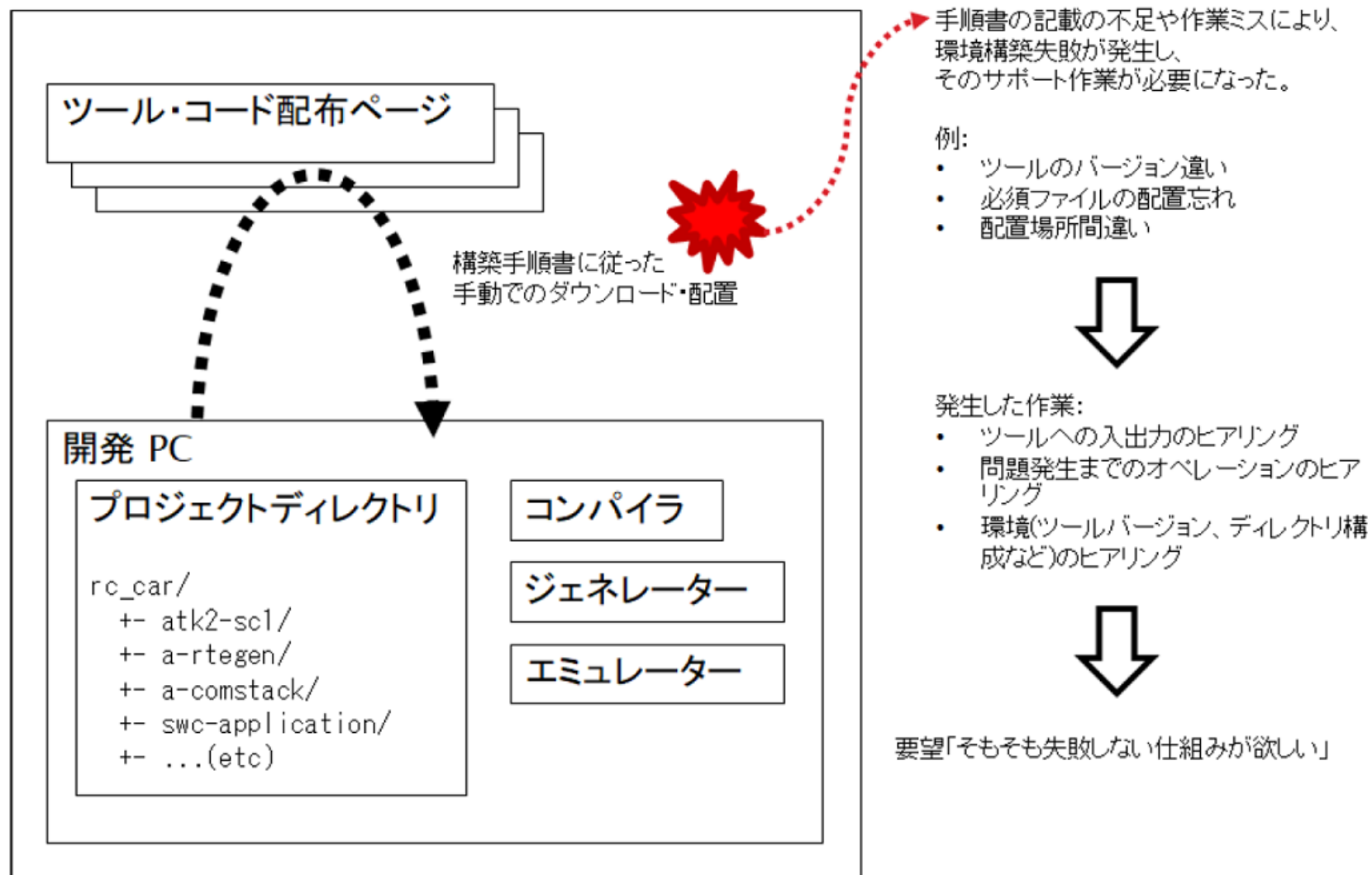


図1: 背景と目的

クラウド IDE 開発環境概要

今回提案したアイデアでは、クラウド IDE に Eclipse Che^{※1} を使用し検証した。

利用者が Eclipse Che 上でどのように操作するかを説明する。

※1: <https://www.eclipse.org/che/>

開発手順(I) -ワークスペースステンププレート選択-

Eclipse Che | New Workspace

← → ↻ 保護されていない通信 | che-che.10-109-64-96.nip.io/dashboard/#/create-workspace ☆ シークレット

Eclipse Che

Dashboard
Workspaces
Stacks
Factories
Administration

New Workspace **CREATE & OPEN**

NAME ? DEMO

SELECT	NAME	DESCRIPTION	REQUIRED MEMORY
STACK ?	TOPPERS/ATK2(V850)	TOPPERS/ATK2(V850)	1.6 GB
	TOPPERS/ATK2(Zynq UltraScale+ MPSoC Cortex-R5)	TOPPERS/ATK2(Zynq UltraScale+ MPSoC Cortex-R5)	1.6 GB
	TOPPERS/ASP3(ZYBO-Z7, ZYBO)	TOPPERS/ASP3(ZYBO-Z7, ZYBO)	1.6 GB
	(ZYBO-Z7, ZYBO)	TOPPERS/HRP3(ZYBO-Z7, ZYBO)	1.6 GB

開発を行いたい環境に対応したワークスペースステンププレートを選択すると、エディタ・コードジェネレータ・コンパイラ・エミュレータ等、必要なツール群一式が導入済みのワークスペースが生成される。

図2: クラウド IDE オーバービュー

開発手順(2) -プロジェクト編集-

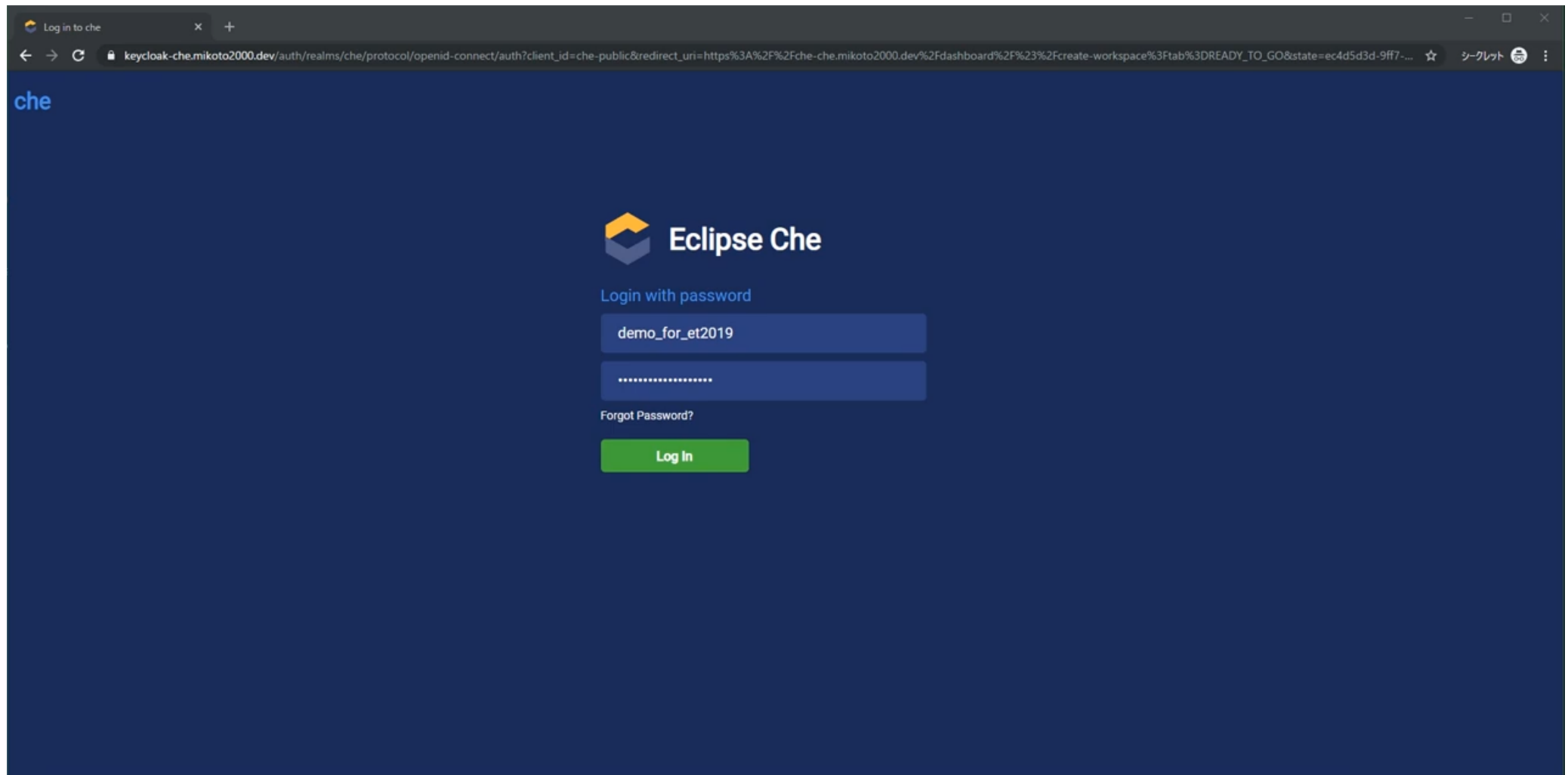
The image shows a cloud IDE interface with several callout boxes highlighting features:

- ファイルエクスプローラー** (File Explorer): Points to the left sidebar showing a project tree with files like `Os_Cfg.h`, `Os_Lcfg.c`, and `toppers_atk2.c`.
- エディタ(コード補完・定義ジャンプ)** (Editor): Points to the main code editor showing C code for `led_init(void)` with syntax highlighting and a tooltip for `LED接続ポート初期化`.
- ジェネレータ・コンパイラ** (Generator/Compiler): Points to the bottom terminal area showing compilation commands and output.
- マイコンエミュレータによるデバッグ** (Debugging with Microcontroller Emulator): Points to the bottom terminal area showing a debugger prompt `[DBG>]`.
- スイッチ情報表示ツール** (Switch Information Display Tool): Points to the right terminal area showing a digital state output: `**** DIGITAL STATE ****` with values for LED1-4, DSW1-4, and PSW1-2.

図3: クラウド IDE オーバービュー

クラウド IDE で TOPPERS カーネルを動かす

DEMO.



構成 -概要-

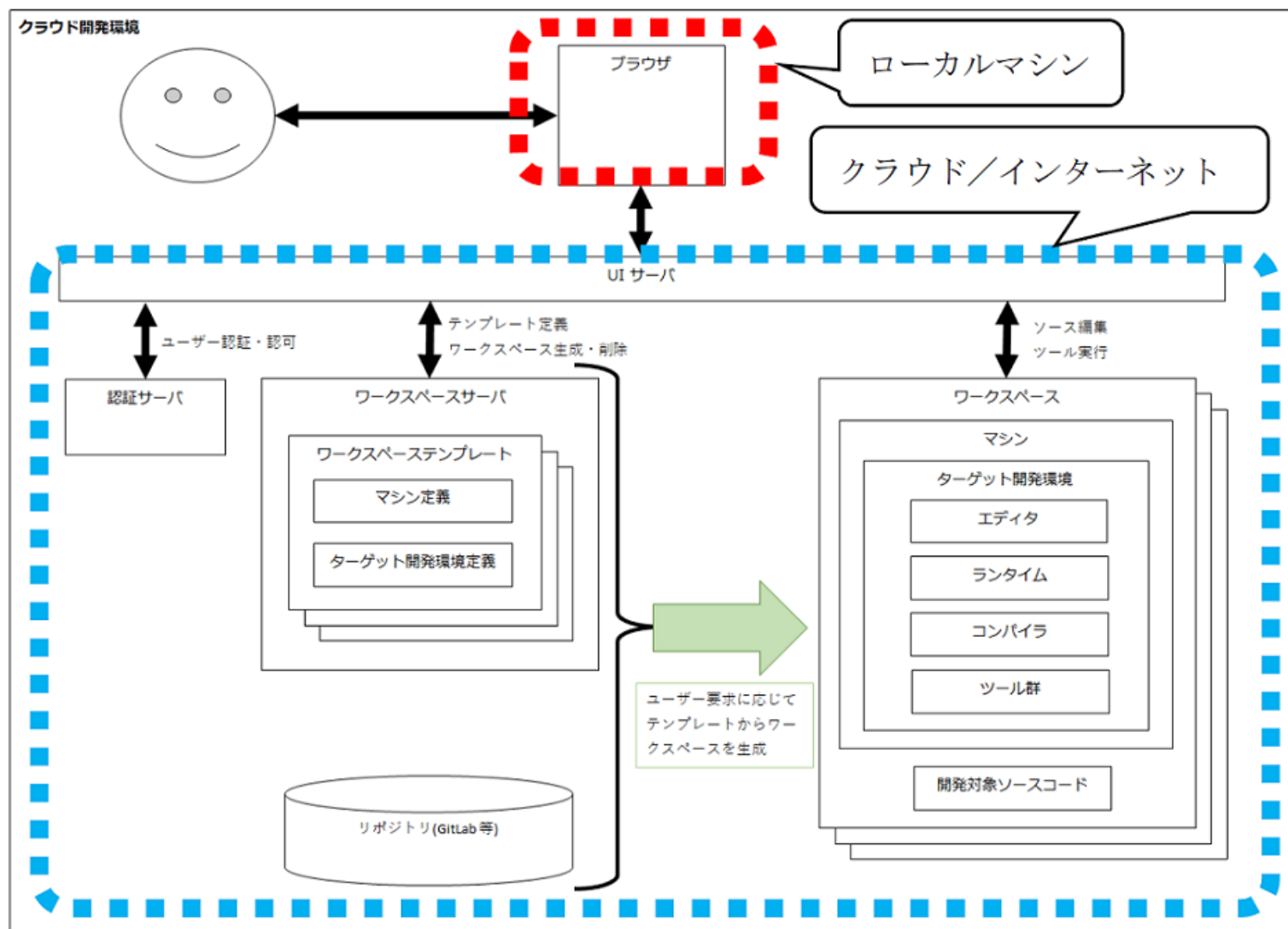


図4: クラウド IDE オーバービュー

独自開発環境構築例

DEMO.

[mikoto2000/che-aarch64-simple-sample](#) を見ながら、[che.openshift.io](#) 上へ独自開発環境を展開し、動作確認を行う。

今後の展望

1. CI サービスとの連携

- コンテナベースのCIサーバーを使用すれば、Eclipse Che用に作成したDockerコンテナを流用し、それをそのまま使ってCIを行うことができるはず
- CIサーバーとの連携のノウハウ・ガイドラインを整備すれば、より効率よくCIを実施することができる。そうすれば、教育だけでなく、開発にも本格的に導入していくことができるかもしれない

2. ワークスペーステンプレート集の作成

- 「独自開発環境構築例」で示した通り、開発環境を定義したyamlを公開しておけば、Eclipse Cheに読み込ませることができる
- 開発環境定義のyamlを集めたGitHubリポジトリを作成すれば、より簡単に試してもらうことができるようになると思われる

参考資料

■ 基本情報

- [Eclipse Che | Eclipse Next-Generation IDE for developer teams](#)
- [Introduction to Eclipse Che | Eclipse Che Documentation](#)

■ インストール

- [Running Che locally | Eclipse Che Documentation](#)
- [Running Eclipse Che on Kubernetes using Docker Desktop on macOS or Windows](#)
- [Windows の Docker Desktop\(with Kubernetes\) に Eclipse Che をデプロイした - mikoto2000 の日記](#)

■ 独自開発環境の追加

- [Overview | Eclipse Che Documentation](#)
- [Eclipse Che に自作の Che Plugin\(java12 & vscode-java\) を追加する - mikoto2000 の日記](#)
- [Eclipse Che に java12 のスタックを追加する - mikoto2000 の日記](#)
- [mikoto2000/che-aarch64-simple-sample: Eclipse Che の独自スタックデモ用プロジェクト。](#)

■ Eclipse Che が開発者向けサービスとして公開しているのでアカウントを作ればすぐに試せる

- [Hosted Che | Eclipse Che Documentation](#)