Visual Studio Codeで始めるGo言語

～高機能エディタとクラウドで効率的な開発を目指そう～

2019/10/24 飛松 清

自己紹介

- ソフトウェア・エンジニア
  - Webサービス、業務システム、ミドルウェアなど

- 著書
  - Go言語入門

Agenda

1. IDEとVisual Studio Code
2. Go言語の紹介
3. Visual Studio CodeとGo言語
4. リモートツールによる連携
5. DockerとAzure

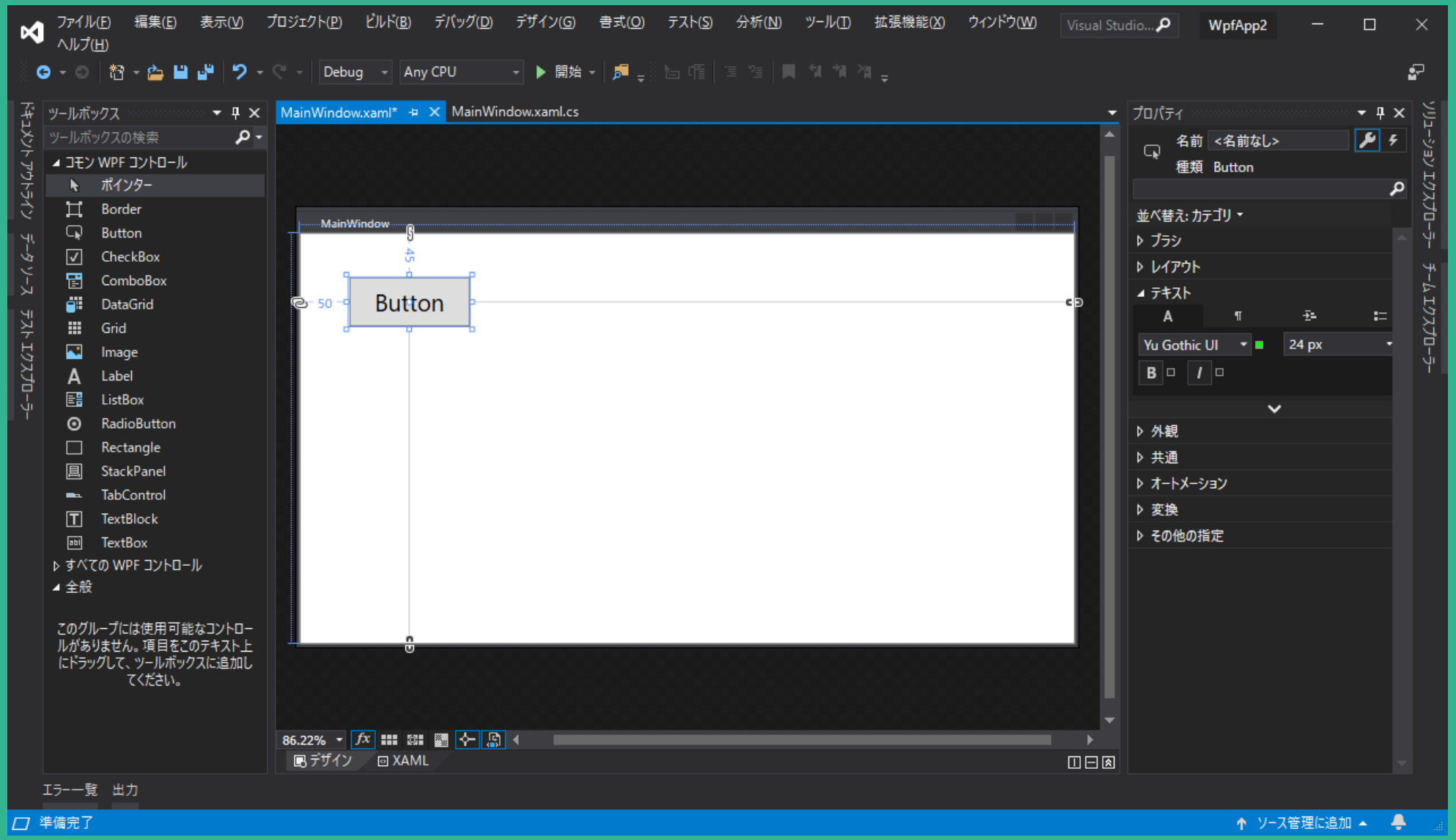
1. IDEとVisual Studio Code

IDE(統合開発環境)とは

- 開発に必要な各種機能を1つの環境（アプリ）で提供
  - コードエディタ
  - コンパイラ、デバッガ
  - バージョン管理

- GUIで操作、作成
  - ボタンクリックで実行
  - ドラッグ＆ドロップでボタン設置

IDEでGUI作成



(Visual studio)

Web開発の事情

- 最近のGUIはWebが主流
  Webはテキストベース(HTML+JavaScript)

- 1つの開発で複数言語を使用することも多い
  - フロントエンド：JavaScript
  - バックエンド：Go言語

- エディタ+拡張機能(プラグイン)は今でも人気
  - Vim(1991年〜)
  - Emacs(1972年〜)

Visual Studio Code(VS Code)の誕生

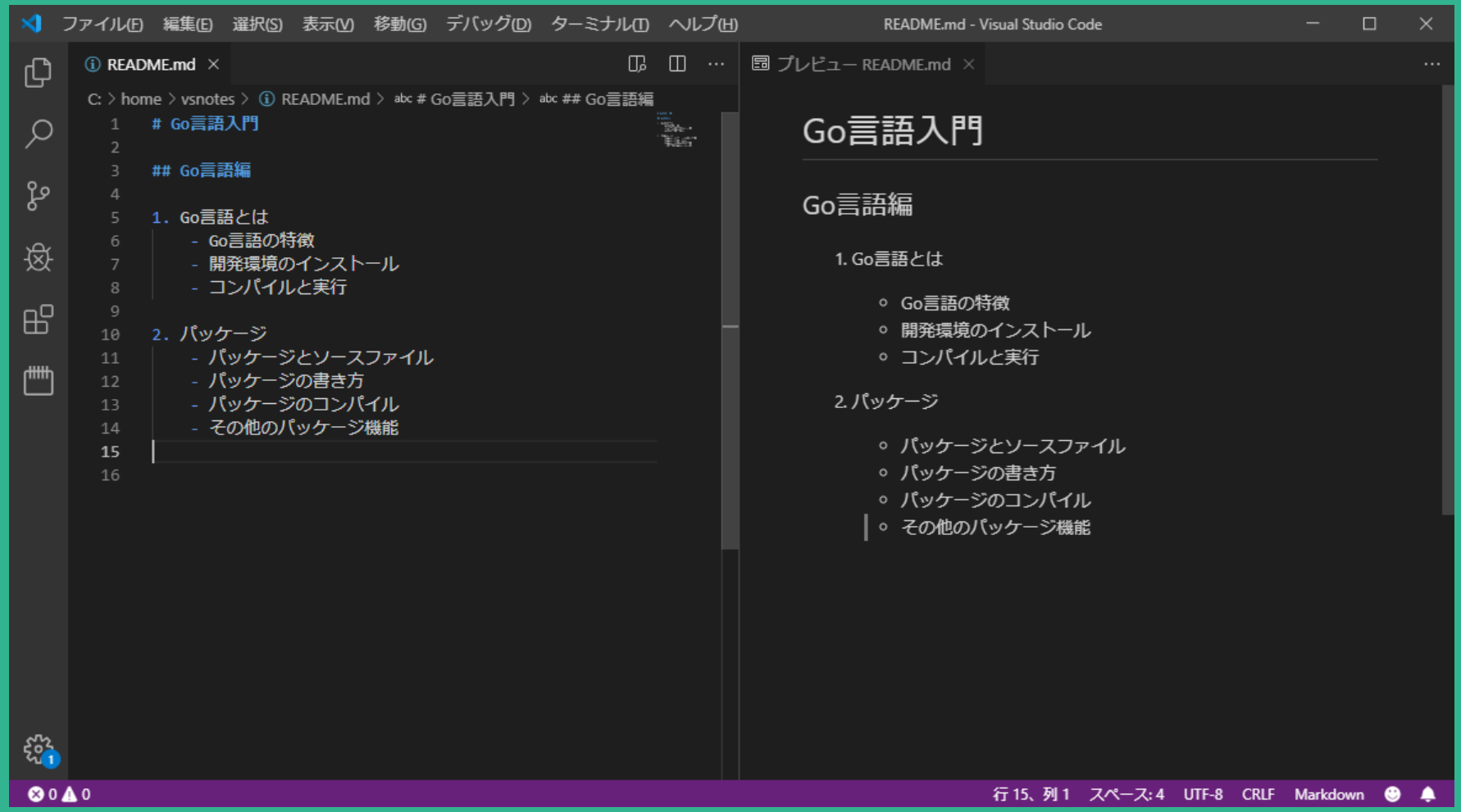
- 2015年にMicrosoftが公開
  - オープンソース
  - 従来のVisual Studioとは異なるもの

- マルチプラットフォーム対応
  - Windows, Mac, Linux

- Electronフレームワークを使用
  - ブラウザエンジン上で動作

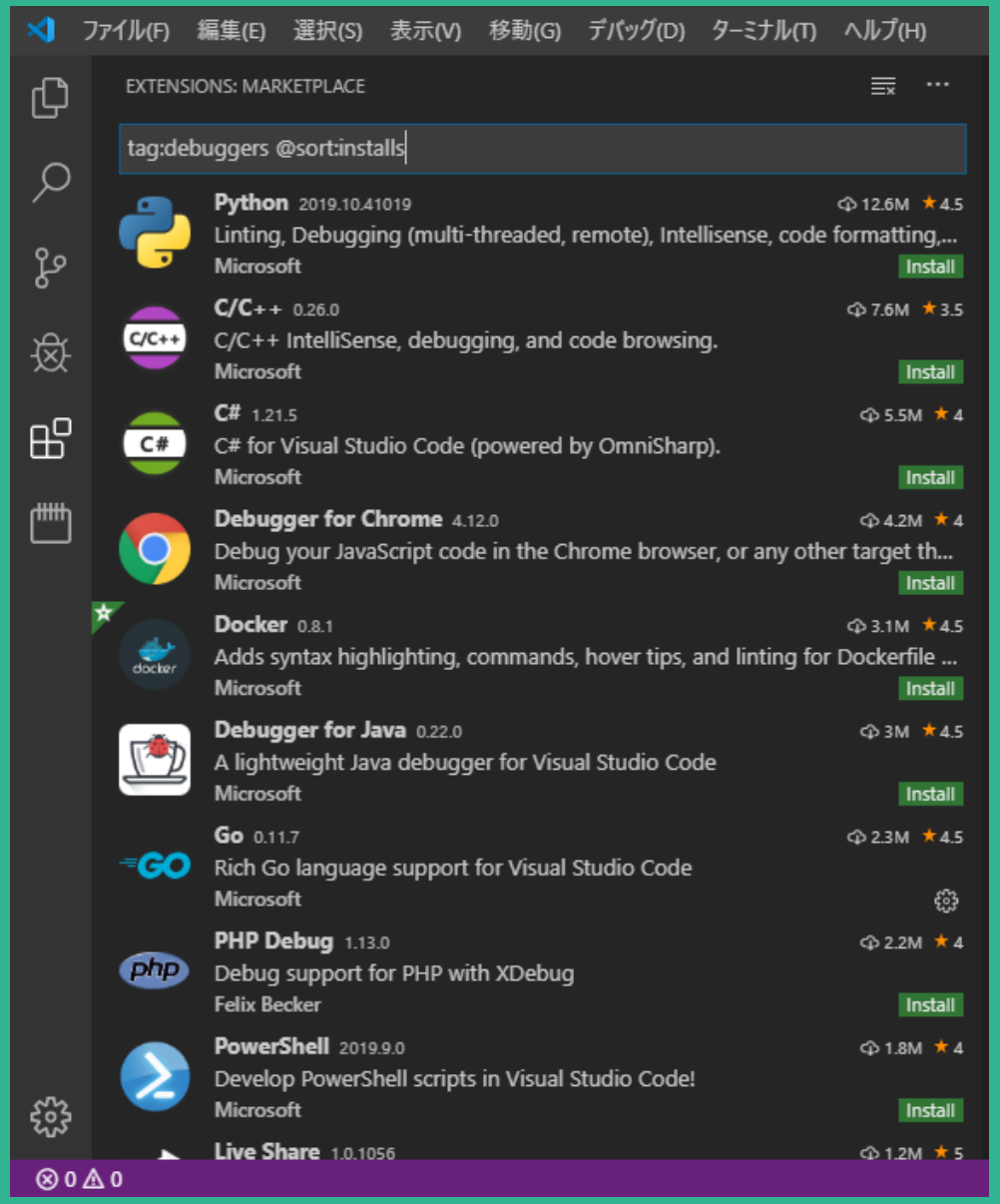
VS Codeの画面(Markdown)

- 左画面の内容が即座に右画面に反映



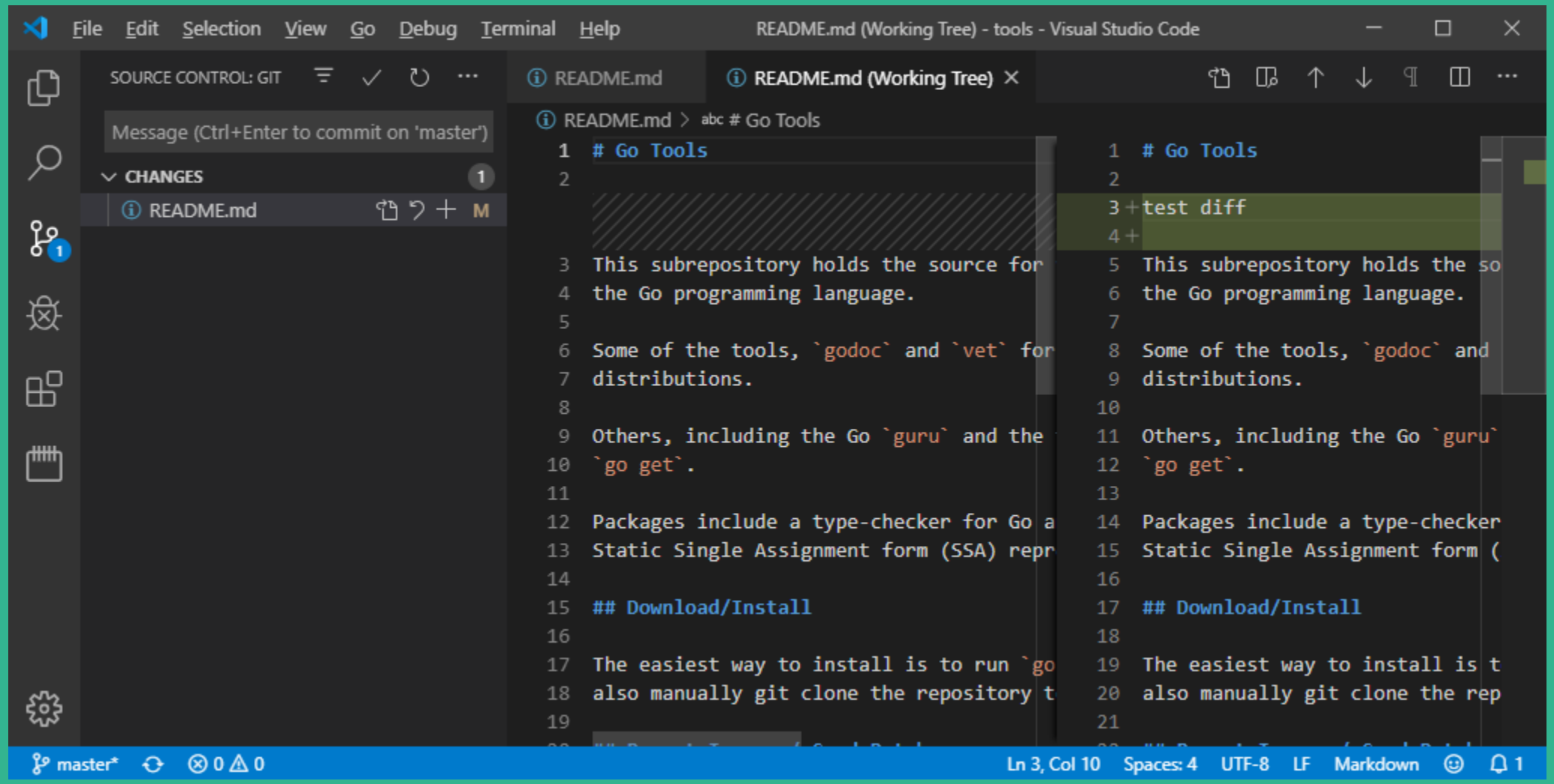
VS Codeの言語対応

- 各言語には拡張機能で対応
  - 複数言語で同時開発可能



バージョン管理

- Gitとdiffに標準対応



2. Go言語の紹介

Go言語の歴史

- 2009年：Googleが発表

- 2011年：Version 1.0 Release
  - 以降Version 1.x では後方互換性あり
  - 約半年ごとにマイナーバージョンアップ

- 2019年：Version 1.13 Release
  - Github Ranking Statsで4位

    (ref. https://madnight.github.io/githut/#/pull_requests/2019/3)

初期の設計者

- Ken Thompson
  - Unix, C言語
- Rob Pike
  - Unix, Plan9
- Robert Griesemer
  - V8(JavaScriptエンジン)

Go言語の開発目的

The goals of the Go project were to eliminate the slowness and clumsiness of software development at Google, and thereby to make the process more productive and scalable.

(ref. https://talks.golang.org/2012/splash.article)

Goプロジェクトの目標は、Googleでのソフトウェア開発の遅延と不器用さを排除し、それによって生産性と拡張性を高めることでした。

Go言語の特徴

- 静的型付け、コンパイル型言語
- Garbage Collection付き
- Windows, Mac, Linuxに対応
- ネイティブコード出力
- 並行処理

Go言語の特徴：ネイティブコード出力

- コンパイル結果は、ネイティブコード
  - Windowsの場合、1つのexeファイル

- 実行先PCに実行環境不要
  - exeファイルのみで実行可能
  - 配布しやすい、デプロイしやすい

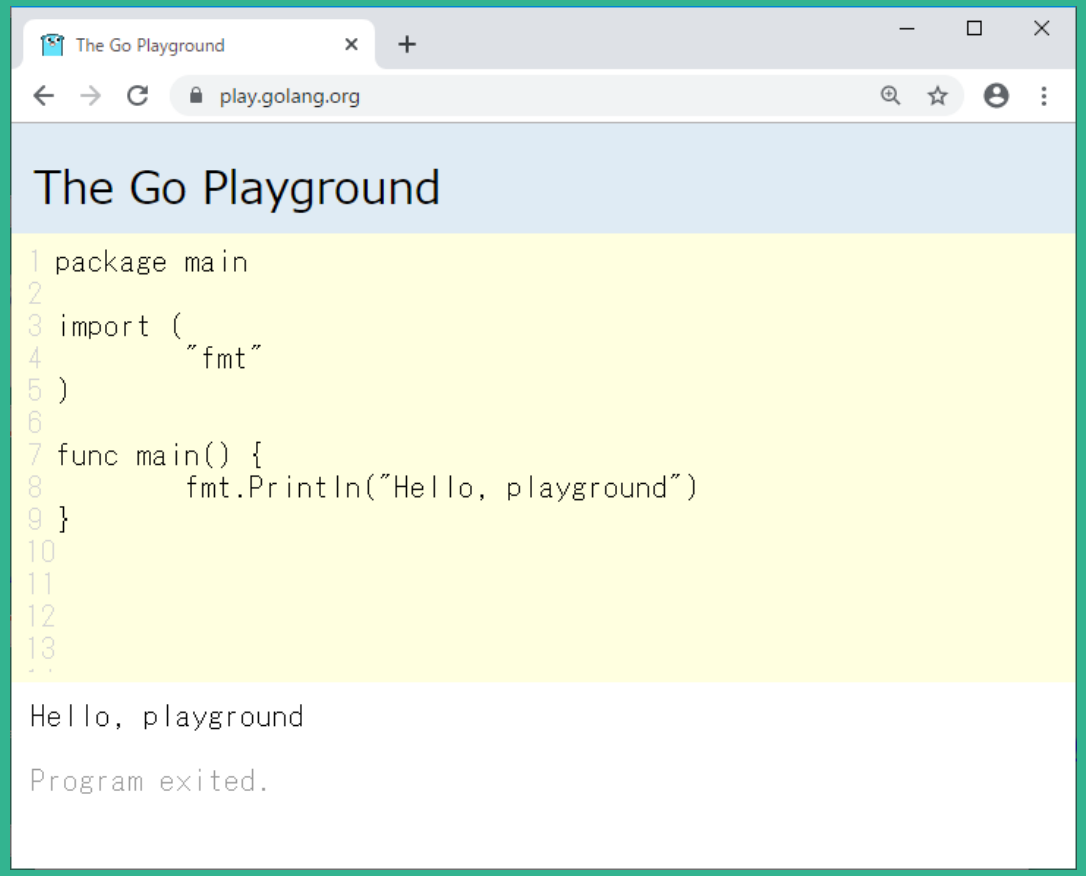
Go言語の特徴：並行処理

- 並行処理
  - 複数の処理を同時に実行すること
  - マルチコアCPUを有効利用できる

- Go-routine
  - Go言語組込みの並行処理機能
  - 簡単に記述可能：「go 関数名()」

Go Playground

- Webブラウザ上で実行できる環境



Go言語の事例

- ツール・インフラ系
  - Docker
  - Kubernetes
  - Terraform

- Webアプリケーション
  - https://github.com/golang/go/wiki/GoUsers

3. VS CodeとGo言語

なぜVS Codeなのか

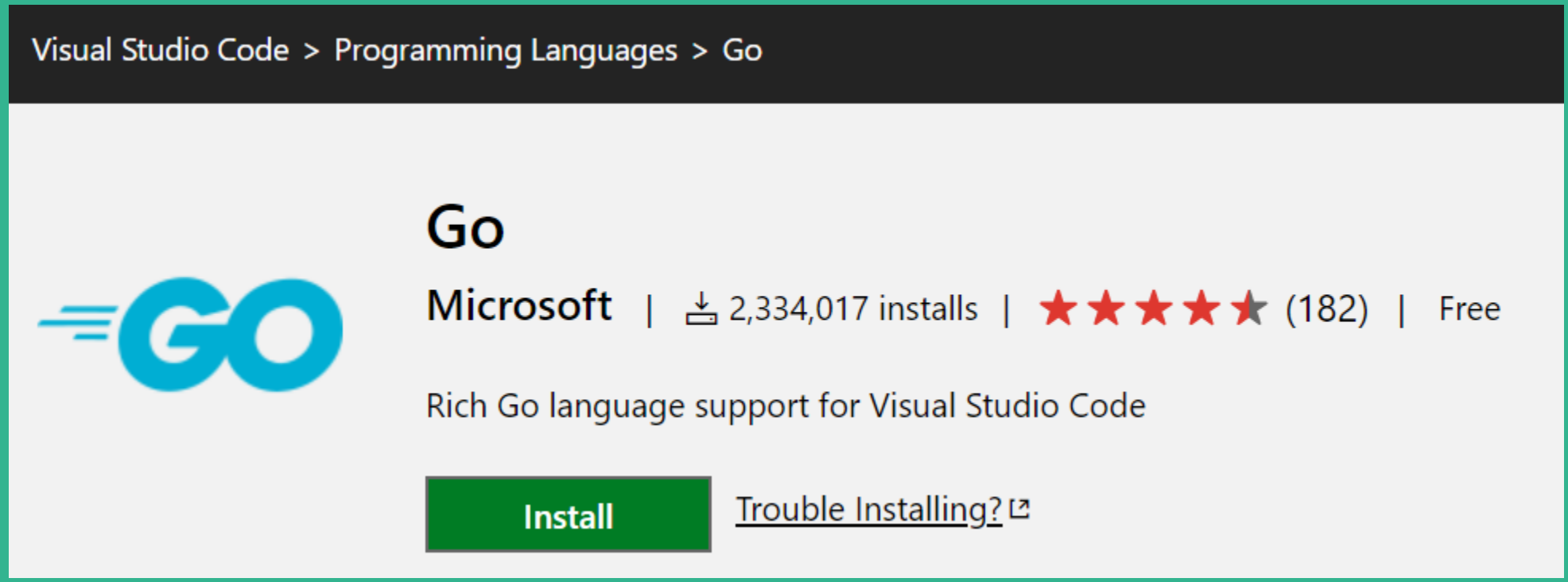
- Go言語に開発ツールはコマンドラインが多い
  - Unix系文化のため

- Windowsユーザは統合環境を好む傾向
  - GUIが主流
  - コマンドプロンプトが貧弱

- VS Codeの拡張機能は、コマンドラインツールを上手くラップする

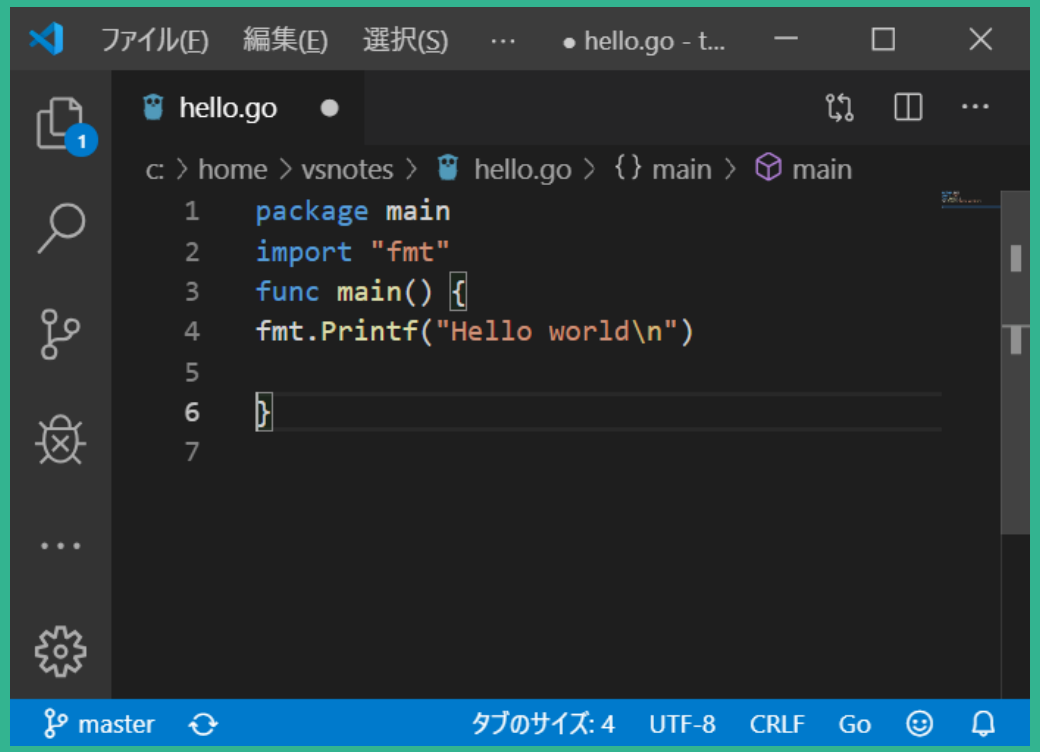
VS CodeのGo言語用拡張機能

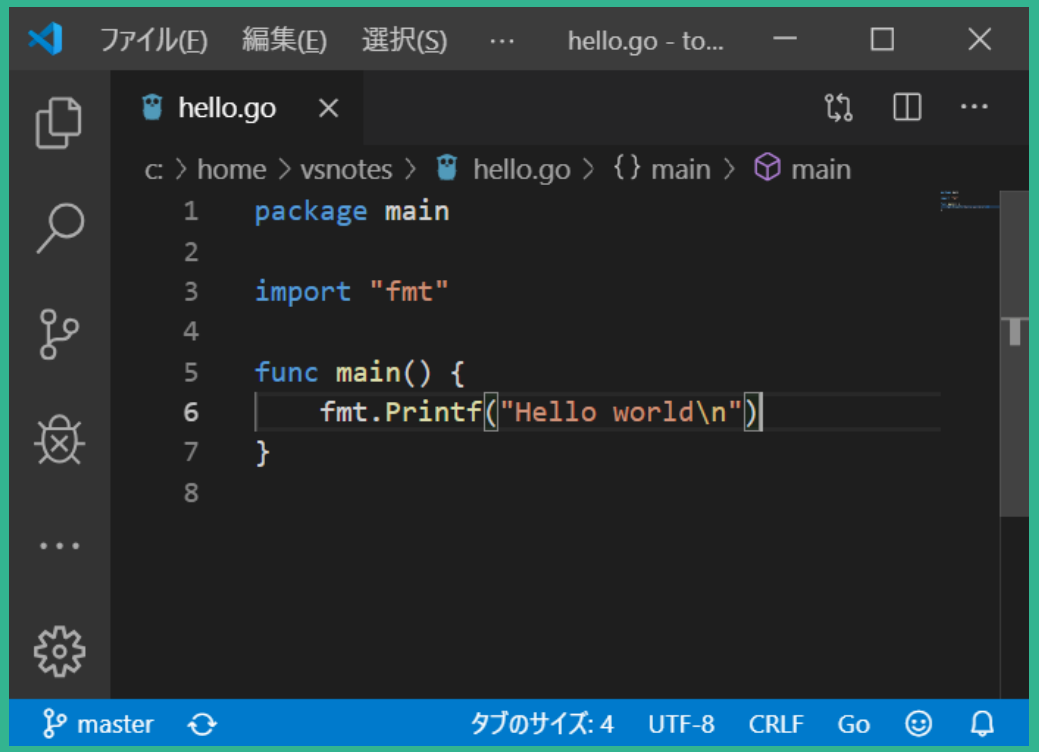
- Microsoftオフィシャルの拡張機能



コード整形

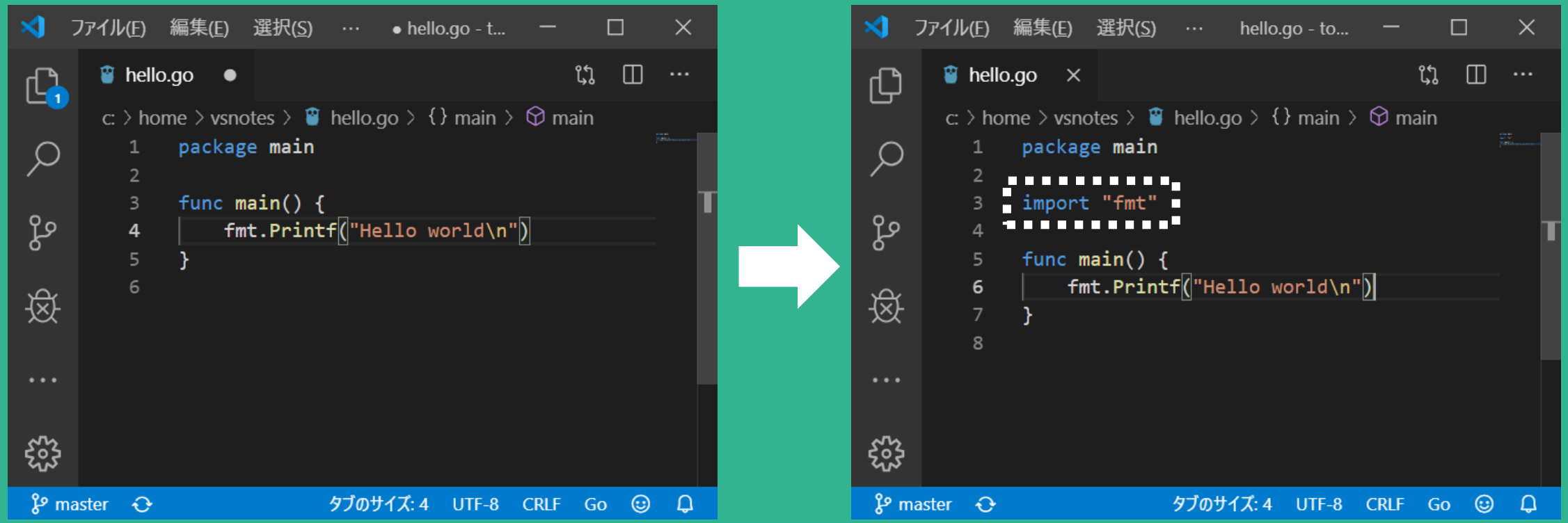
- gofmt ：Go言語の標準整形コマンドを使用
- 保存時に自動実行





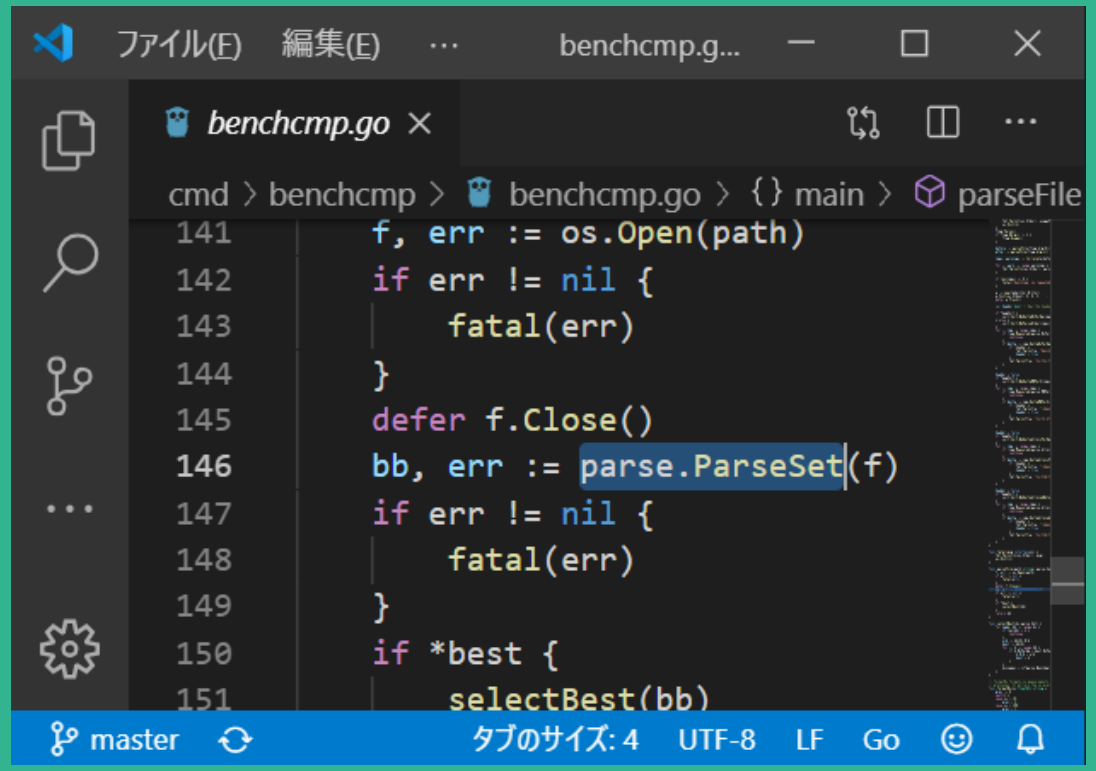
import文の自動追加

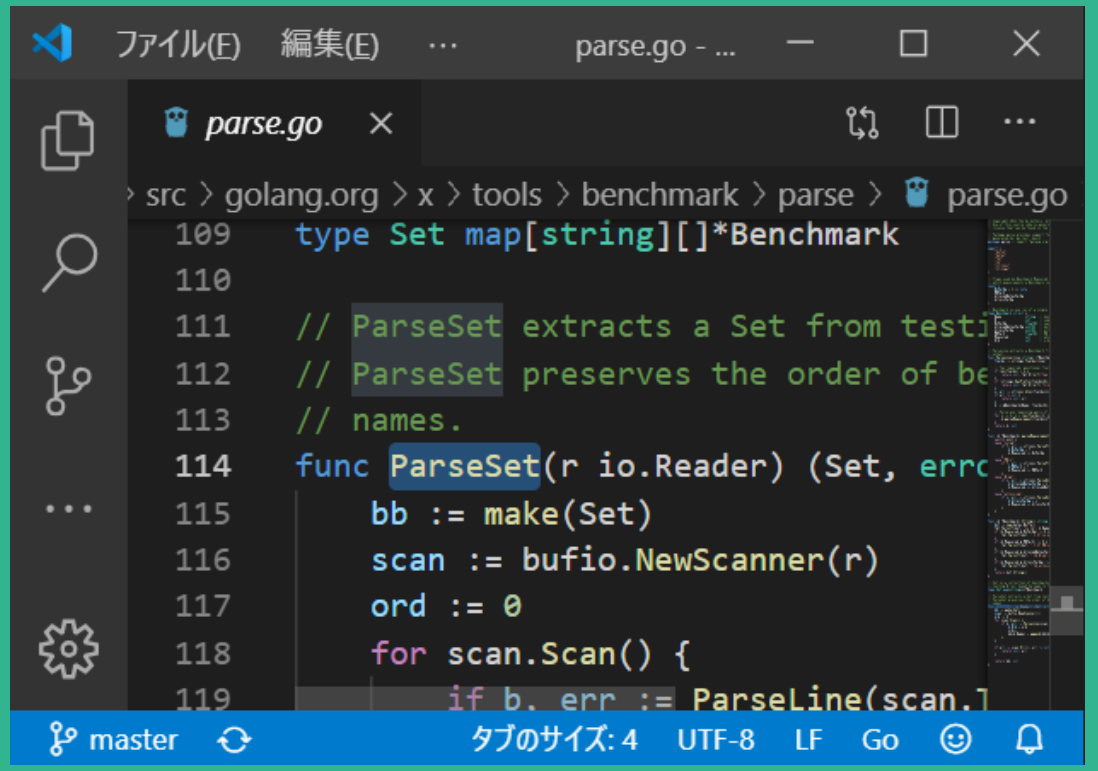
- goimportsコマンド: import文の自動追加
  同様に保存時に自動実行



コード・ナビゲーション

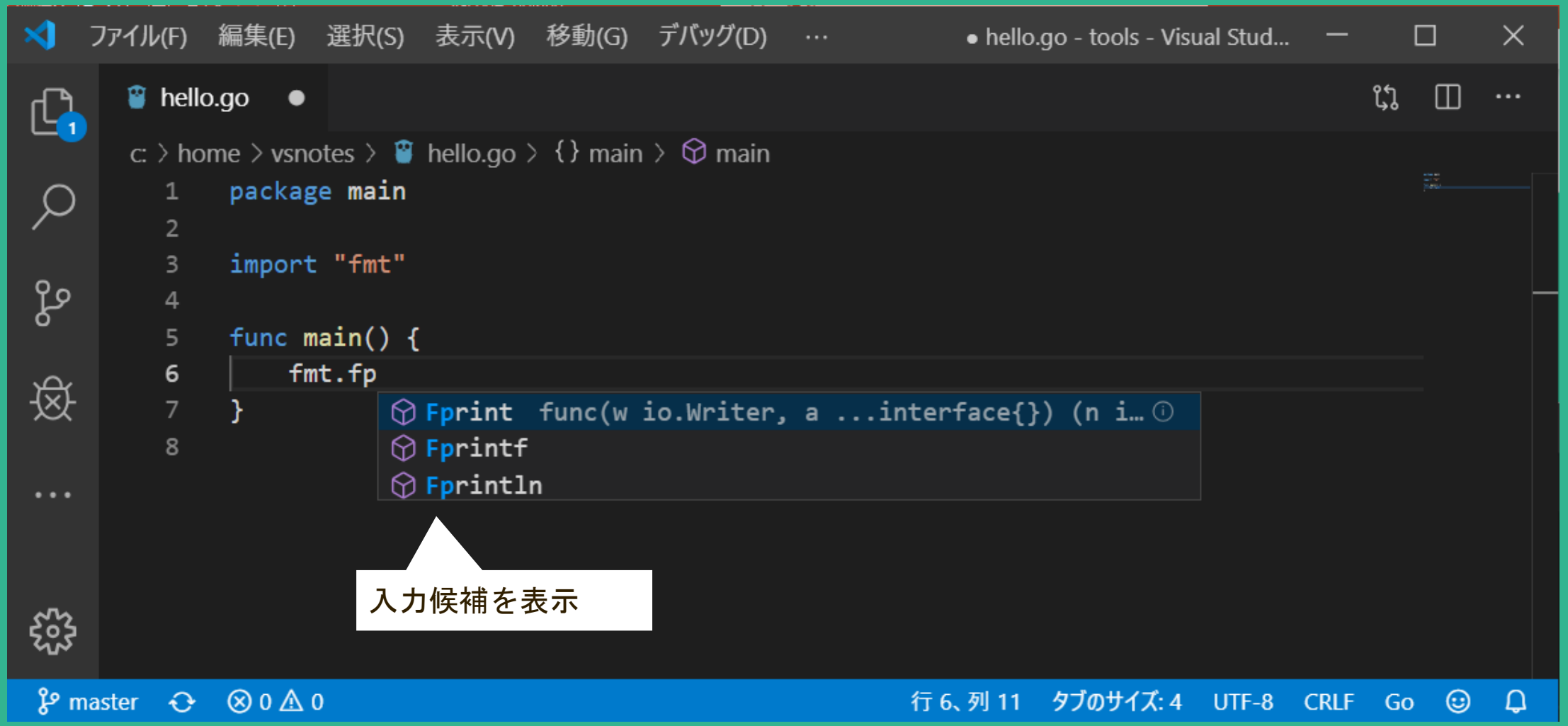
- 関数定義に移動





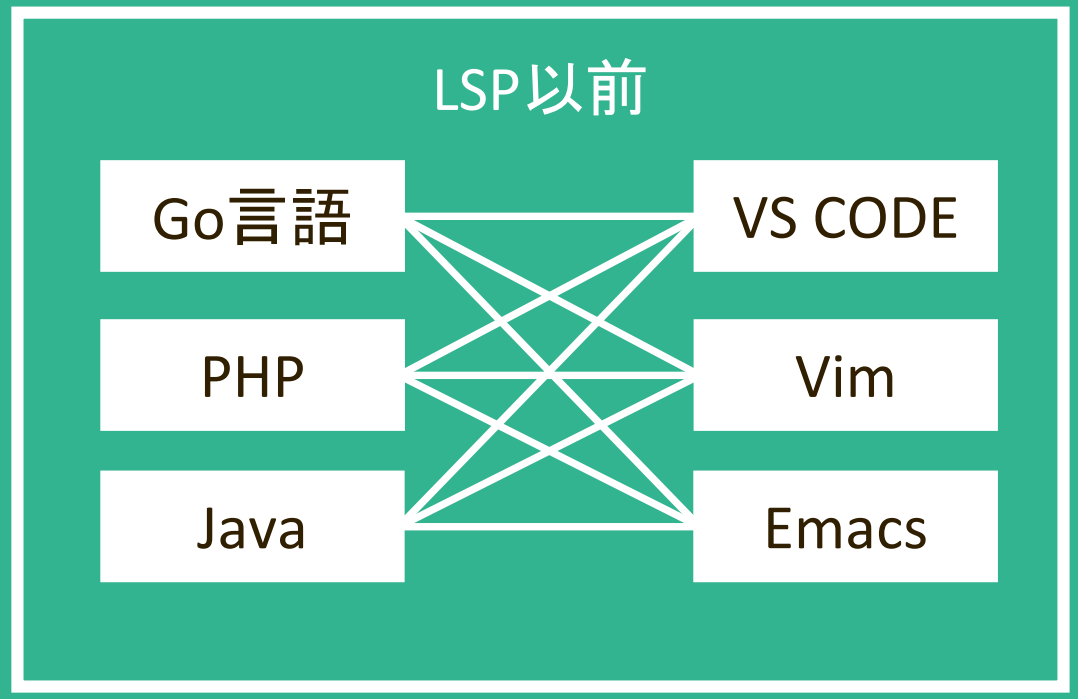
IntelliSense(コード補完)

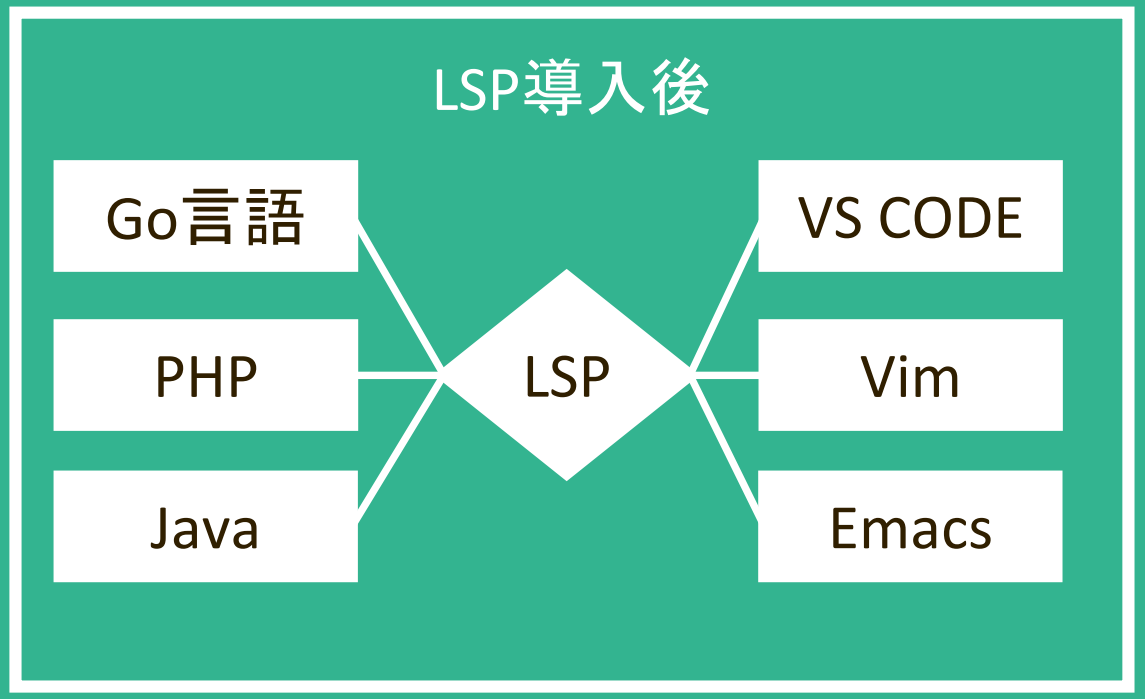
- 関数名などの入力候補を表示(goplsを使用)



Language Server Protocol(LSP)

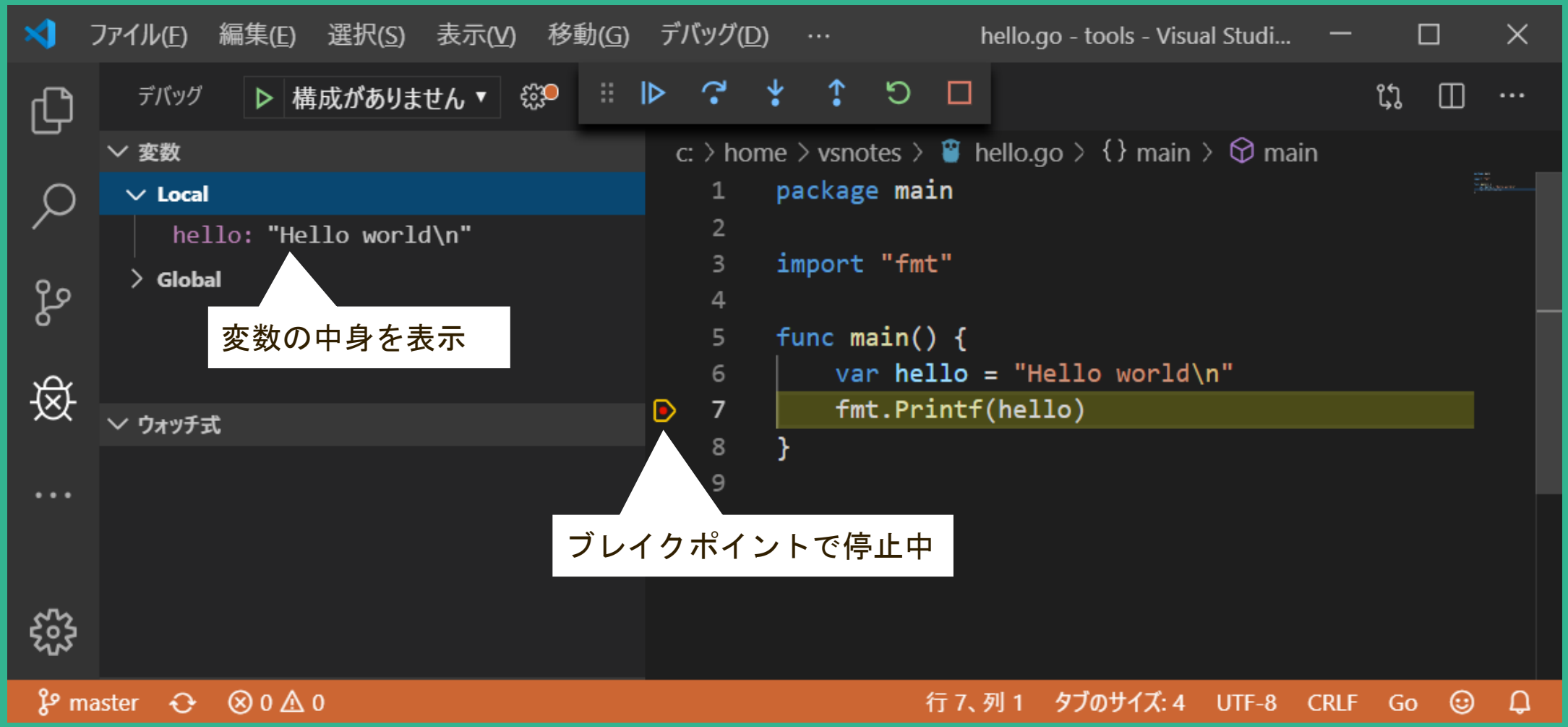
- コード補完などのデータを
  言語とエディタ(IDE)で連携するプロトコル





デバッガ

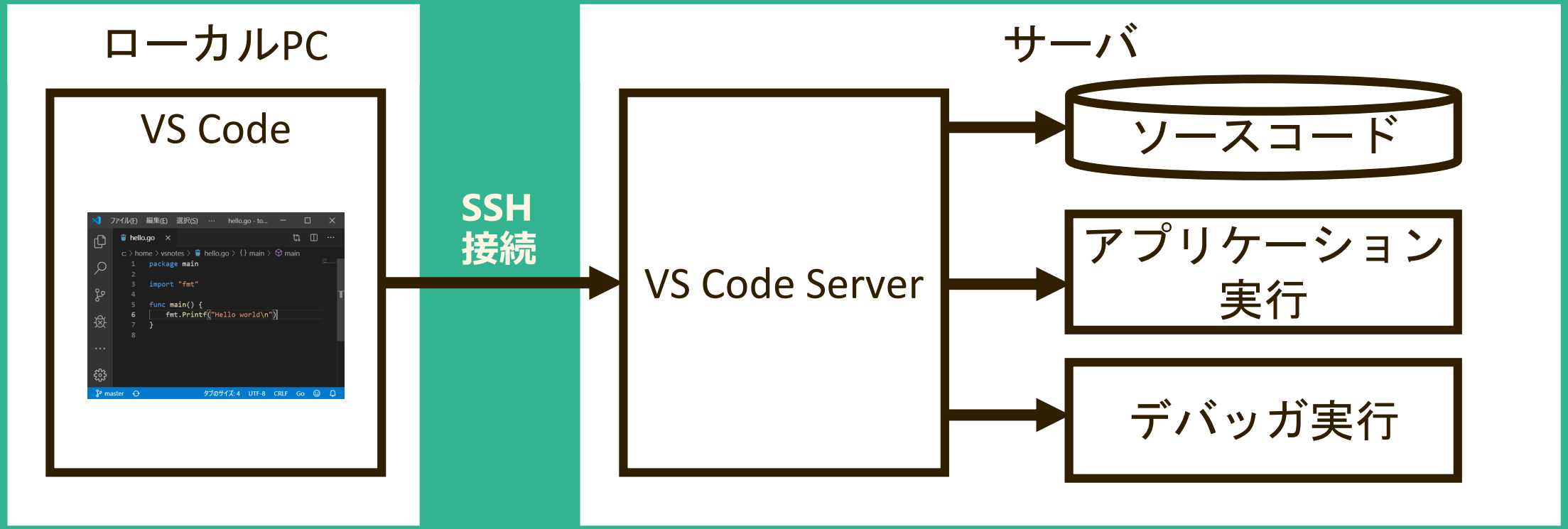
- Delve(Go言語デバッガ)をVS Codeに統合



4. リモートツールによる連携

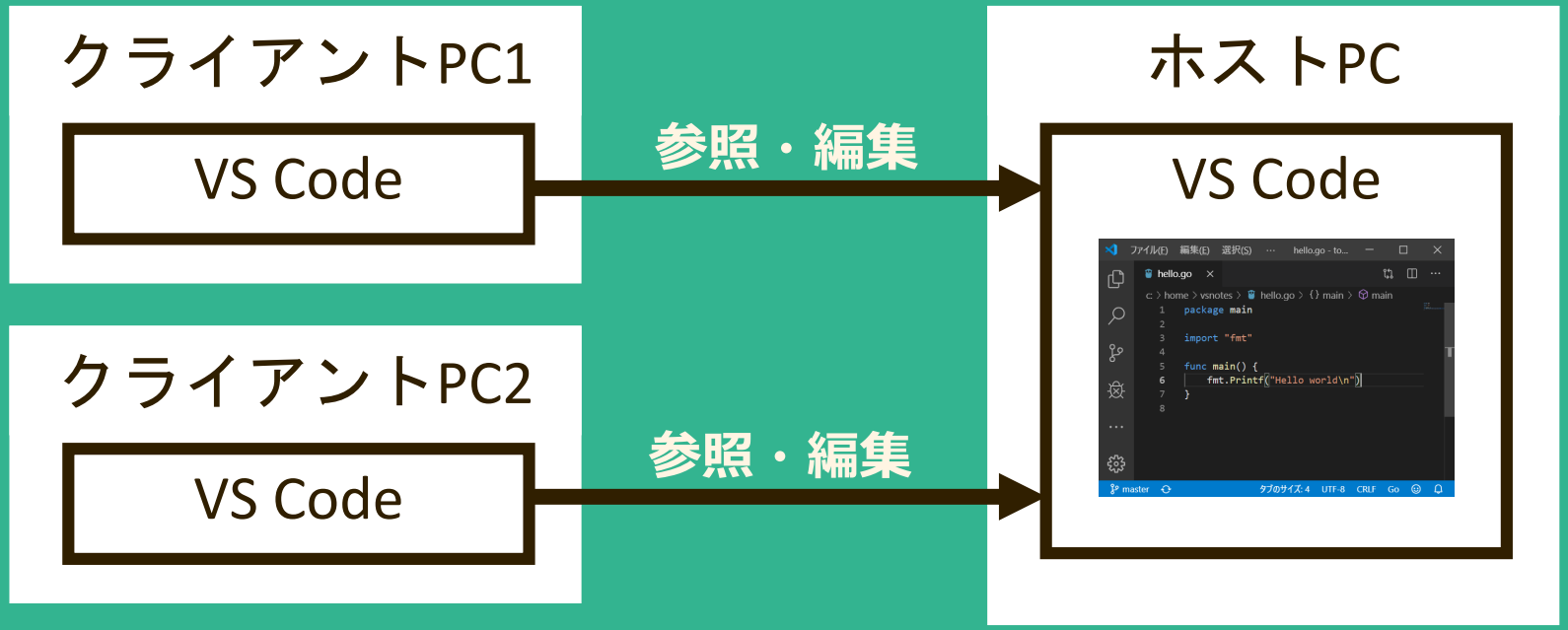
Remote Development

- リモート接続先のコード編集、実行、デバッグがローカルPCで可能



Visual Studio Live Share

- リアルタイムでの共同開発
- ホストPCのコードを、多人数で参照・編集できる
- Microsoft アカウントが必要



Visual Studio Live Share(画面)

(https://visualstudio.microsoft.com/ja/services/live-share/)

5. DockerとAzure

Docker

- Linuxの仮想環境（コンテナ）を提供するソフトウェア

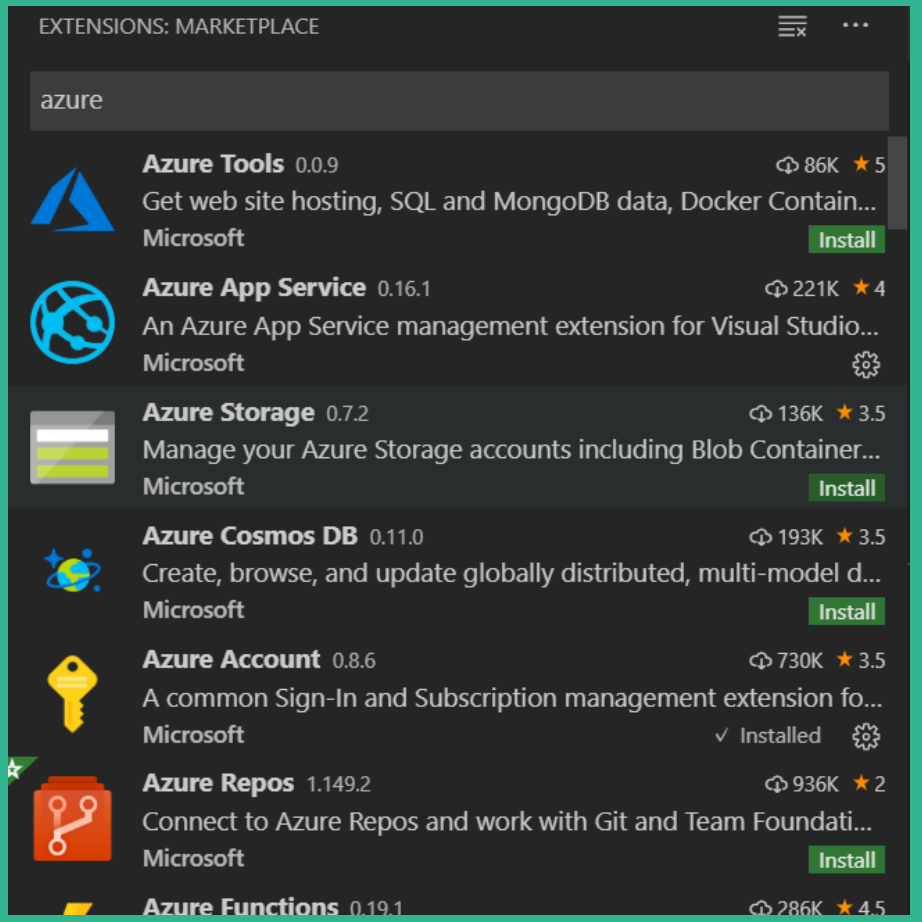
- ソフトウェア開発でDockerを使用するメリット
  - サーバ環境構築が容易
  - ライブラリ等のバージョン統一化

Microsoft Azure

- マイクロソフトのクラウドプラットフォーム

- PaaS(Platform as a Service)と
  IaaS (Infrastructure as a Service)がメイン

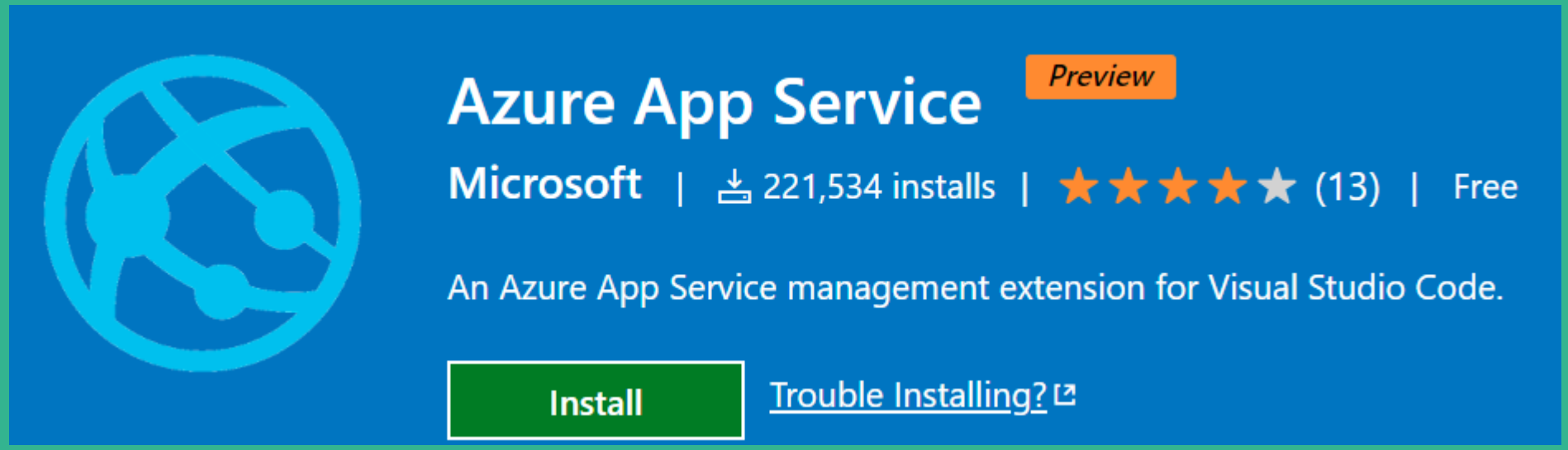
- 今回は**Azure App Service**と
  Azure Container Registoryを使用



Azure App Service

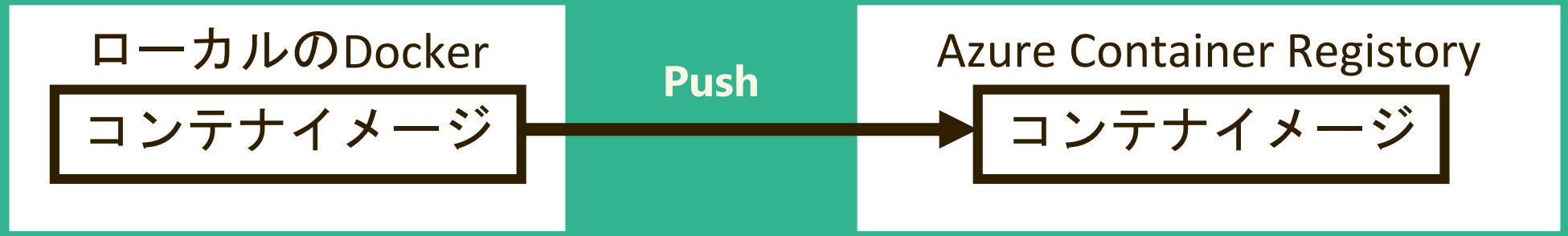
- Web系のフルマネージメント型PaaS

- Dockerのコンテナを設置可能

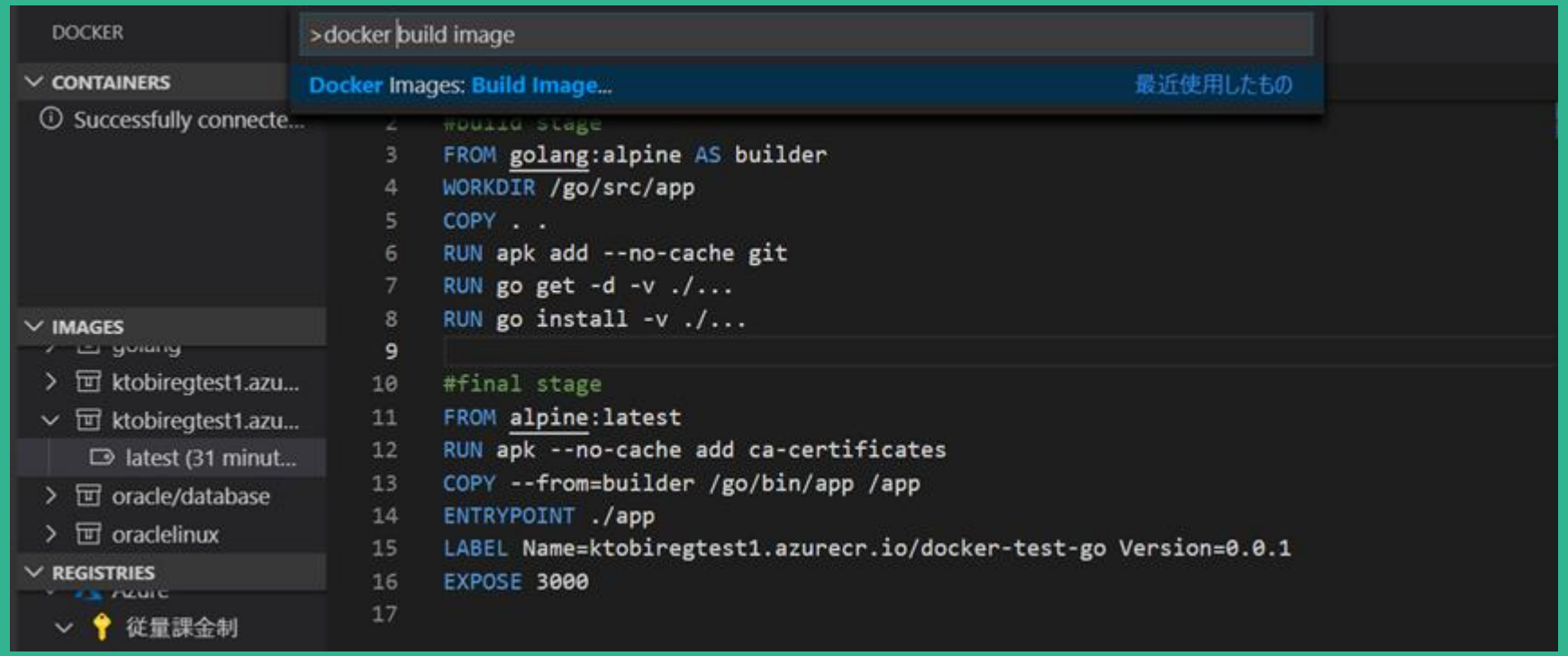


VS Codeでコンテナイメージ作成

- Docker コンテナイメージをローカルに作成
- ローカルのコンテナイメージをAzure Container RegistryにPush
- Azure Container Registory
  - Azureで使用するDockerのコンテナイメージを格納

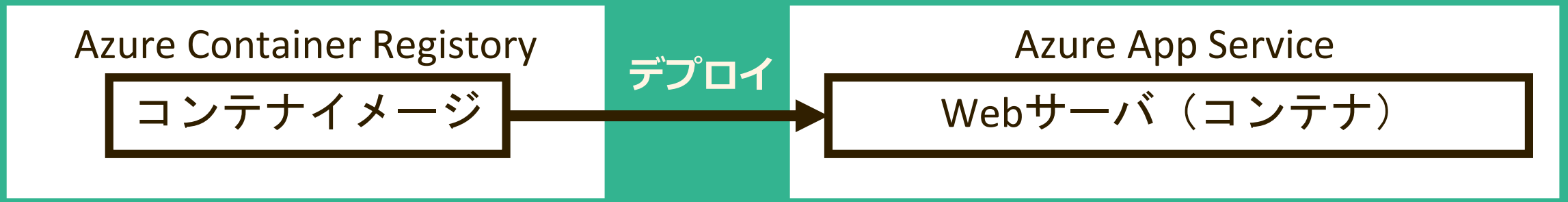


VS Codeでコンテナイメージ作成(画面)

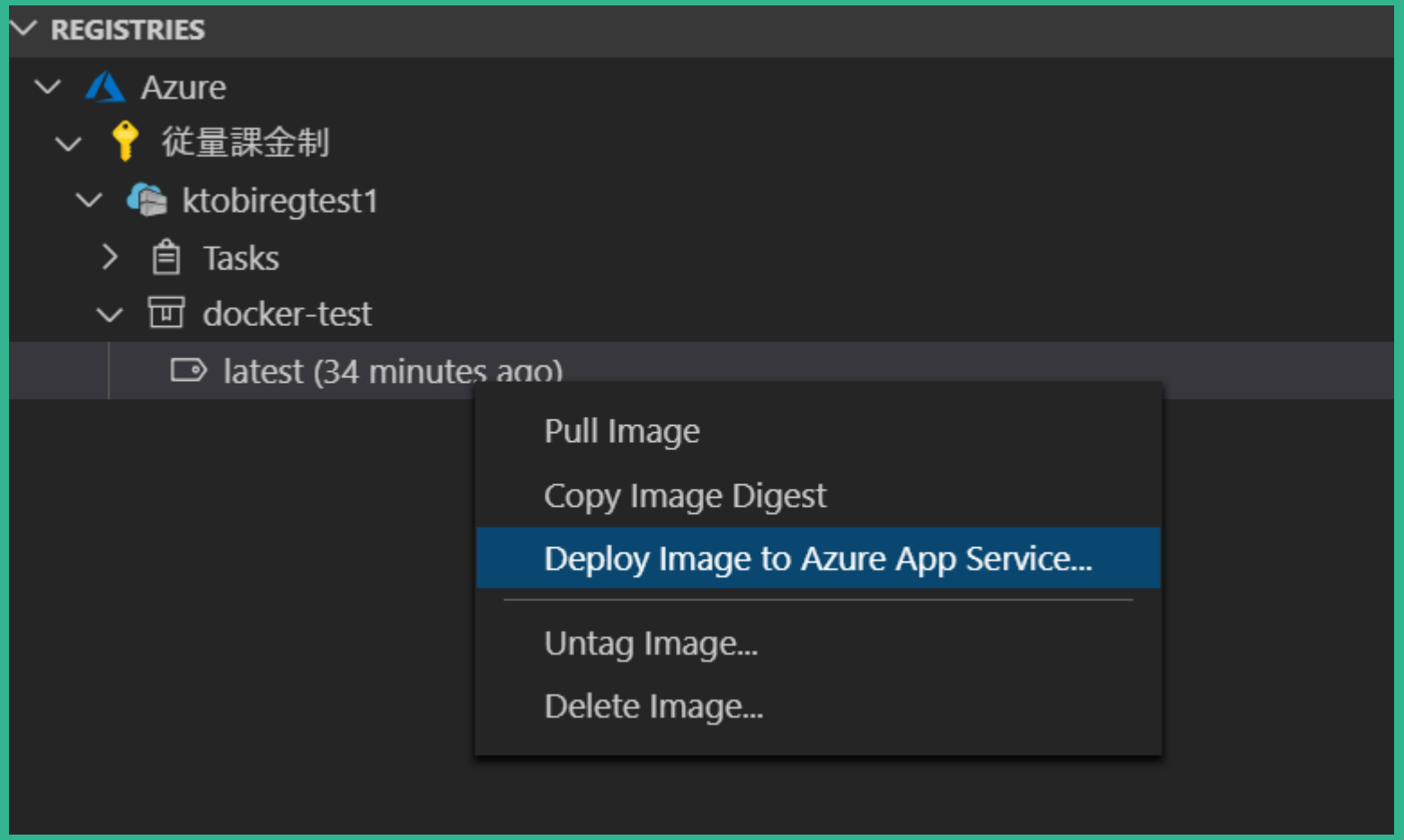


VS Codeでデプロイ

- Azure Container Registoryのコンテナイメージを
  Azure App Serviceにデプロイ



VS Codeでデプロイ（画面）



まとめ

- VS Code上でのGo言語開発
- リモートツールの使用
- AzureによるクラウドとDocker連携

ご清聴ありがとうございました