

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

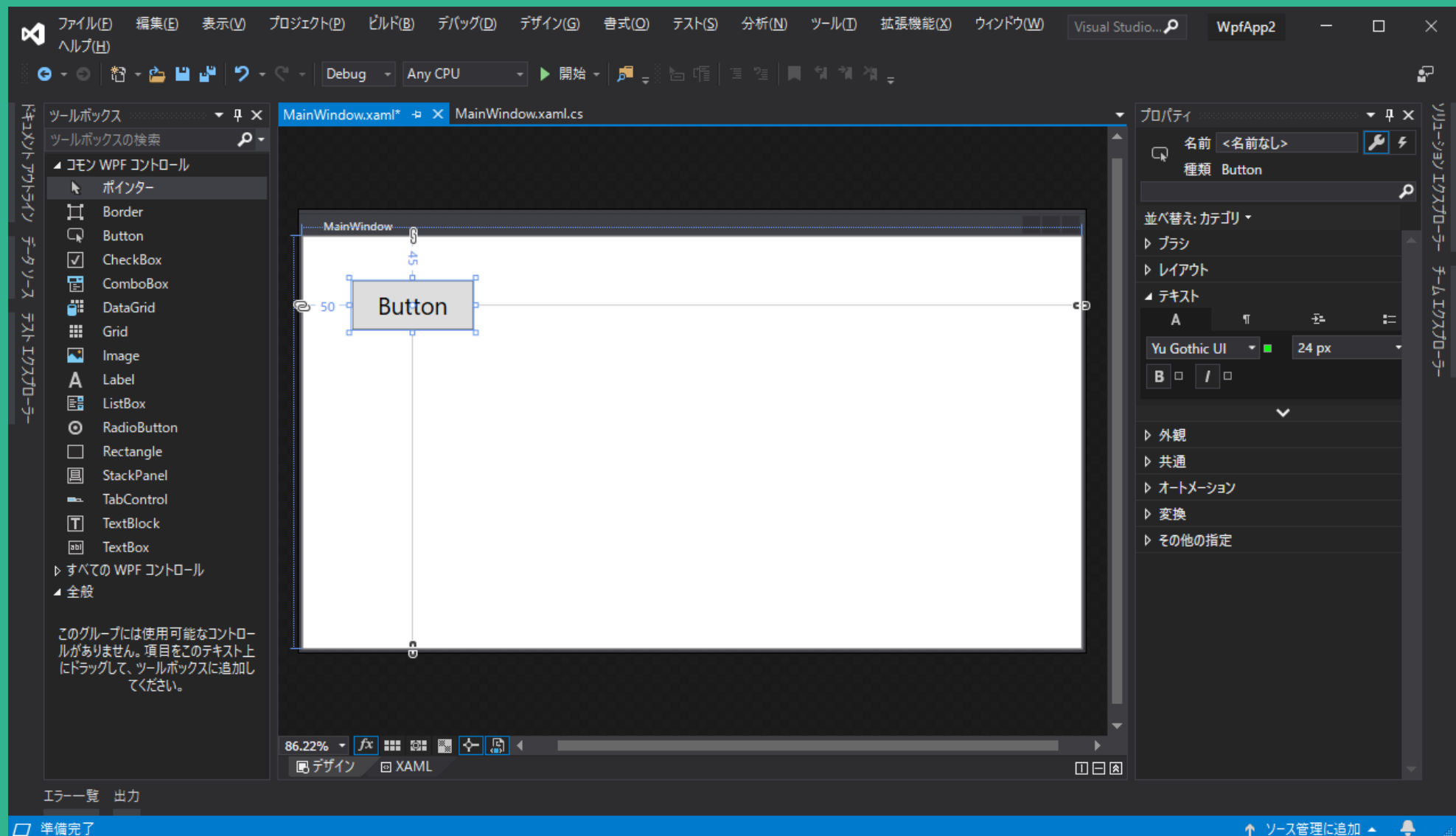
1. IDEとVisual Studio Code
2. Go言語の紹介
3. Visual Studio CodeとGo言語
4. リモートツールによる連携
5. DockerとAzure

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

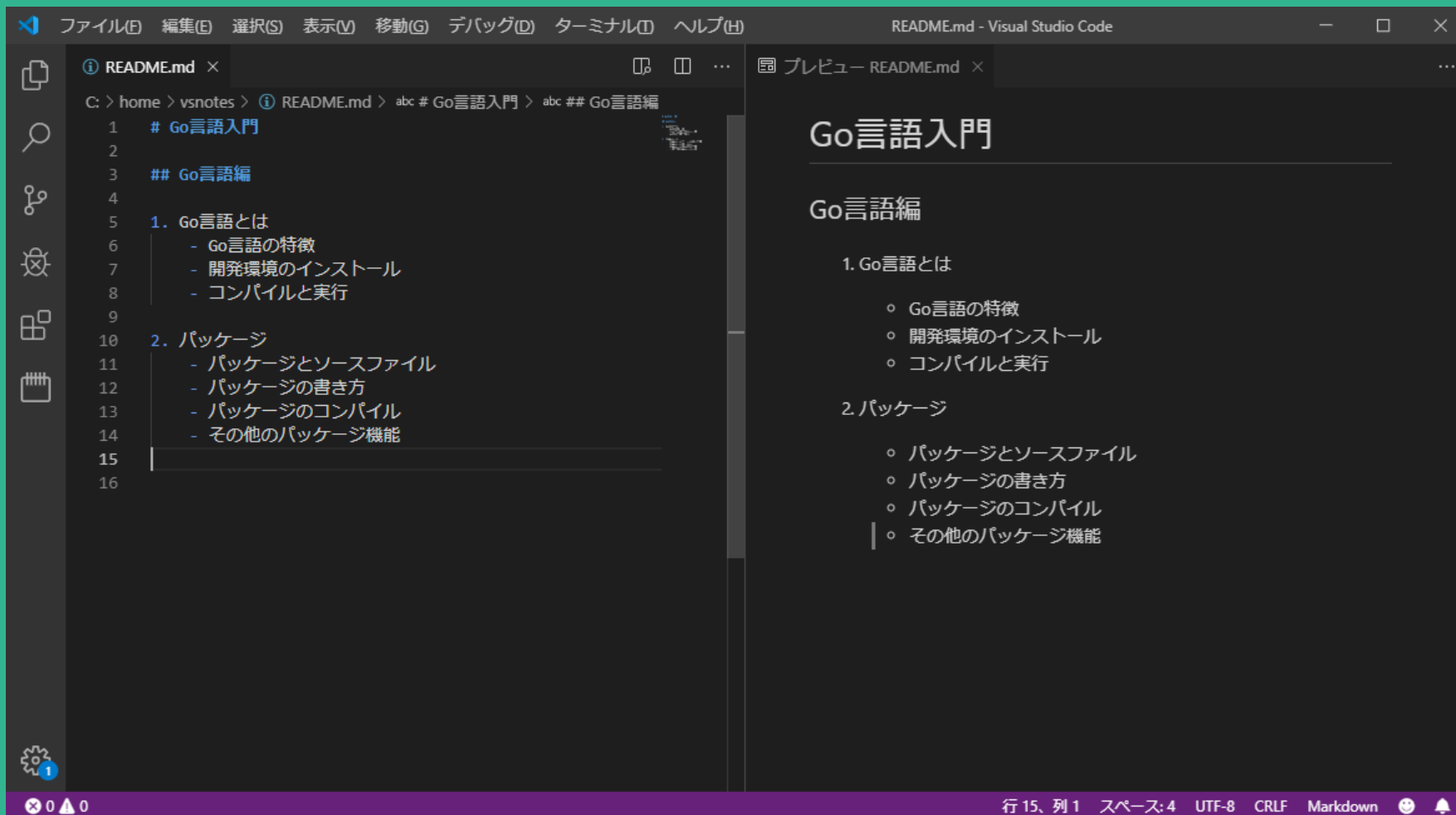
- 最近のGUIはWebが主流
Webはテキストベース(HTML+JavaScript)
- 1つの開発で複数言語を使用することも多い
 - フロントエンド：JavaScript
 - バックエンド：Go言語
- エディタ+拡張機能(プラグイン)は今でも人気
 - Vim(1991年～)
 - Emacs(1972年～)

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

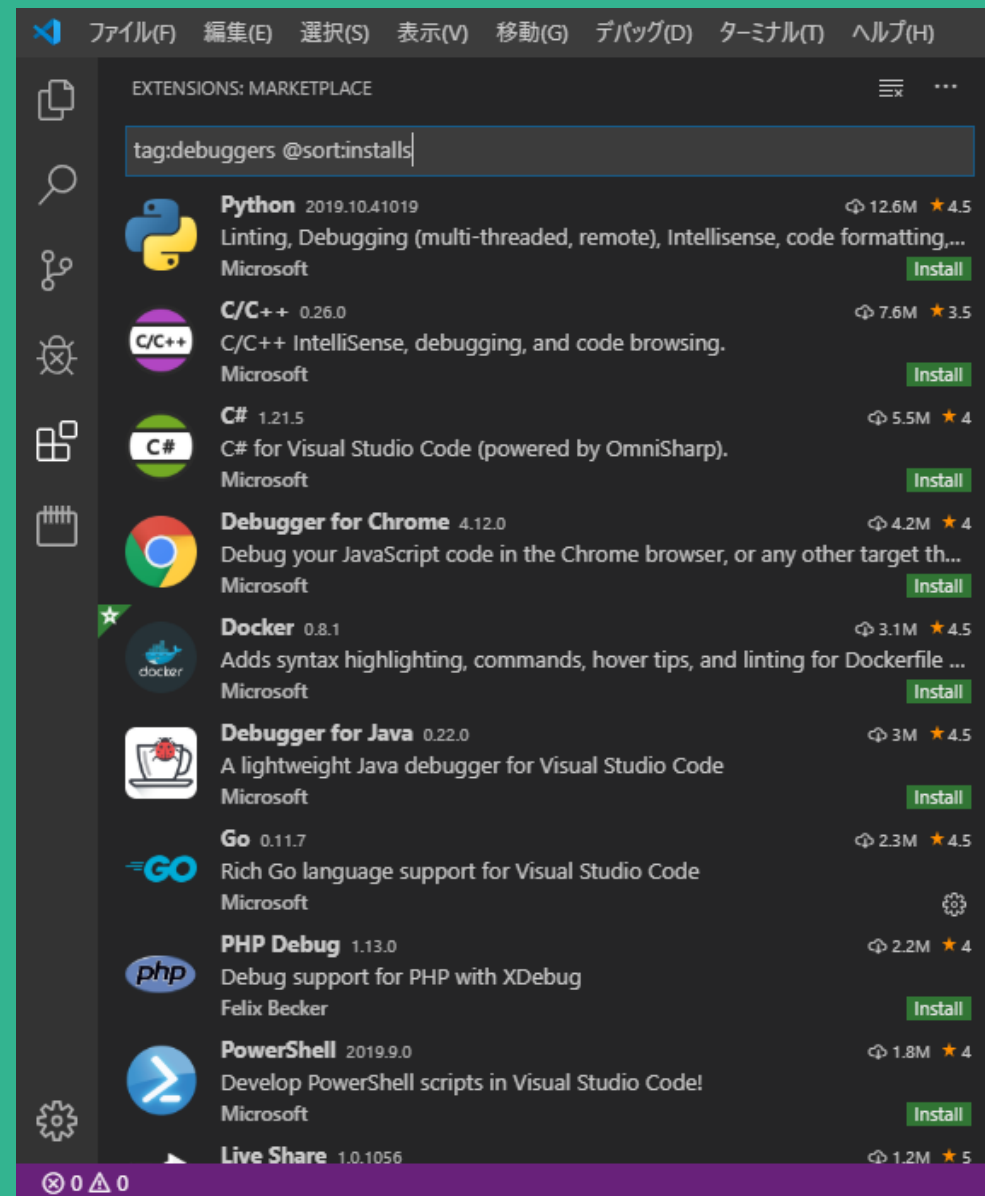
VS Codeの画面(Markdown)

- ・ 左画面の内容が即座に右画面に反映



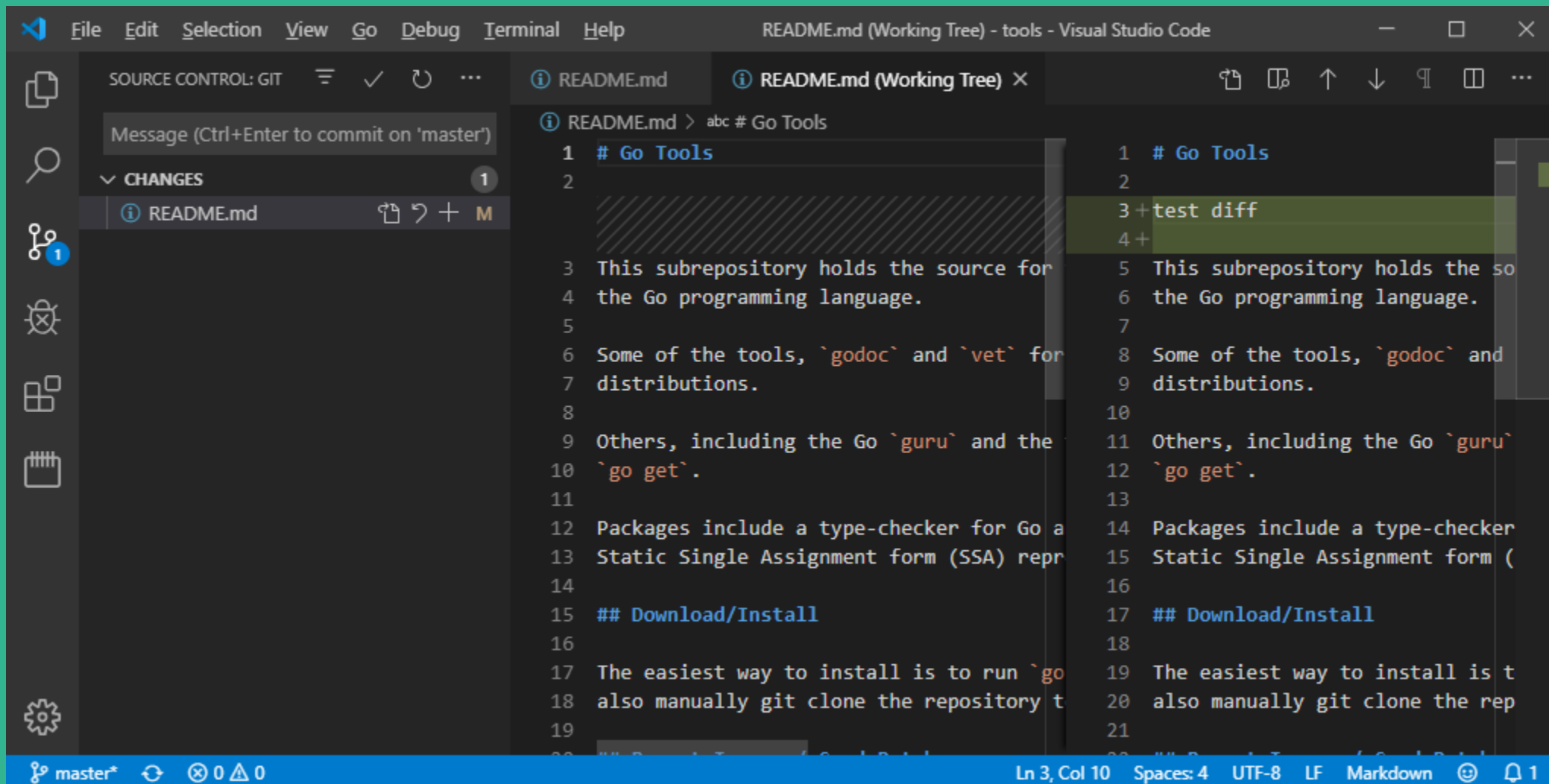
VS Codeの言語対応

- 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

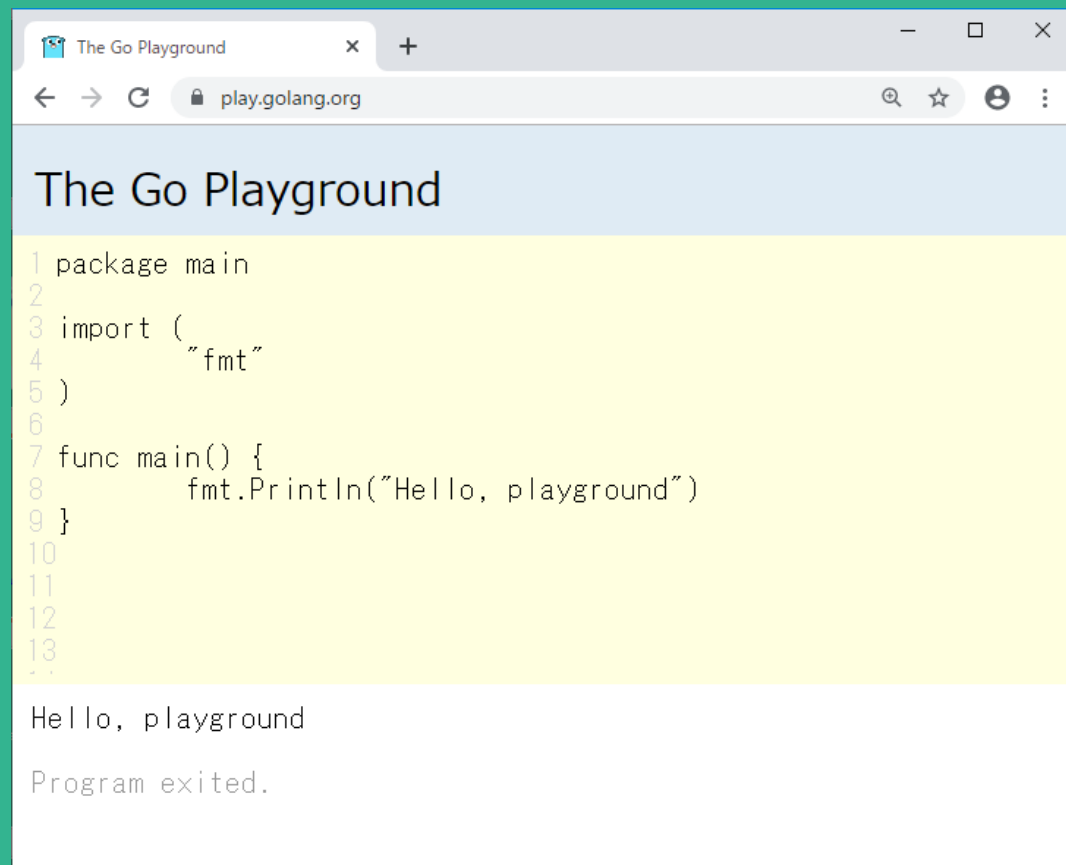
- ・ コンパイル結果は、ネイティブコード
 - Windowsの場合、1つのexeファイル
- ・ 実行先PCに実行環境不要
 - exeファイルのみで実行可能
 - 配布しやすい、デプロイしやすい

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>

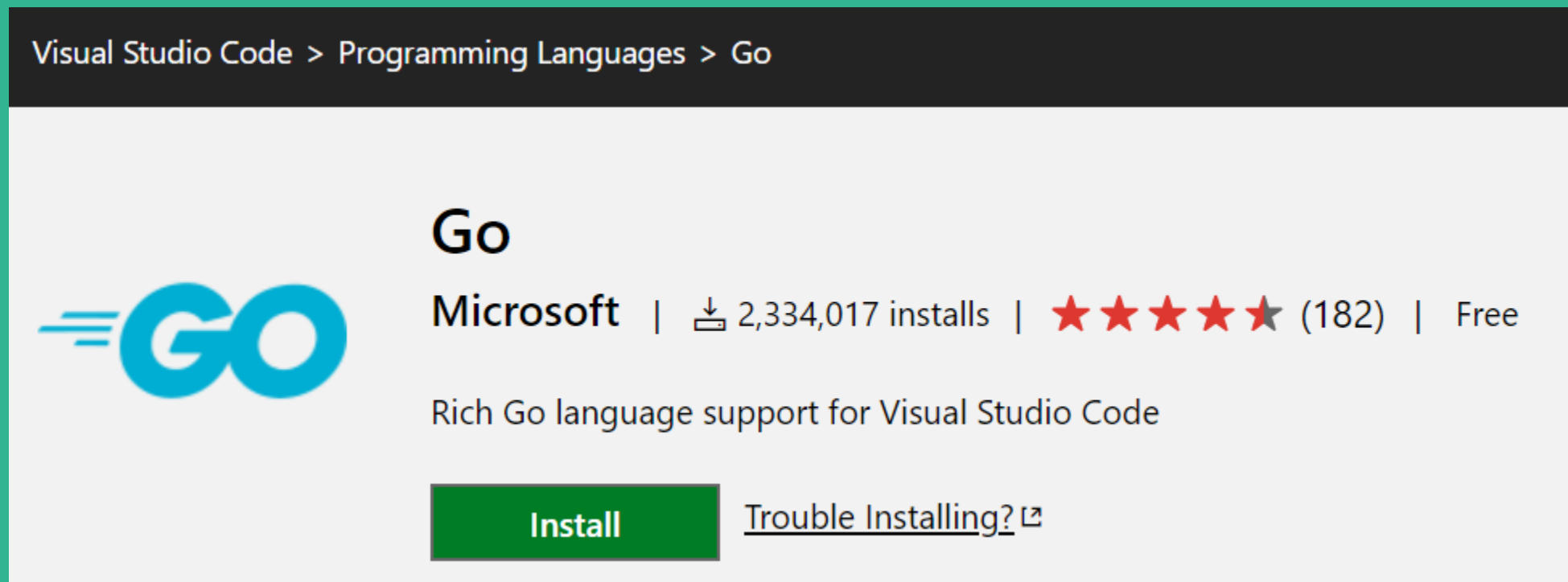
3. VS CodeとGo言語

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

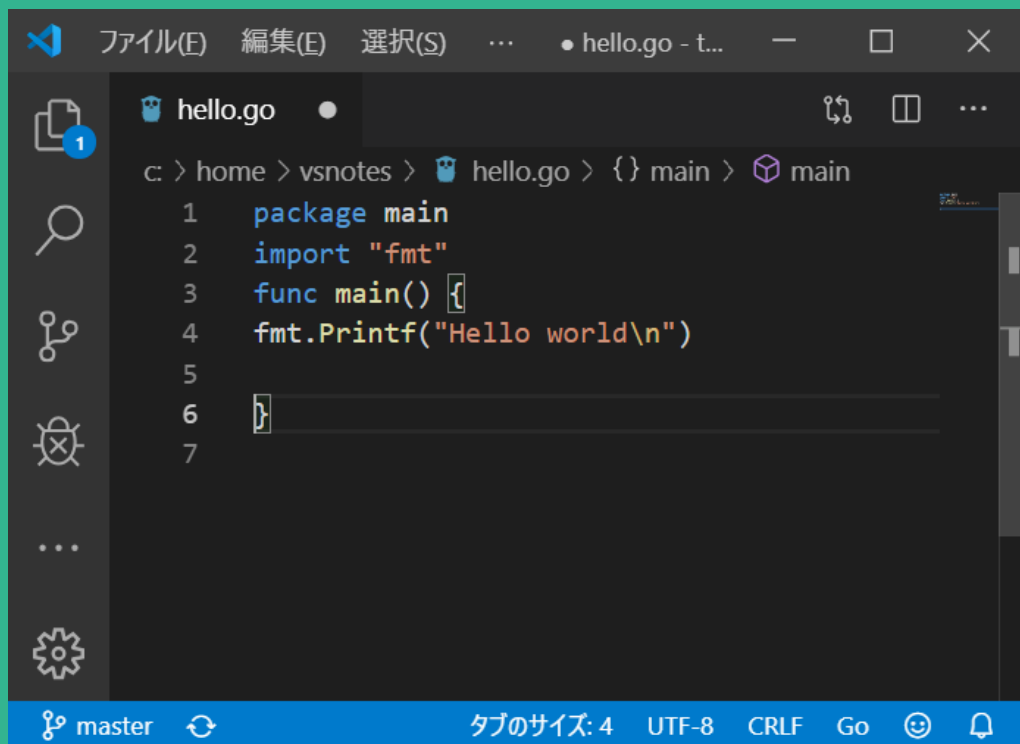
VS CodeのGo言語用拡張機能

- Microsoftオフィシャルの拡張機能



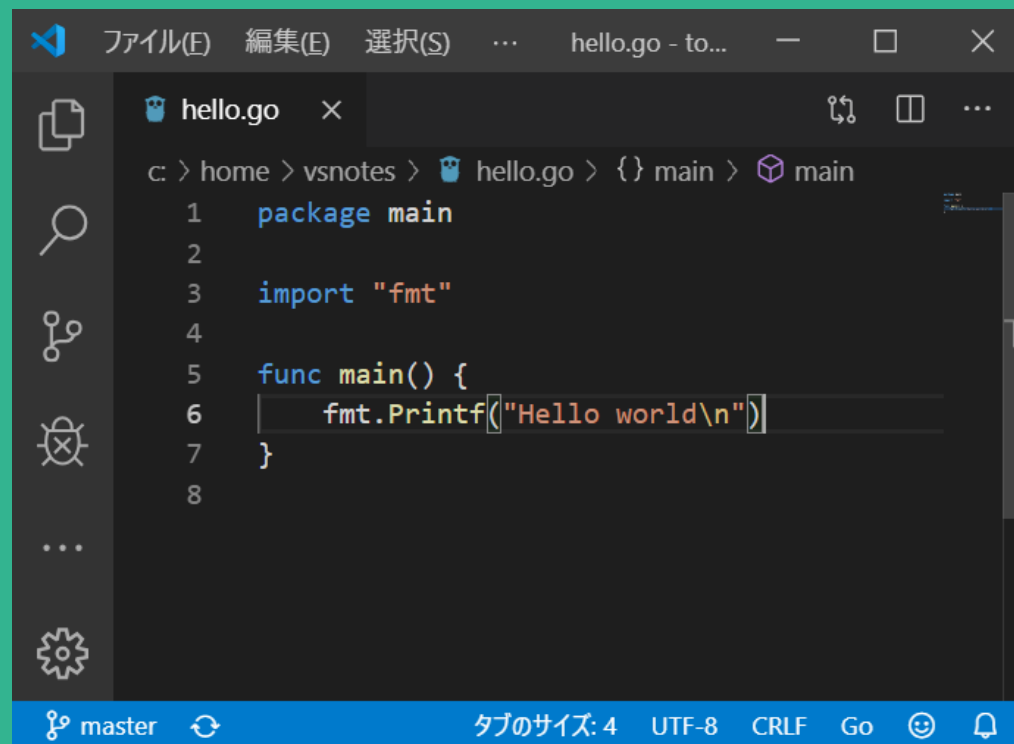
コード整形

- gofmt : Go言語の標準整形コマンドを使用
- 保存時に自動実行



The left screenshot shows the VS Code editor with a file named `hello.go`. The code is unformatted, with the `func main()` block spanning multiple lines. The status bar at the bottom indicates the file is on the `master` branch, with a tab size of 4, UTF-8 encoding, CRLF line endings, and the Go language mode selected.

```
1 package main
2 import "fmt"
3 func main() {
4     fmt.Printf("Hello world\n")
5 }
6
7
```

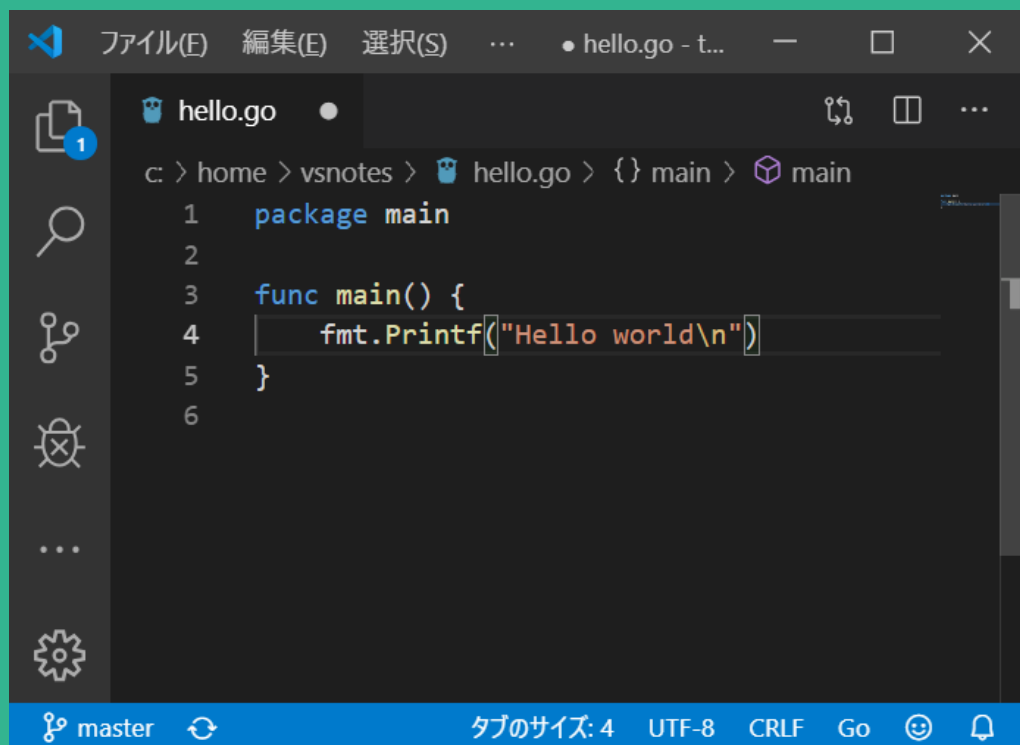


The right screenshot shows the same VS Code editor with the `hello.go` file, but the code is now formatted by `gofmt`. The `func main()` block is properly indented and wrapped. The status bar at the bottom remains the same, indicating the file is on the `master` branch, with a tab size of 4, UTF-8 encoding, CRLF line endings, and the Go language mode selected.

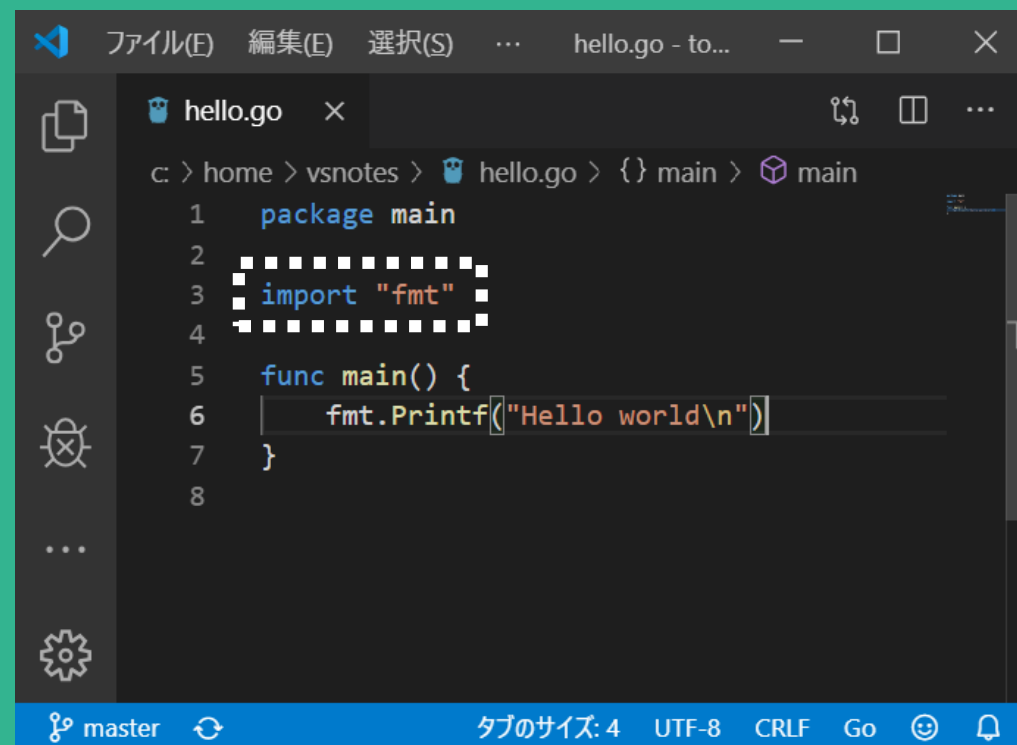
```
1 package main
2
3 import "fmt"
4
5 func main() {
6     fmt.Printf("Hello world\n")
7 }
8
```


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



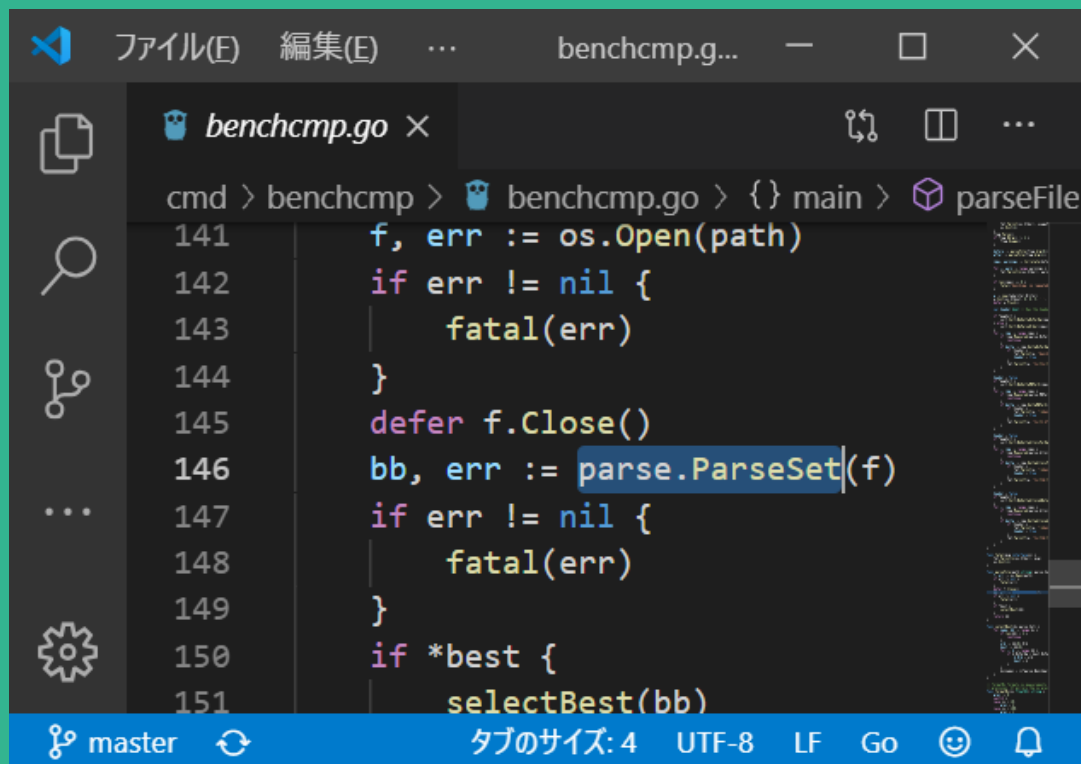
```
1 package main
2
3 func main() {
4     fmt.Printf("Hello world\n")
5 }
6
```



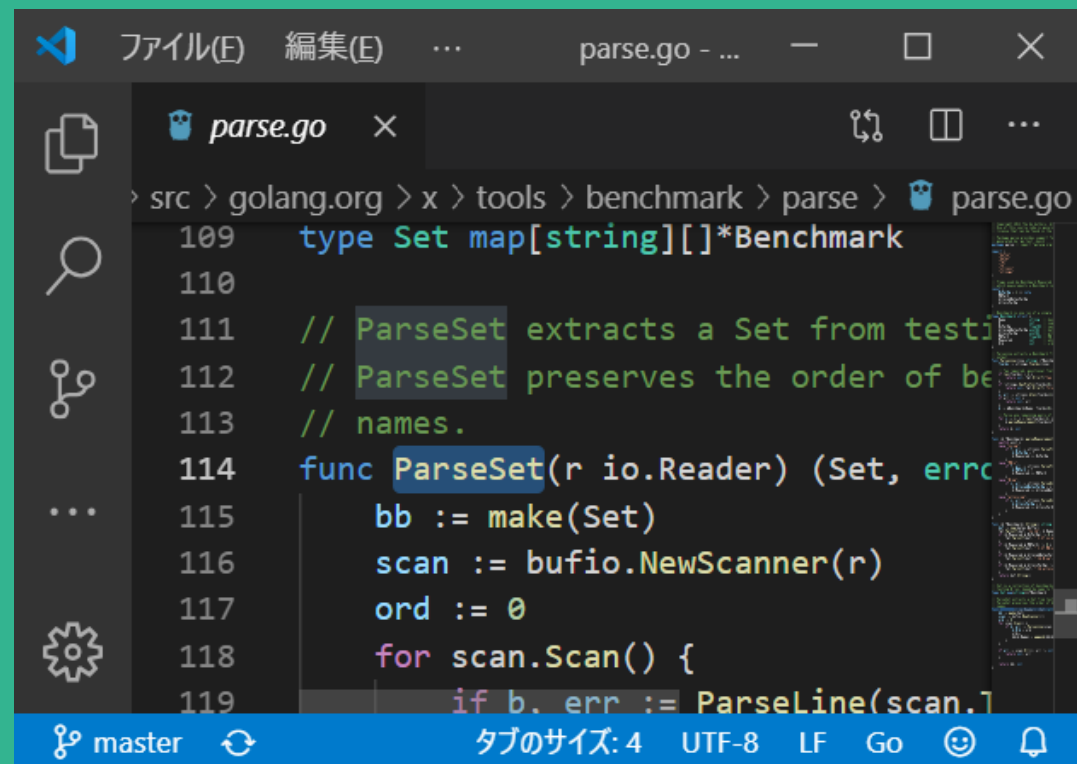
```
1 package main
2
3 import "fmt"
4
5 func main() {
6     fmt.Printf("Hello world\n")
7 }
8
```

コード・ナビゲーション

- 関数定義に移動



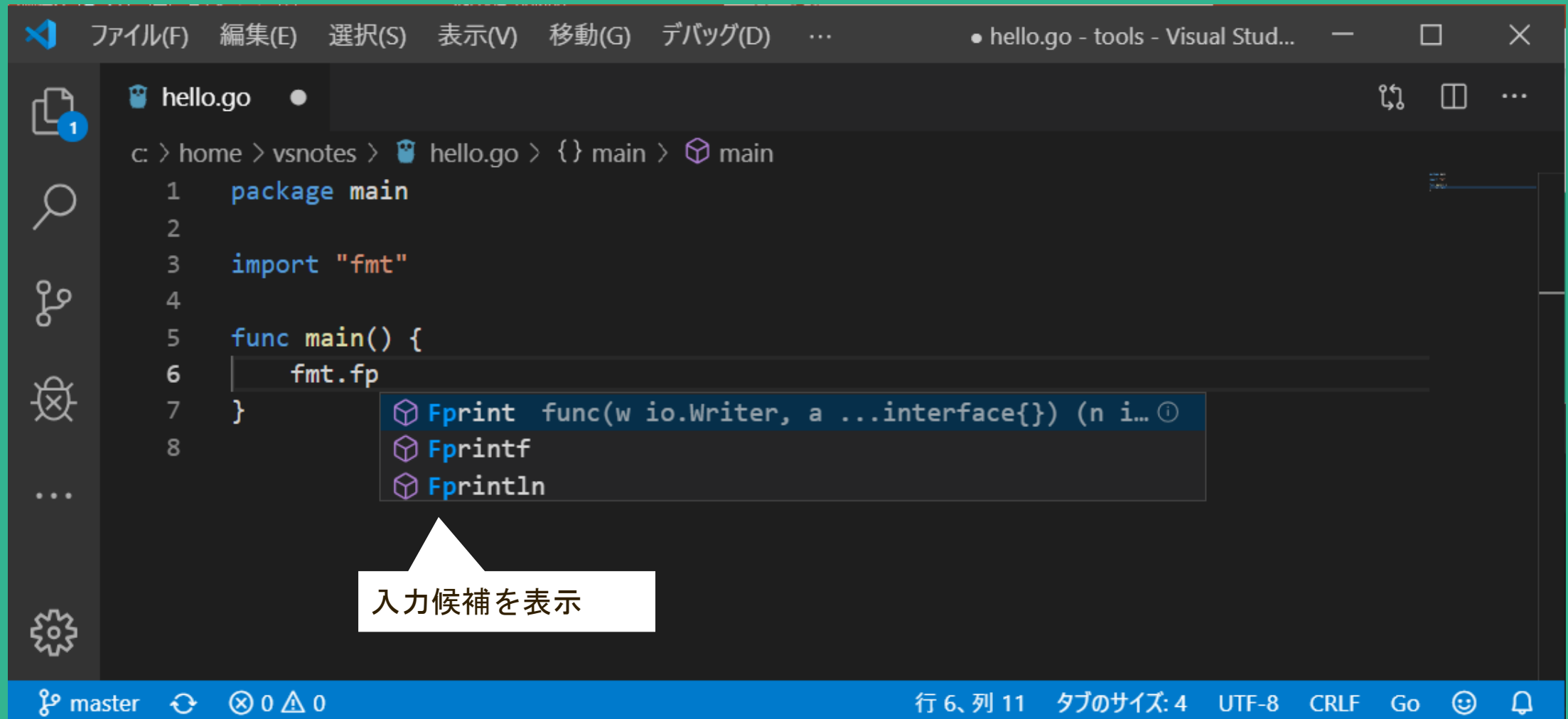
```
cmd > benchcmp > benchcmp.go > {} main > parseFile
141 f, err := os.Open(path)
142 if err != nil {
143     fatal(err)
144 }
145 defer f.Close()
146 bb, err := parse.ParseSet(f)
147 if err != nil {
148     fatal(err)
149 }
150 if *best {
151     selectBest(bb)
```



```
src > golang.org > x > tools > benchmark > parse > parse.go
109 type Set map[string][]*Benchmark
110
111 // ParseSet extracts a Set from test
112 // ParseSet preserves the order of be
113 // names.
114 func ParseSet(r io.Reader) (Set, error) {
115     bb := make(Set)
116     scan := bufio.NewScanner(r)
117     ord := 0
118     for scan.Scan() {
119         if b.err := ParseLine(scan.T
```

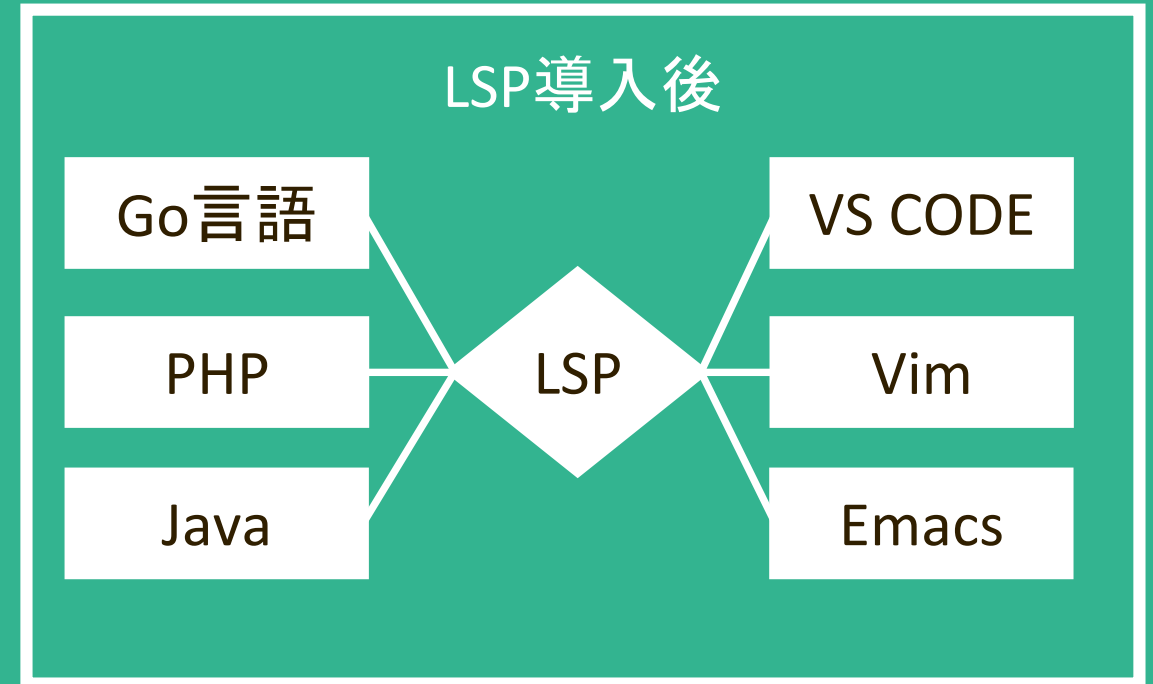
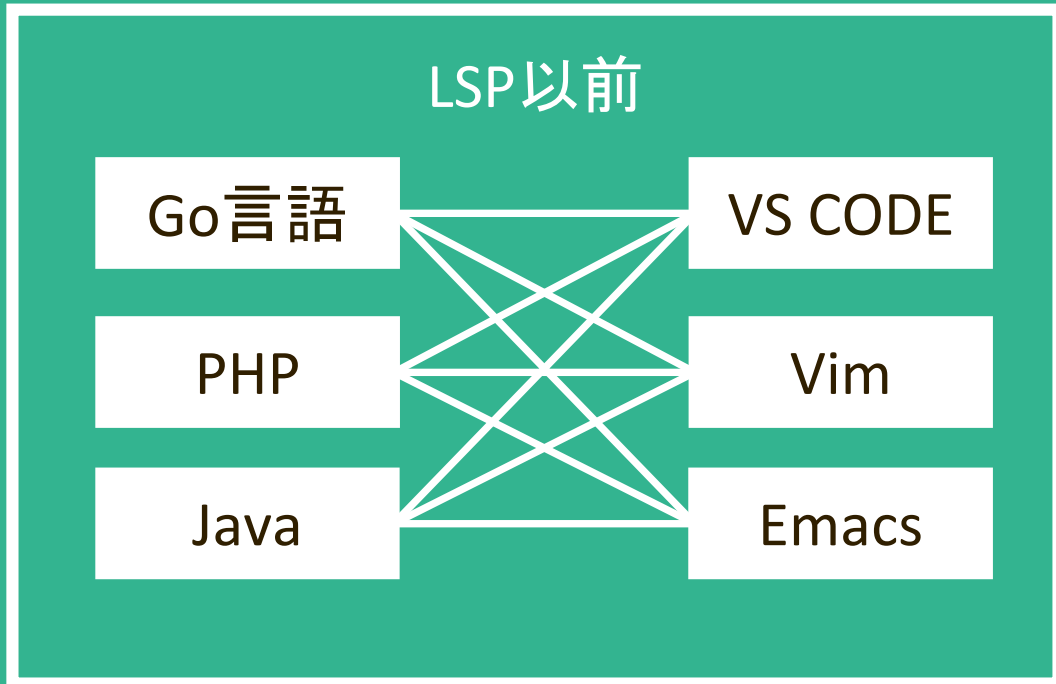
IntelliSense(コード補完)

- 関数名などの入力候補を表示(goplsを使用)



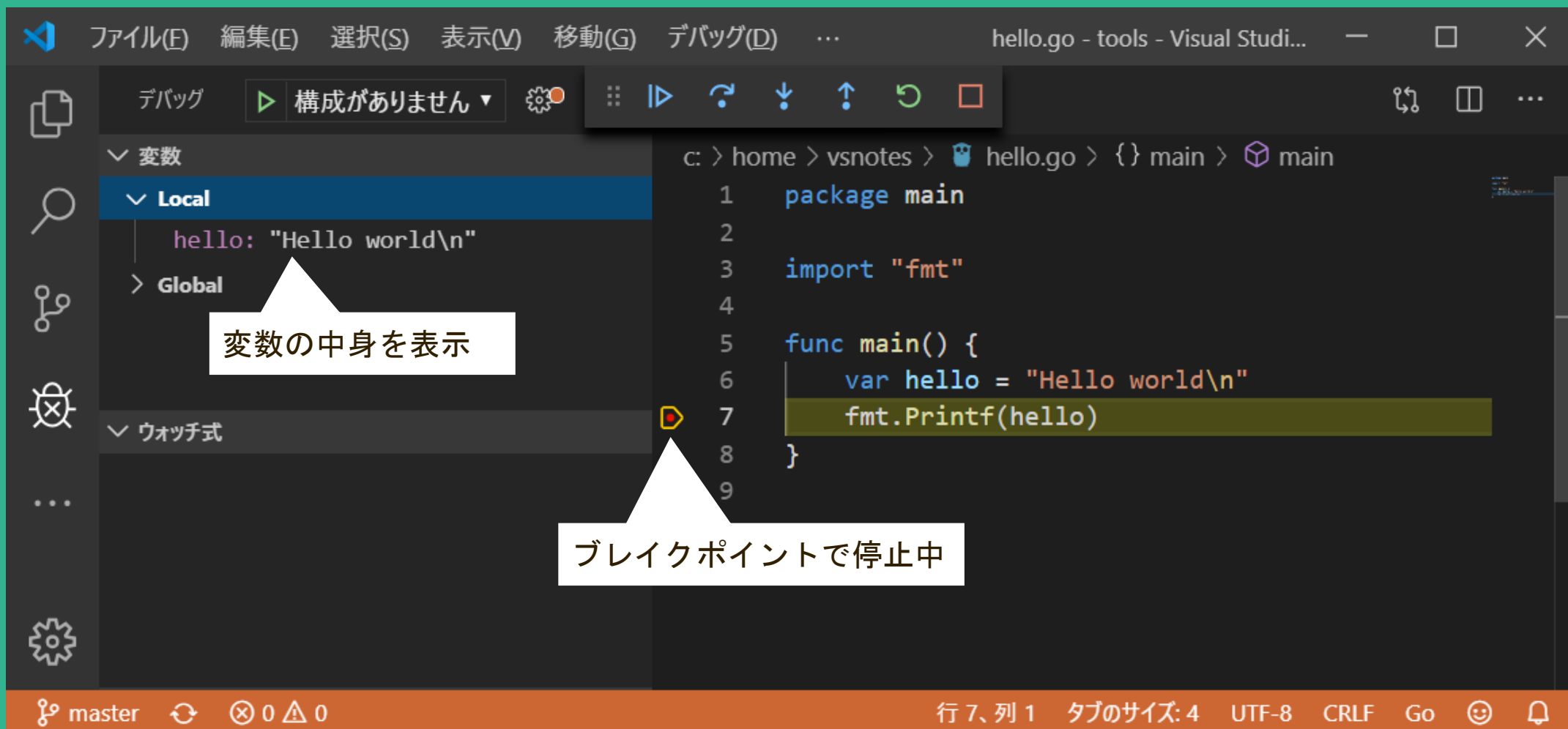
Language Server Protocol(LSP)

- 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



デバッグ

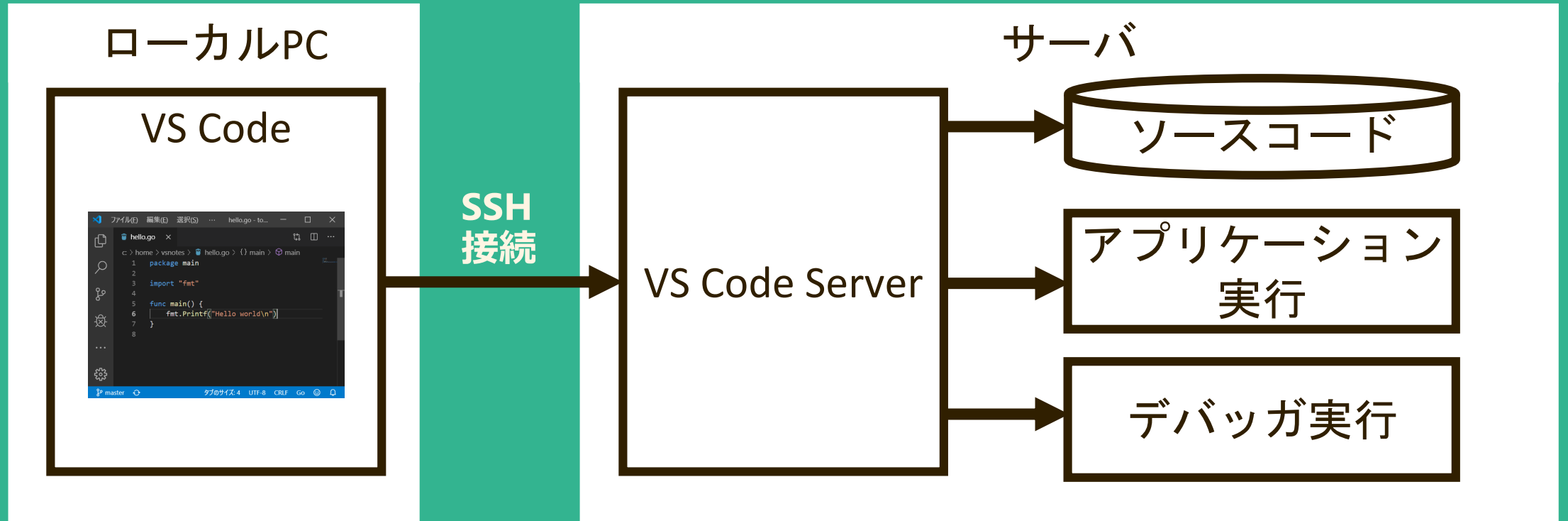
- Delve(Go言語デバッガ)をVS Codeに統合



4. リモートツールによる連携

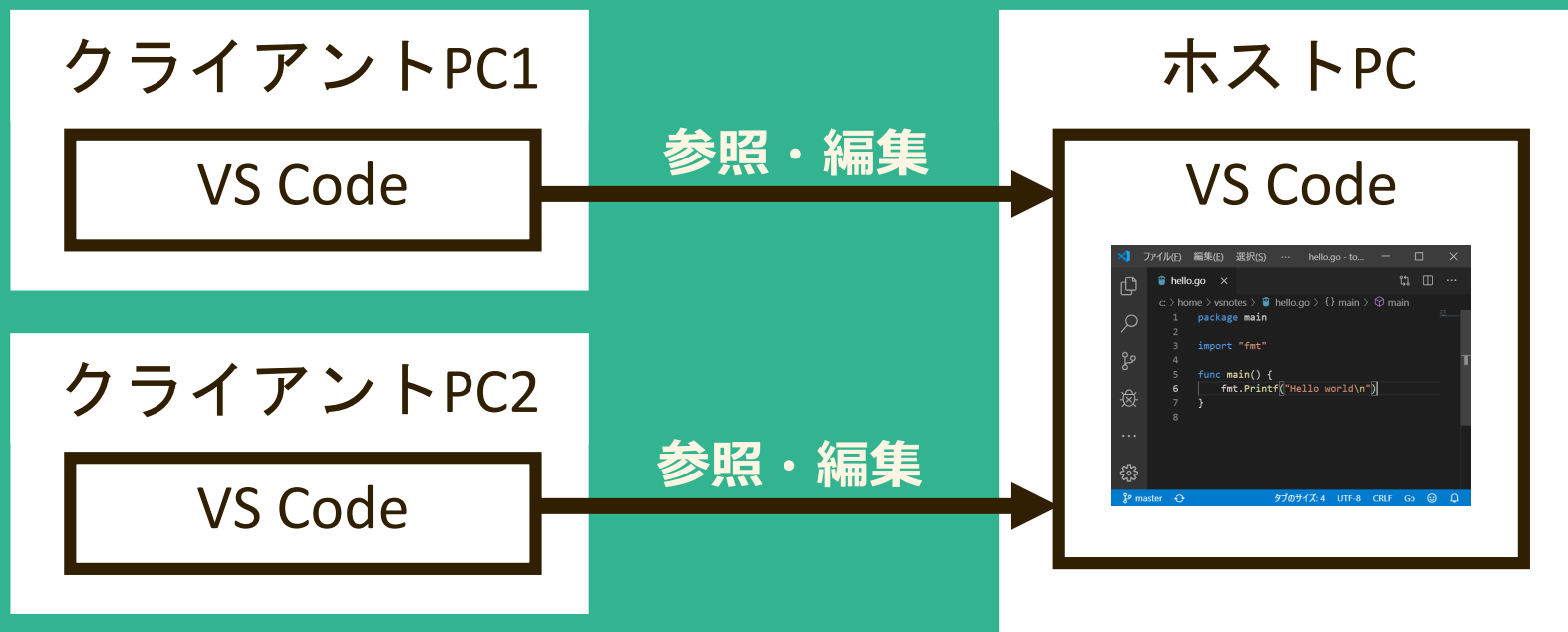
Remote Development

- ・ 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)

The screenshot displays the Visual Studio Live Share interface. On the left, the 'SESSION DETAILS' sidebar lists participants and shared resources. The main area shows a code editor with a JavaScript file named 'GuestbookGrid.js'. Two participants are highlighted: 'Jon W Chu' and 'Amanda Silver'. A callout box on the right, with arrows pointing to the highlighted names, contains the text '複数の人が同時参照・編集' (Multiple people can simultaneously reference and edit).

SESSION DETAILS

- Participants (3)
 - Jon W Chu • Header.js:12
 - Amanda Silver • GuestbookGrid.js:13
 - PJ Meyer • GuestbookGrid.js:9
- Shared Servers (2)
 - localhost:3000
 - REST API
- Shared Terminals (2)
 - bash (Read-only)
 - bash (Read/write)
- Audio Participants (3)
 - Jon W Chu
 - Amanda Silver
 - PJ Meyer

```
1 import GridArrow from "../GridArrow";
2 import GridLegend from "../GridLegend";
3 import GuestbookGridCell from "../GuestbookGridCell";
4
5 export default class GuestbookGrid extends Component {
6   constructor(props) {
7     super(props)
8     this.state = {
9       signatures: signatures
10    }
11  }
12  render() {
13    const cells = this.state.signatures.map((signature, index) => (
14      <GuestbookGridCell key={index} {...signature} />
15    ));
16  }
17 }
18 }
```

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

5. DockerとAzure

Docker

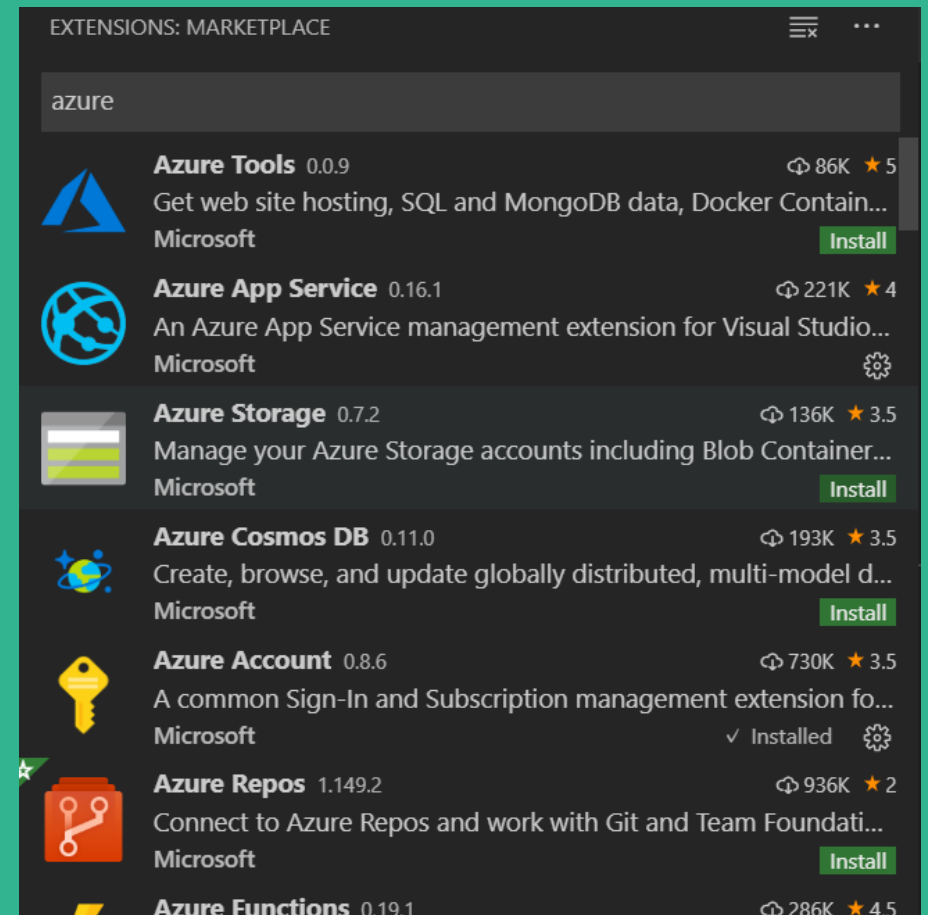
- Linuxの仮想環境（コンテナ）を提供するソフトウェア
- ソフトウェア開発でDockerを使用するメリット
 - サーバ環境構築が容易
 - ライブラリ等のバージョン統一化

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 Registry**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



Azure App Service Preview

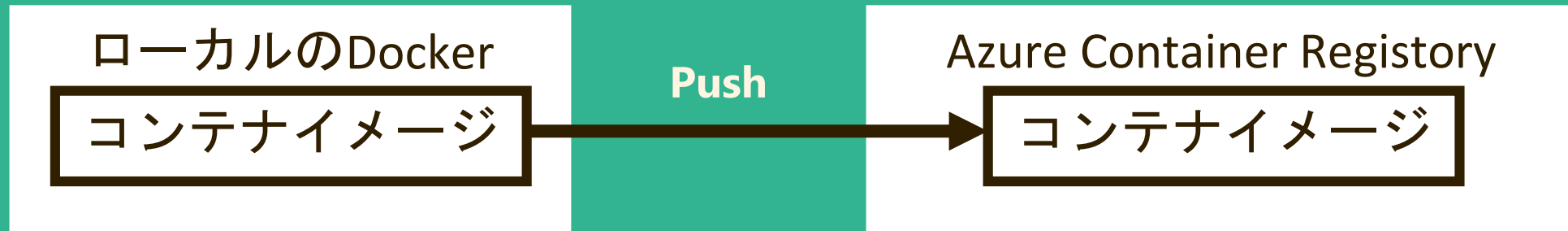
Microsoft | 📦 221,534 installs | ★★★★★ (13) | Free

An Azure App Service management extension for Visual Studio Code.

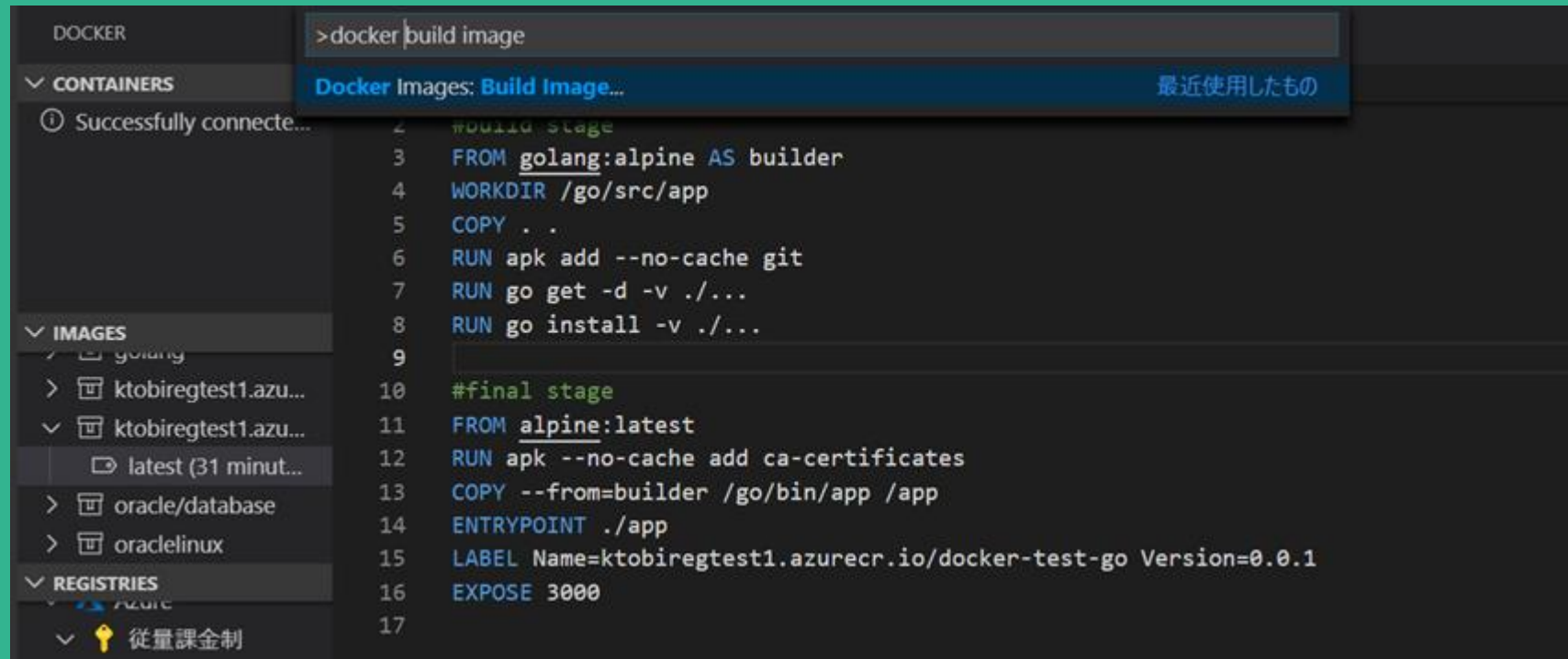
[Install](#) [Trouble Installing?](#)

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 Registry
 - Azureで使用するDockerのコンテナイメージを格納

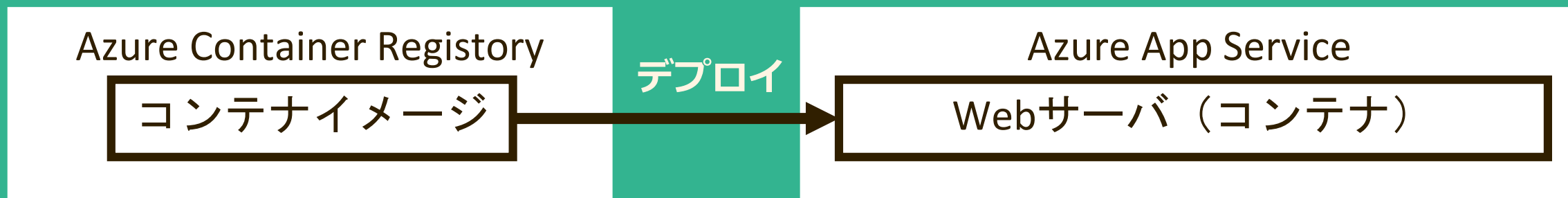


VS Codeでコンテナイメージ作成(画面)

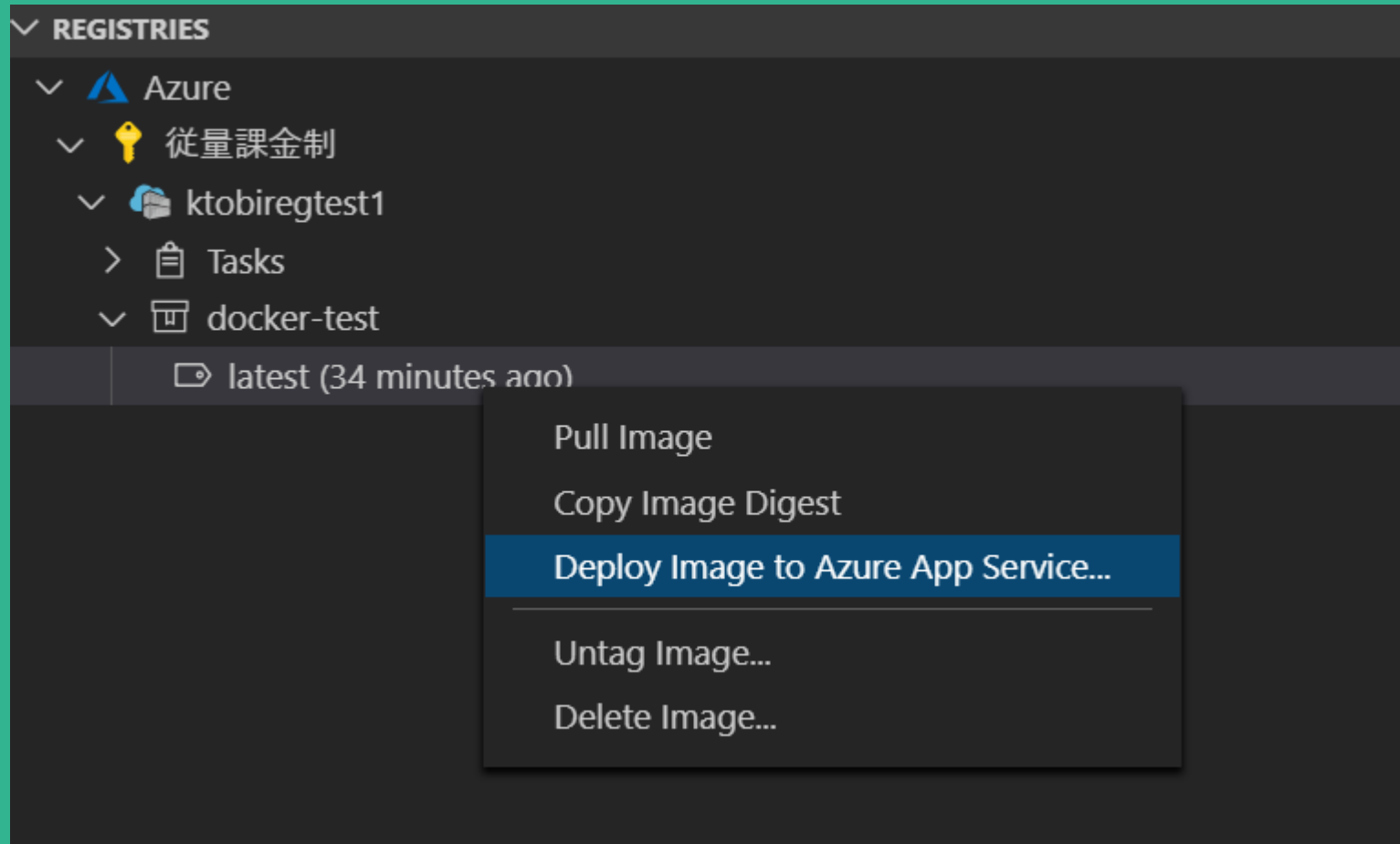


VS Codeでデプロイ

- Azure Container Registry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