

Visual Studio Code Remote Dev & Cloud Code Guide

Atsushi Morimoto(@74th) 

VSCode Remote Dev & Cloud Code Guide

目次

1. VSCodeに集まる世界	3
2. リモート開発機能とは	7
3. リモートSSH機能を使う	10
4. リモートマシンを構築しよう	18
5. Linuxネイティブdockerを使おう	37
6. Dev Containerを使おう	41
7. Cloud Codeを使おう	46
8. Skaffoldを使って、コンテナを自動ビルド、デプロイしよう ...	55

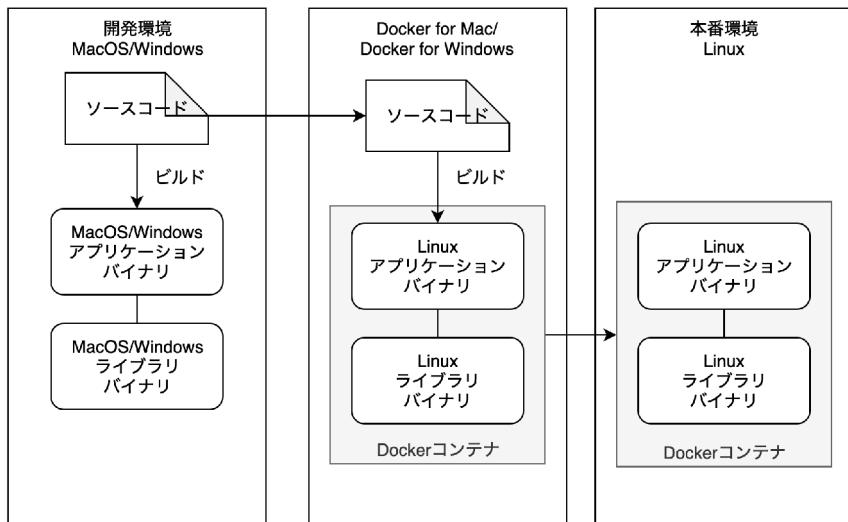
第1章 VSCodeに集まる世界

1.1 リモート開発機能の何が素敵か？

VSCode リモート機能はとても魅力的な機能です。特に私が魅力的に感じている点は、以下の点です。

- MacOS、Windowsを使いながらも、実際にアプリケーションを動作させるバイナリはLinuxのものを使うことができる。
- SSH経由でLinuxを使っても、操作性は全く低下しない。

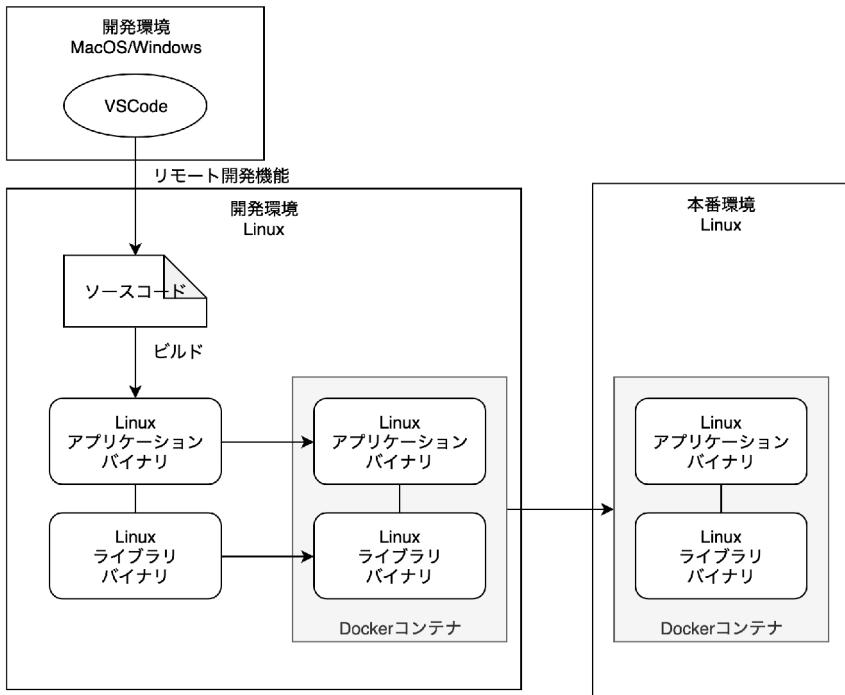
.NETフレームワークを使用している限り、動作させる環境はLinuxサーバ上で動かすアプリケーションです。それにも関わらず、私たちの多くは開発環境としてMacOS、Windowsを使っていました。MacOS、WindowsのバイナリはLinuxで動作させることはできません。スクリプト言語（Ruby、Pythonなど）や、中間言語（Java、.NETなど）は、動作させることはできますが、依存ライブラリによっては完全に動作できるかは開発者によって保証はされません。開発時もライブラリは必要で、MacOS、Windowsで動作するライブラリも必要となります。



▲現在の開発

現在、アプリケーションの実行はライブラリを含めてDockerコンテナによって行われます。それにも関わらず、Dockerコンテナを用いて開発ができるケースは少ないとと思われます。

VSCodeのリモート開発機能はMacOS、Windowsを使いながら、Linuxマシンを開発環境として使う機能です。これを用いることで、コンパイラやプログラミング言語のランタイムで、プログラミング言語のツールは本番環境と同じLinuxで動作するものを利用できます。このため、開発環境のために別途MacOS、Windowsで動作するライブラリを用意する必要がなくなります。



▲リモート開発機能での開発

筆者の場合、MacOS上にLinuxの仮想マシンを構築した上で、VSCodeのリモート開発機能を使って開発作業の全てすべてLinux上で行っています。本書では、リモート機能を使いこなすためtipsと、筆者が実践している仮想マシン構築方法のtipsを紹介します。

1.2 Cloud Codeの何が魅力的か？

GoogleはCloud CodeというKubernetesのフロントエンドを公開しています。これはGoogleが新しいIDEを構築したのではなく、JetBrainsとVSCodeの拡張機能として公開されました。

これによりKubernetesが扱いやすくなりました。今までkubectlコマンドで努力が必要だったものが、マウスを使って直感的に操作することができます。しかし、上記を用いてもkubectlコマンドを全く利用しないことはできません。なおCloud Codeはコンテナに対してデバッグする機能を提供してくれますし、

第1章 VSCodeに集まる世界

コンテナアプリの開発を支援するSkaffoldとの連携もできるようになっています。本書では、Cloud Codeは何を提供してくれているのか、この先のロードマップはどんなものかを見ていくたいと思います。

第2章 リモート開発機能とは

本書で主張したいことはただ1つです。

「なぜLinuxで動かすアプリケーションをMacOS、Windowsで開発しているのか？ Linuxでの開発でいいのではないか？」ということです。

2.1 リモート機能の本質とは

VSCodeの持つ機能として、Language Server Protocol（以下LSP）があります。これはプログラミング言語の支援機能をプロトコルとしてまとめたものです。VSCodeではほとんどのプログラミング言語のための機能が、LSPを用いる言語サーバを通じて提供されます。またLSPは、言語サーバとしてのプログラミング言語の文脈補完機能以外にも、リントツールや、ソースコードフォーマットも提供します。LSPのプロトコルの実装はJSON RPCを利用しています。つまり、JSON RPCが通じるネットワークに接続さえできれば、言語サーバを動かすサーバと、言語サーバを利用するクライアントが成立します。VSCodeのリモート機能では、VSCodeの拡張機能である言語サーバをSSHの接続先サーバや、WSL（Windows Subsystem Linux）、Dockerコンテナ上で動かすことができます。このようにプロトコルとしてLSPを定めたのが、リモート機能の実現につながっているのです。

2.2 ユーザにとってのリモート機能の利点とは

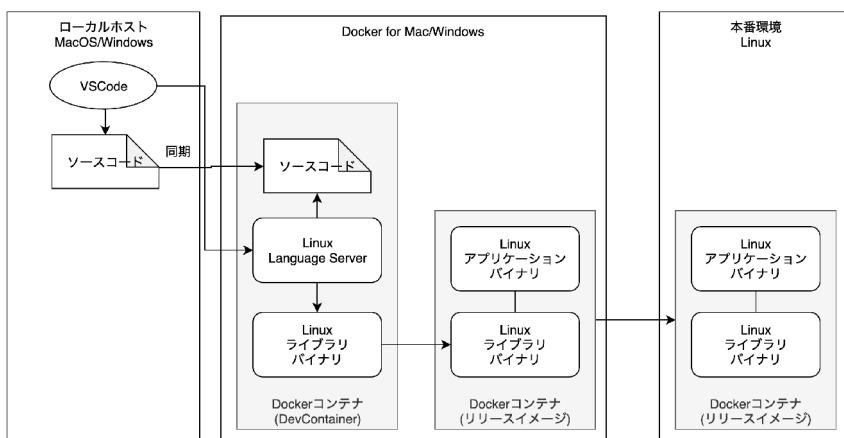
MacOSやWindowsでも開発環境が構築できれば、あまりリモート開発機能に魅力を感じないかもしれません。しかし、以下でつまずいた経験はないでしょうか？

- 使用するライブラリはLinuxのシェアードライブラリを使っているが、それと同様のバージョンのライブラリがMacOS、Windowsで提供されていなかった
- オープンソースライブラリを、開発環境のMacOS、Windowsでビルドしようとしたのに、ビルドできなかつた
- Linuxであればsystemtapや、ptraceを使って解析できるにも関わらず、開発環境のMacOS、Windowsでは同様のツールが使えなかつた

第6章 Dev Containerを使おう

6.1 Dev Containerとは

DevContainerとは、VSCodeの用語で、開発時に用いるDockerコンテナを指します。 Dockerコンテナを開発で使い、リリースするアプリケーションがDockerコンテナであれば、開発環境と本番環境の環境依存の問題はなくなります。 完全に同じコンテナを使うことは難しいですが、できる限り近いコンテナを使えれば大変便利です。 VSCodeのリモートコンテナ機能とは、DevContainerを使いその中でLanguage Serverを起動して開発をすすめることができる機能です。



▲Dev Containerの開発イメージ

リモートコンテナ機能では以下のようなことが実現します。

- ローカルのVSCodeを使って、DevContainerでLanguage Serverなど拡張機能が動作します
- ローカルのソースコードはDevContainer内にマウントされます。リモートSSH開発のように、コンテナ内にコードをチェックアウトする必要はありません

第7章 Cloud Codeを使おう

Cloud CodeとはGoogleが提供するKubernetesのリソース管理、およびKubernetes開発を支援するVSCodeの拡張機能です。現在ベータ版という扱いですが、以下の機能を使うことができます。

- Kubernetesのリソース管理
 - リソースがツリー状に表示できる
 - マニフェストの編集ができる
 - ログをVSCodeで開くことができる
 - クラスタの作成ができる（しかし、クラスタの作成には通常かなりの設定が必要であるため、VSCodeからクラストアを作ることは普通しません）-->
 - 他のリソースを参照するリソースの場合、マニフェストから定義を追うことができる（ConfigMapなど）
- Kubernetesのマニフェストの作成
 - Kubernetesのマニフェストのコード補完ができる
 - Kubernetesのスニペットが利用できる
- Skaffoldを使って、変更の自動ビルド、デプロイができる
- コンテナにアタッチしてデバッグすることができます

Kubernetesのリソースの一部がVSCode上で管理できることは魅力的です。ログの表示や、マニフェストの表示も、VSCodeの使い慣れた機能を使うことができるので、解析が楽になります。

特に、今まで実現されていなかったKubernetesの構文の自動補完や、スニペットはとても便利です。筆者はもうCloud CodeなしでKubernetesのマニフェストを書こうとは思わなくなりました。

また、コンテナに直接アタッチしてデバッグする機能も、制限は大きいですが開発時はとても有用です。