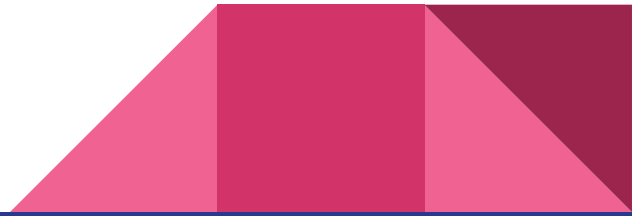
Javaまったくわからんエンジニアが Spring Bootの環境をdevcontainerする

SIOS Tech lab ｜龍ちゃん 

アジェンダ

- ご挨拶
- 前提条件
- 使用するdocker-composeコマンドの確認
- 開発環境構築
- まとめ



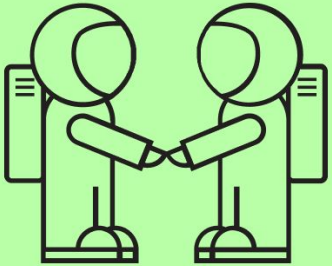
龍ちゃん

@RyuReina_Tech

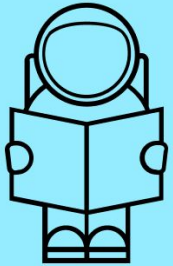
3年目のピヨピヨエンジニア

フロントエンド

デザイン業務

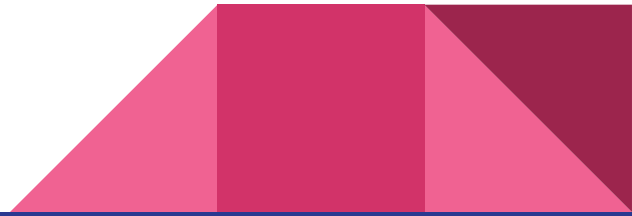


外部発信



今日の目標

Spring Bootの環境をdevcontainerで作成する

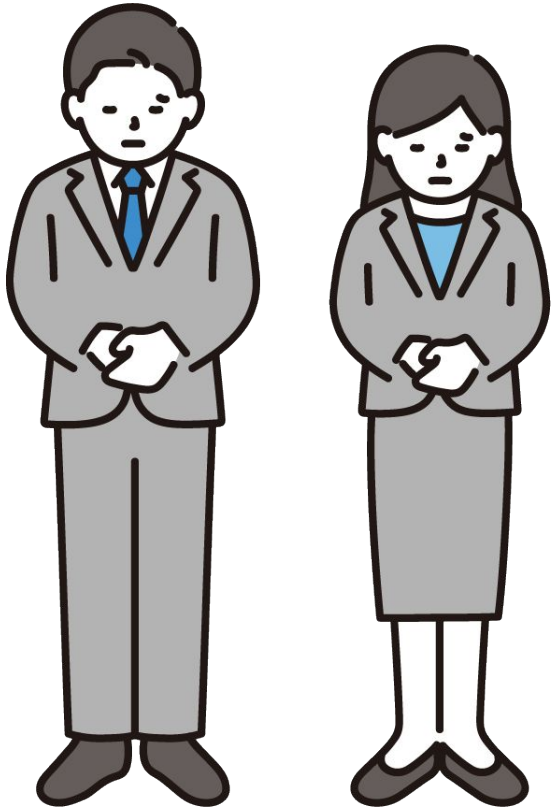


前提条件・お願い

前提条件

- 開発環境がVS Codeの人である*
- docker composeが実行できる環境がある
- 萎えない心

＊Dev Containers tutorial（https://code.visualstudio.com/docs/devcontainers/tutorial）



もし間違ってたら有識者の方教えてくださいませ　(ง👐)パァ

使用するコマンド確認

使用するコマンドの確認

docker compose build	サービスの構築

参考：docker ドキュメント（https://docs.docker.jp/engine/reference/commandline/compose.html#id9）

開発環境構築

開発環境構築：全体の流れ

- 作成する環境の整理
- 環境作成
  - Spring Bootプロジェクトのダウンロード
  - 効率的なプロジェクトに改変
  - devcontainerでup
  - デバック機能の追加

作成する環境の整理：要件整理

JDKバージョン	21
Spring Boot バージョン	3.2.4
ビルドツール	Maven
プロジェクトタイプ	Java（Spring Boot）
拡張機能	Spring Web：Web開発 Spring Boot Dev Tools：ホットリロード

作成する環境の整理：目指すディレクトリ構造

```
.
├── .devcontainer
│   ├── Dockerfile
│   └── devcontainer.json            // devcontainer設定ファイル
├── .dockerignore                    // ビルド時に無視するファイルパス
├── docker-compose.yml
└── app                              // Spring Bootプロジェクト
```

開発環境構築：開発環境の流れ

1. Spring Bootプロジェクトの作成
2. 改善
    a. 改善①：non-rootユーザで実行する
    b. 改善②：Volume Trickを使用してTargetをコンテナ内に収める
    c. 改善③：Docker build時のキャッシュを活用
3. 一旦起動確認
4. VS Codeデバッグできる環境を整備する

開発環境構築：作成した環境の結果

ソースコード見にくいので、GitHubに上げています

開発環境作る

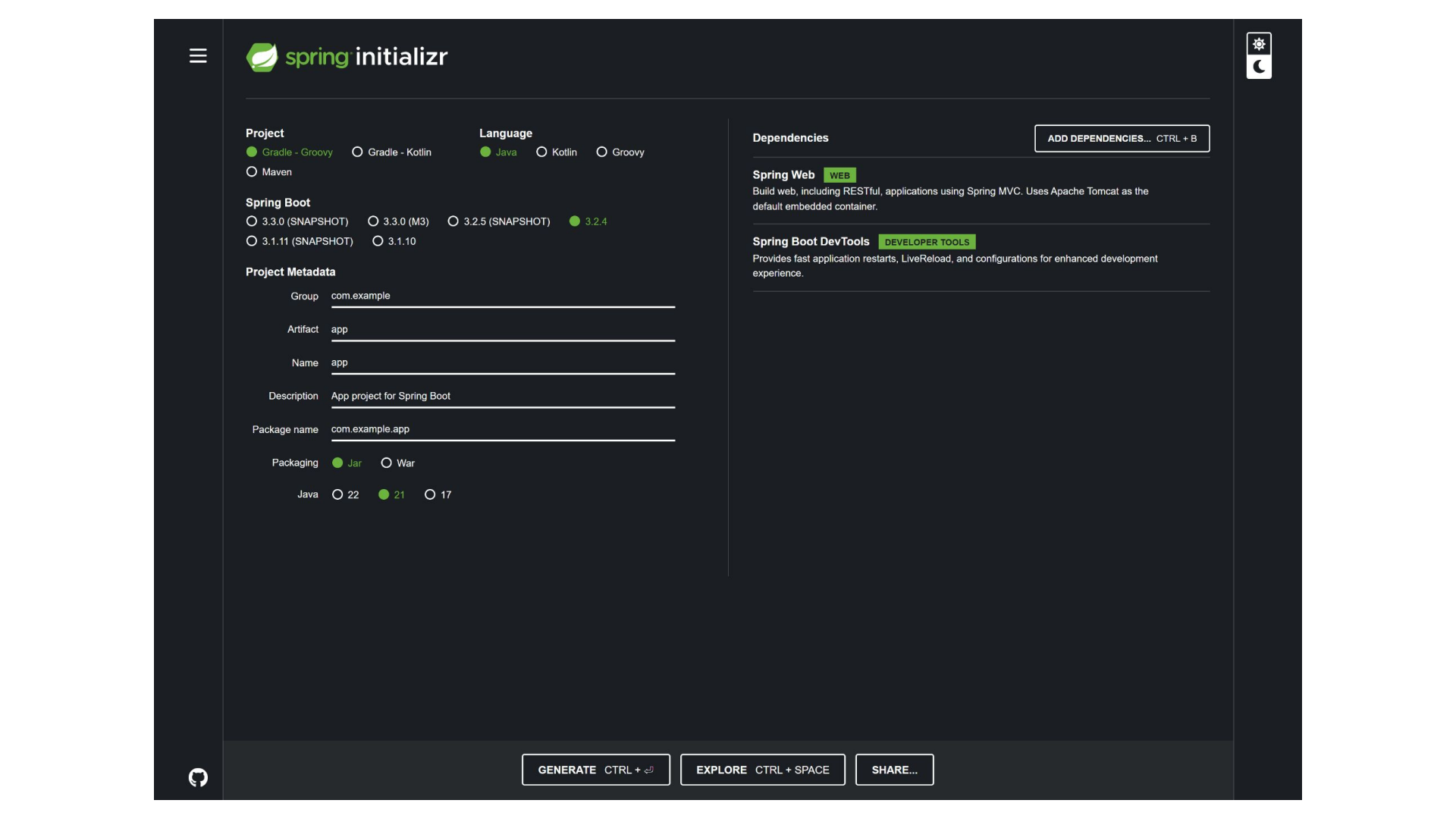
開発環境作る

1. Spring Bootプロジェクトの作成
2. 改善
    a. 改善①：non-rootユーザで実行する
    b. 改善②：Volume Trickを使用してTargetをコンテナ内に収める
    c. 改善③：Docker build時のキャッシュを活用
3. 一旦起動確認
4. VS Codeデバッグできる環境を整備する

Spring Bootプロジェクトの作成

公式が初期構築に便利なWebサービス（spring initializr）を出していました。

- GUIで設定可能
- 拡張機能なども設定することもできる
- アプリケーションはZipファイルで出力される


spring initializr
Project
Gradle - Groovy
Gradle - Kotlin
Maven
Language
Java
Kotlin
Groovy
Spring Boot
3.3.0 (SNAPSHOT)
3.3.0 (M3)
3.2.5 (SNAPSHOT)
3.2.4
3.1.11 (SNAPSHOT)
3.1.10
Project Metadata
Group
com.example
Artifact
app
Name
app
Description
App project for Spring Boot
Package name
com.example.app
Packaging
Jar
War
Java
22
21
17
Dependencies
ADD DEPENDENCIES... CTRL + B
Spring Web
WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.
Spring Boot DevTools
DEVELOPER TOOLS
Provides fast application restarts, LiveReload, and configurations for enhanced development experience.
GENERATE CTRL +
EXPLORE CTRL + SPACE
SHARE...

Spring Bootプロジェクトの作成

起動に必要なおまじない：ローカルで実行してね

```
chmod +x ./app/mvnw
```

なぜこのおまじないが必要か？

```
#0 0.253 /bin/sh: ./mvnw: Permission denied
```

ファイルで実行権限がないため実行をすることができない

Docker内のalpine環境でRUN chmod +x mvnwを実行してもうまくいかない....

Spring Bootプロジェクトの作成

シンプルな構成で一度立ち上げてみる

```
.
├── .devcontainer
│   ├── Dockerfile
│   └── devcontainer.json
├── .dockerignore
├── app                                  // Spring Bootプロジェクト
└── docker-compose.yml
```

Spring Bootプロジェクトの作成

.dockerignore

```
**/target
```

Dockerfile

```
FROM eclipse-temurin:21-jdk-alpine
WORKDIR /app

COPY --chown=spring:spring ./app .
RUN ./mvnw dependency:go-offline

CMD ["./mvnw", "spring-boot:run"]
```

Spring Bootプロジェクトの作成

docker-compose.yml

```yaml
version: "3.7"
services:
  spring:
    build:
      context: .
      dockerfile: ./.devcontainer/Dockerfile.down
    tty: true
    volumes:
      - type: bind
        source: ./app
        target: /app
    ports:
      - 8080:8080
```

Spring Bootプロジェクトの作成

devcontainer.json

```
{
  "dockerComposeFile" : ["../docker-compose.yml" ],
  "service": "spring",
  "workspaceFolder" : "/app",
  "forwardPorts" : [8080],
  "customizations" : {
    "vscode": {
      "extensions": [
        "vscjava.vscode-java-pack" ,
        "vmware.vscode-boot-dev-pack" ,
        "vscjava.vscode-spring-initializr"
      ]
    }
  },
  "remoteUser" : "root"
 }
```

Spring Bootプロジェクトの作成

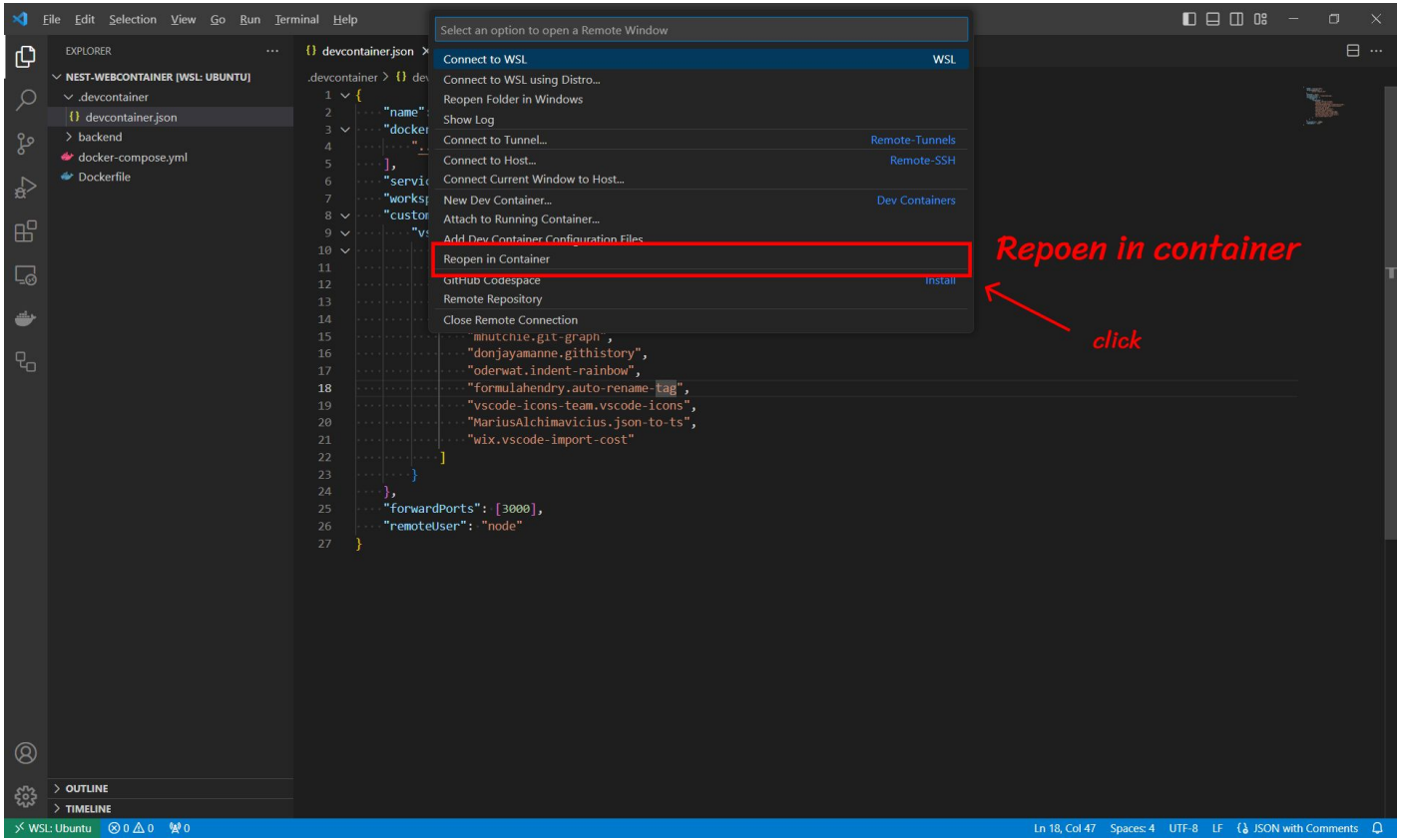
devcontainerで立ち上げてみましょう。（VS Codeの拡張機能が必要です）

参考：Dev Containers（vscode:extension/ms-vscode-remote.remote-containers）

Spring Bootプロジェクトの作成

devcontainerでコンテナを起動する

Spring Bootプロジェクトの作成



Spring Bootプロジェクトの作成

問題点

- 実行ユーザーがrootになる
- targetディレクトリがroot権限で作成される
- レイヤーキャッシュが活用されていない

これを改善していこうと思います

Spring Bootプロジェクトの作成

改善手法

- 改善①：Volume Trickを使用してTargetをコンテナ内に収める
- 改善②：non-rootユーザで実行する
- 改善③：Docker build時のキャッシュを活用

開発環境作る

1. Spring Bootプロジェクトの作成
2. 改善
    a. 改善①：Volume Trickを使用してTargetをコンテナ内に収める
    b. 改善②：non-rootユーザで実行する
    c. 改善③：Docker build時のキャッシュを活用
3. 一旦起動確認
4. VS Codeデバッグできる環境を整備する

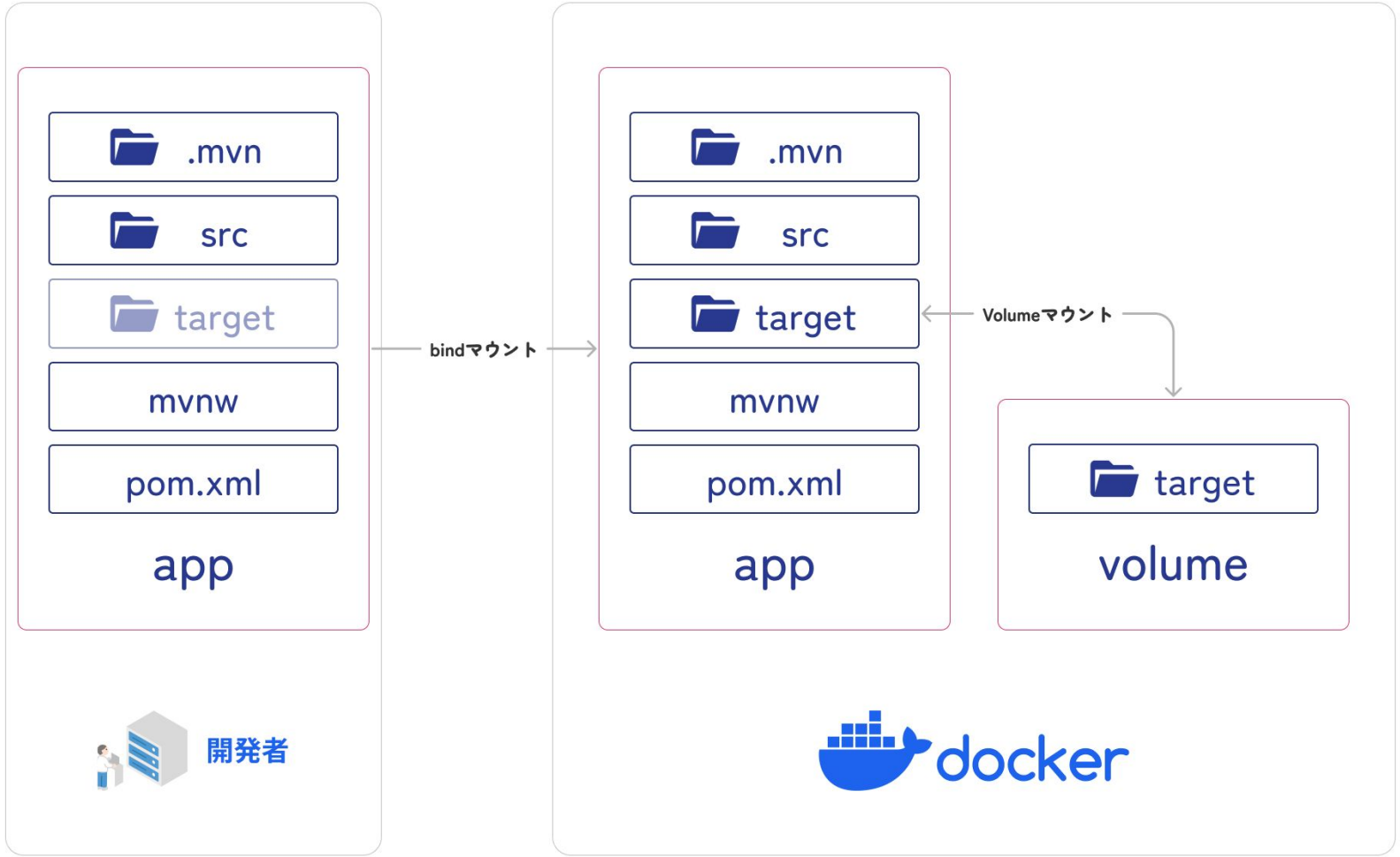
どこをVolume Trickする？

『Spring Bootの***Target***ファイルを Volume Trickでコンテナに収める』

Why?

- Targetはビルド結果を保存されている
- 常に最新のデータである方がうれしいが、変更管理をする必要がない

Volume Trickを使用してTargetをコンテナ内に収める

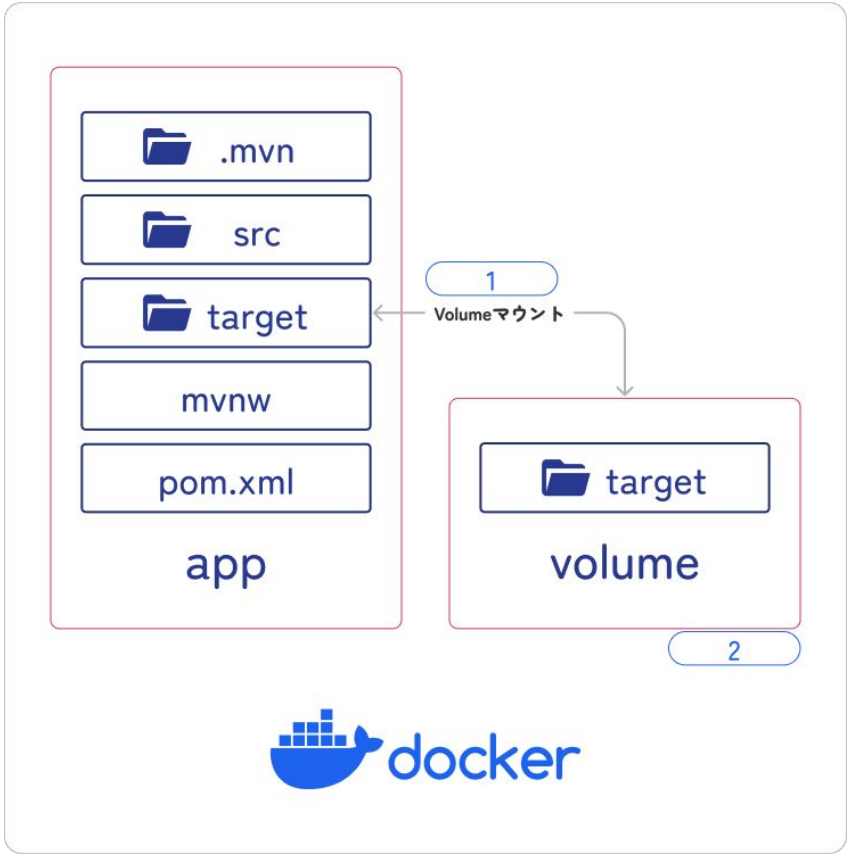

.mvn
src
target
mvnw
pom.xml
app
bindマウント
.mvn
src
target
mvnw
pom.xml
app
Volumeマウント
target
volume
開発者
docker

```yaml
version: "3.7"
services:
  spring:
    build:
      context: .
      dockerfile: ./.devcontainer/Dockerfile
    tty: true
    volumes:
      - type: bind
        source: ./app
        target: /app
      - type: volume
        source: target
        target: /app/target
    ports:
      - 8080:8080

volumes:
  target:
```

1

2



非rootユーザーでアクセスする

アプリケーションをrootで実行すると、アプリケーションからコンテナを実行しているPCへの攻撃が成立する可能性がある！

対策

- 実行ユーザー（spring）を作成する
- 実行ユーザーでアプリケーションを起動させる
- devcontainer側からも、実行ユーザーでアクセスする

Docker build時のキャッシュを活用する

Dockerのレイヤーキャッシュとは？

```
FROM golang:1.20-alpine
WORKDIR /src
COPY . .
RUN go mod download
RUN go build -o /bin/server ./cmd/client
RUN go build -o /bin/server ./cmd/server
ENTRYPOINT [ "/bin/server" ]
```

Dockerレイヤーキャッシュ

- 一行（層）ごとにキャッシュされる
- ビルド時にキャッシュがある場合は、キャッシュを活用する

積極的に分割したほうが良い

```
FROM eclipse-temurin:21-jdk-alpine
WORKDIR /app

RUN addgroup spring && adduser  --ingroup spring --disabled-password spring
RUN chown spring:spring ./

RUN mkdir target && chown spring:spring target

COPY --chown=spring:spring ./app/.mvn .mvn
COPY --chown=spring:spring ./app/mvnw ./app/pom.xml ./

USER spring:spring
RUN ./mvnw dependency:go-offline

COPY --chown=spring:spring ./app/src ./src

CMD ["./mvnw", "spring-boot:run"]
```

```
FROM eclipse-temurin:21-jdk-alpine
WORKDIR /app
```

springユーザー作成

```
RUN addgroup spring && adduser  --ingroup spring --disabled-password spring
RUN chown spring:spring ./
```

ボリュームマウント用ディレクトリ作成

```
RUN mkdir target && chown spring:spring target
```

依存関係とビルドツールのコピー

```
COPY --chown=spring:spring ./app/.mvn .mvn
COPY --chown=spring:spring ./app/mvnw ./app/pom.xml ./
```

依存関係のインストール

```
USER spring:spring
RUN ./mvnw dependency:go-offline
```

アプリケーションファイルのコピー

```
COPY --chown=spring:spring ./app/src ./src

CMD ["./mvnw", "spring-boot:run"]
```

開発環境作る

1. Spring Bootプロジェクトの作成
2. 改善
    a. 改善①：non-rootユーザで実行する
    b. 改善②：Volume Trickを使用してTargetをコンテナ内に収める
    c. 改善③：Docker build時のキャッシュを活用
3. 起動確認
4. VS Codeデバッグできる環境を整備する

Dockerfile

```
FROM eclipse-temurin:21-jdk-alpine
WORKDIR /app

RUN addgroup spring && adduser  --ingroup spring --disabled-password spring
RUN chown spring:spring ./

RUN mkdir target && chown spring:spring target

COPY --chown=spring:spring ./app/.mvn .mvn
COPY --chown=spring:spring ./app/mvnw ./app/pom.xml ./

USER spring:spring
RUN ./mvnw dependency:go-offline

COPY --chown=spring:spring ./app/src ./src

CMD ["./mvnw", "spring-boot:run"]
```

docker-compose.yml

```yaml
version: "3.7"
services:
  spring:
    build:
      context: .
      dockerfile: ./.devcontainer/Dockerfile
    tty: true
    volumes:
      - type: bind
        source: ./app
        target: /app
      - type: volume
        source: target
        target: /app/target
    ports:
      - 8080:8080

volumes:
  target:
```

devcontainer.json

```
{
  "dockerComposeFile": ["../docker-compose.yml"],
  "service": "spring",
  "workspaceFolder": "/app",
  "forwardPorts": [8080],
  "customizations": {
    "vscode": {
      "extensions": [
        "vscjava.vscode-java-pack",
        "vmware.vscode-boot-dev-pack",
        "vscjava.vscode-spring-initializr"
      ]
    }
  },
  "remoteUser": "spring"
}
```

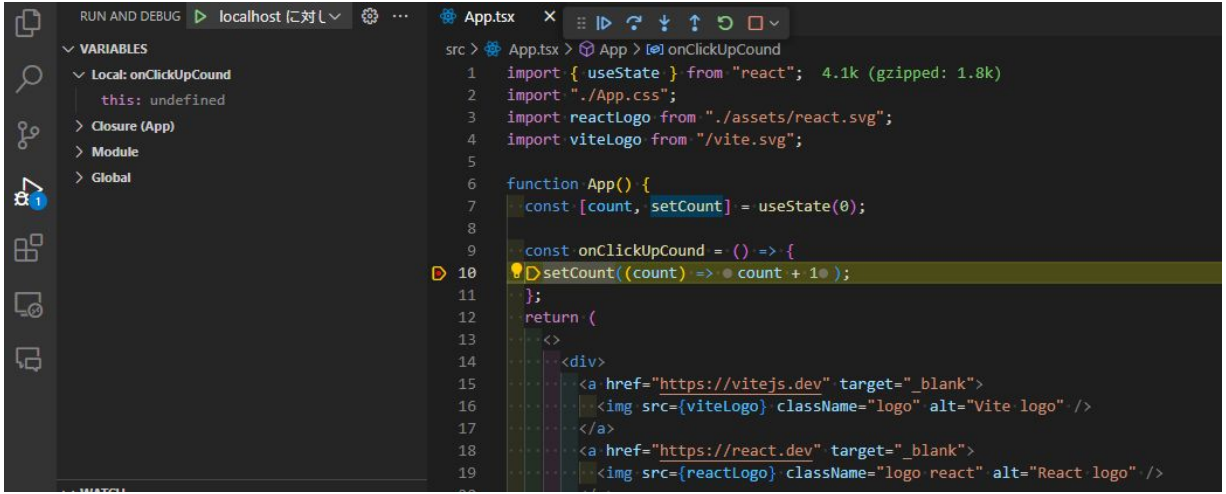
起動

ソースはこちらになります

開発環境作る

1. Spring Bootプロジェクトの作成
2. 改善
    a. 改善①：non-rootユーザで実行する
    b. 改善②：Volume Trickを使用してTargetをコンテナ内に収める
    c. 改善③：Docker build時のキャッシュを活用
3. 起動確認
4. VS Codeデバッグできる環境を整備する

デバック実行とは？



ブレークポイントを設定したり、実行中の値の確認などができる

開発の助けになる

設定方法

VS Codeで開いた際に、ルートディレクトリに.vscode/launch.jsonがあればよい

```
./app
└── .vscode
    └── launch.json
```

ワークスペースに.vscodeファイルがあれば認識される

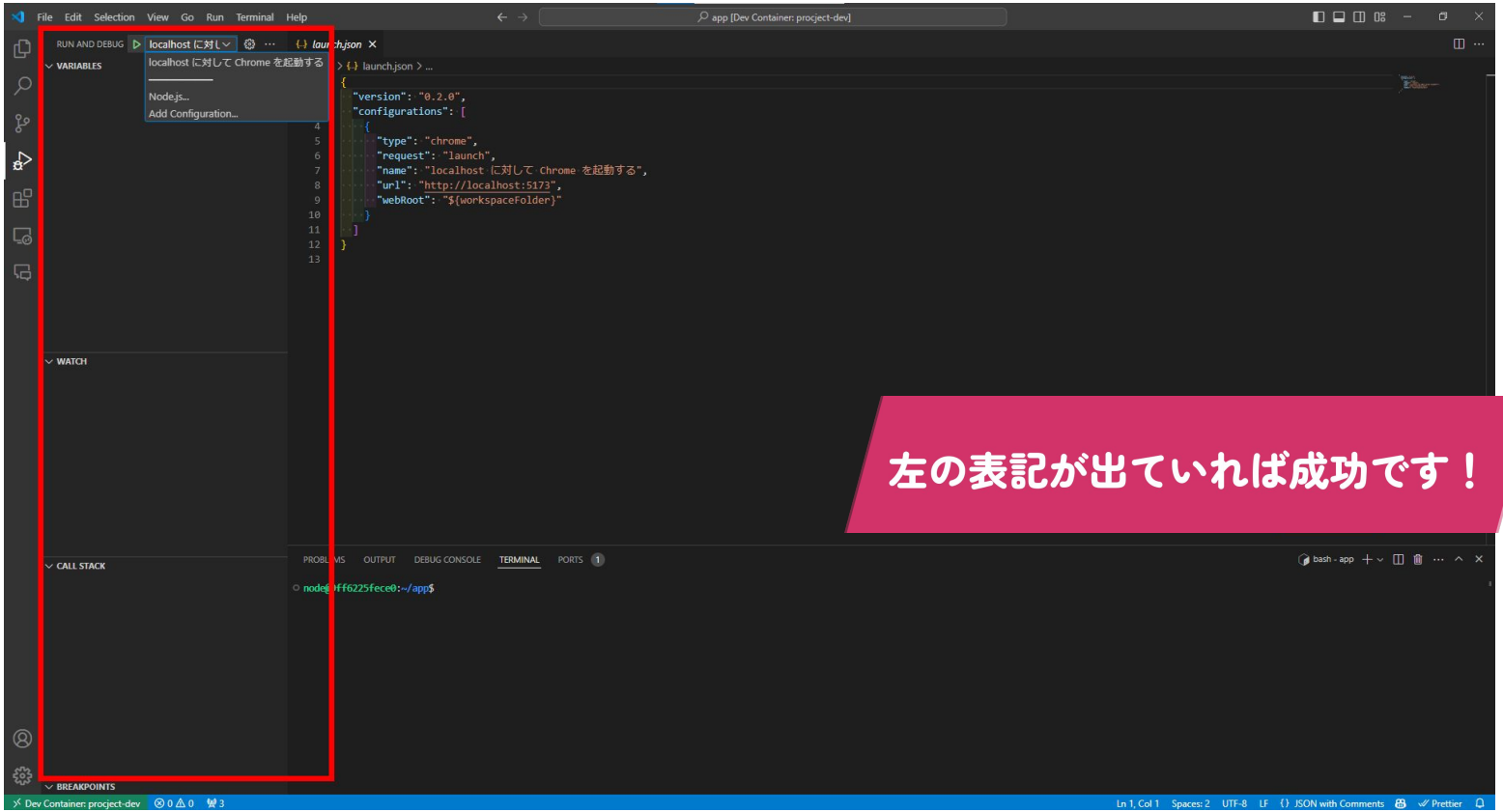
launch.jsonを記載することでデバック実行が可能になる

デバック機能追加

```
{
    "version": "0.2.0",
    "configurations": [
        {
            "type": "java",
            "name": "AppApplication",
            "request": "launch",
            "mainClass": "com.example.app.AppApplication",
            "projectName": "app",
            "env": {
                "SERVER_PORT": "8081",
            },
        }
    ]
}
```

- ○ デバック実行時にアプリを立ち上げる
- ○ アプリを8081で立ち上げる

デバック機能の確認



左の表記が出ていれば成功です！

まとめ

まとめ

こちらのリポジトリにまとまっています

これがdevcontianerのいいところ