Visual COBOL チュートリアル

Visual Studio Code を用いた COBOL 開発

1 目的

本チュートリアルは、Visual Studio Code を用いて COBOL 開発を行う手順の習得を目的としています。

軽量 IDE である Visual Studio Code を用いることで、Eclipse / Visual Studio IDE より低リソースで COBOL アプリケーションの開発やデバッグ実行が行えます。

2 前提

- 本チュートリアルで使用したマシン
  - Windows 環境

    Eclipse 製品： Windows 11

    Visual Studio 製品： Windows 11
  - Linux 環境

    Rocky 9.4
- Visual COBOL 11.0 Patch Update 01 for Windows, Visual COBOL Development Hub 11.0 Patch Update 01 が各環境にインストール済みであること
- Windows 環境側に Visual Studio Code がインストールされていること
- XServer を使用したリモート開発を行う場合は、Windows, Linux 間で X 通信が正常に行えること

下記のリンクから事前にチュートリアル用のサンプルファイルをダウンロードして、任意のフォルダーに解凍しておいてください。

サンプルプログラムのダウンロード

内容

3 チュートリアル

見出し 1 説明

3.1 Windows 環境

Rocket COBOL プラグインが未インストールの場合は、4.1.1 を参照のうえ、インストールしてください。

3.1.1 既存 Eclipse プロジェクトの利用

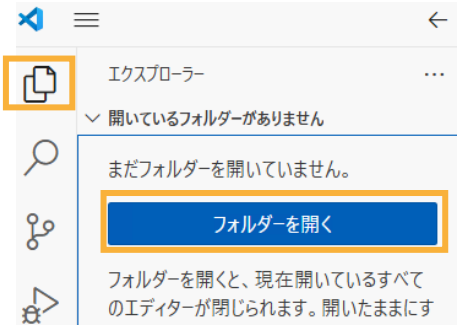
Visual COBOL for Windows 製品の Eclipse 版で作成したプロジェクトを Visual Studio Code で操作します。サンプルファイルを解凍したフォルダー eclipse 配下の tictac フォルダーを任意のフォルダーにコピーしてください。本書では、C:¥直下にコピーしています。異なる場所に配置した場合は、以降の手順にてパスを環境に合わせて変更してください。

1) Visual Studio Code の起動

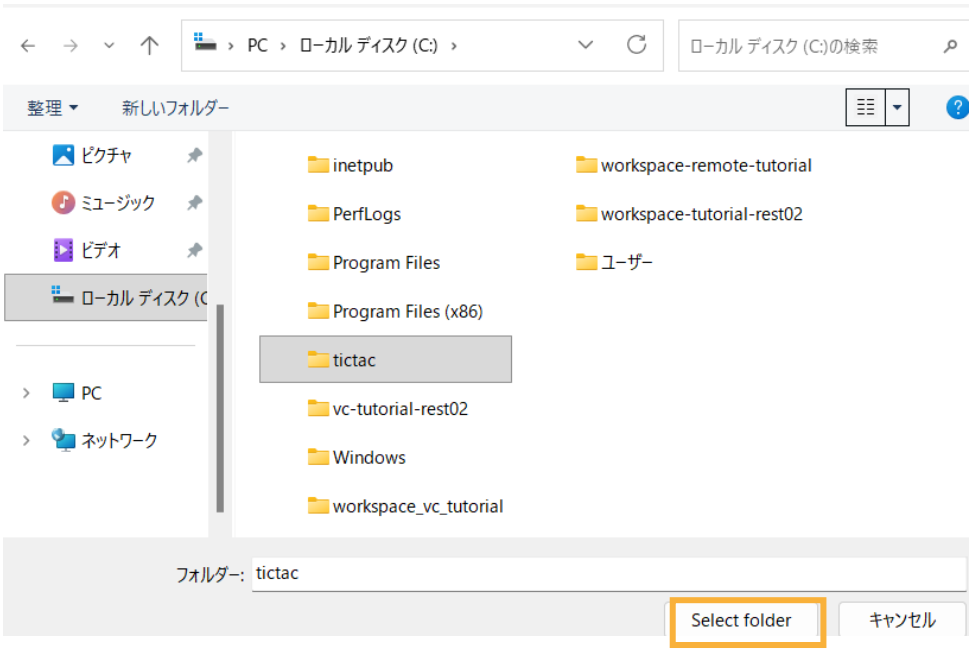
Visual Studio Code を起動します。

2) サンプルプロジェクトのオープン

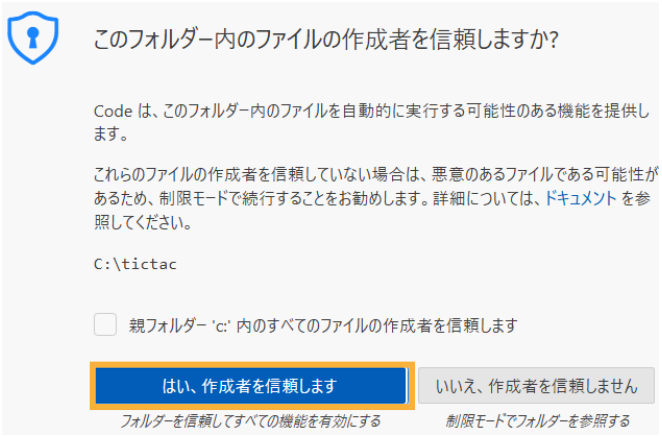
[エクスプローラー] アイコンをクリックして、エクスプローラーを開き、[フォルダーを開く] をクリックします。



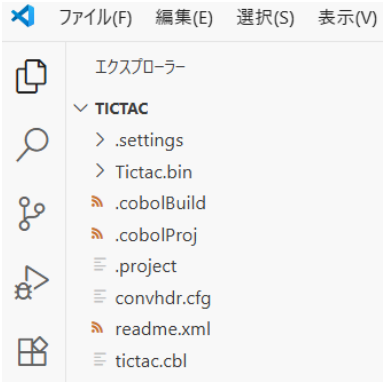
C:¥tictac フォルダーを選択して、[フォルダーの選択] をクリックします。



以下のダイアログでは、[はい、作成者を信頼します] をクリックします。

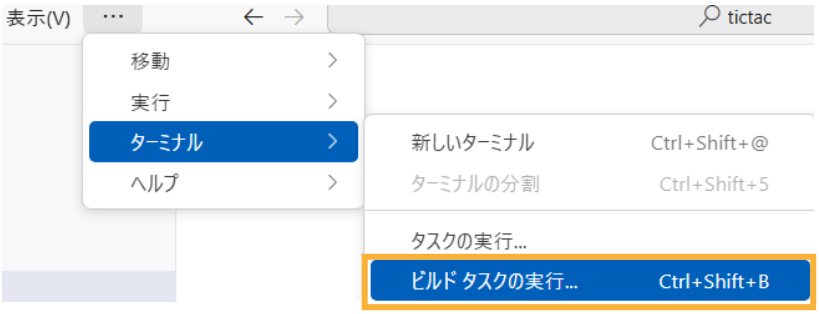


tictac フォルダーが IDE 上で開かれます。

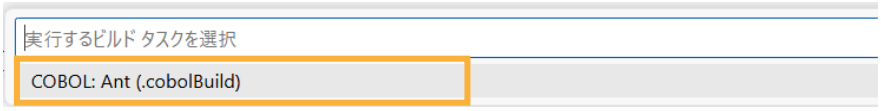


3) アプリケーションのビルド

メニューより [ターミナル] > [ビルド タスクの実行] を選択します。

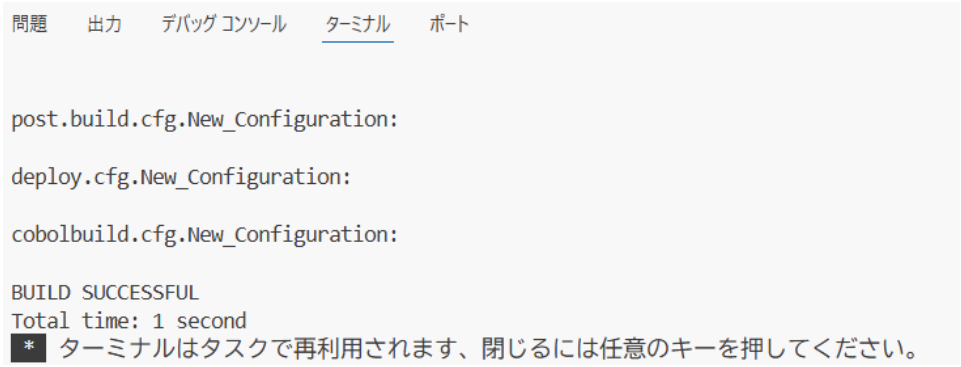


画面上部中央に以下が表示されたら、[COBOL: Ant (.cobolBuild)] を選択します。



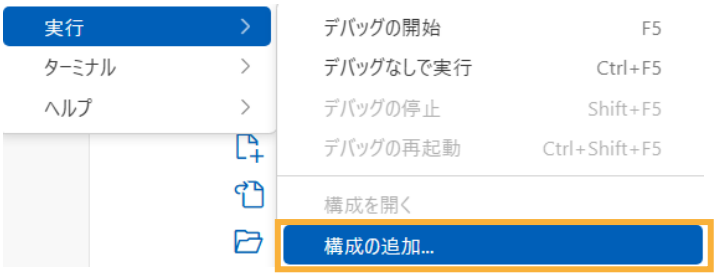
補足）
COBOL: Ant (.cobolBuild) が表示されない場合は、Visual COBOL for Eclipse 製品で有効なプロジェクトであるかをご確認ください。古いバージョンのプロジェクトでも、表示されないことがあります。その場合は、新しいバージョンで開き直したうえで再実行、もしくは、3.1.3 と同様、新規開発のようにタスクを作成します。

画面下部のターミナルで、ビルドが成功したことが確認できます。

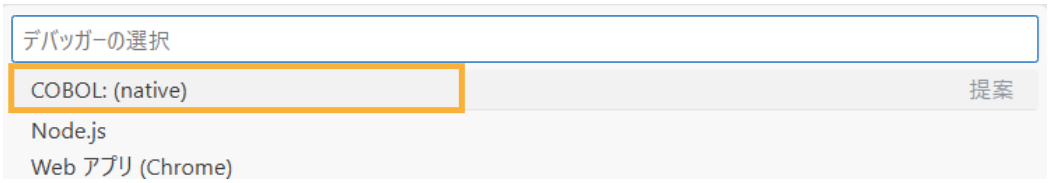


4) アプリケーションの実行

メニューより [実行] > [構成の追加] を選択します。



画面上部中央より、[COBOL: (native)] を選択します。



launch.json がエディター上にオープンされます。以下を修正したうえで保存します。

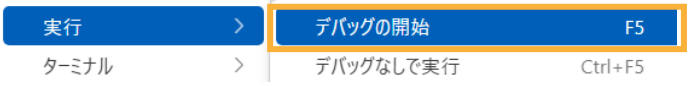
“COBOL (native) : Launch” の “program” の値

旧）"${workspaceFolder}/<insert-program-name-here>"

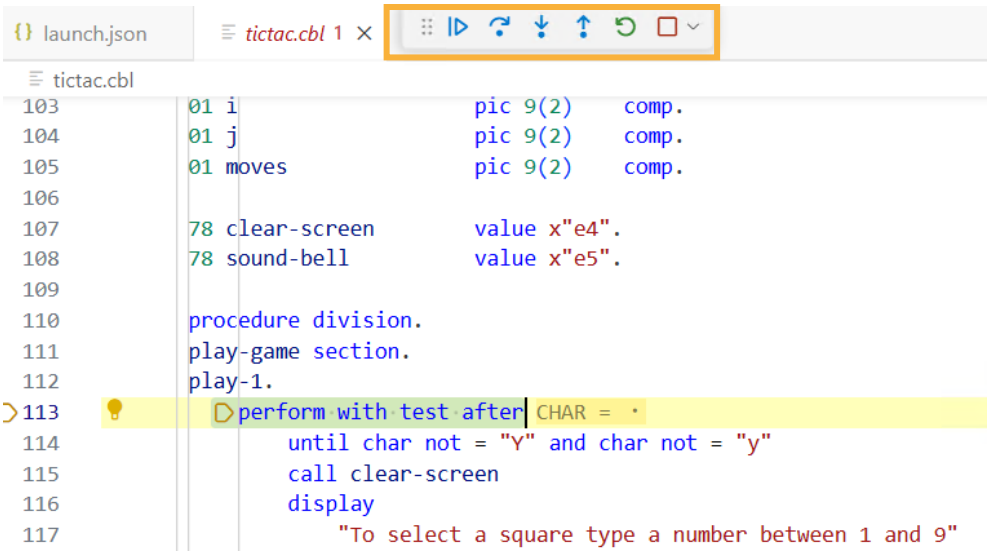
新）”${workspaceFolder}/Tictac.bin/Tictac.exe”

```
    "version": "0.2.0",
    "configurations": [
        {
            "type": "cobol",
            "request": "launch",
            "name": "COBOL (native): Launch",
            "program": "${workspaceFolder}/Tictac.bin/Tictac.exe",
            "cwd": "${workspaceFolder}",
            "stopOnEntry": true
        },
(以降。省略)
```

メニューより、[実行] > [デバッグの開始] を選択します。



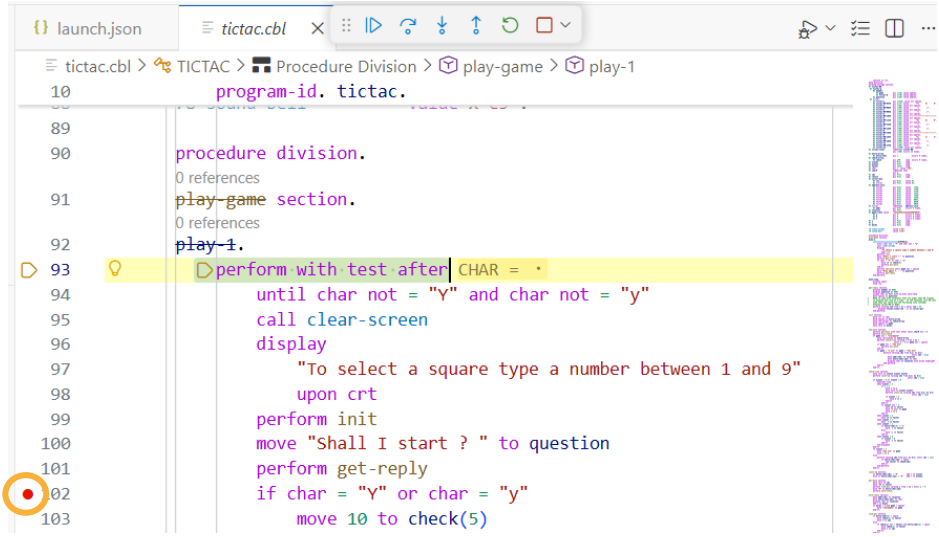
tictac.cbl の 113 行目でプログラムが停止します。



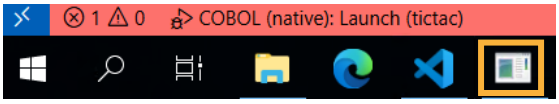
[実行] メニュー配下の項目、もしくは、上部に表示されるアイコンより、各種デバッグ操作が行えます。

[続行] アイコンをクリックします。

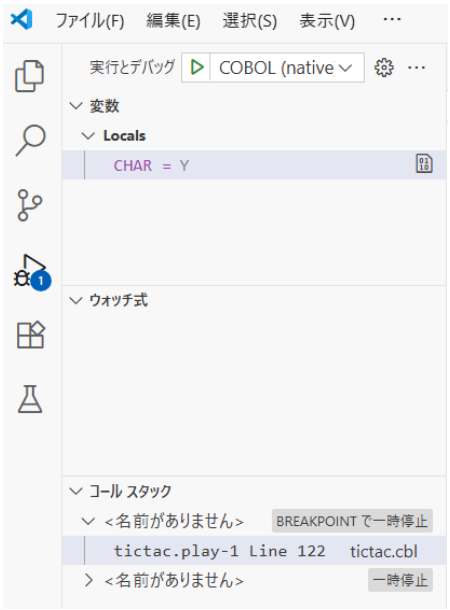
tictac.cbl の 102 行目の行番号左側をクリックして、ブレークポイントを設定します。



タスクバーのプロンプトタブをクリックして、アプリケーションの実行画面を開き、画面上で “Y” を入力すると、102 行目でプログラムが停止します。

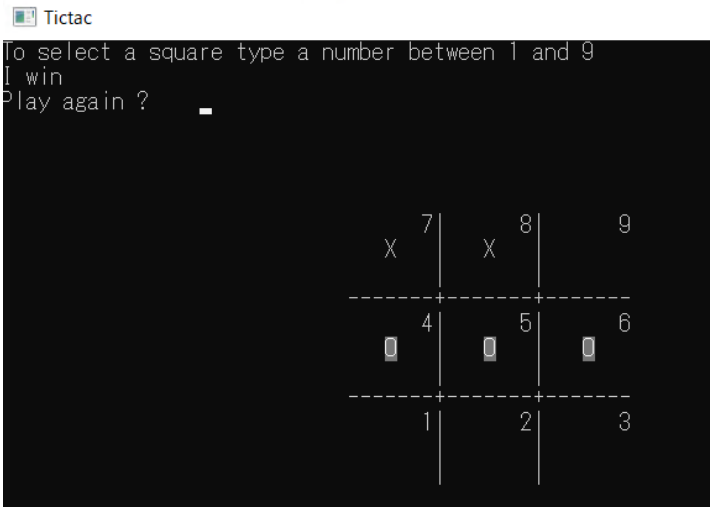


画面左側の [変数] > [Locals] にて、停止箇所での変数の値が確認できます。変数項目（ここでは、CHAR = Y） をクリックすると、変数の値を変更できます。



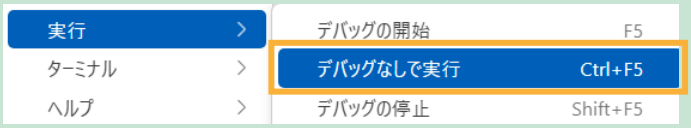
[続行] アイコンをクリックします。

アプリケーション画面に戻り、”7”, “8” と入力すると、以下の画面が表示されますので、”N” を入力してアプリケーションを終了します。



補足）

デバッグ実行しない場合は、メニューより [実行] > [デバッグなしで実行] を選択してアプリケーションを起動してください。



3.1.2 Visual Studio ソリューション・プロジェクトの利用

Visual COBOL for Windows 製品の Visual Studio 版で作成したプロジェクトを Visual Studio Code で操作します。

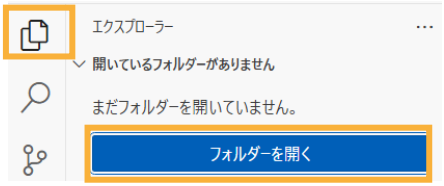
サンプルファイルを解凍したフォルダー visualstudio 配下の tictac フォルダーを任意のフォルダーにコピーしてください。本書では、C:¥直下にコピーしています。異なる場所に配置した場合は、以降の手順にてパスを環境に合わせて変更してください。

1) Visual Studio Code の起動

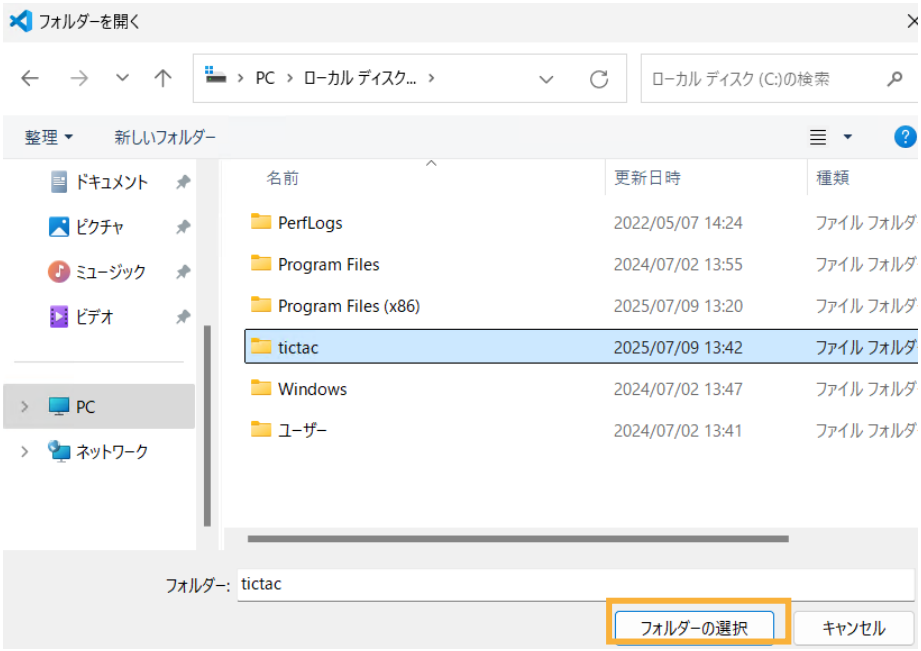
Visual Studio Code を起動します。

2) サンプルソリューションのオープン

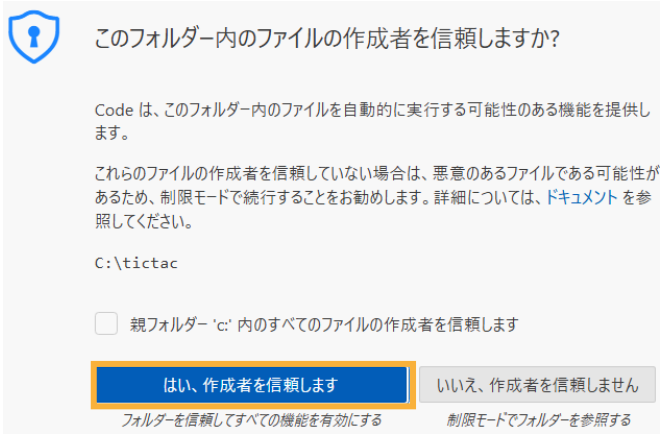
[エクスプローラー] アイコンをクリックして、エクスプローラーを開き、[フォルダーを開く] をクリックします。



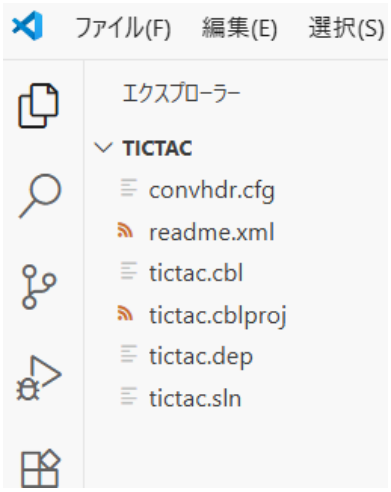
C:¥tictac フォルダーを選択して、[フォルダーの選択] をクリックします。



以下のダイアログでは、[はい、作成者を信頼します] をクリックします。

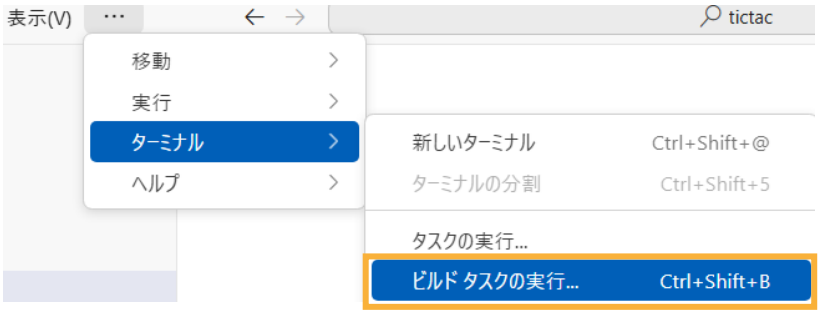


tictac フォルダーが IDE 上で開かれます。

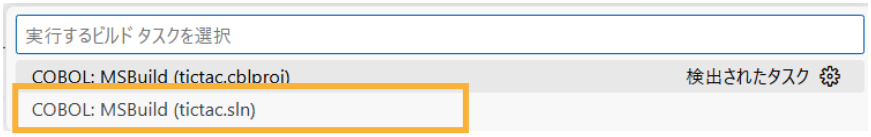


3) アプリケーションのビルド

メニューより [ターミナル] > [ビルド タスクの実行] を選択します。



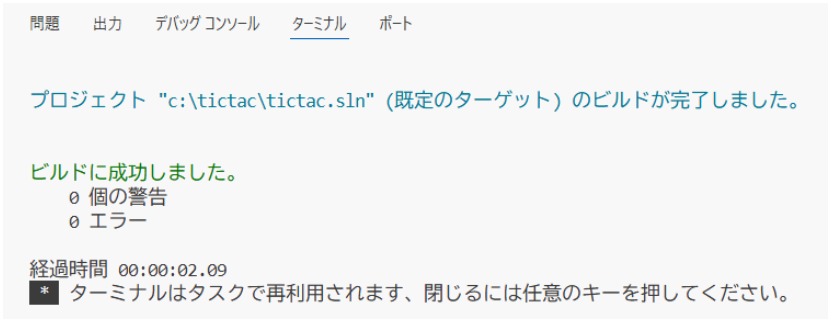
画面上部中央に以下が表示されたら、[COBOL: MSBuild (tictac.sln)] を選択します。



補足）

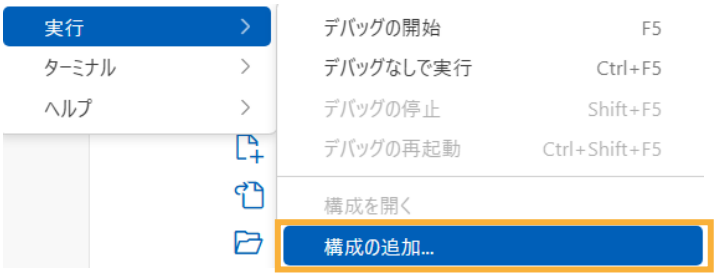
.cblproj を選択することで、Visual Studio IDE と同様、プロジェクト単位にビルドを行うこともできます。

画面下部のターミナルで、ビルドが成功したことが確認できます。

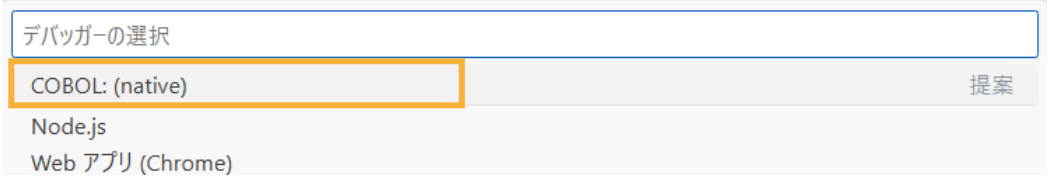


4) アプリケーションの実行

メニューより [実行] > [構成の追加] を選択します。



画面上部中央より、[COBOL: (native)] を選択します。

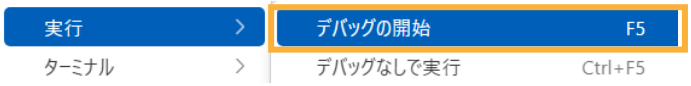


launch.json がエディター上にオープンされるので、以下で上書き保存します。

```
{
    // IntelliSense を使用して利用可能な属性を学べます。
    // 既存の属性の説明をホバーして表示します。
    // 詳細情報は次を確認してください: https://go.microsoft.com/fwlink/?linkid=830387
    "version": "0.2.0",
    "configurations": [
        {
            "type": "cobol",
            "request": "launch",
            "name": "COBOL (native): Launch",
            "program": "${workspaceFolder}/bin/x64/Debug/tictac.exe",
            "cwd": "${workspaceFolder}",
            "is64Bit": true,
            "stopOnEntry": true

        }
    ]
}
```

メニューより、[実行] > [デバッグの開始] を選択します。



デバッグ操作については、3.1.1 と同様です。

3.1.3 Visual Studio Code で新たにアプリケーション開発

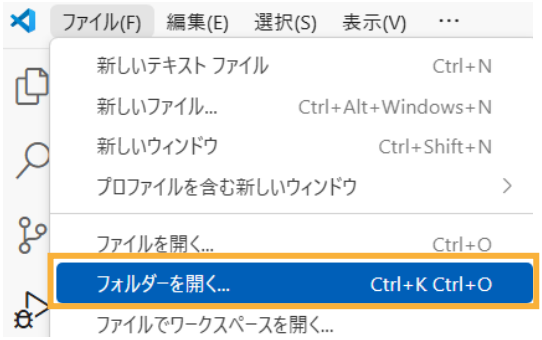
Eclipse や Visual Studio 同様、開発リソースやビルド成果物をまとめるため、任意のフォルダー（本書では c:¥vsfolder）を使用します。3.1.1、もしくは 3.1.2 で使用したサンプルファイル tictac.cbl を使用するフォルダー配下にコピーしてください。異なるフォルダーを使用する場合は、以降の手順にてパスを更新してください。

1) Visual Studio Code の起動

Visual Studio Code を起動します。

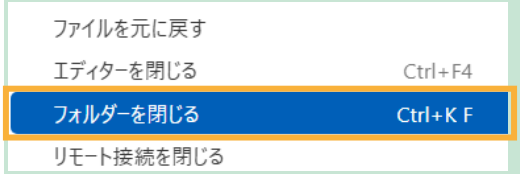
2) 作業フォルダーのオープン

メニューより、[ファイル] > [フォルダーを開く] を選択します。

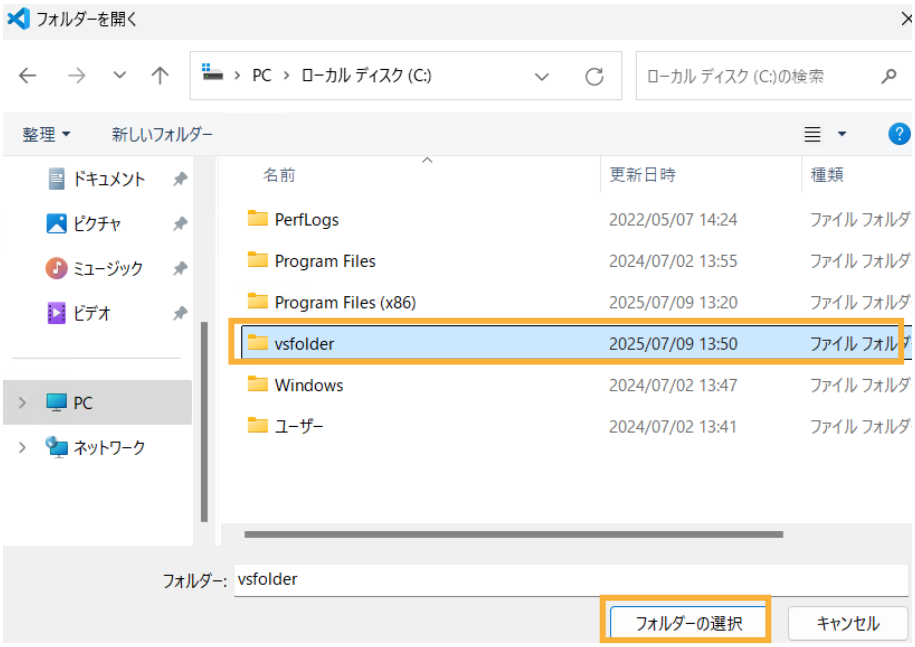


補足）

前手順の実施などにより、別なフォルダーをオープンしている場合は、先にメニューより［ファイル］＞［フォルダーを閉じる］を選択してください。



C:¥vsfolder を選択して、[フォルダーの選択] をクリックします。



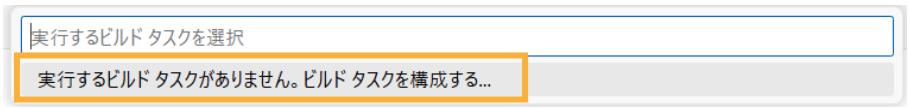
以下のダイアログでは、[はい、作成者を信頼します] をクリックします。

IDE のエクスプローラーに、tictac.cbl が表示されます。

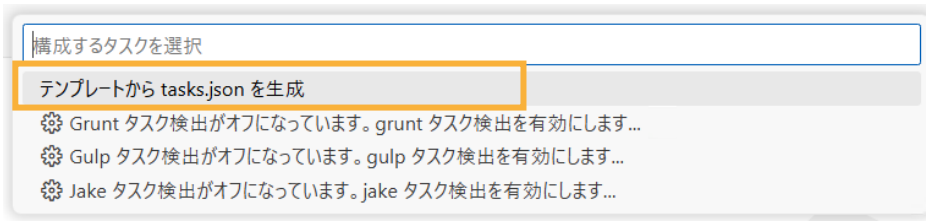
3) アプリケーションのビルド

メニューより、[ターミナル] > [ビルド タスクの実行] を選択します。

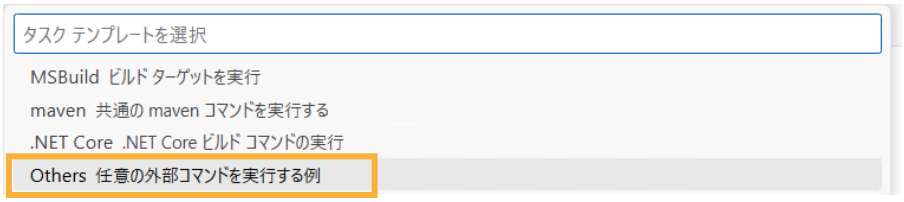
画面上部中央に表示される [実行するビルド タスクがありません。ビルド タスクを構成する] を選択します。



[テンプレートから tasks.json を作成] を選択します。



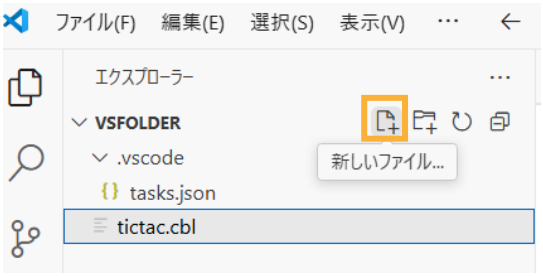
[Others 任意の外部コマンドを実行する例] を選択します。



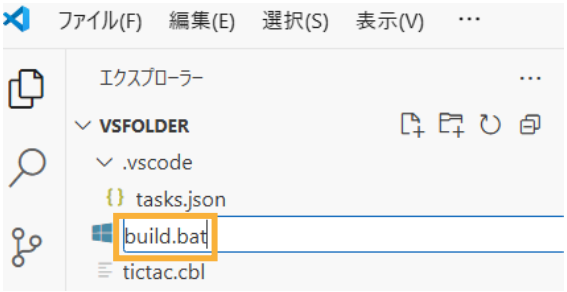
tasks.json が作成され、エディター上にオープンされるので、以下で上書き保存します。

```
{
    // See https://go.microsoft.com/fwlink/?LinkId=733558
    // for the documentation about the tasks.json format
    "version": "2.0.0",
    "tasks": [
        {
            "label": "COBOL-build(64bit)",
            "type": "shell",
            "command": ".¥¥build.bat",
            "problemMatcher": ["$COBOLErrFormat3"],
            "group": "build"
        }
    ]
}
```

Visual Studio IDE のエクスプローラー上の tictac.cbl をクリックしたうえで、 VSFOLDER 欄にカーソルを合わせて表示される [新規ファイル] アイコンをクリックします。



ファイル名に “build.bat” を入力して、Enter を押します。

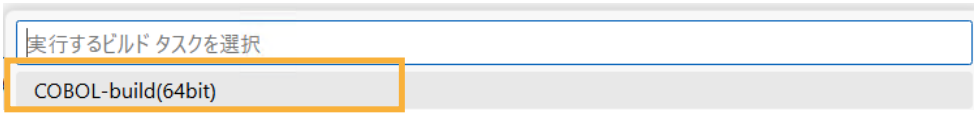


エディター上に build.bat がオープンされるので、以下で上書き保存します。

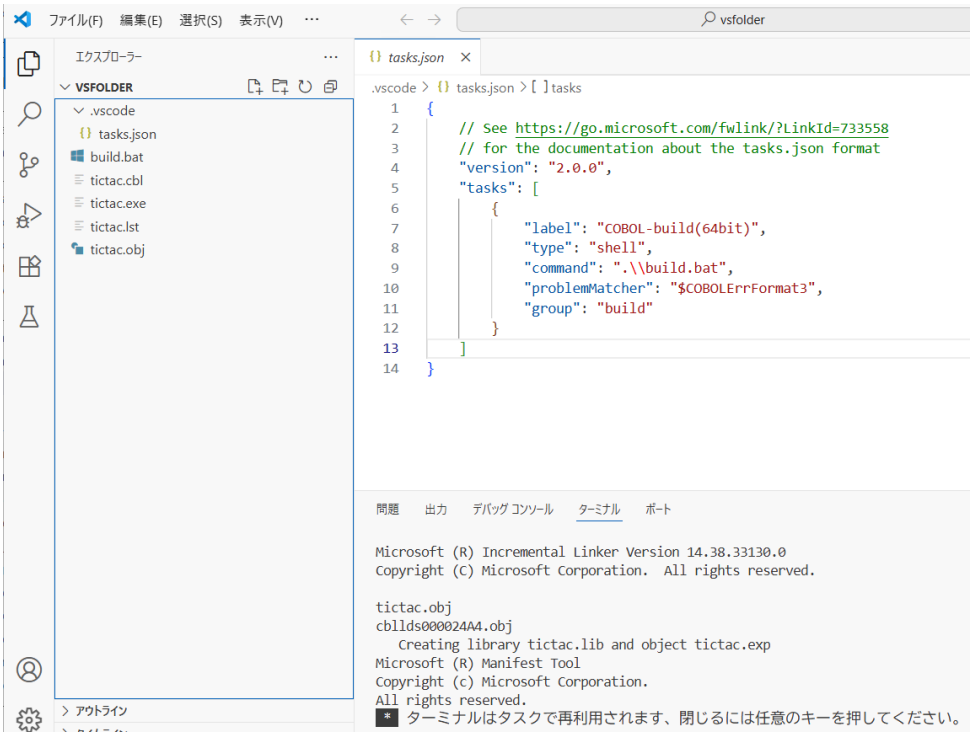
```
call "C:¥Program Files (x86)¥Rocket Software¥Visual COBOL¥setupenv.bat" 64
cobol tictac.cbl list() anim;
cbllink tictac.obj
```

メニューより、 [ターミナル] > [ビルド タスクの実行] を選択します。

画面上部中央より [COBOL-build(64bit)] を選択します。

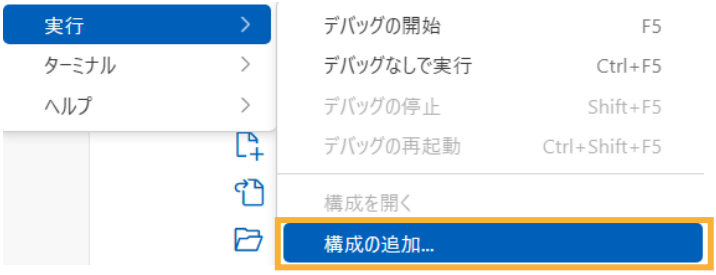


コンパイルとリンクが行われ、tictac.exe などがエクスプローラー上に表示されます。

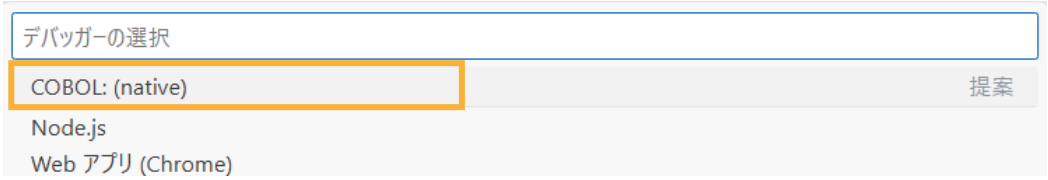


4) アプリケーションの実行

メニューより [実行] > [構成の追加] を選択します。



画面上部中央より、[COBOL: (native)] を選択します。



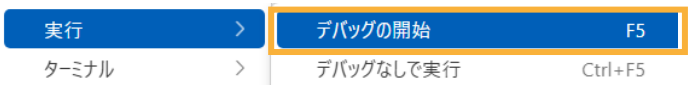
launch.json がエディター上にオープンされるので、以下で上書き保存します。

```
{
    // IntelliSense を使用して利用可能な属性を学べます。
    // 既存の属性の説明をホバーして表示します。
    // 詳細情報は次を確認してください: https://go.microsoft.com/fwlink/?linkid=830387
    "version": "0.2.0",
    "configurations": [
        {
            "type": "cobol",
            "request": "launch",
            "name": "COBOL (native): Launch",
            "program": "${workspaceFolder}/tictac.exe",
            "cwd": "${workspaceFolder}",
            "is64Bit": true,
            "stopOnEntry": true
        },
    ]
}
```

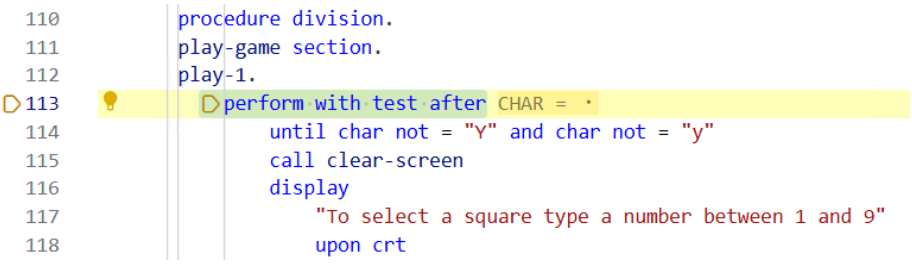
補足）

64bit アプリケーションのため、”is64Bit” 項目が true で設定されています。未設定時のデフォルトは false、すなわち、32bit アプリケーションが対象になります。

メニューより、[実行] > [デバッグの開始] を選択します。



113 行目で停止します。



デバッグ操作については、3.1.1 と同様です。

3.2 Linux 環境

Rocket COBOL プラグインが未インストールの場合は、4.1.2 を参照のうえ、インストールしてください。

なお、既存のリモート開発プロジェクトを使用する際は、Eclipse IDE でリモート COBOL プロジェクトが適切にデバッグ実行まで行えることを確認してください。特に、ファイアウォール機能などによる通信制御にはご注意ください。

サンプルファイルを解凍したフォルダー RemoteDev 配下には、以下の 2 つのフォルダーがあります。

- RemoteTictac

  xterm を Windows 上で表示してデバッグ実行を行います。画面入出力があるアプリケーションや、SJIS など UTF-8 以外の日本語を扱うアプリケーション開発では、こちらをご参照ください。

- RemoteSSHOnly

  xterm は不要で、Visual Studio Code 上のコンソールビュー上でデバッグ実行を行います。上記に当てはまらないアプリケーション開発では、こちらをご参照ください。

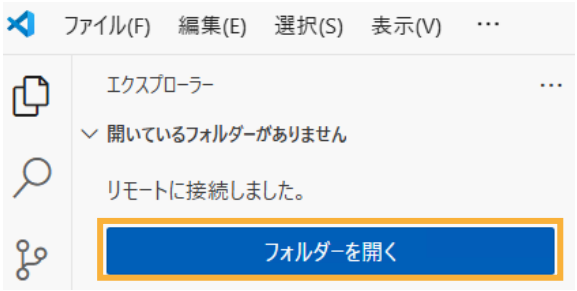
上記フォルダーを、Linux サーバーに転送してください。本手順では、/home/ruser 配下に転送しています。

また、プラグインインストール手順記載の 4.1.2 では、Linux サーバーへの接続手順までが含まれています。チュートリアルを実施する前に、Linux サーバーへの接続までを行ってください。

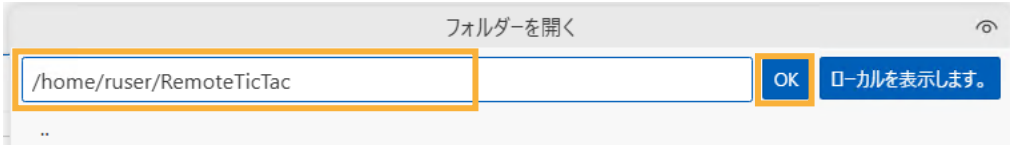
3.2.1 既存のリモート開発プロジェクトの使用 / RemoteTictac

1) Linux フォルダーのオープン

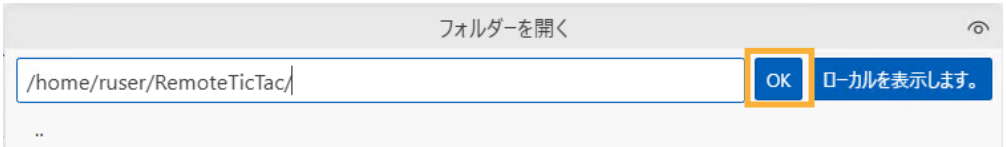
[フォルダーを開く] をクリックします。



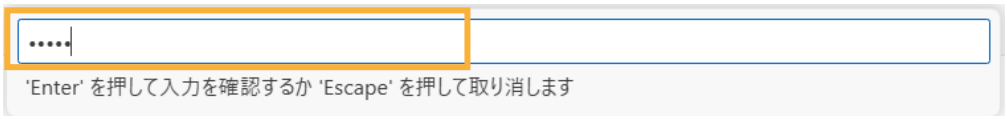
画面中央に以下が表示されたら、"/home/ruser/RemoteTicTac" を入力して、[OK] をクリックします。



そのまま [OK] をクリックします。



パスワード入力を求められた場合は、パスワードを入力して Enter を押します。

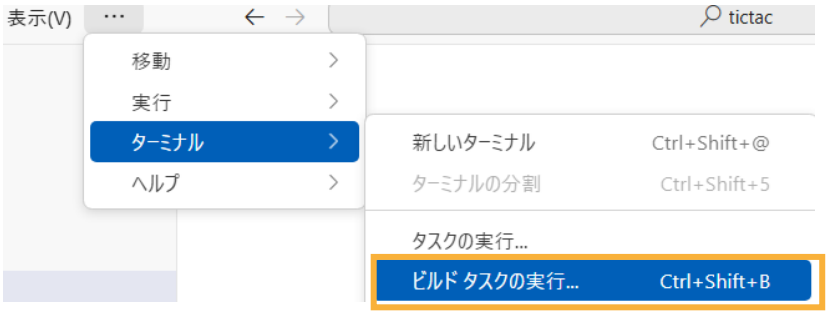


以下のダイアログでは、[はい、作成者を信頼します] をクリックします。

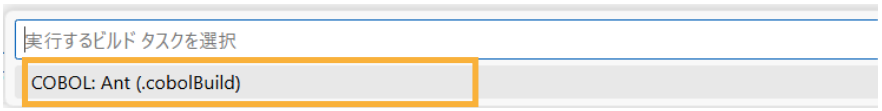
RemoteTicTac フォルダーがエクスプローラー上にオープンされます。

2) アプリケーションのビルド

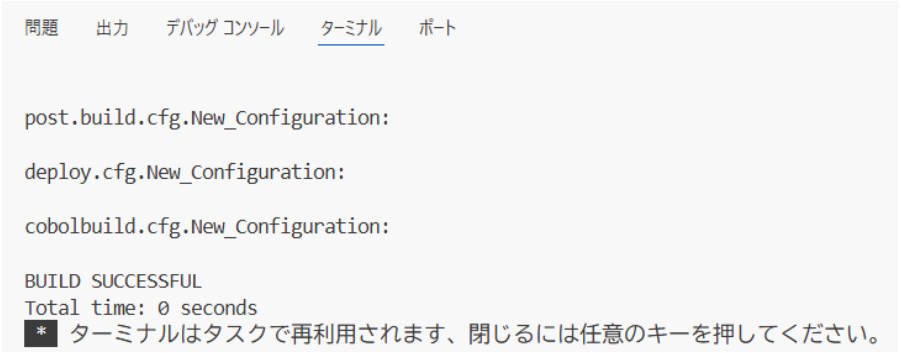
メニューより [ターミナル] > [ビルド タスクの実行] を選択します。



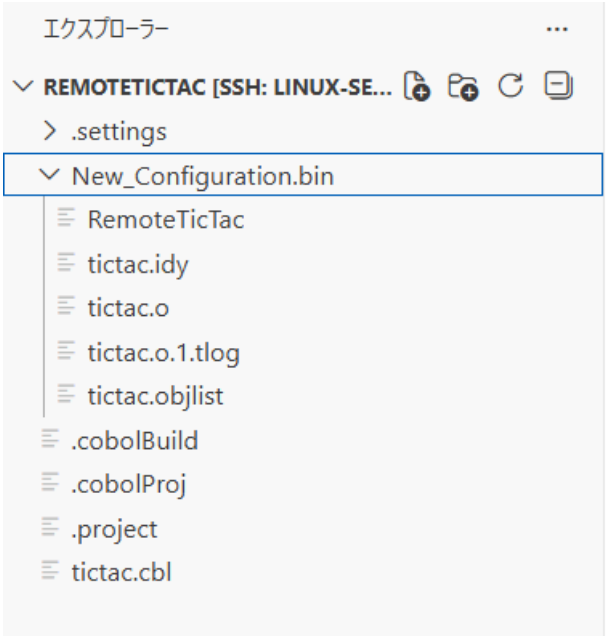
画面上部中央に以下が表示されたら、[COBOL: Ant (.cobolBuild)] を選択します。



ターミナルビューにて、ビルドが正常終了したことが確認できます。



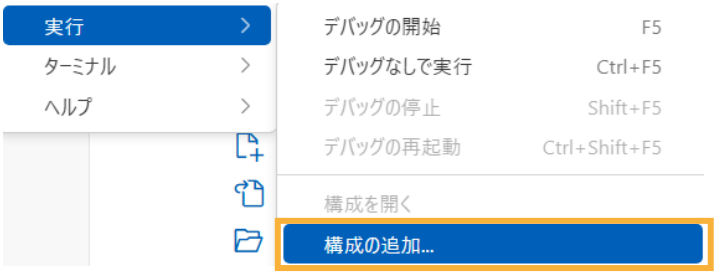
また、エクスプローラー上の New_Configuration.bin 配下に、RemoteTicTac の実行モジュールなどが生成されます。



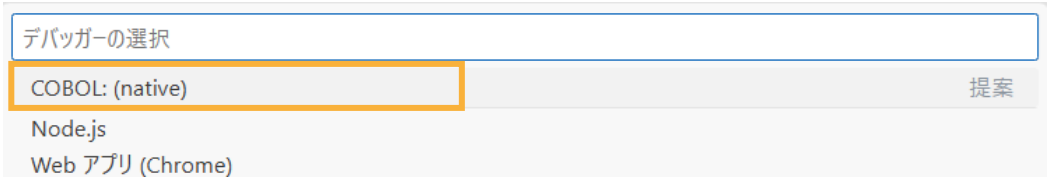
3) アプリケーションの実行

xterm を Windows 上で表示するため、XServer を起動します。

メニューより [実行] > [構成の追加] を選択します。



画面上部中央より、[COBOL: (native)] を選択します。

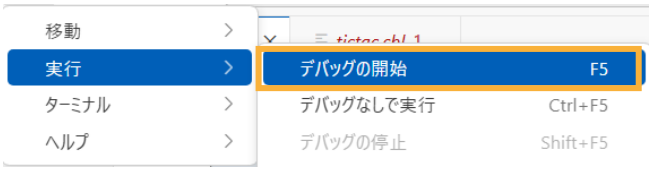


launch.json がエディター上にオープンされるので、以下で上書き保存します。

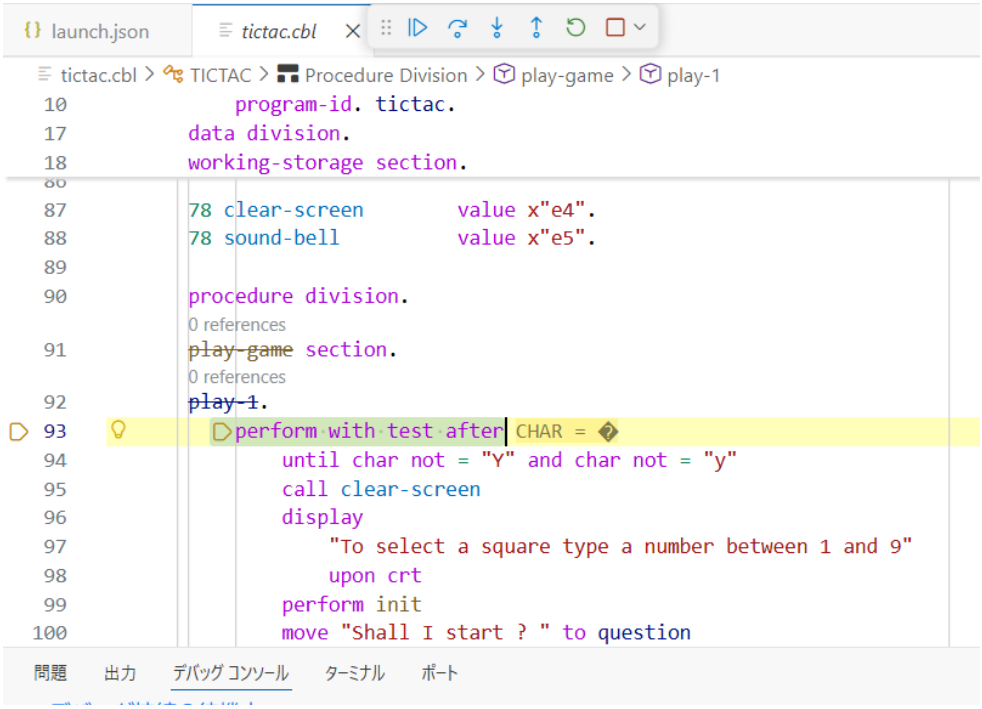
なお、<Windows の アドレス>の箇所は、お使いの Windows のホスト、 IP アドレスなど、Linux サーバーからアクセス可能な値を指定します。

```
{
    // IntelliSense を使用して利用可能な属性を学べます。
    // 既存の属性の説明をホバーして表示します。
    // 詳細情報は次を確認してください: https://go.microsoft.com/fwlink/?linkid=830387
    "version": "0.2.0",
    "configurations": [
        {
            "type": "cobol",
            "request": "launch",
            "name": "COBOL (native): remote",
            "program": "${workspaceFolder}/New_Configuration.bin/RemoteTicTac",
            "is64Bit": false,
            "stopOnEntry": true,
            "console": "externalTerminal",
            "env": [
                { "name": "DISPLAY", "value": "<Windows の IP アドレス>:0.0"},
                { "name": "TERM", "value": "xterm"}
            ]
        },
    ]
}
```

メニューより、[実行] > [デバッグの開始] を選択します。



xterm ウィンドウが起動し、エディター上では tictac.cbl の 93 行目で停止します。

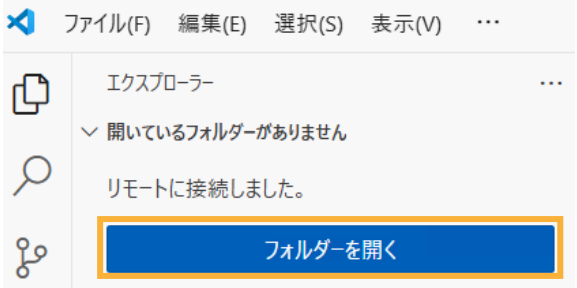


デバッグ操作については、3.1.1 と同様です。

3.2.2 Visual Studio Code で新規にリモート開発 / RemoteSSHOnly

1) Linux フォルダーのオープン

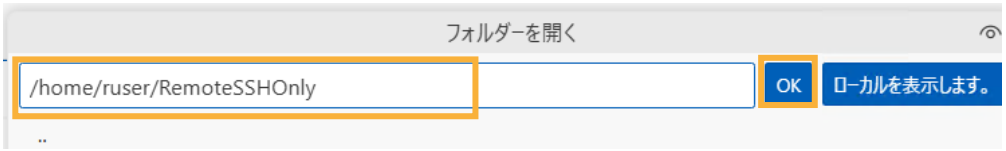
[フォルダーを開く] をクリックします。



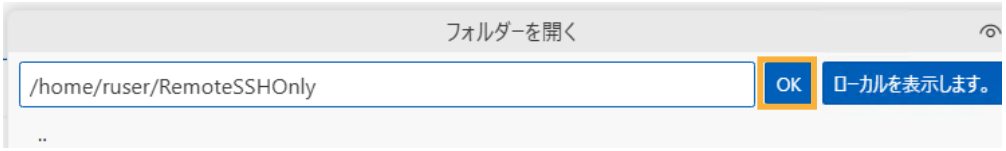
補足）

前手順の実施などにより、別なフォルダーをオープンしている場合は、先にメニューより [ファイル] > [フォルダーを閉じる] を選択してください。

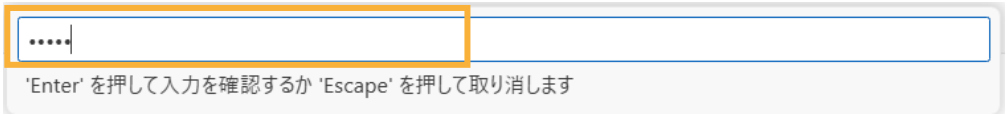
画面中央に以下が表示されたら、”/home/ruser/RemoteSSHOnly” を入力して、[OK] をクリックします。

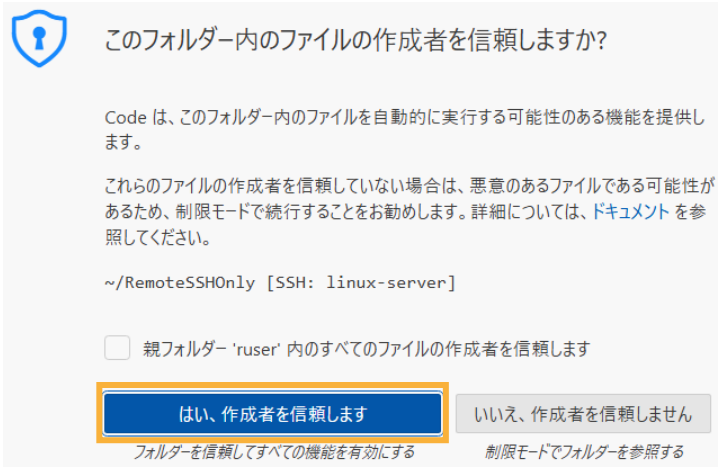


そのまま [OK] をクリックします。



パスワード入力を求められた場合は、パスワードを入力して Enter を押します。



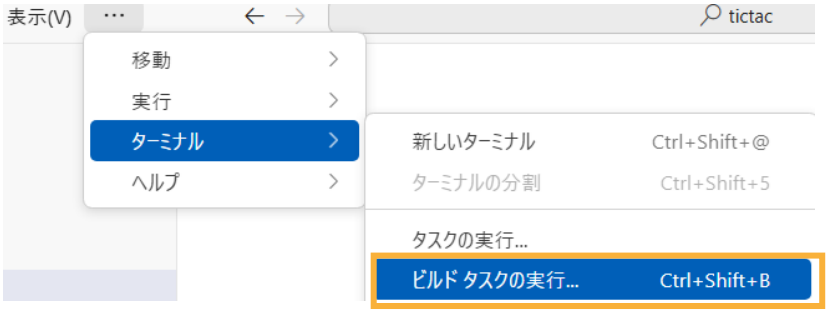


以下のダイアログでは、[はい、作成者を信頼します] をクリックします。

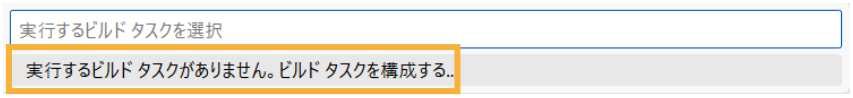
RemoteSSHOnly フォルダーがエクスプローラー上にオープンされます。

2) アプリケーションのビルド

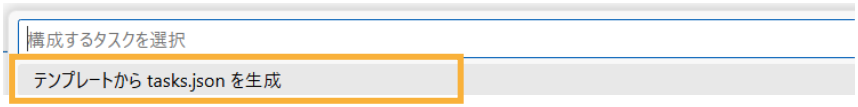
メニューより [ターミナル] > [ビルド タスクの実行] を選択します。



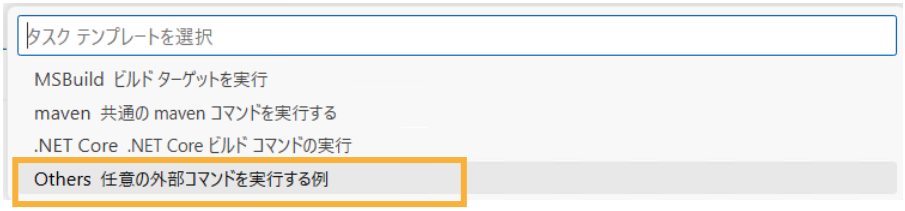
画面上部中央に以下が表示されたら、[実行するビルドタスクがありません。ビルド タスクを構成する] を選択します。



[テンプレートから tasks.json を生成] を選択します。



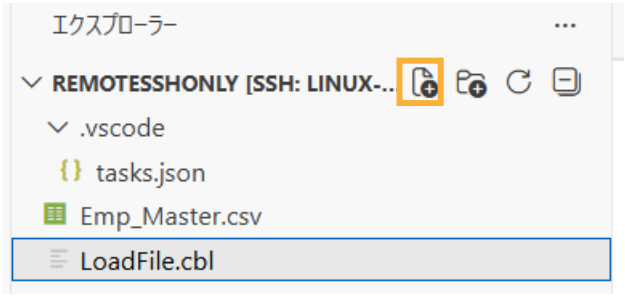
[Others 任意の外部コマンドを実行する例] を選択します。



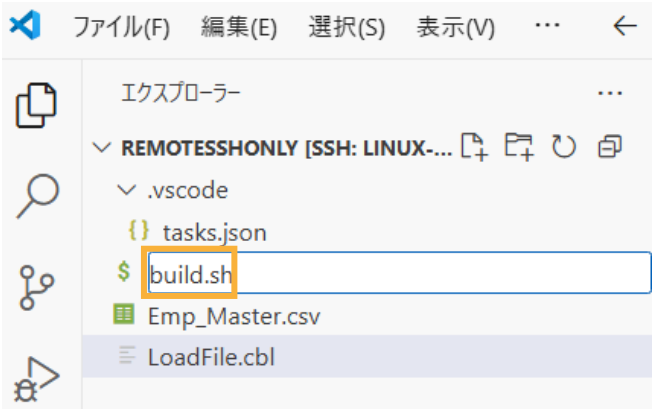
tasks.json が作成され、エディター上にオープンされるので、以下で上書き保存します。

```
{
    // See https://go.microsoft.com/fwlink/?LinkId=733558
    // for the documentation about the tasks.json format
    "version": "2.0.0",
    "tasks": [
        {
            "label": "COBOL(build)",
            "type": "shell",
            "command": "sh build.sh",
            "problemMatcher": ["$COBOLErrFormat3"],
            "group": "build"
        }
    ]
}
```

LoadFile.cbl をクリックした後、[REMOTESSHONLY [SSH: LINUX-SERVER]] にカーソルを合わせて表示されるアイコンから [新しいファイル] アイコンをクリックします。



ファイル名を “build.sh” と入力して Enter を押します。



エディターに build.sh がオープンされるので、以下で上書き保存します。

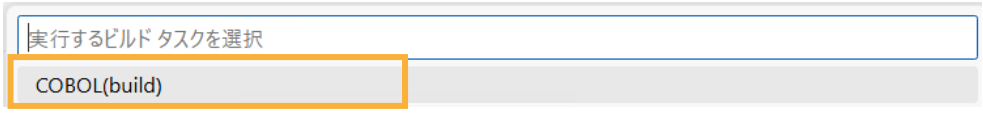
```
. /opt/rocketsoftware/VisualCOBOL/bin/cobsetenv
cob -xg LoadFile.cbl -C"anim assign(external) errformat(3)"
```

注意）

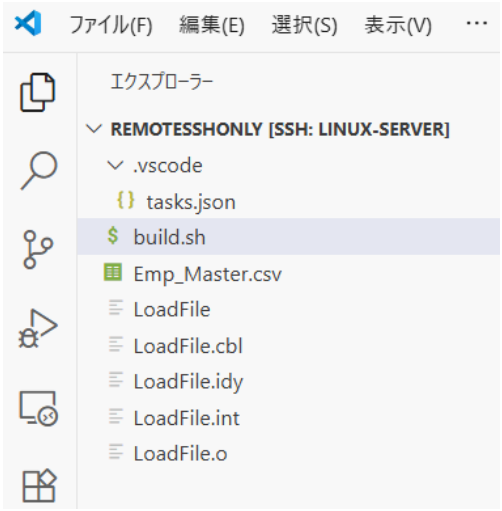
1 行目は、Visual COBOL 製品のインストール先に合わせてパスを変更してください。

メニューより、 [ターミナル] > [ビルド タスクの実行] を選択します。

画面上部中央より [COBOL(build)] を選択します。

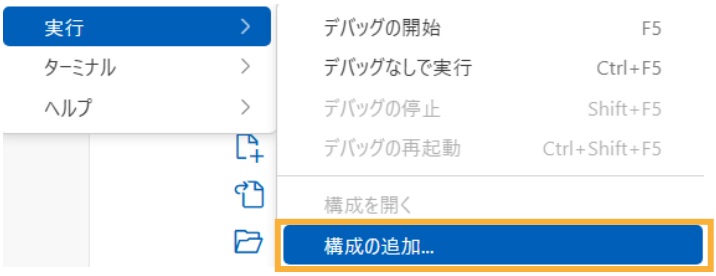


コンパイルとリンクが行われ、LoadFile などがエクスプローラー上に表示されます。

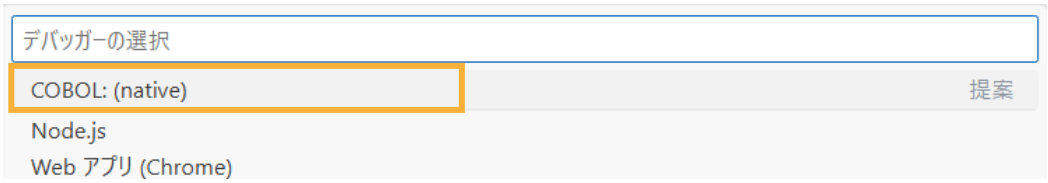


3) アプリケーションの実行

メニューより [実行] > [構成の追加] を選択します。



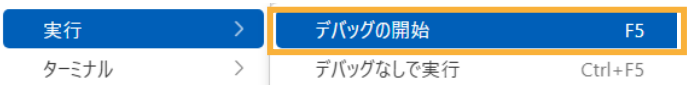
画面上部中央より、[COBOL: (native)] を選択します。



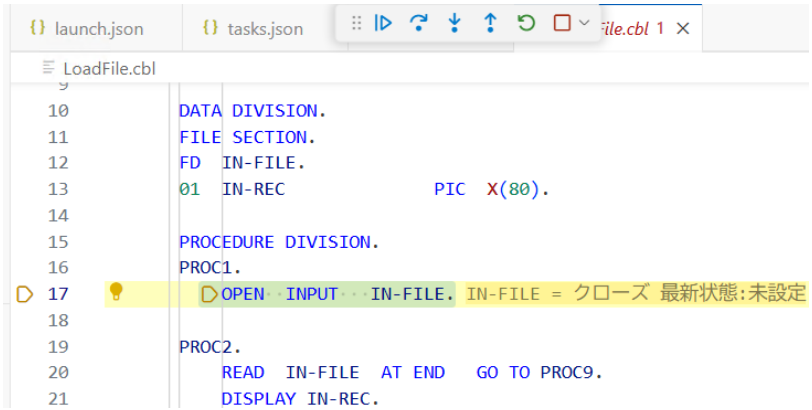
.json がエディター上にオープンされるので、以下で上書き保存します。

```
{
    // IntelliSense を使用して利用可能な属性を学べます。
    // 既存の属性の説明をホバーして表示します。
    // 詳細情報は次を確認してください: https://go.microsoft.com/fwlink/?linkid=830387
    "version": "0.2.0",
    "configurations": [
        {
            "type": "cobol",
            "request": "launch",
            "name": "COBOL (native): Launch",
            "program": "${workspaceFolder}/LoadFile",
            "cwd": "${workspaceFolder}",
            "stopOnEntry": true,
            "console": "integratedTerminal",
            "env":[
                {"name": "INFILE", "value": "Emp_Master.csv"}
            ]
        },
    ]
}
```

メニューより、[実行] > [デバッグの開始] を選択します。



17 行目で停止します。



デバッグ操作については、3.1.1 と同様です。

アプリケーションの出力は、ターミナルビュー上に行われます。

```
$  /usr/bin/env /home/ruser/.vscode-server/extensions/rocketsoftware.rocket-cobol-2.
0.12/bin/debugadapter/linux-x64/MicroFocus.VsCodeDebugProtocol /pipe:cdr_4678
11111113,佐藤,隆,SATO,TAKASHI,M,営業部,19980401,0
22222226,鈴木,尚之,SUZUKI,NAOYUKI,M,技術部,19981015,0
33333339,田中,直美,TANAKA,NAOMI,F,総務部,19990401,0
44444442,山田,洋一,YAMADA,YOICHI ｦ,M,営業部,20000701,0
55555555,伊藤,弘子,ITO,HIROKO,F,技術部,20010401,0
66666668,木村,貴弘,KIMURA,TAKAHIRO,M,営業部,20021220,0
77777771,中村,慎司,NAKAMURA,SINJI,M,技術部,20030401,0
88888884,橋本,悦子,HASHIMOTO,ETSUKO,F,総務部,20040805,0
99999997,三井,薫,MITSUI,KAORU,F,営業部,20050401,0
$
```

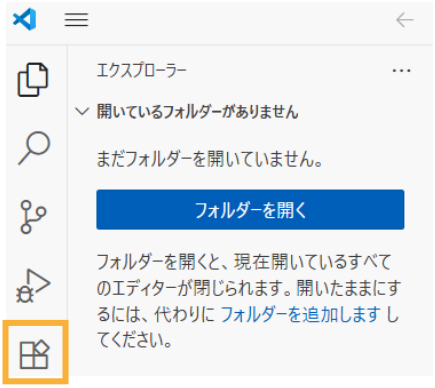
4 補足

4.1 プラグインインストール

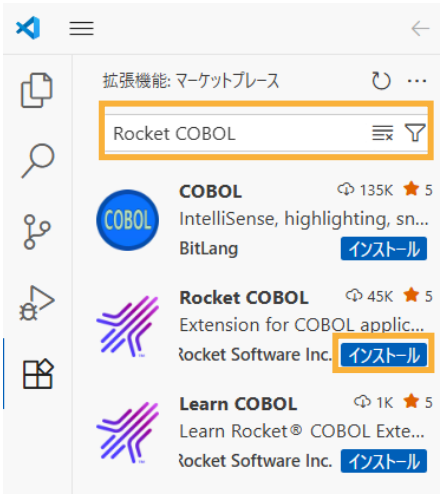
4.1.1 Windows 環境

1) Visual Studio Code を起動します。

2) 画面左より拡張機能のアイコンをクリックします。

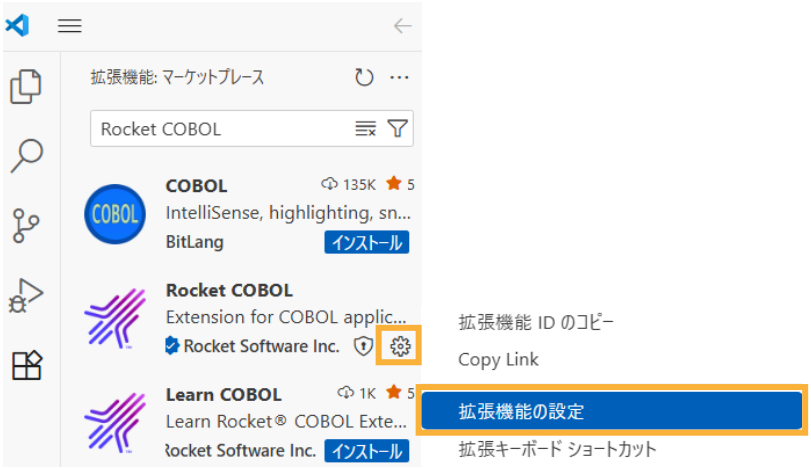


3) 検索欄に “Rocket COBOL” を入力して、[Rocket COBOL] の [インストール] をクリックします。



以下のダイアログが表示された場合は、[発行元を信頼してインストールする] をクリックします。

4) インストール完了後に表示される設定アイコンをクリックしたうえで表示されるポップアップメニューより [設定] を選択します。



5) 必要な設定を行います。

注意）
各種項目のパスは、お使いの環境に合わせて設定してください。

Visual COBOL for Windows (Eclipse) 製品をインストールした環境では、以下を設定します。

- Rocket COBOL > Ant: Ant_home Path
  C:¥Users¥Public¥Rocket Software¥Visual COBOL¥eclipse¥plugins¥org.apache.ant_1.10.14.v20230922-1200
- Rocket COBOL > Ant: Ant_home Path Resolve:
  [User Defined] を選択
- Rocket COBOL > Java: Java_home Path
  C:¥Program Files (x86)¥Rocket Software¥Visual COBOL¥AdoptOpenJDK
- Rocket COBOL > Java: Java_home Path Resolve
  [User Defined] を選択
- Rocket COBOL: Install Location
  C:¥Program Files (x86)¥Rocket Software¥Visual COBOL

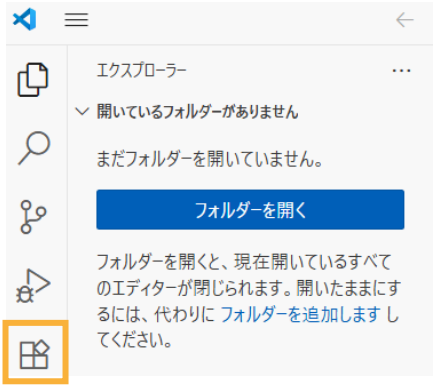
Visual COBOL for Windows (Visual Studio) 製品をインストールした環境では、以下を入力します。

- Rocket COBOL: Install Location
  C:¥Program Files (x86)¥Rocket Software¥Visual COBOL
- Rocket COBOL.MS Build: Path
  C:¥Program Files¥Microsoft Visual Studio¥2022¥Professional¥MSBuild¥Current¥Bin¥MSBuild.exe
- Rocket COBOL.MS Build: Resolve
  [User Defined] を選択

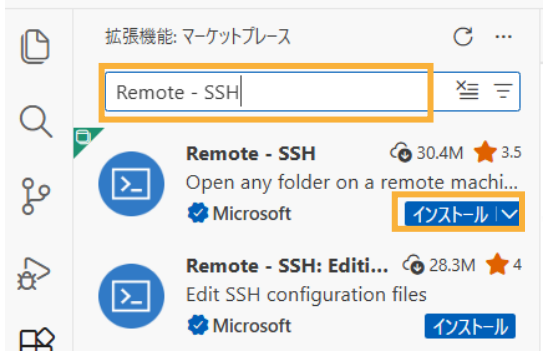
4.1.2 Linux 環境

1) Visual Studio Code を起動します。

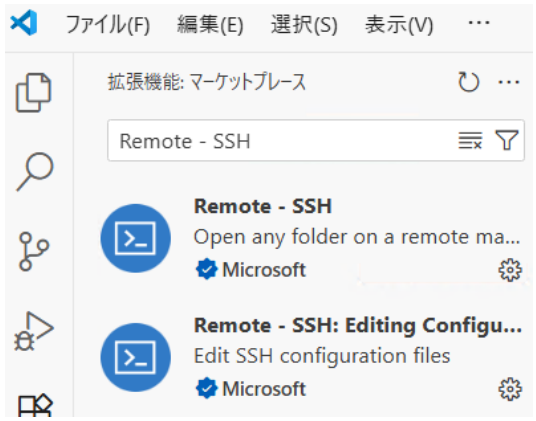
2) 画面左より拡張機能のアイコンをクリックします。



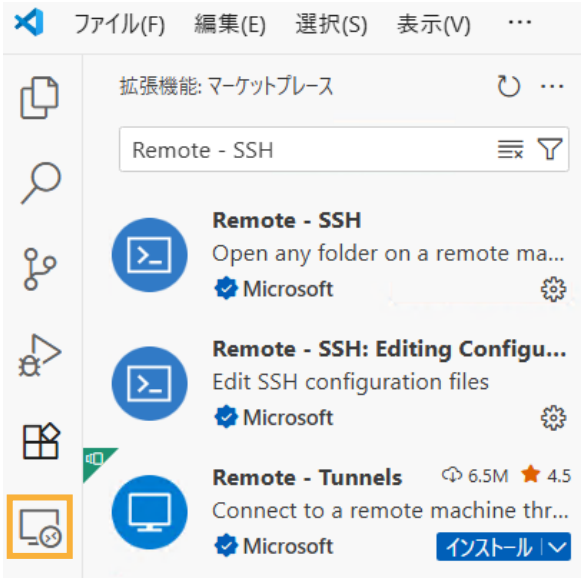
3) 検索欄に “Remote - SSH” を入力して、[Remote - SSH] の [インストール] をクリックします。



インストールが完了すると、歯車アイコンが表示されます。

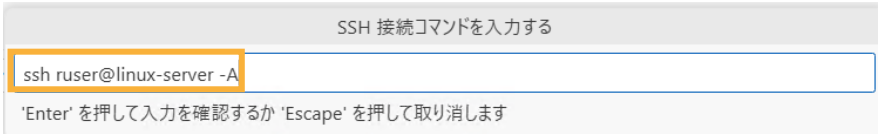


4) 左側のメニューから、[リモート エクスプローラー] ボタンをクリックします。



5) SSH 欄にカーソルを合わせた際に表示される [+] ボタンをクリックします。

6) 画面上部中央に以下が表示されたら、“ssh ruser@linux-server -A" を入力して Enter を押します。



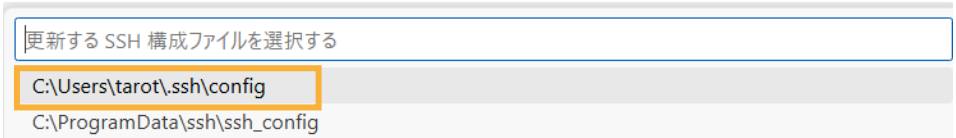
補足）

上記入力は、以下を指定しています。お使いの環境にあわせて変更してください。

ruser: リモート先のユーザー名

linux-server: リモート先のホスト名、または IP アドレス

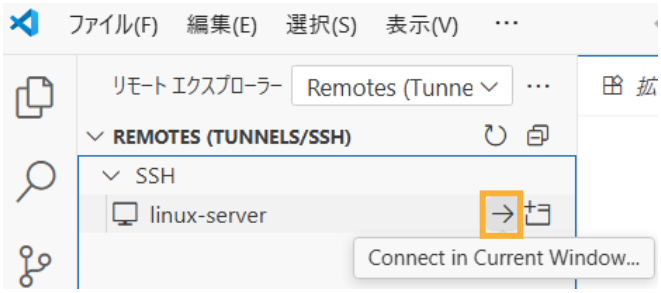
画面上部中央に以下が表示されたら、一番上の項目を選択します。



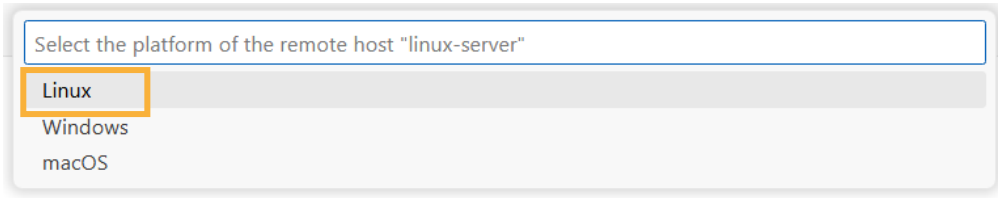
上記入力は、以下を指定しています。お使いの環境にあわせて変更してください。

tarot の場所は、現在 Windows でログイン中のユーザー名になります。

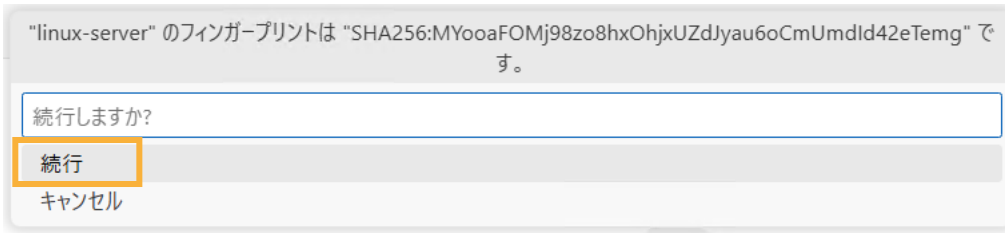
7) 画面右側に追加された [linux-server] 欄にカーソルを合わせて表示される [→] アイコンをクリックします。



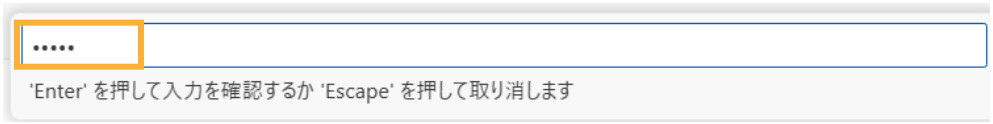
[Linux] を選択します。



[続行] を選択します。



パスワードを入力して、Enter を押します。



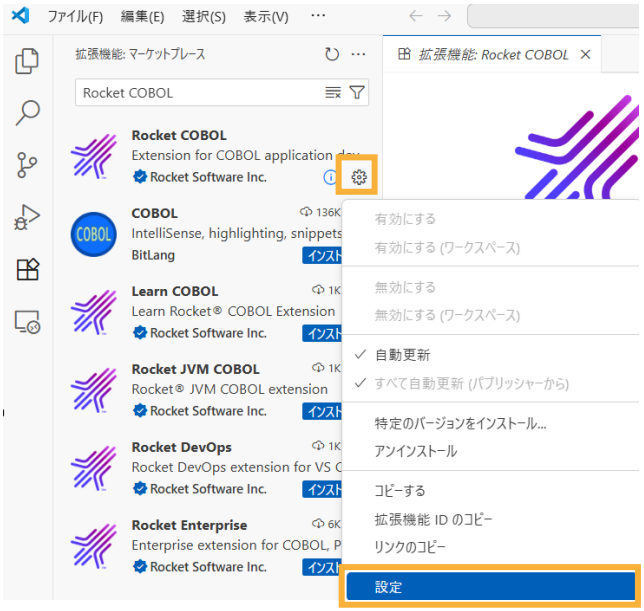
8) 画面左より拡張機能のアイコンをクリックします。

9) 検索欄に “Rocket COBOL” を入力して、表示される [インストール] をクリックします。

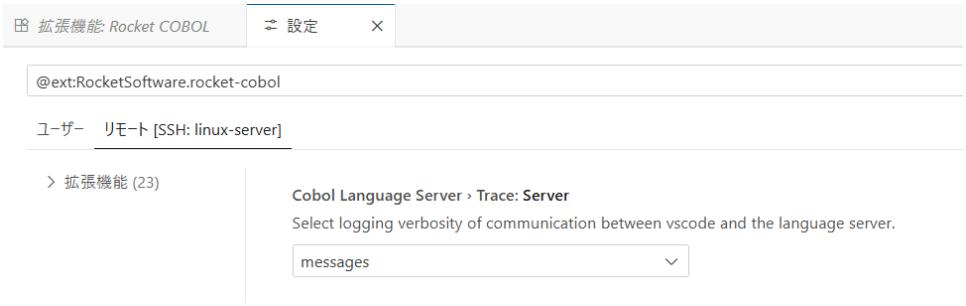
すでに、Windows 環境側にインストール済みの場合は、[SSH: linux-server にインストール] をクリックします。

以下のダイアログが表示された場合は、[発行元を信頼してインストールする] をクリックします。

インストールが完了すると、歯車アイコンが表示されます。歯車アイコンをクリックして表示されるメニューより [設定] を選択します。



画面中央より [リモート [SSH: linux-server]] タブをクリックしたうえで、以下の入力を行います。



- Rocket COBOL > Ant: Ant_home Path
  “/opt/rocketsoftware/VisualCOBOL/remotedev/ant/apache-ant-1.10.12” を入力
- Rocket COBOL > Ant: Ant_home Path Resolve
  [User Defined] を選択
- Rocket COBOL > Ant: Java_home Path
  “/usr/lib/jvm/jdk-11.0.23+9” を入力
- Rocket COBOL > Ant: Java_home Path Resolve
  [User Defined] を選択
- Rocket COBOL > Install Location
  “/opt/rocketsoftware/VisualCOBOL” を入力

注意）
各種項目のパスは、お使いの環境に合わせて設定してください。
また、Windows 環境にプラグインをインストール済みの場合、設定するタブに注意してください。本チュートリアルでは、[リモート [SSH: linux-server]] タブを選択してください。

免責事項

ここで紹介したソースコードは、機能説明のためのサンプルであり、製品の一部ではございません。ソースコードが実際に動作するか、御社業務に適合するかなどに関しまして、一切の保証はございません。 ソースコード、説明、その他すべてについて、無謬性は保障されません。
ここで紹介するソースコードの一部、もしくは全部について、弊社に断りなく、御社の内部に組み込み、そのままご利用頂いても構いません。
本ソースコードの一部もしくは全部を二次的著作物に対して引用する場合、著作権法の精神に基づき、適切な扱いを行ってください。