**アプリケーション開発標準**

開発ツール標準

目次

[0. 改訂履歴 2](#_Toc168927142)

[1. はじめに 3](#_Toc168927143)

[1.1. 目的 3](#_Toc168927144)

[1.2. 対象者 3](#_Toc168927145)

[1.3. 略語集 4](#_Toc168927146)

[2. 開発ツール標準 4](#_Toc168927147)

[2.1. 構築手順 5](#_Toc168927148)

[2.1.1. 事前準備 5](#_Toc168927149)

[2.1.2. コマンドプロンプト 5](#_Toc168927150)

[2.1.3. Scoop 8](#_Toc168927151)

[2.1.4. Git 10](#_Toc168927152)

[2.1.5. サクラエディタ 10](#_Toc168927153)

[2.1.6. WinMerge日本語版 13](#_Toc168927154)

[2.1.7. PowerShell 15](#_Toc168927155)

[2.1.7.1. オプション： 新しいPowerShell 17](#_Toc168927156)

[2.1.7.1. オプション： Windowsターミナル 19](#_Toc168927157)

[2.1.8. Node.js 21](#_Toc168927158)

[2.1.9. Yarn 23](#_Toc168927159)

[2.1.10. Zulu OpenJDK 23](#_Toc168927160)

[2.1.11. Salesforce CLI 24](#_Toc168927161)

[2.1.12. Visual Studio Code 25](#_Toc168927162)

[2.1.13. Visual Studio Code拡張機能 26](#_Toc168927163)

[2.2. Wrap-up 31](#_Toc168927164)

[2.2.1. インストール確認 31](#_Toc168927165)

[2.2.2. Git情報確認 32](#_Toc168927166)

[2.2.3. Gitトラブルシュート 33](#_Toc168927167)

[2.3. 利用ルール 36](#_Toc168927168)

[2.3.1. Salesforce開発 36](#_Toc168927169)

[2.3.2. その他 36](#_Toc168927170)

# 改訂履歴

|  |  |  |  |
| --- | --- | --- | --- |
| 更新日 | バージョン | 内容 | 更新者 |
| 2019/05/01 | 1.0 | * 初版作成 | takahito.miyamoto |
| 2020/04/08 | 2.0 | * 第2版作成 | takahito.miyamoto |
| 2021/05/26 | 2.1 | * インストール方法を更新   + 2.1.8. Zulu OpenJDK   + 2.1.9. Node.js   + 2.1.10. Yarn * インストール対象を更新   + 2.1.13. Visual Studio Code拡張機能 | shuri.wakita  aya.kubota |
| 2021/08/19 | 2.2 | * PowerShellのプロンプト設定を更新 * VS Codeのsettings.jsonを更新 | takahito.miyamoto |
| 2021/10/11 | 2.3 | * sfdx-plugin-prettierを追加 * VS Codeのsettings.jsonを更新 * VS Codeの拡張機能を更新 * Yarnのインストール方法を更新 | takahito.miyamoto |
| 2022/01/12 | 2.4 | * Git Bashを削除 * Zuluのインストール手順を更新 * Node.jsのインストール手順を更新 | takahito.miyamoto |
| 2022/06/08 | 2.5 | * サクラエディタのインストール手順を更新 * ツールのインストール順序を変更 * WinMergeのコンテキストメニューを追加 * Zuluのインストール手順を更新 * Wrap-upを追加 | takahito.miyamoto |
| 2023/4/10 | 2.6 | * 拡張機能にDocument Thisを追加 | miki.jin |
| 2023/4/11 | 2.7 | * 2.1.7 PowerShellのコマンド修正 * 拡張機能からLive HTML Previewerを削除 | miki.jin |
| 2023/5/23 | 2.8 | * 2.1.10と2.1.12の２行になっているコマンドのフォントを小さくして１行に修正。 * 2.2.1のキャプチャ張り替え | miki.jin |
| 2023/5/29 | 2.9 | * 目次の更新 * 2.1.1にある「echo y | sfdx plugins:install sfpowerkit」の文章を削除・キャプチャ張り替え | miki.jin |
| 2023/5/30 | 3.0 | * 目次の更新 | miki.jin |
| 2023/5/30 | 3.1 | * 2.1.8のversionと画像張替え * 2.2の画像張り替え | miki.jin |
| 2023/5/30 | 3.2 | * ヘッダーの更新 | miki.jin |
| 2023/12/05 | 3.3 | * ソースコード管理にGitLabを追加 * 静的コード解析をSalesforce Code Analyzerに変更 * Gitのcredential helperを資格情報マネージャーに変更 * nvmインストール時にPowerShellの管理者モードを有効化するタイミングを変更 * Salesforce CLIのインストールコマンドを変更 * Visual Studio Code拡張機能を変更 | takahito.miyamoto |
| 2023/12/12 | 3.4 | * レジストリ登録のコマンドを修正 | takahito.miyamoto |
| 2024/03/25 | 3.5 | * NVMのインストール手順を修正 | takahito.miyamoto |
| 2024/06/10 | 3.6 | * Copyrightを2024へ修正 * Node.jsのLTSバージョンを20.14.0へ修正 * Javaバージョンを11から17へ変更 | takahito.miyamoto |

* 年1回程度、最新のテクノロジー動向を考慮して本文書を改訂する。
* 不明点は更新者にコンタクトすること。

# はじめに

## 目的

本文書により、Salesforceを利用したアプリケーション開発において対象者が利用すべき開発ツール群を定義する。最終更新日時点でベストと思われるツールの組み合わせを記載しているが、プロジェクトによっては既に導入済みあるいは導入予定の別ツールが存在する可能性があるため、個々のツールの採用可否はプロジェクトごとに判断することとする。

## 対象者

本文書の意図する対象者は、Windows端末にてSalesforceを利用したアプリケーション開発に関わる以下のメンバーとする：

* アーキテクチャ担当者
* コーディング担当者
* テスト担当者
* デプロイ担当者

## 略語集

特に断りのない限り、本文書を通して以下の略語を何の説明もなく用いることがある。

| 略語 | 意味 |
| --- | --- |
| CI | Continuous Integration |
| CLI | Command Line Interface |
| CSS | Cascading Style Sheets |
| MS Teams | Microsoft Teams |
| npm | Node Package Manager |
| nvm | Node.js Version Manager for Windows |
| SFDC | Salesforce.com |
| VS Code | Visual Studio Code |

# 開発ツール標準

原則として、以下のツールをインストールおよびセットアップして利用する。

| テーマ | ツール |
| --- | --- |
| ソースコード管理 - ローカル | Git |
| ソースコード管理 - リモート | GitHubまたはBitbucketまたはGitLab |
| コードレビュー | GitHubまたはBitbucketまたはGitLab |
| Gitクライアント | VS Code |
| テキストエディタ | VS Codeおよびサクラエディタ**(\*)** |
| コンソール | コマンドプロンプトおよびPowershell |
| 静的解析 | Salesforce Code Analyzer |
| コード整形 | Prettier |
| コミュニケーション | MS Teams |
| 自動化スクリプト | Node.jsおよびYarn |

**(\*)** サクラエディタ以外にお気に入りのテキストエディタがあればそれを利用してもよい。Windows標準の「メモ帳」は不可とする。

## 構築手順

### 事前準備

各種ツールのインストーラーによっては、インストール先のフォルダをデフォルトで設定しているものが多く存在する。しかし、それぞれのインストール先がバラバラな状態を放置していると、あとから目的の場所を見つけられなくなり、バージョンアップやアンインストール作業が滞ることがある。そのような事態を避けるため、Cドライブの下にツール専用のフォルダを新規作成し、原則としてその配下に各種ツールをインストールする。



また、ダウンロードした各インストーラーは特定の場所に移動させてからインストール作業を実施する。

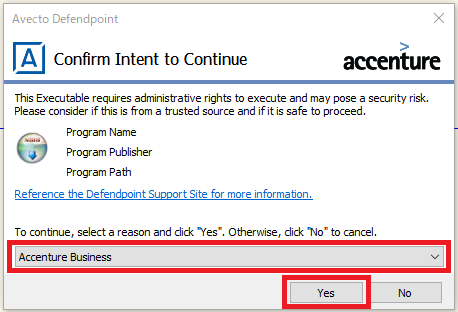


これ以降は、

* インストール場所： C:\tools
* インストーラーの保存場所： C:\tools\\_installer

を前提とした手順を記載する。

また、作業の途中で以下の画面がポップアップすることがある。その場合は、「Accenture Business」を選択して「Yes」をクリックすればよい。

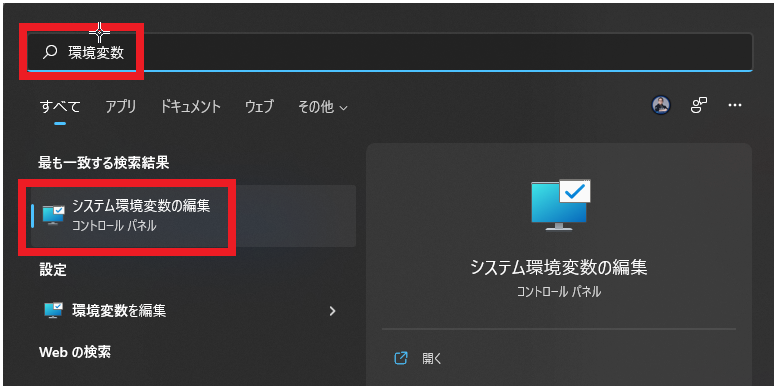


### コマンドプロンプト

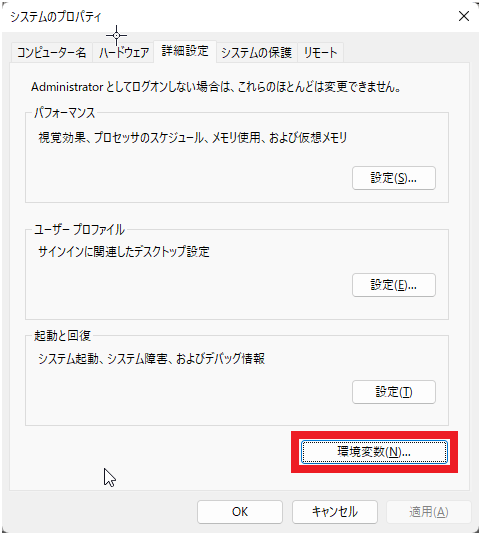
ディレクトリの階層が深くなると、コマンドプロンプトのプロンプト表示が長くなる。コマンド部分が改行されて一番左に来るように変更する。

※好みに応じて表示形式は変えてよい。

スタートメニューから「システム環境変数の編集」を検索して実行する。

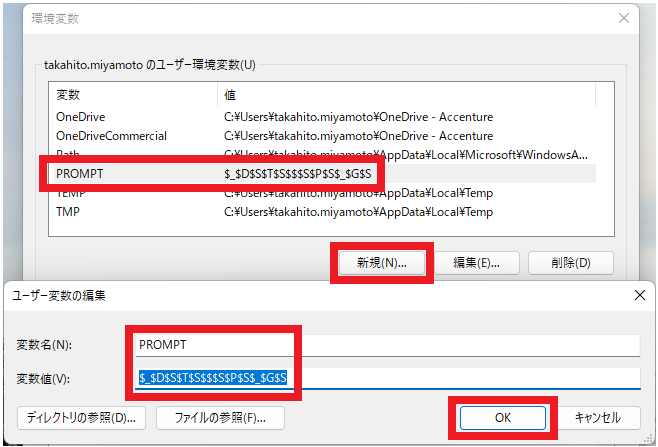


「環境変数」をクリックする。



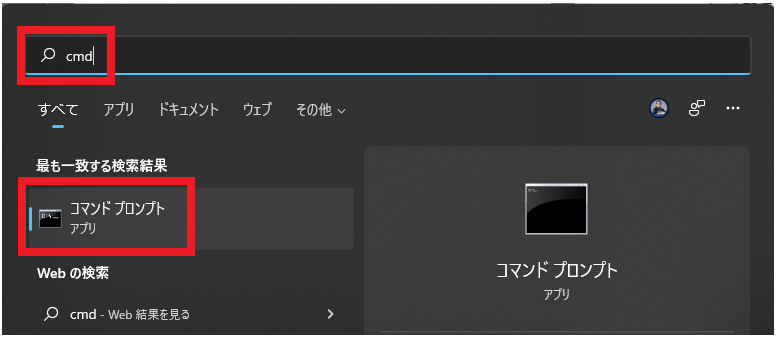
ユーザーの環境変数に「PROMPT」を新規追加する。

* 変数名: PROMPT
* 変数値: $\_$D$S$T$S$$$S$P$S$\_$G$S



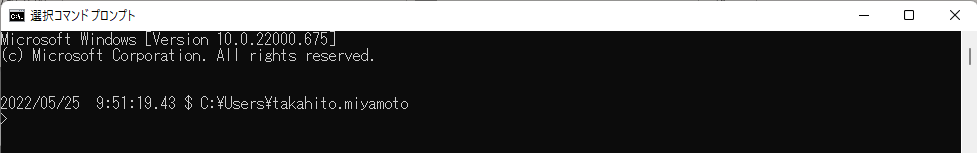
「OK」 > 「OK」 > 「OK」 をクリックして編集を完了する。

スタートメニューから「cmd」を検索して実行する。



表示内容に日付が含まれていて、改行された行からカーソルが開始していることを確認する。

※ 確認後、コマンドプロンプトは閉じる。



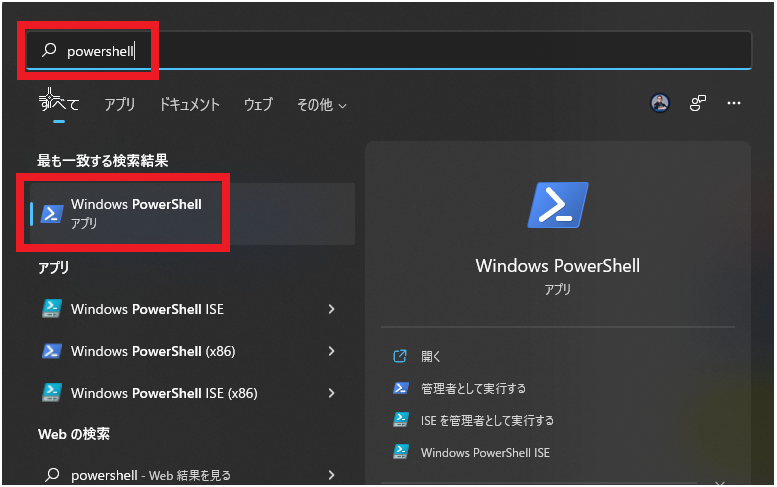
### Scoop

Scoopを活用することでコマンドラインから各種ツールをインストールできる。

以下のツールのインストールのために、Scoopを導入する。

* Git
* サクラエディタ
* Java
* Node.js、Yarn
* Salesforce CLI
* Visual Studio Code
* WinMerge日本語版

スタートメニューからPowerShellを検索して実行する。

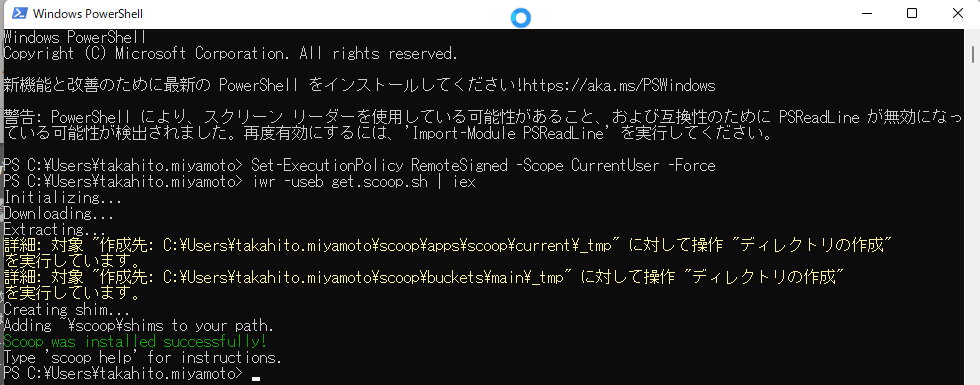


PowerShellで次のコマンドを実行する。

※ コマンドは1行ずつ実行する。

|  |
| --- |
| Set-ExecutionPolicy RemoteSigned -Scope CurrentUser -Force  iwr -useb get.scoop.sh | iex |

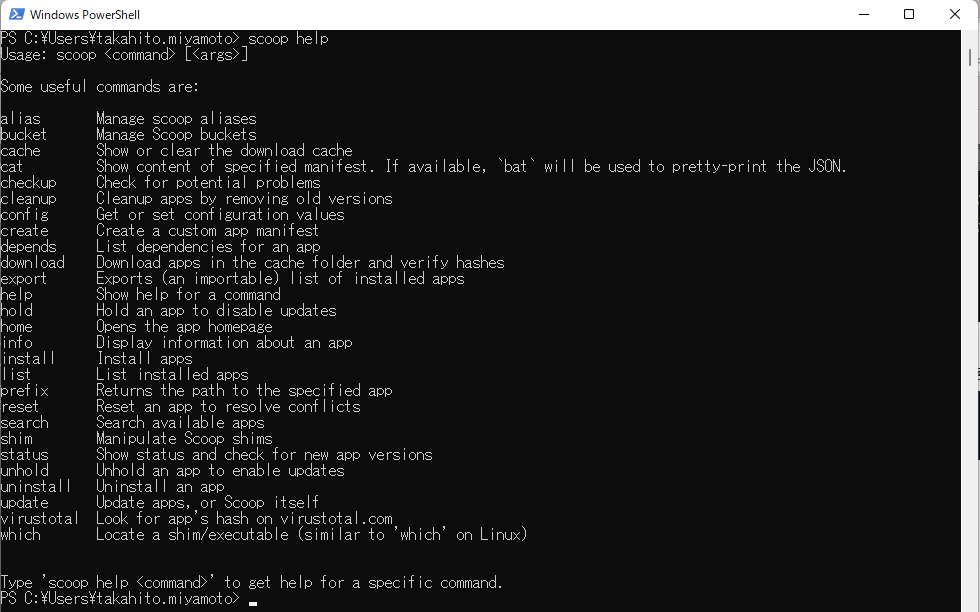
「Scoop was installed successfully!」が表示されていることを確認する。



PowerShellで次のコマンドを実行して、Scoopのヘルプ情報が確認できたらインストール成功。

※ PowerShellは起動したままとする。

|  |
| --- |
| scoop help |



### Git

PowerShellで次のコマンドを実行する。

※ コマンドは1行ずつ実行する。

|  |
| --- |
| scoop install git |

PowerShellで次のコマンドを実行する。

※ コマンドは1行ずつ実行する。

※ ***EnterpriseID*** と ***Email*** は適宜修正してから実行する。

|  |
| --- |
| git config --global user.name ***EnterpriseID***  git config --global user.email ***Email***  git config --global core.quotepath false  git config --global merge.tool vscode  git config --global diff.tool vscode  git config --global core.autocrlf input  git config --global core.editor sakura  git config --global credential.helper wincred |

### サクラエディタ

※サクラエディタ以外のテキストエディタを利用したい場合は、各自手順を調べてインストールすること。

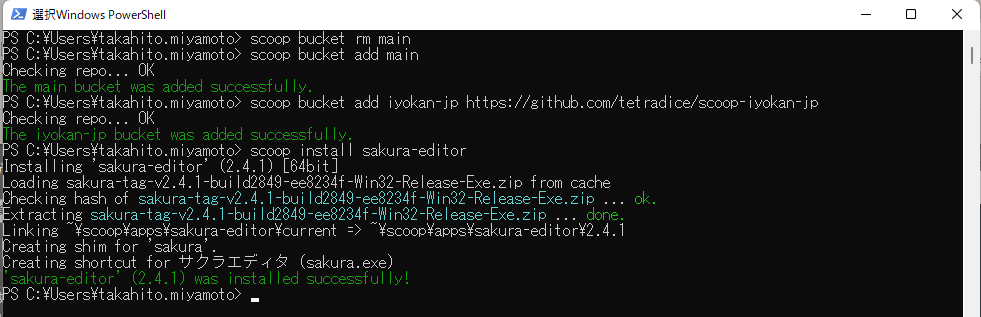
PowerShellで次のコマンドを実行する。

※ コマンドは1行ずつ実行する。

|  |
| --- |
| scoop bucket rm main  scoop bucket add main  scoop bucket add iyokan-jp https://github.com/tetradice/scoop-iyokan-jp  scoop install sakura-editor |

「'sakura-editor' (2.4.1) was installed successfully!」が表示されていることを確認する。

※ バージョンは異なっていてもOK。



PowerShellで次のコマンドを実行する。

※ コマンドは1行ずつ実行する。

|  |
| --- |
| sakura $HOME\openWithSakura.reg |

サクラエディタが開くので次のコードを貼り付けて保存して閉じる。

※ ***EnterpriseID***は適宜修正してから保存する（**6か所**）。

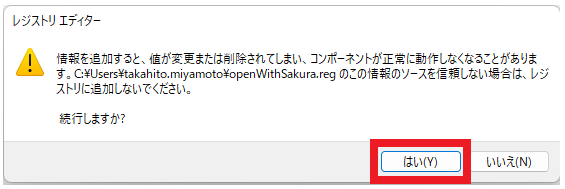
|  |
| --- |
| Windows Registry Editor Version 5.00  [HKEY\_CURRENT\_USER\Software\Classes\\*\shell\OpenWithSAKURA]  @="Open with SAKURA"  "Icon"="C:\\Users\\***EnterpriseID***\\scoop\\apps\\sakura-editor\\current\\sakura.exe"  [HKEY\_CURRENT\_USER\Software\Classes\\*\shell\OpenWithSAKURA\command]  @="\"C:\\Users\\***EnterpriseID***\\scoop\\apps\\sakura-editor\\current\\sakura.exe\" \"%1\""  [HKEY\_CURRENT\_USER\Software\Classes\Directory\shell\GrepWithSAKURA]  @="Grep with SAKURA"  "Icon"="C:\\Users\\***EnterpriseID***\\scoop\\apps\\sakura-editor\\current\\sakura.exe"  [HKEY\_CURRENT\_USER\Software\Classes\Directory\shell\GrepWithSAKURA\command]  @="\"C:\\Users\\***EnterpriseID***\\scoop\\apps\\sakura-editor\\current\\sakura.exe\" -GREPMODE -GREPDLG -GOPT=SLP -GFILE=\"\*.\*\" -GFOLDER=\"%1\""  [HKEY\_CURRENT\_USER\Software\Classes\Directory\Background\shell\GrepWithSAKURA]  @="Grep with SAKURA"  "Icon"="C:\\Users\\***EnterpriseID***\\scoop\\apps\\sakura-editor\\current\\sakura.exe"  [HKEY\_CURRENT\_USER\Software\Classes\Directory\Background\shell\GrepWithSAKURA\command]  @="\"C:\\Users\\***EnterpriseID***\\scoop\\apps\\sakura-editor\\current\\sakura.exe\" -GREPMODE -GREPDLG -GOPT=SLP -GFILE=\"\*.\*\" -GFOLDER=\"%1\"" |
|  |

PowerShellで次のコマンドを実行する。

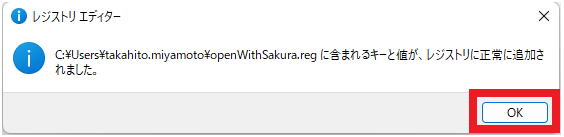
※ コマンドは1行ずつ実行する。

|  |
| --- |
| reg import $HOME\openWithSakura.reg |

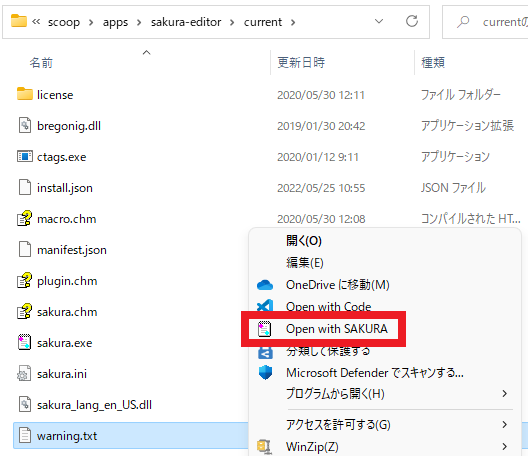
確認メッセージがポップアップされた場合、[はい]をクリックする。



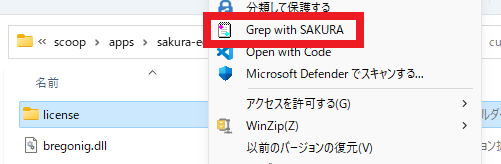
[OK]をクリックする。



任意のテキストファイルを右クリックして、「SAKURAで開く」メニューが追加されたことを確認する。



任意のフォルダを右クリックして、「Grep with SAKURA」メニューが追加されたことを確認する。



### WinMerge日本語版

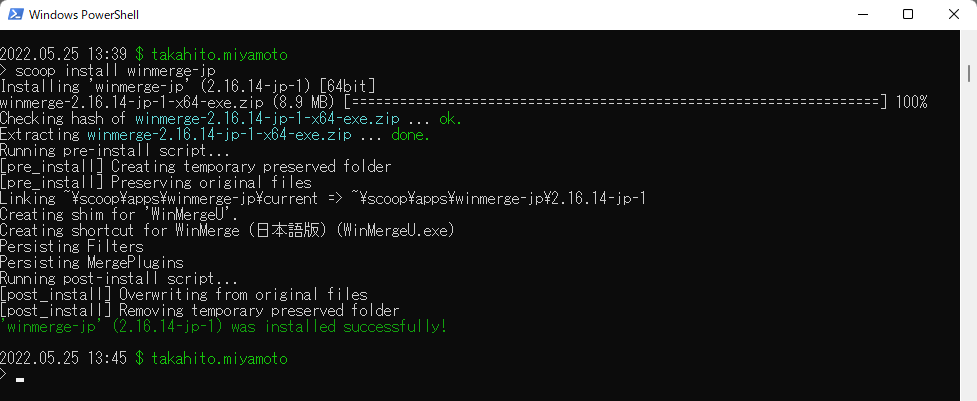
PowerShellで次のコマンドを実行する。

※ コマンドは1行ずつ実行する。

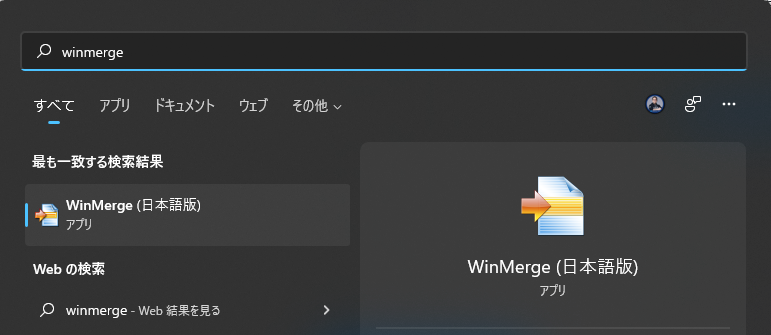
|  |
| --- |
| scoop install winmerge-jp |

「'winmerge-jp' (2.16.14-jp-1) was installed successfully!」が表示されていることを確認する。

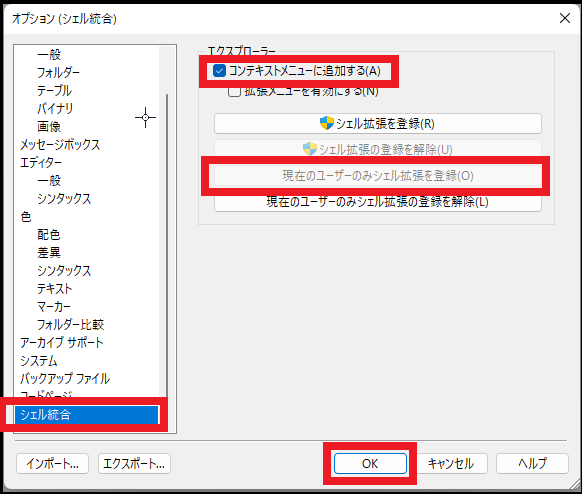
※ バージョンは異なっていてもOK。



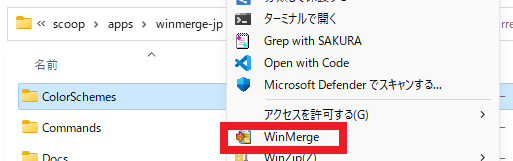
スタートメニューからWinMergeを検索して実行する。



[編集]>[設定]>[シェル統合]を選択する。[現在のユーザーのみシェル拡張を登録]をクリックし、[コンテキストメニューに追加する]をONにして、[OK]をクリックする。



任意のフォルダを右クリックして、「WinMerge」メニューが追加されたことを確認する。



### PowerShell

ディレクトリの階層が深くなると、PowerShellのプロンプト表示が長くなる。コマンド部分が改行されて一番左に来るように変更する。

※好みに応じて表示形式は変えてよい。

PowerShellで次のコマンドを実行する。

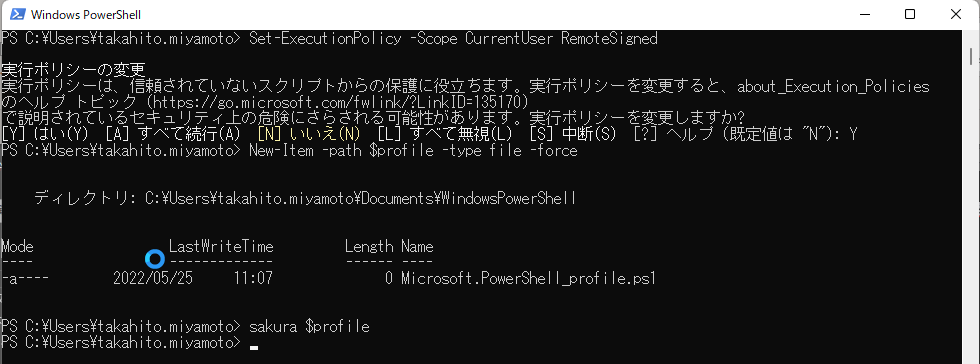
※ コマンドは1行ずつ実行する。

|  |
| --- |
| Set-ExecutionPolicy -Scope CurrentUser RemoteSigned  New-Item -path $profile -type file -force  sakura $profile |

サクラエディタが開くので次のコードを貼り付けて保存して閉じる。

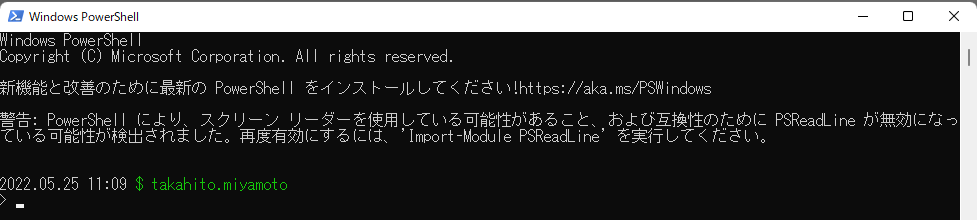
※好みに応じて表示形式は変えてよい。

|  |
| --- |
| Function Write-BranchName () {    try {  *$branch* ***=*** git rev***-***parse ***--***abbrev***-***ref HEAD      if (*$branch* ***-eq*** "HEAD") {  *# we're probably in detached HEAD state*        (git branch | select-string "^\\*").ToString() | set-variable ***-***name branch  *$branch* ***=*** *$branch* ***-replace*** "^\\* \(", ""  *$branch* ***=*** *$branch* ***-replace*** "\)$", ""        Write-Host " (*$branch*)" ***-***ForegroundColor "red"      }      else {  *# we're on an actual branch*        Write-Host " (*$branch*)" ***-***ForegroundColor "blue"      }    } catch {  *# we'll end up here if we're in a newly initiated git repo*      Write-Host " (no branches yet)" ***-***ForegroundColor "yellow"    }  }  function prompt {  *$base* ***=*** (Get-Date ***-***Format "yyyy.MM.dd HH:mm")  *$path* ***=*** (Split-Path (Get-Location) ***-***Leaf)  *$userPrompt* ***=*** "*$*('>' ***\**** ($nestedPromptLevel ***+*** *1*)) "    Write-Host "*`n$base* " ***-***NoNewline    if (Test-Path .git) {      Write-Host $ *$path* ***-***NoNewline ***-***ForegroundColor "green"      Write-BranchName    }    else {  *# we're not in a repo so don't bother displaying branch name/sha*      Write-Host $ *$path* ***-***ForegroundColor "green"    }    return *$userPrompt*  } |



一度PowerShellを閉じて、再度起動する。プロンプトが変更されたことを確認する。

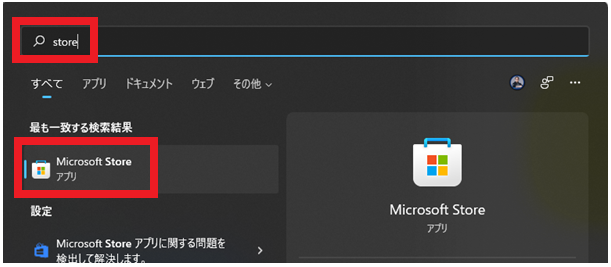
※ 確認したら閉じる。



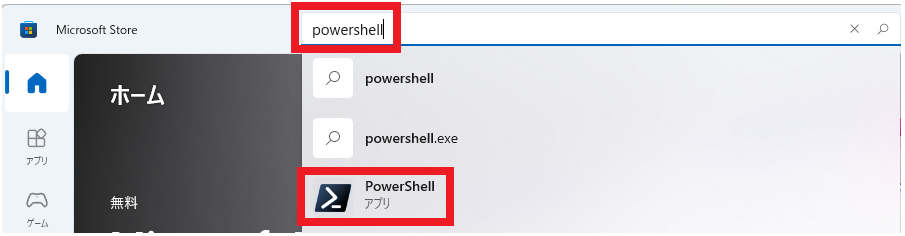
#### オプション： 新しいPowerShell

新しいPowerShellをインストールする。

スタートメニューから「Microsoft Store」を検索して実行する。



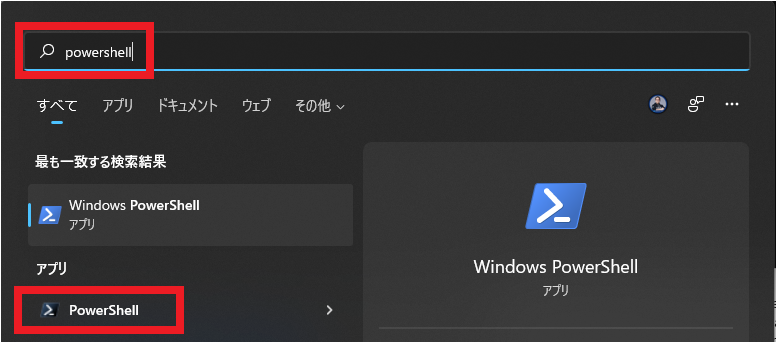
「PowerShell」を検索してクリックする。



「入手」をクリックする。



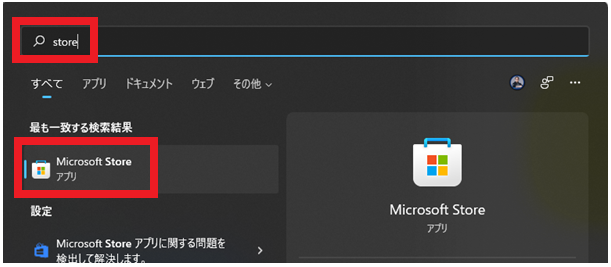
スタートメニューから「PowerShell」を検索して、新しいPowerShellが表示されることを確認する。



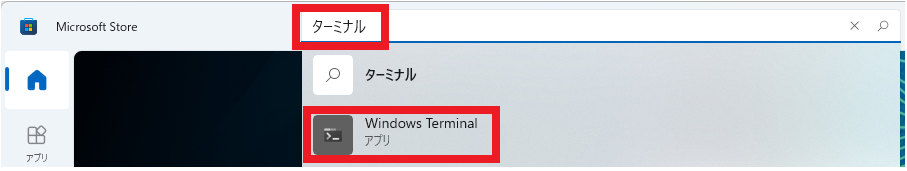
#### オプション： Windowsターミナル

コマンドプロンプトとPowerShellを自在に切り替えるために、Windowsターミナルをインストールする。

スタートメニューから「Microsoft Store」を検索して実行する。



「ターミナル」を検索してクリックする。



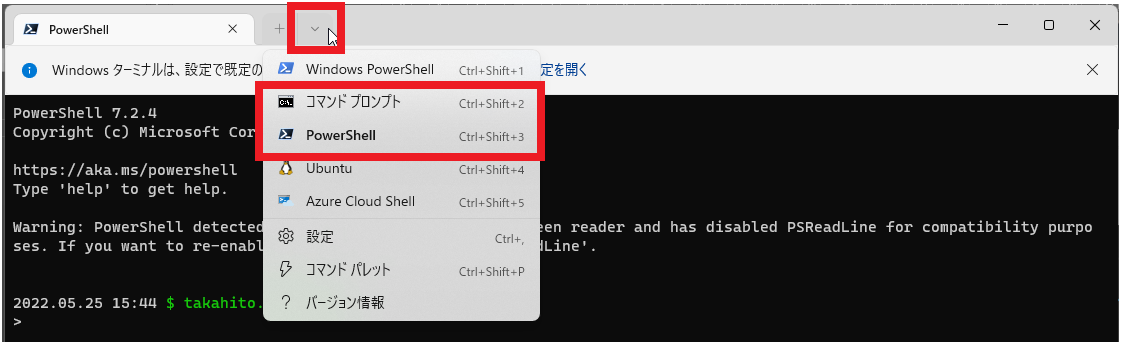
「入手」をクリックする。



スタートメニューから「ターミナル」を検索して実行する。



タブからコマンドプロンプトやPowerShellが選択できることを確認する。



### Node.js

nvm 経由でインストールする手順を採用する。

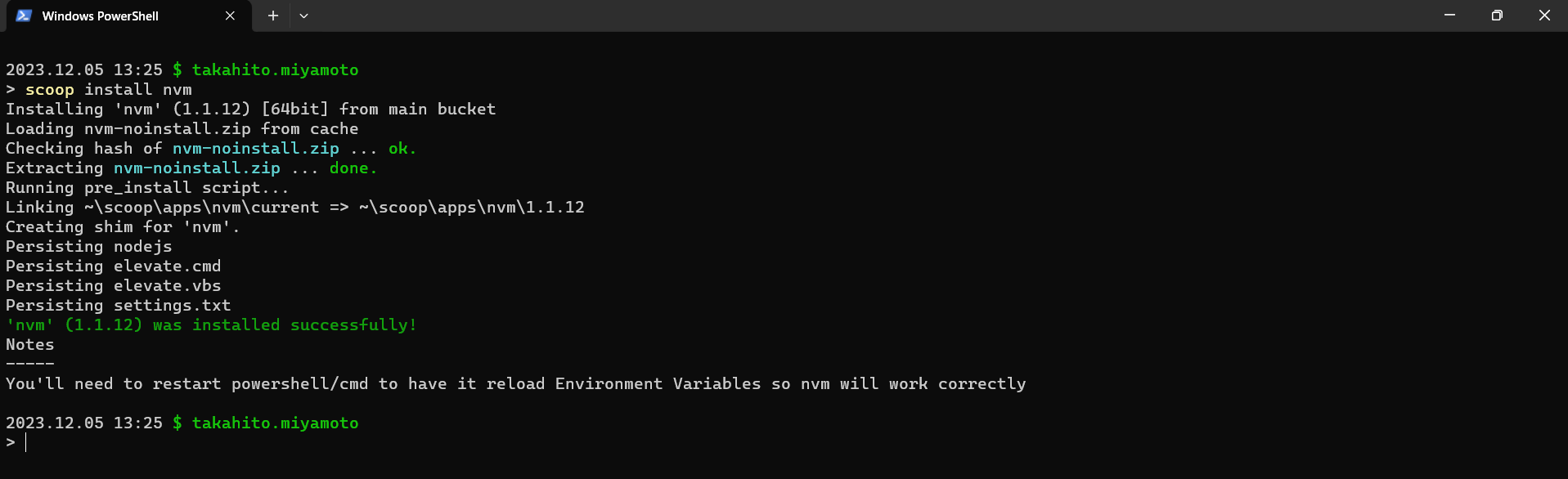
PowerShellで次のコマンドを実行する。

※ コマンドは1行ずつ実行する。

|  |
| --- |
| scoop bucket rm main  scoop bucket add main  scoop install nvm |

「'nvm' (1.1.12) was installed successfully!」が表示されていることを確認する。

※ バージョンは異なっていてもOK。



PowerShellで次のコマンドを実行する。

※ コマンドは1行ずつ実行する。

|  |
| --- |
| nvm list available |

LTS のバージョンを確認する。この場合、20.14.0 がインストール対象の LTS バージョンとなる。

テキスト

自動的に生成された説明

PowerShellで次のコマンドを実行する。

※ コマンドは1行ずつ実行する。

|  |
| --- |
| nvm install 20.14.0 |

テキスト

低い精度で自動的に生成された説明

スタートボタンからcmdを検索し、管理者モードで実行する。



管理者モードのコマンドプロンプトで次のコマンドを実行する。

※ コマンドは1行ずつ実行する。

|  |
| --- |
| nvm use 20.14.0 |

「Now using node v20.14.0 (64-bit)」が表示されていることを確認する。

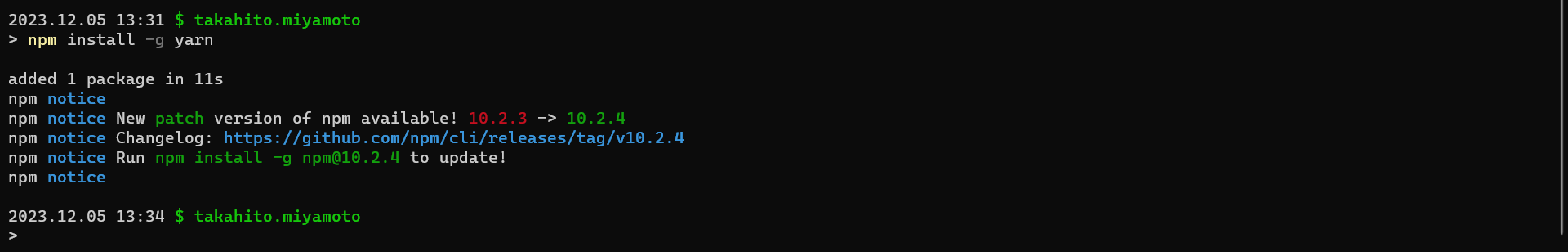


### Yarn

PowerShellで次のコマンドを実行する。

※ コマンドは1行ずつ実行する。

|  |
| --- |
| npm install -g yarn |



### Zulu OpenJDK

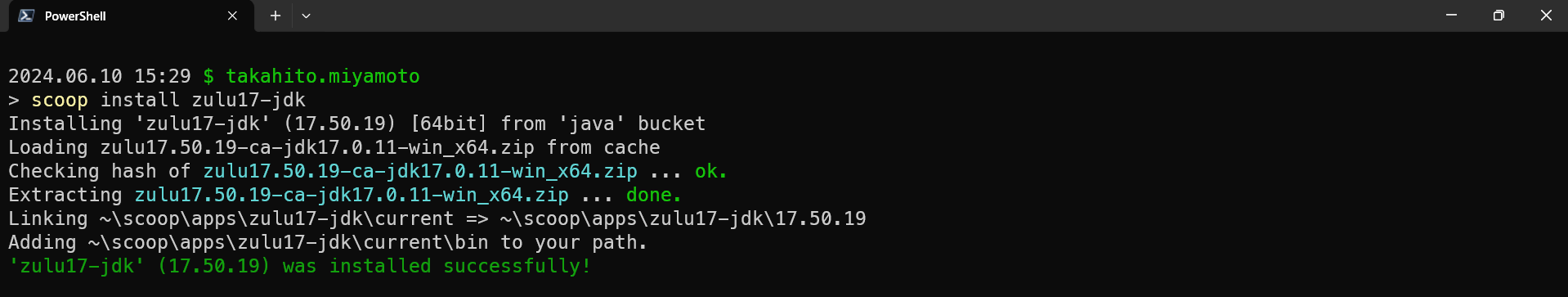
PowerShellで次のコマンドを実行する。

※ コマンドは1行ずつ実行する。

|  |
| --- |
| scoop bucket add java  git config --global --add safe.directory $HOME/scoop/buckets/java  scoop install zulu17-jdk |

「'zulu11-jdk' (17.50.19) was installed successfully!」が表示されていることを確認する。

※ バージョンは異なっていてもOK。



### Salesforce CLI

一度PowerShellを閉じて、再度起動する。

PowerShellで次のコマンドを実行する。

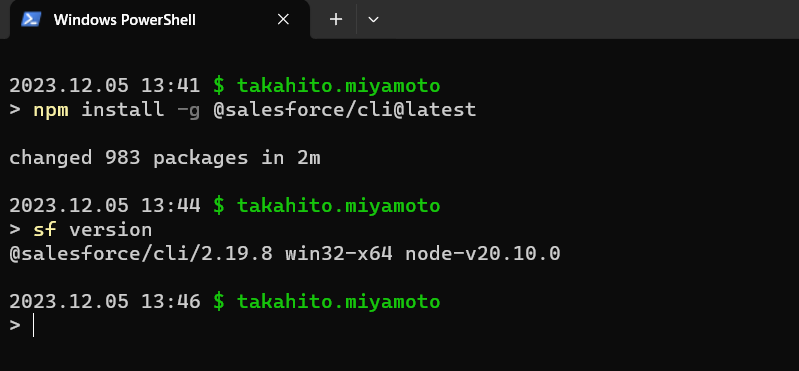
※ コマンドは1行ずつ実行する。

※ 結構時間かかるので終わるまで別作業をしているとよい。

|  |
| --- |
| npm install -g @salesforce/cli@latest  sf version |

sf のバージョンが表示されていることを確認する。

※ バージョンは異なっていてもOK。



PowerShellで次のコマンドを実行する。

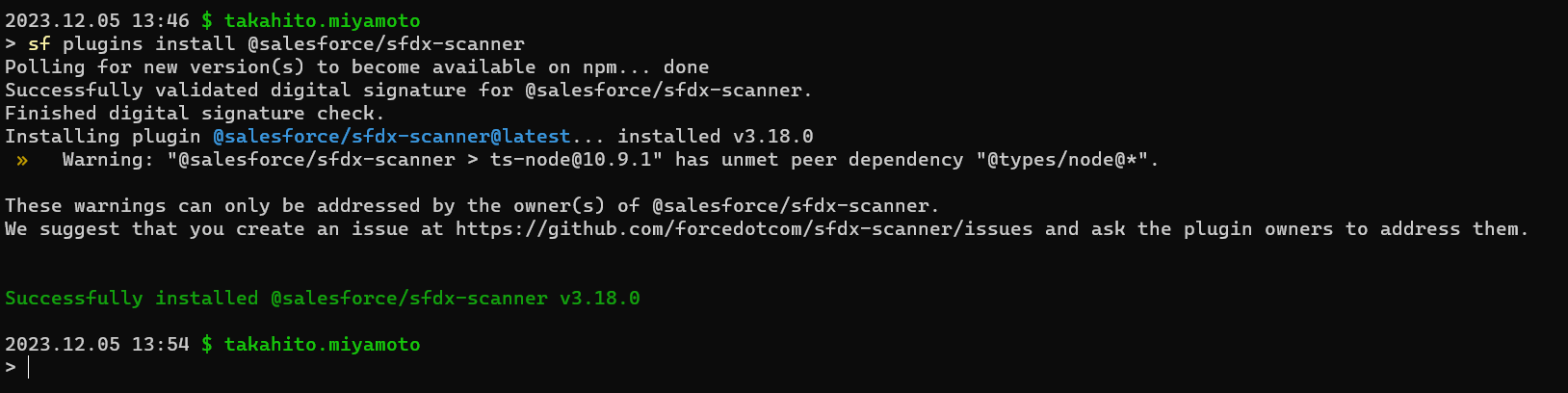
※ コマンドは1行ずつ実行する。

※ 結構時間かかるので終わるまで別作業をしているとよい。

|  |
| --- |
| sf plugins install @salesforce/sfdx-scanner |

「Successfully installed @salesforce/sfdx-scanner v3.18.0」が表示されていることを確認する。

※ バージョンは異なっていてもOK。



### Visual Studio Code

PowerShellで次のコマンドを実行する。

※ コマンドは1行ずつ実行する。

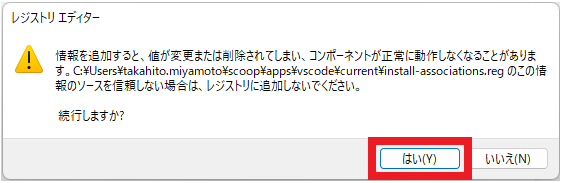
|  |
| --- |
| scoop bucket add extras  git config --global --add safe.directory $HOME/scoop/buckets/extras  scoop install vscode |

PowerShellで次のコマンドを実行する。

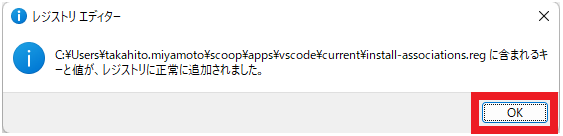
※ コマンドは1行ずつ実行する。

|  |
| --- |
| reg import $HOME\scoop\apps\vscode\current\install-context.reg  reg import $HOME\scoop\apps\vscode\current\install-associations.reg |

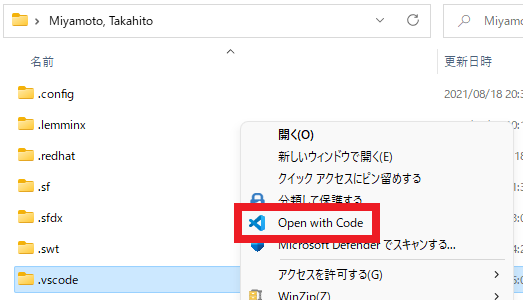
確認メッセージがポップアップされた場合、[はい]をクリックする。



[OK]をクリックする。

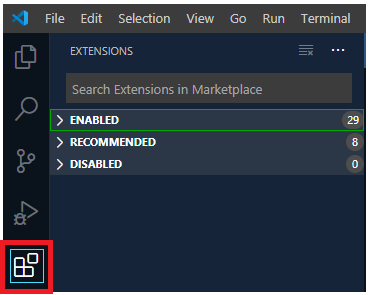


任意のフォルダを右クリックして、「Open with Code」メニューが追加されたことを確認する。



### Visual Studio Code拡張機能

VS Codeを起動し、拡張機能のアイコンをクリックする。



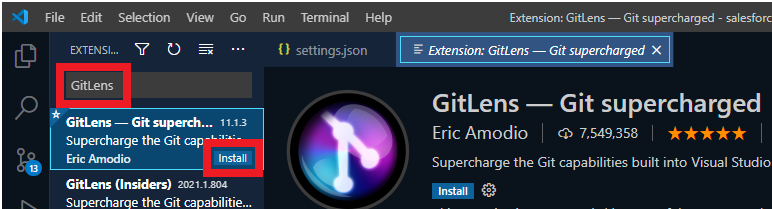
以下の拡張機能を検索して順次インストールする。

| 名称 | 用途 |
| --- | --- |
| 共通 | |
| EditorConfig for VS Code | .editorconfig で定義した内容を適用できるようにする。 |
| Git Graph | コミットログを確認する。ブランチの状況を概観する。 |
| GitLens | 様々な切り口で過去の変更内容を行ったり来たりすることを可能にする。 |
| JSON Tools | JSON をクイックに整形可能にする。 |
| Live Share | 相手を自分の VS Code 画面へ招待し、リアルタイムに同時編集可能にする。 |
| Rainbow CSV | CSV に色を付けて見やすくする。 |
| Todo Tree | TODO コメントを一覧で管理できるようにする。 |
| Trailing Spaces | 文末の余分なスペースや余分な改行を保存時に自動的に削除可能する。 |
| zenkaku | 半角スペース以外の空白文字（全角スペース、タブ）を検知可能にする。 |
| Salesforce開発 | |
| ESLint | JavaScriptに対する静的コード解析を実行する。 |
| Prettier - Code formatter | ソースコードを自動でフォーマットする。 |
| Salesforce Extension Pack | VS Codeを用いて、Salesforce Platformでカスタム開発を行うことを可能にする。特に、以下の拡張機能も同時にインストールされる。   * Apex * Apex Interactive Debugger * Apex Replay Debugger * Salesforce CLI Integration * Aura Components * Visualforce * Lightning Web Components * SOQL * SLDS Validator |
| Einstein for Developers | Einstein GPTによりApexコードを自動で生成する。 |
| Salesforce Code Analyzer | ApexおよびJavaScriptに対する静的コード解析を実行する。 |
| Codey’s in bed by 10 | (Option) Salesforce社推奨のVS Codeテーマを追加する。 |
| Lightning Web Components開発 | |
| Path Autocomplete | インポートするファイルのパスをオートコンプリートで入力できるようにする。 |
| Document This | TypeScript および JavaScript ファイルに詳細な JSDoc コメントを自動的に生成する。 |

例）GitLens の場合

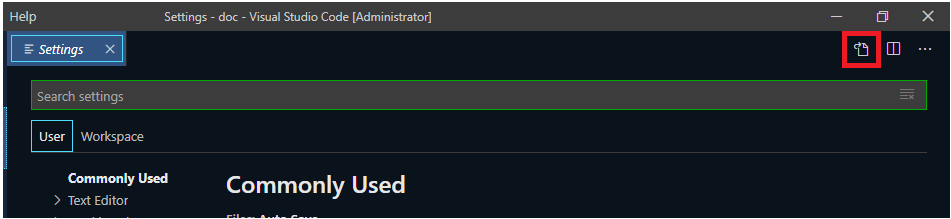
左上の検索ウィンドウへ「GitLens」と入力し、検索結果の中から「GitLens」を選択する。

「Install」をクリックして、完了するまで待機する。



上記の拡張機能をすべてインストールしたら、それらを有効化する設定をする。

「**Ctrl** + **,**」 を押下してSettings画面を開き、JSONのアイコンをクリックする。



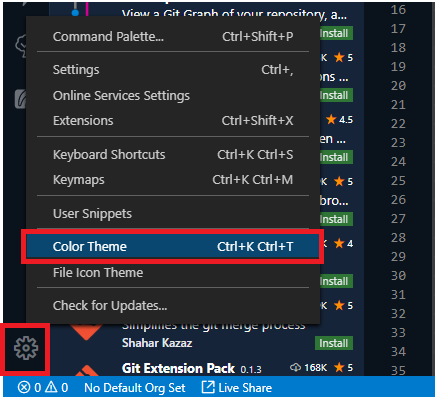
settings.jsonへ以下のコードを貼り付ける。

※「C:\\」で始まる箇所は各々の設定によって異なるため、実際に確認して間違っていたら正しく変更しておく。

※ ***EnterpriseID***は適宜修正してから実行する。(**2か所**)

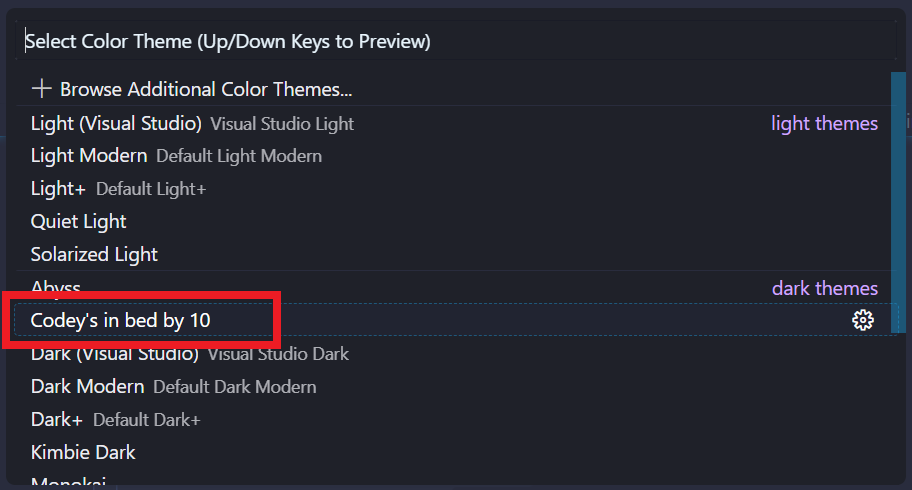
|  |
| --- |
| {    "[html]": { "editor.defaultFormatter": "esbenp.prettier-vscode" },    "[javascript]": { "editor.defaultFormatter": "esbenp.prettier-vscode" },    "[json]": { "editor.defaultFormatter": "esbenp.prettier-vscode" },    "[jsonc]": { "editor.defaultFormatter": "esbenp.prettier-vscode" },    "editor.defaultFormatter": "esbenp.prettier-vscode",    "editor.formatOnSave": *true*,    "editor.largeFileOptimizations": *false*,    "editor.minimap.autohide": *true*,    "editor.minimap.enabled": *false*,    "editor.mouseWheelZoom": *true*,    "editor.renderWhitespace": "all",    "editor.suggestSelection": "first",    "editor.tabSize": *2*,    "editor.wordWrap": "on",    "explorer.confirmDelete": *false*,    "extensions.ignoreRecommendations": *true*,    "files.associations": { "\*.cls": "apex" },    "files.eol": "*\n*",    "files.exclude": {      "/.git": *true*,      "\*\*/node\_modules": *true*,      "\*\*/node\_modules/": *true*    },    "files.trimTrailingWhitespace": *true*,    "files.watcherExclude": {      "\*\*/.git/objects/\*\*": *true*,      "\*\*/.git/subtree-cache/\*\*": *true*,      "\*\*/node\_modules/\*\*": *true*    },    "git.autofetch": *true*,    "git.confirmSync": *false*,    "git.path": "C:*\\*Users*\\*EnterpriseID*\\*scoop*\\*shims*\\*git.exe",    "javascript.updateImportsOnFileMove.enabled": "always",    "npm.autoDetect": "off",    "salesforcedx-vscode-apex.enable-semantic-errors": *false*,    "salesforcedx-vscode-apex.java.home": "C:*\\*Users*\\*EnterpriseID*\\*scoop*\\*apps*\\*zulu11-jdk*\\*current",    "salesforcedx-vscode-core.detectConflictsAtSync": *true*,    "salesforcedx-vscode-core.push-or-deploy-on-save.enabled": *false*,    "salesforcedx-vscode-core.retrieve-test-code-coverage": *true*,    "salesforcedx-vscode-core.show-cli-success-msg": *false*,    "salesforcedx-vscode-core.telemetry.enabled": *false*,    "salesforcedx-vscode-lightning.activationMode": "autodetect",    "search.exclude": { "\*\*/node\_modules": *true* },    "terminal.integrated.profiles.windows": {      "PowerShell -NoProfile": { "source": "PowerShell", "args": ["-NoProfile"] }    },    "todo-tree.filtering.excludedWorkspaces": ["node\_modules"],    "todo-tree.highlights.enabled": *true*,    "trailing-spaces.trimOnSave": *true*,    "typescript.tsc.autoDetect": "off",    "update.showReleaseNotes": *false*,    "window.zoomLevel": *1*,    "workbench.colorTheme": "Codey's in bed by 10",    "workbench.startupEditor": "none"  } |

VS Codeを再起動し、左下の歯車アイコン ＞ 「Color Theme」をクリックする。



「Codey's in bed by 10」をクリックし、見た目が変わったことを確認する。

※好みに応じて別のテーマを選んでもよい。



## Wrap-up

### インストール確認

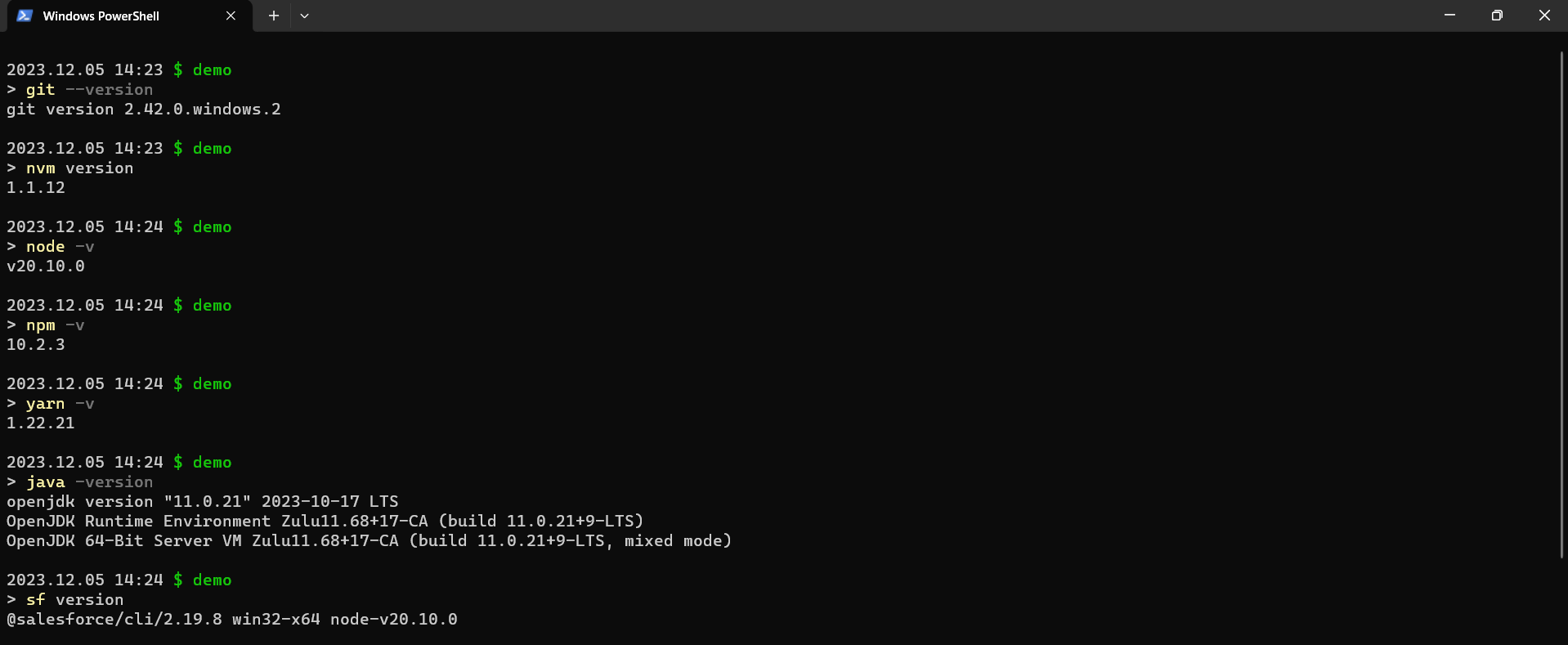
ここまでの手順で各ツールが正しくインストールできているか確認する。もしエラーが発生した場合は、手順に戻って再実施する。

PowerShellで次のコマンドを実行する。

※ コマンドは1行ずつ実行する。

※ バージョンは異なっていてもOK。

|  |
| --- |
| git --version  nvm version  node -v  npm -v  yarn -v  java -version  sf version  sf plugins --core |



テキスト

自動的に生成された説明

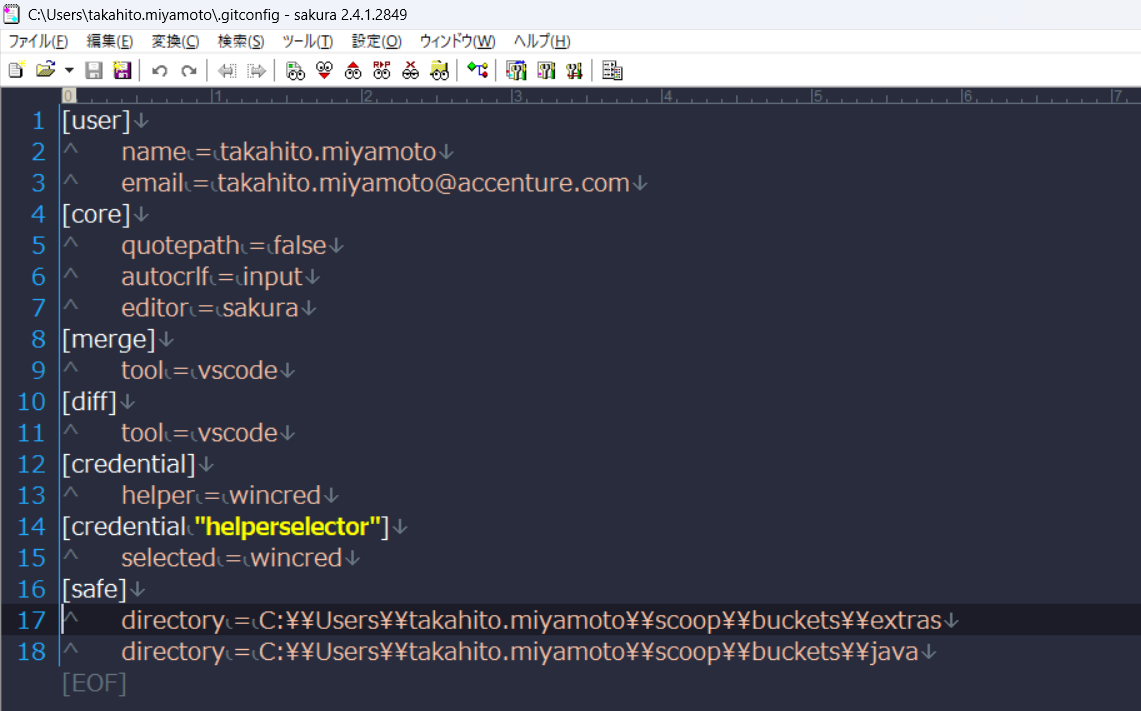
### Git情報確認

PowerShellで次のコマンドを実行する。

※ コマンドは1行ずつ実行する。

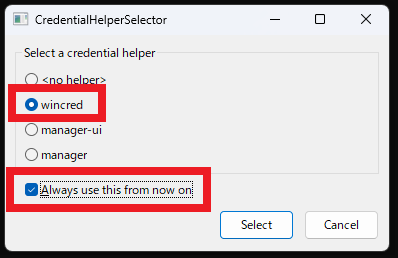
|  |
| --- |
| sakura .gitconfig |

サクラエディタが開き、Gitの設定を参照できることを確認する。確認したらファイルを閉じる。

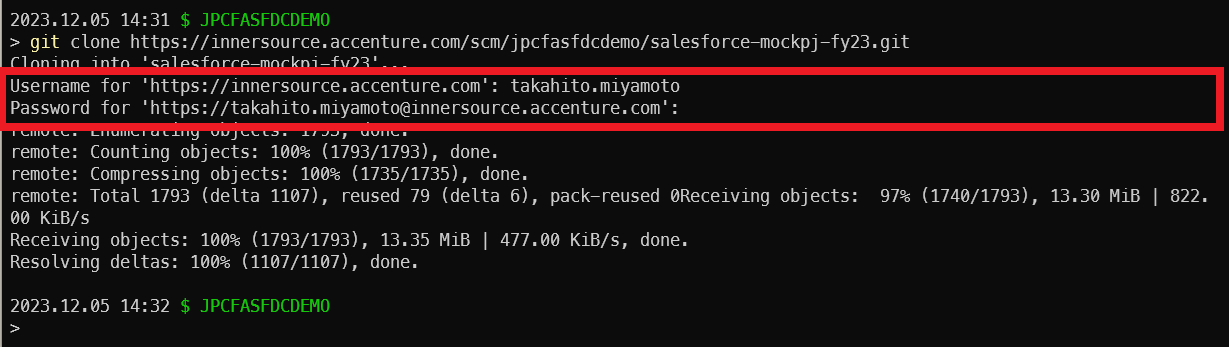


### Gitトラブルシュート

※Git利用時にポップアップ画面が表示されたら、wincredを選択する。



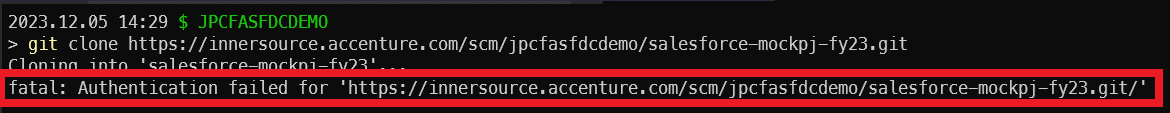
※Gitにログインを求められたら、IDにEnterprise ID、パスワードには**API Token**を入力して[OK]をクリックする。パスワードにAPI Tokenをコピペしても、何も表示されないので要注意。誤って複数回コピペしないこと。



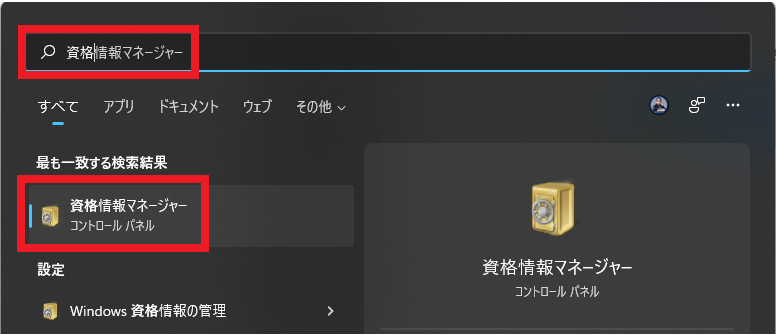
※API Tokenの取得先：

<https://innersource.accenture.com/plugins/servlet/de.resolution.apitokenauth/admin>

※Gitのパスワードを間違えて入力してしまった場合は認証が失敗するので、資格情報を削除してからやり直す。



スタートメニューから「資格情報マネージャー」を検索して実行する。



[Windows資格情報] から innersource 関連の情報を探し、[削除]をクリックする。



## 利用ルール

### Salesforce開発

* VS Codeは定期的にバージョンアップがあるため、プロジェクトが終わったタイミングを見計らって最新バージョンへ更新するようにする。プロジェクト中は更新しないようにする。

※前述の方法で上書きインストールすればよい。

* Node.jsのLTSは定期的にバージョンアップがあるため、プロジェクトが終わったタイミングを見計らって最新のLTSバージョンへ更新するようにする。プロジェクト中は更新しないようにする。

※前述の方法で最新の LTS バージョンをインストールすればよい。

* Salesforce CLIのコマンドを実行する場合、コマンドプロンプトあるいはPowerShellを利用する。Git BashはSalesforce CLIサポート対象外であるためコマンドを実行しないようにする。
* scoopで管理しているツールは定期的にバージョンアップがあるため、プロジェクトが終わったタイミングを見計らって最新バージョンへ更新するようにする。プロジェクト中は更新しないようにする。

※次のコマンドを実行する。

|  |
| --- |
| scoop update |

### その他

* その他、プロジェクトで取り決めたルールを順守する。

以上