の出力結果のように2項目にAndroid toolchainのエラー、5項目にAndroid Studioの警告が表示されます。

```
[✗] Android toolchain - develop for Android devices
    ✗ Unable to locate Android SDK.
      Install Android Studio from: https://developer.android.com/studio/index.html
      On first launch it will assist you in installing the Android SDK components.
      (or visit https://flutter.dev/docs/get-started/install/macos#android-setup
for detailed instructions).
      If the Android SDK has been installed to a custom location, please use
      `flutter config --android-sdk` to update to that location.
```

```
[!] Android Studio (not installed)
```

この場合は最初にAndroidアプリ開発用のIDE（*Integrated Development Environment*、統合開発環境）であるAndroid Studioをインストールするのが簡単です。Androidの開発者向けWebサイトからダウンロードし、セットアップウィザードの手順どおりに進めてください

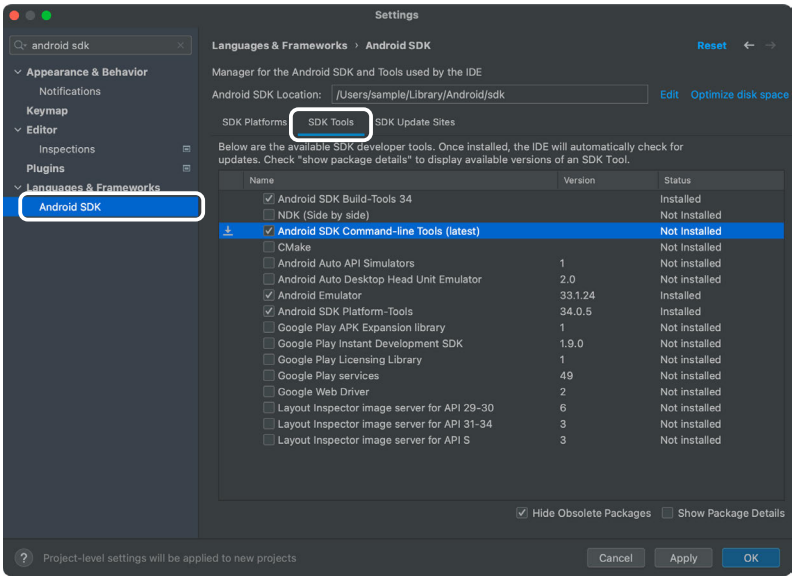
・`https://developer.android.com/studio/index.html`

完了したら、あらためて`flutter doctor`を実行します。Android Studioの項にチェックが付けばOKです。

```
[✓] Android Studio (version 2023.1)
```

続いて、Android toolchainをインストールします。Android Studioを起動し、アプリケーションメニューから「Settings」を選択します。Settingsウィンドウの検索窓に「Android SDK」と入力し、ツリーの中から「Android SDK」を選択します（**図1.3**）。「SDK Tools」タブを選択し、「Android SDK Command-line Tools (latest)」にチェックを入れ、OKボタンを押します。

図1.3 Settingsウィンドウ



再び**flutter doctor**を実行します。以下のようにライセンスの同意に関する内容が表示された場合は、メッセージ内のコマンドを実行します。

```
[!] Android toolchain - develop for Android devices (Android SDK version 34.0.0)
    ! Some Android licenses not accepted. To resolve this, run: flutter doctor
--android-licenses
```

```
# Android toolchainのライセンスに同意する
$ flutter doctor --android-licenses
```

すると、対話形式でライセンスの同意操作を進められます。**All SDK package licenses accepted**と表示されたら完了です。

```
（ライセンス条文）
----------------------------
Accept? (y/N): y
All SDK package licenses accepted
```

再び**flutter doctor**を実行します。以下のようにチェックが付けばAndroidの開発環境のインストールは完了です。

```
[✓] Android toolchain - develop for Android devices (Android SDK version 34.0.0)
```

iOSの開発環境をインストールする

`flutter doctor`の出力結果から、Xcodeの情報を確認しましょう。XcodeはAppleのプラットフォーム向けのIDE(統合開発環境)です。iOSの開発環境がまだ構築されていない場合は、以下のように`Xcode installation is incomplete`とメッセージが表示される可能性があります。

```
[✗] Xcode - develop for iOS and macOS
    ✗ Xcode installation is incomplete; a full installation is necessary for iOS
and macOS development.
      Download at: https://developer.apple.com/xcode/
      Or install Xcode via the App Store.
```

この場合は最初にXcodeをインストールします。最も簡単な方法はMac App StoreからXcodeをインストールする方法です(Tips参照)。Mac App Storeにて、「Xcode」というキーワードで検索すると簡単に見つけられます。

Xcodeをインストールしたら、以下のコマンドを実行します。

```
# Xcodeコマンドラインツールのディレクトリを指定
$ sudo xcode-select --switch /Applications/Xcode.app/Contents/Developer
# Xcodeの関連パッケージをインストール
$ sudo xcodebuild -runFirstLaunch
# XcodeとSDKのライセンスに同意
$ sudo xcodebuild -license
```

途中、何度かXcodeのライセンスに同意を求められることがあります。内容を確認し、同意する場合は「agree」と入力します。

```
By typing 'agree' you are agreeing to the terms of the software license agreement
s. Any other response will cancel. [agree, cancel]
agree
```

再び`flutter doctor`を実行し、状況を確認します。iOS Simulatorのランタイムがインストールされていない場合は、以下のように`Unable to get list of installed Simulator runtimes.`とメッセージが表示される可能性があります。

```
[!] Xcode - develop for iOS and macOS (Xcode 15.2)
    ✗ Unable to get list of installed Simulator runtimes.
```

この場合は以下のコマンドを実行してiOS Simulatorのランタイムをインストールします。

```
# iOS Simulatorのランタイムをインストール
$ xcodebuild -downloadPlatform iOS
```

再び`flutter doctor`を実行し、状況を確認します。以下のようにCocoaPods

がインストールされていない旨のメッセージが表示された場合は、CocoaPodsのインストールを実行します。

```
[!] Xcode - develop for iOS and macOS (Xcode 15.2)
    ✗ CocoaPods not installed.
```

CocoaPodsはiOSアプリ開発で用いられるRuby製のパッケージ管理ツールです。macOS標準のRubyを用いる場合は以下のコマンドでインストールします(Tips参照)。

```
# CocoaPodsをインストール
$ sudo gem install cocoapods
```

CocoaPodsのインストールが完了したら、**flutter doctor**で状況を確認します。以下のようにチェックが付けばiOSの開発環境のインストールは完了です。

```
[✓] Xcode - develop for iOS and macOS (Xcode 15.2)
```

Tips Xcodeのバージョンを使い分けるインストールのしかた

Mac App StoreからXcodeをインストールした場合は、新しいバージョンのXcodeがリリースされると上書きアップデートされます。実際のアプリ開発では複数バージョンのXcodeの使い分けが必要なこともあり、上書きアップデートが問題になるケースもあります。このようなケースに対応するため、筆者はAppleの開発者向けサイト[注a]からXcodeをダウンロードする方法を取っています。

ダウンロードしたXcodeは**/Applications**配下に配置しておきましょう。

注a https://developer.apple.com/download/applications/

Tips CocoaPodsがインストールできない場合

macOS標準のRubyのバージョンではCocoaPodsがインストールできない場合があります。そのようなときの対処法の一例を紹介します。

Rubyのバージョン管理ツールであるrbenvを使ってRubyのバージョンを上げます。まず、rbenvをインストールするためのパッケージマネージャであるHomebrewを用いてインストールします。Homebrewのインストールに関する詳細は公式Webサイト[注a]に記載があります。

注a https://docs.brew.sh/Installation

```
# Homebrewをインストール
$ /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/ins
tall/HEAD/install.sh)"
# Homebrewにパスを通す
$ echo "export PATH=\"\$PATH:/opt/homebrew/bin\"" >> ~/.zshenv
# 実行中のシェルにパスを適用
$ . ~/.zshenv
```

続いて、rbenvをインストールします。

```
# rbenvをインストール
$ brew install rbenv ruby-build
# rbenvにパスを通す
$ echo "eval \"\$(rbenv init - zsh)\"" >> ~/.zshrc
# 実行中のシェルにパスを適用
$ . ~/.zshrc
```

rbenvを使ってRubyをインストールします。今回は2.7.5をインストールします。

```
# バージョン2.7.5のRubyをインストール
$ rbenv install 2.7.5
# グローバルのRubyのバージョンを2.7.5に設定
$ rbenv global 2.7.5
```

以下のコマンドでRubyのバージョンを確認しましょう。

```
# Rubyのバージョンを出力
$ ruby --version
```

バージョン2.7.5が適用されていない場合はターミナルを再起動してください。
続いて、CocoaPodsをインストールします。

```
# CocoaPodsをインストール
$ gem install cocoapods
```

以下のコマンドでCocoaPodsのバージョンを確認しましょう。

```
# CocoaPodsのバージョンを出力
$ pod --version
```

無事にバージョンが表示されれば完了です。

Android Studioの設定 —— Flutterと親和性の高いIDE

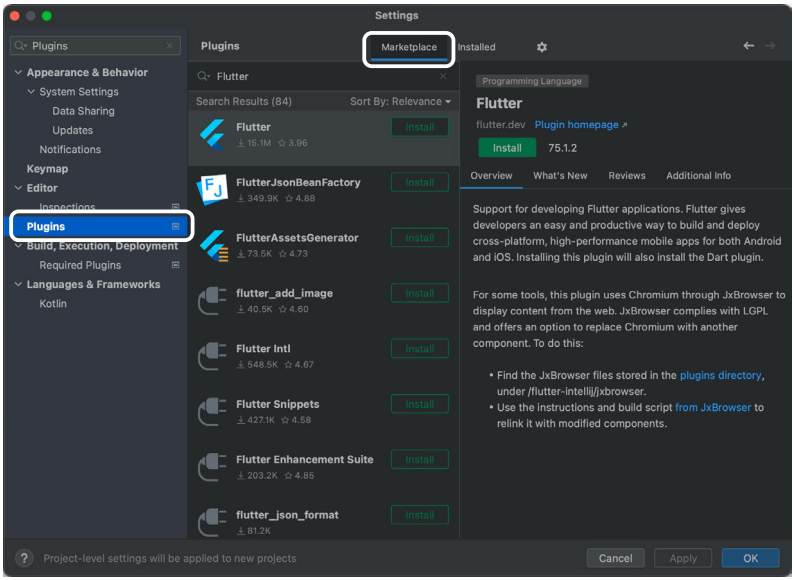
Flutterアプリの開発にはAndroid Studio、IntelliJ IDEAやVisual Studio Code（以降、VS Code）がよく用いられます。これらのIDEにはプラグインが提供されており、コード補完やデバッグ、ステップ実行などの機能が利用できます。

本書では環境構築の過程でAndroid Studioをインストールしているので、Android Studioを中心に解説していきます（VS Codeも軽量で扱いやすく筆者は気に入っています）。

Android Studioでプラグインをインストールするには、アプリケーションメニューから「Settings」を選択します。Settingsウィンドウの検索窓に「Plugins」と入力し、ツリーの中から「Plugins」を選択します（**図1.4**）。「Marketplace」タブを開いたら、「Flutter」というキーワードでプラグインを検索します。Flutterプラグインが見つかったらインストールボタンを押します。Flutterプラグインと併せてDartプラグインも同時にインストールされます。

インストールが完了したらAndroid Studioを再起動しましょう。

図1.4　**Settingsウィンドウ**



1.3 fvmによるFlutterのバージョン管理

プロジェクトごとにFlutterのバージョンを切り替えることができるfvmというツールを紹介します。fvmを利用することで、Flutterのバージョンを切り替えるためにFlutter SDKを再インストールする手間を省くことができます。「過去に開発したFlutterアプリをメンテナンスするために、古いバージョンのFlutterを入れなおす……」といった面倒ごとから解放されます。

fvmのインストール

まずfvmをインストールするために、macOS向けのパッケージ管理ツールHomebrewをインストールします。CocoaPodsのインストール時にHomebrewを導入した場合はこの手順は不要です。

```
# Homebrewをインストール
$ /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
# Homebrewにパスを通す
$ echo "export PATH=\"\$PATH:/opt/homebrew/bin\"" >> ~/.zshenv
# 実行中のシェルにパスを適用
$ . ~/.zshenv
```

次にHomebrewを利用してfvmをインストールします。

```
# fvmのGitHubリポジトリをHomebrewに追加
$ brew tap leoafarias/fvm
# fvmをインストール
$ brew install fvm
```

これでfvmのインストールは完了です。

fvmを利用したFlutterのインストール

続いて、fvmを利用して最新安定版のFlutterおよびそれに対応したDartの環境をインストールしましょう。インストール可能なFlutterバージョンをターミナルから確認します。

```
# インストール可能なFlutterバージョンを出力
$ fvm releases
```

stableと書かれたバージョンが最新の安定版となります。執筆時点のコマンド実行結果を以下に示します。最新の安定版は3.16.9でした。

```
.
.
.
（省略）
Nov 30 23  | 3.16.2
Dec 5 23   | 3.18.0-0.1.pre
Dec 6 23   | 3.16.3
Dec 13 23  | 3.16.4
Dec 14 23  | 3.18.0-0.2.pre
Dec 20 23  | 3.16.5
Jan 10 24  | 3.16.6
Jan 10 24  | 3.19.0-0.1.pre
Jan 11 24  | 3.16.7
Jan 17 24  | 3.16.8
Jan 18 24  | 3.19.0-0.2.pre
-----------------------------------
Jan 25 24  | 3.16.9           stable
-----------------------------------
-----------------------------------
Jan 26 24  | 3.19.0-0.3.pre   beta
-----------------------------------
```

それではバージョン3.16.9のFlutterをインストールします。

```
# バージョン3.16.9のFlutterをインストール
$ fvm install 3.16.9
```

これでFlutterのインストールは完了です。

後述の「1.4 プロジェクトの作成」の節で、プロジェクトで利用するfvmのFlutterのバージョンを指定する方法を解説します。その設定が完了すると、プロジェクトのディレクトリ配下でfvmコマンドを介してFlutterを使用することができます。たとえば、Flutterのバージョン番号を確認するには以下のように実行します。

```
# fvmを経由してFlutterのバージョンを出力
$ fvm flutter --version
# fvmを経由してDartのバージョンを出力
$ fvm dart --version
```

「1.2 Flutterの環境構築」の節でインストールしたFlutterはそのまま残って