

Setting Up OpenCV for C++ using CMake and VS Code on Mac OS



By [Adam McQuistan](#) in C++ 03/30/2020 [Comment](#)

Introduction

In this article I demonstrate how to install OpenCV for C++ from source using cmake. Following this I show how to configure my favorite code editor, Visual Studio Code (VS Code), along with the awesome Microsoft extensions CMake Tools and C/C++ IntelliSense which provides a fantastic, productivity boosting, development environment.

Plano Anual com 10%Off



Tudo sobre investimentos em uma única plataforma.

Ad Inside Research



Installing XCode

In order to utilize the popular Mac OS clang compiler you should have XCode and the XCode tools installed. Since installing XCode is covered quite well by Apple Developer Support and a very common task among technologists utilizing Mac OS platforms I will simple link to [Apple's docs](#) for that part.

Installing Homebrew

[Homebrew](#), aka brew, is a software package manager for the Mac OS platform that is heavily used among developers. To install Homebrew execute this command from your terminal.

```
$ /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install.sh)"
```

Installing CMake

[CMake](#) is a cross platform build tool popular among native C/C++ developers. I use CMake in this tutorial to build and install OpenCV for C++ as well as run the demo project. CMake can be easily installed using brew as follows.

```
$ brew install cmake
```

Installing OpenCV from Source with CMake

Before I can build and install OpenCV I must clone its source repo from GitHub. In addition to the main source I will pull down the extra contribution modules for install as well. The extra modules are optional and contain experimental algorithms and features but in my opinion they are good to have for development purposes.

First make a top level opencv directory, change directories into it and clone the two repositories then checkout the most recent tagged release versions which are 4.2.0 as of the date of writing this article.

```
$ mkdir opencv
$ cd opencv
$ git clone https://github.com/opencv/opencv.git
$ cd opencv
$ git checkout tags/4.2.0
$ cd ..
$ git clone https://github.com/opencv/
$ cd opencv_contrib
$ git checkout tags/4.2.0
$ cd ..
```

I should now be in the top level opencv directory that contains the two aforementioned repositories. Next I create two more directories (i) an install directory which will be the target for my installation of OpenCV and, (ii) another named build_opencv for building the sources in.

```
$ mkdir install build_opencv
$ ls -l
total 0
drwxr-xr-x  2 adammcquistan  staff   64 Mar 29 21:45 build_opencv
drwxr-xr-x  2 adammcquistan  staff   64 Mar 29 21:45 install
drwxr-xr-x 21 adammcquistan  staff  672 Mar 29 21:35 opencv
drwxr-xr-x 13 adammcquistan  staff  416 Mar 29 21:36 opencv_contrib
```

Next I change directories into the build_opencv directory and configure CMake as shown below.

```
$ cd build_opencv
$ cmake -D CMAKE_BUILD_TYPE=RELEASE \
-D CMAKE_INSTALL_PREFIX=../install \
-D INSTALL_C_EXAMPLES=ON \
-D OPENCV_EXTRA_MODULES_PATH=../opencv_contrib/modules \
-D BUILD_EXAMPLES=ON ../opencv
```

The last steps to install is to kick off the multithreaded build to compile the project followed by installing it into the install directory as seen below.

```
$ export CPUS=$(sysctl -n hw.physicalcpu)
$ make -j $CPUS
$ make install
```

A Melhor Plataforma com 10%Off



As melhores carteiras recomendadas você encontra na plataforma da Inside Rese:

Ad Inside Research



Verify OpenCV Install with Simple Demo Project

Before moving on to the VS Code setup its a good idea to verify that OpenCV was properly built and installed using a simple demo project. To do this I change directories back to the top level opencv directory I created at the start of this tutorial (ie, the one that contains the directories opencv, opencv_contrib, build_opencv and install directories). Then I make a new directory called simple_demo and change directories into it.

```
$ cd ..
$ ls -l
drwxr-xr-x  2 adammcquistan  staff   64 Mar 29 21:45 build_opencv
drwxr-xr-x  2 adammcquistan  staff   64 Mar 29 21:45 install
drwxr-xr-x 21 adammcquistan  staff  672 Mar 29 21:35 opencv
drwxr-xr-x 13 adammcquistan  staff  416 Mar 29 21:36 opencv_contrib
$ mkdir simple_demo && cd simple_demo
```

Inside the simple_demo directory I make a main.cpp file and fill it with the following code which includes the opencv.hpp and highgui.hpp headers as well as defines a main() method. The main method looks for a argument to be passed to the program representing the path to a image file to be read in and displayed using OpenCV.

```
// main.cpp

#include <iostream>
#include <opencv2/opencv.hpp>
#include <opencv2/highgui.hpp>

using std::cout;
using std::endl;

int main(int argc, char** argv) {

    if (argc != 2) {
        cout << "Expecting a image file to be passed to program" << endl;
        return -1;
    }

    cv::Mat img = cv::imread(argv[1]);

    if (img.empty()) {
        cout << "Not a valid image file" << endl;
        return -1;
    }

    cv::namedWindow("Simple Demo", cv::WINDOW_AUTOSIZE);
    cv::imshow("Simple Demo", img);

    cv::waitKey(0);
    cv::destroyAllWindows();

    return 0;
}
```

Additionally, inside the `simple_demo` directory I create a `CMakeLists.txt` file, which is essentially a CMake configuration script, and place the following in it.

```
# CMakeLists.txt

# Older versions of CMake are likely to work just fine but, since
# I don't know where to cut off I just use the version I'm using
cmake_minimum_required(VERSION "3.17")

# name of this example project
project(simple-demo)

# set OpenCV_DIR variable equal to the path to the cmake
# files within the previously installed opencv program
set(OpenCV_DIR /Users/adammcquistan/code/c++/opencv/install/lib/cmake/opencv4)

# Tell compiler to use C++ 14 features which is needed because
# Clang version is often behind in the XCode installation
set(CMAKE_CXX_STANDARD 14)

# configure the necessary common CMake environment variables
# needed to include and link the OpenCV program into this
# demo project, namely OpenCV_INCLUDE_DIRS and OpenCV_LIBS
find_package( OpenCV REQUIRED )

# tell the build to include the headers from OpenCV
include_directories( ${OpenCV_INCLUDE_DIRS} )

# specify the executable target to be built
add_executable(simple-demo main.cpp)

# tell it to link the executable target against OpenCV
target_link_libraries(simple-demo ${OpenCV_LIBS} )
```

Almost there!

Now I create a build directory within the `simple_demo` directory containing the `CMakeLists.txt` and `main.cpp` files then configure and build the demo project as shown below.

```
$ mkdir build && cd build
$ cmake ..
$ make
```

A Melhor Plataforma com 10%Off



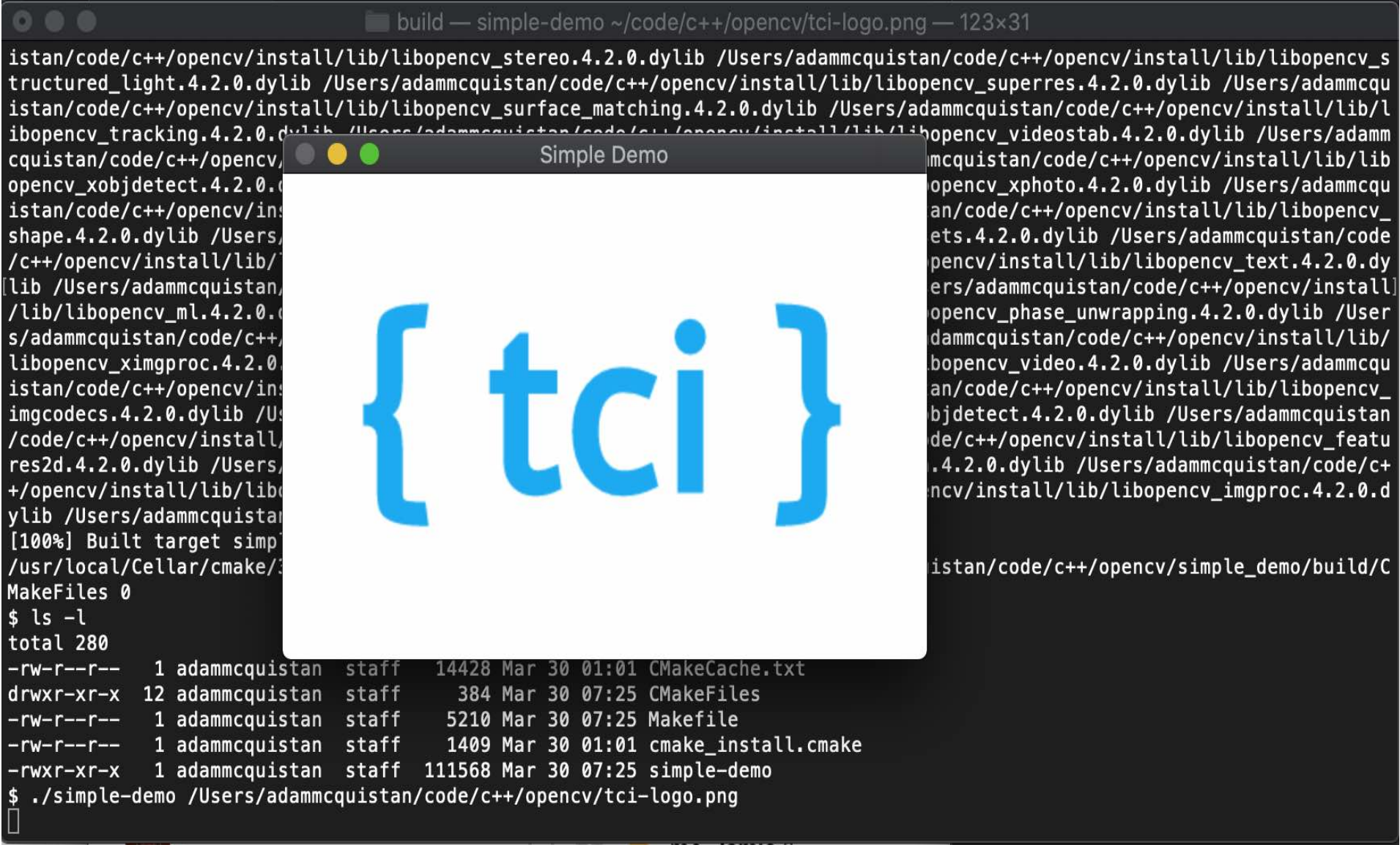
As melhores carteiras recomendadas você encontra na plataforma da Inside Rese:

Ad Inside Research



And finally, I execute the program passing it a path to an image file, the TCI logo in this example, and watch the HighGUI window display the image.

```
$ ./simple-demo /path/to/tci-logo.png
```



Before moving on I delete all the contents inside the build directory because I will use the CMake Tools extension in VS Code to regenerate its contents.

```
$ ls -l
total 280
-rw-r--r--  1 adammcquistan  staff   14428 Mar 30 07:47 CMakeCache.txt
drwxr-xr-x 12 adammcquistan  staff    384 Mar 30 07:48 CMakeFiles
-rw-r--r--  1 adammcquistan  staff   5210 Mar 30 07:47 Makefile
-rw-r--r--  1 adammcquistan  staff   1409 Mar 30 07:47 cmake_install.cmake
-rwxr-xr-x  1 adammcquistan  staff  111568 Mar 30 07:48 simple-demo
$ rm -ifr *
```

A Melhor Plataforma com 10%Off



As melhores carteiras recomendadas você encontra na plataforma da Inside Rese:

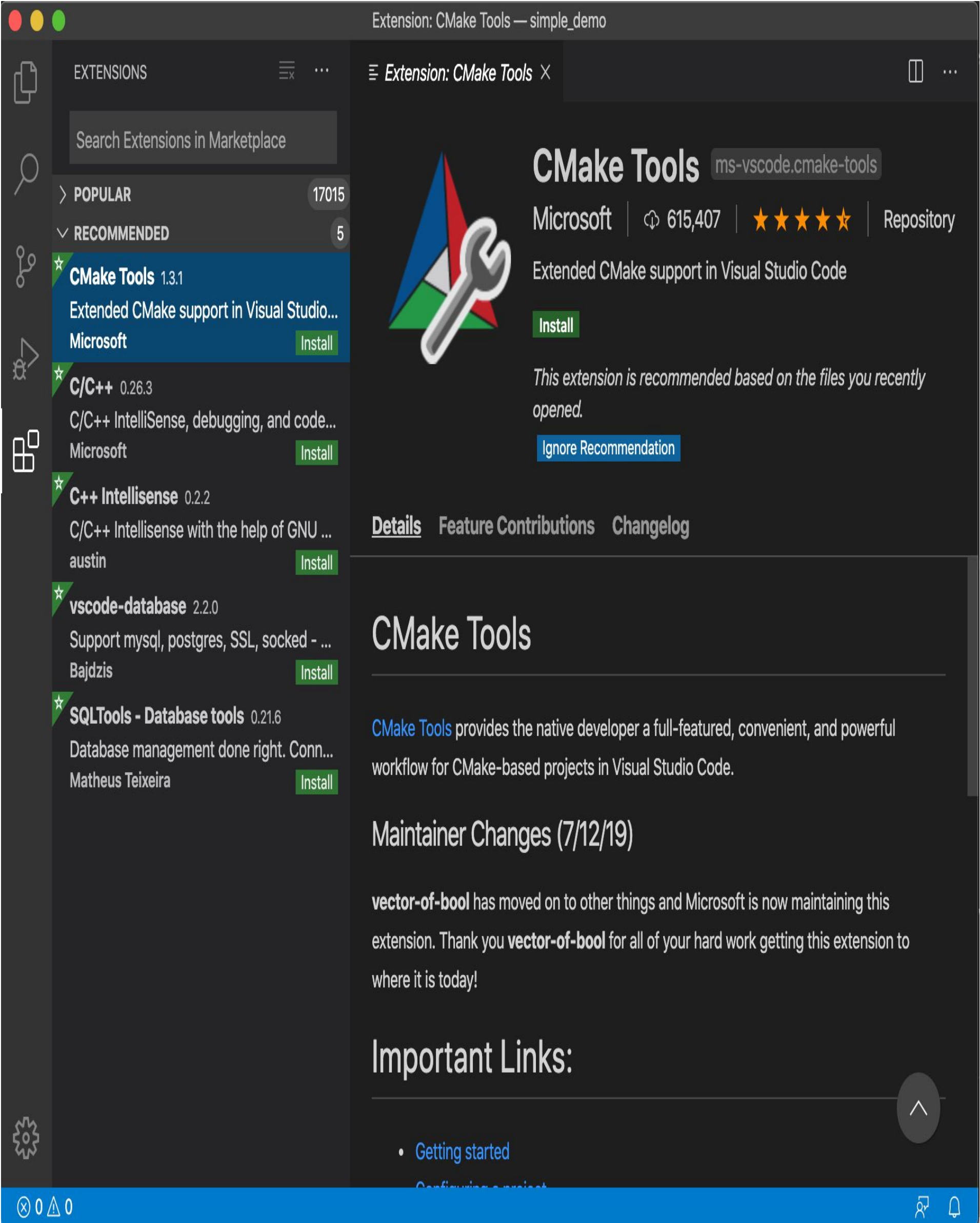
Ad Inside Research



Installing VS Code

If you don't already have Microsoft's VS Code editor installed please do so from [here](#). I really love using VS Code and I'm confident you will enjoy using it too in just about any language.

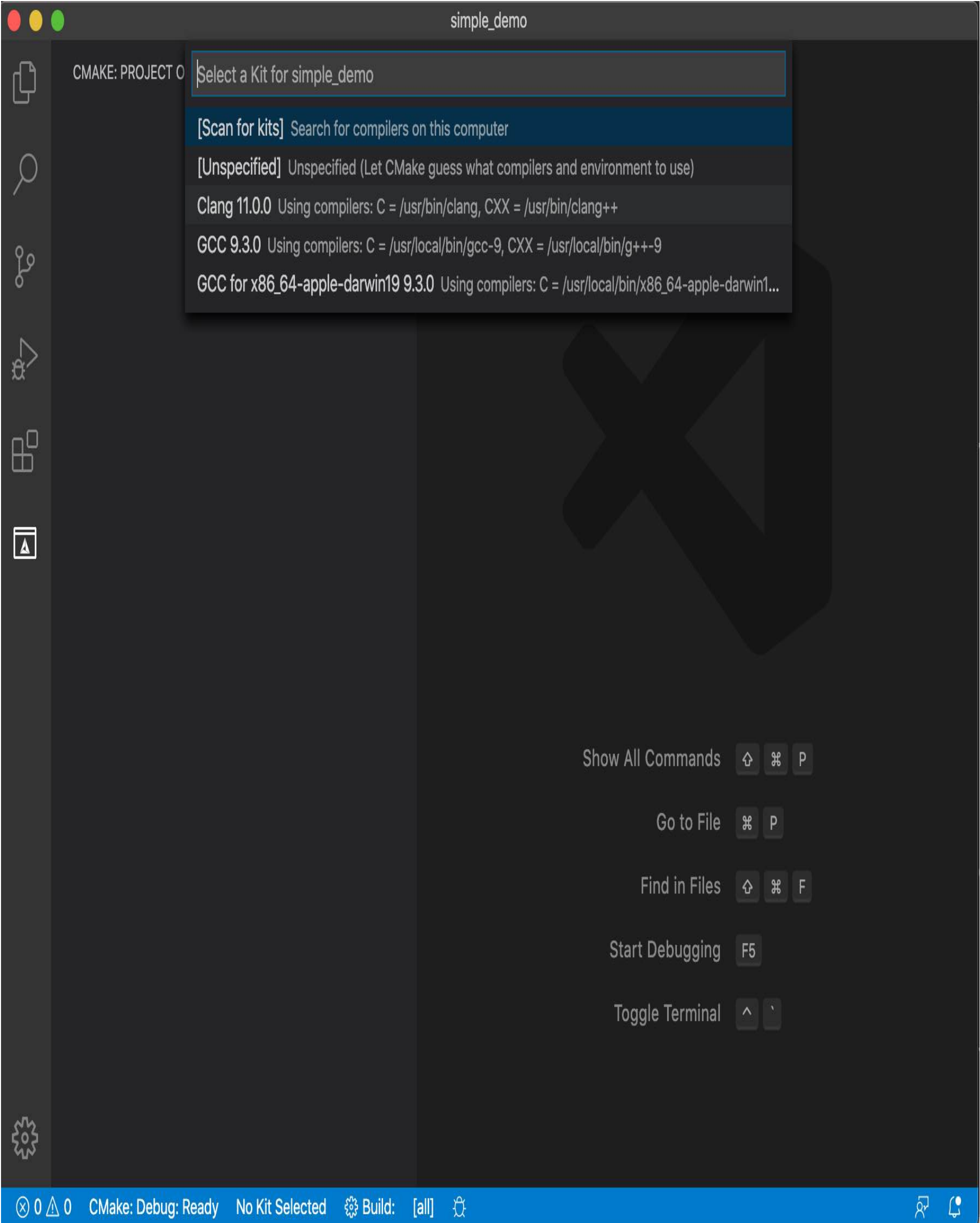
First off I open the editor in the simple_demo directory as shown below. Next I click on the Extensions Tab, expand the recommended submenu (or search for using the search bar) and install C/C++ IntelliSense as well as CMake Tools by Microsoft.



After installing the C/C++ IntelliSense and CMake Tools extensions a new CMake Tools tab within the left toolbar is added. I click on the CMake tab and then click on the Configure action button.

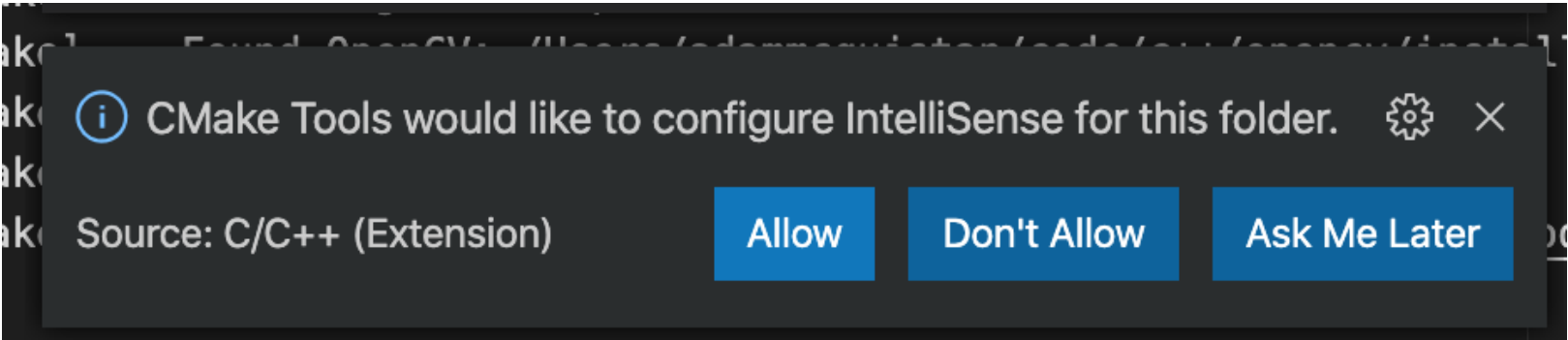


The command menu will open with a dropdown and I select the Clang compiler.



This will result in a list of build targets being displayed in the CMake tab.

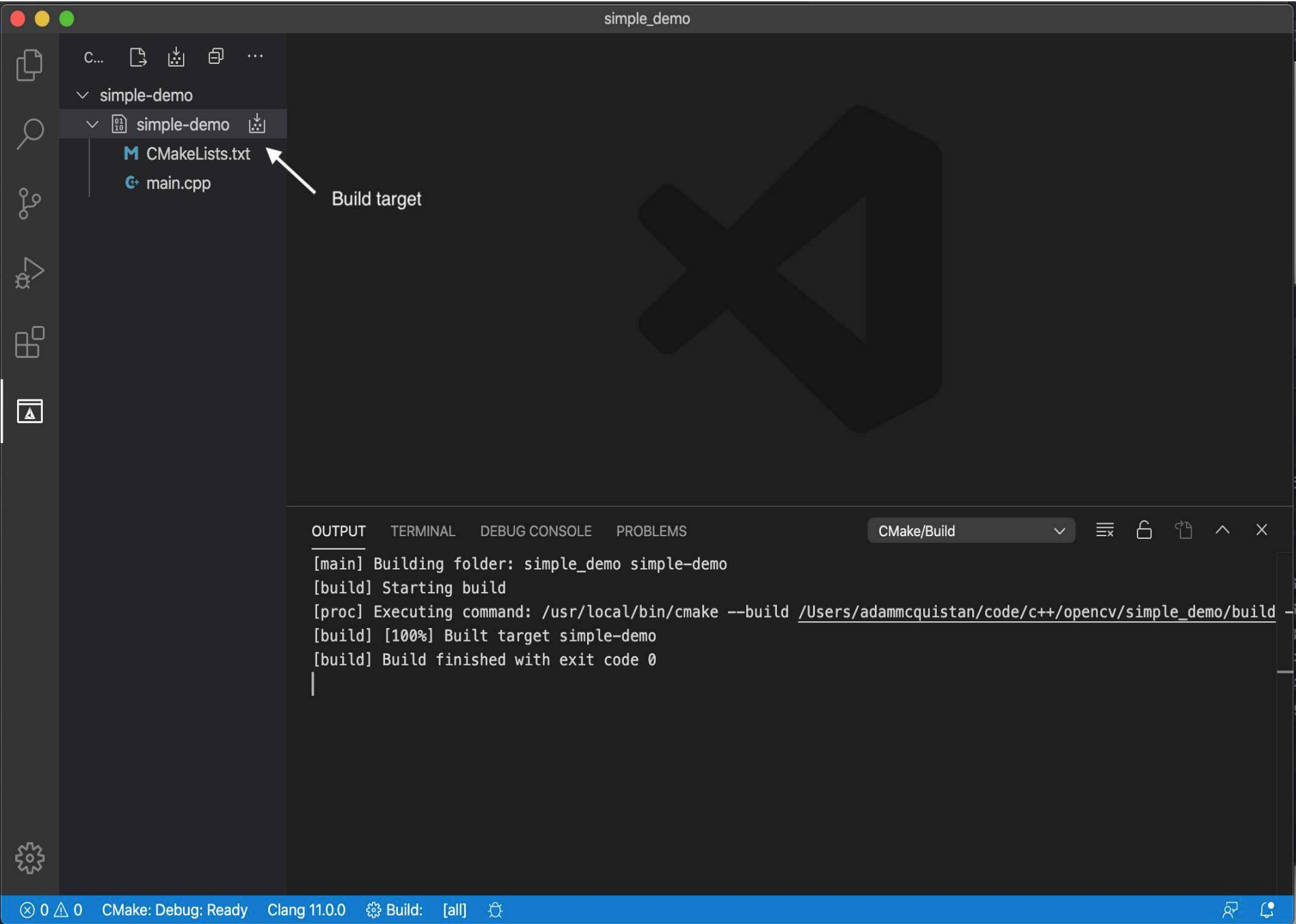
Shortly after this VS Code will prompt you to ask if CMake Tools can configure IntelliSense for the project. Click Allow.



If for some reason this prompt does not appear or you accidentally close it then create a directory named `.vscode` directly under the `simple_demo` directory (it may already exist) and create a `settings.json` file within it and place the following in it.

```
// settings.json
{
  "C_Cpp.default.configurationProvider": "vector-of-bool.cmake-tools"
}
```

Back in the CMake Tools tab I click the build action button next to the simple-demo target to build it.



Then in the terminal I issue the same command shown earlier to test the newly built executable in the build directory.

```
$ ./simple-demo /path/to/tci-logo.png
```

A Melhor Plataforma com 10%Off



As melhores carteiras recomendadas você encontra na plataforma da Inside Rese:

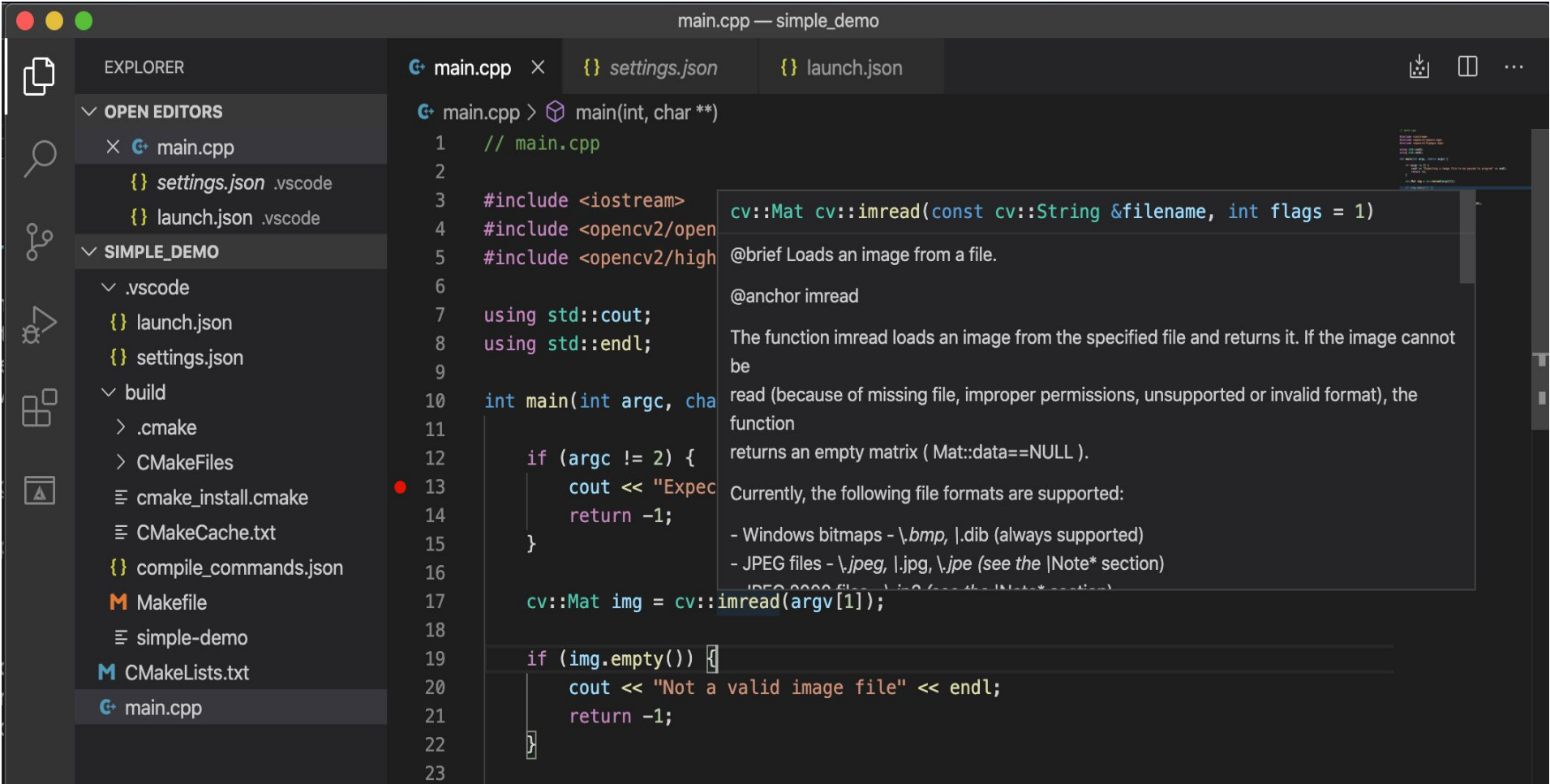
Ad Inside Research



Configuring and Using VS Code Debugging Tools

Ok so thats great but, there are two hugely beneficial things that I'm leaving out (i) IntelliSense and (ii) debugging which are both for helping to write and inspect your C++ code. The IntelliSense part is actually ready to go already.

At this point I can open my main.cpp file in the editor and immediately see the amazing IntelliSense as shown below when I hover my mouse over the cv::imread(...) function.

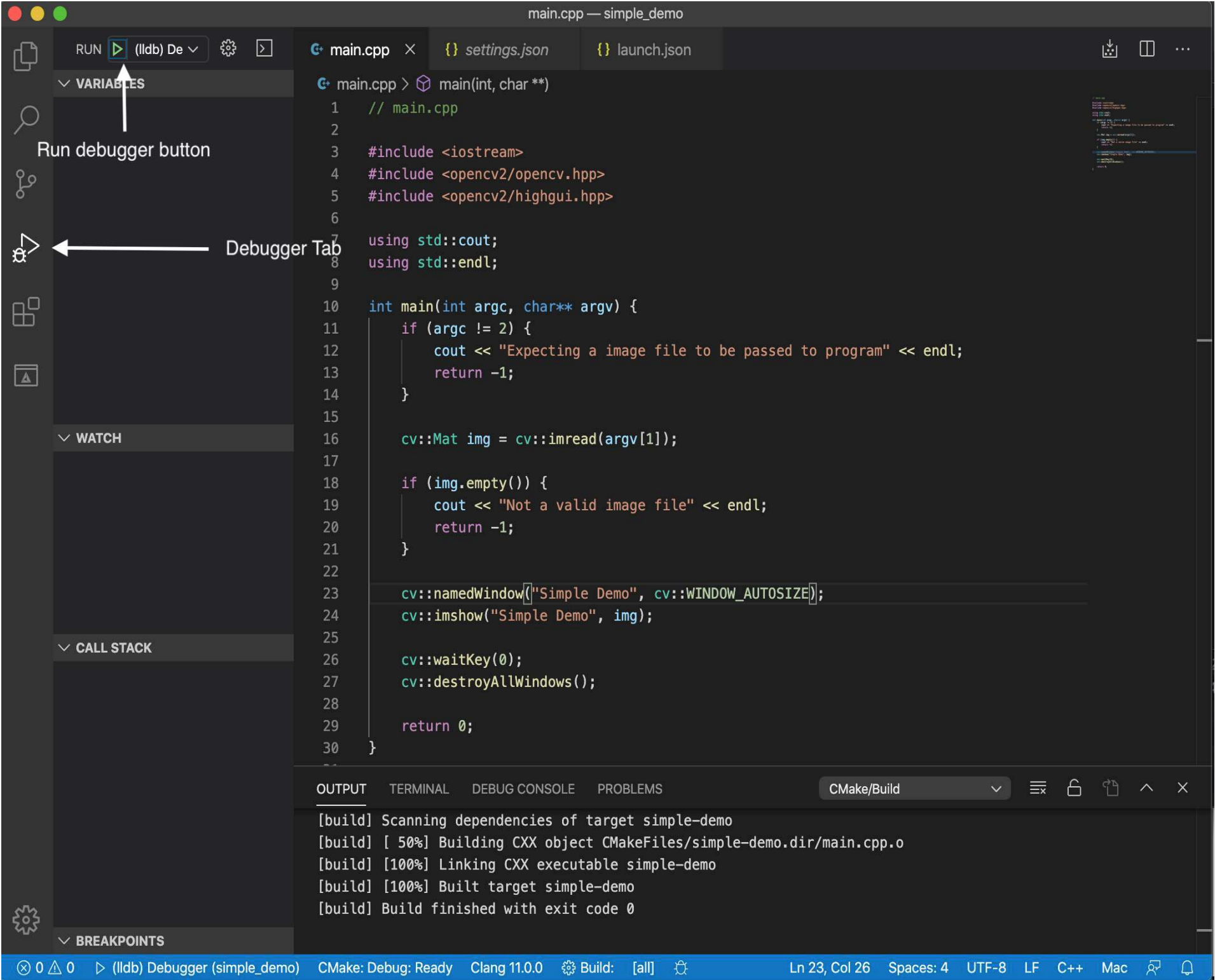


In order to enable the VS Code debugging tools I must create another configuration file within the .vscode directory named launch.json and place the following in it.

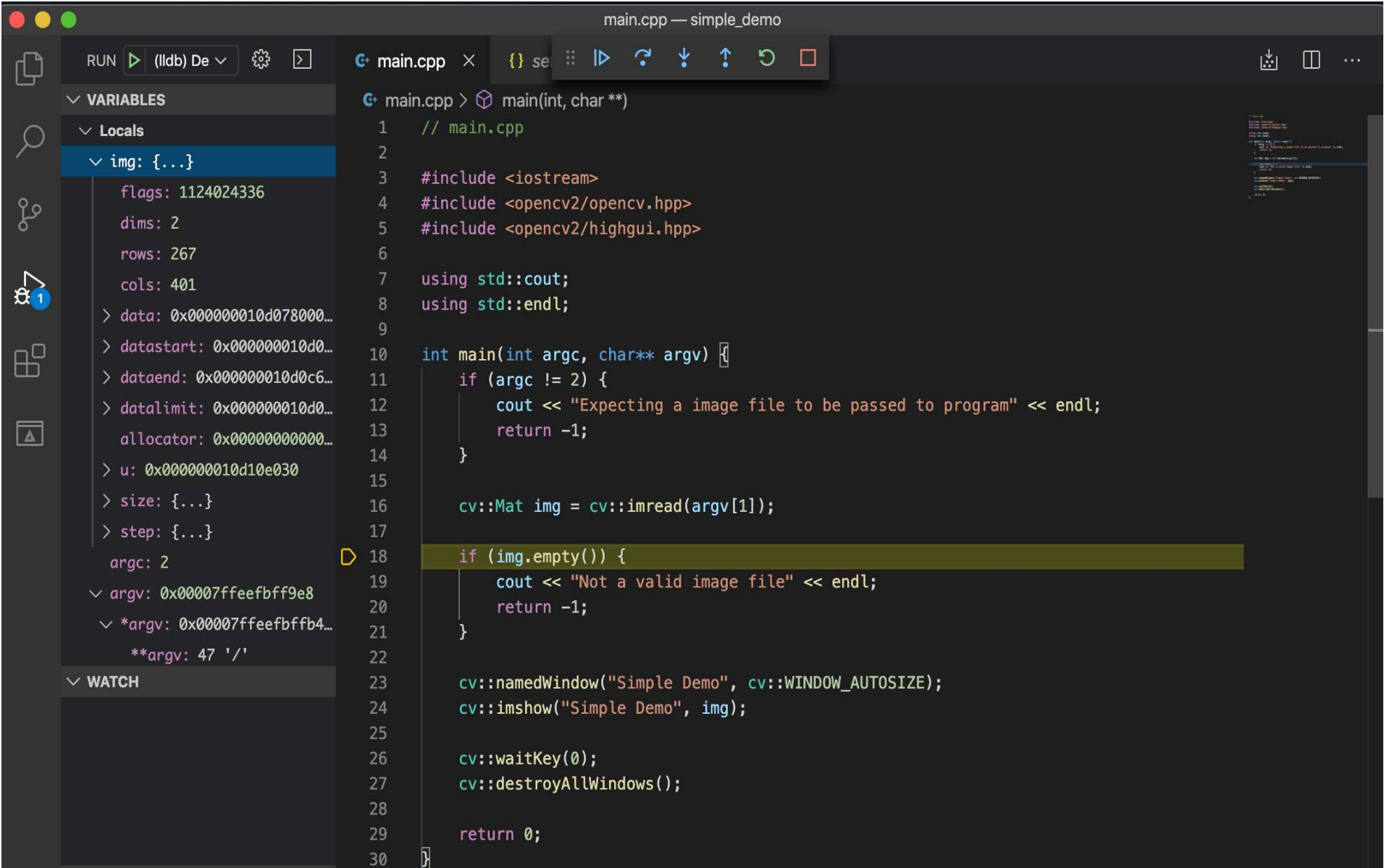
```
// launch.json
{
  "version": "0.2.0",
  "configurations": [
    {
      "name": "(lldb) Debugger",
      "type": "cppdbg",
      "request": "launch",
      "program": "build/simple-demo",
      "args": ["/path/to/tci-logo.png"],
      "stopAtEntry": true,
      "cwd": "${workspaceFolder}",
      "environment": [],
      "externalConsole": true,
      "MIMode": "lldb"
    }
  ]
}
```

This config file says to use the cppdbg debugger, to pass in a path to an image file to display, pause execution at the start of the main() method and to launch a separate terminal for executing the program.

In order to launch the debugger I click on the Debugger tab then click the debug action button as shown below.



What occurs next is the debugger shows the program paused at the first line of the main() method and the debug tab menu shows the program variables as well as buttons for controlling the execution of the program.



Resources For Learning More

- [Building Computer Vision Projects with OpenCV 4 and C++: Implement complex computer vision algorithms and explore deep learning and face detection](#) is a fantastic project based book to building great things using C++ and OpenCV

[thecodinginterface.com](#) earns commision from sales of linked products such as the books above. This enables providing continued free tutorials and content so, thank you for supporting the authors of these resources as well as [thecodinginterface.com](#)

Conclusion

In this article I have demonstrated how to install OpenCV for C++ plus how to utilize VS Code along with CMake Tools and C/C++ IntelliSense extensions for a professional level, high productivity, development environment.

As always, I thank you for reading and please feel free to ask questions or critique in the comments section below.

C++

OpenCV

f

🐦

in

Share with friends and colleagues

★

51 likes

Navigation


★

💬

Categories	
Machine Learning	2
Data Engineering	16
DevOps	2
Python	13
Serverless	3
PostgreSQL	7
Linux	4
C++	2
Java	15
JavaScript	1

Cupom 10%
pit-news-10

amazon



Best-Selling Laptops

[Shop now](#)

Privacy



Community favorites for C++

← [HOW TO UPLOAD AND DOWNLOAD FILES IN AWS S3 WITH PYTHON AND BOTO3](#)

[HOW TO CONSTRUCT AN OPENCV MAT OBJECT FROM C++ ARRAYS AND VECTORS](#) →

ALSO ON THE CODING INTERFACE

Beginner's Guide to Gradle for Java ...

2 years ago • 2 comments

In this tutorial I will be describing how to get up and going quickly with ...

JavaFX with Gradle, Eclipse, Scene ...

2 years ago • 1 comment

In this second article I will be building out the Random Number generator app ...

RESTful Consumption using the Java 11+ ...

2 years ago • 2 comments

In this article I demonstrate using the new HTTP Client API introduced in Java ...

Deploying F AWS: Amazi

a year ago • 1 c

In this article I to deploy a sin Todo app com

Sponsored

Novo método para tratar fungos nas unhas vira febre em Teresópolis

Nail Cure

Médico Brasileiro: Eu imploro aos Brasileiros que abandonem esses três alimentos

Dr. Rafael Freitas

Shampoo que reduz cabelos brancos. Não é tintura

Menfirst

Comprar agora

Luma de Oliveira tem 56 anos de idade e hoje é irreconhecível

Mighty Scoops

Conheça a Máscara KNIT Max95

KNIT

Comprar agora

Cientistas revelam como figuras famosas realmente eram. As fotos são incríveis!

Desafiomundial

Os óculos que faliram as óticas chegaram ao Brasil

5 Comments

The Coding Interface

Disqus' Privacy Policy

1 Login

Favorite

Tweet

Share

Sort by Best



Join the discussion...

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name



Stephan Müller • 4 months ago

Hi,
first of all, thanks for the tutorial. I didn't get too far though, the command `git clone https://github.com/opencv/` fails in my terminal with the error "fatal: destination path 'opencv' already exists and is not an empty directory.", which kind of makes sense since this directory was already created by the `git clone https://github.com/opencv/opencv.git` command before. Any ideas?
Thanks
^ | v • Reply • Share ›



Natan Loterio • 8 months ago • edited

[SOLVED]
Hi there,

I just followed exactly your instructions, but the CMake Could not find a package configuration file provided by "OpenCV" with any of the following names: `OpenCVConfig.cmake`, `opencv-config.cmake`
I changed just the `OpenCV_DIR` to the one created during the install to `set(OpenCV_DIR /Users/natanloterio/workspace/libs/cpp/build_open/CMakeFiles/Export/lib/cmake/opencv4)`
^ | v • Reply • Share ›



Natan Loterio → Natan Loterio • 8 months ago

I just searched by where the `OpenCVConfig.cmake` was created, and used the folder I found. I was
`/Users/natanloterio/workspace/libs/cpp/install/lib/cmake/opencv4/`

So, it worked. Thanks
^ | v • Reply • Share ›



kujahleague • a year ago

Thanks for your explanation. I am very new to using visual studio code in Mac.
I followed everything until `launch.json` and first when I use your code it say could not find the program so I changed the path in "program": full path to the program (inside build folder, without the `.exe`)
Now I could click the debug button, however, the debugger never stop to ask for variable. The bottom bar is orange color. Please suggest.
^ | v • Reply • Share ›



Keng Thé • a year ago

Sponsored

Novo método para tratar fungos nas unhas vira febre em Teresópolis

Nail Cure

3 Erros que donos de cachorro cometem e diminuem a vida canina

Petvi

Nova Máscara KNIT Max95, a única que realmente protege!

KNIT

Médico Brasileiro: Eu imploro aos Brasileiros que abandonem esses três alimentos

Dr. Rafael Freitas

Depois de perder peso Jorge Garcia parece modelo

Healthy George

Mulheres: "É assim que se combate a queda de cabelo após os 50 anos"

Doutor Nature



Tags

- Natural Language Processing
- Apache Flink
- AWS EMR
- Apache Hive
- ksqlDB
- Stream Processing
- Kafka
- PySpark
- Kubernetes
- SciKit Learn
- AWS Glue
- Redshift
- Data Engineering
- AWS Kinesis
- AWS Serverless Application Model
- AWS
- AWS Elastic Transcoder
- AWS-Lambda
- sysadmin
- C++
- OpenCV
- AWS S3
- psql
- PostgreSQL
- Databases
- CentOS
- Ubuntu
- Linux
- Sorting
- Animations
- Try-With-Resources
- Factory Method
- ChoiceBox
- ComboBox
- Java HTTP Client
- REST
- Intro to Java
- Jsoup
- Web Scraping
- MachineLearning
- JavaScript
- NodeJS
- OOP
- FXML
- Eclipse
- JavaFX
- uWSGI
- Nginx
- BeautifulSoup
- requests
- TextBlob
- Build Systems
- DevOps
- Gradle
- static assets
- Flask
- cloning
- Collections
- streams
- Java
- Python Social Auth
- Django Guardian
- VueJS
- Python
- Django Auth
- Django

Community Favorites

How To Construct an OpenCV Mat Object from C++ Arrays and Vectors

🕒 04/07/2020 📁 C++

How To Use Window Functions in SQL

🕒 03/29/2021 📁 Data Engineering

Implementing a Serverless Flask REST API using AWS SAM

🕒 12/02/2020 📁 Serverless

Setting Up OpenCV for C++ using CMake and VS Code on Mac OS

🕒 03/30/2020 📁 C++

Bridging Node.js and Python with PyNode to Predict Home Prices

🕒 08/20/2019 📁 JavaScript

Django Authentication Part 1: Sign Up, Login, Logout

🕒 06/07/2019 📁 Python

Django Authentication Part 4: Email Registration and Password Resets

🕒 06/19/2019 📁 Python

Example Driven High Level Overview of Spark with Python

🕒 03/12/2021 📁 Data Engineering

How To Upload and Download Files in AWS S3 with Python and Boto3

🕒 03/27/2020 📁 Python

Sitemap

- Home
- Blog
- Courses
- About Us
- Terms
- Privacy

Follow The Coding Interface



Copyright © theCodingInterface 2021

Powered by [Django](#) and [Vue.js](#)