Integrated Development Environments

And why you should care

Table of Contents

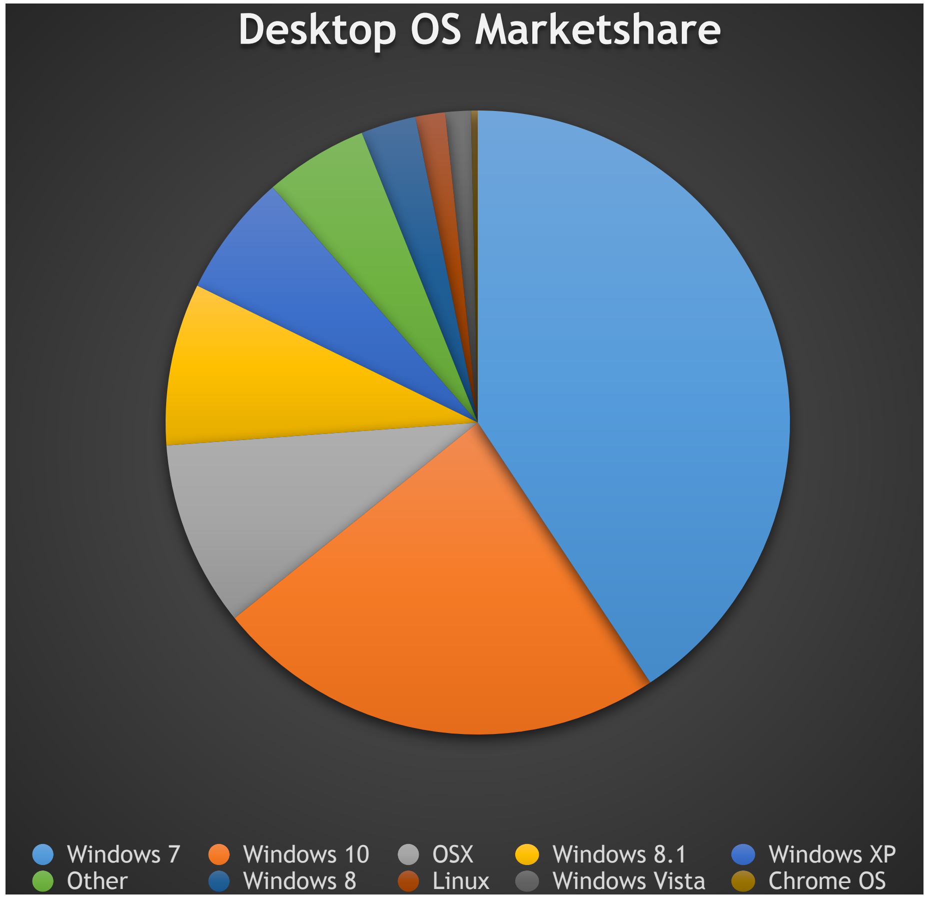
Microsoft and Linux, one happy family

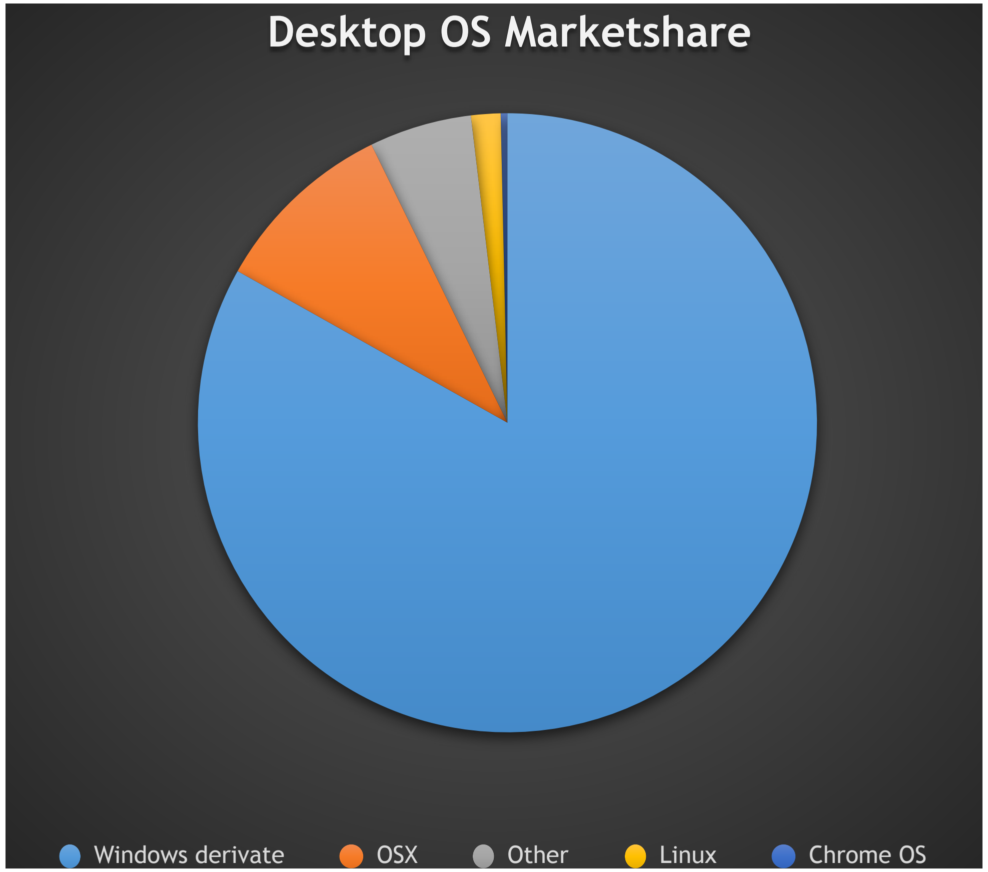
EMBRACE THE DARK SIDE

Embrace the dark side

- There seems to be a widely accepted axiom that Microsoft, Windows in particular and scientific computing don’t mix well
  - Linux is for the pros, command-line FTW
  - Windows is click-click, plus it’s *Microsoft*
- Avoiding starting a flame war, there are a few advancements which are generally „good to know”
- Why is it important what happens in the Windows domain?

Embrace the dark side





Desktop operating system usage based on web browser usage. (Source: https://www.netmarketshare.com)

Embrace the dark side

- Windows as a serious compute platform?
  - Nano Server is an ultra minimalistic Windows Server installation
    - 200 MB taken on disk, no 32-bit support, no GUI
    - No MSI, WSA only (the trusted version of the Windows Store APPX package format)
    - Clean install has only 20 processes running
    - Only PowerShell (command-line) management
  - More information here.

Embrace the dark side

Satya Nadella @ Build 2016 talking about Microsoft's plans in embracing Linux and open-source software

Embrace the dark side

- Microsoft is shifting to being a service provider as opposed to being a software vendor
- Key to having a large subscriber base is the ability to provide service on all of the platforms
  - Microsoft Office Mobile was published on iOS and Android ~6 months ahead of Windows 10 Mobile
  - Discontinue proprietary .NET in favor of open-source version
- Microsoft internally has a Linux flavor of their own (Azure cloud services)

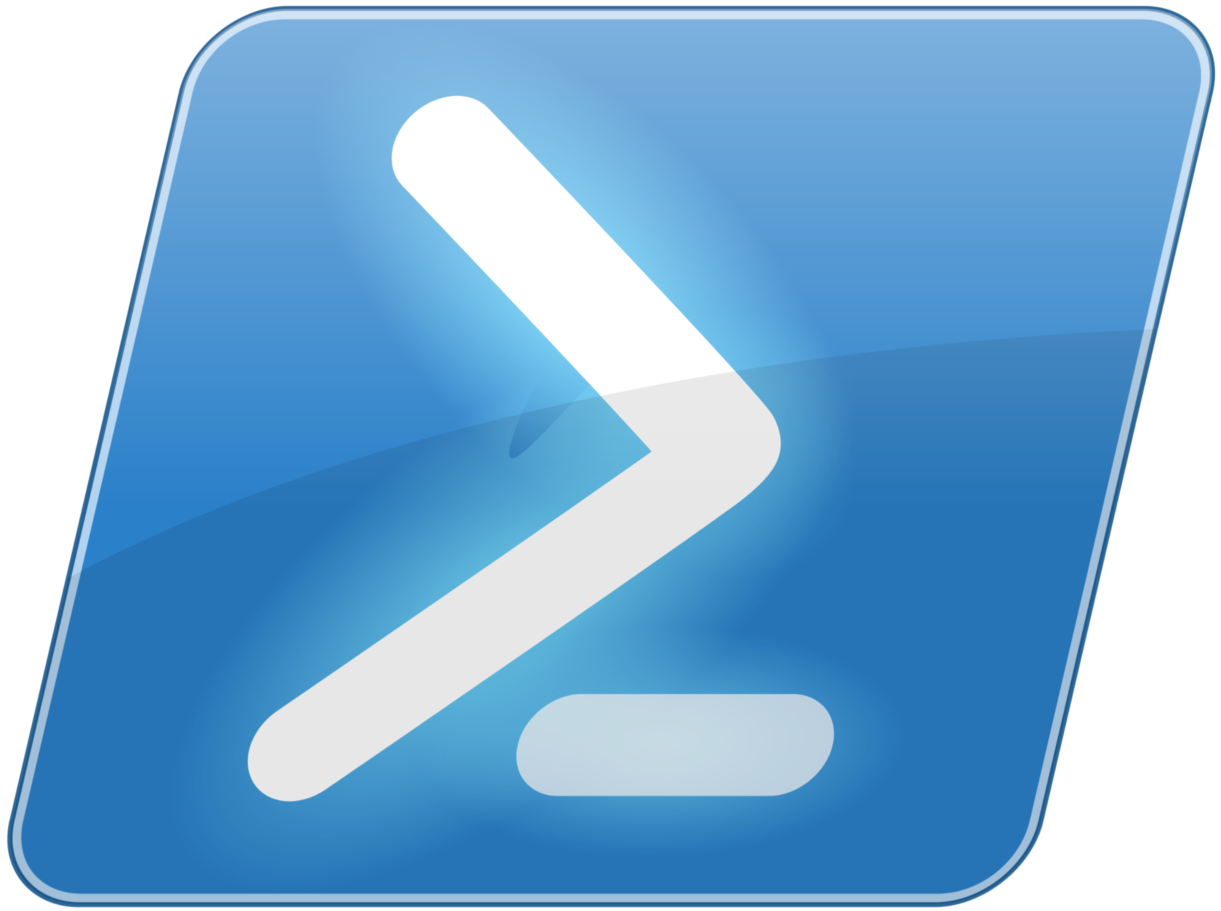
.NET for the masses

- What is .NET?
  - It is a software execution framework (Common Language Runtime, aka. CLR) and a standard framework class library which together provide language interoperability.
- Why .NET?
  - Language interoperability is seamless.
    - Author a library in one language, consume in another
  - Portability
    - On 27 June 2016, .NET Core 1.0 was publicly released, marking the birth of cross-platform .NET

.NET for the masses

- What languages live in the .NET family?
  - C#, the most common .NET language. MS alternative to Java
  - F#, multi-paradigm (purely functional, imperative) language
  - L#, an implementation of Lisp atop .NET
  - C++/CLI, a C++ language extension to interface with managed code of .NET
  - IronPython, an implementation of Python atop .NET
  - PowerShell, an interactive shell and scripting language
  - etc...

May the shell be with you



- PowerShell, a strongly typed:
  - Scripting language
  - Interactive shell
- First release 2006
- It is the Windows counterpart of bash+Ruby/Perl
- Primary purpose is automation
- Runs atop .NET (Core)
- Website

May the shell be with you

- Unlike the *document-oriented* Unix shells, it adopts the *API-oriented* philosophy of Windows
  - Stream-of-chars vs. objects
  - Data structures are first-class citizens
- High-emphasis on security
  - Script execution policies, trusted vendors, signatures, etc.
  - Super-user vs. admin privilege
- Domain-specific languages
  - Introduction of keywords for DSL functionality (DSC, WWF)

May the shell be with you

- Commands are often referred to as Cmdlets
  - They are full blown script entry points with tab-completion, optional parameter validation and all that jazz
- Cmdlet names always consist of a Verb-Noun pair
  - Discoverability
  - Intuitive

```
PowerShell
Copyright (C) 2016 Microsoft Corporation. All rights reserved.

Loading personal and system profiles took 1295ms.
PS C:\Users\Matty>
```

May the shell be with you

```
PS C:\Users\Matty> Get-Verb

Verb        Group
----        -----
Add         Common
Clear       Common
Close       Common
Copy        Common
Enter       Common
Exit        Common
Find        Common
Format      Common
Get         Common
Hide        Common
Join        Common
Lock        Common
Join        Common
Lock        Common
Move        Common
```

May the shell be with you

```
PS C:\Users\Matty> Get-Command -Noun Package

CommandType     Name                                   Version    Source
-----------     ----                                   -------    ------
Cmdlet          Find-Package                           1.0.0.1    PackageManagement
Cmdlet          Get-Package                            1.0.0.1    PackageManagement
Cmdlet          Install-Package                        1.0.0.1    PackageManagement
Cmdlet          Save-Package                           1.0.0.1    PackageManagement
Cmdlet          Uninstall-Package                      1.0.0.1    PackageManagement
```

May the shell be with you

```
PS C:\Users\Matty> Get-Process

Handles  NPM(K)    PM(K)      WS(K)     CPU(s)     Id  SI ProcessName
-------  ------    -----      -----     ------     --  -- -----------
      0      17     6444      22928       0,97  13572   4 ApplicationFrameHost
      0       9     1548       7684              9972   4 atieclxx
      0      12     2812      10788             12392   4 atieclxx
      0       7     1392       2072              1480   0 atiesrxx
      0      13     8188      15168       1,38  16572   0 audiodg
      0      17     4220      17860       0,45  10356   4 browser_broker
      0      11     1872       3184              2620   0 BTDevMgr
      0      33     4528      15208      13,17    972   4 BTServer
      0      10     1796       7796       0,27  13276   4 CAudioFilterAgent64
      0       5      772        652              2612   0 CodeXLDriversLoadService-x64
      0       7     1148       4976              9416   0 conhost
      0      14     6488      17128       7,75  11124   4 conhost
      0      15     1452       2812               524   0 csrss
      0      15     2032       5940             14688   4 csrss
...
```

May the shell be with you

```
PS C:\Users\Matty> Get-Process I Sort-Object -Descending

Handles  NPM(K)    PM(K)      WS(K)     CPU(s)     Id  SI ProcessName
-------  ------    -----      -----     ------     --  -- -----------
      0      17     6444      22928       0,97  13572   4 ApplicationFrameHost
      0       9     1548       7684              9972   4 atieclxx
      0      12     2812      10788             12392   4 atieclxx
      0       7     1392       2072              1480   0 atiesrxx
      0      17     4220      17860       0,45  10356   4 browser_broker
      0      11     1872       3184              2620   0 BTDevMgr
      0      33     4528      15208      13,17    972   4 BTServer
      0      10     1796       7796       0,27  13276   4 CAudioFilterAgent64
      0       5      772        652              2612   0 CodeXLDriversLoadService-x64
      0       7     1148       4976              9416   0 conhost
      0      14     6488      17176       8,83  11124   4 conhost
      0      15     2044       6288             14688   4 csrss
      0      15     1452       2816               524   0 csrss
      0      21     6800      10828              1800   0 dasHost
...
```

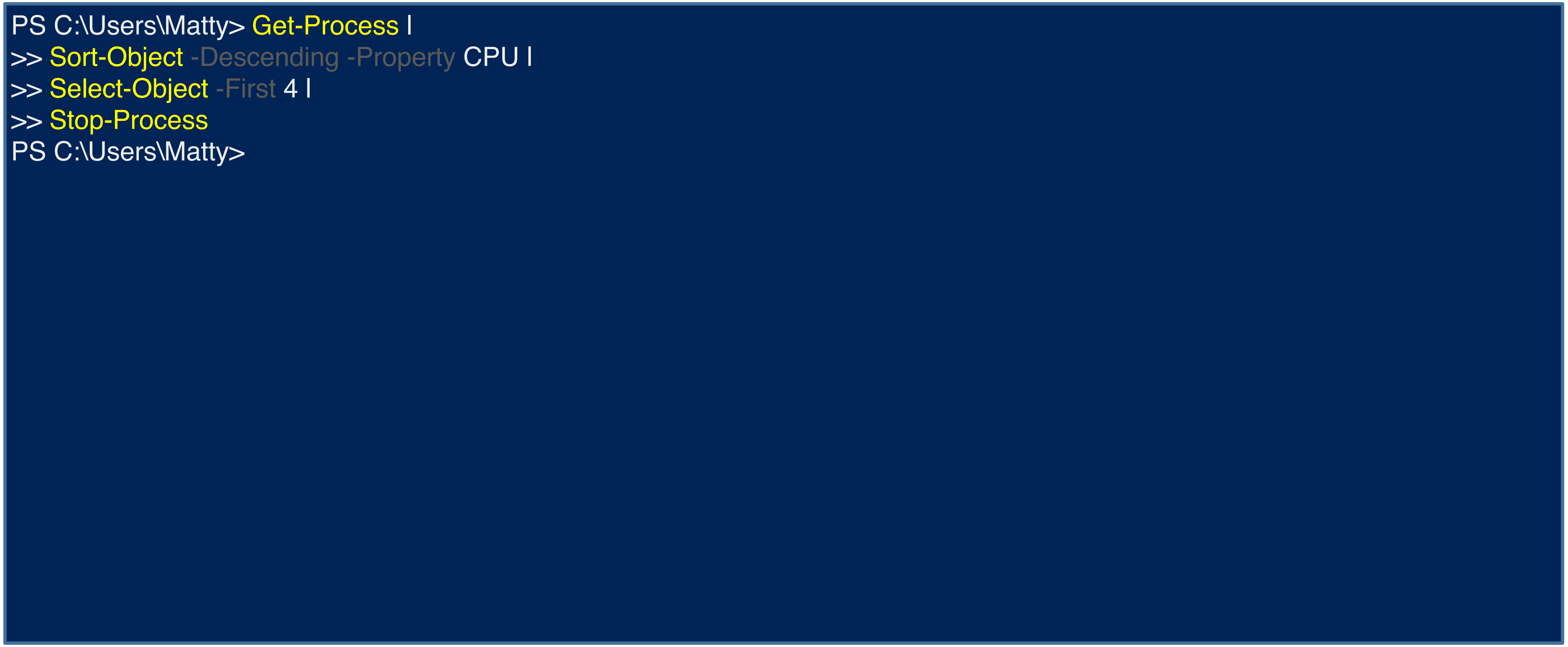
May the shell be with you

```
PS C:\Users\Matty> Get-Process | Sort-Object -Descending -Property CPU

Handles  NPM(K)    PM(K)      WS(K)     CPU(s)     Id  SI ProcessName
-------  ------    -----      -----     ------     --  -- -----------
      0      97   168252     226508     206,80  10044   4 POWERPNT
      0     214   290184     284508      69,20  12700   4 MicrosoftEdgeCP
      0      41    87284     104884      50,75   9936   4 OneDrive
      0      17     9664      21900      35,28  10484   4 ETDCtrl
      0      55    37640       1328      32,61  10004   4 SkypeHost
      0      42    21168      50096      25,58  11316   4 svchost
      0      76    90176     122152      19,22  10152   4 explorer
      0      33     4528      15208      13,17    972   4 BTServer
      0      14     6488      17208      11,98  11124   4 conhost
      0      64    44568      72616      12,06   2304   4 powershell
      0      59    36768      87320      10,53   8364   4 MicrosoftEdge
      0      27    18200      38872       8,97   9984   4 RuntimeBroker
      0      16     6228      23176       6,86   8652   4 sihost
      0      33    25132      43712       6,17  14084   4 Lenovo.Modern.ImController.PluginHost
...
```

May the shell be with you

```
PS C:\Users\Matty> Get-Process | Sort-Object -Descending -Property CPU | Select-Object -First 4

Handles  NPM(K)    PM(K)      WS(K)     CPU(s)     Id  SI ProcessName
-------  ------    -----      -----     ------     --  -- -----------
      0      96   192716     250664     267,80  10044   4 POWERPNT
      0     214   290088     284428      69,44  12700   4 MicrosoftEdgeCP
      0      41    87284     104884      50,75   9936   4 OneDrive
      0      17     9912      22148      38,31  10484   4 ETDCtrl
```

May the shell be with you



```
PS C:\Users\Matty> Get-Process |
>> Sort-Object -Descending -Property CPU |
>> Select-Object -First 4 |
>> Stop-Process
PS C:\Users\Matty>
```

May the shell be with you

```
PS C:\Users\Matty> Get-Process |
>> Sort-Object -Descending -Property CPU |
>> Select-Object -First 4 |
>> Stop-Process
PS C:\Users\Matty> ps | sort -Desc -Prop CPU | select -First 4 | kill
```

May the shell be with you

```
PS C:\Users\Matty> Get-Process |
>> Sort-Object -Descending -Property CPU |
>> Select-Object -First 4 |
>> Stop-Process
PS C:\Users\Matty> ps | sort -Desc -Prop CPU | select -First 4 | kill
PS C:\Users\Matty>
PS C:\Users\Matty> ps | sort -D -P CPU | select -F 4 | kill
```

May the shell be with you

- Okay, I get, the pipe is awesome with objects, so?
- Data structures are first class citizens too. Hm?
  - Arrays, dictionaries are distinct types with meaningful member functions
  - When data presents itself in any of these natural structures, it manifests in the API
    - When functions return multiple objects, it is usually an array of structs
    - When data is organized into a tree structure, it is usually presented as a PSDrive
      - File systems, Windows registry, environmental variables, etc.

May the shell be with you

```
PS C:\Users\Matty> $dirs = ls
PS C:\Users\Matty> $dirs.GetType()

IsPublic IsSerial Name                                     BaseType
-------- -------- ----                                     --------
True     True     Object[]                                 System.Array


PS C:\Users\Matty> $count = $dirs.Length
PS C:\Users\Matty> $count.GetType()

IsPublic IsSerial Name                                     BaseType
-------- -------- ----                                     --------
True     False    Int32                                    System.ValueType


PS C:\Users\Matty> $count
27
PS C:\Users\Matty>
```

May the shell be with you

```
PS C:\Users\Matty> cd env:
PS Env:\> ls

Name                              Value
----                          -----
ALLUSERSPROFILE              C:\ProgramData
AMDAPPSDKROOT                C:\Kellekek\AMD APP SDK\3.0\
APPDATA                      C:\Users\Matty\AppData\Roaming
ChocolateyPath               C:\Chocolatey
...

PS Env:\> (ls).Length
50
PS Env:\> Test-Path .\USERNAME
True
PS Env:\> Test-Path .\BOGUS
False
```

Windows Subsystem for Linux

- Devs want Linux-like developer experience
- NT microkernel was originally intended to support multiple OS
- Microsoft teamed up with Canonical to create WSL
- It is the Ubuntu user-space ‚syscalls'
- Much like Wine, but implemented inside the kernel, not user-space
- Linux ELF binaries running on the NT kernel!

Windows Subsystem for Linux

```
PowerShell
Copyright (C) 2016 Microsoft Corporation. All rights reserved.

Loading personal and system profiles took 2003ms.
PS C:\Users\Matty> bash.exe
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

mnagy@MATTY-Z50-75:/mnt/c/Users/Matty$ cd
mnagy@MATTY-Z50-75:~$ cat /etc/issue
Ubuntu 16.04.1 LTS \n \l

mnagy@MATTY-Z50-75:~$ which cat
/bin/cat
mnagy@MATTY-Z50-75:~$ ldd /bin/cat
        linux-vdso.so.1 =>  (0x00007fffde3a6000)
        libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f06be7f0000)
        /lib64/ld-linux-x86-64.so.2 (0x00007f06bec00000)
```

Windows Subsystem for Linux

```
PowerShell
Copyright (C) 2016 Microsoft Corporation. All rights reserved.

Loading personal and system profiles took 2003ms.
PS C:\Users\Matty> bash.exe
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

mnagy@MATTY-Z50-75:/mnt/c/Users/Matty$ cd
mnagy@MATTY-Z50-75:~$ cat /etc/issue
Ubuntu 16.04.1 LTS \n \l

mnagy@MATTY-Z50-75:~$ which cat
/bin/cat
mnagy@MATTY-Z50-75:~$ ldd /bin/cat
        linux-vdso.so.1 =>  (0x00007fffde3a6000)
        libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f06be7f0000)
        /lib64/ld-linux-x86-64.so.2 (0x00007f06bec00000)
```

Windows Subsystem for Linux

- sudo apt install package-from-ubuntu-repo
- It can
  - Run not just command-line apps
  - GUI ones like Firefox! (SSH X-forwarding to Windows)
  - It actually builds* the AliRoot software suite!
- One can
  - Run Linux applications locally, on a Windows machine
  - Develop cross-platform apps in Windows, test Linux conformance locally

Who doesn't like chocolate?

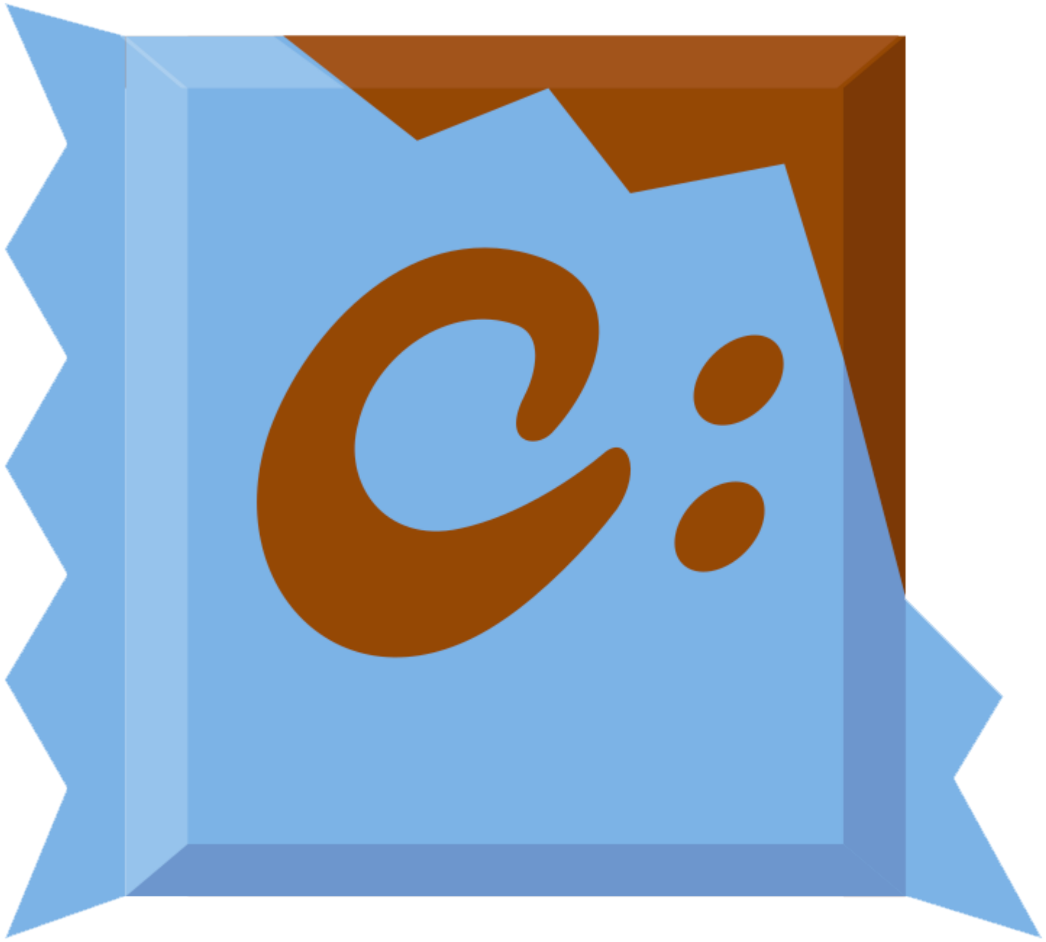
- Installing software from official/3rd party repositories is a commonplace in the Linux world since... like forever.
- On Windows, one generally has to obtain installers from each and every vendor they install software from.
- Each vendor runs a service of their own to detect out-of-date installs and manage updates
  - Waste of resources (human and machine)
- But this is all over!

Who doesn't like chocolate?

Chocolatey is a package manager for Windows (like apt-get or yum but for Windows).

Who doesn't like chocolate?

- Package manager with Windows in mind
- It is implemented in PowerShell
- Community driven repository
  - Open for contribution
- Most common applications can be easily installed with it
- Packages may be MSI, Zip, NuGet, etc.
- https://chocolatey.org/packages



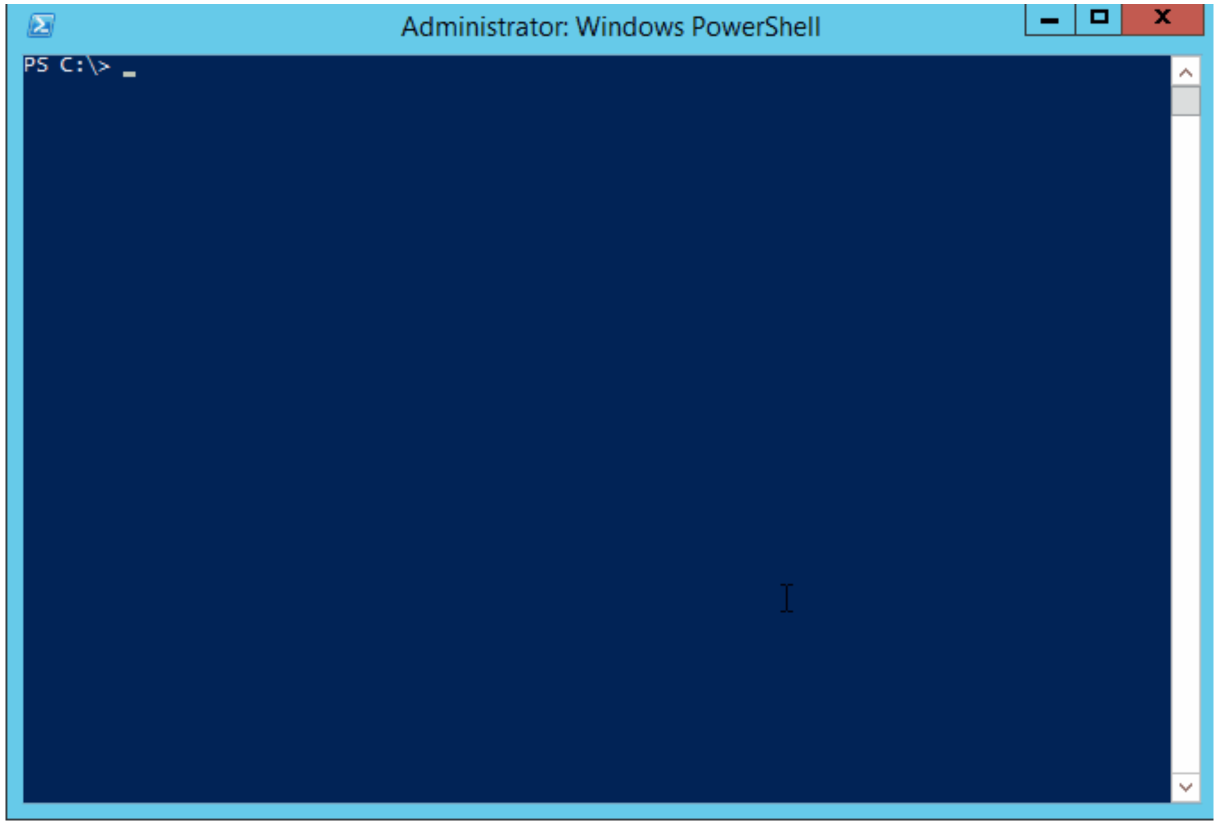
Who doesn’t like chocolate?

```
PS C:\Users\Matty> iwr https://chocolatey.org/install.ps1 | iex
```

- Installation cannot be much simpler
  - In an elevated command prompt invoke the given one-liner
  - What does the script do?
    - iwr is an alias to Invoke-WebRequest (wget is the same alias)
    - The result is passed on the pipe to Invoke-Expression
- Off-topic: how to save the script to disk?
  - Instead of invoking the script, we save it to a file

```
PS C:\Users\Matty> wget https://chocolatey.org/install.ps1 | Out-File install.ps1
```

Who doesn't like chocolate?



```
PS C:\> _
```

- Simple commands
  - List
  - Search
  - Info
  - Install
  - Upgrade
  - Uninstall
  - Pin
  - New
  - Pack
  - Push

Who doesn’t like chocolate?

- Windows PowerShell 5.0: enters PackageManagement module
  - It is a package manager manager
  - Ability to register package sources and manage from a unified interface

```
PS C:\Users\Matty> Find-Package gnuplot

Name                    Version        Source          Summary
----                    -------        ------          -------
gnuplot                  4.6.6        chocolatey      Gnuplot is a portable comma...

PS C:\Users\Matty> Install-Package gnuplot
PS C:\Users\Matty> gnuplot

        G N U P L O T
        Version 5.0 patchlevel 1    last modified 2015-06-07

gnuplot>
```

Actual product may vary

Build systems, version control and other buzzwords

TYPICAL DEVELOPMENT WORKFLOW

Build systems

- This topic was throroughly investigated last year, please refer to last year's slides as well, for a complete tour.
- What is a build system?
  - A tool that takes care of building your application in the fastest way possible with minimal user effort.
  - The input is a make file, and the output is one or more binary/ies (hopefully). ☺️
  - In a broader sense, it's a workflow consisting of general purpose actions to take, for eg. invoking a compiler.

Build systems

- Didn’t I just say „Minimal user effort”?!
  - Build Systems aim at being as comfortable to use as possible
  - User declares the task, instead of specifying what to do
  - Declarative DSL, not imperative
- Didn’t I just say „Maximum throughput”?!
  - Detects the minimal portion of the program that must be recompiled when editing code. (Based on time stamps)
  - Processes independent parts of the build tasks in parallel
- Requires learning, but pays off in the long run!

CMake: Cross-platform make

- Make file generator
- Portable
- Open-source
- Knows most languages by default
- The known ones are EASY to use
- Others can be taught
- DSL script language sometimes unfriendly
- Most cross-platform projects use it

```
PROJECT (my_app)
LIST (SOURCES)
APPEND (SOURCES main.cpp vector.cpp)
ADD_EXECUTABLE (${PROJECT_NAME} SOURCES)
```

CMake+CTest+CPack = EXIT_SUCCESS

- Kitware is the company behind the CMake suite of tools
- Full-fledged scripting language to do virtually anything
  - It is documented
  - Gazillions of tutorials online
- Big projects using CMake suite of tools
  - Bullet Physics Engine, CLion, Compiz, cURL, ROOT, GEANT4, GROMACS, KDE, libPNG, LAPACK, LLVM, Clang, MySQL, OGRE, OpenCV, SFML, zlib, …

Portability

- Portability is important!
  - Today, you might write the code for yourself, but tomorrow you might have to give it to a collegue
  - If your code is bound to a specific OS, compiler, etc. They will be more reluctant to use your code
- Dependencies
  - The portability of code is the union of restrictions imposed by:
    - Tools required to build the application
    - Environment required to run the application
  - Prefer portable tools over non-portable (have good reason to defect)
  - Understand the costs of depending upon external software (even OSS)

Research project

- Physics library

Research project

- Physics library
  - src
    - Phys stuff
    - More

Top-level CMakelists.txt

```
# The supremum of version requirements of the script imposed by features used
cmake_minimum_required (VERSION 2.8.11)

# CMakeLists files in this project can
# refer to the root source directory of the project as ${RESEARCH_SOURCE_DIR}
# and to the root binary directory of the project as ${RESEARCH_BINARY_DIR}.
project (RESEARCH)

# Recurse into the „phys" and „app" subdirectories. This does not actually
# cause another cmake executable to run. The same process will walk through
# the project's entire directory structure.
add_subdirectory (phys)
add_subdirectory (app)
```

Library CMakelists.txt

```
cmake_minimum_required (VERSION 2.8.11)

# Create a library called „Phys" which includes the source files „stuff.cpp” and „more.cpp”.
# The extension is already found. Any number of sources could be listed here.
add_library (Phys src/stuff.cpp src/more.cpp)

# Make sure the compiler can find include files for our Phys library
# when other libraries or executables link to Phys
target_include_directories (Phys PUBLIC ${CMAKE_CURRENT_SOURCE_DIR}/inc)
```

Library CMakelists.txt

```
cmake_minimum_required (VERSION 2.8.11)

# Add executable called „Application" that is built from the source files
# „main.cpp”. The extensions are automatically found.
add_executable (Application src/main.cpp)

# Make sure the compiler can find include files for our Application sources target_include_directories (Application PUBLIC $
{CMAKE_CURRENT_SOURCE_DIR}/inc)

# Link the executable to the Phys library. Since the Phys library has
# public include directories we will use those link directories when building
# Application
target_link_libraries (Application LINK_PUBLIC Phys)
```

Install CMake: Ubuntu

```
mnagy@MATTY-Z50-75:~$ sudo apt install cmake
Reading package lists... Done
Building dependency tree
Reading state information... Done
Suggested packages:
  codeblocks eclipse ninja-build
The following NEW packages will be installed:
  cmake
0 upgraded, 1 newly installed, 0 to remove and 22 not upgraded.
Need to get 0 B/2,623 kB of archives.
After this operation, 14.6 MB of additional disk space will be used.
Selecting previously unselected package cmake.
(Reading database ... 86023 files and directories currently installed.)
Preparing to unpack .../cmake_3.5.1-1ubuntu3_amd64.deb ...
Unpacking cmake (3.5.1-1ubuntu3) ...
Processing triggers for man-db (2.7.5-1) ...
Setting up cmake (3.5.1-1ubuntu3) ...
mnagy@MATTY-Z50-75:~$
```

Install CMake: Windows

- Using Chocolatey via

```
PS C:\Users\Matty> choco install cmake
```

- Download the installer from [Kitware]
- When ready, using the package management framework of PowerShell

„Fear leads to anger, anger leads to hate, and hate leads to… suffffferiiiiing.” – Master Yoda

FEAR OF THE UNKNOWN

Fear of the unknown

- What exactly is an IDE?
  - It is a set of tools related to various aspects of software development that work in synergy to maximize productivity.
- What are these aspects?
  - Source editing, compiling, build automating, debugging, version controling, unit testing, benchmarking / profiling, packaging / distributing...
- Heck, but I'm a physicist, not a software engineer
  - Exactly! That is why you should care!

Fear of the unknown

Integrated Development Environment

- Pro
  - End-to-end automation
  - Workflow is natural
  - Easy to learn, „hard” to master
- Con
  - Gotta cook with what you got
    - The choice of IDE becomes important
    - Extensibility is important

Toolchain

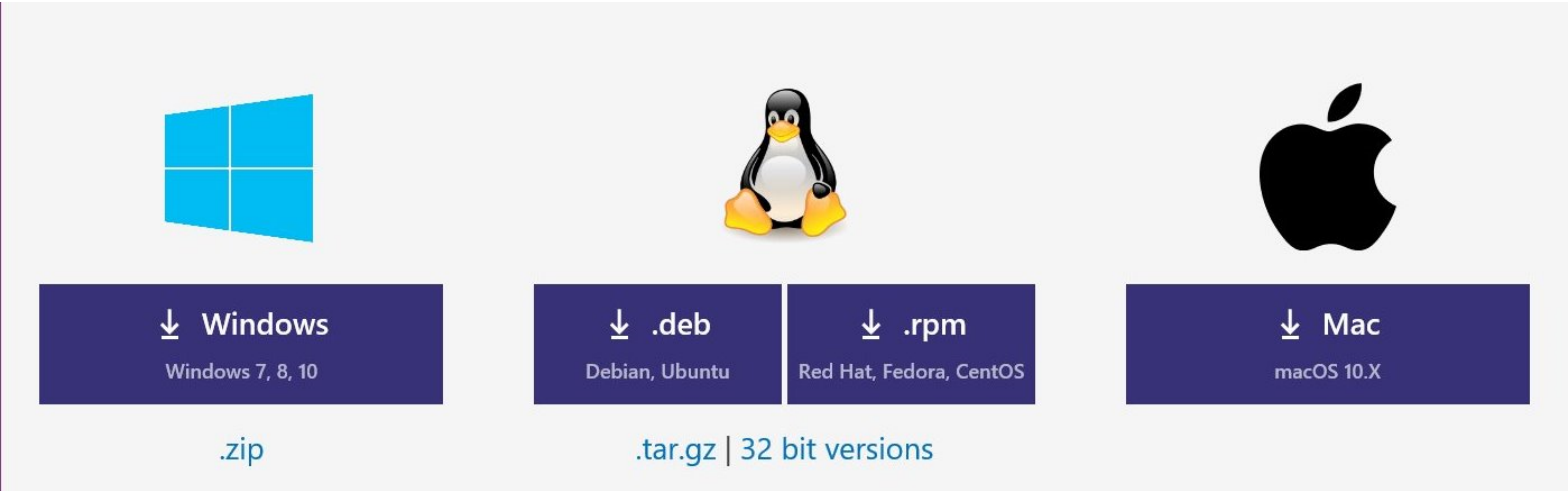
- Con
  - Distinct tools for everything
    - One needs to interface them
  - Some glitches remain
  - Hard to learn, hard to master
    - Get used to workflow (scripting?)
- Con
  - Choose the best of everything!

IDE for the powerusers

- Visual Studio Code is young
  - Initial release: April 29, 2015
  - Open-source: November 18, 2015
- It is cross-platform
  - Windows, Linux, OSX
    - Built atop JavaScript and TypeScript
- Initially a code editor
  - Community pressure lead to an open plug-in system
  - Immense influx of plug-ins since

IDE for the powerusers

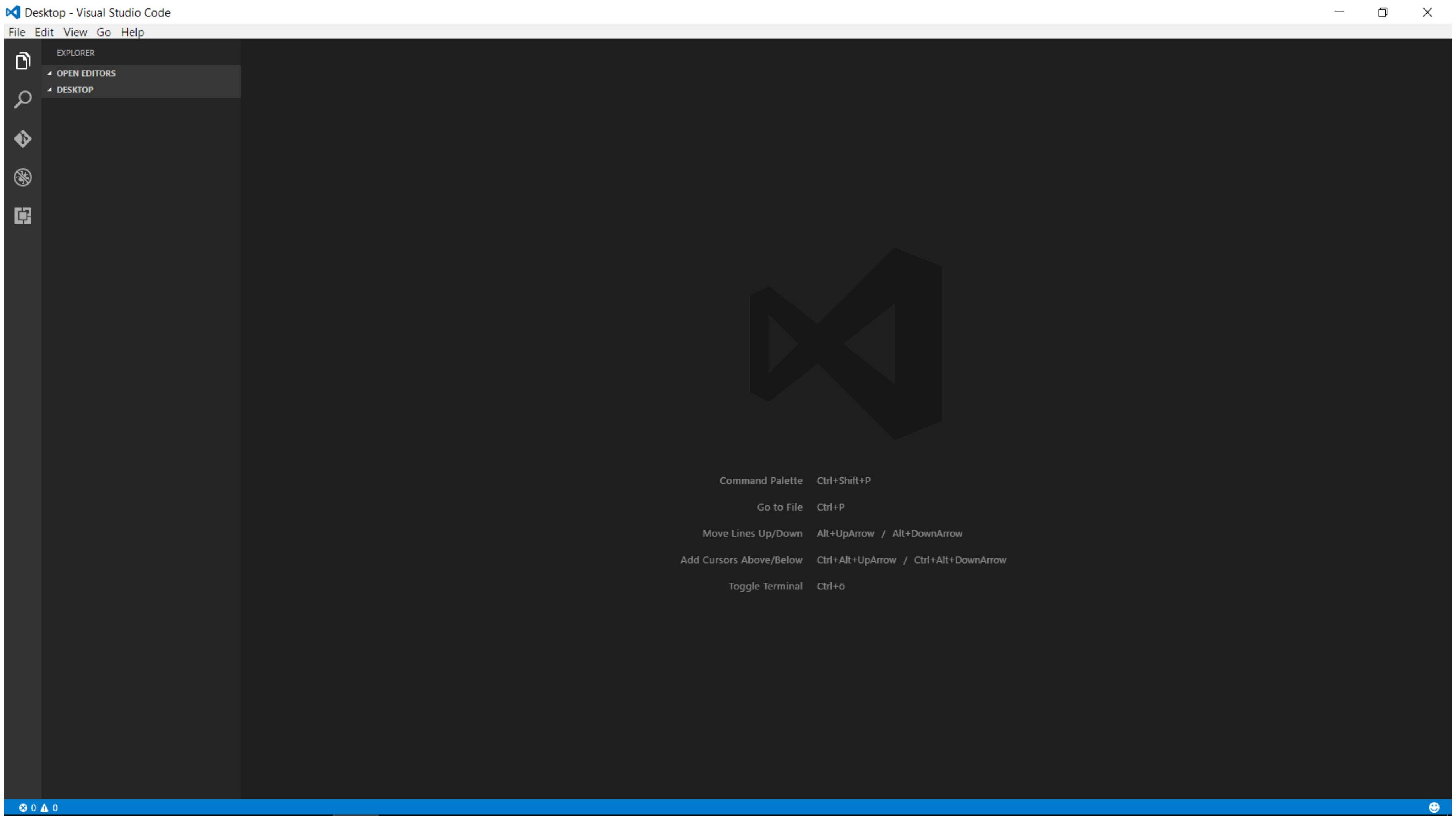
https://code.visualstudio.com



Installation is as easy as 1-2-3.

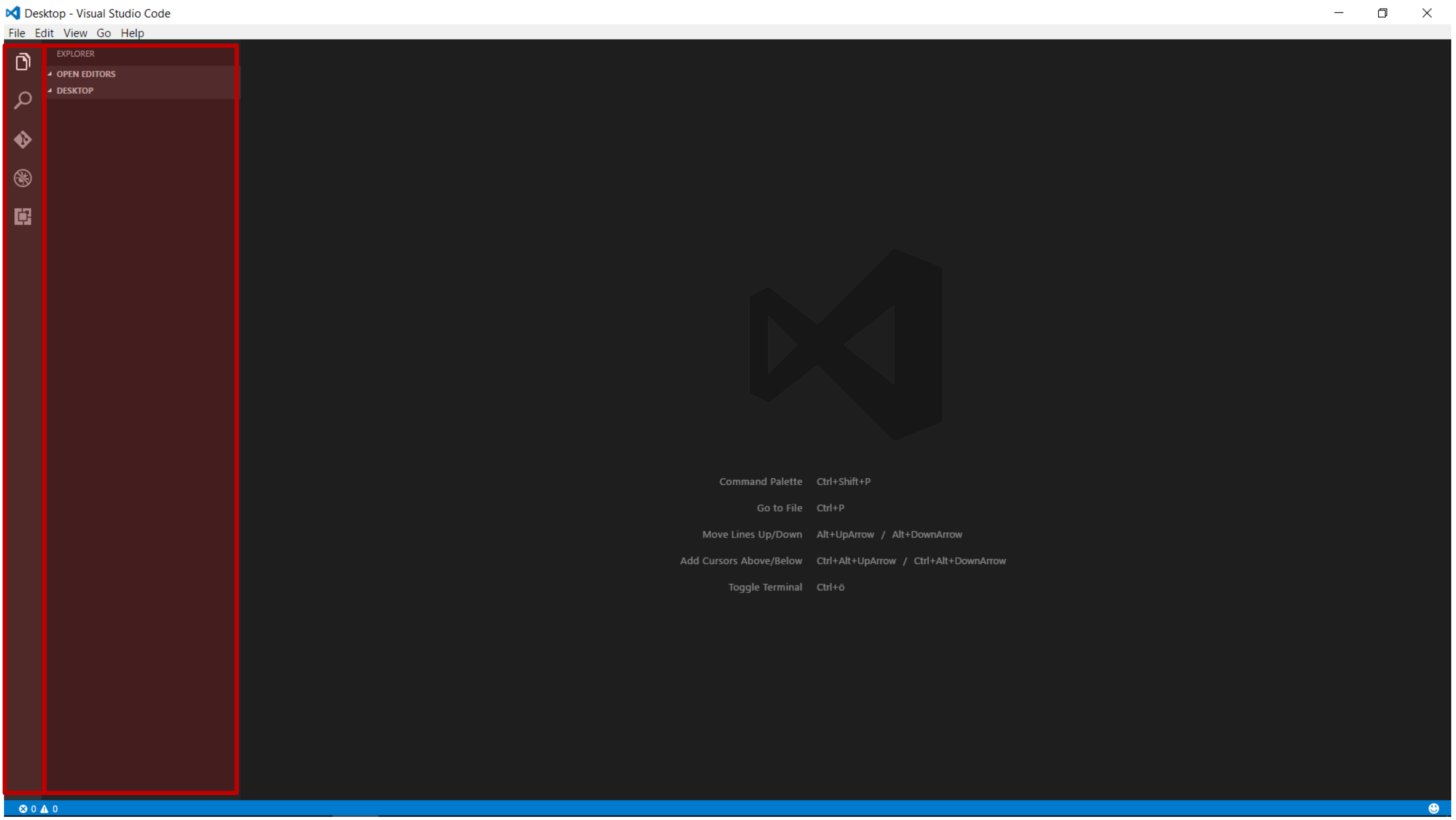
```
mnagy@MATTY-Z50-75:~$ sudo dpkg -i code_1.6.1-1476373175_amd64.deb
```

IDE for the powerusers



Command Palette Ctrl+Shift+P
Go to File Ctrl+P
Move Lines Up/Down Alt+UpArrow / Alt+DownArrow
Add Cursors Above/Below Ctrl+Alt+UpArrow / Ctrl+Alt+DownArrow
Toggle Terminal Ctrl+ö

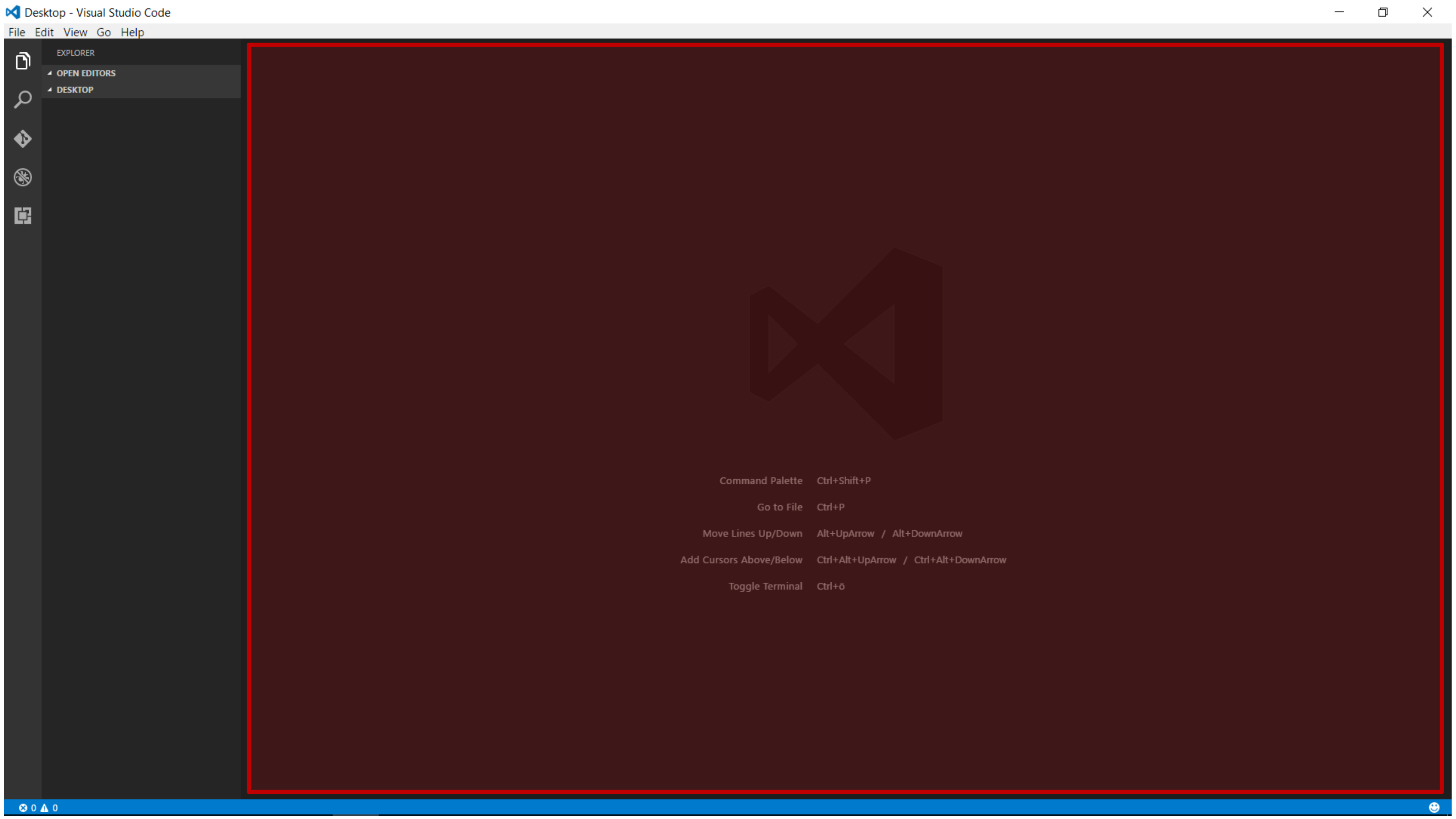
IDE for the powerusers



Desktop - Visual Studio Code
File Edit View Go Help
EXPLORER
OPEN EDITORS
DESKTOP
Command Palette Ctrl+Shift+P
Go to File Ctrl+P
Move Lines Up/Down Alt+UpArrow / Alt+DownArrow
Add Cursors Above/Below Ctrl+Alt+UpArrow / Ctrl+Alt+DownArrow
Toggle Terminal Ctrl+ö

- Sidebar
- Buttons of core commands reside here
  - File explorer
  - Search
  - Git
  - Debug
  - Extensions

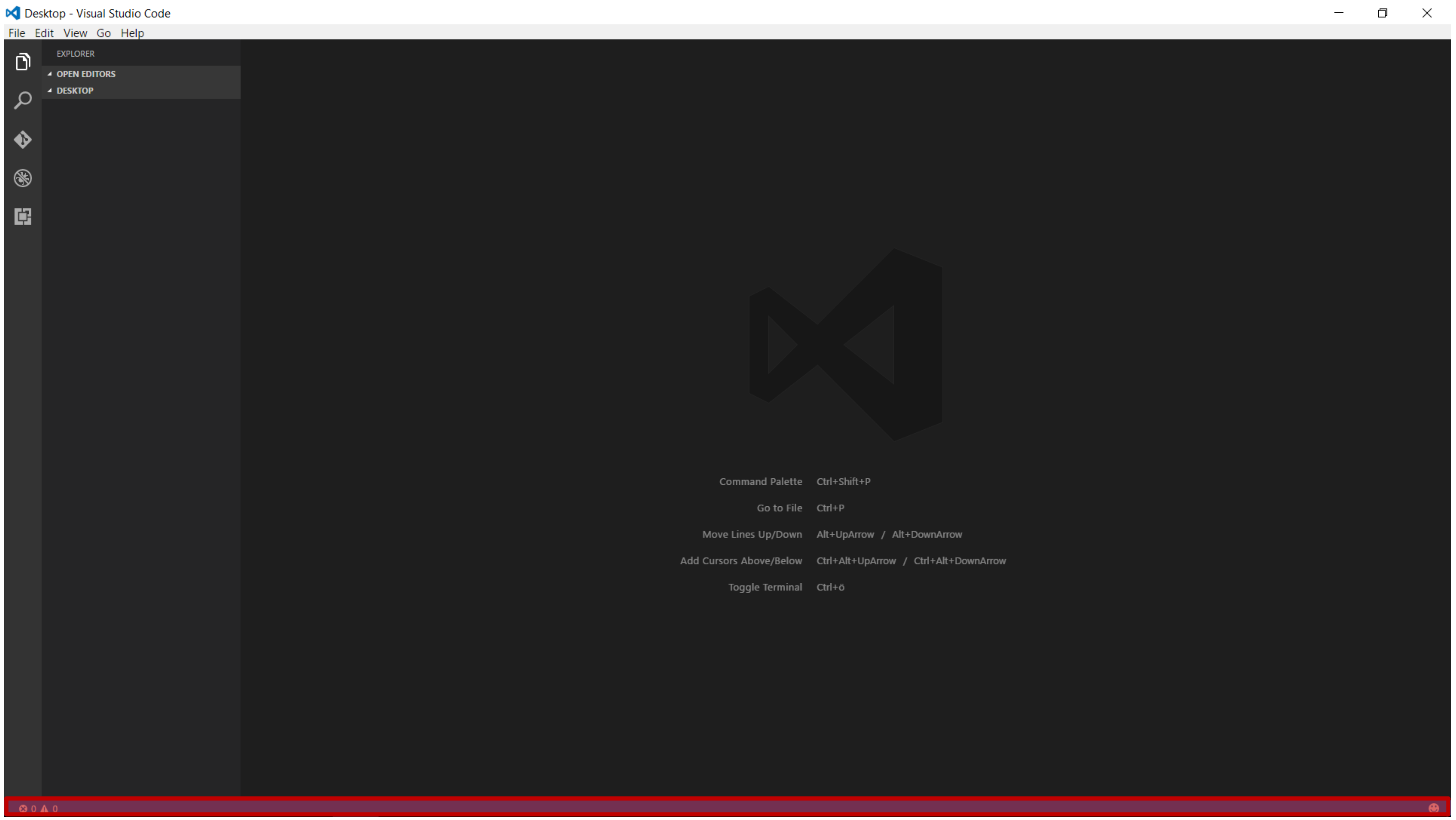
IDE for the powerusers



Command Palette Ctrl+Shift+P
Go to File Ctrl+P
Move Lines Up/Down Alt+UpArrow / Alt+DownArrow
Add Cursors Above/Below Ctrl+Alt+UpArrow / Ctrl+Alt+DownArrow
Toggle Terminal Ctrl+ö

- Editor pane
- This is where one spends most of his/her time
- Can be subdivided
  - Header/Source
  - Diff view

IDE for the powerusers



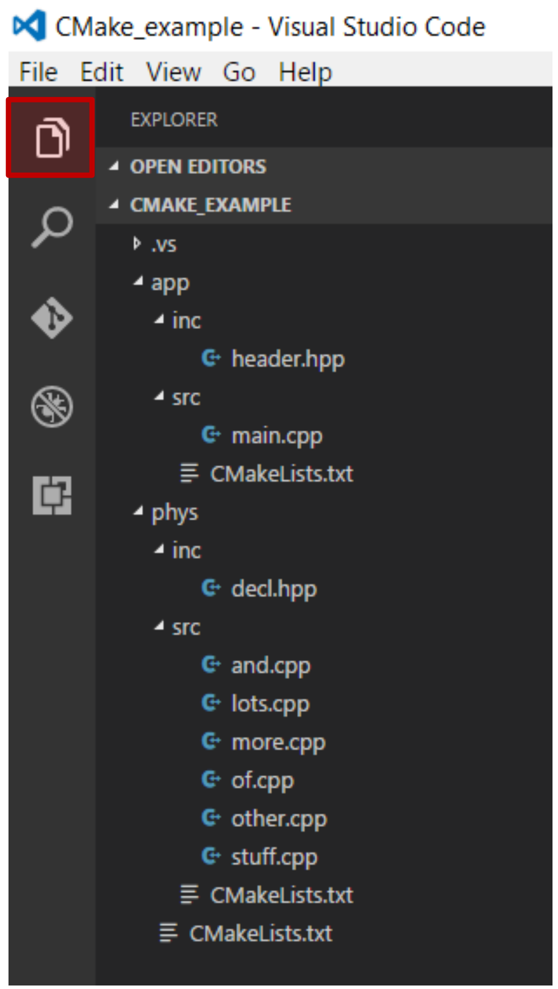
Command Palette	Ctrl+Shift+P
Go to File	Ctrl+P
Move Lines Up/Down	Alt+UpArrow / Alt+DownArrow
Add Cursors Above/Below	Ctrl+Alt+UpArrow / Ctrl+Alt+DownArrow
Toggle Terminal	Ctrl+ö

- Status bar
- Shows various informations
  - Git
  - Extensions
- Buttons may reside
  - Frequent commands

IDE for the powerusers

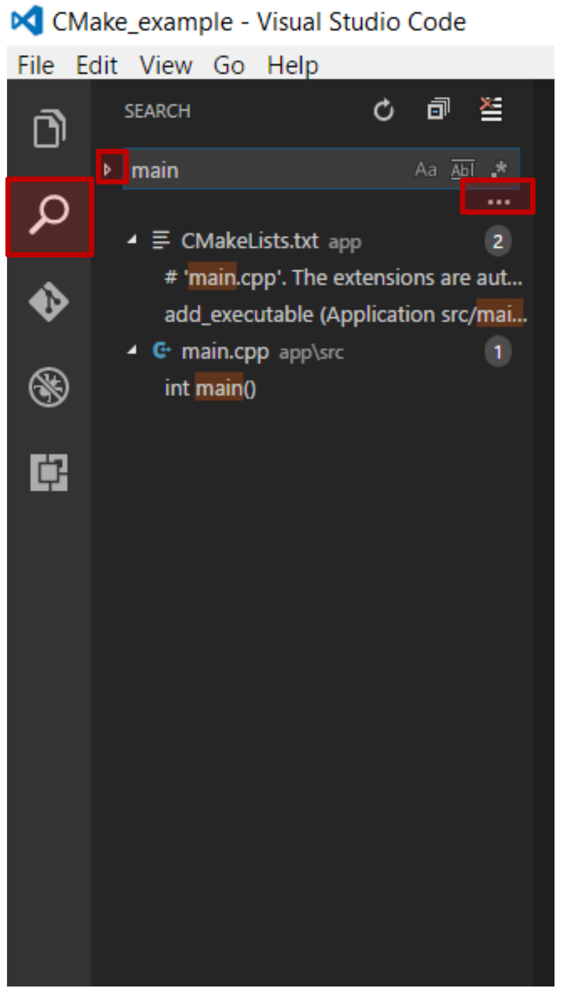
- VS Code can be used as a text editor
  - Simply pop open a file, save, exit
- VS Code can be used as a highly customizable IDE
  - Open an entire folder
  - Global settings of Code are stored in platform-specific places
  - Workspace scope settings (folders opened by Code) are stored in-place in a folder called .vs

IDE for the powerusers



- The Explorer button displays a tree structure of all files inside the folder
- For fast identification of files, icon packs can be installed in just a few clicks
  - File, Preferences, File Icon Theme, Seti
  - Track feature request of CMake icons (GitHub issue)
- Collapse/expand folders as you see fit
- Double-clicking a file opens it in the editor pane

IDE for the powerusers


CMake_example - Visual Studio Code

File Edit View Go Help

SEARCH

main

CMakeLists.txt app 2

'main.cpp'. The extensions are aut...

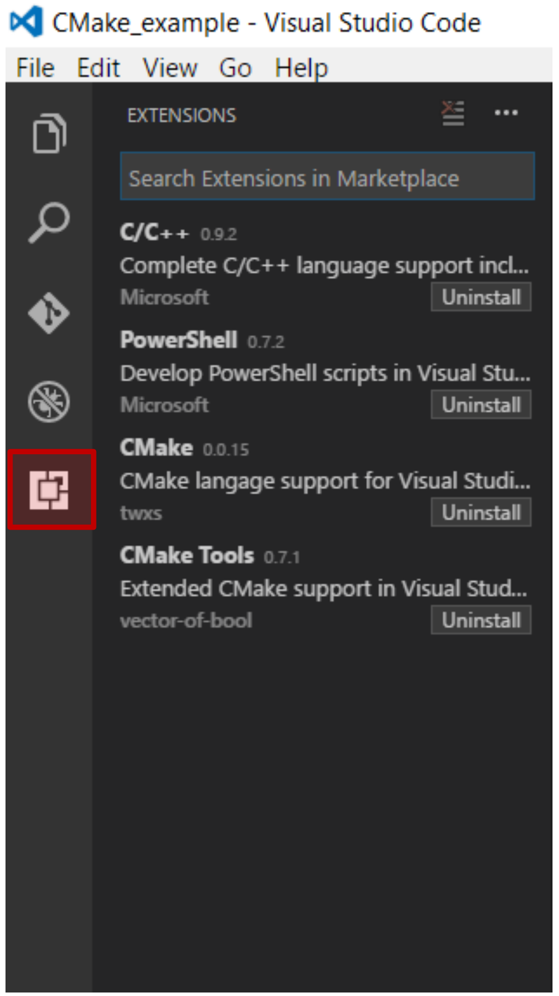
add_executable (Application src/mai...

main.cpp app\src 1

int main()

- The Search button allows to search the contents of the files by regular expressions if needed
- Enter a pharse to find all files containing it
- Find and replace capability is invoked by the collapse/ expand triangle
- The … icon under the text box displays additional search options

IDE for the powerusers


CMake_example - Visual Studio Code

File Edit View Go Help

EXTENSIONS

Search Extensions in Marketplace

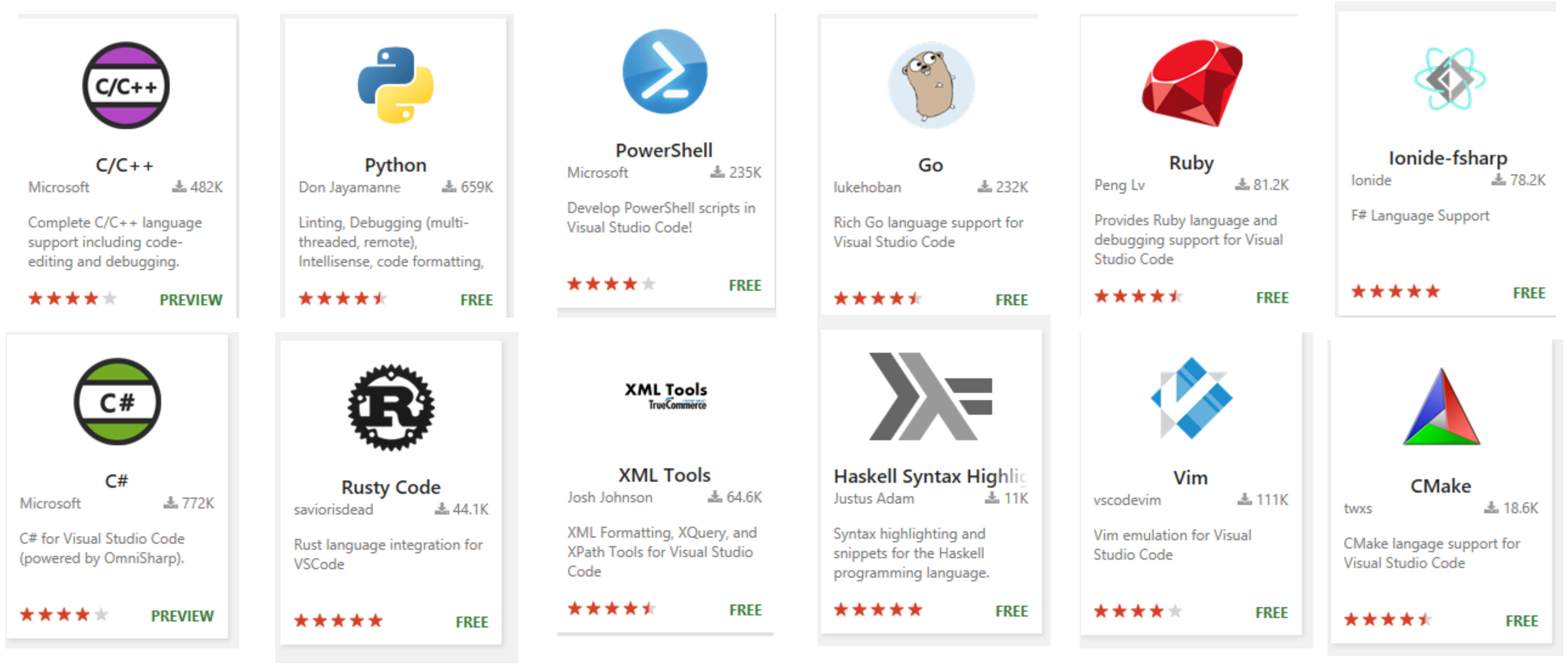
C/C++ 0.9.2
Complete C/C++ language support incl...
Microsoft — Uninstall

PowerShell 0.7.2
Develop PowerShell scripts in Visual Stu...
Microsoft — Uninstall

CMake 0.0.15
CMake langage support for Visual Studi...
twxs — Uninstall

CMake Tools 0.7.1
Extended CMake support in Visual Stud...
vector-of-bool — Uninstall

- The Extensions button let's us:
  - explore the marketplace for extensions
  - manage already installed extensions


C/C++ — Microsoft — 482K — Complete C/C++ language support including code-editing and debugging. — PREVIEW

Python — Don Jayamanne — 659K — Linting, Debugging (multi-threaded, remote), Intellisense, code formatting, — FREE

PowerShell — Microsoft — 235K — Develop PowerShell scripts in Visual Studio Code! — FREE

Go — lukehoban — 232K — Rich Go language support for Visual Studio Code — FREE

Ruby — Peng Lv — 81.2K — Provides Ruby language and debugging support for Visual Studio Code — FREE

Ionide-fsharp — Ionide — 78.2K — F# Language Support — FREE

C# — Microsoft — 772K — C# for Visual Studio Code (powered by OmniSharp). — PREVIEW

Rusty Code — saviorisdead — 44.1K — Rust language integration for VSCode — FREE

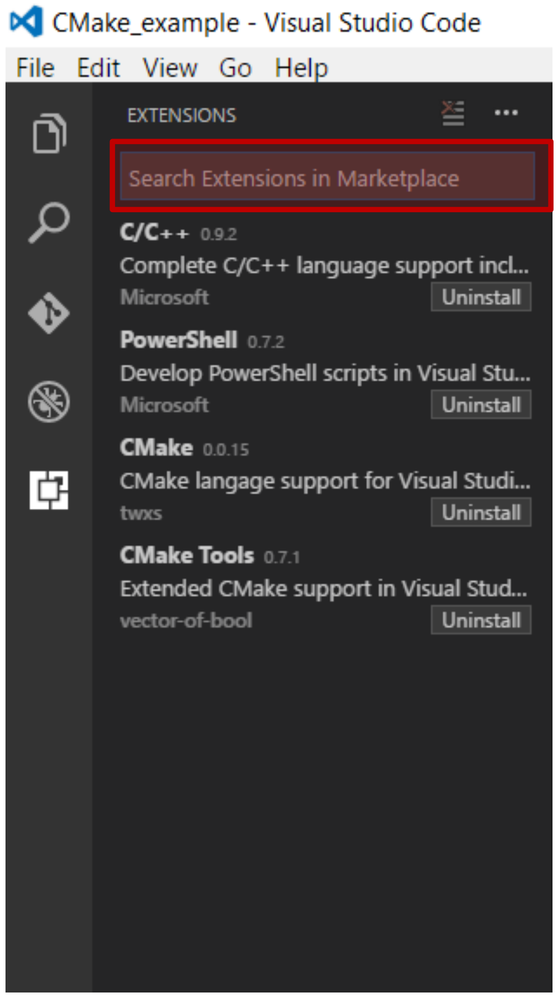
XML Tools — Josh Johnson — 64.6K — XML Formatting, XQuery, and XPath Tools for Visual Studio Code — FREE

Haskell Syntax Highlig — Justus Adam — 11K — Syntax highlighting and snippets for the Haskell programming language. — FREE

Vim — vscodevim — 111K — Vim emulation for Visual Studio Code — FREE

CMake — twxs — 18.6K — CMake langage support for Visual Studio Code — FREE

IDE for the powerusers


CMake_example - Visual Studio Code

File Edit View Go Help

EXTENSIONS

Search Extensions in Marketplace

C/C++ 0.9.2
Complete C/C++ language support incl...
Microsoft — Uninstall

PowerShell 0.7.2
Develop PowerShell scripts in Visual Stu...
Microsoft — Uninstall

CMake 0.0.15
CMake langage support for Visual Studi...
twxs — Uninstall

CMake Tools 0.7.1
Extended CMake support in Visual Stud...
vector-of-bool — Uninstall

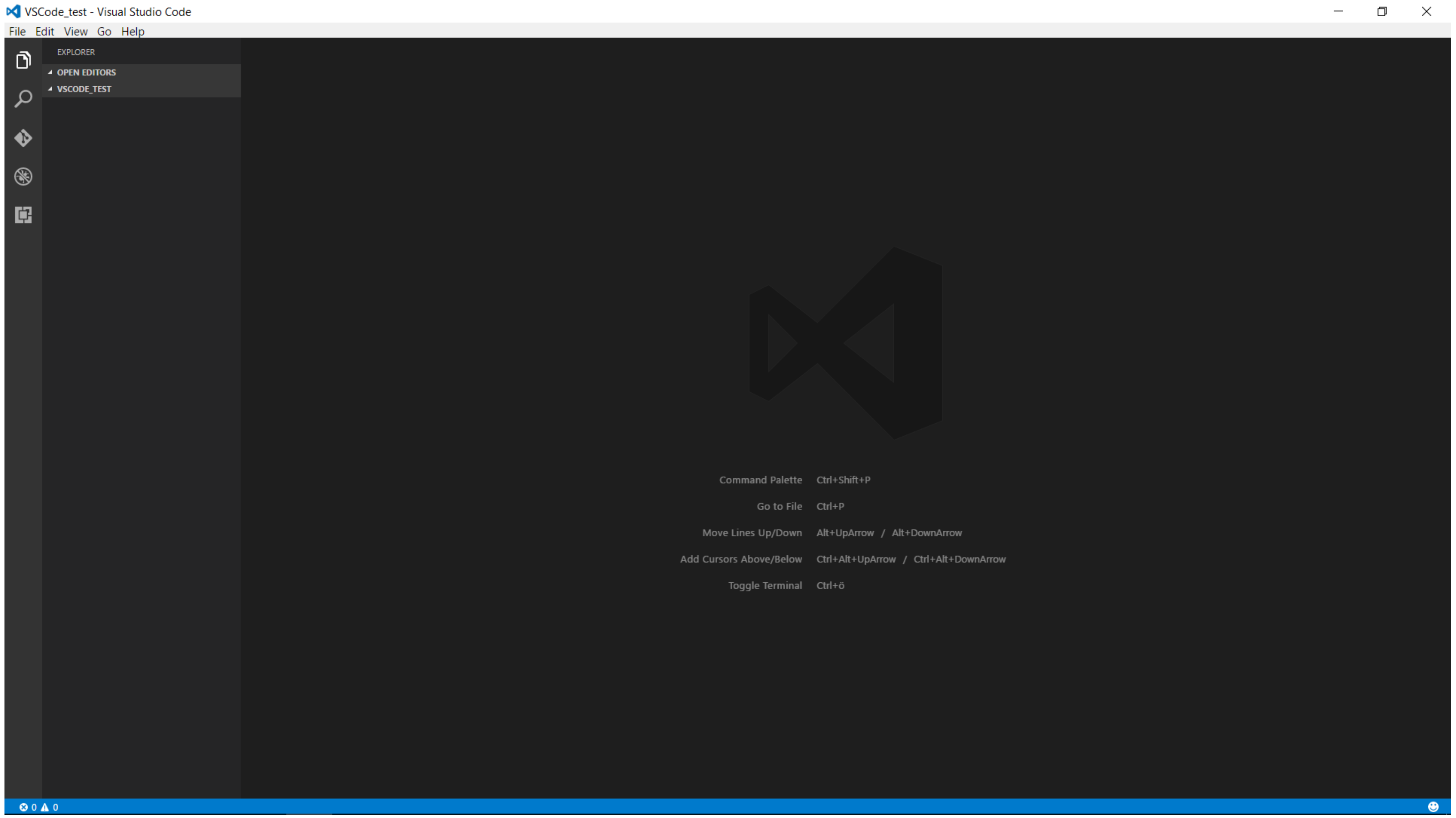
- To install an extension:
  - Click the search bar
  - Type the name of the extension
  - Click on the Install button
  - When it's done, click on the Reload button
- This tutorial will show the synergy of these:
  - C/C++
  - CMake
  - CMake Tools

IDE for the powerusers

- On Linux
  - I assume you have g++ installed via your distributions package manager and thus the compiler and linker are in your $PATH
- On Windows
  - I assume you either have
    - Visual Studio 2015 Community Edition installed
    - Standalone Visual C++ Build Tools installed
  - In case you want to use the legacy NMake build system, you must launch VS Code from a developer command prompt
    - Launch one of the shortcuts installed in your Start Menu
    - Invoke vcvarsall.bat which sets up your environment

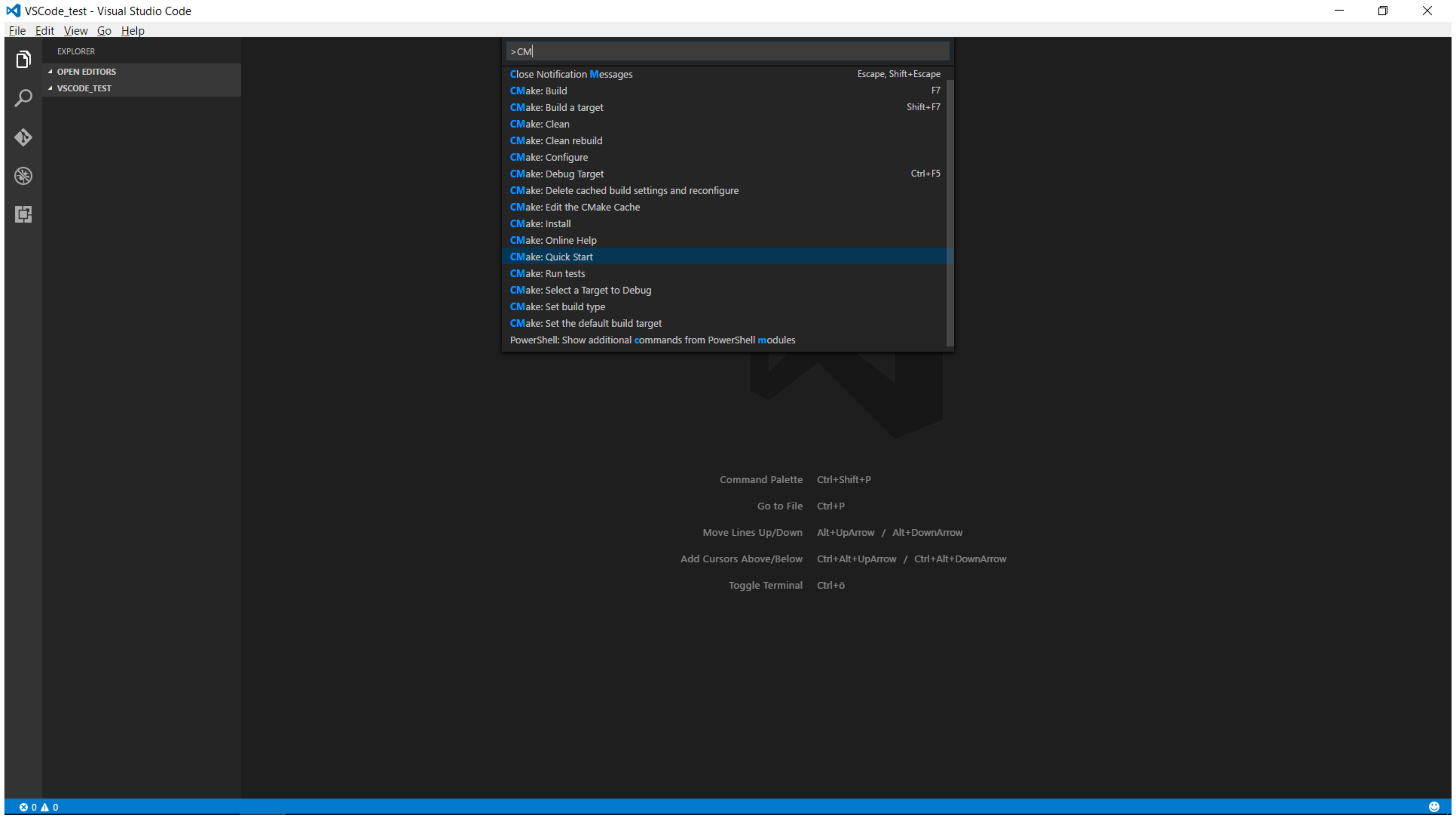
IDE for the powerusers

- Navigate to a directory where you want to start a new project
- You will be greeted with an empty workspace
- We will use the Quick Start feature of CMake Tools to initialize a simple source file and build script


VSCode_test - Visual Studio Code
File Edit View Go Help
EXPLORER
OPEN EDITORS
VSCODE_TEST
Command Palette Ctrl+Shift+P
Go to File Ctrl+P
Move Lines Up/Down Alt+UpArrow / Alt+DownArrow
Add Cursors Above/Below Ctrl+Alt+UpArrow / Ctrl+Alt+DownArrow
Toggle Terminal Ctrl+ö
0 0

IDE for the powerusers

- To open the Command Palette, either
  - Ctrl+Shift+P
  - Click View, then Show Command Palette
- This will display all the available commands from the core editor and all the extensions
- Type in „CMake”
- After, select „Quick Start”

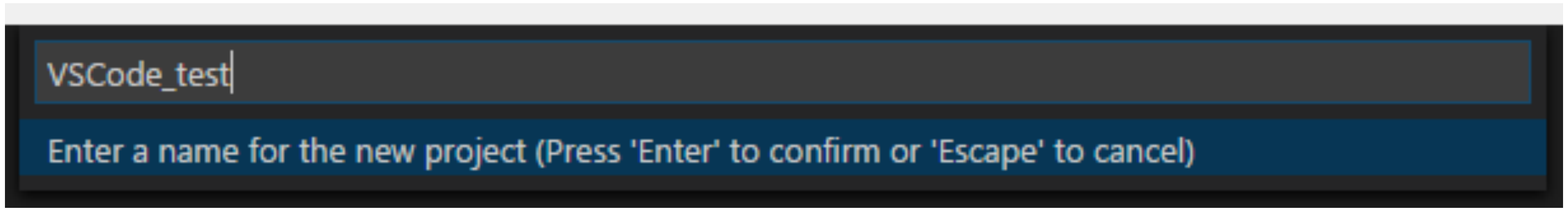


```
>CM
Close Notification Messages                          Escape, Shift+Escape
CMake: Build                                         F7
CMake: Build a target                                Shift+F7
CMake: Clean
CMake: Clean rebuild
CMake: Configure
CMake: Debug Target                                  Ctrl+F5
CMake: Delete cached build settings and reconfigure
CMake: Edit the CMake Cache
CMake: Install
CMake: Online Help
CMake: Quick Start
CMake: Run tests
CMake: Select a Target to Debug
CMake: Set build type
CMake: Set the default build target
PowerShell: Show additional commands from PowerShell modules
```

Command Palette	Ctrl+Shift+P
Go to File	Ctrl+P
Move Lines Up/Down	Alt+UpArrow / Alt+DownArrow
Add Cursors Above/Below	Ctrl+Alt+UpArrow / Ctrl+Alt+DownArrow
Toggle Terminal	Ctrl+ö

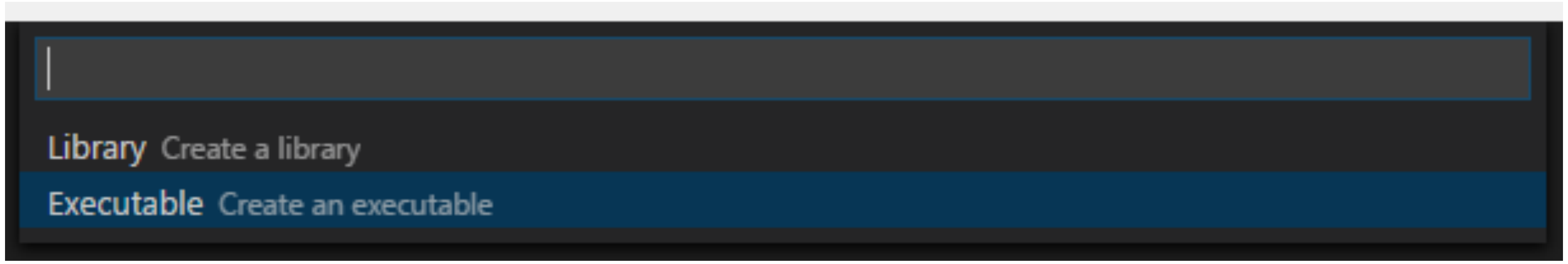
IDE for the powerusers

- Next, it is going to prompt for a project name
- VSCode_test for eg.
- After giving a project name, it is going to ask what type of project will this be
- Choose „Executable”
- Finally, it’s going to as for a default build configuration
- Choose Debug



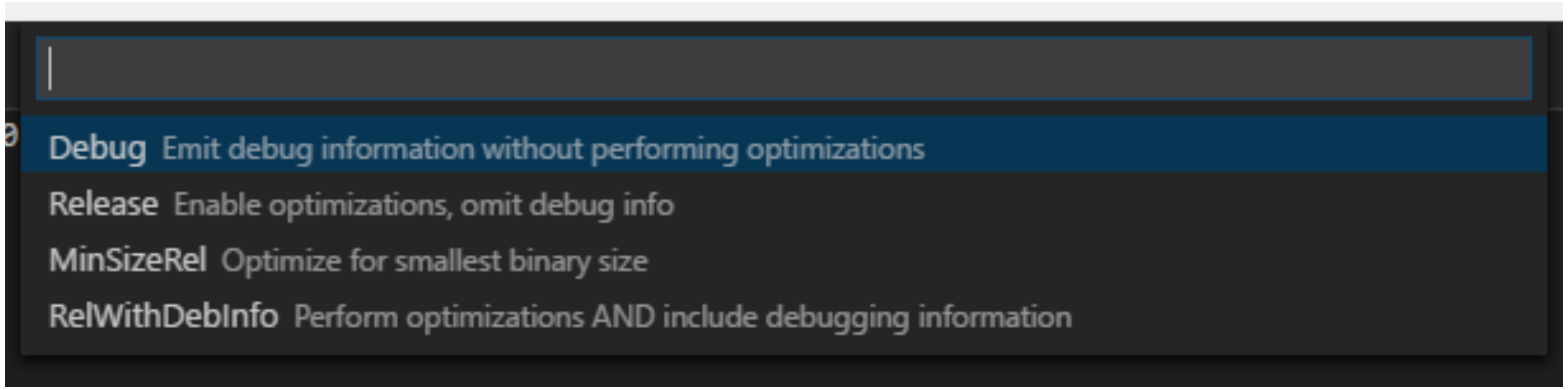
VSCode_test

Enter a name for the new project (Press 'Enter' to confirm or 'Escape' to cancel)



Library Create a library

Executable Create an executable



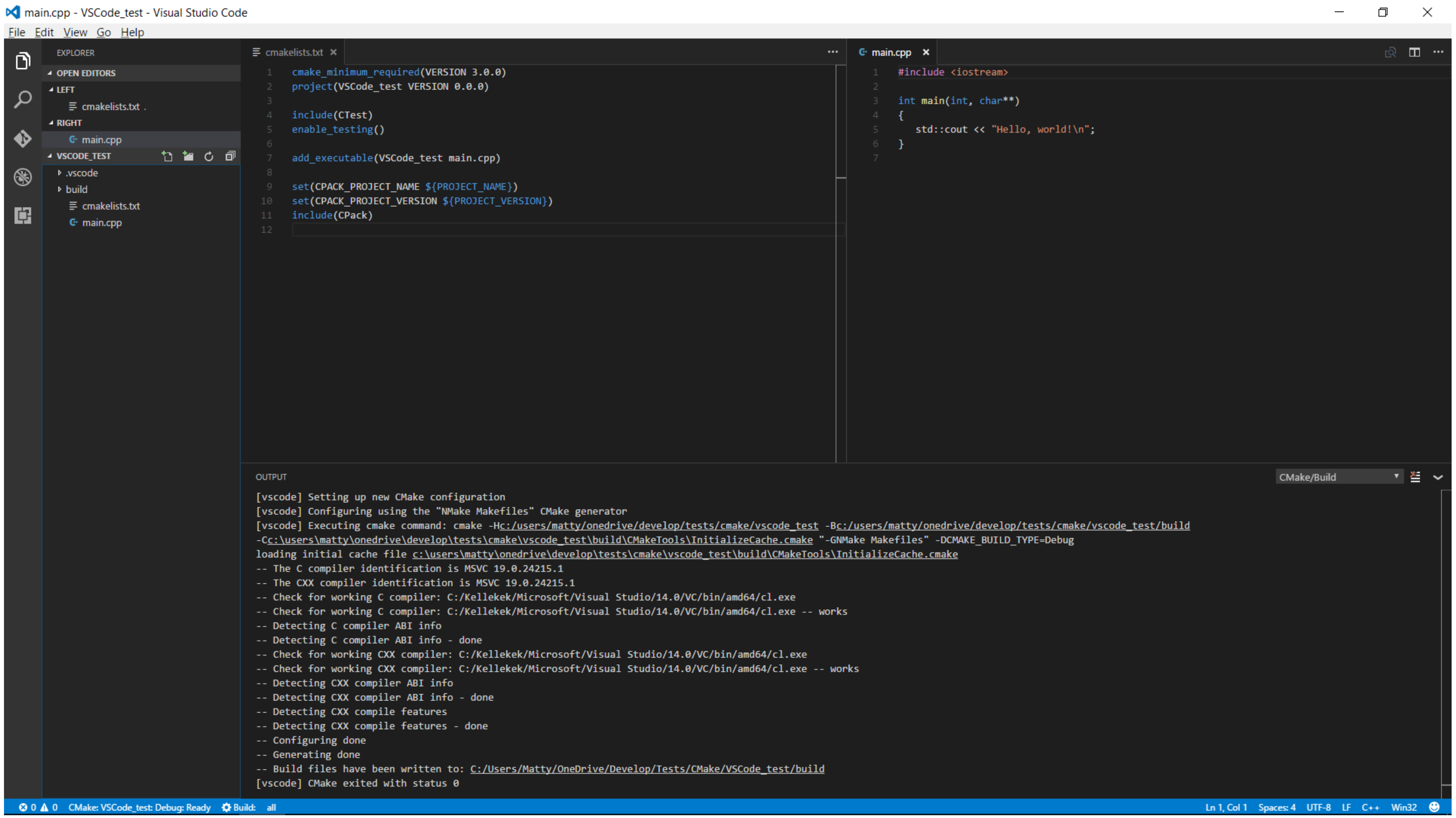
Debug Emit debug information without performing optimizations

Release Enable optimizations, omit debug info

MinSizeRel Optimize for smallest binary size

RelWithDebInfo Perform optimizations AND include debugging information

IDE for the powerusers



```
cmake_minimum_required(VERSION 3.0.0)
project(VSCode_test VERSION 0.0.0)

include(CTest)
enable_testing()

add_executable(VSCode_test main.cpp)

set(CPACK_PROJECT_NAME ${PROJECT_NAME})
set(CPACK_PROJECT_VERSION ${PROJECT_VERSION})
include(CPack)
```

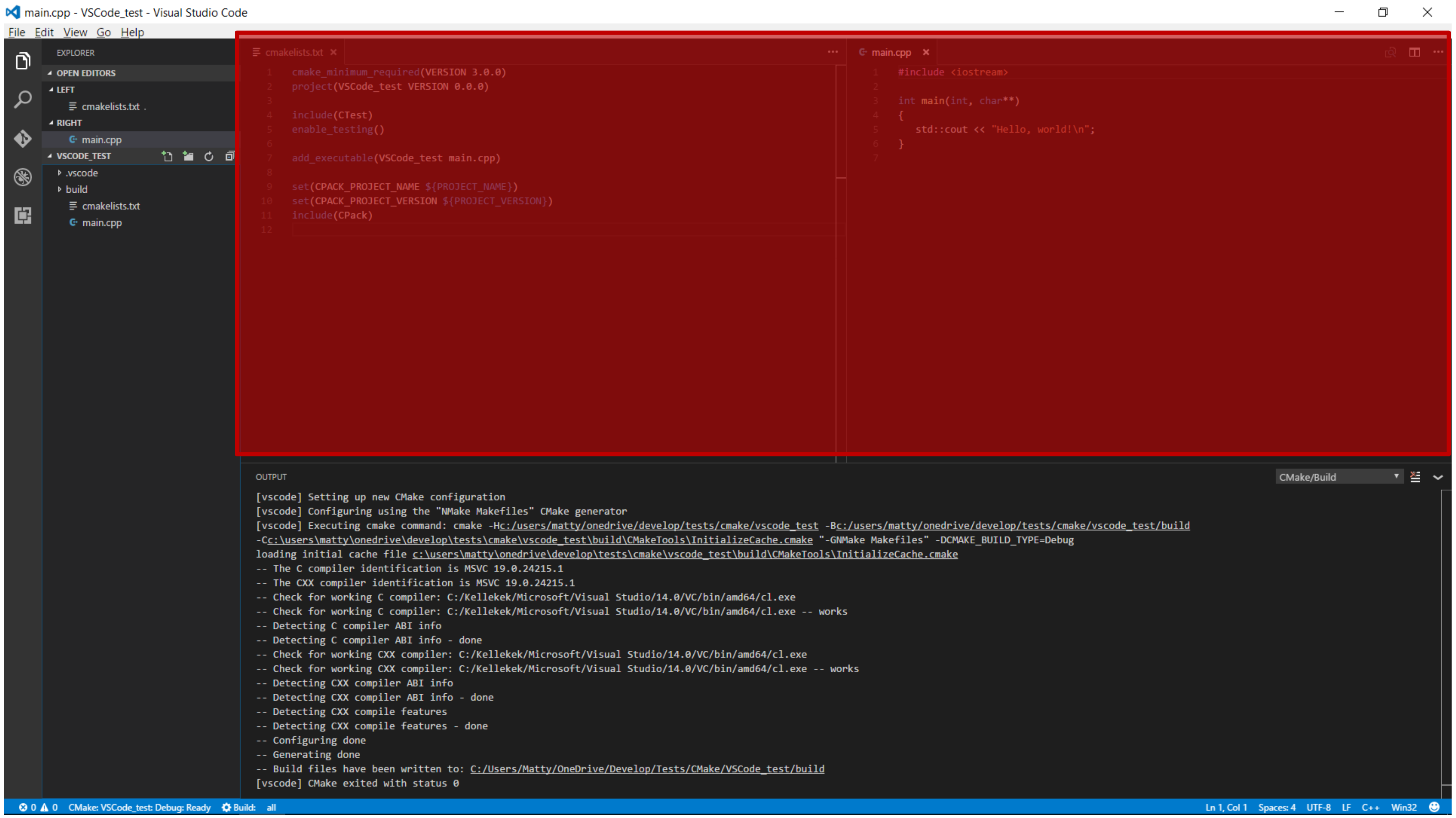
```
#include <iostream>

int main(int, char**)
{
    std::cout << "Hello, world!\n";
}
```

```
[vscode] Setting up new CMake configuration
[vscode] Configuring using the "NMake Makefiles" CMake generator
[vscode] Executing cmake command: cmake -Hc:/users/matty/onedrive/develop/tests/cmake/vscode_test -Bc:/users/matty/onedrive/develop/tests/cmake/vscode_test/build
-Cc:\users\matty\onedrive\develop\tests\cmake\vscode_test\build\CMakeTools\InitializeCache.cmake "-GNMake Makefiles" -DCMAKE_BUILD_TYPE=Debug
loading initial cache file c:\users\matty\onedrive\develop\tests\cmake\vscode_test\build\CMakeTools\InitializeCache.cmake
-- The C compiler identification is MSVC 19.0.24215.1
-- The CXX compiler identification is MSVC 19.0.24215.1
-- Check for working C compiler: C:/Kellekek/Microsoft/Visual Studio/14.0/VC/bin/amd64/cl.exe
-- Check for working C compiler: C:/Kellekek/Microsoft/Visual Studio/14.0/VC/bin/amd64/cl.exe -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working CXX compiler: C:/Kellekek/Microsoft/Visual Studio/14.0/VC/bin/amd64/cl.exe
-- Check for working CXX compiler: C:/Kellekek/Microsoft/Visual Studio/14.0/VC/bin/amd64/cl.exe -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Configuring done
-- Generating done
-- Build files have been written to: C:/Users/Matty/OneDrive/Develop/Tests/CMake/VSCode_test/build
[vscode] CMake exited with status 0
```

```
⊗ 0 ⚠ 0   CMake: VSCode_test: Debug: Ready   Build:  all        Ln 1, Col 1   Spaces: 4   UTF-8   LF   C++   Win32
```

- What happened?

IDE for the powerusers



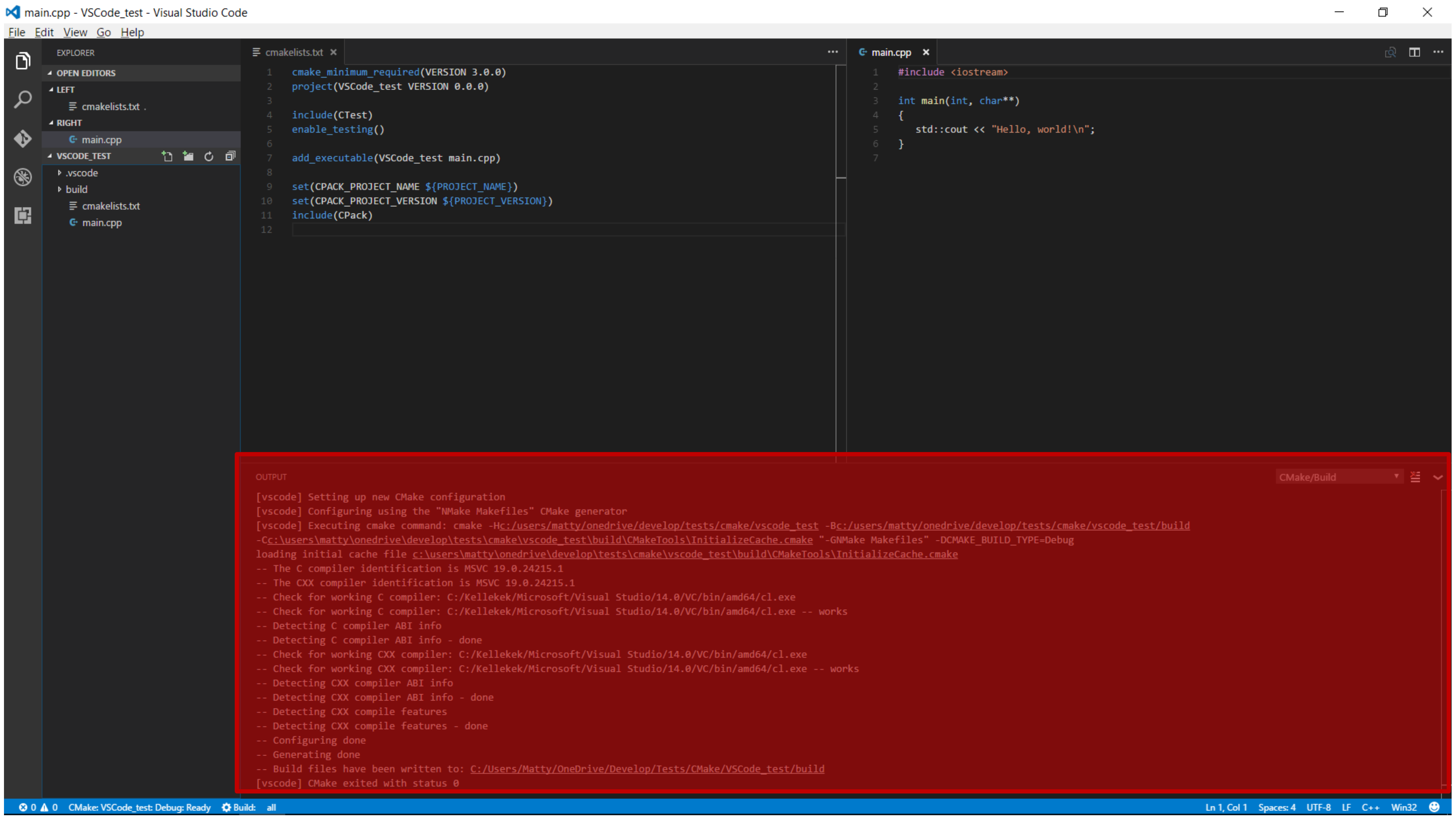
```
cmake_minimum_required(VERSION 3.0.0)
project(VSCode_test VERSION 0.0.0)

include(CTest)
enable_testing()

add_executable(VSCode_test main.cpp)

set(CPACK_PROJECT_NAME ${PROJECT_NAME})
set(CPACK_PROJECT_VERSION ${PROJECT_VERSION})
include(CPack)
```

```cpp
#include <iostream>

int main(int, char**)
{
    std::cout << "Hello, world!\n";
}
```

```
[vscode] Setting up new CMake configuration
[vscode] Configuring using the "NMake Makefiles" CMake generator
[vscode] Executing cmake command: cmake -Hc:/users/matty/onedrive/develop/tests/cmake/vscode_test -Bc:/users/matty/onedrive/develop/tests/cmake/vscode_test/build
-Cc:\users\matty\onedrive\develop\tests\cmake\vscode_test\build\CMakeTools\InitializeCache.cmake "-GNMake Makefiles" -DCMAKE_BUILD_TYPE=Debug
loading initial cache file c:\users\matty\onedrive\develop\tests\cmake\vscode_test\build\CMakeTools\InitializeCache.cmake
-- The C compiler identification is MSVC 19.0.24215.1
-- The CXX compiler identification is MSVC 19.0.24215.1
-- Check for working C compiler: C:/Kellekek/Microsoft/Visual Studio/14.0/VC/bin/amd64/cl.exe
-- Check for working C compiler: C:/Kellekek/Microsoft/Visual Studio/14.0/VC/bin/amd64/cl.exe -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working CXX compiler: C:/Kellekek/Microsoft/Visual Studio/14.0/VC/bin/amd64/cl.exe
-- Check for working CXX compiler: C:/Kellekek/Microsoft/Visual Studio/14.0/VC/bin/amd64/cl.exe -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Configuring done
-- Generating done
-- Build files have been written to: C:/Users/Matty/OneDrive/Develop/Tests/CMake/VSCode_test/build
[vscode] CMake exited with status 0
```

- What happened?
- CMake Tools
  - generated a simple
    - CMakeLists.txt file
    - main.cpp

IDE for the powerusers



```
cmake_minimum_required(VERSION 3.0.0)
project(VSCode_test VERSION 0.0.0)

include(CTest)
enable_testing()

add_executable(VSCode_test main.cpp)

set(CPACK_PROJECT_NAME ${PROJECT_NAME})
set(CPACK_PROJECT_VERSION ${PROJECT_VERSION})
include(CPack)
```

```
#include <iostream>

int main(int, char**)
{
    std::cout << "Hello, world!\n";
}
```

```
[vscode] Setting up new CMake configuration
[vscode] Configuring using the "NMake Makefiles" CMake generator
[vscode] Executing cmake command: cmake -Hc:/users/matty/onedrive/develop/tests/cmake/vscode_test -Bc:/users/matty/onedrive/develop/tests/cmake/vscode_test/build
-Cc:\users\matty\onedrive\develop\tests\cmake\vscode_test\build\CMakeTools\InitializeCache.cmake "-GNMake Makefiles" -DCMAKE_BUILD_TYPE=Debug
loading initial cache file c:\users\matty\onedrive\develop\tests\cmake\vscode_test\build\CMakeTools\InitializeCache.cmake
-- The C compiler identification is MSVC 19.0.24215.1
-- The CXX compiler identification is MSVC 19.0.24215.1
-- Check for working C compiler: C:/Kellekek/Microsoft/Visual Studio/14.0/VC/bin/amd64/cl.exe
-- Check for working C compiler: C:/Kellekek/Microsoft/Visual Studio/14.0/VC/bin/amd64/cl.exe -- works
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working CXX compiler: C:/Kellekek/Microsoft/Visual Studio/14.0/VC/bin/amd64/cl.exe
-- Check for working CXX compiler: C:/Kellekek/Microsoft/Visual Studio/14.0/VC/bin/amd64/cl.exe -- works
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Configuring done
-- Generating done
-- Build files have been written to: C:/Users/Matty/OneDrive/Develop/Tests/CMake/VSCode_test/build
[vscode] CMake exited with status 0
```

- What happened?
- CMake Tools
  - generated a simple
    - CMakeLists.txt file
    - main.cpp
  - called cmake and generated our preferred build systems make files

IDE for the powerusers

CMakeLists.txt

```
cmake_minimum_required(VERSION 3.0.0)
project(VSCode_test VERSION 0.0.0)

include(CTest)
enable_testing()

add_executable(VSCode_test main.cpp)
set(CPACK_PROJECT_NAME $
{PROJECT_NAME})
set(CPACK_PROJECT_VERSION $
{PROJECT_VERSION})
include(CPack)
```

Main.cpp

```
#include <iostream>

int main(int, char**)
{
    std::cout << "Hello, world!\n";
}
```

IDE for the powerusers

CMakeLists.txt

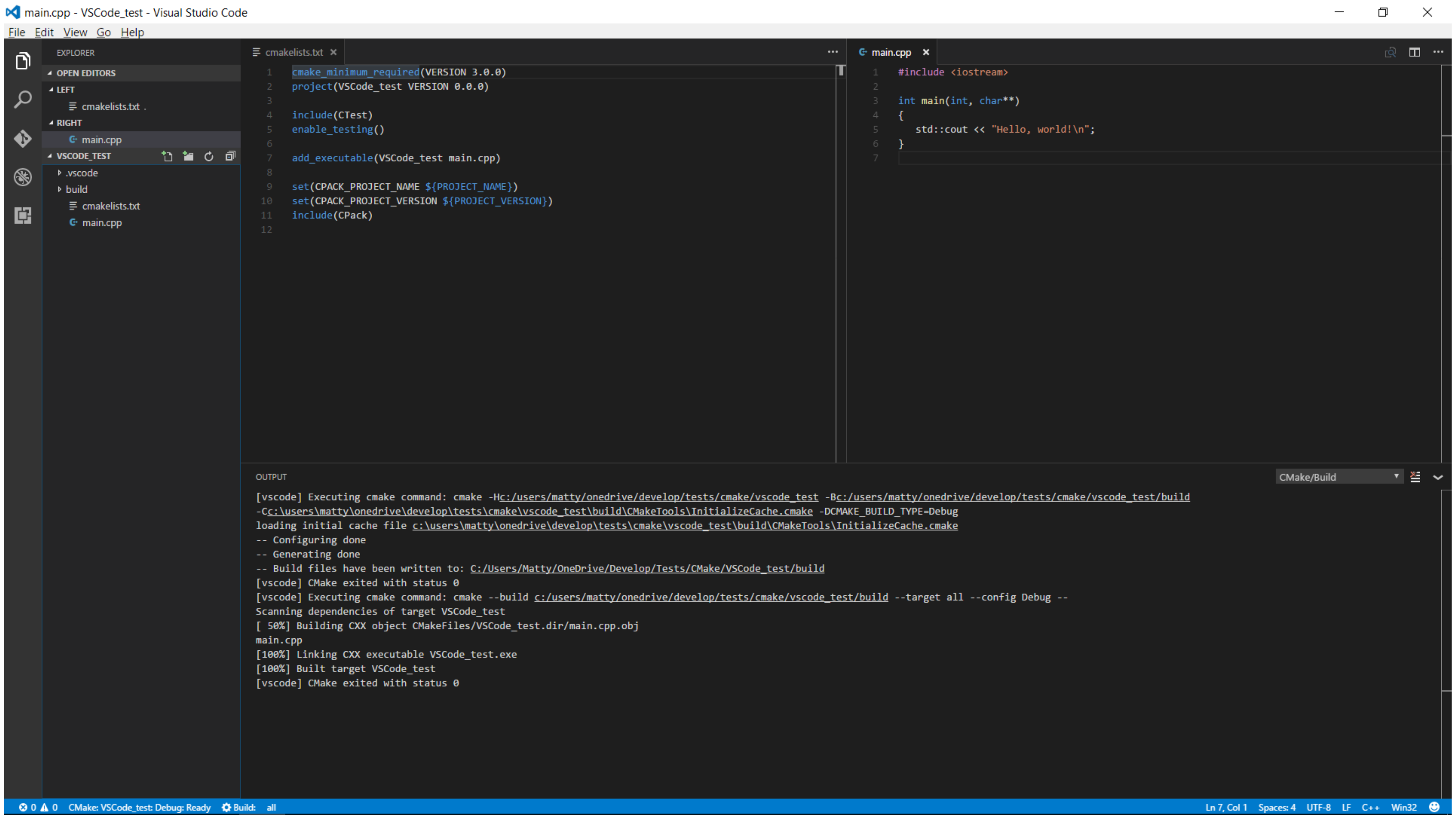
```
cmake_minimum_required(VERSION 3.0.0)
project(VSCode_test VERSION 0.0.0)

include(CTest)
enable_testing()

add_executable(VSCode_test main.cpp)
set(CPACK_PROJECT_NAME $
{PROJECT_NAME})
set(CPACK_PROJECT_VERSION $
{PROJECT_VERSION})
include(CPack)
```

Main.cpp

```
#include <iostream>

int main(int, char**)
{
    std::cout << "Hello, world!\n";
}
```

Now press „F7”

IDE for the powerusers



cmakelists.txt

```
cmake_minimum_required(VERSION 3.0.0)
project(VSCode_test VERSION 0.0.0)

include(CTest)
enable_testing()

add_executable(VSCode_test main.cpp)

set(CPACK_PROJECT_NAME ${PROJECT_NAME})
set(CPACK_PROJECT_VERSION ${PROJECT_VERSION})
include(CPack)
```

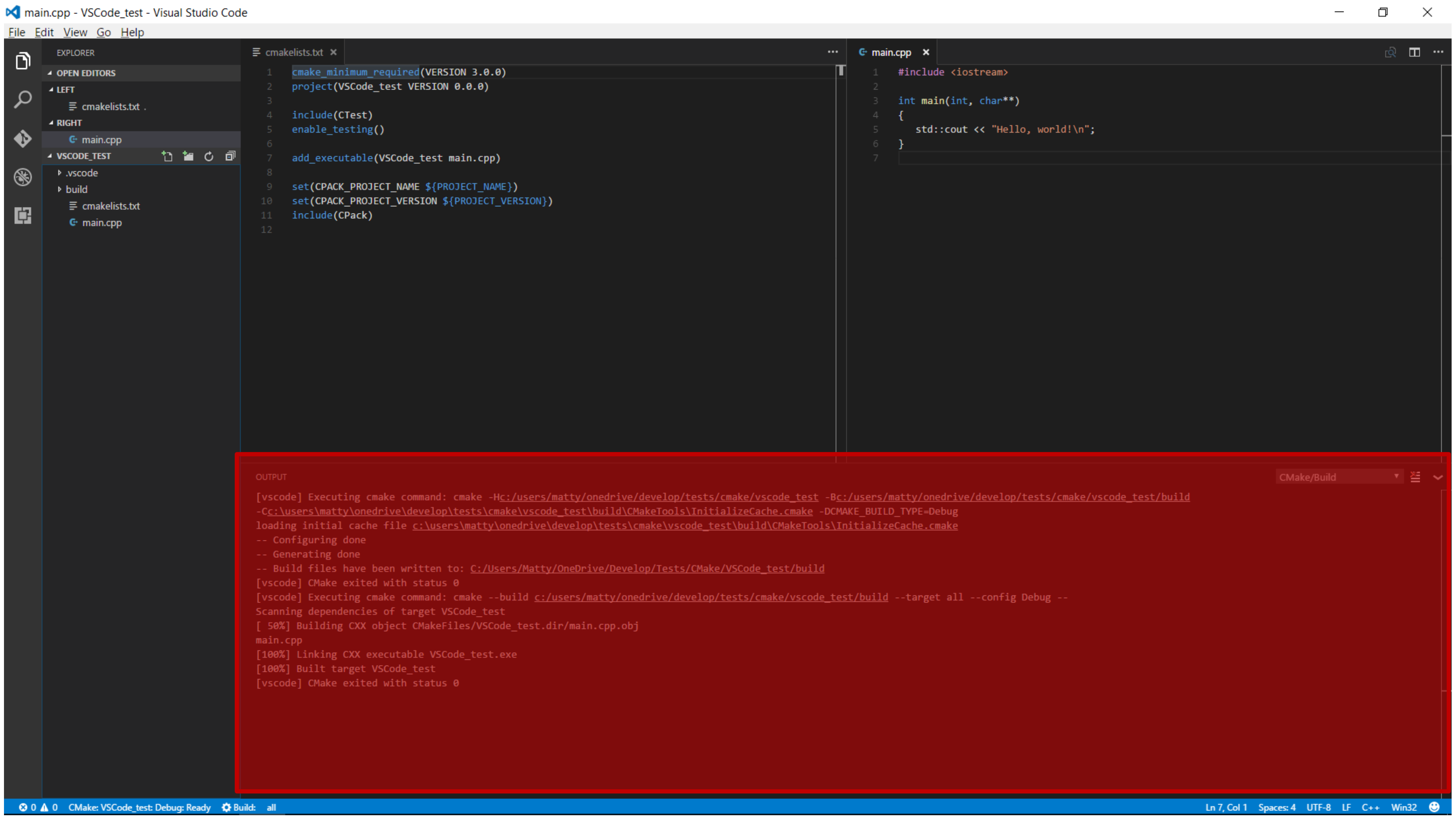
main.cpp

```cpp
#include <iostream>

int main(int, char**)
{
    std::cout << "Hello, world!\n";
}
```

```
[vscode] Executing cmake command: cmake -Hc:/users/matty/onedrive/develop/tests/cmake/vscode_test -Bc:/users/matty/onedrive/develop/tests/cmake/vscode_test/build
-Cc:\users\matty\onedrive\develop\tests\cmake\vscode_test\build\CMakeTools\InitializeCache.cmake -DCMAKE_BUILD_TYPE=Debug
loading initial cache file c:\users\matty\onedrive\develop\tests\cmake\vscode_test\build\CMakeTools\InitializeCache.cmake
-- Configuring done
-- Generating done
-- Build files have been written to: C:/Users/Matty/OneDrive/Develop/Tests/CMake/VSCode_test/build
[vscode] CMake exited with status 0
[vscode] Executing cmake command: cmake --build c:/users/matty/onedrive/develop/tests/cmake/vscode_test/build --target all --config Debug --
Scanning dependencies of target VSCode_test
[ 50%] Building CXX object CMakeFiles/VSCode_test.dir/main.cpp.obj
main.cpp
[100%] Linking CXX executable VSCode_test.exe
[100%] Built target VSCode_test
[vscode] CMake exited with status 0
```

- What happened?

IDE for the powerusers



```
cmake_minimum_required(VERSION 3.0.0)
project(VSCode_test VERSION 0.0.0)

include(CTest)
enable_testing()

add_executable(VSCode_test main.cpp)

set(CPACK_PROJECT_NAME ${PROJECT_NAME})
set(CPACK_PROJECT_VERSION ${PROJECT_VERSION})
include(CPack)
```

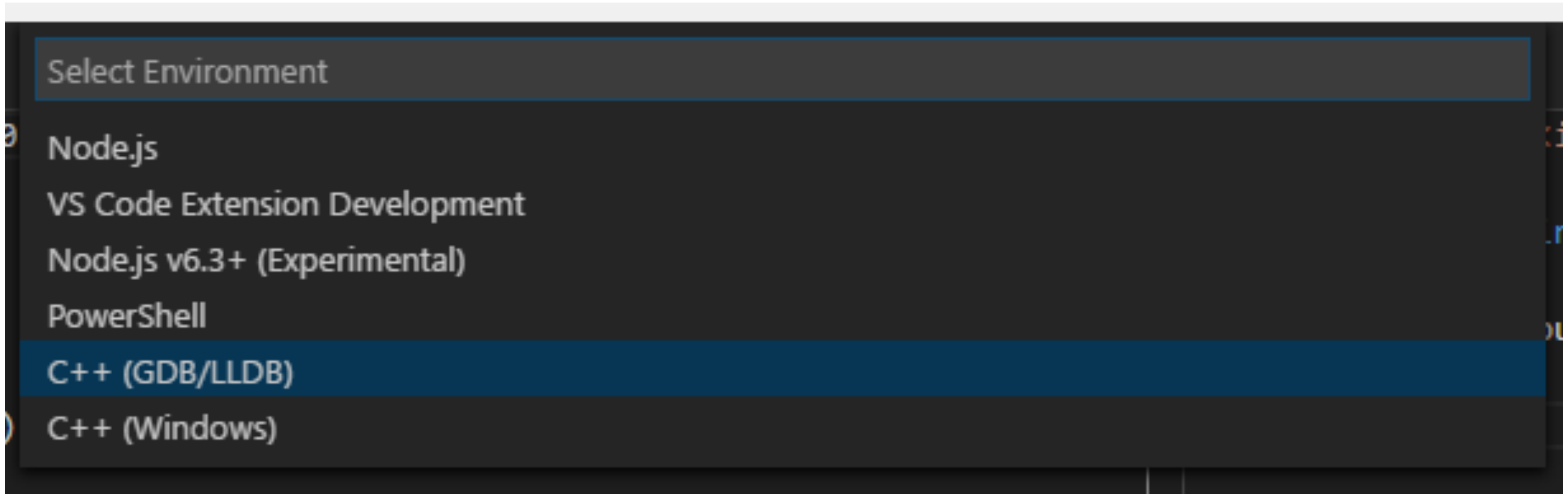
```
#include <iostream>

int main(int, char**)
{
    std::cout << "Hello, world!\n";
}
```

```
[vscode] Executing cmake command: cmake -Hc:/users/matty/onedrive/develop/tests/cmake/vscode_test -Bc:/users/matty/onedrive/develop/tests/cmake/vscode_test/build
-Cc:\users\matty\onedrive\develop\tests\cmake\vscode_test\build\CMakeTools\InitializeCache.cmake -DCMAKE_BUILD_TYPE=Debug
loading initial cache file c:\users\matty\onedrive\develop\tests\cmake\vscode_test\build\CMakeTools\InitializeCache.cmake
-- Configuring done
-- Generating done
-- Build files have been written to: C:/Users/Matty/OneDrive/Develop/Tests/CMake/VSCode_test/build
[vscode] CMake exited with status 0
[vscode] Executing cmake command: cmake --build c:/users/matty/onedrive/develop/tests/cmake/vscode_test/build --target all --config Debug --
Scanning dependencies of target VSCode_test
[ 50%] Building CXX object CMakeFiles/VSCode_test.dir/main.cpp.obj
main.cpp
[100%] Linking CXX executable VSCode_test.exe
[100%] Built target VSCode_test
[vscode] CMake exited with status 0
```

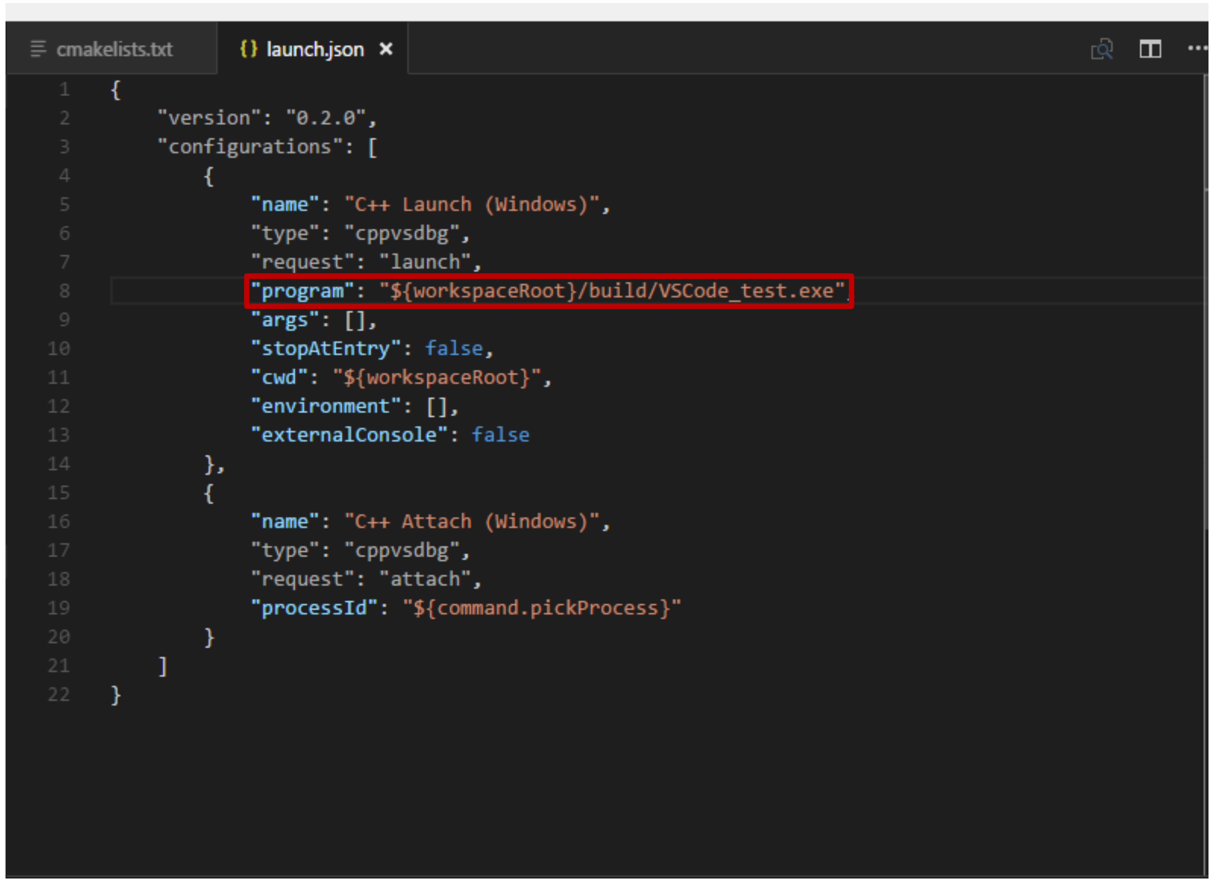
- What happened?
- CMake Tools
  - called cmake which in turn invokes our build system
  - you can see the very same output, as if you had invoked make from the command line

IDE for the powerusers

- Next, press „F5”
- All extensions and core features will prompt for a predefined task to run when pressing the hotkey
- On Linux
  - select „C++ (GDB/LLDB)” for GCC or Clang
- On Windows
  - select „C++ (Windows)” for MSVC
  - select „C++ (GDB/LLDB)” for Clang



IDE for the powerusers



cmakelists.txt | {} launch.json ×

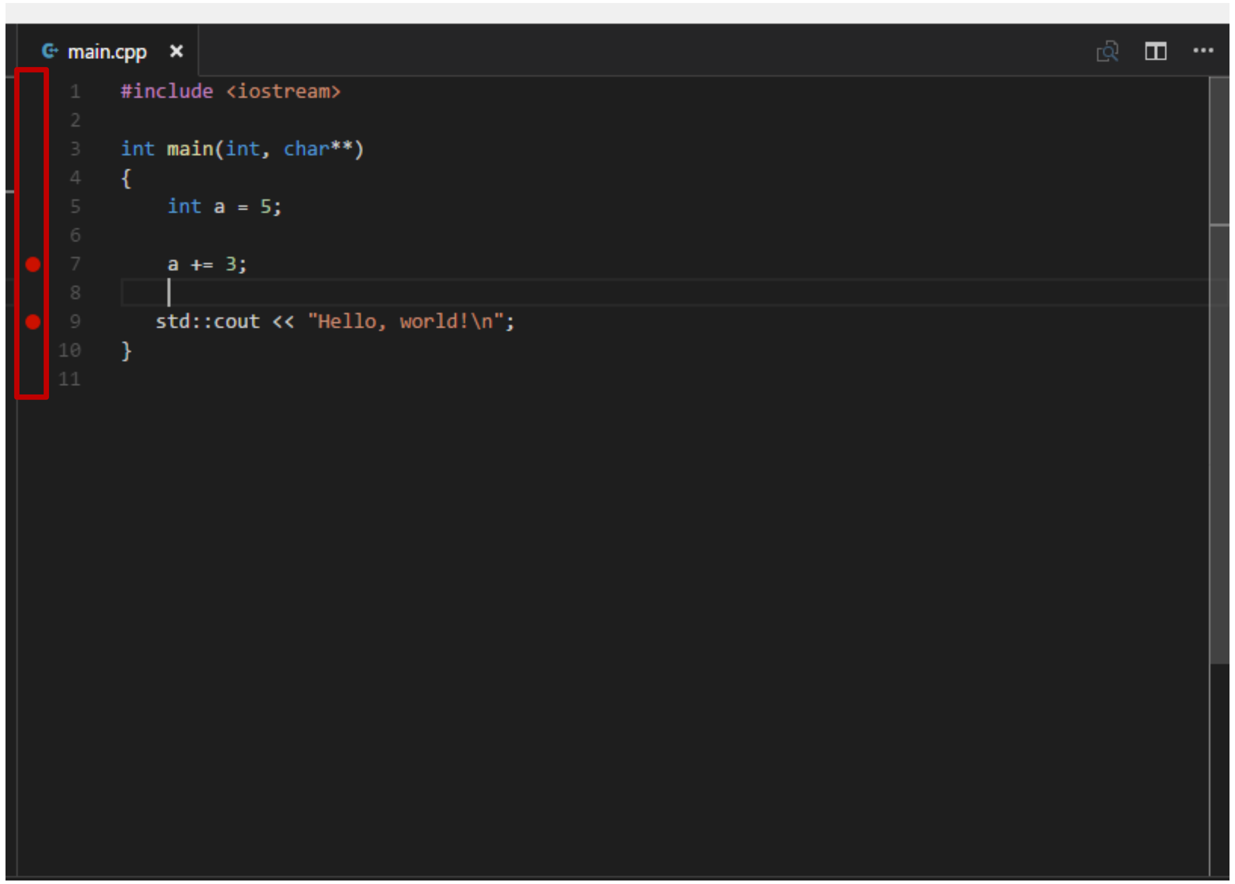
```
{
    "version": "0.2.0",
    "configurations": [
        {
            "name": "C++ Launch (Windows)",
            "type": "cppvsdbg",
            "request": "launch",
            "program": "${workspaceRoot}/build/VSCode_test.exe",
            "args": [],
            "stopAtEntry": false,
            "cwd": "${workspaceRoot}",
            "environment": [],
            "externalConsole": false
        },
        {
            "name": "C++ Attach (Windows)",
            "type": "cppvsdbg",
            "request": "attach",
            "processId": "${command.pickProcess}"
        }
    ]
}
```

- When trying to launch debug for the first time, the extension will not find the executable
- In the .json config file that pops open, specify the location of the executable
- Default location is

`${workspaceRoot}/build/<project_name>`

IDE for the powerusers

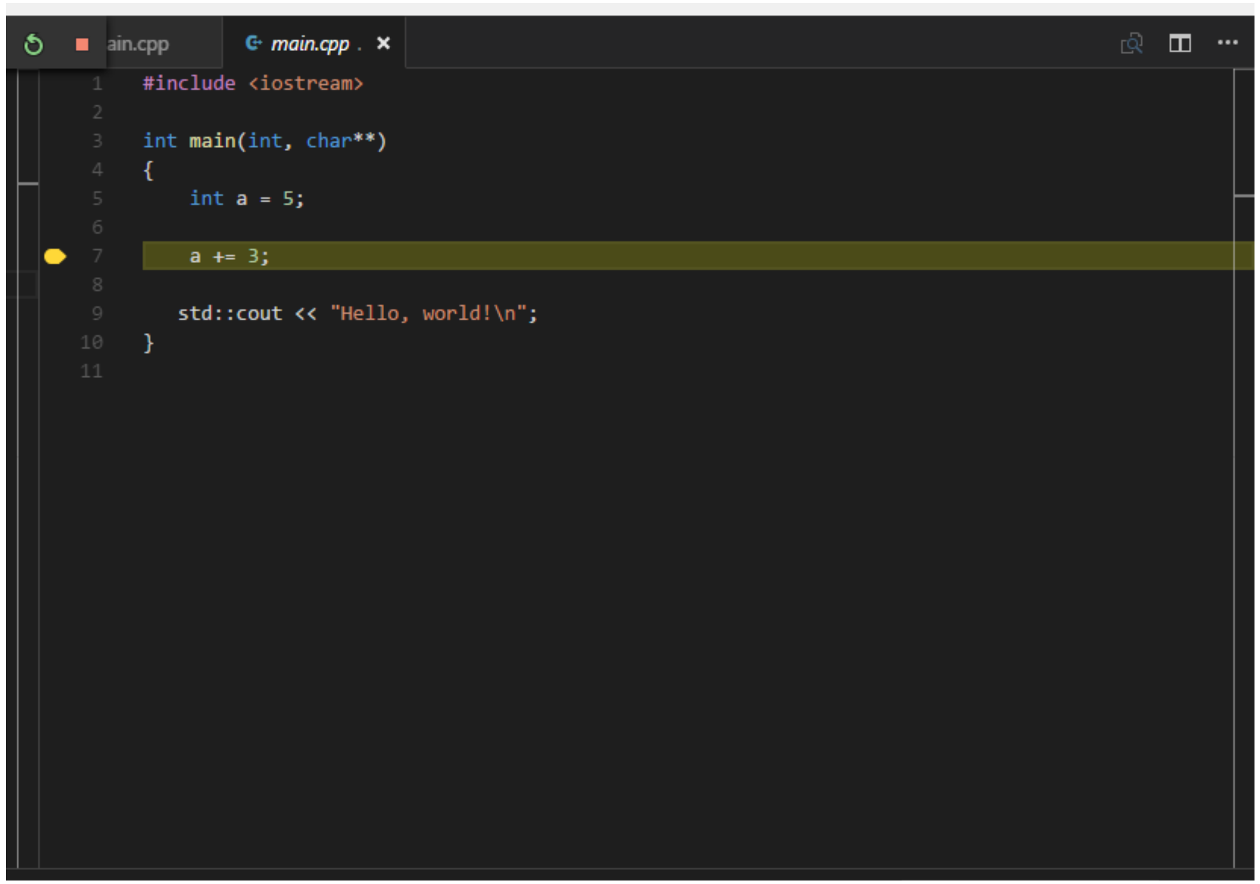
- Adding a few statements to the code makes debugging a little more interesting
- Click on the left of the line number where you want to put a breakpoint
- When the application reaches this point, execution will halt allowing you to inspect program state



```
#include <iostream>

int main(int, char**)
{
    int a = 5;

    a += 3;

   std::cout << "Hello, world!\n";
}

```

IDE for the powerusers

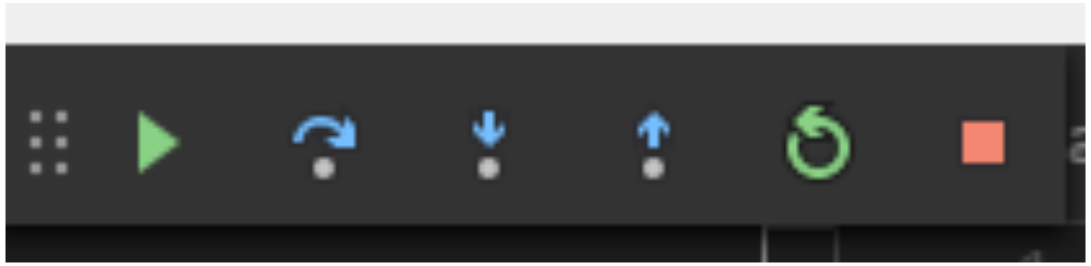
- Pressing „F5” once again will finally (and henceforth) launch debugging.
- Hovering over variables in this halted state will show their actual value in a floating tooltip



```
#include <iostream>

int main(int, char**)
{
    int a = 5;

    a += 3;

   std::cout << "Hello, world!\n";
}

```

IDE for the powerusers

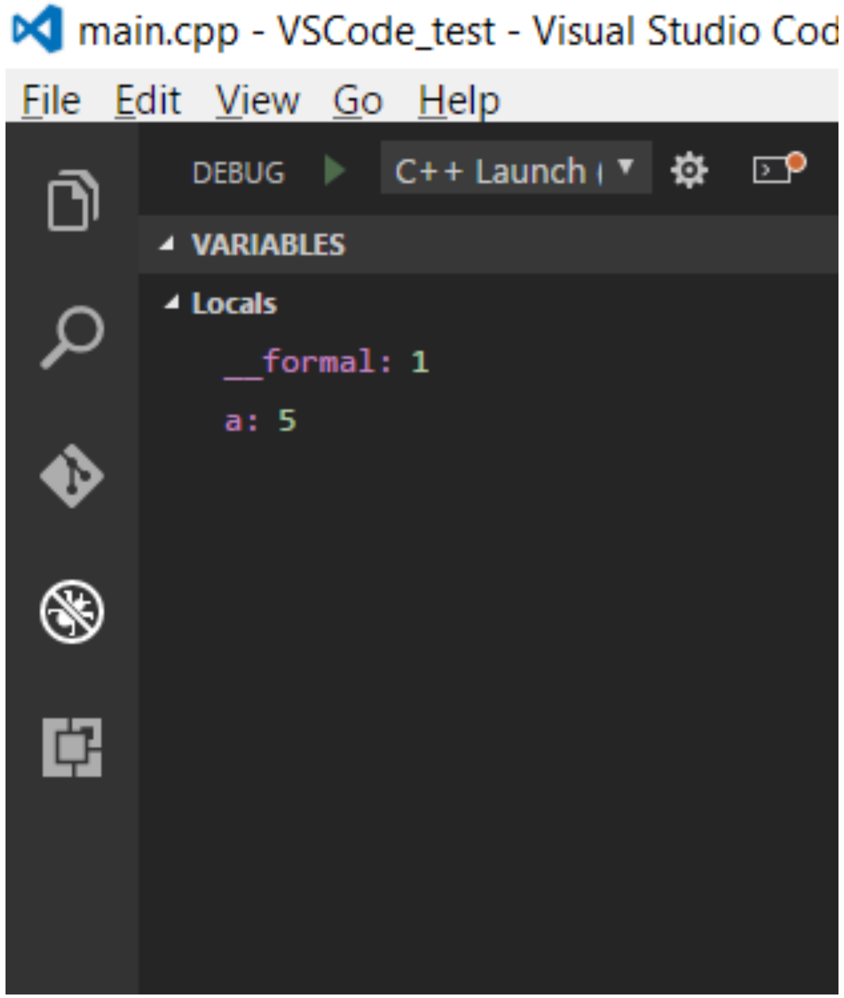
- Pressing „F5” once again will finally (and henceforth) launch debugging.
- Hovering over variables in this halted state will show their actual value in a floating tooltip
- The top-center part shows debugging controls



- Jump to next breakpoint (F5)
- Step over statement (F10)
- Step into statement (F11)
- Step out statement (Shift+F11)
- Restart (Ctrl+F5)
- Stop debugging (Shift+F5)

IDE for the powerusers

Locals


main.cpp - VSCode_test - Visual Studio Cod

File Edit View Go Help

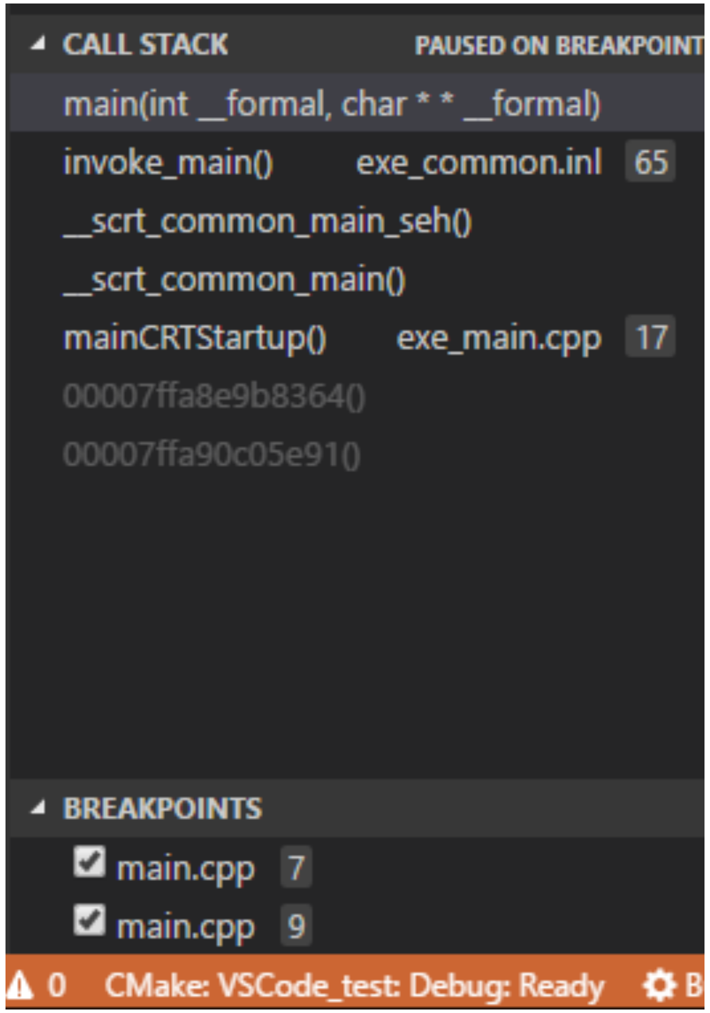
DEBUG C++ Launch

VARIABLES

Locals

```
__formal: 1
a: 5
```

Call stack


CALL STACK PAUSED ON BREAKPOINT

```
main(int __formal, char * * __formal)
invoke_main()          exe_common.inl  65
__scrt_common_main_seh()
__scrt_common_main()
mainCRTStartup()       exe_main.cpp    17
00007ffa8e9b8364()
00007ffa90c05e910
```

BREAKPOINTS

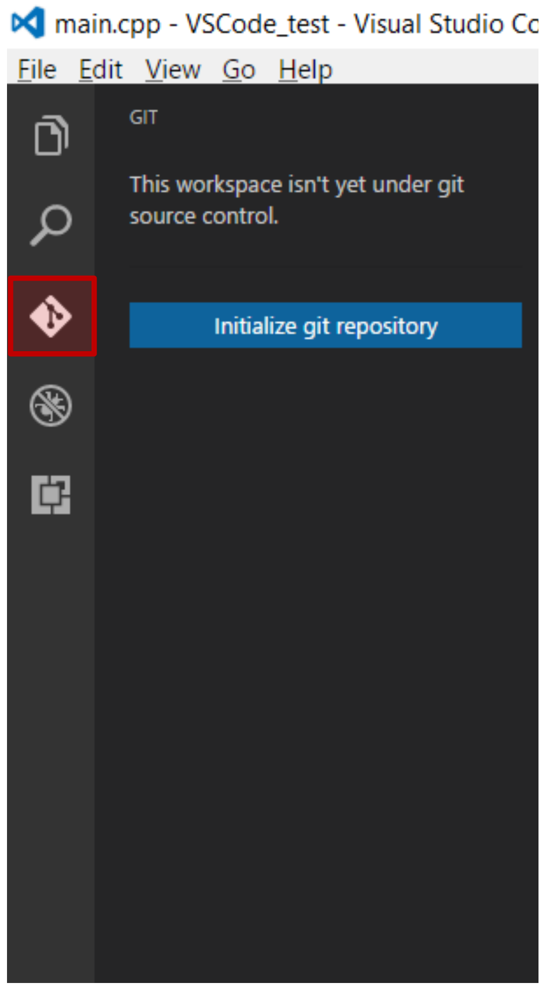
- ✓ main.cpp 7
- ✓ main.cpp 9

⚠ 0 CMake: VSCode_test: Debug: Ready

IDE for the powerusers

- Git is the most common version control system used today
- It is very useful if one wishes to
  - Roll back to earlier versions of the code
  - Use a functioning version while developing a new feature
  - Experiment with code
  - Collaborate with other people on a single codebase
- A more thorough introduction to Git can be found in last year’s slides

IDE for the powerusers

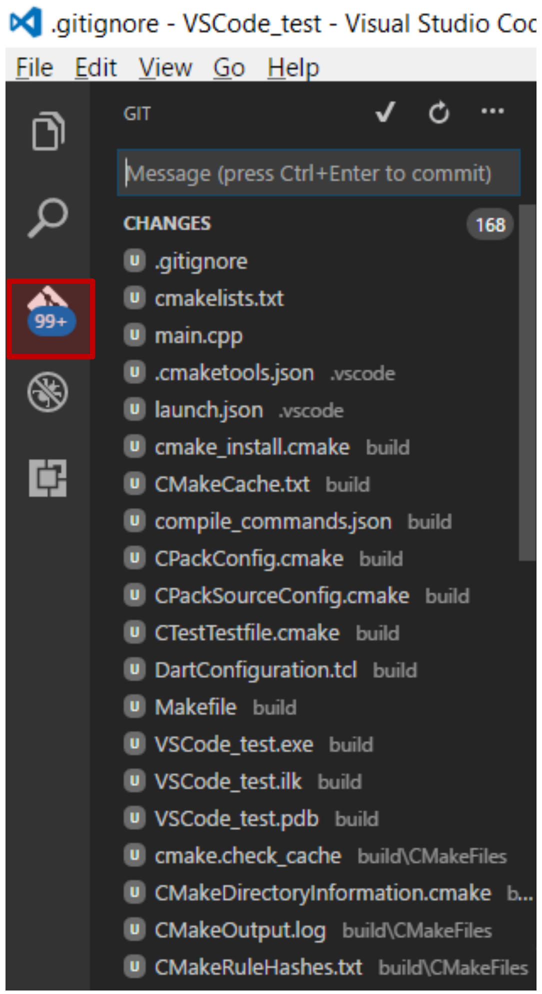

main.cpp - VSCode_test - Visual Studio Cc

This workspace isn't yet under git source control.

- Git is so common in just about all programming languages that Git support is a core feature in VS Code
- Our CMake Quick Start project initially has not been setup to be a Git repository
- We can do so by opening the Git sidebar in the IDE
- It will tell us that this folder is not part of a repository,
  – we can make it so by clicking

Initialize git repository

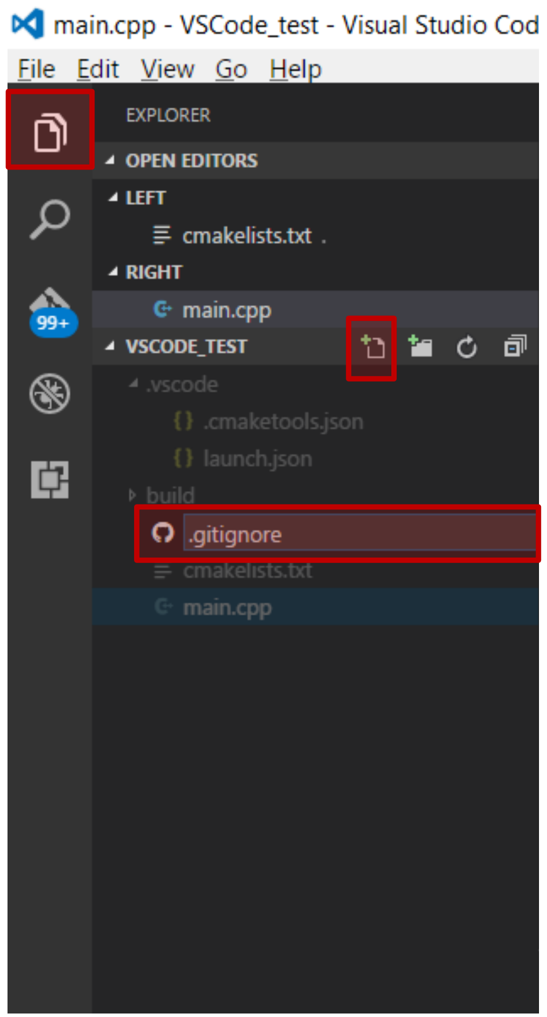
IDE for the powerusers


.gitignore - VSCode_test - Visual Studio Coc
File Edit View Go Help
GIT
Message (press Ctrl+Enter to commit)
CHANGES 168
99+

```
U .gitignore
U cmakelists.txt
U main.cpp
U .cmaketools.json  .vscode
U launch.json  .vscode
U cmake_install.cmake  build
U CMakeCache.txt  build
U compile_commands.json  build
U CPackConfig.cmake  build
U CPackSourceConfig.cmake  build
U CTestTestfile.cmake  build
U DartConfiguration.tcl  build
U Makefile  build
U VSCode_test.exe  build
U VSCode_test.ilk  build
U VSCode_test.pdb  build
U cmake.check_cache  build\CMakeFiles
U CMakeDirectoryInformation.cmake  b...
U CMakeOutput.log  build\CMakeFiles
U CMakeRuleHashes.txt  build\CMakeFiles
```

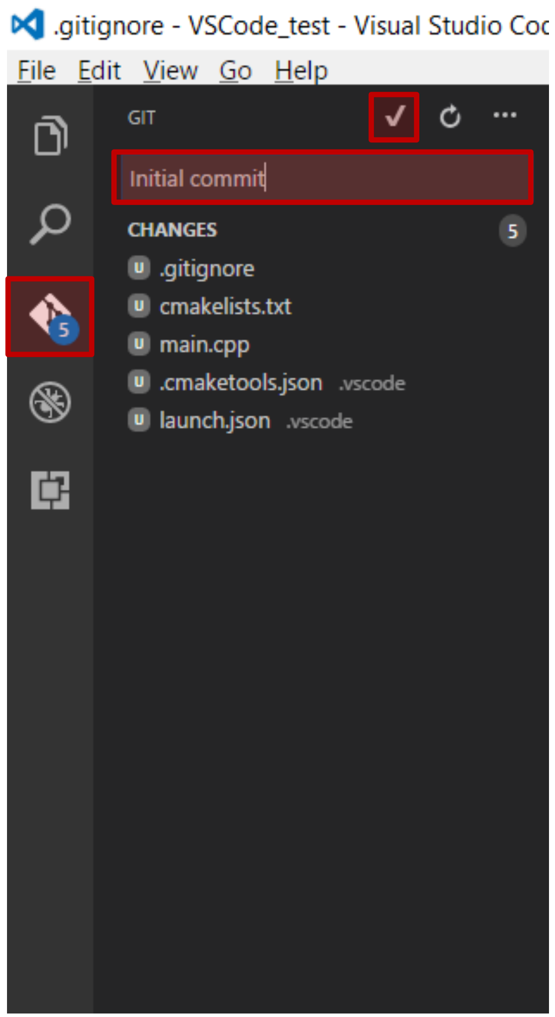
- Next off, Git will show gazillions of files which are staged to be commited
  - This is because we went with the default CMake Tools behavior, which is that the build directory resides inside the same source tree that we develop in
  - This is called in-source building, which is generally not a good practice
  - Ideally one configures CMake Tools to place the side-effects and the final build in a directory outside the source directory
  - This is called out-of-source building

IDE for the powerusers



- Just to play along with the default behavior, we'll instead inform Git that we do not wish to keep track of such files (and thus inflate our repo size with binaries)
  - Because all these files are located in a dir called „build", our job is as simple as it gets
  - We go back to the Explorer pane and click on the „New File" button
  - When prompted for a name, say „.gitignore"
  - Inside put a single line saying „build/"
  - Save the file (Ctrl+s)

IDE for the powerusers



- Going back to the Git pane, now it will show only the source files and the .vscode directory where VS Code stores it’s project-specific configuration
  - When multiple people collaborate, it is good practice to .gitignore the .gitignore file itself too and all IDE related files, as others might be using different IDEs
- Add a short comment to the commit that is informative to you
- Press the ✓ button to make the commit
- Congrats, you got yourself a basic git repo!

IDE for the powerusers

- Most other Git commands are accessed from the Command Palette
  - Just type „git” and a list of possible commands will be filtered out with additional help in the form of drop downs
- Check last year’s slides for typical Git workflow and scenarios when they come in handy
  - The overhead of using Git comes in handy anytime when the project is of moderate seriousness
- For keeping code safe
  - keep the repo in a folder tracked by a cloud storage vendor
  - occaisonally push commits to a remote server (working/master)

IDE for the powerusers

Wigner Git server

- When you want to keep your code confidential (in relation to Wigner)
  - Mail to admin&wigner.mta.hu
    - Initially send a public rsa key
    - Create repo by sending a mail
  - Basic howto can be found here
- Now you may set this repo as remote/origin and push

Public repositories

- Most popular public repo is definately Github
  - Free accounts may only have public repos
  - Use it if the code is not confidential
- If you want the best of both worlds, show a demo to our admins how to setup a private Github server