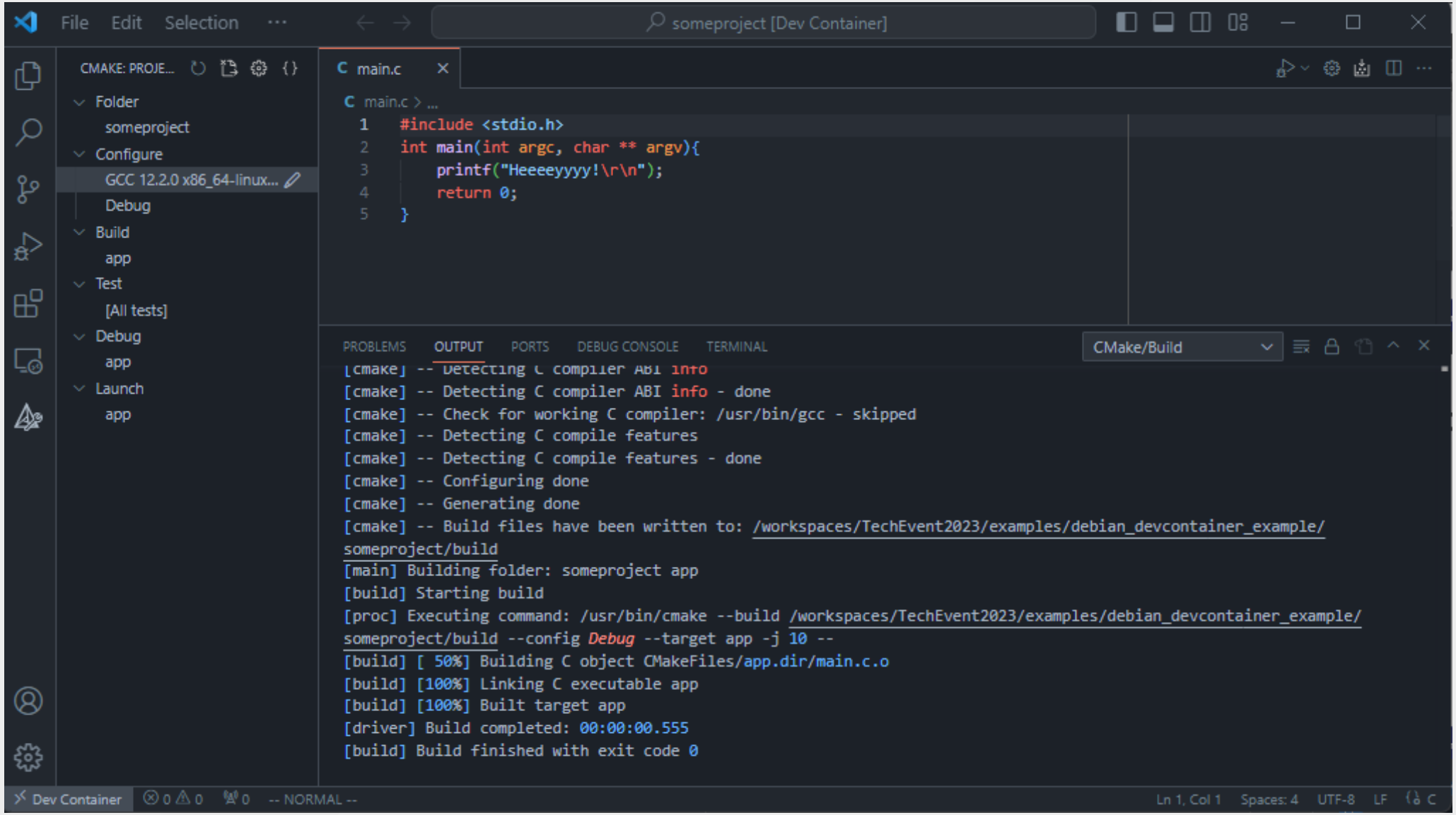
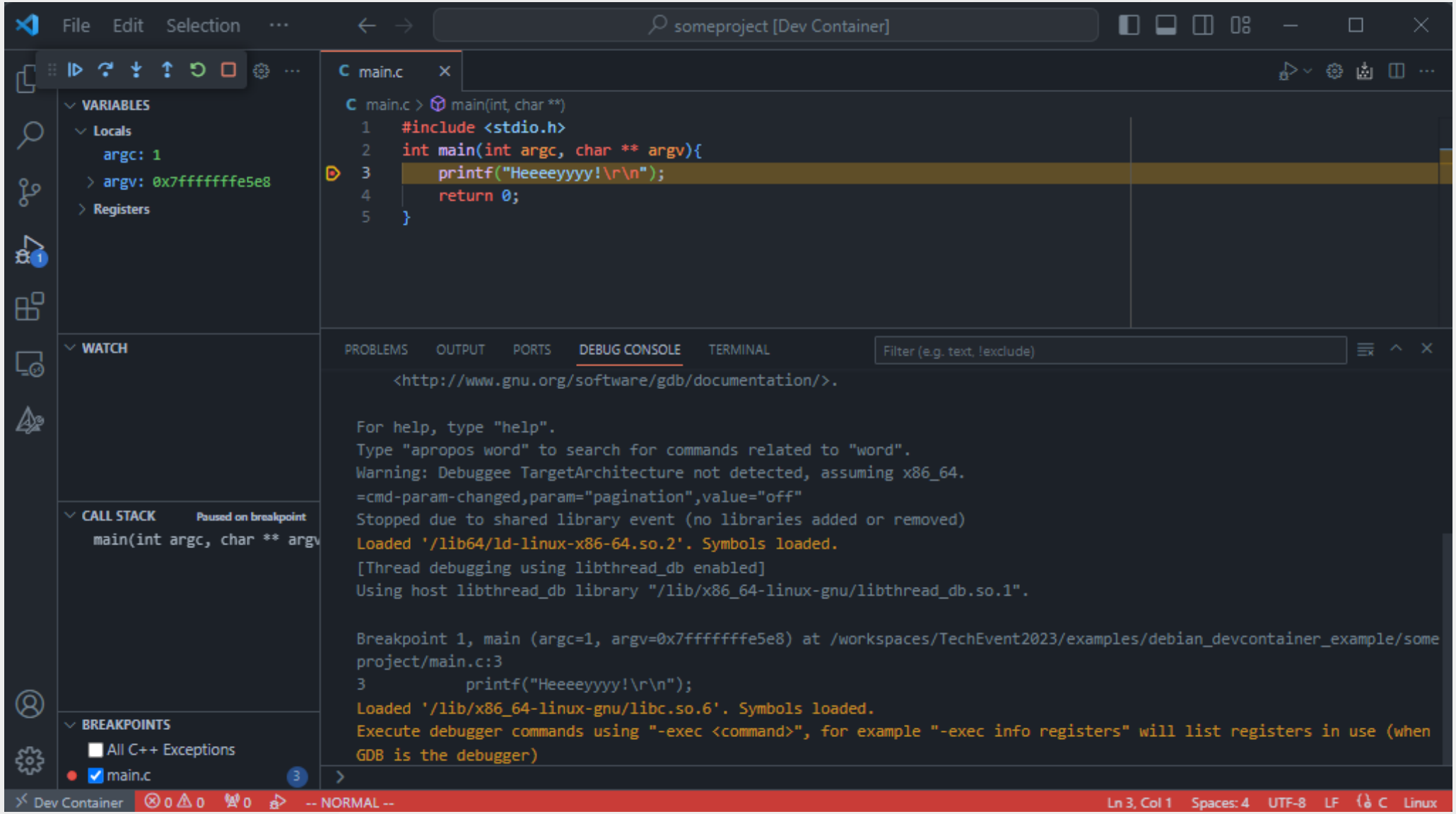
Devcontainers and Embedded software development

How-to and some notes

What's the goal?

- VSCode
- Simple hello-world application in C
- Easy building from IDE
- Integrated debugging of application
- Running in devcontainer


File Edit Selection ···
someproject [Dev Container]
CMAKE: PROJE...
Folder
someproject
Configure
GCC 12.2.0 x86_64-linux...
Debug
Build
app
Test
[All tests]
Debug
app
Launch
app
main.c
main.c > ...
1 #include <stdio.h>
2 int main(int argc, char ** argv){
3 printf("Heeeeyyyy!\r\n");
4 return 0;
5 }
PROBLEMS OUTPUT PORTS DEBUG CONSOLE TERMINAL
CMake/Build
[cmake] -- Detecting C compiler ABI info
[cmake] -- Detecting C compiler ABI info - done
[cmake] -- Check for working C compiler: /usr/bin/gcc - skipped
[cmake] -- Detecting C compile features
[cmake] -- Detecting C compile features - done
[cmake] -- Configuring done
[cmake] -- Generating done
[cmake] -- Build files have been written to: /workspaces/TechEvent2023/examples/debian_devcontainer_example/someproject/build
[main] Building folder: someproject app
[build] Starting build
[proc] Executing command: /usr/bin/cmake --build /workspaces/TechEvent2023/examples/debian_devcontainer_example/someproject/build --config Debug --target app -j 10 --
[build] [50%] Building C object CMakeFiles/app.dir/main.c.o
[build] [100%] Linking C executable app
[build] [100%] Built target app
[driver] Build completed: 00:00:00.555
[build] Build finished with exit code 0
Dev Container
-- NORMAL --
Ln 1, Col 1
Spaces: 4
UTF-8
LF
C

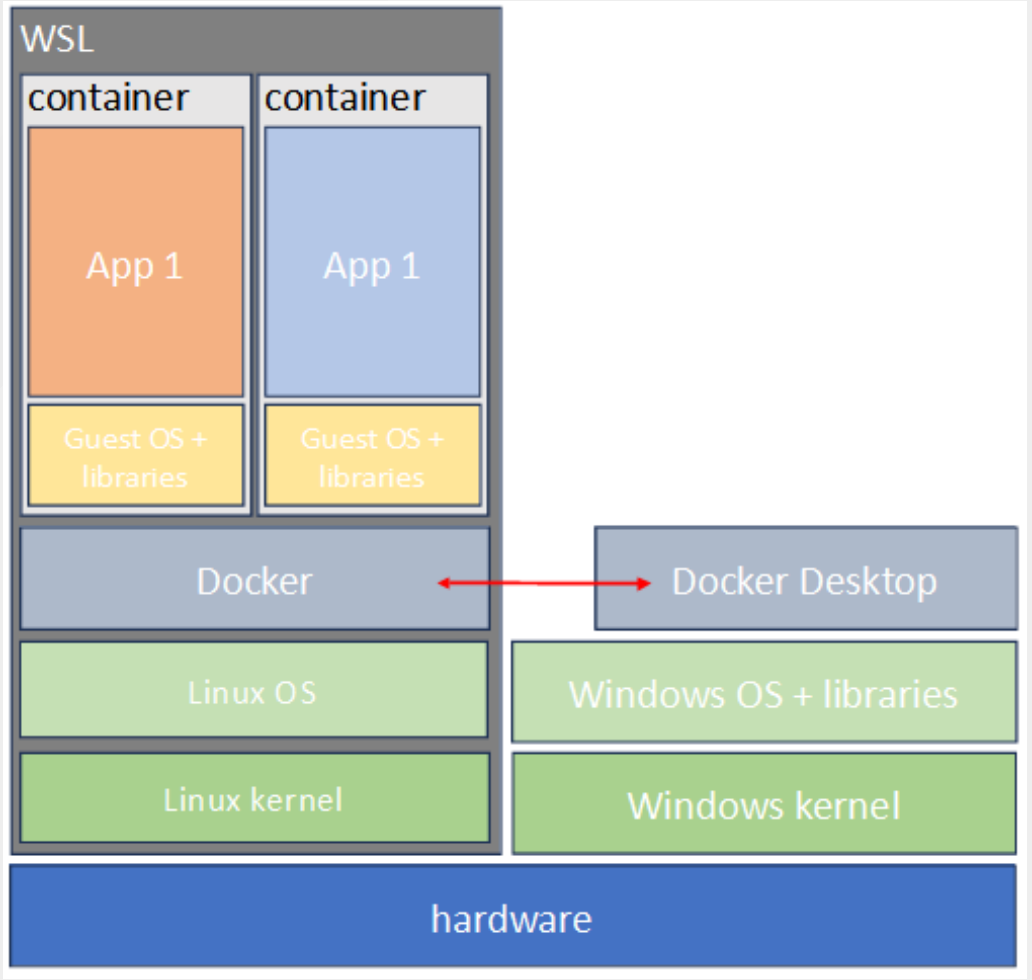

File Edit Selection
someproject [Dev Container]
main.c
main.c > main(int, char **)
#include <stdio.h>
int main(int argc, char ** argv){
printf("Heeeeyyyy!\r\n");
return 0;
}
VARIABLES
Locals
argc: 1
argv: 0x7fffffffe5e8
Registers
WATCH
CALL STACK
Paused on breakpoint
main(int argc, char ** argv
BREAKPOINTS
All C++ Exceptions
main.c
PROBLEMS OUTPUT PORTS DEBUG CONSOLE TERMINAL
Filter (e.g. text, !exclude)
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word".
Warning: Debuggee TargetArchitecture not detected, assuming x86_64.
=cmd-param-changed,param="pagination",value="off"
Stopped due to shared library event (no libraries added or removed)
Loaded '/lib64/ld-linux-x86-64.so.2'. Symbols loaded.
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
Breakpoint 1, main (argc=1, argv=0x7fffffffe5e8) at /workspaces/TechEvent2023/examples/debian_devcontainer_example/someproject/main.c:3
3 printf("Heeeeyyyy!\r\n");
Loaded '/lib/x86_64-linux-gnu/libc.so.6'. Symbols loaded.
Execute debugger commands using "-exec <command>", for example "-exec info registers" will list registers in use (when GDB is the debugger)
Dev Container
-- NORMAL --
Ln 3, Col 1
Spaces: 4
UTF-8
LF
C
Linux

What do we need

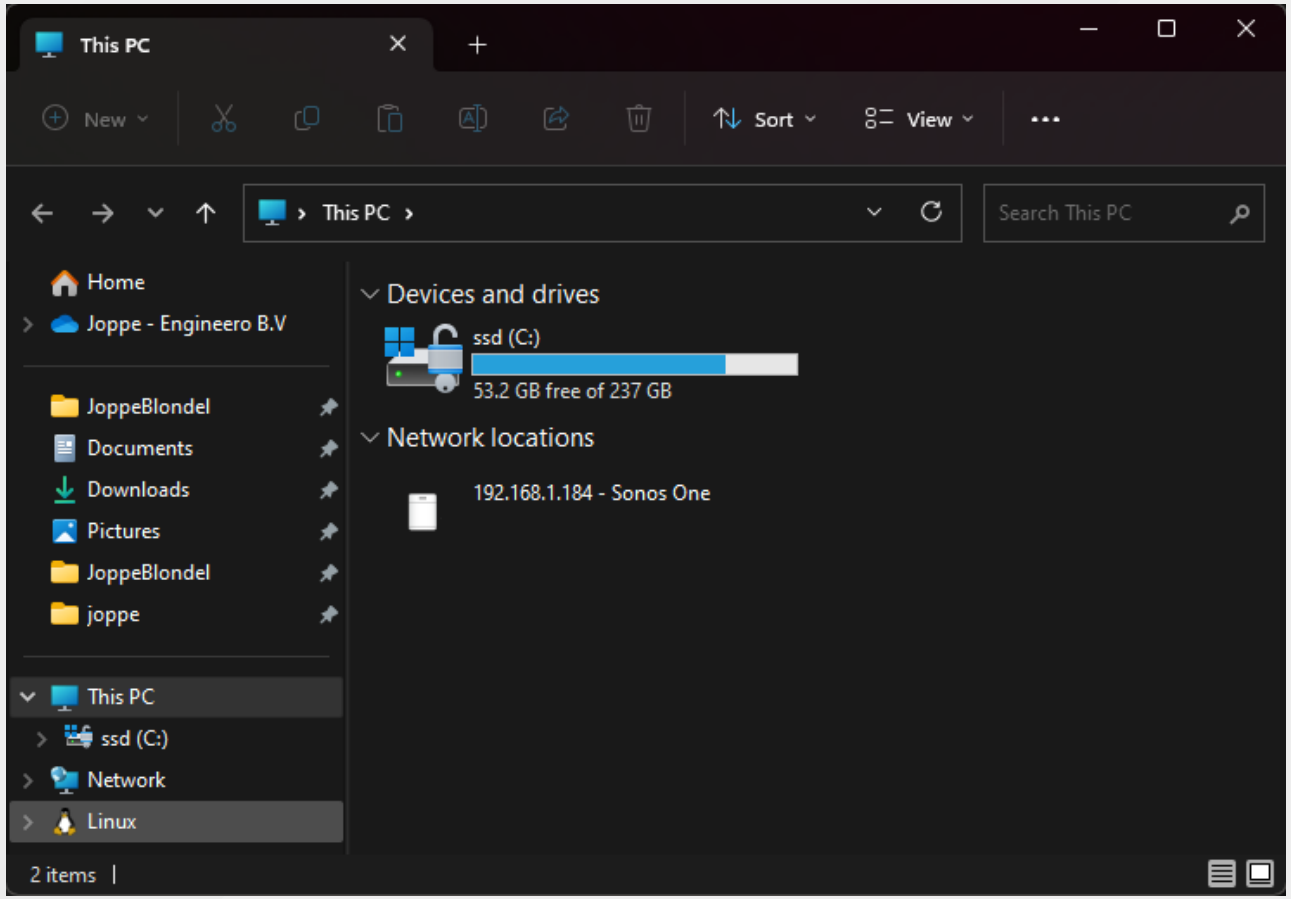
- Definition of Docker container
- Definition of the devcontainer
- Simple buildsystem
- VSCode configuration for building and debugging

A small note

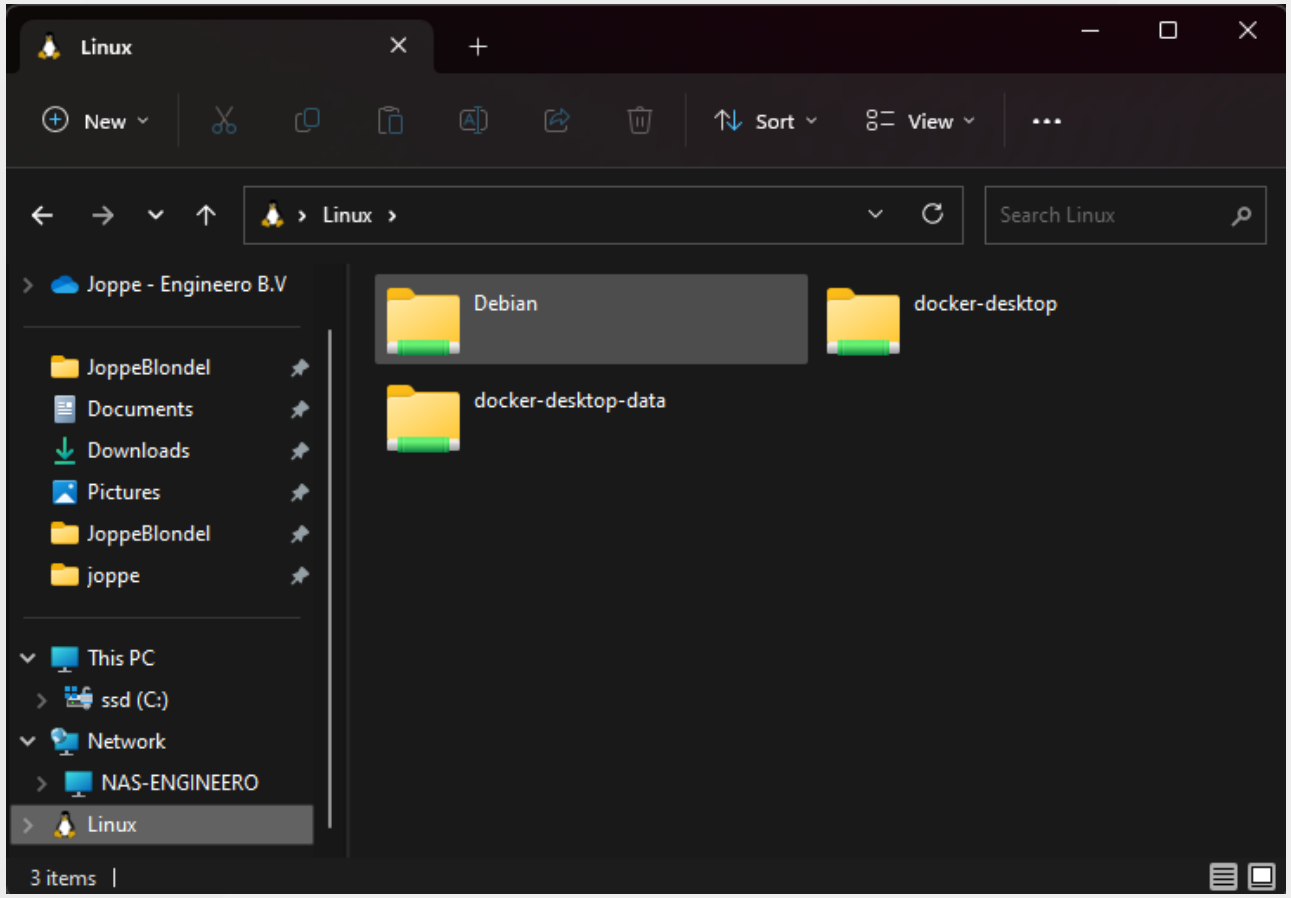
- Containers run on WSL
- Files on Windows must be shared to WSL: slow...
- Fix: Work from WSL



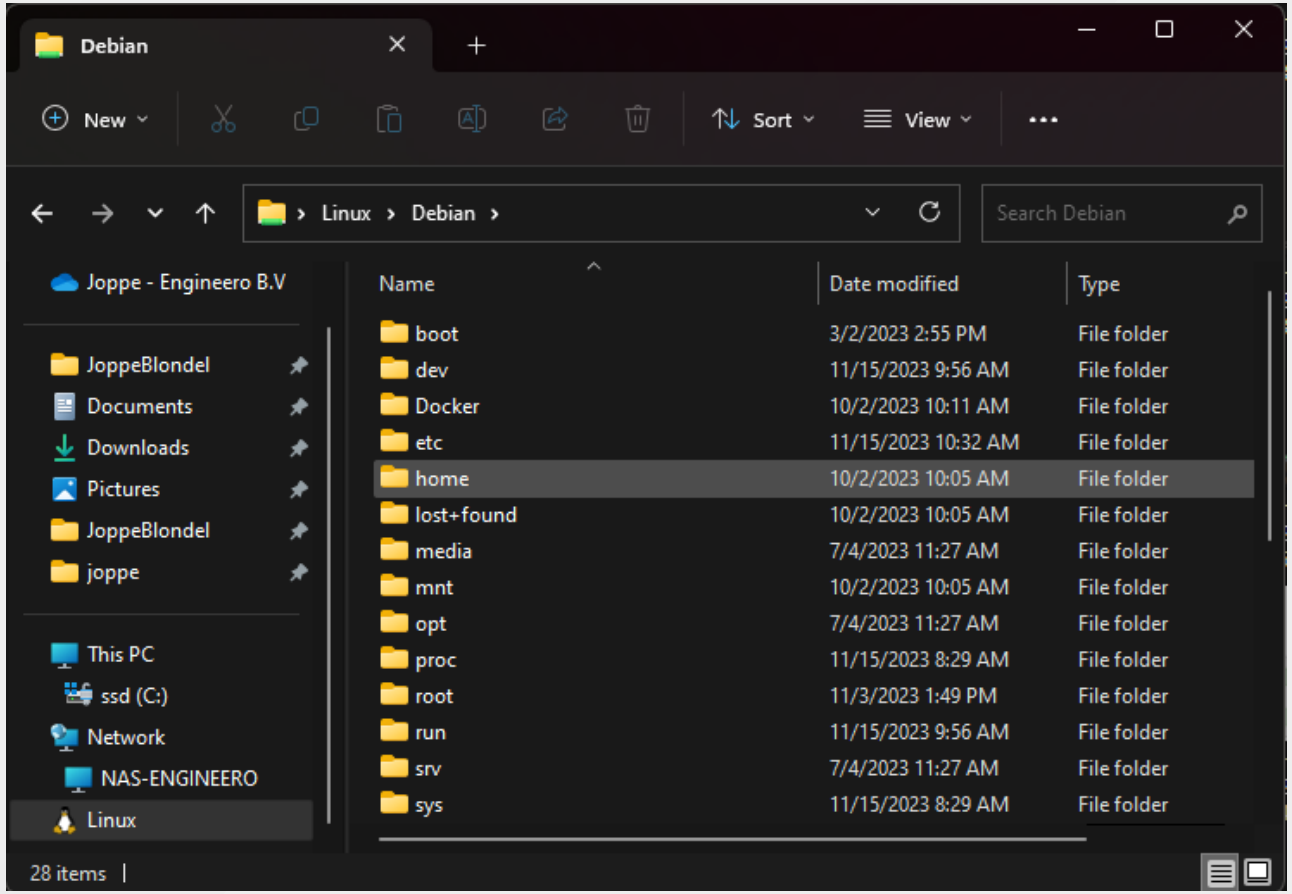
Work from WSL: Create project location



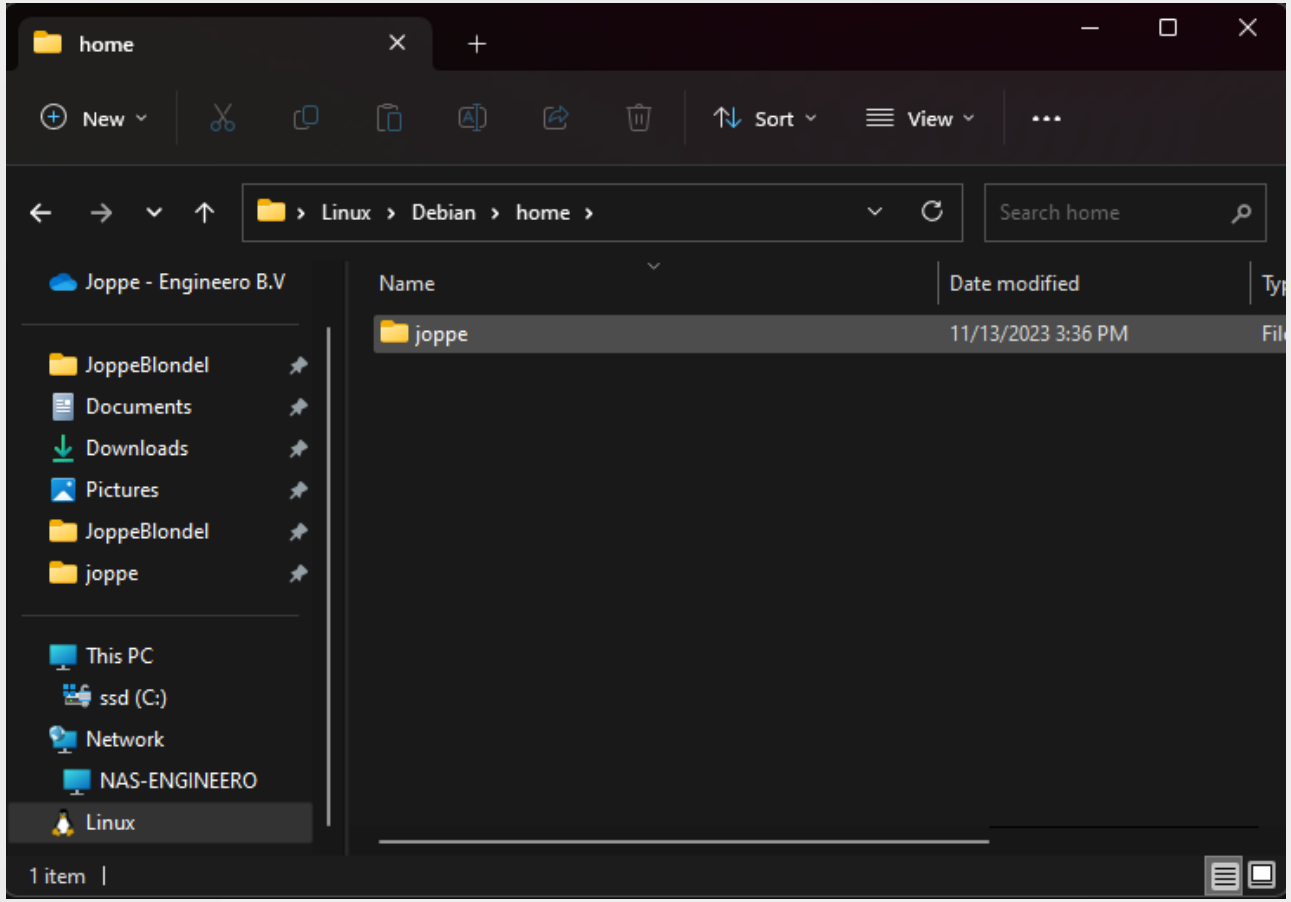
Work from WSL: Create project location



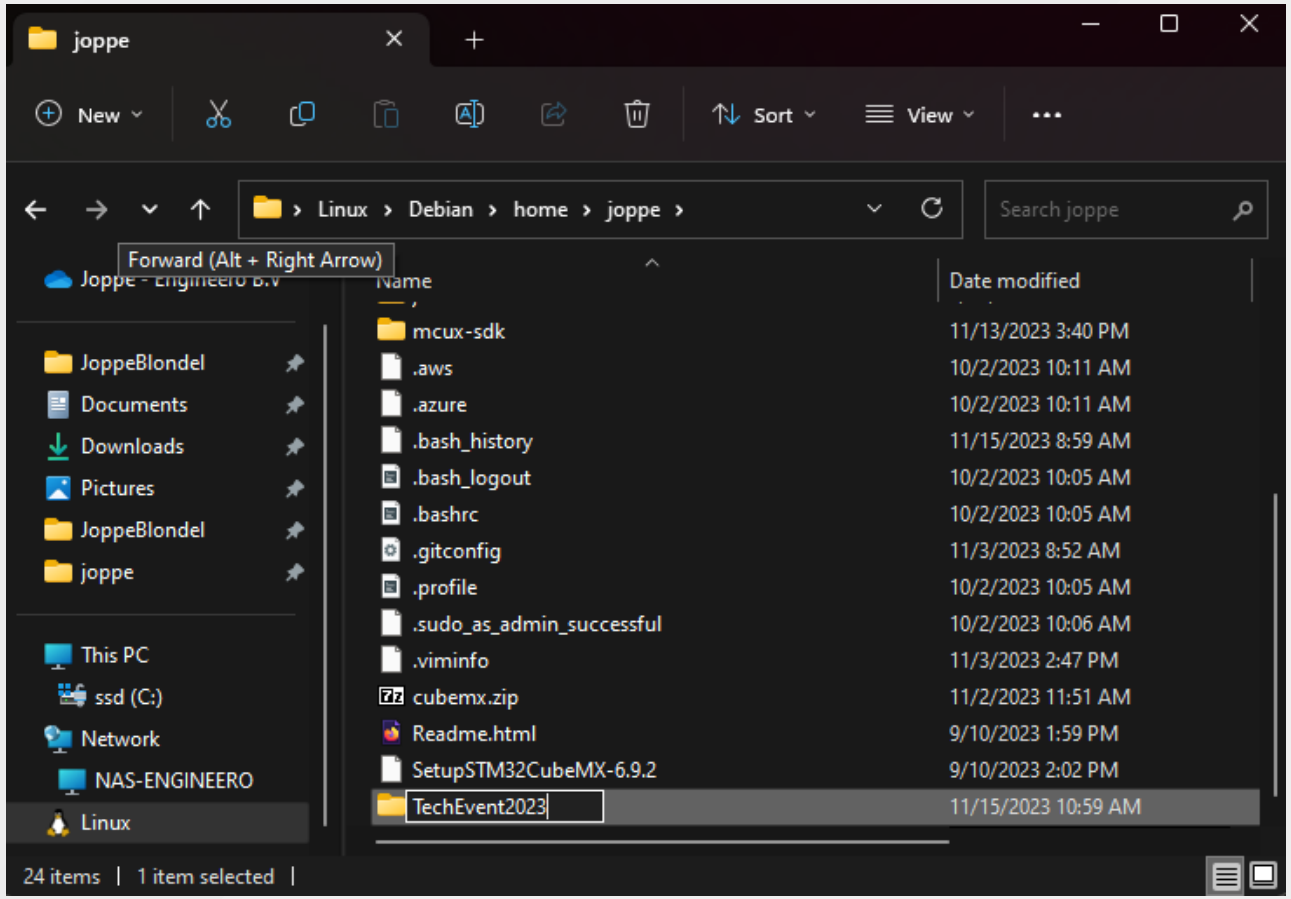
Work from WSL: Create project location



Work from WSL: Create project location



Work from WSL: Create project location

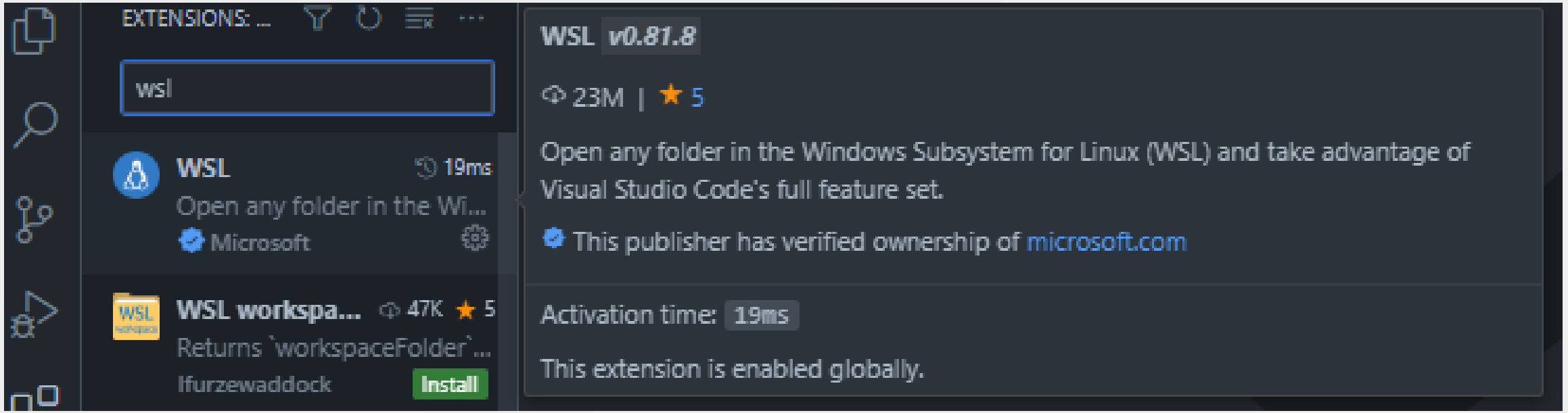


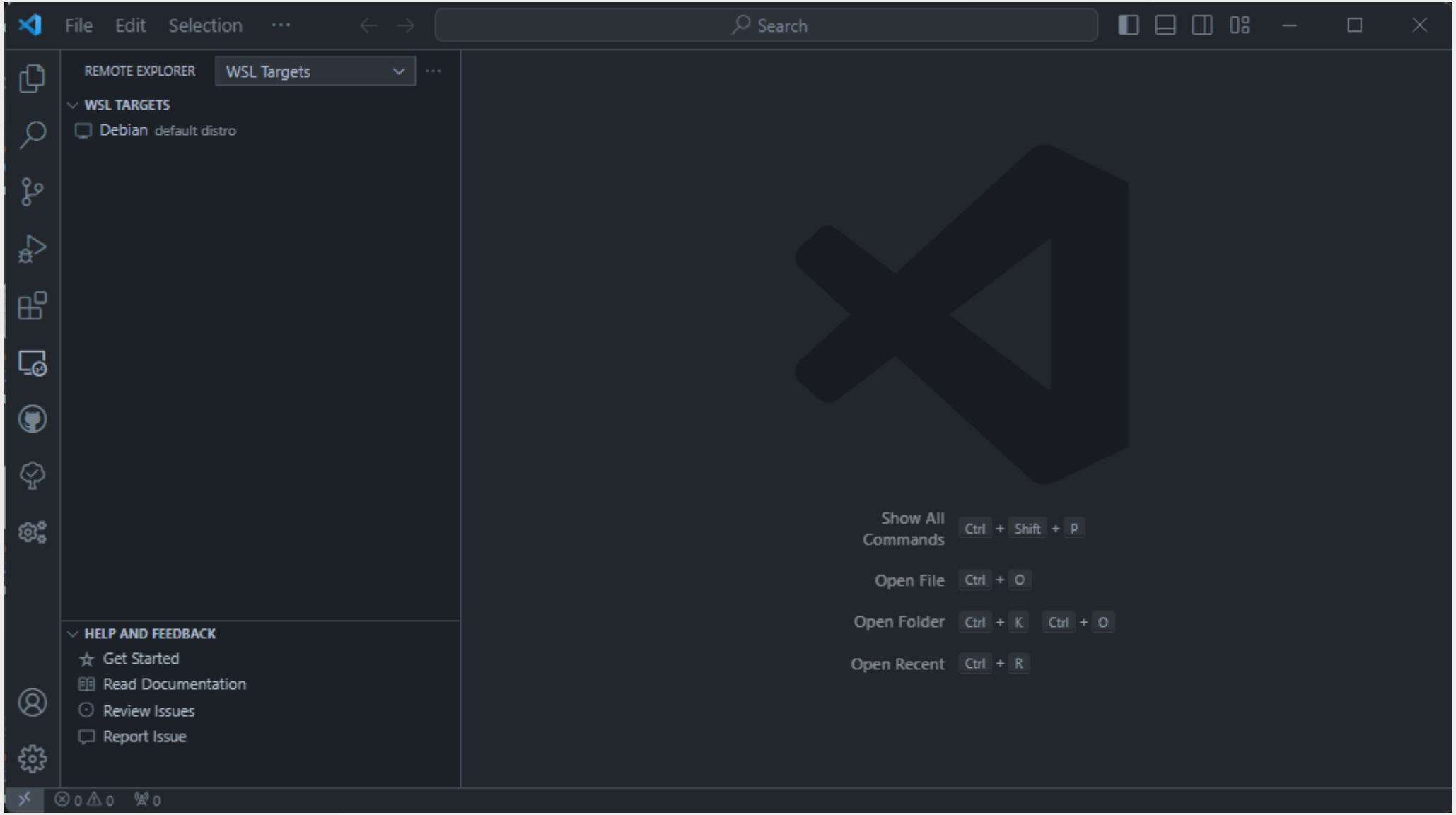
Work from WSL: Create project location

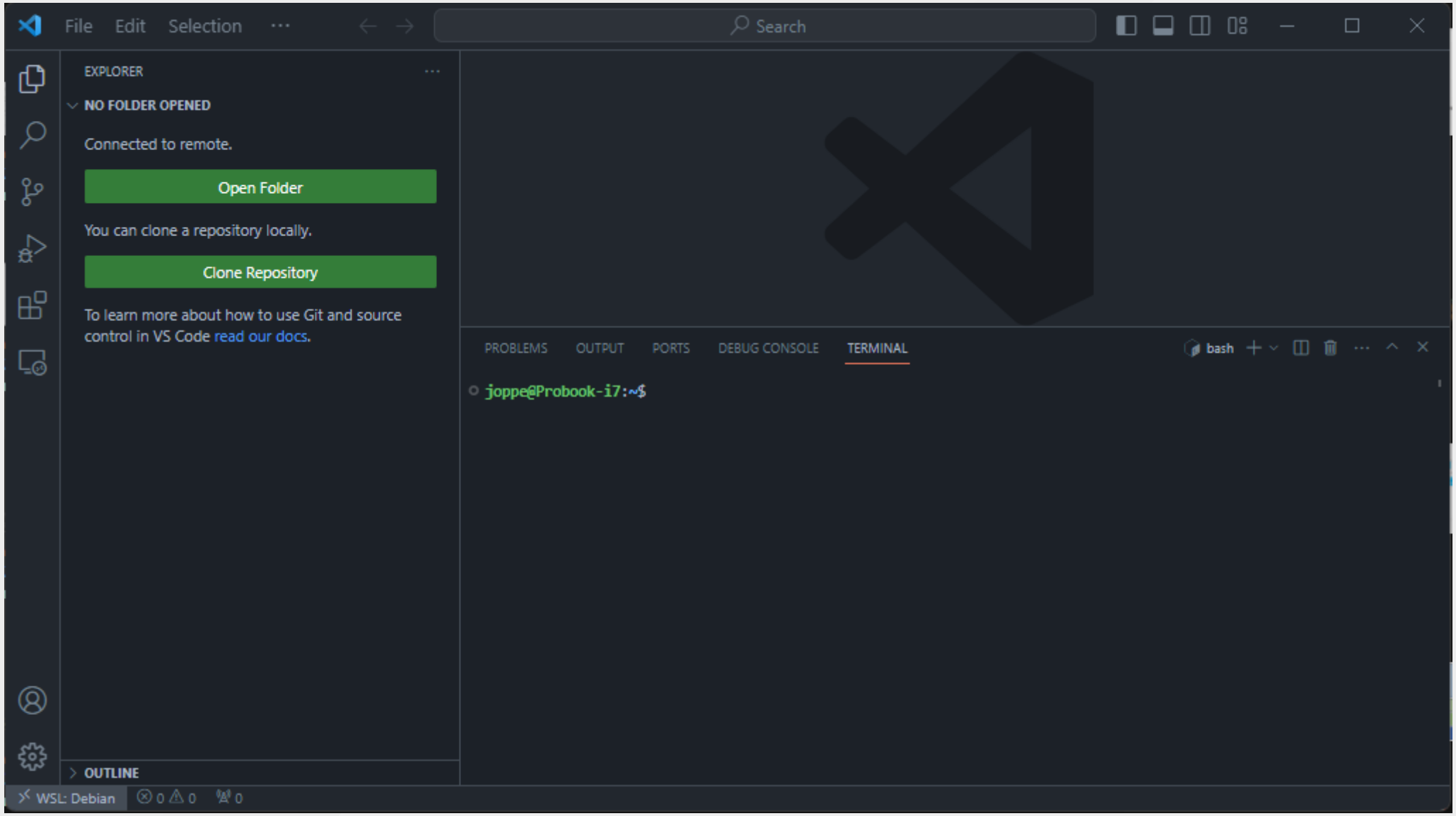
Or clone a repo in WSL

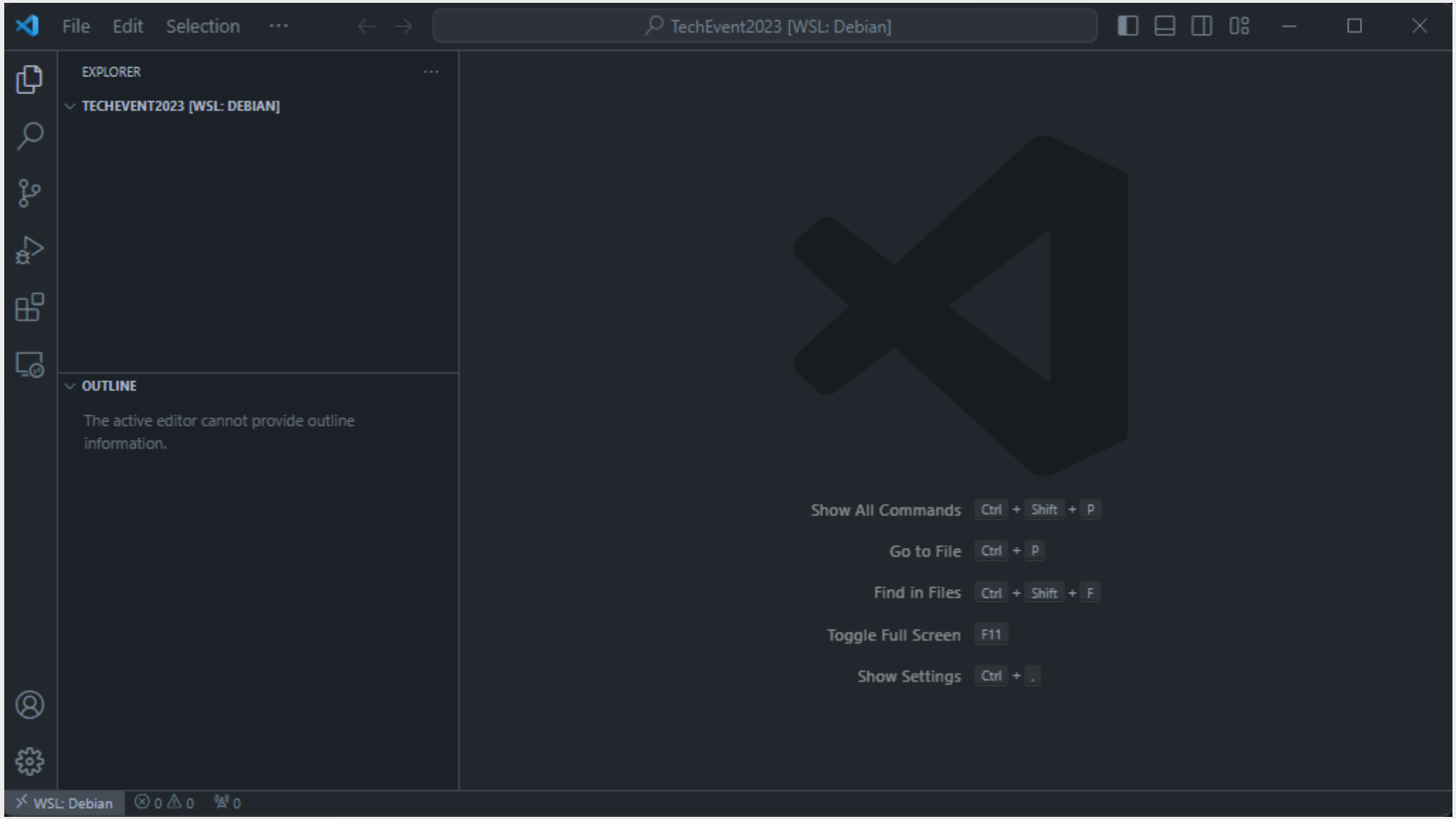
```
AzureAD+JoppeBlondel@Probook-i7 UCRT64 ~
$ wsl
joppe@Probook-i7:/mnt/c/msys64/home/JoppeBlondel$ cd
joppe@Probook-i7:~$ git clone http://somerepohere.git
```

Work from WSL: Open project in WSL




File
Edit
Selection
Search
REMOTE EXPLORER
WSL Targets
WSL TARGETS
Debian default distro
HELP AND FEEDBACK
Get Started
Read Documentation
Review Issues
Report Issue
Show All Commands Ctrl + Shift + P
Open File Ctrl + O
Open Folder Ctrl + K Ctrl + O
Open Recent Ctrl + R


File
Edit
Selection
Search
EXPLORER
NO FOLDER OPENED
Connected to remote.
Open Folder
You can clone a repository locally.
Clone Repository
To learn more about how to use Git and source control in VS Code read our docs.
PROBLEMS
OUTPUT
PORTS
DEBUG CONSOLE
TERMINAL
bash
joppe@Probook-i7:~$
OUTLINE
WSL: Debian


File
Edit
Selection
TechEvent2023 [WSL: Debian]
EXPLORER
TECHEVENT2023 [WSL: DEBIAN]
OUTLINE
The active editor cannot provide outline information.
Show All Commands Ctrl + Shift + P
Go to File Ctrl + P
Find in Files Ctrl + Shift + F
Toggle Full Screen F11
Show Settings Ctrl + ,
WSL: Debian

OK we're in... and now?

- `.devcontainer` directory with
  - `devcontainer.json` -> Definition of the devcontainer to use
  - `Dockerfile` -> Definition of the underlying Docker container

.devcontainer/devcontainer.json

minimal setup:

```
{
    // Give the devcontainer a name, optional
    "name": "TechEvent2023_A",
    // Use a docker container and point to the Dockerfile
    "build": {
        "dockerfile": "Dockerfile"
    }
}
```

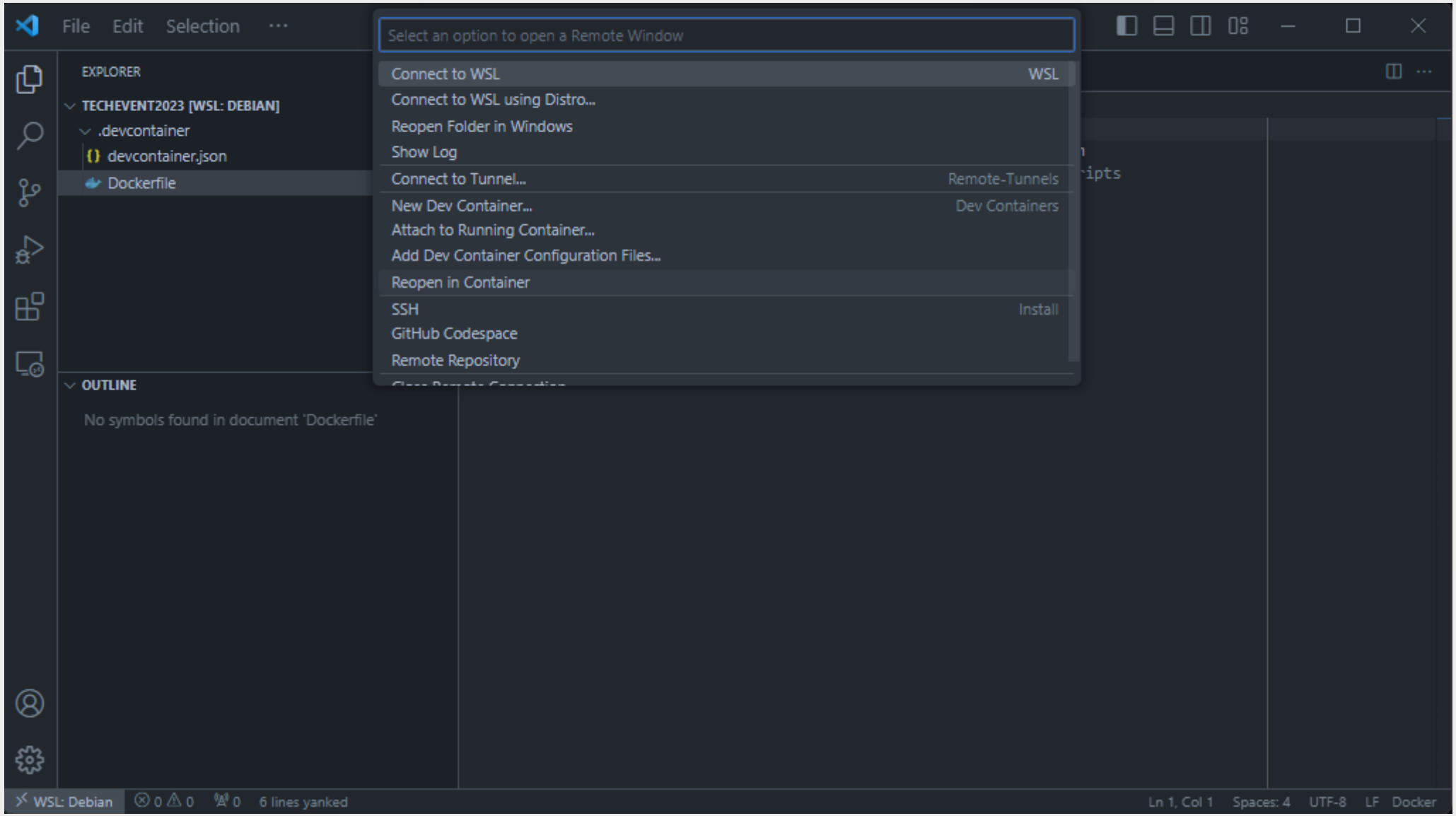
`.devcontainer/Dockerfile`

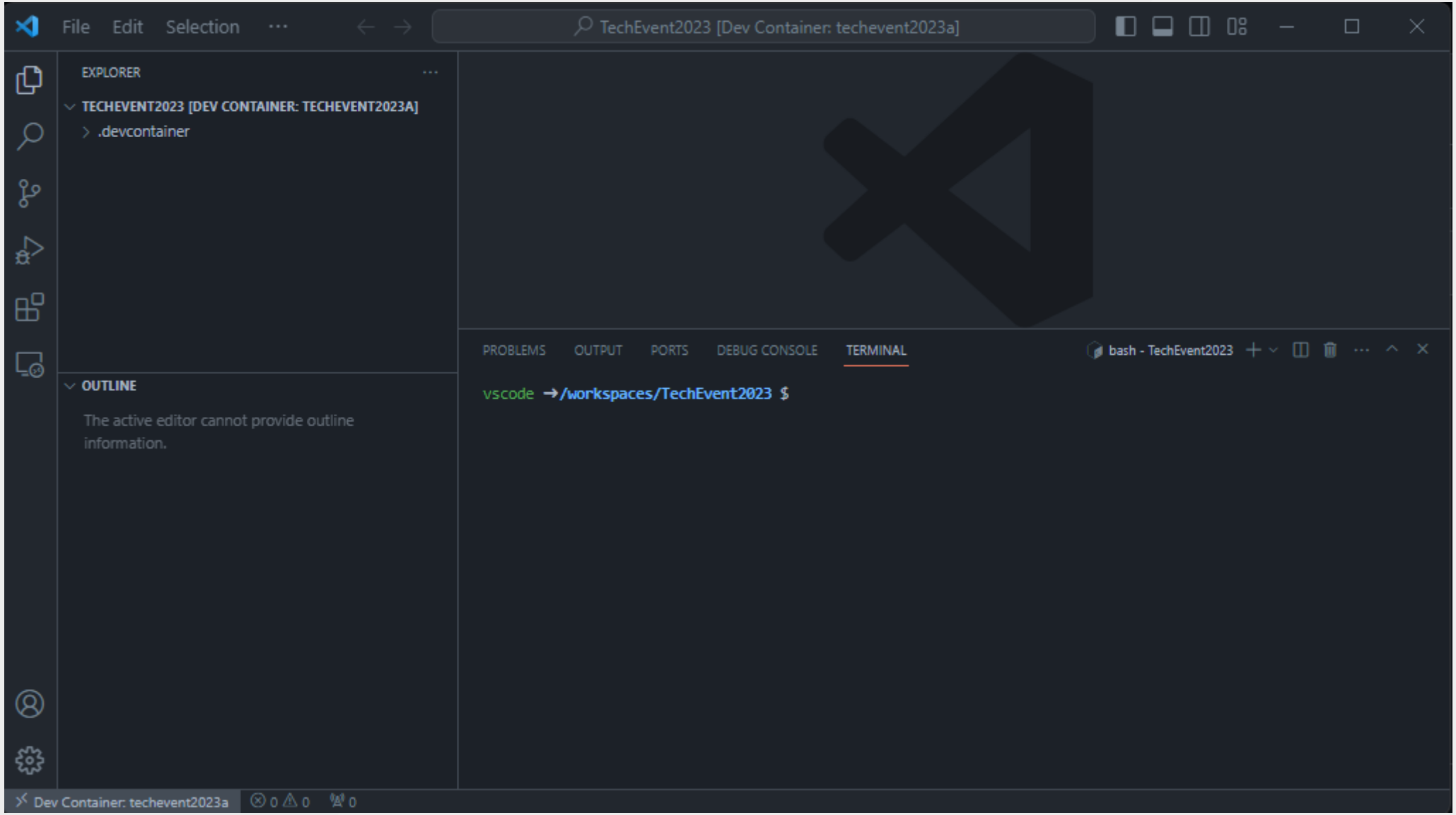
minimal setup with gcc and cmake:

```
# Base image: Microsoft's debian image
FROM mcr.microsoft.com/vscode/devcontainers/base:debian
# Things are ran from a script, apt needs this to run in scripts
ENV DEBIAN_FRONTEND=noninteractive
# Install needed software
RUN apt update -y && apt install -y build-essential cmake gdb
```

Microsofts images: Alpine, Ubuntu and Debian

... Beyond git, this image / Dockerfile includes zsh, Oh My Zsh!, a non-root vscode user with sudo access, and a set of common dependencies for development.


File
Edit
Selection
Select an option to open a Remote Window
Connect to WSL
WSL
Connect to WSL using Distro...
Reopen Folder in Windows
Show Log
Connect to Tunnel...
Remote-Tunnels
New Dev Container...
Dev Containers
Attach to Running Container...
Add Dev Container Configuration Files...
Reopen in Container
SSH
Install
GitHub Codespace
Remote Repository
EXPLORER
TECHEVENT2023 [WSL: DEBIAN]
.devcontainer
devcontainer.json
Dockerfile
OUTLINE
No symbols found in document 'Dockerfile'
WSL: Debian
6 lines yanked
Ln 1, Col 1
Spaces: 4
UTF-8
LF
Docker


File Edit Selection ···
TechEvent2023 [Dev Container: techevent2023a]
EXPLORER
TECHEVENT2023 [DEV CONTAINER: TECHEVENT2023A]
.devcontainer
OUTLINE
The active editor cannot provide outline information.
PROBLEMS OUTPUT PORTS DEBUG CONSOLE TERMINAL
bash - TechEvent2023
vscode →/workspaces/TechEvent2023 $
Dev Container: techevent2023a

Lets add a hello world

```c
// main.c
#include <stdio.h>
int main(int argc, char ** argv){
    printf("Hello World!\r\n");
    return 0;
}
```

```
# CMakeLists.txt
cmake_minimum_required(VERSION 3.10)
project(hello_world LANGUAGES C)
add_executable(app main.c)
```

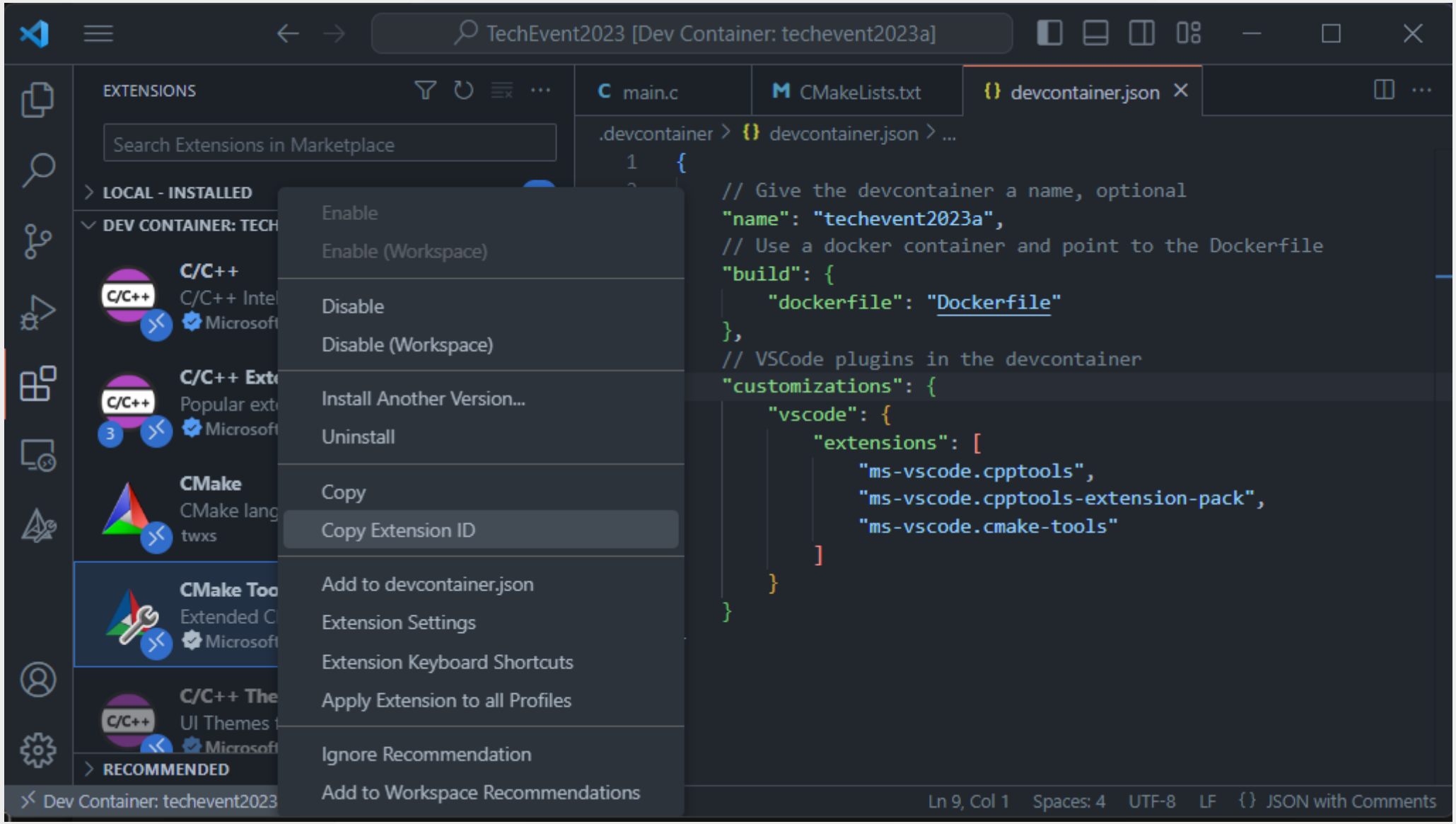
VSCode: plugins for everything

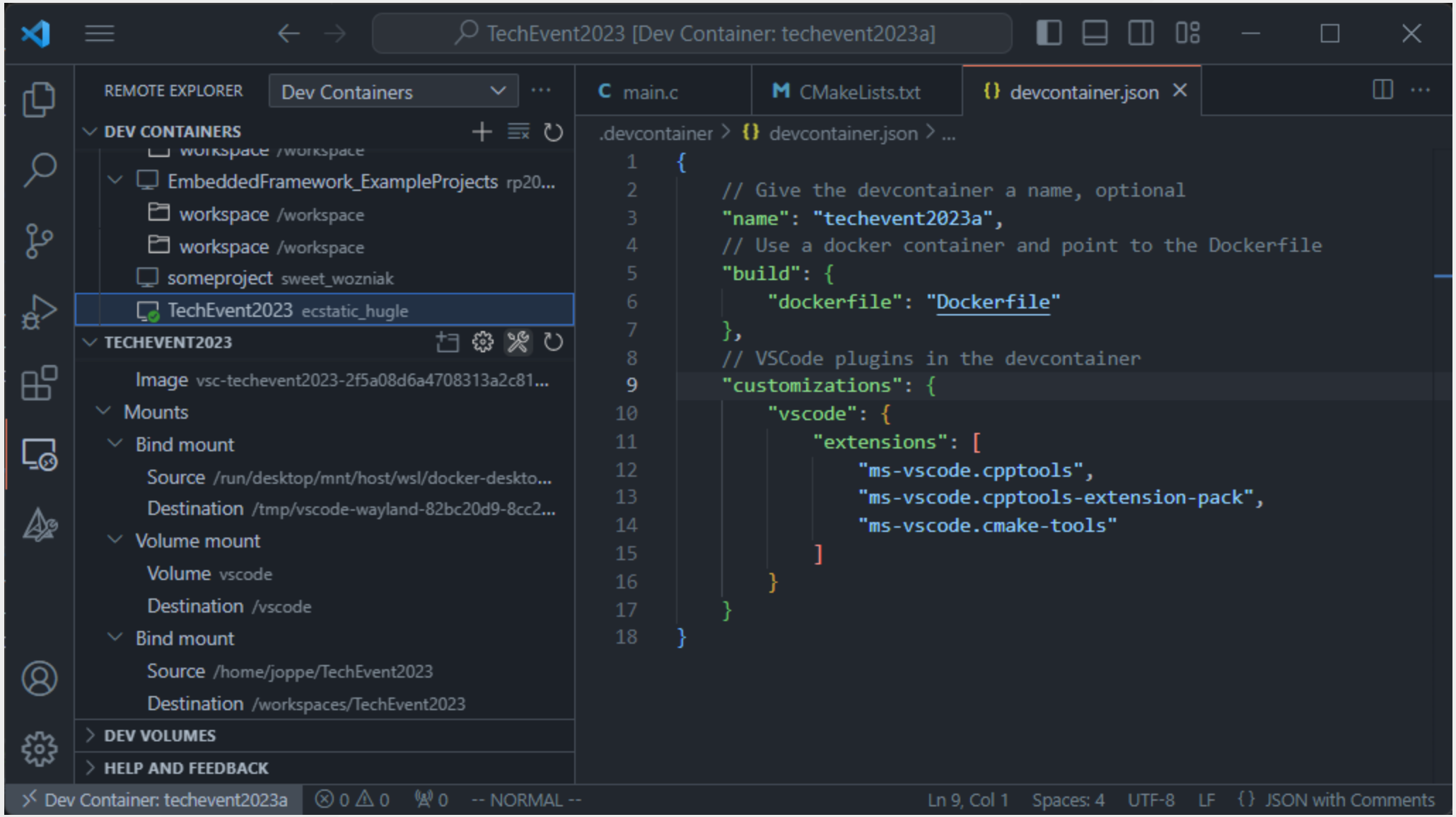
- With cmake installed we can build from the terminal but...
- `CMake Tools` plugin for VSCode for easy CMake build and debug functionality
- `C/C++` and `C/C++ Extension Pack` plugins for VSCode for C/C++ autocomplete

Add them to the devcontainer

add the following to `devcontainer.json`:

```
"customizations": {
    "vscode": {
        "extensions": [
            "ms-vscode.cpptools",
            "ms-vscode.cpptools-extension-pack",
            "ms-vscode.cmake-tools"
        ]
    }
}
```


TechEvent2023 [Dev Container: techevent2023a]
EXTENSIONS
Search Extensions in Marketplace
LOCAL - INSTALLED
DEV CONTAINER: TECH
C/C++
C/C++ Inte
Microsoft
C/C++ Exte
Popular ext
Microsoft
CMake
CMake lang
twxs
CMake Too
Extended C
Microsoft
C/C++ The
UI Themes
Microsoft
RECOMMENDED
Enable
Enable (Workspace)
Disable
Disable (Workspace)
Install Another Version...
Uninstall
Copy
Copy Extension ID
Add to devcontainer.json
Extension Settings
Extension Keyboard Shortcuts
Apply Extension to all Profiles
Ignore Recommendation
Add to Workspace Recommendations
main.c
CMakeLists.txt
devcontainer.json
.devcontainer > devcontainer.json > ...
{
// Give the devcontainer a name, optional
"name": "techevent2023a",
// Use a docker container and point to the Dockerfile
"build": {
"dockerfile": "Dockerfile"
},
// VSCode plugins in the devcontainer
"customizations": {
"vscode": {
"extensions": [
"ms-vscode.cpptools",
"ms-vscode.cpptools-extension-pack",
"ms-vscode.cmake-tools"
]
}
}
Dev Container: techevent2023
Ln 9, Col 1
Spaces: 4
UTF-8
LF
{} JSON with Comments


TechEvent2023 [Dev Container: techevent2023a]
REMOTE EXPLORER
Dev Containers
DEV CONTAINERS
EmbeddedFramework_ExampleProjects rp20...
workspace /workspace
workspace /workspace
someproject sweet_wozniak
TechEvent2023 ecstatic_hugle
TECHEVENT2023
Image vsc-techevent2023-2f5a08d6a4708313a2c81...
Mounts
Bind mount
Source /run/desktop/mnt/host/wsl/docker-deskto...
Destination /tmp/vscode-wayland-82bc20d9-8cc2...
Volume mount
Volume vscode
Destination /vscode
Bind mount
Source /home/joppe/TechEvent2023
Destination /workspaces/TechEvent2023
DEV VOLUMES
HELP AND FEEDBACK
main.c
CMakeLists.txt
devcontainer.json
.devcontainer > devcontainer.json > ...
{
// Give the devcontainer a name, optional
"name": "techevent2023a",
// Use a docker container and point to the Dockerfile
"build": {
"dockerfile": "Dockerfile"
},
// VSCode plugins in the devcontainer
"customizations": {
"vscode": {
"extensions": [
"ms-vscode.cpptools",
"ms-vscode.cpptools-extension-pack",
"ms-vscode.cmake-tools"
]
}
}
}
Dev Container: techevent2023a
0 0
0
-- NORMAL --
Ln 9, Col 1
Spaces: 4
UTF-8
LF
JSON with Comments

This is not embedded yet right?

- Just linux gcc
- Native executable debugging
- Things we miss
  - USB
  - USB <-> C debugging
  - C Debugger which is embedded aware
  - Maybe some nice embedded specific features in VSCode

USB

See readme in TechEvent repo

note: When using Ubuntu as WSL distro udev is already running

USB <-> C debugging

- Piece of software connecting to the physical debugger
- Implementing a GDB server which can set breakpoints, single step, etc
- Examples:
  - **openocd**
  - pyocd
  - black magic probe
  - (j)linkserver

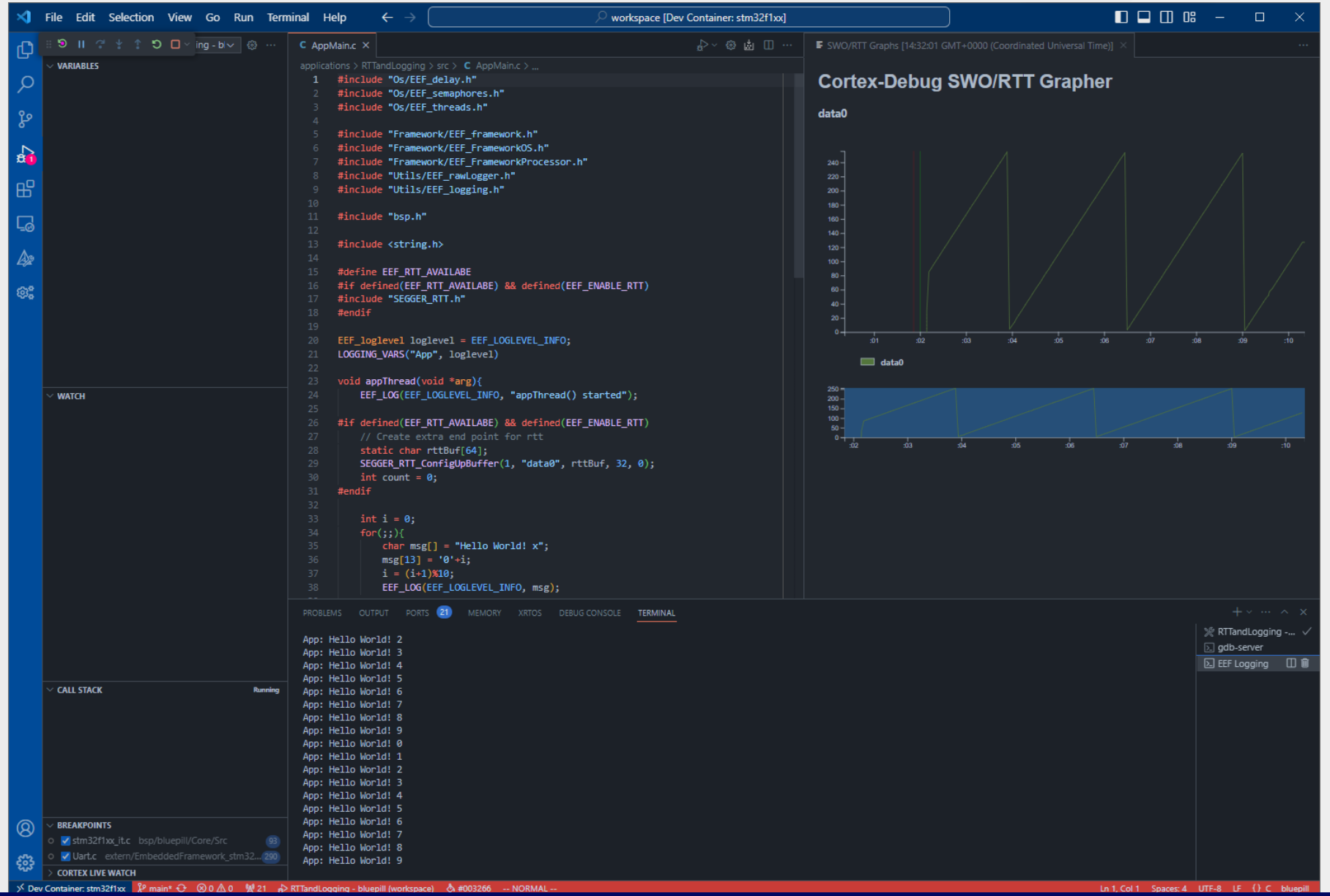
C Debugger which is embedded aware

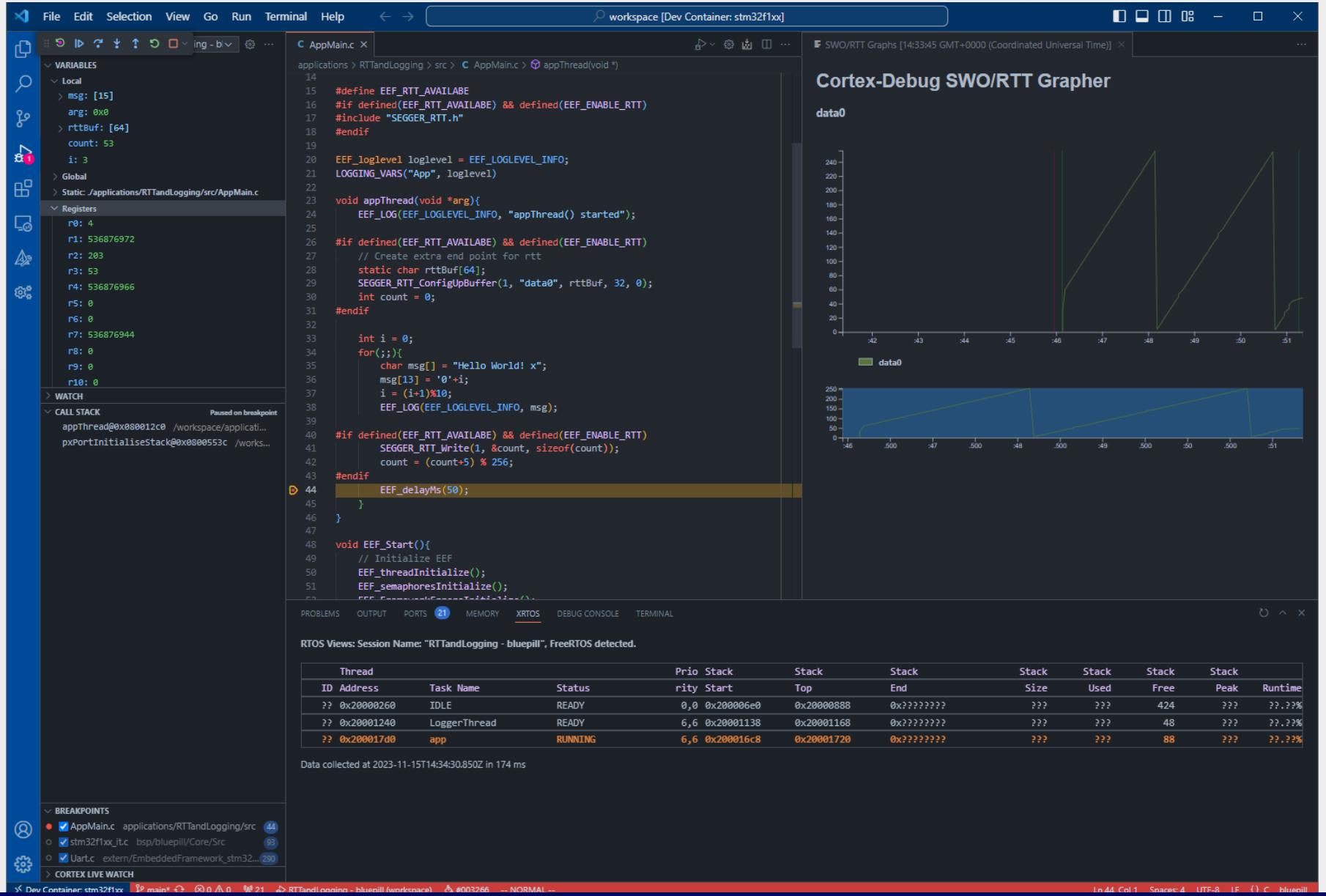
- arm-none-eabi-gdb
- riscv-gdb
- ....

All depricated now and combined to
gdb-multiarch

Nice embedded specific features

- cortex-debug (`marus25.cortex-debug`)
- device packages (`marus25.cortex-debug-dp-xxxx`)
- Peripheral Viewer (`mcu-debug.peripheral-viewer`)
- MemoryView (`mcu-debug.memory-view`)
- Embedded Tools (`ms-vscode.vscode-embedded-tools`)
- Task Explorer (`spmeesseman.vscode-taskexplorer`)


File Edit Selection View Go Run Terminal Help
workspace [Dev Container: stm32f1xx]
AppMain.c
applications > RTTandLogging > src > AppMain.c > ...
#include "Os/EEF_delay.h"
#include "Os/EEF_semaphores.h"
#include "Os/EEF_threads.h"
#include "Framework/EEF_framework.h"
#include "Framework/EEF_FrameworkOS.h"
#include "Framework/EEF_FrameworkProcessor.h"
#include "Utils/EEF_rawLogger.h"
#include "Utils/EEF_logging.h"
#include "bsp.h"
#include <string.h>
#define EEF_RTT_AVAILABE
#if defined(EEF_RTT_AVAILABE) && defined(EEF_ENABLE_RTT)
#include "SEGGER_RTT.h"
#endif
EEF_loglevel loglevel = EEF_LOGLEVEL_INFO;
LOGGING_VARS("App", loglevel)
void appThread(void *arg){
EEF_LOG(EEF_LOGLEVEL_INFO, "appThread() started");
#if defined(EEF_RTT_AVAILABE) && defined(EEF_ENABLE_RTT)
// Create extra end point for rtt
static char rttBuf[64];
SEGGER_RTT_ConfigUpBuffer(1, "data0", rttBuf, 32, 0);
int count = 0;
#endif
int i = 0;
for(;;){
char msg[] = "Hello World! x";
msg[13] = '0'+i;
i = (i+1)%10;
EEF_LOG(EEF_LOGLEVEL_INFO, msg);
SWO/RTT Graphs [14:32:01 GMT+0000 (Coordinated Universal Time)]
Cortex-Debug SWO/RTT Grapher
data0
data0
VARIABLES
WATCH
CALL STACK
Running
BREAKPOINTS
stm32f1xx_it.c bsp/bluepill/Core/Src
Uart.c extern/EmbeddedFramework_stm32...
CORTEX LIVE WATCH
PROBLEMS OUTPUT PORTS 21 MEMORY XRTOS DEBUG CONSOLE TERMINAL
App: Hello World! 2
App: Hello World! 3
App: Hello World! 4
App: Hello World! 5
App: Hello World! 6
App: Hello World! 7
App: Hello World! 8
App: Hello World! 9
App: Hello World! 0
App: Hello World! 1
App: Hello World! 2
App: Hello World! 3
App: Hello World! 4
App: Hello World! 5
App: Hello World! 6
App: Hello World! 7
App: Hello World! 8
App: Hello World! 9
RTTandLogging -...
gdb-server
EEF Logging
Dev Container: stm32f1xx
main*
RTTandLogging - bluepill (workspace)
-- NORMAL --
Ln 1, Col 1
Spaces: 4
UTF-8
LF
C
bluepill


File Edit Selection View Go Run Terminal Help
workspace [Dev Container: stm32f1xx]
AppMain.c
applications > RTTandLogging > src > AppMain.c > appThread(void *)
#define EEF_RTT_AVAILABE
#if defined(EEF_RTT_AVAILABE) && defined(EEF_ENABLE_RTT)
#include "SEGGER_RTT.h"
#endif
EEF_loglevel loglevel = EEF_LOGLEVEL_INFO;
LOGGING_VARS("App", loglevel)
void appThread(void *arg){
EEF_LOG(EEF_LOGLEVEL_INFO, "appThread() started");
#if defined(EEF_RTT_AVAILABE) && defined(EEF_ENABLE_RTT)
// Create extra end point for rtt
static char rttBuf[64];
SEGGER_RTT_ConfigUpBuffer(1, "data0", rttBuf, 32, 0);
int count = 0;
#endif
int i = 0;
for(;;){
char msg[] = "Hello World! x";
msg[13] = '0'+i;
i = (i+1)%10;
EEF_LOG(EEF_LOGLEVEL_INFO, msg);
#if defined(EEF_RTT_AVAILABE) && defined(EEF_ENABLE_RTT)
SEGGER_RTT_Write(1, &count, sizeof(count));
count = (count+5) % 256;
#endif
EEF_delayMs(50);
}
}
void EEF_Start(){
// Initialize EEF
EEF_threadInitialize();
EEF_semaphoresInitialize();
VARIABLES
Local
msg: [15]
arg: 0x0
rttBuf: [64]
count: 53
i: 3
Global
Static: /applications/RTTandLogging/src/AppMain.c
Registers
r0: 4
r1: 536876972
r2: 203
r3: 53
r4: 536876966
r5: 0
r6: 0
r7: 536876944
r8: 0
r9: 0
r10: 0
WATCH
CALL STACK
Paused on breakpoint
appThread@0x080012c0 /workspace/applicati...
pxPortInitialiseStack@0x0800553c /works...
BREAKPOINTS
AppMain.c applications/RTTandLogging/src 44
stm32f1xx_it.c bsp/bluepill/Core/Src 93
Uart.c extern/EmbeddedFramework_stm32... 290
CORTEX LIVE WATCH
SWO/RTT Graphs [14:33:45 GMT+0000 (Coordinated Universal Time)]
Cortex-Debug SWO/RTT Grapher
data0
data0
PROBLEMS OUTPUT PORTS 21 MEMORY XRTOS DEBUG CONSOLE TERMINAL
RTOS Views: Session Name: "RTTandLogging - bluepill", FreeRTOS detected.
ID Thread Address Task Name Status Prio rity Stack Start Stack Top Stack End Stack Size Stack Used Stack Free Stack Peak Runtime
?? 0x20000260 IDLE READY 0,0 0x200006e0 0x20000888 0x???????? ??? ??? 424 ??? ??.??%
?? 0x20001240 LoggerThread READY 6,6 0x20001138 0x20001168 0x???????? ??? ??? 48 ??? ??.??%
?? 0x200017d0 app RUNNING 6,6 0x200016c8 0x20001720 0x???????? ??? ??? 88 ??? ??.??%
Data collected at 2023-11-15T14:34:30.850Z in 174 ms