How To Run CUDA C or C++ on Microsoft Visual Studio.

medium.com/@p190036/how-to-run-cuda-c-or-c-on-microsoft-visual-studio-c6398892bc10

Muhammad Abdullah April 7, 2022

Muhammad Abdullah

Apr 7

·

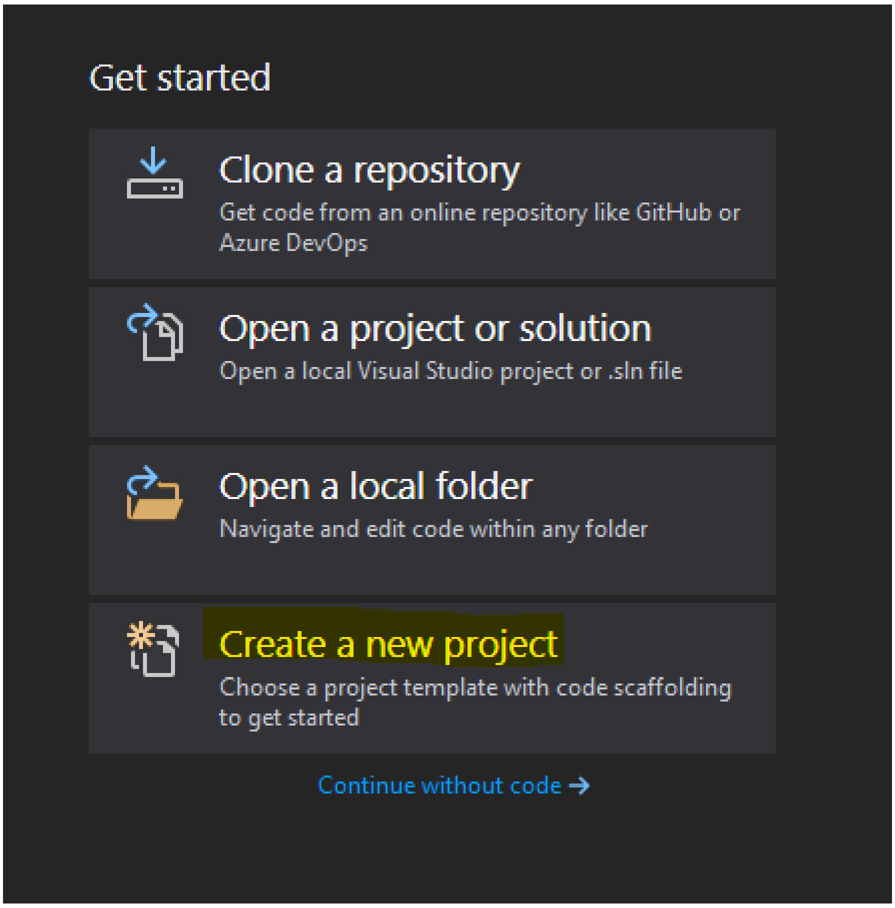
3 min read

CUDA code doesn’t run on AMD CPU or Intel HD graphics unless you have a NVIDIA Hardware inside your Machine. If you don’t have NVIDIA hardware then you need to run CUDA code on Google COLAB. You can check how to do that on the following link. How To Run CUDA C or C++ on Google Colab. | by Muhammad Abdullah | Apr, 2022 | Medium

Step1: InstallMicrosoft Visual Studio from Visual Studio: IDE and Code Editor for Software Developers and Teams (microsoft.com)

Step 2: Install CUDA Toolkit from CUDA Toolkit 11.6 Update 2 Downloads | NVIDIA Developer

Step 3: Open Visual Studio, and create new project



Get started

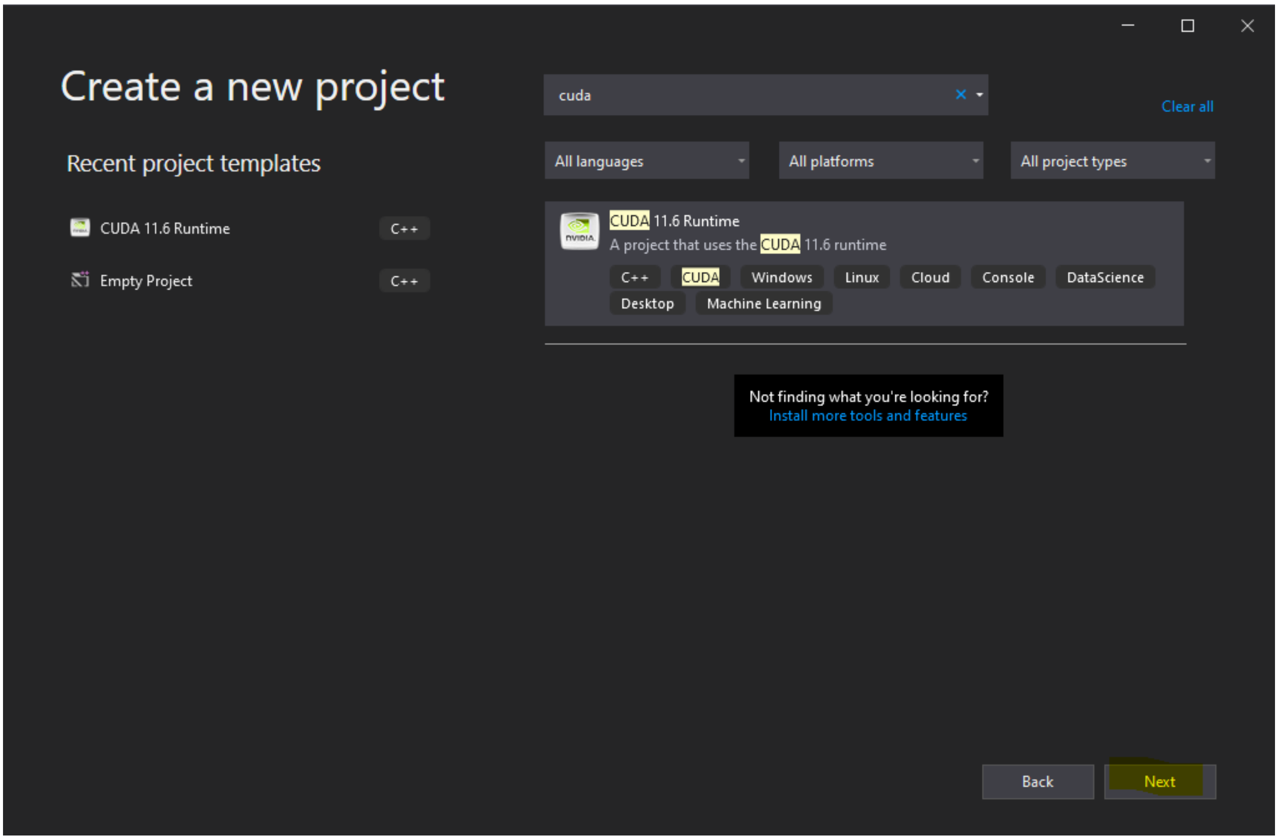
Clone a repository
Get code from an online repository like GitHub or Azure DevOps

Open a project or solution
Open a local Visual Studio project or .sln file

Open a local folder
Navigate and edit code within any folder

Create a new project
Choose a project template with code scaffolding to get started

Step 4: After clicking on new project search for CUDA select that and click on next.



Create a new project

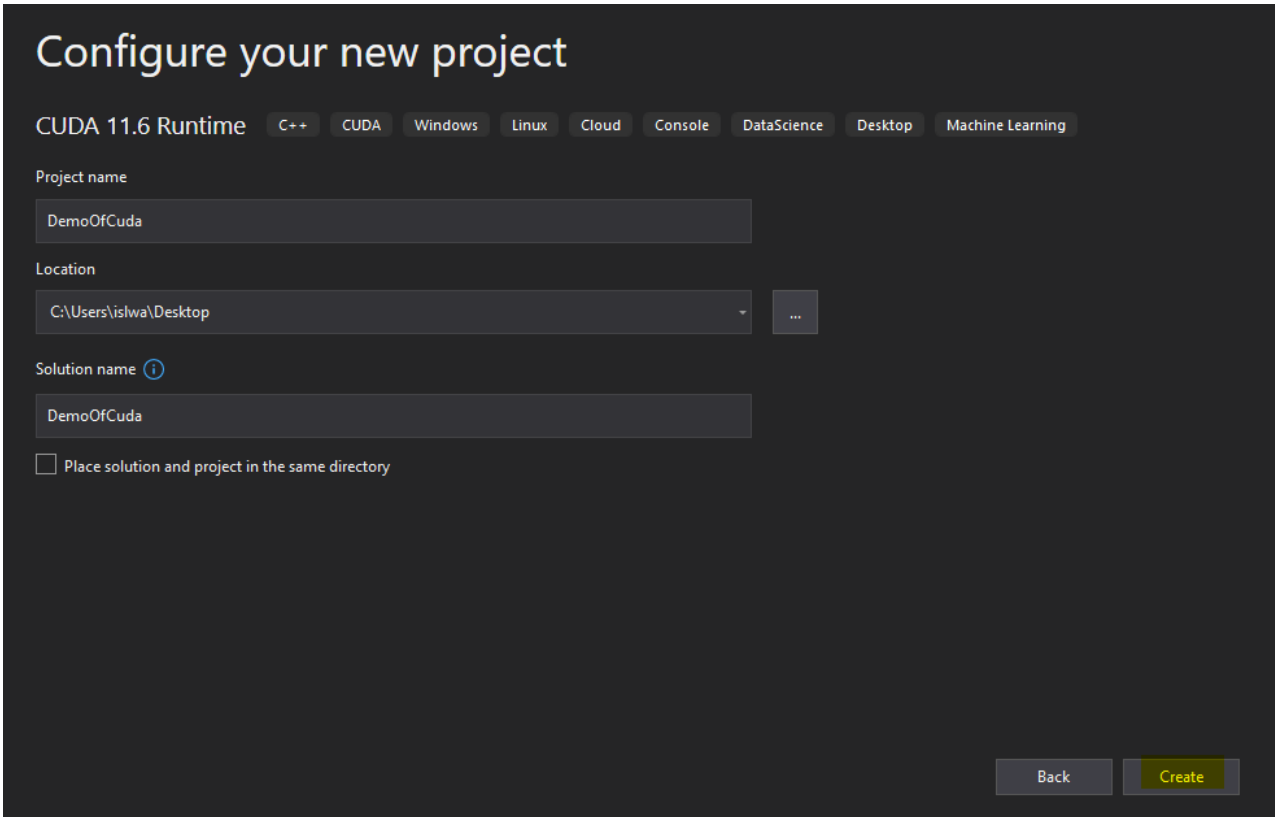
Recent project templates

Search: cuda

CUDA 11.6 Runtime
A project that uses the CUDA 11.6 runtime

Not finding what you're looking for?
Install more tools and features

Step 5: Configure your project and click on create



Configure your new project

CUDA 11.6 Runtime

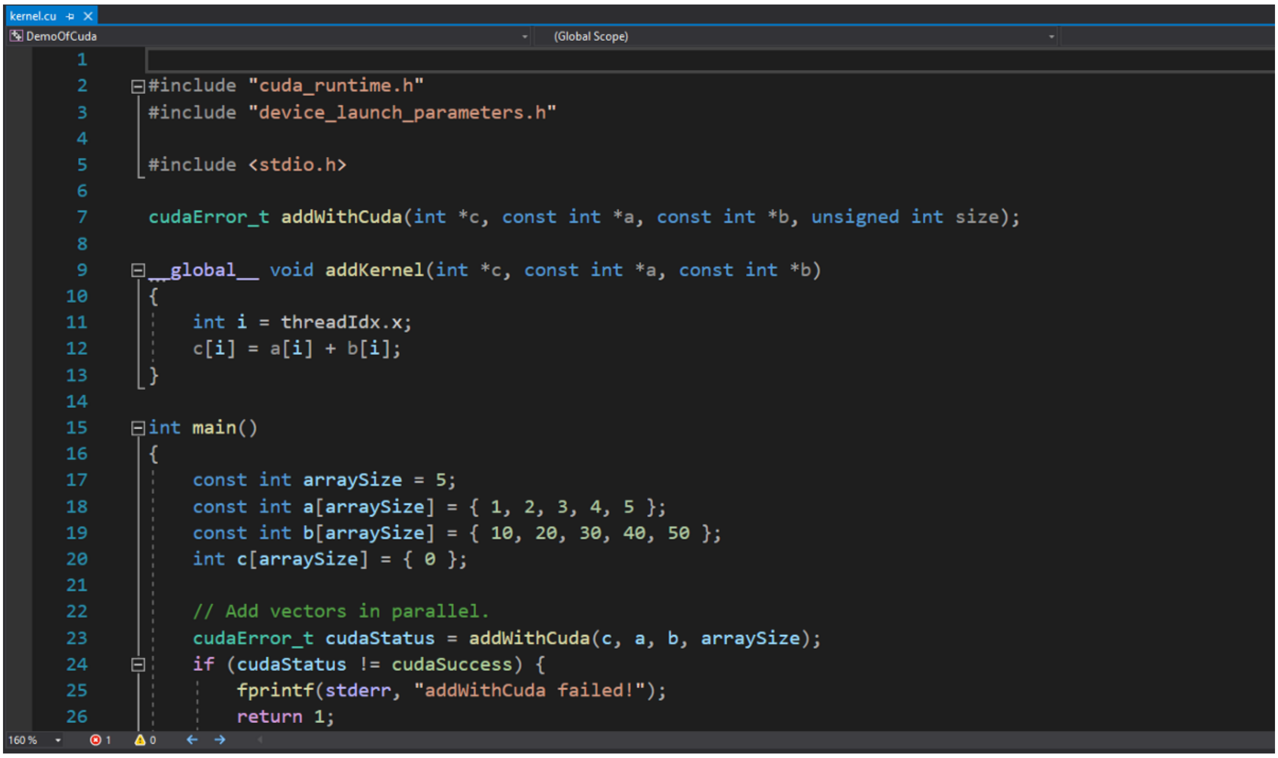
Project name: DemoOfCuda

Location: C:\Users\islwa\Desktop

Solution name: DemoOfCuda

□ Place solution and project in the same directory

After creating your project you will have a pre coded CUDA program you can remove that, and code your own program there.



```

#include "cuda_runtime.h"
#include "device_launch_parameters.h"

#include <stdio.h>

cudaError_t addWithCuda(int *c, const int *a, const int *b, unsigned int size);

__global__ void addKernel(int *c, const int *a, const int *b)
{
    int i = threadIdx.x;
    c[i] = a[i] + b[i];
}

int main()
{
    const int arraySize = 5;
    const int a[arraySize] = { 1, 2, 3, 4, 5 };
    const int b[arraySize] = { 10, 20, 30, 40, 50 };
    int c[arraySize] = { 0 };

    // Add vectors in parallel.
    cudaError_t cudaStatus = addWithCuda(c, a, b, arraySize);
    if (cudaStatus != cudaSuccess) {
        fprintf(stderr, "addWithCuda failed!");
        return 1;
```

Step 6: To check if your code is working you can do that by running pre coded CUDA program or code your own program.

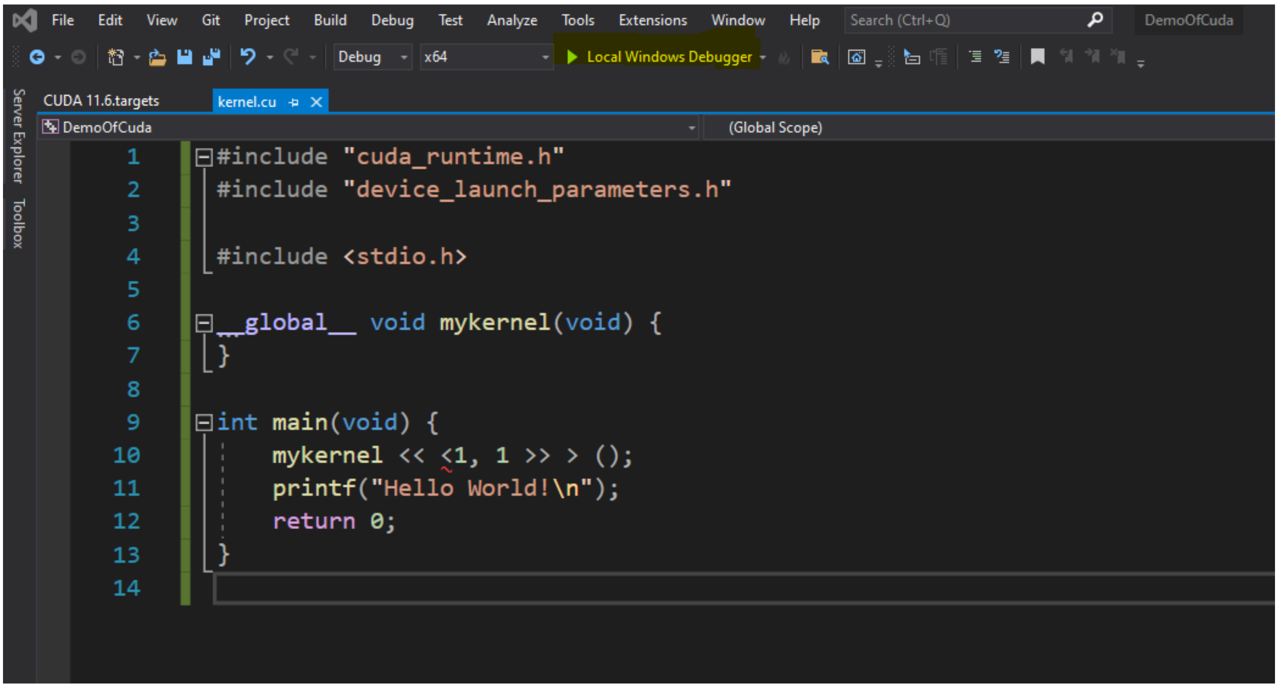
Run the following example to check

```
#include “cuda_runtime.h”
#include “device_launch_parameters.h”

#include <stdio.h>

__global__ void mykernel(void) {
}

int main(void) {
mykernel<<<1,1>>>();
printf(“Hello World!\n”);
return 0;
}
```



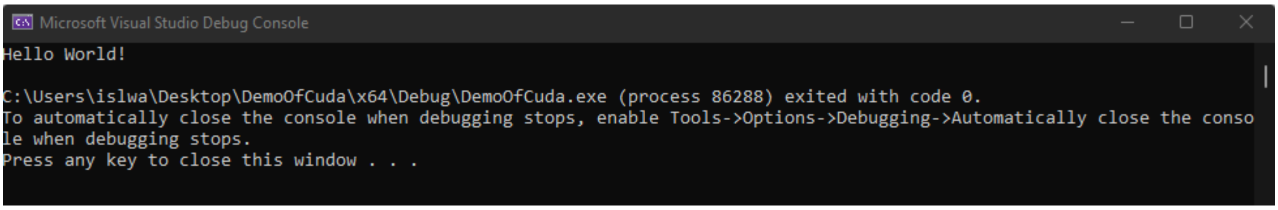
```
#include "cuda_runtime.h"
#include "device_launch_parameters.h"

#include <stdio.h>

__global__ void mykernel(void) {
}

int main(void) {
    mykernel << <1, 1 >> > ();
    printf("Hello World!\n");
    return 0;
}
```

Output should be Hello World!



```
Hello World!

C:\Users\islwa\Desktop\DemoOfCuda\x64\Debug\DemoOfCuda.exe (process 86288) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the conso
le when debugging stops.
Press any key to close this window . . .
```

If you’re interested in seeing more examples of CUDA code you can see them on the following link NVIDIA/cuda-samples: Samples for CUDA Developers which demonstrates features in CUDA Toolkit (github.com)