```
!git clone https://github.com/andreinechaev/nvcc4jupyter
!pip install git+file:/content/nvcc4jupyter
```

```
Cloning into 'nvcc4jupyter'...
remote: Enumerating objects: 48, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 48 (delta 0), reused 0 (delta 0), pack-reused 45
Unpacking objects: 100% (48/48), 8.29 KiB | 653.00 KiB/s, done.
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting git+file:/content/nvcc4jupyter
  Cloning file:///content/nvcc4jupyter to /tmp/pip-req-build-ywj95tds
  Running command git clone --filter=blob:none --quiet file:///content/nvcc4jupyter /tmp/pip-req-build-ywj95tds
  warning: filtering not recognized by server, ignoring
  Resolved file:///content/nvcc4jupyter to commit aac710a35f52bb78ab34d2e52517237941399eff
  Preparing metadata (setup.py) ... done
Building wheels for collected packages: NVCCPlugin
  Building wheel for NVCCPlugin (setup.py) ... done
  Created wheel for NVCCPlugin: filename=NVCCPlugin-0.0.2-py3-none-any.whl size=4287 sha256=c1af8a37295e912278c02ee73674980ac499df86432c
  Stored in directory: /tmp/pip-ephem-wheel-cache-hh5e74kd/wheels/fd/06/13/b58cc0bebf3cd854f673ad262798e45f5ab5a6624b569b745b
Successfully built NVCCPlugin
Installing collected packages: NVCCPlugin
Successfully installed NVCCPlugin-0.0.2
```

+ Code + Text

```
%load_ext nvcc_plugin
```

```
created output directory at /content/src
Out bin /content/result.out
```

```
%%cu
#include <cuda_runtime.h>
#include <iostream>
#define N 1000000

using namespace std;

__global__ void add(int *a, int *b, int *c)
{
    int i = blockIdx.x * blockDim.x + threadIdx.x;
    if (i < N)
    {
        c[i] = a[i] + b[i];
    }
}

int main()
{
    // host vectors - Host means cpu
    int *a, *b, *c;

    // device vectors - Device means gpu
    int *dev_a, *dev_b, *dev_c;

    int size = N * sizeof(int);

    // Allocate memory on host
    a = (int *)malloc(size);
    b = (int *)malloc(size);
    c = (int *)malloc(size);

    // Allocate memory on device
    cudaMalloc((void **)&dev_a, size);
    cudaMalloc((void **)&dev_b, size);
    cudaMalloc((void **)&dev_c, size);

    // Initialize host vectors
    for (int i = 0; i < N; i++)
    {
        a[i] = i;
        b[i] = i * 2;
    }

    // Copy host vectors to device
    cudaMemcpy(dev_a, a, size, cudaMemcpyHostToDevice);
    cudaMemcpy(dev_b, b, size, cudaMemcpyHostToDevice);
```

```
    // Launch kernel with 1 block and 1024 threads per block
    add<<<1, 1024>>>(dev_a, dev_b, dev_c);

    // Copy result from device to host
    cudaMemcpy(c, dev_c, size, cudaMemcpyDeviceToHost);

    cout<<"first 10 values of Array a = ";
    for (int i = 0; i < 10; i++)
    {
        cout << a[i] << " ";
    }
    cout<<endl;

    cout<<"first 10 values of Array b = ";
    for (int i = 0; i < 10; i++)
    {
        cout << b[i] << " ";
    }
    cout<<endl;

    // Print the first 10 values of the result
    cout<<"first 10 values of Array c = ";
    for (int i = 0; i < 10; i++)
    {
        cout << c[i] << " ";
    }
    cout<<endl;

    // Free memory
    cudaFree(dev_a);
    cudaFree(dev_b);
    cudaFree(dev_c);
    free(a);
    free(b);
    free(c);

    return 0;
}
```

```
 first 10 values of Array a = 0 1 2 3 4 5 6 7 8 9
 first 10 values of Array b = 0 2 4 6 8 10 12 14 16 18
 first 10 values of Array c = 0 3 6 9 12 15 18 21 24 27
```

## Steps to Create & run this program

1. Change runtime type to gpu path = Runtime > change runtime type > Hardware accelerator > GPU
2. Setup of Cuda:

    1. Install Cuda

    ```
    !git clone https://github.com/andreinechaev/nvcc4jupyter
    !pip install git+file:/content/nvcc4jupyter
    ```

    2. Load nvcc plugin

    ```
    %load_ext nvcc_plugin
    ```

3. Enter your program && always add %%cu as first line of code to recognize it's cuda program