This example is taken from <https://www.tensorflow.org/api_guides/cc/guide>

// tensorflow/cc/example/example.cc  
  
#include "tensorflow/cc/client/client\_session.h"  
#include "tensorflow/cc/ops/standard\_ops.h"  
#include "tensorflow/core/framework/tensor.h"  
  
int main() {  
  using namespace tensorflow;  
  using namespace tensorflow::ops;  
  Scope root = Scope::NewRootScope();  
  // Matrix A = [3 2; -1 0]  
  auto A = Const(root, { {3.f, 2.f}, {-1.f, 0.f} });  
  // Vector b = [3 5]  
  auto b = Const(root, { {3.f, 5.f} });  
  // v = Ab^T  
  auto v = MatMul(root.WithOpName("v"), A, b, MatMul::TransposeB(true));  
  std::vector<Tensor> outputs;  
  ClientSession session(root);  
  // Run and fetch v  
  TF\_CHECK\_OK(session.Run({v}, &outputs));  
  // Expect outputs[0] == [19; -3]  
  LOG(INFO) << outputs[0].matrix<float>();  
  return 0;  
}

The “ClientSession” object defined in first include statement

#include "tensorflow/cc/client/client\_session.h"

lets the caller, **the main function** in this case to drive the evaluation of the **tensorflow graph** constructed with C++ API.

------------------------------------------------------------------------------------------------------------------------------------------

The corresponding line of code is

ClientSession session(root);

That can be created only when the above header file is included.

This can take three function calls:

ClientSession(const Scope& scope) - New session is created to evaluate the graph in *scope* by connecting to the TensorFlow runtime specified by *none*.

There are two more ways of calling ClientSession.

ClientSession(const Scope& scope, const string& target) – New session is created to evaluate the graph in *scope* by connecting to the TensorFlow runtime specified by *target*.

ClientSession(const Scope& scope, const SessionOptions& session\_options) – New session is created to evaluate the graph in *scope* and configure it with session\_options.

The way session runs with multiple calls can also be seen in the same header file.

------------------------------------------------------------------------------------------------------------------------------------------

The

#include "tensorflow/cc/ops/standard\_ops.h"

includes all the header files of the operations that can be performed.

------------------------------------------------------------------------------------------------------------------------------------------

The header file

#include "tensorflow/core/framework/tensor.h"

must be included to create a tensor in line

std::vector<Tensor> outputs;

------------------------------------------------------------------------------------------------------------------------------------------

**Support Questions and Answers – Please read them – they are useful:**

**What is Scope?**

Let us use these two lines to understand concept of scope.

using namespace tensorflow::ops;

A scope object is a container for Tensorflow Op properties. Op constructors requires the scope as the mandatory first argument and constructs Op. The op constructed takes the properties of the object.

*TensorFlow Operations*, also known as *Ops*, are nodes that perform computations on or with Tensor objects. After computation, they return zero or more tensors, which can be used by other Ops later in the graph.

Now root is a node that can perform .

Scope root = Scope::NewRootScope();

A new scope called root is created using the above statement.

The Op-constructor functions must check the scope's status by calling the ok() method before proceeding to construct the op.

See [this](https://www.tensorflow.org/api_docs/cc/class/tensorflow/scope) for more. Information related to Child scope and the functions that a scope supports is discussed here.

------------------------------------------------------------------------------------------------------------------------------------------

**What is TensorFlow runtime?**

Tensorflow runtime has things to do with enhancing the performance. The TF runtime parallelizes the graph execution across many dimensions. These options can be set as second parameter if either of the last two calls are made.

------------------------------------------------------------------------------------------------------------------------------------------

**What is Status?**

Status of a call is done using the status

This line

TF\_CHECK\_OK(session.Run({v}, &outputs)); has TF\_CHECK\_OK that is defined as macro in the file below header file tells the same thing.

<https://github.com/tensorflow/tensorflow/blob/master/tensorflow/core/lib/core/status.h>

------------------------------------------------------------------------------------------------------------------------------------------

Status can be used as a return type as well.

For example,

return Status::OK();

is a valid argument.

------------------------------------------------------------------------------------------------------------------------------------------

The function call

Status load\_graph\_status = LoadGraph(graph\_path, &session);

and the

load\_graph\_status.ok()

can be used as condition in any of the conditional operators and errors can be logged. The function LoadGraph has the below statement as the potential return argument.

return Status::OK();

------------------------------------------------------------------------------------------------------------------------------------------

In our example, the line

TF\_CHECK\_OK(session.Run({v}, &outputs));

expects the session.run to run v and get outputs. The very next line

LOG(INFO) << outputs[0].matrix<float>();

is a C++ syntax for logging and returning the values.

------------------------------------------------------------------------------------------------------------------------------------------

The two lines

  TF\_CHECK\_OK(session.Run({v}, &outputs));  
  // Expect outputs[0] == [19; -3]  
  LOG(INFO) << outputs[0].matrix<float>();

can be rewritten if we need some error as below:

  if(!TF\_CHECK\_OK(session.Run({v}, &outputs));)  
  // Expect outputs[0] == [19; -3]  
  LOG(INFO) << outputs[0].matrix<float>();

else

LOG(ERROR) << “THE EXECUTION DIDN’T RETURN THE EXPECTED VALUES”;

------------------------------------------------------------------------------------------------------------------------------------------

The line

root.WithOpName("v") returns the new scope with name “root/v”.

Now copy the code below – (same as the code in first page) in tensorflow/cc/example/example.cc

// tensorflow/cc/example/example.cc  
  
#include "tensorflow/cc/client/client\_session.h"  
#include "tensorflow/cc/ops/standard\_ops.h"  
#include "tensorflow/core/framework/tensor.h"  
  
int main() {  
  using namespace tensorflow;  
  using namespace tensorflow::ops;  
  Scope root = Scope::NewRootScope();  
  // Matrix A = [3 2; -1 0]  
  auto A = Const(root, { {3.f, 2.f}, {-1.f, 0.f} });  
  // Vector b = [3 5]  
  auto b = Const(root, { {3.f, 5.f} });  
  // v = Ab^T  
  auto v = MatMul(root.WithOpName("v"), A, b, MatMul::TransposeB(true));  
  std::vector<Tensor> outputs;  
  ClientSession session(root);  
  // Run and fetch v  
  TF\_CHECK\_OK(session.Run({v}, &outputs));  
  // Expect outputs[0] == [19; -3]  
  LOG(INFO) << outputs[0].matrix<float>();  
  return 0;  
}

Also place the BUILD file with the contents as shown below.

load("//tensorflow:tensorflow.bzl", "tf\_cc\_binary")  
  
tf\_cc\_binary(  
    name = "example",  
    srcs = ["example.cc"],  
    deps = [  
        "//tensorflow/cc:cc\_ops",  
        "//tensorflow/cc:client\_session",  
        "//tensorflow/core:tensorflow",  
    ],  
)

You should run

./configure in build sandbox

and then

bazel run -c opt //tensorflow/cc/example:example

Here are some of the properties controlled by a Scope object:

* Operation names
* Set of control dependencies for an operation
* Device placement for an operation
* Kernel attribute for an operation

The first parameter for all operation constructors is always a Scope object. Tensor inputs and mandatory attributes form the rest of the arguments.

auto m = MatMul(scope, a, b, MatMul::TransposeA(true).TransposeB(true));

auto is required and is the recommended way of constructing operations.

m is the object returned by the operation constructor.

MatMul is the operation constructor.

Scope is the first argument to be passed every time to the Op constructors.

a and b are the necessary arguments.

The fourth argument is optional, and the syntax is as shown.

See more about them online.

Scope root = Scope::NewRootScope();  
auto a = Placeholder(root, DT\_INT32);  
// [3 3; 3 3]  
auto b = Const(root, 3, {2, 2});  
auto c = Add(root, a, b);  
ClientSession session(root);  
std::vector<Tensor> outputs;  
  
// Feed a <- [1 2; 3 4]  
session.Run({ {a, { {1, 2}, {3, 4} } } }, {c}, &outputs);  
// outputs[0] == [4 5; 6 7]

This example shows the most of it.

Interpret on your own.

Read more about it here. [[redundant link](https://www.tensorflow.org/api_guides/cc/guide)]