C++ the serverless way with Cloud Functions

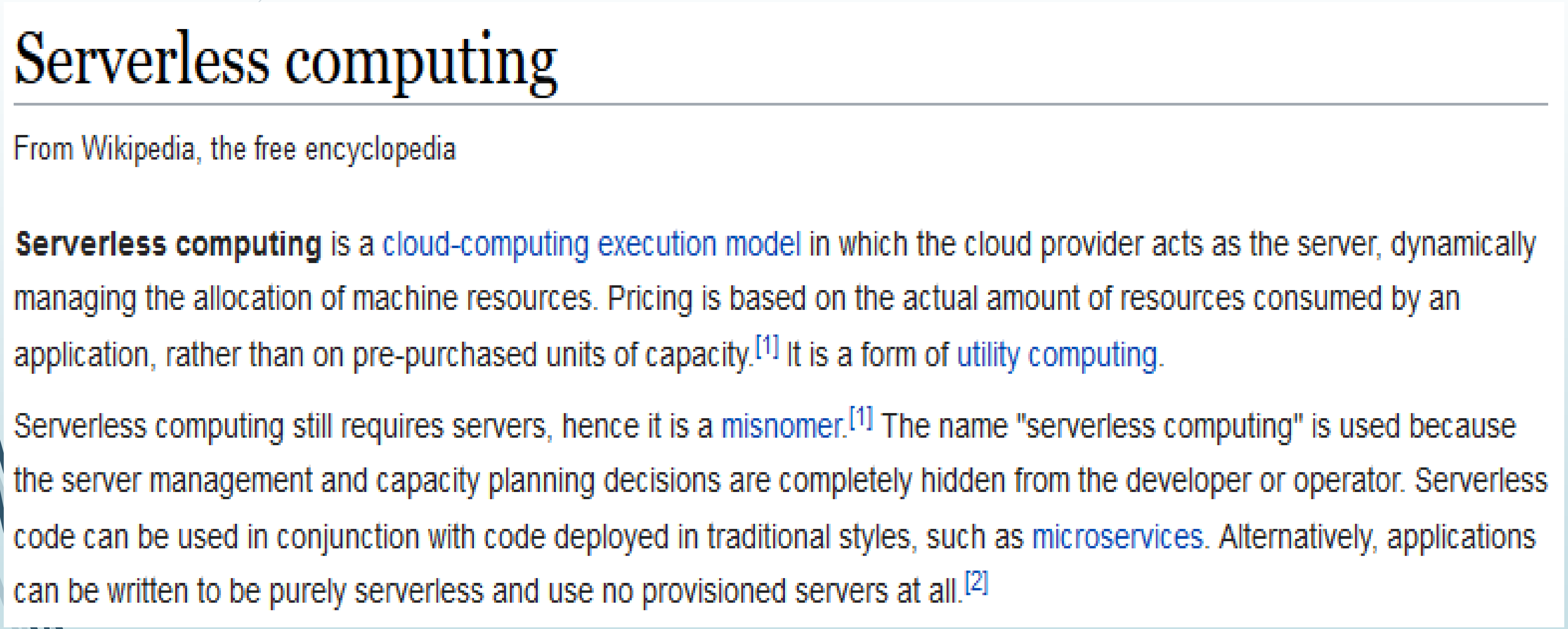
Runcy Oommen

@runcyoommen

/roommen

https://runcy.me

Serverless - Recap/Quick Intro



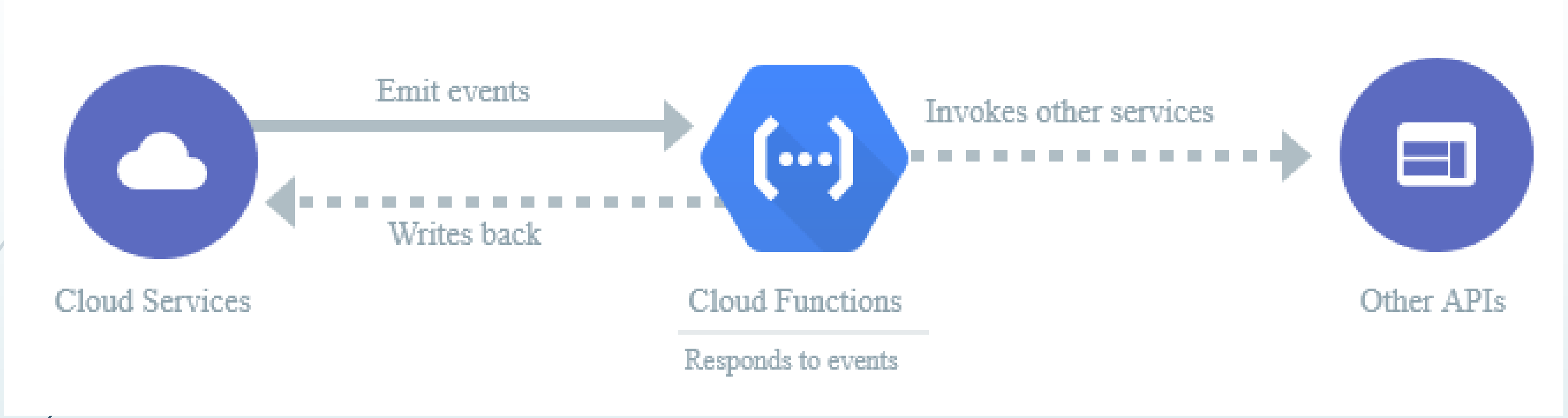
Serverless computing

From Wikipedia, the free encyclopedia

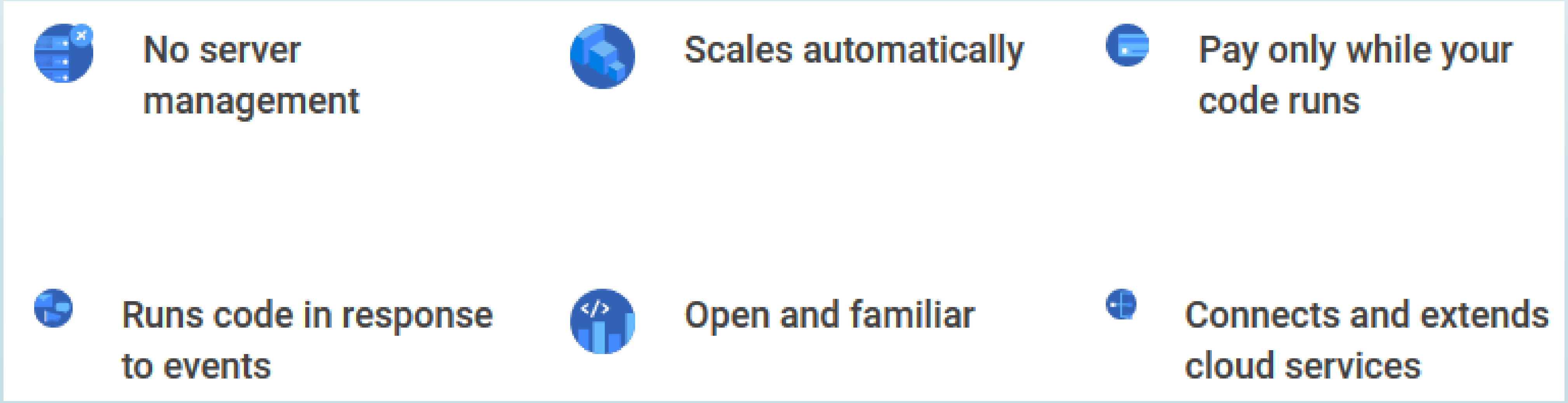
Serverless computing is a cloud-computing execution model in which the cloud provider acts as the server, dynamically managing the allocation of machine resources. Pricing is based on the actual amount of resources consumed by an application, rather than on pre-purchased units of capacity.[1] It is a form of utility computing.

Serverless computing still requires servers, hence it is a misnomer.[1] The name "serverless computing" is used because the server management and capacity planning decisions are completely hidden from the developer or operator. Serverless code can be used in conjunction with code deployed in traditional styles, such as microservices. Alternatively, applications can be written to be purely serverless and use no provisioned servers at all.[2]

Google Cloud Functions - How It Works



Major Features



No server management

Scales automatically

Pay only while your code runs

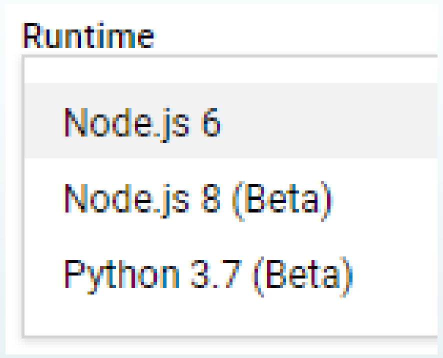
Runs code in response to events

Open and familiar

Connects and extends cloud services

Language Support

- Cloud Functions can be written in Node.js and Python
- Executed in language specific runtimes
- Node.js runtime is based on v6.14.0 and v8.11.1 (Beta)
- Python runtime is based on v3.7.0

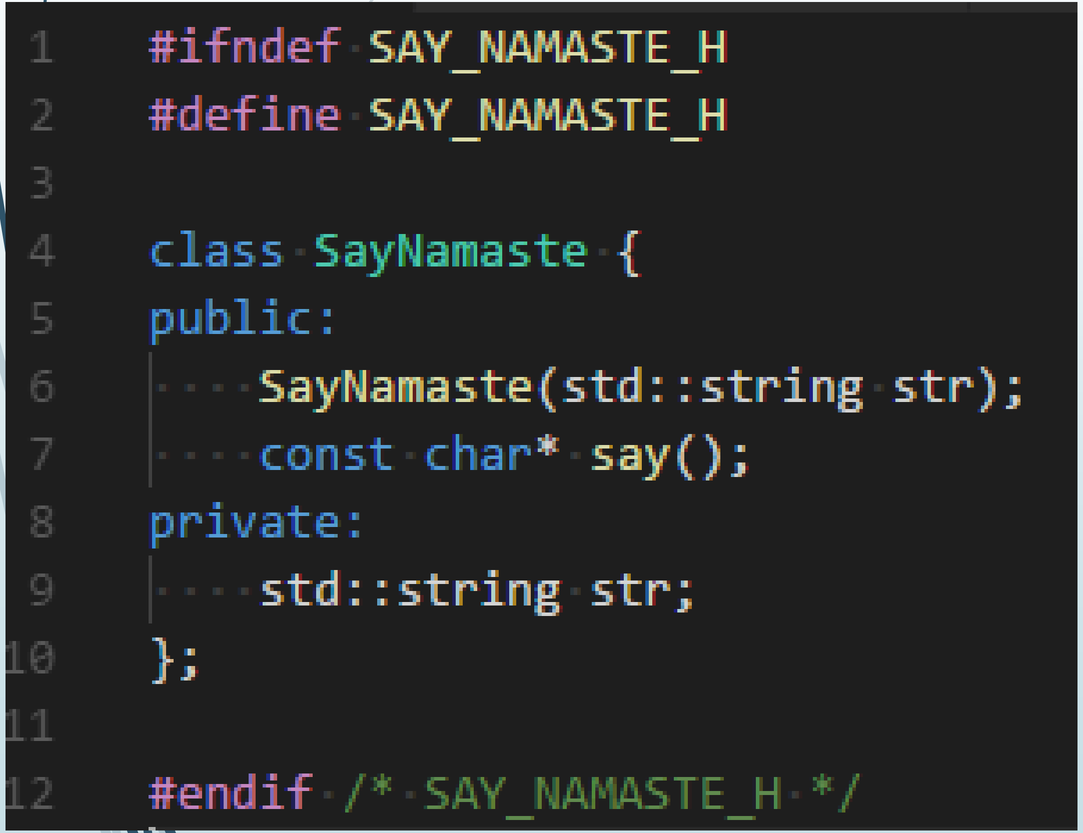


Generic Support

- Cloud Functions can access almost all major GCP services
- Cloud Functions can be triggered by events from:
  - HTTP
  - Cloud Storage
  - Cloud Pub/Sub
  - Firebase

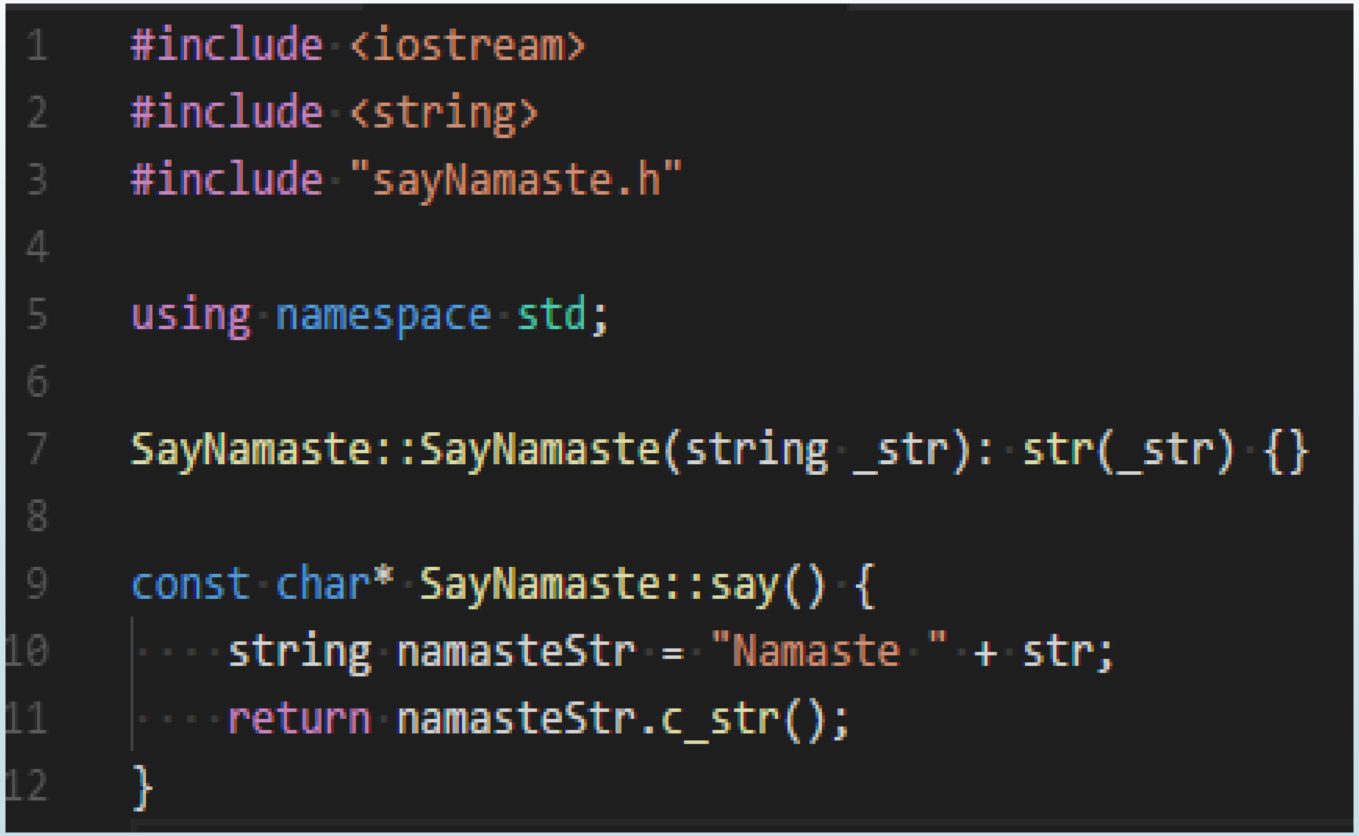
Let’s Start – Namaste Duniya!

sayNamaste.h



```cpp
#ifndef SAY_NAMASTE_H
#define SAY_NAMASTE_H

class SayNamaste {
public:
    SayNamaste(std::string str);
    const char* say();
private:
    std::string str;
};

#endif /* SAY_NAMASTE_H */
```

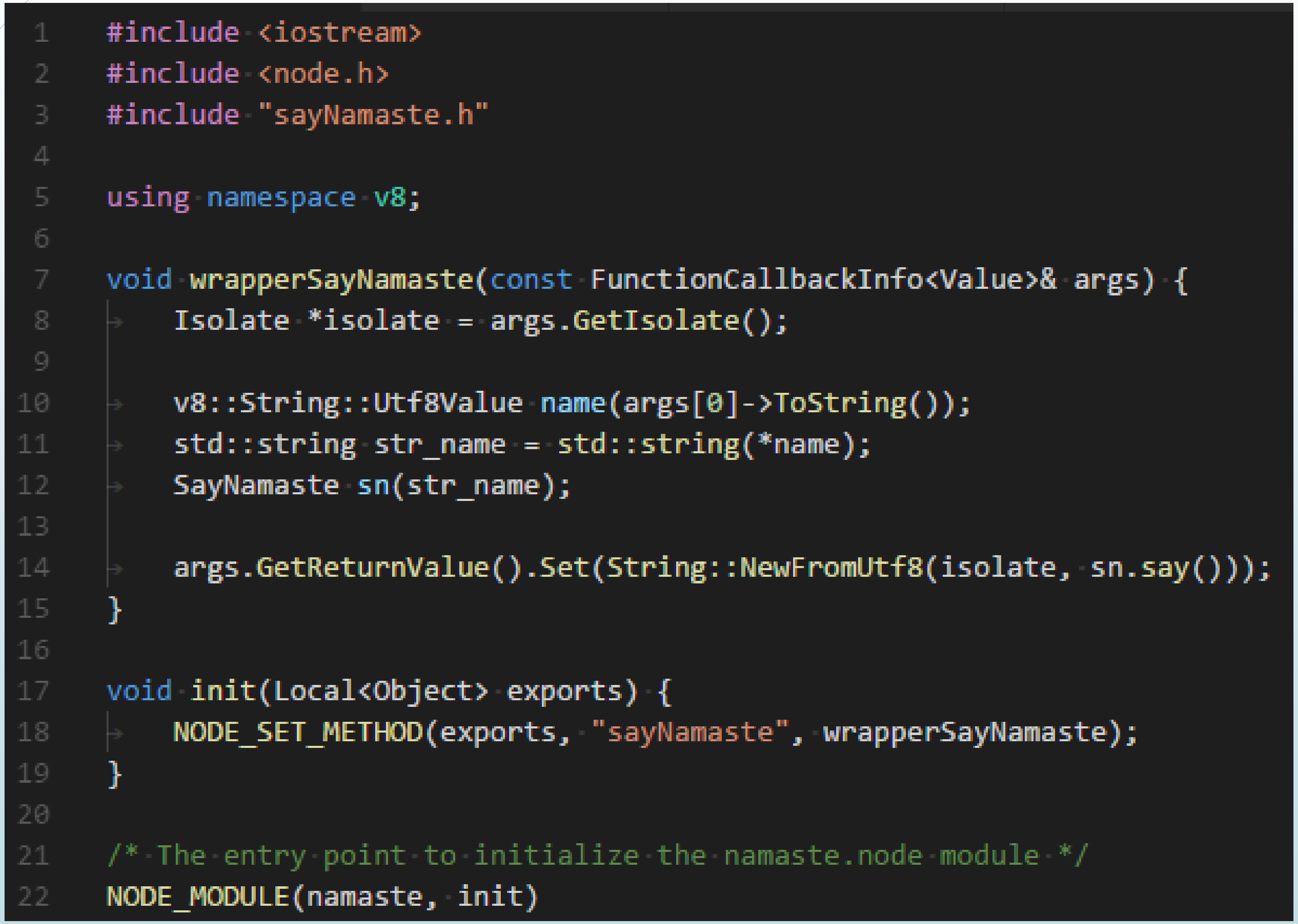
sayNamaste.cpp



```cpp
#include <iostream>
#include <string>
#include "sayNamaste.h"

using namespace std;

SayNamaste::SayNamaste(string _str): str(_str) {}

const char* SayNamaste::say() {
    string namasteStr = "Namaste " + str;
    return namasteStr.c_str();
}
```

Warp it with V8 runtime for add-on invocation

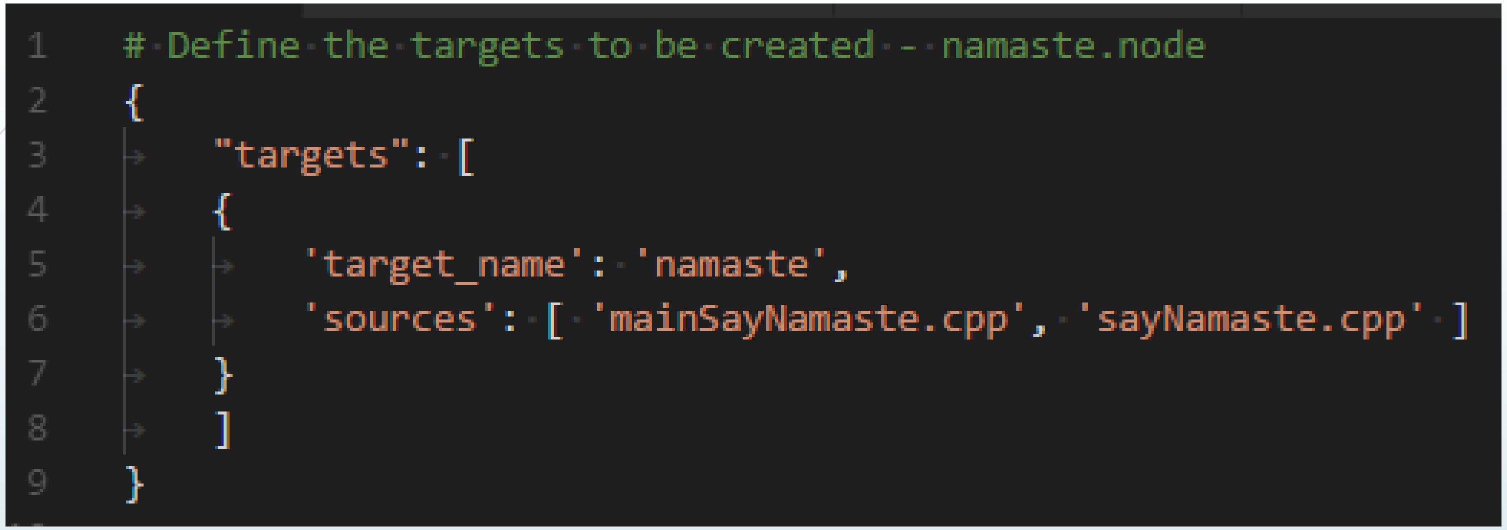
mainSayNamaste.h



```
#include <iostream>
#include <node.h>
#include "sayNamaste.h"

using namespace v8;

void wrapperSayNamaste(const FunctionCallbackInfo<Value>& args) {
    Isolate *isolate = args.GetIsolate();

    v8::String::Utf8Value name(args[0]->ToString());
    std::string str_name = std::string(*name);
    SayNamaste sn(str_name);

    args.GetReturnValue().Set(String::NewFromUtf8(isolate, sn.say()));
}

void init(Local<Object> exports) {
    NODE_SET_METHOD(exports, "sayNamaste", wrapperSayNamaste);
}

/* The entry point to initialize the namaste.node module */
NODE_MODULE(namaste, init)
```

Addon target definition

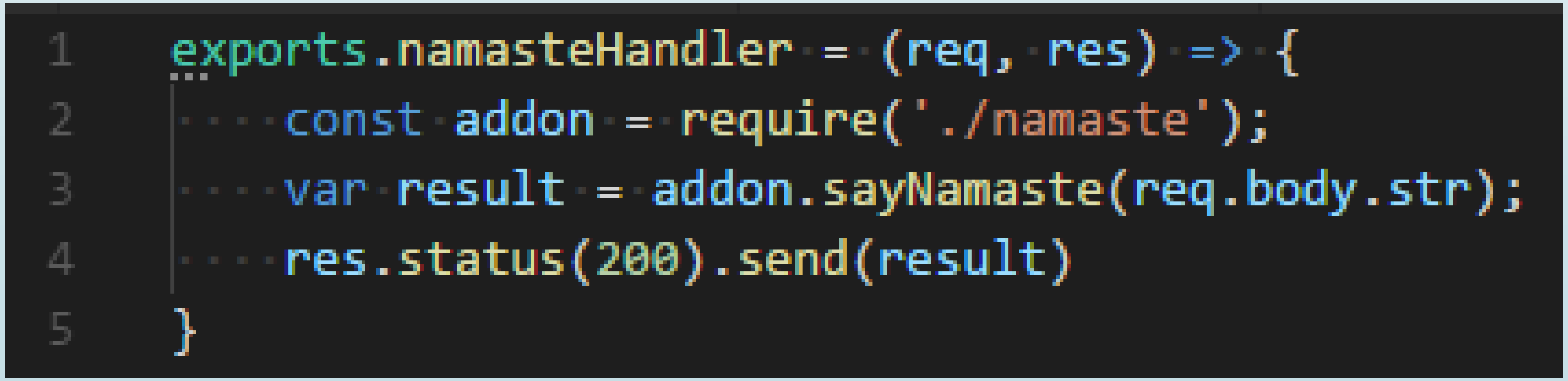
binding.gyp



```
# Define the targets to be created - namaste.node
{
    "targets": [
    {
        'target_name': 'namaste',
        'sources': [ 'mainSayNamaste.cpp', 'sayNamaste.cpp' ]
    }
    ]
}
```

Here’s how it’s invoked

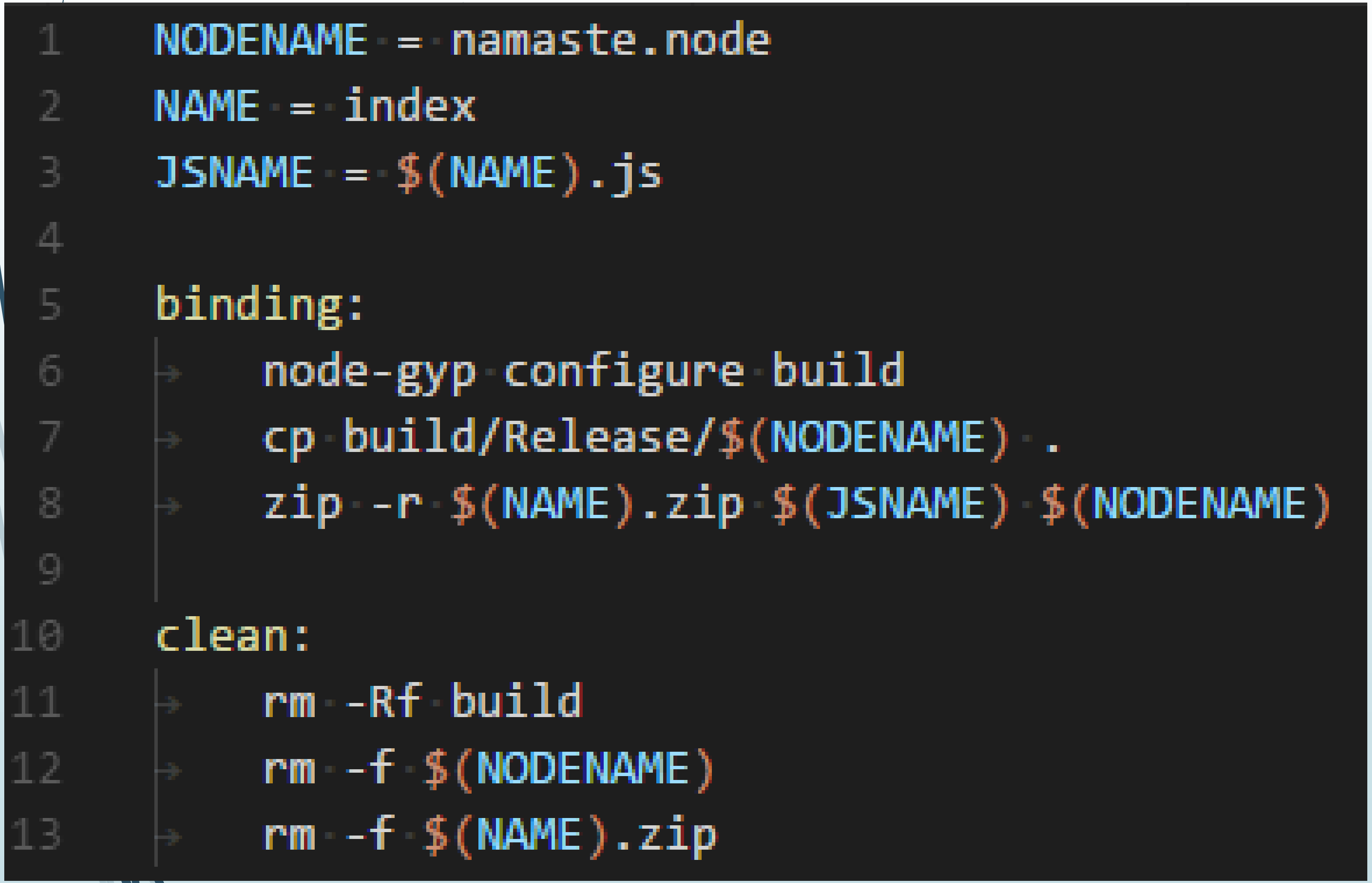
Index.js



```
exports.namasteHandler = (req, res) => {
    const addon = require('./namaste');
    var result = addon.sayNamaste(req.body.str);
    res.status(200).send(result)
}
```

Next steps…

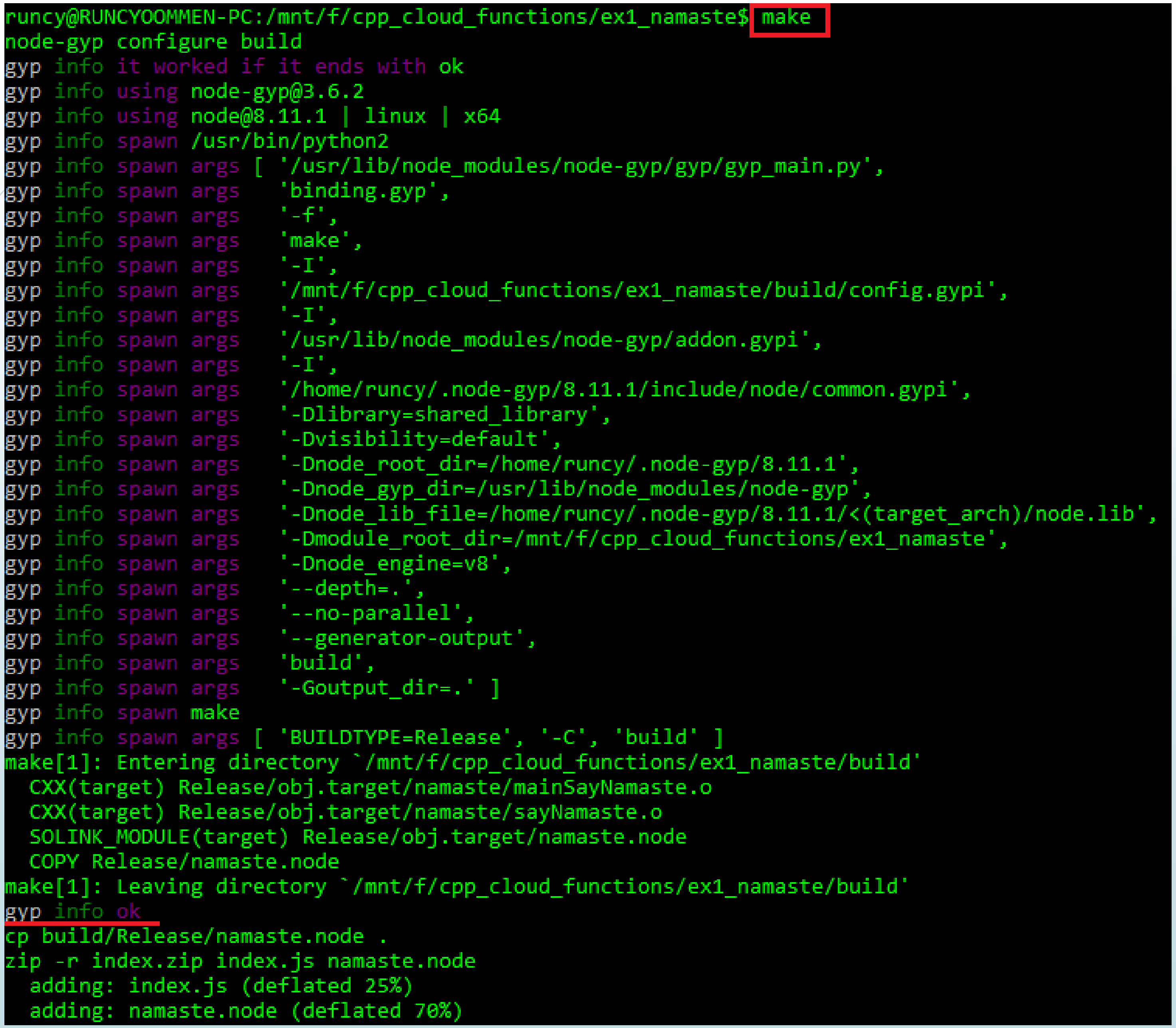
Makefile



```
NODENAME = namaste.node
NAME = index
JSNAME = $(NAME).js

binding:
	node-gyp configure build
	cp build/Release/$(NODENAME) .
	zip -r $(NAME).zip $(JSNAME) $(NODENAME)

clean:
	rm -Rf build
	rm -f $(NODENAME)
	rm -f $(NAME).zip
```

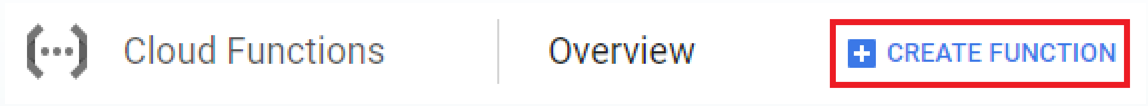
- Build the addon target
- Zips them together
- Creates index.zip for upload

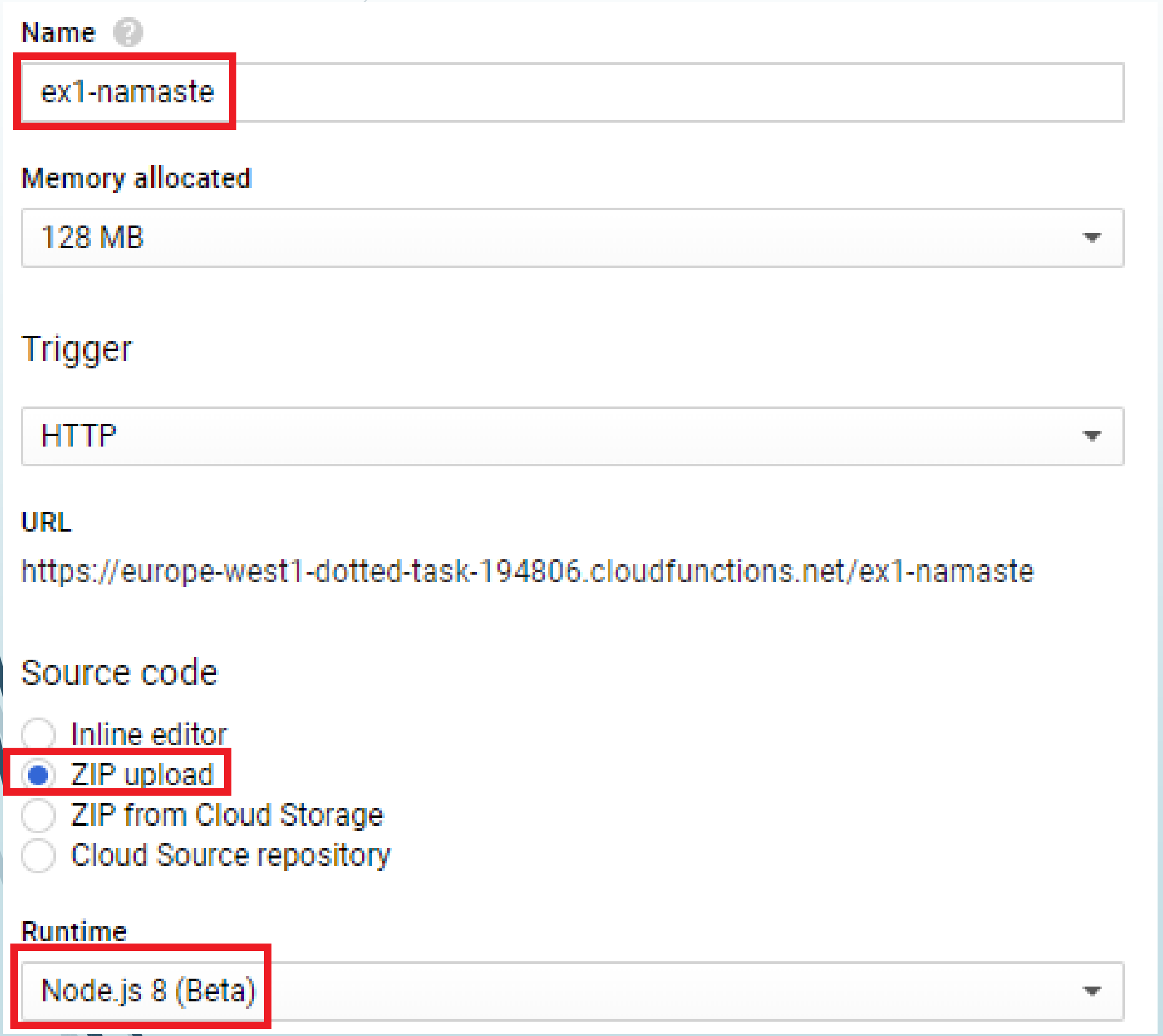
Time to “make”



```
runcy@RUNCYOOMMEN-PC:/mnt/f/cpp_cloud_functions/ex1_namaste$ make
node-gyp configure build
gyp info it worked if it ends with ok
gyp info using node-gyp@3.6.2
gyp info using node@8.11.1 | linux | x64
gyp info spawn /usr/bin/python2
gyp info spawn args [ '/usr/lib/node_modules/node-gyp/gyp/gyp_main.py',
gyp info spawn args   'binding.gyp',
gyp info spawn args   '-f',
gyp info spawn args   'make',
gyp info spawn args   '-I',
gyp info spawn args   '/mnt/f/cpp_cloud_functions/ex1_namaste/build/config.gypi',
gyp info spawn args   '-I',
gyp info spawn args   '/usr/lib/node_modules/node-gyp/addon.gypi',
gyp info spawn args   '-I',
gyp info spawn args   '/home/runcy/.node-gyp/8.11.1/include/node/common.gypi',
gyp info spawn args   '-Dlibrary=shared_library',
gyp info spawn args   '-Dvisibility=default',
gyp info spawn args   '-Dnode_root_dir=/home/runcy/.node-gyp/8.11.1',
gyp info spawn args   '-Dnode_gyp_dir=/usr/lib/node_modules/node-gyp',
gyp info spawn args   '-Dnode_lib_file=/home/runcy/.node-gyp/8.11.1/<(target_arch)/node.lib',
gyp info spawn args   '-Dmodule_root_dir=/mnt/f/cpp_cloud_functions/ex1_namaste',
gyp info spawn args   '-Dnode_engine=v8',
gyp info spawn args   '--depth=.',
gyp info spawn args   '--no-parallel',
gyp info spawn args   '--generator-output',
gyp info spawn args   'build',
gyp info spawn args   '-Goutput_dir=.' ]
gyp info spawn make
gyp info spawn args [ 'BUILDTYPE=Release', '-C', 'build' ]
make[1]: Entering directory `/mnt/f/cpp_cloud_functions/ex1_namaste/build'
  CXX(target) Release/obj.target/namaste/mainSayNamaste.o
  CXX(target) Release/obj.target/namaste/sayNamaste.o
  SOLINK_MODULE(target) Release/obj.target/namaste.node
  COPY Release/namaste.node
make[1]: Leaving directory `/mnt/f/cpp_cloud_functions/ex1_namaste/build'
gyp info ok
cp build/Release/namaste.node .
zip -r index.zip index.js namaste.node
  adding: index.js (deflated 25%)
  adding: namaste.node (deflated 70%)
```

Creating the cloud function





Name: ex1-namaste

Memory allocated: 128 MB

Trigger

HTTP

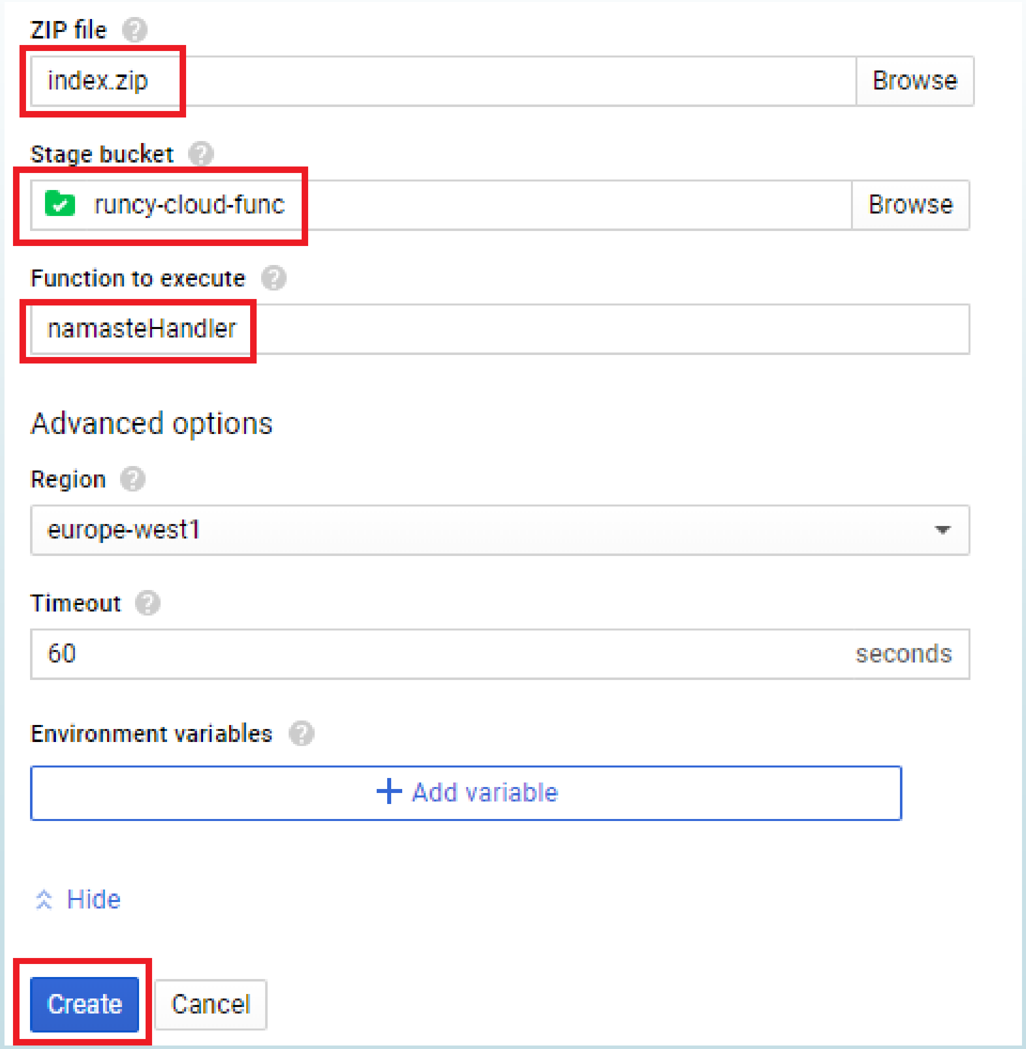
URL

https://europe-west1-dotted-task-194806.cloudfunctions.net/ex1-namaste

Source code

Inline editor [✓ ZIP upload] ZIP from Cloud Storage Cloud Source repository

Runtime: Node.js 8 (Beta)



ZIP file: index.zip

Stage bucket: runcy-cloud-func

Function to execute: namasteHandler

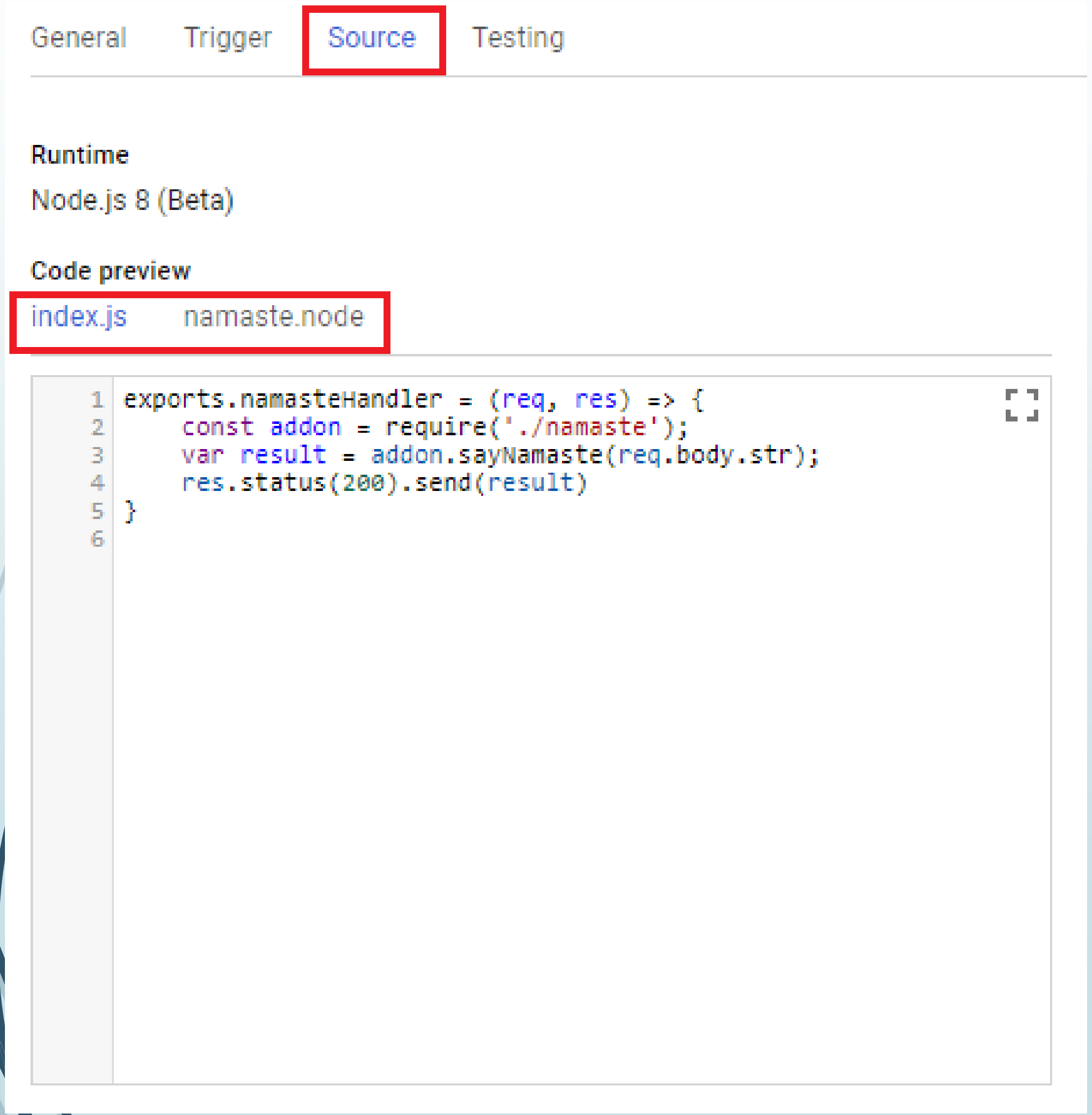
Advanced options

Region: europe-west1

Timeout: 60 seconds

Environment variables

Preview uploaded source



Runtime

Node.js 8 (Beta)

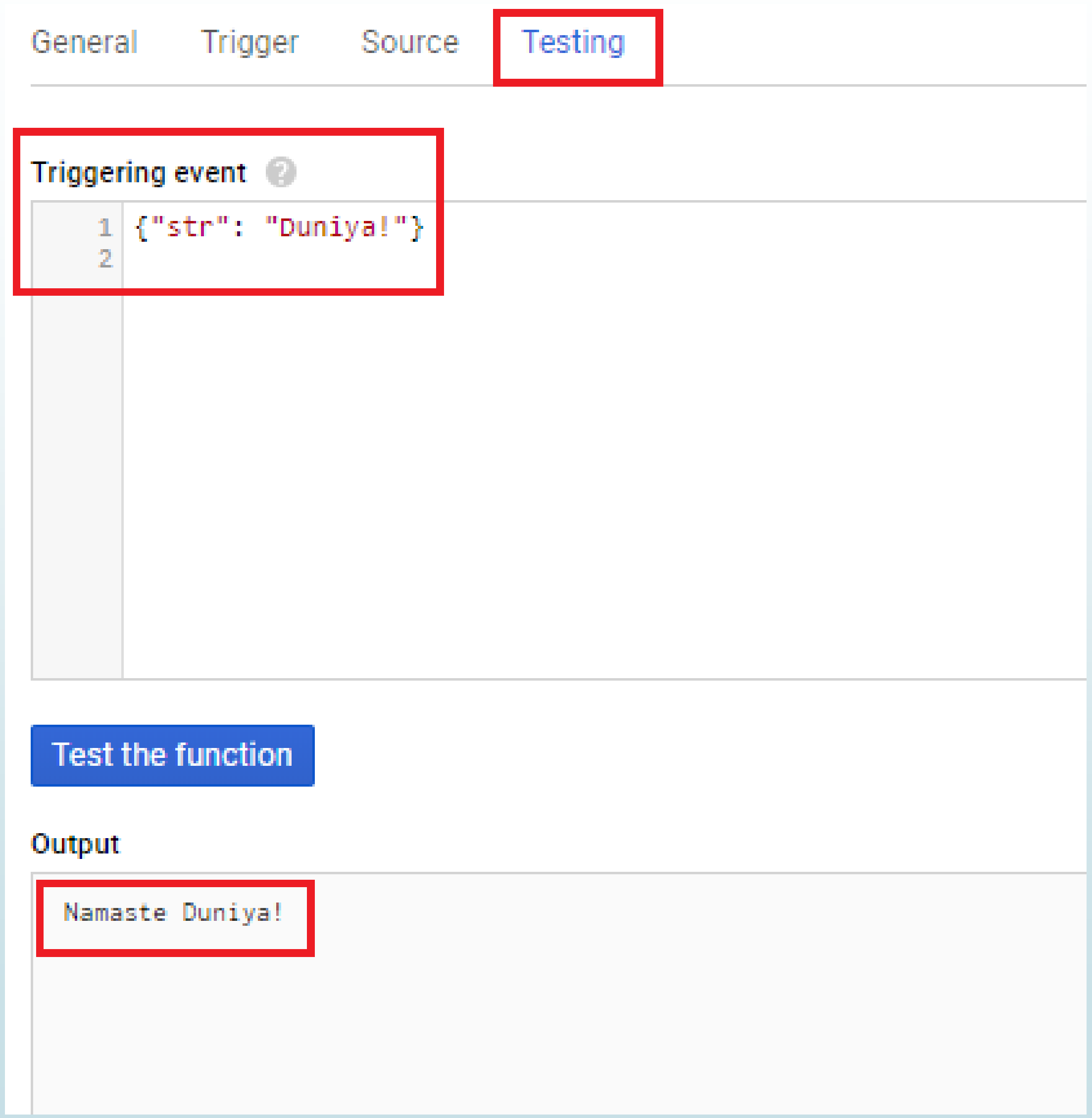
Code preview

index.js namaste.node

```
exports.namasteHandler = (req, res) => {
    const addon = require('./namaste');
    var result = addon.sayNamaste(req.body.str);
    res.status(200).send(result)
}
```

- You can preview the source files including the node to check if it got uploaded properly

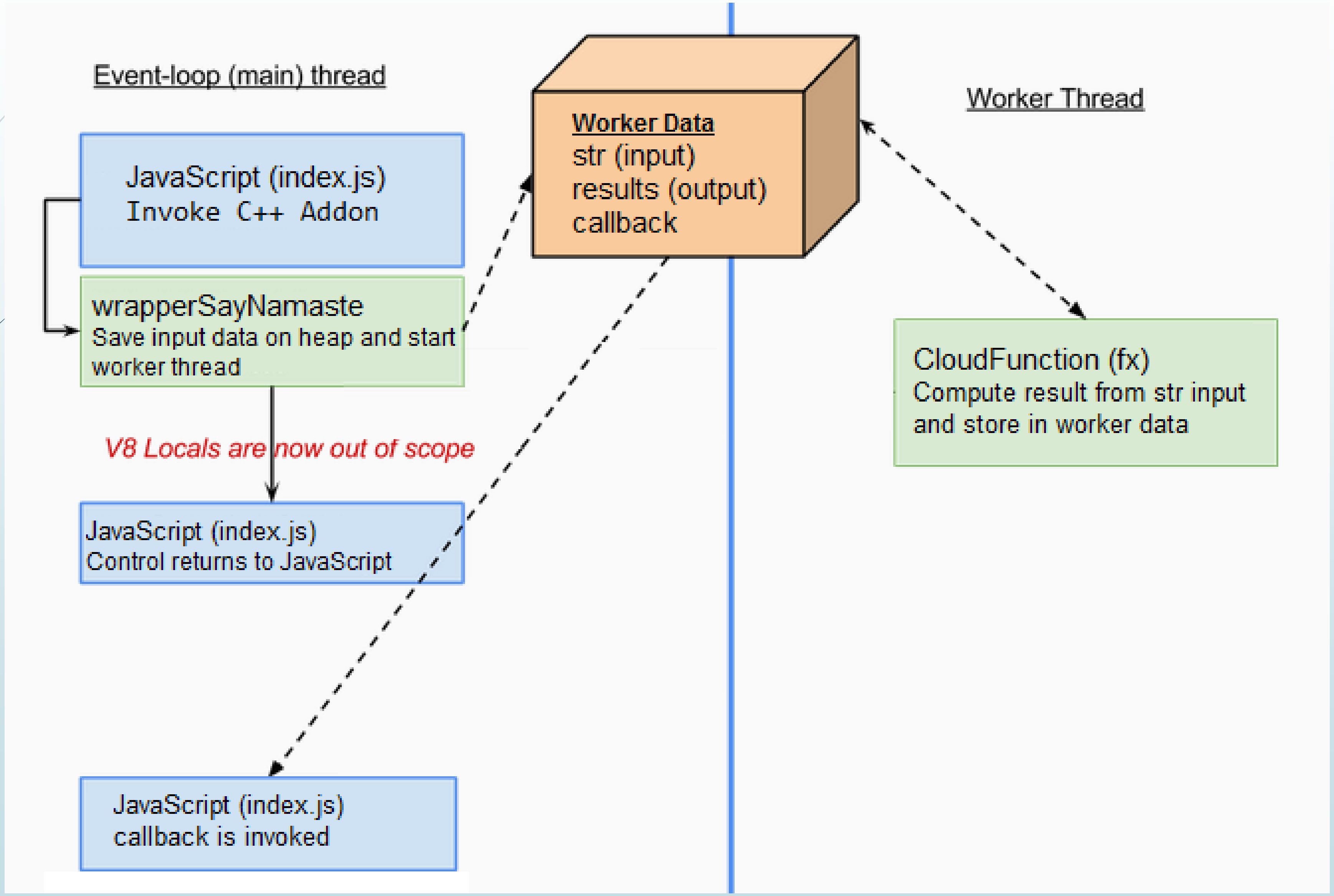
Let's test!



```
{"str": "Duniya!"}
```

```
Namaste Duniya!
```

Depicting the flow



References/Links

- **V8 Runtime Addon:** https://nodejs.org/api/addons.html
- **Node.js native addon build tool:** https://github.com/nodejs/node-gyp
- **C++ processing from Node.js:** https://nodeaddons.com/c-processing-from-node-js/
- **Source Repo:** https://github.com/roommen/cpp_cloud_functions

Let’s make an offering to the Demo Gods…

Q & A