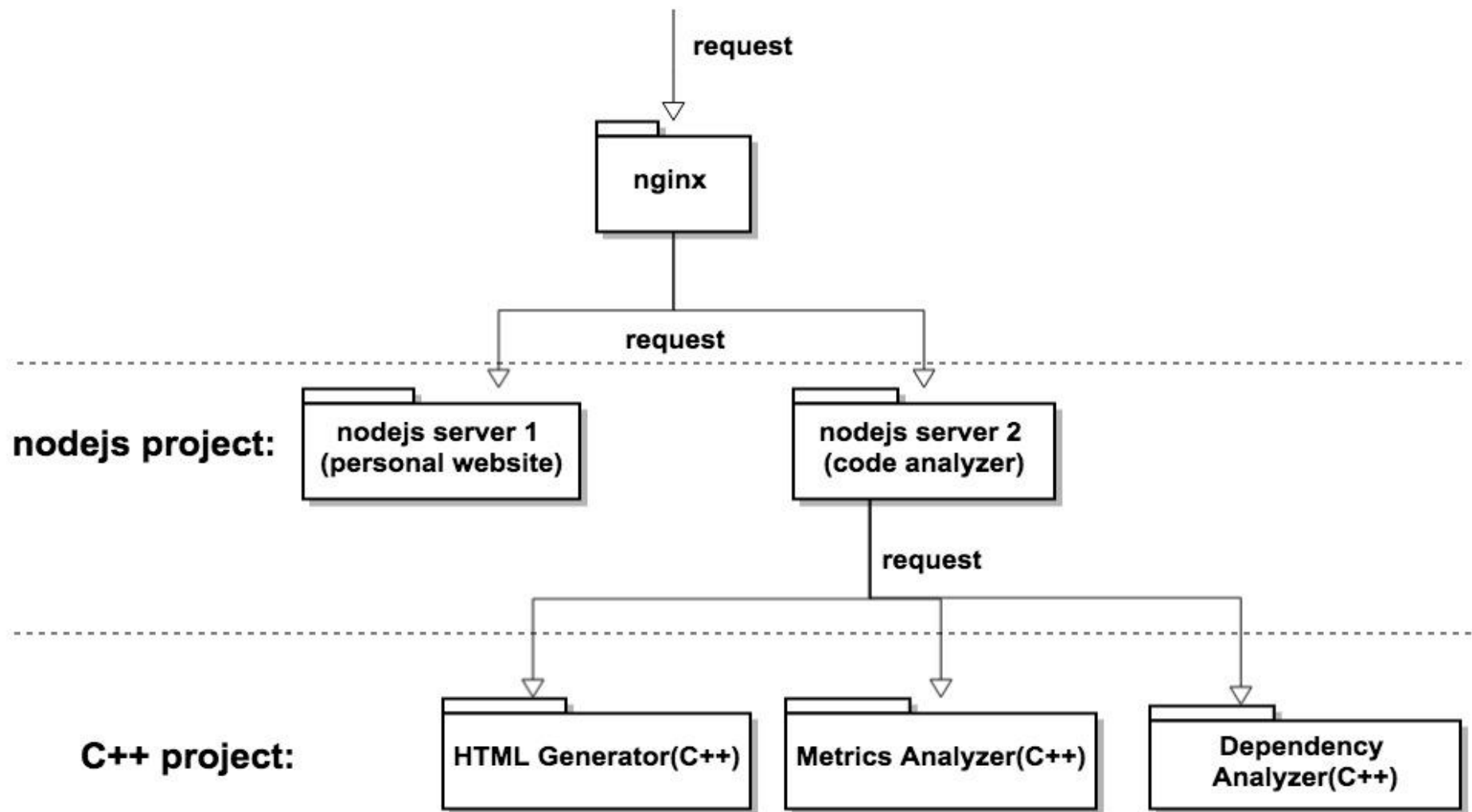

Probing example

— - Distributed Code Master —



How does nodejs talk to C++

1. Use socket.
2. Use node's child-process module to load C++ program.
3. Use V8's support for C++ and node-gyp tool.

GYP: i.e. “generate your project”

1. A building tool for cross-platform auto building.
2. Json configuration

```
"target_defaults": {  
  "cflags": [],  
  "default_configuration": "Release",  
  "defines": [],  
  "include_dirs": [],  
  "libraries": []  
},
```

V8 is a JS engine

1. Compile JS into binary code
2. Written in C++
3. Fast as its name suggests



An example of chatting between C++ and Node

In c++:

1. Include header
2. User V8's wrapper class

```
#include <sys/types.h>
#include <node.h>
#include <v8.h>
#include "../FileSystem/FileSystem.h"
using namespace v8;

void generateHTML(const FunctionCallbackInfo<Value>& args) {

    //convert to std::string
    String::Utf8Value _templateFile(args[0]->ToString());
    std::string templateFile = std::string(*_templateFile);

    String::Utf8Value _destHTMLFolder(args[1]->ToString());
    std::string destHTMLFolder = std::string(*_destHTMLFolder);

    String::Utf8Value _searchPath(args[2]->ToString());
    std::string searchPath = std::string(*_searchPath);

    HTMLgenerator g( templateFile, destHTMLFolder);

    // provide patterns for file manager
    std::vector<std::string> searchPatterns = {"*.h", "*.cpp", "*.c", "*.cc", "*.C"};
    FileManager::IFileMgr *pFileMgr = new FileManager::FileMgr(searchPath);
    for (auto p : searchPatterns) {
        pFileMgr->addPattern(p);
    }

    pFileMgr->regForFiles(&g);
    pFileMgr->search();

    bool isComplete = true;
    args.GetReturnValue().Set(isComplete);
}
```

In nodejs:

- Just call and wait return

```
const generateHTML = function(templateHTMLPath, destHTMLPath, searchPath) {  
    let isFinished = htmlGenerator(templateHTMLPath, destHTMLPath, searchPath);  
    return isFinished;  
}
```

C++ project: HTML Generator

1. Provide template HTML files with some placeholders.
2. Transfer some C++ symbols into HTML representation
 - '<' => '<'
 - '>' => '>'
 - ' ' => ' '
3. Store generated HTML files.

[illegible]

C++ project: Metrics analyzer

1. Use the parser to generate a AST for each C++ file.
2. Analyze based on the info stored in the node of the AST.
3. Store result into a file.

```

void MetricExecutive::analyze()
{
    // do analysis
    FileCollection fileCollection = fileSearch(_path);
    if (fileCollection.size() == 0)
    {
        std::cout << "whooops....can not find any file matches : ( ";
        return;
    }
    for (int i = 0; i < fileCollection.size(); i++)
    {

        std::string filename = FileSystem::Path::getName(fileCollection[i]);
        stream(filename);
        ASTTree* tree = generateTree(fileCollection[i]);
        analyzer.setTree(tree);
        analyzer.analyzeFunc();
        _metricInfos[filename] = analyzer.metricInfo();
        analyzer.clean();
    }
}

void stream(File filename) {
    std::cout << "\n =====";
    std::cout << "\n The No." << i + 1 << " file is: " << filename << std::endl;
    analyzer.metricsResultBuf << "\n =====";
    analyzer.metricsResultBuf << "\n The No." << i + 1 << " file is: " << filename;
}

```

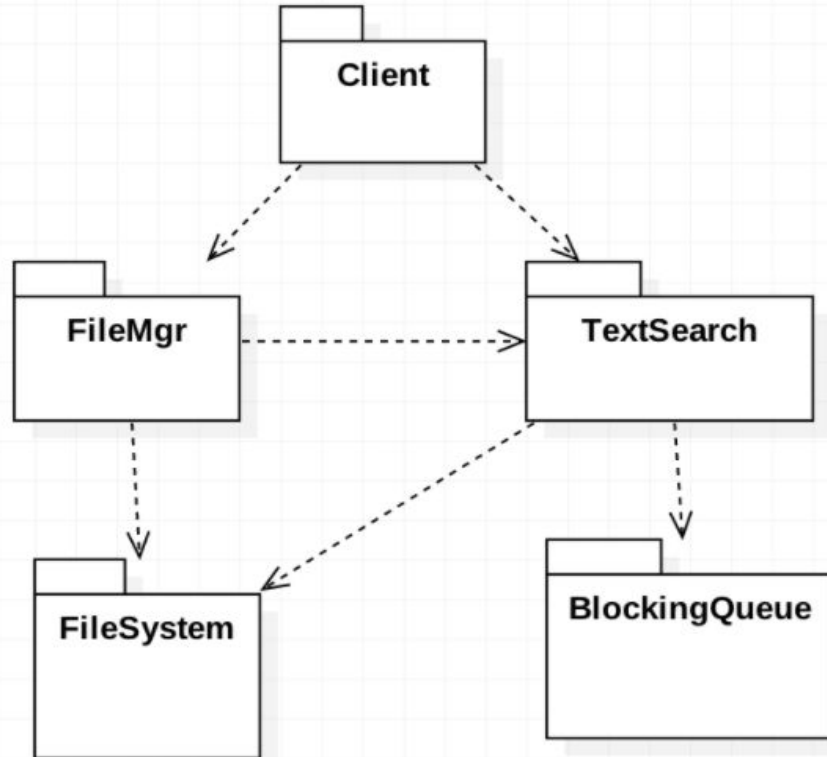
C++ project: Dependency analyzer

1. Same idea as in the OOD project.
2. Output a dependency graph on the webpage based on the result.

(Coming soon..)

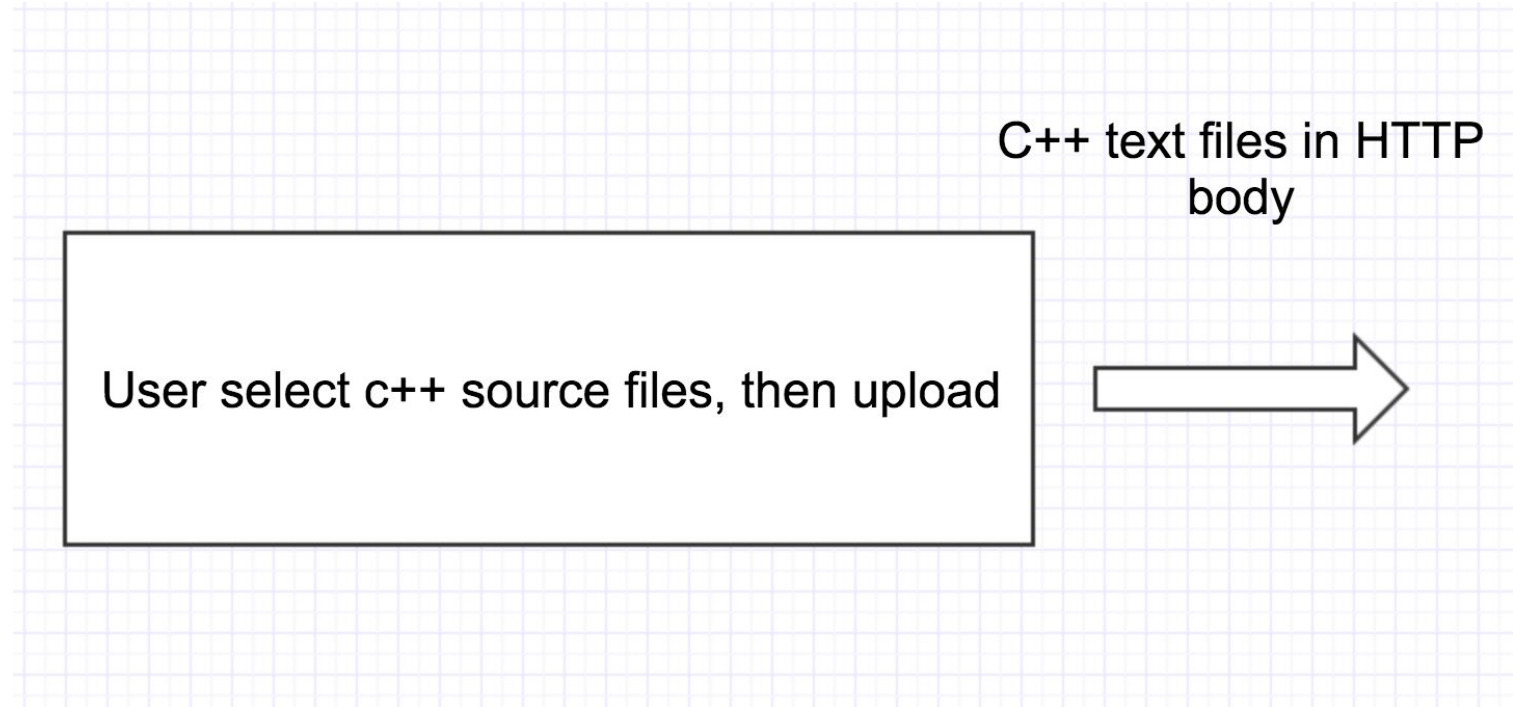
Dependency graph for your project:

Expected outcome:



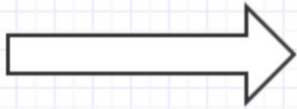
Full Stack Demo:

Flow of the project



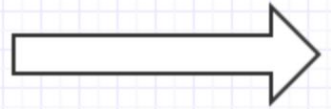
Flow of the project

C++ text files in HTTP
body



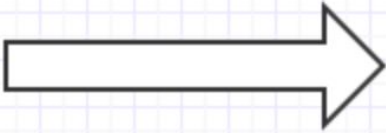
Server receive HTTP message
1. Use "multer" middleware to process files
2. Use "Express" to route the request.

A request object



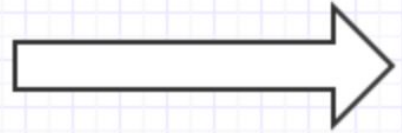
Flow of the project

A request object



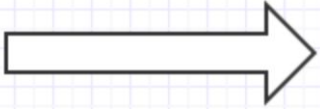
My handler get the request,
1. Extract files out.
2. Store them

Files in the disk



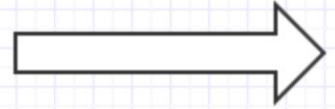
Flow of the project

Files in the disk



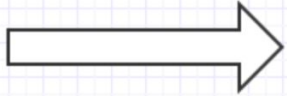
Call HTMLgenerator and MetricsAnalyzer sequentially, to process files in a given path.

Files in the disk
OR JSON String



Flow of the project

Files in the disk
OR JSON String



1. Generate JSON result, and return to the nodejs server,
2. nodejs process the JSON, update the UI.



End

Future Plan:

1. Get Dependency Analyzer done.
2. Add color theme for code display (molokai, monaco, etc.).
3. Provide online compiler for C++ code.
4. Apply design patterns to the project.

Thank you