1 Overview

2 server

2.1 package.json

I should modify `dependencies` and `devDependencies` for new versions of vscode, nodejs and npm.

2.2 tsconfig.json

This tsconfig.json file is a configuration file for the TypeScript compiler. It specifies various compiler options that control how the TypeScript compiler should behave when compiling TypeScript code.

2.3 webpack.config.js

The JavaScript code is a configuration file for a webpack build process. It exports a function that takes a default configuration object and modifies it to suit the needs of the server build.

2.4 package-lock.json

This file is a lock file generated by npm. It contains information about the dependencies and their versions.

2.5 customInstallServerIntoExtension.js

customInstallServerIntoExtension.js is a Node.js script that installs server npm modules into an extension's server location. It copies the package.json and tsconfig.json files to the extension's server location, and runs npm install –production to install the necessary npm modules. Additionally, it copies the package-lock.json file to the extension's server location.

3 client

4 ikos-vscode-pluign

4.1 package.json

The package.json file is a configuration file used in Node.js-based projects. It contains metadata about the project and its dependencies.

Here are some of the key fields in a typical package.json file:

name: The name of the project. version: The version of the project. description: A brief description of the project. author: Information about the author of the project. license: The license under which the project is released. dependencies: A list of dependencies required by

the project. devDependencies: A list of development dependencies required for building and testing the project. scripts: A set of scripts that can be run using the npm run command. In the context of the package.json file you mentioned, it seems to be a configuration file for a VS Code extension. It likely contains information about the extension, its dependencies, and the scripts that can be run to build and test it.

4.1.1 galleryBanner

`galleryBanner` specifies the color and theme of the banner that is displayed in the Visual Studio Code Marketplace when the extension is listed. For example, the color can be set to `#303f9f` and the theme can be set to `dark`.

```
1      "galleryBanner": {
2          "color": "#303f9f",
3          "theme": "dark"
4        }
```

In addition, this file defines many other fields that can be used to customize the behavior of the extension. For example, you can use the `engines` field to specify the Node.js version that the extension requires. Many information can be displayed in the Visual Studio Code Marketplace when the extension is listed. For example, you can use the `publisher` field to specify the publisher of the extension.

4.1.2 engines

I have modified the `engines` field to adapt ikos-vscode plugin. I extended the `publisher` field to zoush99, and made it support ikos.

I have modified the `activationEvents` to support LLVM IR. In the future, I would like to enable this extensoin to support Fortran programs.

4.1.3 dependencies and devDependencies

But I also need to modify `dependencies` and `devDependencies` fields because they may be too old to work with current version of nodejs or npm.

The configuration provides a comprehensive set of options for the c-cpp-flylint extension, allowing users to customize the behavior of the linter, including when it runs, logging verbosity, language standards, include paths, and web query settings for diagnostic code lookups. The configuration uses various data types, including booleans, strings, arrays, and enums, to define the settings and their defaults.

```
1      "c-cpp-flylint.run": {
2          "order": 2,
3          "type": "string",
4          "enum": [
5            "onSave",
6            "onType",
7            "onBuild"
8          ],
9          "default": "onSave",
10         "description": "Run the linter upon typing (onType), upon a build task (onBuild)
```

```
11      }
```

This configuration setting for ‘c-cpp-flylint‘ allows you to specify when the linter should run. You can choose between running it on typing, on saving the file, or on building the project, with the default behavior being to run the linter when the file is saved.

4.1.4 configurations of different analyzers

There are many configuration of different analyzers, such as `Clang, Cppcheck, Flexelint, PC-lint, PC-lint Plus and lizard`. But I get confused when I try to find the configuration of the analyzers' parameters. Where is the configuration of the analyzers' parameters?

4.2 tsconfig.based.json

This is a TypeScript configuration file that sets compiler options for TypeScript.

`emitDecoratorMetadata` enables the emitting of metadata for decorators. `experimentalDecorators` enables experimental support for decorators. `noImplicitAny` disallows implicit any types. `noImplicitReturns` disallows functions that don't explicitly return a value. `noUnusedLocals` disallows unused local variables. `noUnusedParameters` disallows unused function parameters. `strict` enables strict type checking options.

4.3 tsconfig.json

This tsconfig.json file is a configuration file for the TypeScript compiler. It specifies various compiler options that control how the TypeScript compiler should behave when compiling TypeScript code.

4.4 typings.d.ts

This file is a file that declare the types for the ikos-vscode plugin. Some lines are added by zoush99, because ikos-vscode plugin needs them.

4.5 shared.webpack.config.js

This file is a common webpack configuration file for the ikos-vscode plugin.

4.6 package-lock.json

This file is a lock file generated by npm. It contains information about the dependencies and their versions.

4.6.1 code-climate.sh

This is a shell script that runs a Docker container.