judge-image-for-annaforces

Lohit P Talavar, Adarsh Mishra, Priyanshi Meena

This project provides a Python script that allows you to compile and execute code (Python, C, C++) securely in a Docker container, enforcing time and memory limits. It is primarily designed for automated code judging, competitive programming, or sandboxed code evaluation environments.

Features

- **Language support**: Python, C, C++
- **Security**: Runs code inside a Docker container
- **Resource limits**: Enforces memory and time limits for execution
- **Automatic Docker image management**: Builds the judge image if not present
- **Logging**: Outputs events and errors to `judge_runner.log`
- **Easy file management**: Functions to create, delete, and manage test files

Usage

1. Prerequisites

- Docker must be installed and running on your system.
- Python 3.x

Defaults

Parameter	Default Value	Description
Docker Image Name	`judge-image`	The name of the Docker image
Docker Image Tag	`latest`	The tag for the Docker image
Full Docker Image	`judge-image:latest`	Image name used for execution
Test Folder	`test`	Folder in which files are created
Test File	`main.py`	Default filename for Python
Default Compile Time Limit	5 seconds	Time allowed for compiling C/C++
Default Run Time Limit	1 second	Time allowed for running code
Default Memory Limit	1024 MB	Max memory allowed in container
Logging File	`judge_runner.log`	Log file for execution events/errors

2. Quick Start

Clone the repository and run the script:

```
python app.py
```

3. Main Functions

create_folder_and_file(folder_name, file_name, content) Creates a folder and writes the specified content to a file.
Defaults:

- folder_name: `test`
- file_name: `main.py`
- content: Minimal Python print statement

delete_folder(folder_name) Deletes the folder (and its contents) specified.
Default: `test`

create_image(image) Checks if the Docker image exists. If not, builds the image from the current directory.
Default: `judge-image:latest`

run_code_in_container(image, file_path, language, stdin, time_limit, memory_limit)

- **image**: Docker image to use (default: `judge-image:latest`)
- **file_path**: Path to the source code file (default: `test/main.py`)
- **language**: `'python'`, `'c'`, or `'c++'` (default: `'python'`)

- **stdin**: Input to provide to the program (default: '')
- **time_limit**: Seconds allowed for execution (default: 1)
- **memory_limit**: Memory limit in MB (default: 1024)

Returns a result dictionary:

```
{
    'success': True/False,
    'stdout': 'Program output',
    'stderr': 'Error output',
    'error': 'Error message if any'
}
```

4. Example

```
from app import create_folder_and_file, create_image, run_code_in_container, delete_folder

cpp_code = (
    '#include <iostream>\n'
    'using namespace std;\n'
    'int main() {\n'
    '    string s; cin >> s;\n'
    '    cout << "Echo: " << s << endl;\n'
    '    return 0;\n'
    '}\n'
)
create_folder_and_file(file_name='main.cpp', content=cpp_code)
create_image()
result = run_code_in_container(language='c++', file_path='test/main.cpp', stdin='HelloWorld'
print(result)
delete_folder()
```

5. Logging

All execution logs, errors, and build information are written to `judge_runner.log` in append mode.

6. Troubleshooting

- Ensure Docker is running and you have permissions to execute containers.
- If you get permission errors, check file ownership and Docker setup.
- The Docker image must support the target language (update Dockerfile if needed).

7. Extending

- To add more languages, update the command generation logic in `run_code_in_container`.
- To change resource limits, adjust the `--memory` flag and `timeout` parameters.

License

MIT License

Authors

- Lohit P Talavar
- Adarsh Mishra
- Priyanshi Meena

Credits

- Uses Docker for sandboxing.
- Python standard libraries for subprocess and file management.