Documentation for the Glimlach Docker Image Runner (cli.py)

Table of Contents

1. Introduction

   The Glimlach Docker Image Runner, represented by the Python script cli.py, is designed to automate the execution of Docker images based on a provided JSON configuration file. This script is particularly useful for running multiple Docker images with different configurations concurrently.

2. Features

   - Parallel Execution: The script utilizes the multiprocessing module to execute Docker images concurrently, enhancing overall performance.

   - Configuration File: Docker images and their configurations are specified in a JSON configuration file, making it easy to manage and customize the execution process.

   - Placeholder Replacement: The script supports the replacement of placeholders in the configuration file with actual values, allowing for dynamic configuration.

   - Logging: The script provides informative logging at different levels (INFO, ERROR) to track the progress and identify any issues during execution.

   - Persistence: Completed Docker images are tracked in a file (completed_images.txt) to prevent redundant executions.

3. Usage

   The script is executed from the command line, and it requires a path to a JSON configuration file as an argument. For example:

   “python cli.py config.json “ or “python cli.py config.json “

4. Configuration File

   The configuration file is a JSON file that contains the following sections:

   - Values: Key-value pairs used for replacing placeholders in the configuration file.

   - Images: An array of Docker image configurations. Each configuration includes the Docker image ID, command line arguments, and other necessary details.

   - Output-directory: The directory where output files generated by Docker images will be stored.

5. Placeholder Replacement

   To make the configuration file more dynamic, placeholders can be used and replaced with actual values. Placeholders should be enclosed in < > brackets, and corresponding values are provided in the values section of the configuration file.

   In the provided configuration file example:

   ```
   “”””
   <ip> is replaced with the value 123.456.101.1.
   <out_dir> is replaced with the value /path/to/dir/.
   <web> is replaced with the value https://abc.com.

   “”””
   ```

6. Running the Script

   The script is executed from the command line using the python command, followed by the script file and the path to the configuration file.

   ```
   “”””
   python cli.py config.json

   “”””
   ```

7. Error Handling

   Errors during the execution of Docker images are caught, logged, and the script continues with the remaining images. This ensures that the script does not terminate prematurely due to a single failure.

8. Logging

   The script uses Python's logging module to provide detailed information about the execution process. Information is logged at the INFO level, while errors are logged at the ERROR level.

9. Notes and Considerations

- Completed Images Tracking: The script maintains a file (completed_images.txt) to track completed Docker images. This prevents unnecessary re-execution of images.

- Dependency: Ensure that Docker is installed and available in the system's PATH.

- Concurrency: The script leverages multiprocessing to run Docker images concurrently. Adjust the number of processes based on system capabilities.

This documentation provides an overview of the features and usage of the Glimlach Docker Image Runner script (cli.py). For more detailed information, refer to the inline comments in the script and the provided configuration file example. Customize the configuration file according to your specific Docker image execution requirements.