Open-Source Report for TCP_connections

General Information & Licensing

Code Repository	Flask / Python
License Type	BSD-3-Clause Source
License Description	• THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
License Restrictions	• Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. • Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. • Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

e

Magic

Dispel the magic of this technology. Replace this text with some that answers the following questions for the above tech:
(pattern: links – explanation -links – explanation)

https://github.com/MaRonggg/5bytes/blob/325c16e9d3659d55ecb4abeef3986cdb32cee78a/app.py#L25

https://github.com/miguelgrinberg/Flask-SocketIO/blob/91b5ddc31bebeb6241d281252c711b160550ce01/src/flask_socketio/__init__.py#L242
Analysis: Since the project is using websocket, SocketIO is created and set its async_mode to threading by default. Then, inside the SocketIO, a server is created and tcp_connection is made before websocket connection.

https://github.com/MaRonggg/5bytes/blob/325c16e9d3659d55ecb4abeef3986cdb32cee78a/app.py#L102
Analysis: First, with socketio.run function with flask app as parameter, socketio.run is used for creating bi-directional communications between the clients and the server

https://github.com/miguelgrinberg/Flask-SocketIO/blob/91b5ddc31bebeb6241d281252c711b160550ce01/src/flask_socketio/__init__.py#L553
https://github.com/miguelgrinberg/Flask-SocketIO/blob/91b5ddc31bebeb6241d281252c711b160550ce01/src/flask_socketio/__init__.py#L651

Analysis: Inside the run() function called by socketio.run, the host and port will be set to be default if there are no specific host and are provided. After setting some debug variable, it goes into if statements that check server is on threading mode, since our SocketIO is set to threading by default, so go inside the if statement. Inside if statement, after continuing setting up debug techniques such as checking if the server runs in production deployment. If not, app.run is used to call run() function with parament host, port and threading status.

https://github.com/pallets/flask/blob/066a35dd322f689ec07d7c0e82b19eacadac3c6b/src/flask/app.py#L1067
https://github.com/pallets/flask/blob/066a35dd322f689ec07d7c0e82b19eacadac3c6b/src/flask/app.py#L1191

Analysis: The run() function is for running the application on a local development server, it contains host and port from parameters of app.run. And after fetch the value of port and host, the werkzeug imports a run_simple library and call run_simple with host and port as parameter so that we could create a WSGI application that helps server push requests received to framework and web applications, which is the whole meaning of making tcp connection.

https://github.com/pallets/werkzeug/blob/3115aa6a6276939f5fd6efa46282e0256ff21f1a/src/werkzeug/serving.py#L907

https://github.com/pallets/werkzeug/blob/3115aa6a6276939f5fd6efa46282e0256ff21f1a/src/werkzeug/serving.py#L1055

https://github.com/pallets/werkzeug/blob/3115aa6a6276939f5fd6efa46282e0256ff21f1a/src/werkzeug/serving.py#L1037

Analysis: inside the run_simple function, the validity of port will be checked, next, if the function is not in the production (not using reloader), the server will be set to handle request in an infinite loop. However, before running WSGI application, a server is required. Therefore, the first thing is to create an appropriate WSGI server instance, so make_server() function is called with several input; especially, hostname, port, application that indicate which is the desired WSGI application to run, threaded which's value deicides whether to handle concurrent requests using threads, and processes which's value deicides handle concurrent requests using up to this number of processes; threaded is set to false and process is set to 1 because it is just for tcp_connection.

https://github.com/pallets/werkzeug/blob/3115aa6a6276939f5fd6efa46282e0256ff21f1a/src/werkzeug/serving.py#L853
https://github.com/pallets/werkzeug/blob/3115aa6a6276939f5fd6efa46282e0256ff21f1a/src/werkzeug/serving.py#L893

Analysis: inside the make_server function, creating an appropriate WSGI server instance based on the value of threaded and processes. Since, the threaded is set to false and the process is 1, the BaseWSGIServer is chosen and BaseWSGIServer class is called to initate the server that handles one request at a time by inheriting functionalities from HTTPServer.

https://github.com/pallets/werkzeug/blob/3115aa6a6276939f5fd6efa46282e0256ff21f1a/src/werkzeug/serving.py#L651

Analysis: in this case, the BaseWSGIServer server is set with no multithread and no multiprocess and has the functionalities of httpserver, which supports its parent function.

https://github.com/python/cpython/blob/51ee0a29e9b20c3e4a94a675e73a894ee2fe447b/Lib/http/server.py#L129

Analysis: Inside the HTTPServer, socketserver.TCPServer is called, in this step a server instance is created. server_bind() function is called with the functionalities from TCPServer to bind the socket to the desired address.

https://github.com/pallets/werkzeug/blob/3115aa6a6276939f5fd6efa46282e0256ff21f1a/src/werkzeug/serving.py#L1069
https://github.com/pallets/werkzeug/blob/3115aa6a6276939f5fd6efa46282e0256ff21f1a/src/werkzeug/serving.py#L766

Analysis: After we created a server, and since no reloader is being used, serve_forver() function is executed to calls handle_request() in an infinite loop using the line super().serve_forever(poll_interval=poll_interval) as shown in the Homeworks. For now, a WSGI application is ready for deployment.