Twitter Coding Challenge

Overview
Your objective for this challenge is to produce a tool that will query the 'status' page on 1000 servers, and produce 2 reports based on data within the 'status' page. More details will be provided below.

Materials
Below is the complete list of materials you should have from us. If you haven't received them, or they appear unusable, please let us know and we'll get you updated copies. The materials critical for completing this assignment:

1. This document of instructions, which you're reading
2. 'servers.txt'
3. 'responses.txt'

Task Details

- Servers:
  - The 'servers.txt' file we've sent you contains a list of fictitious servers. Your solution should read this file for its list of endpoints. We have not used FQDNs in our example, just to keep it simple.
  - Although the challenge is to connect to 1000 servers, your **solution should scale,** at least one order of magnitude higher, with performance in mind. Also consider that having a list of servers, like this one, does not guarantee a given server will be online when you attempt to reach it.
  - As mentioned, each server has a 'status' endpoint that returns a JSON object. For example, http://ServerA/status. More details about the JSON data, being returned by 'status', will be provided below.
  - Please implement the code required to call the 'status' endpoint (as an HTTP request) and retrieve the data.
- Response from 'status' endpoint:
  - The JSON object which is returned from the 'status' endpoint contains several key/value pairs. Here are the keys: "Application", "Version", "Uptime", "Request_Count", "Error_Count", "Success_Count"
  - The 'responses.txt' file we've sent you contains example JSON data, to serve as a reference for as you complete this exercise.
  - Please **DO NOT** read the 'responses.txt' file in your code. This is just example JSON data so you can see what could be returned from the endpoints. Each endpoint can be considered to reply with a single entry from the list in 'responses.txt'.
- Output:
  - Your tool should scan each endpoint and produce a report that aggregates:
    - **Success Rate**, by Application, by Version. Note, this is not just a success count, but is instead an aggregate of information. For example,

information about version “1.001” of the “Cache1” application will calculate a different Success Rate than information about version “1.002” of “Cache1”. These may also be different Success Rates than version “9.007” of “Database2” application.

- Your tool should write its output to **standard out** and to a **local file.**
  - Output written to **standard out** should be in a human readable format.
  - Output written to a **local file** should be easily programmatically parsable (plan for this to be consumed by a downstream application).

Rules/Expectations

- You’re welcome to implement this in the language of your choice, however we prefer no shell-scripting for this exercise.
- You may use online resources (Google, Stack Overflow, books, etc.). Please write this code on your own (do not reach out to any individuals or groups to contribute to your solution).
- Your code should be free of compilation and logic errors. Your code runnable in a Linux, command line, production-like environment.
- In addition to correctness, other areas that are also important are clean and easy to understand code, speed, and effective resource utilization within the confines of the language used for your solution.
- Please make sure that whatever you submit is something that would pass a team code review for a tool that would be used in production.
- Your solution should include a **README** with instructions on how to compile/run your code, and any additional information you feel is relevant.
- Submit your solution as specified in the accompanying e-mail, in an archive file. Do not post the challenge or solution publicly.

Additional Information

- While we recognize everyone’s expertise, experiences, and circumstances differ, we’ve attempted to tune this assignment to take under three hours to complete. That being said, we do evaluate code submissions with the rigor that is appropriate for the role and level you’re applying for.
- Please be prepared to discuss your solution if you are selected for an ‘on-site’ (virtual or in person) interview. We may ask you to explain design decisions, fix bugs, and extend your solution to address additional use-cases.