Fetch Take Home Exercise

Site Reliability Engineering

Overview

As a Site Reliability Engineer, it is our job to not only identify issues that will impact the reliability of systems, but also develop processes and procedures that will make it easier for others to do the same work in the future.

For this exercise, we are asking that you evaluate the code example that we provide, find any existing issues with the code, improve the code to ensure that it meets the provided requirements, and write the missing README for the code.

Requirements

Code

- Must use either the provided Go or Python code as your starting point
- Must accept a YAML configuration as command line argument
    - YAML format must match that in the sample provided
- Must accurately determine the availability of all endpoints during every check cycle
- Endpoints are only considered available if they meet the following conditions
    - Status code is between 200 and 299
    - Endpoint responds in 500ms or less
- Must determine availability cumulatively
- Must return availability by domain
- Must ignore port numbers when determining domain
- Check cycles must run and log availability results every 15 seconds regardless of the number of endpoints or their response times

Readme

- Must outline how to install and run the code
- Must include a section that outlines how each of the issues were identified and why each change to the code was made

Submission

- Use of AI tools to complete or assist in this exercise is not allowed
- Final submission must be made through a public git repository

Starting Code and Sample YAML

- **Go**: https://github.com/fetch-rewards/sre-take-home-exercise-go
- **Python**: https://github.com/fetch-rewards/sre-take-home-exercise-python

YAML Explanation

You can assume that the YAML file being used will always be valid

name (string, required) — A free-text name to describe the HTTP endpoint.
url (string, required) — The URL of the HTTP endpoint.
- You may assume that the URL is always a valid HTTP or HTTPS address.

method (string, optional) — The HTTP method of the endpoint.
- If this field is present, you may assume it's a valid HTTP method (e.g. GET, POST, etc.).
- If this field is omitted, the default is GET.

headers (dictionary, optional) — The HTTP headers to include in the request.
- If this field is present, you may assume that the keys and values of this dictionary are strings that are valid HTTP header names and values.
- If this field is omitted, no headers need to be added to or modified in the HTTP request.

body (string, optional) — The HTTP body to include in the request.
- If this field is present, you should assume it's a valid JSON-encoded string.
- If this field is omitted, no body is sent in the request.