Qualification Test

We would like you to deploy The Responding Dark Laughter (hereafter TRDL).

TRDL is a web service that returns the value 42.

Here is a sample interaction with TRDL

```
$ curl http://1.2.3.4/
42
```

The requirements on TRDL are that the production system is a high availability service with a strict service level agreement (several “9”s uptime). Your task is to design the proof-of-concept system which will not run in production, but is required to evaluate if it can fulfill the stakeholders wishes.

The system consists of an HTTP server with some code running inside or behind it. It responds to a GET request as in the example. If you need to extend it, the specific input and output format are not critical.

You must provide reasoning and justifications why you have planned the system the way you have. It should run on public cloud infrastructure (for example AWS or Google Cloud) in a container orchestrator (for example Kubernetes or Docker Swarm). You are also required to ensure that the developer understands what are the additional requirements that TRDL needs to implement to deploy and run the PoC system.

There is a fine line to walk in exercises like this between the effort justified by the simplicity of the problem and showing what you can do in a production setting. When in doubt, err on the side of simplicity and doing less work. You should put some thought into error and edge cases in the implementation and we would like to see how automated testing of some kind should be done, either unit testing of the code or end-to-end testing of the system.

We expect you to spend at most 20 hours on this, and we want you to start by making a “discovery” outlining what you need to do from these directions, and then the time you think this will take for each task needed to complete the exercise.

You need to produce a report containing:
- a plan on how this should be setup and how it should be deployed.
- documentation of how developers need to extend their implementation to make the system work in the environment you define.

Things we'll assess:
- your competence with the technologies you've chosen
- the comprehensibility and correctness of the documentation that you have written
- the clarity and usability of any instructions that you provide to help us deploy the system
- how you have planned the automated build and/or deployment of the system

Things we will not assess:
 - how long it takes you for each task (we are mostly interested in your planning skill)
 - your telepathy: if anything is unclear its ok to make assumptions as long as you define them; or you need a requirements decision, please ask
 - adherence to some arbitrary set of secret criteria that we're not telling you about

Things we'd like to discuss at a follow-up interview:
 - the reason for technical decisions you made along the way
 - what would you do differently for a production system?
 - how would you monitor the system in production?
 - how would you approach upgrading the system?
 - what might a path-to-production for the system look like?

You can assume that any script should run natively on Linux, and you can assume that any meta code you provide “works”.