Take home test: API

Summary:

Design/implement stub REST API for resource limits for snaps using the https://github.com/mvo5/qrest-skeleton repo as a starting point.
Please do *not* fork the repo to your personal GH to avoid others seeing that you are attempting the test. Just clone it locally and work locally.

This should take 2h-4h. If it takes longer please simplify some of the requirements or talk to me.

If there are any questions you can reach me by mail with "michael.vogt@ubuntu.com" or on irc on freenode during European hours as "mvo".

Background:

Snaps are managed by snapd which talks over a REST API. One feature we plan to add to our REST API is allowing resource limits for snaps.

The core concept of this is a "quota-group" which is a named collection of quotas for system-resources. Groups can contain snaps or other quota groups.

The API should support:

- creating/removing quota-groups with user-controlled names
- adding/removing snaps/groups to a quota-group
- set/change/view resource limits for quota-groups (memory initially but should be designed with extensibility in mind)
- Query resource usage for quota-groups (i.e. get information how much quota of a quota group is already used)

Example:

Device has 768MB RAM and User wants:

- application quota group for sub-group UI and sub-group daemons assigns 512MB memory
    - UI group contains ui-snap1,ui-snap2
    - Daemon group contains daemon-snap1
- Special-snap quota group for special-snap with 256Mb memory

Goals:

Design the REST API (an informal txt/markdown to describe design considerations is fine, no need to be super-formal here) and implement the design in Go that talks to a mock backend. Use the github.com/mvo5/qrest-skeleton repo as a starting point/example. Feel free to modify anything in their (including backend.go) to best suit your solution

When finished, zip/tar the git repo and mail to michael.vogt@canonical.com

Non-Goals:

Avoid adding formal API descriptions like WSDL2, WADL (just to avoid spending too much time on this exercise, not because we have anything against them).
Ignore the XXX in backend.go (they are just my stream of consciousness)

References:

The repo with the skeleton: https://github.com/mvo5/qrest-skeleton/
Docs for the Gorilla web toolkit: https://www.gorillatoolkit.org/
Docs for the check.v1 test framework: http://labix.org/gocheck
The existing snapd REST API docs: https://snapcraft.io/docs/snapd-api