Technical Test, Engineer - Multipass

Background

Multipass is a multi-platform tool for allowing users to easily spin up Ubuntu Cloud instances. It's written in C++ and interfaces with a number of 3rd party libraries and technologies. Multipass strives to have a clean and concise CLI without adding too many options to burden end users.

These technologies/libraries include, but are not limited to:

- C++17
- gRPC
- REST
- cmake
- libssh
- Qt
- Poco
- JSON/YAML
- Simplestreams
- QEMU
- LXD
- libvirt
- Hyper-V
- Windows
- macOS
- Linux
- Snap/Windows/macOS packaging
- Networking

Introduction

The goal of this exercise is to get a sense of your understanding of C++, the standard library, and show how you may incorporate third party libraries to accomplish tasks. We are aiming for you to take approximately 4 hours to complete this, but take more time if necessary.

Requirements

This task is to define an interface and implement a derived class in C++ that fetches the latest Ubuntu Cloud image information in Simplestreams format from https://cloud-images.ubuntu.com/releases/streams/v1/com.ubuntu.cloud:released:download.json and provides methods that :

- Return a list of all currently supported Ubuntu releases.
- Return the current Ubuntu LTS version.
- Return the sha256 of the disk1.img item of a given Ubuntu release.

This will be wrapped in a CLI that outputs the above returned values depending on the option given.

The definition of the Simplestreams format is found at https://canonical-simplestreams.readthedocs-hosted.com/en/latest/reference/streams-product-jsonschema/#jsonschema-product.

You need only be concerned with the amd64 architecture for the cloud images. You may choose to use a third party JSON parser and HTTP downloader library if you wish.

It is recommended to build the project using a CMakeLists.txt and cmake and it should be able to be built on Linux, macOS, and Windows. If you do not have access to any of these platforms for testing, please state as such in the submission.

Be sure to document the methods. The completed assignment should be a tarball, including build instructions in a README and ideally including a git repository to show the different development steps taken.

If you find the requirements are open to some interpretation, that is intentional and please use your best judgment on the design and implementation and explain your decisions where necessary. Please reach out to us if you have any questions.