Dockers - Task #3

Goals

- Build an AI docker service based on JARVIS
- Use customized Ubuntu 18.04 image with python and git
- Modify the cloned repository to simplify execution
- Image construction will progress from interactive, standalone and mount usage

Estimated Time: 45-60 minutes

Prerequisites

All steps below should be performed in your personal Cloud9 environment. It is better to create a new folder with *Dockerfile* for it.

Steps

1. Instructions for base docker image "*jarvis-service*"
    a. Image is based on Ubuntu 18.04
    b. Install *python*, *python-pip* and *git* system packages
    c. Use *pip* to install required *aiml* python package
    d. Use ***WORKDIR*** to change file operations to target */root*
    e. Clone git repository https://github.com/A-I-Projects/J.A.R.V.I.S into jarvis folder
    f. Use ***WORKDIR*** to change file operations to target the new jarvis folder
    g. Make a backup copy of *script.py*
    h. Use ***COPY*** to replace the supplied *script.py* into the jarvis folder

    It may be advantageous to start testing individual commands in an interactive docker before finalizing the *Dockerfile*.

2. Start an interactive instance of the image and experiment with jarvis
    a. Run *python script.py* to launch the program
    b. Check various *.aiml files in the repo (basic_chat.aiml) for inputs JARVIS can understand
    c. Type different inputs and observe the output

3. Modify the *Dockerfile* to include an ***ENTRYPOINT*** that launches jarvis as in 2.a and rebuild the image
4. Start a new interactive instance of the image, but now without */bin/bash* argument

5. Modify the *Dockerfile* ***ENTRYPOINT*** to take predefined input filename and save output to a predefined filename (assume those files should reside in a mount folder = */mnt*)
    a. It is best to use ***ENTRYPOINT*** shell form and normal pipes to redirect stdin/out
6. Use mount to expose an external (Cloud9) directory to the instance (in */mnt*)
    a. Prepare an input file with words/sentences for JARVIS
    b. Run a new instance that launches JARVIS as a standalone process
    c. Check for file output after the instance finishes