Docker Exercise

General

In this exercise you are required to create a mini topology of 2 Docker containers.

Criteria for Success
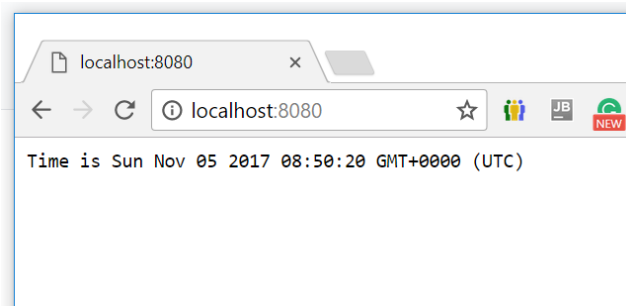
These are the criteria for success, <u>by importance</u>:

(1) It works.
(2) Code is clean.
(3) Improvement suggestions make sense

Tasks

(1) Create "Timeservice" container

The "Timerservice" is a docker image that contains a web server which returns a simple screen with the current time:

All source files for your container (e.g. dockerfile, scripts, etc.) should be put in folder `~/exam/timerservice`.
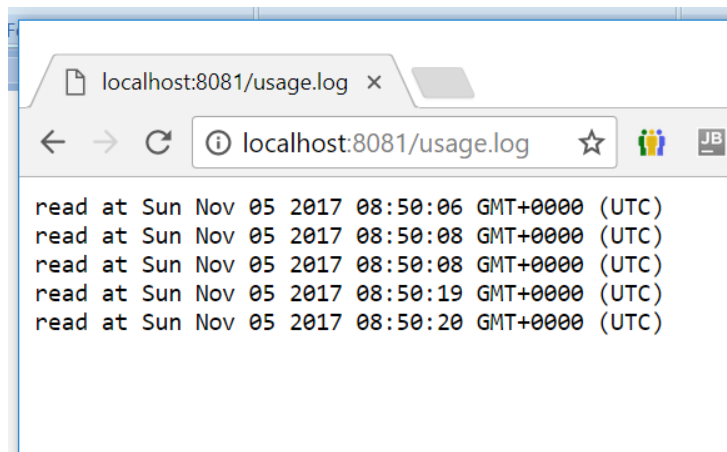
There are no limitations on the technology you use inside the docker container: python, node, ruby or whatever – use what makes you efficient.

(2) Create "Loggerservice" container

We would like to monitor calls to the "Timerservice" using a new container called "loggerservice".

- Modify "timerservice" so that it logs each time a user accesses it to the "<u>loggerservice</u>" (Note: this is not the web servers' access.log)

- The "loggerservice" is also a container with a web server. It displays the content of the "timerservice" usage log at url `/usage.log`:

All source files for your container (e.g. dockerfile, scripts, etc.) should be put in folder `~/exam/loggerservice`.

There are no limitations on the technology you use inside the docker container: python, node, ruby or whatever – use what makes you efficient.

(3) Start Topology with Both

Create a bash file which will start both servers mentioned and connect them:

- Timerservice on port 8080
- Loggerservice on port 8081
- Loggerservice displays the logs of Timerservice

The topology should be started using single "`./start.sh`" command in `~/exam`

(4) Bonus questions

(a) Write a script that lists all the layers in your **timerservice** docker image, ordered by their size:

```
0B        /bin/sh -c #(nop)  CMD ["/bin/sh"]
0B        /bin/sh -c #(nop)  CMD ["node" "/src/server.…
0B        /bin/sh -c #(nop)  ENV VERSION=v10.7.0 NPM_V…
0B        /bin/sh -c #(nop)  EXPOSE 4000
486B      /bin/sh -c cd /src && npm install
826B      /bin/sh -c #(nop) COPY dir:ad8b10e40ec0ca2ed…
4.2MB     /bin/sh -c #(nop) ADD file:6ee19b92d5cb1bf14…
63.6MB    /bin/sh -c apk add --no-cache curl make gcc …
```

(b) What is the most expensive layer? in our example it was the addition of
packages related to nodejs:

```
63.6MB  /bin/sh -c apk add --no-cache curl make gcc g++ python linux-headers binutils-gold gnupg libstdc++ &&    for server in ipv4.pool.sks-keyservers.net keyse
rver.pgp.com ha.pool.sks-keyservers.net; do      gpg --keyserver $server --recv-keys      94AE36675C464D64BAFA68DD7434390BDBE9B9C5      B9AE9905FFD7803F2571466
1B63B535A4C206CA9      77984A986EBC2AA786BC0F66B01FBB92821C587A      71DCFD284A79C3B38668286BC97EC7A07EDE3FC1      FD3A5288F042B6850C66B31F09FE44734EB7990E
    8FCCA13FEF1D0C2E91008E09770F7A9A5AE15600      C4F0DFFF4E8C1A8236409D08E73BC641CC11F4C8      DD8F2338BAE7501E3DD5AC78C273792F7D83545D && break;    done &&
  curl -sfSLO https://nodejs.org/dist/${VERSION}/node-${VERSION}.tar.xz &&    curl -sfSL https://nodejs.org/dist/${VERSION}/SHASUMS256.txt.asc | gpg --batch --de
crypt |     grep " node-${VERSION}.tar.xz\$" | sha256sum -c | grep ': OK$' &&    tar -xf node-${VERSION}.tar.xz &&    cd node-${VERSION} &&    ./configure --prefix
=/usr ${CONFIG_FLAGS} &&    make -j$(getconf _NPROCESSORS_ONLN) &&    make install &&    cd / &&    if [ -z "$CONFIG_FLAGS" ]; then      if [ -n "$NPM_VERSION" ]; th
en      npm install -g npm@${NPM_VERSION};      fi;      find /usr/lib/node_modules/npm -name test -o -name .bin -type d | xargs rm -rf;      if [ -n "$YARN_VERSI
ON" ]; then      for server in ipv4.pool.sks-keyservers.net keyserver.pgp.com ha.pool.sks-keyservers.net; do      gpg --keyserver $server --recv-keys
    6A010C5166006599AA17F08146C2130DFD2497F5 && break;      done &&      curl -sfSL -O https://yarnpkg.com/${YARN_VERSION}.tar.gz -O https://yarnpkg.com/${YAR
N_VERSION}.tar.gz.asc &&      gpg --batch --verify ${YARN_VERSION}.tar.gz.asc ${YARN_VERSION}.tar.gz &&      mkdir /usr/local/share/yarn &&      tar -xf ${YA
RN_VERSION}.tar.gz -C /usr/local/share/yarn --strip 1 &&      ln -s /usr/local/share/yarn/bin/yarn /usr/local/bin/ &&      ln -s /usr/local/share/yarn/bin/yar
npkg /usr/local/bin/ &&      rm ${YARN_VERSION}.tar.gz*;      fi;    fi &&    apk del curl make gcc g++ python linux-headers binutils-gold gnupg ${DEL_PKGS} &&
rm -rf ${RM_DIRS} /node-${VERSION}* /usr/share/man /tmp/* /var/cache/apk/*      /root/.npm /root/.node-gyp /root/.gnupg /usr/lib/node_modules/npm/man      /usr/li
b/node_modules/npm/doc /usr/lib/node_modules/npm/html /usr/lib/node_modules/npm/scripts
```