有关WIKI的需求烦请在https://servicedesk.iflytek.com/ 内提交，选择：质效平台运营 下分类即可

openvisualcloud环境搭建

由 陈明军创建, 最后修改于十二月 27, 2022

1.安装docker，在centos7环境中输入命令：

curl -fsSL https://get.docker.com | bash -s docker --mirror Aliyun

sudo yum install docker-compose

2.运行命令docker --version查看是否安装成功：

```
[root@vr-8k-test ~]# docker --version
Docker version 20.10.21, build baeda1f
[root@vr-8k-test ~]#
```

3.启动docker：

sudo systemctl start docker

4.拉取openvisualcloud代码并编译：

```
cd WebRTC-Sample/owt-server
mkdir build
cd build
cmake ..
mak
```

5.可能遇到的问题：

a). github拉取代码失败：失败后再拉取，或者使用代理，修改WebRTC-Sample/owt-server/image/owt-immersive/Dockerfile

```
FROM centos:7.6.1810 AS build
WORKDIR /home
SHELL ["/bin/bash", "-o", "pipefail", "-c"]

# COMMON BUILD TOOLS
RUN yum install -y -q bzip2 make autoconf libtool git wget ca-certifi
RUN git config --global http.proxy http://10.40.164.69:8118
RUN git config --global https.proxy https://10.40.164.69:8118
# Install cmake
```

b).编译owt-server模块时会连续编译多个项目，可能因为网络问题拉取github失败，可将不同模块分开，这样之前编译成功的不会重新再编译：

```
# 3. Clone webrtc source code and patch

# hadolint ignore=SC1091
RUN git clone -b ${OWT_BRANCH} ${OWTSERVER_REPO} && \
    source /opt/rh/devtoolset-7/enable && \

    cd ${SERVER_PATH} && git reset --hard ${OWTSERVER_COMMIT} && \
    curl https://patch-diff.githubusercontent.com/raw/open-webrtc-toolkit/owt-se

    # Install node modules for owt
RUN npm config set proxy=${http_proxy} && \
    npm config set https-proxy=${http_proxy} && \
    npm install -g --loglevel error node-gyp@v6.1.0 grunt-cli underscore jsdoc &
    cd ${SERVER_PATH} && npm install nan

    # Get openh264 for owt
RUN cd ${SERVER_PATH}/third_party && \
    mkdir openh264 && cd openh264 && \
    wget ${OPENH264_SOURCE} && \
    wget ${OPENH264_BINARY} && \
    tar xzf ${OPENH264_SOURCENAME} openh264-${OPENH264_MAJOR}.${OPENH264_MINOR}.
    ln -s -v openh264-${OPENH264_MAJOR}.${OPENH264_MINOR}.0/codec codec && \
    bzip2 -d ${OPENH264_BINARYNAME}.bz2 && \
    ln -s -v ${OPENH264_BINARYNAME} libopenh264.so.${OPENH264_SOVER} && \
    ln -s -v libopenh264.so.${OPENH264_SOVER} libopenh264.so && \
    echo 'const char* stub() {return "this is a stub lib";}' > pseudo-openh264.c
    gcc pseudo-openh264.cpp -fPIC -shared -o pseudo-openh264.so

    # Get licode for owt
RUN cd ${SERVER_PATH}/third_party && git clone ${LICODE_REPO} && \
```

c).拉取chrome webrtc代码时可能会一直retry失败，可加上http代理：

```
    # Install webrtc for owt
RUN cd ${SERVER_PATH}/third_party && mkdir webrtc  && cd webrtc &&\
    export GIT_SSL_NO_VERIFY=1 && \
    export http_proxy=http://10.40.164.69:8118 && \
    export https_proxy=http://10.40.164.69:8118 && \
    git clone -b 59-server ${WEBRTC_REPO} src && cd src && \
    git reset --hard ${WEBRTC_COMMIT} && \
    ./tools-woogeen/install.sh && \
    patch -p1 < ${SERVER_PATH}/scripts/patches/0001-Implement-RtcpFOVObserver.patch && \
    ./tools-woogeen/build.sh
```

d).编译js client时候可能报gcc版本不支持，修改：

```
    # Get js client sdk for owt
RUN echo "----------Get js client sdk for owt start" && \
    source /opt/rh/devtoolset-7/enable && \
    cd /home && git clone -b ${OWT_BRANCH_JS} ${OWT_SDK_REPO} && cd owt-client-javascript/scripts && git re
    export LD_LIBRARY_PATH=/usr/local/lib64 && \
    #Build and pack owt
    cd ${SERVER_PATH} && export CPLUS_INCLUDE_PATH=/usr/local/include/svt-hevc && export PKG_CONFIG_PATH=/u
    ./scripts/pack.js -t all --install-module --no-pseudo --sample-path /home/owt-client-javascript/dist/sa
    echo "----------Get js client sdk for owt end"
```

e).npm过程中中断，后续再重新编译可能报npm失败，修改：

```
RUN cd /home/scripts &&\
    npm cache clean --force && \
    npm config set proxy=${http_proxy} && \
    npm config set https-proxy=${http_proxy} && \
    npm install
```

6.在ubuntu22.04上编译player

cd WebRTC-Sample/owt-linux-player

build owt linux sdk and dependencies
./build_webrtc_linux_client_sdk.sh

build player
./build_player.sh

7.编译过程中可能遇到的问题：

a).build_webrtc_linux_client_sdk.sh每一次编译都是删除上一次编译的文件，这样太麻烦，可修改脚本不删除，类似都这样：

```
install_openssl() {
    cd ${DEPS}
    local SSL_VERSION="1_1_1h"
    #rm -rf openssl* OpenSSL*

    #wget -c https://github.com/openssl/openssl/archive/OpenSSL_${SSL_VERSION}.tar.gz
    #tar xf OpenSSL_${SSL_VERSION}.tar.gz
    cd openssl-OpenSSL_${SSL_VERSION}

    #./config shared -m64 --prefix=${PREFIX} --openssldir=${PREFIX}
    #make -j
    #make install
}
```

b).

https://blog.csdn.net/VictoriaW/article/details/60480269?locationNum=1&fps=1

https://blog.csdn.net/dong_beijing/article/details/80654464

无标签

ATLASSIAN