Article
Guillaume Rongier · Mar 8, 2023 5m read

Keep your docker iris images in shape ;)

iris-docker-multi-stage-script

A python script to keep your docker iris images in shape ;)

Before **After**

Witout changing your dockerfile or your code you can reduce the size of your image by 50% or more !

TL;DR

Name the builder image builder and the final image final and add this to end of your Dockerfile:

Modify your Dockerfile to use a multi-stage build:

```
ARG IMAGE=intersystemsdc/irishealth-community:latest
FROM $IMAGE as builder
```

Add this to end of your Dockerfile:

```
FROM $IMAGE as final

ADD --chown=${ISC_PACKAGE_MGRUSER}:${ISC_PACKAGE_IRISGROUP} https://github.com/grongi
erisc/iris-docker-multi-stage-script/releases/latest/download/copy-
data.py /irisdev/app/copy-data.py

RUN --mount=type=bind,source=/,target=/builder/root,from=builder \
    cp -f /builder/root/usr/irissys/iris.cpf /usr/irissys/iris.cpf && \
    python3 /irisdev/app/copy-data.py -c /usr/irissys/iris.cpf -d /builder/root/
```

Boom! You're done!

Usage

```
usage: copy-data.py [-h] -c CPF -d DATA_DIR [--csp] [-p] [-o OTHER [OTHER ...]]

Copy data from a directory to the IRIS data directory

optional arguments:
  -h, --help            show this help message and exit
  -c CPF, --cpf CPF     path to the iris.cpf file
  -d DATA_DIR, --data_dir DATA_DIR
```

```
                        path to the directory where the data files are located
  --csp                 toggle the copy of the whole CSP folder
  -p, --python          toggle the copy of python libs
  -o OTHER [OTHER ...], --other OTHER [OTHER ...]
                        toggle the copy of other folders
```

How to use it

First have a look at a non-multi-stage Dockerfile for iris:

```
ARG IMAGE=intersystemsdc/irishealth-community:latest
FROM $IMAGE

WORKDIR /irisdev/app
RUN chown ${ISC_PACKAGE_MGRUSER}:${ISC_PACKAGE_IRISGROUP} /irisdev/app
USER ${ISC_PACKAGE_MGRUSER}

# copy source code
COPY src src
COPY misc misc
COPY data/fhir fhirdata
COPY iris.script /tmp/iris.script
COPY fhirUI /usr/irissys/csp/user/fhirUI

# run iris and initial
RUN iris start IRIS \
    && iris session IRIS < /tmp/iris.script \
    && iris stop IRIS quietly
```

This is a simple Dockerfile that will build an image with the iris source code and the fhir data. It will also run the iris.script to create the fhir database and load the data.

With this kind of dockerfile you will end up with a big image. This is not a problem if you are using a CI/CD pipeline to build your images. But if you are using this image in production you will end up with a big image that will take a lot of space on your server.

Then have a look at a multi-stage Dockerfile for iris

```
ARG IMAGE=intersystemsdc/irishealth-community:latest
FROM $IMAGE as builder

WORKDIR /irisdev/app
RUN chown ${ISC_PACKAGE_MGRUSER}:${ISC_PACKAGE_IRISGROUP} /irisdev/app
USER ${ISC_PACKAGE_MGRUSER}

# copy source code
COPY src src
COPY misc misc
COPY data/fhir fhirdata
COPY iris.script /tmp/iris.script
COPY fhirUI /usr/irissys/csp/user/fhirUI

# run iris and initial
RUN iris start IRIS \
    && iris session IRIS < /tmp/iris.script \
```

```
    && iris stop IRIS quietly

# copy data from builder
FROM $IMAGE as final

ADD --chown=${ISC_PACKAGE_MGRUSER}:${ISC_PACKAGE_IRISGROUP} https://github.com/grongi
erisc/iris-docker-multi-stage-script/releases/latest/download/copy-
data.py /irisdev/app/copy-data.py

RUN --mount=type=bind,source=/,target=/builder/root,from=builder \
    cp -f /builder/root/usr/irissys/iris.cpf /usr/irissys/iris.cpf && \
    python3 /irisdev/app/copy-data.py -c /usr/irissys/iris.cpf -d /builder/root/
```

This is a multi-stage Dockerfile that will build an image with the iris source code and the fhir data. It will also run the iris.script to create the fhir database and load the data. But it will also copy the data from the builder image to the final image. This will reduce the size of the final image.

Let read in details the multi-stage Dockerfile:

```
ARG IMAGE=intersystemsdc/irishealth-community:latest
FROM $IMAGE as builder
```

Define the base image and the name of the builder image

```
WORKDIR /irisdev/app
RUN chown ${ISC_PACKAGE_MGRUSER}:${ISC_PACKAGE_IRISGROUP} /irisdev/app
USER ${ISC_PACKAGE_MGRUSER}

# copy source code
COPY src src
COPY misc misc
COPY data/fhir fhirdata
COPY iris.script /tmp/iris.script
COPY fhirUI /usr/irissys/csp/user/fhirUI

# run iris and initial
RUN iris start IRIS \
    && iris session IRIS < /tmp/iris.script \
    && iris stop IRIS quietly
```

Basically the same as the non-multi-stage Dockerfile

```
FROM $IMAGE as final
```

Start with the base image

```
ADD --chown=${ISC_PACKAGE_MGRUSER}:${ISC_PACKAGE_IRISGROUP} https://github.com/grongi
erisc/iris-docker-multi-stage-script/releases/latest/download/copy-
data.py /irisdev/app/copy-data.py
```

Add the copy-data.py script to the image with the right user and group

```
RUN --mount=type=bind,source=/,target=/builder/root,from=builder \
    cp -f /builder/root/usr/irissys/iris.cpf /usr/irissys/iris.cpf && \
    python3 /irisdev/app/copy-data.py -c /usr/irissys/iris.cpf -d /builder/root/
```

A lot is happening here.

First we are using the --mount option to mount the builder image.
- --mount=type=bind is the type of mount
- source=/ is the root of the builder image
- target=/builder/root is the root of the builder image mounted in the final
- from=builder is the name of the builder image

Then we are copying the iris.cpf file from the builder image to the final image.

```
cp -f /builder/root/usr/irissys/iris.cpf /usr/irissys/iris.cpf
```

Finally we are running the copy-data.py script to copy the data from the builder image to the final image.

```
python3 /irisdev/app/copy-data.py -c /usr/irissys/iris.cpf -d /builder/root/
```

Side by side comparison

Non multi-stage Dockerfile

```
ARG IMAGE=intersystemsdc/irishealth-community:latest
FROM $IMAGE

WORKDIR /irisdev/app
RUN chown ${ISC_PACKAGE_MGRUSER}:${ISC_PACKAGE_IRISGROUP} /irisdev/app
USER ${ISC_PACKAGE_MGRUSER}

COPY . .

RUN iris start IRIS \
    && iris session IRIS < /tmp/iris.script \
    && iris stop IRIS quietly
```

Multi-stage Dockerfile

```
ARG IMAGE=intersystemsdc/irishealth-community:latest
FROM $IMAGE as builder

WORKDIR /irisdev/app
RUN chown ${ISC_PACKAGE_MGRUSER}:${ISC_PACKAGE_IRISGROUP} /irisdev/app
USER ${ISC_PACKAGE_MGRUSER}

COPY . .
```

```
RUN iris start IRIS \
    && iris session IRIS < /tmp/iris.script \
    && iris stop IRIS quietly

FROM $IMAGE as final

ADD --chown=${ISC_PACKAGE_MGRUSER}:${ISC_PACKAGE_IRISGROUP} https://github.com/grongi
erisc/iris-docker-multi-stage-script/releases/latest/download/copy-
data.py /irisdev/app/copy-data.py

RUN --mount=type=bind,source=/,target=/builder/root,from=builder \
    cp -f /builder/root/usr/irissys/iris.cpf /usr/irissys/iris.cpf && \
    python3 /irisdev/app/copy-data.py -c /usr/irissys/iris.cpf -d /builder/root/
```

#Docker #InterSystems IRIS