**CREATING OPTIMISED DOCKER IMAGES USING MULTI STAGE BULIDS:**

CREATE DOCKER FILES

The Multi-Stage feature allows a single Dockerfile to contain multiple stages in order to produce the desired, optimised, Docker Image.

Previously, the problem would have been solved with two Dockerfiles. One file would have the steps to build the binary and artifacts using a development container, the second would be optimised for production and not include the development tools.

MULTI STAGE DOCKER FILE

Using the editor, create a Multi-Stage Dockerfile. The first stage using the Golang SDK to build a binary. The second stage copies the resulting binary into a optimised Docker Image.

Copy to Editor*# First Stage*

FROM golang:1.6-alpine

RUN mkdir /app

ADD . /app/

WORKDIR /app

RUN CGO\_ENABLED=0 GOOS=linux go build -a -installsuffix cgo -o main .

*# Second Stage*

FROM alpine

EXPOSE 80

CMD ["/app"]

*# Copy from first stage*

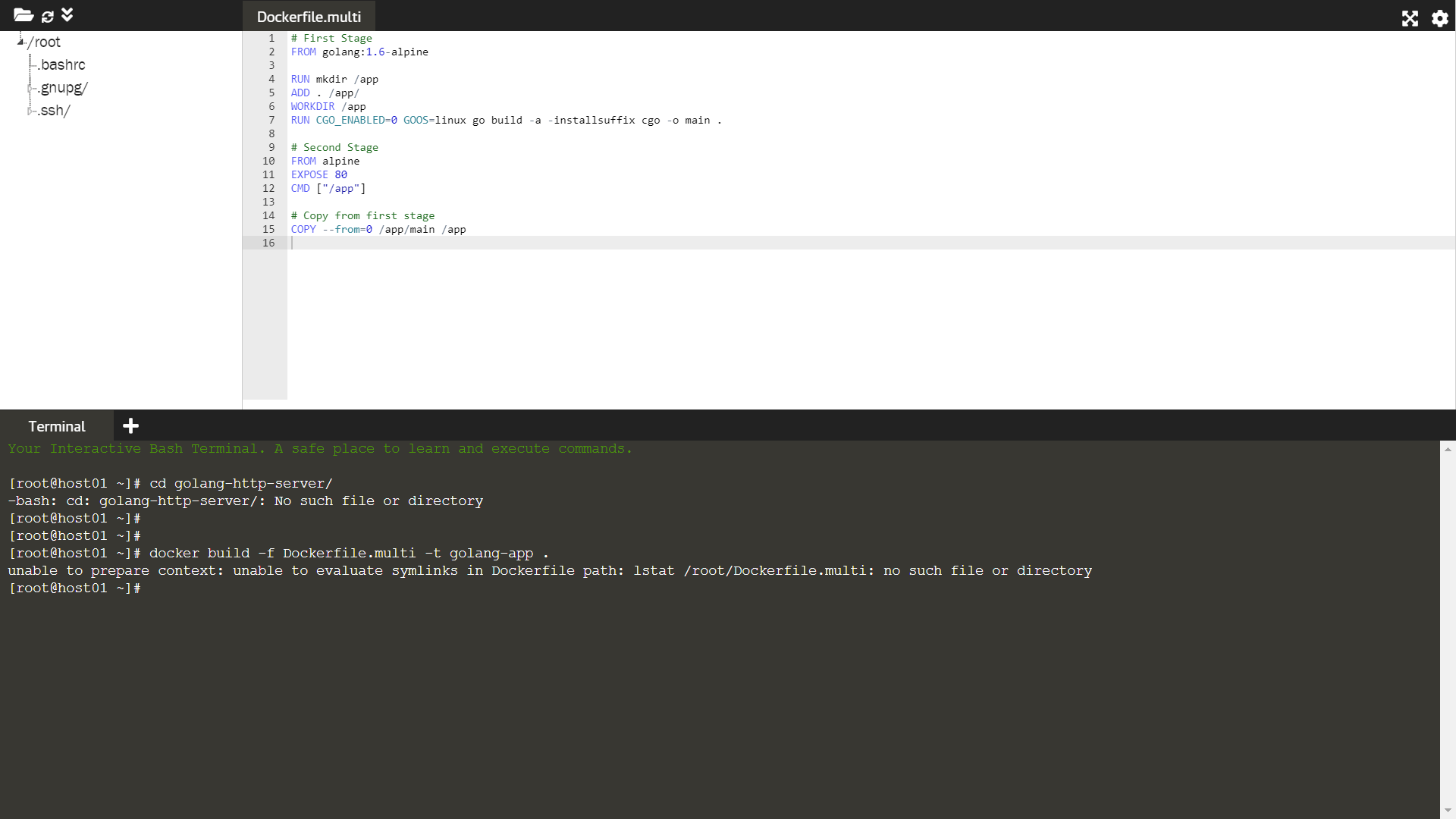
COPY --from=0 /app/main /app

BUILD MULTI STAGE BULIDERS:

With the new syntax for the Dockerfile in place, the build process is identical to previously.

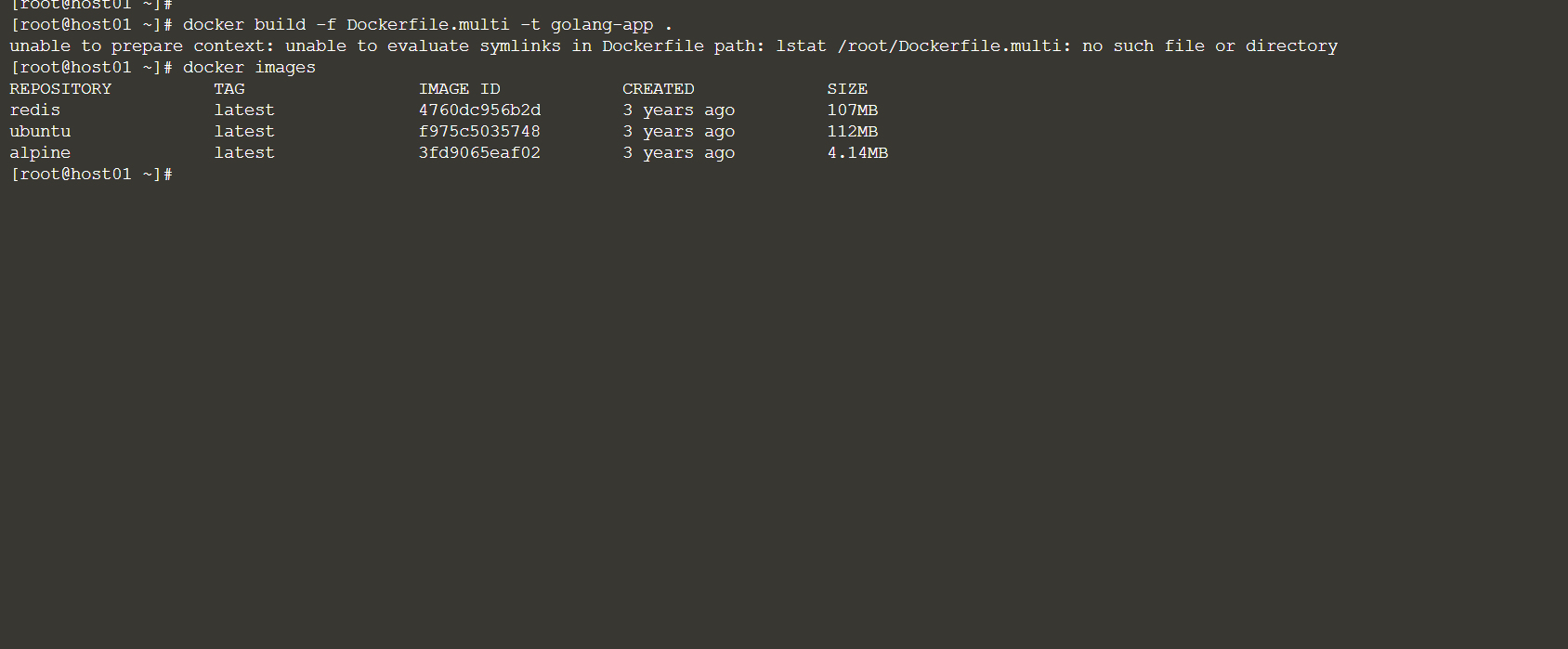
Create the desired Docker Image using the build command below.

docker build -f Dockerfile.multi -t golang-app .



The result will be two images. One untagged that was used for the first stage and the second, smaller image, our target image.

docker images



TEST IMAGE:

The image can be launched and deployed without any changes required.

docker run -d -p 80:80 golang-app

curl localhost

