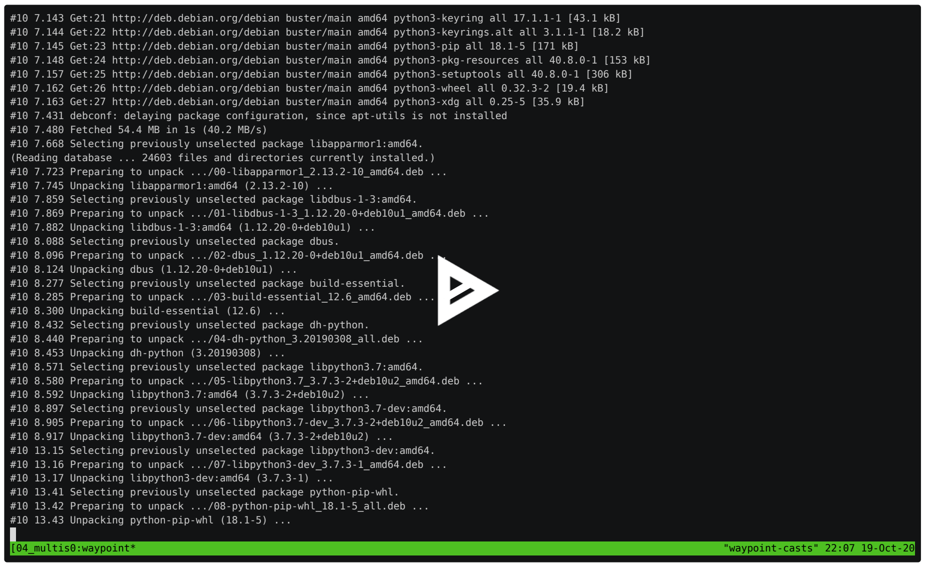
multistage dockerfile deployment



```
#10 7.143 Get:21 http://deb.debian.org/debian buster/main amd64 python3-keyring all 17.1.1-1 [43.1 kB]
#10 7.144 Get:22 http://deb.debian.org/debian buster/main amd64 python3-keyrings.alt all 3.1.1-1 [18.2 kB]
#10 7.145 Get:23 http://deb.debian.org/debian buster/main amd64 python3-pip all 18.1-5 [171 kB]
#10 7.148 Get:24 http://deb.debian.org/debian buster/main amd64 python3-pkg-resources all 40.8.0-1 [153 kB]
#10 7.157 Get:25 http://deb.debian.org/debian buster/main amd64 python3-setuptools all 40.8.0-1 [306 kB]
#10 7.162 Get:26 http://deb.debian.org/debian buster/main amd64 python3-wheel all 0.32.3-2 [19.4 kB]
#10 7.163 Get:27 http://deb.debian.org/debian buster/main amd64 python3-xdg all 0.25-5 [35.9 kB]
#10 7.431 debconf: delaying package configuration, since apt-utils is not installed
#10 7.480 Fetched 54.4 MB in 1s (40.2 MB/s)
#10 7.668 Selecting previously unselected package libapparmor1:amd64.
(Reading database ... 24603 files and directories currently installed.)
#10 7.723 Preparing to unpack .../00-libapparmor1_2.13.2-10_amd64.deb ...
#10 7.745 Unpacking libapparmor1:amd64 (2.13.2-10) ...
#10 7.859 Selecting previously unselected package libdbus-1-3:amd64.
#10 7.869 Preparing to unpack .../01-libdbus-1-3_1.12.20-0+deb10u1_amd64.deb ...
#10 7.882 Unpacking libdbus-1-3:amd64 (1.12.20-0+deb10u1) ...
#10 8.088 Selecting previously unselected package dbus.
#10 8.096 Preparing to unpack .../02-dbus_1.12.20-0+deb10u1_amd64.deb ...
#10 8.124 Unpacking dbus (1.12.20-0+deb10u1) ...
#10 8.277 Selecting previously unselected package build-essential.
#10 8.285 Preparing to unpack .../03-build-essential_12.6_amd64.deb ...
#10 8.300 Unpacking build-essential (12.6) ...
#10 8.432 Selecting previously unselected package dh-python.
#10 8.440 Preparing to unpack .../04-dh-python_3.20190308_all.deb ...
#10 8.453 Unpacking dh-python (3.20190308) ...
#10 8.571 Selecting previously unselected package libpython3.7:amd64.
#10 8.580 Preparing to unpack .../05-libpython3.7_3.7.3-2+deb10u2_amd64.deb ...
#10 8.592 Unpacking libpython3.7:amd64 (3.7.3-2+deb10u2) ...
#10 8.897 Selecting previously unselected package libpython3.7-dev:amd64.
#10 8.905 Preparing to unpack .../06-libpython3.7-dev_3.7.3-2+deb10u2_amd64.deb ...
#10 8.917 Unpacking libpython3.7-dev:amd64 (3.7.3-2+deb10u2) ...
#10 13.15 Selecting previously unselected package libpython3-dev:amd64.
#10 13.16 Preparing to unpack .../07-libpython3-dev_3.7.3-1_amd64.deb ...
#10 13.17 Unpacking libpython3-dev:amd64 (3.7.3-1) ...
#10 13.41 Selecting previously unselected package python-pip-whl.
#10 13.42 Preparing to unpack .../08-python-pip-whl_18.1-5_all.deb ...
#10 13.43 Unpacking python-pip-whl (18.1-5) ...

[04_multis0:waypoint*                                                          "waypoint-casts" 22:07 19-Oct-20
```

as you have already noticed that the docker images built with `pack` plugin are hardly minimal images. I would recommend having a multi-stage builder docker file in which in one stage , application is built and in the second stage, application runs.

to make multi-stage builds work, use a `hook` (or github actions pipeline) to build the image first and then use `docker-pull` plugin to add waypoint entrypoint to the image and push it to the artifact repository.

this approach has the benefit of seamless integration with github actions pipeline.

this approach has two stages :

- pull target git repository in docker container and run build
- move the artifact from the first stage into a second minimal stage

to make this work, the artifact must be statically linked or self contained.

in this example, we will make a simple selfcontainer django based echo webserver.

the following is the base template for our docker image

```
FROM python:alpine as base

ARG GITHUB_REPOSITORY_OWNER
ENV GITHUB_REPOSITORY_OWNER $GITHUB_REPOSITORY_OWNER

ARG GITHUB_REPOSITORY
ENV GITHUB_REPOSITORY $GITHUB_REPOSITORY
```

```
ARG GITHUB_ACTOR
ENV GITHUB_ACTOR $GITHUB_ACTOR

ARG GITHUB_TOKEN
ENV GITHUB_TOKEN $GITHUB_TOKEN

ENV TERM=xterm
# [NOTE] => packages installed here are some of the most common base build dependencie for
RUN echo "http://dl-cdn.alpinelinux.org/alpine/edge/main" > /etc/apk/repositories && \
    echo "http://dl-cdn.alpinelinux.org/alpine/edge/community" >> /etc/apk/repositories && \
    echo "http://dl-cdn.alpinelinux.org/alpine/edge/testing" >> /etc/apk/repositories && \
    apk upgrade -U -a && \
    apk add build-base make git bash ncurses-static curl libressl-dev musl-dev libffi-dev
SHELL ["/bin/bash", "-c"]
# [TODO] => install and customize your image how ever you like here
RUN git clone "https://$GITHUB_ACTOR:$GITHUB_TOKEN@github.com/<repo_owner>/<repo_name>.git"
WORKDIR /workspace/<repo_name>
# [TODO] => add build commands here
FROM python:alpine
COPY --from=base /workspace/<artifact> /<artifact>
ENTRYPOINT ["/<artifact>"]
```

the environment variables defined in this file are present in github actions exection pipeline. we will build the image with github actions before using `docker-pull` to inject waypoint entrypoint and pushing it to a docker repository. to see how building the image with github actions would look like, look into this github `repo`

in this demo, we are not using github actions but a `hook` to build the image locally.

our target repo is `da-moon/upstream-gen` I have already create a github token that can pull the repo. to securily use the token, store it in `~/.git_token` and use Docker build secrets to inject it into the image.

first, lets setup the docker image :

```
mkdir -p /tmp/upstream-gen
cat << EOF | tee /tmp/upstream-gen/Dockerfile
# syntax = docker/dockerfile:1.0-experimental

FROM python:buster as base

ARG GITHUB_REPOSITORY_OWNER
ENV GITHUB_REPOSITORY_OWNER \$GITHUB_REPOSITORY_OWNER

ARG GITHUB_REPOSITORY
```

```
ENV GITHUB_REPOSITORY \$GITHUB_REPOSITORY

ARG GITHUB_ACTOR
ENV GITHUB_ACTOR \$GITHUB_ACTOR

ENV TERM=xterm
# [NOTE] => git token is stored at '\$HOME/.git_token'
ENV PIP_USER=false
RUN export DEBIAN_FRONTEND=noninteractive; \
    apt-get update && \
    apt-get install -y  make git curl wget build-essential python3 python3-pip
SHELL ["/bin/bash", "-c"]
RUN mkdir -p "/workspace" && \
    mkdir -p "~/.local/bin" && \
    mkdir -p "~/.poetry/bin" && \
    curl -sSL https://raw.githubusercontent.com/python-poetry/poetry/master/get-poetry.py |
    python3 -m pip install pex dephell[full] && \
    dephell --version && \
    pex --version
RUN git clone "https://\$GITHUB_ACTOR:\$(cat \$HOME/.git_token)@github.com/da-moon/upstream-
WORKDIR /workspace/upstream-gen
RUN make python-pex && \
    dist/pex/upstream-gen version
FROM python:buster
COPY --from=base /workspace/upstream-gen/dist/pex/upstream-gen /upstream-gen
ENTRYPOINT ["/upstream-gen"]
CMD ["--log", "TRACE" ,"server"]
EOF
sed -i -e 's/\s\s*/ /g' -e '/^\s*$/d' /tmp/upstream-gen/Dockerfile
```

so, we will create a script file to have waypoint run in a hook to build the image.

```
cat << EOF | tee /tmp/upstream-gen/build.sh
#!/usr/bin/env bash

export GITHUB_REPOSITORY=upstream-gen
export GITHUB_REPOSITORY_OWNER=da-moon
export GITHUB_ACTOR=da-moon
docker system prune -f && \
DOCKER_BUILDKIT=1 docker build \
        --progress=plain \
        --secret id=github_token,src="\$HOME/.git_token" \
        --build-arg GITHUB_REPOSITORY="\$GITHUB_REPOSITORY_OWNER/\$GITHUB_REPOSITORY" \
        --build-arg GITHUB_REPOSITORY_OWNER=\$GITHUB_REPOSITORY_OWNER \
        --build-arg GITHUB_ACTOR=\$GITHUB_ACTOR \
        -t "fjolsvin/\$GITHUB_REPOSITORY:latest" . && \
docker push "fjolsvin/\$GITHUB_REPOSITORY:latest"
```

```
EOF
chmod +x /tmp/upstream-gen/build.sh && \
sed -i -e 's/\s\s*/ /g' -e '/^\s*$/d' /tmp/upstream-gen/build.sh
```

as you can see, the script reads a git token stored in `~/.git_token` so generate a token and store it there before moving along.

we will use the following `waypoint.hcl` rebuild the image and inject waypoint

```
cat << EOF | tee /tmp/upstream-gen/waypoint.hcl
project = "waypoint-http-echo-example"
app "waypoint-http-echo-example" {
  labels = {
    "service" = "waypoint-http-echo-example",
  }
  build {
    hook {
      when = "before"
      command = ["./build.sh"]
    }
    use "docker-pull" {
        image = "fjolsvin/upstream-gen"
        tag = "latest"
        encoded_auth = file("~/.docker_auth")
    }
    registry {
      use "docker" {
        image = "fjolsvin/waypoint-http-echo-example"
        tag = "latest"
        encoded_auth = file("~/.docker_auth")
      }
    }
  }
  deploy {
    use "nomad" {
      datacenter = "dc1"
      region = "global"
      replicas = 1
      service_port = 9090
    }
  }
}
EOF
pushd /tmp/upstream-gen/ && \
waypoint init && \
NOMAD_ADDR="http://10.33.235.43:4646" waypoint up
popd
```

```
rm ~/.git_token
```