Title: multi-stage build in docker compose?

Post Body:

How can I specify multi-stage build with in a `docker-compose.yml`?

For each variant (e.g. dev, prod...) I have a multi-stage build with 2 docker files:

- dev: `Dockerfile.base` + `Dockerfile.dev`
- or prod: `Dockerfile.base` + `Dockerfile.prod`

File `Dockerfile.base` (common for all variants):

```
FROM python:3.6 RUN apt-get update && apt-get upgrade -y RUN pip install pipenv pip COPY Pipfile ./ # some more common configu
```

File `Dockerfile.dev`:

```
FROM flaskapp:base RUN pipenv install --system --skip-lock --dev ENV FLASK_ENV development ENV FLASK_DEBUG 1
```

File `Dockerfile.prod`:

```
FROM flaskapp:base RUN pipenv install --system --skip-lock ENV FLASK_ENV production
```

Without docker-compose, I can build as:

```
# Building dev docker build --tag flaskapp:base -f Dockerfile.base . docker build --tag flaskapp:dev -f Dockerfile.dev . # or
```

According to the compose-file doc, I can specify a Dockerfile to build.

```
# docker-compose.yml version: '3' services:   webapp:     build:       context: ./dir       dockerfile: Dockerfile-alternate
```

But how can I specify 2 Dockerfiles in `docker-compose.yml` (for multi-stage build)?

Accepted Answer:

As mentioned in the comments, a multi-stage build involves a single Dockerfile to perform multiple stages. What you have is a common base image.

You could convert these to a non-traditional multi-stage build with a syntax like (I say non-traditional because you do not perform any copying between the layers and instead use just the from line to pick from a prior stage):

```
FROM python:3.6 as base RUN apt-get update && apt-get upgrade -y RUN pip install pipenv pip COPY Pipfile ./ # some more common
```

Then you can build one stage or another using the `--target` syntax to build, or a compose file like:

```
# docker-compose.yml version: '3.4' services:   webapp:     build:       context: ./dir       dockerfile: Dockerfile       tar
```

The biggest downside is the current build engine will go through every stage until it reaches the target. Build caching can mean that's only a sub-second process. And BuildKit which is coming out of experimental in 18.09 and will need upstream support from docker-compose will be more intelligent about only running the needed commands to get your desired target built.

All that said, I believe this is trying to fit a square peg in a round hole. The docker-compose developer is encouraging users to move away from doing the build within the compose file itself since it's not supported in swarm mode. Instead, the recommended solution is to perform builds with a CI/CD build server, and push those images to a registry. Then you can run the same compose file with `docker-compose` or `docker stack deploy` or even some k8s equivalents, without needing to redesign your workflow.

Highest Rated Answer:

you can use as well concating of docker-compose files, with including both `dockerfile` pointing to your existing dockerfiles and run `docker-compose -f docker-compose.yml -f docker-compose.prod.yml build`