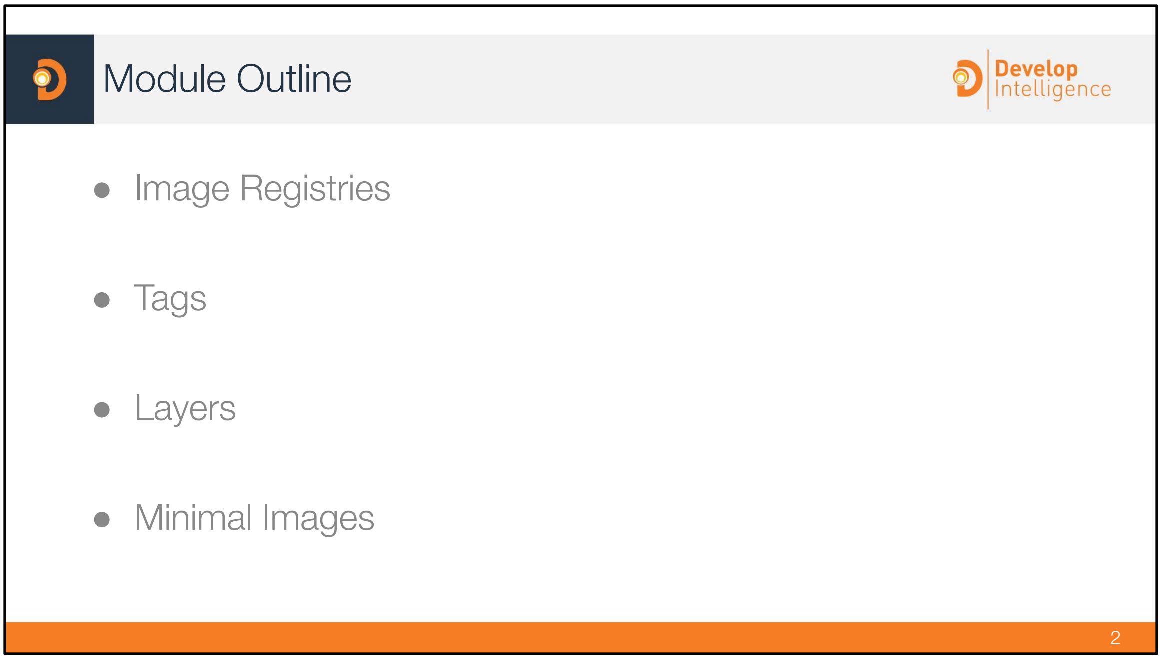
Docker Image Deep Dive

Develop Intelligence



Module Outline

Develop Intelligence

- Image Registries
- Tags
- Layers
- Minimal Images

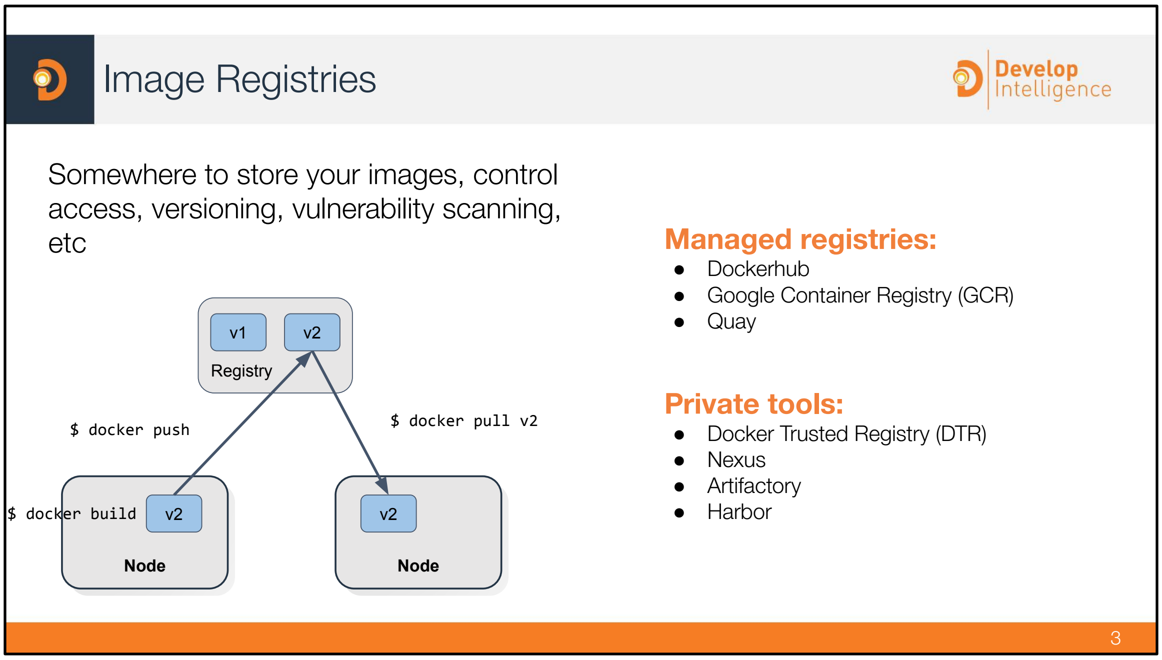
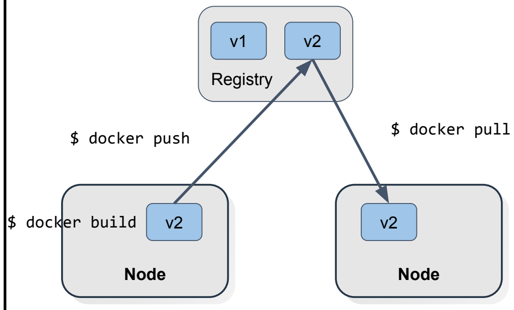


Image Registries

Somewhere to store your images, control access, versioning, vulnerability scanning, etc



Managed registries:

- Dockerhub
- Google Container Registry (GCR)
- Quay

Private tools:

- Docker Trusted Registry (DTR)
- Nexus
- Artifactory
- Harbor

3

hub.docker.com

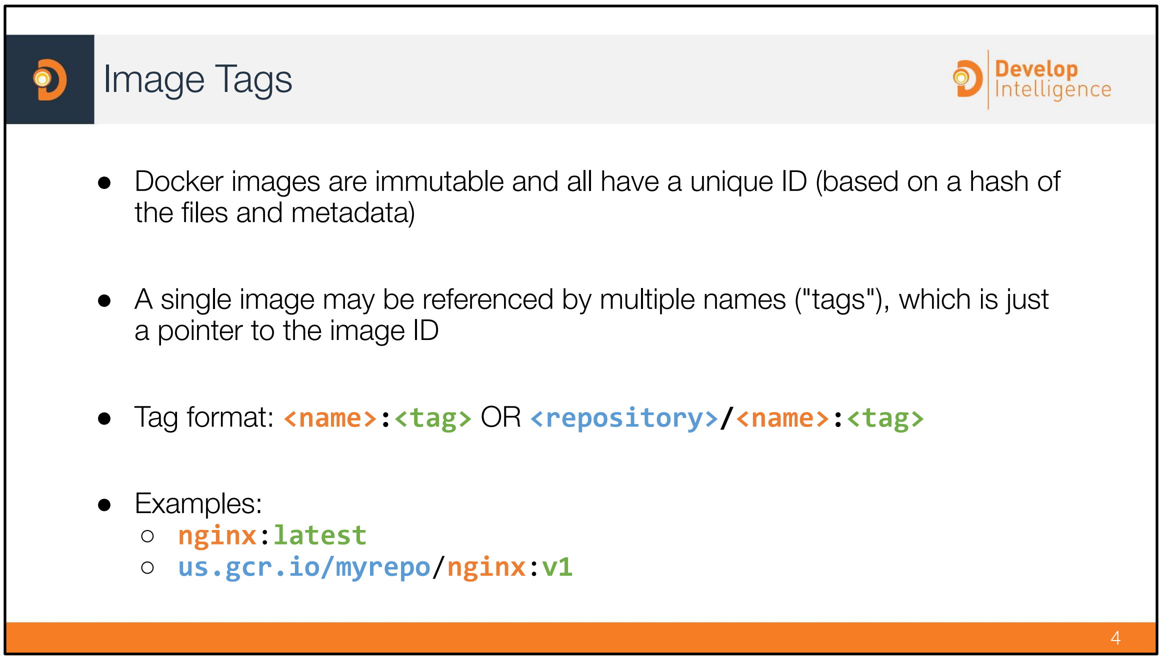


Image Tags

- Docker images are immutable and all have a unique ID (based on a hash of the files and metadata)
- A single image may be referenced by multiple names ("tags"), which is just a pointer to the image ID
- Tag format: `<name>:<tag>` OR `<repository>/<name>:<tag>`
- Examples:
  - `nginx:latest`
  - `us.gcr.io/myrepo/nginx:v1`

4

If you try to use <name>:<tag> and the image does not exist locally, docker will look in it's default registry(s). If a tag is not provided, ":latest" will be used by default. ":latest" is also added automatically when building a new image.

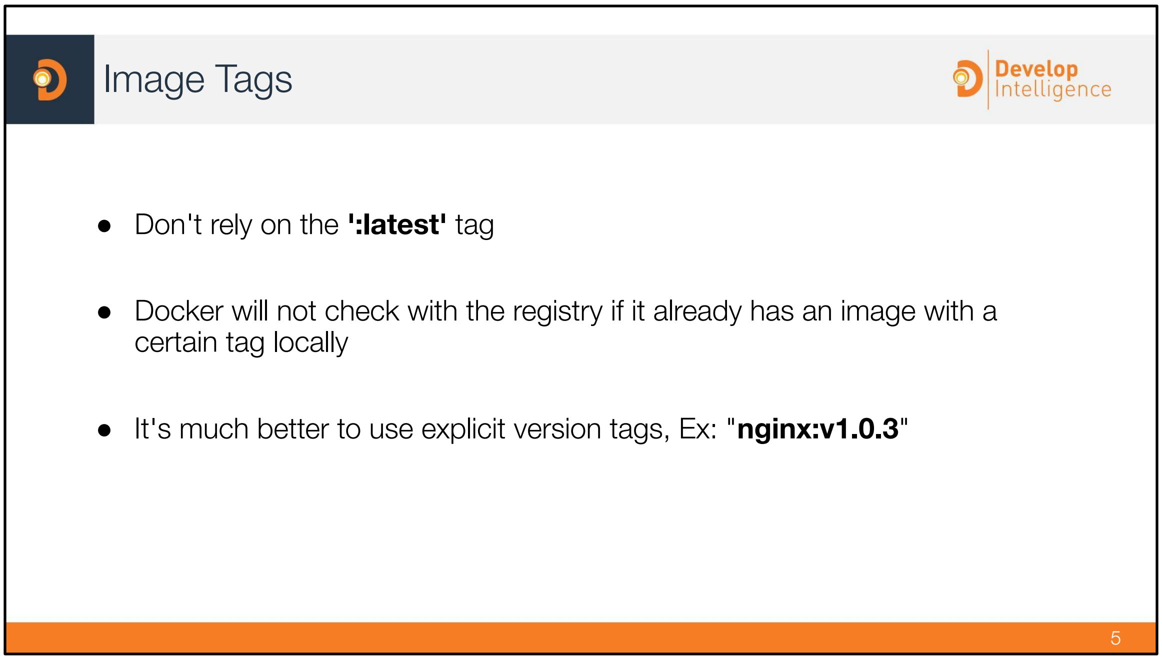


Image Tags

Develop Intelligence

- Don't rely on the **':latest'** tag
- Docker will not check with the registry if it already has an image with a certain tag locally
- It's much better to use explicit version tags, Ex: "**nginx:v1.0.3**"

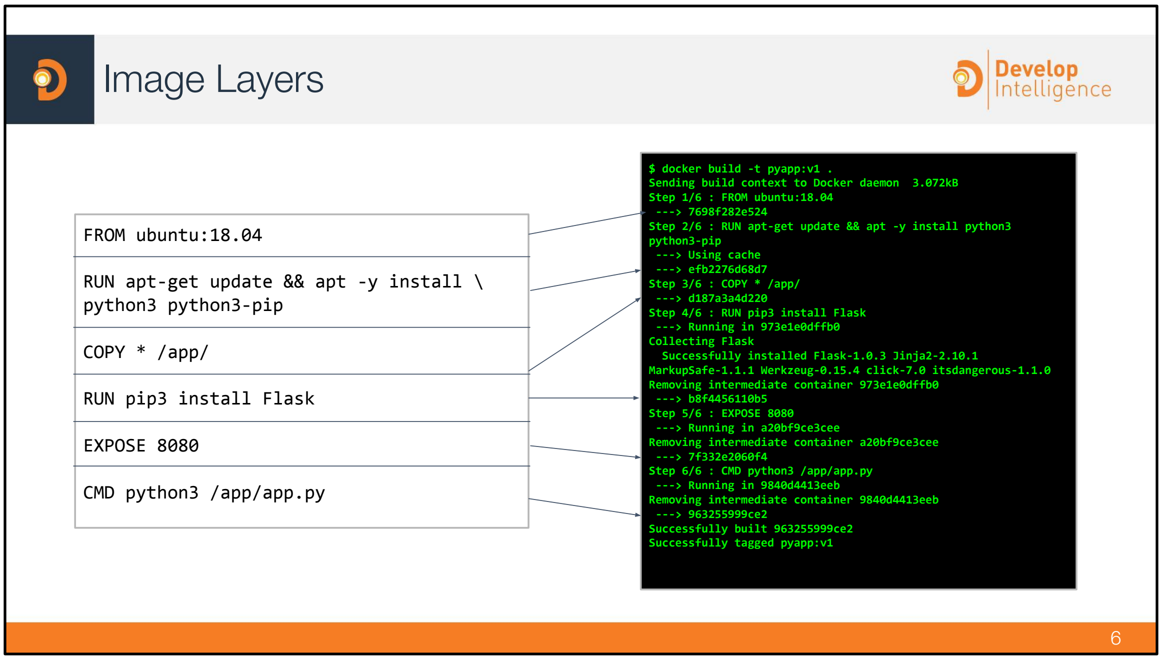


Image Layers

```
FROM ubuntu:18.04
RUN apt-get update && apt -y install \
python3 python3-pip
COPY * /app/
RUN pip3 install Flask
EXPOSE 8080
CMD python3 /app/app.py
```

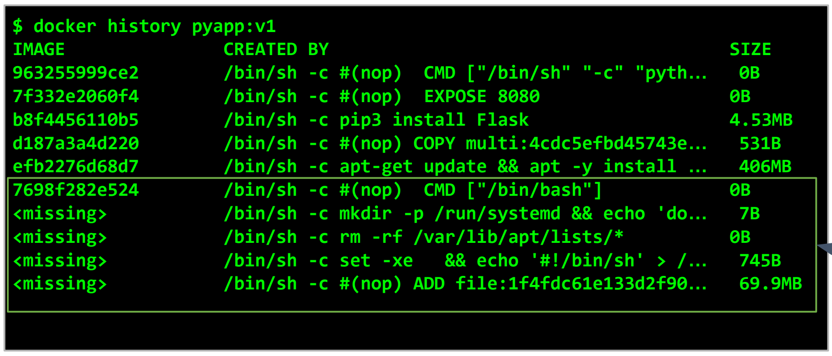
```
$ docker build -t pyapp:v1 .
Sending build context to Docker daemon  3.072kB
Step 1/6 : FROM ubuntu:18.04
 ---> 7698f282e524
Step 2/6 : RUN apt-get update && apt -y install python3
python3-pip
 ---> Using cache
 ---> efb2276d68d7
Step 3/6 : COPY * /app/
 ---> d187a3a4d220
Step 4/6 : RUN pip3 install Flask
 ---> Running in 973e1e0dffb0
Collecting Flask
  Successfully installed Flask-1.0.3 Jinja2-2.10.1
MarkupSafe-1.1.1 Werkzeug-0.15.4 click-7.0 itsdangerous-1.1.0
Removing intermediate container 973e1e0dffb0
 ---> b8f4456110b5
Step 5/6 : EXPOSE 8080
 ---> Running in a20bf9ce3cee
Removing intermediate container a20bf9ce3cee
 ---> 7f332e2060f4
Step 6/6 : CMD python3 /app/app.py
 ---> Running in 9840d4413eeb
Removing intermediate container 9840d4413eeb
 ---> 963255999ce2
Successfully built 963255999ce2
Successfully tagged pyapp:v1
```

Docker images consist of layers. When a container is run, each layer is joined together to produce the resulting container filesystem. Roughly every command in a docker file will produce a new layer. Each layer is a differential snapshot of the container filesystem at that point in time. Docker is intelligent about only downloading any layers it does not already have locally, it will also only store each layer exactly once on the host...even if multiple images have layers in common.

OverlayFS - https://docs.docker.com/storage/storagedriver/overlayfs-driver/

Image Layers

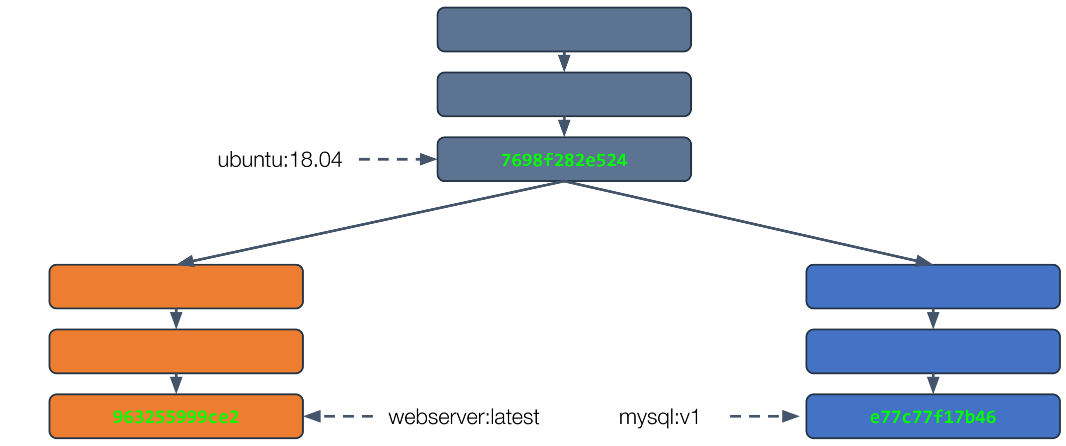
Develop Intelligence

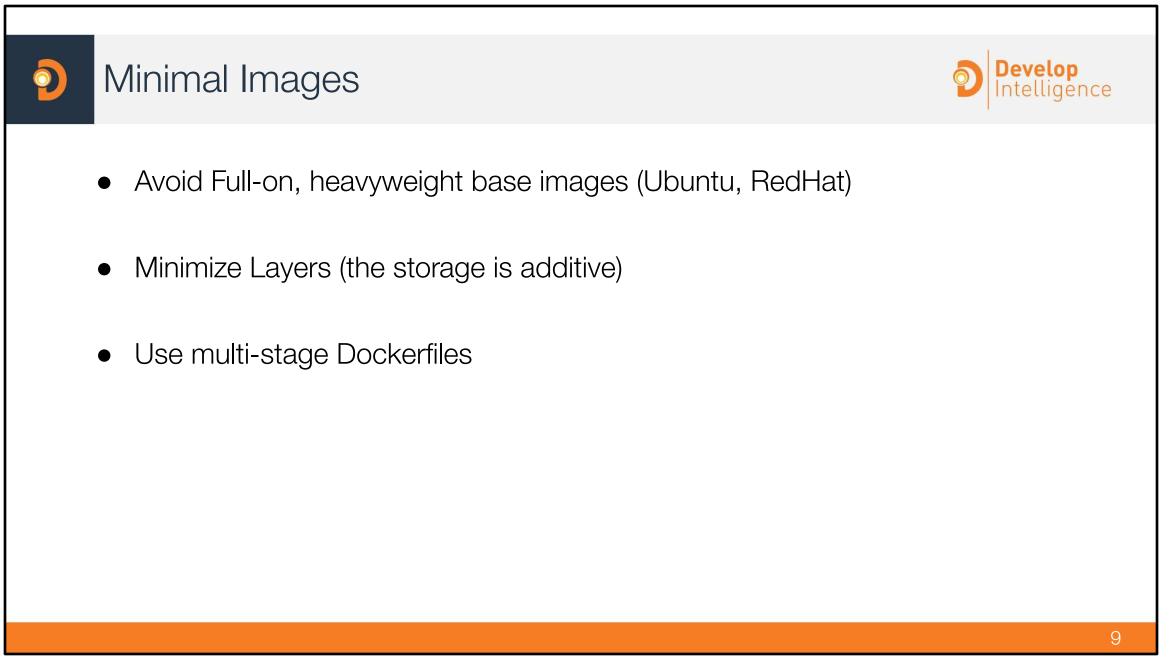


```
$ docker history pyapp:v1
IMAGE               CREATED BY                                      SIZE
963255999ce2        /bin/sh -c #(nop)  CMD ["/bin/sh" "-c" "pyth…   0B
7f332e2060f4        /bin/sh -c #(nop)  EXPOSE 8080                  0B
b8f4456110b5        /bin/sh -c pip3 install Flask                   4.53MB
d187a3a4d220        /bin/sh -c #(nop) COPY multi:4cdc5efbd45743e…   531B
efb2276d68d7        /bin/sh -c apt-get update && apt -y install …   406MB
7698f282e524        /bin/sh -c #(nop)  CMD ["/bin/bash"]            0B
<missing>           /bin/sh -c mkdir -p /run/systemd && echo 'do…   7B
<missing>           /bin/sh -c rm -rf /var/lib/apt/lists/*          0B
<missing>           /bin/sh -c set -xe   && echo '#!/bin/sh' > /…   745B
<missing>           /bin/sh -c #(nop) ADD file:1f4fdc61e133d2f90…   69.9MB
```

Base Image Layers

Image Layer Reuse

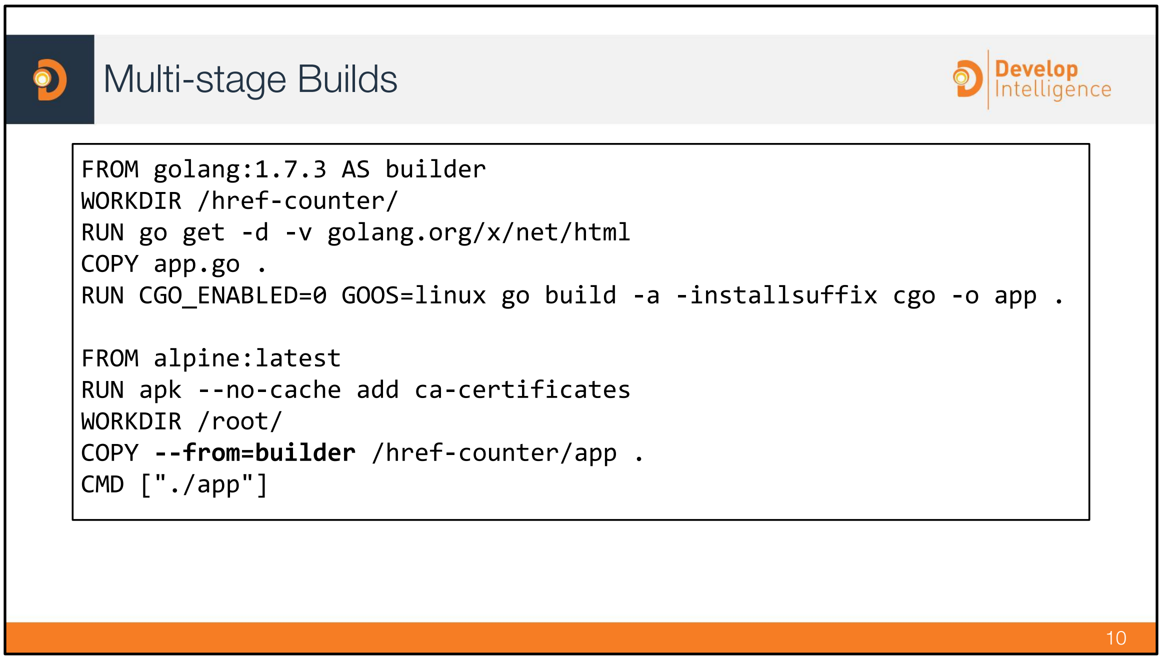



Minimal Images

- Avoid Full-on, heavyweight base images (Ubuntu, RedHat)
- Minimize Layers (the storage is additive)
- Use multi-stage Dockerfiles

9

https://docs.docker.com/develop/develop-images/multistage-build/



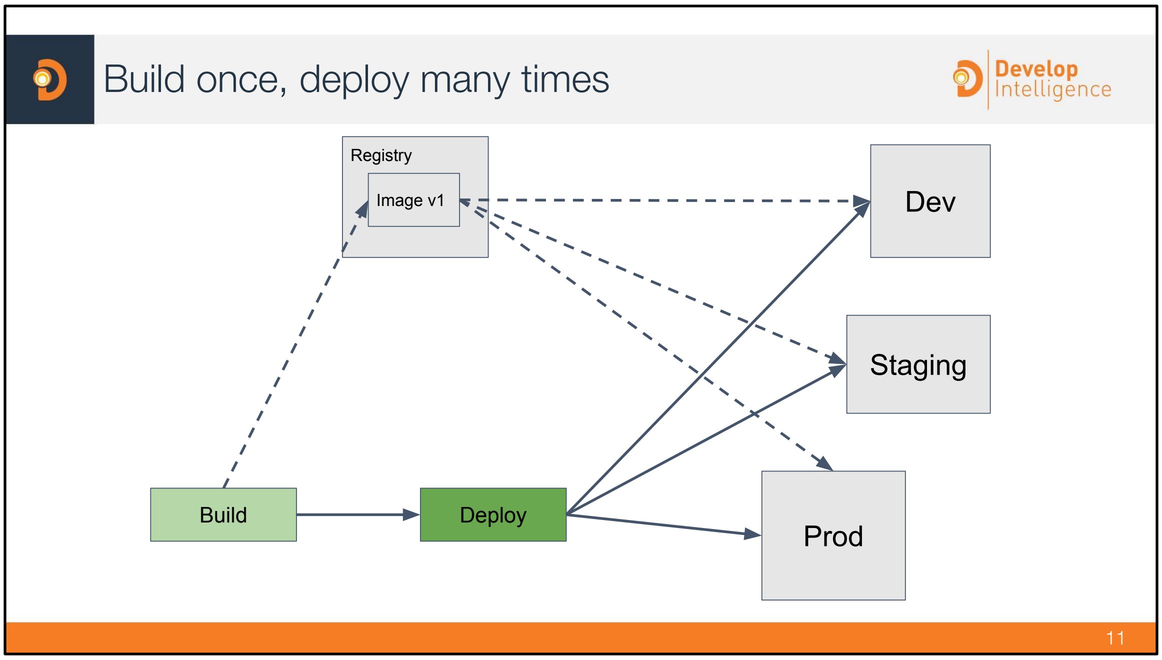
Multi-stage Builds

```
FROM golang:1.7.3 AS builder
WORKDIR /href-counter/
RUN go get -d -v golang.org/x/net/html
COPY app.go .
RUN CGO_ENABLED=0 GOOS=linux go build -a -installsuffix cgo -o app .

FROM alpine:latest
RUN apk --no-cache add ca-certificates
WORKDIR /root/
COPY --from=builder /href-counter/app .
CMD ["./app"]
```

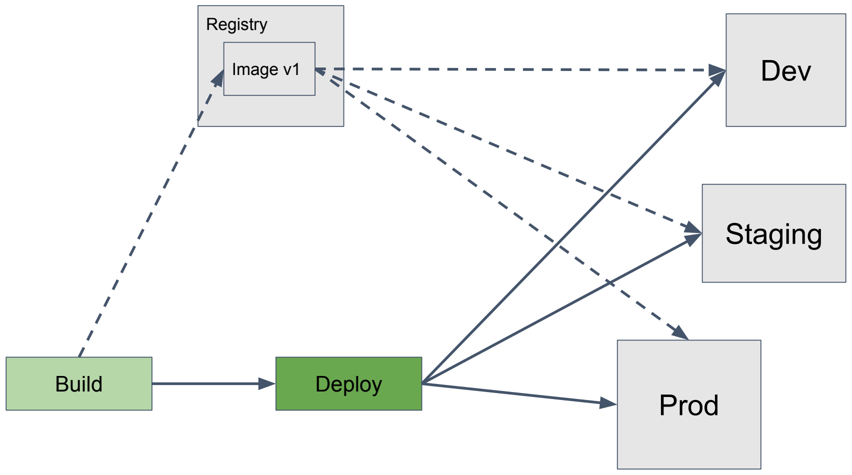
10

https://docs.docker.com/engine/reference/builder/
https://docs.docker.com/develop/develop-images/multistage-build/

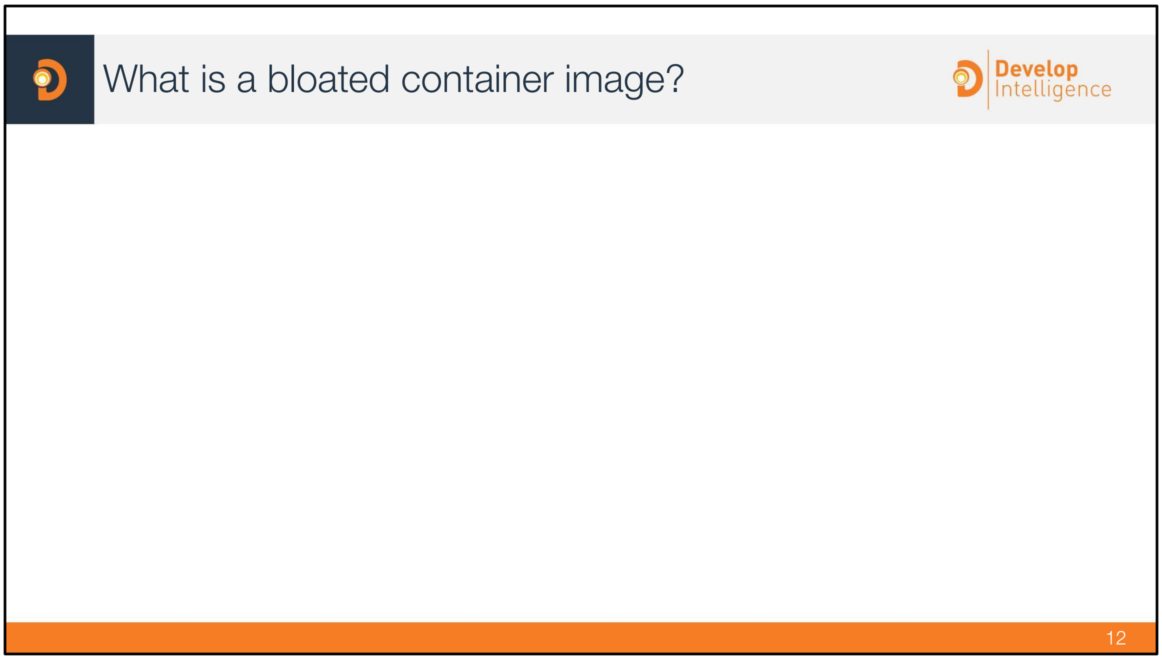


Build once, deploy many times

Develop Intelligence



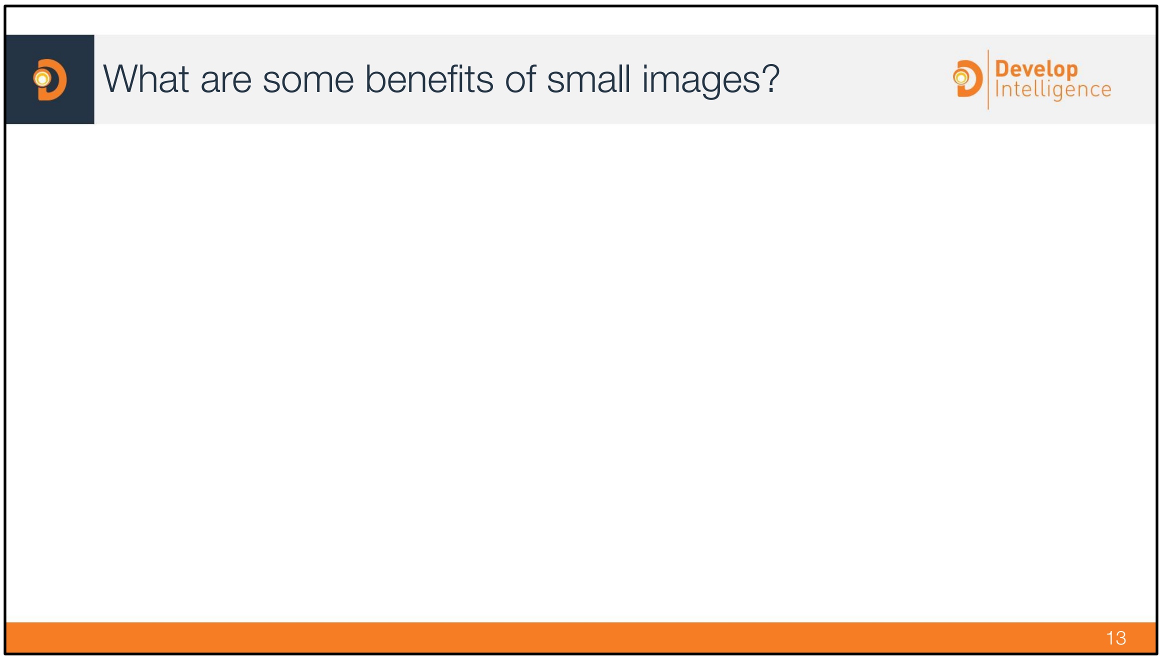
11



What is a bloated container image?

12

Contains unused dependencies
Advanced: Intermediate layers contain data not needed in final image



What are some benefits of small images?

Develop Intelligence