

Frequently Answered Queries from StackOverflow



Brandon Mitchell
Twitter: @sudo_bmitch
StackOverflow: bmitch

How Do We Learn?



How Do We Learn?

- RTFM
- Practice
- Drills
- Teaching

Typical StackOverflow User Background

- Mostly developers
- Often more comfortable with an IDE than a CLI
- DevOps is shifting those Devs into more Ops tasks
- Pro: devs no longer depend on ops to manage their app runtime environment

Typical StackOverflow User Background

- Mostly developers
- Often more comfortable with an IDE than a CLI
- DevOps is shifting those Devs into more Ops tasks
- Pro: devs no longer depend on ops to manage their app runtime environment
- Con: devs no longer depend on ops to manage their app runtime environment
- Devs are now learning OS/Linux/distributions, scripting, package managers, networking, and storage.

General Docker Questions

Q: How do containers differ from VM's

- Containers have a shared kernel, application isolation vs hardware isolation
- How do we change the mindset of people using containers as a lightweight VM?

Q: How do containers differ from VM's

- Containers have a shared kernel, application isolation vs hardware isolation
- How do we change the mindset of people using containers as a lightweight VM?
 - Who likes uptime?

Q: How do containers differ from VM's

- Containers have a shared kernel, application isolation vs hardware isolation
- How do we change the mindset of people using containers as a lightweight VM?
 - Who likes uptime?
 - Who wants to maintain a server that hasn't been rebooted for 3 years, and the original admin has left?

Q: How do containers differ from VM's

- Containers have a shared kernel, application isolation vs hardware isolation
- How do we change the mindset of people using containers as a lightweight VM?
 - Who likes uptime?
 - Who wants to maintain a server that hasn't been rebooted for 3 years, and the original admin has left?
 - Uptime quickly becomes a ticking time bomb.

Q: How do containers differ from VM's

- Containers have a shared kernel, application isolation vs hardware isolation
- How do we change the mindset of people using containers as a lightweight VM?
 - Who likes uptime?
 - Who wants to maintain a server that hasn't been rebooted for 3 years, and the original admin has left?
 - Uptime quickly becomes a ticking time bomb.
- What we want is availability, not uptime. We want a LB pointing to replicas spread across multiple AZ's so we can have **low uptime** and **high availability**.

Q: How do containers differ from VM's

Practical differences:

- Don't ssh into containers (exec, and only in dev)
- Don't upgrade containers in place (replace them)
- Don't install multiple apps inside a single container (compose files)
- Don't give containers static IP's (LB/reverse proxies)
- Don't backup containers (backup volumes)
- Don't export containers to make new images (use a Dockerfile)

Dockerfile

Q: What's the difference between the string and json syntax?

- RUN, CMD, and ENTRYPOINT can each use either syntax
- The string syntax includes a shell, `/bin/sh -c "${cmd}"` by default.
- The json syntax executes the command directly, without a shell.

Q: What's the difference between the string and json syntax?

- RUN, CMD, and ENTRYPOINT can each use either syntax
- The string syntax includes a shell, `/bin/sh -c "${cmd}"` by default.
- The json syntax executes the command directly, without a shell.

Shell Pros:

- Expands variables
- Command chaining (`&&`)
- I/O redirection

Q: What's the difference between the string and json syntax?

- RUN, CMD, and ENTRYPOINT can each use either syntax
- The string syntax includes a shell, `/bin/sh -c "${cmd}"` by default.
- The json syntax executes the command directly, without a shell.

Shell Pros:

- Expands variables
- Command chaining (`&&`)
- I/O redirection

Shell Cons:

- Intercepts signals
- Extra processing to merge entrypoint with cmd

Q: What's the difference between the string and json syntax?

String/Shell Syntax:

```
RUN echo hello world  
ENTRYPOINT /entrypoint.sh  
CMD run a b c
```

Q: What's the difference between the string and json syntax?

String/Shell Syntax:

```
RUN echo hello world  
ENTRYPOINT /entrypoint.sh  
CMD run a b c
```

Json/Exec Syntax:

```
RUN ["echo", "hello", "world"]  
ENTRYPOINT ["/entrypoint.sh"]  
CMD ["run", "a", "b", "c"]
```

Q: What's the difference between the string and json syntax?

What if cmd is a string and you have an entrypoint?

```
/entrypoint.sh /bin/sh -c "args to entrypoint"
```

Q: What's the difference between the string and json syntax?

What if cmd is a string and you have an entrypoint?

```
/entrypoint.sh /bin/sh -c "args to entrypoint"
```

To fix this in the entrypoint:

```
#!/bin/sh
if [ $# -gt 1 -a "$1" = "/bin/sh" -a "$2" = "-c" ]; then
    shift 2
    eval "set -- $1"
fi
exec "$@"
```

Q: Why doesn't build use the cache?

Cache requires:

- Same command to be run
- Same checksum on all files
- Same previous layer
- Image was built locally

Q: Why doesn't build use the cache?

Cache requires:

- Same command to be run
- Same checksum on all files
- Same previous layer
- Image was built locally

Cache can be broken by:

- Changing a build ARG value
- Changing a timestamp
- The previous layer being rebuilt

Q: Why doesn't build use the cache?

Cache requires:

- Same command to be run
- Same checksum on all files
- Same previous layer
- Image was built locally

Cache can be broken by:

- Changing a build ARG value
- Changing a timestamp
- The previous layer being rebuilt

To trust images pulled from a registry, use:

```
docker build --cache-from my_image ...
```

Q: Why is my image so large?

How big are the layers resulting from this Dockerfile:

```
FROM busybox

RUN mkdir /data
RUN dd if=/dev/zero bs=1024 count=1024 of=/data/one
RUN chmod -R 0777 /data
RUN dd if=/dev/zero bs=1024 count=1024 of=/data/two
RUN chmod -R 0777 /data
RUN rm /data/one

CMD ls -alh /data
```


Q: Why is my image so large?

- Running the image you see the 1MB file:

```
-rwxrwxrwx    1 root    root      1.0M May 12 00:14 two
```

- Each `dd` command adds a 1MB layer.

Q: Why is my image so large?

- Running the image you see the 1MB file:

```
-rwxrwxrwx    1 root    root      1.0M May 12 00:14 two
```

- Each `dd` command adds a 1MB layer.
- Each `chmod` command will change permissions and copy the entire 1MB file to the next layer.

Q: Why is my image so large?

- Running the image you see the 1MB file:

```
-rwxrwxrwx    1 root    root      1.0M May 12 00:14 two
```

- Each `dd` command adds a 1MB layer.
- Each `chmod` command will change permissions and copy the entire 1MB file to the next layer.
- What does the `rm` command do to the image size?

Q: Why is my image so large?

The `rm` command only changes directory metadata in the next layer:

```
Step 6/7 : RUN chmod -R 0777 /data
---> Running in 038bd2bc5aea
---> 77793bf30d5f
Step 7/8 : RUN rm /data/one
---> Running in 504c6e9b6637
---> 9fe0e2f18893
...

$ docker image ls -a | grep 77793bf30d5f
REPOSITORY      TAG          IMAGE ID      CREATED        SIZE
<none>         <none>       77793bf30d5f  10 minutes ago 6.37MB
$ docker image ls -a | grep 9fe0e2f18893
REPOSITORY      TAG          IMAGE ID      CREATED        SIZE
<none>         <none>       9fe0e2f18893  10 minutes ago 6.37MB
```

Q: Why is my image so large?

- Resulting 1MB file has become 4MB on disk and over the network
- Compare the two resulting images to see the added disk space:

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
busybox	latest	54511612f1c4	8 months ago	1.13MB
test-layers	latest	757ce49dd12f	10 minutes ago	6.37MB

- Subtracting the two you get the expected ~5MB

Q: Why is my image so large?

- 5MB? Not 4MB? Where did the extra 1MB come from?

```
FROM busybox
RUN mkdir /data
RUN dd if=/dev/zero bs=1024 count=1024 of=/data/one
RUN chmod -R 0777 /data
RUN dd if=/dev/zero bs=1024 count=1024 of=/data/two
RUN chmod -R 0777 /data
RUN rm /data/one
CMD ls -alh /data
```

Q: Why is my image so large?

- 5MB? Not 4MB? Where did the extra 1MB come from?

```
FROM busybox
RUN mkdir /data
RUN dd if=/dev/zero bs=1024 count=1024 of=/data/one
RUN chmod -R 0777 /data
RUN dd if=/dev/zero bs=1024 count=1024 of=/data/two
RUN chmod -R 0777 /data
RUN rm /data/one
CMD ls -alh /data
```

- A `chmod` or `chown` changes a timestamp on the file *even when there is no permission or ownership change made.*

Q: Why is my image so large?

How can we examine layers? Build with `docker build --rm=false .`

```
...  
Step 2/7 : RUN mkdir /data  
---> Running in 04c5fa1360b0  
---> 9b4368667b8c  
Step 3/7 : RUN dd if=/dev/zero bs=1024 count=1024 of=/data/one  
---> Running in f1b72db3bfaa  
1024+0 records in  
1024+0 records out  
1048576 bytes (1.0MB) copied, 0.006002 seconds, 166.6MB/s  
---> ea2506fc6e11
```


Q: Why is my image so large?

Check each temp image with `docker diff ${cid}`

```
$ docker diff 04c5fa1360b0 # mkdir /data
A /data
$ docker diff f1b72db3bfaa # dd if=/dev/zero bs=1024 count=1024 of=/data/one
C /data
A /data/one
$ docker diff 81c607555a7d # chmod -R 0777 /data
C /data
C /data/one
$ docker diff 1bd249e1a47b # dd if=/dev/zero bs=1024 count=1024 of=/data/two
C /data
A /data/two
$ docker diff 038bd2bc5aea # chmod -R 0777 /data
C /data/one
C /data/two
$ docker diff 504c6e9b6637 # rm /data/one
C /data
D /data/one
```

Q: Why is my image so large?

Reduce the image size by merging RUN lines:

```
FROM busybox

RUN mkdir /data \
  && dd if=/dev/zero bs=1024 count=1024 of=/data/one \
  && chmod -R 0777 /data \
  && dd if=/dev/zero bs=1024 count=1024 of=/data/two \
  && chmod -R 0777 /data \
  && rm /data/one

CMD ls -alh /data
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
busybox	latest	54511612f1c4	8 months ago	1.13MB
test-layers2	latest	951252cf34ed	25 seconds ago	2.18MB

Run

Q: What does "invalid reference format" mean?

- A reference is a pointer to an image.
- The docker command line is order dependent:

```
docker ${docker_args} run ${run_args} image ${cmd}
```

- Frequently happens when an invalid arg gets parsed as the image name or invalid characters from copy/pasting from a source that changes dashes and quotes.
- What does docker interpret as the image name here:

```
my project$ docker run -it -rm -v $(pwd):/data "my_image"
```

Q: Why do I get "executable not found"?

- Did you run the intended command?

```
docker run --rm my_image -it echo hello world
```

- Is docker trying to run a json string?
- Does the file exist... in the path and inside the container?
- If it is a shell script, check the first line

```
#!/bin/bash
```

- If it is a binary, there is likely a missing library

Q: What is this TTY error?

```
the input device is not a TTY
```

- The TTY is a terminal in linux
- `docker run -it`: Interactive terminal
- `docker run -i`: Input but no terminal, piping in a file
- `docker run -t`: Setup terminal but no input, color output in logs
- `docker run`: No input, no terminal, typically used for scripts/cron/ci-cd

Networking

Q: EXPOSE vs Publishing a port?

- EXPOSE
 - Documentation from the image creator to the person running the image
 - Not needed to publish
 - Not needed for container-to-container communication
- Publish
 - Maps a port on the host to connect to a port in the container.
 - One-way, from host to container, it does not allow containers to access applications running on the host.

Q: Networking issues between containers?

- Make sure app is listening on 0.0.0.0, not 127.0.0.1
- Use a user generated network
- Use DNS: container id, container name, service name, or network alias
- Connect to the container port, not the host published port
- With overlay networking, open 7946/both, 4789/tcp, and protocol 50.

Q: Networking issues between containers?

Follow-up Q: Do I need to expose the port?

- Nope, expose is documentation.

Follow-up Q: Do I need to publish the port?

- Nope, that only makes the container accessible from outside of docker.

Follow-up Q: Do I need links?

- Nope, links are deprecated, use user created networks.

Q: Networking issues between containers?

Follow-up Q: Do I need to expose the port?

- Nope, expose is documentation.

Follow-up Q: Do I need to publish the port?

- Nope, that only makes the container accessible from outside of docker.

Follow-up Q: Do I need links?

- Nope, links are deprecated, use user created networks.

Follow-up Q: What's a network alias?

- You can give containers or services additional names on any network.

Q: Networking issues accessing published port?

- Make sure app is listening on 0.0.0.0, not 127.0.0.1.
- Connect to an IP on the host, not the 0.0.0.0 listener wildcard.
- Verify the publish command. `-p 8080:80` maps host port 8080 to container port 80.
- With overlay networking, open 7946/both, 4789/tcp, and protocol 50.
- Verify the docker host you are using with `echo $DOCKER_HOST`. If this is set, connect to that IP instead.

Networking Tips

Follow-up Q: How to check if the app is listening on 0.0.0.0?

```
docker run -it --rm --net container:${cid}  
nicolaka/netshoot netstat -lnt
```

Networking Tips

Follow-up Q: How to check if the app is listening on 0.0.0.0?

```
docker run -it --rm --net container:${cid}  
nicolaka/netshoot netstat -lnt
```

Follow-up Q: Why doesn't localhost work?

- IPv6 is the likely cause, use 127.0.0.1 instead.

Volumes

Q: Build isn't updating a directory?

Typically caused by a volume:

- A volume attached to the container persists the old state
- A volume defined in the image prevents changes to that directory

PSA: Remove VOLUME in Dockerfiles

- Users cannot extend the image
- Anonymous volumes clutter the filesystem
- Not required for creating volumes at runtime

PSA: Remove VOLUME in Dockerfiles

- Users cannot extend the image
- Anonymous volumes clutter the filesystem
- Not required for creating volumes at runtime
- Solution: define volumes in a compose file

Q: How do I handle UID/GID and permission issues with host volumes?

- Option 1: `chmod 777`
- Option 2: Update image user to match host uid/gid
- Option 3: Use named volumes and manage data with containers
- Option 4: Correct permissions with entrypoint

Q: How do I handle UID/GID and permission issues?

Option 2: Update image to match host uid/gid:

```
FROM debian:latest
ARG UID=1000
ARG GID=1000
RUN groupadd -g $GID cuser \
  && useradd -m -u $UID -g $GID -s /bin/bash cuser
USER cuser
```

```
$ docker build \
  --build-arg UID=$(id -u) --build-arg GID=$(id -g) .
```

Q: How do I handle UID/GID and permission issues?

Option 4: Entrypoint to correct uid/gid:

```
FROM jenkins/jenkins:lts
USER root
RUN apt-get update \
    && wget -O /usr/local/bin/gosu "https://github.com/..." \
    && chmod +x /usr/local/bin/gosu \
    && curl -sSL https://get.docker.com/ | sh \
    && usermod -aG docker jenkins
COPY entrypoint.sh /entrypoint.sh
ENTRYPOINT ["/entrypoint.sh"]
```

Q: How do I handle UID/GID and permission issues?

Option 4: Entrypoint to correct uid/gid:

```
#!/bin/sh

# if image and volume gid do not match, modify container user
SOCK_DOCKER_GID=$(ls -ng /var/run/docker.sock | cut -f3 -d' ')
CUR_DOCKER_GID=$(getent group docker | cut -f3 -d: || true)
if [ "$SOCK_DOCKER_GID" != "$CUR_DOCKER_GID" ]; then
    groupmod -g ${SOCK_DOCKER_GID} docker
fi

# drop access to jenkins user and run jenkins entrypoint
exec gosu jenkins /bin/tini -- /usr/local/bin/jenkins.sh "$@"
```

Q: How to initialize a host volume from image?

- Option 1: Don't. Initialize outside of docker, before starting the container
- Option 2: Copy with an entrypoint from a saved location in the image.

Q: How to initialize a host volume from image?

- Option 1: Don't. Initialize outside of docker, before starting the container
- Option 2: Copy with an entrypoint from a saved location in the image.
- Option 3: Define a named volume that's a bind mount.

```
$ docker volume create --driver local \  
  --opt type=None \  
  --opt device=/home/user/test \  
  --opt o=bind \  
  test_vol
```


Q: How to initialize a host volume from image?

Walk-through of example 3 - Dockerfile:

```
FROM busybox:latest
RUN adduser --home /home/user --uid 5001 \
    --disabled-password user
USER user
COPY --chown=user sample-data/ /home/user/data/
```

Q: How to initialize a host volume from image?

Walk-through of example 3 - Sample data:

```
$ ls -al sample-data/  
total 24  
drwxr-xr-x  3 bmitch bmitch 4096 Jan 22  2017 .  
drwxr-xr-x 30 bmitch bmitch 4096 May 14 09:41 ..  
drwxr-xr-x  2 bmitch bmitch 4096 Jan 22  2017 dir  
-rw-r--r--  1 bmitch bmitch  14 Jan 22  2017 file2.txt  
-rw-r--r--  1 bmitch bmitch  12 Jan 22  2017 file.txt  
-rw-r--r--  1 bmitch bmitch 214 Jan 22  2017 tar-file.tgz
```

Q: How to initialize a host volume from image?

Walk-through of example 3 - create volume:

```
$ mkdir test-vol  
  
$ ls -al test-vol  
total 8  
drwxr-sr-x  2 bmitch bmitch 4096 May 14 09:40 .  
drwxr-xr-x 30 bmitch bmitch 4096 May 14 09:33 ..  
  
$ docker volume create --driver local --opt type=none \  
  --opt device=$(pwd)/test-vol --opt o=bind test-vol  
test-vol
```

Q: How to initialize a host volume from image?

Walk-through of example 3 - Run the container:

```
$ docker run -it --rm -v test-vol:/home/user/data test-vol \
/bin/sh -c "\
    echo hello world >/home/user/data/inside-container.txt \
    && ls -l /home/user/data"
total 20
drwxr-xr-x  2 user  user  4096 May 14 13:43 dir
-rw-r--r--  1 user  user   12 Jan 23  2017 file.txt
-rw-r--r--  1 user  user   14 Jan 23  2017 file2.txt
-rw-r--r--  1 user  user   12 May 14 13:43 inside-container.txt
-rw-r--r--  1 user  user  214 Jan 23  2017 tar-file.tgz
```

Q: How to initialize a host volume from image?

Walk-through of example 3 - Show the local directory from the host:

```
$ ls -al test-vol/
total 28
drwxr-sr-x  3   5001   5001 4096 May 14 09:43 .
drwxr-xr-x 30 bmitch bmitch 4096 May 14 09:41 ..
drwxr-xr-x  2   5001   5001 4096 May 14 09:43 dir
-rw-r--r--  1   5001   5001  14 Jan 22  2017 file2.txt
-rw-r--r--  1   5001   5001  12 Jan 22  2017 file.txt
-rw-r--r--  1   5001   5001  12 May 14 09:43 inside-container.txt
-rw-r--r--  1   5001   5001 214 Jan 22  2017 tar-file.tgz
```

Thank You

Slides: <https://github.com/sudo-bmitch/dc2018>



Brandon Mitchell
Twitter: @sudo_bmitch
StackOverflow: bmitch

