Working with Volumes and Persistent Data

The Big Picture

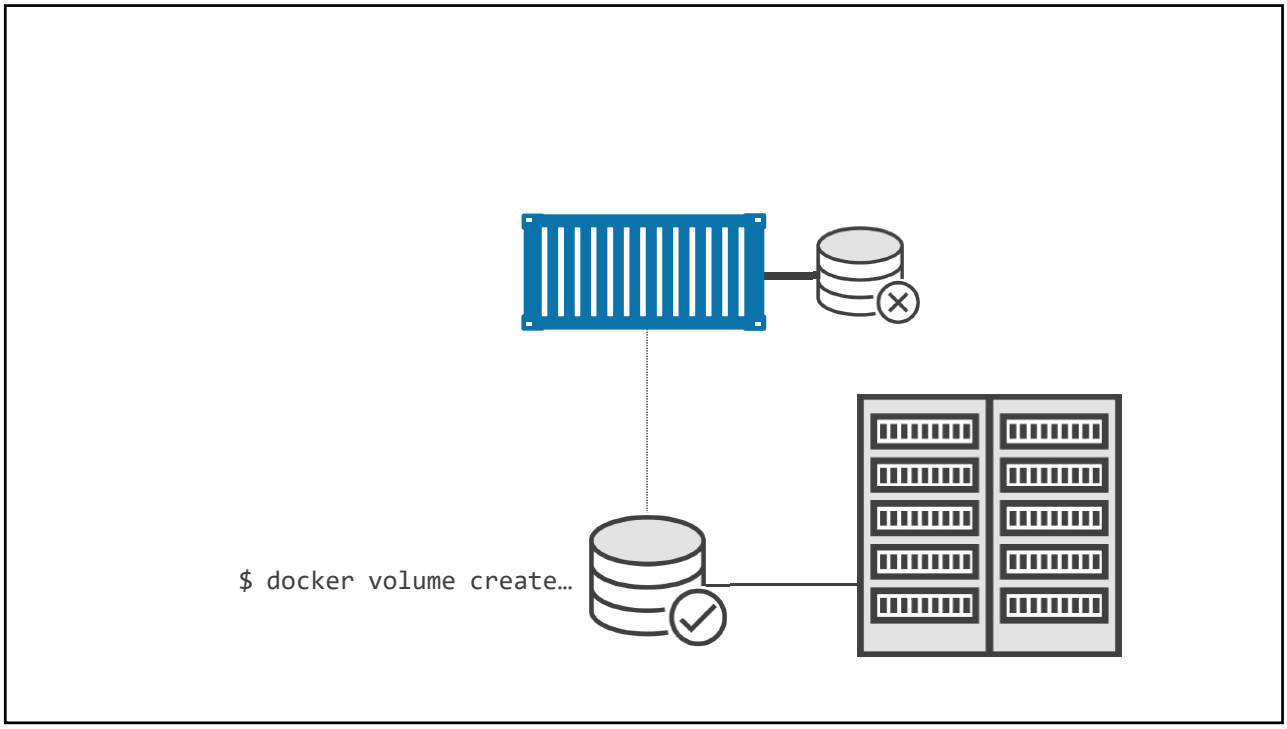
Laying the foundation

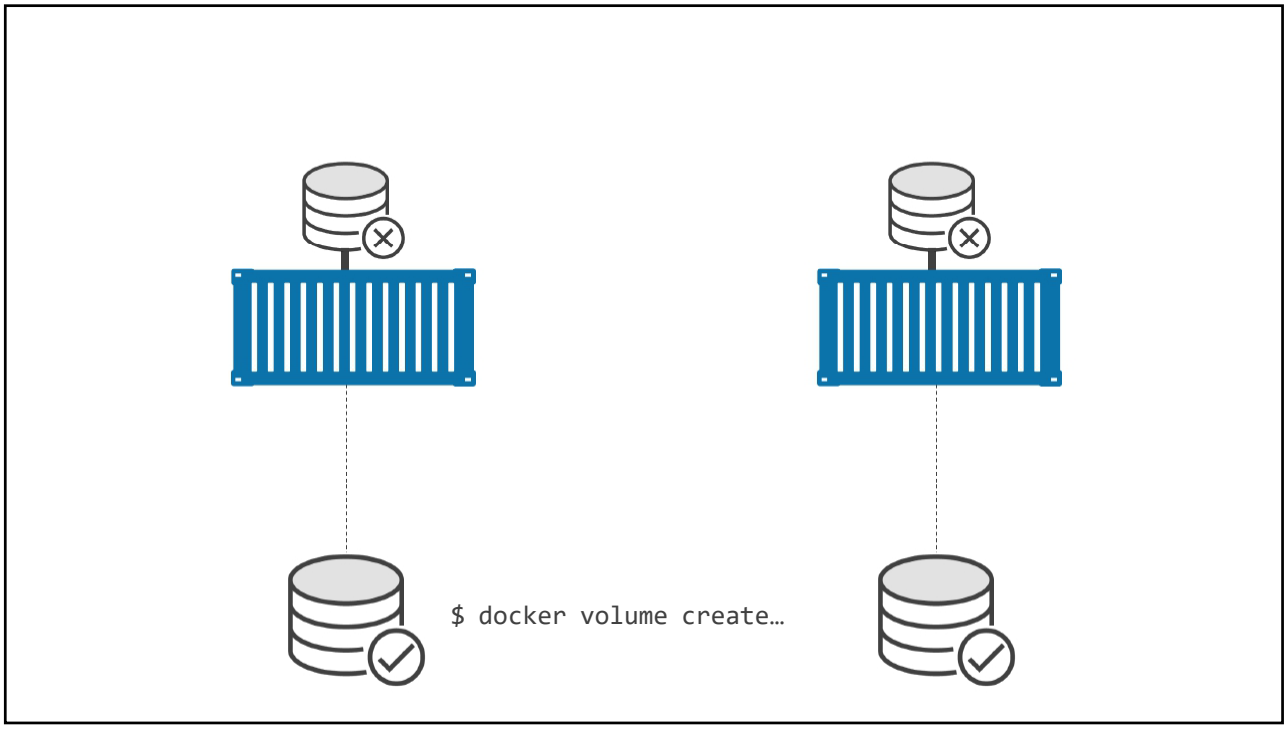
Persistent

Immutable

Container Lifetime & Persistent Data

- Containers are **usually** immutable
- "immutable infrastructure": only re-deploy containers, never change
- This is the ideal scenario, but what about databases, or unique data?
- Docker gives us features to ensure these "separation of concerns"
- This is known as "persistent data"
- Two ways: Volumes and Bind Mounts
- Volumes: make special location outside of container
- Bind Mounts: link container path to host path


$ docker volume create…


$ docker volume create…

Persistent Data: Volumes

- VOLUME command in Dockerfile
- Also override with docker run -v /path/in/container
- Bypasses Union File System and stores in alt location on host
- Includes it's own management commands under docker volume
- By default they only have a unique ID, but you can assign name
- Then it's a "named volume"

Persistent Data: Bind Mounting

- Maps a host file or directory to a container file or directory
- Can't use in Dockerfile, must be at container run
- ... run -v /Users/bret/stuff:/path/container (mac/linux)
- ... run -v //c/Users/bret/stuff:/path/container (windows)