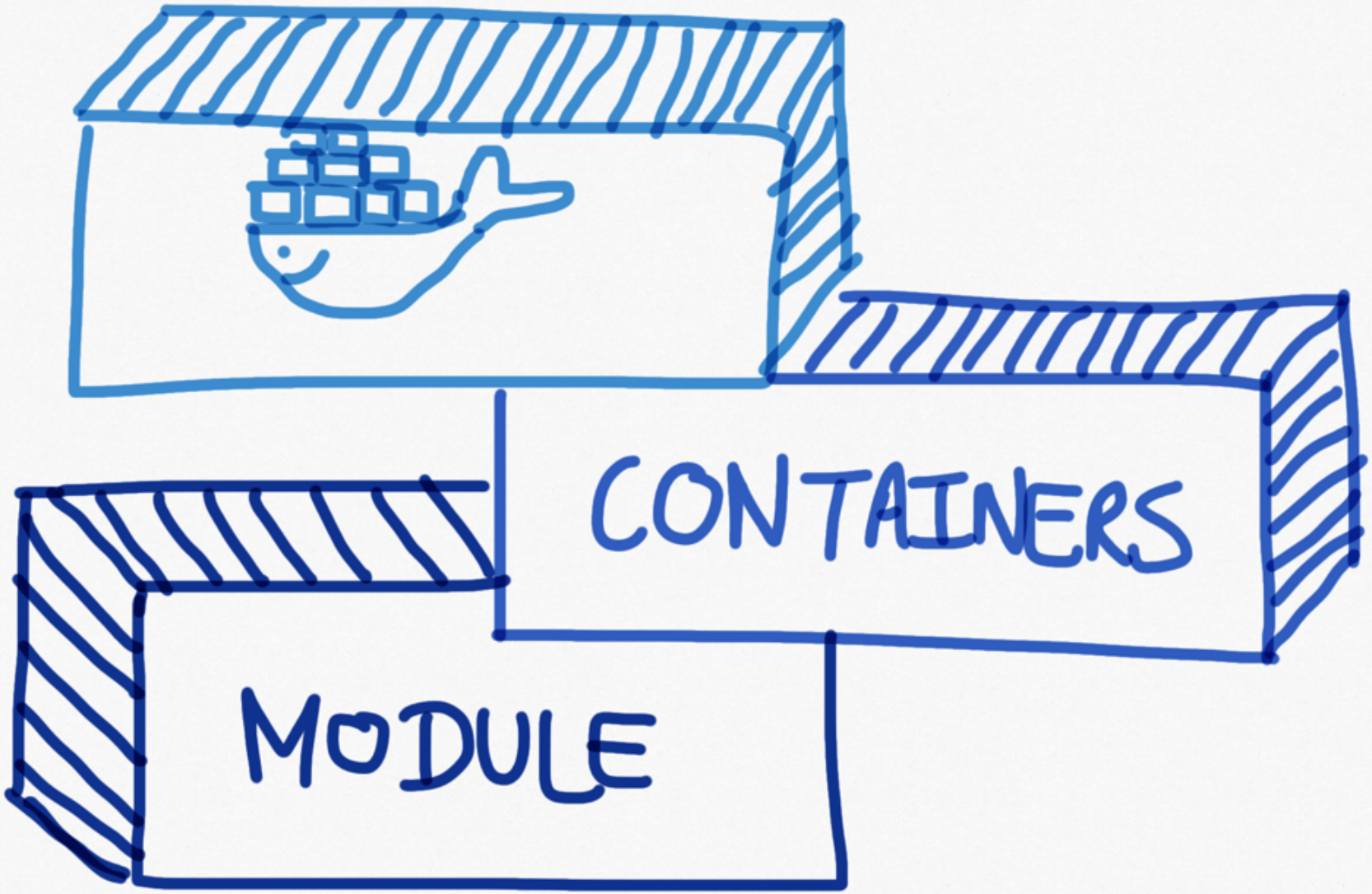

CONTAINERS
MODULE

# What are we going to learn

1. Container basics
2. Exposing ports
3. Networking
4. Storage
5. Logs
6. Environment variables

Time variant

# Lesson 1

- [ ] WHAT is a CONTAINER
- [ ] Get container to say CIAO!
  + something else → ooooh

- [ ] Delete everything

# Defining a container

- A container packages up all the code and dependencies needed for an app to run
- Containers isolate software from environment
- Solves the problem of getting software to run reliably in different environments
- Images become containers at runtime
- Containers use portions of the host OS' resources
- Stateless
  - There are ways to persist data in a container

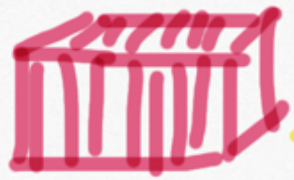
# Container pros

- Containers enable isolation
  - Decouple applications from OS
  - Secure workspace
- Containers enable portability
  - Can run on any machine, can deploy on any machine
  - What you build and run locally is same in production
- Containers enable scaling
  - Lightweight and can be modified with ease
- Their use
  - Most commonly at REA we use containers to run our app images and any dependencies

# Extra reading

- What is a container
- Why containers?
- Container vs VM

# Create Our Own... Container



...But How ??

↳ Docker Docs

# Best resource

- https://docs.docker.com/engine/reference/commandline
- Look at the side tab, it consists of all docker commands you could ever need.

# Exercise Time

# What Do We Have Running

- [ ] List currently running containers
- [ ] List ALL containers
- [ ] Echo containers to terminal by id

# Extra Spice

- [ ] Run a container in daemon mode

does the file persist?

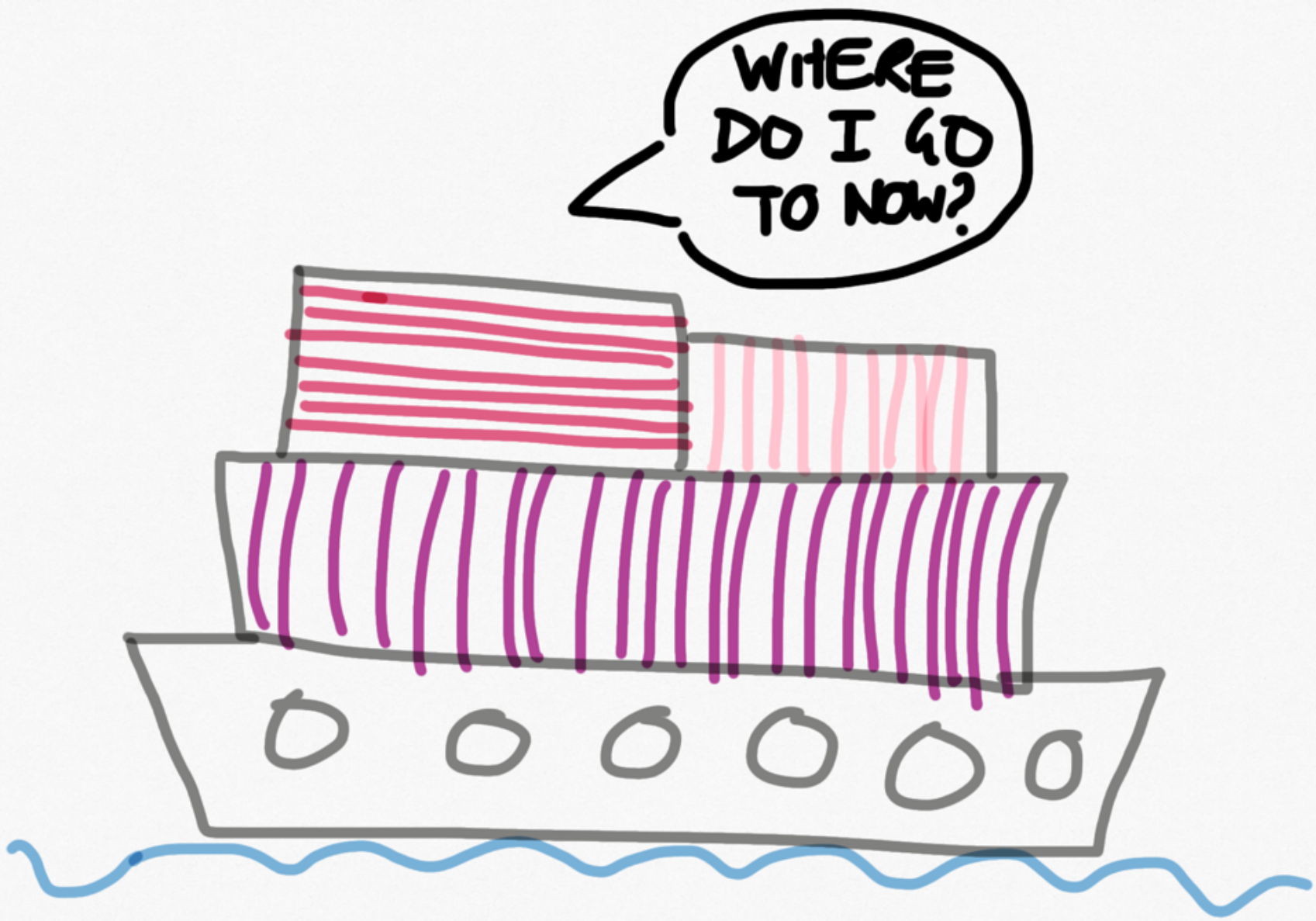
- [ ] Run bash shell on container
- [ ] Create a file your container
- [ ] Stop & re-start container

# Clean It All Up

- [ ] Stop all running containers
- [ ] Remove the containers
- [ ] Remove a container automatically after it has finished running
- [ ] Force a running container to be removed

# Lesson 2

- [ ] Understand how ports work
- [ ] Expose / publish port for a container


WHERE DO I GO TO NOW?

# Like a shipping port?

- A ship is assigned a specific port to deliver goods.
- In a network, a port is a defined as a place where things can communicate with each other.
  - E.g. port 80 for HTTP/TCP connections
- In Docker, a container needs a port to talk to other things too.
  - E.g. Talk to other Docker containers

# Exercise Time

# Resources

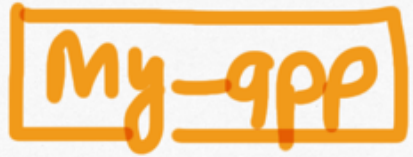
- Docker run
- Exposing port

# Expose a Port

- ☐ What ports are currently occupied
- ☐ Map host port 1234 to port 80 on your container
- ☐ Check the app is running on port 1234
- ☐ Expose a port without specifying a port, check app is running
  - ↳ what port is it on now?

# Port Mapping

CURRENT




docker ps

... ... ..... PORTS

80/tcp

- By default my-app listens on port 80

IDEAL

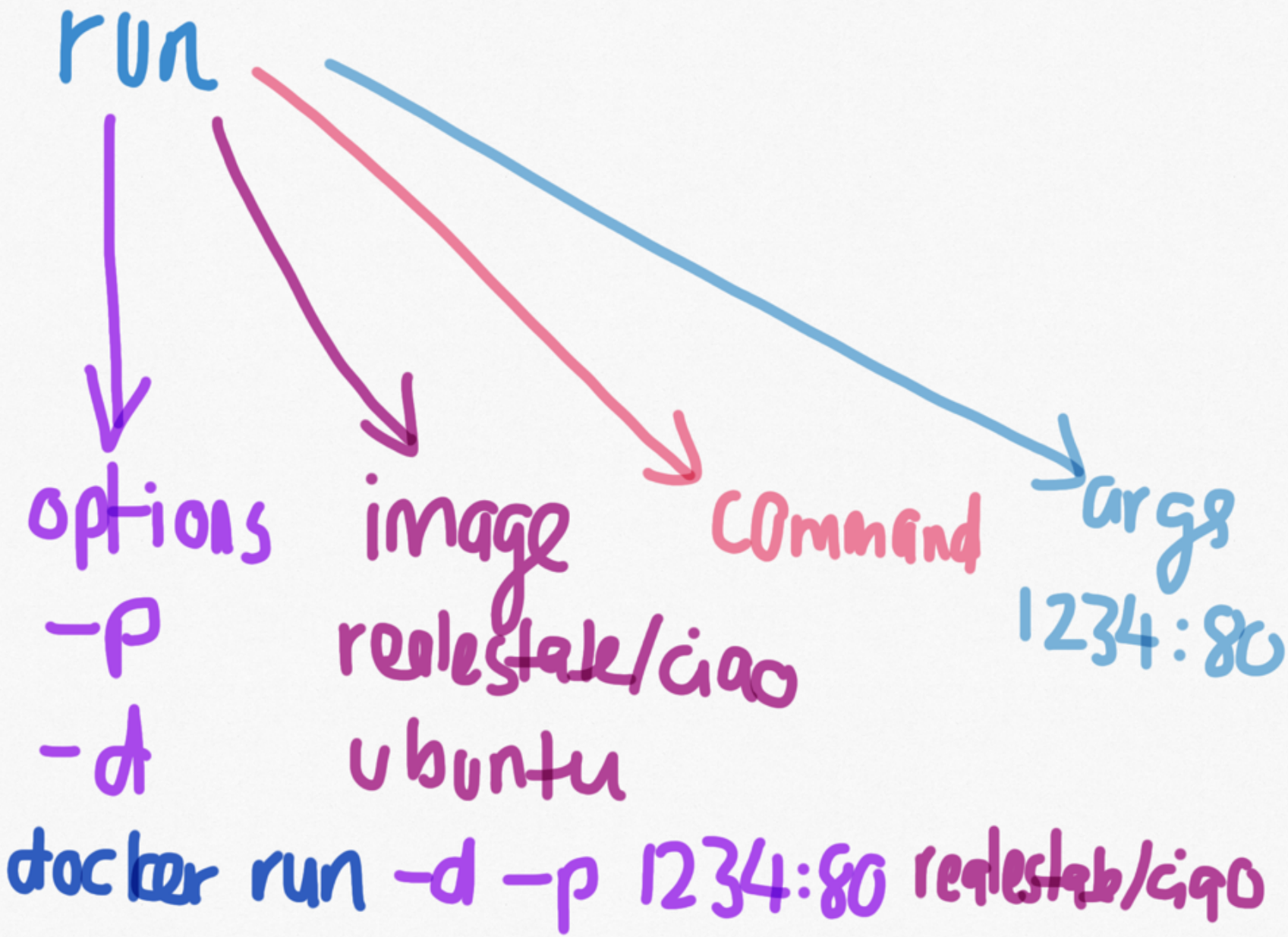

docker ps

... ... ... .. PORTS

1234 → 80/tcp

↳ localhost

- We want my-app to listen on port 1234


run
options
-p
-d
image
realestate/ciao
ubuntu
command
args
1234:80
docker run -d -p 1234:80 realestate/ciao

# Lesson 3

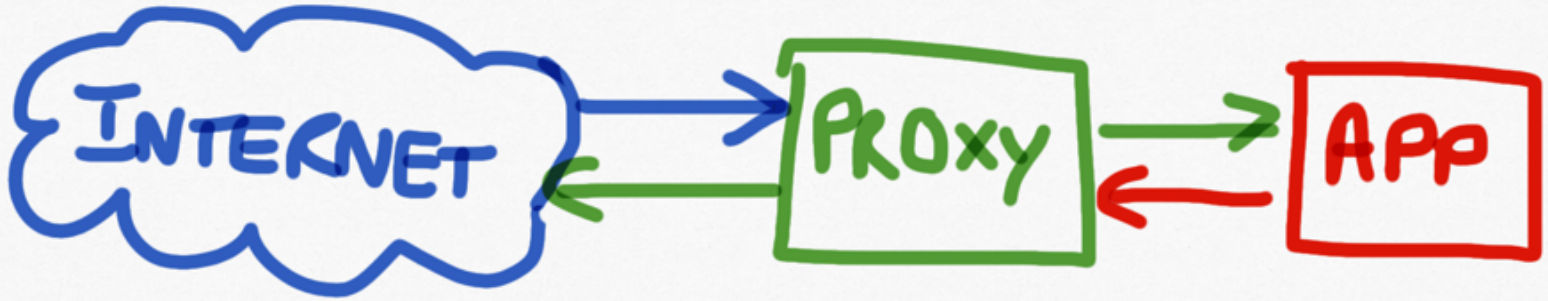
- [ ] Understand why networks are useful
- [ ] Create our own network with 2 different containers
- [ ] Inspect the network & delete it!

# Our problem

PROXY ??? APP

NOTE: this is a common REA setup

# We Want:



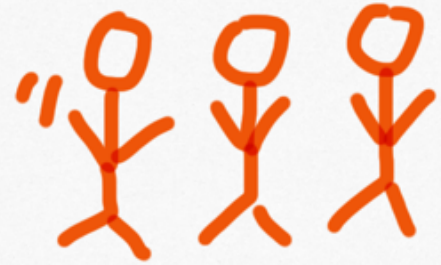

1. Proxy to be able to talk to the app
2. Proxy to be able to talk out to the internet
3. Be able to see where the proxy & app are sitting locally

# Exercise Time

# Resources

- Network
- Use realestate/ciao-proxy as image for the proxy
  - This is an NGINX server that is configured as a reverse proxy
  - It is currently listening on port 80
  - It is set up to forward requests specifically to ‘app’
  - NB: Make sure that you call your app container ‘app’

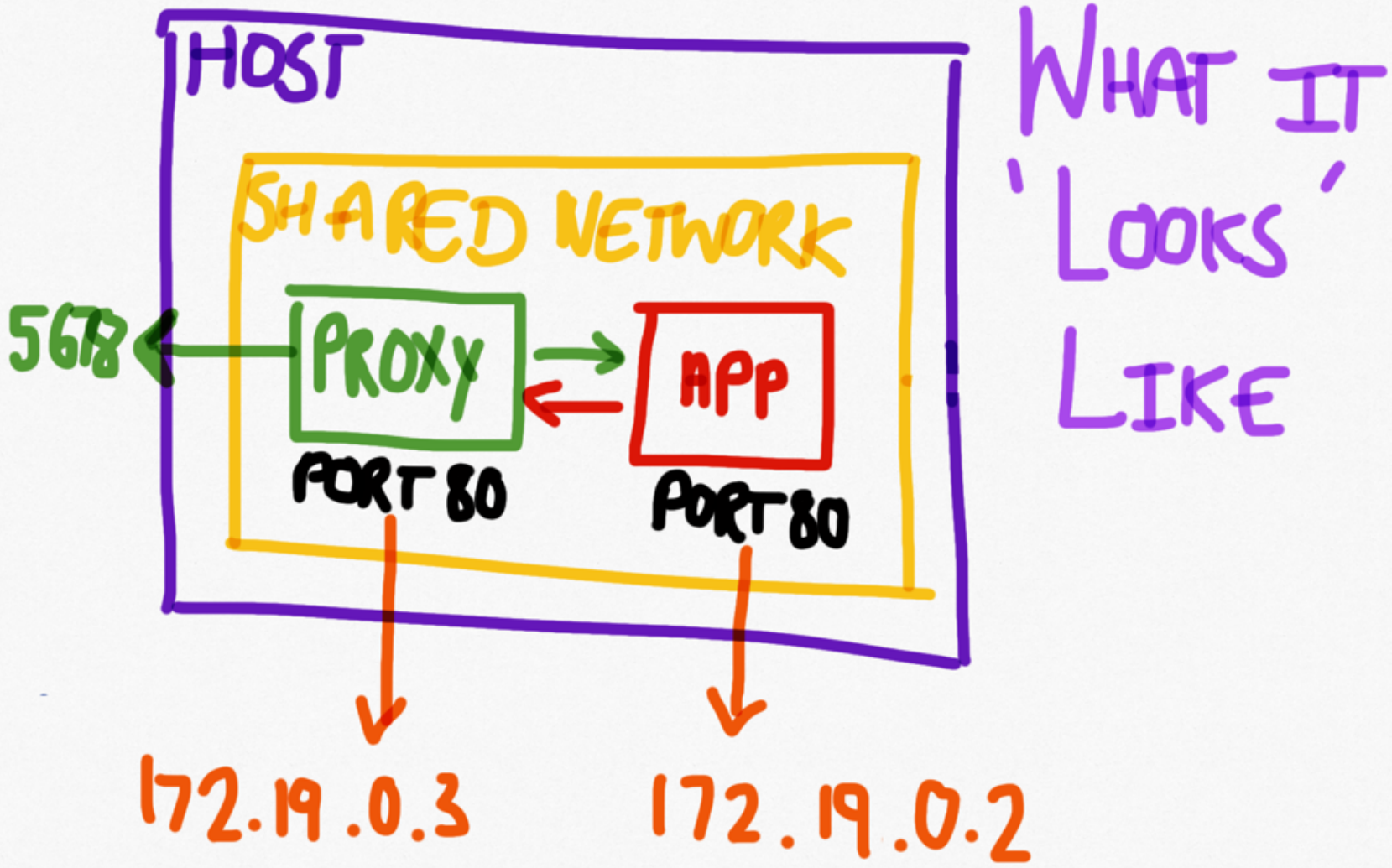
# Lets Get Networking



- [ ] List current networks
- [ ] Create a network
- [ ] Run an app container in the network
- [ ] Run a proxy container in the network
- [ ] Expose a port on proxy, check we can access the app

# Explore Our Network

- [ ] Check the containers are running in the network
- [ ] What are the IP addresses for our app and proxy container?
- [ ] Hit the proxy! Do you get a greeting? What about on port 80?
- [ ] Remove the network


HOST
SHARED NETWORK
5678
PROXY
APP
PORT 80
PORT 80
172.19.0.3
172.19.0.2
WHAT IT 'LOOKS' LIKE

# Whew, feeling a bit like this?

# Lesson 4

- [ ] Understand what bind mount & volumes are
- [ ] Do a bind mount & create a volume → how are they diff?
- [ ] Create named, anonymous & shared volumes

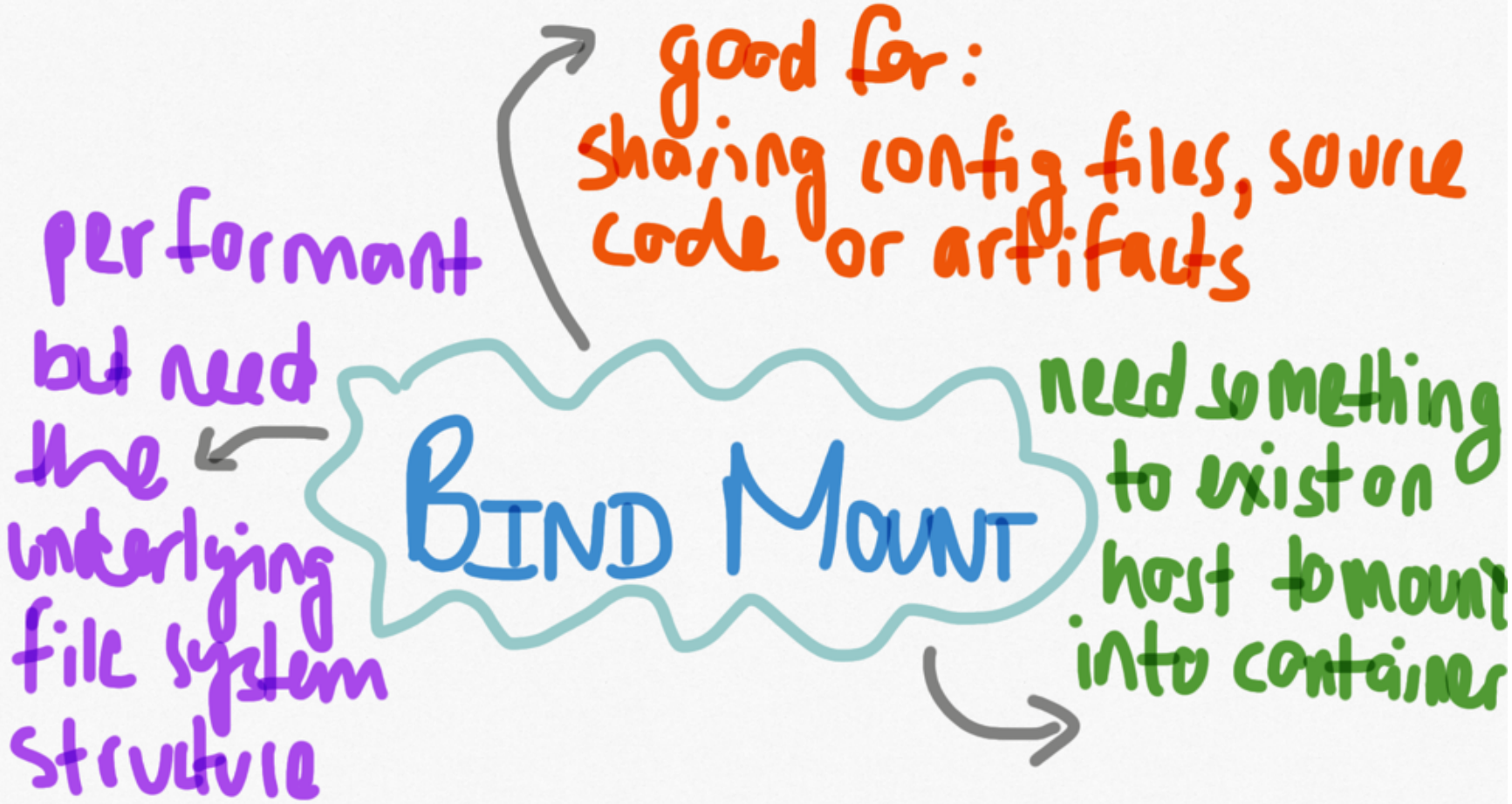
# Resources

- Docker volume
- Docker bind mount
- Understanding volumes

# Exercise Time

# Bind Mount a Directory

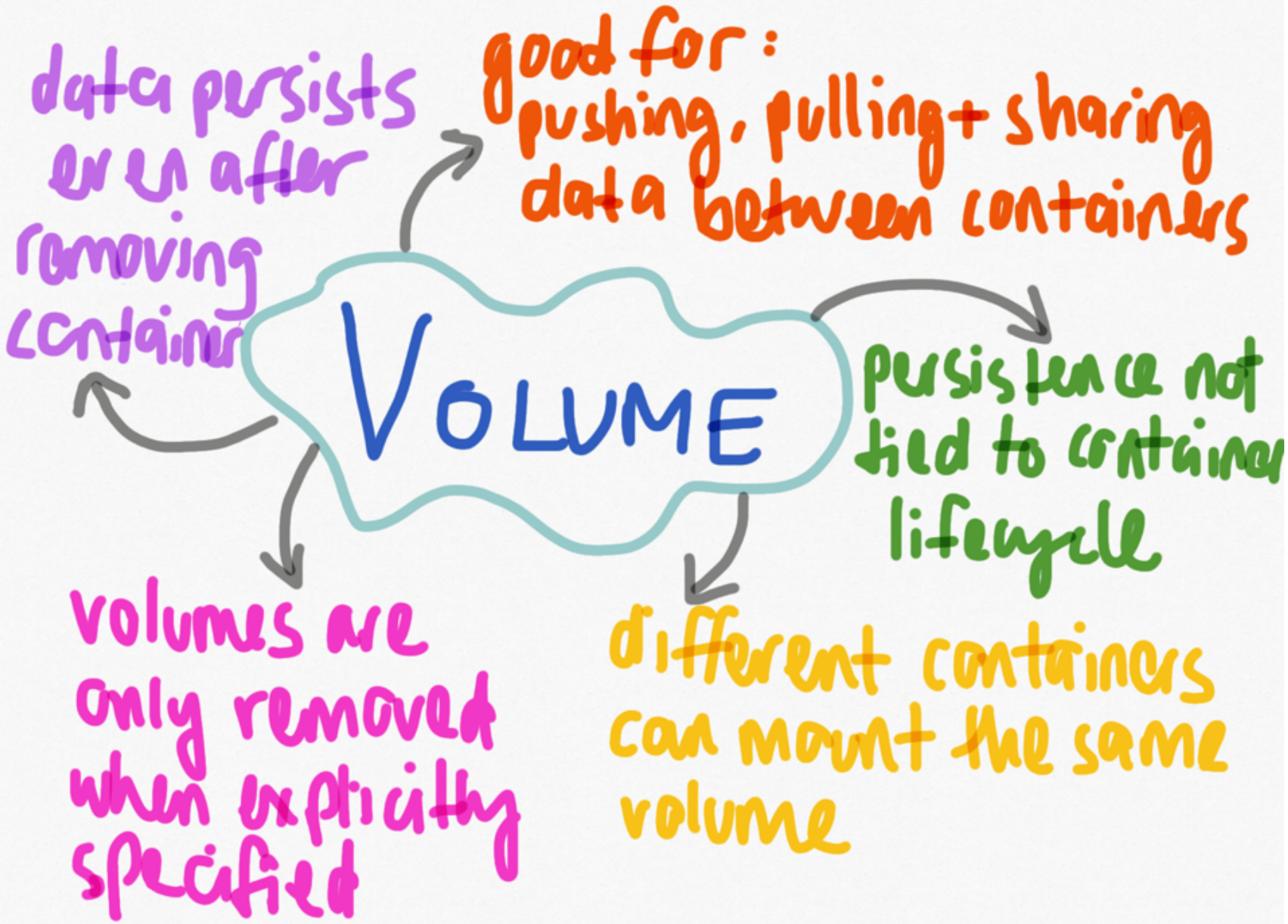
- [ ] Set a $PATH you want to have a copy of on your container
- [ ] Run a container using bind mount w bash shell attached
- [ ] Try to write within the mounted directory
- [ ] Do your changes persist after you stop the container?
- [ ] Do they persist if container is removed?



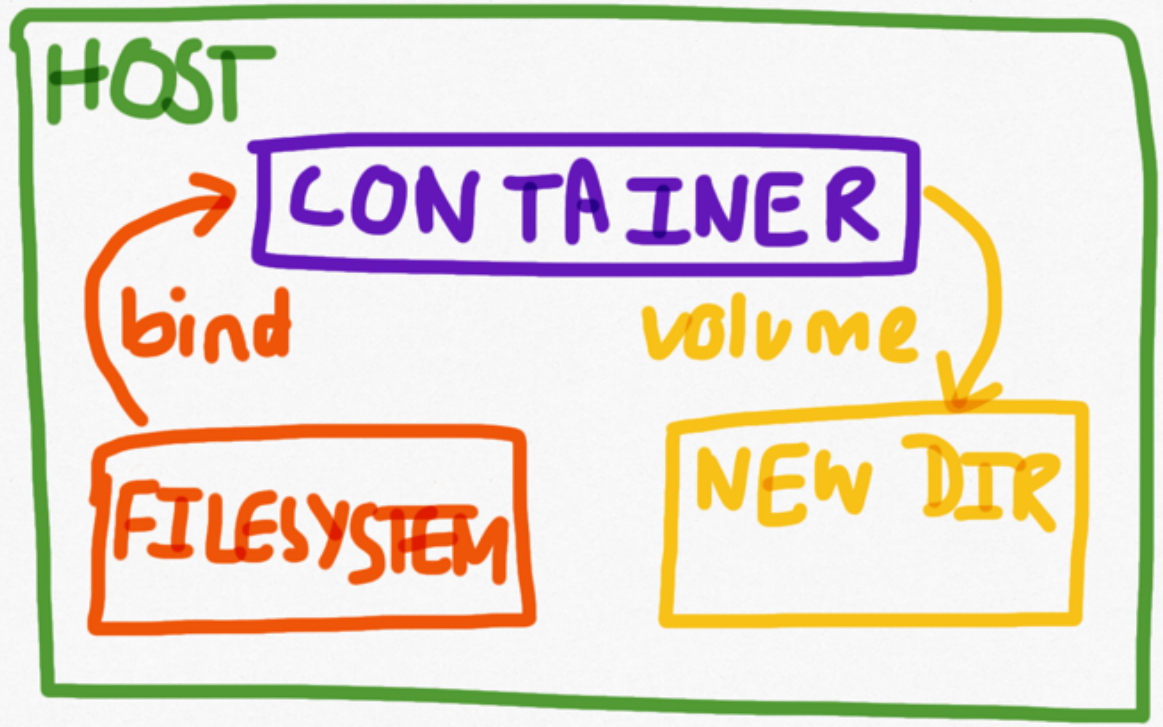

AND THATS ABOUT ALL...

# All About The... Volume

- [ ] Create a named volume that mounts my-cache → /cache
- [ ] Create a file in /cache
- [ ] Stop the container and check if the volume exists
- [ ] Remove the container and check if the volume exists... why?


data persists even after removing container
good for: pushing, pulling + sharing data between containers
VOLUME
persistence not tied to container lifecycle
volumes are only removed when explicitly specified
different containers can mount the same volume

# Bind Mount vs. Volume



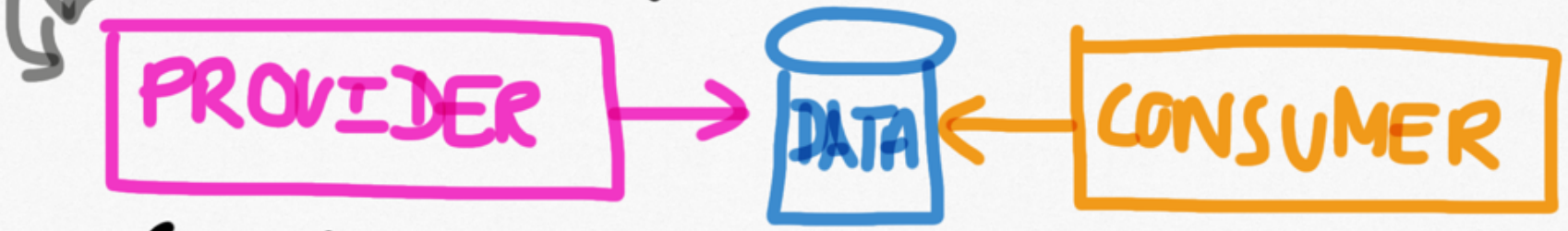
# Anonymous Volume


✓ more storage

- ☐ Start a container using --rm & mount an anon volume at /very_temp
- ☐ How could we make this more secure?
- ☐ Stop the container, what happens?

# Shared Volume

- Why would we wont to share a volume?




- Create a container which 'provides' the volume
- Create a container which 'consumes' it

# Are we done yet? …so close

# Lesson 5

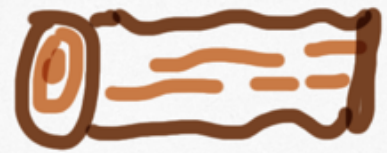
- [ ] Check logs for our container
- [ ] How could we utilise these?

# Exercise Time

# Resources

- Docker Logs

# Loggin' Time



- [ ] Get logs for a container
- [ ] Display time and details in the logs
- [ ] Get real time output in the logs

# Lesson 6

- [ ] Change the greeting of our app using ENV variables
- [ ] Understand how / where ENV variables can be most useful

# Exercise Time

# Resources

- Docker run search for –env
- Code for where environment variable is: https://github.com/realestate-com-au/intro-to-docker/blob/gh-pages/exercises/ciao/index.js#L5

# Use Environment Vars

The application's greeting message is stored in MESSAGE env variable.

- [ ] Set your own greeting for your container

# Clean It All Up

# Woo, finito!


And here are the solutions for reference