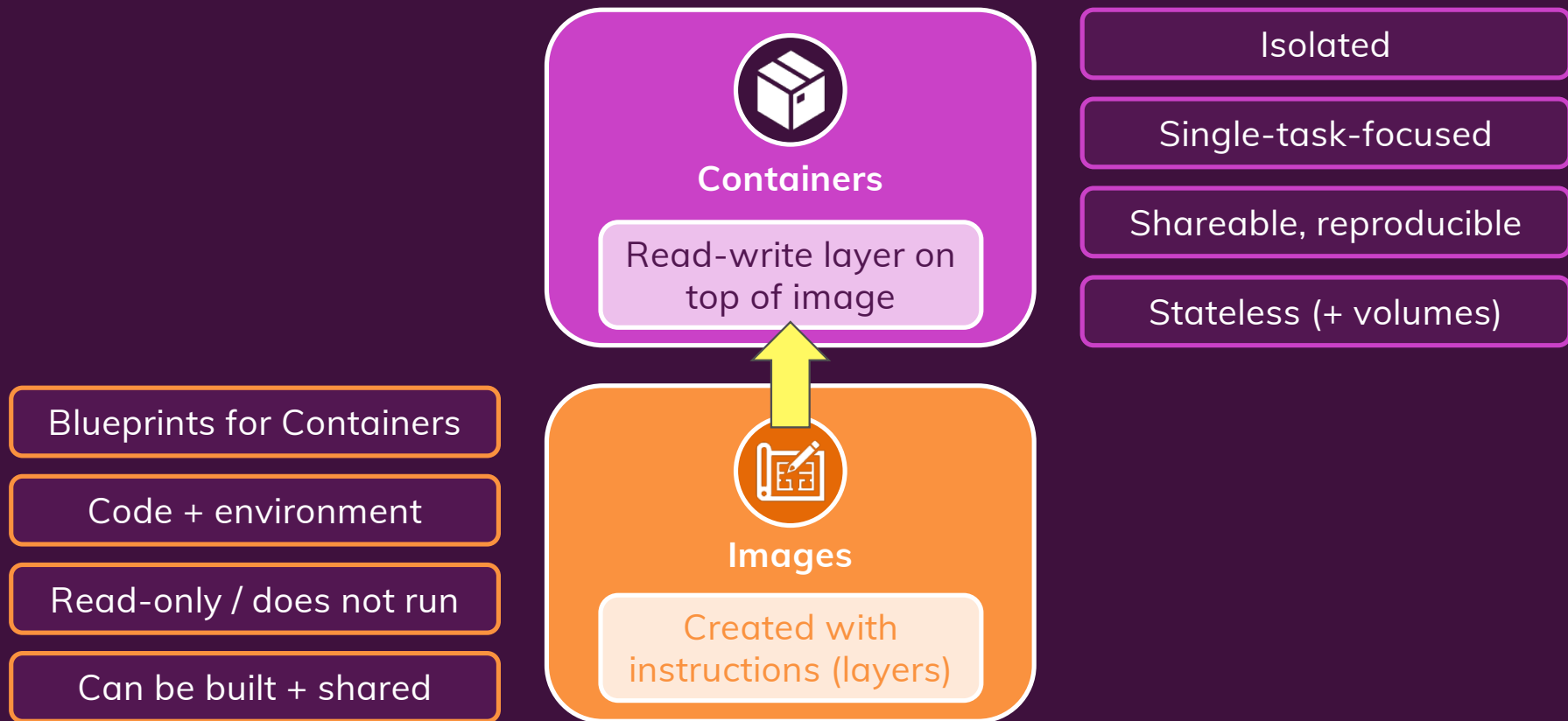


Docker Core Concepts



Key Commands

Build an image based on a Dockerfile

```
docker build -t NAME:TAG .
```

Name & versions
of an image

Build context

Run a container based on a remote or local Image

```
docker run --name NAME --rm -d IMAGE
```

Container
name

Remove once
stopped

Detached
mode

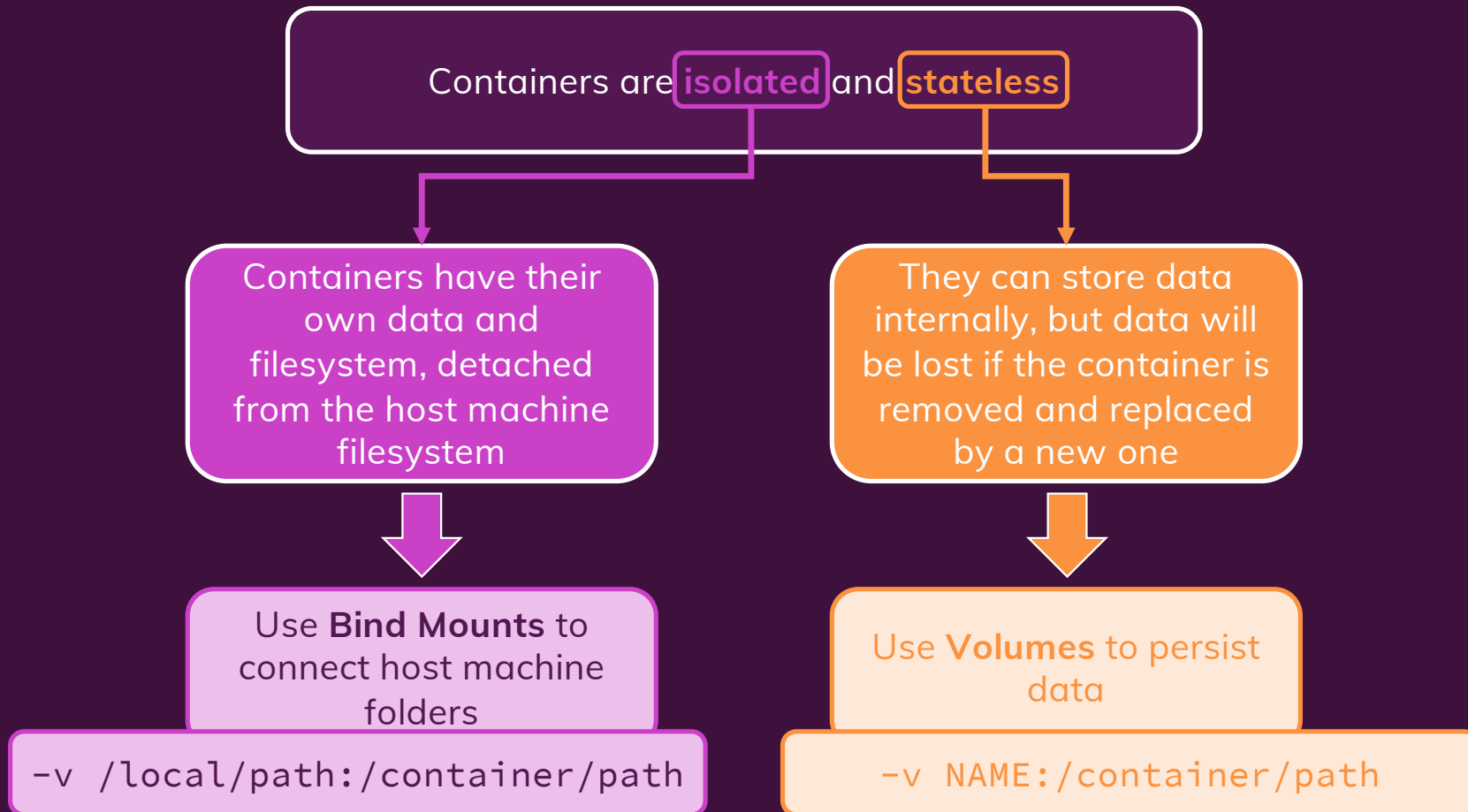
Share (push) an Image to a Registry
(default: DockerHub)

```
docker push REPOSITORY/NAME:TAG
```

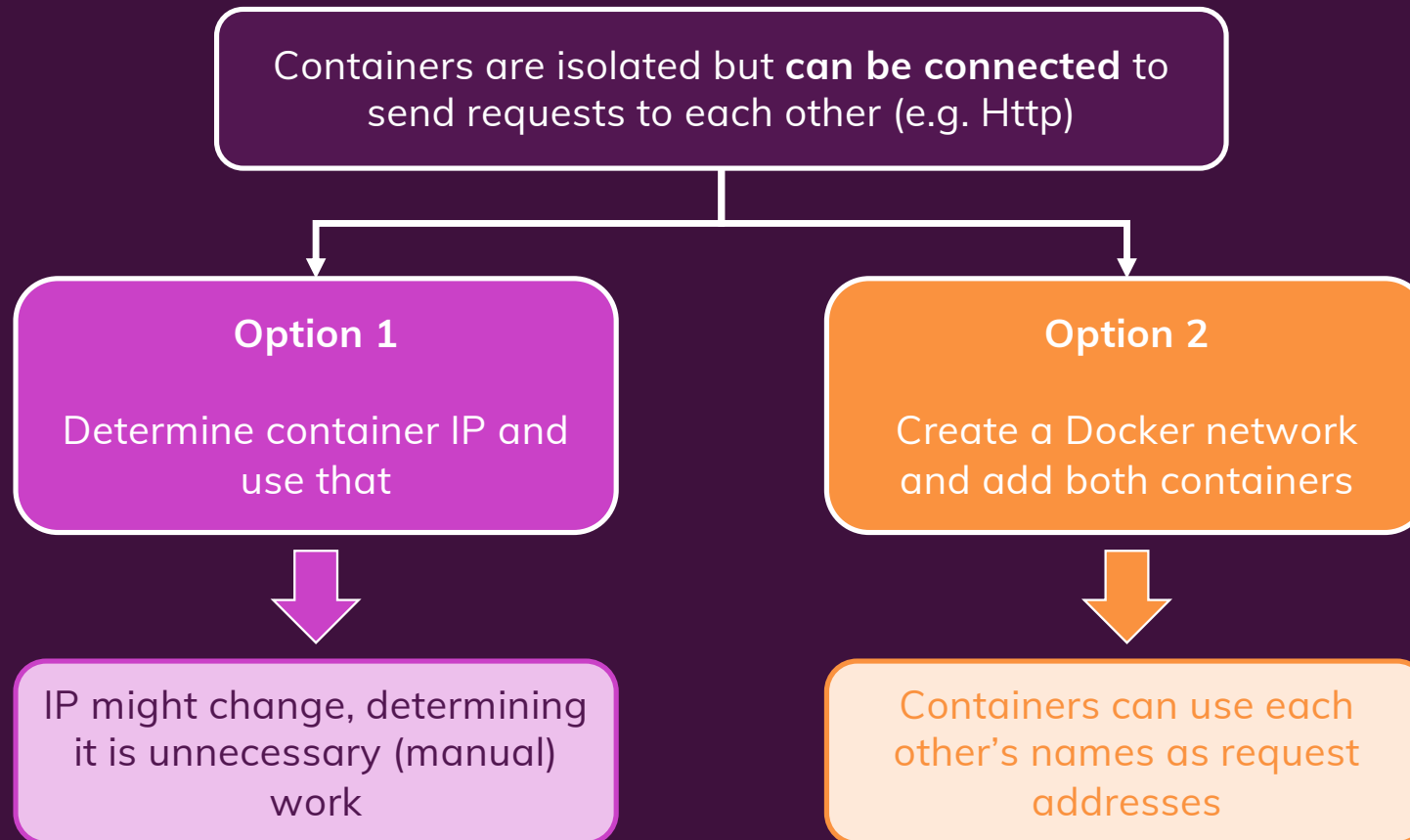
Fetch (pull) an Image from a Registry
(default: DockerHub)

```
docker pull REPOSITORY/NAME:TAG
```

Docker Containers & Data



Docker Containers & Networks



Docker vs Docker Compose

Repeating long **docker build** and **docker run** commands gets annoying – especially when working **with multiple containers**

Docker Compose allows you to pre-define build and run configuration in a **.yaml** file

docker-compose up

Build missing images and start all containers

docker-compose down

Stop all started containers

Local Host (Development) vs Remote Host (Production)

Local Host / Development

Isolated, encapsulated, reproducible development environments

No dependency or software clashes

Remote Host / Production

Isolated, encapsulated, reproducible environments

Easy updates: Simply replace a running container with an updated one

Develop your application in the same environment you'll run it in after deployment

Deployment Is Optional!

It's perfectly fine to use Docker (and Docker Compose) for local development!

Encapsulated
environments for
different projects

No global
installation of tools

Easy to share and
re-produce

Deployment Considerations

Replace **Bind Mounts** with **Volumes** or **COPY**

Multiple containers might need **multiple hosts**

But they can also run on the **same host** (depends on application)

Multi-stage builds help with apps that need a **build step**

Control vs Ease-of-use

You can launch a **remote server**, install **Docker** and run your containers

Full control but you also need to manage everything

You can use a **managed service** instead

Less control and extra knowledge required but easier to use, less responsibility