Docker

- **Docker image** - operating system with preconfigured application (service)
- **Docker container** - running instance created from docker image
- **Data volume** - persistent storage of data outside of container. Can be shared between containers.
- **Dockerfile** - is a recipe which describes the files, environment, and commands that make up an image.
- **docker-compose** - tool for defining and running multi-container Docker application (e.g. web app + mysql db). Compose preserves all volumes used by your services. When docker-compose up runs, if it finds any containers from previous runs, it copies the volumes from the old container to the new container. This process ensures that any data you've created in volumes isn't lost.

command line

run

Create new container based on image and execute optional command inside.

```
docker run -i --rm busybox
docker run -i --rm busybox ps aux
docker run -i --rm debian:jessie-slim
docker run -i --rm nginx bash
```

- **docker run** - run new isolated container ...
  - **-rm** to Automatically remove the container when it exits
  - **–restart always**
  - **–name**

```
docker run --rm -ti -v `pwd`:/opt/myservice remote/path
```

```
--rm                            Automatically remove the container when it
exits
-i, --interactive               Keep STDIN open even if not attached
-t, --tty                       Allocate a pseudo-TTY
-v, --volume=[]                 Bind mount a volume
```

How to run multiple shell commands in docker at once:

```
docker run image /bin/bash -c "cd /some/path && some_command"
```

stopping

`docker stop` sends SIGTERM to PID1 and waits 10 seconds before force kill SIGKILL.

```
docker stop ----time=30 foo
```

```
docker kill ----signal=SIGWINCH apache
docker kill ----signal=SIGQUIT nginx
```

More on handling signals
https://www.ctl.io/developers/blog/post/gracefully-stopping-docker-containers/

restart policy

```
docker update --restart=always 5ba1f7f3d67e
# or usign container name
docker update --restart=always portainer
```

GUIs

https://blog.ouseful.info/2015/08/10/seven-graphical-interfaces-to-docker/

Usefull:

- https://github.com/kevana/ui-for-docker deprecated by:
- https://github.com/portainer/portainer
  - ```
    docker run -d -p 9000:9000 -v
    /var/run/docker.sock:/var/run/docker.sock --name portainer --
    restart always portainer/portainer
    ```
  - https://portainer.readthedocs.io/en/latest/configuration.html

Working:

- docker-compose-ui
  - https://github.com/francescou/docker-compose-ui

Not working:

- http://panamax.io/
  - CoreOS required, so they provide ready Virtualbox or Vagrant recipe
  - https://github.com/CenturyLinkLabs/panamax-ui/wiki/Installing-Panamax
  - Install command:

    ```
    curl http://download.panamax.io/installer/ubuntu.sh | bash
    ```

- Generates a nice graph showing the hierarchy of Docker images in your local image cache
  https://github.com/CenturyLinkLabs/docker-image-graph
  - Doesn't work with current docker version
    ```

Other managers

- https://github.com/ClusterHQ/flocker

Backup

No universal backup solution. Possible scenarios:

- docker images - use `docker save`
- running container:
  - pause (but what with not flushed data?)
  - commit container as image (volumes are not included!)
  - backup a image using `save`
- backup data volumes
  - gracefully stop container to ensure all data are flushed
  - run new container only to execute backup script on other container's volume –`volumes-from`
- connect remotely to service to get dump (i.e. mysql)
- configure service to make daily backup to bind mounted host directory

by committing state to images

To make backup of running container it is need to `commit` its current state and save as docker image. With option `-p` container will be paused before saving snapshot.

```
# docker commit -p portainer portainer1
sha256:c48af304eed09c0ef7f557e6f5e02f10b2637c4c02dd765c186ee29805c31272

# docker images
REPOSITORY              TAG                 IMAGE ID            CREATED
SIZE
portainer1              latest              c48af304eed0        About a
minute ago   9.16 MB
```

Now image can be pushed to remote hub or registry. See `man docker push` Or dumped to file:

```
docker save -o portainer1.tar portainer1
```

To load dump file check `man docker load`

run backup on existing volume

To get data from container outside:

```
$ sudo docker run --rm --volumes-from dbdata -v $(pwd):/backup busybox tar
```

```
cvf /backup/backup.tar /dbdata
```

exporting filesystem

`export` exports only container filesystem and has some limitations: it won't export the data volume (`VOLUME` in Dockerfile or specified by `-v`)

```
docker export $CONTAINER_ID > $CONTAINER_ID-backup.tar
docker import - slava/$CONTAINER_ID-backup < $CONTAINER_ID-backup.tar
```

Build image

Create build directory and `Dockerfile`

Dockerfile

```
FROM debian:jessie-slim
RUN apt-get -y update && apt-get install -y fortunes
CMD /usr/games/fortune -a | cowsay
```

```
docker build -t mydocker .
docker run mydocker
```

ns

Issues

endpoint with name portainer already exists in network bridge

```
# docker start 7cda5b580e16
Error response from daemon: endpoint with name portainer already exists in
network bridge
Error: failed to start containers: 7cda5b580e16
```

https://github.com/moby/moby/issues/23302

Typically when you see containers in docker network inspect output with a ep- prefix, that means it

can be either of 2 cases -

```
  these are stale endpoints left over in the DB. For those cases, docker
network disconnect should help.
  these are remote endpoints seen in other nodes that are part of the
overlay network. The only way to clean them up are from that specific host.
```

Not helping:

```
docker network prune
```

```
docker network disconnect -f bridge portainer
```

Helps:

```
/etc/init.d/docker restart
```