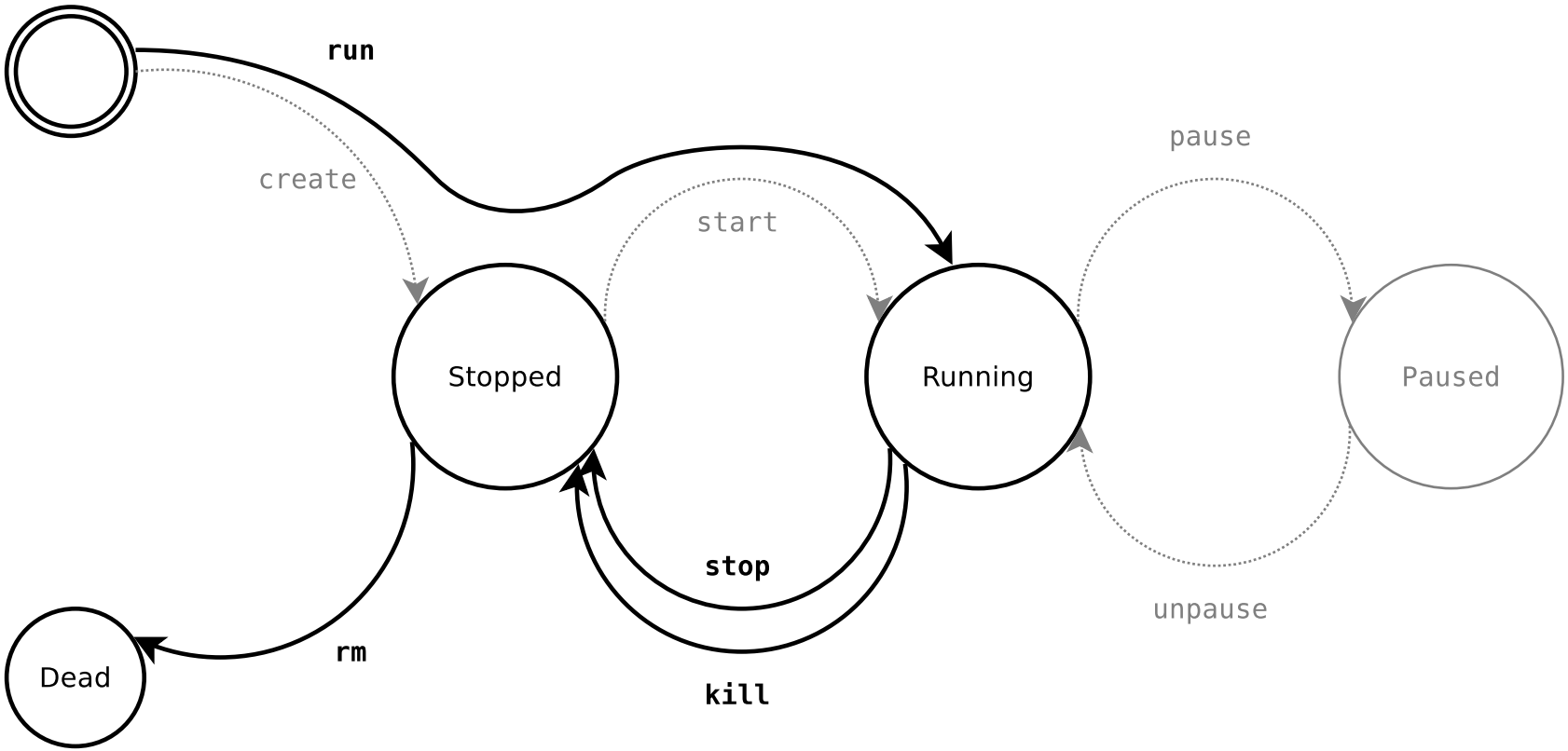
Part 2.
Managing containers

- create/start/stop/remove containers
- inspect containers
- interact, commit new images

Lifecycle of a docker container



Container management commands

command	description
`docker create` *image* [*command*] `docker run` *image* [*command*]	create the container = `create` + `start`
`docker rename` *container new_name* `docker update` *container*	rename the container update the container config
`docker start` *container...* `docker stop` *container...* `docker kill` *container...* `docker restart` *container...*	start the container graceful[2] stop kill (SIGKILL) the container = `stop` + `start`
`docker pause` *container...* `docker unpause` *container...*	suspend the container resume the container
`docker rm` [`-f`[3]] *container...*	destroy the container

[2] send SIGTERM to the main process + SIGKILL 10 seconds later

[3] `-f` allows removing running containers (= `docker kill` + `docker rm`)

Notes about the container lifecycle

- the container filesystem is created in `docker create` and dropped in `docker rm`
  - it is persistent across `stop`/`start`
- the container configuration is mostly static
  - config is set in `create`/`run`
  - `docker update` may change only a few parameters (eg: cpu/ram/blkio allocations)
  - changing other parameters requires destroying and re-creating the container
- other commands are rather basic

```
Usage:  docker create [OPTIONS] IMAGE [COMMAND] [ARG...]

Create a new container

  -a, --attach=[]                 Attach to STDIN, STDOUT or STDERR
  --add-host=[]                   Add a custom host-to-IP mapping (host:ip)
  --blkio-weight=0                Block IO (relative weight), between 10 and 1000
  --cpu-shares=0                  CPU shares (relative weight)
  --cap-add=[]                    Add Linux capabilities
  --cap-drop=[]                   Drop Linux capabilities
  --cgroup-parent=                Optional parent cgroup for the container
  --cidfile=                      Write the container ID to the file
  --cpu-period=0                  Limit CPU CFS (Completely Fair Scheduler) period
  --cpu-quota=0                   Limit CPU CFS (Completely Fair Scheduler) quota
  --cpuset-cpus=                  CPUs in which to allow execution (0-3, 0,1)
  --cpuset-mems=                  MEMs in which to allow execution (0-3, 0,1)
  --device=[]                     Add a host device to the container
  --disable-content-trust=true    Skip image verification
  --dns=[]                        Set custom DNS servers
  --dns-opt=[]                    Set DNS options
  --dns-search=[]                 Set custom DNS search domains
  -e, --env=[]                    Set environment variables
  --entrypoint=                   Overwrite the default ENTRYPOINT of the image
  --env-file=[]                   Read in a file of environment variables
  --expose=[]                     Expose a port or a range of ports
  --group-add=[]                  Add additional groups to join
  -h, --hostname=                 Container host name
  --help=false                    Print usage
  -i, --interactive=false         Keep STDIN open even if not attached
  --ipc=                          IPC namespace to use
  --kernel-memory=                Kernel memory limit
  -l, --label=[]                  Set meta data on a container
  --label-file=[]                 Read in a line delimited file of labels
  --link=[]                       Add link to another container
  --log-driver=                   Logging driver for container
  --log-opt=[]                    Log driver options
  --lxc-conf=[]                   Add custom lxc options
  -m, --memory=                   Memory limit
  --mac-address=                  Container MAC address (e.g. 92:d0:c6:0a:29:33)
  --memory-reservation=           Memory soft limit
  --memory-swap=                  Total memory (memory + swap), '-1' to disable swap
  --memory-swappiness=-1          Tuning container memory swappiness (0 to 100)
  --name=                         Assign a name to the container
  --net=default                   Set the Network for the container
  --oom-kill-disable=false        Disable OOM Killer
  -P, --publish-all=false         Publish all exposed ports to random ports
  -p, --publish=[]                Publish a container's port(s) to the host
  --pid=                          PID namespace to use
  --privileged=false              Give extended privileges to this container
  --read-only=false               Mount the container's root filesystem as read only
  --restart=no                    Restart policy to apply when a container exits
  --security-opt=[]               Security Options
  --stop-signal=SIGTERM           Signal to stop a container, SIGTERM by default
  -t, --tty=false                 Allocate a pseudo-TTY
  -u, --user=                     Username or UID (format: <name|uid>[:<group|gid>])
  --ulimit=[]                     Ulimit options
  --uts=                          UTS namespace to use
  -v, --volume=[]                 Bind mount a volume
  --volume-driver=                Optional volume driver for the container
  --volumes-from=[]               Mount volumes from the specified container(s)
  -w, --workdir=                  Working directory inside the container
```

```
Usage:  docker start [OPTIONS] CONTAINER [CONTAINER...]

Start one or more stopped containers

  -a, --attach=false          Attach STDOUT/STDERR and forward signals
  --help=false                Print usage
  -i, --interactive=false     Attach container's STDIN
```

```
Usage:  docker stop [OPTIONS] CONTAINER [CONTAINER...]

Stop a running container.
Sending SIGTERM and then SIGKILL after a grace period

  --help=false       Print usage
  -t, --time=10      Seconds to wait for stop before killing it
```

```
Usage:  docker restart [OPTIONS] CONTAINER [CONTAINER...]

Restart a container

  --help=false       Print usage
  -t, --time=10      Seconds to wait for stop before killing the container
```

```
Usage:  docker kill [OPTIONS] CONTAINER [CONTAINER...]

Kill a running container

  --help=false         Print usage
  -s, --signal=KILL    Signal to send to the container
```

```
Usage:  docker rm [OPTIONS] CONTAINER [CONTAINER...]

Remove one or more containers

  -f, --force=false      Force the removal of a running container (uses SIGKILL)
  --help=false           Print usage
  -l, --link=false       Remove the specified link
  -v, --volumes=false    Remove the volumes associated with the container
```

```
Usage:  docker pause [OPTIONS] CONTAINER [CONTAINER...]

Pause all processes within a container

  --help=false    Print usage
```

`docker run` — Run a container

https://docs.docker.com/reference/run/

`docker run [ options ] image [ arg0 arg1...]`

→ create a container and start it

- the container filesystem is initialised from image *image*
- *arg0..argN* is the command run inside the container (as PID 1)

```
$ docker run debian /bin/hostname
f0d0720bd373
$ docker run debian date +%H:%M:%S
17:10:13
$ docker run debian true ; echo $?
0
$ docker run debian false ; echo $?
1
```

`docker run` — Foreground mode vs. Detached mode

- Foreground mode is the default
  - *stdout* and *stderr* are redirected to the terminal
  - `docker run` propagates the exit code of the main process
- With `-d`, the container is run in detached mode:
  - displays the ID of the container
  - returns immediately

```
$ docker run debian date
Tue Jan 20 17:32:07 UTC 2015
$ docker run -d debian date
4cbdefb3d3e1331ccf7783b32b47774fefca426e03a2005d69549f3ff06b9306
$ docker logs 4cbdef
Tue Jan 20 17:32:16 UTC 2015
```

`docker run` — TTY allocation

Use `-t` to allocate a pseudo-terminal for the container

→ without a tty

```
$ docker run debian ls
bin
boot
dev
...
$ docker run debian bash
$
```

→ with a tty (`-t`)

```
$ docker run -t debian ls
bin   dev  home  lib64   mnt  proc  run   selinux  sys  usr
boot  etc  lib   media   opt  root  sbin  srv      tmp  var
$ docker run -t debian bash
root@10d90c09d9ac:/#
```

`docker run` — interactive mode

- By default containers are non-interactive
  - *stdin* is closed immediately
  - terminal signals are not forwarded[4]

```
$ docker run -t debian bash
root@6fecc2e8ab22:/# date
^C
$
```

- With `-i` the container runs interactively
  - *stdin* is usable
  - terminal signals are forwarded to the container

```
$ docker run -t -i debian bash
root@78ff08f46cdb:/# date
Tue Jan 20 17:52:01 UTC 2015
root@78ff08f46cdb:/# ^C
root@78ff08f46cdb:/#
```

[4] `^C` only detaches the terminal, the container keeps running in background

`docker run` — override defaults (1/2)

user (`-u`)

```
$ docker run debian whoami
root
$ docker run -u nobody debian whoami
nobody
```

working directory (`-w`)

```
$ docker run debian pwd
/
$ docker run -w /opt debian pwd
/opt
```

`docker run` — override defaults (2/2)

environment variables (`-e`)

```
$ docker run debian sh -c 'echo $FOO $BAR'

$ docker run -e FOO=foo -e BAR=bar debian sh -c 'echo $FOO $BAR'
foo bar
```

hostname (`-h`)

```
$ docker run debian hostname
830e47237187
$ docker run -h my-nice-container debian hostname
my-nice-hostname
```

`docker run` — set the container name

`--name` assigns a name for the container
(by default a random name is generated)

```
$ docker run -d -t debian
da005df0d3aca345323e373e1239216434c05d01699b048c5ff277dd691ad535
$ docker run -d -t --name blahblah debian
0bd3cb464ff68eaf9fc43f0241911eb207fefd9c1341a0850e8804b7445ccd21
$ docker ps
CONTAINER ID   IMAGE       COMMAND      CREATED             ..  NAMES
0bd3cb464ff6   debian:7.5  "/bin/bash"  6 seconds ago           blahblah
da005df0d3ac   debian:7.5  "/bin/bash"  About a minute ago      drunk_darwin
$ docker stop blahblah drunk_darwin
```

Note: Names must be unique

```
$ docker run --name blahblah debian true
2015/01/20 19:31:21 Error response from daemon: Conflict, The name blahblah is already assigned
to 0bd3cb464ff6. You have to delete (or rename) that container to be able to assign blahblah to a
 container again.
```

docker run — autoremove

By default the container still exists after command exit

```
$ docker run --name date-ctr debian date
Tue Jan 20 18:38:21 UTC 2015
$ docker start date-ctr
date-ctr
$ docker logs date-ctr
Tue Jan 20 18:38:21 UTC 2015
Tue Jan 20 18:38:29 UTC 2015
$ docker rm date-ctr
date-ctr
$ docker start date-ctr
Error response from daemon: No such container: date-ctr
2015/01/20 19:39:27 Error: failed to start one or more containers
```

With `--rm` the container is automatically removed after exit

```
$ docker run --rm --name date-ctr debian date
Tue Jan 20 18:41:49 UTC 2015
$ docker rm date-ctr
Error response from daemon: No such container: date-ctr
2015/01/20 19:41:53 Error: failed to remove one or more containers
```

Common `rm` idioms

Launch a throwaway container for debugging/testing purpose

```
$ docker run --rm -t -i debian
root@4b71c9a39326:/#
```

Remove all zombie containers

```
$ docker ps -a
CONTAINER ID  IMAGE       COMMAND     CREATED              STATUS
2b291251a415  debian:7.5  "hostname"  About a minute ago   Exited (0) About a mi
6d36a2f07e18  debian:7.5  "false"     2 minutes ago        Exited (1) 2 minutes
0f563f110328  debian:7.5  "true"      2 minutes ago        Exited (0) 2 minutes
4b57d0327a20  debian:7.5  "uname -a"  5 minutes ago        Exited (0) 5 minutes
$ docker container prune
WARNING! This will remove all stopped containers.
Are you sure you want to continue? [y/N] y
Deleted Containers:
2b291251a415
6d36a2f07e18
0f563f110328
4b57d0327a20
```

Inspecting the container

command	description
`docker ps`	list running containers
`docker ps -a`	list all containers
`docker logs` [`-f`[5]] *container*	show the container output (*stdout+stderr*)
`docker top` *container* [*ps options*]	list the processes running inside the containers[6]
`docker stats` [*container*]	display live usage statistics[7]
`docker diff` *container*	show the differences with the image (modified files)
`docker port` *container*	list port mappings
`docker inspect` *container...*	show low-level infos (in json format)

[5] with `-f`, `docker logs` follows the output (à la `tail -f`)

[6] `docker top` is the equivalent of the `ps` command in unix

[7] `docker stats` is the equivalent of the `top` command in unix

Interacting with the container

command	description
`docker attach` *container*	attach to a running container (stdin/stdout/stderr)
`docker cp` *container:path hostpath*|-	copy files from the container
`docker cp` *hostpath*|- *container:path*	copy files into the container
`docker export` *container*	export the content of the container (tar archive)
`docker exec` *container args...*	run a command in an existing container (**useful** for debugging)
`docker wait` *container*	wait until the container terminates and return the exit code
`docker commit` *container image*	commit a new docker image (snapshot of the container)

docker commit example

```
$ docker run --name my-container -t -i debian
root@3b397d383faf:/# cat >> /etc/bash.bashrc <<EOF
> echo 'hello!'
> EOF
root@3b397d383faf:/# exit
$ docker start --attach my-container
my-container
hello!
root@3b397d383faf:/# exit
$ docker diff my-container
C /etc
C /etc/bash.bashrc
A /.bash_history
C /tmp
$ docker commit my-container hello
a57e91bc3b0f5f72641f19cab85a7f3f860a1e5e9629439007c39fd76f37c5dd
$ docker rm my-container
my-container
$ docker run --rm -t -i hello
hello!
root@386ed3934b44:/# exit
$ docker images -t
511136ea3c5a Virtual Size: 0 B
  af6bdc397692 Virtual Size: 115 MB
    667250f9a437 Virtual Size: 115 MB Tags: debian:wheezy, debian:latest
      a57e91bc3b0f Virtual Size: 115 MB Tags: hello:latest
```