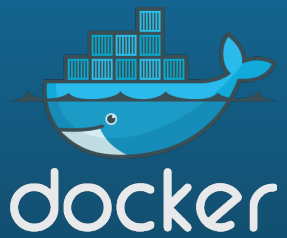
What's new in Docker 1.13.0

spoiler alert: a lot!!

Victor Vieux / vieux@docker.com / @vieux



Restructure CLI commands

```
docker images -> docker image list
docker create -> docker container create
…

docker build and docker run unchanged

fully backward compatible, legacy commands still showed in --help.
Use DOCKER_HIDE_LEGACY_COMMANDS=1 to hide legacy commands.
```

```
[node1] (local) root@10.0.17.3 ~
$ DOCKER_HIDE_LEGACY_COMMANDS=1 docker --help

Usage:  docker COMMAND

A self-sufficient runtime for containers

Management Commands:
  checkpoint  Manage checkpoints
  container   Manage containers
  image       Manage images
  network     Manage networks
  node        Manage Swarm nodes
  plugin      Manage plugins
  secret      Manage Docker secrets
  service     Manage services
  stack       Manage Docker stacks
  swarm       Manage Swarm
  system      Manage Docker
  volume      Manage volumes

Commands:
  build       Build an image from a Dockerfile
  deploy      Deploy a new stack or update an existing stack
  login       Log in to a Docker registry
  logout      Log out from a Docker registry
  run         Run a command in a new container
  search      Search the Docker Hub for images
  version     Show the Docker version information
```

```
Management Commands:
  checkpoint  Manage checkpoints
  container   Manage containers
  image       Manage images
  network     Manage networks
  node        Manage Swarm nodes
  plugin      Manage plugins
  secret      Manage Docker secrets
  service     Manage services
  stack       Manage Docker stacks
  swarm       Manage Swarm
  system      Manage Docker
  volume      Manage volumes

Commands:
  attach      Attach to a running container
  build       Build an image from a Dockerfile
  commit      Create a new image from a container's changes
  cp          Copy files/folders between a container and the local filesystem
  create      Create a new container
  deploy      Deploy a new stack or update an existing stack
  diff        Inspect changes on a container's filesystem
  events      Get real time events from the server
  exec        Run a command in a running container
  export      Export a container's filesystem as a tar archive
  history     Show the history of an image
  images      List images
  import      Import the contents from a tarball to create a filesystem image
  info        Display system-wide information
  inspect     Return low-level information on Docker objects
  kill        Kill one or more running containers
  load        Load an image from a tar archive or STDIN
  login       Log in to a Docker registry
```

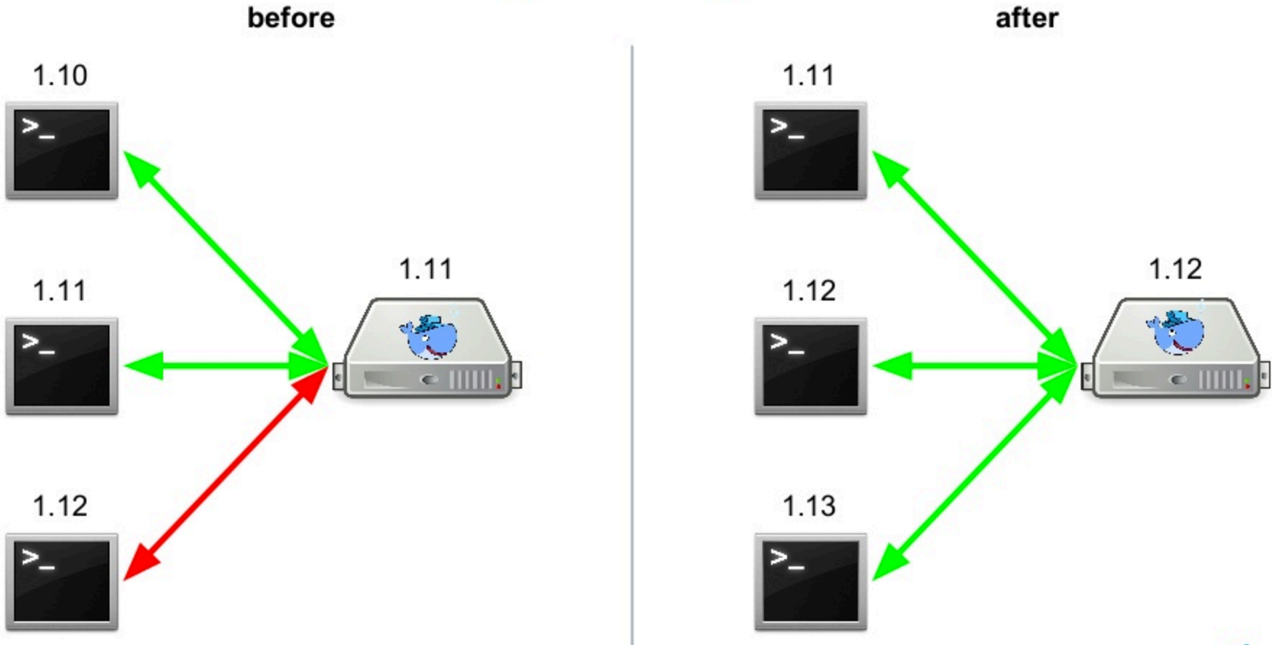
Experimental not a separate build anymore

`--experimental` added to the `dockerd`

docker will round-trip with the daemon to show usage accordingly.

CLI backward compatibility

An older cli will now be able to talk to a newer daemon (same round-trip as experimental).



docker

Default encryption at rest

All of the data stored in swarm mode is encrypted at rest, and allows users to own key

```
docker swarm update --autolock
docker swarm unlock
docker swarm unlock-key --rotate
```

```
[node1] (local) root@10.0.17.3 ~
$ docker swarm init --autolock --advertise-addr 10.0.17.3
Swarm initialized: current node (gcviujg6pcgxbifp8l1hda6l6) is now a manager.

To add a worker to this swarm, run the following command:

    docker swarm join \
    --token SWMTKN-1-0llug9vurcgt8sujy79lmvkqusxzi43x4ac0rer9mgb5jzqoo6-ev963i7gsuhy03y4bfadfct2g
    10.0.17.3:2377

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.

To unlock a swarm manager after it restarts, run the `docker swarm unlock`
command and provide the following key:

    SWMKEY-1-taR/pCo6Xk0+QP2hQcoNiN5y38YVIfb9Gw7FuUH+2vQ

Please remember to store this key in a password manager, since without it you
will not be able to restart the manager.
[node1] (local) root@10.0.17.3 ~
$ docker swarm unlock-key
To unlock a swarm manager after it restarts, run the `docker swarm unlock`
command and provide the following key:

    SWMKEY-1-taR/pCo6Xk0+QP2hQcoNiN5y38YVIfb9Gw7FuUH+2vQ

Please remember to store this key in a password manager, since without it you
will not be able to restart the manager.
```

```
[node1] (local) root@10.0.17.3 ~
$ docker swarm unlock-key --rotate
Successfully rotated manager unlock key.

To unlock a swarm manager after it restarts, run the `docker swarm unlock`
command and provide the following key:

    SWMKEY-1-cG44wX8+4l6hVdszvE99Y6Ug/MQf2yMudKx/Qch53Ys

Please remember to store this key in a password manager, since without it you
will not be able to restart the manager.


[node1] (local) root@10.0.17.3 ~
$ reboot
[node1] (local) root@10.0.17.3 ~
$ docker swarm unlock
Please enter unlock key:
```

```
[node1] (local) root@10.0.17.3 ~
$ docker swarm update --autolock
Swarm updated.
To unlock a swarm manager after it restarts, run the `docker swarm unlock`
command and provide the following key:

    SWMKEY-1-G8PUWFEKLXTWCqx30+R3+QEWH74vn7EvZL4vOvp8OMI

Please remember to store this key in a password manager, since without it you
will not be able to restart the manager.


[node1] (local) root@10.0.17.3 ~
$ docker swarm update --autolock=false
Swarm updated.
```

Secrets Object - Only in Swarm mode

- A new secrets object, along with API endpoints and CLI commands
- Available in swarm mode only
- Secrets are stored in the Raft log associated with the swarm cluster
- Mounted in tmpfs inside a container
- Secrets are immutable

```
docker secret create SECRET_NAME FILE (may be STDIN)
docker secret rm SECRET_NAME
docker secret inspect SECRET_NAME
docker service create --name app --secret SECRET_NAME my_app
```

Create a secret

```
[node1] (local) root@10.0.17.3 ~
$ echo 'supa_duba_secret_string' > my_secret.txt
[node1] (local) root@10.0.17.3 ~
$ docker secret create my_secret my_secret.txt
n2w3pis53oz5t604fix3m8edx
```

```
[node1] (local) root@10.0.17.3 ~
$ docker secret inspect my_secret
[
    {
        "ID": "n2w3pis53oz5t604fix3m8edx",
        "Version": {
            "Index": 59
        },
        "CreatedAt": "2017-02-02T15:36:13.801005863Z",
        "UpdatedAt": "2017-02-02T15:36:13.801005863Z",
        "Spec": {
            "Name": "my_secret"
        }
```

Use a secret

```
[node1] (local) root@10.0.17.3 ~
$ docker service create --name app --secret my_secret nginx
m1a8e7sl23kw734tq1y10ugz8
[node1] (local) root@10.0.17.3 ~
$ docker service inspect app
[
    { "ContainerSpec": {
          "Image": "nginx:latest@sha256:f2d384a6ca8ada733df555be3edc427f2e5f285ebf468a
          "DNSConfig": {},
          "Secrets": [
              {
                  "File": {
                      "Name": "my_secret",
                      "UID": "0",
                      "GID": "0",
                      "Mode": 292
                  },
                  "SecretID": "n2w3pis53oz5t604fix3m8edx",
                  "SecretName": "my_secret"
              }
          ]
```

Use a secret / Delete a secret

```
[node2] (local) root@10.0.17.4 ~
$ docker exec -ti 99d679751be4 bash
root@99d679751be4:/# ls /var/run/secrets/
my_secret
root@99d679751be4:/# cat /var/run/secrets/my_secret
supa_duba_secret_string
```

```
[node1] (local) root@10.0.17.3 ~
$ docker secret rm my_secret
Error response from daemon: rpc error: code = 3 desc = secret 'my_secret
' is in use by the following service: app
```

Rotate a secret

```
[node1] (local) root@10.0.17.3 ~
$ echo 'new_secret' | docker secret create my_secretv2 -
dtt3rw4boxsph4q9zx8whe4g7

[node1] (local) root@10.0.17.3 ~
$ docker service update --secret-rm my_secret \
>   --secret-add source=my_secretv2,target=my_secret \
>   app
app
[node1] (local) root@10.0.17.3 ~
$ docker service ps app
ID             NAME       IMAGE          NODE   DESIRED STATE  CURRENT STATE
jgyhsjq6wgbn   app.1      nginx:latest   node1  Running        Running 50 seconds
616vyp8hcmo6    \_ app.1  nginx:latest   node2  Shutdown       Shutdown 55 second

[node1] (local) root@10.0.17.3 ~
$ docker exec -ti 1258febcba9b bash
root@1258febcba9b:/# cat /var/run/secrets/my_secret
new_secret
```

Plugins out of experimental

```
docker plugin create vieux/sshfs /path/to/rootfs
docker plugin enable vieux/sshfs
```

or

```
docker plugin install vieux/sshfs

docker plugin set vieux/sshfs DEBUG=1
```

Compose to Swarm

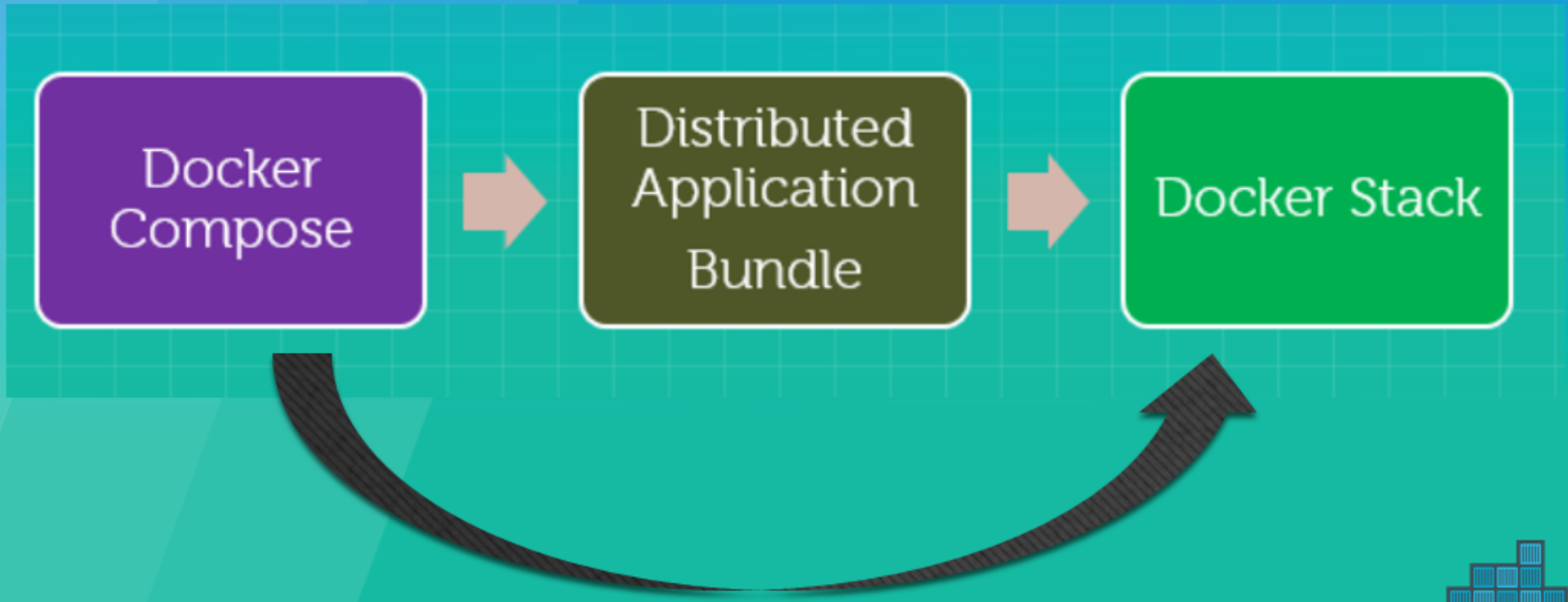
```
docker stack deploy --compose-file= foo

docker stack list
docker stack rm foo
```

New v3 compose format, difference from v2:

- Removed the non portable options (build, volume-from, etc…)
- Added Swarm specific options (replicas, mode, etc…)

Compose to Swarm



$sudo docker stack deploy --compose-file ./docker-compose.yml myapp

docker

Data management commands

```
docker system df to show docker disk usage.

docker system prune to remove unused data.
docker container/image/network/volume prune
```

```
~/work docker system df
TYPE                TOTAL               ACTIVE              SIZE                RECLAIMABLE
Images              49                  21                  10.66 GB            -3.701e+08 B (-3%)
Containers          65                  0                   1.74 GB             1.74 GB (100%)
Local Volumes       62                  25                  7.769 GB            4.173 GB (53%)
```

```
~/work docker system prune
WARNING! This will remove:
        - all stopped containers
        - all volumes not used by at least one container
        - all networks not used by at least one container
        - all dangling images
Are you sure you want to continue? [y/N] y
Deleted Containers:

Total reclaimed space: 9.774 GB
```

```
~/work docker system df
TYPE                TOTAL               ACTIVE              SIZE                RECLAIMABLE
Images              41                  0                   10.39 GB            10.39 GB (100%)
Containers          0                   0                   0 B                 0 B
Local Volumes       0                   0                   0 B                 0 B
```

(experimental) other features

`docker build --squash` will produce only one layer.

`docker service logs` to show aggregated logs of a service.

docker-init

New binary **docker-init** is now shipped with docker to kill zombies processes.

It uses tini, but can be replaced by your own with **--init** and **--init-path** on **dockerd**

A bunch of new flags

```
docker build --network
docker service create --host
docker service create --tty --hostname --dns …
```

Various Orchestration Enhancements

Pin image by digest for rolling updates

```
redis:latest -> redis:latest@sha256:13478120c…
```

Service update failure thresholds and rollback

Various Orchestration Enhancements

HA scheduling, spreads out tasks from a service

Improvements to constraints, UX

- Provide reason for unschedulable tasks
- Global services should obey constraints
- Remove tasks that no longer satisfy constraints


dockercon 17
austin APRIL 17-20