### Docker常用命令

docker build -t friendlyhello . # Create image using this directory's Dockerfile

docker run -p 4000:80 friendlyhello # Run "friendlyname" mapping port 4000 to 80

docker run -d -p 4000:80 friendlyhello # Same thing, but in detached mode

docker container ls # List all running containers

docker container ls -a # List all containers, even those not running

docker container stop <hash> # Gracefully stop the specified container

docker container kill <hash> # Force shutdown of the specified container

docker container rm <hash> # Remove specified container from this machine

docker container rm $(docker container ls -a -q) # Remove all containers

docker image ls -a # List all images on this machine

docker image rm <image id> # Remove specified image from this machine

docker image rm $(docker image ls -a -q) # Remove all images from this machine

docker login # Log in this CLI session using your Docker credentials

docker tag <image> username/repository:tag # Tag <image> for upload to registry

docker push username/repository:tag # Upload tagged image to registry

docker run username/repository:tag # Run image from a registry

docker stack ls # List stacks or apps

docker stack deploy -c <composefile> <appname> # Run the specified Compose file

docker service ls # List running services associated with an app

docker service ps <service> # List tasks associated with an app

docker inspect <task or container> # Inspect task or container

docker container ls -q # List container IDs

docker stack rm <appname> # Tear down an application

docker swarm leave --force # Take down a single node swarm from the manager

docker-machine create --driver virtualbox myvm1 # Create a VM (Mac, Win7, Linux)

docker-machine create -d hyperv --hyperv-virtual-switch "myswitch" myvm1 # Win10

docker-machine env myvm1 # View basic information about your node

docker-machine ssh myvm1 "docker node ls" # List the nodes in your swarm

docker-machine ssh myvm1 "docker node inspect <node ID>" # Inspect a node

docker-machine ssh myvm1 "docker swarm join-token -q worker" # View join token

docker-machine ssh myvm1 # Open an SSH session with the VM; type "exit" to end

docker node ls # View nodes in swarm (while logged on to manager)

docker-machine ssh myvm2 "docker swarm leave" # Make the worker leave the swarm

docker-machine ssh myvm1 "docker swarm leave -f" # Make master leave, kill swarm

docker-machine ls # list VMs, asterisk shows which VM this shell is talking to

docker-machine start myvm1 # Start a VM that is currently not running

docker-machine env myvm1 # show environment variables and command for myvm1

eval $(docker-machine env myvm1) # Mac command to connect shell to myvm1

& "C:\Program Files\Docker\Docker\Resources\bin\docker-machine.exe" env myvm1 | Invoke-Expression # Windows command to connect shell to myvm1

docker stack deploy -c <file> <app> # Deploy an app; command shell must be set to talk to manager (myvm1), uses local Compose file

docker-machine scp docker-compose.yml myvm1:~ # Copy file to node's home dir (only required if you use ssh to connect to manager and deploy the app)

docker-machine ssh myvm1 "docker stack deploy -c <file> <app>" # Deploy an app using ssh (you must have first copied the Compose file to myvm1)

eval $(docker-machine env -u) # Disconnect shell from VMs, use native docker

docker-machine stop $(docker-machine ls -q) # Stop all running VMs

docker-machine rm $(docker-machine ls -q) # Delete all VMs and their disk images

//////////////////for docker machine////////////////////////

$ docker-machine config <docker-machine name>

$ docker-machine env <docker-machine name>

$ docker-machine inspect <docer-machine name>

$ docker-machine ip <docker-machine name>

$ docker-machine kill <docker-machine name>

$ docker-machine provision <docker-machine name>`

$ docker-machine regenerate-certs <docker-machine name>

$ docker-machine restart <docker-machine name>

$ docker-machine ssh <docker-machine name>

$ docker-machine start <docker-machine name>

$ docker-machine status <docker-machine name>

$ docker-machine stop <docker-machine name>

$ docker-machine upgrade <docker-machine name>

$ docker-machine url <docker-machine name>

修改docker daemon的options:

### Configure Docker to start on boot

Most current Linux distributions (RHEL, CentOS, Fedora, Ubuntu 16.04 and higher) use [systemd](https://docs.docker.com/engine/installation/linux/linux-postinstall/#systemd) to manage which services start when the system boots. Ubuntu 14.10 and below use [upstart](https://docs.docker.com/engine/installation/linux/linux-postinstall/#upstart).

systemd

$ sudo systemctl enable docker

To disable this behavior, use disable instead.

$ sudo systemctl disable docker

### 配置docker registry中国镜像

The URL of the registry mirror for China is registry.docker-cn.com. You can pull images from this mirror just like you do for other registries by specifying the full path, including the registry, in your docker pull command, for example:

$ docker pull registry.docker-cn.com/library/ubuntu

You can add "https://registry.docker-cn.com" to the registry-mirrors array in [/etc/docker/daemon.json](https://docs.docker.com/engine/reference/commandline/dockerd/#daemon-configuration-file) to pull from the China registry mirror by default.

{

"registry-mirrors": ["https://registry.docker-cn.com"]

}

Save the file and reload Docker for the change to take effect.

Or, you can configure the Docker daemon with the --registry-mirror startup parameter:

$ dockerd --registry-mirror=https://registry.docker-cn.com

### 加入现有host VM到docker-machine

docker-machine create -d generic \

--generic-ip-address 192.168.100.2 \

--generic-ssh-key $HOME/.ssh/id\_rsa \

--generic-ssh-user dockeradmin \

--generic-ssh-port 22 \

dev-db

### docker compose

$ cat docker-compose.yml

web:

image: myusername/web

$ docker-compose up -d

$ docker-compose scale web=3

### Force DHCP address to renew in Ubuntu Linux

//force the DHCP address to renew

sudo dhclient -r; sudo dhclient

### set environment for container and/or service

docker run -e VARIABLE=VALUE ...:

web:

environment:

- DEBUG=1

Just like with docker run -e, you can set environment variables on a one-off container with docker-compose run -e:

docker-compose run -e DEBUG=1 web python console.py

You can set default values for any environment variables referenced in the Compose file, or used to configure Compose, in an [environment file](https://docs.docker.com/compose/env-file/) named .env:

$ cat .env

TAG=v1.5

$ cat docker-compose.yml

version: '3'

services:

web:

image: "webapp:${TAG}"

### Container name的用处(default bridge network没法通过ping <container name>，除非container启动时用 --link)

**Service and Container Hostnames** update automatically when a service scales up or down or redeploys. As a user, you can configure service names, and Docker Cloud uses these names to find the IP of the services and containers for you. You can use hostnames in your code to provide abstraction that allows you to easily swap service containers or components.

**Service links** create environment variables which allow containers to communicate with each other within a stack, or with other services outside of a stack. You can specify service links explicitly when you create a new service or edit an existing one, or specify them in the stackfile for a service stack.

You can use hostnames to connect any container in your Docker Cloud account to any other container on your account without having to create service links or manage environment variables. This is the recommended service discovery method.

### Discovering containers on the same service or stack

A container can always discover other containers on the same stack using just the **container name** as hostname. This includes containers of the same service. Similarly, a container can always discover other services on the same stack using the **service name**.

For example, a container webapp-1 in the service webapp can connect to the container db-1 in the service db by using db-1 as the hostname. It can also connect to a peer container, webapp-2, by using webapp-2 as the hostname.

A container proxy-1 on the same stack could discover all webapp containers by using the **service name** webapp as hostname. Connecting to the service name resolves as an A [round-robin](http://en.wikipedia.org/wiki/Round-robin_DNS) record, listing all IPs of all containers on the service webapp.

To detach from the container1 container and leave it running, use the keyboard sequence **CTRL-p CTRL-q**. If you wish, attach to container2 and repeat the commands above.

//attach to docker container

docker container attach <docker name/docker container id>

//container linkage

sudo docker run -itd --name=container6 --link container3:container3 busybox

### login the container(docker exec)

docker exec -it <container id /container name> bash

### Run the .war inside tomcat with docker environment

//the Dockerfile

FROM tomcat

ADD test.war /usr/local/tomcat/webapps/

CMD ["catalina.sh", "run"]

//copy the .war in the same folder as Dockerfile

//build the image

sudo docker build -t testapp .

//run the docker container and access the testapp via host port 8082

sudo docker run -itd -p 8082:8080 testapp

### 将Maven的dependency刷新到eclipse的build path

mvn eclipse:clean

mvn eclipse:eclipse

if will prepare the eclipse .classpath file for you. That is, these commands are run against maven from the command line i.e. outside of eclipse.

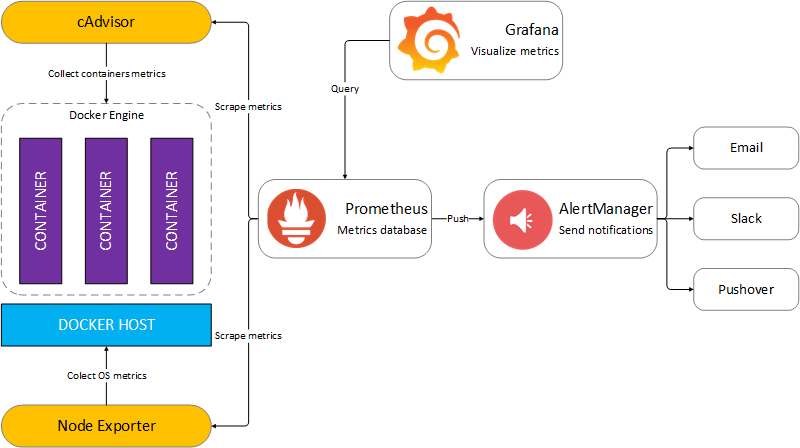
### Docker监控

开源软件：Prometheus

Prometheus配置：<https://docs.docker.com/config/thirdparty/prometheus/>

Prometheus（metrics database） + cAdvisor(container metrics) +Grafana(图形化工具)+ NodeExporter(主机Metrics)+ AlertManager (Alert管理)+Caddy(反向代理/认证)

<https://stefanprodan.com/2016/a-monitoring-solution-for-docker-hosts-containers-and-containerized-services/>



git clone https://github.com/stefanprodan/dockprom

cd dockprom

ADMIN\_USER=admin ADMIN\_PASSWORD=admin docker-compose up -d

### Tensorflow Docker Image

$ sudo docker run -itd -p 8888:8888 -p 6606:6606 -p 6006:6006 -v ~/tensorflow\_models:/models tensorflow/tensorflow:latest-py3

### Kill the container running a given service

sudo docker kill -s HUP `sudo docker container ls -f "name=monitoring\_caddy\*" -q`

monitoring\_caddy=docker service name

### make the Linux Bash shell prompt shorter:

PS1='\u:\W\$ '