- normalnie chcemy 10 apek, kazda na osobnej maszynce

- male zuzycie na serwerze

- wirtualka pomaga, odpalamy 10 na tej samej maszynce

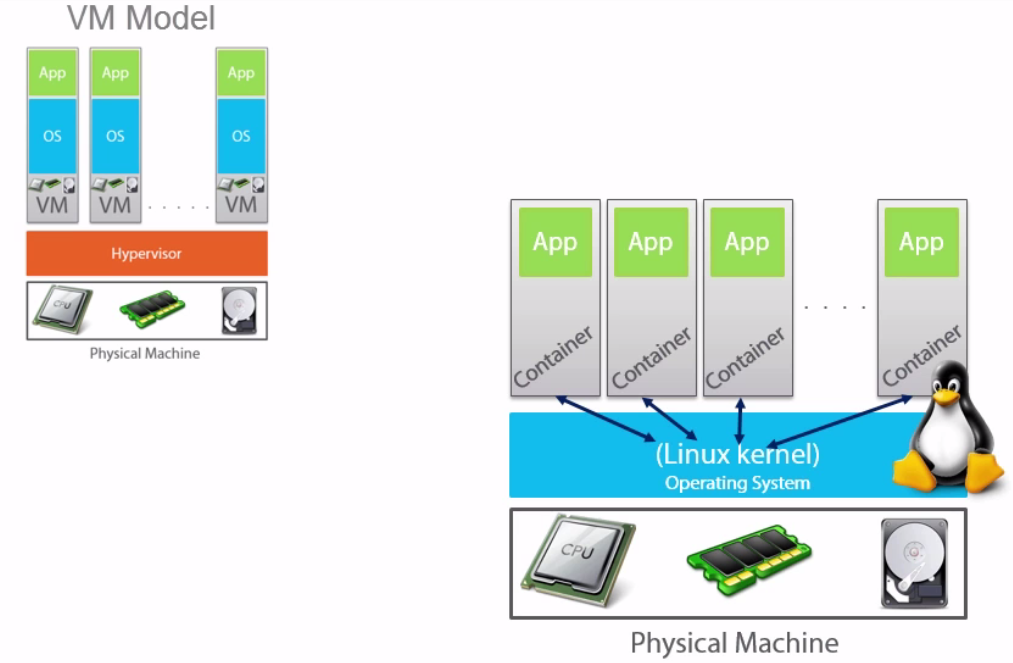
- minusem, ze potrzebujemy 10 OS dla 10 apek, czyli duzy kosz dla malej apki

- kazdy VM moze wymagac drogiej licencji

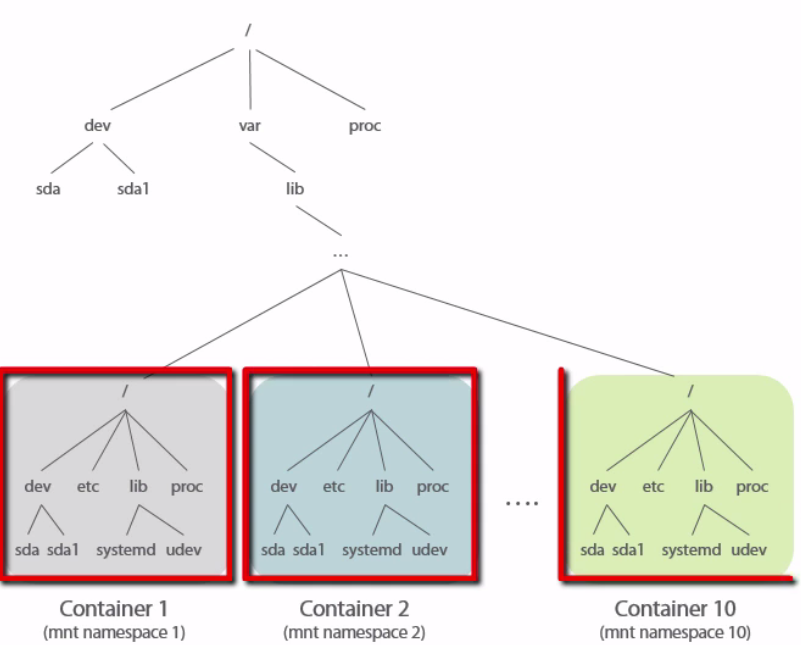
- z pomoca nadchodzi kontener – docker , jest duzo lzejszy

-- zapewnie bezpieczny odizolowany runetime, pobiera mniej procka, ram i powiezchni dysku

-- kontener to wyizolowana user space



- kazdy kontener ma swoj root file system

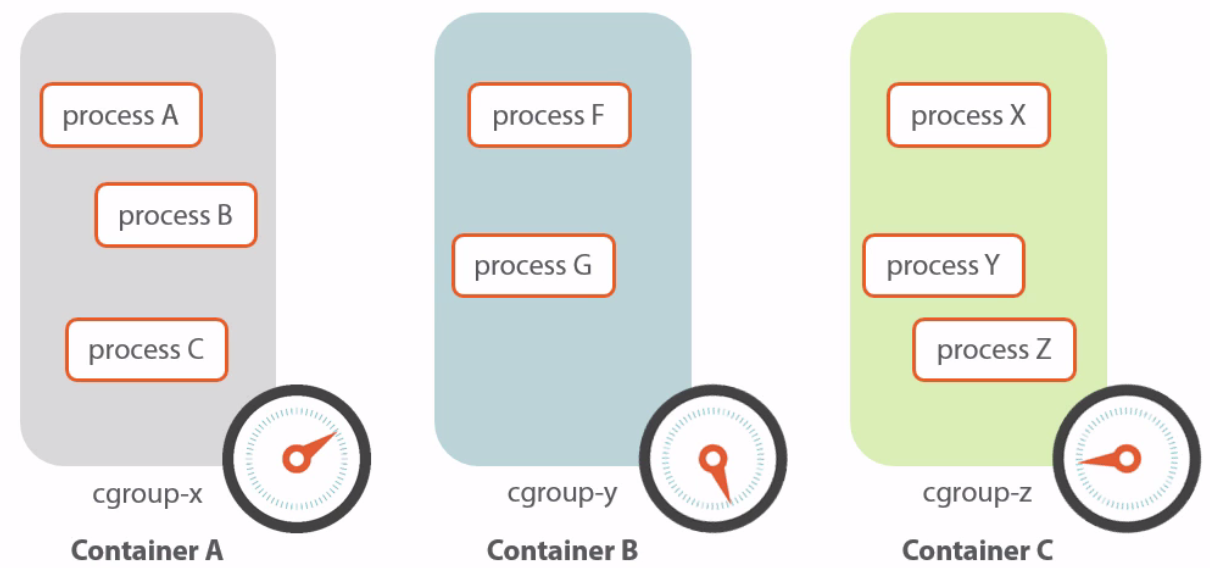


- kazdy ma swoj process tree – proces w jednym nie moze wyslac sygnalu/zabic procesu w innym , wlasna tablica rutowania, to dzieki kernel namespace linuxa, partycjom , dzieki user mozemy miec root wewnatrz kontenera, nie na zewnatrz





- dodatkowo mamy cgroups – kontrol grupy – mozna gupowac resources i aplikowac limity , CPU, memory, block IO

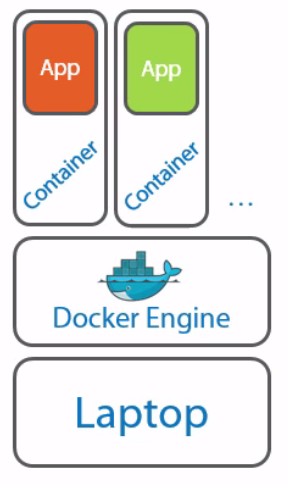


- CAPABILITIES – grain control user or proces get , np. Dzielimy root na mniejsze uprawnienia, mozna do procesu tylko wybrane uprawnienie, nie dajemy za duzo, np. Cap\_net\_bind\_service, dzial na white list, wszystko poza nie dozwolone

DOCKER

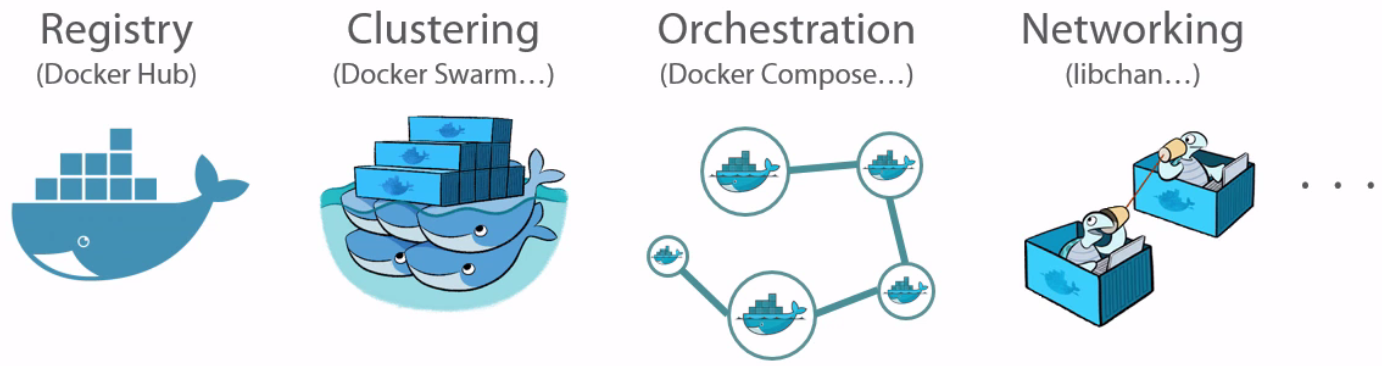
- implementacja technologii kontenerów, środowisko uruchomieniowe, silnik dokera

- zapewnie standard runtime



Taki kontener z laptopa mozna potem wrzucic wszedzie, na chmure – AWS, Azure, data center

- staje sie platformą – docker image format, kontener runtime, (3ci to service discovery)



FUTURE

* Powstanie dla windows, ale uruchamiane na windowsach obrazy, bo wspoldzieli OS
* Dzialanie na VM

Instalacja ubuntu i dokera

- doker jako klient serwer, klient komendy sle do serwera, moze byc na tej samej maszynce lub przez siec, oba na raz instalowane jako paczka, serwer tworzy kontenery

- na raz sciaga i instaluje

- pisany w go

- docker info – mowi o kontenerach i obrazach , storage driver zależy od linuxa , ubuntu ma aufs

- przed update backup danych !

- potrzebuje ROOT!

- ls –l /run – pokazuje że docker nasluchuje na docker.sock , root ale grupa docker, wiec musimy dodac do niej uzytkownikow, zeby mogli dzialac , wazne kto tam jest, bo moze wysadzic dockera

Tworzymy kontener – docker run –it ubuntu /bin/bash, startuje nowy kontener, (-it – interactive and …), jako Ubuntu image, w nim bash process mozemy odpalic

- musi user byc w grupie : sudo gpasswd –a myuser docker, sprawdzamy cat /etc/group – widzimy, ze dodany , trzeba sie przelogowac

- po odpaleniu jestesmy w kontener command line , na koncu id konenera

---

- konfiguracja by docker sluchal na network port zamiast local unique socket I polaczymy sie do niego z innej maszynki

- netstat –tlp – widzimy ze docker nie slucha na porcie

- service docker stop – zatrzymanie

- startujemy docker by sluchal na porcie i jako demon

-- docker –H ip:port –d &

- teraz wewnetrzne nie mozemy odpalic docker info, bo nie slucha na socket

- teraz na innej maszynce ustawiam DOCKER\_HOST na ta co nasluchuje :

-- 

- po wywlolaniu docker version, wersja tego do ktorego sie laczymy zdalnie

- export DOCKER\_HOST = - wraca do poprzedniej wersji

- jak wystawimy na porcie, to nie ma restrykcji do grupy i kazdy moze uzyc !

- mozna 2 opcje , wew i zew - 

- mozna pod ubuntu obraz dokerowy, centos contener pod ubuntu , czyli np. Instaluje yum, ale lekkie, ale moge kolejne instalowac rzeczy jak np. Vim

- exit wychodzi z dockera, zamyka

- docker ps – widac,ze procesu nie ma

- docker ps –a – pokazuje wszystkie kontenery co byly wystartowane na maszynce , widzimy ich id

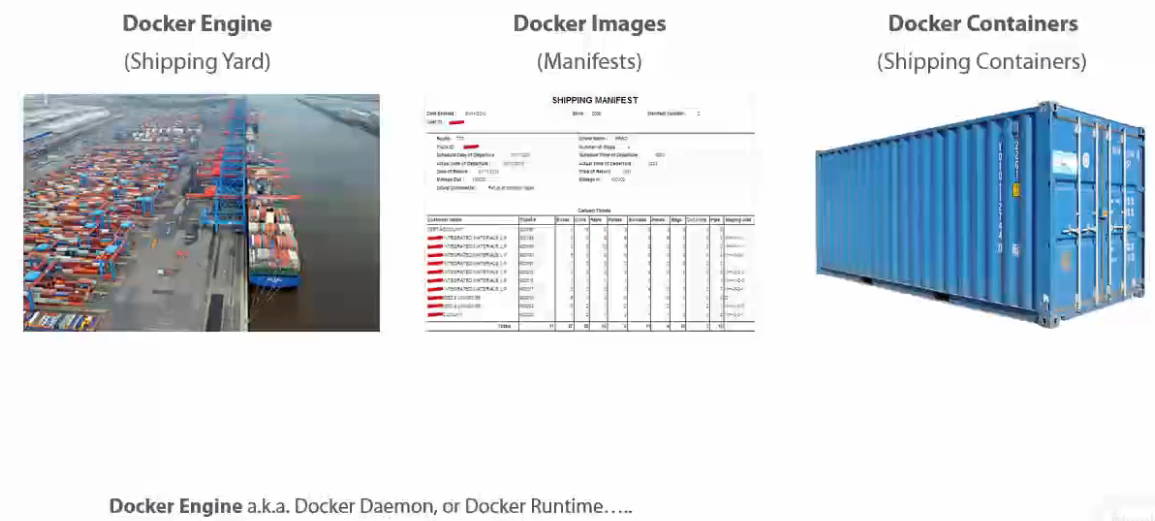
-  - wypisze pelne hash instancji z poczatkowym id

- teraz 

- jest tam temp, gdzie mamy dokumenty zapisane podczas wczesniejszej sesji kontenera – MAMY DOSTEP Z HOSTA DO PLIKOW STWORZONYCH PRZEZ KONTENER !!!!

- MOZEMY WROCIC – docker start id – staruje docker, ale nie przelacza na niego

- docker attach ID, teraz konsola konenera i mozemy czytac pliki wczesniejsze



- programy tez po malu dostarczane w dni na test w tygodnie na prod, wiec tu kontenery tez , kodujemy w kontenerze na lapku, a zmieniamy/odpalamy w kontenerzy na prod , nie ma u mnie dziala ;) pamietac o zaufanym kodzie – obrazy z prywatnych z hub

- engine – ma standardy, wszedzie taki sam, daje nam cala infrastrukture, polaczenia, system plikow itc., czyli gdzie kolwiek odpalamy to dzialanie takie samo

--

Major docker componnts -> docker images

- image to jak manifest przesylki

- mamy kontener i instrukcje jak zbudowac

- image potrzebne do odpalenia,

jest to konkretny kontener pod konkretnym systemem

- image jest jak template w swiecie VM

- jak okresle system bez wersji to sciagnie z docker hub ostatnia wersje



- jak docker pull –a fedora, sciagnie wszystkie

- docker images fedora – wylistuje dostepne lokalnie, widzimy id i tagi

- mozna docker pull nazwa



- np. /var/lib/docker/aufs

Kontenery

- to instancje odpalane z danego image, jak image jest build-time to kontener runtime

- kontener to instancja linuxa, okreslamy jaki image i jakie polecenie/proces podczas run



- kontener optymalizuje i zaciaga tylko niezbedne pakiety, bez gui i bez programow, by byla mala paczka, dla cereos tylko 20MG, jak chcemy cos wiecej to dociagam/instaluje, fedora wieksza,

- najpierw odpalam a potem attach by sie do niego podlaczyc

- jak odpale jakis program to moze nie byc, bo lekkie, trzeba dociagac

- wychodze z kontenera – ctrl+P+Q

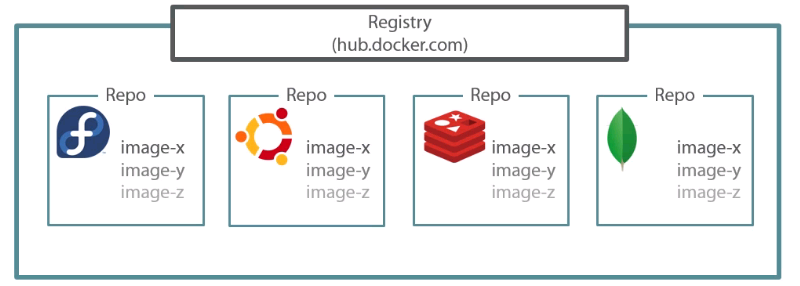
- docker ps pokaze odpalone obrazy

- docker ps –a pokaze tez wczesniej odpalone

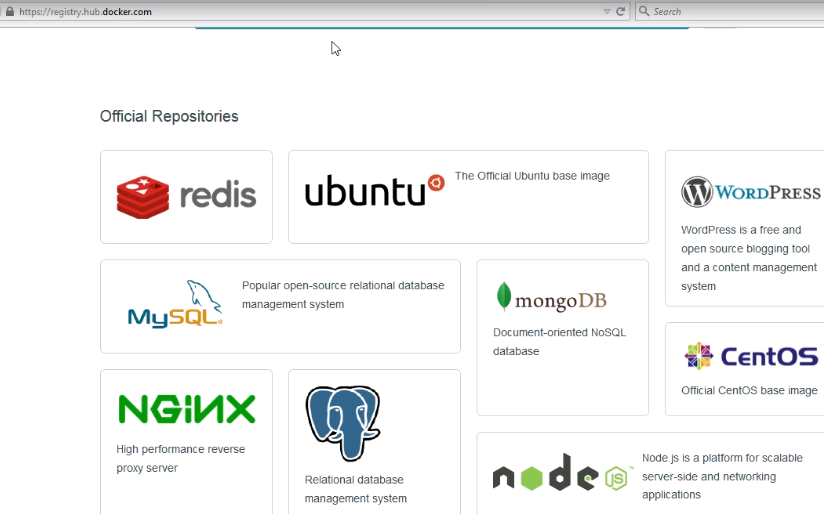


- obraz sciagamy z repository,

- repository zyje wewnatrz registry



- mozna z przegladarki



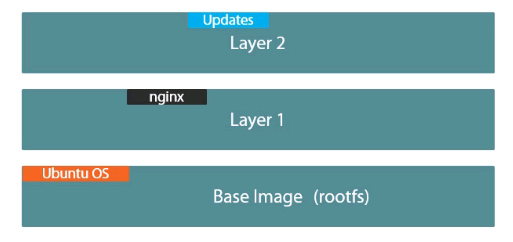
- tu sa oficjalne repo, pod danym ile wersj, jak pobrac etc.

- poza tym sa user repos , tworzone przez docker community, mozemy je tez pobierac, ale wazne by ufac kontenerowi, bo moze namieszac

Wiecej o images

- 1 image sklada sie z np. 3 warst – images

- na przyklad na dole mamy system (root file system), co pozwala nam, ze nasz obraz jest ubuntu, potem nad tym instalujemy apke np. Nginx, a nad tym updates, apchech albo inne



- to uzyteczne, bo warstwy nakladamy po kolei, i mozemy podmieniac warstwe

- na danym image mozna wiele kontenerow

- to efektywne dla cache i config changes

- kolejne config changes to moze byc nowa wartwa

- jak sciaga, to te warstwy co juz ma to juz ich nie

- Kazdy image – warstwa ma swoje id, wiec mozemy powiedziec by zlozwyl na dole wartwa id, potem id, na tym id

- wszystkie wartwy skladaja sie w jedna, przy czym gorna warstwa przykrywa dolne, czyli jak sa konflikty to gorna nadpisuje



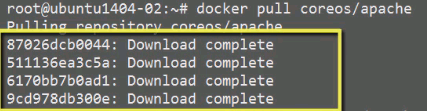
- wszystkie warstwy sa jako read only, tylko gorna jest modyfikowalna

- ponizej rootfs jest jeszcze na starcie bootfs, ale zanika po starcie

- wszystkie zmiany w kontenerze na rune time sa poprzez top layer

- kazdy obraz z docker images ma kilka warstw

- gdy na przyklad siagamy, to ida 4 warstwy

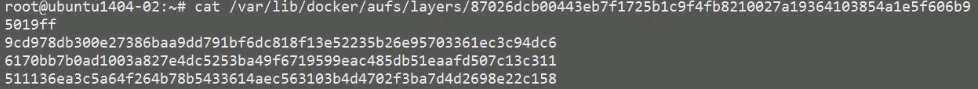


- docker images –tree – wylisuje nam z warstwami obrazy, na dole gorna wartwa, to polecenie moze byc usuniete, by engine lzejszy byl

- warstwy trzymane w osobnym katalogu, na przyklad :



- jak przeczytamy ktorys, to wypisze ponizsze warstwy



- mozna zobaczyc wewnetrzne pliki danej warstwy, configi



- najnizsza warstwa ma kompletny ubunto root file system

Kopiowanie obrazu czy konteneru na inny host

- jeden ze sposobow to push na docker hub i pobranie w innym host

- jak odpalam docker bez wersji to ostatnia bierze

- na bash moge cos odpalic

- ponizej nie dajemy –it, bo odpalamy z poleceniem i sie zamknie po wykonaniu, poza tym nie robimy attach do niego



- powyzej zrobilismy zmiane, mozna tak dodawac konfigi, cos zainstalowac np. Apache

- teraz docker ps –a – pokaze nam wczesniej odpalone i mozemy zamrozic ten ostatni – zrobic z niego image

Czyli moge dodawac kolejne obrazy instalacja czegos, a potem z nich obraz calosciowy



- nazywamy go fridge, taki ma tag – po wypisaniu docker images, i wersje teraz latest

- teraz docker history fridge wypisze nam polecenia wywolane przy tworzeniu danych fridge

- w ten sposob mozna tysiace image z cieka warstwa, lekkich

- jak chce przeniesc na inny host obraz to pakuje do tara



- mozna sprawdzic co w srodku



- kopiujemy w dowolny sposob na inny host

- teraz importujemy na maszynce innej :



- widac po docker images ze jest

- - startujemy

- mozna teraz przeczytac plik obrazu – widac co zapisaz z zewnatrz, z maszynki

Na podstawie metadata w image odpalony kontener, stawiane wszystkie warstwy, ale kazdy kontener stawia na gorze swoja wlasna writable layer

- na poczatku ta warstwa pusta, ale po kolejnych zapytaniach zajmuje przestrzen

- dzieki gornej warstwie zachowuje sie jak normalny system, mimo ze dolne warstwy readonly

- wszystkie dockery dzielajądro core OS (procesy)

- z perspekrywy deva to jest full sandbox

- kontenery startuja zwykle z jedna apka

- dobre dla arch micro serwisow

Moge odpalic w tle – detach, sie bedzie wykonywal i sie zamknie, np. Pinganie 30 razy



- jak sprawdze docker ps, to widze ze jest up

- jak docker top id – pokazuje jaki proce wykonywany w kontenerze – czyli wypisze nasze ping x –c 30

- kontener zyje dopoki proces w nim zyje

- jak chcemy konkretny obraz ubuntu to docker run ubuntu:4.14

Mozna przy run okreslic ile cpu dajemy, ile memory

- to wszystkie cpu - 

- 

- docker inspect id – daje nam szczegolowe info o kontenerze – id dostaje po run

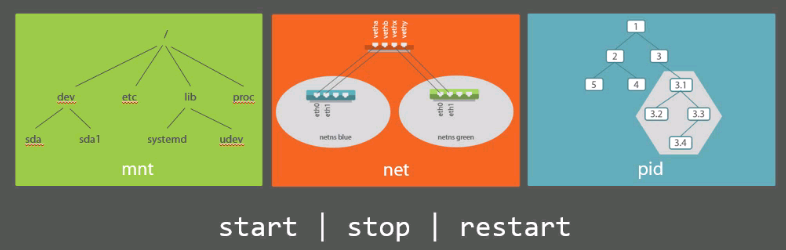
Ale tez z docker ps –a

- mozemy sie podlaczy po nazwie – docker attach name

- widac jakie procesy ida – np. Ping

- ctrl c – powrot do konsoli hosta, konczy dzialanie kontenera

Kontener ma wlasny system plikow, polaczenia i procesy



- z –it interactive mode i attach, widzimy w prompt short id

- mozemy wyjsc exit, ctrl+c – terminate, a ctrl p+q – detach robi

- docker stop id – zatrzyma obraz z poza obrazu

- mozna tez kill

- oba stopuja rpoces/aplikacje w kontenerze z pid1, ta wewnatrz kontenera

- docker ps –l – ostatnio startowany

- docker start id – uruchomi go , a docker attach id, sie podlaczy

- by ponownie - docker restart

- tak samo attach , robi attach do proces PID1 w kontenerze

PID1 – to specjalny init process , ktory zarzadza innymi procesami

- ale kontener mozna jako mongo, redis etc., nie zarzadza innymi procesami, ale mamy 1 w kontenerze

- choc mozemy kilka procesow w kontenerze, troszke bardziej skomplikowane

- one concern per container (problem)

Stopowane kontenery moga byc restartowane, ich stan trzymany w lokalnej pamieci

- mimo, ze docker ps mowi, ze 1 proces chodzi to

- docker info , mowi ze mamy 25 images i 15 kontenerow

-  !!!!!!!!!

- docker rm id – usunie kontener, jak running, to rzyczy error , najpierw stop, -f to force

- docker rmi id

- mozna aliasy na najpopularniejsze np.

- alias dps=”docker ps” I teraz wystarczy,ze dam dps ☺

Z hosta mozna jakie procesy dzialaja w kontenerze

- docker top id – pokazuje procesy w danym kontenerze

- z zewnatrz id procesow maja wysokie numery, ale jak attach do kontenera to juz normalne

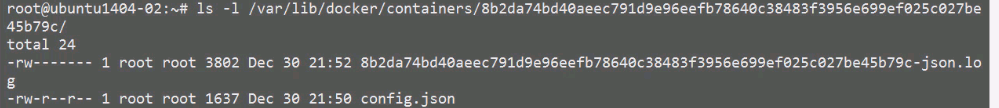
- jak w kontenerze wywolam ps –ef – wypisze procesy, potem wyjde z niego i docker logs – to mi pokaze to co na konsoli w kontenerze – czyli ps –ef

- dziala jak tail

Docker info id

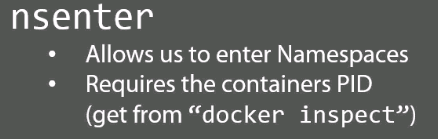
- daje duzo informacji o procesach, net ,

Duzo z tego jest w config.json pod kontenerem



Docker attach polancza do PID1 kontenera, ale zadko tam jest shell, tylko jakas apka

Pomocny nsenter



- potrzebuje pid z kontenera

- na host go pobierzemy za pomoca docker inspect

- docker inspec id | grep Pid – daje pid procesu kontenera

 - mount, uts, network, process, ipc namespaces, -t to target pid

Koncowo :  - czyli lanczymy sie do kontenera, ale zamiast attach to do konsoli I mozemy wykonywac zapytania

- nie jestesmy w pid1, wiec robiac exit, wrocimy do hosta, ale kontener nadal chodzi

- drugie to docker-enter id – w nim exit tez wraca

- 3ci to wywolanie czegos na docker – docker exec –it id /bin/bash – laczy do shell, mozna exit powrot do hosta

- moze nie byc nsenter na kontenerze, instalowac mozna



Budowanie obrazu z docker file, jest elastyczne i ma duzo mozliwosci

Dockerfile – wazna pisownia

- plik textowy

- prosty format, latwo czytany

- instrukcje by zbudowac image

- wazne gdzie jest, dostep do plikow

- jak mamy to docker build – buduje obraz na instrukcjach z pliku

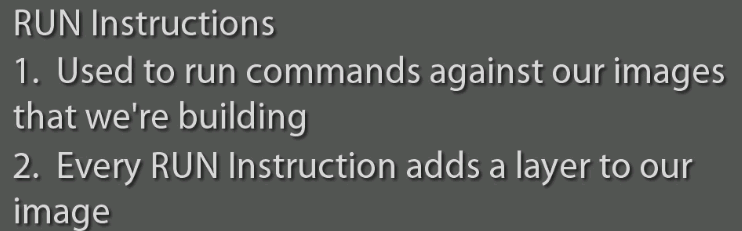
Hash to komentaz w pkiku #

- pierwsza instrukcja FROM – jaki obraz, mozna z wersja, mozna bez-ostatnia

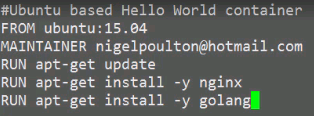
- dobra praktyka to instrukcje wielką litera, mozna podac kto utrzymuje

- format to instrukcja (z duzej) i wartosc z malej

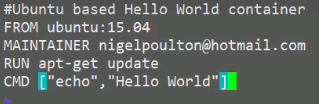
- instrukcje RUN :



- kazdy run odpala, wykonuje robote, robi commit i stop kontener, kolejny run to nowa warstwa, otwiera kontener z poprzedniego stanu



- na koniec command CMD, robi wypis, albo komenda tu echo



Jak budujemy z Dockerfile, to wszystkie pliki i katalogi w tej samej lokalizacji zostana zawarte w build

- czyli lepiej nie na root Dockerfile

- teraz budujemy obraz, -t jako tag (nazwa), opcjonalnie wersja, potem . – jako ze w tym samym katalogu



- buduje kontext do socker deamon – czyli nie klient, moze przez siec

- potem poszczegolne kroki z pliku – stepy

- na koniec wypluwa z danym id

- w miedzyczasie przy run/cmd odpala kontenery, ale je zaraz usuwa, widac w logach

- w docker images –tree – pokaze, ze warstwy z pliku powiazane, i stoi na ubuntu

- jak damy docker history id – wypisze polecenia wywolane na tym image

- teraz mozemy odpalic docker run helloworld:0.1 i nam wypisze cmd

- mozna w docker hub zobaczyc przykladowe, np. Pod ubuntu

- dodatkowo ADD moze pakowac cos

- push pull to registry – docker hub , public docker registry

- mozna swoje wlasne prywatne registry

Hub docker to publiczne , default registry, zawiera repozytoria

- np. Jak wpiszemy mongo to jest w wielu repozytoriach usera,

- kazdy image na razie byl pull z docker hub, musimy wiedziec jak push zrobic

- zeby to zrobic musimy miec konto na registry.hub.docker.com

- mozna tu tworzyc konto, albo przez command line – komenda login

- po kliknieciu w rybke – home, po prawej u gory add repository

- pod userem, nazwa , opis, czy public

- jedno private repo na usera za free

- jak docker images, to po lewej z jakiego repository

- przed push musimy otagowac, by nasz wlasny image mial odpowiednie repo –

, przed push potrzebny user/repo gdzie ma trafic

- mozna to od razy przy budowaniu docker build –t

- poniewaz tag=repository to mozna push



- trzeba przy tym zalogowac sie

- sprawdza wszystkie warstwy, nie pushuje tych co sa, tylko nowe

- moge usunac kilka kontenerow na raz



- usuwam obrazow kilka , razem z warstwami



- teraz pobierm moj



Pobiera wszystkie warstwy

- ta sama nazwa wersja i id na wielu postach

- latwo pobrac z dowolnego hosta teraz

- nie mozna repo z command line, trzeba z gui

Private registry, ktore ma kilka repo, mozna na dokerze , nie koniecznie na linux-host



-p to port wew slucha i na zew tez, a registry to python registry, czyli nazwa image ktory pobierze

- mozemy sie do tego teraz z przegladarki podlaczyc, hostname:5000

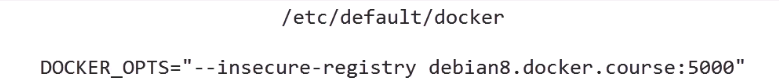
- jak na innym host, to trzeba bedzie zmienic link

Zeby oplublikowac musze najpierw otagowac obraz ktory chce pushowac, tag ma host:port/nazwa



- teraz moege push zrobic

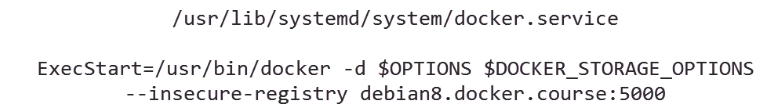




- sciagam na innym i install



Po stronie cenrtos tez insecure option



Docker hub enterprise

- docker hub w korporacji

- obrazy jao program, wszedzie odpalane pod kazdym systemem

- jak google store or play in enterprise IT

- jak drugi raz robimy build z tego samego dockerfile ale z innym tagiem –t, to jest duzo szybciej, dziala cache – build cache i w logach widzimy – using cache

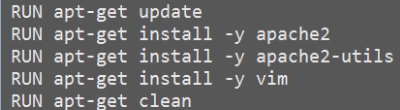
- przy kolejnym wywolaniu sprawdzenie w build cache czy dana instrukcja istnieje juz, jak tak to ja z cache pobiera

- docker info – 5 images

- docker image –tree – to 5 warstw

- kazda instrukcja maintainer, run I cmd to warstwa (widac w docker history id)

- dobra praktyka to clean po wszystkich run na apt-get



- expose wystawia port na zewnatrz

- startujemy serwer za pomoca CMD



Ale i tak po zbudowaniu



Odpalam z portem, bo mapuje port hosta na port dockera :

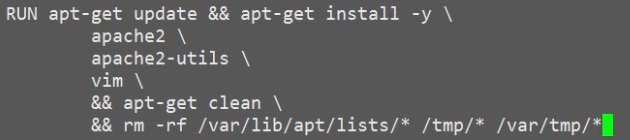


- teraz po wejsciu w przegadarce na url hosta pokazuje mi apache z dockera

- jak wszystkie run po kolei co w liniach polacze w jedna linie za pomoca &&

(dodoatkowo moge formatowac z nowej linii za pomoca \)

- w jednej instrukcji wszystkie apt-get i czyszczenie prawgopodobnych temp plikow



- dzieki temu ten caly RUN jako jedna warstwa, wiec mam mniej warstw, lzejsze

- tez docker images pokaze, ze obraz calosciowo jest mniejszy

CMD jest runtime, gdy RUN jest build-time, do instalacji

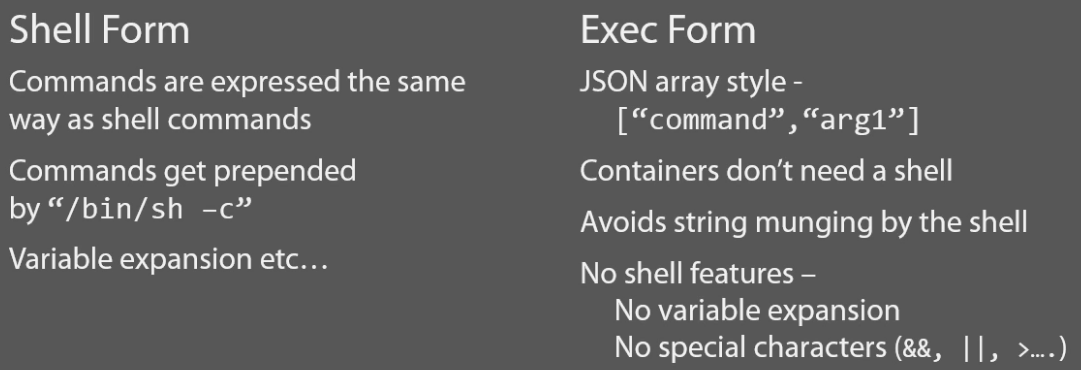


- jak startujemy docker z komenda na runtime (/bin/bash), to ona nadpisze nasz CMD

- tylko 1 CMD na Dockerfile

- mozna wiecej, ale tylko ostatnia da efekt

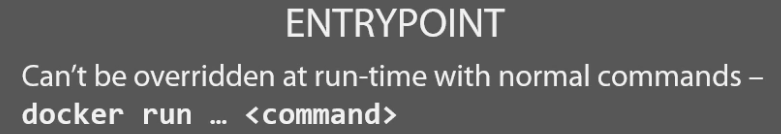
- mamy 2 mozliwosci wywolania CMD, shell dodatkowo znaki specjalne i zmienne



CMD odpala PID1

ENTRYPOINT

- podobne do CMD, ale nie nadpisywane na runtime



- dodatkowo kazda komenda na runtime jest jako argument do entrypoint

Np.





- normalnie jak ponizsze startowalo bash, to teraz to wejdzie na entrypoint i wypisze /bin/bash



- np jak entrypoint odpala apache, to w run mozemy argumenty podac do niego , albo po nim cmd z argumentami

ENV

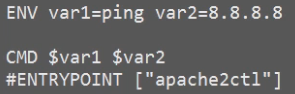
- dodanie environment variable do kontenera

- lepeij w jednej, jak w 2 linie to 2 warstwy



- po odpaleniu docker z bash i wywolaniu env nam je wypisze

- przy takich zmiennych CMD bedzie wywolywal pingi



- docker volume create, ls, rm

- wydziela dane z kontenera i wspoldzieli pomiedzy

- pozwala zapisywac dane poza kontenerem

- jak kontener pada, inne maja dostep do danych

- odpalam okreslajac volume



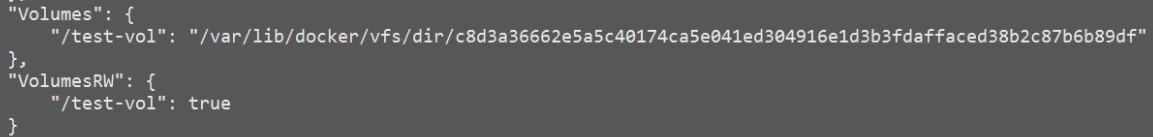
- tworzy test-vol katalog pod dockerem

- dodaje tam plik 

- wychodze do host

- moge sprawdzic ten voluen po nazwie – docker inspect voltainer

- mowi gdzie dane trzyma na host



- teraz inny kontener moze z tego korzystac, przy starcie okresla –volumes-from

- w jego katalogach tez pojawia sie test-vol z dodanym plikiem

- mozna tez udostepnic katalog z hots do kontenera



- -v i pierwsze to jaki katalog z host 2gie pod jakim w kontener dostepny , ale to nie skalowalne, uzaleznia od danych na host, np. Moze byc problem z dev na prod

- wystarczy VOLUME w Dockerfile, i wrzuca dane do podanego na host



- przy wyrzuceniu kontenera volum zostaje, musze podac –v by wyrzucic volume



-----

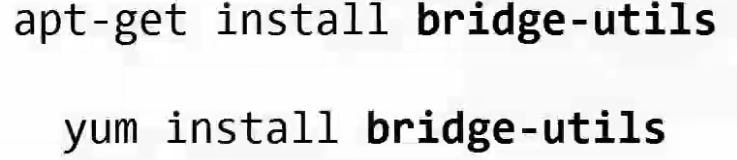
COPY – dodaje artefakty do image

ip –a – pokaze wszystkie polaczenia, olewamy lo, eth0,eth1, wazny docker0

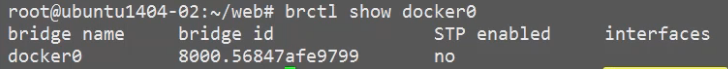
- docker0 interface tworzony gdy startuje docker deamon/engine to wlasciwie bridge/virtual switch , software in linux kernel, jest wymagany do kontener networking

- jak switcher, przezuca pakiet pomiedzy 2 polaczone urzadzenia (normalnie uzadzenia na porty)

- zeby zobaczyc co podlaczane instalujemy bridge-utils



Teraz :

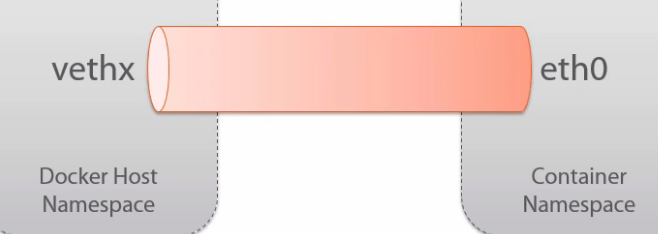


- jak odpalimy kontener jakis pojawia sie interfaces , odpalimy kontener to sie pojawi interface z 1 attached port do brige

- kazdy kontener ma 1 interface automatycznie podlaczany do docker virtual bridge

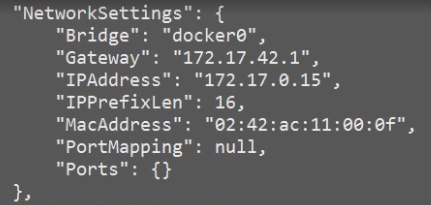


- jak robimy z kontenera zapytania przez siec typu ping to laczy sie z bridge



Docker inspect net2 – po odpaleniu sprawdzam

- w nim widac :



Ze bridge to docker0, a gateway xxx, co widac bylo jak odpalalismy traceroot na kontenerze , te ip kontenera widac

- resolv.conf kazdego kontenera to kopia /etc/resolv.conf hosta

Na kontenarze znajdziemy w



Mozemy nadpisac dns default z hosta w run



- w docker inspect dnstest widac,ze zmienione

- czyli kazdy kontener moze miec swoj resolv.conf i hostfile nadpisujacy ten z image

Exponowanie network ports z kontenera na zewnatrz

- jeden ze sposobow to EXPOSE z Dockfile i wystawienie przy run



Zeby to w host wystawic przy run –p option



Mapujemy 5001 z host na 80 w kontenerze

- docker port web1 – pokaze mapowanie portow na kontenerze web1

- normalne mapowanie tcp, ale mozna udp





- jak na host ip –f inet –a – wypisze nap dostepne

Chce sie podlaczyc tylko z kontenera na eth1, to biore jego ip i w run je podaje



--

Jak w Dockfile ESPOSE kilka portow, zbuduje jako throw-away

- i uruchomi z –P – wszystkie wystawia



To jak podejrze wypisze wszystkie mapowania



Linking containers

- bespieczniejsze niz exposing ports

- dziala pomiedzy kontenerami nie z zewnetrznym swiatem

- source EXPOSE port , ale nie na runtime, czyli nie widac na zewnarz

- reciever sie poadlacza i source mu przesyla dane



- doker linkuje kontenery po nazwie

- po uruchomieniu source



- i docker ps, pokazuje, ze port jest zdefiniowany w kontenerze, ale nie nasluchuje na nim, nie jest exponowany przez host

- receiver na starcie sie laczy do surce

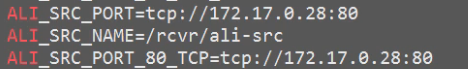


- wymagane sa oba – nazwa source i alias, moga byc takie same

- przy docker inspect rcvr widzimy ze link jest

- po podlanczeniu do rcvr

Przy env | grep ALI widac

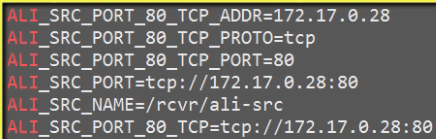


Przy

Cat /etc/hosts

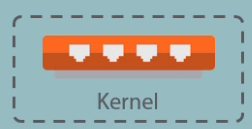


Gyd receiver kontener ma appscript ktory wstaje po wstaniu kontenera, moze uzywac tych wartosci i odwolywac sie do wystawionego serwisu

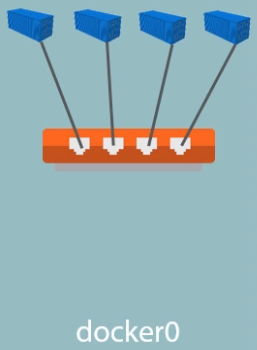


- mozna linkowac wiele do wielu, wiele rcv do jednego source i 1 rcv do wielu scr

Bridge tworzone przez docker demon na starcie



Podlaczone kontenery do niego, moga sie ze soba komunikowac lub z zewnetrznym swiatem



Docker deamon logging



Service docker stop – stopuje docker

- docker –d –l debug & - odpala deamon z poziomem logow debug, amp na koncu – terminal back

- moge tez ustawic w /etc/default/docker



- ps –ef | grep docker – pokazuje pid i loging level ustawiony na starcie

- kill pid

- service docker start – teraz po ps pokaze ze logging ten z default

Logi z PID1

- wszystko co na standard out i standard error zapisywane w filesystem od docker host

Wypiszemy z host : docker logs nazwa\_kontenera

Flaga –f – to dorzuca dynamicznie

Dzieki volume mozemy udostepnic logi innym i je zapisywac , do zapisu danych

Zamiast od razu z Dockerfile mozna instrukcje po kolei

From



Run-Update, run- apt-get ping, cmd – ping

Po test wrzucenie do dockerfile I refactor by run na raz

Jak podamy np. Zly update, to leci bald przy tworzeniu image z dockera

- jak podam docker images – to widac, ze powstal image <none>, ten nie tagowany image bez repo to ostatni zakonczony z sukcesem z Dockerfile – widac z logu budowania po id

- mozemy z niego bledy sprawdzac, wiemy do kiedy dobrze bylo



Przy starcie docker0 okresla adres, ale jak duza siec, to moze sobie chcec zakres adresow, co juz zajete, mozemy je okreslic

Zakres ustawiam w 

Zakres adresow 

Najpierw stopuje, potem staruje ze zmiana,

Service docker status czy dziala

W ip –a widzimy nowe ustawienia

- nowe kontenery dostaja teraz ip z tego range

Normainie wszystkie komponenty na tym samym host moga sie komunikowac ze soba

Reguluja to

- icc – inter component communication

- iptables

Oba true default

Jak to nadpiszemy, to wstawi docker droprule to iptables config

Jak wpisze : iptables –L –v – to pokaz\uje 3 accept rules

Ustawiam

, zastopuje komunikacje pomiedzy

Teraz po iptables –L –v widac drop rules

Przy ustawieniu ponizej tez nie bedzie polaczenia, bo nie moze zmienic iptables



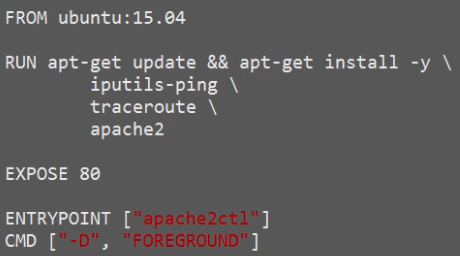
Czyli kontenery sie nie moga komunikowac, mimo icc=true, kontenery nie moga sie pingowac

- Docker version

- docker info

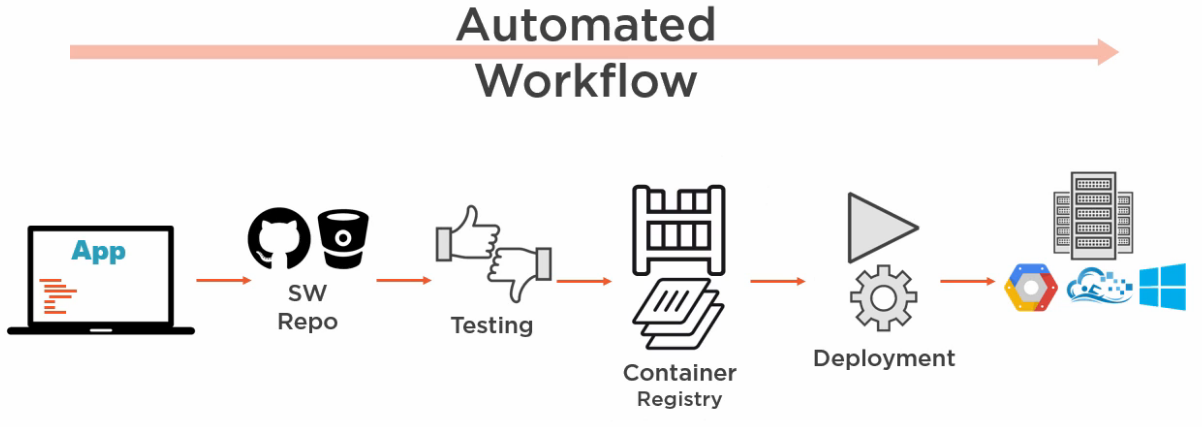
Ile kontenerow i images (warstw)

driver to aufs, root dir

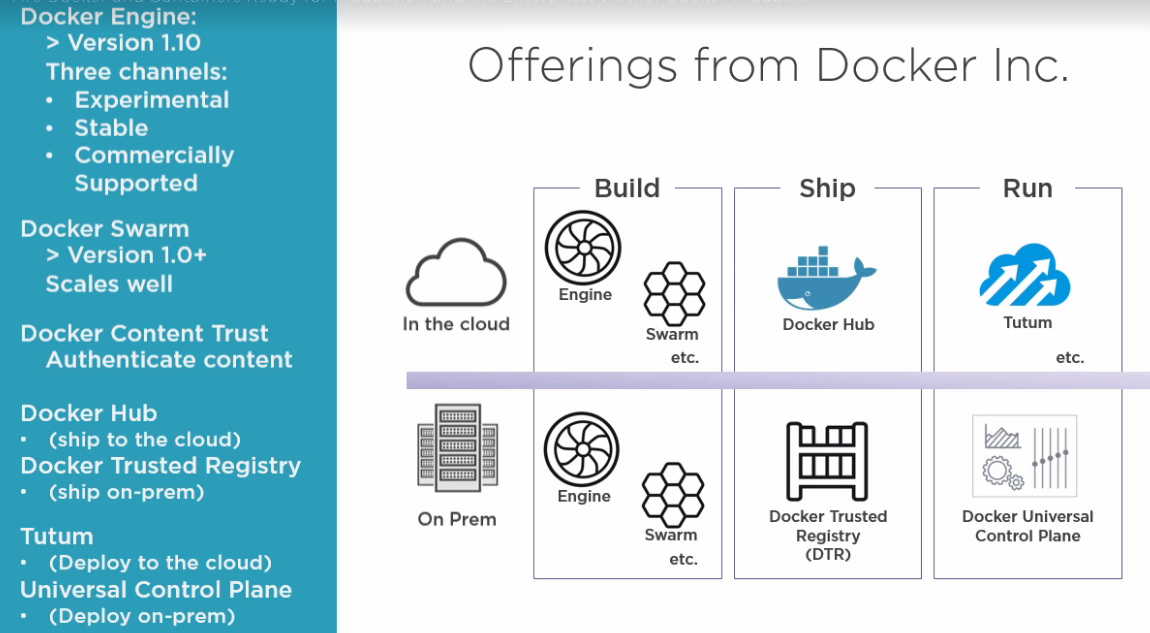
* Docker images, jakie obrazy, z –tree – pokazuje warstwy
* Docker history – jakie zapytania chodzily
* Docker pull nazwa, do zaciagania
* Docker ps – obecnie dzialajace, z ich PID1, jaki port exponowany poikaze
* Docker ps –a – jakie byly odpalane kiedykolwiek
* Odpalamy i zamykamy docker stop, docker start
* Przyklad Dockerfile, uwaga na warstwy, mniej to lepiej
* 
* Budujemy 

BIG PICTURE

Automatyzacja przez wrzucanie obrazow do registry

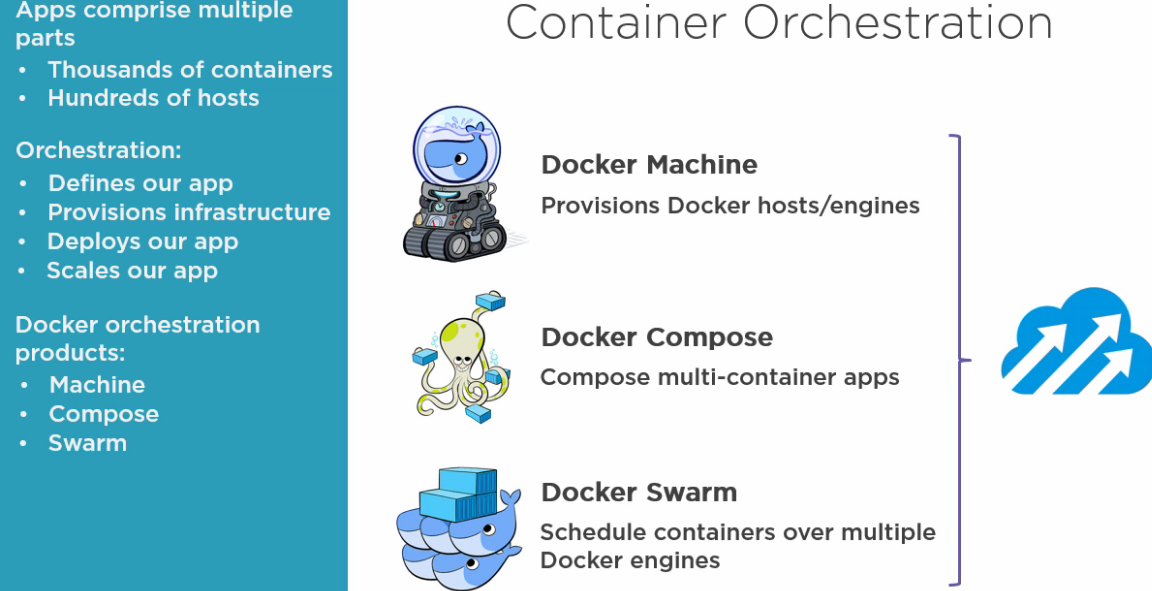


Duzo narzedzi



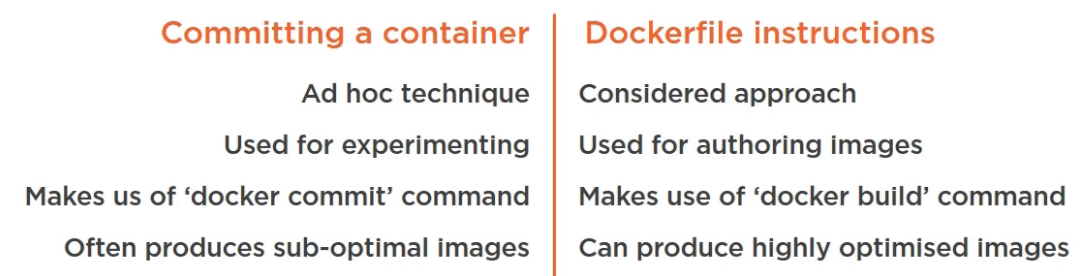
Swarm- robi cluster z kontenerow

- orchestration, to polaczenia pomiedzy komponentami, majacymi rozne odpowiedzialnosci, httpe, aut, search, mysql, logs, ...



APPLICATIONS WITH DOCKER

-2 sposoby tworzenia obrazow, Dockerfile i commit



OPENSHIFT

- zaarzadza komponentafi dockerowymi, monitoruje, wybudza jak padnie,

- moze je jako 1 serwis wystawiac, user na 1 adres a miedzy nimi load balancing

- mozna ustawic adres to taki serwis:port

- mozna nawet takie, ze na bitbucket , wchodzi nowa wersja buduje obrazy i podmienia ze starymi – zapewnia nam ciaglasc

----------

<https://www.javacodegeeks.com/2017/05/deploying-docker-images-openshift.html>

- dziala na kubernetis, rozszerzenie

- mamy obrazy dokerowe i je wrzucamy na openshift, mamy swoje w chmurze i mamy swoje „repo” – openshift cluster

- po zalogowaniu tworzymy tam nowy projekt, jak kubernetis namespace

- teraz tworzymy nowa aplikacje bazujac na obrazie

oc new-app lordofthejars/crimes:1.0 --name crimes

* Teraz robimy route by serwis byl dostepny w sieci

oc expose svc crimes --name=crimeswelcome

* You can find the public URL by going to **OpenShift** dashboard, at top of pods definition.

Dodajemy nowa wersje

|  |
| --- |
| oc import-image crimes:1.1 --from=lordofthejars/crimes:1.1 |

Konfigurujemy wewnetrzna OpenShift Docker rejestr, z nowym obrazem dockera do wydania

* Wydajemy nowa wersje, jak blad mozemy rollback, mozna wyrzucic

- to implementacja Kubernetis , providing some features missed in Kubernetes, such as automation of the build process of the containers, health management, dynamic provision storage, or multi-tenancy, to cite a few.

---------------------------------------------

- dokery w repon, np. Github

- na openshift robie proj, i pod niego image jako apka

- image ma manifest – json lub yaml

<https://blog.pavelsklenar.com/10-useful-docker-commands-tip-tricks/>