DOCKER

Docker je nastroj poskytnu jednotne rozhranie pre zjednotenie aplikacii či už linux alebo windows

Kontajner – izolovane služby ako pre predstavu docker je miesto ako lod a kontajner je nejaka služba pre danu vec.

Webova aplikacia – kontajner pre webovy server a tak

Rozdiel medzi VM a docker

-VM zabera vela miesta oproti kontajneru

Obrázok, na ktorom je text, snímka obrazovky, písmo, číslo

Automaticky generovaný popis

-vm trva dost dlho a nastartovanie vm začne sa prehrievat a využiva sa hlavne pri vačšich serveroch a tak

-docker je nalavo – je tam nejaka infraštruktura (HW) mame OS a potom mame docker,kontajner nepotrebuje nejaky samostatny OW ale zdiela operačny system daneho PC

Image – nejaky recept by sa dalo povedat, zaklad toho z čoho ten kontajner vznikne,je to zaklkad napisany v subore a na tom beži container (definicia kontajnera)

-vybuildvany subor – zapisany polda určitych inštrukcii v subore (dockerfile

)

Cez cmd sme spustili

docker run hello-world

(naštartuj kontajner ktory použiva image hello-world)

a to nam ide vphode

docker ps – zoznam bežiačich kontajnerov

tymto vidim všetky aj tie vypnute

C:\Users\riki0>docker ps -a

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

b403148a8492 hello-world "/hello" 17 hours ago Exited (0) 2 minutes ago peaceful\_mendeleev

ak chceme použit naming aby sme vedeli ako sa to bude volat

C:\Users\riki0>docker run --name=hello-world hello-world



Vytvorenie vlastneho image

Obrázok, na ktorom je snímka obrazovky, softvér, multimediálny softvér, grafický softvér

Automaticky generovaný popis

From znamena aky image chceme stiahnut z dockehub

LABEL com.example.version-is-production="labeltest"

Lepiše vyhladanie

RUN – pridavanie nejykch packegov ako napriklad

RUN apt-get install -y curl \

Git \

Stiahnut curl a git to to znamena

RUN apt-get update && apt-get install -y curl \

git \

vždy sa upodatne ta cache s tymi inforamciami natiahne si data a začne to instalovat

musi to byt v ramci jednej vrsty

CMD – spusta nejaky rpogram ktory je v tom image

EXPOSE 80 – port na ktoorm počuva

ENV – nejaka premenna ktoru vieme použivat

COPY – vtom istom priečinku mali byt aj configuračne subory lebo rie aj idu v tom foldri (nakopirovat svoje subory v workdir priečinku ktore si spusitme vo sovjom projekte,nakopirovat config do kontajnera)

ADD- podobne ako copy ale v pripade žeby sme napisali nejaky zazipovany subor a chceli by sme ho nakopirovat do rootu tak ho rozbali a cely ten obsah nakopiruje do rootu(to by copy neurobil)

Entrypoint – spustanie v image,nepočuva na použitelkse argumenty z vonku

VOLUME – niečo ako mkdir(vytvorenie preičinku)

USER – beži to defaultne v roote

Kontajner je jednotka sofweru , ktore ma už image , nejaky vysleodk našej prace

CLI-



Bodka znamena že v tom foldri by sme mali mat nejaky ten Dockerfile

|  |  |  |
| --- | --- | --- |
| --detach , -d |  | Run container in background and print container ID |
| [--publish](https://docs.docker.com/engine/reference/commandline/run/#publish) , [-p](https://docs.docker.com/engine/reference/commandline/run/#publish) |  | Publish a container’s port(s) to the host |

docker run -d -p 127.0.0.1:8080:80 --name test tutum/hello-world

najprv ho hlada lokalne , potom celkovo

-d po zadani nam nelockne terminal

-p port – mapovanie z localneho hosta do kontajnera

1x je vonkajši a druhy je vnutorny

Aplikacia

Obrázok, na ktorom je text, snímka obrazovky, písmo

Automaticky generovaný popis

Dockerfile

FROM php:7.4-apache

COPY /webroot /var/www/html

EXPOSE 80

Y toho webroot kde bol php skopiroval všetko do kontajnera polda ktoreho bol vztvoreny

Z php pre image vidime že toto nastavi defaultne   
Obrázok, na ktorom je text, snímka obrazovky, písmo, dizajn

Automaticky generovaný popis

C:\Users\riki0\hello-world-app>docker build -t hello-world-app .

Buildneme -t je tag a – „.“ Znamena že v ramci toho prešinku najdi Dockerfile

Tymto sa buildne image   


Ale treba ešte spsutit kontajner

C:\Users\riki0\hello-world-app>docker run -d -p 8080:80 --name=web\_stranka\_c hello-world-app  
tu zadame už naš buildnuty image

Obrázok, na ktorom je text, snímka obrazovky, ľudská tvár, chlap

Automaticky generovaný popis

Toto ma nevyhodu v tom že ten php sme zabetonovali do toho imagu, taže ak by som chcel niečo menit tak musim toto cele menit

Volume – može vyriešit problem ohladne tohto issue ktore sme mali

Volume miesto na disku alokovane dockrom ktore ostane aj po zmazani kontajnera

(DB je typicky priklad kde mame data v db a nechcme o ne prist) stale tam ostanu

V mysql

Obrázok, na ktorom je text, snímka obrazovky, písmo, dizajn

Automaticky generovaný popis

Je tento image ktory ma tu alokovane miesto.

C:\Users\riki0\hello-world-app>dcoker run -d --name=mysql -e MYSQL\_ALLOW\_EMPTY\_PASSWORD=true mysql

Takto zadefinujeme premennu ktora bez toho by sa nesputsila

V kontajneri nas zaujima   
Obrázok, na ktorom je text, snímka obrazovky, písmo

Automaticky generovaný popistato cesta je realna cesta k tomu ale je to miesto v tejto virtualke

Taka nasa friendly cesta ako sa k tomu dostat

Z toho konfigu v kontajneri vidime že sa nam to tu uchovava   
Obrázok, na ktorom je text, snímka obrazovky, písmo, rad

Automaticky generovaný popis

**Volume vieme pomenovat iba pri štarte kontajnera !!!!!**

Zmazanie kontajnerov

C:\Users\riki0\hello-world-app>docker rm -f mysql mysql2

mysql

mysql2

kontajner sme vypli ale volume beži  
Obrázok, na ktorom je text, snímka obrazovky, písmo

Automaticky generovaný popistaže o tieto data sme neprišli ak sme tam data vytvorili

ako pomenovat volueme –

C:\Users\riki0\hello-world-app>docker run -d --name=mysql2 -e MYSQL\_ALLOW\_EMPTY\_PASSWORD=true -v mysql-db:var/lib/mysql mysql

Obrázok, na ktorom je snímka obrazovky, text, písmo

Automaticky generovaný popis

Perzistencia dat znamena trvacnost aj po reštarte všetkeho

(uchovanie dat)

Bind mounts

docker run -d --name=hello\_world\_app\_c -p 8080:80 -v %cd%/webroot:/var/www/html hello-world-app

mapujeme to v tom a musiem dat cestu ktorz kopirujeme kde,

 aby som dosiahol prepojenie priečinka na hosťovskom OS, kde sa nachádza zdrojový kód aplikácie do kontajnera s webovým serverom.

**Docker compose** –

Nástroj pre konfiguráciu viac-kontajnerových aplikácií.

Nastroj ulahčuje pracu s docker aplikaciami

Kombinacia konfigu subora a CLI nastroja

Určeny pre multikontajnerove aplikacie

Nastavenie sa zapisuje v YML subore – je sučastou docker mac/windows

Obrázok, na ktorom je text, snímka obrazovky, písmo

Automaticky generovaný popis

Zakladna čast docker compose suboru –

# Ak nie je definovaná verzia, default je 1, odporúča sa aspoň 2

# Verzia závisí od verzie Docker engine

version: "3.8"

# Služby (kontajnery)

services:

service\_1: # sekcia pre nastavenie kontajnera (zároveň DNS názov)

container\_name: # názov kontajnera (nepovinné)

build: # cesta ku Dockerfile pre build

image: # alebo názov hotového image-u (lokálne, Dockerhub)

ports: # mapovanie portov (zvonku do Dockeru) (nepovinné)

volumes: # argument -v alebo aj COPY inštrukcia (nepovinné)

environment: # argument -e, environment premenné pre danú službu (nepovinné)

command: # definovanie príkazov po spustení kontajnera (prepisuje image)(nepovinné)

service\_2: # další kontajner

C:\Users\riki0\hello-world-app>docker-compose -f docker-compose.yml

Takto možeme volat docker compose avšak ak nemame nič

Webova aplikacia

-webovy server , databaza (server,klient)

- Dockerfile – webovyserver,php,potrebne tools,composer

- Docker-compose.yml – webovy server

- Projekt – composer create-project netta/web-project

Dockerfile

stiahnutie potrebnych nastrojov do Linuxu (Debian-u) # sposob zapisu RUN setri pocet vrstiev (velkost image-u) ale pri malej zmene sa prebuilduje cely RUN

verzia sa menila ma

FROM php:8.0-apache

Koli projektu ktory nepodporuje 7.4

Obrázok, na ktorom je text, snímka obrazovky, softvér, multimediálny softvér

Automaticky generovaný popis

Zakladny OS z ktoreho vychadza nas BASE image

Curl , git sa tam potrebujeme pripojit a potom dalšie knižnice ako rozpakovanie suborov a tak

Obrázok, na ktorom je text, rad, písmo, softvér

Automaticky generovaný popis

Pootm ako extension

Pomocou composeru stiahneme projekt

Stiahne composer na urovni OS – ide o verziu 2

Expose je klasika pre port

Obrázok, na ktorom je text, snímka obrazovky, písmo

Automaticky generovaný popis

A on stamat stiahne všetky veci lebo je to v webe a tam je dockerfile ktory ma hovorit čo stiahnut

Potom do kontajnera už viem stiahnut cely projekt ako sme hore hovorili



Obrázok, na ktorom je text, snímka obrazovky, softvér, webová stránka

Automaticky generovaný popis

Dockerfile –

FROM php:8.0-apache

RUN apt-get update && apt-get install -y \

curl \

git \

libzip-dev \

unzip \

freetds-bin \

freetds-dev \

freetds-common

RUN docker-php-ext-install -j$(nproc) \

zip \

pdo \

pdo\_mysql \

&& docker-php-ext-enable \

pdo\_mysql

RUN curl -s https://getcomposer.org/installer | php -- --install-dir=/usr/local/bin --filename=composer --2

EXPOSE 80

Docker-compose

version: "3.8"

services:

web:

container\_name: hello-world-app-c

build: ./web

ports:

- "8080:80"

volumes:

- ../webroot:/var/www/html

database:

container\_name: database

image: mariadb:10.1.37

environment:

- MYSQL\_ROOT\_PASSWORD=root

volumes:

- ./database/data:/var/lib/mysql

adminer:

container\_name: adminer

image: adminer:4.7.2

ports:

- "8081:8080"

depends\_on:

- database

Na poste – „localhost:8081“

Obrázok, na ktorom je text, snímka obrazovky, písmo, číslo

Automaticky generovaný popis

Zatial o sebe aplikacia nevie že ma nejaku DB hladat na sieti

Treba v stiahnut porjkete najst veci a pridat aby DB o sebe vedela teraz je to multikontajner avšak momentalne nevedia spolu komunikovat lebo o sebe nevedia.

**Docker Network**

Skupina kontajnerov, ktoré navzájom komunikujú

Kontajnery mendzi sebou nemôžu komunikovať pomocou lokálnych portov.

Compose automaticky vytvorí defaultnú virtuálnu sieť pre všetky služby.

**Docker @ cluster**

Swarm Mode Docker

Jeden alebo viac výpočtových uzlov

Sú spojené sieťou

Manager Node

Udržiava stav klastra

Prideľuje prostriedky

Udržiava informácie o službách a prideľuje DNS mená.

RAFT algoritmus pre zvolenie "vedúceho" uzla.

Worker Node

Spúšťa a zastavuje kontajnery

Odolnosť služby voči výpadku výpočtového uzla.

Škálovanie - pridávanie uzlov podľa potreby.

Upgrade alebo update výpočtových uzlov bez prerušenia služby.

Jednoduchý orchestrátor pre súkromný klaud.

Obrázok, na ktorom je text, písmo, číslo, diagram

Automaticky generovaný popis

Obrázok, na ktorom je text, snímka obrazovky, písmo, diagram

Automaticky generovaný popis

volumes:

- ../webroot:/var/www/html

Tam sa zapisuju data a je to viac perzistentne

**Bind mounts**

Obrázok, na ktorom je text, snímka obrazovky, diagram, písmo

Automaticky generovaný popis

Synchronizacia dat medzi virtualkou a hostom

Obrázok, na ktorom je text, snímka obrazovky, písmo

Automaticky generovaný popis

Obrázok, na ktorom je text, snímka obrazovky, softvér, operačný systém

Automaticky generovaný popis

Prepinače –

Prepínač -it otvorí konzolu kontajnera a napojí ju na aktuálny štandardný vstup. Prepínač --rm zabezpečí, aby bol kontajner dočasný a po vypnutí kontajnera neostali žiadne zvyšné súbory.

Automatický reštart kontajnera zabezpečíte prepínačom --restart always. Ak Vaša Docker aplikácia havaruje alebo sa skončí, systém ju automaticky reštartuje:

docker run -d --restart always busybox sleep 60

Zistite identifikačné meno bežiaceho kontajnera a pripojte sa naňho:

docker exec -it <ID Kontajnera>

Mapovanie zväzkov robíme pomocou prepínača -v. Uvedieme špeciálny reťazec s mapovaním. Časť pred dvojbodkou je meno pomenovaného zväzku alebo lokálneho adresára, časť za dvojbodkou je meno adresára v kontajneri.

-v pgdata:/var/lib/postgresql/data

Virtuálne siete

Pomocou mapovania portov je síce možné sa z vonkajšieho prostredia pripojiť na bežiaci kontajner, ale nie je to možné medzi kontajnermi navzájom. A nebolo by to ani bezpečné. Kontajnery by mali byť od seba maximálne oddelené, aby nebolo možné ľahko pristupovať k cudzím procesom. Musíme explicitne povedať, ktoré kontajnery budú vidieť ktoré.

docker network create pgnet

priklad

docker run \

-p 8080:80 \

--name pgadmin \

--network pgnet \

-e 'PGADMIN\_DEFAULT\_EMAIL=user@domain.com' \

-e 'PGADMIN\_DEFAULT\_PASSWORD=SuperSecret' \

-d dpage/pgadmin4

docker run \

--name postgres \

--network pgnet \

-e 'POSTGRES\_PASSWORD=mysecretpassword' \

-v pgdata:/var/lib/postgresql/data \

-d postgres:10.4-alpine

Obrázok, na ktorom je text, snímka obrazovky, písmo, rad

Automaticky generovaný popis

**KUBERNETES –**

Klastrový operačný systém

Orchestrátor kontajnerov

Kubernetes nie je Docker

Aplikácia beží v kontajneri ale...

Kubertenes sa stará kde a ako

manažuje viac strojov (virtuálnych alebo fyzických) kde bežia kontajnery

**rozdiel**

docker sa stará o beh procesov

Kubernetes sa stará o celý cluster

**Na čo je Kubernetes**

Nástroj na manažment klastra, pridelenie a nepretržité využitie zdrojov:

diskový priestor

CPU

GPU

sieťové spojenie

RAM

Vaša aplikácia beží kontajneri na Vašom klastri alebo na cloude.

Aby sme mohli jendoducho horizontálne škálovať

**Odolnost kubernetesu**

Ak havaruje aplikácia , reštartuje sa (možno na inom uzle).

ak havaruje celý uzol, záťaž preberie iný uzol.

**Scalability – škalovatelnost**

Horizontálna škálovateľnosť - pridáme nové (virtuálne) zariadenie.

Rozširime servre (viac RAM, viac CPU, vymena servra)

Vertikálna škálovatenosť - rozšírime existujúce zariadenie (pridáme disk, procesor alebo pamäť).

(kupime albo prenajmene ešte jeden server)

KOMPONENTY – Kubernetes clustra

api server

Má svoje URL pomocou ktorého vieme riadiť klaster.

Etcd

distribuovaná databáza: (log,stav objektu,konfiguracia)

scheduler

rozhoduje kde sa bude spúšťať zátaž

controller manager

zabezpečuje súlad stavu klastra s konfiguráciou

spúšťa alebo ruší záťaž podľa potreby

kubelet

stará sa o beh záťaže v kontajneri na pracovnom uzle

spúšťa a ruší procesy (kontajnery)

kubeproxy

sprostredkuje spojenie medzi uzlami

container run-time

Kubernetes na cloude –

Amazon Elastic Kubernetes Service EKS

Amazon Elastic Kubernetes Service (Amazon EKS) je spravovaná služba Kubernetes na prevádzku Kubernetes v cloude AWS a lokálnych dátových centrách. V cloude Amazon EKS automaticky riadi dostupnosť a škálovateľnosť uzlov riadiacej roviny Kubernetes zodpovedných za plánovanie kontajnerov, správu dostupnosti aplikácií, ukladanie údajov klastra a ďalšie kľúčové úlohy

Microsoft Azure Kubernetes Service AKS

Typi uložisk

Volumes

PersistentVolumes

Trvalý zväzok (PV) je úložný prostriedok vytvorený a spravovaný rozhraním Kubernetes API, ktorý môže existovať aj po skončení životnosti jednotlivého podu.

Deklarujeme zväzky k dispozícii

"Obalí" konkrétnu zložku alebo blokové zariadenie

PersistentVolumeClaims

Žiadosť PersistentVolumeClaim požaduje úložisko konkrétnej triedy StorageClass, režimu prístupu a veľkosti. Server Kubernetes API môže dynamicky poskytnúť základný prostriedok úložiska Azure, ak žiadny existujúci prostriedok nemôže splniť požiadavku na základe definovanej triedy úložiska StorageClass.

deklarujeme požiadavku na nejaký PersistentVolume

definuje väzbu medzi aplikáciou a úložiskom

Storage Classes

Rozne druhy uložisko ako napriklad premium a standard

PRACA s kubernetes –

S klastrom budete komunikovať pomocou príkazu kubectl

POD

Základnou jednotkou klaudovej aplikácie v systéme Kubernetes je objekt typu Pod. Jeden Pod je tvorený viacerými kontajnermi a zväzkami. Je zaručené, že jeden Pod bude bežať na maximálne jednom uzle. Vďaka tomu môžu jeho kontajnery navzájom ľahko komunikovať pomocou lokálnych portov a zdieľaných zväzkov.

Pozriem si konkrétny druh objektov v klastri:

kubectl get pods

* ClusterIP: dostupná v rámci klastra.
* NodePort: dostupná z vonku na každom uzle.
* LoadBalancer: dostupná pomocou nástrojov poskytovateľa.

Mapovanie medzi trvalým zväzkom a Podom zabezpečuje objekt typu PersistentVolumeClaim. Tento objekt vyjadruje konkrétnu požiadavku aplikácie na úložný priestor. Ak ku nemu existuje PersistentVolume ktorý vyhovuje požiadavke, tak sa vytvorí mapovanie.

Každý Kubernetes objekt má definované tieto atribúty:

* apiVersion: Verzia API ktorú objekt využíva
* kind: Druh objektu
* meta: Informácie o objekte
  + name: Meno objektu - to čo sa nám zobrazí v kubectl get
  + labels: zoznam štítkov objektu - identita pre iné Kubernetes objekty
* spec: špecifikácia objektu

Je možné, že na jednom klastri beží viac aplikácií naraz. Objekty ktoré sme vytvorili doteraz sa umiestnili do východiskového menného priestoru default.Po čase sa môže stať, že vo východiskovom mennom priestore sa nachádza príliš veľa objektov. Pokiaľ nepovieme inak, tak sa všetky príkazy kubectl týkajú východiskového menného priestoru.

Vytvorte si textový súbor nginx-deployment.yaml a zapíšte do neho:

# verziu API zistíte z dokumentácie

apiVersion: apps/v1

# Druh objektu

kind: Deployment

# O objekte

metadata:

# Meno objektu

name: nginxdep

# špecifikácia objektu

spec:

# Počet podov ktorý sa má vytvoriť

replicas: 1

# Selektor vytvára prepojenie Deploymentu a Podu

# Vyberá tie PODy ktoré majú štítok nginx

selector:

matchLabels:

app: nginx

# Šablóna PODu

template:

metadata:

# štítok PODu - na spojenie Deploymentu a Podu

labels:

app: nginx

spec:

# kontajnery PODu

containers:

# Len jeden kontajner s aplikáciou nginx

- name: nginx

# Meno a verzia obrazu

image: nginx:1.14.2

ports:

# POD má otvorený port 80

- containerPort: 80

# Ak je to potrebné, tu uvedieme ďalšiu

# konfiguráciu kontajnera, napr. mapovanie zväzkov

# alebo premenné prostredia

K deploymentu je potrebné vytvoriť aj službu aby porty kontajnerov boli prístupne pre iné kontajnery alebo pre používateľov.

Vytvorte si súbor nginx-service.yaml a zapíšte do neho:

# verzia API z dokumentácie

apiVersion: v1

# Druh objektu

kind: Service

# informácie o objekte

metadata:

# Meno služby

name: nginxservice

# Špecifikácia služby

spec:

# Ktoré pody vykonávajú službu

selector:

app: nginx

# Porty služby

ports:

- protocol: TCP

# Port služby

port: 8800

# Port na Pode

targetPort: 80

Kubernetes sme vytvarali pomocou vytvorenia AKS clustra