TSS Demo:

* Konzolovka, ktora zapisuje do konzoly a tiez do suboru. Delay v cykle je 1s, bez throw exception
* Definicia dockera je zapisana v subore DockerFile bez pripony. Tento subor by mal byt takto pomenovany, inaksie jeho nazov musim v kazdom commande vyplnat.

**DockerFile:**

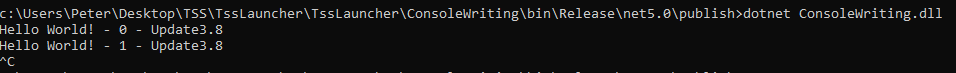
FROM mcr.microsoft.com/dotnet/aspnet:5.0

COPY bin/Release/net5.0/publish App/

WORKDIR /App

ENTRYPOINT ["dotnet", "ConsoleWriting.dll"]

* From: definicia docker imagu, ktory sa ma pre docker pouzit
* Copy: skopirovanie publishnutej konzolovky do adresara „APP” noveho docker imagu. DockerFile je pre tento priklad umiestneny v priecinku rovnakom ako ConsoleWriting.csproj
* Workdir: nasetovanie working directory na dockeri
* Entrypoint: spustenie commandu dotnet s argumentom ConsoleWriting.dll, v cmd mimo dockera by to bolo nasledovne:



**Docker commands:**

* Pre nasledujuce commandy je potrebne ist do priecinku, kde sa DockerFile nachadza, v mojom pripade priecinok s ConsoleWriting.csproj

dotnet publish -c Release

* + - publishnutie ConsoleWriting app do priecinku bin/Release/net5.0/publish

docker build -t writing-docker-image -f Dockerfile .

* + - vybuildovanie noveho docker image na zaklade DockerFile s nazvom writing-docker-image

docker images

* + - zoznam stiahnutych alebo lokalne vybuildovanych docker images

docker create --name core-writing-console writing-docker-image

* + - vytvorenie docker containera s nazvom core-writing-console z image writing-docker-image

docker container ps -all

docker ps -a

* + - zoznam vsetkych containerov (vytvorenych, beziacich, padnutych...)

docker start core-writing-console

* + - nastartovanie dockera s nazvom core-writing-console v stave “Detached”. Znamena to, ze docker sa nastartoval a bezi na pozadi, jeho konzolu nevidim.

docker attach --sig-proxy=false core-writing-console

* + - pripojenie sa na beziaci docker container pre citanie jeho konzoly

docker stop core-writing-console

* + - zastavenie dockera s nazvom core-writing-console

docker rm core-writing-console

* + - vymazanie docker container s nazvom core-writing-console

docker image rm writing-docker-image

* + - vymazanie docker imagu

**Update konzolovky:**

* Pokial konzolovku updatnem, je ju potrebne publishnut a pokracovat rovnako – build image, create container...

**Pripojenie sa na beziaci container pre vykonavanie prikazov:**

* nezabudnut urobit publish/docker build/ docker create/docker start

docker exec core-writing-console dir

* + vykonanie prikazu ‘dir’ na beziacom containery s nazvom core-writing-console

docker exec -it core-writing-console bash

* + pripojenie sa na bash beziaceho containera

apt-get update

apt-get install vim

vim WriteLines.txt

* + Precitanie suboru, ktory vytvorila konzolovka cez ‘vim’

**Docker volumes:**

* Preferovany sposob pre ukladanie dat dockerom a zdielanie dat medzi dockermi. Detaily som neriesil, kedze pre demo to nie je dolezite

docker volume create sharing-folder

docker volume inspect sharing-folder

docker run -v sharing-folder:/etc/shared --name core-writing-console writing-docker-image

* + vytvori aj nastartuje core-writing-console docker container s definovanym volume

**Docker volume – zdielanie windows priecinku**

<https://token2shell.com/howto/docker/sharing-windows-folders-with-containers/>

* TODO: https://docs.docker.com/docker-for-windows/

**Docker image registry:**

* Storovanie a distribucia docker images, e.g. Docker hub, Azure, Gitlab

docker login registry.gitlab.com

* + prihlasenie sa do gitlabu

dotnet publish -c Release

docker build -t registry.gitlab.com/petriceko/dockerdemo:Version3 .

docker push registry.gitlab.com/petriceko/dockerdemo:Version3

* + vybuildovanie docker imagu, ktory uploadnem do gitlabu

https://gitlab.com/petriceko/DockerDemo/container\_registry/1900894

* + Prehlad docker images na gitlabe

docker pull registry.gitlab.com/petriceko/dockerdemo:Version3

* + stiahnutie docker imagu, pokial by som ho uz lokalne stiahnuty nemal

docker create --name core-writing-console registry.gitlab.com/petriceko/dockerdemo:Version3

docker start core-writing-console

docker attach --sig-proxy=false core-writing-console

* + overenie, ci naozaj stiahnuty image funguje

**Docker compose:**

* Sposob ako vybuildovat celu docker infrastrukturu (definicia dockerov, siete, volumes...)
* Zauzivany nazov pre docker composu suboru je: docker-compose.yml
* Pripravene su dva compose subory, jeden pre mysql db a druhy pre autoupdate

Docker compose – MYSQL

version: "3.9"

services:

  mysql:

    image: mysql:5.7

    restart: always

    volumes:

      - todo-mysql-data:/var/lib/mysql

    ports:

      - '3306:3306'

    environment:

      MYSQL\_ROOT\_PASSWORD: pass123

      MYSQL\_DATABASE: mysql\_database\_name

      MYSQL\_USER: clientuser

      MYSQL\_PASSWORD: clientpass

  consolewriting:

    build: ./ConsoleWriting

    restart: always

    depends\_on:

      - mysql

volumes:

  todo-mysql-data:

* Docker compose sa sklada z dvoch docker images: mysql a consolewriting.
* ‘image: mysql:5.7’ - Mysql image sa stiahne s public registry s tagom 5.7 – v tomto pripade je to verzia mysql servera
* ‘restart: always’ – v pripade, ze by nastala chyba a server padne, automaticky sa docker container opat nastartuje
* ‘volumes’ – dolezite definovat z dovodu, ze volume je prave storage pre mysql data. Cize ak by mysql container padol, opat sa nastartuje s datami, ktore tam boli pred padnutim. Ak by volume zadefinovane nebolo, po restarte by bol docker cisty bez dat.
* ‘ports’ – znamena to external:internal port mapping. Ak sa teda na docker container pripajam na port 3306, vo vnutri containera pojdem tiez na tento port. Pokial porty takto nedefinujem, ziadne ani nevystavujem a teda vytvorenie pripojenia s definiciou portu nebude mozne.
* ‘environment’ – definicia enfironment variables, pouzite variables su z definicie mysql imagu z dokumentacie
* ‘build’ – v podpriecinku ConsoleWriting sa nachadza DockerFile, ktory sa pouzije pre build docker imagu a jeho nasledne nastartovanie
* ‘depends\_on’: kontrola poradia startovania a stopovania docker containerov
* ‘volumes’: definicia volume

Docker compose prikazy: (potrebujem byt v priecinku, kde sa docker compose file nachadza):

docker-compose build

* + vybuildovanie docker composu

docker-compose up

* + nastartovanie docker composu v attached mode

docker-compose up -d

* + nastartovanie docker composu v detached mode
* Ked sa docker compose nastartuje, konzola v attach mode vypisuje merge konzolu vsetkych docker containerov. Pre overenie, ci mysql storuje data vo volumes, connection string: localhost:3306

docker-compose down

* + zastavenie vsetkych docker containerov. Volumes sa necistia, pokial nedam force delete. Ak dam teraz vo Workbenchi refresh/select, tak nepojde

docker-compose pull consolewriting

docker-compose images

docker-compose rm

docker-compose restart

docker-compose pause

docker-compose unpause

* + Zoznam dalsich dolezitych komandov

docker-compose exec consolewriting bash

* + pripojenie sa na bash beziaceho dockera s nazvom consolewriting

**Demo padajucej konzolovky:**

* upravit konzolovku tak aby throwla exception

dotnet publish -c Release

docker-compose build

docker-compose up

docker-compose down

**Demo autoupdate konzolovky:**

* Pouzit druhy docker compose

docker-compose up

* v novom VS Code ist na ConsoleWriting
* Urobit zmenu v ConsoleWriting

docker login registry.gitlab.com

dotnet publish -c Release

docker build -t registry.gitlab.com/petriceko/dockerdemo:Version3 .

docker push registry.gitlab.com/petriceko/dockerdemo:Version3

* Pozor: pokial docker push vypise pre vsetky layers: Layer already exists, tak je potrebne urobit napriklad rebuild conzolovky, pretoze docker build si “nevsimol”, ze sa appka zmenila a preto push neurobil ziadnu zmenu. Ocakavane je, ze jeden layer musi byt pushnuty, cize hlaska: „Pushed”