ÚAMT
FEI STU

Docker Kontajner Nginx

Ing. Lukáš Beňo, PhD.

Obsah

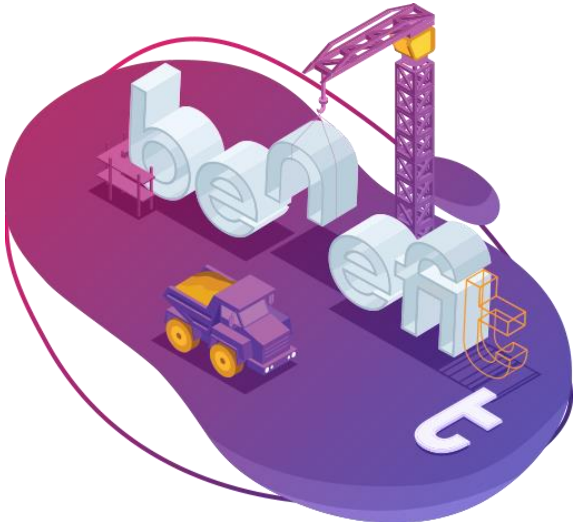
Čo je docker ?

- Open-source platforma, ktorá sa používa na nasadenie a vývoj aplikácií s cieľom využiť **mikroservisovú architektúru**
- Vytvorené ako odpoveď na **odklon** od **monolitického dizajnu aplikácií**
- Nástroj používaný v kombinácii s **Kubernetes** na nasadenie a správu mikroservisových aplikácií

Výhody docker-u

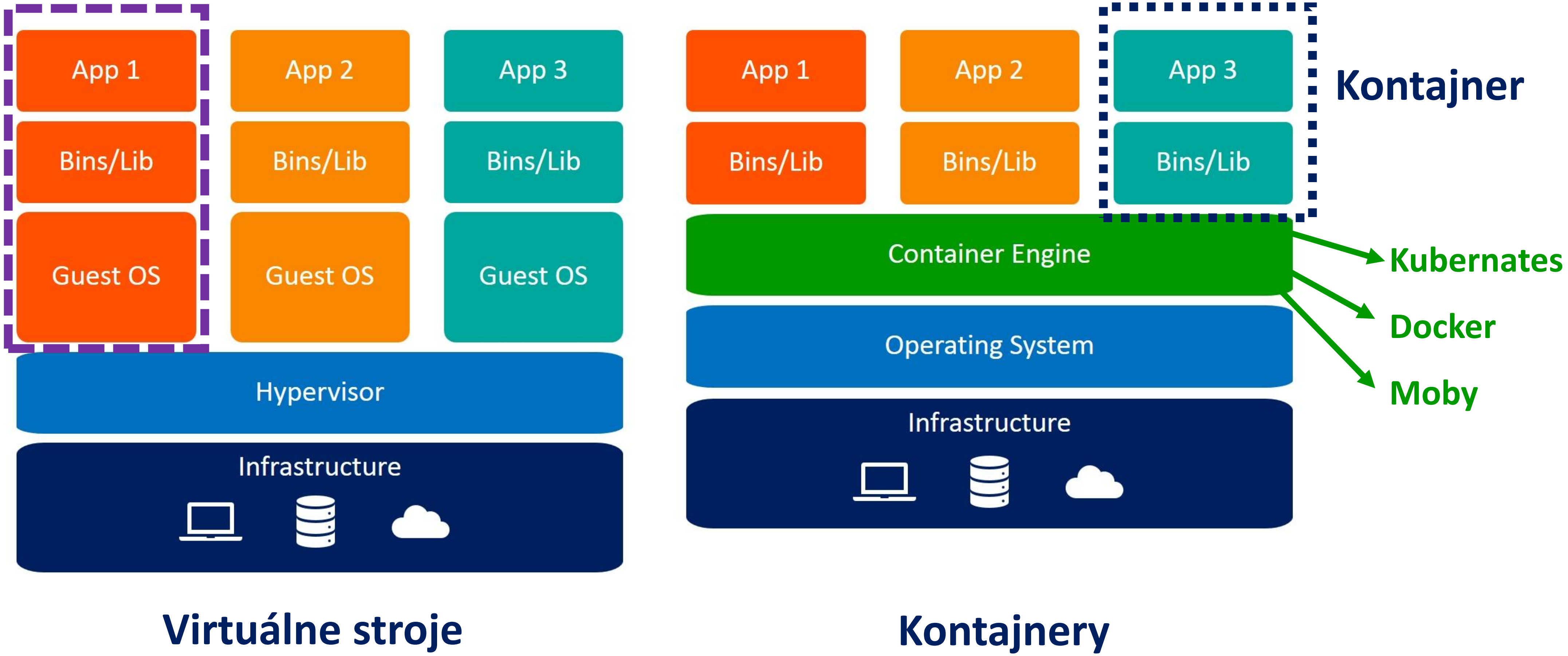
- **Štandardizované** vývojové prostredie - *Development, CSE, QA a Produkcia*
- **Lepší výkon** v porovnaní s klasickými Virtuálnymi Prístrojmi (VM)
- **CI/CD** (Continues Integration/Continues Delivery) integrácia
- Určitý **druh** izolácie od hardvéru
- **Jednoduchší proces** vývoja jednotlivých komponentov vďaka portabilite

a izolácie v kontraste s monolitickou aplikáciou



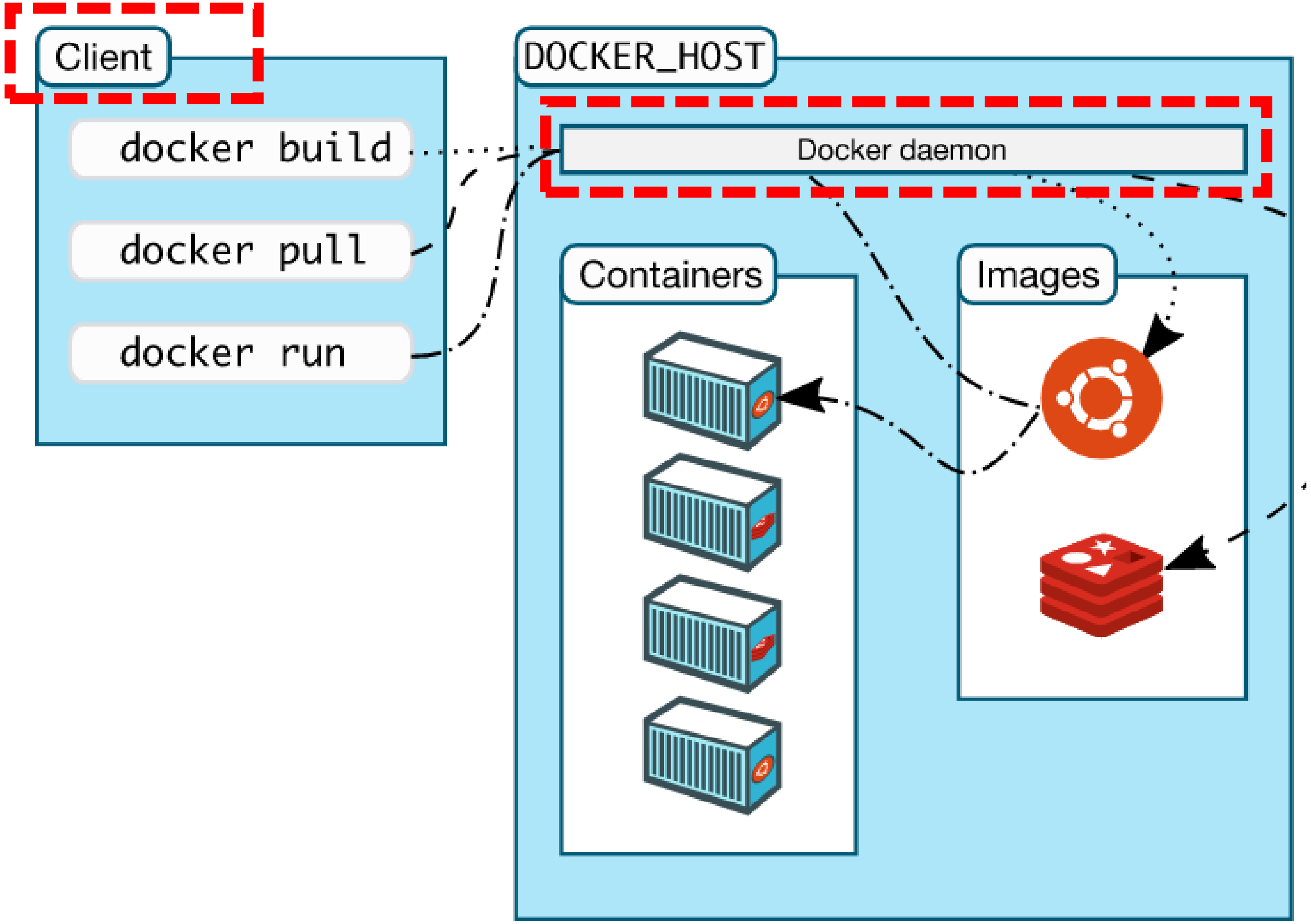
Docker vs Virtuálny stroj

- **Menšia veľkosť**
- **Rýchlejšie nasadenie**



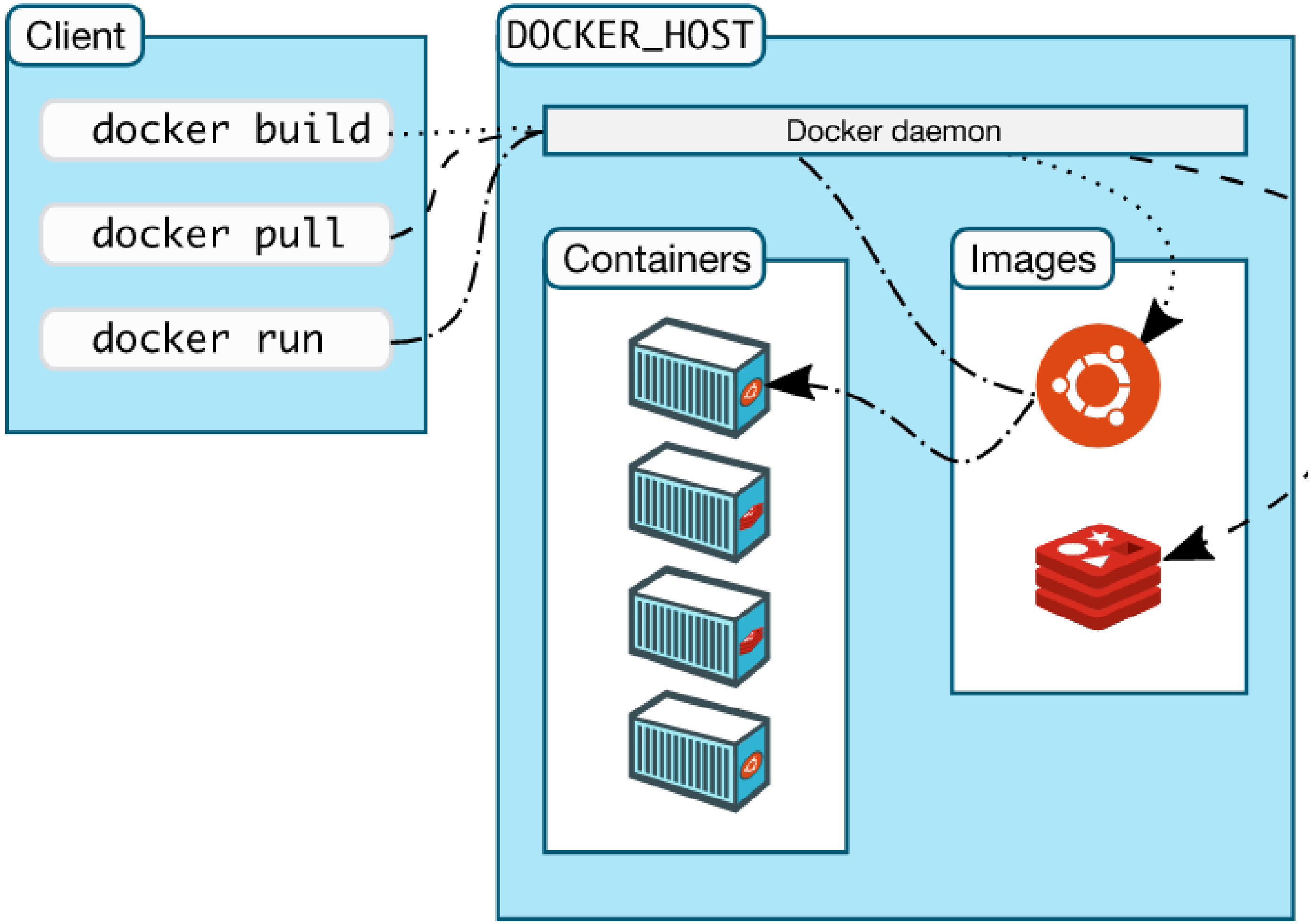
- **Virtualizácia** na úrovni **aplikácie** verzus virtualizácia na úrovni **OS** (hyper-V)
- **Menej bezpečné** ako VM kvôli zdieľanému **jadru hostiteľa**

Docker architektúra



- Architektúra **klient-server**
- **Docker klient** komunikuje s **Docker daemon**
- Docker daemon zabezpečuje **vytvorenie, beh a distribúciu kontajnerov**
- Klient a Docker daemon môžu bežať na **rovnakom systéme**, alebo Docker klienta možno pripojiť k vzdialenému Docker deamon

Docker architektúra



- Komunikácia s docker deamon prebieha prostredníctvom Docker API:
  - **CLI** (terminál), možné príkazy
    - **docker pull**
    - **docker run**
    - **docker push**
  - **Docker desktop** (*Minulé cvičenie*)

Docker - Image

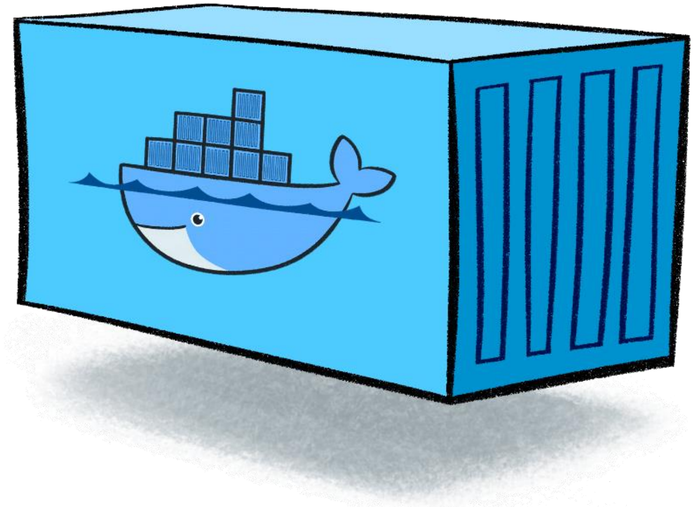
- **Sada inštrukcií (šablóna)** používaná na zostavenie kontajnera
- Obsahuje definíciu potrebných **knižníc, nástrojov, závislosti a podobne pre beh aplikácie**
- Zložený z vrstiev
- Dostupný z **Docker registra**, **lokálneho úložiska Docker Desktop** alebo je zostavený pomocou **nášho vlastného súboru Dockerfile**

Docker - Kontajner

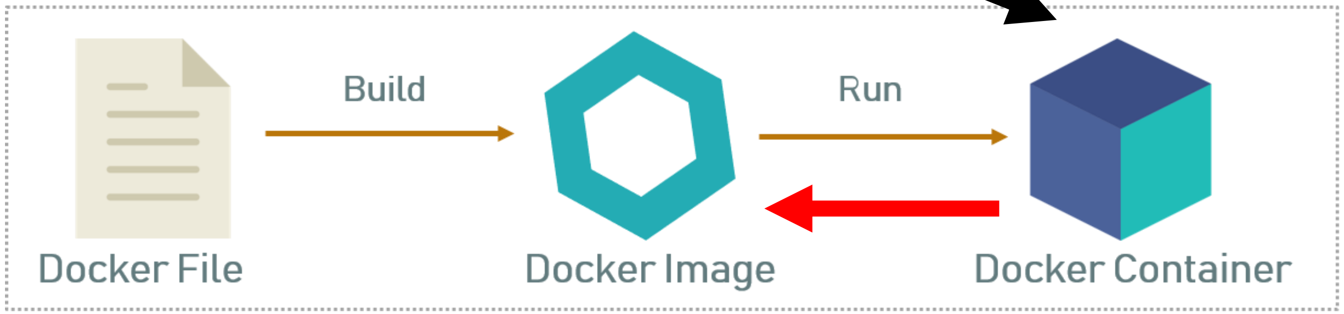
- Kontajner je **spustiteľná inštancia** Docker image
- Kontajner môžeme **vytvoriť, spustiť, zastaviť, presunúť** alebo **odstrániť** pomocou rozhrania Docker API alebo CLI.
- Kontajner môžete **pripojiť** k jednej alebo viacerým sieťam, pripojiť k nemu **úložisko** alebo dokonca vytvoriť nový **Docker Image** na základe jeho aktuálneho stavu (*t.j. z bežiaceho kontajnera*)

Docker - Kontajner

Konfigurácia



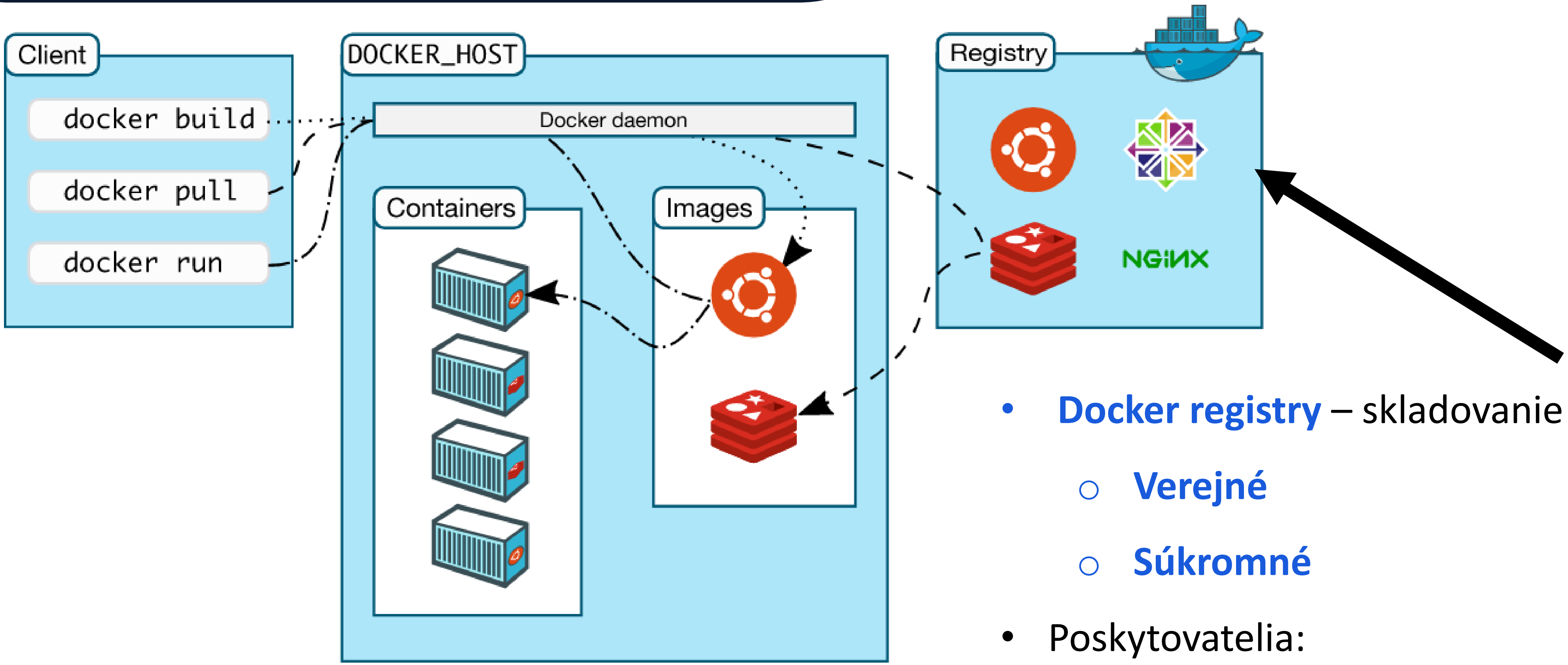
Bežiaca inštancia kontajnera



Docker - Kontajner

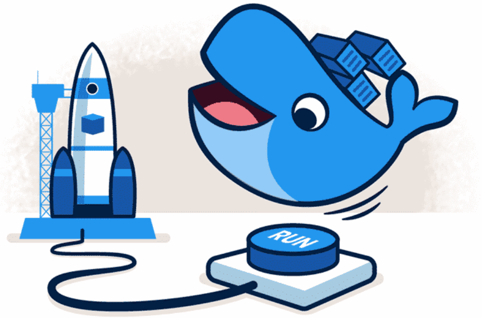
- Vysoko **škálovateľné**
- **Nemenné** – po reštarte pôvodný stav
- **Jednoduchá údržba**
- Prístup cez **sieťové porty** pre **aplikáciu** *(napr. localhost:8080 pre web aplikáciu)*, **príkaz docker exec pre shell** *(vhodné pre debugovanie ale testovanie nových vecí vo vnútri kontajnera bez nutnosti znovu vytvorenia)*

Docker registry



- **Docker registry** – skladovanie Docker images
    - **Verejné**
    - **Súkromné**
- Poskytovatelia:
    - **Docker Hub**
    - **Azure Container Registry**
    - **Amazon Elastic Container Registry**

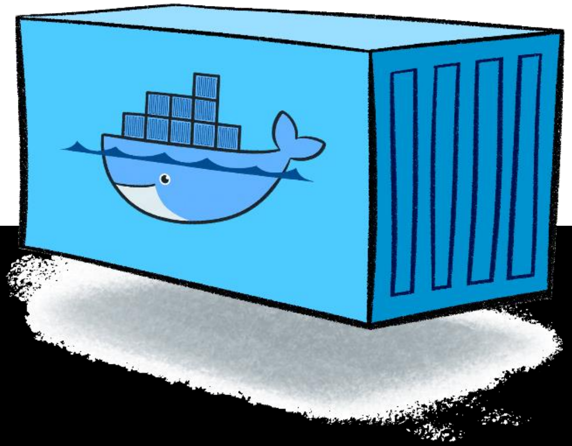
Vytvorenie kontajnera



Docker file

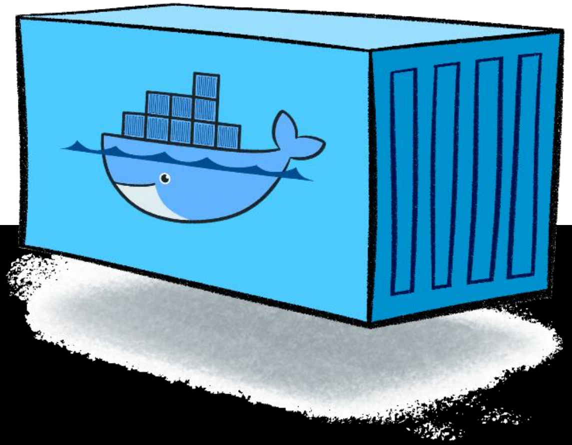
- Dockerfile je **textový dokument**, ktorý obsahuje všetky príkazy, ktoré slúžia na **zostavenie docker image**

Vzorový Docker file pre .NET Core



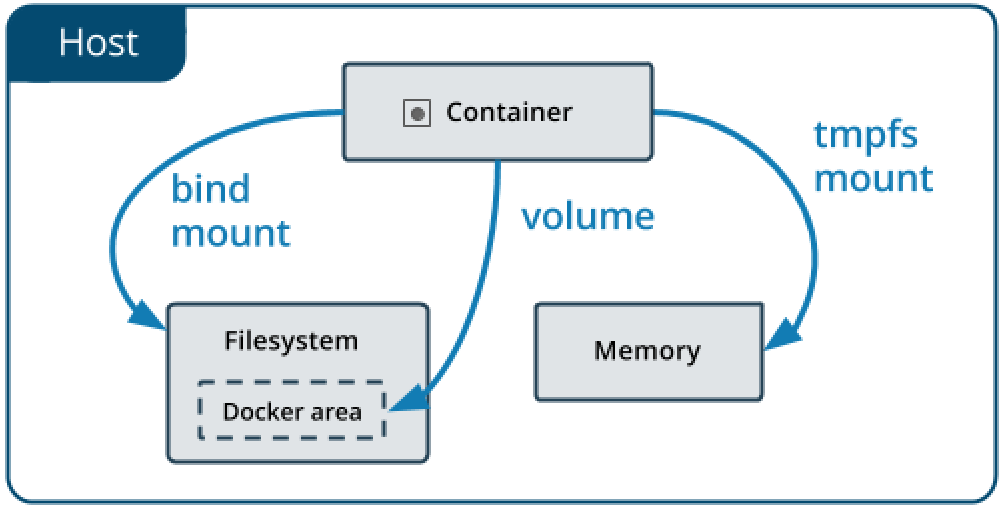
```
FROM mcr.microsoft.com/dotnet/core/sdk:3.1-buster AS build-env
WORKDIR /app

COPY *.csproj ./
RUN dotnet restore

COPY . ./
RUN dotnet publish -c Release -o out

FROM mcr.microsoft.com/dotnet/core/runtime:3.1-buster-slim-arm64v7
WORKDIR /app
COPY --from=build-env /app/out ./
COPY KeyWord.table ./
COPY MotorVoiceCommands.Json ./
COPY EdgeHost.txt ./

RUN apt-get update -y
RUN apt-get -y install alsa-utils
RUN apt-get -y install procps

ENTRYPOINT ["dotnet", "VoiceRecognitionModule.dll"]
```

Vzorový Docker file pre NodeRed



```
# Use node-red official containers
FROM nodered/node-red:1.3.5-12-minimal-arm64v8

WORKDIR /usr/src/node-red
USER node-red

COPY ./flows.json /data/flows.json

# Install the opcua extension
RUN npm install node-red-contrib-opcua@0.2.263

# Install the dashboard extension
RUN npm install node-red-dashboard

# Install Azure IoT Edge Module nodes
RUN npm install node-red-contrib-azure-iot-edge-module

# Expose Port 53880 for OPC UA Server
EXPOSE 53880
```

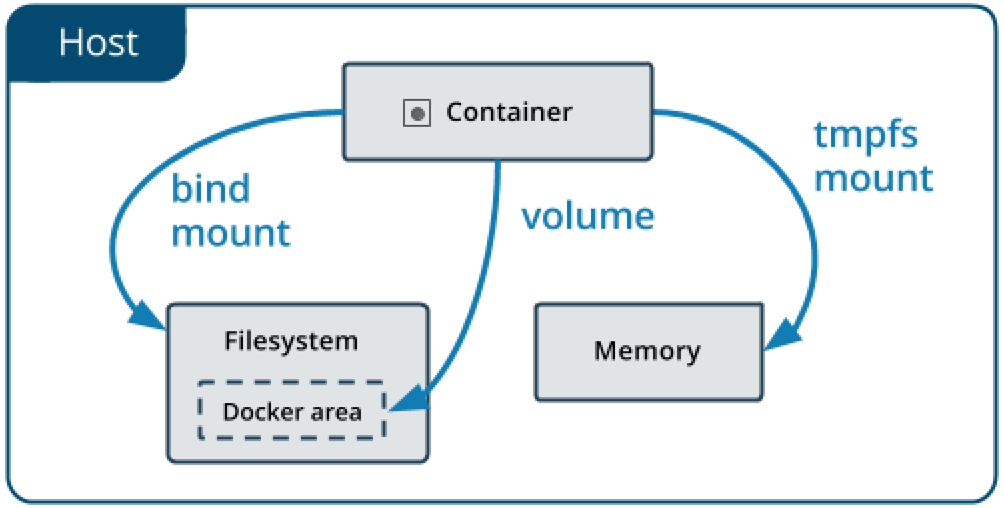
Docker perzistencia údajov



- Perzistencia údajov prostredníctvom **Volumes**
  - **Volumes** - sú uložené v časti hostiteľského súborového systému, ktorý spravuje Docker *(/var/lib/docker/volumes/ v systéme Linux*).

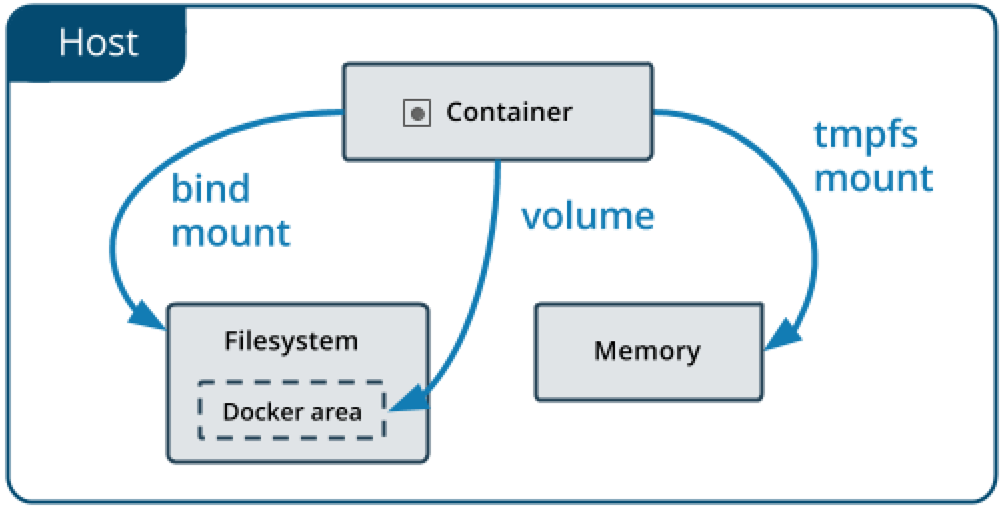
  Non-Docker procesy by **nemali upravovať túto časť súborového systému**. **Volumes** sú **najlepším spôsobom** uchovávania údajov v Dockeri.

Docker perzistencia údajov



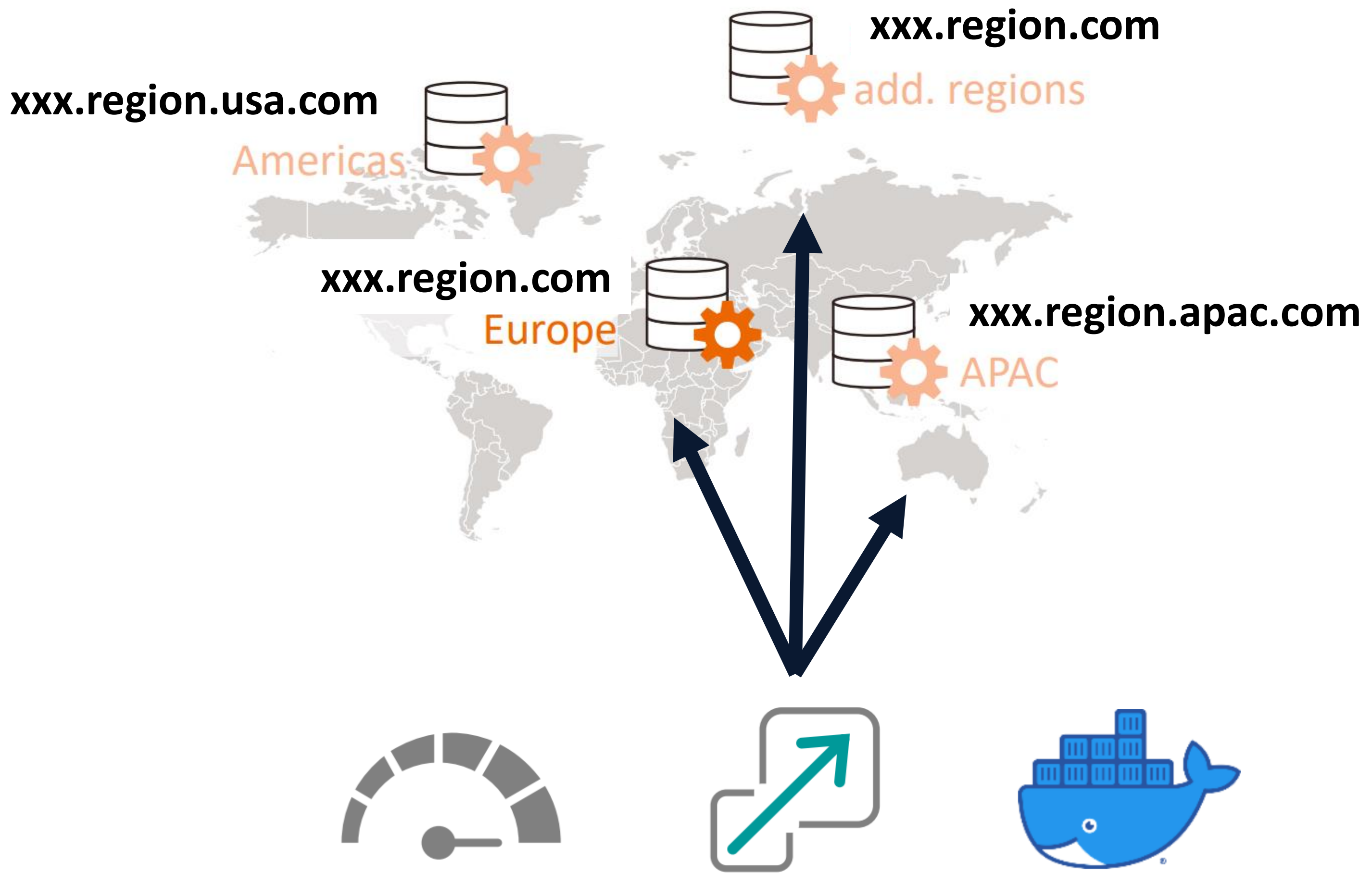
- Perzistencia údajov prostredníctvom **Bind mount**
  - **Bind mount** môžu byť **uložené kdekoľvek v hostiteľskom systéme**. Môžu to byť dokonca dôležité systémové súbory alebo adresáre. **Procesy mimo Docker ich môžu kedykoľvek upraviť.**

Docker perzistencia údajov



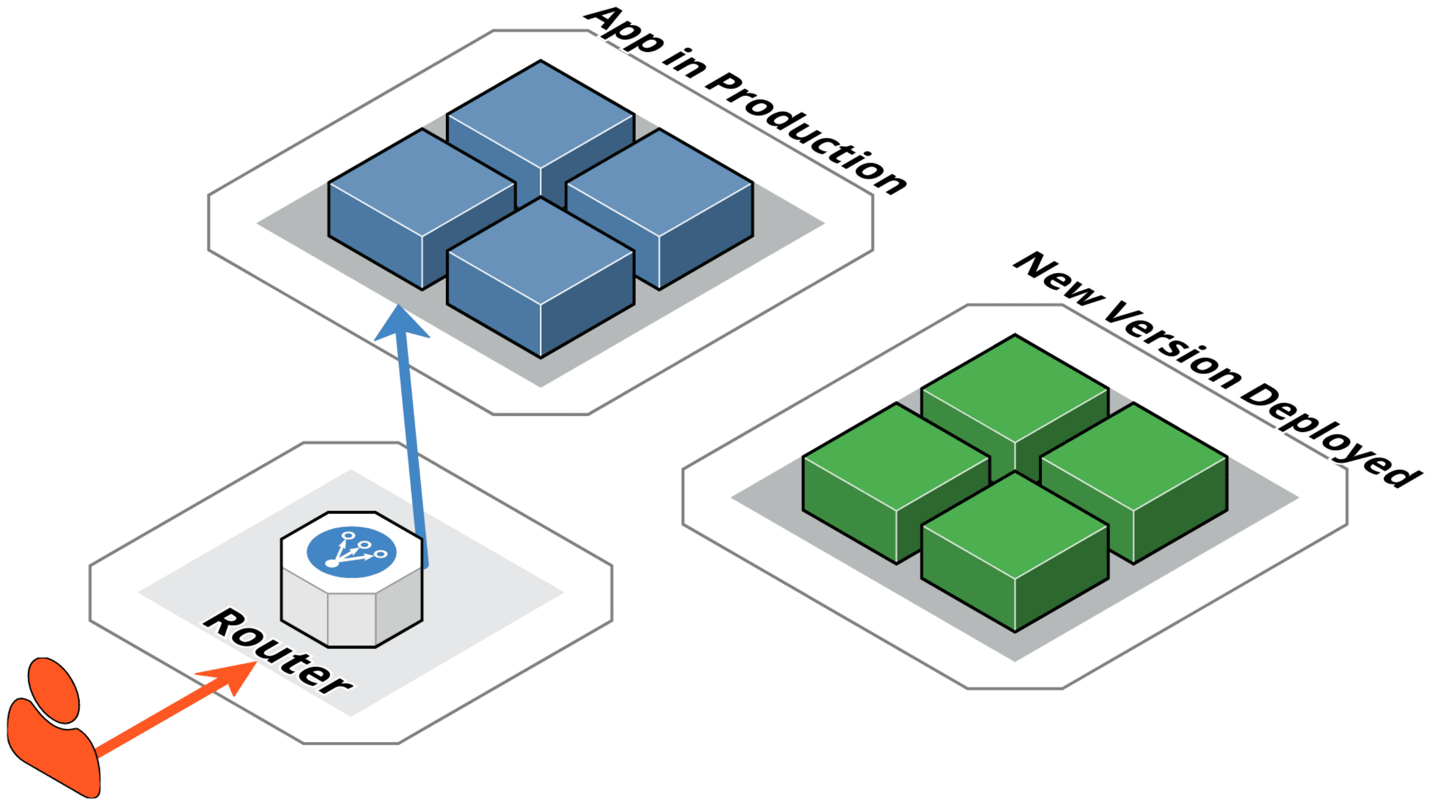
- Perzistencia údajov prostredníctvom **Tmpfs**
  - **Tmpfs** - **sú uložené iba v pamäti hostiteľského systému** a nikdy sa nezapisujú do súborového systému hostiteľského systému

Výhody kontajnerizácie



I. Výkonnosť II. Škálovateľnosť III. Automatické nasadenie

Green to Blue nasadanie



V **Green** to **Blue** nasadeniach sú udržiavané dva servery: „modrý" server a „zelený" server. V každom danom čase iba jeden server spracováva požiadavky (napr. na ktoré odkazuje DNS).

Verejné požiadavky môžu byť napríklad smerované na modrý server, čo z neho robí produkčný server a „zelený" server sa stáva zaisťovacím serverom, ku ktorému je možné pristupovať iba v súkromnej sieti.

Zmeny sa nainštalujú na neaktívny „zelený" server, ktorý sa potom testuje prostredníctvom súkromnej siete, aby sa overilo, či zmeny fungujú podľa očakávania. Po overení sa neaktívny server vymení za aktívny server, čím sa nasadené zmeny efektívne prevedú do produkcie.

Vytvorenie a spustenie kontajnera

1. **Vytvorenie priečinku napr.** *C:\DigitechProject\App*
2. **Vytvorenie súboru app.js na adrese** *C:\DigitechProject\App*
3. **Nakopírovanie aplikácie do app.js**

```
const express = require("express");
const http = require("http");

const app = express();

const server = http.createServer(app);

app.get('/', function(req,res){
	res.send("Hello World");
});

server.listen(3000, function(){
	console.log("Server is listening on port: 3000");
});
```

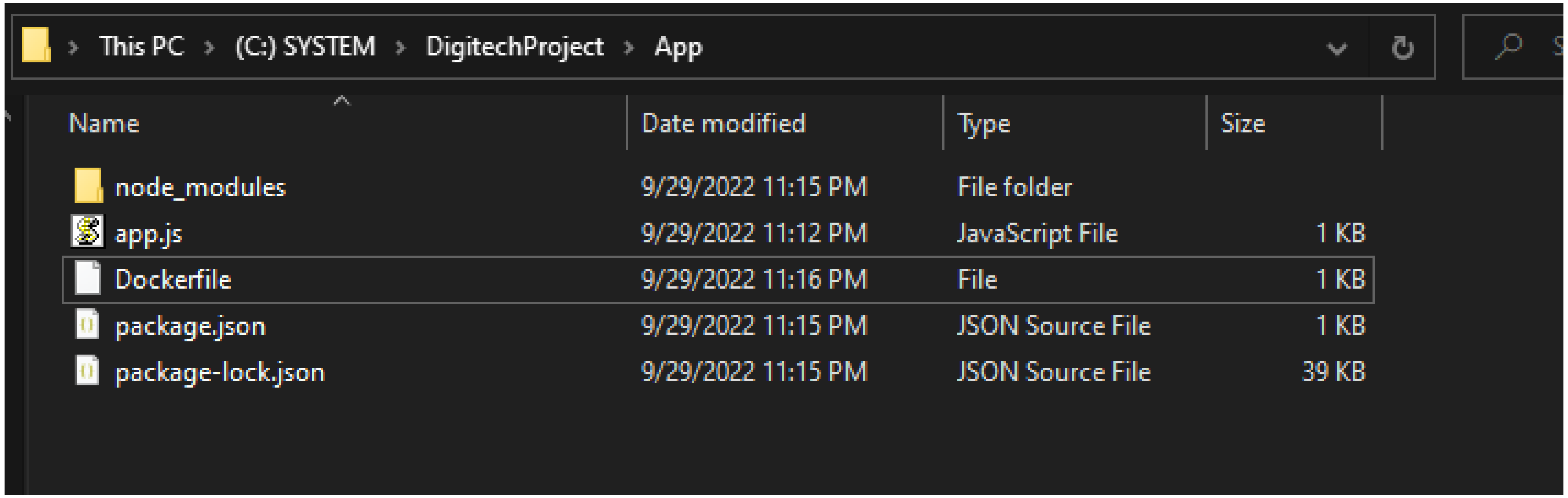
Úprava aplikácie

4. Inštalácia npm - *https://nodejs.org/en/download/*

5. Spustenie príkazu v termináli *npm*

6. Spustenie príkazu *npm init* v termináli v lokalite *C:\DigitechProject\App*

7. Spustenie príkazu *npm install express --save* v termináli v lokalite *C:\DigitechProject\App*



Vytvorenie a spustenie kontajnera – Javascript

8. **Vytvorenie Docker File na mieste** *C:\ DigitechProject\App* pričom názov súboru je „**Dockerfile**“ – bez prípony
9. **Otvorenie súboru a nakopírovanie obsahu nižšie**

```
FROM node:10
WORKDIR /app
COPY package.json /app
RUN npm install
COPY . /app
CMD node app.js
EXPOSE 8080
```

Vytvorenie a spustenie kontajnera

Tagovanie

Názov pre Docker Image

Bodka zabezpečí hľadanie docker file v lokalite odkiaľ bol spustený terminál

10. Spustenie príkazu z terminálu *docker build -t digi-tech-one .*

Musíte sa nachádzať v priečinku *C:\ DigitechProject\App*

11. Spustenie príkazu z terminálu *docker images*

Mal by byť viditeľný náš vytvorený image *digi-tech-one*

```
PS C:\DigitechProject\App> docker images
REPOSITORY                                  TAG              IMAGE ID       CREATED          SIZE
digi-tech-one                               latest           ead3a798d9e8   37 seconds ago   209MB
getting-started                             latest           201be78e[illegible]7a2   16 minutes ago   213MB
researchphd.azurecr.io/noderedrevpimodule   1.3.6-arm32v7    49572c0a[illegible]461   6 months ago     229MB
researchphd.azurecr.io/noderedopcuamodule   1.6.3-arm32v7    c0983395[illegible]a72   6 months ago     325MB
researchphd.azurecr.io/noderedrevpimodule   1.3.5-arm32v7    f15c5fae[illegible]836   6 months ago     229MB
```

12. Spustenie docker image pomocou príkazu *docker run -d -p 8080:3000 digi-tech-one*

Kontajner beží v pozadí, nedôjde k vypnutie po zatvorení terminálu

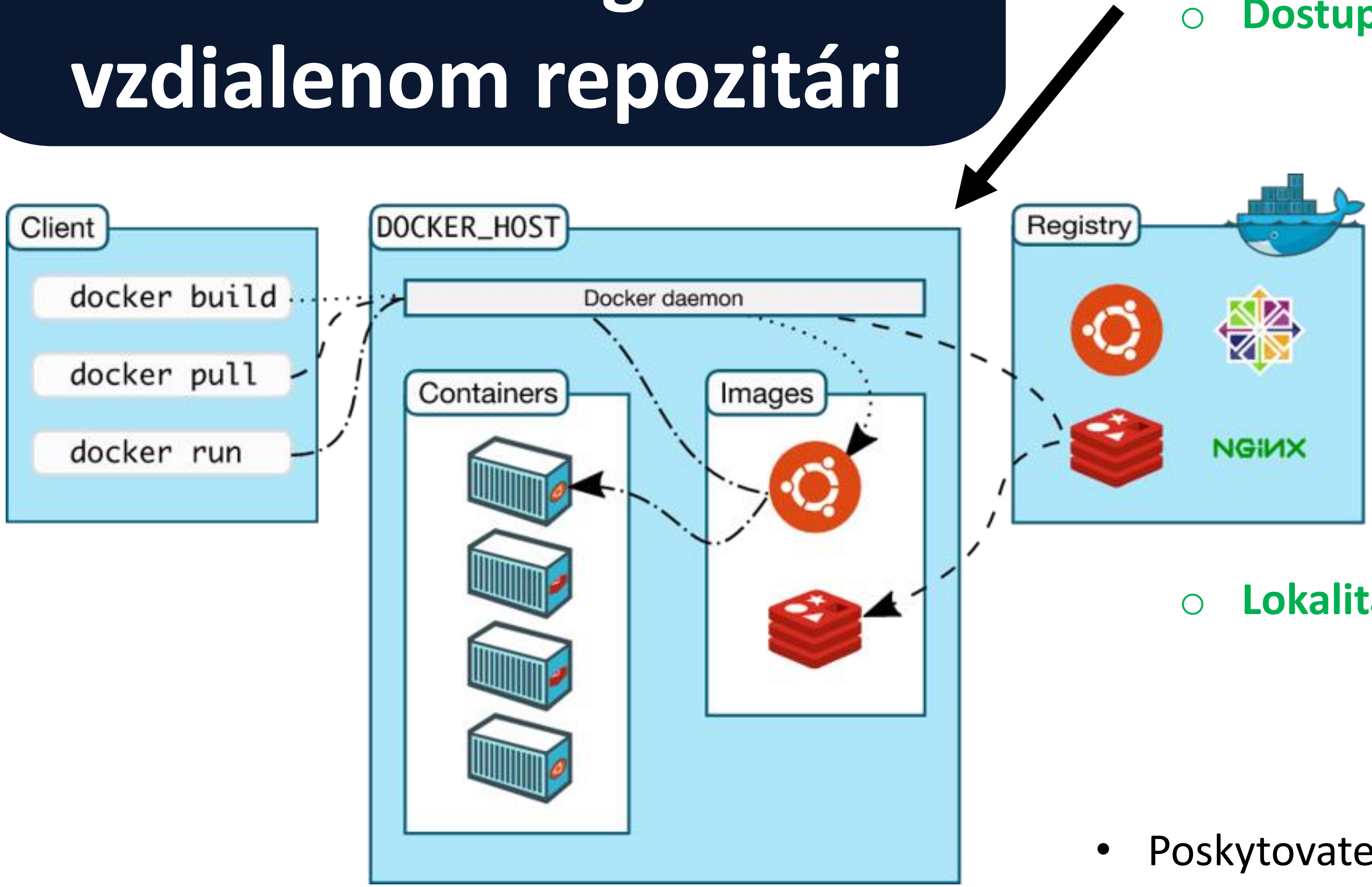
Mapovanie portov do vonkajšieho sveta

Vytvorenie a spustenie kontajnera

13. Spustenie príkazu z terminálu *docker ps* mal by sa zobraziť bežiaci vytvorený kontajner :)

```
CONTAINER ID   IMAGE           COMMAND                  CREATED         STATUS         PORTS
  NAMES
80675e702b2f   digi-tech-one   "docker-entrypoint.s…"   9 minutes ago   Up 9 minutes   8080/tcp, 0.0.0.0:8080->3000/tcp
  nervous_swirles
```

Docker image vo vzdialenom repozitári

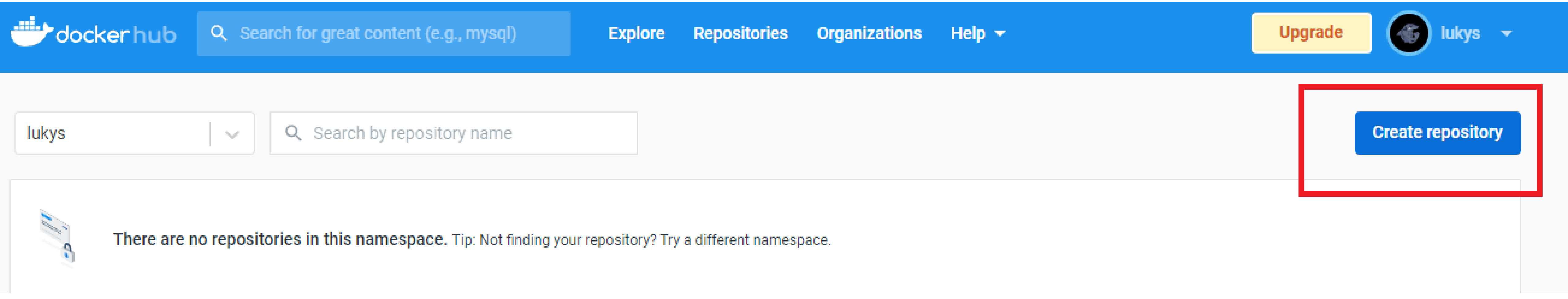


- **Docker registry** – skladovanie Docker images
  - **Dostupnosť:**
    - **Verejné**
    - **Súkromné**
  - **Lokalita:**
    - **Lokálne**
    - **Vzdialené**
- Poskytovatelia vzdialeného uloženia:
  - **Docker Hub**
  - **Azure Container Registry**
  - **Amazon Elastic Container Registry**

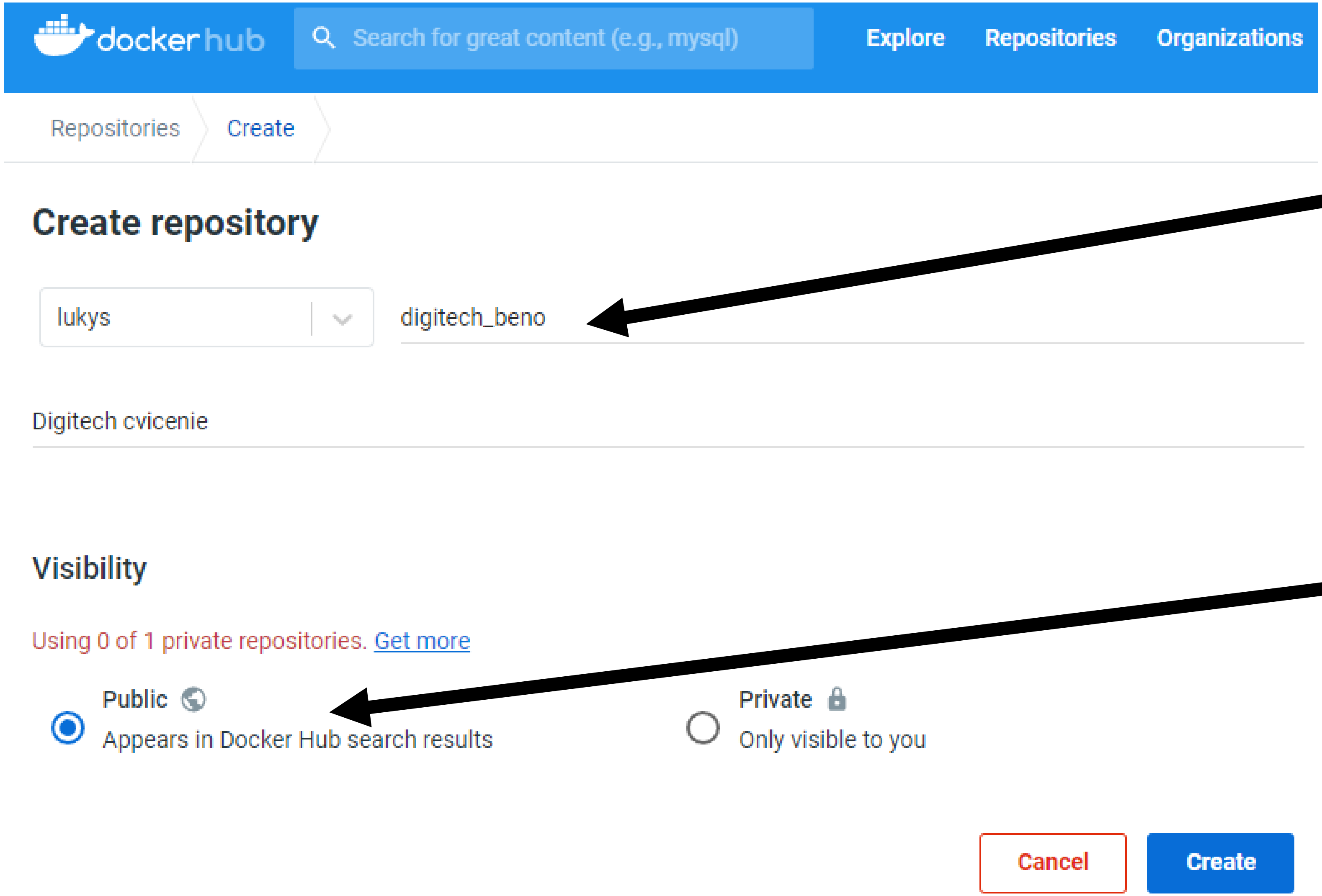
Docker hub

- **Podmienky:**
  - Nainštalovaný Docker desktop
  - Vytvorený účet na https://hub.docker.com

1. **Vytvorenie Docker registry repozitára -** https://hub.docker.com/repositories



Docker hub



digitech_priezvisko

Verejné dostupný

Docker hub

Príkaz - docker images

2. **Spustenie príkazu z cmd:** *docker login*

3. **Tagovanie image**

   Predpis: *docker tag imageid new-repo:tagname*

   Spustenie príkazu z cmd – ukážka:

   docker tag afc8b4ede07e lukys/digitech_beno:version2.0

```
C:\Users\Z003VC0N>docker images
REPOSITORY                                   TAG              IMAGE ID       CREATED         SIZE
digitech                                     version1.0       afc8b4ede07e   13 days ago     917MB
```

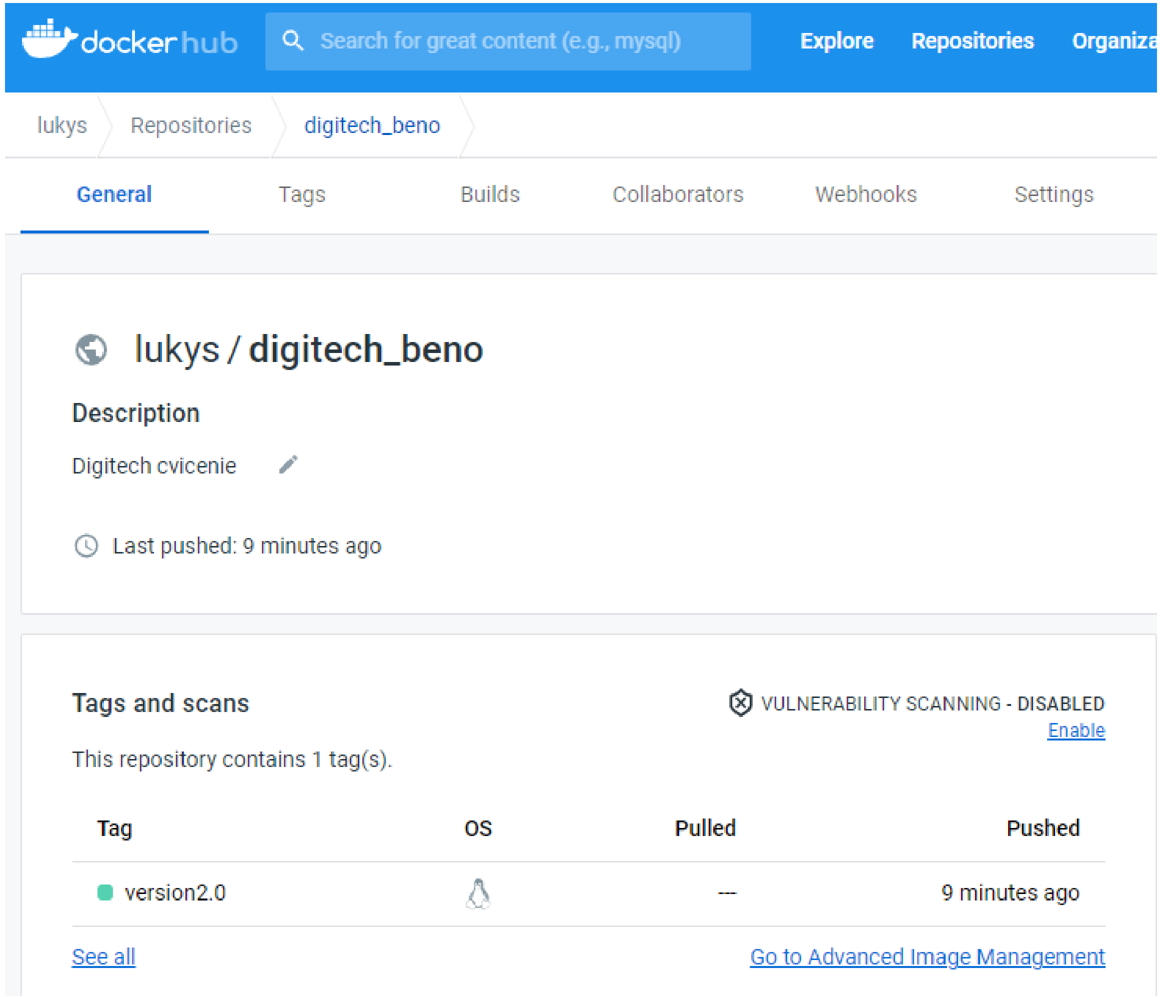
Docker hub

4. **Uloženie image vo vzdialenom repozitári**

   Predpis: *docker push new-repo:tagname*

   Spustenie príkazu z cmd:

   docker push lukys/digitech_beno:version2.0



Nginx – load balancer

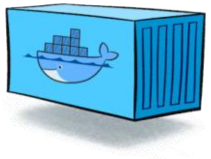
- Vyvažovanie záťaže naprieč viacerými inštanciami aplikácií prostredníctvom distribúcie prevádzky na niekoľko aplikačných serverov - optimalizácia využitia zdrojov.

- Maximalizácia priepustnosti, zníženie latencie a zabezpečenie konfigurácií odolných voči chybám.

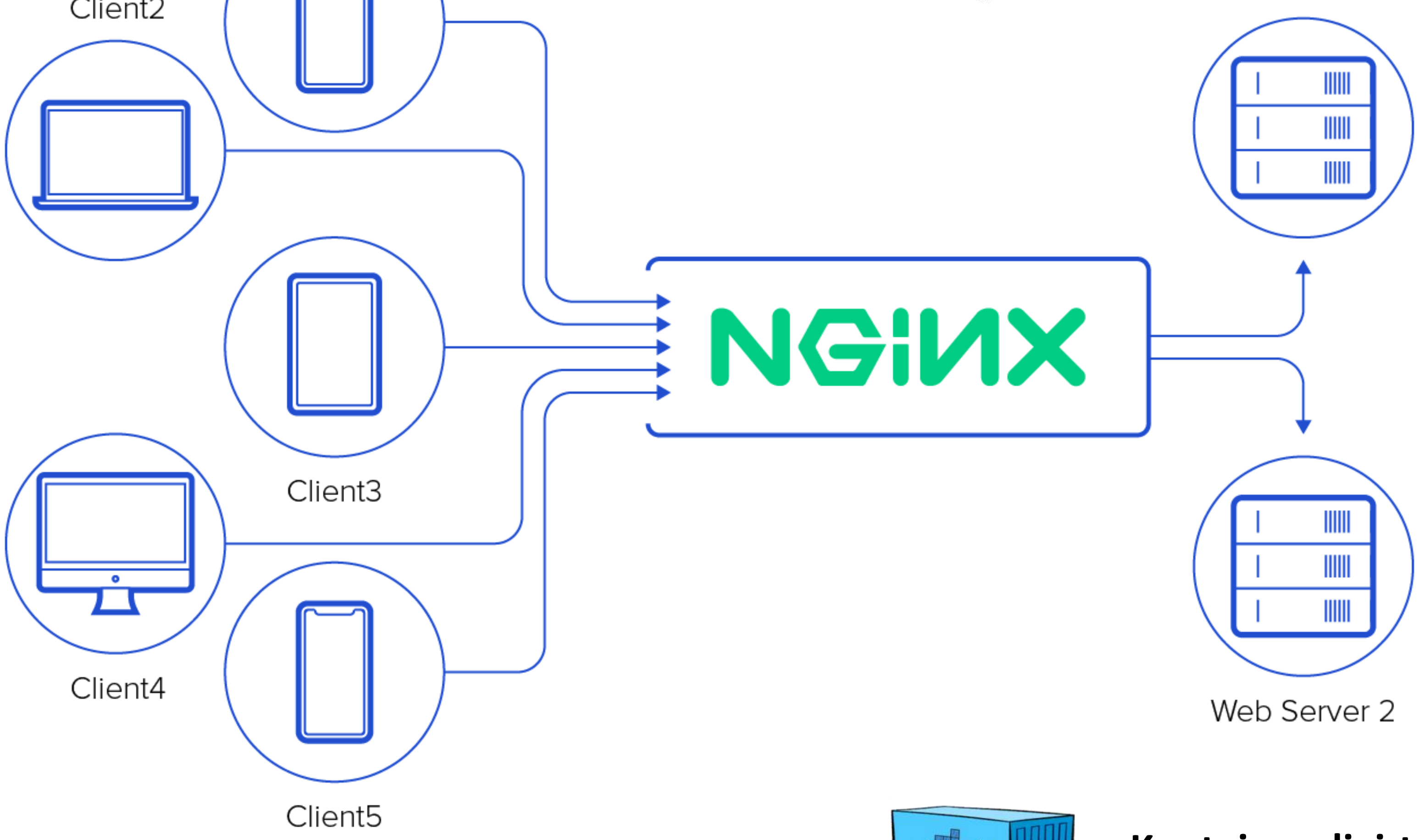
- Zlepšenie výkonu, škálovateľnosti a spoľahlivosti webových aplikácií.

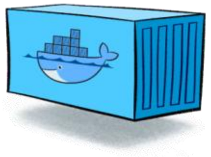
- Efektívny nástroj na vyrovnávanie záťaže HTTP.

Nginx – load balancer



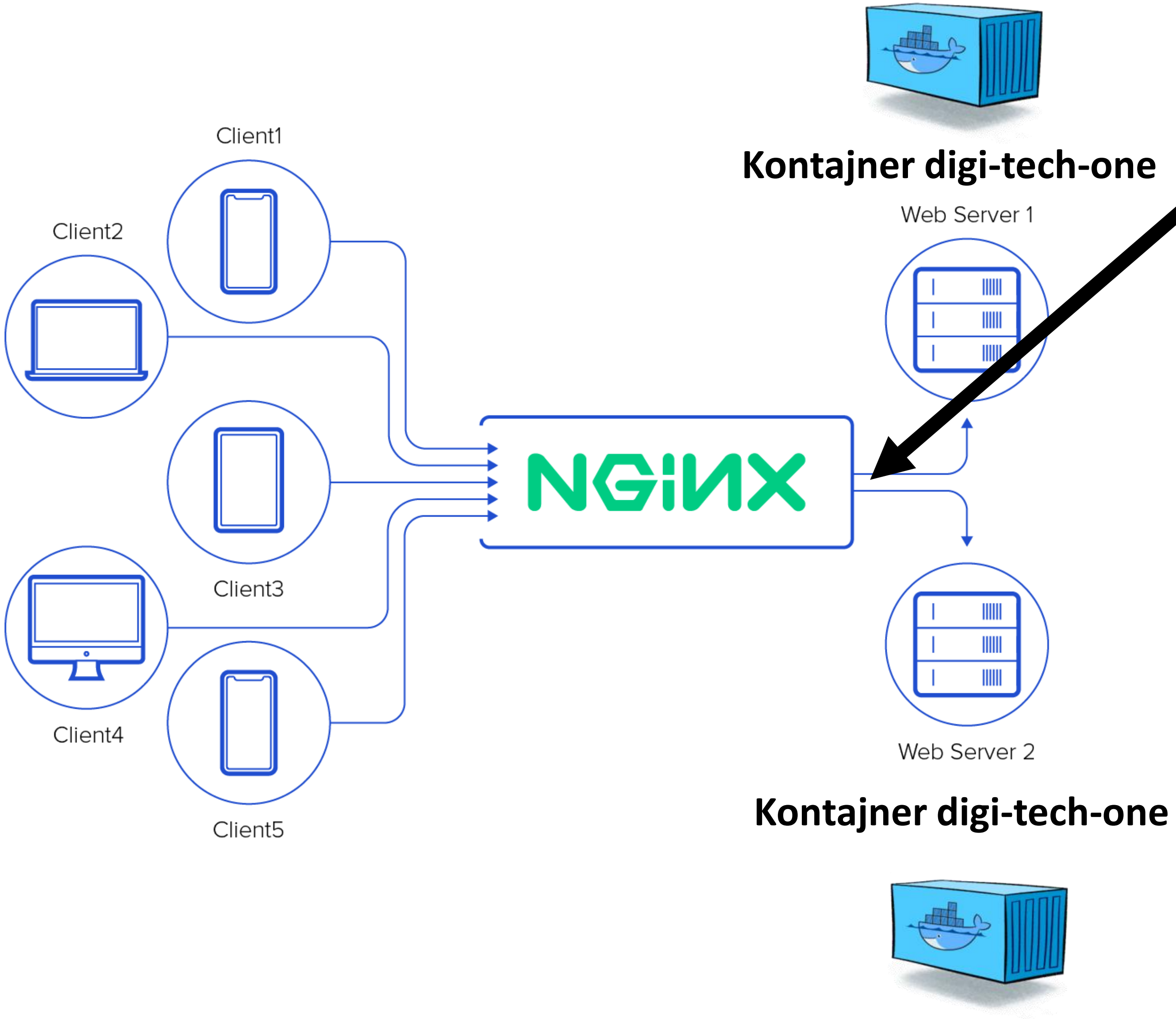
Kontajner digi-tech-one





Kontajner digi-tech-one

Nginx – Round-robin DNS



Round-robin DNS:

- **Technika rozloženia záťaže.**
- **Rovnomerné rozloženie požiadaviek medzi mnohými webovými servermi.**

Nginx – Round-robin DNS

NGINX

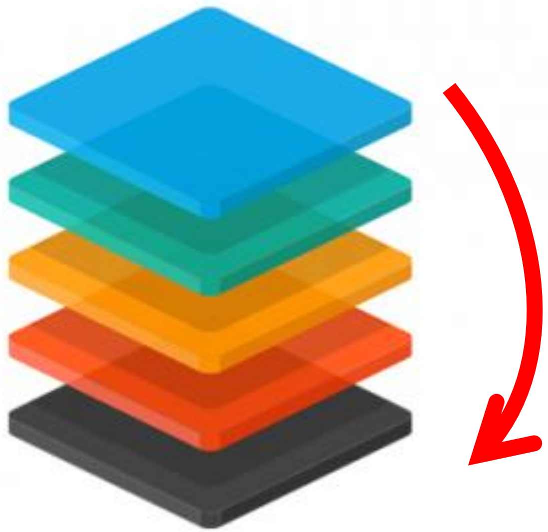
Napríklad spoločnosť má jeden názov domény a tri identické kópie rovnakej webovej stránky s umiestnením na troch serveroch s tromi rôznymi IP adresami.

Keď jeden používateľ pristupuje na domovskú stránku bude odoslaná na prvú IP adresu.

Druhý používateľ, ktorý pristupuje na domovskú stránku bude zaslaný na ďalšie IP adresy, a tretí používateľ bude zaslaný na tretiu IP adresu.

V každom prípade, akonáhle je IP adresa je vydaná, prejde sa na koniec zoznamu.

Štvrtý užívateľ teda bude zaslaný na prvú IP adresu a tak ďalej.



Docker compose

Docker Compose

- **Nástroj pre definovanie a spustenie multi-kontajnerov Docker aplikácií.**
- **YAML súbor na konfiguráciu**

```yaml
version: "3"

services:
  digi-tech-one:
    image: "digi-tech-one:latest"
    expose:
      - "3000"
  nginx:
    image: nginx:latest
    volumes:
      - ./nginx.conf:/etc/nginx/nginx.conf:ro
    depends_on:
      - digi-tech-one
    ports:
      - "5000:4000"
```

Image, ktorý chceme škálovať

Port využitý našou aplikáciou

Image pre Nginx

Úložisko pre Nginx

Port, kde budú smerovať požiadavky z vonkajšieho sveta

Port, kde načúva Nginx

Implementácia

1. *Vytvorenie priečinku napr. Yaml*
2. *Vytvorenie súboru v priečinku s názvom docker-compose.yml a nakopírovanie obsahu nižšie*

```yaml
version: "3"

services:
  digi-tech-one:
    image: "digi-tech-one:latest"
    expose:
      - "3000"
  nginx:
    image: nginx:latest
    volumes:
      - ./nginx.conf:/etc/nginx/nginx.conf:ro
    depends_on:
      - digi-tech-one
    ports:
      - "5000:4000"
```

Implementácia

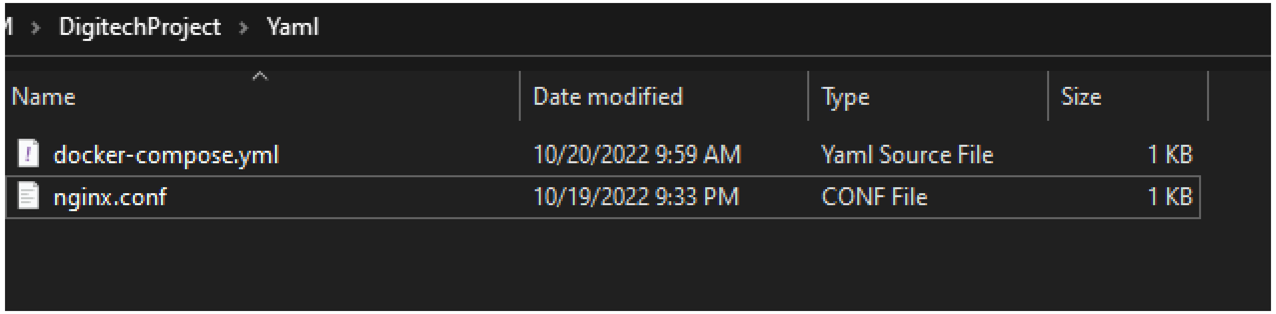
3. Vytvorenie súboru v priečinku s názvom nginx.conf a nakopírovanie obsahu nižšie

Maximálny počet pripojení

```
user  nginx;

events {
    worker_connections   1000;
}
http {
        server {
              listen 4000;
              location / {
                proxy_pass http://digi-tech-one:3000;
              }
        }
}
```

Implementácia

Štruktúra v priečinku je nasledovná:



› DigitechProject › Yaml

Name	Date modified	Type	Size
docker-compose.yml	10/20/2022 9:59 AM	Yaml Source File	1 KB
nginx.conf	10/19/2022 9:33 PM	CONF File	1 KB

4. ***Otvorenie terminálu v priečinku a spustenie nasledovného príkazu:***

docker-compose up --scale digi-tech-one=5

Spustenie piatich inštancií

Implementácia

5. Vyčkanie kým docker compose nabehne:

```
PS C:\DigitechProject\Yaml> docker-compose up --scale digi-tech-one=5
[+] Running 6/6
 - Container yaml-digi-tech-one-4  Created                                                     0.0s
 - Container yaml-digi-tech-one-2  Created                                                     0.0s
 - Container yaml-digi-tech-one-3  Created                                                     0.0s
 - Container yaml-digi-tech-one-5  Created                                                     0.0s
 - Container yaml-digi-tech-one-1  Created                                                     0.0s
 - Container yaml-nginx-1          Recreated                                                   0.2s
Attaching to yaml-digi-tech-one-1, yaml-digi-tech-one-2, yaml-digi-tech-one-3, yaml-digi-tech-one-4, yaml-digi-tech-one-
5, yaml-nginx-1
                        Server is listening on port: 3000
yaml-digi-tech-one-5  | Server is listening on port: 3000
yaml-digi-tech-one-2  | Server is listening on port: 3000
yaml-digi-tech-one-3  | Server is listening on port: 3000
yaml-digi-tech-one-1  | Server is listening on port: 3000
yaml-nginx-1          | /docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
yaml-nginx-1          | /docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
yaml-nginx-1          | /docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
yaml-nginx-1          | 10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
yaml-nginx-1          | 10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
yaml-nginx-1          | /docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
yaml-nginx-1          | /docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
yaml-nginx-1          | /docker-entrypoint.sh: Configuration complete; ready for start up
```

Riešenie je pripravené

Implementácia

5. *Spustenie nasledovného príkazu v termináli:*

docker container ls

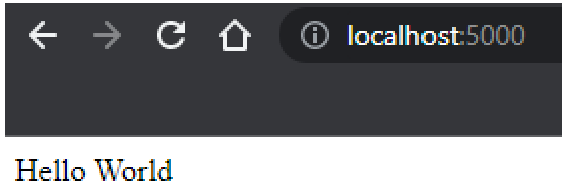
Malo by sa zobraziť 5 bežiacich inštancií:

```
PS C:\Users\Z003VC0N> docker container ls
CONTAINER ID   IMAGE                  COMMAND                  CREATED             STATUS              PORTS
               NAMES
5c41564b529e   nginx:latest           "/docker-entrypoint.…"   About an hour ago   Up About an hour    80/tcp, 0.0.0.0:40
00->5000/tcp   yaml-nginx-1
d648c1c73df9   digi-tech-one:latest   "docker-entrypoint.s…"   12 hours ago        Up About an hour    3000/tcp, 8080/tcp
               yaml-digi-tech-one-4
d481ab8aa5c1   digi-tech-one:latest   "docker-entrypoint.s…"   12 hours ago        Up About an hour    3000/tcp, 8080/tcp
               yaml-digi-tech-one-2
c0afee624714   digi-tech-one:latest   "docker-entrypoint.s…"   12 hours ago        Up About an hour    3000/tcp, 8080/tcp
               yaml-digi-tech-one-3
33db15f744fc   digi-tech-one:latest   "docker-entrypoint.s…"   12 hours ago        Up About an hour    3000/tcp, 8080/tcp
               yaml-digi-tech-one-5
5211e85ad2b5   digi-tech-one:latest   "docker-entrypoint.s…"   12 hours ago        Up About an hour    3000/tcp, 8080/tcp
               yaml-digi-tech-one-1
```

Implementácia

6. Otvorenie adresy http://localhost:5000/

   Mala by sa zobraziť naša web stránka:



Kubernates – rozšírenie obzoru

- https://www.websupport.sk/blog/2022/09/ovladnite-serverove-technologie/?xnpe_tifc=OIQsh.zp4dojhkbjxDoXbjpZhfEWVjQsVuU_O.YpOIHD4Mps4uQNxDHXbD4.4.YJh.Y8&utm_source=exponea&utm_campaign=Polovica%20ceny%20.SK%20adresy%20je%20dnes%20na%20n%C3%A1s&utm_medium=mail

- https://www.atpjournal.sk/buxus/docs/casopisy_cele/ATP%20Journal%209%202022.pdf#page=54

- https://magazin.kpi.fei.tuke.sk/2019/04/kubernetes-a-jeho-zakladne-pouzitie/

Literatúra

- https://levelup.gitconnected.com/dockerize-a-hello-world-node-js-application-a15c0564a50e
- https://docs.docker.com/get-started/overview/
- https://docs.docker.com/storage/
- https://learn.microsoft.com/en-us/dotnet/core/docker/build-container?tabs=windows

Literatúra

- http://nginx.org/en/docs/http/load_balancing.html
- https://cs.wikipedia.org/wiki/Round-robin_DNS
- https://pspdfkit.com/blog/2018/how-to-use-docker-compose-to-run-multiple-instances-of-a-service-in-development/