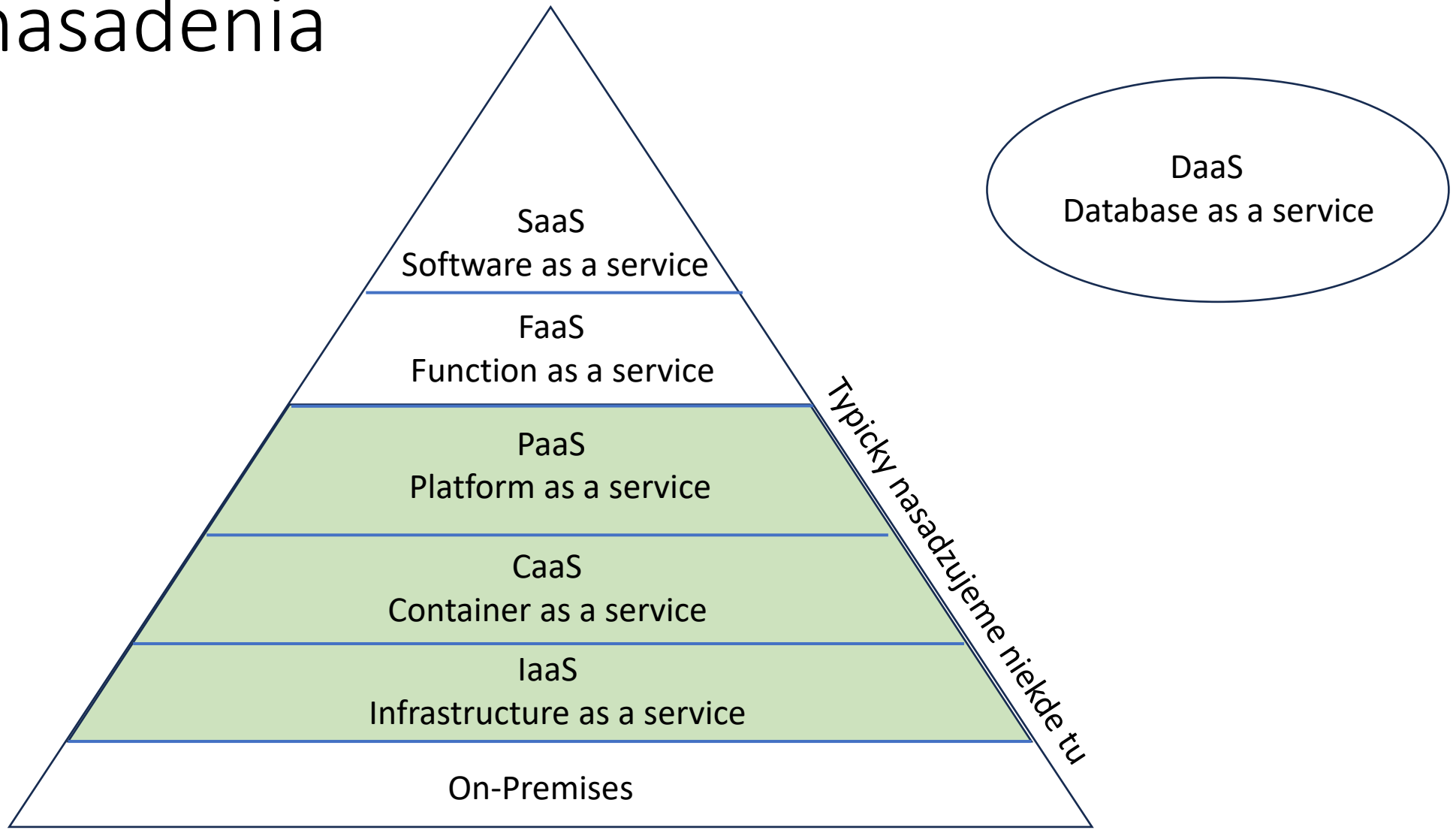
Ako nasadíme web appku?

Projekt 1

Nasadenie

- Kritéria
  - Bezpečnosť
  - Výkon servera
  - Robustnosť a redundantnosť
  - Cena

Typy nasadenia

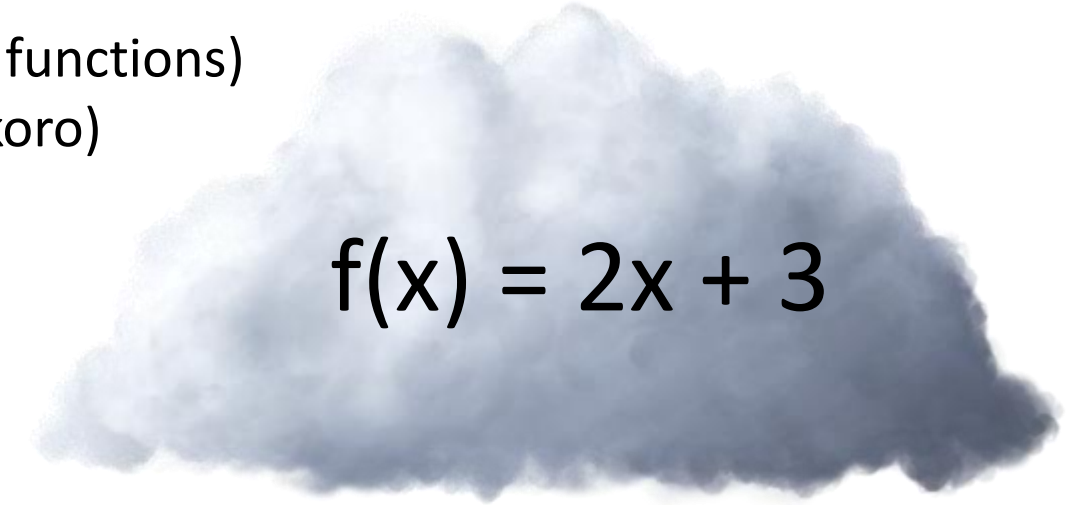


SaaS – Software-as-a-service

- Prenajnete si hotový program
- Free, alebo veľmi lacné pre fyzické osoby
  - Cloud úložisko, email, kalendár, ...
- Pre podnikateľov a firmy
  - Účtovníctvo, správa používateľov, informačné systémy

FaaS – Function-as-a-service

- Tiež známe ako bezserverové funkcie (serverless functions)
- Nasadzujete priamo „triedy/metódy/funkcie“ (skoro)
- Elegantné riešenie pre mikroslužby
- Nevhodné pre monolitické aplikácie (náš prípad)
- Limitovaná podpora jazykov
  - JS, TS, Python – najlepšia podpora
  - Go, Rust – ak treba výkon
  - Java, C# – zvyčajne iba do počtu, samé nevýhody
- Appka musí byť špeciálne navrhnutá pre FaaS nasadenie
  - Používate knižnice a frameworky priamo od cloud poskytovateľa
  - Ťažký vendor-lock
  - Spring Boot síce umožňuje nasadenie ako FaaS, ale nerobil by som to
- Platíte viac-menej iba za využitý CPU čas a pamäť
- AWS **Lambda**
  Azure **Functions**
  Google **Cloud Functions**



PaaS – Platform-as-a-service

- Prenajmem si na cloude „platformu“
  - Môže byť interpreter/runtime jazyka
    - Java, Python, Node.js, ...
  - Alebo aplikačný server
    - Tomcat, Apache
- Heroku
  - (skoro) Daj tam link na svoj Git a tvoja appka je hostovaná
  - ~~Free tier na hranie~~ (do 2022)
  - Celkom lacné na malé veci
  - Veľmi drahé na veľké veci
- AWS Elastic Beanstalk, Azure App Services, Google App Engine, ...

CaaS – Container-as-a-service

- Pracujú s OCI obrazmi (OCI obraz = Docker obraz)
- Univerzálne vzhľadom na jazyky
- Pre korporáty
  - Orchestrátory kontajnerov v cloude
    - Kubernetes, Nomad, OpenShift
  - Orchestrátory orchestrátorov kontajnerov
    - Rancher, Gardener
  - EKS – AWS Elastic Kubernetes Engine
    AKS – Azure Kubernetes Engine
    GKS – Google Kubernetes Engine
    ...
- Pre menšie tímy a sólo vývojárov
  - AWS Fargate, Azure Containers, Google Cloud Run, ...
  - Dáte tam OCI obraz a fičíte

DaaS – Database-as-a-service

- Manažované databázy
- Chcem DB s toľko CPU, RAM, disku
- Alebo chcem iba DB
  - Platíte iba za aktuálne využitý priestor a CPU čas
  - Po novom aj pre „tradičné“ SQL databázy
- Poskytovateľ rieši replikáciu, zálohovanie, ...
- Cenovo prijateľné pre cloud-natívne DB priamo od poskytovateľa
  - AWS DynamoDB, Azure TableStorage, Google Firebase
    - Sú to NoSQL tabuľky
  - AWS **S3**, Azure BS, Google CS
    - Na súbory
- Pre iné DB to už vie byť celkom drahé
  - Najmä pre SQL DB

IaaS – Infrastructure-as-a-service

- Niečo medzi vlastným HW a PaaS
- Cenovo dostupné aj pre jednotlivcov
- Prenajímate si OS
  - A množstvo CPU (vlákien), RAM, disk k nemu
- Dostanete SSH/RDP prístup na vlastnú Linux/Windows Server inštanciu
- Varianty
  - VPS – Virtual Private Server
    - Máte „izolovaný“ OS, ale fyzický HW je zdieľaný medzi viac VPS - virtualizácia
    - Skvelé pre web appky
    - Nevhodné pre dlhotrvajúce výpočty atď.
  - VDS – Virtual Dedicated Server
    - Ako VPS, ale máte garantovaný výkon CPU a disku
  - DS – Dedicated Server
    - Prenajmete si fyzický server alebo stolný počítač

On-Premises - vlastné železo

- Potrebujem internet s aspoň 1 verejnou IP
- Kúpim si hardvér
  - Minimálne desktop s diskami v RAID 1 (doma)
  - Mini-rack (doma)
  - Plnotučný server do racku
    - Prenajať priestor v datacentre
      - Doma to nechcete, lebo to je hlučné a žerie to
    - Mať vlastnú serverovňu
    - Drahé jak sviňa pre súkromnú osobu,
      ale ak ste veľká organizácia, tak vlastné servery sú lacnejšie z dlhodobého hľadiska

Entrance App

- Máme appku, ktorá sa skladá z
  - React FE (**Entrance Web**)
  - Spring Boot BE (**Entrance API**)
  - **Keycloak**
  - **MySQL**

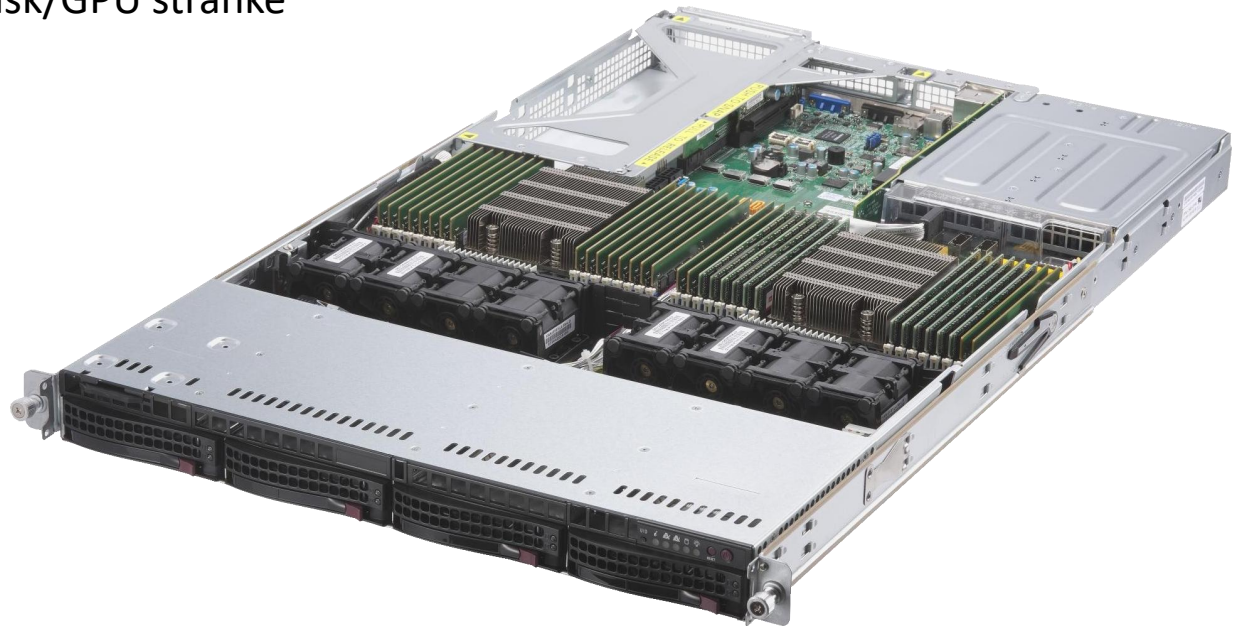
IaaS nasadenie

Koľko serverov?

- Dlho platilo pravidlo, že každý program by mal bežať na vlastnom serveri
  - Programy: MySQL, Keycloak, *Entrance API*, *Entrance Web*,
    ... všetko na samostatnom serveri, ideálne aj na inom fyz. stroji
- Viac programov na 1 serveri sa môžu „biť“
  - Môžu potrebovať iné systémové knižnice
  - Alebo rovno špecifický OS
- Teória: Ak jeden server/program padne, tak zbytok našej appky funguje
  - Realita: ak nám spadne hociktorý program, tak appka je aj tak nepoužiteľná
  - Potrebujeme **repliky** (hlavný server + záložné servery) pre každý program
- S viac VPS je veľmi veľa práce
  - Navyše servery s DB nesmú byť dostupné z internetu

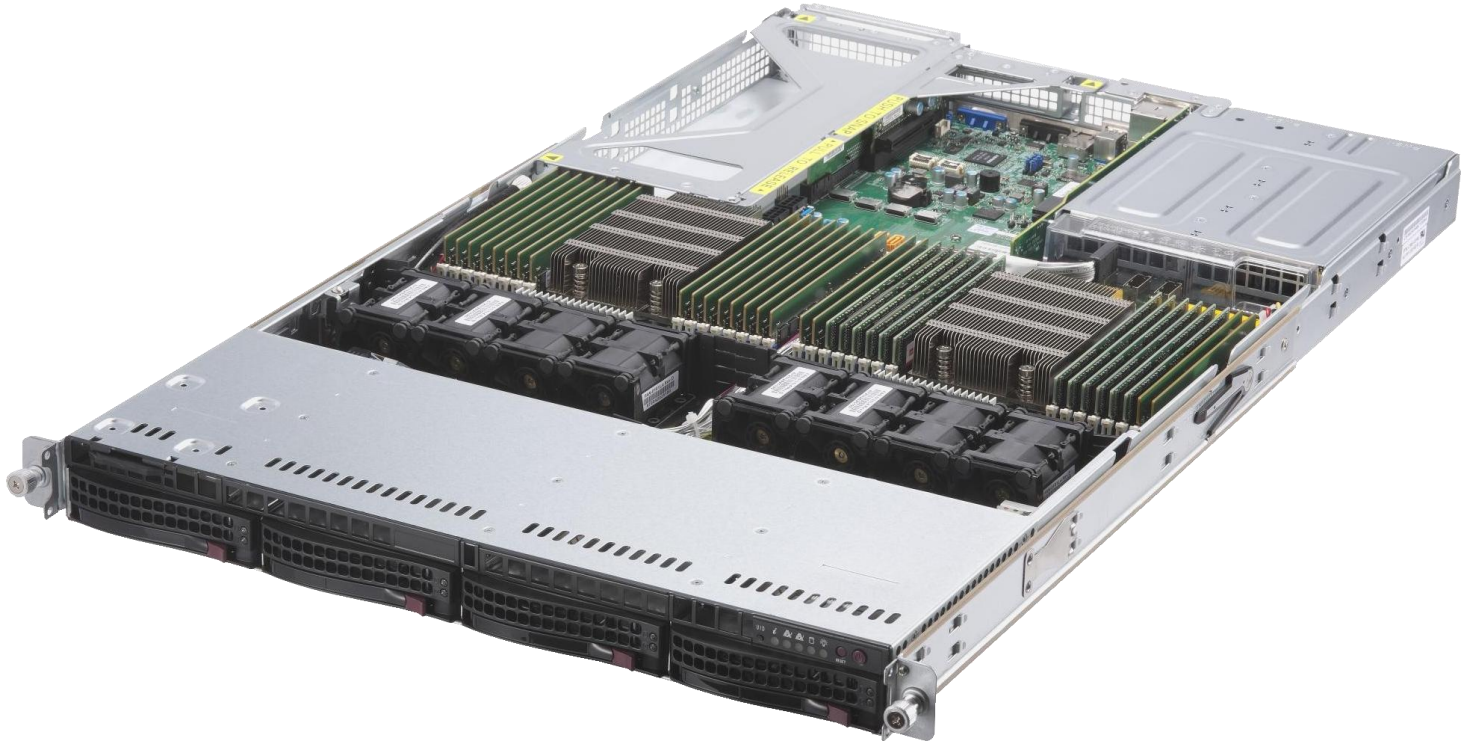
Nasadenie na jediný (virt./fyz.) server

- Vďaka kontajnerom vieme eliminovať mnohé problémy jedno-serverového nasadenia
  - Aj bežný komoditný HW z eshopu má výkonu až-až
  - VPS tiež vedia byť dosť nadupané po CPU/RAM/disk/GPU stránke
- Kontajnerizované programy sa nebudú „biť“
  - Majú vlastný „OS“ (user-space)
- Viete ušetriť veľa € na infraštruktúre
- Jednoduché na nasadenie (Docker Compose)
- Na mnoho projektov to stačí
- Nevýhody
  - Ak sa pokazí OS, tak je vaša appka offline
    - Vďaka kontajnerom je to málo pravdepodobné
  - Ak sa pokazí HW, tak je vaša appka offline
    - Špec. server HW je ale robustnejší ako komoditný HW
    - Viete za behu vymeniť disky, RAM, CPU, …
  - Časom upgrade HW nie je triviálny kvôli migrácii dát
  - Reálne aj tak potrebujete ešte ďalší (NAS) server na zálohy DB



(Moderné) nasadenie na viac serverov

- High Availability Deployment
  - Aplikácia musí prežiť chybu na úrovni HW, OS, DB, aj v samotnej appke
- Orchestrátory kontajnerov na vlastnom železe
  - ~~Docker Swarm~~, **Kubernetes** (K8s), OpenShift, Nomad
  - Servery sa budú správať ako jeden „ultra“ server – **klaster**
  - Ak jeden server vypadne z klastra...
  - ... orchestrátor automaticky presunie kontajnery na iný server
    - Pri DB toto musí podporovať aj ona samotná
  - Stačí vám spravovať iba klaster ako celok (viac-menej)
  - Riešia väčšinu komplexnosti viac-serverového nasadenia
    - Ale pridáva aj nejakú svoju komplexnosť
  - Viete zlepšiť výpočtový výkon pridaním nového servera do klastra
  - Web appky vieme triviálne replikovať
    - Replikácia MySQL a DB obecne je trocha menej triviálna



Nasadzujeme na jediný server

Kde zohnať (virt.) server

- Mnohé tradičné webhostingy ponúkajú aj VPS
- Poskytovateľov je veľa (aj v rámci SK)
- Na školské veci si viete zažiadať u CAI
- Komerčné riešenia:
  - Od lokálnych SK poskytovateľov cez stredne veľkých DE/FR/UK poskytovateľov po US cloudy
  - Nemci majú posledné roky asi najlepšie ceny (4cpu+4ram 5-10 €/m v DE vs 1cpu+1ram 15 €/m na SK)
  - VPS od US cloudov (AWS, Azure, GCP...) majú zďaleka najrýchlejší internet
    - 10-30 Gbps vs 1 Gbps od EÚ poskytovateľov (10 Gpps pre dedikované servery od EÚ poskytovateľov)
    - AWS dokonca ponúka VPS s 3 Tbps sieťou (asi $30k/m pri 3 ročnej viazanosti, $60k/m bez viazanosti)

Pripojenie na server

- Ak si prenajmete server, tak dostanete prihlasovacie údaje
  - Linux VPS cez ssh (terminál)
  - Windows VPS cez RDP
    - RDP je síce GUI, ale je to bazmeg deravý a treba nutne riešiť VPN kvôli bezpečnosti
- Následne sa odporúča (Linux server)
  - nastaviť prihlasovanie cez verejné kľúče (ssh-keygen)
  - vypnúť prihlasovanie cez heslo
    - alebo aspoň zmeniť heslo, čo vám dal poskytovateľ
  - Vypnúť „núdzové“ prihlasovacie spôsoby (VNC)
  - Pozor, ak následne stratíte ssh privátny kľúč, tak ste v keli
    - Šifrovane si zálohujte priečinok .ssh vo vašom PC !!!
      - 7zip s heslom, alebo VeraCrypt a zálohu dajte mimo PC (kľudne aj na cloud drive)

Nastavenie Firewallu

- Firewall – blokuje a povoľuje spojenia
- Štandardne je firewall na Linuxe vypnutý
- iptables/nftables
  - Štandartné Linux firewally
- ufw – Uncomplicated Firewall
  - Jednoduchá nadstavba nad iptables/nftables
- Potrebujeme ssh (22), http (80), https (443)
  - Iba tieto porty budú povolené.
  - Spojenia na iné porty server odmietne

ufw allow ssh
ufw allow http
ufw allow https

ufw enable

- POZOR: pred zapnutím firewallu (ufw enable) sa uistite, že ste povolili ssh, ináč sa už na VPS nedostanete

Doména

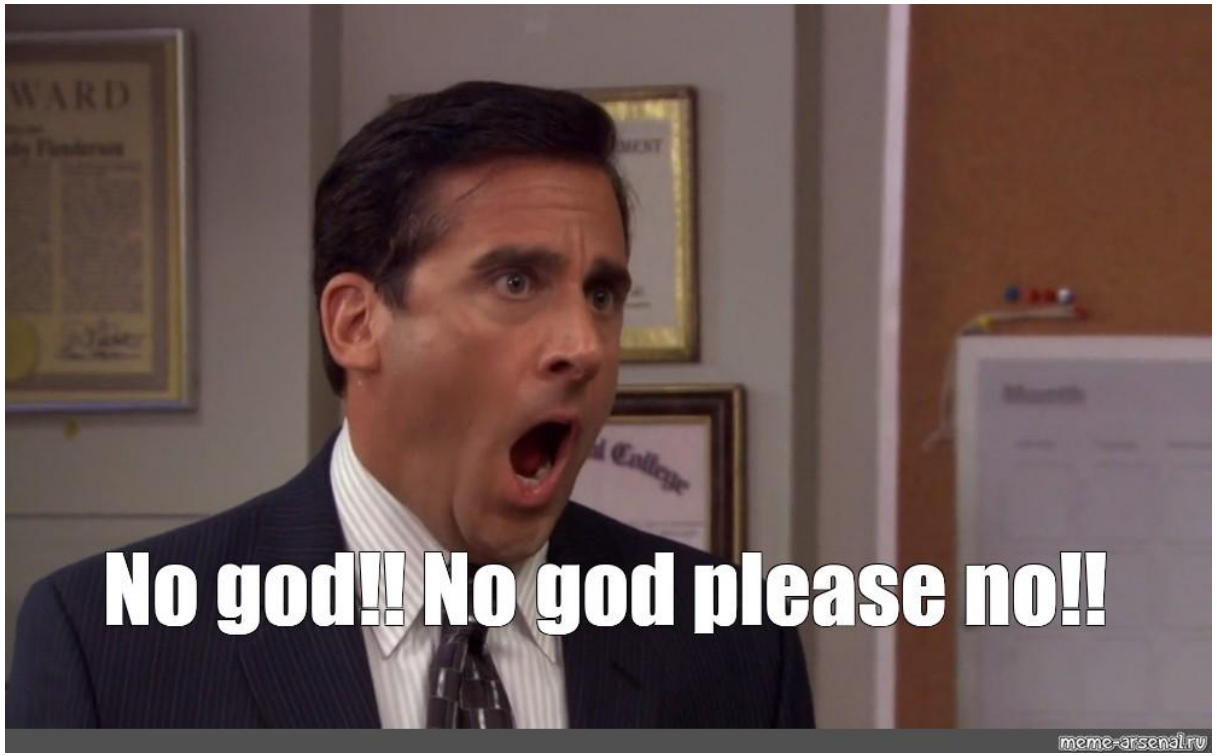
- VPS má verejnú IP a je dostupná z internetu
  - Pripojili sme sa predsa priamo pomocou ssh
- Ak VPS mám pre svoje interné potreby, tak to stačí
- Čo ak ale na nej chcem hostovať web appku?
  - Používatelia musia v browseri zadávať surové IP adresy (napr. 62.169.24.109)
  - Niektorí poskytovatelia dajú aj nejakú subdoménu
    - napr. entrance.eu-central-1.compute.amazonaws.com
- Potrebujeme vlastnú doménu (napr. entrance.sk)
  - Kupuje sa vždy na rok (.sk a .com asi 10 €, .eu asi 5 €)
  - VPS poskytovatelia zvyčajne umožňujú doménu dokúpiť
  - Môžem si doménu prenajať aj inde a nastaviť záznam na VPS
    - Dá sa ušetriť aj pár desiatok €

Nastavenie DNS

- U poskytovateľa cez webové rozhranie
- Treba nastaviť A záznamy na IP adresu našej VPS
  - AAAA záznamy v prípade IPv6
  - CNAME záznamy, ak už VPS má nejakú subdoménu od poskytovateľa
  - Môžeme si vytvárať **subdomény**
    - **auth**.entrance.sk na adresu servera, kde máme Keycloak
    - **api**.entrance.sk na SpringBoot backend
    - **app**.entrance.sk na náš React frontend
    - entrance.sk necháme pre prípadnú statickú web stránku
    - Ak všetky komponenty máme na 1 serveri, tak všetko bude ukazovať na tú istú IP

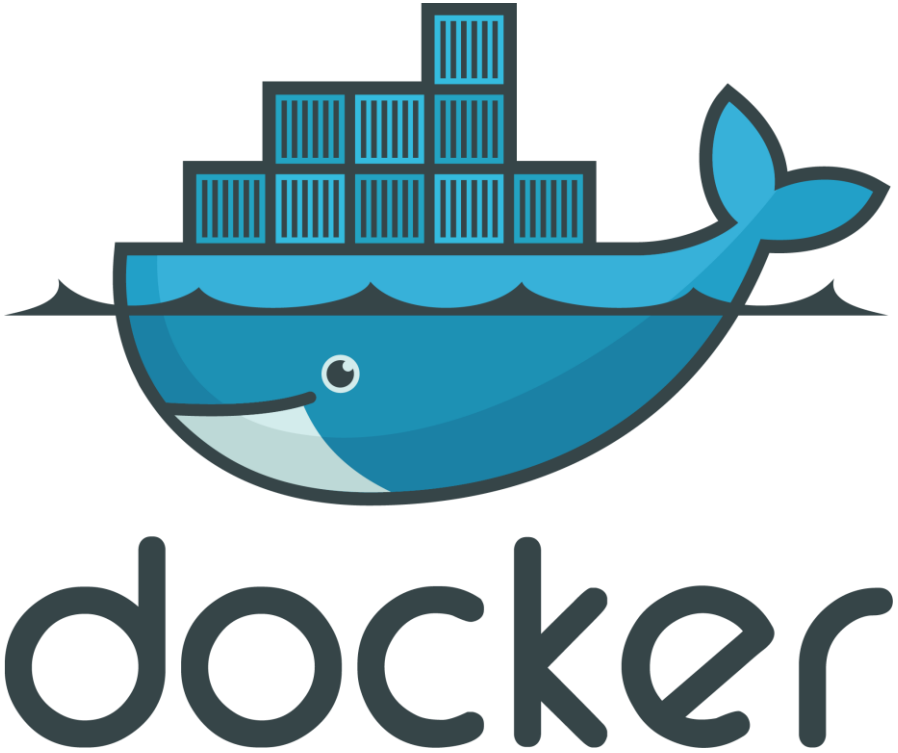
Upload appky na server

- Potrebujeme teraz dostať našu appku na náš VPS server
- MySQL+Keycloak môžeme rozbehať cez Docker
- ~~Naše zdrojáky môžeme zazipovať, uploadovať cez scp na server a tam kompilovať~~



Docker Hub

- Urobíme si free konto na hub.docker.com
  - Alebo na ľubovoľný iný OCI register (Quay.io, JFrog, ...)
  - Alebo si budeme hostovať vlastný OCI register
  - OCI – Open Container Initiative – otvorený štandard do ktorého sa Docker kontajnery časom vyvinuli
    - Podman – Open source alternatíva ku Dockeru pre OCI obrazy
- Kliknite vpravo hore na vaše konto
  - Account Settings/Security a vytvorte **access token** pre váš počítač
  - Vygenerovaný token sa vám ukáže iba raz, tak si ho starostlivo uložte
- Zostavíme obrazy appky cez Docker
  - docker build -t [hub-konto]/entrance-api:v0.1.0 –f Dockerfile-api .
  - docker build -t [hub-konto]/entrance-web:v0.1.0 –f Dockerfile-web .
- Nahráme obrazy na Docker Hub
  - docker login -u [hub-konto]
    - Ako heslo použite vygenerovaný **access token**
  - docker push [hub-konto]/entrance-api:v0.1.0
  - docker push [hub-konto]/entrance-web:v0.1.0
- Zvykom je aj zbuild-iť a push-núť aj obrazy s tagom „latest“
- Pozor: Pri free konte sú vaše obrazy dostupné verejne
  - Free kontá môžu mať iba 1 privátny repozitár



Produkčná konfigurácia našej appky

- Vytvorme si priečinky **prod_deployment/docker**
- Potrebujeme tam nový docker-compose.yaml a keycloak-init.json
  - Pozor na defaultné heslá pre DB a Keycloak
    - Musíme ich zmeniť na niečo iné a bezpečné
    - Snáď každý verejne dostupný server sa raz stane terčom kyber-útoku
  - docker-compose.yaml vieme „šablónovať“
    - Všetko v tvare ${SOME_VAR} bude nahradené premennou prostredia SOME_VAR.
    - Premenné prostredia môžeme zhrnúť v súbore **.env**
    - Takto môžeme mať v gite docker-compose.prod.yaml bez hesiel a iných nastavení jedinečných pre VPS (napr. jej doména)
    - Pri nasadení skopírujeme docker-compose.prod.yaml z gitu na VPS
      - Každá VPS bude mať svoj nový jedinečný súbor **.env** s premennými prostredia
  - keycloak-init.json nepodporuje šablónovanie z premenných prostredia
    - Musíme ho upraviť ručne, alebo si spraviť vlastný script na šablónovanie

Produkčné nastavenie Keycloak-u

Parametrizovanie Keycloak-u

- Keycloak bude dostupný z internetu
- NESMIEME používať implicitné nastavenia
  - Heslá ako „CHANGE_ME“
    - To nás potom hocikto vyhekuje
  - Zmeniť adresy BE a FE appiek z localhost na doménu nášho servera
- Náš súčasný init_keycloak.json nebude fungovať

Šablónovaný init_keycloak.json

prod_deployment/docker/init_keycloak.template.json

```
{
  ...
 "users": [
  {
    "username": "entrance-admin",
    ...
    "credentials": [
     {
       "type": "password",
       "value": "${ENTRANCE_ADMIN_PASS}",
       "temporary": false
     }
   ], ...
  }, ...
 ],
 "clients": [
  {
    "clientId": "entrance-app",
    "secret": "${KEYCLOAK_ENTRANCE_CLIENT_SECRET}",
    "rootUrl": "${APP_HOSTNAME}",
    "adminUrl": "${APP_HOSTNAME}",
   ...
   }, ...
 ]
}
```

- Na serveri definujeme tieto premenné prostredia
- Ak máme viac nasadení na viac serveroch
  - Každá organizácia potrebuje vlastný Entrance systém
  - Každý nasadenie ich bude mať unikátne
- Teda každý server musí mať unikátny init_keycloak.json

Šablónovací skript pre Keycloak

- Keycloak nepodporuje šablónovaný init_keycloak.json
  - Nebude sám vedieť dosadiť do neho správne premenné prostredia
- Pre každé nasadenie spustíme na serveri tento skriptík:

prod_deployment/docker/init_keycloak_generate.sh

```
# Export all variables from .env file
export $(sed -e '/^[[:space:]]*$/d' -e '/^[[:space:]]*#/d' .env | xargs)

# Use envsubst to replace all ${ENV_VAR} like strings in your file
envsubst < init_keycloak.template.json > init_keycloak.json
```

Produkčné nastavenie Dockera

(Voliteľné) Izolovanie sieťovej komunikácie

- Pre produkčné nasadenie cez Docker sa odporúča vzájomne izolovať sieťovú komunikáciu aplikácií
- Vytvoríme si siete v Dockeri (na serveri)

```
docker network create entrance-network
docker network create keycloak-network
docker network create traefik-network
```

prod_deployment/docker/docker-compose.yaml

```
networks:
  entrance-network:
    external: true
  keycloak-network:
    external: true
  traefik-network:
    external: true
```

Tu bude Entrance API a jeho DB

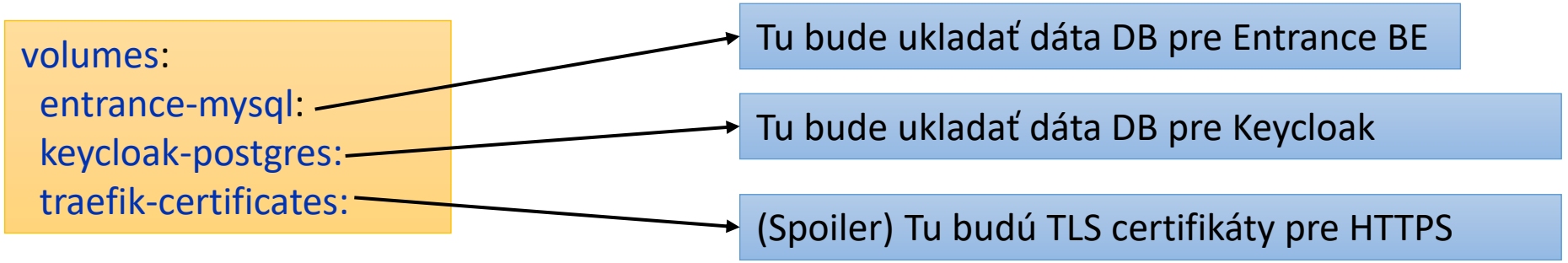
Tu bude Keycloak a jeho DB

Tu bude všetko, čo má byť dostupné z internetu
- Entrance API, Entrance Web, Keycloak
- Databázy už nie

Perzistencia dát kontajnerov

- Ak vymažeme kontajner (napr. MySQL) napr. cez docker compose down, tak jeho dáta sa stratia
- Na produkcii si toto nemôžeme dovoliť
  - Ak stratíme dáta (platiacich) zákazníkov, tak sme v keli
    - Nastavme si pri VPS denné zálohy, ale aj tak to nie je 100%
- Vytvoríme si perzistentné priečinky (zväzky/volumes), ktoré budú oddelené od kontajnerov
- Každý kontajner, ktorý potrebuje perzistenciu bude mať vlastný zväzok
  - T.j. najmä databázy
  - Na Linuxe sú zväzky štandardne v priečinku /var/lib/docker/volumes

prod_deployment/docker/docker-compose.yaml



Nasadenie viacero aplikácií

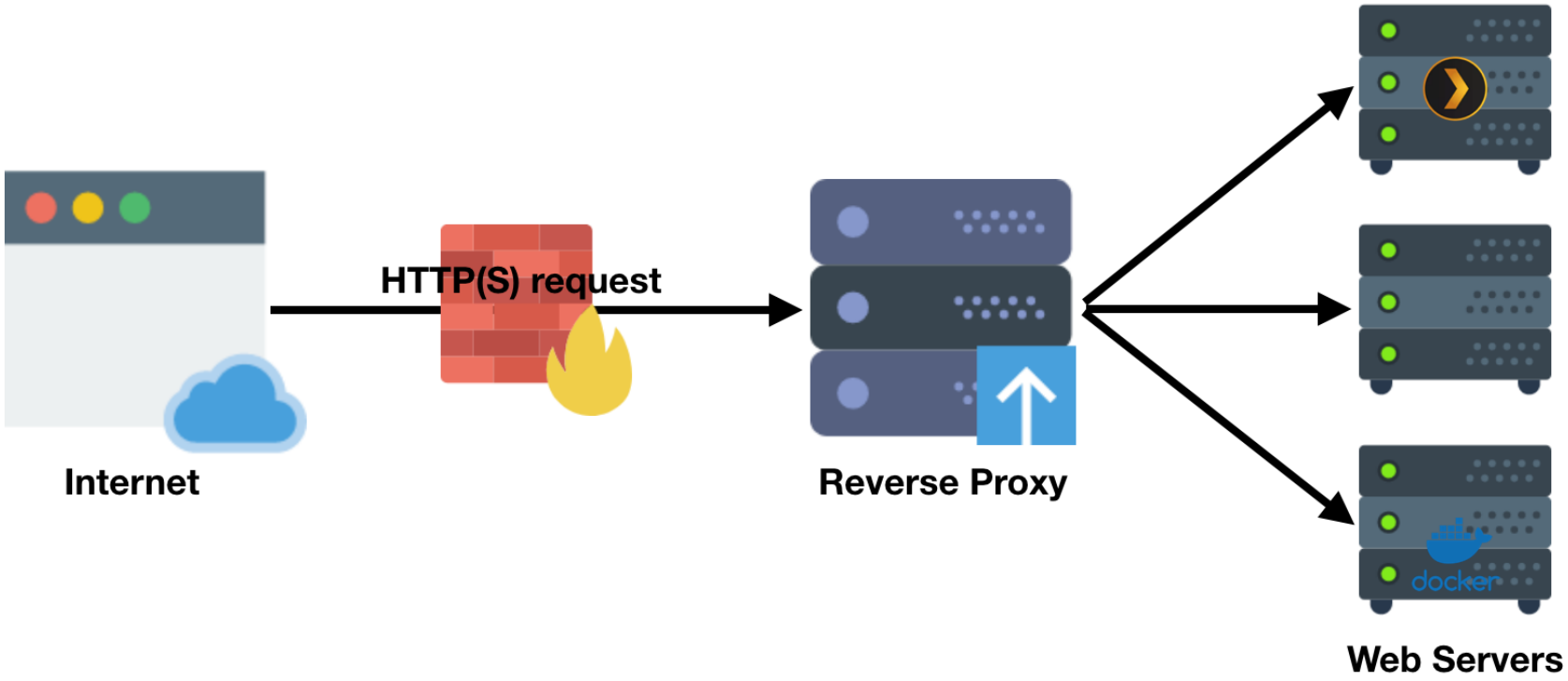
- Máme 3 aplikácie, ktoré chceme mať prístupné z internetu
- Pri lokálnom nasadení sme mali
  - Keycloak na porte 9080
  - Entrance API (SpringBoot) na porte 8080
  - Entrance Web (React) na porte 3000
- Pri produkčnom nasadení by všetky 3 mali byť na jedinom porte 443
  - Ale veď predsa nevieme spustiť dve serverové appky na jedinom porte
    - Či?

Nasadenie viacero aplikácií

prod_deployment/docker/docker-compose.yaml

```
entrance-api:
  image: vildibald/entrance-api:latest
  environment:
    CORS_ALLOWED_ORIGINS: ${API_HOSTNAME},${APP_HOSTNAME}
    DB_JDBC: jdbc:mysql://mysql:3306/entrance
    KEYCLOAK_URL: http://keycloak:8080
    KEYCLOAK_ADMIN_CLIENT_SECRET: ${KEYCLOAK_ADMIN_CLIENT_SECRET}
    KEYCLOAK_ADMIN_PASSWORD: ${ENTRANCE_ADMIN_REALM_PASS}
    DB_USERNAME: entrance
    DB_PASSWORD: ${ENTRANCE_MYSQL_PASSWORD}
  networks:
    - entrance-network
    - traefik-network
  depends_on:
    keycloak:
      condition: service_healthy
    mysql:
      condition: service_healthy
```

- Ubudlo nám mapovanie portu na hosta
  - To nám sprístupnilo appku z kontajnera na hosta
- Teraz chceme sprístupniť appku na celý internet (port 443)
  - To sa robí úplne ináč cez **reverzné proxy**



Reverzné proxy

Reverzné proxy

- Umožňuje, mimo iné, viacerým web aplikáciám zdieľať jeden port na jednej adrese
- Sprístupní web appku do inej siete
  - V našom prípade z internej Docker siete do celého internetu
- Pakety posiela na základe
  - Subdomény
    - **auth**.entrance.sk -> Keycloak na porte 9080
    - **app**.entrance.sk -> React na porte 3000
  - Cesty
    - entrance.sk/**auth** -> Keycloak na porte 9080
    - entrance.sk/**app** -> React na porte 3000
  - Záleží ako si to nastavíme
- Web aplikácie môžu fyzicky bežať na tom istom serveri, alebo na rôznych serveroch
  - Tie môžu byť aj izolované z internetu
- Reverzné proxy je jediným mostíkom medzi internetom a našimi appkami

- Asi najznámejší a najpoužívanejší
- Písaný v C
- Je to aj plnohodnotný webový server
- V benchmarkoch asi o 15% rýchlejší ako Traefik
  - V 99% prípadoch je to jedno
  - Web appky sú brzdené databázou a nie rev. proxy
- Skvelé GUI
  - NGINX Proxy Manager

- Novší a jednoduchší
- Písaný v Go
- Je to prevažne iba rev. proxy a load-balancer
- Lepšia integrácia s Dockerom
- Ľahšia konfigurácia
  - Najmä pri Docker Compose
- Iba základné „read-only“ GUI
- Integrovaná podpora pre Let’s Encrypt

HTTPS = HTTP + TLS

- Dnes je štandardom šifrované HTTP (na porte 443)
- Bez TLS dnes browser odmietne otvoriť webovú appku
  - Iba ak adresa je localhost
- ~~Potrebujeme webovým appkám nastaviť TLS certifikáty, ...~~
- ...alebo ešte lepšie
  - Necháme to na rev. proxy
  - Traefik má vstavanú podporu pre Let’s Encrypt včítane auto obnovenia certifikátu
    - Pri NGINX sa s tým treba trocha babrať

Rozbehanie Traefik-u

prod_deployment/docker/docker-compose.yaml

```
traefik:
  image: traefik:v3.0
  command:
    ...
    - "--entryPoints.web.address=:80"
    - "--entrypoints.web.http.redirections.entrypoint.to=websecure"
    - "--entrypoints.web.http.redirections.entrypoint.scheme=https"
    - "--entryPoints.websecure.address=:443"
    ...
    - "traefik.http.routers.keycloak.tls=true"
    - "traefik.http.routers.keycloak.tls.certresolver=letsencrypt"
    - "--certificatesresolvers.letsencrypt.acme.tlschallenge=true"
    - "--certificatesresolvers.letsencrypt.acme.email=${TRAEFIK_ACME_EMAIL}"
    ...
  volumes:
    - /var/run/docker.sock:/var/run/docker.sock
    - traefik-certificates:/etc/traefik/acme
  networks:
    - traefik-network
  ports:
    - "80:80"
    - "443:443"
  ...
```

Nastavenie pre TLS cez Let’s Encrypt

Traefik musí byť jediný kontajner s mapovaním portov na hosta.
A súčasne sú tieto porty otvorené firewallom na hostovi.
A súčasne má náš server verejnú IP.
Výsledkom bude, že Traefik vie komunikovať s internetom.

Pridanie appky do Traefik-u

prod_deployment/docker/docker-compose.yaml

```
entrance-api:
  ...
  labels:
    - "traefik.enable=true"
    - "traefik.http.routers.entrance-api.rule=Host(`${API_HOSTNAME}`)"
    - "traefik.http.routers.entrance-api.service=entrance-api"
    - "traefik.http.routers.entrance-api.entrypoints=websecure"
    - "traefik.http.services.entrance-api.loadbalancer.server.port=8080"
    - "traefik.http.routers.entrance-api.tls=true"
    - "traefik.http.routers.entrance-api.tls.certresolver=letsencrypt"
    - "traefik.http.services.entrance-api.loadbalancer.passhostheader=true"
    - "traefik.http.routers.entrance-api.middlewares=compresstraefik"
    - "traefik.http.middlewares.compresstraefik.compress=true"
    - "traefik.docker.network=traefik-network"
```

Povieme Traefiku, že Entrance BE chcem dostupný z internetu na tejto adrese (napr. api.entrance.sk)

Povieme Traefiku, že Entrance BE interne počúva v kontajneri na porte 8080

Vo výsledku bude bude naša appka dostupná z internetu

Šablóna pre šablóny

prod_deployment/docker/.env_template

```
# Host names - these should be changed to your (sub)domain
KEYCLOAK_HOSTNAME=https://auth.domain.com
API_HOSTNAME=https://api.domain.com
APP_HOSTNAME=https://app.domain.com
TRAEFIK_HOSTNAME=https://traefik.domain.com

# TLS - this should be changed to your email
TRAEFIK_ACME_EMAIL=your@email.com

# Secrets - these should be changed to random values
KEYCLOAK_DB_PASSWORD=this-should-be-random
MYSQL_ROOT_PASSWORD=this-should-be-random
ENTRANCE_MYSQL_PASSWORD=this-should-be-random
KEYCLOAK_ADMIN_PASSWORD=this-should-be-random
KEYCLOAK_ADMIN_CLIENT_SECRET=this-should-be-random
KEYCLOAK_ENTRANCE_CLIENT_SECRET=this-should-be-random
ENTRANCE_ADMIN_PASS=this-should-be-random
ENTRANCE_ADMIN_REALM_PASS=this-should-be-random
# Value should be `some_user_name:bcrypted_password`. Use e.g.
https://hostingcanada.org/htpasswd-generator/
TRAEFIK_BASIC_AUTH=admin:this-should-be-bcrypted
```

Nastavíme presne podľa domény servera

Mal by byť reálny email „vlastníka“ servera.

Defaultné heslá pre Keycloak. Mali by byť vždy unikátne a „náhodné“

Samotné nasadenie

- Nech máme teda server s verejnou IP, doménou a Docker-om
- Cez ssh/scp tam dáme z nášho priečinka prod_deployment/docker
  - docker-compose.yaml
  - init_keycloak.template.json
  - init_keycloak_generate.sh
  - .env_template ako .env **+ vyplniť až na serveri samotnom**
- Následne spustíme na serveri v presnom poradí
  - init_keycloak_generate.sh
  - docker compose up –d
- Voilá a appka fičí
- Pôjdeme na app.[nasa_domena] a môžeme sa kochať

Ďakujem za pozornosť