

Önálló laboratórium beszámoló

Távközlési és Médiainformatikai Tanszék

|  |  |
| --- | --- |
| Készítette: | **Fazekas Ferenc Albert** |
| Neptun-kód: | **IW4FRZ** |
| Ágazat: | **FELHŐ ALAPÚ HÁLÓZATOK** |
| E-mail cím: | **khhun12@gmail.com** |
| Konzulens(ek): | **Dr. Maliosz Markosz Dr. Simon Csaba** |
| E-mail címe(ik): | **maliosz@tmit.bme.hu simon@tmit.bme.hu** |

Téma címe: Szavazó alkalmazás megvalósítása kubernetes segítségével

Feladat

Ha nem az előtted lévő számítógéped erőforrásait használod, máris könnyen a felhőben találod magad! A műhelycsoport elsősorban a felhőben futó hálózati szolgáltatások (pl. mobil alkalmazások háttértámogatása vagy tartalomelosztó rendszerek) optimalizált létesítésére és működtetésére fókuszál. Ehhez a következő ismereteket lehet elsajátítani: a szerverek és a hálózatok virtulaizációja, virtualizációs környezetek (virtuális gép, konténer (pl. Docker) stb.) Fontos cél egy komplex szolgáltatást megvalósító virtuális komponensek automatizált telepítése, skálázása és működtetése, e menedzsment (orchestration) szoftver eszközök ismerete hasznos gyakorlati tudást biztosít, és a DevOps szemlélet is elsajátítható. A technológia megismerésén felül a megtanultak lehetővé teszik a gyakorlati alkalmazás során a felhő platformok telepítését, konfigurálását, üzemeltetését, illetve optimalizálását.

**2020/2021. 1. félév**

# A laboratóriumi munka környezetének ismertetése, a munka előzményei és kiindulási állapota

## 1.1 Bevezető

A témalabor során megismerkedtem több virtualizációs technológiával, többek között a Docker-rel és Kubernetes-el, melyek konténer alapú virtualizációt valósítanak meg. A tárgy során alulról felfelé haladtunk az architektúra működésének megismerésében. Az első felében a tárgynak a Docker alapjait tanultam meg ami a felhő alapú szolgáltatások mondhatni alapvető építőköve, a másodikban pedig a Kubernetes-ről tanultam, azaz, hogy hogyan kell konténereket egységbe szervezni, valamint szolgáltatásokat skálázhatóan és effektíven működtetni.

A feladatom ehhez kapcsolódóan, egy szavazó alkalmazás megvalósítása volt, Docker és Kubernetes segítségével.

## Elméleti összefoglaló

A virtualizáció már régóta az informatikában jelenlévő dolog, hiszen így hatékonyabban tudunk szolgáltatásokat, applikációkat megvalósítani. Alapvető célja az erőforrások hatékonyabb kihasználása, a fizikai réteg problémáinak kikerülése, és a platformfüggetlenség, absztakció.

Az első lépcsőfok a virtualizáció fejlődésébe a virtuális gépek alkalmazása volt, itt a host-OS hardveres és szoftverestámogatásával egy guest-OS-t futtattunk, egy külön operációs renszert annak minden hozományával együtt, itt az volt a probléma, hogy viszonylag lassú a beüzemelés, nagy overhead (filesystem, processzekm stb…), ezért nem erőforrás optimális egy célfeladat megoldására.

Erre találták ki később a konténerizációt, ahol az atomi egység az a „container”, ami process szintjén virtualizál, jóval kisebb erőforrás igénnyel, kényelmesebb beüzemeléssel, valamint sokkal jobb erőforráskihasználással. A container egy container image-ből áll elő, ez tárolja a különféle filesystem layer-eket és egyéb, fontos paramétereit.

Még egy lépcsőfoknak tekinthető ,például ha olyan szolgáltatást akarunk megvalósítani ami időben változó erőforrásigénnyel rendelkezik, hogy hogyan orchestraljuk az előző szinten létrehozott konténer szolgáltatásokat, mikor indítunk belőlük, hányat, milyen paraméterekkel, stb…, ezt hivatott megvalósítani a Kubernetes, amivel egy magasabb szintről tudunk konténereket menedzselni, itt az atomi egység az a „pod”, ami tartalmazhat több container imaget is. A Kubernetes bevezet számos egyéb funkciót is, mint például a service és deployment fogalmát. Amelyek nagyban megkönnyítik a magasabb szintről való menedzselést. Többek között számos beépített funkcióval rendelkezik, például beépített terhelés elosztó, ami szintén levesz némi terhet a fejlesztő válláról.

Valamint fontos része mind a Kubernetes-nek mind a Docker-nek az úgynevezett hálózati abszrakció, aminek segítségével egyszerűbb hálózatkezelést valósíthatunk meg a szolgáltatásainknak, pl.: webszerver példányok között a terheléselosztó automatikusan forwardolja a kintről érkező csomagokat.

# Az elvégzett munka és eredmények ismertetése

## 1.1 A munkám ismertetése

### Bevezető

A feladatom egy szavazó alkalmazás megvalósítása volt, ehhez Kubernetes-t és Docker-t használtam egy VMware virtuális gépen. Az előre kiadott minta projekt alapján[1]. 5 db entitás van jelen ebben. A voting-app ami egy python segítségével elkészített szavazó webfelület, itt tudunk szavazni. A redis, ami eltárolja Redis segítségével a bevitt szavazatot. A worker, ami a redis és a db közötti összeköttetésért felelős. A db, ami egy PostgreSQL adatbázis, a perzisztens adattárolásért felel. Valamint egy result-app, ami Node.js segítségével kiírja a bevitt szavazatokat. Feladat része még a 3 nem sima public image-t használó container image legyártása valamint saját docker repository-ba való feltöltése.

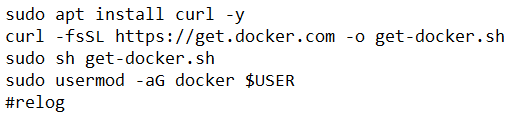
### Elképzelés

Az elképzelésem egy 5 podban működő rendszer volt, úgy, hogy minden pod 1 image-t tartalmaz, mivel ez szerintem így a leginkább átlátható. A megadott minta projektben lévő entitások közül a redis és a db az egyszerű public image-t használja, tehát azokkal Docker szinten nincs dolgom. A worker-t,a result-ot és a voting-app-ot viszont Docker segítségével a mintaprojekthez hasonló módon meg kell valósítani, buildelni kell belőlük image-eket. Ezek után a megfelelő paraméterezéssel a különböző image-ket deployment-be és service-be szervezni. Valamint tesztelni, hogy helyesen működik-e a megvalósított alkalmazás.

### A környezet telepítése

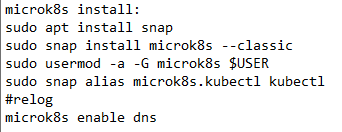
A legelső lépés a virtuális gép telepítése, ehhez VMware-t használva feltelepítettem egy Ubuntu18.04LTS linux operációs rendszert, 3GB RAM, 2 CPU core és 20GB háttértárral.

Ezek után szükséges a Docker telepítése amely az 1.ábrán látható parancsokkal valósítható meg.



*1. ábra: Docker telepítési parancsok*

Ezek után a Kubernetes-t kell telepíteni, amihez én Microk8s-t használtam. Ennek a telepítése a 2.ábrán látható parancsokkal valósítható meg. Az egyszerűbb kezeléshez még beállítottam egy aliast és adtam sudo jogot mindkét esetben. Valamint, a hálózat megfelelő működése érdekében engedélyeztem a dns-t.

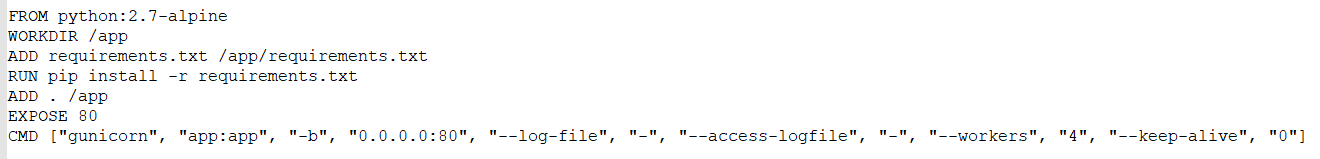


*2. ábra: Kubernetes környezet telepítése*

### Container image-k megvalósítása

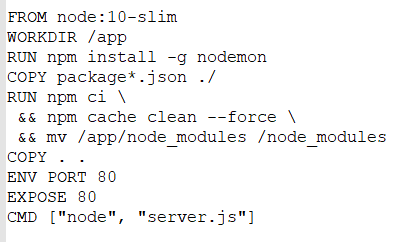
A környezet telepítése után a következő lépés a 3 container image (vote-app,result,worker) megvalósítása Docker segítségével. Itt a szükséges fileokat az image-k buildeléséhez a mintaprojektből vettem[1].

A vote-app Dockerfile a 3.ábrán látható, a megfelelő paraméterekkel és fileokkal. Python alap image-val, 80-as alapértelmezett portot használva(web).



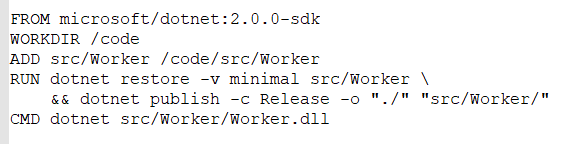
*3. ábra: vote-app Dockerfile*

A result Dockerfile a 4.ábrán látható, szintén egy user interface 80-as porttal(web), NodeJS-t használva az adatbázisból lekéri szavazatokat és megjeleníti a weblapon.



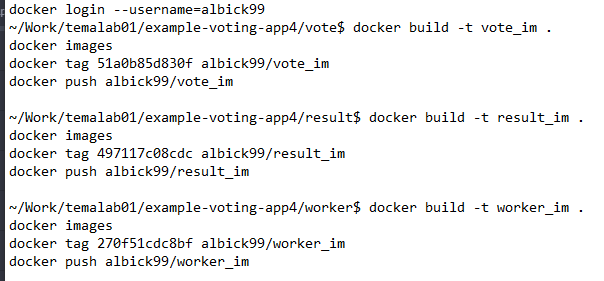
*4. ábra: result Dockerfile*

A worker Dockerfile az 5.ábrán látható, ez valósítja majd meg a kommunikációt a redis és a db között, .NET framework alapú.



*5. ábra: worker Dockerfile*

Így előálltak a szükséges Dockerfile-ok, mostmár csak buildelni kell őket és a saját dockerhub repositorymba feltölteni. A buildeléshez szükséges parancsokat a 6.ábra tartalmazza. Először be kell jelentkezni a saját repositoryba, majd egyenként buildelni és feltölteni a megfelelő cimkével ellátott image-ket. Ez után rendelkezésre állnak további felhasználásra.



*6. ábra: image build parancsok*

### Deployment és service megvalósítása

A deployment a dep.yaml file alapján megy végbe, mellékelve látható[2]. Itt definiáljuk 5db pod deploymentjét külön egy vote nevű namespace-be az könnyebb átláthatóság végett. A db sima „postgres:9.4”-es imaget használ, ez a perzisztens adatbázisunk, be van állítva külön, hol tárolja az adatokat a filerendszerünkben, valamint a szükséges környezeti változók át vannak adva neki, az 5432-es porton kommunikál. A redis egy „redis:alpine” imaget használ, 6379-es porton kommunikál, és szintén be van állítva nekü adattárolásra útvonal. A result a dockerhub repository-nkból a „result\_im” imaget használja, 80-as porton. A vote, az előbb elkészített „vote\_im” imaget használja, 80-as porton. A worker szintén az előre legyártott „worker\_im” imaget használja. Fontos megemlíteni, hogy most az egyszerűség kedvéért „replicas: 1” értéket használok mindenhol.

Már csak a service előállítása maradt, hogy működőképes legyen az alkalmazás, ennek a service.yaml file-ja mellékelve látható[3]. Itt létrehozzuk a serviceket. Hálózati beállításokat tekintve ClusterIP a worker,redis és db, tehát csak cluster-en belülről érhető el, mivel nekik csak egymással kell kommunikálni és nem is lenne biztonságok ha kívülről szükségtelenül elérhetőek lennének. A 2db webes felület(result,vote) viszont NodePort, tehát kívülről is elérhető. Az egymással való kommunikációra szolgálnak a megadott targetPort-ok, a port segítségével érthető el közvetlenül a service, ha ismerjük IP címét, és nodePort segítségével érhető el kívülről.

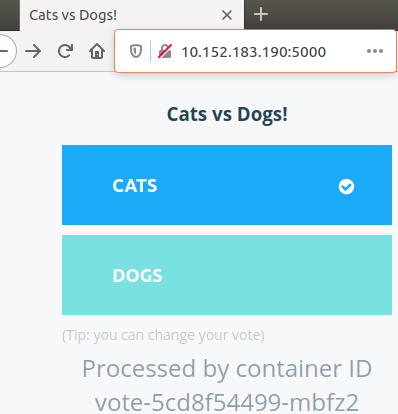
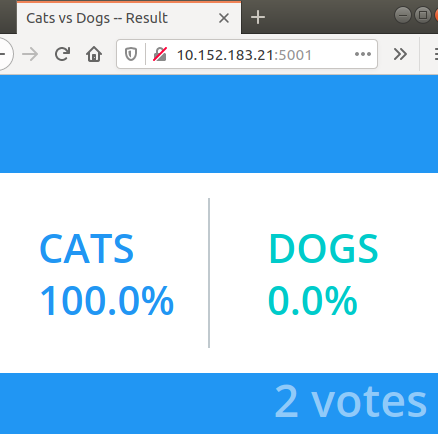
Ahhoz hogy működésre bírjuk a rendszerünk, az alábbi paranccsal el kell indítanunk, ami a 7.ábrán látszik.



*7. ábra: service és deployment indítása*

### Eredmények tesztelése

A „kubectl get po -n vote” paranccsal ellenőriztem, hogy az összes pod megfelelően elindult-e. Miután mind „running” lett , a „kubectl get services -n vote” paranccsal megnéztem a szavazó felület(vote) ip címét és egy böngészőben megnyitottam a service ip címét a service.yaml fileban definiált 5000-es porttal és a 8.ábrán látható eredményt kaptam, tehát a vote rész működik. Hasonló eredményre jutottam parancssori böngészővel is. A szavazat leadását követően a tényleges tesztje az alkalmazásunknak a result page volt, ami a 9.ábrán láthatóan megfelelően működött és valós eredményeket megfelelő formában adott vissza. Ebből arra a konklúzióra juthatunk, hogy a rendszer egészét vizsgálva végponttól-végpontig megfelelően funkcionál.

*8. ábra: vote felület 9. ábra: result felület*

## 2.2 Összefoglalás

A félév során sok új dolgot megtanultam a Docker és Kubernetes témakörébe, ezek segítségéve

Ebben a részben az *adott* *félévre vonatkozó, az Önálló laboratórium tárgy keretében elvégzett munka során* elért ***új*** eredmények ismételt, vázlatos, **tömör** összefoglalását várjuk, lehetőleg nem felsorolásként. Itt még egyszer ki lehet térni a leglényegesebb eredményekre, valamint a félév során felmerülő nehézségekre, de meg lehet említeni a továbbfejlesztési irányokat, lehetőségeket is.

Ezt a részt tagolható a következő pontok megválaszolásával:

* Mi volt az aktuális kérdés, **probléma,** amivel a félév során foglalkoztál?
* Mi a dolgozat **célja**, miért érdekes egyáltalán ezzel a problémával foglalkozni?
* Milyen **módszereket** használtál a probléma megoldása érdekében?
* Mik a legfontosabb **eredmények**?
* Milyen **következtetéseket** lehet levonni?

Ha valaki elolvassa ezt a részt, képet kell kapnia az egész dolgozatról!

Fontos, hogy az itt megadott sablontól el lehet térni, használata nem kötelező, csak segítséget jelenthet, viszont a fedőlap lehetőleg maradjon ugyanez és tartalmilag egyezzen meg a sablon irányelveivel. A beszámoló felépítésében nem érdemes eltérni a Bevezető - Féléves munka és eredmények bemutatása - Összefoglaló hármastól.

# Irodalom, és csatlakozó dokumentumok jegyzéke

## A tanulmányozott irodalom jegyzéke:

|  |  |
| --- | --- |
| [] | Umberto Eco, *Hogyan írjunk szakdolgozatot?,* Kairosz Kiadó, 2000, ISBN: 9639137537. |
| [] | Esterházy Péter, *Termelési-regény (Kisssregény)*, Magvető Könyvkiadó, 2004, ISBN: 9631423948. |
| [] | *Tájékoztató a Műszaki Informatika Szak önálló laboratórium tantárgyainak 2008/9. tanév I. félévi lezárásáról a BME TMIT-en* (VITMA367, VITMA380, VITT4353, VITT4330), http://inflab.tmit.bme.hu/08o/lezar.shtml, szerk.: Németh Felicián, 2008. november 5. Utolsó letöltés ideje: 2010-10-12 |

## Egyéb tartalmak jegyzéke: