**Bevezetés a Docker használatába**

**Docker**

A SpringBoot beépítetten tartalmaz Docker támogatást

= **op rendszer szintű virtualizáció;** hasonlít a klasszikus virtualizációra, azzal a különbséggel, hogy ez **kevésbé erőforrás igényes, nem kell a teljes op rendszert újra telepíteni,** minden egyes részét újra futtatni, hanem a **host op rendszernek** a **szolgáltatásait tudjuk igénybe venni**

*mire jó a virtualizáció?*

**különálló jól elkülönített környezeteket tudunk létrehozni; Docker esetén saját filerendszerrel és saját telepített szoftverekkel**

a létrehozott környezetek csak a jól meghatározott módon kommunikálnak egymással

*Gyakorlati példák arra, hogy mire jó a docker által biztosított virtualizáció:*

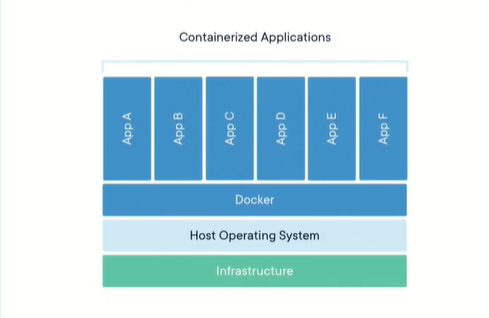
**ha fejlesztünk a saját gépes környezet eltérhet az éles környezettől** (az alkalmazásunk másképpen is működhet);de ha Dockert használunk, a saját gépünkön is Docker containerrel és az éles környezetben is docker containerben futtathatjuk az alkalmazásunkat -> így a két környezet nem tér el egymástól, az **éles környezetben nem fognak nem várt hibák előjönni**

**Ha egyszerre több alkalmazás fejlesztésében veszünk részt, nem kell azzal küzdeni, hogy a sajátgépre különböző fejlesztőeszközök különböző verziói mind legyenek feltelepítve** -> **helyette Docker környezet docker containerrel** a **Java egyik verziójával**, és egy másik környezetet használunk **másik** feltelepített Java **verzióval**

**Ha több számítógépből álló hálózatot szeretnénk reprodukálni**, azt is meg tudjuk tenni -> egy docker containerből elindítunk egy adatbázist, másikból egy alkalmazást, egy harmadikból egy elosztott clasht stb.

**microservice-ek esetén különösen fontos lehet a docker használat** -> nem kell egy alkalmazást feltelepíteni minden komponensével, hanem **kapunk egy docker containert, amit ha elindítunk, kapunk egy már kész környezetet**

**Archirtektúra**



**az infrastruktúra van alul, arra telepítjük az op rendszert és arra a Dockert;** utána már csak a különböző környezeteket/containereket futtatjuk -> nem kell még minden egyes container alá egy külön op rendszert telepíteni

**Docker felhasználási módja**

*mikor hasznos?*

**Saját fejlesztői környezetben** akarunk mindenféle erőforrást, backend servicet elindítani

saját fejlesztői környezettől való **izolációt** szolgálja

**Docker image** tartalmazza **a teljes környezetet és függőségeket** is

**Portabilitás** – **ugyanaz a container futtatható a sajátgépemen és éles környezetben is,** privát vagy publikus felhőből pl.

**További Docker komponensek**

sok komponensből áll;

**Docker Hub** – publikus szolgáltatás; **docker image-eket lehet itt tárolni**; a Java Maven világban a **Maven repositoryhoz hasonlítható**

**Docker Compose** – **egyszerre több Docker containert lehet vele indítani és kezelni** (pl. adatbázis, alkalmazásszerver és arra telepített alkalmazás és még valami harmadik komponens, akkor ezeket egy paranccsal mindent el tudjuk indítani)

**Docker Swarm** – eszköz, amivel **teljes hálózatot lehet felépíteni** és native cluster megoldást is tartalmaz (=a containereket clusterbe s lehet tenni)

**Docker Machine – távoli Docker környezetek üzemeltetéséhez hasznos modul,** amivel egy **cloudon futó Docker infrastruktúrához tud kapcsolódni** és ott futtatja le a parancsokat

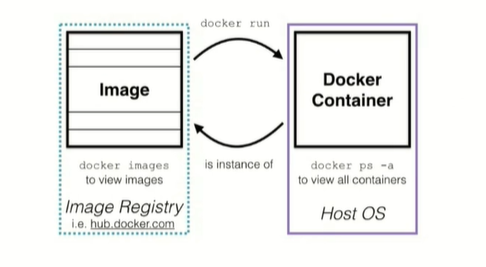
**Docker forgalmak**

**Image** – egy **fizikai állomány, állományok összessége** -> tartalmazza az **adott környezetet, annak filerendszerét és annak a feltelepített szoftvereit;** az image registryben helyezhető el, pl. a Dcoker hubon publikusan

**Container** – ha ezt az **image-t letöltjük sajátgépre és elindítjuk**, akkor létrejön egy **konkrét futó példány a saját gépünkön**

**Java megfelelője az osztály az imagere,** ami egy **publikus repositoryban van**, a **container pedig maga a példány, amit elindítunk**

**Imagek lekérdezése ’docker images’ paranccsal sajátgépen, a ’docker run’ indítja az imaget, és a futó containereket a docker paranccsal lehet kilistázni, ha -a kapcsolót is mögé rakjuk, akkor kilistázza az összes, leállított image-eket is**



**Docker konténerek - parancsok**

**docker version**

**docker run** hello-world (ha az image nem található a sajátgépen, a Docker automatikusan a Docker Hubhoz fordul és onnan letölti a lefrissebb verzióját és lemásolja a sajátgépre)

**docker run -p 8080:80 nginx** – az adott konténer portját tudjuk kivezetni a saját host számítógépünk valamilyen portjára; elindít egy webservert ami a 80as porton a konténeren belül el fogja kezdeni a kiszolgálást, ezt a 80as80as portot kivezetjük a host számítógépünk 8080as portjára -> ha hozzákapcsolódunk a sajátgépünk/localhost 8080asára, akkor az automatikusan a most elindított 80as portjához fog továbbítódni

**docker stop nginx** - ha elindítunk egy containert, egy hasht fog kapni, egy számot -> ezzel tudunk rá hivatkozni és pl. leállítani; elég csak beírni az első 3 számot (pl. docker stop 517)

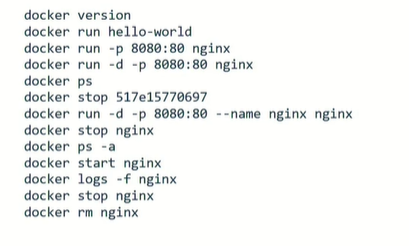
**docker --name** - hogy ne számokkal kelljen dolgozzunk, nevet is adhatunk az adott konténernek

**docker start xxx** – leállított konténer újraindítása

->run és start különbözik!!

**docker logs -f xxx** – csatlakozás a konténerhez és a konzolra üzenetek kiíratása, folyamatos pörgetéssel

**docker rm xxx** – álló konténer törlése



**Műveletek image-ekkel**

**docker images** – Dokcer által létrehozott imagek kilistázása

**docker rmi xxx** – az adott image kitörlése

**Bevezetés a Docker használatába – gyakorlat**

0.Docker feltelepítés

1.docker version

2.docker run hello-world -> fent van a hubon de sajátgépen nincs -> észreveszi és letölti majd elindítja

kiírt üzenet: Hello from Docker!

3.port kivezetése, hogy meg tudjuk szólítani a konténerbe futó webservert:

docker run -p 80:80 nginx -> lokálisan nincs meg, ezért letölti a docker hubról, létrehozza az image alapján a konténert és elindítja; a konténer folyamatosan fut, ezért nem adja vissza nekünk a vezérlést -> átmegyünk böngészőbe és beírjuk, hogy localhost:8080 -> a konténerben futó webserver fog minket már kiszolgálni, megkapjuk az nginx üdvözlő oldalát = a host 80as portja továbbítva van a konténer 80as portjára

4.paracssorosan CTRL C-vle a futás megszakítása -> a konténer nem áll le, csak visszakapom a parancssort, és a konzolra kiírt üzeneteket a továbbiakban már nem látom;

5. docker ps – futó konténerek kilistázása -> egy fut valamilyen egyedi azonosítóval, látható továbbá hogy az nginx image alapján került létrehozásra és hogy a 80as portja lett kivezetve a locahost 80asára

6. docker stop 5xx – átmegyünk böngészőbe és látjuk hogy leállt, már nem szolgál ki kéréseket

7. docker run -d -p 80:80 –name my-web nginx (-d – azonnal adja vissza a vezérlést)

8.docker ps

9. docker stop myweb

10.docker ps – már nem fut

11- docker ps -a -> a nem futókat is mutatja

12. docker start my-web -> újra indítás

13. docker ps -> mutatja megint

14. docker logs my-web

15. docker log my-web -f -> nem kapcsolódunk le a konténerről, és folyamatosan kiírásra kerülnek a logbejegyzések a böngészős refreshre

16. docker exec -it my-web bash -# kapunk egy promptot az adott konténeren belül, ahol linuxos utasításokat adtunk ki.., bejelentkeztünk a konténerbe majd exit, majd megint docker ps és docker stop

17. docker rm my-web -> konténer letörlése

18. docker ps -a -> már nem mutatja

19. docker images -> letöltött imagek kilistázása

20. docker rmi nginx -> van egy olyan konténer, ami ezt az imaget használja, ezért nem törölhető elsőre, ezért először a konténert kell törölni -> docker rm 581 majd újra docker rmi nginx, végül docker images az ellenőrzésre, hogy tényleg törlésre került-e

**Kérdések**

*Mi a docker és milyen előnyei vannak?*

*Mire jó a virtualizáció?*

*Milyen gyakorlati példái vannak a docker használatnak és milyen előnyökkel jár?*

*Hogy épül fel a Docker architektúra?*

*Milyen docker komponenseket ismersz?*

*Mi az a docker image?*

*Mi az a container?*

*Mik ezeknek a javas megfelelőjük?*

*Hogyan kérdezzük le az imageket?*

*Hogyna indítunk el konténert? Mi történik, ha az image nem található a sajátgépen?*

*Mire jó a –p kapcsoló?*

*hogy állítunk le futó konténert?*

*Hogy indítunk újra álló konténert?*

*Hogy adhatunk nevet a konténernek?*

*Hogy törlünk álló konténert?*

*Hogy törlünk imaget?*