1. **Szintfelmérő feladat**

Belépési kulcs

Az ssh kulcs elérése: https://pastebin.com/JfPPBmtt

(a raw változatot lehet kimásolni) -> Ha a kimásolt ssh kulcsot *hakap.pem* nek nevezzük el, akkor a belépés:

*ssh –i hakap.pem hakap@**<IP-cim>***

Fontos az SSH kulcs megfelelő jogosultságait beállítani:

chmod 600 hakap.pem

Csoportonként különböző klasztert, emiatt külön IP címet kell használni. Minden csoport számára két-két jump szervert indítottam, a terhelés elosztása érdekében.

A jump szerverről a *node-<x>* -re kell továbblépni, ahol **<x>** a csoporton belüli sorszám. Minden csoportnak feltöltöttem a sorszámot tartalmazó fájlt (NEPTUN kód szerint betűrendi sorrend alapján). Fontos: itt *root* felhasználói nevet kell használni.

*ssh root@node-**<x>***

Érdemes root-ként dolgozni. Megj. a node-okra ubuntu felhasználói névvel is be lehet lépni.

A node-okra nincs telepítve a Docker. Telepíteni pl. az alábbi Git repóból elérhető szkripttel lehet:
git clone https://github.com/suhard/hakap.git
cd hakap
chmod u+x install_docker.sh
./install_docker.sh

Windows-os használat

Windows10-es gépeken már a konzol indítása (*cmd*) után ugyanúgy lehet használni az ssh parancsot, mint a Linuxban.

Alternatíva: Windows-os gépeken sokan a PuTTy alkalmazást használják SSH belépésre.

A putty.exe letöltése és indítása után érdemes egy dedikált "*session*"-t létrehozni, és a baloldali menüsorban a "*Connection -> SSH -> Auth*" almenüre kattintva megjelennek a részletes opciók ("*Options controlling SSH authentication*"). A fenti PPK kulcsot ott kell "betölteni", a "*Private key file for authentication*"/"*Browse*" gombra kattintva.

Aki saját laptopról dolgozik, és azon Windows fut, az alternatíva egy VirtualBox installálása és abban indítani egy Linux VM-et.

Csoportokhoz rendelt IP nyilvános IP címek

G1 csoport

128.110.223.36
128.110.223.37

G2 csoport

128.110.223.46
128.110.223.47

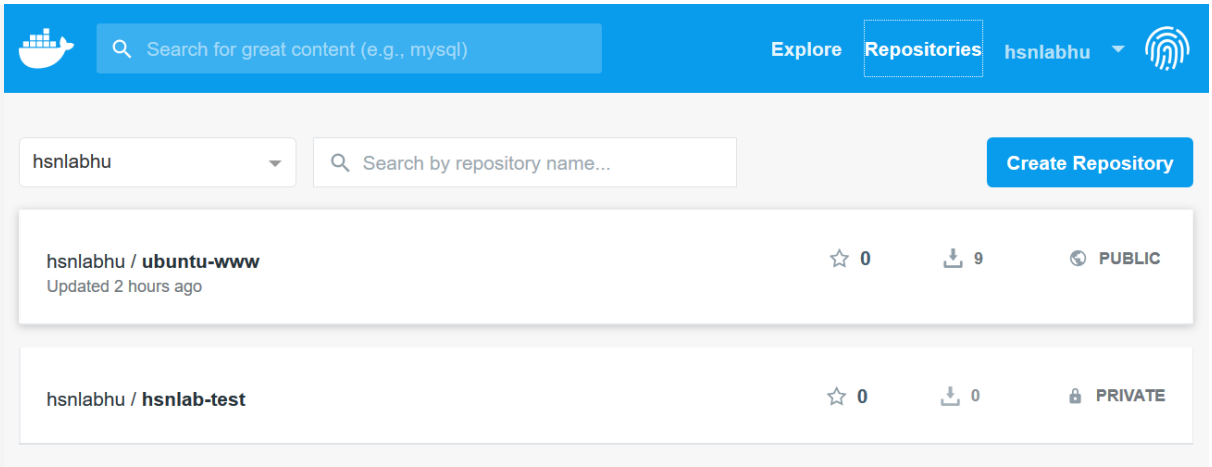
G3 csoport

128.110.223.56
128.110.223.57

Docker registry használat

A nyilvános Docker registry-n létre kell hozni egy fiókot: https://hub.docker.com/

DockerHub-on a regisztráció után létre kell hozni egy saját “repository”-t is!
Az alábbi példában a “hsnlabhu” a fiók neve (a docker hub id, vagy docker_id).
Két repo van létrehozva: “ubuntu-www” és “hsnlab-test”



Az ingyenes regisztrációhoz egy privát (titkos) repo jár, valamint korlátlan számú nyilvános repó (egy korábban privátnak beállított repo-t bármikor publikussá tehetünk és vissza).

Csak egy létező, már létrehozott repository-ba lehet feltölteni (push) az image-t. Tulajdonképpen a “repository” neve lesz az a név, amire a lokális, feltöltésre kész image-t át fogjuk nevezni.

1. Szintfelmérő feladat leírása

A "szolgáltatás" működése során egy Flask környezetben megírt egyszerű alkalmazás fogadja a beérkezett kéréseket és válaszként megadja, hogy hányszor hívták meg. Az egyetlen állapotváltozó (a lekérdezések száma) egy külön Redis adatbázisban van nyilvántartva.

A példában a "web frontend" és az "adatbázis backend" két külön konténerben fut, ez a két konténer a "web" és a "redis". A szolgáltatás helyes működése érdekében a web frontendnek tudnia kell a redis backendről, valamint a két konténer össze kell legyen kötve.

A két konténer közti elérést ti kell megoldanotok! (Február 28-i előadásban ismertetett módszereket használhatjátok)

A „szolgáltatás" futtatásához szükség lesz a két képfájlra (image), amelyekből létre lehet hozni a 2 konténert. A redis image-ként a standard DockerHub-on elérhető image-t használunk, de a webes frontend image-t a compose parancs kiadása után, egy dedikált Dockerfile alapján hozza létre a rendszer.

A teljes feladat forrásfájljai, a szolgáltatás logikáját jól leírja ez a hivatkozás:
https://docs.docker.com/compose/gettingstarted/
Ami FONTOS, hogy ezen a linken a példa docker-compose segítségével oldja meg a két konténer összekapcsolását, a szintfelmérő során nem ezt az eljárást kell követni!

Ehhez egyrészt kell egy segédfájl, amely megadja, hogy milyen python lib-eket telepítsen fel.

---
requirements.txt

```
flask
redis
```

Másrészt szükség lesz arra a Python alkalmazásra, amely a webes frontendet valósítja meg.
---
app.py

```
import time

import redis
from flask import Flask

app = Flask(__name__)
cache = redis.Redis(host='redis', port=6379)

def get_hit_count():
    retries = 5
    while True:
        try:
            return cache.incr('hits')
        except redis.exceptions.ConnectionError as exc:
            if retries == 0:
                raise exc
            retries -= 1
            time.sleep(0.5)

@app.route('/')
def hello():
    count = get_hit_count()
    return 'HAKAP class hello! I have been seen {} times.\n'.format(count)

if __name__ == "__main__":
    app.run(host="0.0.0.0", debug=True)
```

Ha megvannak ezek a fájlok, akkor az alábbi Dockerfile alapján lehet a „web” konténernek az image-t.

Dockerfile

```
FROM python:3.4-alpine
ADD . /code
WORKDIR /code
RUN pip install -r requirements.txt
CMD ["python", "app.py"]
```

FONTOS: Lehet, hogy érdemesebb onnan, a fenti linkről kimásolni a forrásfájlokat, mert ebből a PDF-ből kimásolva hibás lesz.

A docker host-on használt port ne a 80-as port legyen!