2021.12.01.

Horváth Dávid - PSZQPH

Óbudai egyetem

Projektmunka III.

Féléves feladat dokumentáció

Tartalomjegyzék

[Cloudkarafka 2](#_Toc89347771)

[Első feladat - Batch alapon relációs tábla 3](#_Toc89347772)

[Consumer 3](#_Toc89347773)

[Producer 6](#_Toc89347774)

[DBprocessor 8](#_Toc89347775)

[DBviewer 10](#_Toc89347776)

[Második feladat – Streaming 11](#_Toc89347777)

[Consumer 11](#_Toc89347778)

[Producer 13](#_Toc89347779)

[Dataviewer 15](#_Toc89347780)

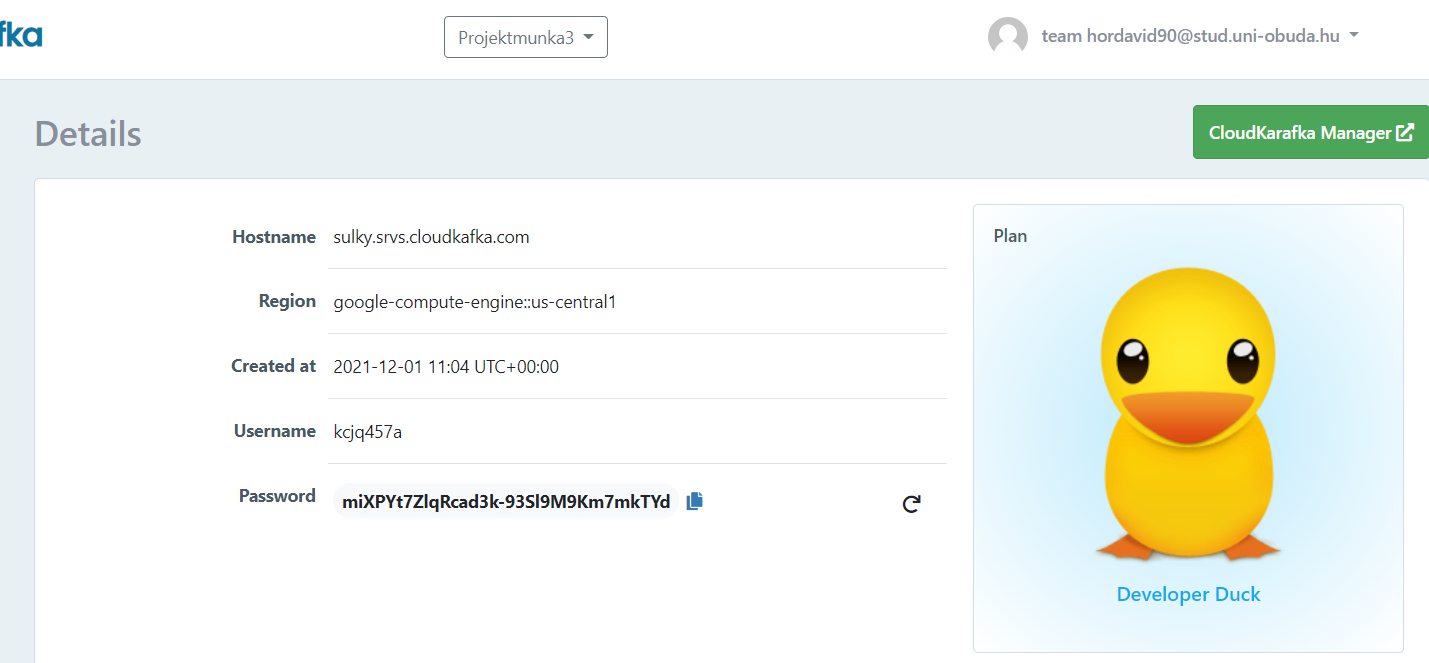
[RealtimeProcess 15](#_Toc89347781)

[Harmadik feladat – XML/JSON feldolgozás 18](#_Toc89347782)

[Github 18](#_Toc89347783)

# Cloudkarafka

Mielőtt belekezdtem volna a féléves feladatba, regisztráltam az ingyenes cloudkarafka oldalra, ahol sikeres regisztráció után csináltam egy új instance-t, így a pruducer és a consumer tud egymással kommunikálni.



A képen szöveg látható

Automatikusan generált leírás A képen szöveg látható

Automatikusan generált leírás

# Első feladat - Batch alapon relációs tábla

## Consumer

Első lépésként fel kellett telepíteni a kafka könyvtárat, majd beimportálni a dependenciákat. A képen szöveg látható

Automatikusan generált leírás

A topicon keresztül lehet begyűjteni az adatot, fel tud iratkozni egy cosumer egy vagy több topicra is.

A topics értéknek a cloudkarafkában kapott topics nevet írjuk.

Bootstrap.servers: a szerverek portszáma mindig a 9094, a szervereket szintén a cloudkarafka oldalon lehet megtalálni.

Session.timeout.ms: megszünik a kapcsolat max 6 másodpercet várakozik.

Default.topic.config: ha nem akarjuk az elejétől lekérni az adatokat, csak a rácsatlakozástól akarjuk lekérni az adatokat, ezért van a „largest”.

Ezt követik bizonyos biztonsági protokollok.

Van egy kulcs érték pár. UTF-8-as kódolással.

Ezután található a credentials. A felhasználónév és a jelszó, amiket szintén a cloudkarafkán érünk el, mindig egyedi.

Utána ezt el kell indítani. Létrehozzuk a consumer példányát, átadjuk neki a configot.

Subscribe segítségével feliratkozunk a topicra.

A képen szöveg látható

Automatikusan generált leírás

Ezt követően ismét beimportálunk pár dependenciát, ami szükséges még. A képen szöveg látható

Automatikusan generált leírás

Létrehozunk egy helyi szervert az 12349-es porton. Azok a kliensek akik erre felcsatlakoznak, nekik fogjuk küldeni az adatokat a consumer segítségével. Összegyűjti az adatokat a karafkától és kiküldni a feliratkozóknak.

Az s.listen, figyeli a portot és ha valaki csatlakozik elkezdi küldeni az adatokat neki.

Van egy végtelen ciklus, ahol várakozás történik. Az s.accept blokkolja a program futását, addig amíg nem csatlakozik fel a kliens. Ha csatalkozott akkor figyelni fogja 2,5 másodpercenként az adatokat, ha nem kap semmit kiírja, hogy nem jött semmi. Más esetben kiszedi a consumertől megadott adatokat, ellenőrzi és kiírja.

A képen szöveg látható

Automatikusan generált leírás A képen szöveg látható

Automatikusan generált leírás

## Producer

Első lépéként fel kellett telepíteni pár könyvtárat pyhtonba, majd pár dependenciát beimportáltam. A képen szöveg látható

Automatikusan generált leírás A képen szöveg látható

Automatikusan generált leírás

Ezt követően jön a topicos rész, ami majdnem megegyezik a consumer részben találhatóval. Az a különbség a kettő között, hogy a producernél nincs szükség group id-ra, mivel az adatokat mindenkinek kiküldni akik erre a topicra rá vannak csatlakozva. A képen szöveg látható

Automatikusan generált leírás

Van a végtelenített while ciklus. Csatlakozik a coinos apira, így json formában megkapja az adatokat. Átalakítja a kérést úgy, hogy lekérjük a coinnak az azonosítóját, a sorszámát, a nevét, az értékét és a változását az utolsó 24 órában.

Ha 200-as a státuszkód, a válasz, akkor a response-t átalakítjuk DataFrame-é. Minden soron végigmegy és megadja az egyes sorok közül miket szedjen ki, amiket mi megadtunk. Végén átalakítja listává. Végigmegyünk for ciklussal a kapott elemeken. A p.produce fogja szolgáltatni az adott topicra az adatot.



## DBprocessor

Létrehozunk egy táblát a szükséges adatokkal. A képen szöveg látható

Automatikusan generált leírás

Ezt követően beimportálunk pár szükséges dependenciát. A képen szöveg látható

Automatikusan generált leírás

Beolvassuk az adatokat. A képen szöveg látható

Automatikusan generált leírás

A vesszőknél fogva elválasztjuk a stringeket. A képen szöveg látható

Automatikusan generált leírás

Átalakítjuk a szükséges adatokat. A képen szöveg látható

Automatikusan generált leírás

A képen szöveg látható

Automatikusan generált leírás

## DBviewer

Kilistázzuk a tábla tartalmát. A képen asztal látható

Automatikusan generált leírás

A képen asztal látható

Automatikusan generált leírás

# Második feladat – Streaming

## Consumer

Első lépésként fel kellett telepíteni a kafka könyvtárat, majd beimportálni a dependenciákat. 

A topicon keresztül lehet begyűjteni az adatot, fel tud iratkozni egy cosumer egy vagy több topicra is.

A topics értéknek a cloudkarafkában kapott topics nevet írjuk.

Bootstrap.servers: a szerverek portszáma mindig a 9094, a szervereket szintén a cloudkarafka oldalon lehet megtalálni.

Session.timeout.ms: megszünik a kapcsolat max 6 másodpercet várakozik.

Default.topic.config: ha nem akarjuk az elejétől lekérni az adatokat, csak a rácsatlakozástól akarjuk lekérni az adatokat, ezért van a „largest”.

Ezt követik bizonyos biztonsági protokollok.

Van egy kulcs érték pár. UTF-8-as kódolással.

Ezután található a credentials. A felhasználónév és a jelszó, amiket szintén a cloudkarafkán érünk el, mindig egyedi.

Utána ezt el kell indítani. Létrehozzuk a consumer példányát, átadjuk neki a configot.

Subscribe segítségével feliratkozunk a topicra. A képen szöveg látható

Automatikusan generált leírás

Ezt követően ismét beimportálunk pár dependenciát, ami szükséges még. A képen szöveg látható

Automatikusan generált leírás

Létrehozunk egy helyi szervert az 12346-es porton. Azok a kliensek akik erre felcsatlakoznak, nekik fogjuk küldeni az adatokat a consumer segítségével. Összegyűjti az adatokat a karafkától és kiküldni a feliratkozóknak.

Az s.listen, figyeli a portot és ha valaki csatlakozik elkezdi küldeni az adatokat neki.

Van egy végtelen ciklus, ahol várakozás történik. Az s.accept blokkolja a program futását, addig amíg nem csatlakozik fel a kliens. Ha csatalkozott akkor figyelni fogja 1 másodpercenként az adatokat, ha nem kap semmit kiírja, hogy nem jött semmi. Más esetben kiszedi a consumertől megadott adatokat, ellenőrzi és kiírja. A képen szöveg látható

Automatikusan generált leírás

## Producer

Első lépéként fel kellett telepíteni pár könyvtárat pyhtonba, majd pár dependenciát beimportáltam. A képen szöveg látható

Automatikusan generált leírás

A képen szöveg látható

Automatikusan generált leírás

Ezt követően jön a topicos rész, ami majdnem megegyezik a consumer részben találhatóval. Az a különbség a kettő között, hogy a producernél nincs szükség group id-ra, mivel az adatokat mindenkinek kiküldni akik erre a topicra rá vannak csatlakozva. A képen szöveg látható

Automatikusan generált leírás

Van a végtelenített while ciklus. Csatlakozik a coinos apira, így json formában megkapja az adatokat. Átalakítja a kérést úgy, hogy lekérjük a coinnak az azonosítóját, a sorszámát, a nevét, az értékét és a változását az utolsó 24 órában.

Ha 200-as a státuszkód, a válasz, akkor a response-t átalakítjuk DataFrame-é. Minden soron végigmegy és megadja az egyes sorok közül miket szedjen ki, amiket mi megadtunk. Végén átalakítja listává. Végigmegyünk for ciklussal a kapott elemeken. A p.produce fogja szolgáltatni az adott topicra az adatot. A képen szöveg látható

Automatikusan generált leírás

A képen asztal látható

Automatikusan generált leírás

## Dataviewer

A képen szöveg látható

Automatikusan generált leírás

## RealtimeProcess

Első lépésként itt is beimportáljuk a dependenciákat. A képen szöveg látható

Automatikusan generált leírás

A képen szöveg látható

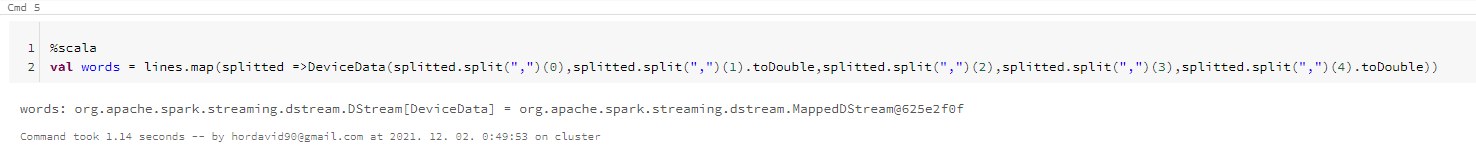
Automatikusan generált leírás

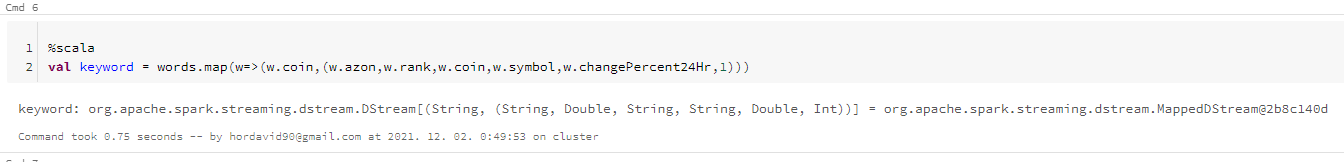
A képen szöveg látható

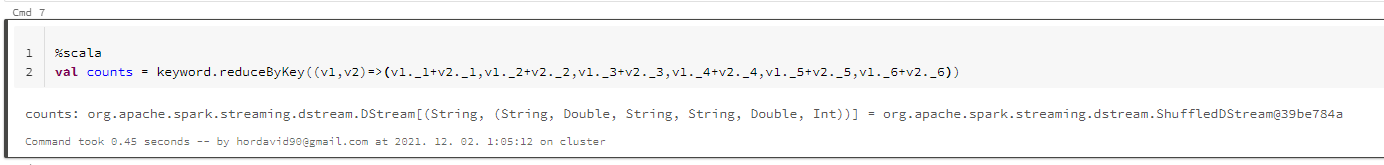
Automatikusan generált leírás

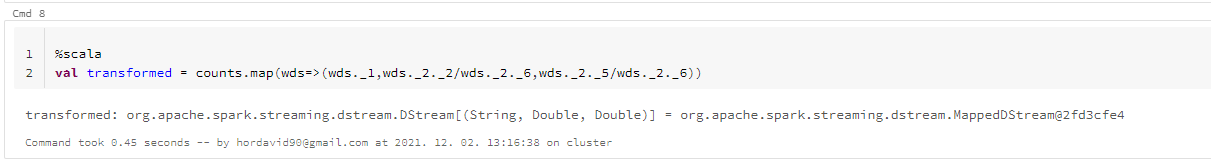
A képen szöveg látható

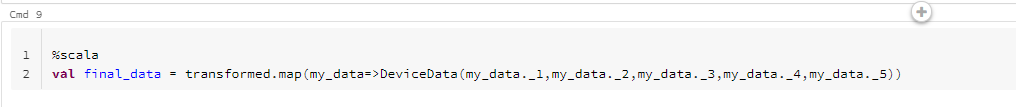
Automatikusan generált leírás







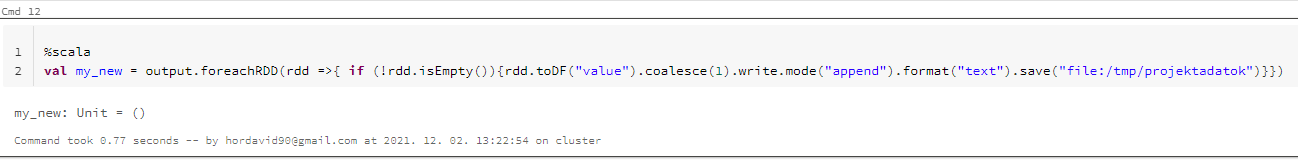




A képen szöveg látható

Automatikusan generált leírás





# Harmadik feladat – XML/JSON feldolgozás

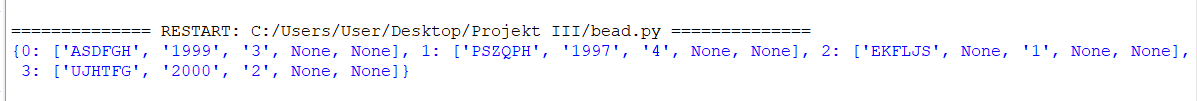
Első lépésként letöltöttem a python 3.10.0-ás verziójátA képen szöveg látható

Automatikusan generált leírás

Ezt követően importáltam a szükséges dolgokat. A képen szöveg látható

Automatikusan generált leírás

Beolvassuk az XML fájlt és kigyűjtsük az adatokat az XML-bőlA képen szöveg látható

Automatikusan generált leírás

A képen szöveg látható

Automatikusan generált leírás

# Github

Github link: <https://github.com/hordavid90/Projekt3.git>