APACHE SPARK KLASZTER PYTHONNAL FELÉPÍTÉSI ÚTMUTATÓ

MTA Cloud felhasználói számára készült Apache Spark klaszter felépítési útmutató v1.0

MTA Cloud csapat
info@mta.cloud.hu

Tartalom

Áttekintés

Az Apache Spark egy gyors és általános célú klaszter keretrendszer. Magas szintű API-kat biztosít Java, Scala, Python és R programnyelvekhez. Továbbá számos magas szintű eszközt támogat, többet között a Spark SQL-t a strukturált adatfeldolgozáshoz, MLlib-et a gépi tanuláshoz, GraphX-et a gráf feldolgozáshoz, és Spark Streaming-et a nagy mennyiségű adatok valós idejű feldolgozásához. További információkért látogasson el az Apache Spark hivatalos weboldalára.

Az Apache Spark klaszter a HDFS-el (Hadoop Distributed File System) együtt a Big Data és a a gépi tanulási alkalmazások egyik legfontosabb eszköze, amely lehetővé teszi a nagy adatállományok párhuzamos feldolgozását több virtuális gépen, amelyek a Spark Workerek. Azonban, egy Spark klaszter létrehozása a HDFS-el a felhőben nem egyszerű, a felhő rendszerek és az Apache Spark architektúrájának mély ismeretét igényli. Azért, hogy a kutatókat megóvjuk ettől a munkától, létrehoztuk és közzétettük azokat a szükséges infrastruktúra leírókat, amelyek segítségével az Occopus automatikusan építi a Spark klasztert, a felhasználó által megadott Workerek számával. A Spark egy "MLlib" nevű speciális könyvtárat biztosít a gépi tanulási alkalmazások támogatására. Hasonlóképpen, az R-orientált Spark környezethez, kifejlesztettük az infrastruktúra-leírókat a gépi tanulási környezet létrehozásához a felhőben. Itt, a programozási nyelv a Python és a felhasználói programozási környezet a Jupyter Notebook. A teljes gépi tanulási környezet a következő összetevőkből áll: Jupyter, Python, Spark és HDFS. Ezt a gépi tanulási környezetet az Occopus automatikusan építi ki és a Spark Workerek számát a felhasználó határozhatja meg.

Ez a bemutató egy teljes Apache Spark infratstruktúrát hoz létre, amely integrálva van a HDFS, a Python és a Jupyter Notebook-al. Tartalmaz egy Spark Master csomópontot és Spark Worker csomópontokat, amelyek számát felfelé vagy lefelé lehet skálázni.

Jellemzők:

- kétféle csomópont létrehozása kontextualizáció által
- egészség ellenőrzés használata előre definiált porttal szemben (health check)
- skálázási paraméterek használata a Spark Worker csomópontok számának korlátozására

Előfeltételek:

- felhő elérése valamilyen Occopus kompatibilis felületen (pl. EC2, Nova, OCCI stb.)

    - Az Occopus eszköz képes az MTA Cloud-on virtuális gépeket indítani, ehhez a nova interfész használatát ajánljuk (ennek címe a felületen megtekinthető: Compute/Access & Security / API Access / Identity)
    - **Fontos: az Occopus eszköz használata jelenleg csak az MTA Cloud SZTAKI ágán működik!**
- a célfelhő tartalmaz egy felhőalapú támogatással rendelkező 16.04 alapú Ubuntu OS képfájlt
    - Ez a képfájl a Compute/Images/Public/Ubuntu 16.04 LTS Cloud image alatt megtalálható

Tehát ha Ön tagja egy, az MTA Cloud SZTAKI-ági béli projektnek, akkor lehetősége nyílik az Occopus eszköz telepítése után Spark infrastruktúra létrehozására.

Lépések

Occopus telepítése

Az automatikusan felépülő Apache Spark architektúra kiépítése az Occopus eszköz segítségével fog megvalósulni, ezért elsőként telepítenünk kell az Occopus eszközt. Az Occopus orkesztrációs eszközt csupán egyetlen parancs segítségével is telepíteni tudja. A telepítéshez szükséges lépésekről, illetve az Occopus-ról bővebben az alábbi linken tájékozódhat. Javasoljuk, hogy indítson el egy Ubuntu alapú virtuális gépet az MTA Cloud-ban, majd oda telepítse az Occopus eszközt.

Letöltés

Az Occopus leírók alapján működik. Ezen leírók alapján fogja az Occopus felépíteni a cél felhőben az infrastruktúrát. A Spark klaszter telepítéséhez szükséges leírókat elkészítettük a felhasználók számára. Ezeket az alábbi linkről lehet letölteni: tutorial.examples.spark-cluster. Az Occopus-t futtató virtuális gépre telepítsük a leírókat.

Megjegyzés: Ebben bemutatóban nova felhő erőforrásokat használunk (az alap bemutató részben található nova oktatóanyagok alapján). Ugyanakkor nyugodtan használhat bármilyen Occopus – kompatibilis felhő erőforrást a csomópontok számára, de javasoljuk, hogy az összes csomópontot azonos felhőben építse fel.

1. Erőforrás szekció kitöltése

Nyissa meg a nodes/node_definitions.yaml fájlt és szerkessze a csomópontok „node_def:”címkével ellátott erőforrás szekcióját:

- Válasszon ki egy Occopus kompatibilis erőforrás plugin-t
    - Ez az MTA Cloud esetében a bemutatóban szereplő erőforrás plugin-nal azonos, nova erőforrás lesz
- Meghatározhatja a plugin attribútumainak megfelelő listát
- Követheti a bővítmény attribútumainak értékeinek összegyűjtésére vonatkozó segítséget
- A plugin-hez tartozó erőforrás sablonokhoz találhat erőforrás plugin bemutatókat is

Fontos, hogy indítás előtt a node definíciós fájlt személyre kell szabnia a felhasználónak. Ebben a fájlban megadjuk azokat az erőforrás azonosítókat, amelyeket használni fogunk, pld.: projekt ID, virtuális gép mérete stb. Ezeket az azonosítókat nem tudjuk megadni a felhasználó helyett, azonban azokat az MTA Cloud felületéről könnyedén össze lehet szedni. Ehhez részletes segítség ezen linken, vagy az alábbi dokumentációban található. A példában található letölthető csomag a Nova plugin erőforrás sablont tartalmazza (ezt használjuk az MTA Cloud használatához is).

Fontos: Ne módosítsa a kontextualizáció és a health_check szekció attribútumait!

Fontos: Ne adjon „server_name” attribútumot a Spark Worker csomópontnak, így az Occopus eszköz képes lesz automatikusan elnevezni őket, így biztosítva azt, hogy a csomópont nevek egyediek legyenek.

2. Tűzfalszabály létrehozása

Az infrastruktúra komponensei egymáshoz kapcsolódnak, ezért több port tartományt is ki kell nyitni a virtuális gépek számára. Jelentkezzünk be az MTA Cloud OpenStack felületére. A „Compute/Access&Security” menüpont alatt létrehozhatunk új tűzfalszabályt a „Create Security Group” gombra kattintva. Létrehozás után szerkeszteni tudjuk a tűzfalszabályt a „Manage Rules/Add Rule” gombra kattintva. Az alábbi portokat adjuk hozzá a tűzfalszabályhoz:

- TCP 22
- TCP 4040
- TCP 8080
- TCP 8888
- TCP 50070

3. Hitelesítés

Az Occopus számára szükség van az MTA Cloudban használt felhasználónév/jelszó párosra, hogy az hitelesíteni tudja magát és képes legyen virtuális gépeket/infrastruktúrákat létrehozni egy adott projekt alatt. Kérjük, ellenőrizze, hogy az Occopus telepítése során helyesen adta-e meg a hitelesítési adatait! Az autentikációs adatok megadásához az alábbi linken talál segítséget.

4. Klaszter méterének beállítása

Ha szükséges, frissítse a Spark Worker csomópontok számát. Ehhez szerkessze az infra-spark-cluster.yaml fájlt és módosítsa a „min” és „max” paramétereket a „scaling” kulcsszó alatt. A skálázás az az intervallum, amelyben a csomópontok száma megváltozhat (min, max). Jelenleg a minimális érték 2-re van állítva (ami az indításkor a kezdeti szám lesz), és a maximális értéke 10.

Az infrastruktúra leíróban (infra-spark-cluster.yaml) személyre szabhatjuk, hogy mennyi worker node induljunk el minimálisan és maximálisan a klaszterben.

```
- &S
   name: spark-worker
   type: spark_worker_node
   scaling:
           min: 2
           max: 10
```

Fontos: Ne feledje, hogy az Occopusnak legalább egy csomópontot el kell indítania minden egyes csomóponttípusból, hogy az infrastruktúra megfelelően működjön, valamint a skálázás csak a Spark Worker csomópontokra alkalmazható ebben a példában!

5. Importálás

Importáljuk a személyre szabott leírókat az Occopus adatbázisába:

```
occopus-import nodes/node_definitions.yaml
```

Győződjön meg róla, hogy a megfelelő virtualenv aktiválva van! Amennyiben ezt még nem tette volna meg korábban, az alábbi parancs segítségével aktiválható az Occopus virtuális környezete:

```
source occopus/bin/activate
```

Megjegyzés: A nodes mappában található cloud init fájlok szerkesztésével a haladó felhasználók személyre szabhatják a Spark konfigurációs fájljait (cloud_init_spark_master.yaml, cloud_init_spark_worker.yaml).

Fontos: Az Occopus akkor veszi fel a csomópont definíciókat az adatbázisból, amikor felépíti az infrastruktúrát, így mindig importálásra van szükség, ha a node definíciós fájl, vagy bármelyik (pld.: kontextualizációs) fájl megváltozik!

6. Felépítés

Az alábbi parancs segítségével megkezdhetjük a Spark klaszter felépítését:

```
occopus-build infra-spark-cluster.yaml
```

7. Felépült infrastruktúra

Sikeres lefutás után a virtuális gépek IP címei, node ID –jai, valamint az infrastruktúra azonosítója megjelenik a log üzenetek alján, listába szedve. Az infrastruktúra azonosító elmenthető, vagy lekérdezhető az occopus-maintain parancs segítségével:

```
List of nodes/ip addresses:
spark-master:
    192.168.xxx.xxx (3116eaf5-89e7-405f-ab94-9550ba1d0a7c)
spark-worker:
    192.168.xxx.xxx (23f13bd1-25e7-30a1-c1b4-39c3da15a456)
    192.168.xxx.xxx (7b387348-b3a3-5556-83c3-26c43d498f39)

14032858-d628-40a2-b611-71381bd463fa
```

8. Tesztelés Jupyter Notebook-al

A Jupyter Notebook webes interfésze a http://<SparkMasterIP>:8888 címen érhető el. Itt felölthet Jupyter Notebook fájlokat és futtathatja azokat.

Megjegyzés: A webes felület védett, a belépéshez bejelentkezésre van szükség. Az alapértelmezett jelszó „lpds”, amit meg lehet változtatni telepítés előtt.

9. Infrastruktúra törlése

Végül, lebonthatja az infrastruktúrát az occopus-build által visszaadott infrastruktúraazonosító használatával:

```
occopus-destroy -i 14032858-d628-40a2-b611-71381bd463fa
```