Standalone에 application 붙여보기

About

resource manager 중 하나인 standalone에 application을 붙여본다. 붙여볼 application은 가장 기초적인 application으로, spark-shell이다.

Prerequisite

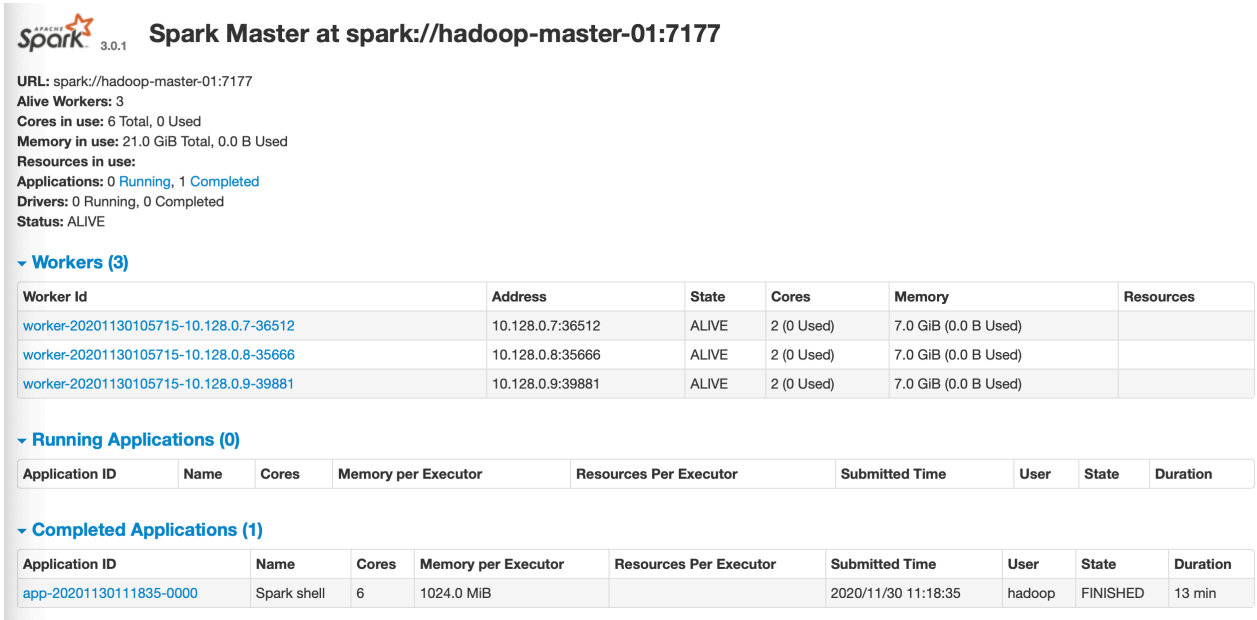
standalone은 spark 설치 시에, 기본적으로 제공되는 가벼운 bundle이다. resource manager로 유명한 yarn이 있지만, 여기서는 다소 가볍게 살펴볼 것이기 때문에 standalone을 사용한다. 따라서 spark 및 standalone이 설치되어 있어야 한다. 본 실습을 위해서 hadoop이 설치되어 있을 필요는 없다.

Let's get started !

standalone?

앞서 언급했듯이, cpu, memory 등을 할당해주는 resource manager이다. 기본적으로 application을 관리한다. 따라서 헷갈리지 말아야할게, standalone은 application이 아니다. application의 자원을 관리해주는 애이다.

standalone도 깔끔한 UI를 제공한다. UI를 살펴보면서 standalone의 구성을 살펴보도록 하자.



Spark 3.0.1 **Spark Master at spark://hadoop-master-01:7177**

URL: spark://hadoop-master-01:7177
Alive Workers: 3
Cores in use: 6 Total, 0 Used
Memory in use: 21.0 GiB Total, 0.0 B Used
Resources in use:
Applications: 0 Running, 1 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers (3)

Worker Id	Address	State	Cores	Memory	Resources
worker-20201130105715-10.128.0.7-36512	10.128.0.7:36512	ALIVE	2 (0 Used)	7.0 GiB (0.0 B Used)	
worker-20201130105715-10.128.0.8-35666	10.128.0.8:35666	ALIVE	2 (0 Used)	7.0 GiB (0.0 B Used)	
worker-20201130105715-10.128.0.9-39881	10.128.0.9:39881	ALIVE	2 (0 Used)	7.0 GiB (0.0 B Used)	

Running Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration

Completed Applications (1)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20201130111835-0000	Spark shell	6	1024.0 MiB		2020/11/30 11:18:35	hadoop	FINISHED	13 min

우선 이 UI는 미리 지정한 포트로 들어갈 수 있다. 보면 세 개의 워커가 있음을 알 수 있고 각 워커는 두 개의 코어, 7G의 메모리를 부여받았다. 이는 `/kikang/spark3/conf/spark-env.sh` 에서 지정한 환경설정이다.

Running Applications는 현재 구동되고 있는 application의 목록을 보여준다. 현재는 돌아가고 있는 application이 없으므로 아무 내용도 표시되지 않았다.

Completed Application은 구동이 종료된 application을 보여준다. 방금 전에, spark-shell을 붙였다가 종료해서 구동이 종료된 리스트에 올라가 있다.

spark-shell

spark-shell은 scala 기반의 interpreter이다. interpreter란, 코드를 치면 바로 결과를 제공해주는 것을 의미한다. spark-shell을 아래와 같이 실행해보자.

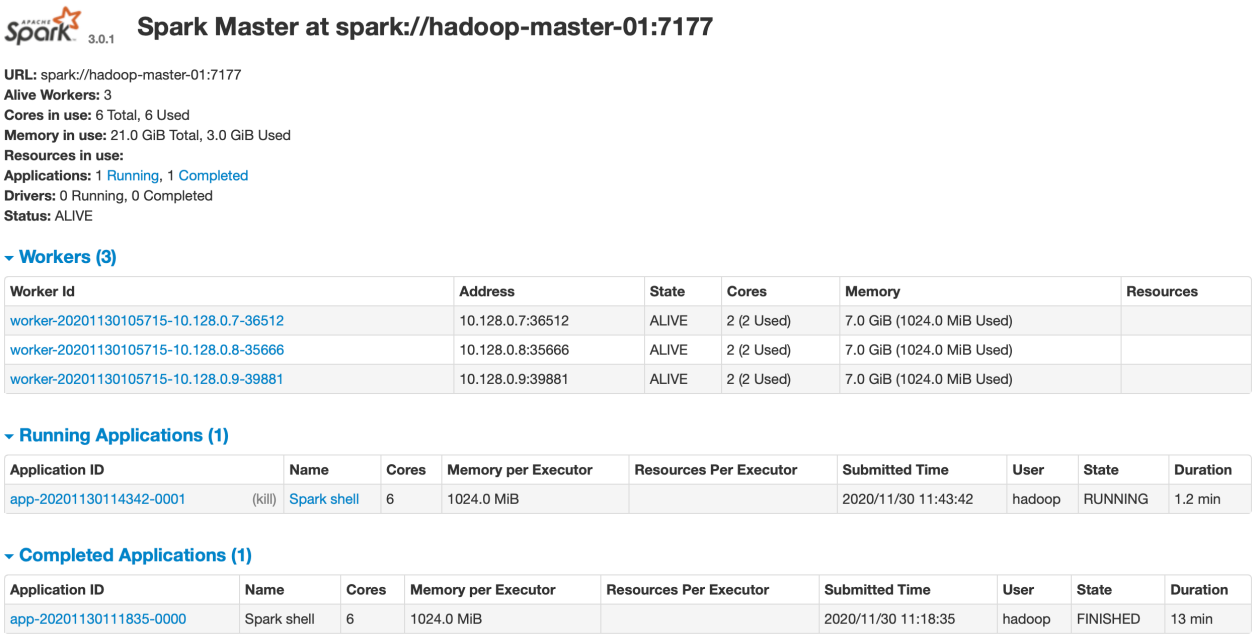
```
$ /kikang/spark3/bin/spark-shell --master local[*]
```

spark-shell을 local로, 모든 코어를 사용(*)해서 실행하겠다는 뜻이다. 로컬로 실행했으니 standalone에 붙지 않는 것은 당연하다.

이제 아래와 같은 명령어를 실행해보자.

```
$ /kikang/spark3/bin/spark-shell --master spark://hadoop-master-01:7177
```

standalone의 주소인 spark://hadoop-master-01:7177을 local[*] 대신 적어줬다. 즉, spark-shell을 위 standalone 주소로 붙이겠다는 뜻이다. 이제 standalone UI을 새로고침해보면, Running Application 목록에 spark-shell이 추가되어 있을 것이다.



Spark Master at spark://hadoop-master-01:7177

URL: spark://hadoop-master-01:7177
Alive Workers: 3
Cores in use: 6 Total, 6 Used
Memory in use: 21.0 GiB Total, 3.0 GiB Used
Resources in use:
Applications: 1 Running, 1 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers (3)

Worker Id	Address	State	Cores	Memory	Resources
worker-20201130105715-10.128.0.7-36512	10.128.0.7:36512	ALIVE	2 (2 Used)	7.0 GiB (1024.0 MiB Used)	
worker-20201130105715-10.128.0.8-35666	10.128.0.8:35666	ALIVE	2 (2 Used)	7.0 GiB (1024.0 MiB Used)	
worker-20201130105715-10.128.0.9-39881	10.128.0.9:39881	ALIVE	2 (2 Used)	7.0 GiB (1024.0 MiB Used)	

Running Applications (1)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20201130114342-0001 (kill)	Spark shell	6	1024.0 MiB		2020/11/30 11:43:42	hadoop	RUNNING	1.2 min

Completed Applications (1)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20201130111835-0000	Spark shell	6	1024.0 MiB		2020/11/30 11:18:35	hadoop	FINISHED	13 min

구동되고 있는 spark-shell application을 클릭해보자.

Application: Spark shell

ID: app-20201130114342-0001
Name: Spark shell
User: hadoop
Cores: Unlimited (6 granted)
Executor Limit: Unlimited (3 granted)
Executor Memory: 1024.0 MiB
Executor Resources:
Submit Date: 2020/11/30 11:43:42
State: RUNNING
Application Detail UI

Executor Summary (3)

ExecutorID	Worker	Cores	Memory	Resources	State	Logs
2	worker-20201130105715-10.128.0.8-35666	2	1024		RUNNING	stdout stderr
1	worker-20201130105715-10.128.0.7-36512	2	1024		RUNNING	stdout stderr
0	worker-20201130105715-10.128.0.9-39881	2	1024		RUNNING	stdout stderr

세 개의 executor ID, 즉 워커가 있음을 확인할 수 있다. 이 세 개의 워커는 앞서 지정한 세 개와 일치한다.

그 중 하나를 클릭해보자.

Spark Worker at 10.128.0.8:35666

ID: worker-20201130105715-10.128.0.8-35666
Master URL: spark://hadoop-master-01:7177
Cores: 2 (2 Used)
Memory: 7.0 GiB (1024.0 MiB Used)
Resources:

Back to Master

Running Executors (1)

ExecutorID	State	Cores	Memory	Resources	Job Details	Logs
2	RUNNING	2	1024.0 MiB		**ID:** app-20201130114342-0001 **Name:** Spark shell **User:** hadoop	stdout stderr

Finished Executors (1)

ExecutorID	State	Cores	Memory	Resources	Job Details	Logs
2	KILLED	2	1024.0 MiB		**ID:** app-20201130111835-0000 **Name:** Spark shell **User:** hadoop	stdout stderr

Master URL을 보면 hadoop-master-01이라는 이름의 워커임을 알 수 있다. 코어는 2개이고 메모리는 7G이다.

이제 pyspark application을 또 붙여보자.

```
$ ./pyspark --master spark://hadoop-master-01:7177
```

Spark Master at spark://hadoop-master-01:7177

URL: spark://hadoop-master-01:7177
Alive Workers: 3
Cores in use: 6 Total, 6 Used
Memory in use: 21.0 GiB Total, 3.0 GiB Used
Resources in use:
Applications: 2 Running, 1 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers (3)

Worker Id	Address	State	Cores	Memory	Resources
worker-20201130105715-10.128.0.7-36512	10.128.0.7:36512	ALIVE	2 (2 Used)	7.0 GiB (1024.0 MiB Used)	
worker-20201130105715-10.128.0.8-35666	10.128.0.8:35666	ALIVE	2 (2 Used)	7.0 GiB (1024.0 MiB Used)	
worker-20201130105715-10.128.0.9-39881	10.128.0.9:39881	ALIVE	2 (2 Used)	7.0 GiB (1024.0 MiB Used)	

Running Applications (2)

Application ID		Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20201130115212-0002	(kill)	PySparkShell	0	1024.0 MiB		2020/11/30 11:52:12	hadoop	WAITING	9 s
app-20201130114342-0001	(kill)	Spark shell	6	1024.0 MiB		2020/11/30 11:43:42	hadoop	RUNNING	8.7 min

Completed Applications (1)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20201130111835-0000	Spark shell	6	1024.0 MiB		2020/11/30 11:18:35	hadoop	FINISHED	13 min

Running Applications 목록에 pysparkShell이 추가되었음을 확인할 수 있다. 하지만 이 application이 할당 받은 코어의 개수를 보자. 한 개도 할당 받지 못했다! 이는, spark-shell application이 이미 6개의 코어를 모두 할당받았기 때문이다. 이를 해결하기 위해서는 환경설정에서 지정하는 워커당 코어의 개수를 늘리거나, application에 할당하는 코어의 개수를 줄이는 방법이 있다.

코어가 하나도 없으면 spark job을 수행할 수 없으므로 위 사항을 고려하여 코어의 개수를 정해야할 것이다.