

Setting up Clustering Environment with AWS EMR & Apache Spark

서석인

Contents

1. Configuration	2
2. Data Preprocessing	2
1. LIBSVM Format	2
2. Networking with AWS S3 Cloud	2
3. Creating AWS EMR Cluster	3
4. Spark Programming with Apache Zeppelin	7

1. Configuration

- Data Preprocessing Environment
 - AWS EC2 r4.xlarge
 - OS: Ubuntu Server 16.04 LTS 64bit (HVM)
 - Spec : 52.8 ECUs, 16 vCPUs, 2.3 GHz, Intel Broadwell E5-2686v4, 122 GiB memory, EBS only
- AWS S3 (Simple Storage Service)
- AWS EMR (Elastic Map-Reduce)
- Ganglia 3.7.2, Spark 2.1.0, Zeppelin 0.6.2

2. Data Preprocessing

1. LIBSVM Format

Spark MLlib에서 데이터를 프로세싱하기 위해서는 우선 LabeledPoint 자료형으로 데이터를 읽어와야 한다. 이를 위해 가장 널리 사용하는 데이터 포맷은 libsvm¹ 인데, 이는 다른 텍스트 형식과는 다르게 다음과 같은 형식을 가지고 있다.

```
label index1:value1 index2:value2 ...
```

각 행(row)는 개행문자(newline character)로 구분되고, 모든 열(column)을 저장하는 것이 아니라, Value 가 0이 아닌 Index들만 저장하도록 하여 저장공간을 절약한다. 따라서, 원래 저장되어 있던 데이터 형식이 csv, txt, binary 포맷일 경우 LIBSVM 포맷에 맞게 데이터를 전처리(Preprocessing)를 해줄 필요가 있다.

2. Networking with AWS S3 Cloud

AWS EC2와 EMR에서 데이터를 공유하기 위해서 AWS S3를 사용할 수 있다. 터미널을 통해 AWS S3에 접근하기 위해서는 우선 해당 인스턴스에 AWS IAM (Identity and Access Management) 서비스를 통해 AWS 계정정보를 저장해야 한다.

¹ <https://www.csie.ntu.edu.tw/~cjlin/libsvm/>

계정정보를 저장한 후에는 s3cmd 라는 S3 클라이언트 커맨드라인 툴을 이용해서 IAM 계정정보로 등록된 AWS S3 버킷에 접근할 수 있다. Ubuntu에서는 `sudo apt-get install s3cmd` 커맨드를 통해서 설치할 수 있다. s3cmd 패키지를 설치한 후에는 `s3cmd --configure` 커맨드를 통해서 IAM 을 통해 생성한 Access Key ID와 Secret Access Key, Default Region을 입력해서 .s3cfg 파일을 생성한다. 이 작업을 통해서 해당 인스턴스(또는 로컬 컴퓨터)에 AWS 계정을 등록할 수 있다.

위 설정까지 완료 했다면, `s3cmd ls` 명령을 통해 해당 계정의 bucket에 어떤 파일이 있는지 다음과 같이 확인해 볼 수 있다.

```
ubuntu@ip-172-31-37-188:~$ s3cmd ls
2017-01-02 09:39 s3://aws-logs-159644160686-ap-northeast-2
2017-03-08 17:06 s3://aws-logs-159644160686-us-east-1
2017-01-02 09:26 s3://cvip-rf
2017-03-09 08:17 s3://cvip-rf2
2016-07-30 23:23 s3://tzsdefault
```

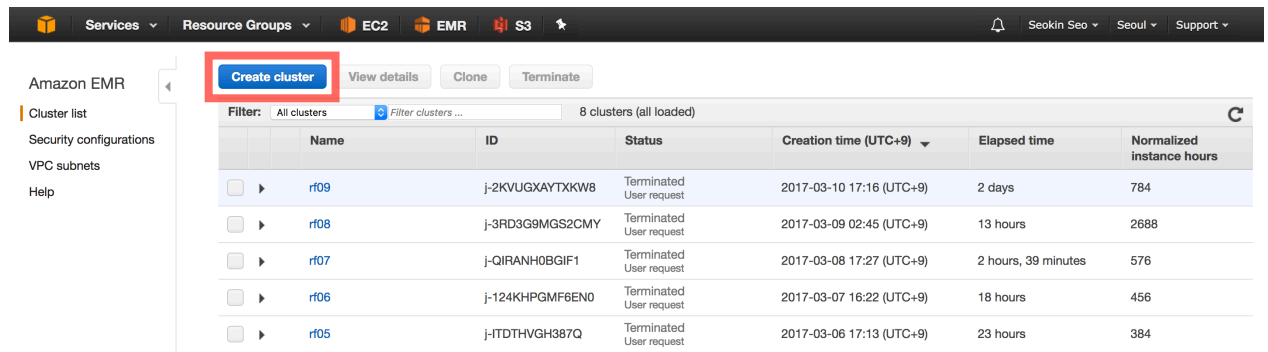
위 명령까지 성공했다면, 각각 `get`과 `put` 명령어를 통해 AWS S3에서 파일을 다운로드하거나 업로드 할 수 있다. 따라서, 원래 바이너리 파일을 S3에 업로드한 후에 이것을 EC2 인스턴스에 다운로드 한 후, 2-1에서 언급한 데이터 전처리(Preprocessing) 과정을 거친 다음 다시 S3에 결과파일을 업로드하도록 한다. 따라서, 2-1에서 처리한 LIBSVM 파일을 `s3cmd put` 명령어를 통해서 AWS S3에 업로드한다.

```
ubuntu@ip-172-31-37-188:~/SparkRandomForestProj$ s3cmd get s3://cvip-rf2/INTER_NN_try1.fmat
download: 's3://cvip-rf2/INTER_NN_try1.fmat' -> './INTER_NN_try1.fmat' [1 of 1]
7020000008 of 7020000008 100% in 83s 80.50 MB/s done
```

3. Creating AWS EMR Cluster

AWS EMR(Elastic Map-Reduce)은 Apache Hadoop이나 Apache Spark 등 관리형 빅데이터 프레임워크를 구동하기 위한 클러스터 플랫폼을 제공하는 서비스이다. EMR을 사용하면 기본적인 환경부터 시작해서 당장 데이터를 처리하기 위한 기본적인 프레임워크와 디펜던시들을 한번에 설치할 수 있어서 Apache Hadoop 또는 Spark를 별도의 추가적인 환경설정 없이 빠르게 구동하기에 매우 간편하다. 뿐만 아니라 Spark는 웹페이지를 통하여 대화형 협업 노트북을 생성하는 Zeppelin이라는 일종의 그래픽 유저 인터페이스를 제공해서 사용자가 더욱 간편하게 데이터를 처리할 수 있는데 AWS EMR은 Zeppelin까지 제공하여 클러스터를 생성하고 별도의 작업 없이 빠르게 데이터를 처리할 수 있다.

EMR 클러스터를 생성하는 방법은 어렵지 않은데, 우선 AWS 콘솔에서 EMR 메뉴를 선택한다. EMR 메뉴를 선택하면 다음과 같은 화면을 볼 수 있다. 다음 화면에서 Create cluster 버튼을 클릭하여 클러스터 생성을 시작할 수 있다.



Amazon EMR

Create cluster View details Clone Terminate

Filter: All clusters 8 clusters (all loaded)

	Name	ID	Status	Creation time (UTC+9)	Elapsed time	Normalized instance hours
<input type="checkbox"/>	▶ r109	j-2KVUGXAYTXKW8	Terminated User request	2017-03-10 17:16 (UTC+9)	2 days	784
<input type="checkbox"/>	▶ r108	j-3RD3G9MGS2CMY	Terminated User request	2017-03-09 02:45 (UTC+9)	13 hours	2688
<input type="checkbox"/>	▶ r107	j-QIRANH0BGIF1	Terminated User request	2017-03-08 17:27 (UTC+9)	2 hours, 39 minutes	576
<input type="checkbox"/>	▶ r106	j-124KHGPMF6EN0	Terminated User request	2017-03-07 16:22 (UTC+9)	18 hours	456
<input type="checkbox"/>	▶ r105	j-ITDTHVGH387Q	Terminated User request	2017-03-06 17:13 (UTC+9)	23 hours	384

클러스터 생성화면에 진입하면 다음과 같은 화면을 볼 수 있다. 아래 화면은 Quick options 일때의 화면이다.

General Configuration

Cluster name

☒ Logging ⓘ

S3 folder ⓘ

Launch mode ☒ Cluster ⓘ ☐ Step execution ⓘ

Software configuration

Vendor ☒ Amazon

Release ⓘ

Applications

- ☒ Core Hadoop: Hadoop 2.7.3 with Ganglia 3.7.2, Hive 2.1.1, Hue 3.11.0, Mahout 0.12.2, Pig 0.16.0, and Tez 0.8.4
- ☐ HBase: HBase 1.3.0 with Ganglia 3.7.2, Hadoop 2.7.3, Hive 2.1.1, Hue 3.11.0, Phoenix 4.9.0, and ZooKeeper 3.4.9
- ☐ Presto: Presto 0.166 with Hadoop 2.7.3 HDFS and Hive 2.1.1 Metastore
- ☐ Spark: Spark 2.1.0 on Hadoop 2.7.3 YARN with Ganglia 3.7.2 and Zeppelin 0.7.0

Hardware configuration

Instance type

Number of instances (1 master and 2 core nodes)

Security and access

EC2 key pair ⓘ [Learn how to create an EC2 key pair.](#)

Permissions ☒ Default ☐ Custom

Use default IAM roles. If roles are not present, they will be automatically created for you with managed policies for automatic policy updates.

EMR role [EMR_DefaultRole](#) ⓘ

EC2 instance profile [EMR_EC2_DefaultRole](#) ⓘ

Cancel Create cluster

1. General Configuration

기본적인 설정들에 대한 항목이다. Cluster name에 클러스터 이름으로 사용하고 싶은 이름을 적으면 되고, 로깅을 하고 싶은 경우 Logging 체크박스에 체크해주고 로그가 저장될 S3 버킷 또는 주소를 지정해주면 된다. Launch mode의 경우 ongoing cluster로 실행할지, transient cluster로 실행할지 결정하는 옵션이다. 두 클러스터링 방식의 차이점은 다음과 같다.

- **Cluster** 옵션에서는 별도의 소프트웨어 설정(Software Configuration)을 통해 어플리케이션을 실행할 클러스터를 생성한다. 클러스터를 생성한 후에 Step들을 추가할 수 있고, Step이 종료(terminate)된 후에도 클러스터는 동작하게 된다.
- **Step execution** 옵션에서는 클러스터를 생성한 후에 실행될 Step들을 바로 추가한다. 어떠한 Step들이 추가되었는지에 따라 클러스터에 포함될 어플리케이션들이 생성시 추가될 지 결정되게 되고, Step이 완료된 후에는 클러스터도 자동으로 종료되게 된다.

이번 프로젝트에서는 Cluster 옵션을 이용하여 클러스터링 환경을 구축하도록 한다.

2. Software Configuration

Cluster 옵션을 선택했을 때 별도로 설정을 하게 되는 부분이다. Vendor는 Cluster Framework를 제공할 업체를 뜻하고, Release는 프레임워크의 릴리즈 버전을 의미한다. Applications에서는 어떤 클러스터링 어플리케이션을 사용할 지 선택하는 부분인데, 이번 프로젝트에서는 Spark를 사용할 것이므로 Spark를 선택한다.

3. Hardware Configuration

어떤 하드웨어를 통해 클러스터링 노드를 구축할 지 설정하는 부분이다. Instance Type은 어떠한 AWS EC2 서버를 노드로 사용할지 선택하는 부분이고, Number of Instance는 몇 개의 노드를 사용할지 선택하는 부분이다. 기본적으로 1개는 마스터 노드를 담당하게 되고, 나머지 노드들은 코어 노드를 담당하게 된다. 마스터 노드를 통해 클러스터링 시스템을 관리하고, 클러스터링 시스템과 네트워킹할 수 있다.

4. Security and access

마지막으로 보안과 접근에 대해 설정하는 부분이다. EC2 key pair의 경우 마스터 노드에 SSH로 접근할 때 필요한 key pair를 설정해주는 부분이다. pem키의 경우 여기서는 생성할 수 없고 EC2 메뉴에서 별도로 생성할 수 있다. 사실 내부 서버로 접근을 필요로 하지 않는 경우에는 따로 설정하지 않아도 된다.

Permission의 경우 EMR 자체의 권장설정을 사용하는 Default와 그 외 별도로 커스터마이징 할 수 있는 Custom 두 가지 설정으로 되어 있다.

모든 설정을 마치고 클러스터를 생성하면 다음과 같은 화면을 볼 수 있다.

Add step

Resize

Clone

Terminate

AWS CLI export

Cluster: rf-09 Starting

Connections: --

Master public DNS: --

Tags: -- [View All / Edit](#)

Summary

ID: j-36KG27CWZ2XPPL

Creation date: 2017-04-05 20:07 (UTC+9)

Elapsed time: 0 seconds

Auto-terminate: No

Termination protection: Off [Change](#)

Configuration Details

Release label: emr-5.4.0

Hadoop distribution: Amazon 2.7.3

Applications: Ganglia 3.7.2, Spark 2.1.0, Zeppelin 0.7.0

Log URI: s3://aws-logs-159644160686-ap-northeast-2/elasticmapreduce/

EMRFS consistent view: Disabled

Network and Hardware

Availability zone: --

Subnet ID: subnet-e4ed758d

Master: Provisioning 1 r3.xlarge

Core: Provisioning 2 r3.xlarge

Task: --

Security and Access

Key name: --

EC2 instance profile: EMR_EC2_DefaultRole

EMR role: EMR_DefaultRole

Visible to all users: All [Change](#)

Security groups for Master:

Security groups for Core & Task:

▶ Monitoring

▶ Hardware

▶ Steps

▶ Configurations

▶ Events

▶ Bootstrap Actions

Starting으로 되어 있는 부분이 Waiting으로 바뀌었다면 클러스터 생성에 성공한 것이다.

4. Spark Programming with Apache Zeppelin

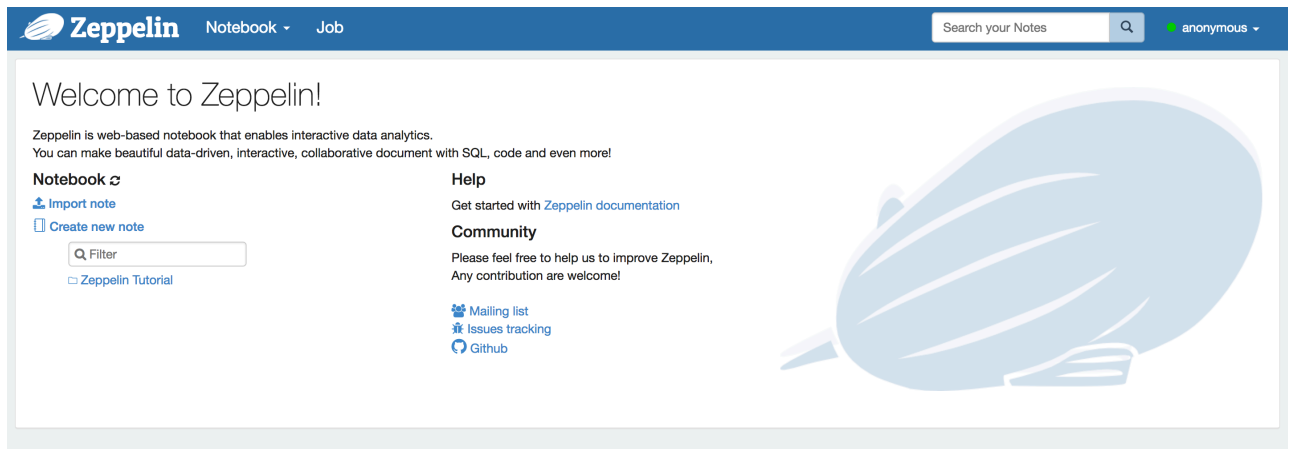
Apache Spark는 Apache Hadoop 위에서 동작하는 프레임 워크로, 기존 HDFS 기반으로 데이터를 처리하던 방식을 인메모리 캐싱 방식을 사용한 RDD(Resident Distributed Datasets)라는 개념을 기반으로 이용하여 데이터를 처리하는 속도를 대폭 향상시킨 빅데이터 클러스터링 프레임워크이다. 3에서 언급했듯 Spark에는 Zeppelin이라는 편리한 대화형 노트북을 제공하므로, 주로 Zeppelin 상에서 노트들을 추가해가면서 Spark 프레임워크 상에서 프로그래밍을 진행하게 된다.

우선 Zeppelin에 접근하려면 마스터 노드(master node)의 8890 포트가 열려있어야 되는데, 마스터 노드가 속한 Security Group의 Inbound의 규칙을 수정해주어야 된다. 먼저 3에서 생성했던 클러스터 환경 설정에서 Security groups for master를 클릭해서 마스터 그룹의 Security Group의 설정을 바꿔주자. Security group을 클릭하면 다음과 같이 Security group을 관리할 수 있는 페이지를 볼 수 있는데, 마스터 노드의 Security group을 클릭한 후 (아래 그룹에서는 ElasticMapReduce-master) Actions를 눌러 Inbound rules를 클릭하여 Inbound 설정을 바꿀 수 있다. 좌측 하단의 Add Rule 버튼을 눌러 8890 포트에 접근할 수 있는 규칙을 추가해주자.

The image shows two screenshots from the AWS Management Console. The top screenshot displays a list of Security Groups for an Elastic MapReduce cluster. A context menu is open over the 'sg-dc50b6b4' group, with 'Edit inbound rules' selected. The bottom screenshot is a detailed view of the 'Edit inbound rules' dialog for the 'sg-dc50b6b4' group. It shows a table of existing inbound rules. A new rule is being added, highlighted with a red box: a Custom TCP Rule on port 8890, allowing traffic from 0.0.0.0/0.

Type	Protocol	Port Range	Source
All TCP	TCP	0 - 65535	Custom sg-db50b6b3
All TCP	TCP	0 - 65535	Custom sg-dc50b6b4
Custom TCP Rule	TCP	8890	Custom 0.0.0.0/0
Custom TCP Rule	TCP	8157	Custom 0.0.0.0/0
Custom TCP Rule	TCP	8157	Custom :::/0
SSH	TCP	22	Custom 0.0.0.0/0
Custom TCP Rule	TCP	8443	Custom 0.0.0.0/0
Custom TCP Rule	TCP	8443	Custom 54.239.116.0/23
All UDP	UDP	0 - 65535	Custom sg-db50b6b3
All UDP	UDP	0 - 65535	Custom sg-dc50b6b4
All ICMP - IPv4	ICMP	0 - 65535	Custom sg-db50b6b3
All ICMP - IPv4	ICMP	0 - 65535	Custom sg-dc50b6b4

이 규칙을 수정한 후, 브라우저에서 마스터 노드의 Public DNS 의 8890 포트로 접근하면 Zeppelin 대쉬 보드로 접속하게 된다. 성공적으로 설정을 마쳤다면 다음과 같은 화면을 보게 된다. 이와 같은 화면을 띄웠다면 클러스터링 환경구축에 성공한 것이다.



위 화면에서 새로 Note를 만들거나 사용자의 로컬 또는 링크로부터 Note를 Import 해올 수 있는데, 동봉된 .json 파일을 Import하여 작업을 재개할 수 있다. 추후에 Note를 Export할 수 있으니 염두 해두자. Import된 Note를 열면 다음과 같은 화면을 볼 수 있다.

