

클라우드 기반 빅데이터 환경구축 (Part 1)

실습조교: 이윤성, 양영석
서울대학교 컴퓨터공학부

Overview

0. 실습에서 사용할 소프트웨어
1. 빅데이터 분석을 위한 클라우드 환경 셋업
2. 클라우드상 배치 처리 분석
3. 클라우드상 대화형 질의 분석
4. 클라우드상 스트림 처리 분석
5. 클라우드상 기계학습 분석
6. 클라우드상 딥러닝 분석

0. 실습에서 사용할 소프트웨어

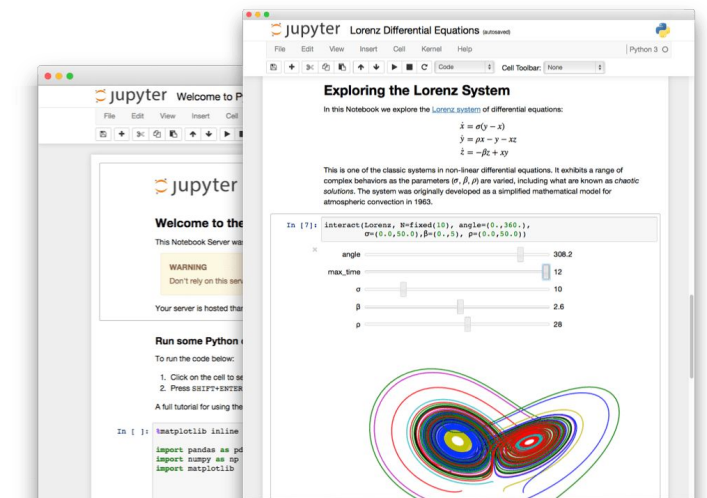
Jupyter

- Interactive computing on web browser

- Code + Documentation
- Interactive Programming
- Supports many languages (Python, R, Scala, etc)
- Integration with many frameworks (Spark, TensorFlow, Ray, etc)

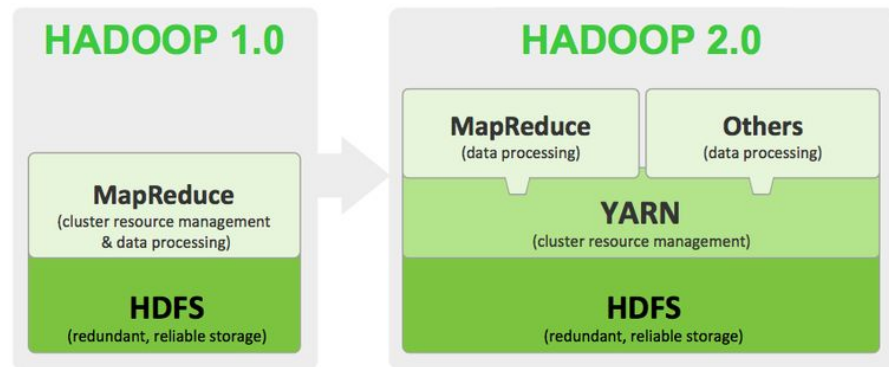
- References

- <http://jupyter.org>
- http://jupyter-notebook.readthedocs.io/en/stable/public_server.html



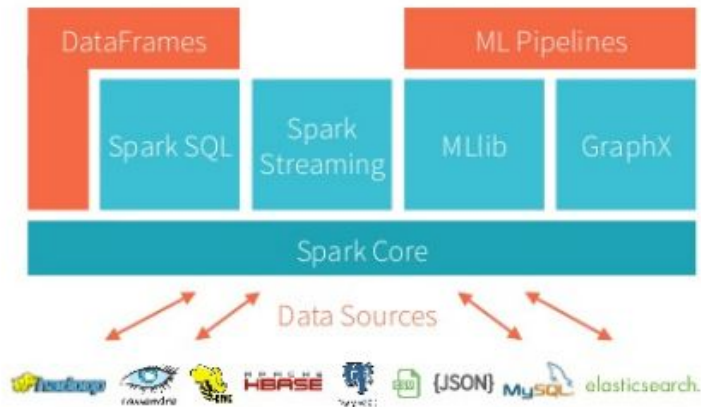
Apache Hadoop

- Basic building block for reliable, scalable, distributed computing
 - **HDFS** is a distributed file system, where the large data is sharded to multiple machines with replication for reliability
 - **YARN** is a resource manager for scheduling cluster resources across multiple data processing applications
- References
 - <http://hadoop.apache.org>



Apache Spark

- Fast and general engine for large-scale data processing
 - Speed: Runs programs up to 100X faster than Hadoop MapReduce in memory.
 - Ease of use: Write applications quickly.
 - Generality: Combines SQL, streaming, and complex analytics.
 - Runs everywhere: Runs on Hadoop, Mesos, etc. Can access diverse data sources like HDFS.
- References
 - <http://spark.apache.org>

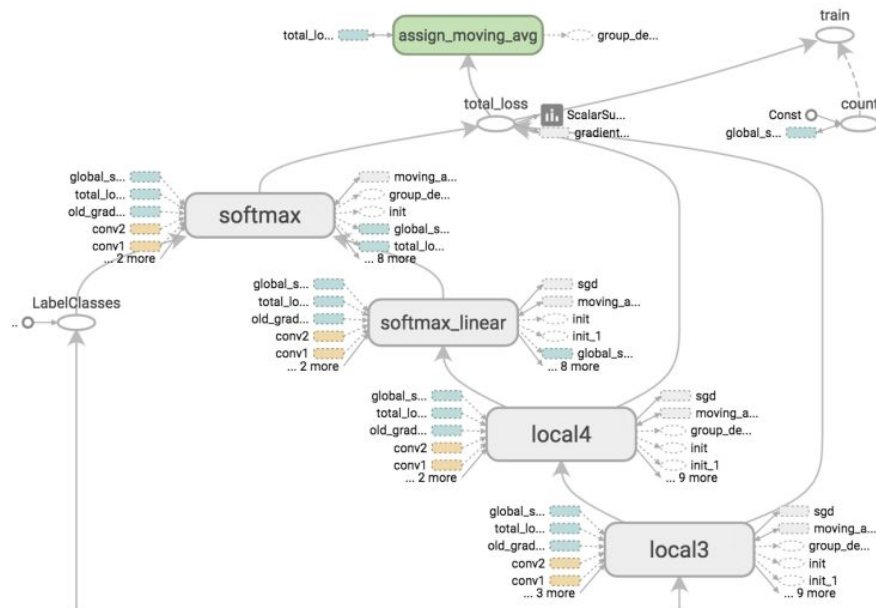


TensorFlow

- SW library for deep learning
 - Express common deep learning models as well as custom, unique models for research
 - Efficient execution of models for fast training and inference
 - Rich set of tutorials and documents
 - Big open source community

- References

- <https://www.tensorflow.org>
- https://www.tensorflow.org/get_started/



1. 빅데이터 분석을 위한 클라우드 환경 셋업

In this session

Goal1: Learn how to deploy your own cluster

- EC2 Overview
- Configure machines

Goal2: Learn how to install and use Hadoop

- HDFS (Hadoop Distributed File System)
- YARN (Yet Another Resource Negotiator, Hadoop v2)

Deploy your own cluster on AWS EC2

EC2 Overview

EC2: Elastic Cloud Computing

- You can use VM (Virtual Machine) instances as you own actual machines
- Choose which instance type(s) you want, then start, terminate, and monitor as many instances as needed.
- Pay only for the resources that you actually consume, like instance-hours or data transfer.
- Can scale elastically as load increases
- More information on <https://aws.amazon.com>

EC2 Overview

EC2: Elastic Cloud Computing

- **You can use VM (Virtual Machine) instances as you own actual machines**



<Your view>



<Amazon Datacenter>

Let's deploy our own cluster

- We will launch 3 instances (1 Master + 2 Workers)
- We prepared AMI (Amazon Machine Images) where most of programs are downloaded/installed
- Our practice consists of minimal configurations to focus on our goal of deploying a cluster
 - EC2 provides many enhanced features such as Elastic IP, Elastic Scaling, etc
 - Please visit <https://aws.amazon.com/ec2/details/> if you are interested

[NOTICE] Account Information

- An account is issued to each team

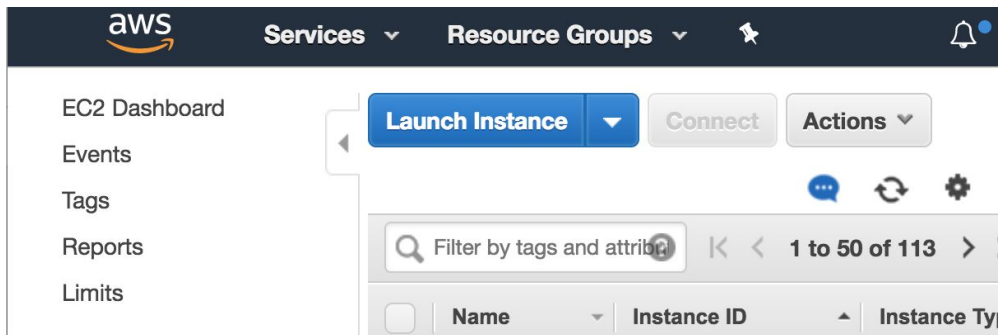
ID: snuds2students+teamX@gmail.com

PASSWORD: * * * * * * * * * *

Step0. Launch Instances

Go to EC2 Management Console

Press the **Launch Instances** button, then you will see the 4 steps



1. Choose AMI

2. Choose Instance Type

3. Configure Instance

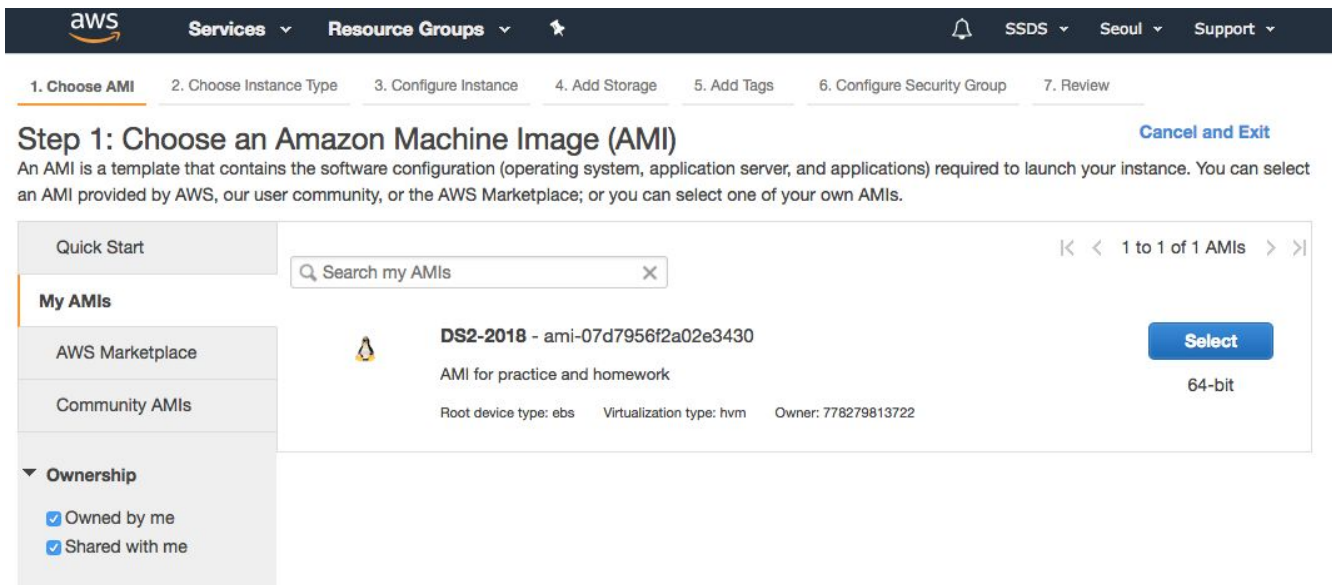
4. Add Storage

Step 1: Choose an Amazon Machine Image (AMI)

server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.

Step1. Choose the AMI

Click **My AMIs** on the left and select the AMI named **DS2-2018**
(If you can't see on the list, check ***Shared with me***)



The screenshot shows the AWS IAM console interface. At the top, there's a navigation bar with the AWS logo, 'Services', 'Resource Groups', and a star icon. Below this is a progress bar with seven steps: '1. Choose AMI' (highlighted), '2. Choose Instance Type', '3. Configure Instance', '4. Add Storage', '5. Add Tags', '6. Configure Security Group', and '7. Review'. The main heading is 'Step 1: Choose an Amazon Machine Image (AMI)' with a 'Cancel and Exit' link. Below the heading is a descriptive paragraph about AMIs. On the left, there's a sidebar with 'Quick Start' and 'My AMIs' (selected). Under 'My AMIs', there are 'AWS Marketplace' and 'Community AMIs' sections. Below these is an 'Ownership' section with two checkboxes: 'Owned by me' (checked) and 'Shared with me' (checked). The main content area shows a search bar 'Search my AMIs' and a list of AMIs. The first AMI is 'DS2-2018 - ami-07d7956f2a02e3430', described as 'AMI for practice and homework'. It has a 'Select' button and shows details: 'Root device type: ebs', 'Virtualization type: hvm', and 'Owner: 778279813722'.

Quick Start

My AMIs

AWS Marketplace

Community AMIs

Ownership

- ☒ Owned by me
- ☒ Shared with me

Search my AMIs

DS2-2018 - ami-07d7956f2a02e3430

AMI for practice and homework



Root device type: ebs Virtualization type: hvm Owner: 778279813722


Select

64-bit

Step2. Choose an Instance Type

Choose **r4.xlarge** type and press **Next: Configure Instance Details**

 Services ▾ Resource Groups ▾ 

 SSDS ▾ Seoul ▾ Support ▾

[1. Choose AMI](#) [2. Choose Instance Type](#) [3. Configure Instance](#) [4. Add Storage](#) [5. Add Tags](#) [6. Configure Security Group](#) [7. Review](#)

Step 2: Choose an Instance Type

<input type="checkbox"/>	Memory optimized	r4.large	2	15.25	EBS only	Yes	Up to 10 Gigabit	Yes
<input type="checkbox"/>	Memory optimized	r4.xlarge	4	30.5	EBS only	Yes	Up to 10 Gigabit	Yes
<input checked="" type="checkbox"/>	Memory optimized	r4.2xlarge	8	61	EBS only	Yes	Up to 10 Gigabit	Yes
<input type="checkbox"/>	Memory optimized	r4.4xlarge	16	122	EBS only	Yes	Up to 10 Gigabit	Yes
<input type="checkbox"/>	Memory optimized	r4.8xlarge	32	244	EBS only	Yes	10 Gigabit	Yes
<input type="checkbox"/>	Memory optimized	r4.16xlarge	64	488	EBS only	Yes	25 Gigabit	Yes
<input type="checkbox"/>	Memory optimized	r4.16xlarge	64	976	1 x 1000 (SSD)	Yes	10 Gigabit	Yes

[Cancel](#) [Previous](#) [Review and Launch](#) [Next: Configure Instance Details](#)

Step3. Configure Instance (1/3)

1. Choose AMI 2. Choose Instance Type **3. Configure Instance** 4. Add Storage 5. Add Tags 6. Configure

Step 3: Configure Instance Details

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request S3 management role to the instance, and more.

Number of instances ⓘ

3

Launch into Auto Scaling Group

Purchasing option ⓘ

☐ Request Spot instances

Network ⓘ

vpc-dadf8ab2 (default) ⌵



Subnet ⓘ

No preference (default subnet in any Availability Zone) ⌵

Auto-assign Public IP ⓘ

Use subnet setting (Enable) ⌵

Set the **Number of instances** as 3

We will leave other network configurations as default

Let's move on to the **Next: Add Storage** on the bottom

Cancel

Previous

Review and Launch

Next: Add Storage

Step3. Configure Instance (2/3)

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

Step 4: Add Storage

Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes. [Learn more](#) about storage options in Amazon EC2.

Volume Type ⓘ	Device ⓘ	Snapshot ⓘ	Size (GiB) ⓘ	Volume Type ⓘ	IOPS ⓘ	Throughput (MB/s) ⓘ	Delete on Termination ⓘ	Encrypted ⓘ
Root	/dev/sda1	snap-0576f06437ebf1698	8	General Purpose GP3	100 / 3000	N/A	<input checked="" type="checkbox"/>	Not Encrypted

[Add New Volume](#)

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage. [Learn more](#) about free usage tier eligibility and usage restrictions.

[Cancel](#) [Previous](#) [Review and Launch](#) [Next: Add Tags](#)

- We can add new volumes (EBS, SSD, HDD)
- But we will stick to the default in this session
- Move to the **Next: Add Tags**

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

Step 5: Add Tags

A tag consists of a case-sensitive key-value pair. For example, you could define a tag with key = Name and value = Webserver. A copy of a tag can be applied to volumes, instances or both. Tags will be applied to all instances and volumes. [Learn more](#) about tagging your Amazon EC2 resources.

Key (127 characters maximum)	Value (255 characters maximum)	Instances ⓘ	Volumes ⓘ
Name	student1	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

[Add another tag](#) (Up to 50 tags maximum)

[Cancel](#) [Previous](#) [Review and Launch](#) [Next: Configure Security Group](#)

- Tags are useful to set (key, value) information
- We will use this feature to identify your machines within the group
- Please put **Name** on the **Key** and your name (e.g., **student1**) on the **Value**
- Move to the **Next: Configure Security Group**

Step3. Configure Instance (3/3)

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags **6. Configure Security Group** 7. Review

Step 6: Configure Security Group

Assign a security group: ☐ Create a new security group
☒ Select an existing security group

Security Group ID	Name	Description	Actions
<input type="checkbox"/> sg-3326c059	default	default VPC security group	Copy to new
<input checked="" type="checkbox"/> sg-07ecacf66abc15637	ds2-2018	Allows ports that are used in our practice	Copy to new

- Security Group restricts the network access (e.g., allow port 22 only from 147.46 *.*.)
- **[Important]** Please choose or create a security group that allows TCP requests (Choose **All TCP, Anywhere**)
- Now it's time to [Review and Launch](#)

Select an existing key pair or create a new key pair ✕

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about [removing existing key pairs from a public AMI](#).

Choose an existing key pair
Select a key pair
ssds2-2018

☒ I acknowledge that I have access to the selected private key file (ssds2-2018.pem), and that without this file, I won't be able to log into my instance.

[Cancel](#) [Launch Instances](#)


- If you review and click [Launch](#) button on the next page, you will encounter this alert box
- You can choose any option (including [Proceed without a key](#)) then click [Launch Instances](#)

Now instances are up!

Click **View Instances** then you will see your machines are up!

Launch Status

 **Your instances are now launching**
The following instance launches have been initiated: i-0e8c97a314f48f1ff, i-099c3817ce9f8d924, i-087a0414d8eb6c1c8 [View launch log](#)

 **Get notified of estimated charges**
Create billing alerts to get an email notification when estimated charges on your AWS bill exceed an amount you define (for example free usage tier).

How to connect to your instances

Your instances are launching, and it may take a few minutes until they are in the **running** state, when they will be ready for you to use. Usage for instances will start immediately and continue to accrue until you stop or terminate your instances.

Click **View Instances** to monitor your instances' status. Once your instances are in the **running** state, you can **connect** to them from the Instance console.

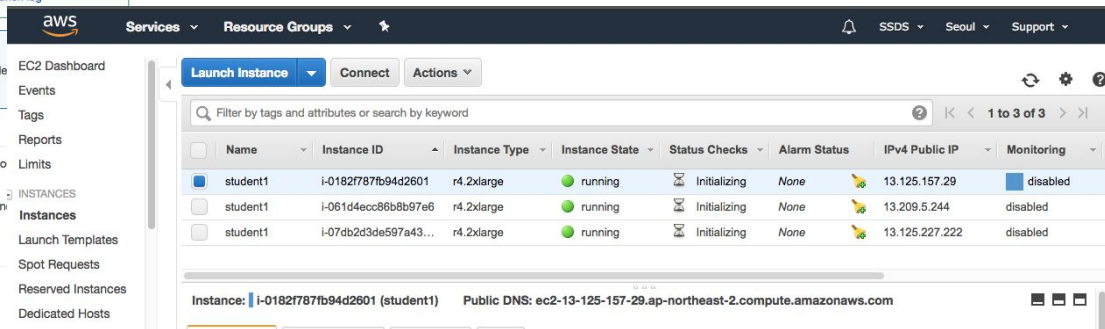
Here are some helpful resources to get you started

- [How to connect to your Linux instance](#)
- [Amazon EC2: User Guide](#)
- [Learn about AWS Free Usage Tier](#)
- [Amazon EC2: Discussion Forum](#)

While your instances are launching you can also

- [Create status check alarms](#) to be notified when these instances fail status checks. (Additional charges may apply)
- [Create and attach additional EBS volumes](#) (Additional charges may apply)
- [Manage security groups](#)

[View Instances](#)



Name	Instance ID	Instance Type	Instance State	Status Checks	Alarm Status	IPv4 Public IP	Monitoring
student1	i-0182f787fb94d2601	r4.2xlarge	running	Initializing	None	13.125.157.29	disabled
student1	i-061d4ecc86b8b97e6	r4.2xlarge	running	Initializing	None	13.209.5.244	disabled
student1	i-07db2d3de597a43...	r4.2xlarge	running	Initializing	None	13.125.227.222	disabled

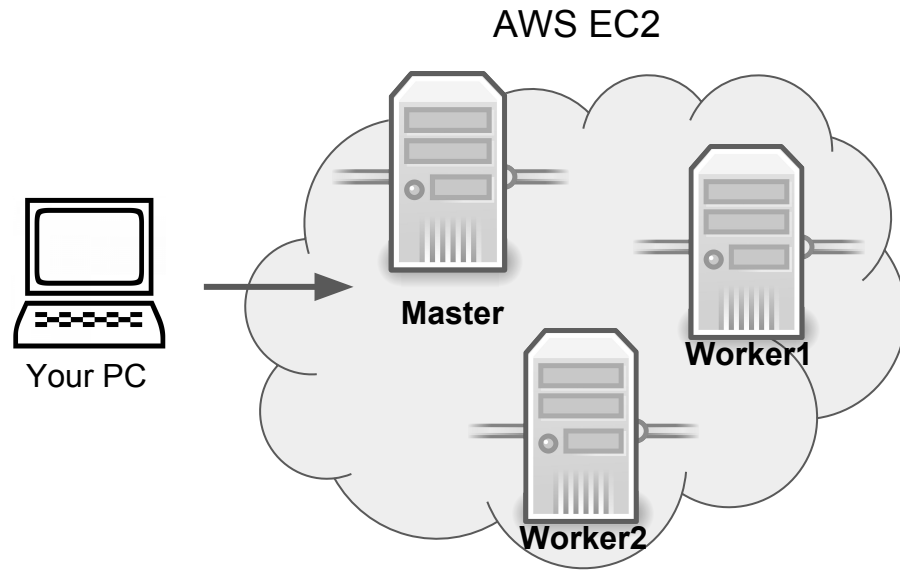
Instance: **i-0182f787fb94d2601 (student1)** Public DNS: **ec2-13-125-157-29.ap-northeast-2.compute.amazonaws.com**

Our small-scale cluster

Now we have 3 machines

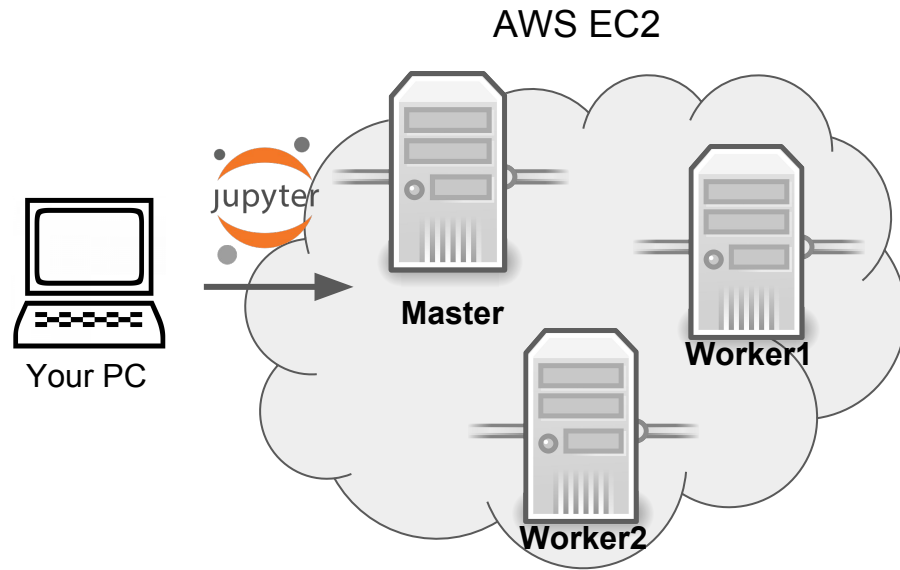
You can assume that you purchased actual machines

- Each machine has its own IP address
- You can access the machine through network
- An OS is installed on them



To access the machines

- Typically, people (including us) use a CLI-based terminal
- We will use Jupyter in this tutorial
 - You don't have to set up anything on your PC
→ You just need a browser!
 - You don't have to type (or copy-and-paste) every command
→ We prepared!



Setup network configurations

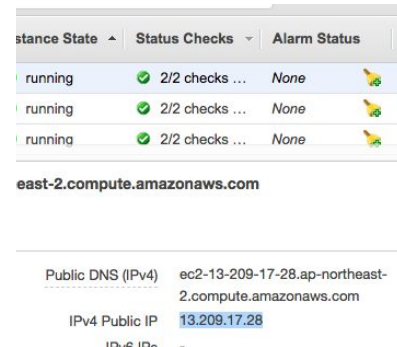
- To use hadoop (and most SWs in the hadoop ecosystem), we need configure
 - Hosts and hostname to let all machines be accessible
 - SSH keys since most programs are executed in remote machines through SSH
 - Profiles such as paths, environment variables

- You will modify `/etc/hosts`, `/etc/hostname`,
`~/.ssh/authorized_keys`, `/etc/environment`

- You can find the commands in the notebook

(`<MASTER_PUBLIC_IP>:8888`)

** MASTER_PUBLIC_IP can be found at the **IPv4 Public IP** field on the instances page*



Instance State	Status Checks	Alarm Status
running	2/2 checks ...	None
running	2/2 checks ...	None
running	2/2 checks ...	None

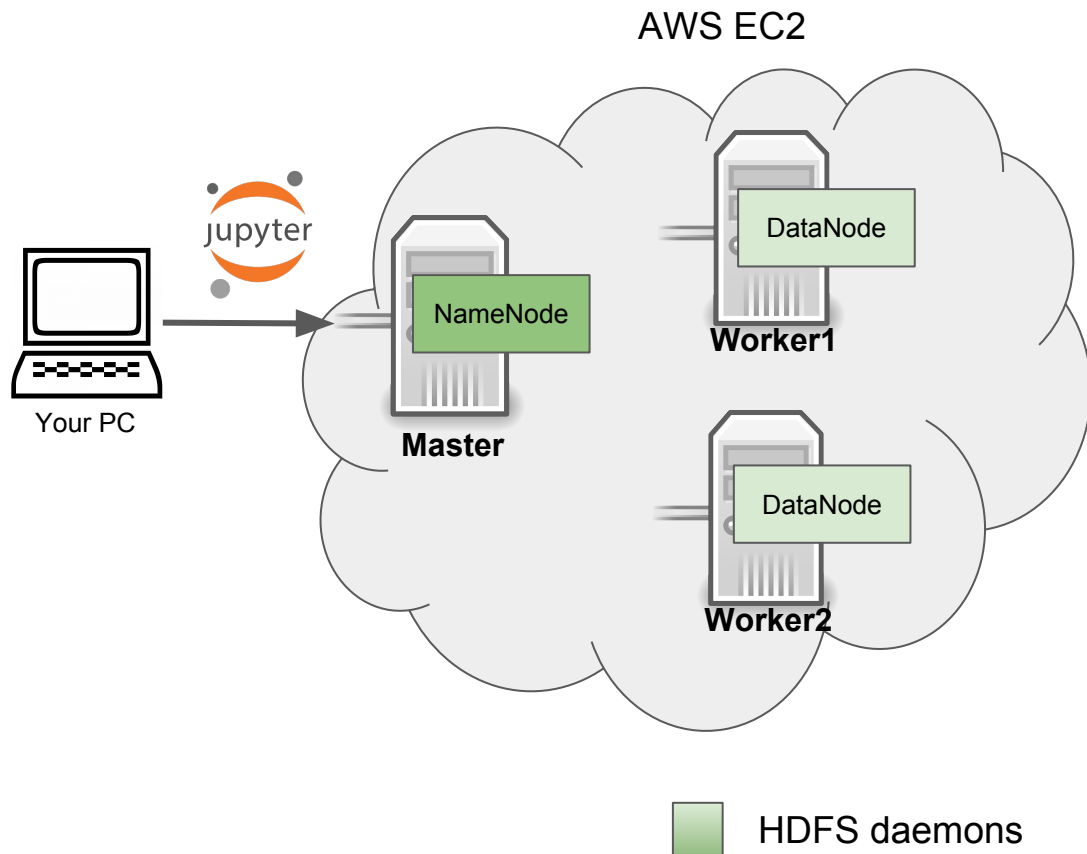
east-2.compute.amazonaws.com

Public DNS (IPv4)	ec2-13-209-17-28.ap-northeast-2.compute.amazonaws.com
IPv4 Public IP	13.209.17.28

Install Hadoop (1/2)

Step 1. We will install HDFS

- Namenode on master
- Datanode on workers
- You will modify
 - core-site.xml
 - hdfs-site.xml
 - workers



Let's turn on HDFS

- We need to format namenode to initiate metadata

```
hdfs namenode format
```

- Run HDFS daemons

```
start-dfs.sh
```

- Go to

<MASTER_PUBLIC_IP>:50070,
then you will see a UI
if everything is set properly

Exercise: Play with HDFS

- Try several commands

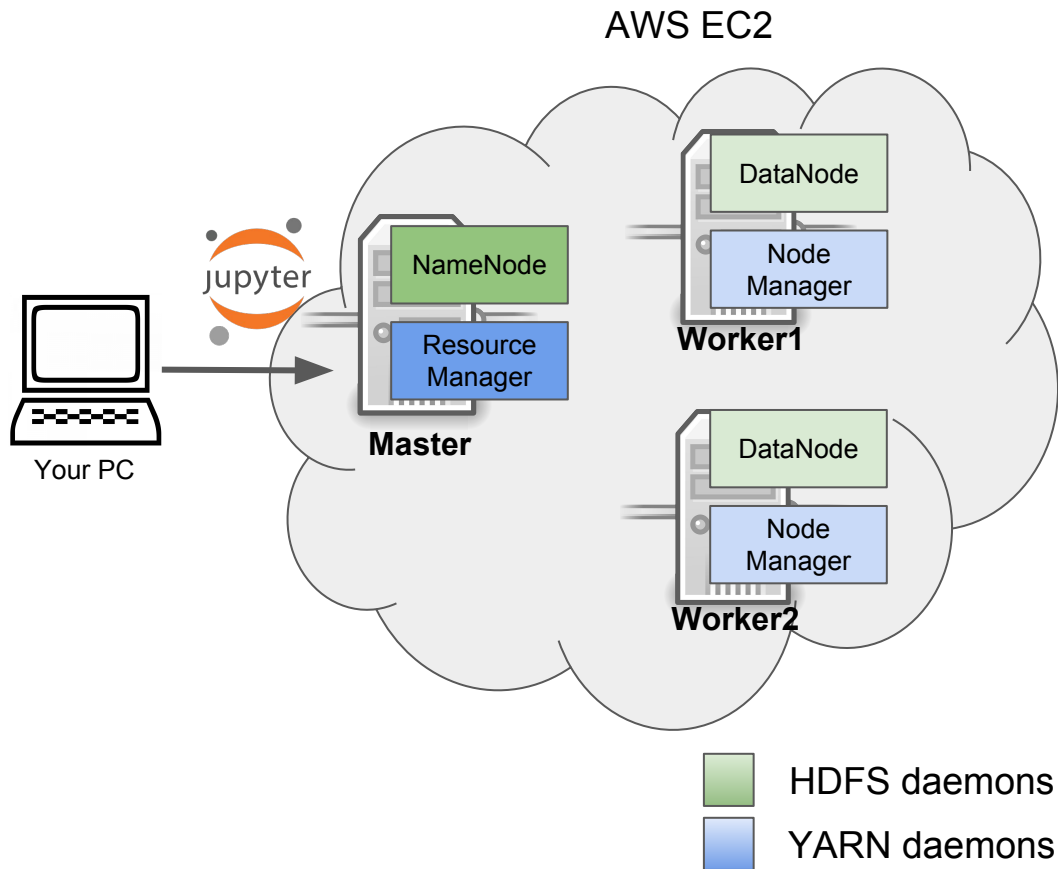
```
hdfs dfs
```

- Upload a file (stored in your local file system) to HDFS
(Let's find which commands you can use)
- Find the file in the web UI (Utilities → File Browser)

Install Hadoop (2/2)

Step 2. We will install YARN

- ResourceManager on Master
- NodeManagers on Workers
- You will modify
 - `yarn-site.xml`



Let's turn on YARN

- Run YARN daemons

```
start-yarn.sh
```

- Go to <MASTER_PUBLIC_IP>:8088,
then you will see a UI
if everything is set properly



Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes	Unhealthy Nodes	Rebooted Nodes
3	0	0	3	0	0 B	110 GB	0 B	0	40	0	5	0	0	0	0

Scheduler Metrics

Capacity Scheduler	Scheduling Resource Type	Minimum Allocation	Maximum Allocation
	[MEMORY]	<memory:10240, vCores:1>	<memory:22528, vCores:8>

Show 20 entries

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Progress	Tracking UI	Blacklisted Nodes
application_1497406530101_0003	ubuntu	mr	YARN	default	Wed Jun 14 11:49:22 +0900 2017	Wed Jun 14 11:52:40 +0900 2017	FINISHED	SUCCEEDED		History	N/A
application_1497406530101_0002	ubuntu	mr	YARN	default	Wed Jun 14 11:40:32 +0900 2017	Wed Jun 14 11:41:11 +0900 2017	KILLED	KILLED		History	N/A
application_1497406530101_0001	ubuntu	mr	YARN	default	Wed Jun 14 11:28:56 +0900 2017	Wed Jun 14 11:32:08 +0900 2017	FINISHED	SUCCEEDED		History	N/A

Showing 1 to 3 of 3 entries

Exercise: Play with YARN

- Let's run DistributedShell example

```
yarn jar
~/hadoop/share/hadoop/yarn/hadoop-yarn-applications-distributedshell
-3.0.1.jar -shell_command 'sleep 120; echo hello yarn' -jar
~/hadoop/share/hadoop/yarn/hadoop-yarn-applications-distributedshell
-3.0.1.jar
```

- Visit web UI and see how the application runs
- This application prints the *commands* to HDFS.

Let's find the file and see its contents

2. 클라우드상 배치 처리 분석

Spark

Case study: SNU Bolt(낙사) manufacturing company

- You're working at SNU Bolt(낙사) manufacturing company
- You're the only data analyst in the company
- Your boss tells you to analyze bolt manufacturing logs
 - The size of the logs is 100TB
- You can use 100 computers for data analytics
- Which technology should you use?

MapReduce vs. Apache Spark

- MapReduce programming interface
 - Only 'Map' and then 'Reduce' (Only 2 steps)
 - Hard to express complex applications
- Apache Spark programming interface
 - Combination of Map, Reduce, Join, .. (Unlimited steps)
 - SQL, Machine learning, Graph processing, ...

Use Apache Spark!

- Distributed system = process data in parallel
- e.g., Processing 100TB manufacturing logs
 - Processing 100TB with 1 computer is slow
 - Processing 100TB with 100 computers is 100X faster
- Compatible with HDFS explained earlier
 - Can read data from HDFS, and write data to HDFS

Spark cluster (+ HDFS cluster)

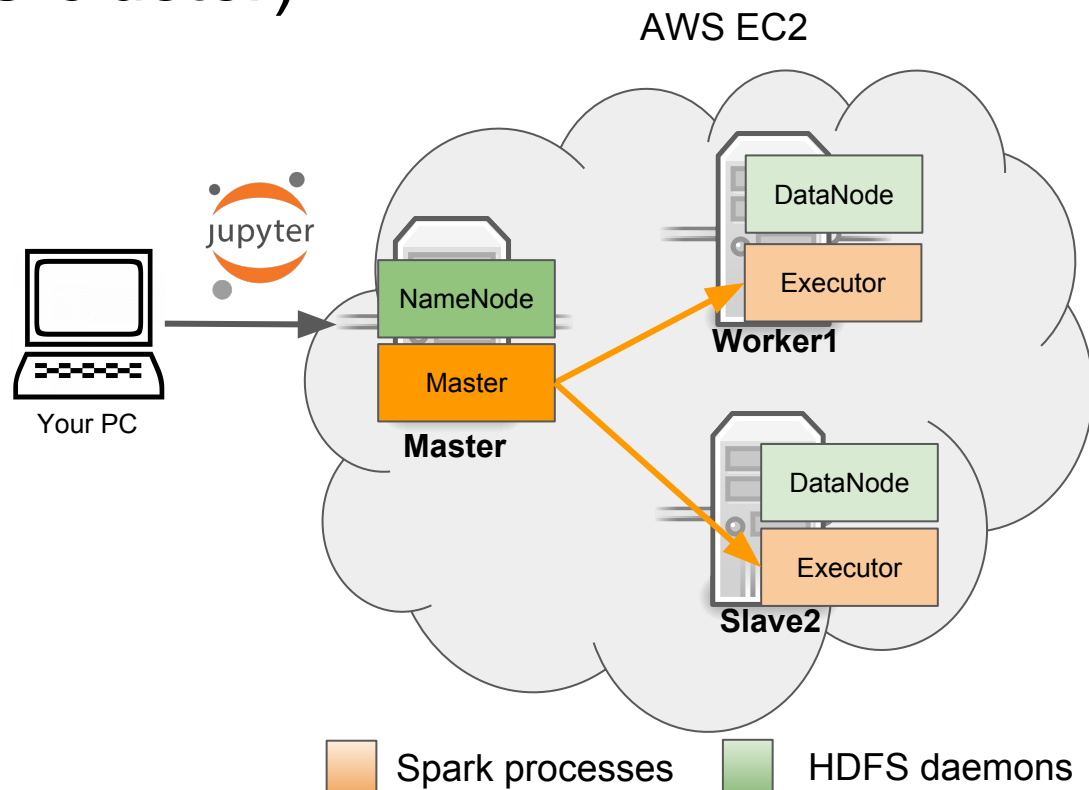
Master on master

Executor on workers

Jupyter command

→ Master (task scheduling)

→ Executor (task execution)



What we will do today

1. Set up a Spark environment in Jupyter
2. Prepare input dataset
3. Run a simple Spark application
4. Analyze job performance using the Spark web ui
5. Visualize the output data

1. Set up a Spark environment in Jupyter

 jupyter Untitled Last Checkpoint: 41 minutes ago (unsaved changes)



Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted

Python 3

           Code 

```
In [8]: import findspark
findspark.init('/home/ubuntu/spark-2.3.0-bin-hadoop2.7')

from pyspark.sql import SparkSession

spark = SparkSession.builder.appName("SNU Bolt").getOrCreate()
textFile = spark.read.text("/home/ubuntu/spark-2.3.0-bin-hadoop2.7/README.md")
```

```
In [9]: textFile.count()
```

```
Out[9]: 103
```

```
In [10]: textFile.first()
```

```
Out[10]: Row(value='# Apache Spark')
```

2. Prepare input dataset


- Copy S3 → HDFS
- Now the dataset is stored in HDFS

3. Run a simple Spark application

- Let's analyze the downloaded data to find top 5 entries
- Our application consists of...
 - Map: Convert each entry into a Tuple
 - Reduce: Group Tuples by key and filter top 5 entries

4. Analyze job performance using the Spark Web UI

- Jobs / Stages / Storage / Environment / Executors / ...

 2.1.0-SNAPSHOT

Jobs

Stages

Storage

Environment

Executors

SQL

Spark shell application UI

Details for Stage 26 (Attempt 0)

Total Time Across All Tasks: 94 ms
Locality Level Summary: Node local: 1; Process local: 74
Shuffle Read: 126.0 B / 2

[DAG Visualization](#)
[Show Additional Metrics](#)
[Event Timeline](#)

Summary Metrics for 75 Completed Tasks

Metric	Min	25th percentile	Median	75th percentile	Max
Duration	0 ms	1 ms	1 ms	2 ms	3 ms
GC Time	0 ms	0 ms	0 ms	0 ms	0 ms
Shuffle Read Blocked Time	0 ms	0 ms	0 ms	0 ms	0 ms
Shuffle Read Size / Records	0.0 B / 0	0.0 B / 0	0.0 B / 0	0.0 B / 0	126.0 B / 2

Aggregated Metrics by Executor

Executor ID ▲	Address	Task Time	Total Tasks	Failed Tasks	Killed Tasks	Succeeded Tasks	Shuffle Read Size / Records
0	stdout stderr 192.168.65.1:60723	0.3 s	75	0	0	75	126.0 B / 2

Tasks (75)

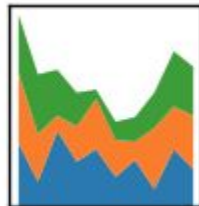
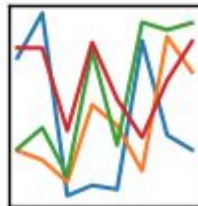
Index	ID	Attempt	Status	Locality Level	Executor ID / Host	Launch Time	Duration	GC Time	Shuffle Read Blocked Time	Shuffle Read Size / Records ▼	Errors
66	333	0	SUCCESS	NODE_LOCAL	0 / 192.168.65.1 stdout stderr	2016/10/17 20:11:22	2 ms		0 ms	126.0 B / 2	
0	334	0	SUCCESS	PROCESS_LOCAL	0 / 192.168.65.1 stdout stderr	2016/10/17 20:11:22	1 ms		0 ms	0.0 B / 0	
1	335	0	SUCCESS	PROCESS_LOCAL	0 / 192.168.65.1 stdout stderr	2016/10/17 20:11:22	2 ms		0 ms	0.0 B / 0	

5. Visualize the output

- The size of the output is small enough to fit in memory
- We can use Pandas to visualize the output

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$



3. 클라우드상 대화형 질의 분석

Spark SQL

Recap: What we've learned in the last class

- Launching AWS instances
- Setting up HDFS
- Loading data into HDFS
- Setting up Jupyter/Spark/Pandas
- Running a Spark job
- Analyzing job performance using the Spark web UI
- Visualizing output data with Pandas


Case study: SNU Bolt(나사) manufacturing company

- You're working at SNU Bolt(나사) manufacturing company
- You'd like to analyze some logs
- This time, the logs are preprocessed and structured as tables with columns and rows
- You want to use **SQL** to process the structured data
- However, the data size is still 100TB
- Which technology should you use?

Use Spark SQL!

- Structured data: Table = Rows X Columns
- SQL language: e.g., “SELECT type FROM bolts”
- Distributed processing
 - vs. single-node MySQL, PostgreSQL
- SQL query \Rightarrow Spark RDD \Rightarrow Distributed execution

1. Set up a Spark environment in Jupyter

 jupyter Untitled Last Checkpoint: 41 minutes ago (unsaved changes)



Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted

Python 3

           Code 

```
In [8]: import findspark
findspark.init('/home/ubuntu/spark-2.3.0-bin-hadoop2.7')

from pyspark.sql import SparkSession

spark = SparkSession.builder.appName("SNU Bolt").getOrCreate()
textFile = spark.read.text("/home/ubuntu/spark-2.3.0-bin-hadoop2.7/README.md")
```

```
In [9]: textFile.count()
```

```
Out[9]: 103
```

```
In [10]: textFile.first()
```

```
Out[10]: Row(value='# Apache Spark')
```


2. Prepare input dataset


- Copy S3 → HDFS
- Now the dataset is stored in HDFS

3. Run a simple SparkSQL application

- We'll run several different queries
- Each query is expanded into a RDD graph
 - We'll examine how the query is expanded
 - We'll examine how the RDD graph is executed

4. Analyze job performance using the Spark Web UI

- Jobs / Stages / Storage / Environment / Executors / ...

 2.1.0-SNAPSHOT

Jobs

Stages

Storage

Environment

Executors

SQL

Spark shell application UI

Details for Stage 26 (Attempt 0)

Total Time Across All Tasks: 94 ms
Locality Level Summary: Node local: 1; Process local: 74
Shuffle Read: 126.0 B / 2

[DAG Visualization](#)
[Show Additional Metrics](#)
[Event Timeline](#)

Summary Metrics for 75 Completed Tasks

Metric	Min	25th percentile	Median	75th percentile	Max
Duration	0 ms	1 ms	1 ms	2 ms	3 ms
GC Time	0 ms	0 ms	0 ms	0 ms	0 ms
Shuffle Read Blocked Time	0 ms	0 ms	0 ms	0 ms	0 ms
Shuffle Read Size / Records	0.0 B / 0	0.0 B / 0	0.0 B / 0	0.0 B / 0	126.0 B / 2

Aggregated Metrics by Executor

Executor ID ▲	Address	Task Time	Total Tasks	Failed Tasks	Killed Tasks	Succeeded Tasks	Shuffle Read Size / Records
0	stdout stderr 192.168.65.1:60723	0.3 s	75	0	0	75	126.0 B / 2

Tasks (75)

Index	ID	Attempt	Status	Locality Level	Executor ID / Host	Launch Time	Duration	GC Time	Shuffle Read Blocked Time	Shuffle Read Size / Records ▼	Errors
66	333	0	SUCCESS	NODE_LOCAL	0 / 192.168.65.1 stdout stderr	2016/10/17 20:11:22	2 ms		0 ms	126.0 B / 2	
0	334	0	SUCCESS	PROCESS_LOCAL	0 / 192.168.65.1 stdout stderr	2016/10/17 20:11:22	1 ms		0 ms	0.0 B / 0	
1	335	0	SUCCESS	PROCESS_LOCAL	0 / 192.168.65.1 stdout stderr	2016/10/17 20:11:22	2 ms		0 ms	0.0 B / 0	

5. Visualize the output

- The size of the output is small enough to fit in memory
- We can use Pandas to visualize the output

pandas

$$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$$

