Yandex

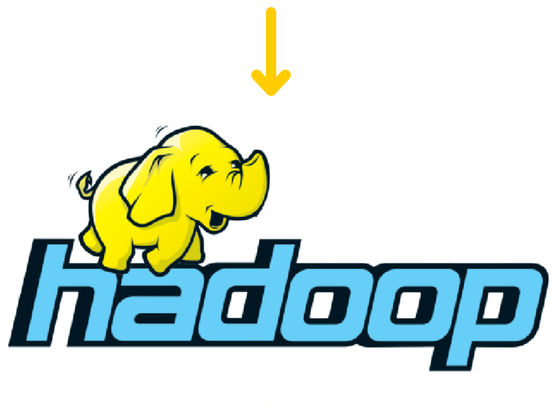
MapReduce

Distributed Cache

Worker들이 사용해야 할 파일을 미리 Node에 업로드해서 공유하는 개념

<article id> <tab> <article content>

key value



특정 단어에 대한 값을 알아보고 싶을 경우

...

zymodemes 1

zymogen 2

zymosan 1

zyu1 4

zz 2

```python
from __future__ import print_function
import re
import sys

def read_vocabulary(file_path):
    return set(word.strip() for word in open(file_path))

vocabulary = read_vocabulary("vocabulary.txt")

for line in sys.stdin:
    article_id, content = line.split("\t", 1)
    words = re.split("\W+", content)
    for word in words:
        if word in vocabulary:
            print(word, 1, sep="\t")
```

특정 단어가 저장된 파일에 있는 단어만 카운트

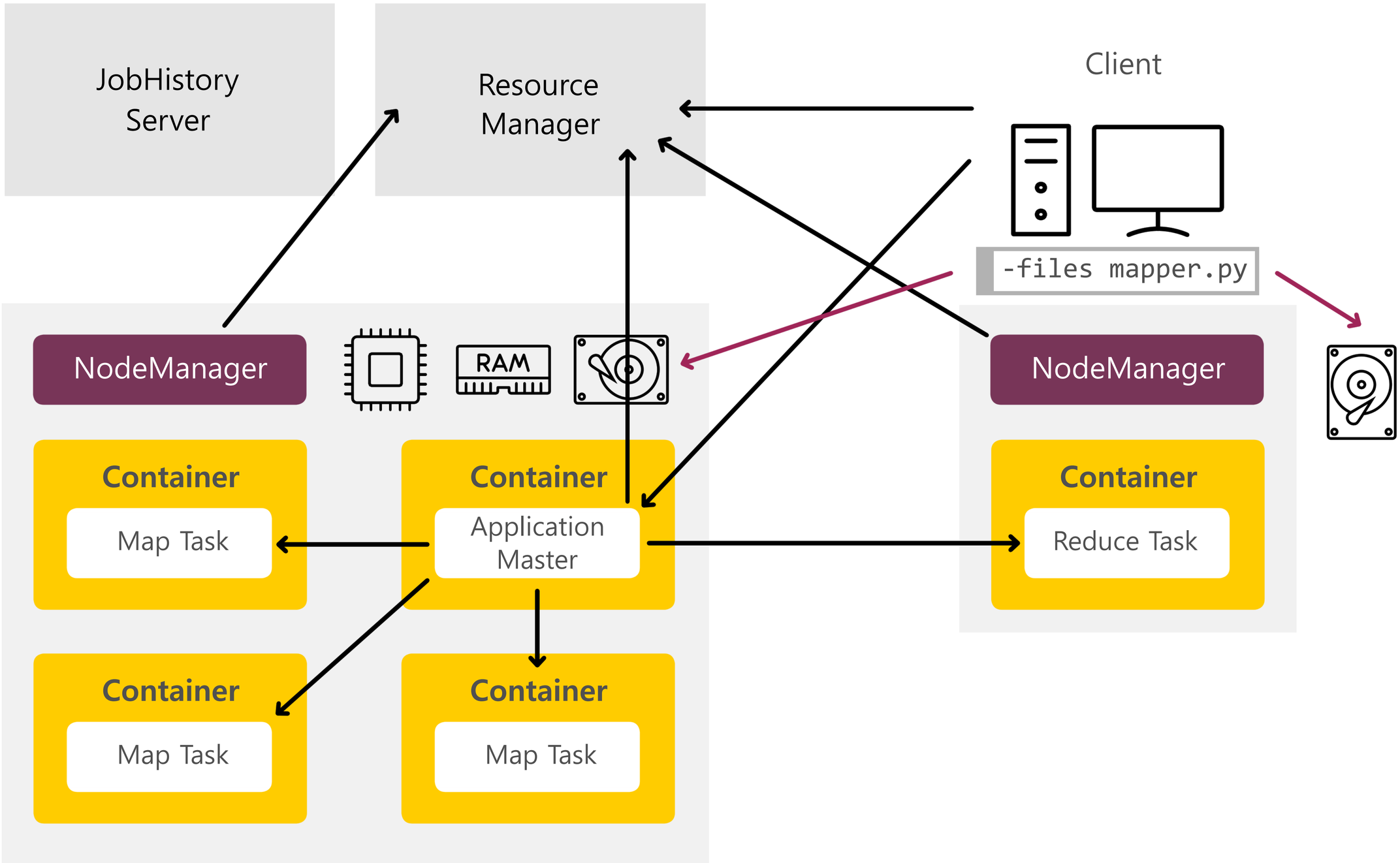
단어 파일을 worker Node에 업로드해야 파일을 읽을 수 있다.

```
yarn jar $HADOOP_STREAMING_JAR \
                    -files mapper.py,reducer.py,vocabulary.txt \
                    -mapper 'python mapper.py' \
                    -reducer 'python reducer.py' \
                    -input /data/wiki/en_articles \
                    -output word_count
```

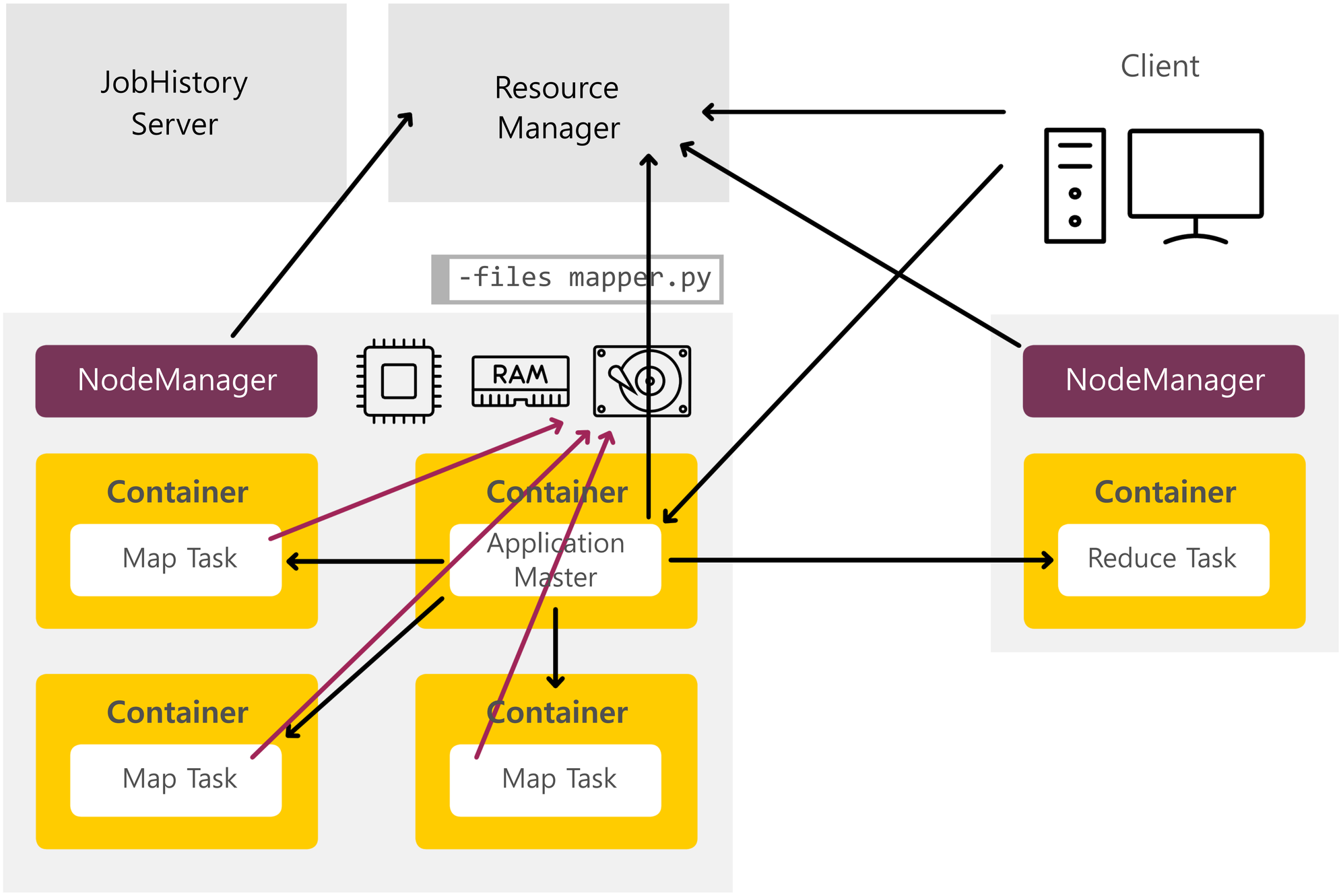
Mapper (Python): mapper.py

```
def read_vocabulary(file_path):
    return set(word.strip() for word in open(file_path))


vocabulary = read_vocabulary("vocabulary.txt")
```

-files 에 지정된 파일들을 모든 Worker Node에 업로드 한다.



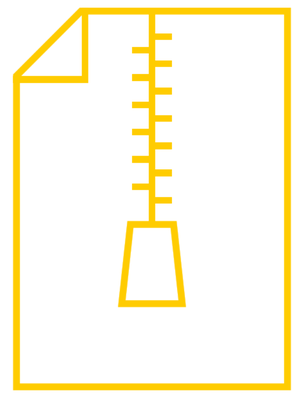
Task Container들은 Node의 디스크에 접근하여 데이터를 읽는다.



Distributed Cache

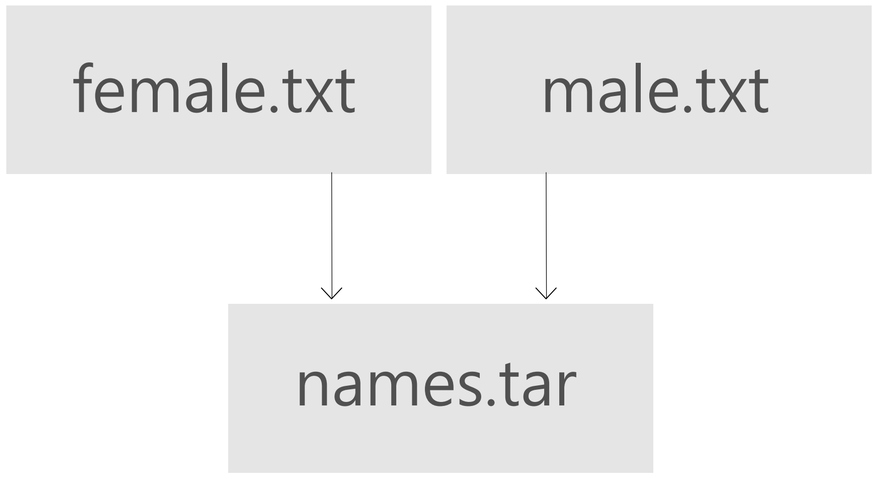
3가지 업로드 가능한 타입

-files



-archives

-libjars




```
$ tar -cf names.tar male.txt female.txt
```

male.txt와 female.txt로 names.tar 압축파일을 만드는 명령어

```
yarn jar $HADOOP_STREAMING_JAR \
                    -files mapper.py,reducer.py,vocabulary.txt \
                    -archives names.tar \
                    -mapper 'python mapper.py' \
                    -reducer 'python reducer.py' \
                    -input /data/wiki/en_articles \
                    -output word_count
```

```python
from __future__ import print_function
import re
import sys

def read_vocabulary(file_path):
    return set(word.strip() for word in open(file_path))

male_names = read_vocabulary("names.tar/male.txt")
female_names = read_vocabulary("names.tar/female.txt")

for line in sys.stdin:
    article_id, content = line.split("\t", 1)
    words = re.split("\W+", content)
    for word in words:
        if word in male_names or word in female_names:
            print(word, 1, sep="\t")
```

경로를 제대로 지정하면
worker에서 압축을 풀고 파일을 읽을 수 있다.


```
$ hdfs dfs -text word_count/* | sort -nrk2,2
James   2284
Thomas 1941
Jack    786
Harry   504
Muhammad        444
Oliver 250
Charlie 250
Jacob   234
Emily   128
Isabella        99
Sophia 92
Noah    80
Sophie 64
Lily    31
Olivia 27
Ethan   27
Ella    25
Amelia 25
Isabelle        18
Chloe   9
```

Summary

- you know how to distribute files and archives to MapReduce workers
- you know how to use distributed files and archives in MapReduce applications
- you can identify the number of a distributed file available on each node (based on 

)

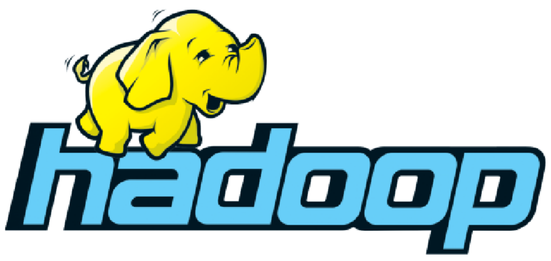
MapReduce

Environment, Counters

worker에게 설정에 필요한 값을 환경변수로 전달할 수 있다.

environment variables
(Job / Task config)

WIKIPEDIA
The Free Encyclopedia



Distributed
Cache

```
...
zymodemes 1
zymogen 2
zymosan 1
zyu1 4
zz 2
```

Mapper (Python): mapper.py

```
from __future__ import print_function
import re
import sys

if os.environ["mapred_task_is_map"] == "true":
print("input_file:{}, start:{}, size:{}".format(
os.environ["mapreduce_map_input_file"],
os.environ["mapreduce_map_input_start"],
os.environ["mapreduce_map_input_length"],
))

for line in sys.stdin:
    pass
```

default 환경변수 예시

environment variables
(Job / Task config)

Mapper (Python): mapper.py

```
os.environ["mapreduce_task_id"]
os.environ["mapreduce_task_partition"]
```

1 task_1488734338480_1443_m_000001
8 task_1488734338480_1443_r_000008

Logged in as: dr.who

Map Tasks for job_1488734338480_1443

- Application
- Job
  - Overview
  - Counters
  - Configuration
  - Map tasks
  - Reduce tasks
- Tools

Show 20 entries Search:

Task					Successful Attempt		
Name	State	Start Time	Finish Time	Elapsed Time	Start Time	Finish Time	Elapsed Time
task_1488734338480_1443_m_000000	SUCCEEDED	Sat Mar 25 19:57:01 +0300 2017	Sat Mar 25 19:57:05 +0300 2017	4sec	Sat Mar 25 19:57:01 +0300 2017	Sat Mar 25 19:57:05 +0300 2017	4sec
task_1488734338480_1443_m_000001	SUCCEEDED	Sat Mar 25 19:57:01 +0300 2017	Sat Mar 25 19:57:05 +0300 2017	4sec	Sat Mar 25 19:57:01 +0300 2017	Sat Mar 25 19:57:05 +0300 2017	4sec
ID	State	Start Time	Finish Time	Elapsed Time	Start Time	Finish Time	Elapsed Time

Showing 1 to 2 of 2 entries First Previous 1 Next Last

```
from __future__ import print_function
import re
import sys

CHARS_IN_LINE = 9

if os.environ["mapred_task_is_map"] == "true":
split_input_start = int(
        os.environ["mapreduce_map_input_start"]
   ) // CHARS_IN_LINE
```

나눠진 파일의 시작 index (line num)

```
for split_line_index, line in enumerate(sys.stdin):
line_number = split_line_index + split_input_start
   if (line_number < 10):
   print(line_number, line, sep='\t')
```

WIKIPEDIA
The Free Encyclopedia

<article id> <tab> <article content>

key value

Mapper (Python): wc_mapper.py

```
from __future__ import print_function
import os
import re
import sys
```

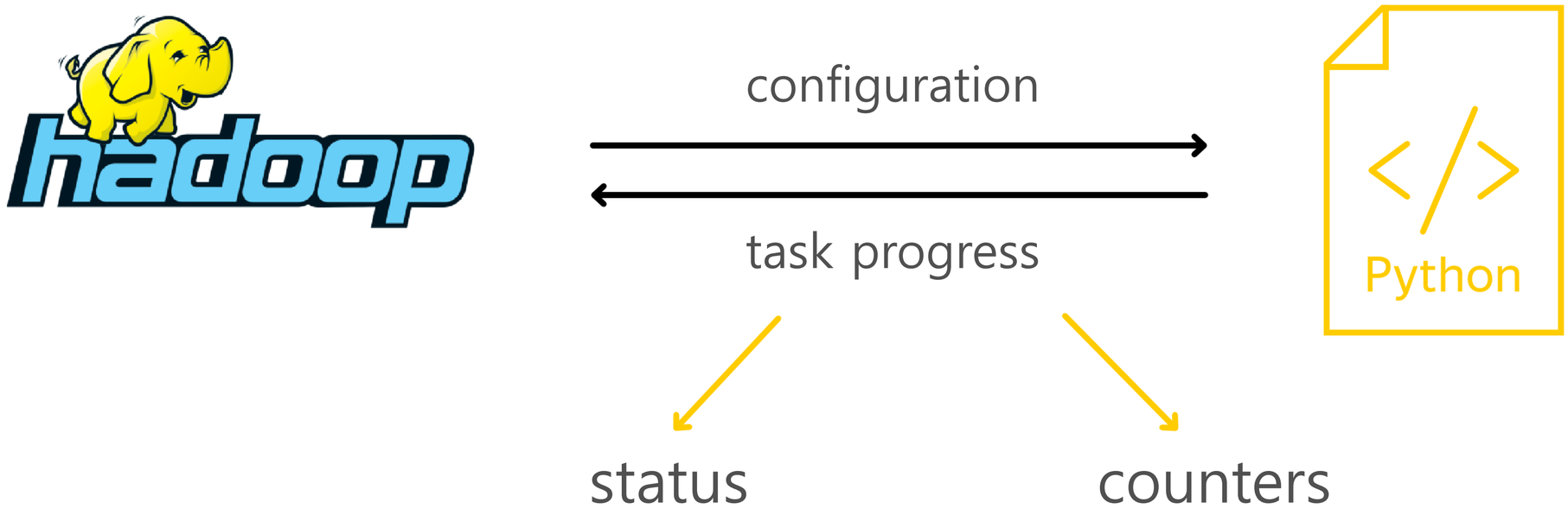
유저가 생성한 환경변수

```
pattern = re.compile(os.environ["word_pattern"])

for line in sys.stdin:
    article_id, content = line.split("\t", 1)
    words = re.findall(pattern, content)
    for word in words:
        print (word,1, sep="\t")
```

유저가 생성한 환경변수 -D를 활용

```
yarn jar $HADOOP_STREAMING_JAR -D word_pattern="\w+\d+" \
                    -files mapper.py \
                    -mapper 'python mapper.py' \
                    -reducer 'python reducer.py' \
                    -input /data/wiki/en_articles \
                    -output word_count
```

```
$ hdfs dfs -text word_count/*
…
test2   4
times11 1
times48 3
tinctoria1      1
titan2
…
```



반대로 Script 코드에서 Hadoop에게 status와 counters 값을 전달할 수 있다.

Status example

Show 20 entries

Attempt	State	Status
attempt_1488734338480_1448_m_000000_0	SUCCEEDED	processed 6374 words
attempt_1488734338480_1448_m_000001_0	SUCCEEDED	processed 1778 words
Attempt	State	Status

Showing 1 to 2 of 2 entries

Mapper (Python): reporter_mapper.py

```
from __future__ import print_function
import re
import sys

for line in sys.stdin:
    article_id, content = line.split("\t", 1)
    words = re.findall("\w+", content)
    for index, word in enumerate(words):
            print(word, 1, sep="\t")
            print("reporter:status:processed {} words"
                .format(index + 1), file=sys.stderr)
```

현재 worker의 상태를 표준출력으로 전달한다.

Mapper (Python): reporter_mapper.py

Counter example

```
from __future__ import print_function
import re
import sys

for line in sys.stdin:
    article_id, content = line.split("\t", 1)
    words = re.findall("\w+", content)
    for index, word in enumerate(words):
            print(word, 1, sep="\t")
            print("reporter:status:processed {} words"
                .format(index + 1), file=sys.stderr)

print(" reporter:counter:Personal Counters,word found,1", file=sys.stderr)
```

reporter:counter:<group>,<counter>,<amount>

각 worker에서 보내는 amount의 합계를 보여준다.

Map-Reduce Framework	Map output materialized bytes	9634808	0	9634808
	Map output records	12396473	0	12396473
	Merged Map outputs	0	20	20
	Physical memory (bytes) snapshot	3370938368	2752958464	6123896832
	Reduce input groups	0	306456	306456
	Reduce input records	0	12396473	12396473
	Reduce output records	0	306456	306456
	Reduce shuffle bytes	0	9634808	9634808
	Shuffled Maps	0	20	20
	Spilled Records	12396473	12396473	24792946
	Total committed heap usage (bytes)	3642228736	6243221504	9885450240
	Virtual memory (bytes) snapshot	7705399296	83205345280	90910744576
Personal Counters	Name	Map	Reduce	Total
	word found	12396473	0	12396473

Summary

- You know how to:
  - provide environment variables (global configuration) to your streaming scripts
  - access job configuration options (e.g. map input file)
  - report progress back to Hadoop MapReduce Framework

MapReduce

Testing

Distributed Cache

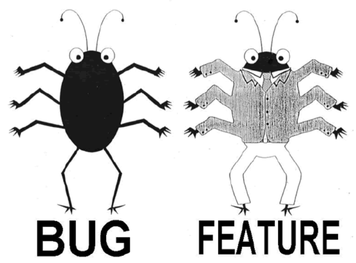
configuration

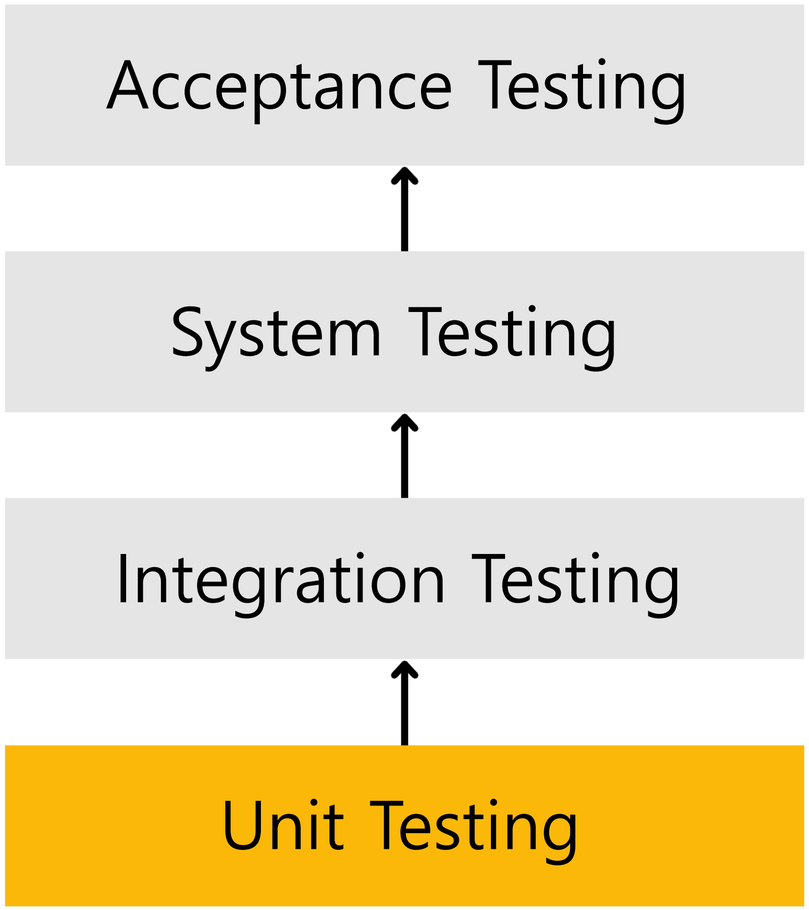
task progress

status

counters

Python





https://wiki.python.org/moin/PythonTestingToolsTaxonomy

python™

» PythonTestingToolsTaxonomy

» PythonTesti...olsTaxonomy

FrontPage
RecentChanges
FindPage
HelpContents
PythonTesti...olsTaxonomy

Page

- Immutable Page
- Info
- Attachments
- More Actions:

User

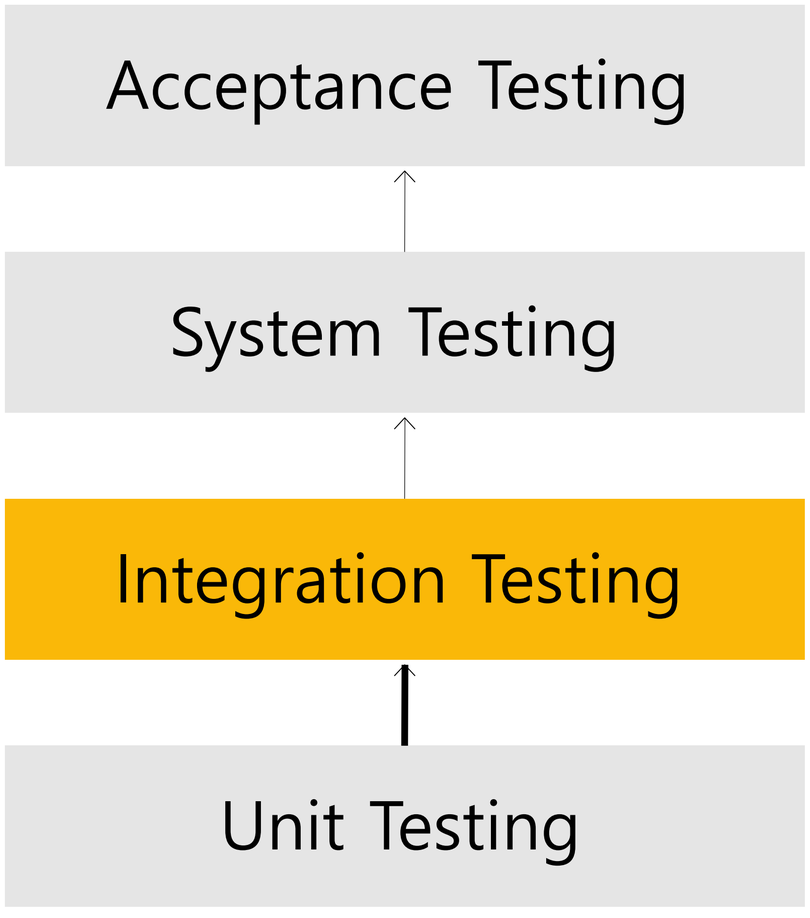
Contents

1. Unit Testing Tools
2. Mock Testing Tools
3. Fuzz Testing Tools
4. Web Testing Tools
5. Acceptance/Business Logic Testing Tools
6. GUI Testing Tools
7. Source Code Checking Tools
8. Code Coverage Tools
9. Continuous Integration Tools
10. Automatic Test Runners
11. Test Fixtures
12. Miscellaneous Python Testing Tools

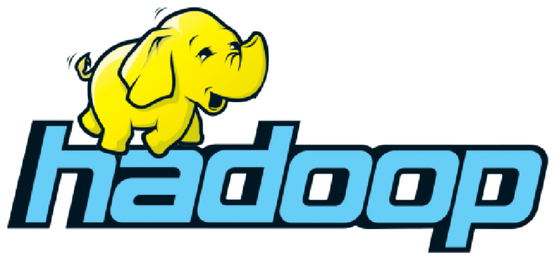
```python
def get_words(input_line):
    return input_line.split()

def test_get_words_parse_simple_string():
    assert get_words("a b cd efg ") == ["a", "b", "cd", "efg"]

def test_get_words_parse_empty_string():
    assert get_words("") == []

def test_get_words_raise_exception_if_no_input():
    with pytest.raises(AttributeError):
        get_words(None)
```

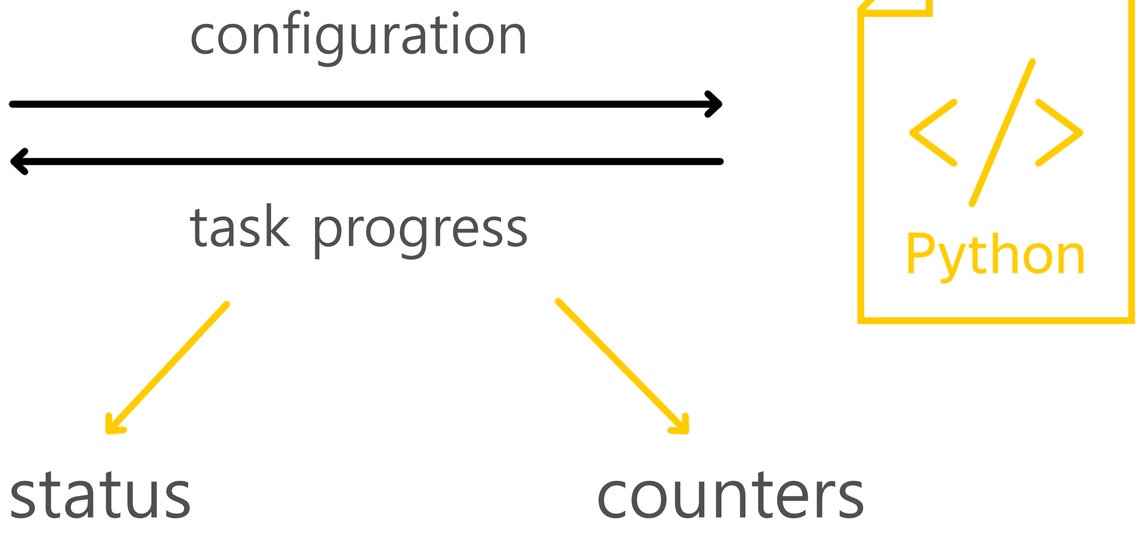
```
$ cat -n wikipedia.dump | tr ' ' '\n' | sort | uniq -c
```

```
$ cat | mapper | sort | reducer
```



Unit 을 조합해가며 파이프라인을 테스트하는 단계





Acceptance Testing

↑

System Testing

↑

Integration Testing

↑

Unit Testing

하나의 node에서 모든 task를 수행해보는 단계
(task가 잘 돌아가는 지 체크)

Hadoop/conf.empty를 활용해야 적용이 가능하다.

```
hdfs --config $HADOOP_EMPTY_CONFIG dfs -rm -r word_count
yarn --config $HADOOP_EMPTY_CONFIG jar $HADOOP_STREAMING_JAR
```

```
$ locate "hadoop/conf.empty"
```

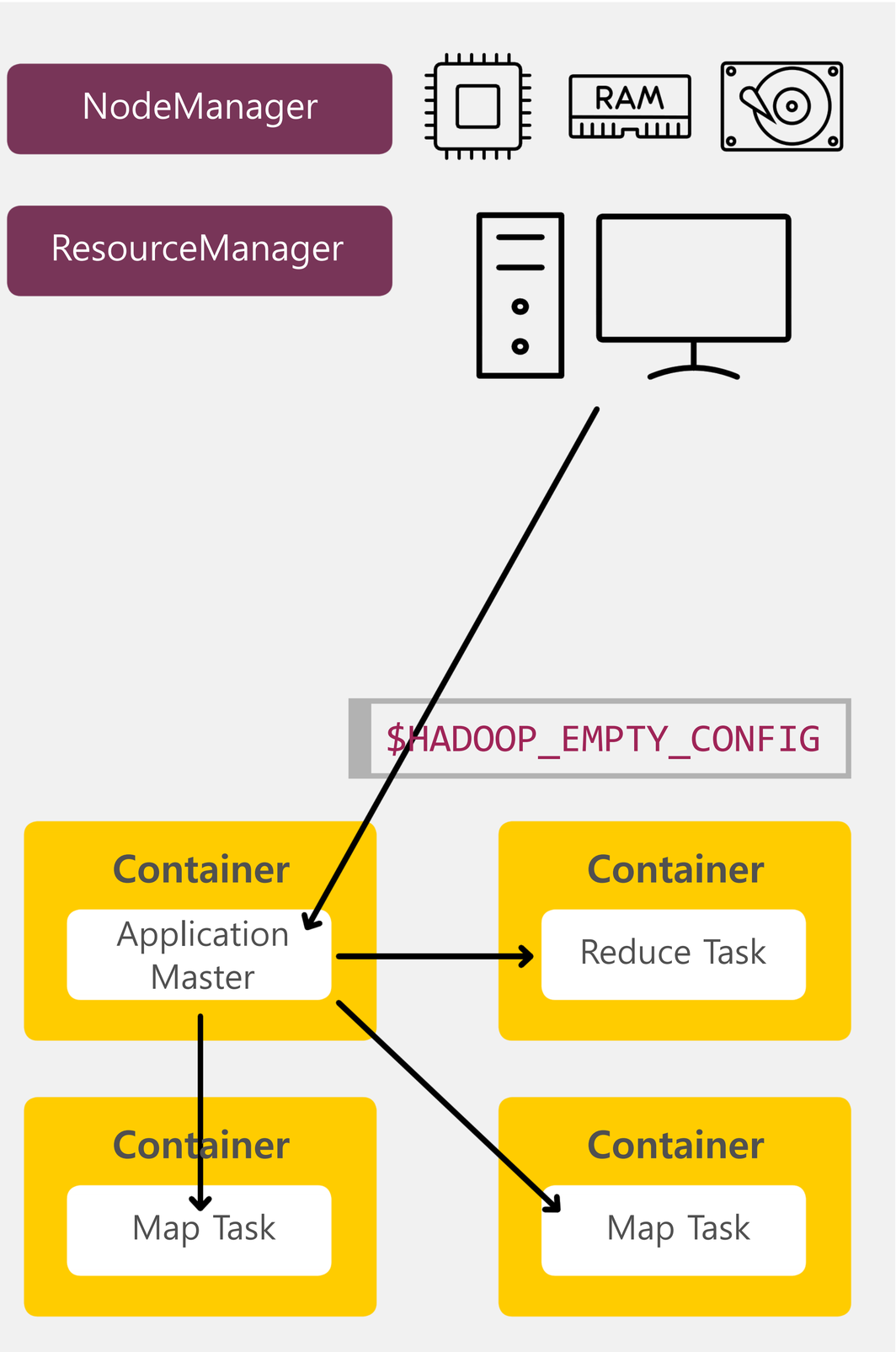
Acceptance Testing

System Testing

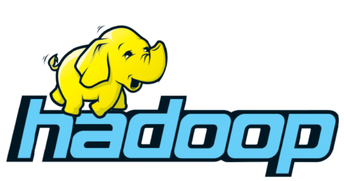
Integration Testing

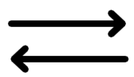
Unit Testing

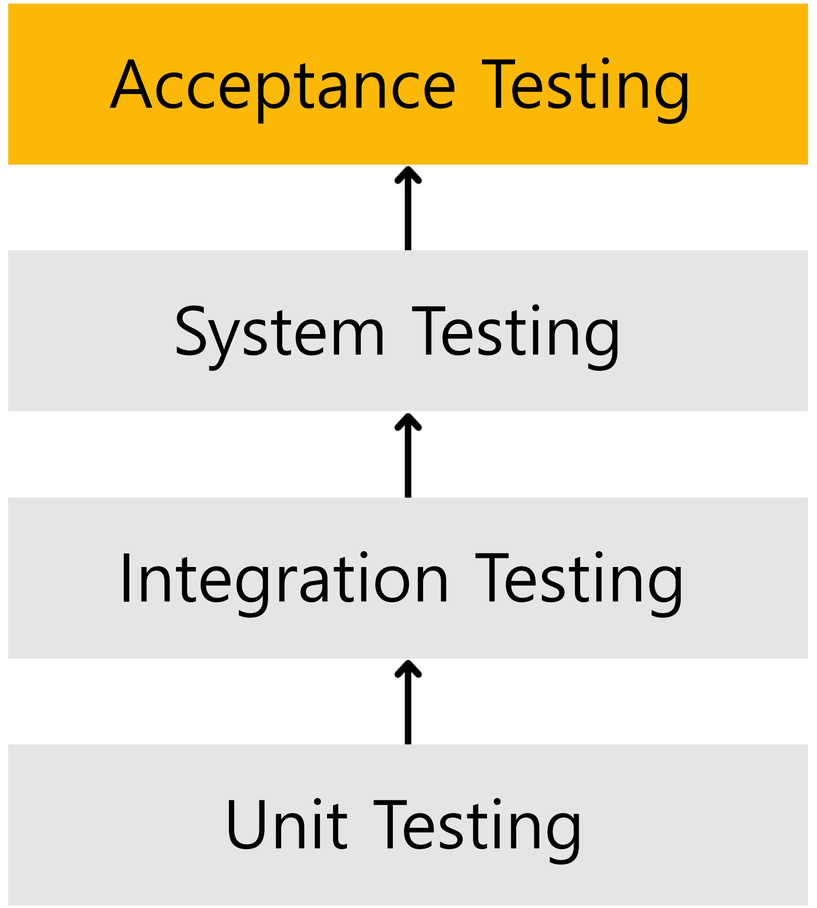
```
        Personal Counters
                word found=182
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=197193
        File Output Format Counters
                Bytes Written=769
17/03/26 18:10:13 INFO streaming.StreamJob: Output directory: w
```



Small Dataset으로 테스트하는 단계







Summary

- you know how to use Python unit testing
- you know how to emulate MapReduce locally with (cat | map | sort | reduce)
- you know how to run MapReduce in a standalone mode (hadoop/conf.empty)
- you know why you need to execute MapReduce against sample datasets

BigDATAteam