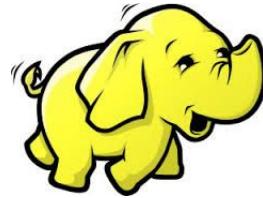


# BIG DATA



DATA IS THE NEW OIL

# Who am I ?

# Chinnasamy

Consultant - Big Data Architect

13+ Years of expertise in Java/J2EE, Scala, Python and Big Data

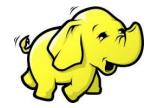
- ~ Wipro
- ~ General Electric
- ~ Verizon
- ~ Infosoft Join
- ~ Jipree
- ~ Vunya
- ~ Evolv
- ~ Scholastic
- ~ iPRO
- ~ Equatortek
- ~ Relevance Lab
- ~ Mobax
- ~ HCL - AADHAAR Project
- ~ Metrolinx

As a consultant consulted for more than 15+ companies.

Conducted more than 40+ training programs on Big Data and allied technologies.

[chinnasamyad@gmail.com](mailto:chinnasamyad@gmail.com)

+91 96000 45955.



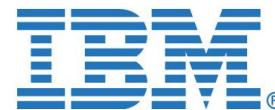
# Chinnasamy

Consultant - Big Data Architect

[chinnasamyad@gmail.com](mailto:chinnasamyad@gmail.com)

+91 96000 45955.

# Who am I ?



Training Partners

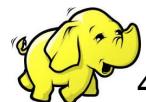


# What's inside ?

Topic	Slide Numbers
Big Data	5
Hadoop-HDFS	27
Impala	103
HBase	108
Python	160
Scala	191
Import code using eclipse	424
Spark	503
Pycharm-Spark IDE Integration	462
Spark SQL	659

Topic	Slide Numbers
ORC	701
Kafka	750
ELK Stack	760
Twitter	775
Natural Language Processing	778
Hortonworks	821
Cassandra	822
Zookeeper	830
Kafka	847

Topic	Slide Numbers
ELK Stack	979
Kibana	994
Solr	1006



# What is Big Data ?

Any system which dictates the necessity of one 'V' said below

A big data environment has one or more of the following characteristics : Volume, Velocity, Variety and Veracity.

Volume	Facebook generates 500 TB/day
Velocity	One million events per second
Variety	Geospatial Data, Audio, Video data etc.,
Veracity	Data in doubt. Uncertainty due to data inconsistency and incompleteness.

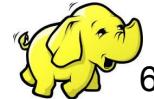
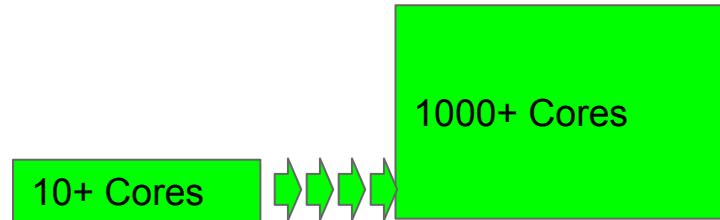
# Current trends in Big Data

Facebook      350 millions images uploaded every day.

Walmart      2.5 PB of customer data hourly.

Youtube      300 hours of video uploaded every minute.

The world of high CPU and GPU.



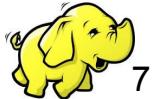
# Data unit measurement

Bit	1 Bit
Byte	8 Bits
KiloByte	1024 bytes
MegaByte	1024 Kilobytes
Gigabyte	1024 Megabytes
Terabyte	1024 Gigabytes
Petabyte	1024 Terabytes
Exabyte	1024 Petabytes
Zettabyte	1024 Exabytes
Yottabyte	1024 Zettabytes
Bronabyte	1024 Yottabytes
Geopbyte	1024 Bronabytes

AADHAAR data is approximately 10 PB.

A gram of DNA will be around 450 exabytes.

In 2013, approximately the WWW has reached around 4 zettabytes.

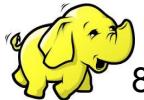


# Why Big Data now ?

SmartPhones, connectivity with Internet/ IoT (Internet of Things), Social Networks etc.., has generated tons of data. Moreover the devices used for accessing became cheaper, faster and smaller.

This made the data manipulation a challenge paving way for a new trend of data management called - The Big Data.

If one ox can plough a field of 1 acre of land and if a requirement pops up to plough a 10 acre land, you cannot grow an ox that is 10 times big. Instead use 10 oxen to plough the 10 acre land and this phenomenon is basically done by any big data system.



# CAP Theorem

C - Consistency

A - Availability

P - Partition Tolerance

Consistency : stating that all the nodes in the system see the same data at the same time.

Availability : stating every request receives the response.

Partition Tolerance : If any node fails, then the system continues to operate with the help of other nodes.

Cassandra, Voldemort is a AP system.

HBASE is a CP system.



# Is 1 PB Big Data ?

It depends on what I am going to do with the one petabyte of data.

If i am not going to operate on at least few terabytes,  
then there is no need for big data environment.

# Types of Architectures in Big Data

Lambda Architecture

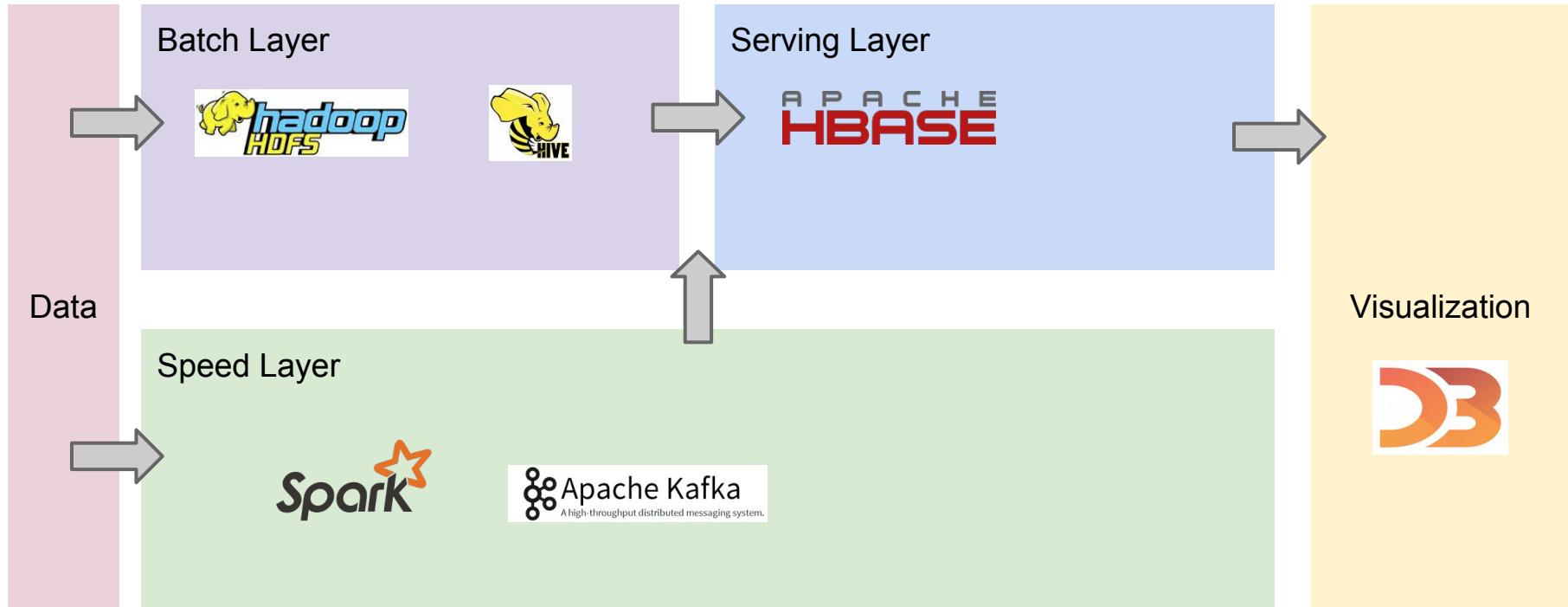
Kappa Architecture

Staged Event Driven Architecture (SEDA)

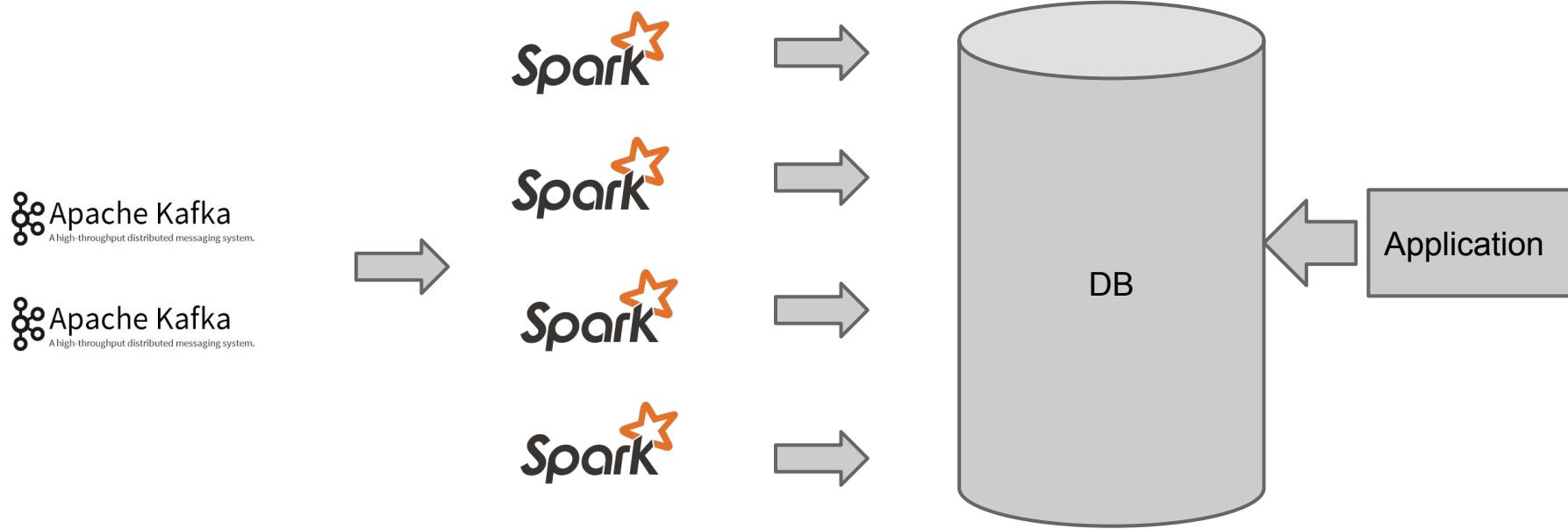
Real Time Streaming Architecture

Zeta Architecture

# Lambda Architecture



# Kappa Architecture



# SEDA Architecture

## Staged Event Driven Architecture



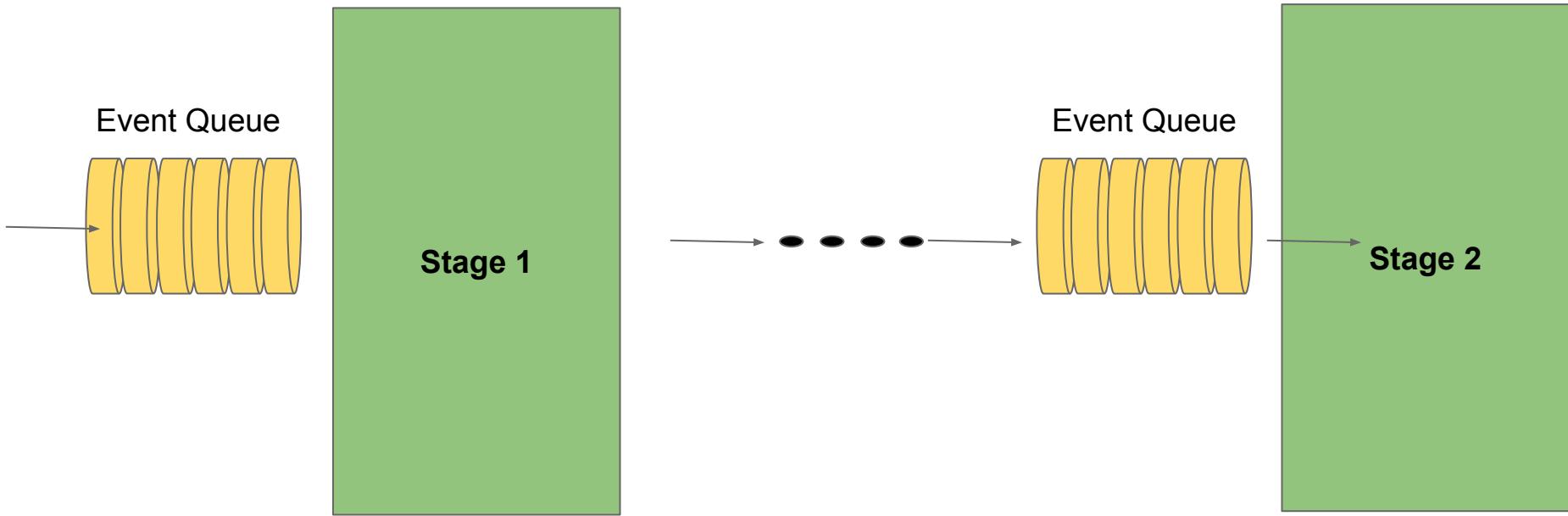
SEDA is a software design that decomposes a complex, event driven application into a set of stages connected by queues.

Every stage in SEDA is autonomous and connected to the next one by infrastructure and not by code.

All stages of the aadhaar are stateless and message driven in a request response model. Each stage is given complete data that it requires for execution.

# SEDA Architecture

Staged Event Driven Architecture



# Zeta Architecture



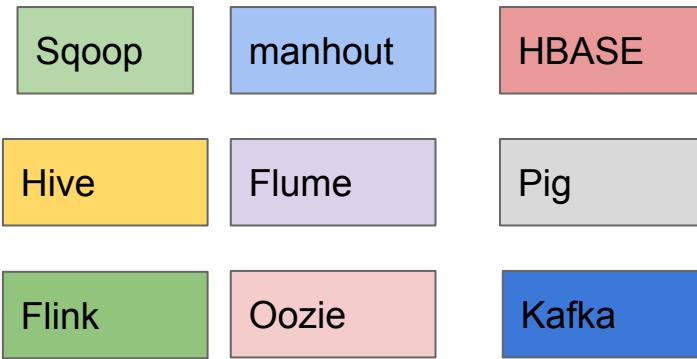
# Myths of Big Data

Once a myth. Not anymore....

Will be the future : It's not the future, it is already the present

We will build it, when we need it : Never. If you can forecast your data size, start building it right now.

# Hadoop Ecosystem



HDFS - Stores data in the cluster  
Namenode - manages hdfs and the data

Mapreduce - process data in the cluster  
JobTracker - manages Mapreduce



# NoSQL Stores

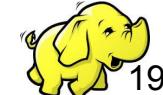
Aravindh Chinnasamy

## Key-Value Store



## Columnar Store

## Document Store



# Memcache - Redis comparison



An in-memory key-value store

Used for operating caches on small footprints of data, mostly strings

Simple LRU cache eviction policy

No replication support

An in-memory key-value store

Used for operating on heavy data structure objects.

Fine grained LRU cache policies

noeviction, allkeys-lru, volatile-lru, allkeys-random, volatile-random, volatile-ttl

Supports master-slave replication, which is a major advantage in a big data system.

# Memcache - Redis comparison



Primary used for reducing the load on the database.

Used for key-value retrieval for all types of keys.

Limited key-name size

Comparatively larger key-size when compared to memcache.

Cannot store data to disk.

Optional and tunable data persistence.

# Data Lakes

- [\*] Nothing but a HDFS Implementation.
- [\*] People also use Amazon S3.
- [\*] Write everything, don't purge anything. Data is the King.
- [\*] Corporations also use Enterprise Data Lakes for their organizations.



# Messaging System

Kafka

RabbitMQ

ActiveMQ

ZeroMQ

# Parallel Threads

```
final ForkJoinPool forkJoinExecutor = new ForkJoinPool  
    (Runtime.getRuntime().availableProcessors());  
  
final Object[] destination = new Object[inputList.size ()];  
  
final ComponentTask<X, Y> action = new ComponentTask<> (  
    inputList, destination, function, 0, inputList.size () - 1);  
  
forkJoinExecutor.invoke (action);  
  
final List<Y> resultList = new ArrayList (destination.length);  
  
for (final Object o : destination) {  
    @SuppressWarnings ("unchecked") final Y y = (Y) o;  
    resultList.add (y);
```

# Parallel Threads

```
public class ComponentTask<X, Y> extends RecursiveAction {  
}
```

```
import java.util.concurrent.ForkJoinTask;  
import java.util.concurrent.RecursiveAction;  
import java.util.function.Function;
```

# Parallel Threads

```
@Override
protected void compute () {

    final int bracketSize = (this.end - this.start);

    if (bracketSize < MINIMUM_THRESHOLD) {

        this.computeFully ();

    } else {

        final int middle = this.start + (bracketSize
/ 2);

        final ComponentTask<X, Y> firstSubtask = new
ComponentTask<> (this.list,
                           this.destination,
                           this.function,
                           this.start,
```

# Parallel Threads

```
/**  
 * Do the computation directly.  
 */  
private void computeFully () {  
  
    for (int i = this.start; i <= this.end; i++) {  
        final Y result = this.function.apply (this.list.get  
(i));  
        this.destination[i] = result;  
    }  
}
```

# Hadoop - HDFS



# What is Hadoop ?

Hadoop is the open source implementation of GFS (Google File System) and MR (MapReduce) concept of Google.

Hadoop is a open source, distributed data storage and processing framework built on commodity hardware that makes economics of unstructured data attractive.

Doug Cutting and Mike Cafarella were the creators of hadoop.

Hadoop was the name of Doug Cutting son's toy elephant.

# How hadoop came into existence

2003 - Google releases papers with GFS (Google File System)

2004 - Google releases papers with MapReduce

2005 - Nutch used GFS and MapReduce to perform operations

2006 - Yahoo! created Hadoop based on GFS and MapReduce (with Doug Cutting and team)

2007 - Yahoo started using Hadoop on a 1000 node cluster

2008 - Apache took over Hadoop

2009 - Hadoop successfully sorted a petabyte of data in less than 17 hours to handle billions of searches and indexing millions of web pages.

2011 - Hadoop releases version 1.0

2013 - Version 2.0.6 is available

# Hadoop Distributions

**cloudera**

Pivotal®

**MAPR®**



**Hortonworks**

# HDFS

HDFS stands for Hadoop Distributed File System

It is the file system distributed across the nodes of the cluster.

The data is replicated across the nodes, thereby giving the fault tolerant behavior.

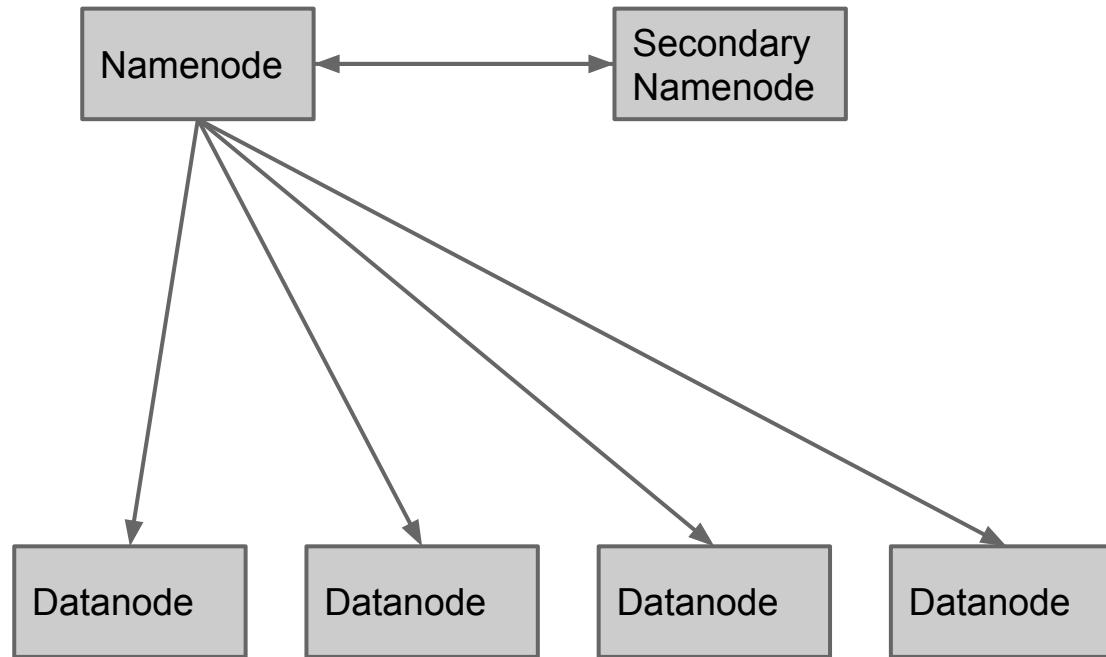
# Namenode and Datanode

Namenode                  Node which contains pointers (references) to data

Datanode                  Node which contains the data.

Secondary namenode      Failsafe node for the namenode.

# Namenode and Datanode



# Namenode

Namenode: The namenode stores metadata of hdfs.

The state of the hdfs is stored in a file called *fsimage*.

During runtime updation, the modifications are written to an edit file called *edits*.

On the next startup of the namenode, the state is read from the fsimage and the changes in the edits are read written back to the fsimage.

Then after which the edits are cleared.

# Checkpoint Namenode & Backup Node

The checkpoint namenode does the merging of fsimage and edits in the runtime.

The Backup node does the same functionality of checkpoint node but it synchronizes the namenode with the help of in-memory computation.

# hdfs commands

```
$ hdfs namenode -format
```

```
$ hdfs dfsadmin -report
```

```
$ hdfs dfs -ls /data
```

```
$ hdfs dfs -cat /data/sshd/wordcount.txt
```

```
$ hdfs dfs -chmod -R 1777 /data/sshd/wordcount.txt
```

# hadoop commands

```
$ ./hadoop fs -ls /
```

```
$ ./hadoop fs -mkdir /aravindh
```

```
$ ./hadoop fs -du /
```

# hadoop commands

- appendToFile
- cat
- checksum
- chgrp
- chmod
- chown
- copyFromLocal
- copyToLocal
- count
- cp
- createSnapshot
- deleteSnapshot
- df
- du
- dus
- expunge
- find
- get
- getfacl
- getfattr
- getmerge
- help
- ls
- lsr
- mkdir
- moveFromLocal
- moveToLocal
- mv
- put
- renameSnapshot
- rm
- rmdir
- rmr
- setfacl
- setfattr
- setrep
- stat
- tail
- test
- text

# Upload a file to hdfs

```
private static String HDFS_PATH = "hdfs://localhost:9000";  
  
Configuration conf = new Configuration();  
  
BufferedReader inputStream =null ;  
inputStream = new BufferedReader(new FileReader("/Users/tester/myfile.csv"));  
  
FileSystem hdfs = FileSystem.get( new URI( HDFS_PATH ), conf );  
Path file = new Path("/data/destination_file.csv");  
  
OutputStream os = hdfs.create( file);  
  
BufferedWriter br = new BufferedWriter( new OutputStreamWriter( os, "UTF-8" ) );  
  
String line = null;  
  
while ((line = inputStream.readLine()) != null) {  
    br.write(line);  
}  
  
br.close();  
hdfs.close();
```

# Configuring HDFS

The config files core-site.html and hdfs-site.xml are in hadoop\_installation [/etc/hadoop](#) directory.

Step 1: core-site.xml (in etc/hadoop)

Step 2: hdfs-site.xml (in etc/hadoop)

Step 3: ./hdfs namenode -format

Step 4: ./start-dfs.sh

# core-site.xml

```
<configuration>

    <property>
        <name>hadoop.tmp.dir</name>
        <value>/Users/tester/hadoopdata</value>
    </property>

    <property>
        <name>fs.defaultFS</name>
        <value>hdfs://localhost:9000</value>
    </property>

</configuration>
```

# hdfs-site.xml

```
<configuration>

    <property>
        <name>dfs.replication</name>
        <value>1</value>
    </property>

    <property>
        <name>dfs.datanode.data.dir</name>
        <value>/Users/tester/hadoopdata/hdfs</value>
    </property>

</configuration>
```

# Do namenode formatting and start hdfs

Goto hadoop\_installation/bin

./hdfs namenode -format

Goto hadoop\_installation/sbin

./start-dfs.sh

```
Mynenis-MacBook-Pro:hadoop koteshwar$ jps
8481 Jps
8118 DataNode
8218 SecondaryNameNode
8038 NameNode
Mynenis-MacBook-Pro:hadoop koteshwar$
```

```
$ ssh-keygen -t rsa -P "" -f ~/.ssh/id_rsa
$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
$ chmod 0600 ~/.ssh/authorized_keys
```

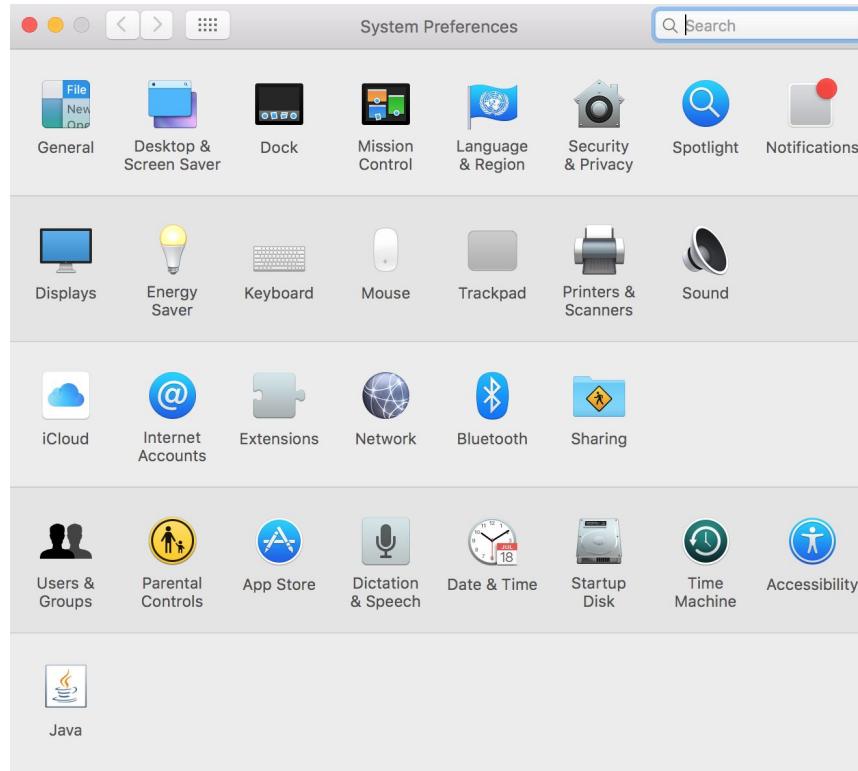
# Do namenode formatting and start hdfs

`./start-dfs.sh`

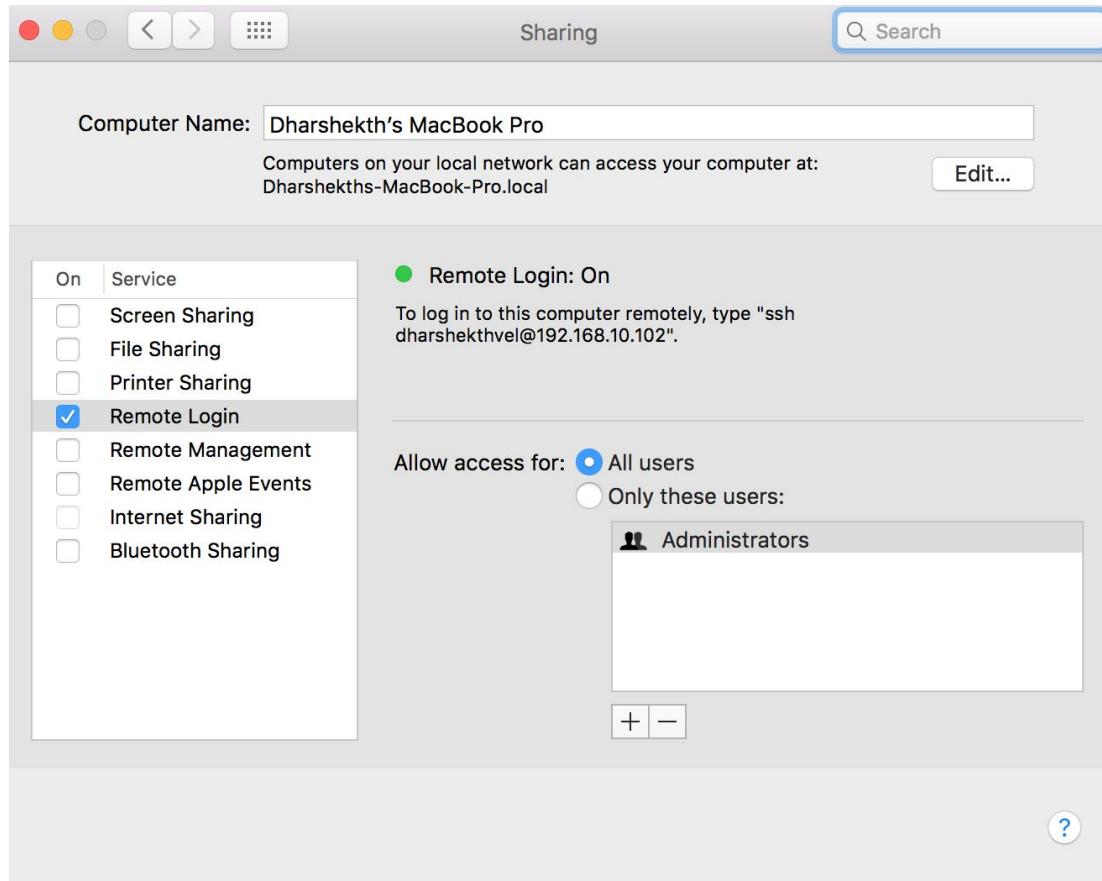
*While starting hadoop with start-dfs.sh if you are getting error ensure, JAVA\_HOME is set in hadoop-env.sh.*

`/home/dharshekthvel/ac/bin/hadoop3/etc/hadoop/hadoop-env.sh`

To enable ssh in mac, go to **System Preferences**,  
and click on **Sharing**.



Click on Remote Login and give access to All users.



```
SCHMAC-TESTER-3:hadoop27 tester$ ls
LICENSE.txt      README.txt      etc      include      libexec      sbin
ree
NOTICE.txt      bin      hadoop-2.7.0-src  lib      logs      share
SCHMAC-TESTER-3:hadoop27 tester$ █
```

# Directory Structure of Hadoop

```
SCHMAC-TESTER-3:hadoop27 tester$ ./tree bin  
bin  
|___container-executor  
|___hadoop  
|___hadoop.cmd  
|___hadoopdata  
|___hdfs  
|___hdfs.cmd  
|___mapred  
|___mapred.cmd  
|___rcc  
|___test-container-executor  
|___yarn  
|___yarn.cmd  
SCHMAC-TESTER-3:hadoop27 tester$ ./tree sbin  
sbin  
|___distribute-exclude.sh  
|___hadoop-daemon.sh  
|___hadoop-daemons.sh  
|___hdfs-config.cmd  
|___hdfs-config.sh  
|___httpfs.sh  
|___kms.sh  
|___mr-jobhistory-daemon.sh  
|___refresh-namenodes.sh  
|___slaves.sh  
|___start-all.cmd  
|___start-all.sh  
|___start-balancer.sh  
|___start-dfs.cmd  
|___start-dfs.sh  
|___start-secure-dns.sh  
|___start-yarn.cmd  
|___start-yarn.sh  
|___stop-all.cmd  
|___stop-all.sh  
|___stop-balancer.sh  
|___stop-dfs.cmd  
|___stop-dfs.sh  
|___stop-secure-dns.sh  
|___stop-yarn.cmd  
|___stop-yarn.sh  
|___yarn-daemon.sh  
|___yarn-daemons.sh  
SCHMAC-TESTER-3:hadoop27 tester$
```

# Directory Structure of Hadoop

SCHMAC-TESTER-3:hadoop27 tester\$ ./tree etc

etc  
|\_\_ hadoop  
| |\_\_ capacity-scheduler.xml  
| |\_\_ configuration.xsl  
| |\_\_ container-executor.cfg  
| |\_\_ core-site.xml  
| |\_\_ hadoop-env.cmd  
| |\_\_ hadoop-env.sh  
| |\_\_ hadoop-metrics.properties  
| |\_\_ hadoop-metrics2.properties  
| |\_\_ hadoop-policy.xml  
| |\_\_ hdfs-site.xml  
| |\_\_ httpfs-env.sh  
| |\_\_ httpfs-log4j.properties  
| |\_\_ httpfs-signature.secret  
| |\_\_ httpfs-site.xml  
| |\_\_ kms-acls.xml  
| |\_\_ kms-env.sh  
| |\_\_ kms-log4j.properties  
| |\_\_ kms-site.xml  
| |\_\_ log4j.properties  
| |\_\_ mapred-env.cmd  
| |\_\_ mapred-env.sh  
| |\_\_ mapred-queues.xml.template  
| |\_\_ mapred-site.xml  
| |\_\_ mapred-site.xml.template  
| |\_\_ slaves  
| |\_\_ ssl-client.xml.example  
| |\_\_ ssl-server.xml.example  
| |\_\_ yarn-env.cmd  
| |\_\_ yarn-env.sh  
| |\_\_ yarn-site.xml

chin

SCHMAC-TESTER-3:hadoop27 tester\$

# Directory Structure of Hadoop



```
mynenis-MacBook-Pro:hadoop koteshwars$  
Mynenis-MacBook-Pro:hadoop koteshwars$  
Mynenis-MacBook-Pro:hadoop koteshwars$ cat core-site.xml  
<?xml version="1.0" encoding="UTF-8"?>  
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>  
<!--  
 Licensed under the Apache License, Version 2.0 (the "License");  
 you may not use this file except in compliance with the License.  
 You may obtain a copy of the License at  
  
 http://www.apache.org/licenses/LICENSE-2.0  
  
 Unless required by applicable law or agreed to in writing, software  
 distributed under the License is distributed on an "AS IS" BASIS,  
 WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
 See the License for the specific language governing permissions and  
 limitations under the License. See accompanying LICENSE file.  
-->  
  
<!-- Put site-specific property overrides in this file. -->  
  
<configuration>  
    <property>  
        <name>fs.default.name</name>  
        <value>hdfs://localhost:9000</value>  
    </property>  
</configuration>  
Mynenis-MacBook-Pro:hadoop koteshwars$  
Mynenis-MacBook-Pro:hadoop koteshwars$  
Mynenis-MacBook-Pro:hadoop koteshwars$  
Mynenis-MacBook-Pro:hadoop koteshwars$ cat hdfs-site.xml  
<?xml version="1.0" encoding="UTF-8"?>  
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>  
<!--  
 Licensed under the Apache License, Version 2.0 (the "License");  
 you may not use this file except in compliance with the License.  
 You may obtain a copy of the License at  
  
 http://www.apache.org/licenses/LICENSE-2.0  
  
 Unless required by applicable law or agreed to in writing, software  
 distributed under the License is distributed on an "AS IS" BASIS,  
 WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
 See the License for the specific language governing permissions and  
 limitations under the License. See accompanying LICENSE file.  
-->  
  
<!-- Put site-specific property overrides in this file. -->  
  
<configuration>  
    <property>  
        <name>dfs.replication</name>  
        <value>1</value>  
    </property>  
</configuration>  
Mynenis-MacBook-Pro:hadoop koteshwars$ █
```

# Configuration of Hadoop

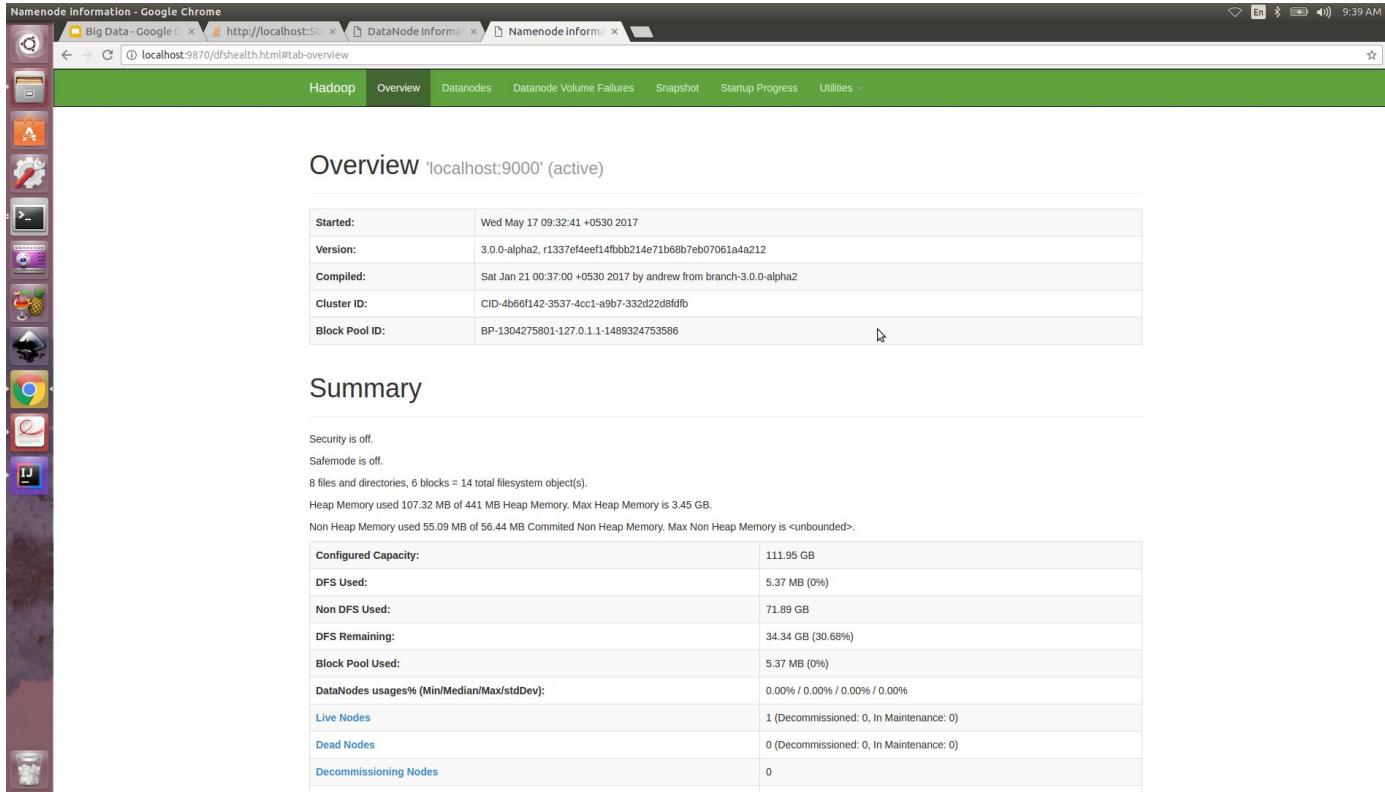
# Starting hadoop

`./start-dfs.sh`

`./start-yarn.sh`

# Starting hadoop - http://localhost:9870/

From Hadoop 3, the port for DFS is on 9870



The screenshot shows a Linux desktop environment with a Unity interface. A Google Chrome window is open, displaying the Hadoop DFS Health Overview page. The URL in the address bar is `http://localhost:9870/dfshealth.html#tab-overview`. The page has a green header with tabs for Hadoop, Overview, Datanodes, Datanode Volume Failures, Snapshot, Startup Progress, and Utilities. The Overview section shows details for the cluster 'localhost:9000' (active). It includes a table with the following data:

Started:	Wed May 17 09:32:41 +0530 2017
Version:	3.0.0-alpha2, r1337ef4ee14fb8b214e71b68b7eb07061a4a212
Compiled:	Sat Jan 21 00:37:00 +0530 2017 by andrew from branch-3.0.0-alpha2
Cluster ID:	CID-4b66f142-3537-4cc1-a9b7-332d22d8fd8b
Block Pool ID:	BP-1304275801-127.0.1.1-1489324753586

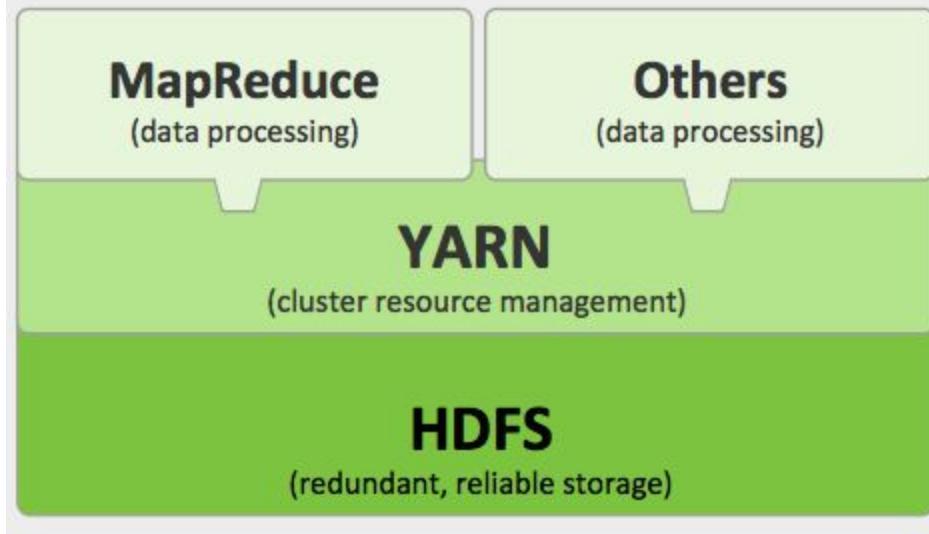
Below the table is a Summary section with the following text:

Security is off.  
Safemode is off.  
8 files and directories, 6 blocks = 14 total filesystem object(s).  
Heap Memory used 107.32 MB of 441 MB Heap Memory. Max Heap Memory is 3.45 GB.  
Non Heap Memory used 55.09 MB of 56.44 MB Committed Non Heap Memory. Max Non Heap Memory is <unbounded>.

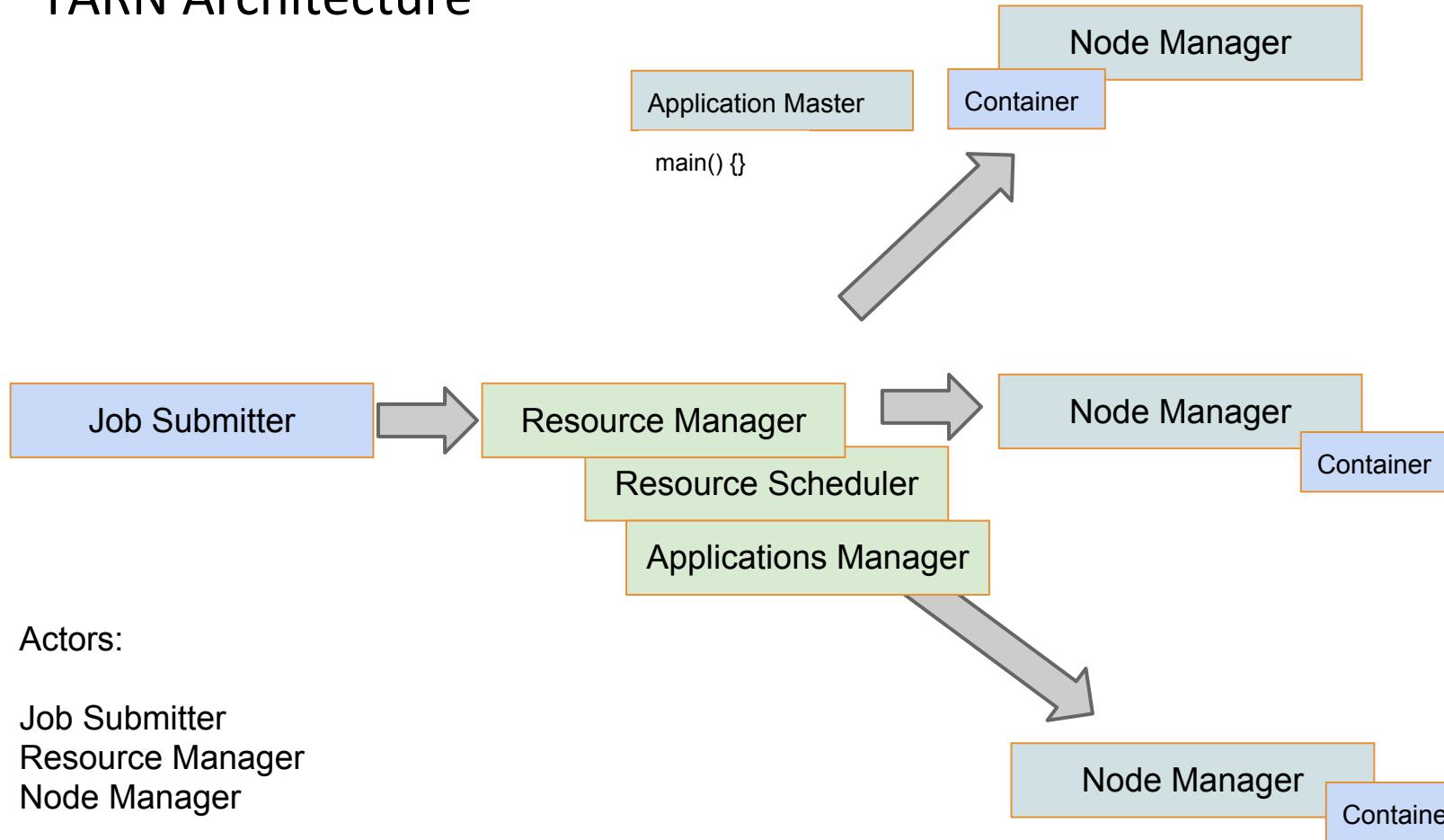
Below the summary is another table with the following data:

Configured Capacity:	111.95 GB
DFS Used:	5.37 MB (0%)
Non DFS Used:	71.89 GB
DFS Remaining:	34.34 GB (30.68%)
Block Pool Used:	5.37 MB (0%)
DataNodes usages% (Min/Median/Max/stdDev):	0.00% / 0.00% / 0.00% / 0.00%
Live Nodes	1 (Decommissioned: 0, In Maintenance: 0)
Dead Nodes	0 (Decommissioned: 0, In Maintenance: 0)
Decommissioning Nodes	0

# HADOOP 2.0



# YARN Architecture



# YARN Architecture

- [1] Client submits an application to the Resource Manager
- [2] Resource manager assigns a container by contacting the Node manager
- [3] The node manager launches the container
- [4] And at last the node manager executes the Application Master

# YARN Architecture

```
localhost:hadoop tester$ cat mapred-site.xml
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
    Licensed under the Apache License, Version 2.0 (the "License");
    you may not use this file except in compliance with the License.
    You may obtain a copy of the License at

        http://www.apache.org/licenses/LICENSE-2.0

    Unless required by applicable law or agreed to in writing, software
    distributed under the License is distributed on an "AS IS" BASIS,
    WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
    See the License for the specific language governing permissions and
    limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->
<configuration>
    <property>
        <name>mapreduce.framework.name</name>
        <value>yarn</value>
    </property>

    <property>
        <name>mapreduce.jobhistory.address</name>
        <value>localhost:10020</value>
        <description>Host and port for Job History Server (default 0.0.0.0:10020)</description>
    </property>
</configuration>
localhost:hadoop tester$ cat yarn-site.xml
<?xml version="1.0"?>
<!--
    Licensed under the Apache License, Version 2.0 (the "License");
    you may not use this file except in compliance with the License.
    You may obtain a copy of the License at

        http://www.apache.org/licenses/LICENSE-2.0

    Unless required by applicable law or agreed to in writing, software
    distributed under the License is distributed on an "AS IS" BASIS,
    WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
    See the License for the specific language governing permissions and
    limitations under the License. See accompanying LICENSE file.
-->
<configuration>
    <property>
        <name>yarn.nodemanager.aux-services</name>
        <value>mapreduce_shuffle</value>
    </property>
</configuration>
localhost:hadoop tester$
```

# Speculative Execution

It is an optimization technique adopted by the mapreduce framework.

When a JobTracker finds out that a task takes too much of time to execute, it can start additional instance of the same task.

This process of execution is called **speculative execution**.

Speculative execution ensures that a slowness in a machine will not slow down a task.

# Speculative Execution

By default, speculative execution is enabled by default. If you want to disable, give in the below properties in the mapred-site.xml.

```
<property>
<name>mapreduce.map.speculative</name>
<value>false</value>
</property>
```

```
<property>
<name>mapreduce.reduce.speculative</name>
<value>false</value>
</property>
```

# Hadoop 3

1. Erasure encoding in HDFS
2. YARN Timeline Service 2.2
3. Support for more than two Namenodes
4. Java 8 is the minimum runtime environment
5. Intra-Datanode Balancer

Erasure coding is a error correcting technique by which the the original data can be constructed using ‘k’ symbols even if ‘m’ symbols are lost. Hadoop 3 uses Reed-Solomon algorithm for erasure coding.



Mapreduce is a method of distributing task across nodes.

In a more elaborate way, mapreduce is a framework for writing in applications that processes tera and peta bytes of data parallelly on many commodity machines.

# Mapper

The mapper performs the first task of a mapreduce

# Reducer

The reducer performs the second task of a mapreduce

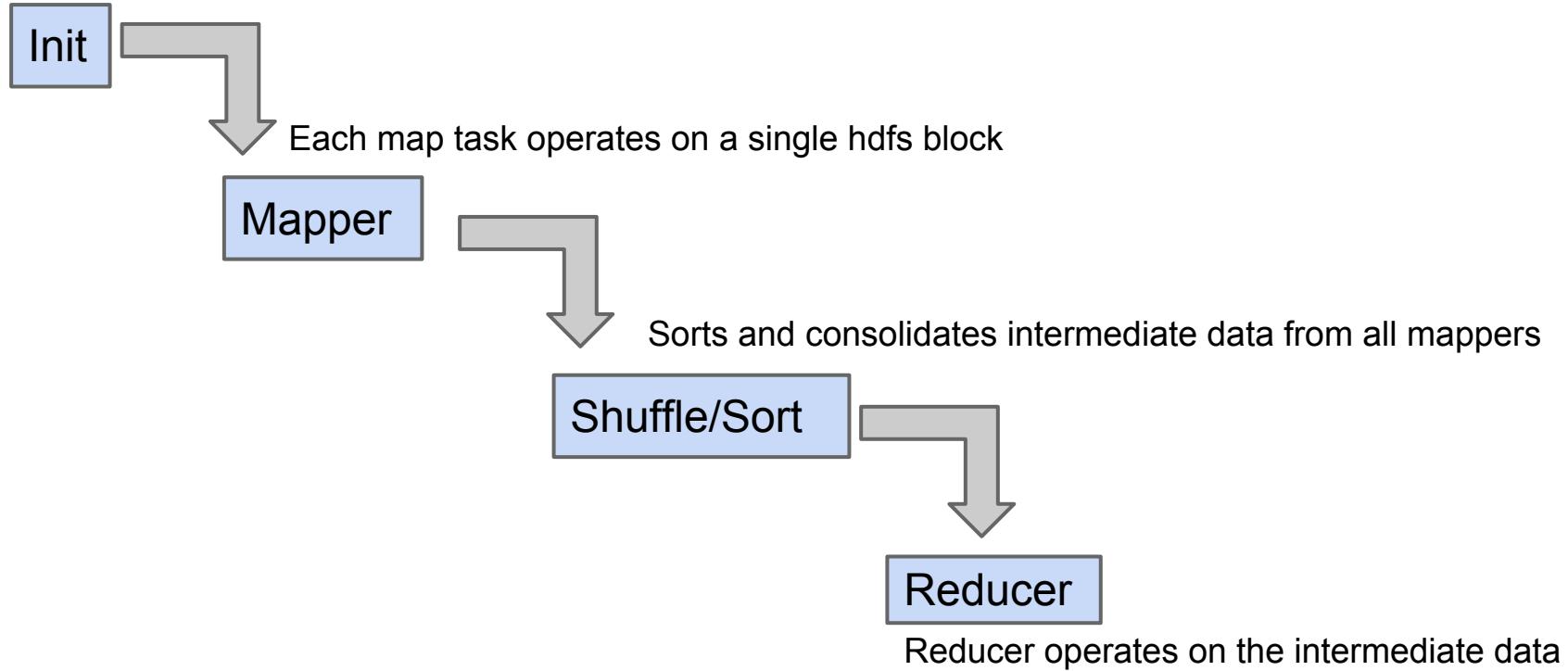
The data is processed to chunks and listed to key-value pairs and then the key-value pairs are given to the reducer

# map

It's a higher order function, that applies the given function to each element in the list.

# reduce

It's a higher order function, that analyze a recursive data structure, recursively processing its constituent parts building up an output.



# init

The input file stored in hdfs is divided into splits. Each split is of a hdfs block. Every split is assigned to a mapper, where the split physically resides.

# mapper

Mapper reads the split of the mapper line by line. Mapper computes the application logic and emits key/value pairs.

# shuffle and sort

Hadoop partitioner divides the emitted output of the mapper into partitions. Hadoop collects all partitions received from the mappers and sort them by key.

# reducer

Hadoop computes the logic in the reduce method and then writes the output to hdfs.

## Input

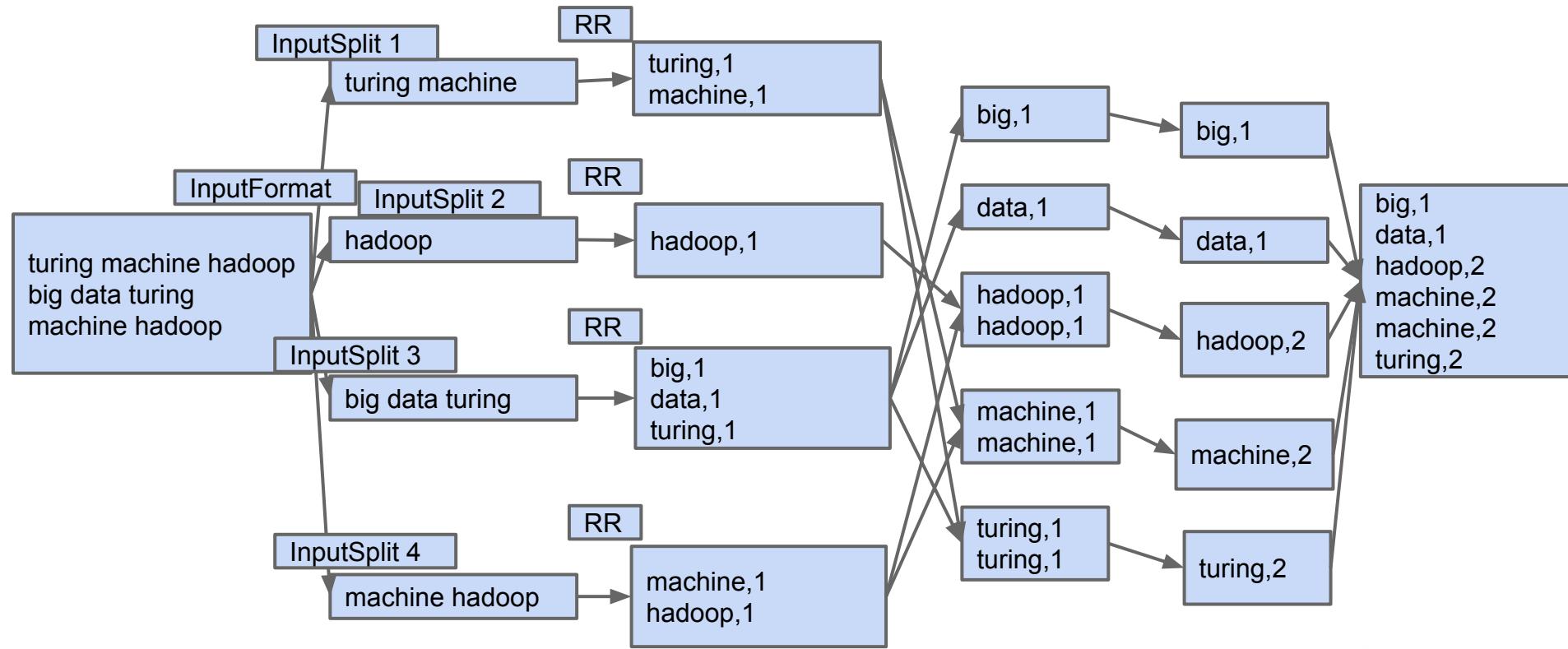
## Splitting

## Mapping

## Shuffling

## Reducing

## Final Result



## **Input Files**

Data for MapReduce task is stored as Input Files. (typically in hdfs)

## **InputFormat**

The protocol of how the input files are split up and read is defined by the Input Format.

The role of InputFormat is that

- ~ it defines the input splits that break a file into tasks.
- ~ provides a factory for RecordReader.

FileInputFormat reads all files. The default InputFormat is the TextInputFormat. The InputFormat can be set in the setInputFormat() method of the configuration of a Job.

TextInputFormat, KeyValueInputFormat,

SequenceFileInputFormat

## **InputSplit**

An InputSplit defines an unit of work that comprises a single map task. Each map task corresponds to a single input split.

## **RecordReader**

InputSplit has defined a slice of work, but does not describe how to access it. The RecordReader class actually loads the data from its source to (key,value) pairs suitable for reading by the mapper.

## **Mapper**

The map() method ejects out (key,value) parameters which are given as input to the reducers. For every map task, a new instance of mapper class is instantiated in a separate java process.

## **Partition & Shuffling**

The process of moving mapper output to the reducers is called shuffling. In order to help shuffling, an intermediate key space is assigned to each reduce node called by Partition. The partitions are the input to the reducers.

## **Sort**

Each key-value pairs are sorted before producing it to any reducer.

## **Reducer**

The code inside the reduce method is called once for each partition.

## **OutputFormat**

The default is TextOutputFormat. The instances of OutputFormat write files to hdfs.

## **RecordWriter**

The OutputFormat is a factory of RecordWriter objects, that write the individual records to the files as dictated by the OutputFormat.

```
class LineTokenizerMapper extends Mapper<Object, Text, Text, IntWritable> {

    private IntWritable one = new IntWritable(1);
    private Text word = new Text();

    @Override
    protected void map(Object key,
                       Text inputLine,
                       Mapper<Object, Text, Text, IntWritable>.Context context)
        throws IOException, InterruptedException
    {

        StringTokenizer tokenizedString = new StringTokenizer(inputLine.toString());

        while (tokenizedString.hasMoreTokens()) {
            word.set(tokenizedString.nextToken().trim());
            context.write(word, one);
        }
    }
}
```

```
class CountReducer extends Reducer<Text,IntWritable,Text,IntWritable> {

    private IntWritable result = new IntWritable();

    @Override
    protected void reduce(Text key, Iterable<IntWritable> values,
        Reducer<Text, IntWritable, Text, IntWritable>.Context context)
        throws IOException, InterruptedException {

        int sum = 0;
        for (IntWritable val : values) {
            sum += val.get();
        }
        result.set(sum);

        context.write(key, result);
    }

}
```

```
public static void main(String[] args) throws IOException {  
  
    Configuration conf = new Configuration();  
  
    Job job = Job.getInstance(conf, "WordCount");  
  
    job.setJarByClass(WordCountJob.class);  
  
    job.setMapperClass(LineTokenizerMapper.class);  
  
    job.setReducerClass(CountReducer.class);  
  
    job.setOutputKeyClass(Text.class);  
  
    job.setOutputValueClass(IntWritable.class);  
  
}  
}
```

# Job Tracker & Task Tracker

**Job Tracker** is the master of the system which manages the jobs and resources of the cluster.

**Task Tracker** are the slaves deployed on each of the machines. They are responsible for running i



# What is pig ?

Pig is a program used for transformation, operation and exploration on large datasets of data.

Pig Latin is the data flow language used to write pig scripts.

Pig power comes from the fact that it can analyze terabytes of code by issuing few lines of code rather than writing complex-boiler plate map-reduce programs.

Developed by Yahoo and now a Apache guy.

# grunt

```
localhost:input tester$  
localhost:input tester$ pig  
15/08/10 13:47:25 INFO pig.ExecTypeProvider: Trying ExecType : LOCAL  
15/08/10 13:47:25 INFO pig.ExecTypeProvider: Trying ExecType : MAPREDUCE  
15/08/10 13:47:25 INFO pig.ExecTypeProvider: Picked MAPREDUCE as the ExecType  
2015-08-10 13:47:25,059 [main] INFO org.apache.pig.Main - Apache Pig version 0.15.0 (r1682971) compiled Jun 01 2015, 11:44:35  
2015-08-10 13:47:25,059 [main] INFO org.apache.pig.Main - Logging error messages to: /Users/tester/examples/sshd/input/pig_1439194645058.log  
2015-08-10 13:47:25,074 [main] INFO org.apache.pig.impl.util.Utils - Default bootup file /Users/tester/.pigbootup not found  
2015-08-10 13:47:25,467 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated. Instead, use mapreduce.jobtracker.address  
2015-08-10 13:47:25,467 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS  
2015-08-10 13:47:25,467 [main] INFO org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - Connecting to hadoop file system at: hdfs://localhost:9000  
2015-08-10 13:47:25,622 java[65450:1903] Unable to load realm info from SCDynamicStore  
2015-08-10 13:47:25,634 [main] WARN org.apache.hadoop.util.NativeCodeLoader - Unable to load native-hadoop library for your platform... using builtin-  
java classes where applicable  
2015-08-10 13:47:26,018 [main] INFO org.apache.hadoop.conf.Configuration.deprecation - fs.default.name is deprecated. Instead, use fs.defaultFS  
grunt>  
grunt>  
grunt>
```

# command line

Runs in the local mode

```
$ pig -x local
```

Runs in the cluster

```
$ pig or $ pig -x mapreduce
```

Running Pig Script as a command line:

```
$ pig -param input=/data/sshd.log pigscript.pig  
pigscript.pig
```

```
sshdLog = LOAD '$input' USING PigStorage AS line:chararray;  
  
sshd_schema = FOREACH sshdLog GENERATE FLATTEN(REGEX_EXTRACT_ALL(line,  
'(.\\s\\d{2}\\s\\d{2}:\\d{2})\\s.*(Accepted.*password.*|\\sfo.*\\s(\\S.*|\\sfr.*\\s(.*)\\spo.*)) as (timestamp:chararray, eventtype:chararray, userid:chararray,  
ipaddr:chararray);  
complete_sshd_schema = FILTER sshd_schema BY timestamp != '';  
grouped_complete_sshd_schema = GROUP complete_sshd_schema BY timestamp;  
STORE grouped_complete_sshd_schema INTO '/data/output_two' USING PigStorage('');
```

Quit from grunt

```
grunt> quit
```



# Load and Store Functions

```
grunt> titanic_data = LOAD '/data/titanic.csv' USING  
    PigStorage(',')
```

AS

```
(Passengerid:int,Survived:int,Pclass:int,LName:chararray,  
FName:chararray,Sex:chararray,Age:int,SibSp:int,Parch:int,  
Ticket:chararray,Fare:double,Cabin:chararray,Embarked  
:chararray);
```

```
grunt> STORE titanic_data INTO '/data/titanic_data_store'  
    USING PigStorage(':'');
```

```
grunt> cat /data/titanic_data_store
```

# Load and Store Functions

- PigStorage : Loads and stores data using field delimiter. It is the default storage.
- BinStorage : Loads and stores data from binary files.
- TextLoader : Loads data from text format.
- JsonLoader, JsonStorage : Loads and stores data from a JSON.
- HBaseStorage : Loads and stores data from HBASE

# grunt supports hadoop filesystem commands

ls

cat

cd

mkdir

rm

```
grunt> ls /data
hdfs://localhost:9000/data/group_four    <dir>
hdfs://localhost:9000/data/group_three   <dir>
hdfs://localhost:9000/data/grouped_complete_sshd_schema <dir>
hdfs://localhost:9000/data/output_one_grouped_complete_sshd_schema      <dir>
hdfs://localhost:9000/data/output_two     <dir>
hdfs://localhost:9000/data/sshd<r 1>    34409657
hdfs://localhost:9000/data/sshd.log<r 1>  34409657
hdfs://localhost:9000/data/sshd_store    <dir>
hdfs://localhost:9000/data/sshd_store_pam_unix <dir>
hdfs://localhost:9000/data/titanic.csv<r 1>  61194
hdfs://localhost:9000/data/titanic_data_store <dir>
grunt>
grunt>
grunt> cat /data/group_four
Nov 01 21:07:02:{(Nov 01 21:07:02,Accepted password,krr,122.172.254.242)}
Nov 02 06:48:39:{(Nov 02 06:48:39,Accepted password,krr,122.172.254.242)}
Nov 02 12:15:04:{(Nov 02 12:15:04,Accepted password,krr,122.172.254.242)}
Nov 02 12:42:52:{(Nov 02 12:42:52,Accepted password,krr,122.172.254.242)}
Nov 04 12:10:09:{(Nov 04 12:10:09,Accepted password,krr,122.172.254.242)}
grunt> cd /data
grunt> ls
hdfs://localhost:9000/data/group_four    <dir>
hdfs://localhost:9000/data/group_three   <dir>
hdfs://localhost:9000/data/grouped_complete_sshd_schema <dir>
hdfs://localhost:9000/data/output_one_grouped_complete_sshd_schema      <dir>
hdfs://localhost:9000/data/output_two     <dir>
hdfs://localhost:9000/data/sshd<r 1>    34409657
hdfs://localhost:9000/data/sshd.log<r 1>  34409657
hdfs://localhost:9000/data/sshd_store    <dir>
hdfs://localhost:9000/data/sshd_store_pam_unix <dir>
hdfs://localhost:9000/data/titanic.csv<r 1>  61194
hdfs://localhost:9000/data/titanic_data_store <dir>
grunt>
```



# Load and Store Functions

If you are running in a local mode, you can give in a file too

```
grunt> sshdLog = LOAD
```

```
'file:/Users/tester/examples/sshd/input/sshd.log'
```

```
USING PigStorage AS line:chararray;
```

# DUMP and DESCRIBE

The contents of the alias or relations are examined using the dump function and the describe operator gives the structure of the relation.

```
grunt>
```

```
titanic_data = LOAD '/data/titanic.csv'  
USING PigStorage(',')  
AS  
(Passengerid:int,Survived:int,Pclass:int,LName:chararray,FName:chararray,Sex:chararray,Age:int,SibSp:int,Parch:int,Ticket:chararray  
,Fare:double,Cabin:chararray,Embarked:chararray);
```

```
grunt> dump titanic_data;
```

# DUMP and DESCRIBE

```
grunt>
```

```
titanic_data = LOAD '/data/titanic.csv'  
USING PigStorage(',')  
AS  
(Passengerid:int,Survived:int,Pclass:int,LName:chararray,FName:chararray,Sex:chararray,Age:int,SibSp:int,Parch:int,Ticket:chararray  
,Fare:double,Cabin:chararray,Embarked:chararray);
```

```
grunt> describe titanic_data;
```

```
titanic_data: {Passengerid: int,Survived: int,Pclass: int,LName: chararray,FName: chararray,Sex: chararray,Age: int,SibSp: int,Parch:  
int,Ticket: chararray,Fare: double,Cabin: chararray,Embarked: chararray}
```

# Pig Data Types

Numeric                    int, long, float, double

Text                      chararray

Binary                    bytearray

Complex                  tuple, bag, map

# Illustrate and Explain

Explain gives in the logical and physical plan

```
grunt> explain grouped_complete_sshd_schema;
```

Illustrate gives in the sample execution of the logical plan with a sample output

```
grunt> illustrate grouped_limited_schema;
```

# Explain

```
# Map Reduce Plan
#-----
MapReduce node scope-600
Map Plan
grouped_complete_sshd_schema: Local Rearrange[tuple]{chararray}{false} - scope-597
| |
| Project[chararray][0] - scope-598
|---complete_sshd_schema: Filter[bag] - scope-591
| |
| Not Equal To[boolean] - scope-594
|---Project[chararray][0] - scope-592
|---Constant() - scope-593
|---sshd_schema: New For Each(true)[bag] - scope-590
| |
| POUUserFunc(org.apache.pig.builtin.REGEX_EXTRACT_ALL)[tuple] - scope-588
| |
|---Cast[chararray] - scope-586
| |
|---Project[bytearray][0] - scope-585
|---Constant((.*\s\d{2}\s\d{2}:\d{2}:\d{2})\s*(Accepted.*password.*)\sfo.*\s(\S.*)\sfr.*\s(.*)\spo.*) - scope-587
|---sshdLog: Load(/data/sshd.log:PigStorage) - scope-584-----
Reduce Plan
grouped_complete_sshd_schema: Store(fakefile:org.apache.pig.builtin.PigStorage) - scope-599
|
|---grouped_complete_sshd_schema: Package(Packager)[tuple]{chararray} - scope-596-----
Global sort: false
-----
```

# Illustrate

```
| sshdLog    | line:chararray
|           |
| Nov 01 21:07:02 localhost.localdomain sshd[21558]: Accepted password for krr from 122.172.254.242 port 61776 ssh2
| Nov 01 21:07:02 localhost.localdomain sshd[21558]: Accepted password for krr from 122.172.254.242 port 61776 ssh2
| Nov 01 06:15:17 localhost.localdomain sshd[13193]: PAM service(sshd) ignoring max retries; 6 > 3
| Nov 01 08:47:17 localhost.localdomain sshd[13763]: Failed password for root from 122.225.97.87 port 50169 ssh2
| Nov 01 11:44:11 localhost.localdomain sshd[15484]: error: Could not load host key: /etc/ssh/ssh_host_dsa_key
| Nov 02 01:39:19 localhost.localdomain sshd[27634]: Failed password for invalid user admin from 122.225.109.115 port 40829 ssh2
| Nov 01 21:07:02 localhost.localdomain sshd[21558]: Accepted password for krr from 122.172.254.242 port 61776 ssh2
| Nov 01 21:07:02 localhost.localdomain sshd[21558]: Accepted password for krr from 122.172.254.242 port 61776 ssh2
| Nov 02 01:39:19 localhost.localdomain sshd[27634]: Failed password for invalid user admin from 122.225.109.115 port 40829 ssh2
| Nov 01 06:15:17 localhost.localdomain sshd[13193]: PAM service(sshd) ignoring max retries; 6 > 3
| Nov 01 11:44:11 localhost.localdomain sshd[15484]: error: Could not load host key: /etc/ssh/ssh_host_dsa_key
| Nov 01 08:47:17 localhost.localdomain sshd[13763]: Failed password for root from 122.225.97.87 port 50169 ssh2
| complete_sshd_schema | timestamp:chararray      | eventtype:chararray   | userid:chararray     | ipaddr:chararray
|           |
| Nov 01 21:07:02   | Accepted password   | krr                 | 122.172.254.242
| Nov 01 21:07:02   | Accepted password   | krr                 | 122.172.254.242
| Nov 01 21:07:02   | Accepted password   | krr                 | 122.172.254.242
| Nov 01 21:07:02   | Accepted password   | krr                 | 122.172.254.242
| grouped_complete_sshd_schema | group:chararray      | complete_sshd_schema:bag{:tuple(timestamp:chararray,eventtype:chararray,userid:chararray,ipaddr:chararray)}
|           |
| Nov 01 21:07:02   | {(Nov 01 21:07:02, ..., 122.172.254.242), ..., (Nov 01 21:07:02, ..., 122.172.254.242)}
```



# FILTER, GROUP and LIMIT

```
sshdLog = LOAD '/data/sshd.log' USING PigStorage AS line:chararray;
```

```
sshd_schema = FOREACH sshdLog GENERATE FLATTEN(REGEX_EXTRACT_ALL(line,  
'(.*\s\d{2}\s\d{2}:\d{2}:\d{2})\s.*(Accepted.*password.*)\sfo.*\s(\S.*\sfr.*\s(.*)\spo.*)')  
as  
(timestamp:chararray, eventtype:chararray, userid:chararray, ipaddr:chararray);
```

```
complete_sshd_schema = FILTER sshd_schema BY timestamp != ";
```

```
grouped_complete_sshd_schema = GROUP complete_sshd_schema BY timestamp;
```

```
grouped_limited_schemea = LIMIT grouped_complete_sshd_schema 5;
```

# train1000.csv

8,2011-06-11,20.11

8,2011-06-13,1.19

8,2011-06-14,15.80

8,2011-06-15,7.09

8,2011-06-16,32.51

8,2011-06-17,16.67

8,2011-06-19,31.37

11,2010-04-04,13.88

11,2010-04-05,72.78

11,2010-04-10,115.87

11,2010-04-12,10.06

11,2010-04-22,1.83

11,2010-05-01,38.95

11,2010-05-05,12.50

11,2010-05-07,13.23

11,2010-05-18,36.18

# TOBAG and TOTUPLE

```
rfm_data = LOAD '/data/train1000.csv' USING PigStorage(',') AS (custid:int, date:chararray, amount:float);  
  
custid_and_amount_bag = FOREACH rfm_data GENERATE TOBAG(custid, amount);  
  
custid_time_amount_tuple = FOREACH rfm_data GENERATE TOTUPLE(custid, date, amount);
```

<i>BAG</i>	<i>TUPLE</i>
((8),(3.28))	((8,2011-06-14,15.8))
((8),(20.11))	((8,2011-06-15,7.09))
((8),(1.19))	((8,2011-06-16,32.51))
((8),(31.37))	((8,2011-06-17,16.67))
((11),(13.88))	((8,2011-06-19,31.37))
((11),(72.78))	((11,2010-04-04,13.88))
((11),(115.87))	((11,2010-04-05,72.78))
	((11,2010-04-10,115.87))
	((11,2010-04-12,10.06))
	((11,2010-04-22,1.83))
	((11,2010-05-01,38.95))

# Load using \$0, \$1, \$2....

```
rfm_data = LOAD '/data/train1000.csv' USING PigStorage(',');
```

```
cust_id_and_amount = FOREACH rfm_data
```

```
    GENERATE $0 AS custid:int,  
           $1 AS date,  
           $2 AS amount;
```

```
grunt> describe cust_id_and_amount;
```

```
cust_id_and_amount: {custid: int,date: bytearray,amount: bytearray}
```

# SUM

```
rfm_data = LOAD '/data/train1000.csv' USING PigStorage(',') AS (custid:int, date:chararray, amount:float)
cust_id_amount = FOREACH rfm_data generate custid, amount;
custid_group = GROUP cust_id_amount BY custid;

monitory = FOREACH custid_group GENERATE group, SUM(cust_id_amount.amount);
```

# UDF - Pig User Defined Functions

```
package com.dmac.pig;
```

```
import java.io.IOException;
```

```
import org.apache.pig.EvalFunc;
```

```
import org.apache.pig.data.Tuple;
```

```
import org.apache.pig.data.TupleFactory;
```

```
public class CustomerIDToNameMapperUDF extends EvalFunc<Tuple> {
```

```
    @Override
```

```
    public Tuple exec(Tuple input) throws IOException {
```

```
        String customerName      = "";
```

```
        String typeOfCustomer   = "";
```

```
        String timeStamp         =      "";
```

```
        Integer id = (Integer)input.get(0);
```

```
        timeStamp = (String)input.get(1);
```

```
        Float amount = (Float)input.get(2);
```

```
        if (amount > 100)
```

```
            typeOfCustomer = "HIGH-END-CUSTOMER";
```

```
        else
```

```
            typeOfCustomer = "LOW-END-CUSTOMER";
```

```
        if (id.intValue() == 11)
```

```
            customerName = "Aravindh";
```

```
        else if (id.intValue() == 8)
```

```
            customerName = "Alan";
```

```
        else
```

```
            customerName = "Turing";
```

```
        Tuple output = TupleFactory.getInstance().newTuple(3);
```

```
        output.set(0, customerName);
```

```
        output.set(1, timeStamp);
```

```
        output.set(2, typeOfCustomer);
```

```
        return output;
```

# UDF

```
$ javac -classpath
```

```
/home/bdsa/hadoop/pig-0.15.0/pig-0.15.0-core-h1.jar:/home/bdsa/hadoop-common-2  
.3.0.jar CustomerIDToNameMapperUDF.java
```

```
$ jar cvf usemyudf.jar com/
```

```
grunt> REGISTER usemyudf.jar
```

```
grunt> rfm_data = LOAD '/data/dunhumby/input/train1000.csv' USING  
PigStorage(',') AS (custid:int, date:chararray, amount:float);
```

```
grunt> formatted_data = FOREACH rfm_data generate  
com.dmac.pig.CustomerIDToNameMapperUDF(custid, date, amount);  
grunt> dump formatted_data;
```

# Parallelism

```
GROUP complete_sshd_schema BY timestamp PARALLEL 40;
```

```
set default_parallel 30
```



# Apache Hive

Apache Hive is a warehouse infrastructure tool to analyze structured data.

A platform where we write in SQL scripts to do map-reduce applications.

The Hive Query Language (HQL) is used to write in queries for hive.



# Install Hive

[1] Download derby. Hive metastore needs a database.

[https://db.apache.org/derby/derby\\_downloads.html](https://db.apache.org/derby/derby_downloads.html)

[2] Edit .bashrc in the home folder to have the derby bin path included in the .bashrc file.  
export PATH="/home/dharshekthvel/anaconda2/bin:/home/dharshekthvel/ac/bin/db-derby-10.13.1.1-bin/bin:\$PATH"

[3] Rename *hive-default.xml.template* to *hive-site.xml* which is in the *hive/conf*



# Install Hive

## [4] Edit the `hive-site.xml` to have this property

```
<property>
  <name>javax.jdo.option.ConnectionURL</name>
  <value>jdbc:derby://localhost:1527/metastore_db;create=true</value>
  <description>
    JDBC connect string for a JDBC metastore.
    To use SSL to encrypt/authenticate the connection, provide database-specific SSL flag in the connection URL.
    For example, jdbc:postgresql://myhost/db?ssl=true for postgres database.
  </description>
</property>
```

## [5] Copy the jar files to the lib directory of the hive. Copy `derbyclient.jar` and `derbytools.jar` file to the lib folder of the hive.

[6] Start the Derby database. `/ac/bin/db-derby-10.13.1.1-bin/bin$ ./startNetworkServer`

[7] `hive/bin./schematool -dbType derby -initSchema`

# Hive Shell Commands

```
hive> create schema bigdatadb;          (or)
hive> create database analyticsdb;
```

```
hive> CREATE TABLE IF NOT EXISTS SRILANKA(id int, name String);
hive> CREATE TABLE IF NOT EXISTS DHARSHEKTH(name String) STORED AS TEXTFILE;
```

```
hive> show databases;
hive> show tables;
```

# Hive UDF

**UDF**      User Defined Functions

**UDAF**      User Defined Aggregate Functions

**UDTF**      User Defined Table Functions

# Hive UDF

## UDF      User Defined Functions

```
import org.apache.hadoop.hive.ql.  
exec.UDF;  
  
import org.apache.hadoop.hive.ql.exec.UDFArgumentException;  
import org.apache.hadoop.hive.ql.metadata.HiveException;  
import org.apache.hadoop.hive.ql.udf.generic.GenericUDF;  
import org.apache.hadoop.hive.serde2.objectinspector.ObjectInspector;  
  
public class StockValuePer extends GenericUDF {  
  
    @Override  
    public Object evaluate(DeferredObject[] arg0) throws HiveException {  
        // TODO Auto-generated method stub  
        return null;  
    }  
  
    @Override  
    public String getDisplayString(String[] arg0) {
```



# Hive UDTF

UDTF            User Defined Table Functions

```
import org.apache.hadoop.hive.ql.exec.UDFArgumentException;
import org.apache.hadoop.hive.ql.metadata.HiveException;
import org.apache.hadoop.hive.ql.udf.generic.GenericUDTF;
import org.apache.hadoop.hive.serde2.objectinspector.ObjectInspector;
import org.apache.hadoop.hive.serde2.objectinspector.StructObjectInspector;
```

```
public class StockPercentageCalculator extends GenericUDTF{
```

```
    @Override
```

```
    public void close() throws HiveException {
        // TODO Auto-generated method stub
```

# Hive UDAF

UDAF for hive is the User Defined Aggregate Function which is used to write in aggregations.

# Hive MapJoin

MapJoin allows a table to be loaded into the memory, so by which a join is performed only with mappers, without having to use Map/Reduce step.

If the queries frequently rely on small table joins then we could use a MapJoin.

```
select /*+ MAPJOIN(dept) */ emp.name, dept.dept from employee emp JOIN department dept ON (emp.empid=dept.empid);
```

# Hive Metastore

The Hive metastore stores the metadata of hive tables.

Used in capturing analytics of big data.

Usually configured to an external database. Typically MYSQL.

# Hive Metastore - Tables in metastore

```
mysql> use metastore  
Reading table information for completion of table and column names  
You can turn off this feature to get a quicker startup with -A
```

metastore is the schema used  
by hive

```
Database changed  
mysql> show tables;  
+-----+  
| Tables_in_metastore |  
+-----+  
| BUCKETING_COLS      |  
| CDS                 |  
| COLUMNS_V2          |  
| COMPACTION_QUEUE    |  
| COMPLETED_TXN_COMPONENTS |  
| DATABASE_PARAMS     |  
| DBS                 |  
| DB_PRIVS            |  
| DELEGATION_TOKENS   |  
| FUNCS               |  
| FUNC_RU             |  
| GLOBAL_PRIVS        |
```

# Hive Metastore - Tables in metastore

HIVE_LOCKS		SD_PARAMS	
IDXS		SEQUENCE_TABLE	
INDEX_PARAMS		SERDES	
MASTER_KEYS		SERDE_PARAMS	
NEXT_COMPACTION_QUEUE_ID		SKEWED_COL_NAMES	
NEXT_LOCK_ID		SKEWED_COL_VALUE_LOC_MAP	
NEXT_TXN_ID		SKEWED_STRING_LIST	
NUCLEUS_TABLES		SKEWED_STRING_LIST_VALUES	
PARTITIONS		SKEWED_VALUES	
PARTITION_EVENTS		SORT_COLS	
PARTITION_KEYS		TABLE_PARAMS	
PARTITION_KEY_VALS		TAB_COL_STATS	
PARTITION_PARAMS		TBLS	
PART_COL_PRIVS		TBL_COL_PRIVS	
PART_COL_STATS		TBL_PRIVS	
PART_PRIVS		TXNS	
ROLES		TXN_COMPONENTS	
ROLE_MAP		TYPES	
SDS		TYPE_FIELDS	
		VERSION	
		+-----+	

# Impala



# Impala

Impala is a massive SQL query engine for processing huge volumes of data.

Written in C++ and Java.

Not runs MR-JOB, but runs impala-daemon process.



# Impala

Impala is a product from Cloudera which acts a SQL on top of HDFS.

# Impala

## Main components

Impala daemon

Impala StateStore

Impala Metastore

## Query processing Interfaces

Impala-shell

Hue Interface

ODBC/JDBC Driver

# Impala

## Impala daemon

Impala daemon runs on each and every system.

## Impala StateStore

The StateStore is responsible for checking the health of each daemon.

## Impala Metastore

Impala uses mysql or postgresql for storing the metadata.

APACHE  
**HBASE**

# Apache HBase

Apache HBase is a columnar storage for the hadoop by Mike Cafarella.

# hbase shell

```
#list the tables
list

# Create a table
create 'hbase_data_table', 'personal_data', 'professional_data'

# Describing the table
describe 'hbase_data_table'

# Inserting Data
put 'hbase_data_table','1','personal_data:fathers_name','chinnasamy'
put 'hbase_data_table','1','professional_data:experience','2'

# Retrieving Data
scan 'hbase_data_table'
```

# hbase - create table

```
HTableDescriptor ht = new HTableDescriptor("PEOPLES");

ht.addFamily( new HColumnDescriptor("professional_data"));

ht.addFamily( new HColumnDescriptor("personal_data"));

HBaseAdmin hba = new HBaseAdmin( conf );

hba.createTable(ht);
```

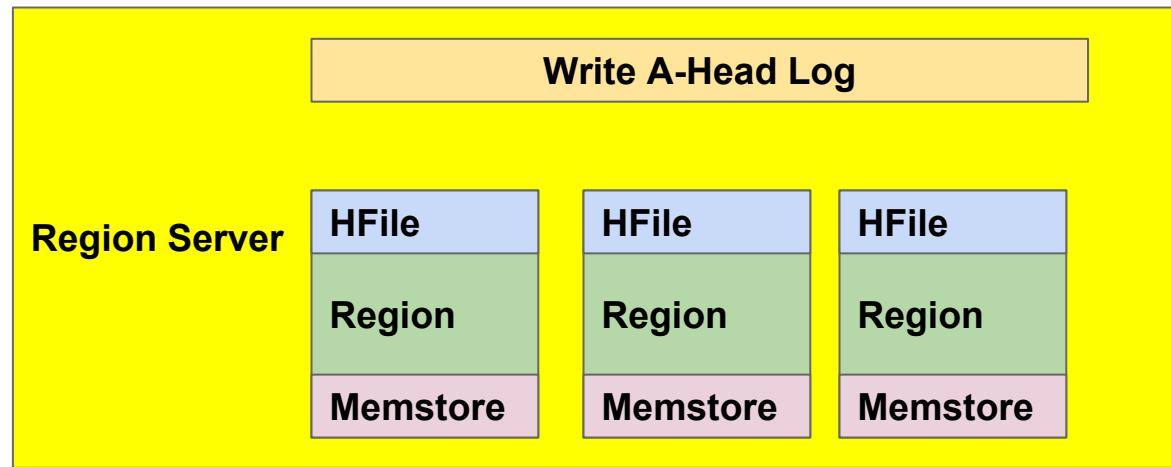
# hbase - insert data

```
HTable htable = new HTable(conf, "PEOPLES");

Put put = new Put(Bytes.toBytes("rowKey-11002233"));
put.add(Bytes.toBytes("professional_data"), Bytes.toBytes("experience"), Bytes.toBytes("2"));
put.add(Bytes.toBytes("professional_data"), Bytes.toBytes("technology"), Bytes.toBytes("JAVA"));

htable.put(put);
```

# HBASE Architecture



# hbase architecture

HBase is the counterpart of the google's Big Table. The following are the various components in HBase.

Client Program

Master Server

Region Server

Regions

Store

Memstore

HFile

WAL

Each component's functionality is described below.

Master Server: Coordinates with zookeeper. Master use the zookeeper zNodes to check on the availability of the zookeeper nodes. Also zookeeper is used for checking on the network failures on the nodes.

Region Servers: The regions are stored in the region servers. Handles all read and write and data updates. The Region server contains store and regions.

# hbase architecture

Store: The store contains Memstore and HFile.

Memstore is a cache. HBase stores data in memstore first and then flushes data to HFile as blocks which are stored in the file system.

WAL (Write A-Head Log): WAL stores the set of writes and when a server crashes it writes.

Regions: The tables are split and stored in regions. Each region is assigned to a region server.

Regions contain in-memory data store (Memstore) and a persistent data store (HFile) and all regions share a reference to WAL(Write AHead Log).

# hbase shell

```
=> ["PEOPLES", "Table_name", "countries", "hbase_data_table"]
hbase(main):007:0> scan 'PEOPLES'
ROW                                         COLUMN+CELL
  rowKey-11002233                          column=professional_data:experience, timestamp=1441810670218, value=2
  rowKey-11002233                          column=professional_data:technology, timestamp=1441810670218, value=JAVA
1 row(s) in 0.1890 seconds

hbase(main):008:0> █
```

# hbase shell

```
hbase(main):001:0> list 'TABLE_NAME'
```

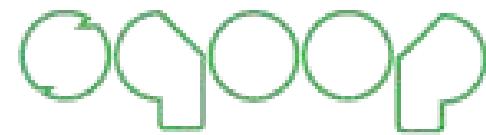
```
hbase (main):001:0> list '/template'  
TABLE  
/template/kycresponseaudit  
/template/genericotpaudittrail  
/template/dbaudittrail  
/template/updateaudittrail  
/template/authdataextractiontracker  
/template/authaudittrail  
/template/sample_emp  
/template/bfdaudittrail  
/template/residentdetail  
/template/authauditindex  
/template/authauditindexgndc  
/template/authaudittracker  
/template/otpauditindex  
/template/residentbiometricdata  
/template/authaudittrailgndc  
/template/otpaudittrail  
/template/residentdetail_nonprod  
/template/test  
/template/genericotpauditindex  
/template/mouresponseaudit  
/template/bfdauditindex  
21 row(s) in 0.3930 seconds  
  
hbase (main):002:0>
```

# hbase shell - Filters

```
hbase(main):001:0> show_filters
```

```
hbase(main):004:0> show filters
Documentation on filters mentioned b
ColumnPrefixFilter
TimestampsFilter
PageFilter
MultipleColumnPrefixFilter
FamilyFilter
ColumnPaginationFilter
SingleColumnValueFilter
RowFilter
QualifierFilter
ColumnRangeFilter
ValueFilter
PrefixFilter
SingleColumnValueExcludeFilter
ColumnCountGetFilter
InclusiveStopFilter
DependentColumnFilter
FirstKeyOnlyFilter
KeyOnlyFilter

hbase(main):005:0>
```



# Apache Sqoop

Sqoop is a command line tool used to transfer data from relational database to hadoop.

# Import Command

```
$ sqoop  
import  
--connect jdbc:mysql://localhost/nav --username root --password  
cloudera --table movies  
--m 1  
--target-dir /movie1
```

# Apache Flume

# Apache Flume - Config

```
# Naming the components on the current agent.
```

```
TwitterAgent.sources = Twitter
```

```
TwitterAgent.channels = MemChannel
```

```
TwitterAgent.sinks = HDFS
```

```
# Describing/Configuring the source
```

```
TwitterAgent.sources.Twitter.type = org.apache.flume.source.twitter.TwitterSource
```

```
TwitterAgent.sources.Twitter.consumerKey = H7gCcVkw7HI8C4hbgEo62h6zF
```

```
TwitterAgent.sources.Twitter.consumerSecret = yGgJ8iT5FKoNcrMWCE31tmQqtKzPiSnhiX9XuXIFP4P54XVdxg
```

```
TwitterAgent.sources.Twitter.accessToken = 837928192170209280-v0p7XGPdehsV0VSTouN6YisQReaEQGt
```

```
TwitterAgent.sources.Twitter.accessTokenSecret = 6HnqRgWUyQWPSUfKKpthSP7jXUBFfU7hVop6vlfJxUQpL
```

```
TwitterAgent.sources.Twitter.keywords = jallikattu, neduvusal
```

```
# Describing/Configuring the sink
```

```
TwitterAgent.sinks.HDFS.type = hdfs
```

```
TwitterAgent.sinks.HDFS.hdfs.path = hdfs://localhost:9000/user/Hadoop/twitter_data/
```

```
TwitterAgent.sinks.HDFS.hdfs.fileType = DataStream
```

```
TwitterAgent.sinks.HDFS.hdfs.writeFormat = Text
```

```
TwitterAgent.sinks.HDFS.hdfs.batchSize = 1000
```

```
TwitterAgent.sinks.HDFS.hdfs.rollSize = 0
```

```
TwitterAgent.sinks.HDFS.hdfs.rollCount = 10000
```

```
# Describing/Configuring the channel TwitterAgent.channels.MemChannel.type = memory
```

```
TwitterAgent.channels.MemChannel.capacity = 10000
```

```
TwitterAgent.channels.MemChannel.transactionCapacity = 100
```

```
# Binding the source and sink to the channel
```

```
TwitterAgent.sources.Twitter.channels = MemChannel
```

```
TwitterAgent.sinks.HDFS.channel = MemChannel
```

chinnasamyad@gmail.com

# Apache Flume

In the bin folder of flume\_installation, give the below command

```
$ ./flume-ng agent --conf ../conf/ -f ../conf/twitter.conf Dflume.root.logger=DEBUG,console -n TwitterAgent
```



# MapR Architecture

## MapR-FS and HDFS

MapR-FS is an implementation of HDFS.

HDFS API's are compliant with MapR-FS.

MapR-FS, written in C, is a POSIX compliant read-write file system

MapR-FS : No Namenode. The metadata is distributed across the cluster.  
So SPOF is eliminated.

# MapR Architecture

Data is written to containers. Container are managed by CLDB.  
CLDB - Container Location Database.

MapR-FS Client talks to CLDB to assign in container to write in data.

MAPR writes data to Container through chunks. Each chunk is of 256 MB writing at 8 KB per IO write. 256 MB is the default block size when compared to HDFS block size of 64 MB.

Unit of sharding	: Chunk - 256 MB
Unit of Replication	: Container - 32 GB
Unit of disk I/O	: Block - 8 KB

# MapR Architecture

Storage Pools - A Storage pool consists of many disks grouped together.  
Three disks are pooled together by default.

Containers reside on storage pool. Each storage pool can store many containers. The default container size is 32 G. Container is replicated three times by default.

cldb.container.sizemb

Each container contain directories, tables, files, data blocks and metadata.

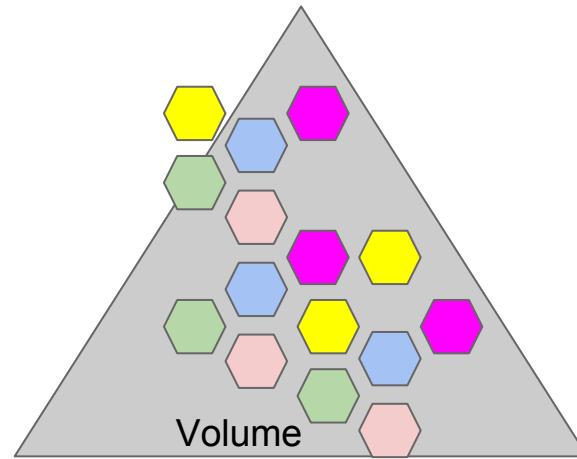
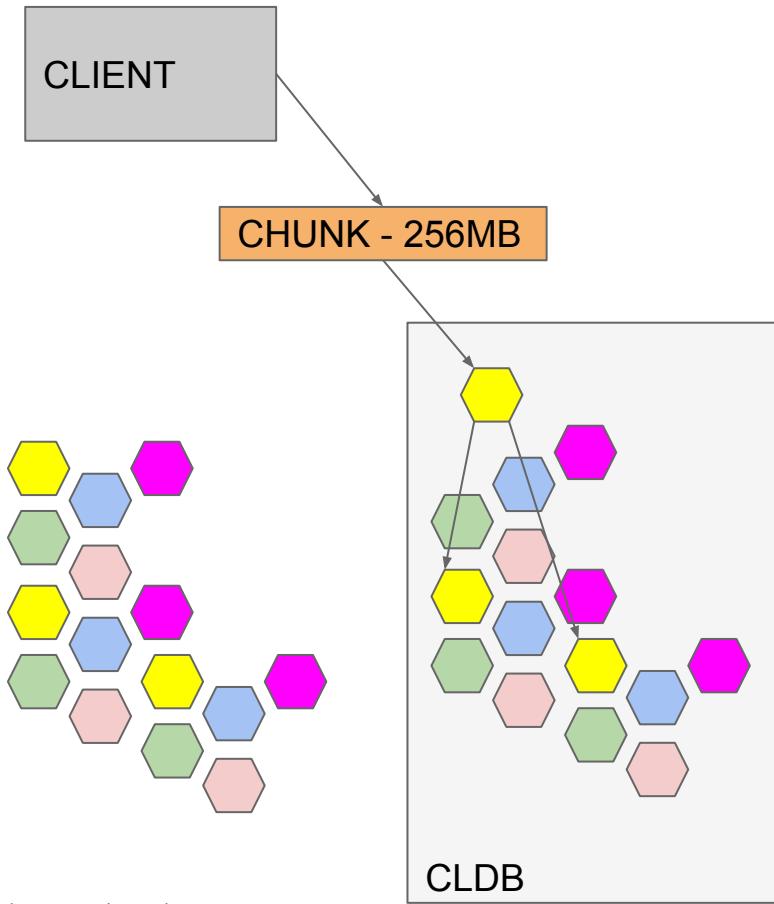
A volume is a group of containers.

# MapR Architecture

CLDB is Container Location Data Base is a management service which finds out containers and root of volume.

CLDB only locates containers, not files and directories.

# MapR Architecture



# MapR Architecture

```
idapp@MNDCTEDC2B10 conf]$ cat warden.conf
rvice=webserver:all:cldb:jobtracker:1:cldb:t_sktracker:all:jobtracker;nfs:all:cldb;kvstore:all:cldb:all:kvsto
rvice.command.jt.start=/opt/mapr/hadoop/hadoop-0.20.2/bin/hadoop-daemon.sh start jobtracker
rvice.command.tt.start=/opt/mapr/hadoop/hadoop-0.20.2/bin/hadoop-daemon.sh start tasktracker
rvice.command.hbmaster.start=/opt/mapr/hbase/hbase-0.94.17/bin/hbase-daemon.sh start master
rvice.command.hbregion.start=/opt/mapr/hbase/hbase-0.94.17/bin/hbase-daemon.sh start regionserver
rvice.command.cldb.start=/etc/init.d/mapr-cldb start
rvice.command.kvstore.start=/etc/init.d/mapr-mfs start
rvice.command.mfs.start=/etc/init.d/mapr-mfs start
rvice.command.nfs.start=/etc/init.d/mapr-nfsserver start
rvice.command.hoststats.start=/etc/init.d/mapr-hoststats start
rvice.command.webserver.start=/opt/mapr/adminuiapp/webserver start
rvice.command.jt.stop=/opt/mapr/hadoop/hadoop-0.20.2/bin/hadoop-daemon.sh stop jobtracker
rvice.command.tt.stop=/opt/mapr/hadoop/hadoop-0.20.2/bin/hadoop-daemon.sh stop tasktracker
rvice.command.hbmaster.stop=/opt/mapr/hbase/hbase-0.94.17/bin/hbase-daemon.sh stop master
rvice.command.hbregion.stop=/opt/mapr/hbase/hbase-0.94.17/bin/hbase-daemon.sh stop regionserver
rvice.command.cldb.stop=/etc/init.d/mapr-cldb stop
rvice.command.kvstore.stop=/etc/init.d/mapr-mfs stop
rvice.command.mfs.stop=/etc/init.d/mapr-mfs stop
rvice.command.nfs.stop=/etc/init.d/mapr-nfsserver stop
rvice.command.hoststats.stop=/etc/init.d/mapr-hoststats stop
rvice.command.webserver.stop=/opt/mapr/adminuiapp/webserver stop
rvice.command.jt.type=BACKGROUND
rvice.command.tt.type=BACKGROUND
rvice.command.hbmaster.type=BACKGROUND
rvice.command.hbregion.type=BACKGROUND
rvice.command.cldb.type=BACKGROUND
rvice.command.kvstore.type=BACKGROUND
rvice.command.mfs.type=BACKGROUND
rvice.command.nfs.type=BACKGROUND
rvice.command.hoststats.type=BACKGROUND
rvice.command.webserver.type=BACKGROUND
rvice.command.jt.monitor=org.apache.hadoop.mapred.JobTracker
rvice.command.tt.monitor=org.apache.hadoop.mapred.TaskTracker
rvice.command.hbmaster.monitor=org.apache.hadoop.hbase.master.HMaster start
rvice.command.hbregion.monitor=org.apache.hadoop.hbase.regionserver.HRegionServer start
rvice.command.cldb.monitor=com.mapr.fs.cldb.CLDB
rvice.command.kvstore.monitor=server/mfs
rvice.command.mfs.monitor=server/mfs
rvice.command.nfs.monitor=server/nfsserver
rvice.command.jt.monitorcommand=/opt/mapr/hadoop/hadoop-0.20.2/bin/hadoop-daemon.sh status jobtracker
```

# MapR Architecture

Files =>

Chunks =>

Containers =>

Storage Pools =>

Disks => Nodes on the Disk

A container exactly belongs to one volume.

## MapR - maprcli

maprcli is the MapR Command Line Interface for MapR Distribution of Hadoop

**Lists the recent tables**

```
$ maprcli table listrecent
```

## MapR - maprcli

maprcli is the MapR Command Line Interface for MapR Distribution of Hadoop

**Lists the recent tables**

```
$ maprcli table listrecent
```

# Container Location Database - CLDB UI

CLDB UI

<http://10.66.204.170:7221/>

CLDB UI runs on 7221 port by default.

The screenshot shows the CLDB UI interface. At the top, it displays the URL as 10.66.204.170:7221/cldb.jsp. The main title is "Container Location Database". Below it, the CLDB mode is listed as "MASTER\_READ\_WRITE". It also shows the CLDB BuildVersion as 3.1.0.23703.GA and the Master for CLDB volume ready as true. The CLDB Status is shown as "RUNNING" with details: Cluster Capacity at 974.65 GB, Cluster Used at 1.85 GB, Cluster Available at 972.8 GB, and Cluster Used Percentage at 0. The "Active FileServers" section contains a table with one row, showing the following data:

ServerID (Hex)	ServerId	HostPort	HostName	Network Location	Last Heartbeat (s)	State	Capacity (MB)	Used (MB)	Available (MB)	In Transit (MB)
13364b4c39c57cf0	1384376726221847805	10.66.204.170-10.67.204.170-192.168.122.1-	MNDCTEDC2B10	data/default-rack/MNDCTEDC2B10	0	ACTIVE	974.65 GB	1.85 GB	972.8 GB	0

Below this, there is a section titled "Active NFS Servers" which is currently empty.



# RabbitMQ Installation

The screenshot shows the Erlang download page at [www.erlang.org/download.html](http://www.erlang.org/download.html). The top navigation bar includes links for NEWS, ARTICLES, EVENTS, DOWNLOADS, COMMUNITY, LINKS, DOCUMENTATION, and ABOUT. The DOWNLOADS section is highlighted with an orange box. Below it, the DOWNLOAD OTP 18.1 section is also highlighted with an orange box. The page lists several OTP 18.1 files, with the 64-bit Windows binary file being the one highlighted by a blue arrow pointing towards the RabbitMQ installation section.

## DOWNLOAD OTP 18.1

Erlang/OTP 18.1 has been released

[OTP 18.1 Readme File](#)

[OTP 18.1 Source File \(65.0 MB\)](#)

[OTP 18.1 Windows 32-bit Binary File \(93.6 MB\)](#)

[OTP 18.1 Windows 64-bit Binary File \(94.1 MB\)](#)

[OTP 18.1 HTML Documentation File \(32.4 MB\)](#)

[OTP 18.1 Man Pages File \(1.3 MB\)](#)

Some highlights of the release are:

- ssl: Add possibility to downgrade an SSL/TLS connection to a tcp connection, and give back the socket control to a user process.
- ssh: The following new key exchange algorithms are implemented: 'ecdh-sha2-nistp256', 'ecdh-sha2-nistp384', 'ecdh-sha2-nistp521', 'diffie-hellman-group14-sha1', 'diffie-hellman-group-exchange-sha1' and 'diffie-hellman-group-exchange-sha256'. This raises the security level considerably.

kernel,stdlib,sasl: A mechanism for limiting the amount of text that the built-in error logger events will produce has been introduced. It is useful for limiting both the size of log files and the CPU time used to produce them.

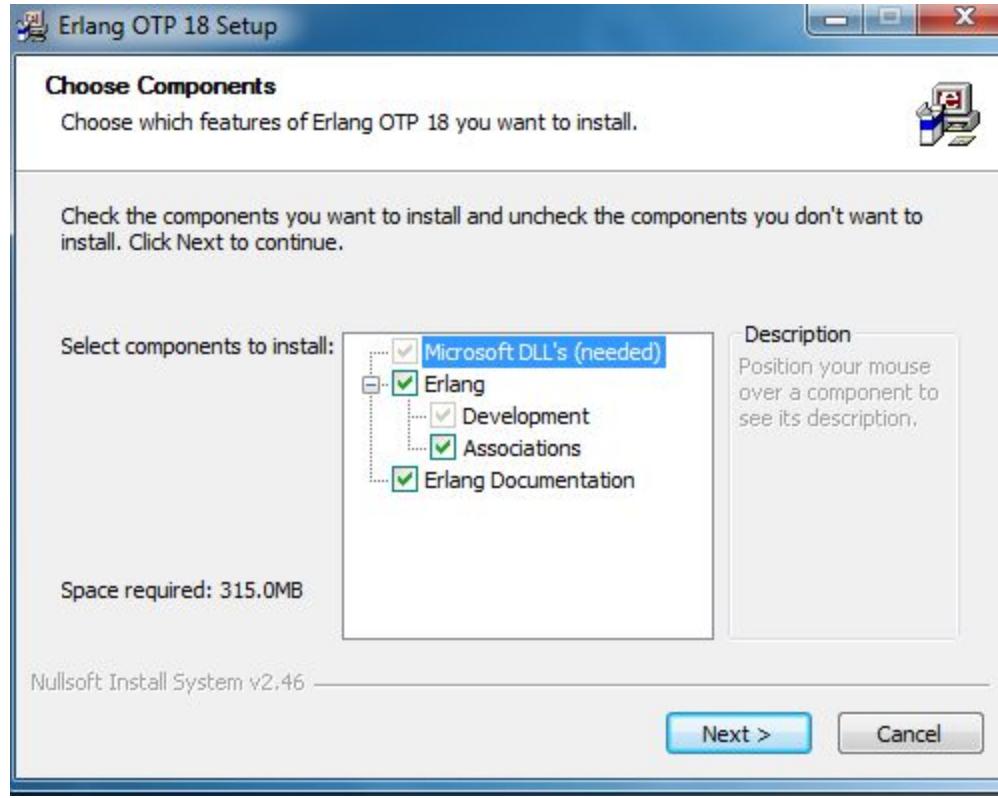
This mechanism is experimental in the sense that it may be changed based on feedback. See config parameter `error_logger_format_depth` in the Kernel application.

RabbitMQ First needs erlang to be installed.  
So install erlang first.

### Available releases

- OTP 18.1
- OTP 18.0
- OTP 17.5
- OTP 17.4
- OTP 17.3
- OTP 17.1
- OTP 17.0
- OTP R16B03-1
- R16B03
- R16B02
- R16B01
- R16B
- R16A
- R15B03-1
- R15B02
- R15B01
- R15B
- R14B04
- R14B03
- R14B02
- R14B01

# RabbitMQ Installation



# RabbitMQ Installation

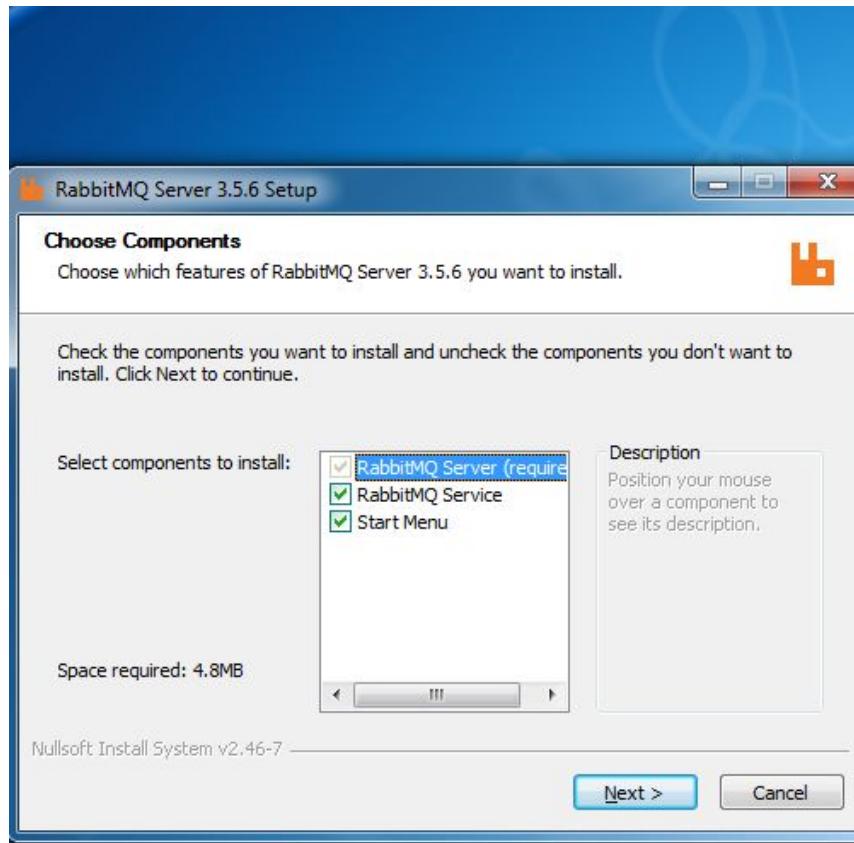
https://www.rabbitmq.com/download.html

The screenshot shows the official RabbitMQ download page. At the top is a navigation bar with links for Features, Installation, Docs, Tutorials, Support (which is underlined in orange), Community, We're Hiring, and Blog. To the right is a search bar with a magnifying glass icon. A large blue watermark reading "Now download RabbitMQ and install it" is diagonally across the page. The main content area has a light orange background and contains sections for "RabbitMQ Server" (with download links for Windows, Debian/Ubuntu, Fedora/RHEL, and source), "Downloads on GitHub" (with similar links), "Installation Guides" (with links for Windows, Linux/Unix, and Mac OS X), "Cloud" (with links for Amazon EC2, Cloud Foundry, and Pivotal Cloud Foundry), and a footer link to "rabbitmq.com/services.html". On the right side, there's a sidebar titled "In This Section" containing a list of installation links for various platforms like Windows, Debian, Ubuntu, RPM-based Linux, Mac OS X, Homebrew, and more.

**In This Section**

- › Install: Windows
- › Install: Debian / Ubuntu
- › Install: RPM-based Linux
- › Install: Mac OS X
- › Install: Homebrew
- › Install: Windows (manual)
- › Install: Generic Unix
- › Install: Solaris
- › Install: EC2
- › Supported Platforms
- › Changelog
- › Erlang Versions
- › Signed Packages
- › Java Client Downloads
- › .NET Client Downloads
- › Erlang Client Downloads
- › Community Plugins

# RabbitMQ Installation



# RabbitMQ Commands

Returns the status of the server

\rabbitmq\_server-3.5.6\sbin>*rabbitmqctl.bat status*

# RabbitMQ Commands

```
Status of node 'rabbit@2400002720-LT33' ...
[{pid,5016},
 {running_applications,[{rabbit,"RabbitMQ","3.5.6"},
    {mnesia,"MNESIA CXC 138 12","4.13.1"},
    {os_mon,"CPO CXC 138 46","2.4"},
    {xmerl,"XML parser","1.3.8"},
    {sasl,"SASL CXC 138 11","2.6"},
    {stdlib,"ERTS CXC 138 10","2.6"},
    {kernel,"ERTS CXC 138 10","4.1"}]},
 {os,{win32,nt}},
 {erlang_version,"Erlang/OTP 18 [erts-7.1] [64-bit] [smp:4:4] [async-threads:30]
 \n"},
 {memory,[{total,37069616},
    {connection_readers,0},
    {connection_writers,0},
    {connection_channels,0},
    {connection_other,2712},
    {queue_procs,2712},
    {queue_slave_procs,0},
    {plugins,0},
    {other_proc,13575040},
...
...
...
```

# RabbitMQ Installation



# Agenda

Era of Functional Programming

Functional Interfaces

Lambda Expressions

Stream API

# Functional Programming

The era of functional programming

Functional programming is a programming paradigm (a style of coding)

JDK 8 effectuates functional programming with the help of functional interfaces.

Usage of lambda expressions in the functional programming style makes the code more elegant and supreme.

# Functional Interfaces

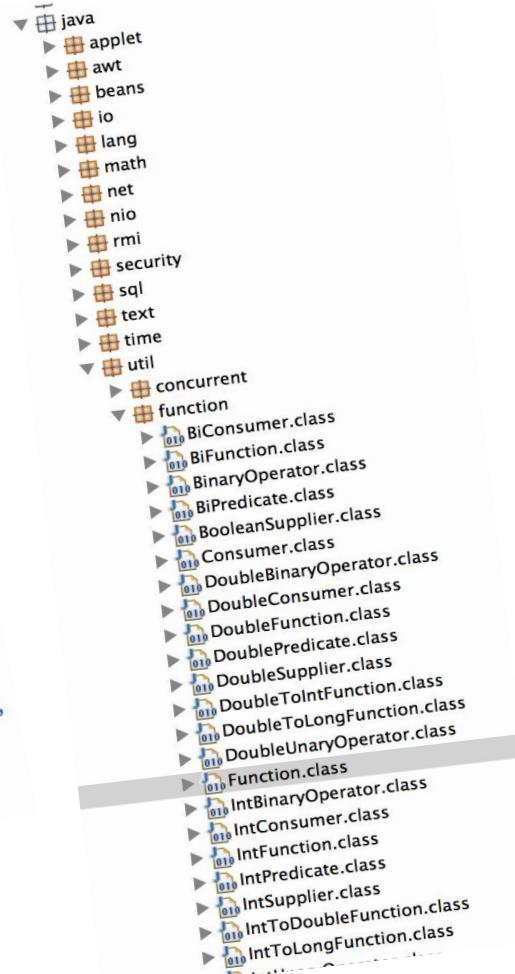
Functional Interfaces also called as Single Abstract Method (SAM) is an interface that contains only one abstract method.

Functional Interfaces may contain one or more default methods or static methods.

A marker interface is an empty interface whereas a functional interface is an interface with one method.

# Functional Interfaces

```
38 * @since 1.8
39 */
40 @FunctionalInterface
41 public interface Function<T, R> {
42     /**
43      * Applies this function to the given argument.
44      *
45      * @param t the function argument
46      * @return the function result
47      */
48     R apply(T t);
49
50     /**
51      * Returns a composed function that first
52      * applies this function to the given argument,
53      * and then applies the given consumer to the
54      * result.
55      *
56      * Note that this function is not guaranteed to
57      * evaluate its argument as part of applying the
58      * composed function; this Java™ Language
59      * Specification does not make any guarantees
60      * as to whether an implementation does this or
61      * not.
62      *
63      * @param consumer the consumer to apply to the
64      * result of this function
65      *
66      * @return a composed function that performs
67      * both operations
68      */
69     Function<T, R&gt; andThen(Consumer<R> consumer);
70
71     /**
72      * Returns a composed function that first
73      * applies this function to the given argument,
74      * and then applies the given function to the
75      * result.
76      *
77      * Note that this function is not guaranteed to
78      * evaluate its argument as part of applying the
79      * composed function; this Java™ Language
80      * Specification does not make any guarantees
81      * as to whether an implementation does this or
82      * not.
83      *
84      * @param function the function to apply to the
85      * result of this function
86      *
87      * @return a composed function that performs
88      * both operations
89      */
90     Function<T, R&gt; compose(Function<R, S> function);
91
92     /**
93      * Returns a composed function that first
94      * applies this function to the given argument,
95      * and then applies the given predicate to the
96      * result.
97      *
98      * Note that this function is not guaranteed to
99      * evaluate its argument as part of applying the
100     * composed function; this Java™ Language
101     * Specification does not make any guarantees
102     * as to whether an implementation does this or
103     * not.
104     *
105     * @param predicate the predicate to apply to the
106     * result of this function
107     *
108     * @return a composed function that performs
109     * both operations
110     */
111     Function<T, R&gt; andThen(Predicate<R> predicate);
112
113     /**
114      * Returns a composed function that first
115      * applies this function to the given argument,
116      * and then applies the given supplier to the
117      * result.
118      *
119      * Note that this function is not guaranteed to
120      * evaluate its argument as part of applying the
121      * composed function; this Java™ Language
122      * Specification does not make any guarantees
123      * as to whether an implementation does this or
124      * not.
125      *
126      * @param supplier the supplier to apply to the
127      * result of this function
128      *
129      * @return a composed function that performs
130      * both operations
131      */
132     Function<T, R&gt; compose(Supplier<S> supplier);
133 }
```



# Simple Functional Program

```
public class IsItAValidYamlDocument implements Predicate<String> {  
    @Override  
    public boolean test(String yamlDocument) {  
        return true;  
    }  
}  
  
import java.util.function.Function;  
public class YamlReaderFunction implements Function<String, Yaml> {  
    @Override  
    public Yaml apply(String yamlDocument) {  
  
        Yaml yaml = new Yaml();  
        // Do complex operations on the Yaml and then return Yaml object.  
        return yaml;  
    }  
}
```

# The Lambda Expression

A lambda expression is a function without a declaration which is passed as a block of code to a place where an object is expected and the expected object is of type functional interface.

A lambda expression is an anonymous function that is treated as an instance of a functional interface.

A definitive epitome programming style that dates back to lisp style of programming.

# The Lambda Expression

A lambda expression is expressed by the below mentioned syntax

( x , y, z, , , ...)      ->      <program\_logic>

# Simplest Lambda Expression Program

```
List<String> listOfStrings = new ArrayList<String>();  
listOfStrings.add("Dhara");  
listOfStrings.add("Bose");  
listOfStrings.add("Che");  
  
listOfStrings.forEach((z) -> System.out.println(z));
```

# Lambda Expression Program

```
List<String> countries = new ArrayList();
countries.add("India");
countries.add("Pakistan");
countries.add("Srilanka");
```

```
List<String> filteredListOfStrings = countries
    .parallelStream()
    .filter((z) -> z.startsWith("S"))
    .collect(Collectors.toList());
```

# Lambda Expression for a Thread

```
Runnable runnable = () -> {  
    for (;;) {  
        System.out.println("I am inside the thread");  
    }  
};  
  
new Thread(runnable).start();
```

# Stream API

A stream is obtained as a source from the collections API components.

A stream is composed of one or more intermediate operation and one terminal operation.

Stream is obtained from a source and then one or many intermediate operation is performed and then followed by a terminal operation.

# Stream API

```
import java.util.Arrays;  
  
Path[] paths = FileUtil.stat2Paths(files);  
  
Arrays.stream(paths)  
    .forEach(path -> System.out.println(path.getName()));
```

# How to get in the total number of processors in Java ?

The `availableProcessors()` method in the `Runtime` class gives in the number of processors available to the JVM.

It is accessed by `Runtime.getRuntime().availableProcessors()`.

Any distributed java based system would use in this property to leverage the multi-core capability.

# How to get in the total number of processors ?

## In Scala:

```
val cpus = Runtime.getRuntime.availableProcessors()  
println(cpus)
```

## In linux:

```
$ lscpu  
Or  
$ sysctl --all | grep cpu
```

## In mac

```
$ sysctl -n hw.cpu  
$ sysctl -a | grep cpu
```



# Python

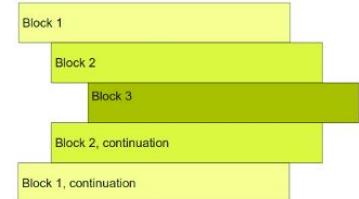


# Python

python is an interpreted,  
object-oriented and high level language.

# Python

- A language structured with indentation, meaning code blocks are defined by their indentation.
- In python, indentation is matter of **language requirement** not a matter of style as compared to other programming languages.



# Simple things first

# You can use a single quote or double quote

```
print ('Hello Chola')
```

```
print ("Hello Chola")
```

# Simple things first

# *Single line comment*

# """ *Three quotes for multi line comment*

""

**Multiline**

**comment**  
    """

# Datatypes

Numeric

Float

String

List

Tuple

Dictionary

Refer [pythondatatypes.py](#)

# Datatypes

Number

String

List

Tuple

Dictionary

# Strings

```
# String can be assigned with single or double quotes
string_variable = 'Aravindh'
another_string_varibale = "Chinnasamy"
```

```
# String concatenation
my_name = string_variable + another_string_varibale

print(my_name)
```

Refer [pythondatatypes.py](#)

# Numbers

```
number_variable = 2016
```

```
" Arithmetic Operations " # a comment
addition = 2 + 2
subtraction = 5 - 3
multiplication = 5 * 6
division = 4 / 2
remainder = 5 % 3
square = 2 ** 3
quotient = 5 // 3
```

```
# Formatting a print statement
print("Addition = %s" % "hi")
print("Addition = %s" % addition)
```

Refer pythondatatypes.py

# List

List is created using [ ]

```
list_of_contrats = ['echencier_1','echencier_2','echencier_3']
print list_of_contrats
```

# Tuple

Tuple is created using ( )

```
list_of_time_line = ('timeline_1','time_line_2','time_line_3')
print list_of_time_line
```

Refer pythondatatypes.py

# Dictionary - Associative arrays

Dictionary is created using { key : value }

```
country_code = {'IN':'INDIA', 'SL':'SRILANKA', 'CM':'CAMEROON'}
```

```
print country_code
```

Refer [pythondatatypes.py](#)

# Sets

Sets are mutable

Sets are created using list

```
set_of_integers = set([1,2,4,4,2,5,6,2,1])  
print ("Printing a set %s" % set_of_integers)
```

# frozensets

frozensets are immutable

```
set_of_frozen_list = frozenset([1,2,4,3,2,3,2,1,7,9,7,8])  
print ("Printing a frozen set %s" % set_of_frozen_list)
```

# Multiple Declaration is allowed

```
# Multiple Assignments in single line
_auaCode_, _auaName_, _auaValue = 1001, "TN", "LICENSED"

print _auaCode_
print _auaName_
print _auaValue
```

# Conditions

```
age = 30
```

```
if (age > 19):  
    print("You are a teenager")  
else:  
    print("You are not a teenager")
```

# Pass variable

The pass statement is a null operation

# Functions - Simple Function

```
def simple_function():
    print("Inside the simple function")
    return
```

# Exceptions

```
try:  
    f = open("a.txt")  
  
except IOError as e:  
    print('File is not found. Exception is %s' % e)  
  
else:  
    pass  
  
finally:  
    pass
```

# Classes

```
class Name_Of_Class:  
  
    def __init__(self):  
  
        def method_name:  
  
            def another_method(self, parameter):
```

To instantiate a class:

```
variable_name = Name_Of_Class()  
variable_name.method_name  
variable_name.another_method('Chinnasamy')
```

# Classes

```
class ReadCSVFile:  
  
    def __init__(self):  
        print("Constructor Initialized")  
  
    def printFileName(self, fileName):  
        print("Inside printFileName - " + fileName)
```

ReadCSVFile.py

```
readCSVFile = ReadCSVFile()  
readCSVFile.printFileName('FILE-001')
```

# Classes

```
class ReadFile:  
  
    def __init__(self, _name_of_file_):  
        self.name_of_file = _name_of_file_  
  
    def readFile(self):  
        resumeTikaFile = open(self.name_of_file, "r")  
        return resumeTikaFile.read(99999)
```

# Regular Expressions

The library `re` is regular-expression based library which can perform regular expression based solutions.

```
import re

matches = re.findall(r'[\w\.-]+@[\\w\.-]+\.', self.filecontent)
```

The above code would give the email id's in the content.

# Functions

```
def simple_function():

    print("Inside the simple function")
    return 0

def function_with_parameter(name) :

    print ("Name is %s" % name)
    return

def function_with_default_parameter(id, algorithm="gini"):

    print ("ID and name is %s and %s" % (id, algorithm))
    return
```

# Functions

```
def function_with_var_args(*arguments):
    #print(arguments)

    for each in arguments:
        print each

    return

print(simple_function())
function_with_parameter("Gini Impurity")
function_with_default_parameter(101)
function_with_default_parameter(102,"logarithmic")
function_with_var_args("arg1","arg2","arg3")
```

# File Handling

# File

## Open and read a file

**Open the file in read mode. "r" represents the read mode.**

```
resumeTikaFile = open(self.name_of_file, "r")
```

Then read the file using open(). The 99999 represents the number of bytes to be read.

```
resumeTikaFile.read(99999)
```

# Serialization using pickle

```
import pickle

from dtopackage.aua_code import AuaCode

county_tuple = ("IN", "FR", "SL")

output = open("/home/dharshekthvel/ac/county.pkl", "wb")

pickle.dump(county_tuple, output)

output.close()
```

*Pickling is the process of serializing an object in python*

# Serialization using pickle

```
import pickle

inputFile = open("/home/dharshekthvel/ac/county.pkl", "r")

tuple1 = pickle.load(inputFile)

print tuple1

inputFile.close()
```

# egg

```
from setuptools import setup, find_packages

setup (
    name='dmac',
    version='1.0.0',
    author='chinnasamy',
    packages = find_packages()

)
```

```
$ python setup.py bdist_egg
```

# REST

```
import requests

data = ""
{
    "name": "BNP",
    "location": {
        "lat": %d,
        "lon": 5
    }
}
"""

range_of_counter = range(800, 1000)

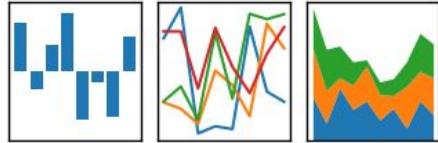
for i in range_of_counter:
    url = 'http://localhost:9200/bgeo/banks/%d?pretty' % i

    for j in range(10, 36):

        response = requests.post(url, data=data)
        print("The response is - %s" % response.text)
```

# pandas

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



API for reading, writing and manipulating data with the notion of dataframes.

# Reading an excel file

```
import pandas as pd

# Read the file
master_file = pd.ExcelFile('/home/dharshekthvel/ac/code/scalatrainingintellij/data/sample.xls')

# Get the sheet names
sheets = master_file.sheet_names

# Get a particular sheet
survey = master_file.parse('choices')

# Print the rows from 0 to 13
print survey[0:13]

# Delete a particular column
del survey['name']

# Drop a particular column
new_survey = survey2.drop('caption', axis = 1)
```

# Reading an excel file

```
# Make a deep copy of the dataframe  
  
new_survey = survey.copy(deep=True)
```

# Flask

Python web server

# Flask Base Program

```
from flask import Flask
app = Flask(__name__)

@app.route("/")
def rootrequest():
    return "Data-Ingestion Home Page"

@app.route("/data")
def data():
    return "Data-Ingestion Started"
```

```
$ export
FLASK_APP=/home/dharshekthvel/PycharmProcts/SparkBasic/
$ flask run
```



# Chinnasamy

Consultant - Big Data Architect

13+ Years of expertise in Java/J2EE, Scala, Python and Big Data

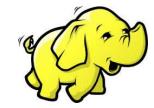
- ~ Wipro
- ~ General Electric
- ~ Verizon
- ~ Infosoft Join
- ~ Jipree
- ~ Vunya
- ~ Evolv
- ~ Scholastic
- ~ iPRO
- ~ Equatortek
- ~ Relevance Lab
- ~ Mobax
- ~ HCL - AADHAAR Project

As a consultant consulted for more than 15+ companies.

Conducted more than 40+ training programs on Big Data and allied technologies.

[chinnasamyad@gmail.com](mailto:chinnasamyad@gmail.com)

+91 96000 45955.



# Agenda

What and Why Scala

Hello world Scala

Class and Object

Conditionals and Expressions - if...else, for, map

Language Specifics and Syntax

traits

Enumerations

Value Classes

Case Classes

Functions and Function Literals

Transformations, Filtering, Informational methods

File Operations

XML Manipulations

Concurrency

SBT

Best Practices

# The Scala Language

Scala is a programming language which fuses object orientation and functional programming in a statically typed model.

# Scala

**Paradigm** - A paradigm describes distinct concepts or thought patterns in a scientific discipline process. Scala way of programming is a paradigm.

## ***Imperative Programming***

- modifying mutable variable.
- using assignments.
- One tends to conceptualize data structure word by word.
- The style changes the program's state.

eg: Java

# Scala

**Paradigm** - A paradigm describes distinct concepts or thought patterns in a scientific discipline process. Scala way of programming is a paradigm.

## ***Functional Programming***

- A programming style that models computation as a evaluation of expression
- avoid mutations
- express operators as functions
- powerful ways to abstract and compose functions. Programming without mutable variables, without assignments, without loops and without control structures. eg: Scala

Functional programming languages: Lisp, Scala, Haskell, Smalltalk, Ruby, Ocaml etc.., First functional language was Lisp.

# Expression vs Statement

## Expression

```
val tellMe = if (100 > 98) true else false
```

## Statement

```
boolean tellMe = false;
```

```
if (100 > 98)
    tellMe = true;
else
    tellMe = false;
```

```
System.out.println(tellMe);
```

# The Scala Language

Scala gets its name from **Scalable Language**

Scala addresses the major needs of java developers  
as it runs on a JVM

Scala has a light weight, concise syntax code.

# The Scala Language

As the saying goes,

Java is slower, but JVM is faster.

That's the reason, the JVM based languages are becoming popular.

# The Scala Language - Advantages

- [\*] Scala is a pure object oriented language. No primitive types.
- [\*] When compared to java, scala is less verbose.
- [\*] Very concise and expressive. Of course less verbose.
- [\*] In the long run, increased productivity.
- [\*] Philosophically, humanity tends to evolve.

# Scala

Scala uses substitution model. The substitution model is formalized in lambda calculus.

It is a good functional programming practice to split up a task into many small function.

*\$ sbt console. Will also start a scala REPL.*

# The Scala Language

Designed by Martin Odersky

Started at around 2001 in switzerland and now a full fledged language running on JVM.

With the Apache Spark using scala, it has become popular in the big data world.



# Scala Language Specifics

Scala does not require semicolons at the end of statements. They are allowed but optional. When you write in two or more statements in the same line, then you need to put in the semicolon.

Scala has Unit, while java has void.

Scala uses the *id: type* syntax whereas java uses *type id*.

# Scala Language Specifics

Embraces both object-oriented as well as functional paradigm.

Scala allows java and code to be mixed since they both run on JVM.

# Hello World Scala Program

HelloWorld.scala

```
package com.dmac

object HelloWorld {

    def main(args: Array[String]) {
        println("The world is good")
    }
}
```

An object definition defines a class with single instance.

Refer [HelloWorld.scala](#)

# Scala Language Specifics - How to write in a method

Scala uses the *id: type* syntax whereas java uses *type id*.

```
/**  
 * A Simple Scala Method  
 */  
def scalaMethod(empName:String, empDepartment : String) : Int = {  
    return 1001  
}  
  
/* A Simple Java Method */
```

```
public Integer scalaMethod(String empName, String empDepartment) {  
    Integer i = new Integer(100);  
    return i;  
}
```

Refer DefiningAMethod.scala

chinnasamyad@gmail.com

# Extend the App class to make your class executable

```
object AppMain extends App {  
    println("The whole block of code is executed  
    when the Object class is extended from the  
    App")  
}
```

Refer AppMain.scala

# var, val and def

def	defines a method
val	defines a fixed value which cannot be changed. Meaning, the variable is immutable.
var	defines a variable that can be modified. Meaning, the variable is mutable.

# Data Types - Variable Definition

```
val simpleNumeric      = 1982
```

```
val simpleNumericaAlias = 1982 : Int
```

```
val simpleFloat       = 32f
```

```
val simpleFloatAlias = 32f : Float
```

```
val simpleDouble      = 33d
```

```
val simpleDoubleAlias = 33d : Double
```

```
val simpleLong        = 48787L
```

```
val simpleLongAlias  = 48787L : Long
```

# Data Types - Variable Definition

All the four are legal and same

**val** *contrats* = 100

**val** *contrats\_1* = 200 : Int

**val** *contracts\_2* : Int = 300

**val** *contrats\_3* = { 100 }

# Datatypes - Variable Definition

**Both are legal definition**

```
val myDataRefNumber = 10034560
```

```
val myDataRefNumberAliter = { 10023456 }
```

# Datatypes

```
var addition = simpleNumeric + simpleNumeric
```

```
addition = simpleNumeric + 1
```

```
addition += 1
```

```
var titanicNumber = BigInt(1234567890)
```

```
var titanicDecimal = BigDecimal(63746.64636)
```

# Datatypes

The \_ (underscore) initializes the variables with the default value.

```
var stringDefault : String = _  
var integerDefault : Int    = _  
var floatDefault : Float   = _  
var doubleDefault : Double = _
```

# Datatypes - Multiple Assignments

```
// Multiple Assignments
val (name: String, age: Int, salary : Float) = ("Chola", 100, 102f)

println(name)
println(age)
println(salary)
```

Scala allows multiple assignments on the single line.

DataTypes.scala

# Syntax Rules

Arrays are written as `Array[T]` rather than `T[]`  
`Array[String]` rather than `String[]`

Array selections are written as `a(i)` rather  
than `a[i]`.

# Scala REPL

Scala provides a REPL for ease of development.

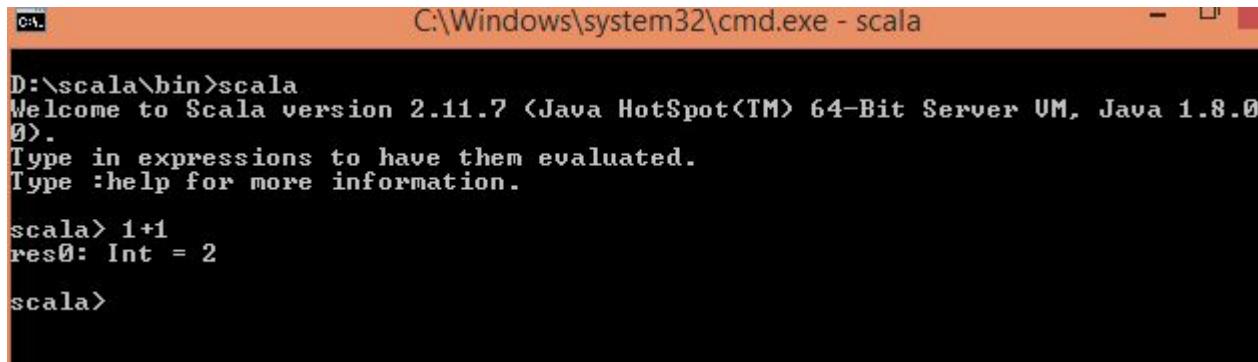
REPL => Read - Eval - Print - Loop

REPL helps in Experiment Driven Development  
(EDD)

# Scala REPL

To invoke scala REPL type in scala command inside the bin folder of the scala.

/scala\_installation/bin/scala



A screenshot of a Windows Command Prompt window titled "C:\Windows\system32\cmd.exe - scala". The window shows the Scala REPL running. The output is as follows:

```
D:\scala\bin>scala
Welcome to Scala version 2.11.7 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_0).
Type in expressions to have them evaluated.
Type :help for more information.

scala> 1+1
res0: Int = 2

scala>
```

# scalac and scala

```
/scala_installation/bin/scalac AppMain.scala
```

```
/scala_installation/bin/scala com.dmac.basic.AppMain
```

## Fast Scala Compiler (fsc)

```
/scala_installation/bin/fsc AppMain.scala
```

fsc uses a compilation daemon and a cache, so compiling repeatedly the same code base becomes much faster after the first dry run.

# Class and Package Names

Class name need not be same as the source file name

Package name need not be coherent with the naming convention

Source file can have more than one public classes



The screenshot shows the Eclipse IDE interface with the Package Explorer and Editor panes. The Package Explorer shows a project structure under 'ScalaTraining' with a 'src' folder containing several Scala files. One file, 'ClassNameNeedNotBeSame.scala', is selected and shown in the Editor pane. The code within this file demonstrates three key points:

- Package name:** The package declaration is `package any.package.name`, highlighted with a blue box.
- Class names:** It contains two class definitions: `ADifferentClassNameOtherThanASourceFile` and `AClassCanHaveMultiplePublicClasses`, both highlighted with orange boxes.
- Source file name:** The source file itself is named `ClassNameNeedNotBeSame.scala`, which is highlighted with a blue box.

```
ClassNameNeedNotBeSame.scala
package any.package.name

class ADifferentClassNameOtherThanASourceFile {

}

class AClassCanHaveMultiplePublicClasses {
```

ClassNameNeedNotBeSame.scala

chinnasamyad@gmail.com

# import statements

```
/* Importing multiple classes */  
import org.apache.spark.{SparkContext, SparkConf}
```

```
/* Importing all the classes */  
/* The _ can be compared as * operator on Java */  
import org.apache.spark._
```

```
/* Renaming a class at the time of import is allowed */  
import javax.swing.{ JFrame => MyAppFrame }
```

```
val myFrame = new MyAppFrame  
myFrame.show(true)
```

```
import javax.swing.JPanel  
val panel = new JPanel  
myFrame.add(panel)
```

Source: Basics.scala

chinnasamyad@gmail.com

# import statements

```
/*
  Importing statements inside the function block is permitted.
  This type of import is valid within that block.

*/
def main(args : Array[String]) {
    import javax.swing.JPanel
    val panel = new JPanel

    myFrame.add(panel)

}
```

Source: Basics.scala

# Invoking methods

```
def main(args : Array[String]) {  
  
    import javax.swing.JPanel  
    val panel = new JPanel  
  
    myFrame.add(panel)  
  
}  
myFrame add panel
```

Instead of invoking the add with a .  
it can be invoked using by

# Invoking methods

```
class MethodClass {  
  
    /**  
     * A Simple Function  
     */  
    def methodName(variableName : String) = {  
        println("Inside the methodName method - " + variableName)  
    }  
  
}
```

// An object is created using the new keyword  
val methodClass = new MethodClass

// All the below three are legal when calling a scala method

```
methodClass.methodName("SOME_STRING")  
  
methodClass methodName "SOME_STRING"
```

```
methodClass methodName {  
    "SOME_STRING"  
}
```

Refer ScalaMethods.scala

# return statement

return keyword is not mandatory.

The last line of the statement in the block is considered the return value.

The value of the last executed statement or expression is the return value.

# The first Simple Transformation map() and foreach()...

```
val list = List("Chinnasamy", "BigData")
```

```
list.map(x => x.toUpperCase)  
.foreach(each => println(each))
```

*// This is also legal*

```
list.map({x => x.toUpperCase})  
.foreach(each => println(each))
```

*// This is also legal*

```
list.map { x => x.toUpperCase }  
.foreach(each => println(each))
```

*// This is also legal*

```
list.map(_.toUpperCase)  
.foreach(println(_))
```

Refer FlatMapExp.scala

# Declarations and Type Inference

```
var declarationOfString = "I am a String Declared"  
  
var anotherVersionOfDeclarationOfString : String = "Another String Declared"  
  
println(declarationOfString); //Semicolon is not mandatory  
  
println(anotherVersionOfDeclarationOfString)  
  
val twelve = 3 + 9  
val assignMe = "Assign Me"
```

Scala language deducts the type automatically.  
This phenomenon is called by the name **Type Inference**.

Scala has a built in type-inference mechanism. The compiler deduces the type from the initialization expression of the variable.

# String Declarations

Scala strings are nothing but java strings,  
so all java string methods are applicable on scala strings.

Functional methods can be invoked in  
Strings

```
val aString = "CHE-GUERA"  
println(aString.drop(4)) // will print GUERA
```

```
val CONSTANT_STRING = "STRING"  
CONSTANT_STRING.drop(2).foreach { x => println(x) }
```

# asInstanceOf

Instead of the casting use asInstanceOf.  
- Child c = (Child) parent -

```
val instance = new InstanceOF  
instance passInRef(new Child)
```

```
class InstanceOF {  
  
  def passInRef(parent : Parent) {  
  
    val child = parent.asInstanceOf[Child]  
    println(child.childName)  
  
  }  
}
```

# Call by name, Call by value

A call-by-name passes a code block to the call and each time the call access the parameter, the code block is executed and the value is calculated.

Parameters to functions are by-value parameters.

If we write a function that accepts a expression as a parameter, we do not want it to be evaluated till it is invoked in our function.

# Call by name, Call by value

```
def returnMeSchema(): String = {
  println("Computing Schema...");
  "MYSQL"
}
```

```
def callByValue(input: String): Unit = {
  println("First Time print = " + input)
  println("Second Time print = " + input)
}
```

```
def callByName(input: => String): Unit = {
  println("First Time print = " + input)
  println("Second Time print = " + input)
}
```

```
/*
 * Computing Schema...
 * First Time print = MYSQL
 * Computing Schema...
 * Second Time print = MYSQL
 */
callByName(returnMeSchema)
```

```
/**
 * Computing Schema...
 * First Time print = MYSQL
 * Second Time print = MYSQL
 */
callByName(returnMeSchema)
```

# Scala Data types

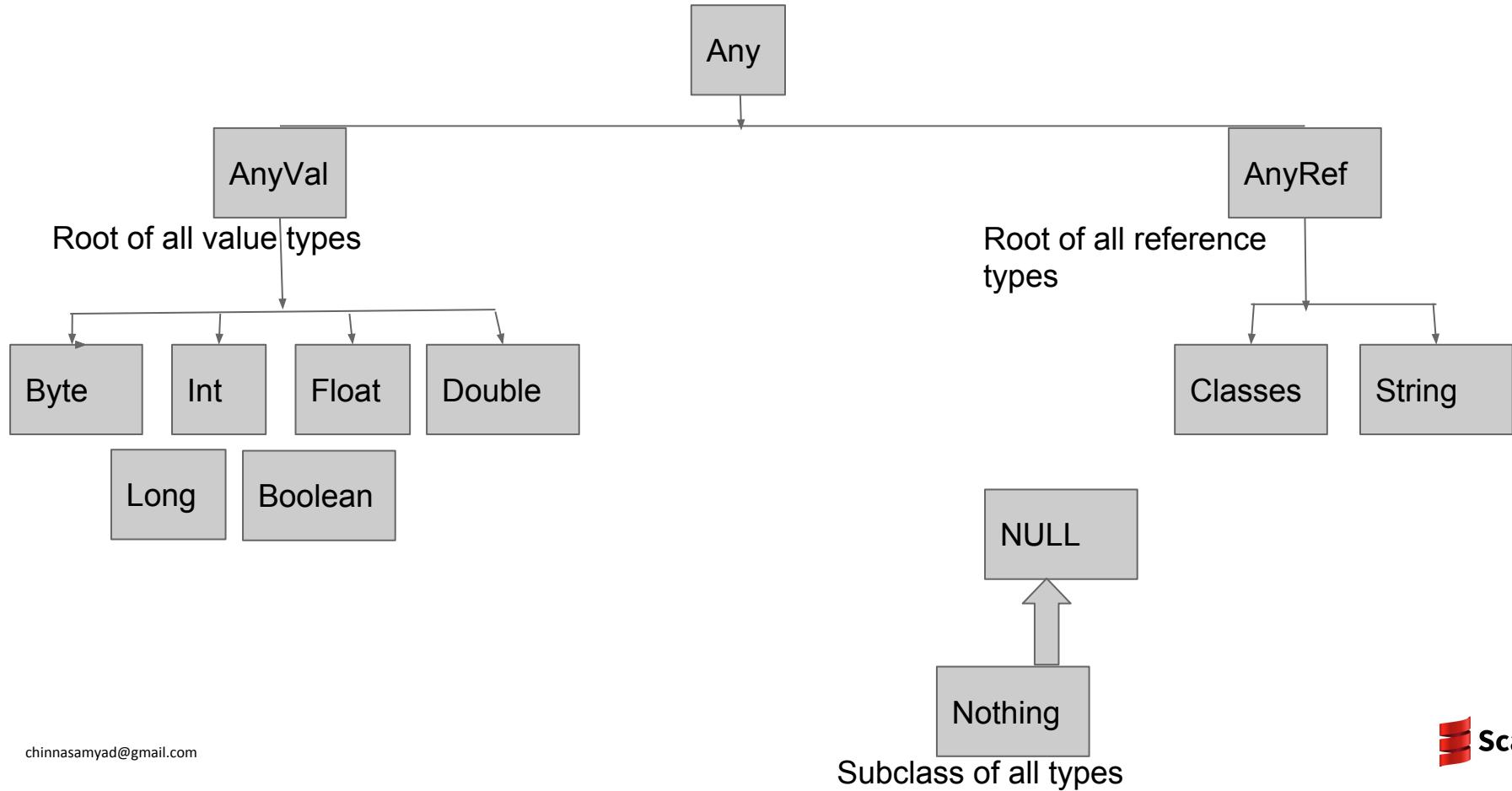
Byte  
Short  
Int  
Long  
Float  
Double  
Char  
String  
Boolean  
Unit  
Null  
Nothing  
Any  
AnyRef

- No value (Equivalent to void in java)

- The subtype of every type. Includes no values

All are objects. There are no primitives as that of Java.

# Scala Data types



# A single source file can contain many packages

```
package parentpackage {  
    class PublicClass {  
        }  
        import com.dmac.basic.parentpackage.PublicClass  
        import com.dmac.basic.parentpackage.ChildClass  
        import com.dmac.basic.childPackage.ChildClass  
  
        package childPackage {  
            class ChildClass extends PublicClass {  
                def printer () {  
                    }  
                }  
            }  
        }
```

# Expressions and Conditionals

# Expressions and Conditionals

def

var and val expressions

if...else

for

do...while

match

try...catch...finally

# for...loop expression

```
// Simple loop //
```

```
for (i <- 1 to 10)  
    println(i)
```

## Syntax

```
for (<identifier> <- <iterator> if (<boolean_expression>) )  
{
```

```
// Nested Loop //
```

```
for { i <- 1 to 10  
      j <- 1 to 10 }
```

```
}
```

```
    println(i, j)
```

Refer to [ConditionsAndExpressions.scala](#)

# for...loop expression

```
println("\n\n\n Reverse Printing")
```

```
for (i <- 50 to 1 by -1)  
    println(i)
```

Prints from 50 decrementing by 1

## Syntax

```
for (<identifier> <- <iterator> if (<boolean_expression>) )  
{  
}  
}
```

Refer to [ConditionsAndExpressions.scala](#)

# for...loop expression

For Loop with Boolean Expression

```
val authProperties = List (  
    "AUA",  
    "ASA-Service Agency",  
    "TID",  
    "Version",  
    "UID",  
    "TXN",  
    "Encrypted-SKEY",  
    "Encrypted-SKEY-CertificateIdentifier",  
    "Data-Type",  
    "Data-Content-Encrypted-PID-XML",  
    "Encrypted-SKEY")
```

```
for (authElement <- authProperties if (authElement.startsWith("Data")))  
  println(authElement)
```

Refer to [ConditionsAndExpressions.scala](#)

# break statement

```
import scala.util.control.Breaks  
  
val loop = new Breaks  
  
println("\n\n\n Reverse Printing")  
  
loop.breakable {  
    for (i <- 50 to 1 by -1) {  
        println(i)  
  
        if (i==40)  
            loop.break  
    }  
}
```

Refer to [ConditionsAndExpressions.scala](#)

# do...while expression

```
var count = 0
```

```
do {  
    count += 1  
    println("My count = " + count)  
}  
while(count < 10)
```

Refer to [ConditionsAndExpressions.scala](#)

# yield

Creates a new data structure and returns as a val.

Yield acts as a buffer and each iteration, the variable gets appended by the item based on the code after yield.

```
val vectorList = Vector(1,2,3,4,5)
```

```
val vectorReturn = for (element <- vectorList) yield element*2
```

Output : Vector(2, 4, 6, 8, 10)

Refer to [ConditionsAndExpressions.scala](#)

# yield

The use of yield with the conditional if is shown below

```
val vectorList = Vector(1,2,3,4,5)
```

```
val evenNumberMultiplier = for (element <- vectorList if (element % 2 == 0))
                                yield element*2
```

Output : Vector(4, 8)

Refer to [ConditionsAndExpressions.scala](#)

# if...else expression

```
if (<boolean_expression>)
    <code>
else  <code>
```

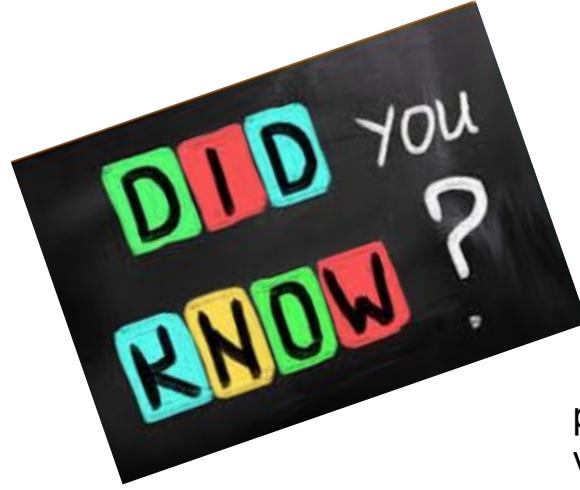
```
val x = if (a) y else z
```

Refer to [ConditionsAndExpressions.scala](#)

# match

```
val month = input match {  
    case 1 => "JAN"  
    case 2 => "FEB"  
    case 3 => "MAR"  
    case 4 => "APR"  
    case 5 => "MAY"  
    case 6 => "JUNE"  
    case 7 => "JULY"  
    case 8 => "AUG"  
    case 9 => "SEP"  
    case 10 => "OCT"  
    case 11 => "NOV"  
    case 12 => "DEC"  
  
    case 0 | 13 => "Give an integer between 1 and 12"  
    case _ => "INVALID_MONTH"  
}
```

Refer to [Matcher.scala](#)



```
print("Enter a month : ")
val input = StdIn.readInt // Read the Input from the console
```

In comparison to the Scanner class of Java.

Refer to [Matcher.scala](#)

# try...catch...finally

```
try {  
  
    val file = new FileReader("");  
    file.close()  
  
}  
catch {  
    case exception: FileNotFoundException => {  
        println("Hey Chin - File is not found")  
    }  
}  
finally {  
    println("Executing Finally")  
}
```

# try...catch...finally

```
try {  
    val f = new FileReader("input.txt")  
  
}  
catch {  
    case ex: FileNotFoundException => {  
        println("File Not Found " + ex.getMessage)  
    }  
    case ex: IOException => {  
        println("IO Exception " + ex.getMessage)  
    }  
}  
finally {  
    println("Executing in finally")  
}  
  
def methodThrowsAException() = {  
    //val f = new FileReader("input.txt")  
    throw new IllegalStateException("Illegal Argument")  
}
```

# try...catch...finally

```
try {  
    val f = new FileReader("input.txt")  
  
}  
catch {  
    case ex: FileNotFoundException => {  
        println("File Not Found " + ex.getMessage)  
    }  
    case ex: IOException => {  
        println("IO Exception " + ex.getMessage)  
    }  
}  
finally {  
    println("Executing in finally")  
}  
  
def methodThrowsAException() = {  
    //val f = new FileReader("input.txt")  
    throw new IllegalStateException("Illegal Argument")  
}
```

# Try...catch...finally

```
val eh = new ExceptionHandler

try {
  eh.methodThrowsAException
}

catch {
  case ex: FileNotFoundException => { println("File Not Found " + ex.getMessage) }
  case ex: IOException      => { println("IO Exception " + ex.getMessage) }
  case ex: IllegalStateException => { println("Message - " + ex.getMessage) }
}
```

*ExceptionHandling.scala*

# String Interpolation

Scala offers a very elegant way to represent strings. The representation is called by the name string interpolation.

```
val goldRate = 2400.34  
println(f"Todos gold rate = $goldRate")
```

```
val exceptionMessage = "SLZ Zone Runtime Exception occurred"  
println(s"Exception message is : $exceptionMessage")
```

In java, you can achieve this using String.format method.

Interpolation.scala

# Classes

Classes are the core building block of object-oriented languages.

A combination of data structures and functions.

The noun names of the problem statement is chosen as the class name.

# class and object

An object has only one instance. A class can have multiple instances.

A component of type object is a singleton object.

```
class ReadCSVFile {  
  
    def readCSVFile(csvFileName : String) : Unit = {  
        // reading the CSV File Logic  
    }  
}  
object SparkReadFile {  
  
    def main(args : Array[String]) : Unit = {  
        println("SparkReadFile")  
    }  
}
```

ReadCSVFile is a class which can have constructors a typical class

SparkReadFile is a single instance and cannot have constructors but can extend other classes

# companion object and a companion class

// Companion class of the object

```
class FactoryMethod {  
  
    def executeSomeComplexBusinessLogic : String = {  
        println("Execute Inside Factory Method")  
        "executed_success"  
    }  
}
```

// Companion Object of the class FactoryMethod

```
object FactoryMethod {  
  
    def execute() : String = {  
        val fm = new FactoryMethod  
        fm.executeSomeComplexBusinessLogic  
    }  
}
```

A companion object shares the same name as that of the class.

You will find this pattern to be used inside the scala source code.

CompanionObject.scala

# Class, subclass, abstract class, traits

Use the **extends** keyword to extend a class.

And use the **override** keyword to override a super class method.

Use abstract class when a constructor is to be given or else you can use a trait.

When extending third party traits, use an abstract class to extend traits.

# Classes

```
class CSVLogReader extends LogReader {  
  
    override def readFileName() : String = {  
        //CSV_FILE"  
        super.readFileName()  
    }  
}  
  
class LogReader {  
  
    def readFileName() : String = {  
        val fileName = "SIMPLE_FILE_1"; fileName  
    }  
}
```

Refer to ScalaClass.scala in the code

## Instantiating a class

```
val csvReader = new CSVLogReader  
  
val csvReader = new CSVLogReader()  
  
val csvReader = new CSVLogReader();  
  
val csvReader = new CSVLogReader;  
  
    println(csvReader.readFileName())
```

# Class and traits

Traits define the object types by specifying the signature of the method.  
(In comparable to Java interfaces)

Scala allows traits to have partial implementations.

In java, classes implements interfaces, whereas in scala a class extends a trait.

Use the override keyword to override the concrete implemented method in trait.

# Class and traits

```
class VoldemortKeyValueStore extends KeyValueStoreTrait {  
  
    def get() : String = {  
        return ""  
    }  
  
    def put(value : Any) : Boolean = {  
  
        val recordInserted = true  
  
        return recordInserted  
    }  
  
    override def dataStoreName(): String = {  
        return "VOLDEMORT"  
    }  
  
    trait KeyValueStoreTrait {  
  
        def put(value : Any) : Boolean  
  
        def get() : Any  
  
        def dataStoreName(): String = {  
            return "REDIS"  
        }  
    }  
}
```

# Traits can extend other traits

```
trait ZeusTrait {  
    def zeusMethod  
}
```

```
trait SupremeTrait {  
    def supremeMethod  
}
```

```
trait SuperTrait {  
    def superMethod() : String = {  
        "SUPER_TRAIT"  
    }  
}
```

```
trait MyTrait extends SuperTrait  
    with SupremeTrait with ZeusTrait {  
}
```

# Traits can be implemented by anonymizing

```
trait Debugger {  
    def printTrace(input : String) = {  
        println(s"Tracing ... : $input")  
    }  
}  
  
// A trait can be instantiated in a anonymous way  
val debugger = new Debugger {  
}  
  
debugger printTrace "debugger trait anonimized"  
  
trait MachineLearning {  
  
    var algorithmMarkedByUser = ""  
  
    def regression(input : String) {  
        println("Implementing Regression")  
    }  
  
    def supportVectorMachine(input : String)  
    def clustering(input: String) = {  
        println("Implementing Clustering")  
    }  
}  
  
// A trait can be instantiated in a anonymous way.  
// But abstract methods has to be implemented.  
val machineLearning = new MachineLearning {  
    override def supportVectorMachine(input: String) = {  
    }  
}
```

# Class constructor

```
// Primary Constructor  
class HighClass(name : String, id: Int) {  
  
    // Auxillary Constructor  
    def this(name: String) {  
        this(name, 88)  
    }  
  
    def this() {  
        this("_NO_ARG_")  
    }  
}
```

```
// Invoking Primary Constructor  
val hc = new HighClass("HIGH_CLASS", 100)
```

```
//Invoking Auxillary Constructor  
val clazz = new HighClass("ONE_ARG")
```

```
//Invoking Auxillary Constructor  
val hClass = new HighClass
```

# Class constructor

Class constructor containing var and val parameters.

```
class HighClass(val name : String, var id: Int) {  
  
    def this(name: String) {  
        this(name, 88)  
    }  
  
    def this() {  
        this("_NO_ARG_")  
    }  
}
```

```
val hc = new HighClass("HIGH_CLASS", 100)
```

```
//hc.name = "HC" // Cannot assign to val  
hc.id = 999
```

*Members defined as val cannot be assigned a new value.*

*Members defined as var can be reassigned.*

# Object class constructor

The object class can have constructor using the companion object.

```
class ObjectClass {  
  
    private var objectName = ""  
    val oc = ObjectClass("Richard_Stallman")  
  
    def getNameOfObject() : String = {  
        objectName  
    }  
}  
  
object ObjectClass {  
  
    def apply(name : String) : ObjectClass = {  
        var objectClass = new ObjectClass  
        objectClass.objectName = name  
        objectClass  
    }  
}
```

# Object class constructor

The object class can have constructor using the companion object.

```
object DataStructure {  
  
    def apply(name: String): DataStructure = {  
        var ds = new DataStructure  
        ds.set(name)  
        ds  
    }  
}  
  
class DataStructure() {  
  
    var _name = ""  
  
    def get(): String = {  
        _name  
    }  
  
    def set(name : String) {  
        _name = name  
    }  
}
```

**val *ds* = DataStructure("HASHMAP")**  
*println(ds.get())*

# Mixin

Mixin is the process by which a trait is mixed with a class to form a robust design.

A mixin is achieved by using the **extends** and **with** keyword.

# trait as mixins

```
class OpenSSO extends Authentication {  
  
    def login(userName : String, password : String) = {  
        true  
    }  
  
    def logout(userName: String) = {  
        true  
    }  
  
    def checkForLogin() = {  
        println("login check using opensso library")  
    }  
  
    override def protocol = {  
        "OPEN_SSO"  
    }  
}
```

```
trait Authentication {  
  
    def login(userName : String, password : String) : Boolean  
  
    def logout(userName: String) : Boolean  
  
    def checkForLogin()  
  
    def protocol = {  
        "_NO_PROTOCOL_IMPLEMENTED_"  
    }  
}  
  
class OpenLDAP extends Authentication {  
  
    def login(userName : String, password : String) = {  
        true  
    }  
  
    def logout(userName: String) = { true }  
  
    def checkForLogin() = { println("login check using ldap library") }  
  
    override def protocol = {  
        "OPEN_LDAP" }  
}
```

# trait as mixins

```
trait AuthenticatioHBASELoggingTrait {  
  
    def logIntoElasticSearch(userName : String) = {  
        println("Logging into HBASE")  
    }  
}  
  
trait NotifyThirdPartyTrait {  
  
    def notify(userName : String) = {  
        println(s"Notifying that $userName has logged in")  
    }  
}
```

```
class OpenSSOClient extends OpenSSO  
    with AuthenticatioHBASELoggingTrait  
    with NotifyThirdPartyTrait {  
    }  
}
```

# When to use abstract class

```
trait DatabaseOperationTrait { // (name : String) {  
    def save  
    def update  
    def delete  
}  
  
abstract class AbstractDatabaseOperation(name:String) {  
  
}  
  
class MySQL(dataBaseName:String) extends  
    AbstractDatabaseOperation(dataBaseName) {  
  
}
```

*Use abstract class when you want to pass in  
a constructor*

*Use it judiciously when interacting with third  
party trait libraries*

*A trait cannot have a constructor, while an abstract class can have a  
constructor.*

# traits with object classes

```
object TraitRules extends App {  
  
    val ocean = new OceanApp with Debugger  
    ocean.printTrace("Tracing an error")  
  
}
```

```
class OceanApp {  
  
}  
  
trait Debugger {  
    def printTrace(input : String) = {  
        println(s"Tracing ... : $input")  
    }  
  
}
```

TraitRules.scala

# sealed classes

The sealed keyword enforces that all subclasses must be declared in the same source file.

*SealedClasses.scala*

```
sealed class FileFactory {
```

```
}
```

```
class XMLFileFactory extends FileFactory {
```

```
}
```

Usage: The Option class in the scala library is a  
sealed class.  
sealed abstract class Option.

# enumeration

An enumeration is defined by extending the Enumeration class of scala.

```
object CountryCode extends Enumeration {  
    type CountryCode = Value  
  
    val INDIA = Value("IN")  
    val FRANCE = Value("FR")  
    val RUSSIA = Value("RU")  
}
```

## Invoking an enumeration

```
for ( country <- CountryCode.values)  
    println(s"$country" + " " + country.id)  
  
println(CountryCode.FRANCE)
```

# Lazy Evaluation

A variable defined `lazy` is evaluated only when it is needed.

Lazy evaluation is used mainly on

- File based properties

- Opening a database collections

```
lazy val expensiveResource : Int = doExpensiveOperation

def doExpensiveOperation() : Int = {
    println("Doing Expensive Operation")
    999
}
```

# Lazy Evaluation

```
class Lazzy {  
  
    var randomVar = { println("Random Var"); Random.nextInt}  
    val randomX = { println("Random X"); Random.nextInt}  
  
    lazy val randomY = { println("Random Y"); Random.nextInt()  
}
```

```
}
```

```
def main(args: Array[String]) {  
    val lazzy = new Lazzy  
  
    //println(lazzy.randomX)  
    //println(lazzy.randomVar)  
  
    //println(lazzy.randomY)  
}
```

# Lazy Collection - Stream

Stream collection are lazy in nature.

```
def returnStream() : Stream[AnyRef] = {
  val stream = Stream("Oracle", "DB2")

  val numberStream = Stream(1,2,3)
  numberStream.take(2).foreach { x => println(x) }

  return stream
}
```

# Case classes

Case classes have the hashCode, equals, method to be created automatically.

By default, all property in a case class have val properties.

Case classes are mainly used as POJO's or DTO(Data Transfer Objects).

# Case classes

```
case class DBProperties(var DataBaseName: String,  
                      var portNo : Int,  
                      var connectionString :String,  
                      var schemaName: String)
```

```
var dbProp = DBProperties("slz_core",3306,"jdbc:mysql:slz-zone3","slz_core")  
  
println(dbProp.connectionString)  
dbProp.connectionString = "jdbc:oracle:slz-zone2"  
  
println(dbProp.connectionString)
```

# Case classes

Case classes can extend other case classes

Case class can also be final.

If the case class is made final, then it can't be extended.

Case class can also extend other normal classes.

# Value classes

Value classes are the custom classes that extend the AnyVal

# Value classes

Value classes are those which extend the AnyVal

```
val dollar = new Dollar(100)  
println(dollar)
```

```
class Dollar(val value: Float) extends AnyVal {  
    override def toString = value.toString  
}
```

# Strings

Representing multi-line strings

```
val multiLineString = "First Line " +  
                      "Second Line"
```

```
val multiLineStringAlias = """ This is first Line  
                           This is second line  
                           This is third line """
```

# Visibility Scopes

**public**

**\_no\_keyword\_**

**protected**

**protected**

**private**

**private**

**scoped protected**

**protected[scope]**

**scoped private**

**private[scope]**

**scope: class name, package name, this**

# Visibility Scopes

```
package parentpackage {  
  
    class PublicClass {  
  
        private          val privateValue      = 22  
        protected       val protectedValue    = 33  
  
        private[parentpackage] val privatePackageScoped = 44  
        private[PublicClass]   val privateClassScoped  = 55  
        protected[PublicClass] val protectedClassScoped = 66  
    }  
  
    class ChildClass extends PublicClass {  
  
        def printer () {  
            println(this.protectedValue)  
            println(this.privatePackageScoped)  
            println(this.protectedClassScoped)  
        }  
    }  
}
```

# Visibility Scopes

```
package childPackage {  
  
    class ChildClass extends PublicClass {  
        def printer () {  
            this.protectedValue  
            this.protectedClassScoped  
        }  
    }  
}
```

VisibilityScopes.scala

chinnasamyad@gmail.com

# Scala Collections

# Scala Collections

Base classes

Iterable, Seq, Set, Map

Immutable Collections

List, Stream, Vector, Range, String, Map, Set

Mutable Collections

Arrays

# Scala Collections

List	An immutable list
Stream	Same as list, but the tail is evaluated on demand.
Set	Collection with unique elements
Map	Key/value pair DS.
Array	A mutable collection
Tuple	Encapsulated store
Range	Range of values containing integers
Vector	For fast access

# List

Fundamental data structure.

Simple Immutable single linked list.

List are immutable in nature.

```
val aadhaarPktState = List("PKT_REJECTED",
                           "PKT_DUPLICATE",
                           "PKT BIOMETRIC_STAGE_ABIS")
```

```
println(aadhaarPktState(1))
```

```
for(i <- aadhaarPktState)
  println(i)
```

List can also be of varied data types.

```
val multiTypeList = List(500, "INTERNAL_SERVER_ERROR", true)
```

multiTypeList is of type List[Any]

# List

To modify a list, convert it to Buffer, modify it and then convert back to list.

```
val countryCodeList = List("IN", "SL", "CA")
```

```
val buffer = countryCodeList.toBuffer
```

```
buffer += "NL"
```

```
print(buffer.toList)
```

# List

The addString() method is completely a different one.  
It's just a utility method.

```
val countryCodeList = List("IN", "SL", "CA")
```

```
val output = new StringBuilder("The various countries are : ")
```

```
val newList = countryCodeList.addString(output, "[", ",", "]")
```

```
print(newList)
```

o/p:

The various countries are : [IN,SL,CA]

# List

List can also be created using the :: (cons - construct) operator as shown below.

When using the cons operator, Nil has to be specified at the end of the list.

```
val errorCodes = "200_OK" :: "404_BAD_REQUEST" :: "500_INTERNAL_SERVER_ERROR"  
                  :: "520_UNKNOWN_ERROR" :: Nil  
  
errorCodes.foreach { errorCode => println(errorCode) }
```

CollectionClasses.scala

# Set

Set is a immutable, unordered collection of unique elements.

```
val setOfNumbers = Set(1,1,2,2,3,3)  
setOfNumbers.foreach { x => println(x) }
```

# Set - Mutable Set

```
{
```

```
import scala.collection.mutable.Set
println("\n\n\n Printing The Mutable Set")
val mutableSet = Set(1,2,2,3,3)
mutableSet.add(4)
mutableSet.foreach { x => println(x) }
```

```
}
```

# Map

Map is a key-value pair.

Aka hashmap, associative arrays in other languages. Elements specified in the Map are represented as Tuples in Scala.

```
val cacheCleanerMap = Map(1 -> "AuaCacheCleaner",
                          2 -> "AsaCacheCleaner",
                          3 -> "ResidentDataCacheCleaner",
                          4 -> "LicenseCacheCleaner")
```

```
cacheCleanerMap.foreach(x => println(x._1 + x._2))
```

# Map - Mutable Map

```
import scala.collection.mutable.Map

var cacheCleanerMutableMap = Map(1 -> "AuaCacheCleaner",
                                2 -> "AsaCacheCleaner",
                                3 -> "ResidentDataCacheCleaner",
                                4 -> "LicenseCacheCleaner")
cacheCleanerMutableMap += (5 -> "_END_")

cacheCleanerMutableMap foreach(x => println(x._1 + x._2))
```

# Map - Mutable Map - getOrElseUpdate

```
val countryCode = collection.mutable.Map("IN" -> "India",  
                                         "FR" -> "France")
```

```
val retrieveCodeNotPresent = countryCode.get("US")
```

// Will update the original map

```
val retrieveCodePresent = countryCode.getOrElseUpdate("US", "UnitedStates")
```

// Will not update the original map

```
val retrieveCodePresent = countryCode.getOrElse("US", "UnitedStates")
```

```
val updatedCode = countryCode.get("US")
```

```
println(retrieveCodeNotPresent)  
println(retrieveCodePresent)  
println(updatedCode)
```

// Will return None  
// Will return UnitedStates  
// Will return UnitedStates

# Mutable vs Immutable

## Immutable Type

`collection.immutable.Set`

`collection.immutable.Map`

`collection.immutable.List`

## Mutable Type

`collection.mutable.Set`

`collection.mutable.Map`

`collection.mutable.Buffer`

# Mutable Stores

```
// Uses List (Linked List) Internally  
val listBufferStore = collection.mutable.ListBuffer(1,2,3)
```

```
override def productsFromUser(user: User): List[Product] = {  
    var listOfProducts = ListBuffer[Product]()  
    user.orders.map(eachOrder => {  
        eachOrder.items.foreach(  
            eachSeq => listOfProducts += eachSeq._1)  
    })  
    listOfProducts.toList  
}  
case class Product(name: String, price: Int, categoryName: Option[String])
```

1. A list buffer containing a custom DTO product

2. A list buffer adding in Product

3. A list buffer converted to a immutable list.

```
// Uses Arrays Internally  
val arrayBufferStore = collection.mutable.ArrayBuffer(1,2,3)
```

# Arrays

Arrays are fixed size, mutable collection.  
It's a wrapper over the Java array type.

# Range

Range generates a sequence. The three operators for sequence are:

<i>to</i>	- <i>inclusive</i>
<i>until</i>	- <i>exclusive</i>
<i>by</i>	

```
var range100 = 1 to 100
```

```
var range50 = 1 to 100 by 2
```

```
println(range100)
```

```
for (i <- range100)  
  println(i)
```

# Range

```
// Will consider the elements from 1 to 20  
val range20 = 1 until 21
```

```
println(range20)
```

Output: Range(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20)

```
range20.foreach { element => println(element) }
```

```
for (element <- range20)  
  println(element)
```

# Range - Character Ranges

```
val characterRange = 'a' to 'z'
```

```
println(characterRange)
```

```
NumericRange(a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q,  
r, s, t, u, v, w, x, y, z)
```

```
val characterRangeCaps = 'A' to 'Z'
```

```
println(characterRangeCaps)
```

```
NumericRange(A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q,  
R, S, T, U, V, W, X, Y, Z)
```

```
val characterRangeByTwo = 'a' to 'z' by 2
```

```
println(characterRangeByTwo)
```

```
NumericRange(a, c, e, g, i, k, m, o, q, s, u, w, y)
```

```
println(characterRange(1))
```

```
b
```

# Tuple

An ordered collection of two or more values.

All values may be of same or different types.

Represented as :

```
val tupleStore = ("Insight into scala", 5, true)
```

# Tuple

```
val tupleStore =  
(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,"17",18,19,20,21,22)
```

```
println(tupleStore._22) // Prints the 22 value in the tuple
```

```
val complexTuple = ("Insight into scala", 5, true)
```

Note: There can be a maximum of only 22 elements in a tuple

# Tuple - Iterating a tuple

```
val auabaseCodes = ("PACK_CBE",
                    "PACK_BANG",
                    "PACK_CHE",
                    "PACK_RAN")
```

```
println(auabaseCodes.productElement(0))
```

```
for (i <- 0 to 3)
  println(auabaseCodes.productElement(i))
```

# Vector

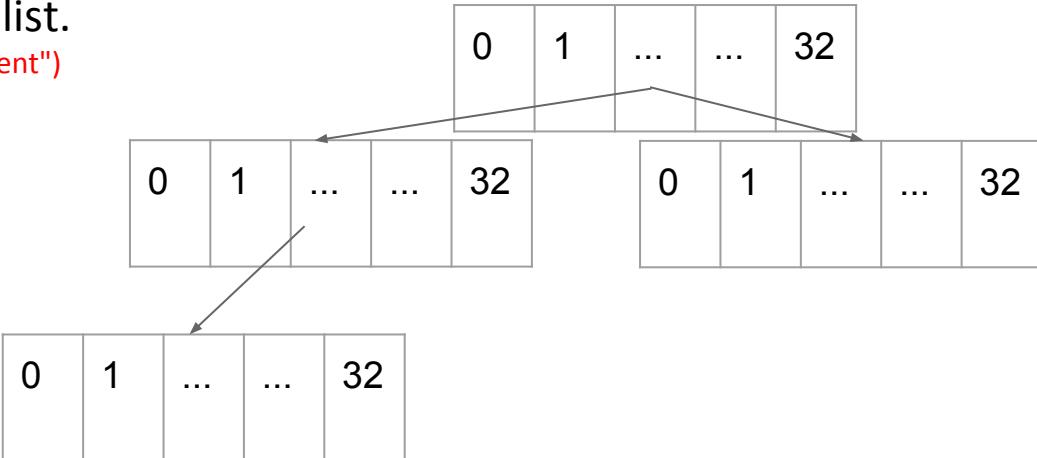
Vector initially with 32 elements is a simple array. i.e., with a branching factor of 32.

Vector has its creation analogy to the list.

```
val vectorStore = Vector(1, 2, 3, 4, "Last Element")
```

Vector forms a shallow tree

Vector is used for fast random access.



# Vector

```
val vectorStore = Vector(1, 2, 3, 4, "Last Element")
val vectorStoreUpdated = 100 +: vectorStore :+ 999

vectorStoreUpdated.foreach { x => println(x) }
```

- +: Appends the element to the head of the dataset.
- :+ Appends the element to the end of the dataset.

# Parallel Collections

Parallel data structure can be obtained from any collection using the par method.

```
val multiTypeList = List(500, "INTERNAL_SERVER_ERROR", true)  
val parallelMultiTypeList = multiTypeList.par
```

It can also be constructed by following means.

```
import scala.collection.parallel.mutable.ParArray  
import scala.collection.parallel.mutable.ParMap
```

```
val parallelArray = ParArray(1,2,3)
```

```
val parallelHashMap = ParMap("key" -> "value")
```

# Parallel Collections

```
import scala.collection.parallel.mutable.ParArray  
import scala.collection.parallel.mutable.ParHashMap  
import scala.collection.parallel.mutable.ParMap  
import scala.collection.parallel.mutable.ParSet  
import scala.collection.parallel.mutable.ParSeq
```

```
import scala.collection.parallel.immutable.ParHashMap  
import scala.collection.parallel.immutable.ParHashSet  
import scala.collection.parallel.immutable.ParVector  
import scala.collection.parallel.immutable.ParMap  
import scala.collection.parallel.immutable.ParSet  
import scala.collection.parallel.immutable.ParSeq
```

# Immutable to mutable

```
val cacheCleanerMap = Map(1 -> "AuaCacheCleaner",
                         2 -> "AsaCacheCleaner",
                         3 -> "ResidentDataCacheCleaner",
                         4 -> "LicenseCacheCleaner")
```

```
// Converting a Immutable Map to Mutable Buffer
var cacheCleanerBuffer = cacheCleanerMap.toBuffer
cacheCleanerBuffer += (5 -> "")
```

```
val immutableCacheCleanerMap = cacheCleanerBuffer.toMap
```

# When to use

## List

$O(1)$  for head element. But  $O(n)$  for remaining elements.  
If you want prepend, then use List.

## Vector

$O(1)$  for almost all operations.  
For random access, use Vector.

## Array

Do not use for prepending elements.

# Scala Function Combinators

map

foreach

filter

zip

partition

.... and many more.

# Scala Functions

Functions are first class citizens in scala

# Scala Method Declaration

```
def method_name(id : type) : return_type = {  
}
```

```
def countNumberOfWords(fileContents : String) : Integer = {  
  
    val st = new StringTokenizer(fileContents)  
    return new Integer(st.countTokens())  
  
}
```

# Procedures

A Procedure is a function that doesn't have a return value.

```
//A interpolated float and string values
def todaysGoldRate(goldRate: Double) = println(f"Todos Gold Rate $goldRate")

def logWarn(warningMessage : String) = println(s"Logging a warning message - $warningMessage")

todaysGoldRate(2390.78)
logWarn("OOM Error may occur")
```

Check out Procedure.scala example

# Function with empty parentheses

```
// Function with empty parameters  
  
def fetchNoSQLStore() : String = "REDIS"
```

```
// Function with empty parameters with the brackets  
  
def fetchCache() : String = {  
    return "voldemort"  
}
```

Check out ScalaFunctions.scala

# Function with named parameters

```
val clazz = new AScalaClass
```

```
// Calling a function with named parameter
```

```
clazz.fetchLicenseRecords(fetchStrategy = "eager", pageStart = 200, tableName =  
"LearningZoneOrganizations", pageEnd = 300)
```

Scala allows to call a method by using parameter names in the method call.  
Which gives the flexibility of specifying orderless parameters.

```
/**  
 * A simple function  
 */  
def fetchLicenseRecords(tableName : String,  
                      pageStart : Int,  
                      pageEnd : Int,  
                      fetchStrategy: String) : Boolean = {  
  
    println("TableName - " + tableName)  
    println("Fetch Strategy - " + fetchStrategy)  
    println("Start - " + pageStart)  
    println("End - " + pageEnd)  
  
    return true  
}
```

# Function with default values

```
clazz.retrieveEnvironmentProperties("araviuser")
clazz.retrieveEnvironmentProperties("chinuser", "QA")
```

```
def retrieveEnvironmentProperties(userName: String, environment : String =
"DEV") = {
    println("Environment - " + environment)
    println("UserName - " + userName)
}
```

# Function names with symbols

```
val methodClass = new MethodClass

println(methodClass.++("ARAVINDH "))
println(methodClass ++ "ARAVINDH " )

/**
 * Methods names can be of symbols
 */
def ++ (input : String) : String = {
    input.concat(input)
}
```

ScalaMethods.scala

# Function names with symbols

```
val methodClass = new MethodClass
```

```
println (methodClass +!@%%%%%%%%%&* "NAME")
```

```
def +!@%%%%%%%%%&* (input : String) : String = {  
  "COMPLEX_NAMING"  
}
```

# Function with vararg parameters

```
/**  
 * Function as a variable argument parameter  
 */  
def saveAUA(aua : String*) : Unit = {  
    for (i <- auas)  
        println(i)  
}  
  
clazz.saveAUA("BANG_AUA", "CHENNAI_AUA", "COIM_AUA")
```

# Function Literals

Function Literals are nothing but lambda expressions which came in to reduce the boiler plate code of what we use to write as Anonymous functions.

Function literals are derived from the lambda calculus syntax.

$\lambda$ -calculus is a mathematical logic for expressing computation based on abstraction and application using variable binding and substitution.  $x \rightarrow x * 2$  is the lambda calculus expression from which lambda calculus was inspired and developed.

# Function Literals

```
{    variable_name  =>  << block_of_code >> }
```

```
val complexQueing = List("QUEUE",
    "TOPIC", "FANOUT", "_NO_QUEUE_");
```

```
complexQueing.foreach { name => println(name) }
```

# Higher Order Functions

Functions that take other functions as parameters or that return function as results are called higher order function.

A function taking a function as a parameter or in other words the result of a function is another function.

To elucidate, it is a function that has a value with a function type as input parameter or a return value.

Very flexible way of composing programs.

# Higher Order Functions

*First order function* is one that acts on list, int, string etc.., types.  
But a higher order function is one that acts on function types.

# Higher Order Functions

```
// Higher Order Function
def process(fn: (String,String) => String, x, y, z, .....)= {
  fn(actual_input, string_to_be_concatenated)
}
```

```
// Higher Order Function
def process(fn: (String,String) => String, actual_input:String, string_to_be_concatenated :String) = {
  fn(actual_input, string_to_be_concatenated)
}
```

# Higher Order Functions

```
// Higher Order Function
def process(fn: (String, String) => String, actual_input: String, string_to_be_concatenated : String) = {
  fn(actual_input, string_to_be_concatenated)
}

val resultAfterPostfix = process(stringProcessor.postFixString, "Martin", "Odersky")
val resultAfterPrefix = process(stringProcessor.prefixString, "Martin", "Odersky")

// An anonymous function executing the same
val anonymousFunctionResult = process((x,y) => x.concat(y), "Martin", "Odersky")

// Using an _ and _
val resultAfterPostfix = process(_.concat(_), "Martin", "Odersky")
```

```
class StringProcessor {

  def prefixString(inputString : String, string_to_be_concatenated: String) : String = {
    string_to_be_concatenated.concat(" " + inputString)
  }

  def postFixString(inputString : String, string_to_be_concatenated: String) : String = {
    inputString.concat(" " + string_to_be_concatenated)
  }
}
```

# Anonymous Functions

// Higher Order Function

```
def process(fn: (String, String) => String, actual_input: String, string_to_be_concatenated : String) = {  
    fn(actual_input, string_to_be_concatenated)  
}
```

// An anonymous function executing the same

```
val anonymousFunctionResult = process((x,y) => x.concat(y), "Martin", "Odersky")
```

```
class StringProcessor {
```

```
    def prefixString(inputString : String, string_to_be_concatenated: String) : String = {  
        string_to_be_concatenated.concat("_" + inputString)  
    }
```

```
    def postFixString(inputString : String, string_to_be_concatenated: String) : String = {  
        inputString.concat("_" + string_to_be_concatenated)  
    }
```

# Function as a variable

A function can be assigned to a variable.

In this case the original object is modified.

```
// A Function Name as a Variable  
val functionNameAsVariable = (i:Int) => { i + 10}
```

Use case : When the original object is to be modified, then you can use this methodology.

# Function as a variable

```
val simpleList = List(1,2,3,4,5)
```

```
// A Function Name as a Variable  
val functionNameAsVariable = (i:Int) => { i + 10}
```

```
simpleList.map(functionNameAsVariable).foreach { x => println(x) }
```

# Function as a variable

```
val fn = (x : Int) => x * x
```

// is expanded to

```
val fnAliter = new Function1[Int, Int] {  
  def apply(x : Int) : Int = {  
    x * x  
  }  
}
```

```
fn(3)  
fn.apply(3)
```

```
fnAliter(3)  
fnAliter.apply(3)
```

# Partially Applied Function

```
val PRIMARY_SCHEMA = "PRIMARY"
val SECONDARY_SCHEMA = "SECONDARY"

def persist(schema : String) (query : String) = {
    println(s"Executing $query in $schema")
}

// Partially Applied Function
def executePAF() {
    val persistPrimary = persist (PRIMARY_SCHEMA) (_:String)
    val persistSecondary = persist (SECONDARY_SCHEMA) (_:String)

    persistPrimary("Select * from RegressionReckoner")
    persistSecondary("Select * from RegressionReckoner")
}
```

# Currying

```
val PRIMARY_SCHEMA = "PRIMARY"
  val SECONDARY_SCHEMA = "SECONDARY"

  def curriedPersist(schema : String) = (query : String) => {
    println(s"Executing $query in $schema")
  }
```

```
// Currying Function
def executeCurrying() {
  val persistPrimary = curriedPersist (PRIMARY_SCHEMA)
  val persistSecondary = curriedPersist (SECONDARY_SCHEMA)

  persistPrimary("Select * from RegressionReckoner")
  persistSecondary("Select * from RegressionReckoner")
}
```

# Currying

```
/** A curried function */
def concatTwoStrings(first:String) = (second:String) => {
    first.concat(second)
}

var result_1 = concatTwoStrings("CHERA")("CHOLA")
var result_2 = concatTwoStrings("PANDYA")
var result_3 = result_2("PALLAVA")

println(result_1)
println(result_2)
println(result_3)
```

A curried function is for a tuple input  
PAF (Partially Applied Function)  
is for multivariate inputs.

Output:

```
CHERACHOLA
<function1>
PANDYAPALLAVA
```

# Type Specifications

# Type Aliases

Helps in elegant naming

Ease of functional programming and coding.

# Type Aliases

```
object TypeAlias {  
  
    type MY_INTEGER = Int  
    type myString = String  
    type RT = Runtime  
    type EMPBEAN = EmployeeBean  
  
}
```

```
object TypeSpecification {  
  
    def main(args : Array[String]) {  
  
        val ainteger : TypeAlias.MY_INTEGER = 5  
        val aString : TypeAlias.myString = "Zeus"  
  
        val empBean : TypeAlias.EMPBEAN = new EmployeeBean  
  
    }  
}
```

# Type Aliases

```
type UserInfo = Tuple2[Int, String]
```

```
val userInfo = new UserInfo(1,  
"Chinnasamy")  
println(userInfo._2)
```

TypeSpecification.scala

# Type Classes

# Type Classes

Type classes provide ad-hoc polymorphism, which states that one can create polymorphism that can be applied to arguments of different types.

Type classes help us by which we can create common behavior for classes without resorting to tradition inheritance (extends) polymorphism.

It's nothing but a programming pattern with the help of implicits.

*Ad Hoc polymorphism refers to functions that can be applied to argument of different types.*

# Type Classes

- [0] Define the type classes
- [1] Define Behavior
- [2] Provide Implementation
- [3] Call the type classes

# Type Classes

[0] Define the type classes

```
final case class REDIS_NOSQL(data : String)
```

```
final case class VOLDEMORT_NOSQL(data : String)
```

```
final case class COUCHDB_NOSQL(datas: String)
```

# Type Classes

[1] Define Behavior

```
trait BigDataWriter[A] {  
    def writeToDB(input : A)  
}
```

# Type Classes

## [2] Provide Implementation

```
object BigDataWriterUtility {

    implicit object MYSQLWriter extends BigDataWriter[REDIS_NOSQL] {
        override def writeToDB(input: REDIS_NOSQL) = {
            println(s"REDIS IS WRITTEN TO MYSQL $input")
        }
    }

    implicit class BigDataToDBConverter[A](input:A) {

        def writeToMYSQL(implicit writer : BigDataWriter[A]) = {
            writer.writeToDB(input)
        }
    }
}
```

# Type Classes

[3] Call the type classes

```
object TypeClassesExplained extends App {  
  
    import BigDataWriterUtility._  
  
    REDIS_NOSQL("key-value").writeToMySQL  
  
}
```

# Scala Generics

# Scala Generics

Covariant

[+T] - Along the inheritance

Contravariant

[-T] - Against the inheritance

Invariant

[T]

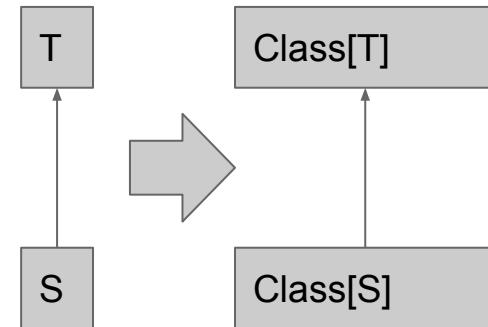
CovariantContravariantInvariant.scala

chinnasamyad@gmail.com

# Rules of Inheritance - Covariant

Covariant - [+A]

If S extends T, then class[S] also extends class[T]



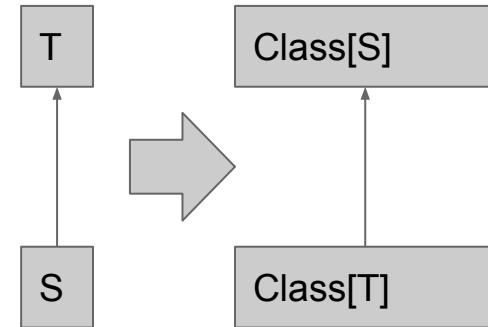
CovariantContravariantInvariant.scala

chinnasamyad@gmail.com

# Rules of Inheritance - Contravariant

Covariant - [-A]

If S extends T, then class[T] also extends class[S]



CovariantContravariantInvariant.scala

chinnasamyad@gmail.com

# Covariance

```
class  
CovariantFamily[+  
A] (val data : A)
```

```
tellMeWhomAmI(new CovariantFamily[Child] (new Child))
```

```
//invokeContravariant(new ContraVariantFamily[Parent] (new  
Parent))
```

```
def tellMeWhomAmI(family : CovariantFamily[Parent]) {  
    println(family.data.parentName)  
}
```

CovariantContravariantInvariant.scala

chinnasamyad@gmail.com

## Variants are of three types in Scala

- 1. Covariant +T
- 2. Contravariant -T
- 3. Invariant T

Covariant - Only subtypes of the defined class

Contravariant - Only supertypes of the defined class

Invariant - Only the particular class

By default scala classes are invariants in nature.

# Variants

```
// Covariant - Only subtypes of WildAnimals are fine
class Zoo1[+T] {

}

// Contravariant - Only supertypes of WildAnimals are fine
class Zoo2[-T] {

}

// Invariant - Only WildAnimals are fine. By default are scala classes
class Zoo3[T] {

}

class Animals

class WildAnimals extends Animals

class Lion extends WildAnimals
```

```
object VariantsOfClass {

    // Only subtypes of WildAnimals are fine
    def covariant(zoo : Zoo1[WildAnimals]) = {

    }

    // Only supertypes of WildAnimals are fine
    def contravariant(zoo : Zoo2[WildAnimals]) = {

    }

    // Only WildAnimals are fine. By default are scala classes are Invariant
    def invariant(zoo : Zoo3[WildAnimals]) = {

    }
}
```

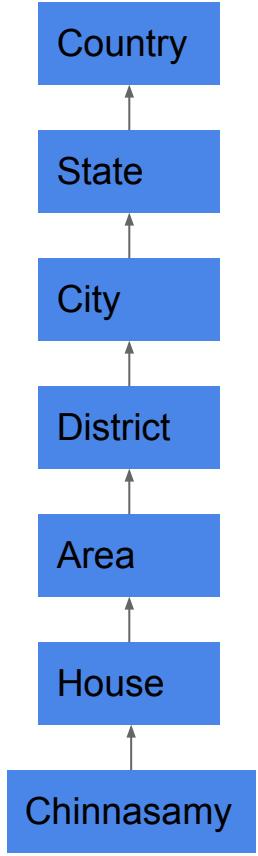
# Variants

```
def main(args : Array[String]) = {  
  
    covariant(new Zoo1[WildAnimals])  
    covariant(new Zoo1[Lion])  
    //covariant(new Zoo1[Animals]) // Compilation fails  
  
    contravariant(new Zoo2[WildAnimals])  
    //contravariant(new Zoo2[Lion]) // Compilation fails  
    contravariant(new Zoo2[Animals])  
  
    invariant(new Zoo3[WildAnimals])  
    //invariant(new Zoo3[Lion]) // Compilation fails  
    //invariant(new Zoo3[Animals]) // Compilation fails  
  
}
```

# Bounds

Upper Bound	$S <: T$	$S$ is a subtype of $T$
Lower Bound	$S >: T$	$S$ is a supertype of $T$

$S >: T <: U$



```
class Country {}  
class State extends Country {}  
class City extends State {}  
class District extends City {}  
class Area extends District {}  
class House extends Area {}  
class Chinnasamy extends House {}
```

# Bounds

```
class World {  
  
    // Upper Bound  
    def isMyDistrict[S <: District](myDistrict:S) : S = {  
        println(myDistrict.getClass)  
        myDistrict  
    }  
  
    // Lower Bound  
    def giveMeAllSupers[S >: District](myDistrict:S) : S = {  
        println(myDistrict.getClass)  
        myDistrict  
    }  
  
    def giveIntermediate[S >: Area <: City](myLiving:S) : S = {  
        println(myLiving.getClass)  
        myLiving  
    }  
  
    class Country {}  
    class State extends Country {}  
    class City extends State {}  
    class District extends City {}  
    class Area extends District {}  
    class House extends Area {}  
    class Chinnasamy extends House {}
```

chinnasamyad@gmail.com

```
val world = new World  
// world.isMyDistrict(new Area)  
// world.isMyDistrict(new House)  
// world.isMyDistrict(new Chinnasamy)  
//world.isMyDistrict(new City) // Fails  
  
// world.giveMeAllSupers(new City())  
// world.giveMeAllSupers(new Country())  
  
world.giveIntermediate(new District)  
// world.giveIntermediate(new State) // fails  
world.giveIntermediate(new House)
```

# Variance

Variance Check =>

Function are contravariant in their argument type and covariant in their result type.

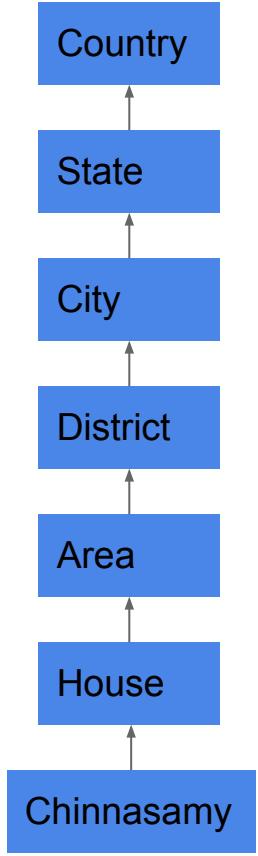
- Covariant type parameters can appear in method results.
- Contravariant type parameters can only appear in method parameters.
- Invariant type parameters can appear anywhere

Variance checks prevents mutable operations in covariant classes.

# Bounds

Upper Bound	$S <: T$	$S$ is a subtype of $T$
Lower Bound	$S >: T$	$S$ is a supertype of $T$

$S >: T <: U$



```
class Country {}  
class State extends Country {}  
class City extends State {}  
class District extends City {}  
class Area extends District {}  
class House extends Area {}  
class Chinnasamy extends House {}
```

# Bounds

```
class World {  
  
    // Upper Bound  
    def isMyDistrict[S <: District](myDistrict:S) : S = {  
        println(myDistrict.getClass)  
        myDistrict  
    }  
  
    // Lower Bound  
    def giveMeAllSupers[S >: District](myDistrict:S) : S = {  
        println(myDistrict.getClass)  
        myDistrict  
    }  
  
    def giveIntermediate[S >: Area <: City](myLiving:S) : S = {  
        println(myLiving.getClass)  
        myLiving  
    }  
  
    class Country {}  
    class State extends Country {}  
    class City extends State {}  
    class District extends City {}  
    class Area extends District {}  
    class House extends Area {}  
    class Chinnasamy extends House {}  
  
    val world = new World  
    // world.isMyDistrict(new Area)  
    // world.isMyDistrict(new House)  
    // world.isMyDistrict(new Chinnasamy)  
    //world.isMyDistrict(new City) // Fails  
  
    // world.giveMeAllSupers(new City())  
    // world.giveMeAllSupers(new Country())  
  
    world.giveIntermediate(new District)  
    // world.giveIntermediate(new State) // fails  
    world.giveIntermediate(new House)
```

# Variance

Variance Check =>

Function are contravariant in their argument type and covariant in their result type.

- Covariant type parameters can appear in method results.
- Contravariant type parameters can only appear in method parameters.
- Invariant type parameters can appear anywhere

Variance checks prevents mutable operations in covariant classes.

# Annotations Usage

```
object AnnotationsUsage {  
  
  def main(args : Array[String]) : Unit = {  
  
    @volatile  
    val volatileKeyword = "A_VOLATILE_DECLARATION"  
  
    @throws(classOf[MeshContractDataFlowException])  
    val dataFlow = new DataFlow  
  
    dataFlow.data("")  
  
  }  
}  
  
class DataFlow {  
  
  def data(message : String) = {  
    throw MeshContractDataFlowException("Dataflow exception")  
  }  
}  
  
case class MeshContractDataFlowException(message:String) extends Exception(message)
```

# Bounded Types

Upper Bounded Type

Lower Bounded Type

# Embrace functional programming

Instead of using the for loop use the foreach

```
def printArray(namesArray : Array[String]) : Unit = {  
    namesArray.foreach { x => println(x) }  
}
```

```
val names = Array("ALAN", "MATHISON", "TURING")  
clazz.printArray(names)
```

# Embrace functional programming - compose , andThen Helpers

[\*] Each function has a compose, andThen methods to help in functional programming.

[\*] compose, andThen build bigger functions from smaller functions.

[\*] The only difference between compose and andThen is that the order of execution is reversed.

Refer ThenAndComposePattern.scala

chinnasamyad@gmail.com

# Embrace functional programming - compose , andThen Helpers

```
def convertStringToInt(input : String) : Int = {  
    Integer.parseInt(input)  
}  
  
def convertIntToFloat(inputInteger : Int) : Float = {  
    inputInteger.toFloat  
}  
  
def convertFloatToString(inputFloat : Float) : String = {  
    inputFloat.toString  
}  
  
val convertStringToFloatUsingAndThen = convertStringToInt _ andThen convertIntToFloat _  
val convertStringToFloatUsingCompose = convertIntToFloat _ compose convertStringToInt _  
val mixOfComposeandThen = convertIntToFloat _ compose convertStringToInt _ andThen  
    convertFloatToString _  
  
println(convertStringToFloatUsingCompose("20"))  
println(mixOfComposeandThen("20"))
```

Refer ThenAndComposePattern.scala

chinnasamyad@gmail.com

# Embrace functional programming - compose , andThen Helpers

```
def getCSVSplitted(input : String) : Array[String] = {  
    input.split(",")  
}
```

```
def returnCSVFormattedString(fileName : String) : String = {  
    val file = Source.fromFile(fileName)  
    file.getLines().mkString  
}
```

```
def analyzeOntheData(inputArray : Array[String]) : Int = {  
    inputArray(2).length  
}
```

```
val getMeData = getCSVSplitted _ compose returnCSVFormattedString _ andThen analyzeOntheData _  
printIn(getMeData("/home/dharshekthvel/ac/code/scalatrainingintellij/data/titanic3.csv"))
```

Refer ThenAndComposePattern.scala

chinnasamyad@gmail.com

# Embrace functional programming - compose , andThen Helpers

```
def convertToUpperCase(input: String) : String = input.toUpperCase  
def returnLength(input : String ) : String = {  
    return input.length.toString  
}
```

val inputList = List("Kudu", "Apache", "Zeppelin")  
/\*  
compose, andThen build bigger functions from smaller functions.  
The only difference between compose and andThen is that the order  
of execution is reversed.  
\*/  
inputList.map(convertToUpperCase\_andThen returnLength\_)  
.foreach(eachElement => println(eachElement))

# PartialFunction

A PartialFunction is a trait that helps in concise code with the help of isDefinedAt() method.

A PartialFunction defines a isDefinedAt() method which helps the caller to know whether the function can handle that element.

Remember: A PartialFunction is no way related to Partially Applied Function (Currying)

# PartialFunction

Implementing a PartialFunction...

```
class One extends PartialFunction[Int, String] {  
    override def isDefinedAt(x: Int): Boolean = if (x==1) true else false  
  
    override def apply(v1: Int): String = "ONE"
```

```
}
```

```
val one = new One
```

```
println(one(1))
```

ElucidatingPartialFunction.scala

# PartialFunction

Implementing a PartialFunction...

```
val two = new PartialFunction[Int, String] {  
    override def isDefinedAt(x: Int): Boolean = if (x==2) true else false  
  
    override def apply(v1: Int): String = "TWO"  
}
```

*println(two(2))*

ElucidatingPartialFunction.scala

# PartialFunction

Implementing a PartialFunction...

```
val three: PartialFunction[Int, String] = { case 3 => "THREE" }
```

```
println(three(3))
```

ElucidatingPartialFunction.scala

# PartialFunction

Implementing a PartialFunction...

```
val mypartialfunction = one orElse two orElse three orElse four
```

```
if (mypartialfunction.isDefinedAt 3)  
  println("Final Output : " + mypartialfunction(3))
```

ElucidatingPartialFunction.scala

# Transformations, Grouping, Filtering Methods

Scala enablers to help in better programming on collections

Transformation, Filtering, Grouping methods are helper methods in collections provided in the scala API to enable ease in programming.

Apart from these there are also mathematical and data structure helper methods to facilitate ease of programming.

# Filter

```
val countries = List("Brazil",
    "Russia",
    "Bahamas",
    "Bangladesh",
    "Cook Islands",
    "UK",
    "Norway",
    "Sweden",
    "Germany",
    "France")
```

```
countries.filter { x => x.startsWith("B") }.foreach { x => println(x) }
```

```
countries.filter { (x:String) => x.startsWith("B") }.foreach { x => println(x) }
```

```
countries.filter ( _.startsWith("B") ).foreach { x => println(x) }
```

# Filter

```
case class SupportVectorMachineBean (var name : String,  
                                   var highSupportVector : Float,  
                                   var lowerSupportVector : Float)
```

## Exercise

```
val svmList =  
    List(  
        SupportVectorMachineBean("AverageCallHandlingTime", 12f, 1f),  
        SupportVectorMachineBean("SupervisorMetric", 15f, 7f),  
        SupportVectorMachineBean("GYRScore", 2f, 1f),  
        SupportVectorMachineBean("CallMetricRatio", 9f, 4f))
```

# Filter

```
svmList.filter { svmBean => svmBean.lowerSupportVector > 2f }  
  .foreach { vector => println(vector.name) }
```

```
    svmList.filter (_ .lowerSupportVector > 2f).foreach( vector =>  
      println(vector.name))
```

# Head and Tail Filter

head returns the first element. Tail returns all elements except the first element.

```
val countries = List("Brazil",
                     "Austria",
                     "Bahamas",
                     "Bangladesh",
                     "Cook Islands")

println("Tail Elements")
countries.tail.foreach { x => println(x) }

println("Head Element")
println(countries.head)
```

# Take and Drop Filter

take(n) returns the first n elements.

drop(n) returns all except the first n elements.

```
val countries = List("Brazil",
    "Austria",
    "Bahamas",
    "Bangladesh",
    "Cook Islands")
```

```
println("Take Elements")
countries.take(3).foreach { x => println(x) }
```

```
println("Drop Elements")
countries.drop(3).foreach { x => println(x) }
```

# find Filter

```
val countries = List("Brazil",  
    "Russia",  
    "Bahamas",  
    "Bangladesh",  
    "Cook Islands",  
    "UK",  
    "Norway",  
    "Sweden",  
    "Germany",  
    "France")
```

Returns the Some of the Option class.

Returns the first occurrence of the search satisfying the predicate.

```
countries.find { x => x.startsWith("Ba") }.foreach { x => println(x) }
```

# Map Transformation

```
val countries = List("Brazil",
    "Austria",
    "Bahamas",
    "Bangladesh",
    "Cook Islands")
```

```
countries.map { country => country.concat("_Country") }
.foreach { x => println(x) }
```

# Map Transformation

```
val countries = List("Brazil",  
                     "Austria",  
                     "Bahamas",  
                     "Bangladesh",  
                     "Cook Islands")  
  
case class CountryNameBean(var name: String)  
  
val countryList = countries  
    .map(country => CountryNameBean(country)).toList  
  
//countryList.foreach { x => println(x.name) }
```

Transformations.scala

chinnasamyad@gmail.com

# Map Transformation

Exercise

```
val svmList =  
  List(  
    SupportVectorMachineBean("AverageCallHandlingTime", 12f, 1f),  
    SupportVectorMachineBean("SupervisorMetric", 15f, 7f),  
    SupportVectorMachineBean("GYRScore", 2f, 1f),  
    SupportVectorMachineBean("CallMetricRatio", 9f, 4f))
```

```
case class SupportVectorMachineBean (var name : String,  
                                    var highSupportVector : Float,  
                                    var lowerSupportVector : Float)
```

```
case class SVMNameBean(var name : String)
```

Transformations.scala

chinnasamyad@gmail.com

# Map Transformation

```
svmList.filter(_.lowerSupportVector > 2f).map { x => SVMNameBean(x.name)}.toList
```

# Map Transformation

```
case class CountryNameCodeBean(var name: String, var code: String)
```

```
val countryNameCodeList = countryList
    .map(x => new CountryCodeBeanTransformation().mapToCountryCode(x)).toList

countryNameCodeList.foreach { x => println(x.name) }
```

```
class CountryCodeBeanTransformation {

    def mapToCountryCode(countryNameBean : CountryNameBean) : CountryNameCodeBean = {

        if (countryNameBean.name == "Brazil") CountryNameCodeBean(countryNameBean.name, "BR")
        else if (countryNameBean.name == "Austria") CountryNameCodeBean(countryNameBean.name, "AU")
        else CountryNameCodeBean(countryNameBean.name, "_NO_CODE_")
    }
}
```

Transformations.scala

# count() Transformation

## (Mathematical Informational Transformation)

```
count(<<predicate>>)
```

count returns the count of elements satisfying the predicate

```
val countries = List("Brazil",  
                     "Austria",  
                     "Bahamas",  
                     "Bangladesh",  
                     "Cook Islands")
```

```
val countOfCountries = countries.count(element => element.startsWith("B"))
```

# collect Transformation

**The collect() transformation transforms the given data as per the logic been written inside the partial function passed to the collect.**

# collect Transformation

```
val partialFunction : PartialFunction[String, String] =  
{  
    case eachElement: String =>  
        eachElement.concat(" - CHOLA")  
}
```

```
val list = List ("CHERA", "PANDYA", "PALLAVA")
```

```
list .collect(partialFunction)  
    .foreach(eachElement => println(eachElement));
```

# Zip Transformation

Zip Transformation does a one to one combination and produces another collection

```
val country = List("Austria", "Bahamas", "Cook Islands")
val code = List("AT", "BS", "CK")
```

```
val zippedList = code.zip(country)
zippedList.foreach(x => println(x))
```

(AT,Austria)  
(BS,Bahamas)  
(CK,Cook Islands)

# Zip Transformation With Index

Zip with Index zip's the list with the indices starting from zero.

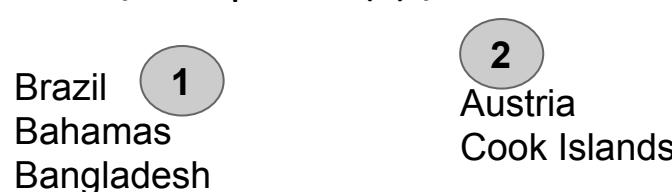
```
val country = List("Austria", "Bahamas", "Cook Islands")
val zippedList = country.zipWithIndex
zippedList.foreach(x => println(x))
```

(Austria,0)  
(Bahamas,1)  
(Cook Islands,2)

# Partition

```
val countries = List("Brazil",  
                     "Austria",  
                     "Bahamas",  
                     "Bangladesh",  
                     "Cook Islands")
```

```
val partitionedList = countries.partition { country => country.startsWith("B") }  
partitionedList._1.foreach { x => println(x) }  
partitionedList._2.foreach { x => println(x) }
```



# reduce Transformation

reduce is a transformation that executes the code written inside the function on the data set.

Basically a reduction on the data set.

```
val listOfNumbers = List(1,2,3,4,5)
```

```
val result = listOfNumbers.reduceLeft(_ + _)
```

```
println(result)
```

# reduce Transformation

```
val stringList = List("A", "String", "in", "a", "list")
```

```
val resultList = stringList.reduceLeft((x,y) => x.concat(" ").concat(y))
```

```
println(resultList)
```

# fold, foldLeft, foldRight

In a nutshell, fold takes data in one format and gives you in another based upon the function logic.

fold takes a seed value as one extra parameter.

reduce transformation acts on the data set in the same way as fold, but reduce doesn't have a seed value as that of fold.

reduce transformation uses the first element as the seed value.

# foldLeft

foldLeft operation to sum the numbers of the list.

```
val listOfNumbers = List(1,2,3,4,5)
```

```
val result = listOfNumbers.foldLeft(0)(_ + _)
```

```
println(result)
```

# foldLeft

A foldLeft operation to concatenate elements in a list

```
val stringList = List("A", "String", "in", "a", "list")
```

```
val resultList = stringList.foldLeft("The final String is : ")((x,y) => x.concat(" ")).concat(y))
```

```
println(resultList)
```

# foldLeft

foldLeft operation to sum the numbers from 1 to 1000

```
val rangeOfNumbers = 1 to 1000
```

```
val rangeResult = rangeOfNumbers.foldLeft(0)((x,y) => x+y)
```

```
println(rangeResult)
```

# Parallel Sequence Transformation

**par** method returns a parallel sequence on the collection

```
val countries = List(  
    "Brazil",  
    "Russia",  
    "Bahamas",  
    "Bangladesh",  
    "Cook Islands")  
  
countries.par.foreach { z => println(z) }
```

# Flatten Transformation

```
val countries = List("Brazil",  
                     "Russia",  
                     "Bahamas",  
                     "Bangladesh",  
                     "Cook Islands")
```

```
val europeanCountries = List("UK",  
                             "Norway",  
                             "Sweden",  
                             "Germany",  
                             "France")
```

```
val allCountries = List(countries,  
                        europeanCountries)
```

```
val flattenedListOfCountries =  
  allCountries.flatten  
  flattenedListOfCountries.foreach { x =>  
    println(x) }
```

# Slice Transformation

```
val countries = List("Brazil",  
  "Russia",  
  "Bahamas",  
  "Bangladesh",  
  "Cook Islands")
```

```
countries.slice(1,3).foreach(x => println(x))
```

Russia  
Bahamas

# SplitAt Transformation

```
val countries = List("Brazil",
  "Russia",
  "Bahamas",
  "Bangladesh",
  "Cook Islands")

countries.splitAt(2)._1.foreach(x => println(x))
```

Since we are printing `_1` of the tuple, we get the first split.

```
Brazil
Russia
```

# Reverse Transformation

Returns the elements in the reversed order

```
val countries = List("Brazil",
    "Russia",
    "Bahamas",
    "Bangladesh",
    "Cook Islands",
    "UK",
    "Norway",
    "Sweden",
    "Germany",
    "France")
```

```
countries.reverse.foreach { x => println(x) }
```

# implicit

The implicit feature in the code allows the scala compiler to adjust code using a lookup mechanism.

implicit can be thought of as a global initializer or identifier.

implicit can either be defined on the variable, method, class or an object.

# implicit

An implicit on the variable

```
implicit val floatValue = 167f
```

```
def getMeRoundedNumberOfFloat(implicit anyFloatValue : Float) {  
    println(anyFloatValue)  
}
```

```
def main(args: Array[String]) {  
    getMeRoundedNumberOfFloat  
}
```

# Implicit methods

An implicit on the method level

```
implicit val defaultHoursOfWork = 8
```

```
def work(nameOfPerson: String) (implicit durationOfWork : Int) = {  
  println("Name of the person and the hours he worked - " + nameOfPerson + " - " + durationOfWork)  
}
```

```
work("Che")  
work("Che")(10)
```

Name of the person and the hours he worked - Che - 8  
Name of the person and the hours he worked - Che - 10

# Implicit - Another Usage

An implicit on the method level

```
def printAge(name : String) {  
    println(s"printing age of $name")  
}
```

```
implicit def printName(age : Int) : String = {  
    println("obtaining name")  
    "CHINNASAMY"  
}
```

```
/* Two implicit functions cannot be possible  
implicit def getName(age: Int) : String = {  
    ""  
} */
```

```
def main(args: Array[String]) {  
    printAge(12)  
}
```

# implicit

## implicit classes as extension methods

```
import com.dmac.basic.ImplicitClassesFeature._  
val out = "NLP_".letMoonBeAddedToString()  
println(out)
```

The String is enriched with the API  
letMoonBeAddedToString().

```
object ImplicitClassesFeature {  
  
    implicit class AddFunctionalityToString(val input: String) {  
  
        def letMoonBeAddedToString(): String = input.concat("MOON")  
    }  
  
}
```

Implicit classes are usually put inside a **object**. So that **import** can be easy.  
Implicit classes can have only one constructor argument  
Implicit classes cannot have two classes with the same input type variable.  
Implicit classes must be defined inside a trait (or) class (or) object.  
Ofcourse, Implicit classes should be imported in the current scope.

# implicit

## implicit classes as extension methods

```
val money = 100;  
println("Printing the implicit" + 100.words)
```

The Integer money is enriched with the API words().

```
implicit class MoneyToWords(val n: Int) {  
  def words(): String = {  
    if (n == 100) "One Hundred"  
    else if (n == 200) "Two Hundred"  
    else "No Money"  
  }  
}
```

Refer ImplicitFeature.scala

# implicit

## implicit on object

```
import org.apache.spark.{SparkContext, SparkConf, AccumulatorParam}

implicit object AuthSuccessAccumulator extends AccumulatorParam[AuthSuccess] {

  override def addInPlace(r1: AuthSuccess, r2: AuthSuccess): AuthSuccess
    = AuthSuccess(r1.uid + " - " + r2.uid)

  override def zero(initialValue: AuthSuccess): AuthSuccess = AuthSuccess("")

}
```

# File Operations

scala.io

# Reading a File

Simple one liner for reading each line in of a file.

```
import scala.io.Source

def readingAFile(fileName : String) : Unit = {
    Source
        .fromFile(fileName)
        .foreach { print }
}
```

# File Read

```
def readFile(fileName: String) : Unit = {  
    val file = io.Source.fromFile(fileName)  
  
    for(line <- file.getLines())  
        println(line)  
  
    file.close  
}
```

FileOperations.scala

# CSV File Read

```
val csvFile = io.Source.fromFile(fileName)

for(line <- csvFile.getLines()) {
    val columns = line.split(",")
    println(columns(0) + " " + columns(2))
}

csvFile.close
```

FileOperations.scala

# CSV File Read

```
case class AuthBean(val authCode:String, val auaCode : String)

val list = new ListBuffer[AuthBean]

for(line <- csvFile.getLines()) {
    val lines = line.split(",")
    val ab = AuthBean(lines(0), lines(2))
    list += ab
}

val authBeanList = list.toList
authBeanList.foreach { authBean => println(authBean.authCode) }

csvFile.close
```

FileOperations.scala

# CSV File Read

```
case class AuthBean(val authCode:String, val auaCode : String)

val list = new ListBuffer[AuthBean]

// Functional way
csvFile.getLines().foreach { line =>
    val column = line.split(",")
    val ab = AuthBean(column(0), column(2))
    list += ab
}

val authBeanList = list.toList
authBeanList.foreach { authBean => println(authBean.authCode) }

csvFile.close
```

FileOperations.scala

# Executing External Command

```
import sys.process._

val LS_COMMAND = "ls -l".!!

println(LS_COMMAND)
```

# Executing External Command

!

!!

lines() - Returns a Stream of string

# sbt - Scala Build Tool

\$ sbt package

Will create a jar without any dependencies.

\$ sbt assembly

Will create a jar with all dependencies.

\$ sbt clean

Deletes the target directory

\$ sbt publish

Publishes the artifacts

# sbt - run

if there are  
more than  
one executable file,  
sbt would  
ask for the prompt.

```
[info] Set current project to sparkscala9 (in build file:/D:/ac/sparkscala9  
[warn] The global sbt directory is now versioned and is located at C:\Users  
.sbt\0.13.  
[warn] You are seeing this warning because there is global configuration  
C:\Users\HCL\.sbt but not in C:\Users\HCL\.sbt\0.13.  
[warn] The global sbt directory may be changed via the sbt.global.base sys-  
property.  
[info] Compiling 25 Scala sources to D:/ac/sparkscala9/target/scala-2.11/cl  
[warn] there were two deprecation warnings; re-run with --deprecation for det  
[warn] one warning found  
[warn] Multiple main classes detected. Run 'show discoveredMainClasses' to  
the list  
  
Multiple main classes detected, select one to run:  
[1] CustomRDD  
[2] DataFrameManipulator  
[3] ElucidateDataset  
[4] SparkGlom  
[5] SparkMain  
[6] StructTypeDemo  
[7] com.dmac.scala.CustomPartitioner  
[8] com.dmac.scala.MapPartitions  
[9] com.dmac.scala.SparkElasticSearch  
[10] com.dmac.scala.WholeTextFilesSpark  
[11] com.dmac.scala.basic.ASimpleMain  
[12] com.dmac.scala.basic.CaseClassExample  
[13] com.dmac.scala.basic.CovariantContravariantInvariant  
[14] com.dmac.scala.basic.CustomAccumulator  
[15] com.dmac.scala.basic.IteratingATuple  
[16] com.dmac.scala.basic.JDBCConnector  
[17] com.dmac.scala.basic.ListToBean  
[18] com.dmac.scala.basic.PartialFunctionMediocre  
[19] com.dmac.scala.basic.RangesOnCharacters  
[20] com.dmac.scala.basic.SomeOtherFile  
[21] com.dmac.scala.basic.SparkReadFile  
[22] com.dmac.scala.basic.SparkTransformations  
[23] com.dmac.scala.basic.WordCountSpark  
[24] com.dmac.scala.main.FirstScalaProgram  
  
Enter number: 1
```

# Metadata Information

*Use the classOf to get information about the class*

```
val clazz = classOf[ETMS]
```

```
println(clazz.getName)  
println(clazz.getInterfaces)  
println(clazz.getConstructors)
```

```
class ETMS(parameter : String) extends AbstractTMS{  
  
    val appName = "Enterprise Trouble Management System";  
  
    def getTotalTickets() : Int = {  
        100  
    }  
}
```

Metadata.scala

# XML File Read

The load function loads the xml file.

```
import scala.xml._  
  
val file = XML.load("D:/ac/data/food.xml")
```

# XML File Read

```
<food>
    <name>Belgian Waffles</name>
    <price>$5.95</price>
    <description>Two of our famous Belgian Waffles with plenty of real maple syrup</description>
    <calories>650</calories>
</food>
<food>
    <name>Strawberry Belgian Waffles</name>
    <price>$7.95</price>
    <description>Light Belgian waffles covered with strawberries and whipped cream</description>
    <calories>900</calories>
</food>
```

```
val file = XML.load("D:/ac/data/food.xml")
val food = file \ "food" \ "name"
```

```
food.foreach { element => println(element.text) }
```

# Scala Best Practices

# Use the Option class where ever possible

Encapsulate each object inside the Option class

Null values are avoided and so NullPointerException is completely wiped off.

Either return Option or None.

Option basically returns the Some.

The Option class is a typical example of the sealed class.

# Use the Option class where ever possible

```
val optionClazz = new UsageOfOption
  val returnValue = optionClazz.divide(3.5, 9.0)

  println(returnValue.get)
  println(optionClazz.upperCaseLogger(null))

  val value = optionClazz.upperCaseLogger("logging otp cache failure").get
  //println(optionClazz.upperCaseLogger("logging otp cache failure"))
  println(value)
```

```
class UsageOfOption {

  def divide(value : Double, divisor: Double) : Option[Double] = {
    if (divisor == 0) None
    else
      Option(value/divisor)
  }

  def upperCaseLogger(inputText : String) : Option[String] = {
    if (inputText == null || inputText.isEmpty()) None
    else {
      val message = "APP_NAME LOG : ".concat(inputText)
      Option(message)
    }
  }
}
```

# Factory Pattern using match

```
val input = "QuickSearch"

val search = input match {

    case "QuickSearch" => new QuickSearch
    case "TimSearch" => new TimSearch

    case _ => new TimSearch
}

search.doSearchAlgorithm
```

```
trait Search {
    def input {
        //Get the Input
    }
    def doSearchAlgorithm
}

class QuickSearch extends Search {
    def doSearchAlgorithm {
        println("Quick Search Algorithm")
    }
}

class TimSearch extends Search {
    def doSearchAlgorithm {
        println("Tim Search Algorithm")
    }
}
```

Refer to [Matcher.scala](#)

# Design by contract

Use the require() to achieve Design by Contract.

```
def designByContract(member : String) {  
    require(!member.isEmpty(), "MethodClass : member should not be empty")  
}
```

ScalaMethods.scala

Refer to [Matcher.scala](#)

# Stuffs that you can do with scala

```
def printMeManyTimes(): String = {  
    return "*" * 100  
}
```

```
println(printMeManyTimes)
```

# Stuffs that you can do with scala

You can mark a function as inline.

```
@inline def divide(value : Double, divisor: Double) : Option[Double] = {  
    ....  
    ....  
    ....  
}  
  
Try it for yourself whether  
it really optimizes the  
compiler
```

Inline functions places the function code in the place where it is called.

# Tail Recursion

If the last action of a function consists of calling a function(which may be the same), one stack frame would be sufficient for both functions. Such calls are called tail-calls.

If a function calls itself as its last action, the functions stack frame can be reused. This is called tail-recursion.

# Tail Recursion

```
def factWithOutTailRecursion(n : Int) : Int = {  
    if (n==0) 1 else n * factWithOutTailRecursion(n-1)  
}
```

```
def factWithTailRecursion(n : Int) : Int = {  
    loop(1,n)  
}
```

```
def loop(acc: Int, n: Int) : Int = {  
    if (n==0) acc else loop(acc*n, n-1)  
}
```

```
println(factWithOutTailRecursion(5))  
println(factWithTailRecursion(5))
```

Premature optimization is the root of all evil.

# Tail Recursion

```
def factWithOutTailRecursion(n : Int) : Int = {  
    if (n==0) 1 else n * factWithOutTailRecursion(n-1)  
}
```

```
def factWithTailRecursion(n : Int) : Int = {  
    loop(1,n)  
}
```

```
def loop(acc: Int, n: Int) : Int = {  
    if (n==0) acc else loop(acc*n, n-1)  
}
```

Premature optimization is the root of all evil.

```
println(factWithOutTailRecursion(5))  
println(factWithTailRecursion(5))
```

# Interoperability with Java

```
import scala.collection.JavaConverters._
```

Use the above import and you can use all conversions possible.

The above import adds a numerous implicit methods to your existing collection.

```
topicPartition.asList
```

# Monad

A Monad is a parametric type with two operations

flatMap

Unit

It should qualify the laws of associativity, Left Unit and Right Unit.

List, Set, Option, Generator are all monads.

# Functor

A functor is a type class which implements a map method as below.

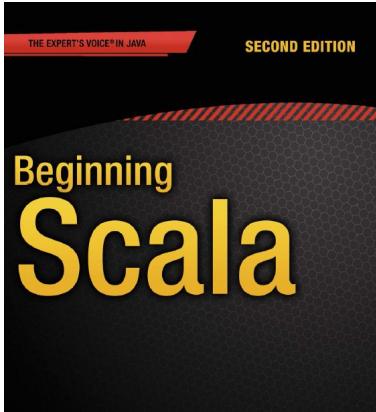
```
def map[A, B](fa: F[A])(f: A => B): F[B]
```

(fa : F[A]) – a data-type that take only one data-type e.g List, Future , Options

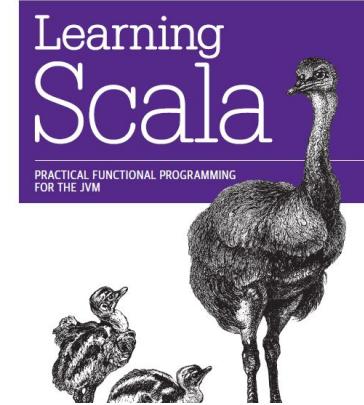
(f: A => B ) – a function that transform a type A to a type B e.g function that takes a string and return an Int.

F[B] – the final return type of the Functor's map function e.g if our "fa" variable is a List[String] and our "f" function is String => Int , then our map function return type will be a List[Int]

# References



O'REILLY®



# Concurrency



# Reactive Programming

<http://www.reactivemanifesto.org/>

# Reactive Programming

**Responsive** : System responds in a timely manner.

**Resilient** : System is fault-tolerant.

**Elastic** : System is responsive under varying Workload.

**Message-Driven** : System uses asynchronous mode of messaging

# Akka

The concept in akka was developed by Carl Hewitt, Peter Bishop and Richard Steiger in 1973.

The concept of actors is used in Erlang language and used by Ericsson to build massive systems after which the actor based programming model became popular.

Akka is heavily influenced from the Erlang way of programming with actors.

# Akka

An open-source project built by Light-Bend.

Akka is centered on actors. An actor based programming model.

By actors, actor way of programming,

One can isolate failure,  
provide non-blocking I/O  
and no shared mutable state for prone errors.

# Actor

An actor is an object that receives messages and takes action based on the logic

It processes the messages or forwards the messages to another actor.  
All actors execute concurrently.

To implement an Actor, extend the actor trait and implement the receive method.

All phenomenon, of creating threads, race condition, synchronization is all handled by akka internally.

# Actor

In short, Concurrency with threads are difficult, but with akka actors is pretty much easy.

An actor is a light-weight process:

Create, Send, Become and Supervise.

# Actor - Creation and Send Messages

Also called as Fire-and-Forget.

ask() : Send-and-Receive-Future

# Concurrent App - Actors

```
import akka.actor.ActorSystem
import akka.actor.Actor
import akka.actor.Props

val system = ActorSystem("ConcurrentApp")
val sender = system.actorOf(Props[Receiver], "receiver")
```

# Concurrent App - Actors

```
class Receiver extends Actor {  
    def receive = {  
        case "MESS" => println("message is received")  
  
        case Bean(name) => println(s"Got it $name")  
        case _ => println("_DEFAULT_")  
    }  
}
```

# Concurrent App - Actors

```
val sender = system.actorOf(Props[Receiver], "receiver")
```

```
sender ! "MESS"  
sender.tell("MESS", sender)
```

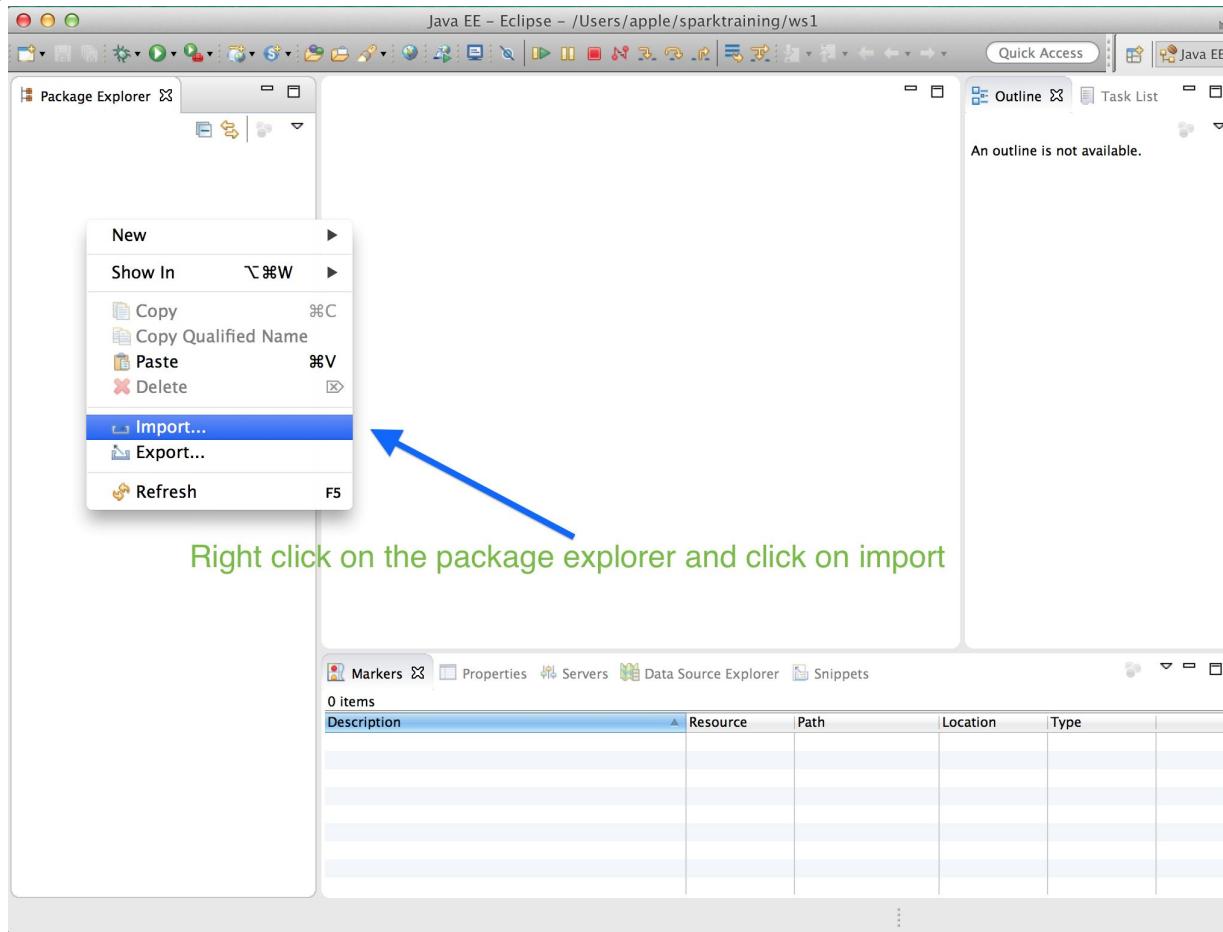
```
val messageBean = Bean("bean")  
sender.tell(messageBean, sender)
```

# Importing the code using Eclipse

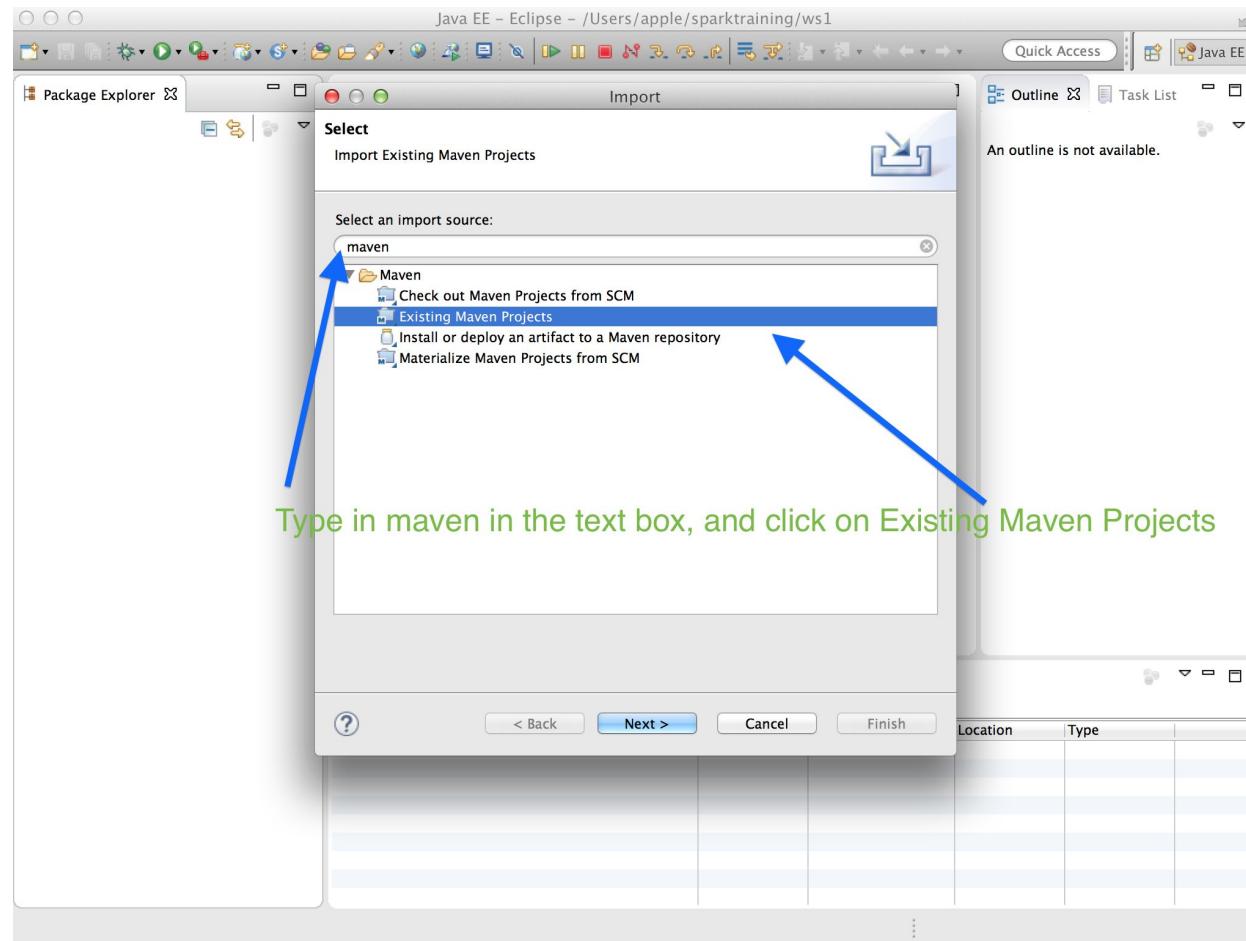
(People who are familiar, can skip this section)



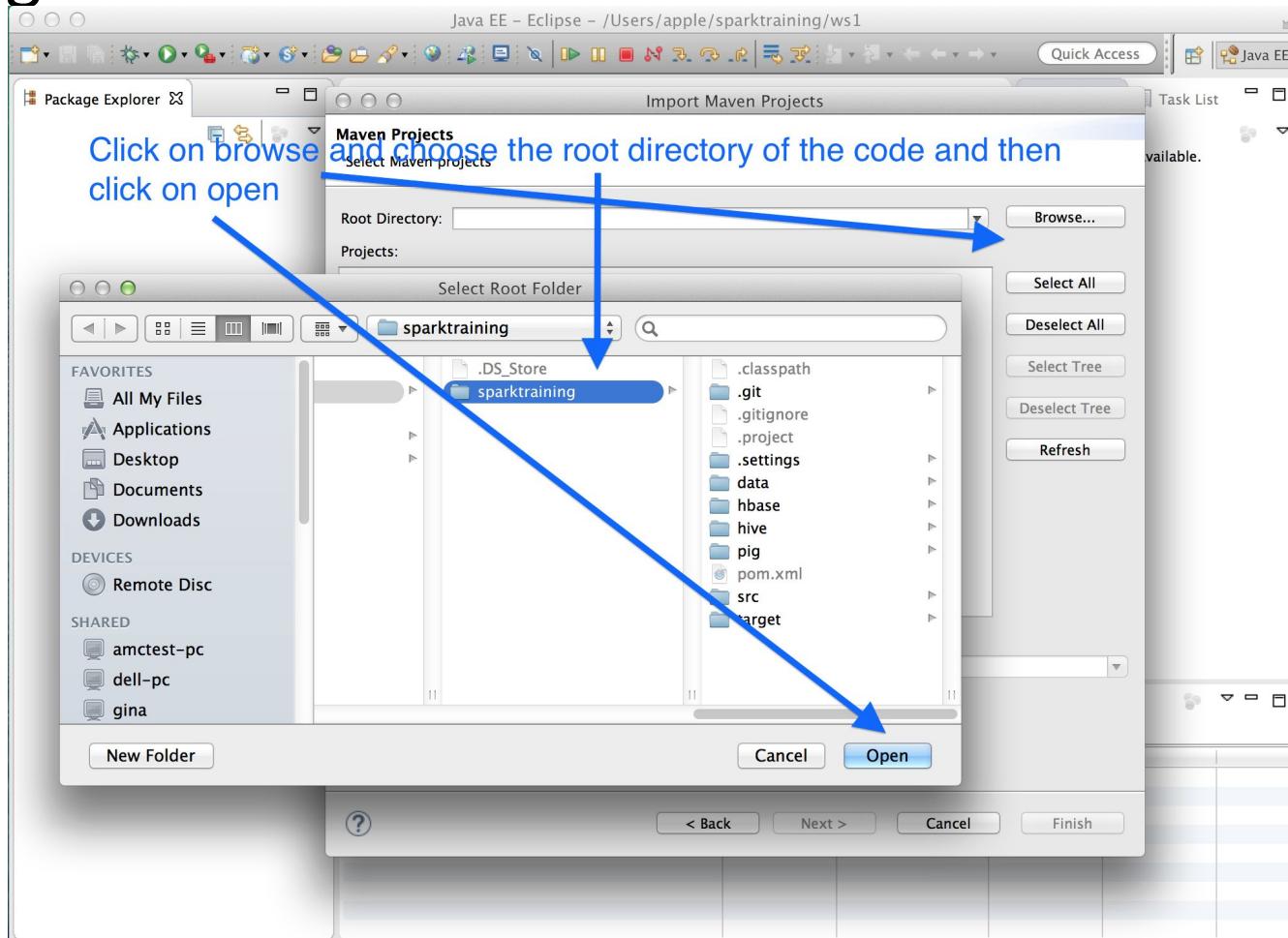
# Importing the code

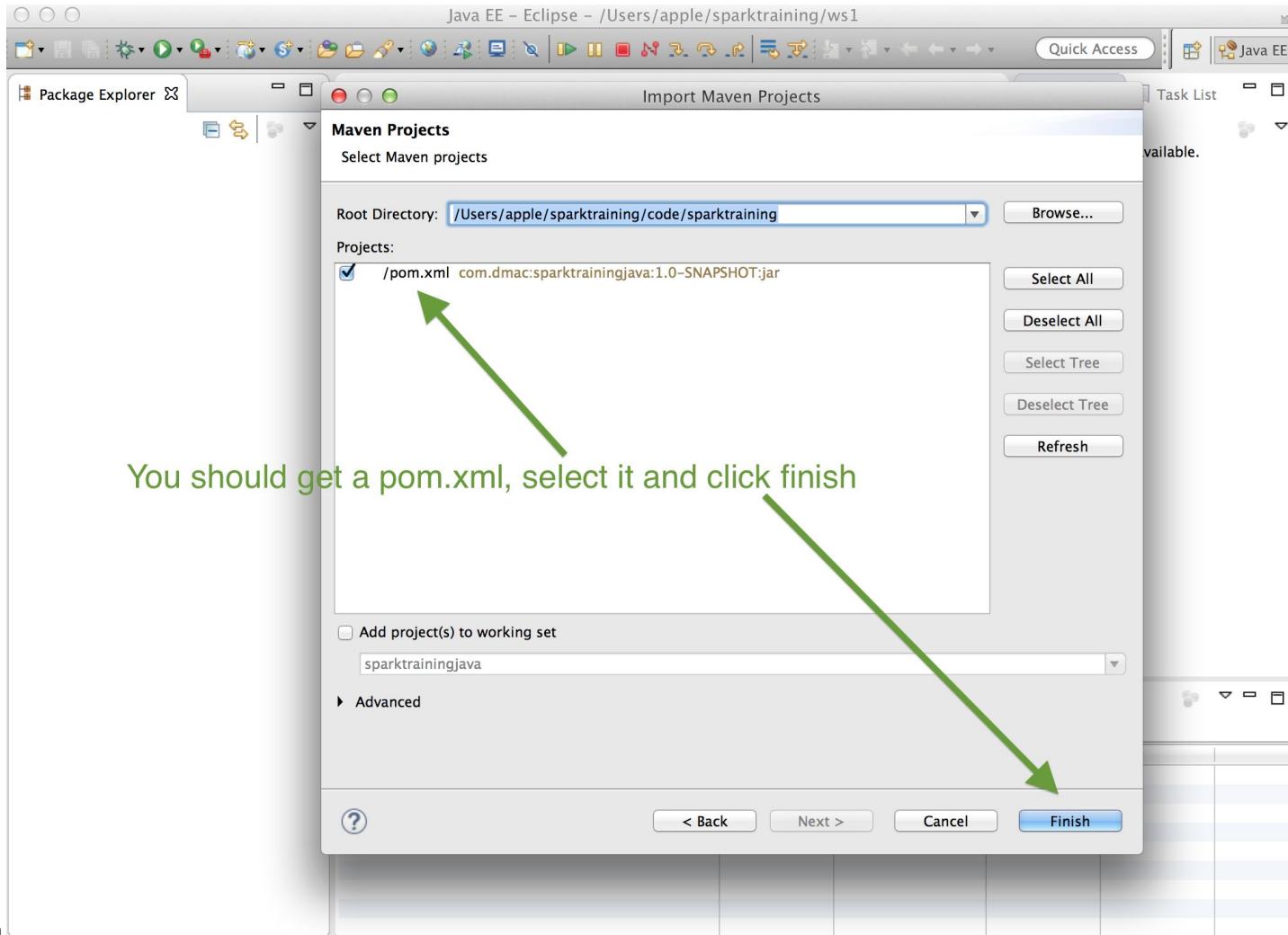


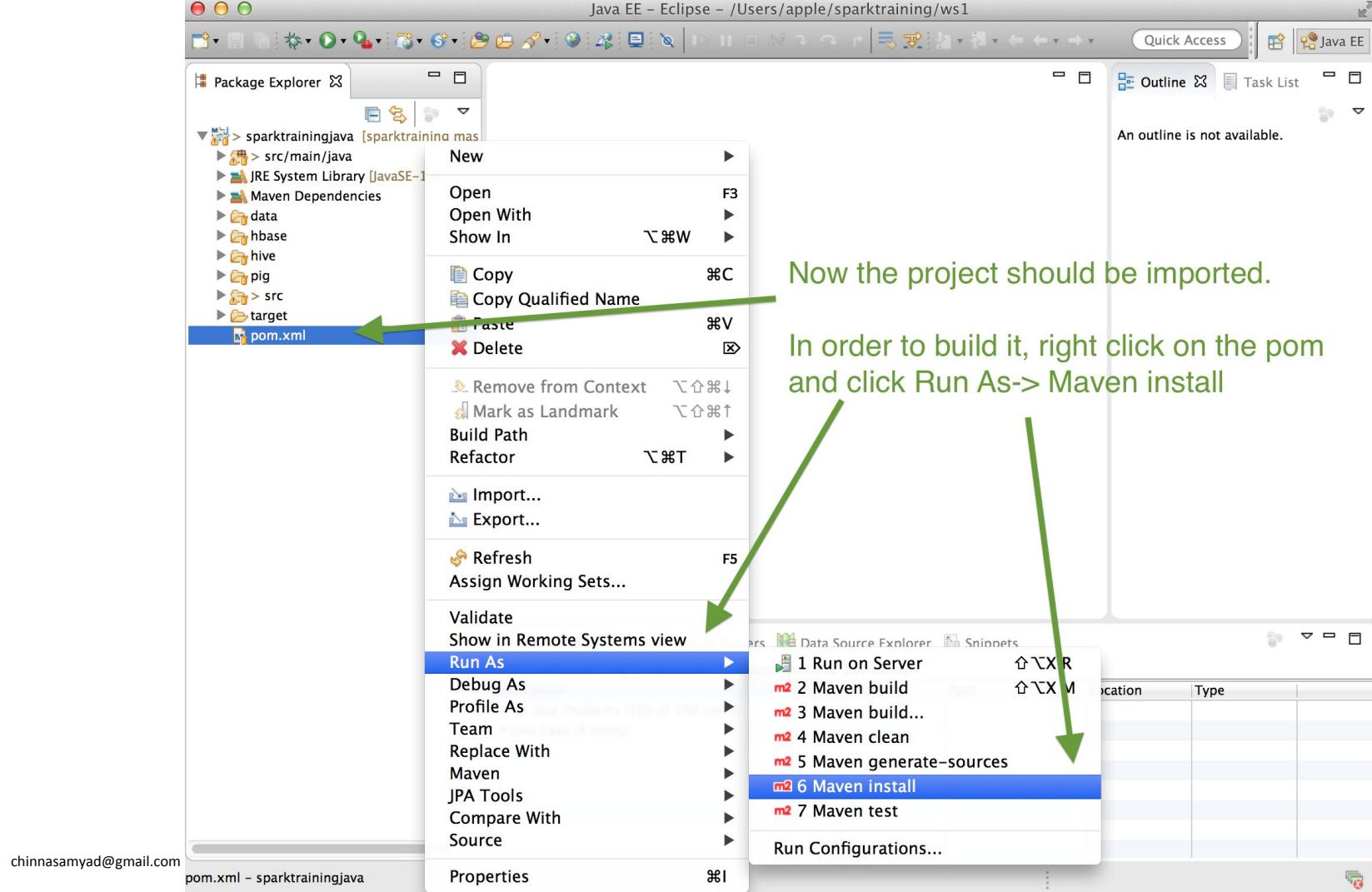
# Importing the code



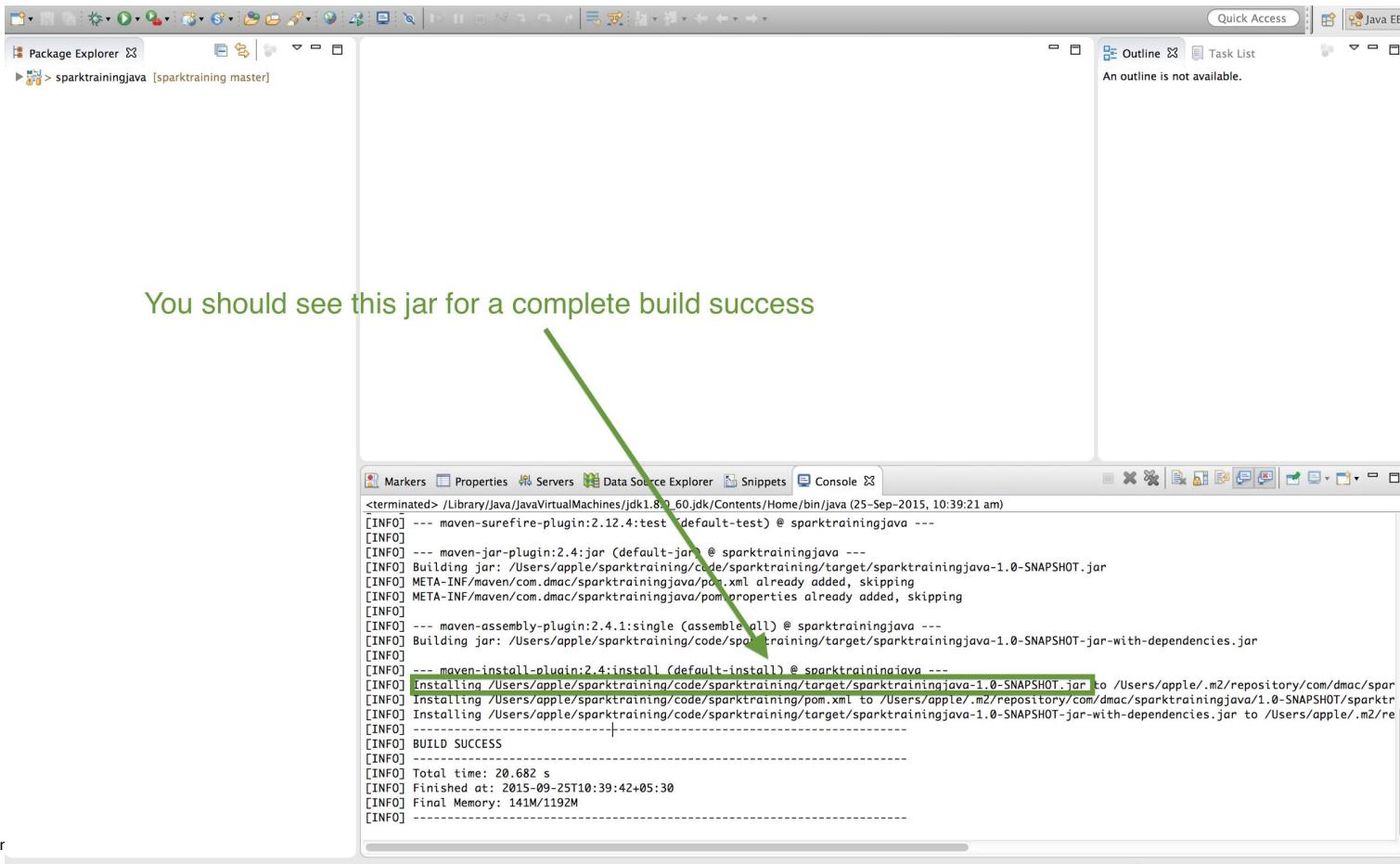
# Importing the code







# Importing the code



# IntelliJ IDE Scala Setup



# IntelliJ - Scala Setup

The screenshot shows a web browser displaying the Scala download page at [www.scala-lang.org/download/](http://www.scala-lang.org/download/). The URL bar is highlighted with an orange box. The page features a dark header with the Scala logo and navigation links for DOCUMENTATION and DOWNLOAD. A large blue "DOWNLOAD" button is centered on a blue background section. The main content area is titled "Choose one of three ways to get started with Scala!"

1

Download Scala 2.11.7 binaries for your system ([All downloads](#)).



[Need help installing?](#)

Or

2

## Get started with Typesafe Activator

Typesafe Activator is a browser-based or command-line tool that helps developers get started with Scala.

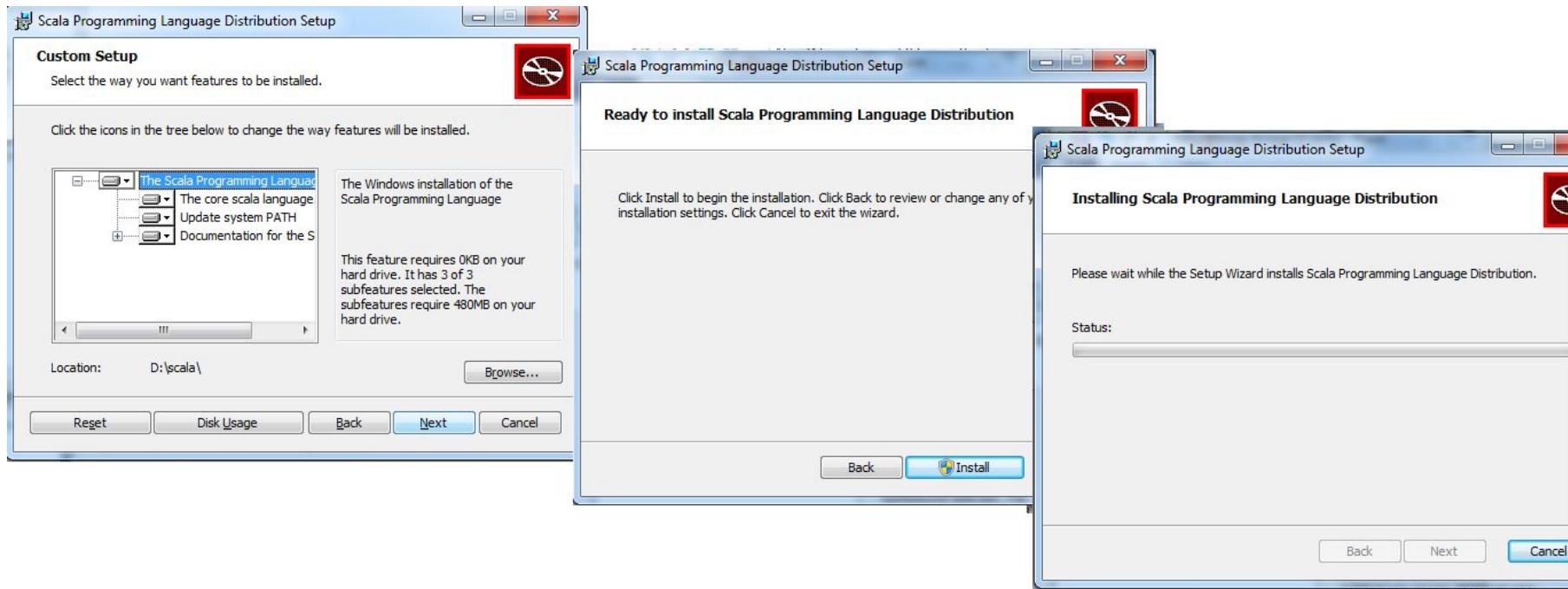


[Need help installing?](#)

# IntelliJ - Scala Setup



# IntelliJ - Scala Setup

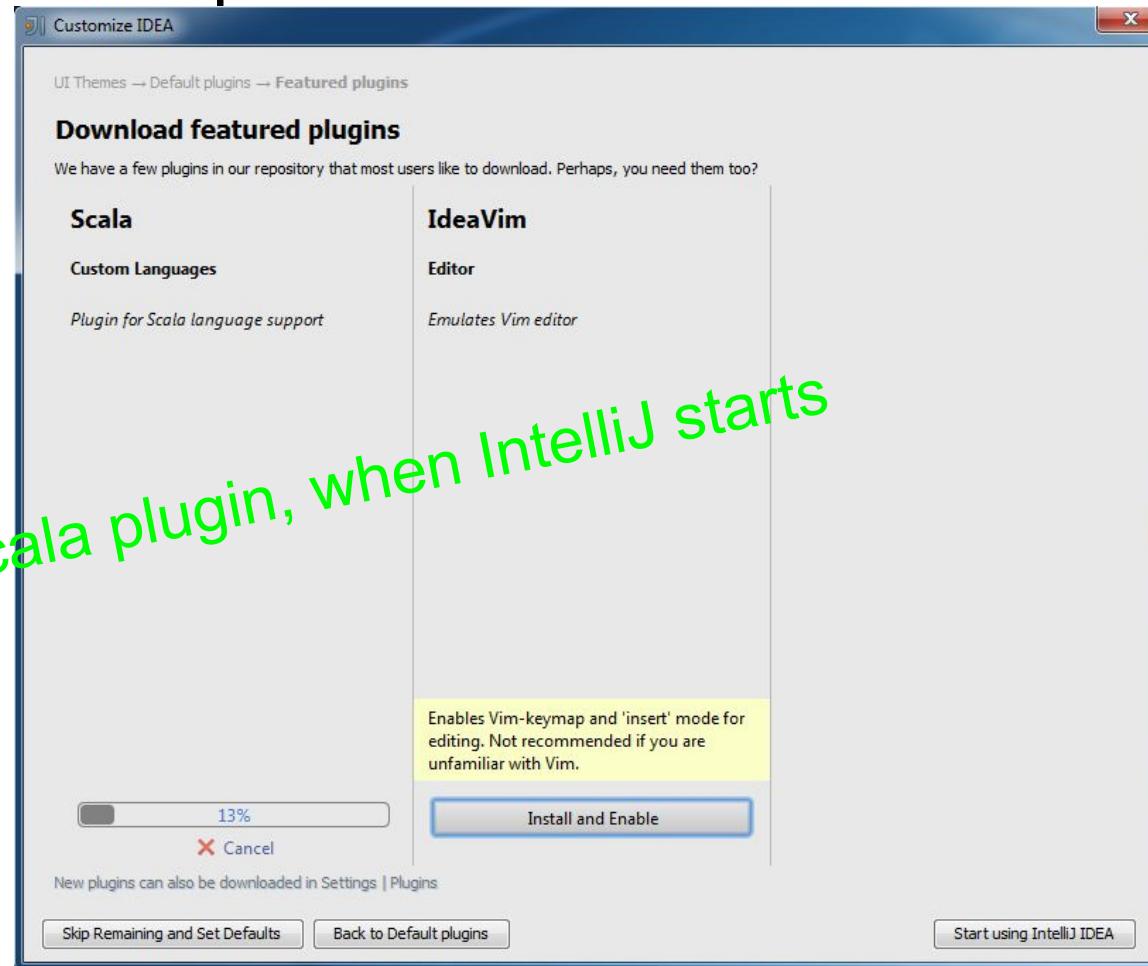


# IntelliJ - Scala Setup

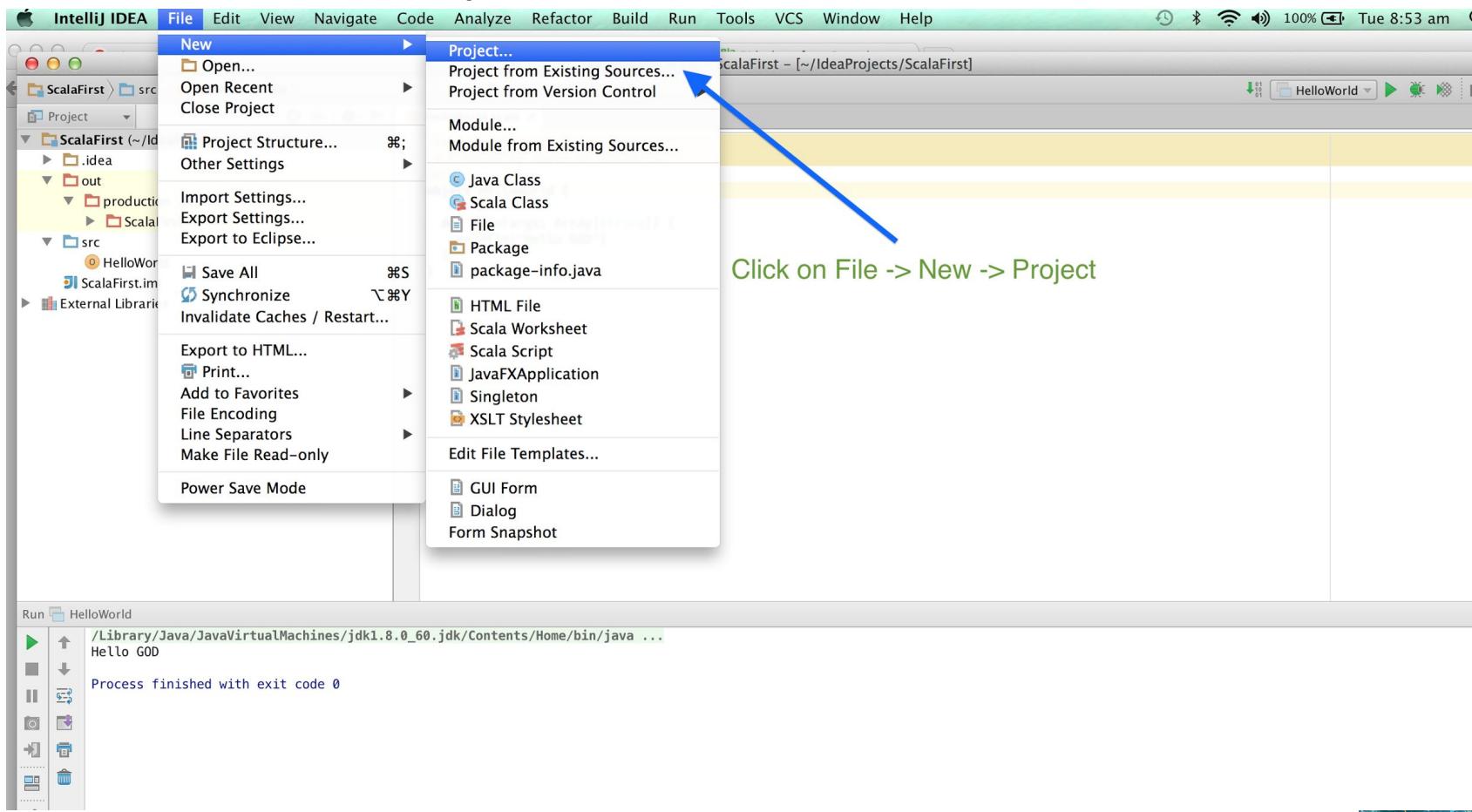


# IntelliJ - Scala Setup

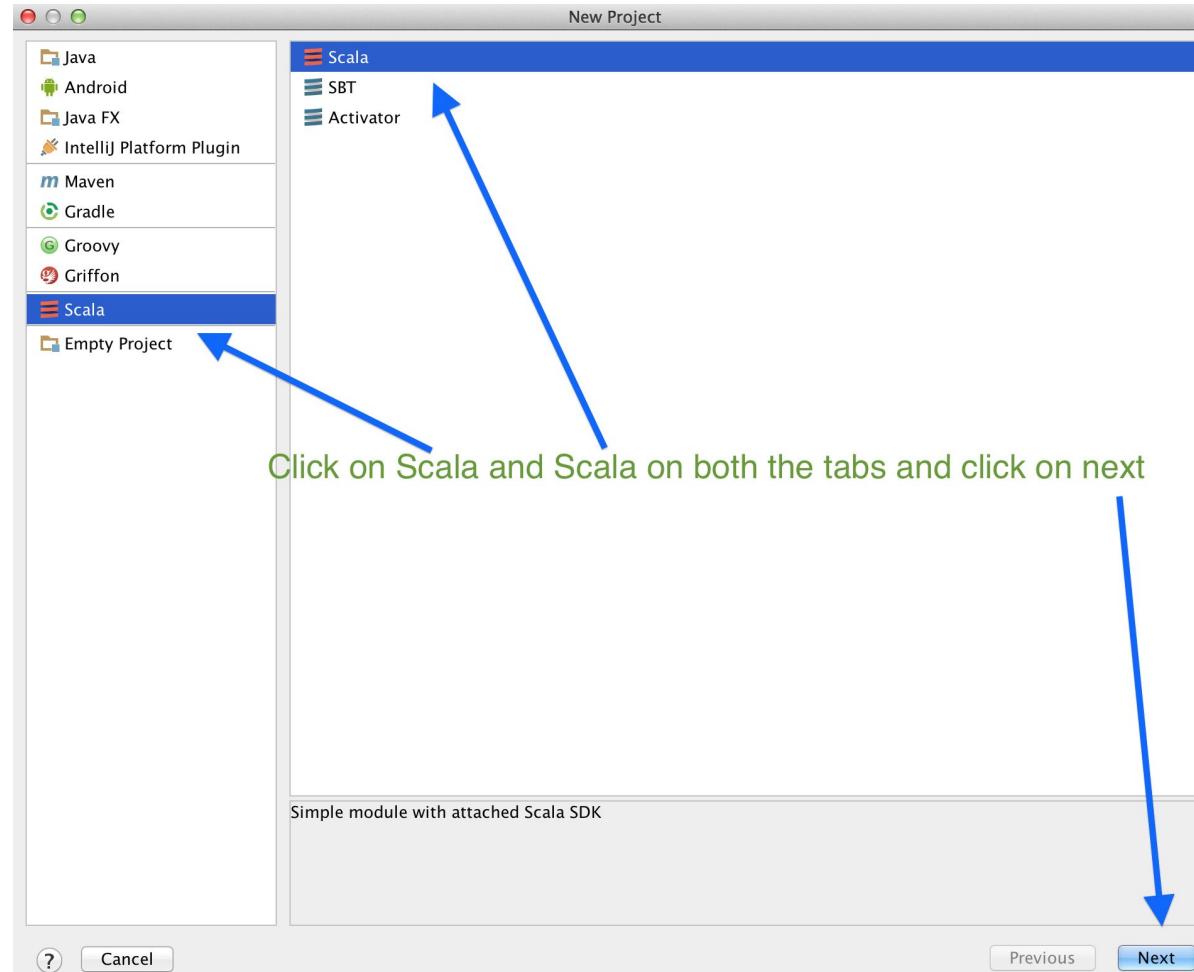
Install the scala plugin, when IntelliJ starts first



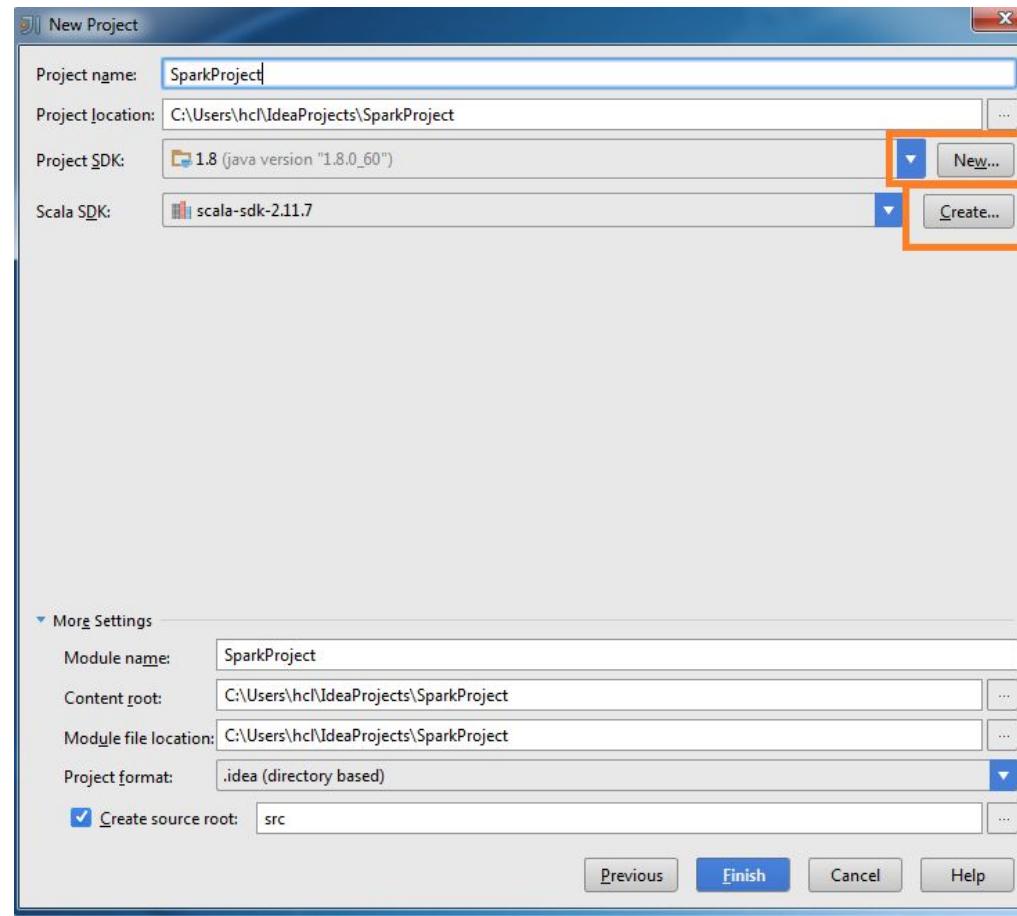
# IntelliJ - Scala Setup



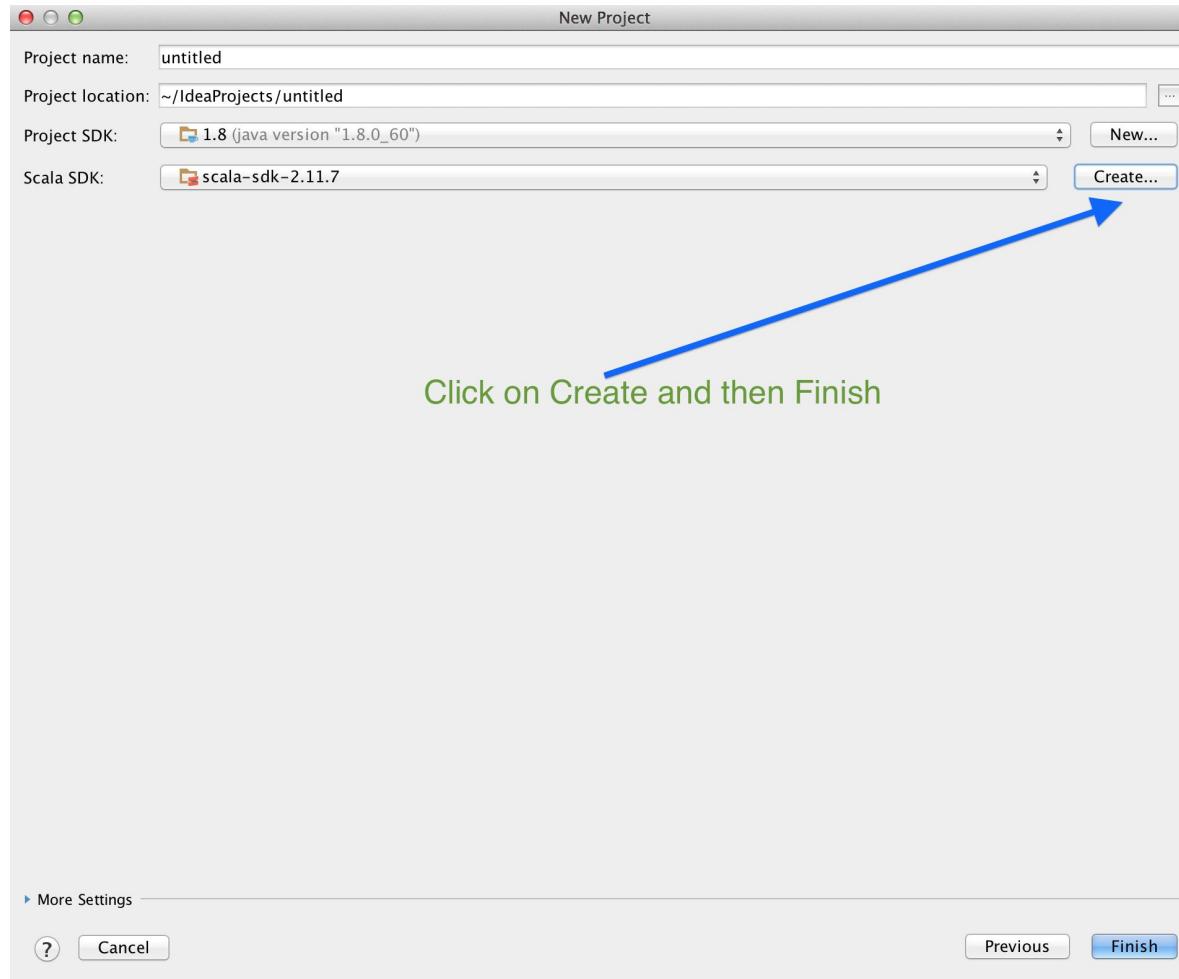
# IntelliJ - Scala Setup



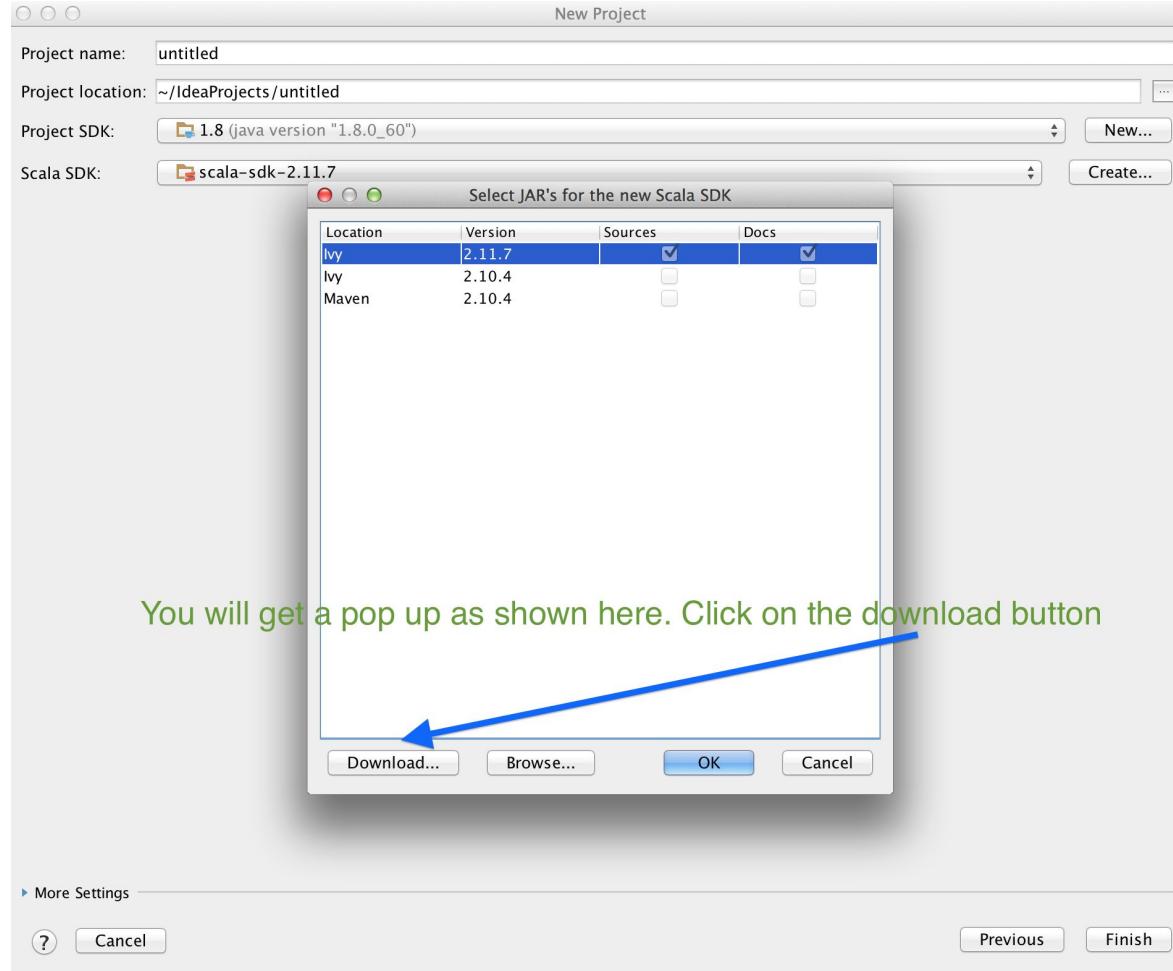
# IntelliJ - Scala Setup



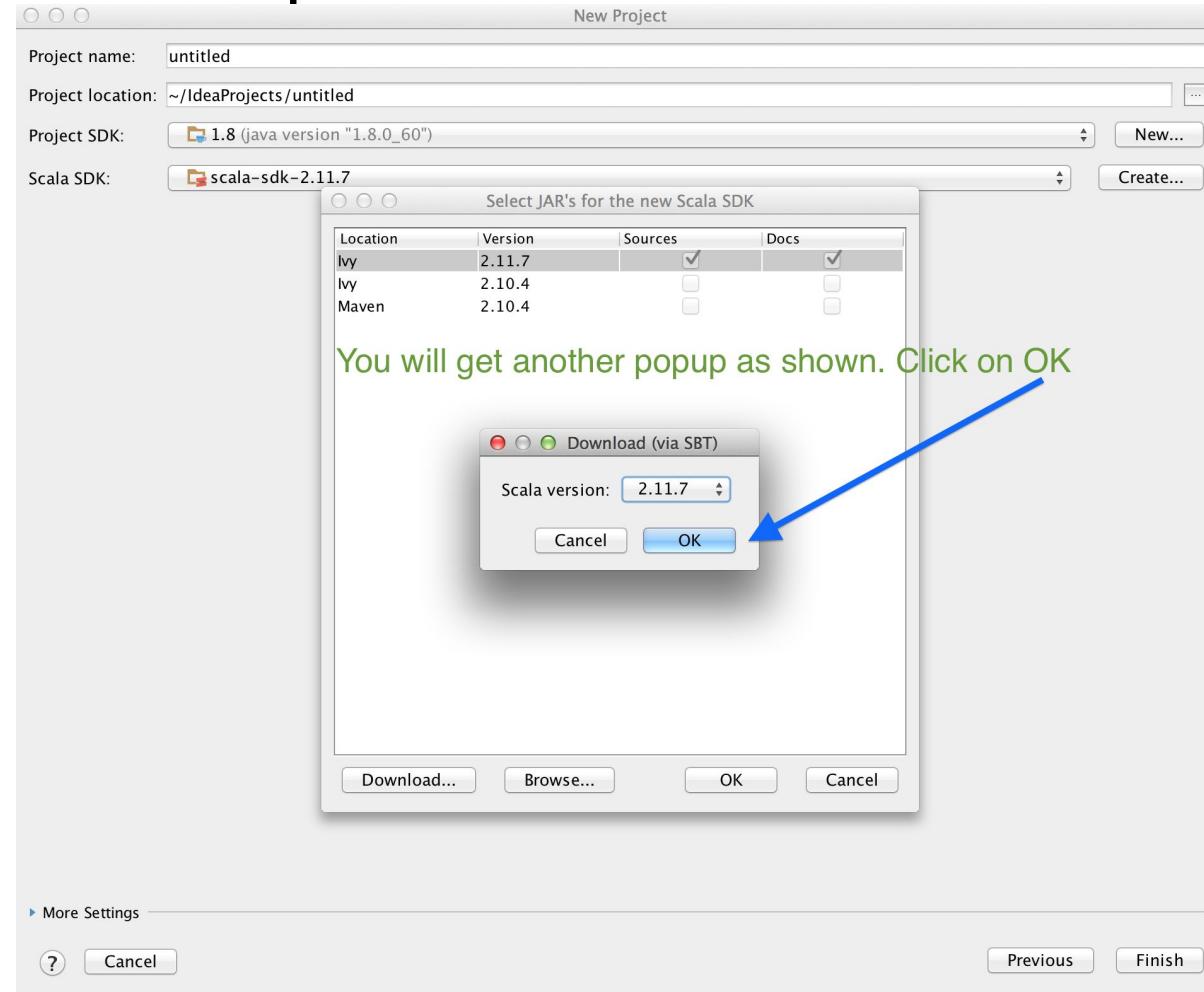
# IntelliJ - Scala Setup



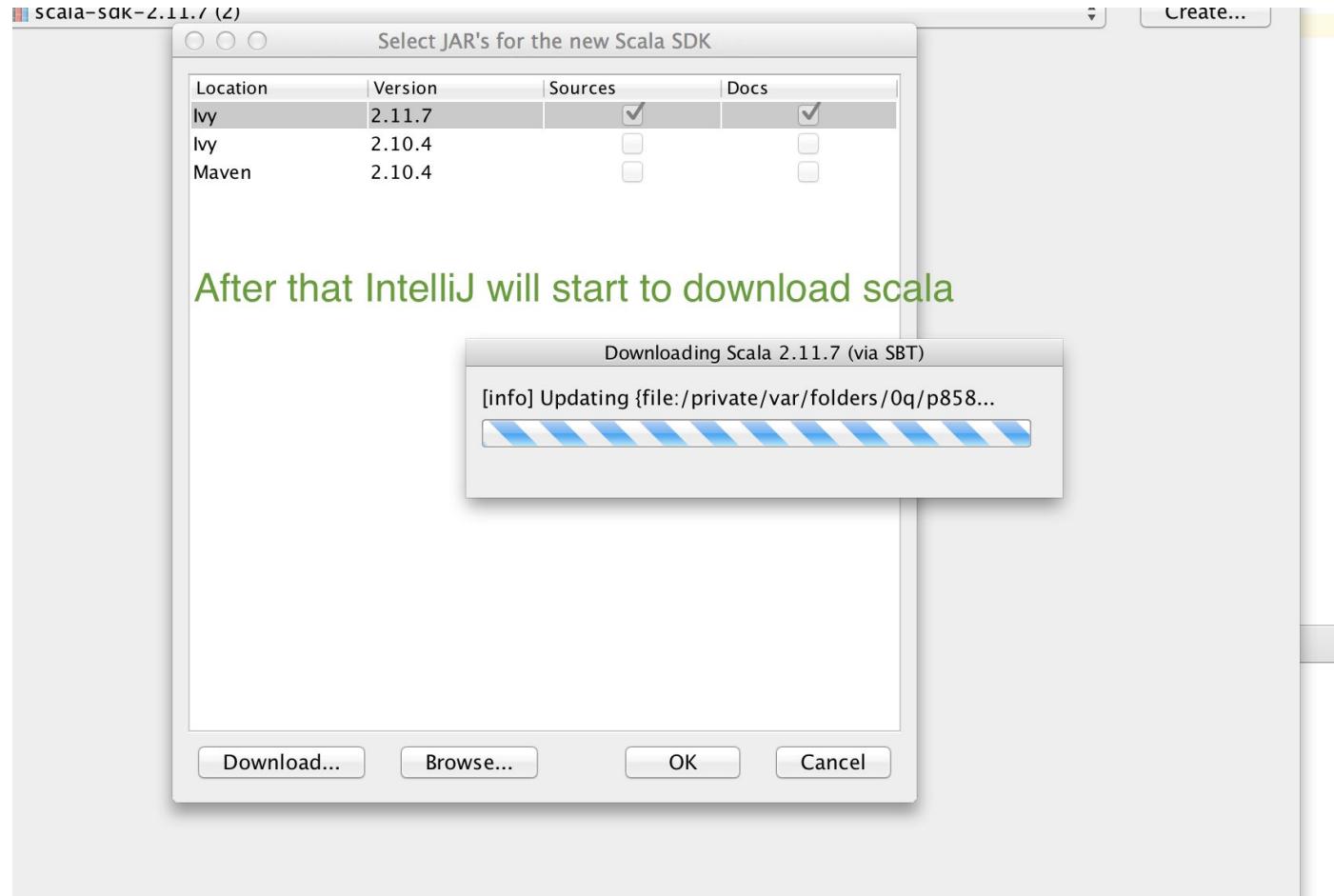
# IntelliJ - Scala Setup



# IntelliJ - Scala Setup



# IntelliJ - Scala Setup



# IntelliJ - Scala Setup

The screenshot shows a web browser window titled "Scala for IntelliJ IDEA :: JetBrains Plugin Repository - Google Chrome". The URL is <https://plugins.jetbrains.com/idea/plugin/1347-scala>. The page displays a table of Scala plugin versions, their compatible IntelliJ builds, update dates, and download links. A "Stable" tab is selected. A "PREVIOUS UPDATES" button is visible at the bottom left. Below the table, there is a section titled "GENERAL USAGE INSTRUCTIONS" with a note about IntelliJ IDEA being an IDE famous for its ergonomics and intelligent coding assistance.

VERSION	COMPATIBLE BUILDS	UPDATE DATE	DOWNLOAD
2017.1.8	171.1834.1–172	03.02.2017	<a href="#">DOWNLOAD</a>
2016.3.8	163.4396–164	24.01.2017	<a href="#">DOWNLOAD</a>
2016.3.7	163.4396–164	20.01.2017	<a href="#">DOWNLOAD</a>
2017.1.6	171.1834.1–172	11.01.2017	<a href="#">DOWNLOAD</a>
2016.3.6	163.4396–164	09.01.2017	<a href="#">DOWNLOAD</a>
2017.1.5	171.1834.1–172	22.12.2016	<a href="#">DOWNLOAD</a>
2016.3.5	163.4396–164	12.12.2016	<a href="#">DOWNLOAD</a>
2016.3.4	163.4396–164	01.12.2016	<a href="#">DOWNLOAD</a>
2016.3.3	163.4396–164	21.11.2016	<a href="#">DOWNLOAD</a>
2016.3.2	163.4396–164	27.09.2016	<a href="#">DOWNLOAD</a>

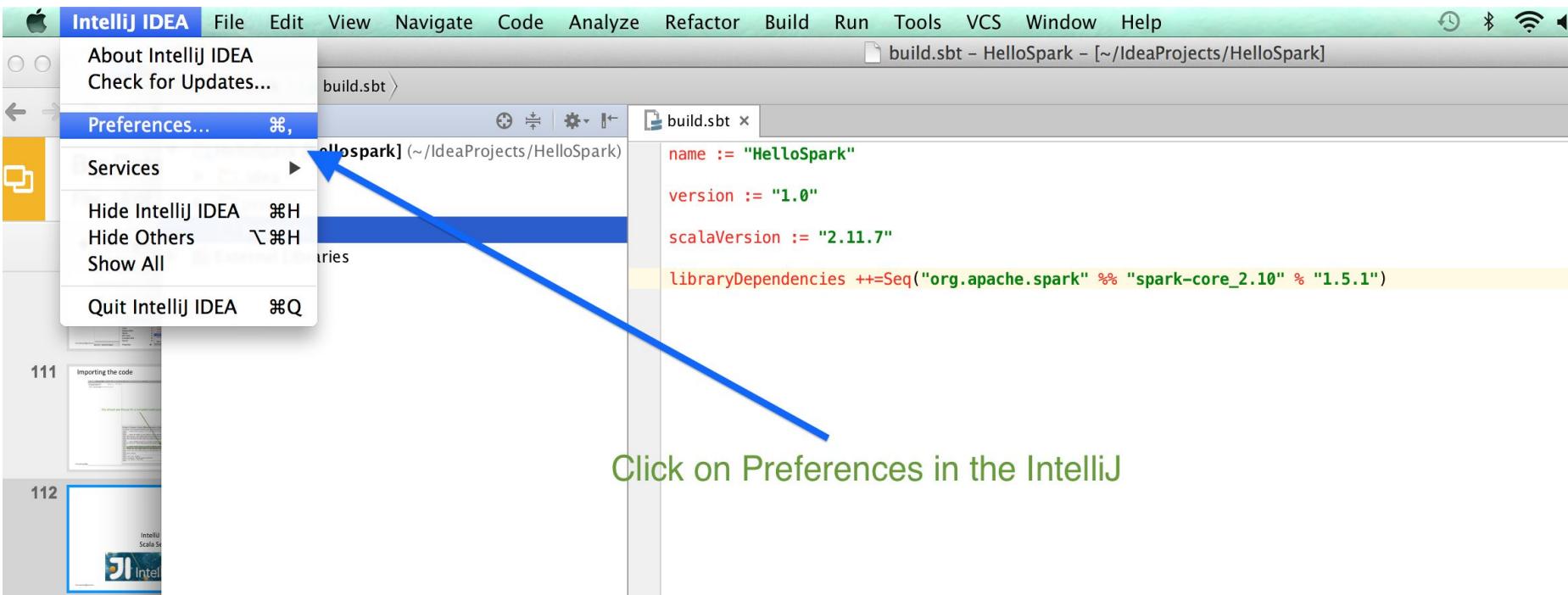
You can also download the scala plugin from and install it manually in the IntelliJ,

<https://plugins.jetbrains.com/idea/plugin/1347-scala>

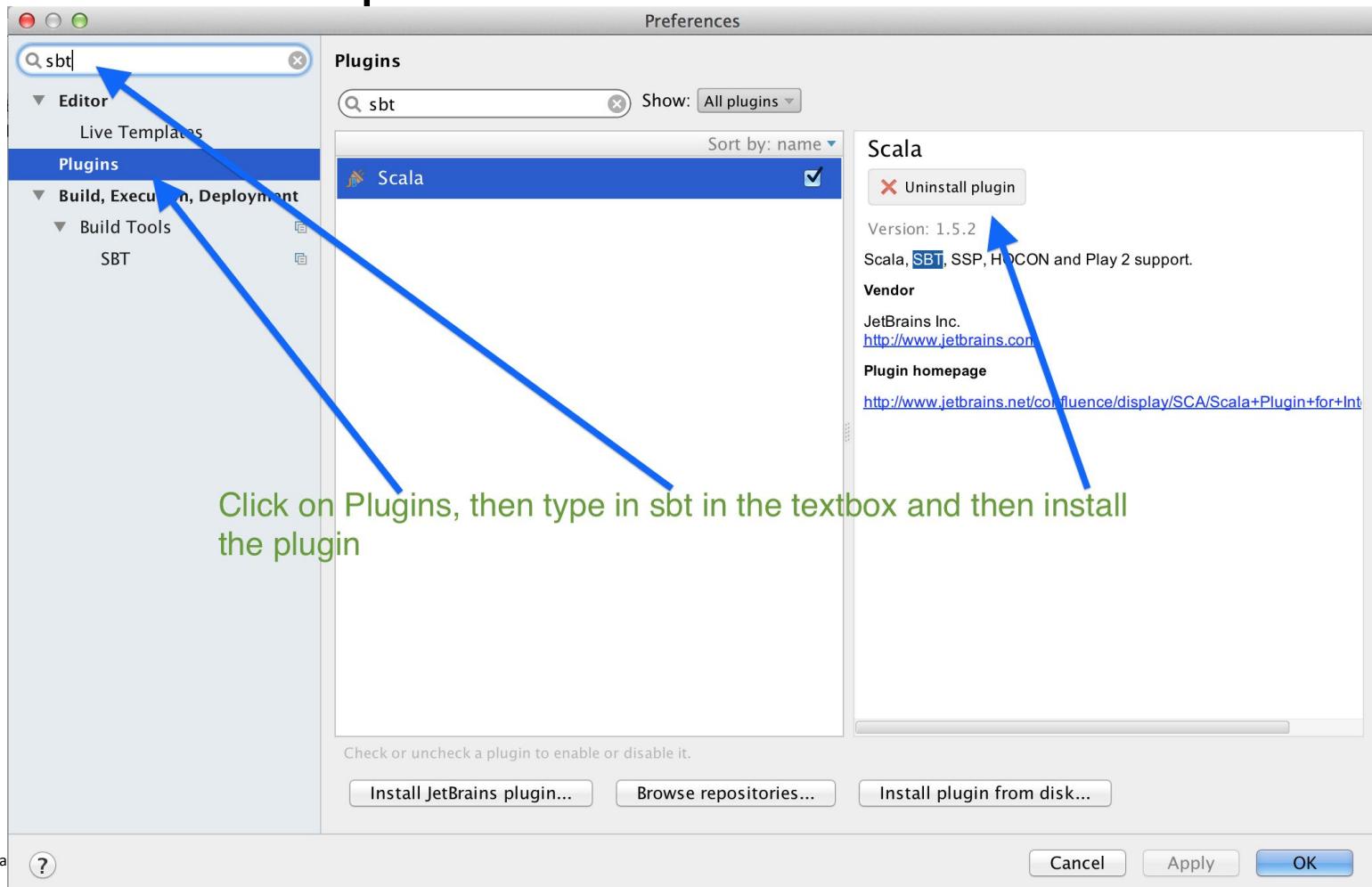
# IntelliJ IDE sbt Setup



# IntelliJ - sbt Setup



# IntelliJ - sbt Setup

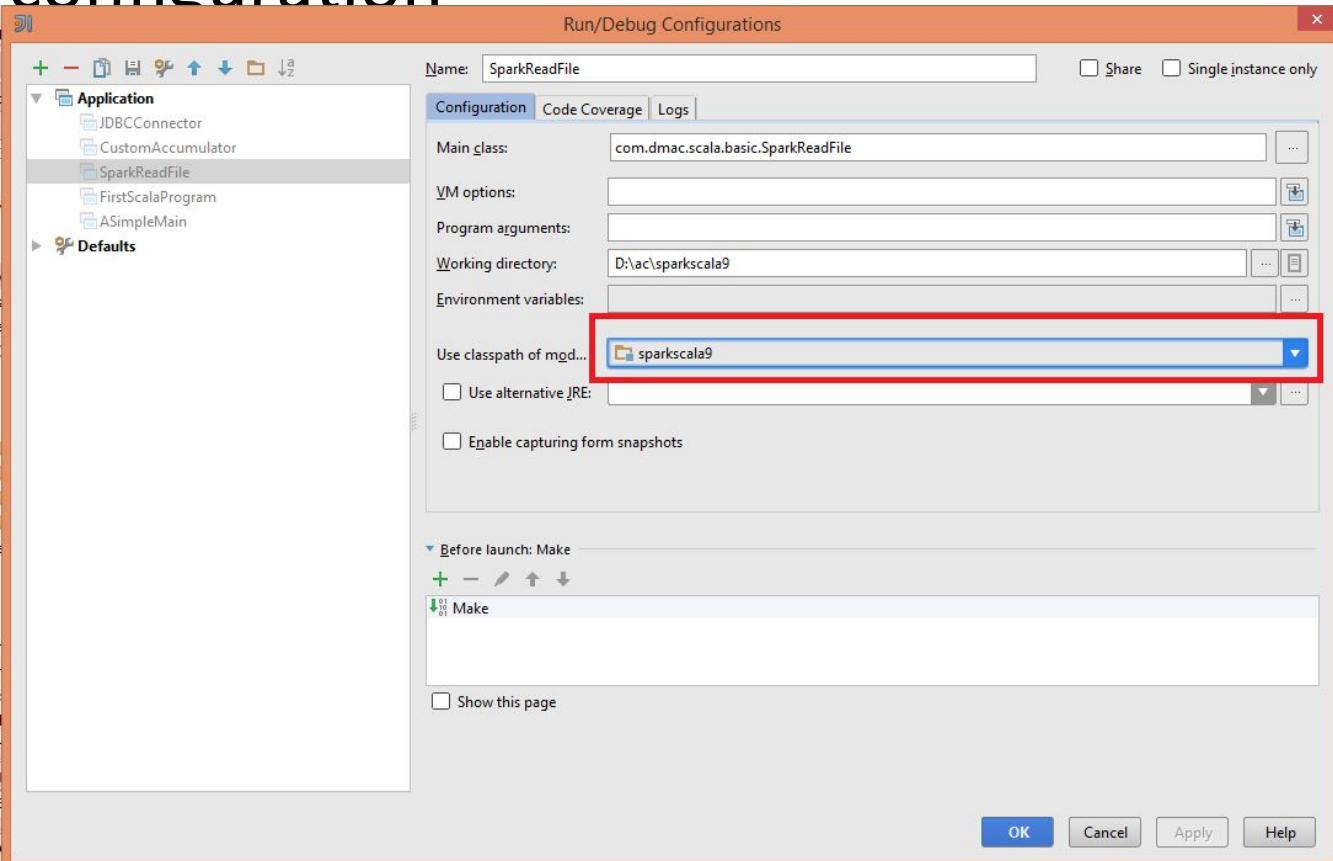


# IntelliJ IDE

## Correctness Check



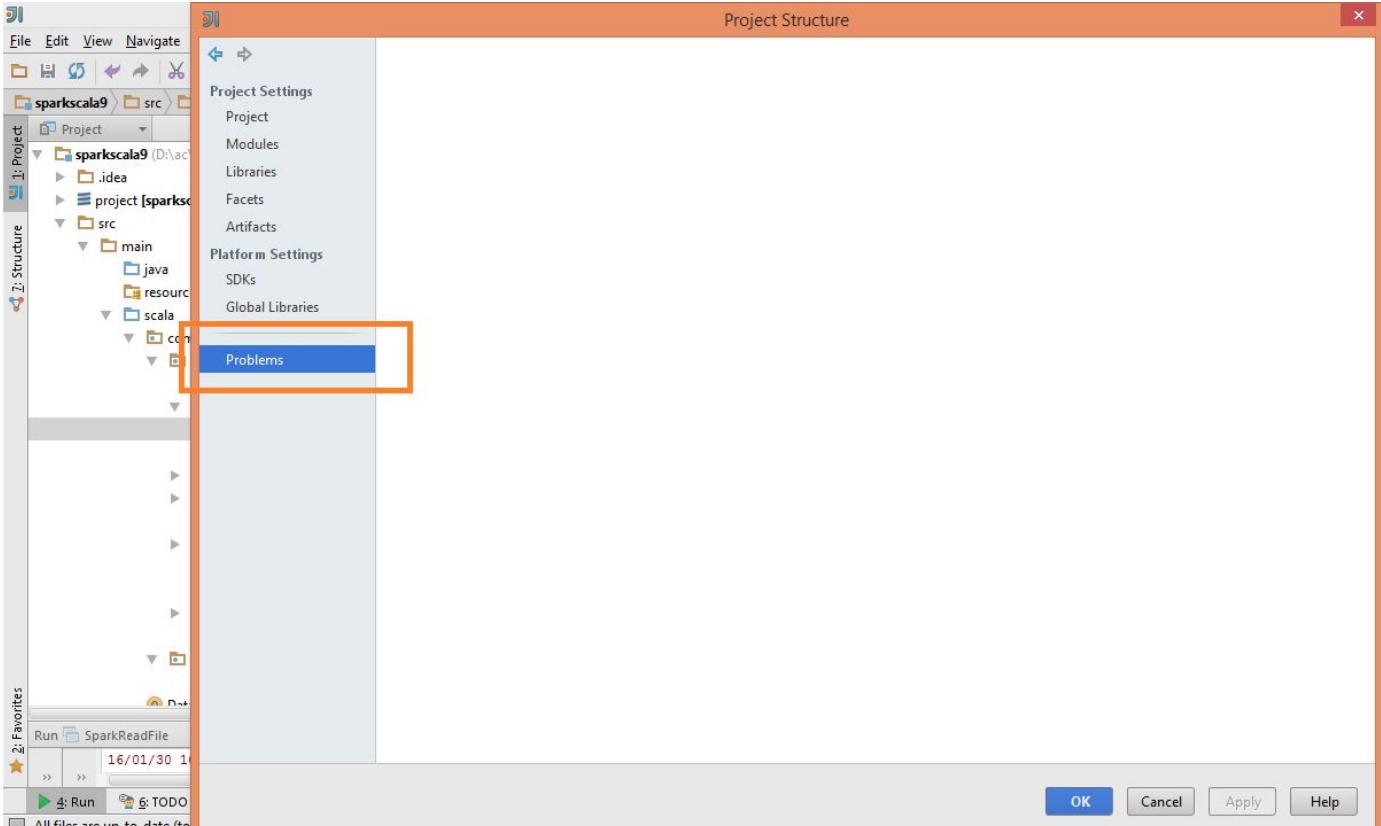
# IntelliJ - Check configuration



Goto Run->  
Edit Configurations

use classpath....  
should contain your project

# IntelliJ - Check configuration



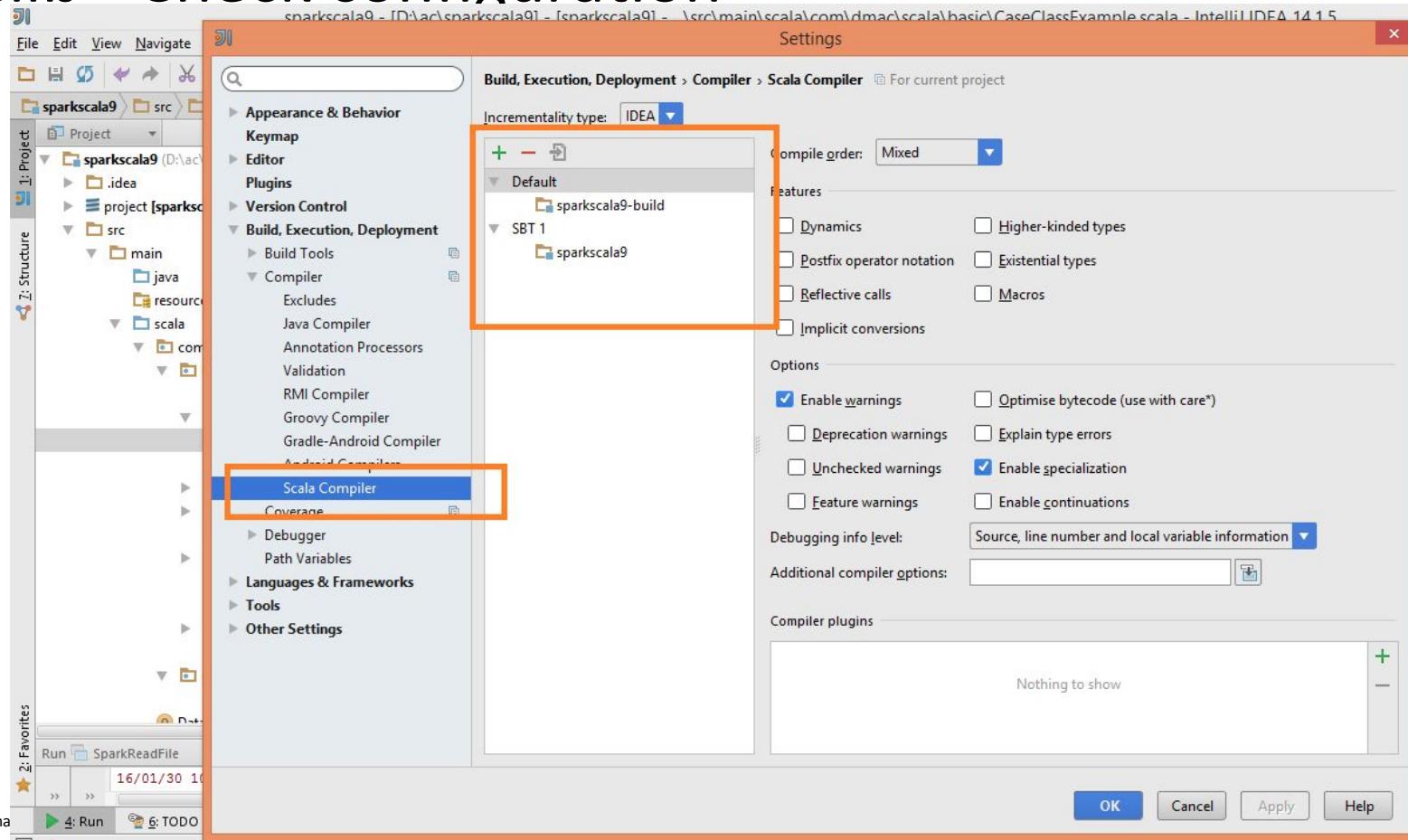
Goto File>  
Project Structure

Problems should not contain any entry

chinnasamyad@gmail.com

# IntelliJ - Check configuration

File -> Settings

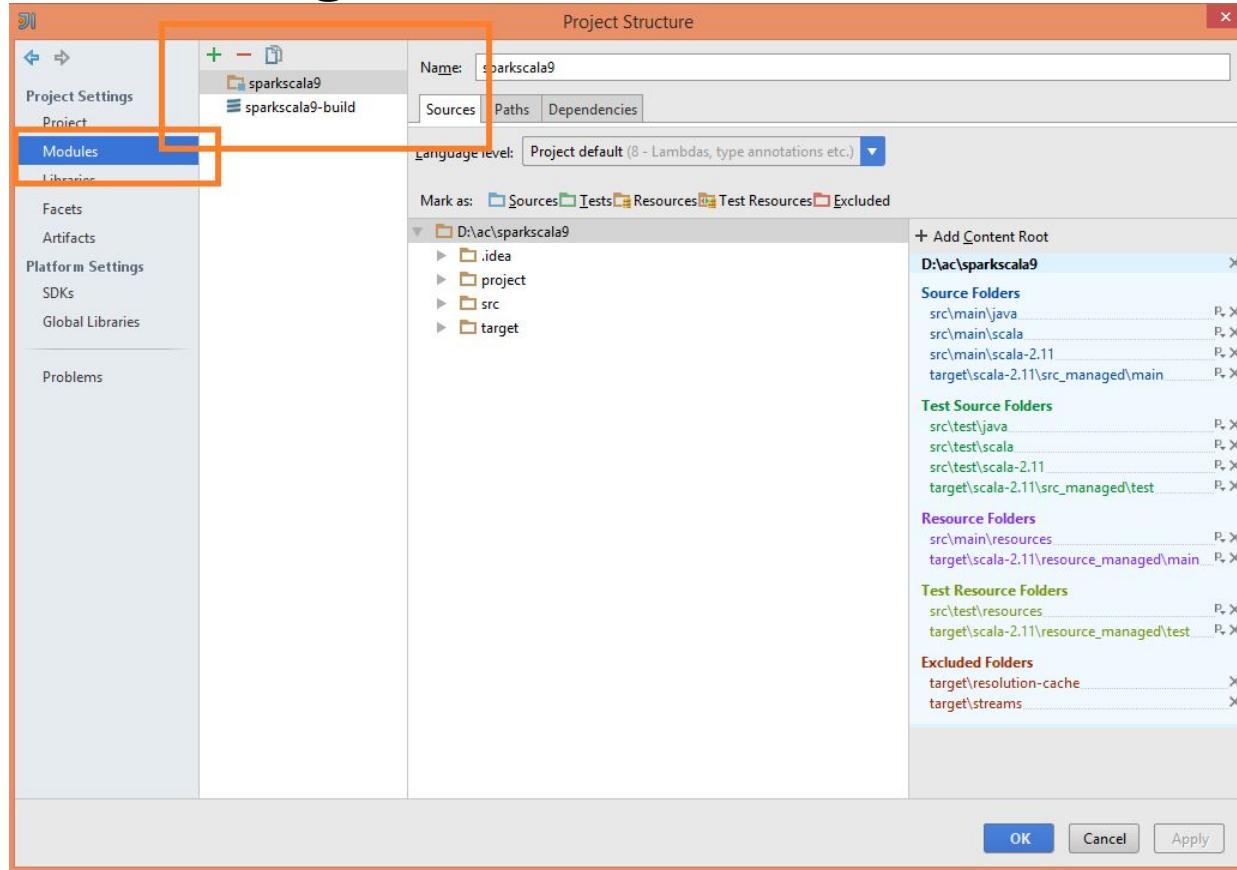


chinnasamyad@gmail.com

IntelliJ 14.8

# IntelliJ - Check configuration

File -> Project  
Structure

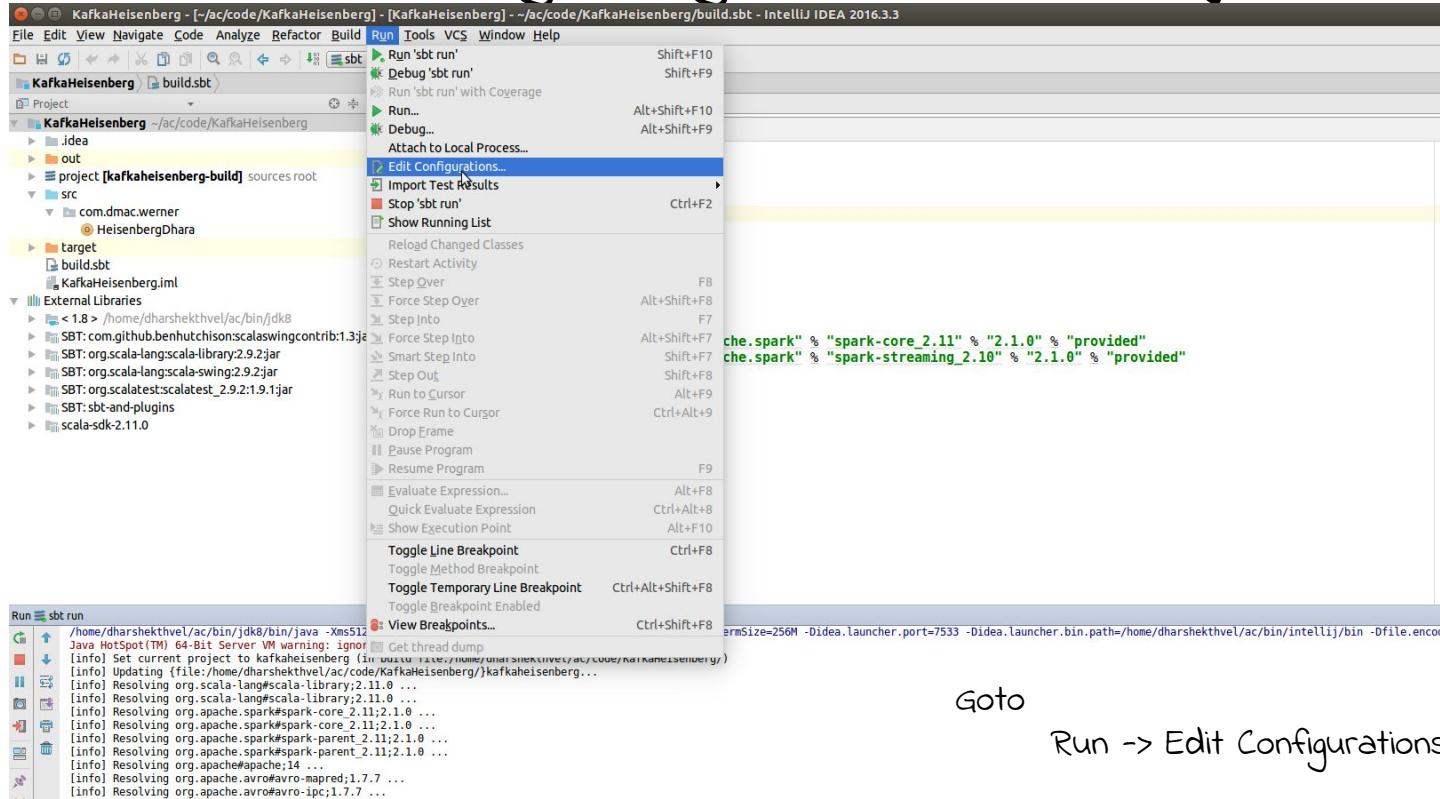


# IntelliJ IDE

## Integrating sbt with IntelliJ



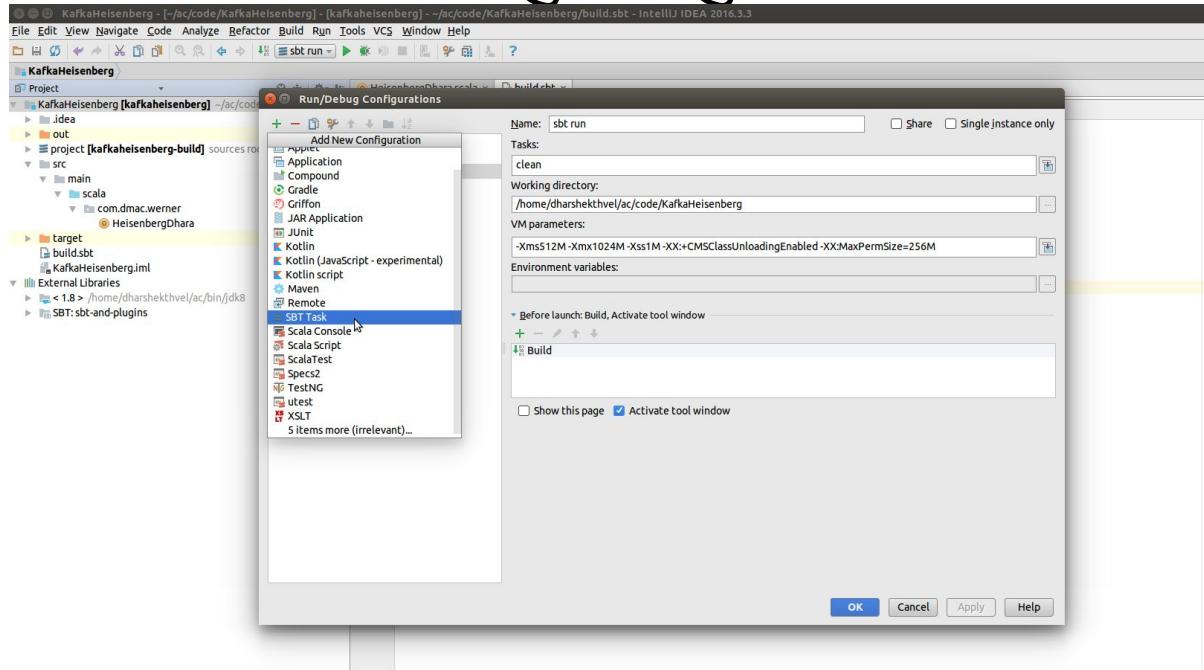
# IntelliJ IDE - Integrating sbt with IntelliJ



Goto

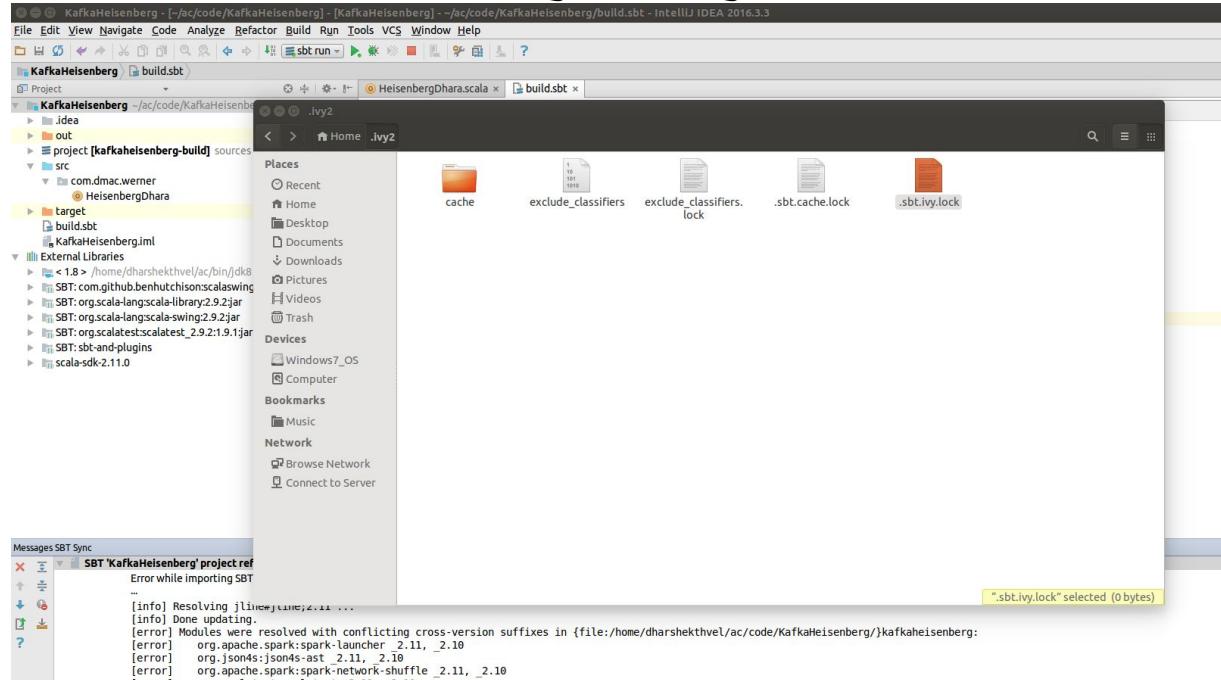
Run -> Edit Configurations

# IntelliJ IDE - Integrating sbt with IntelliJ



In the Run/Debug Configurations dialog, click on the + and choose SBT Task. Give a good name to it and give package as the task.

# IntelliJ IDE - Integrating sbt with IntelliJ



If you are getting a acquire to lock... error, then delete the `sbt.ivy.lock` file in the home `.ivy2` directory. `.ivy2` directory will be hidden.

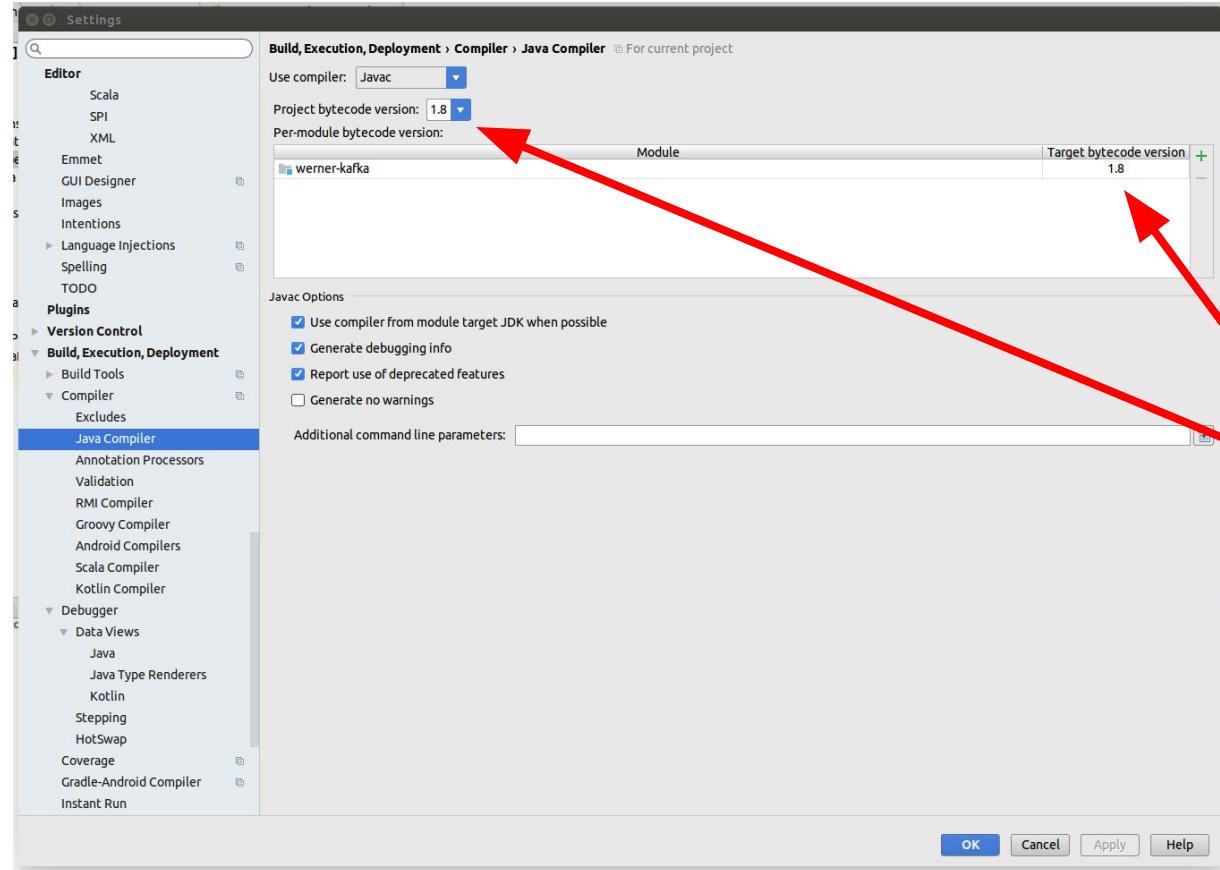
IntelliJ IDE

Byte Code not supported at this level



IntelliJ IDE

Byte Code not supported at this level

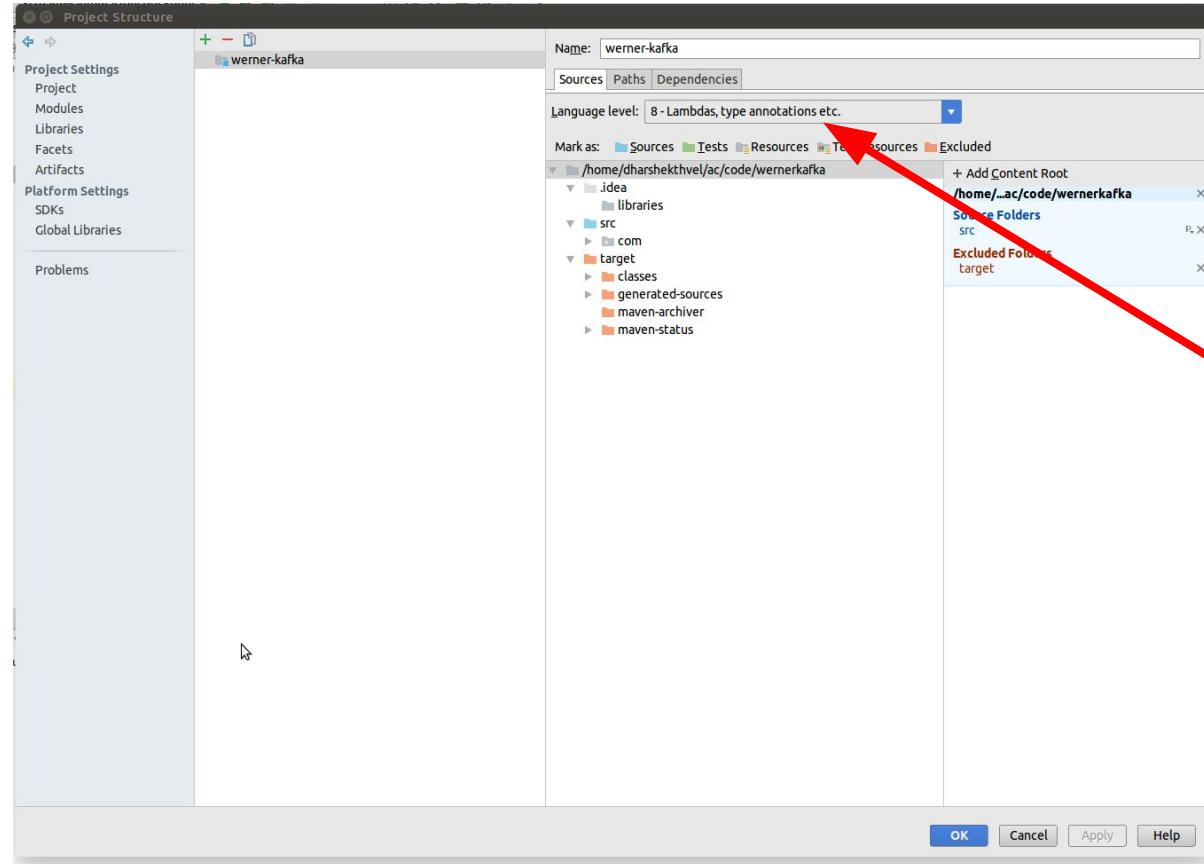


Ensure that you have done 1.8

IntelliJ IDE

Byte Code not supported at this level

File → Project Structure → modules

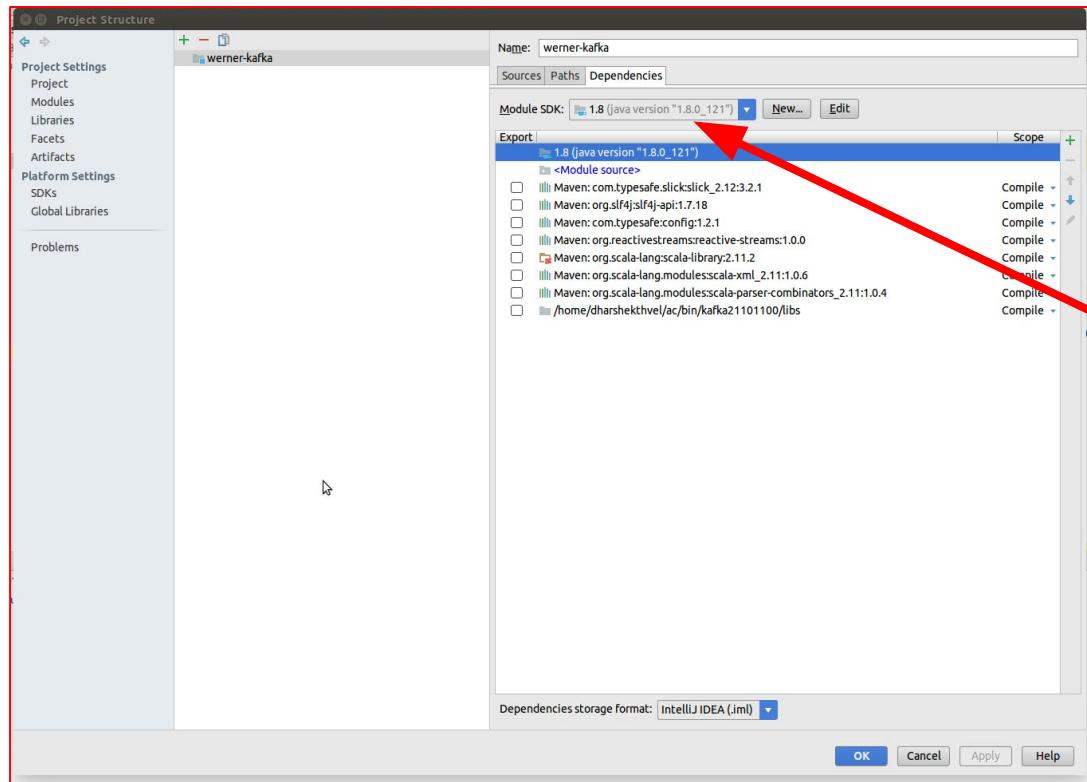


Ensure that you  
have done 1.8

IntelliJ IDE

Byte Code not supported at this level

File → Project Structure → modules

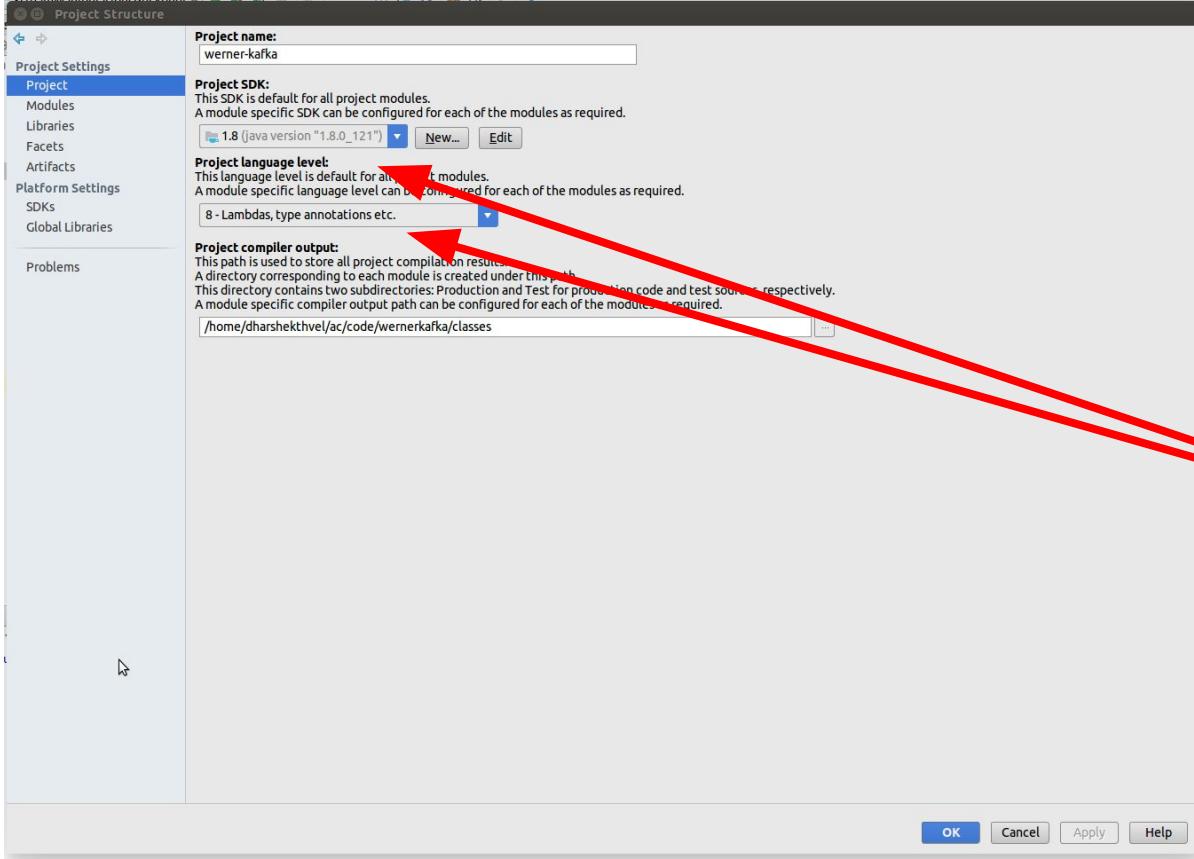


Ensure that you  
have done 1.8

IntelliJ IDE

Byte Code not supported at this level

File → Project Structure → project



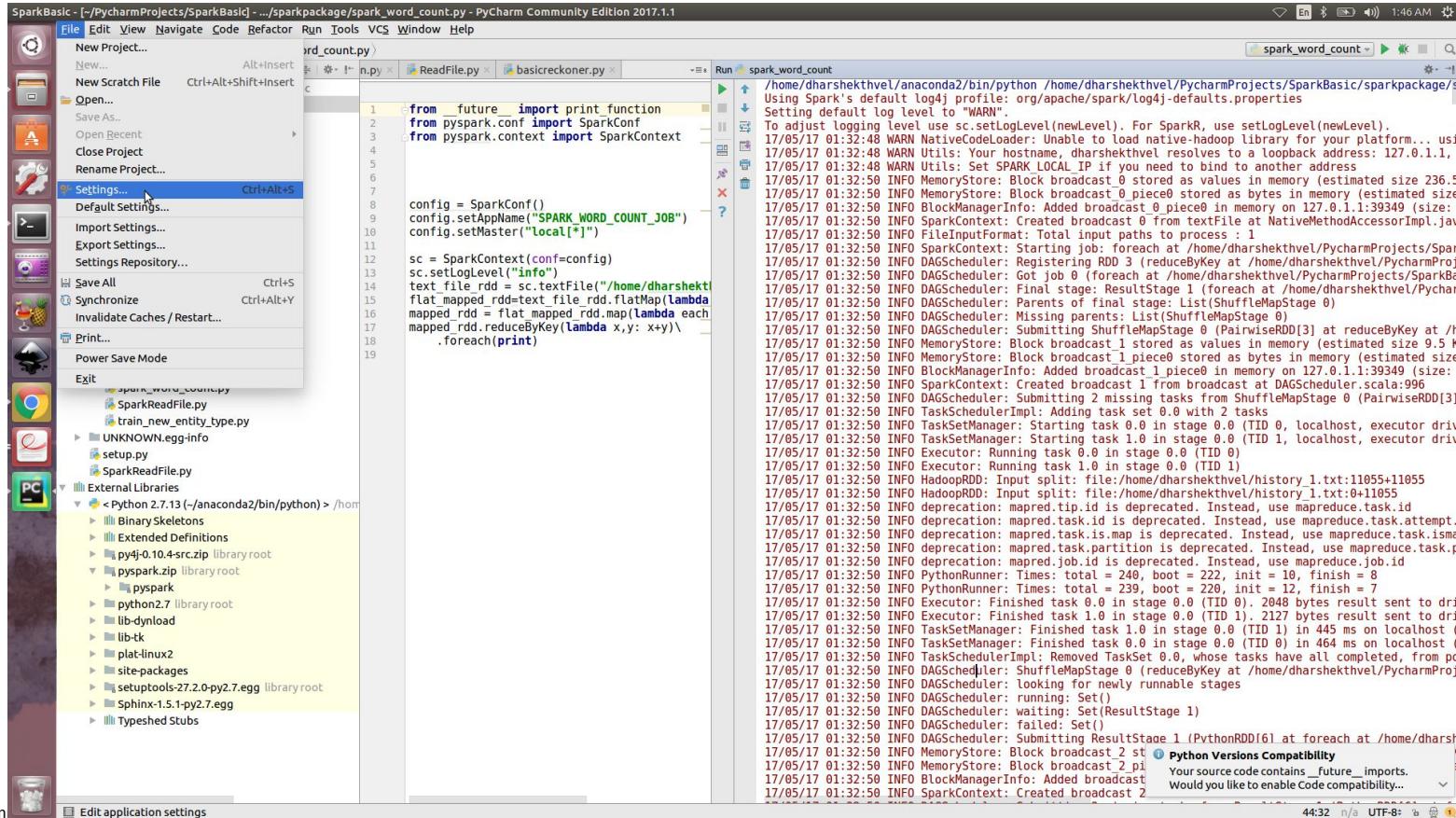
Ensure that you  
have done 1.8

# Pycharm - Spark Setup

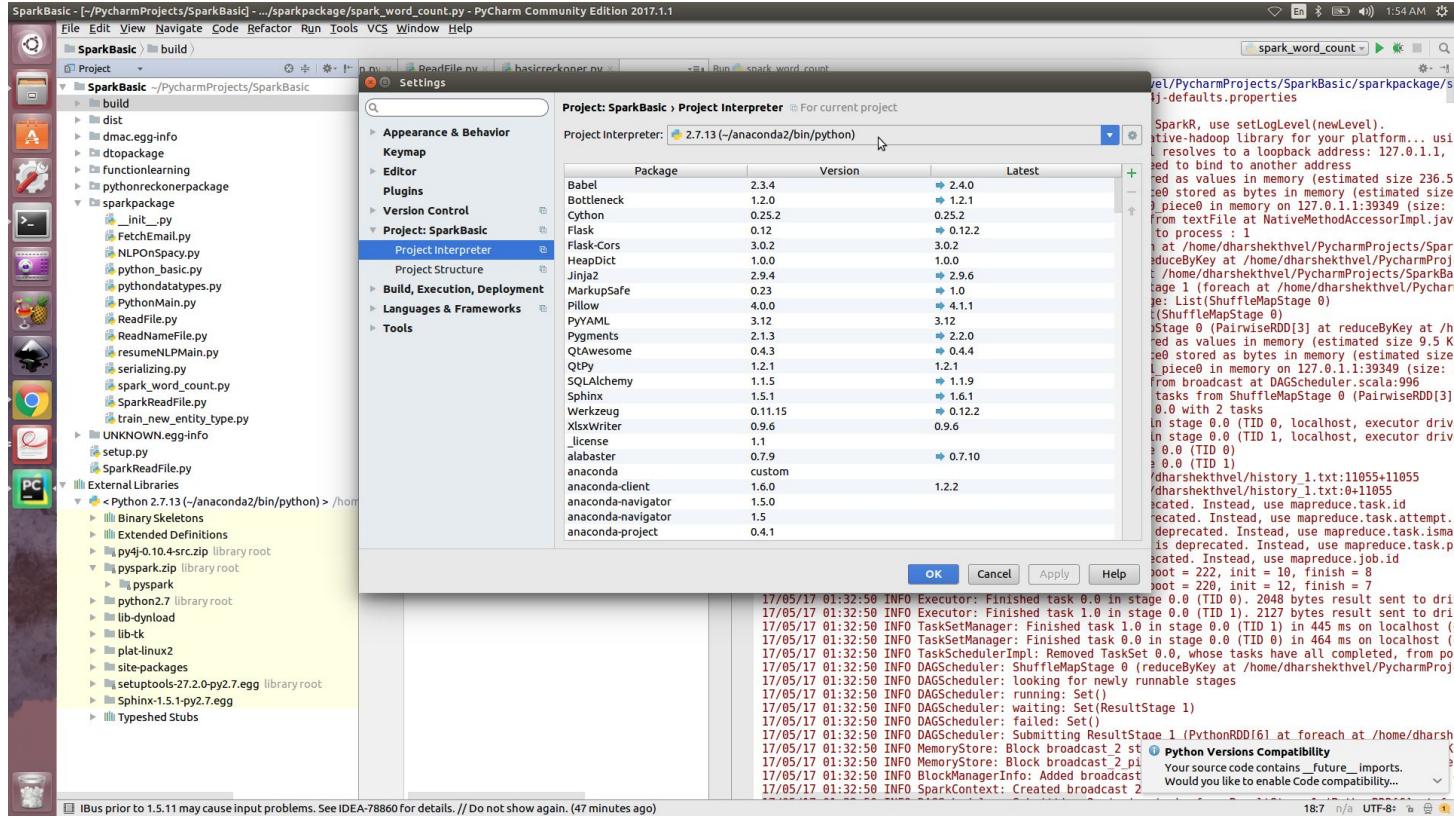


# Pycharm - Spark Setup

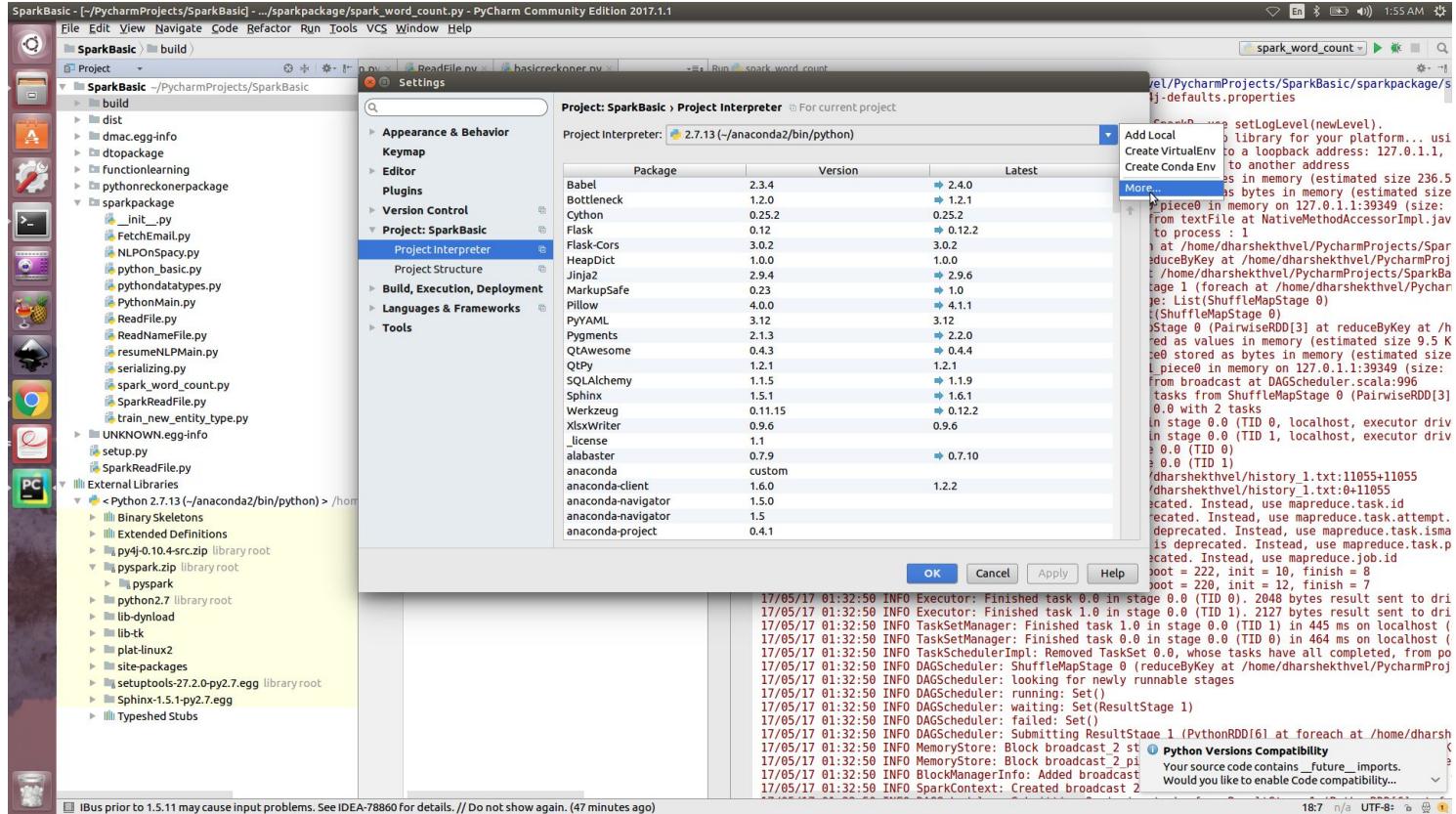
## File -> Settings...



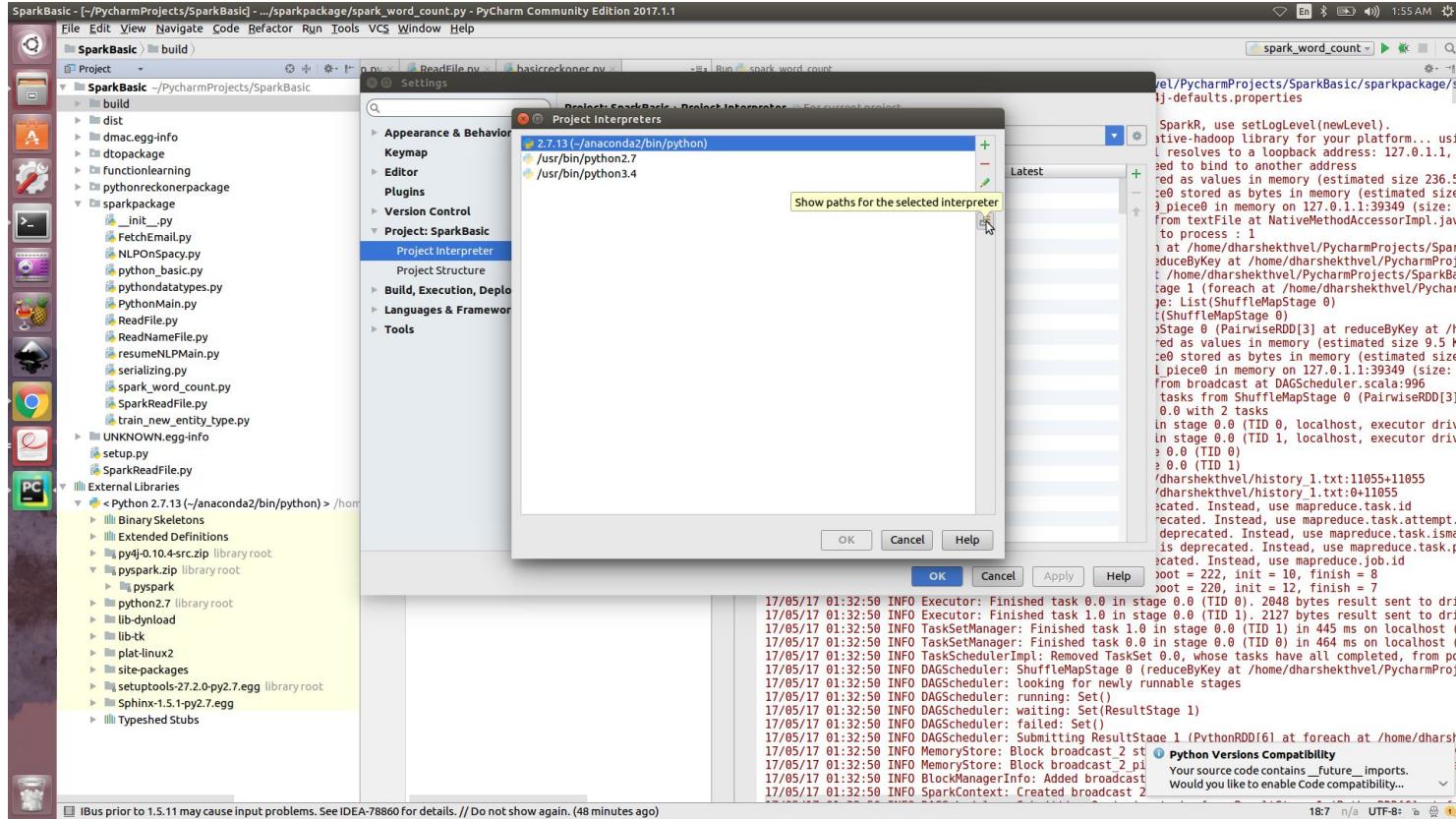
# Pycharm - Spark Setup



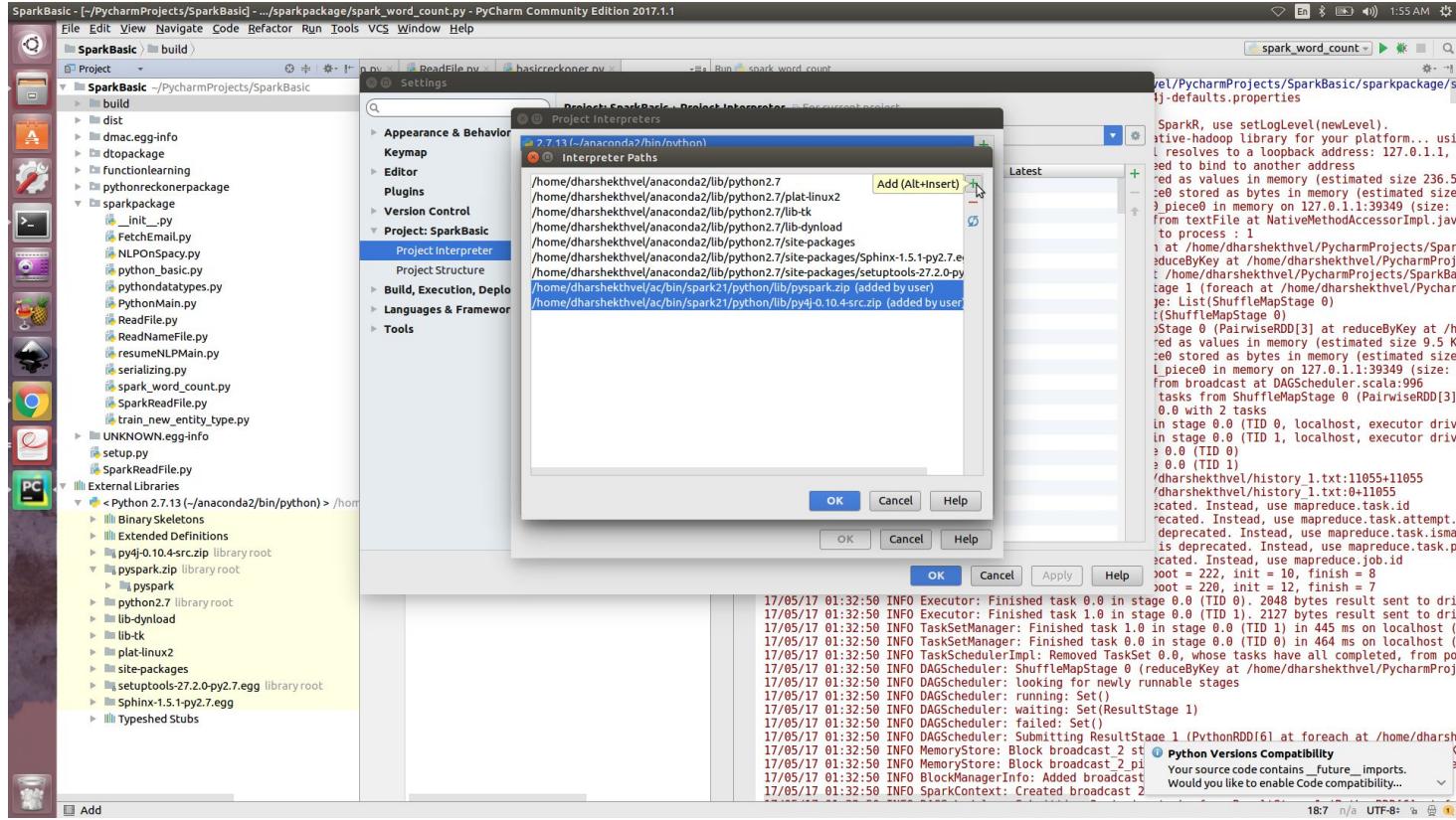
# Pycharm - Spark Setup



# Pycharm - Spark Setup



# Pycharm - Spark Setup





# Agenda

- [] Big Data - Why, What and How
- [] Spark, Why Spark
- [] RDD
- [] JDK 8 (Functional Programming and Lambda Expressions) - If needed
- [] Transformations and Actions
- [] Caching, Storage Levels
- [] Broadcasts and Accumulators
- [] Spark Internals
- [] Spark UI, Job History Server
- [] SparkSQL - DataFrames
- [] SparkStreaming - DStreams
- [] Spark Performance Tuning
- [] Machine Learning

# What is Spark ?

Spark is a fast, general computation engine  
for large scale data processing  
running on a cluster.

# Spark - History

Matei Zaharia started Spark and later in 2013 was donated to Apache.



Algorithms,  
Machines,  
People



# Spark Advantages

Fault - Tolerance

Data is distributed when it is stored

Bring computation to the data

High level programming framework

Hides the plumbing, we can focus on the code

Caches data in memory

# Spark Advantages

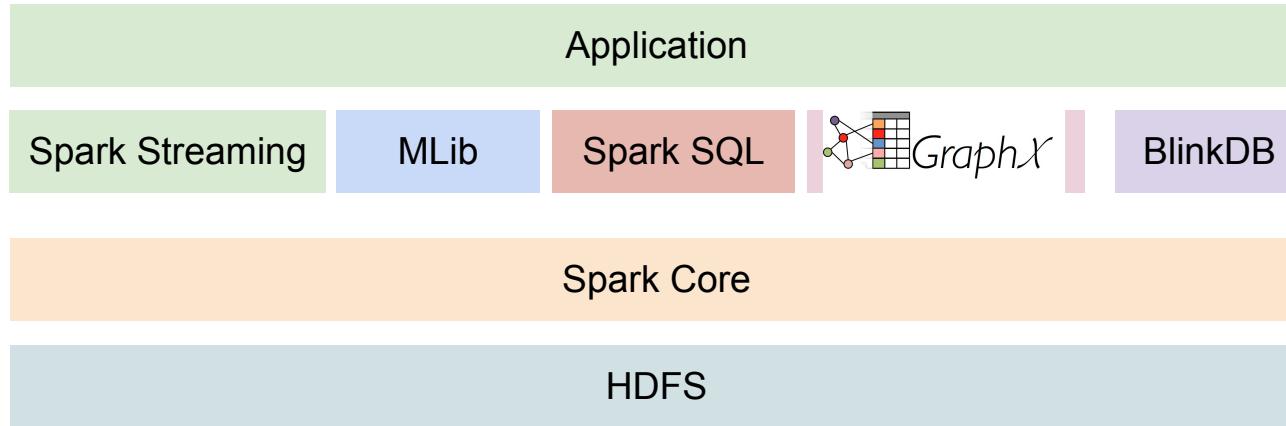
## Lazy computations

Optimize the job before executing

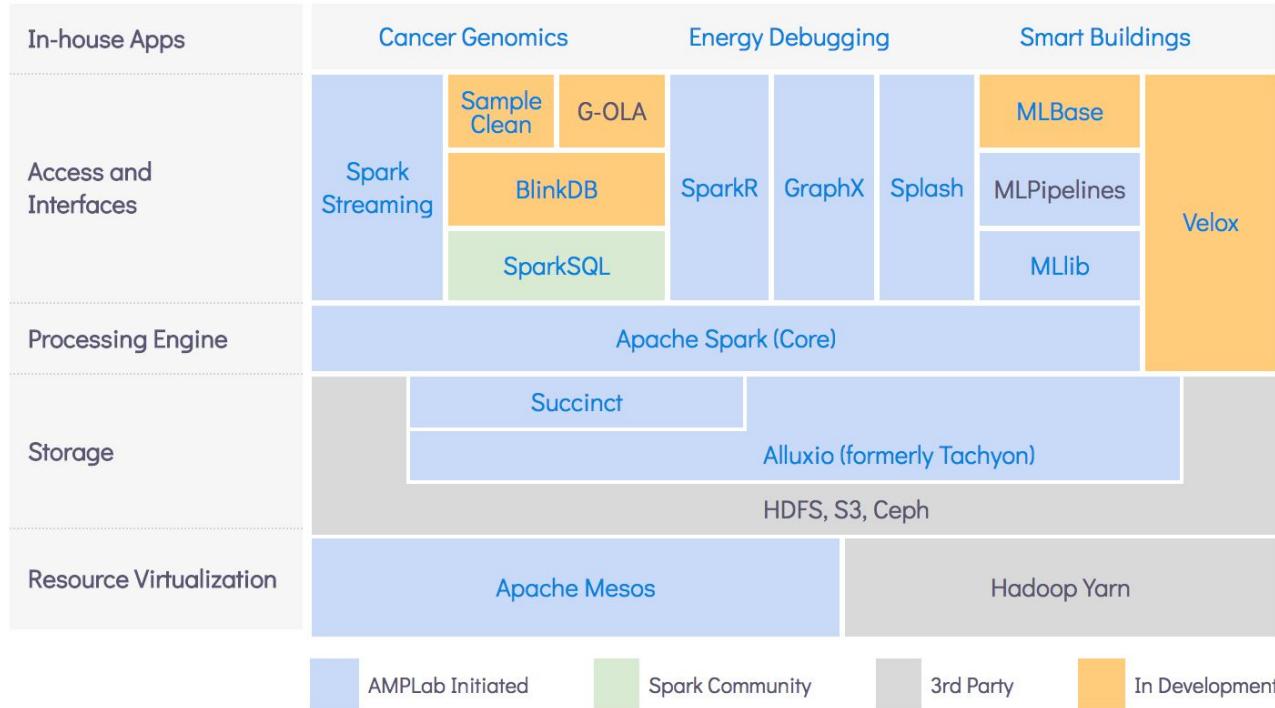
## In memory caching

Scan the hard-disk and then scan the RAM

# BDAS - Berkeley Data Analytics Stack



# BDAS - Berkeley Data Analytics Stack



Source : <https://amplab.cs.berkeley.edu/software/>

# Difference between Storm and Spark



Task Parallel

Data Parallel

Heron is replacing it.

No replacement

Less community active

Active community

# Misconceptions about Spark

It is an in-memory computation engine, it just uses memory for its LRU cache.

It is 100x faster than Hadoop.

Spark's approach to data is completely a novice approach.

View the spark source code.

Writing custom RDD is a good in scala.

To learn the internals of the spark, scanning source code is mandatory.

For Java Developers . . .



# Era of Functional Programming

Functional programming is a programming paradigm (a style of coding)

Spark uses functional programming model heavily to code on the data.

# What is an RDD ?

The Resilient Distributed Dataset (RDD) is the founding data structure of Spark.

An RDD is an abstraction of a collection of data distributed on various nodes.

RDD is a dataset that is distributed and divided into partitions.

It is distributed, immutable, lazily evaluated, type inferred and cacheable.

# RDD's are immutable, distributed and fault-tolerant

RDD's are immutable meaning, any RDD created is computed and given to user. An RDD created is never destroyed.

Distributedness comes from the fact that data is distributed across various partitions which in turn is stored across various nodes.

Since data is available across nodes, there is no possibility of data loss and hence fault tolerant.

# Structure of a Spark Program

1. Create a Spark Config
2. Then create a spark context from the spark config
3. Write one or more transformations.
4. Atlast write only one action.
5. Close the Spark Context.

# Spark Context - sc

```
SparkConf sparkConfig = new SparkConf()  
    .setAppName("ReadCSVFile")  
    .setMaster("local[8]");
```

```
JavaSparkContext javaSparkContext = new JavaSparkContext(sparkConfig);
```

```
javaSparkContext.close();  
(or)  
javaSparkContext.stop();
```

# Spark Context - Read a File

```
    javaSparkContext.textFile("file:///C:/ac/spark/code/sparktraining/data/titanic3.csv")
        .foreach(eachLine -> System.out.println(eachLine));
```

textFile is the transformation, and  
foreach is the action.

We will learn what is a transformation and action a little later. Both are api's in spark but for a different purpose.

# Spark Config

```
SparkConf sparkConfig = new SparkConf()  
    .set("spark.local.dir", "/Users/tester/whitetiger")  
    .setAppName("ReadCSVFile")  
    .setMaster("local[8]");
```

To get rid of the logs printing in the console, put the following code before the Spark Config:

```
import org.apache.log4j.{Level, Logger}  
val rootLogger = Logger.getRootLogger()  
rootLogger.setLevel(Level.ERROR)
```

# Spark Config

To get rid of the logs printing in the console, put the following code before the Spark Config:

```
import org.apache.log4j.{Level, Logger}  
Logger.getLogger("org").setLevel(Level.OFF)
```

# Spark Config

```
SparkConf sparkConfig = new SparkConf()  
    .set("spark.local.dir", "/Users/tester/whitetiger")  
    .setAppName("ReadCSVFile")  
    .setMaster("local[8]");
```

```
sparkConfig.set("spark.driver.host", "localhost")
```

```
sparkConfig.set("spark.driver.allowMultipleContexts", "true")
```

# Spark - pyspark first program



```
from pyspark import SparkContext
```

```
sc = SparkContext()  
lines = sc.parallelize(['first','spark','code'])  
  
for each in lines.take(10):  
    print each
```

```
from pyspark import SparkContext  
from pyspark.conf import SparkConf  
  
config = SparkConf()  
  
config.setAppName("SPARK_READ_FILE_JOB")  
config.setMaster("local[*]")  
sc = SparkContext(conf=config)
```

```
def print_each_line(eachLine):  
    print eachLine  
    return
```

```
textFileRDD = sc.textFile("/home/dharshekthvel/history_1.txt")  
mappedRDD = textFileRDD.map(lambda x : len(x))
```

# pyspark - avoid winutils exception



```
# If you get a winutils.exe exception then add the below line  
# Put the winutils.exe in the bin folder inside D:/ac/winutils/bin
```

```
import os
```

```
os.environ['HADOOP_HOME'] = "D:/ac/winutils/"
```

# Spark Session

SparkSession is the new entry point for spark from Spark 2.0.

SparkSession is the new unified gateway for SQLContext, Streaming Context futuristically.

# Spark Session

```
val sparkSession = SparkSession.builder()
    .appName("SparkJOB")
    .master("local[*]")
    .config("spark.local.dir","/Users/dharshekthvel/ac")
    .getOrCreate()
```

```
val sparkContext = sparkSession.sparkContext
```

```
val file = sparkContext.textFile("/Users/dharshekthvel/ac/auth.csv")
```

```
file.foreach(eachLine => println(eachLine))
```

```
sparkContext.stop()
```

# Spark - Shell

Spark shell is a command line utility which gives spark context (sc) by which any spark API can be evaluated. It is based on REPL (Read-Eval-Print-Loop)

`./spark-shell` which resides in bin directory of the spark installation. This starts in the spark scala REPL.

`./pyspark` which resides in bin directory of the spark installation. This starts in the spark python REPL.

# Spark - Shell

## Scala REPL

## Python REPL

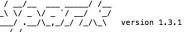
```
15/07/26 13:59:58 INFO sparkEnv: registering mapoutputtracker
15/07/26 13:59:58 INFO SparkEnv: Registering BlockManagerMaster
15/07/26 13:59:58 INFO DiskBlockManager: Created local directory at /var/folders/fs/7n_kjky94b5d0xs_wbkt258c0000gp/T/spark-73aa0b0f-a05f-4941-9430-23cadef75c0
e00
15/07/26 13:59:58 INFO MemoryStore: MemoryStore started with capacity 265.4 MB
15/07/26 13:59:58 INFO HttpFileServer: HTTP File server directory is /var/folders/fs/7n_kjky94b5d0xs_wbkt258c0000gp/T/spark-94d8543c-6bee-4e26-bfa9-c344da126l
b
15/07/26 13:59:58 INFO HttpServer: Starting HTTP Server
15/07/26 13:59:58 INFO Server: jetty-8.y.z-SNAPSHOT
15/07/26 13:59:58 INFO AbstractConnector: Started SocketConnector@0.0.0.0:59160
15/07/26 13:59:58 INFO Utils: Successfully started service "HTTP file server" on port 59160.
15/07/26 13:59:58 INFO SparkEnv: Registering OutputCommitCoordinator
15/07/26 13:59:58 INFO Server: jetty-8.y.z-SNAPSHOT
15/07/26 13:59:58 INFO AbstractConnector: Started SelectChannelConnector@0.0.0.0:4040
15/07/26 13:59:58 INFO Utils: Successfully started service "SparkUI" on port 4040.
15/07/26 13:59:58 INFO SparkUI: Started SparkUI at http://localhost:4040
15/07/26 13:59:58 INFO Executor: Submitting executor ID driver on host localhost
15/07/26 13:59:58 INFO Utils: Connecting to HeartbeatReceiver: akka.tcp://sparkDriver@localhost:59159/user/HeartbeatReceiver
15/07/26 13:59:59 INFO NettyBlockTransferService: Server created on 59161
15/07/26 13:59:59 INFO BlockManagerMasterActor: Trying to register BlockManager
15/07/26 13:59:59 INFO BlockManagerMasterActor: Registering block manager localhost:59161 with 265.4 MB RAM, BlockManagerId(<driver>, localhost, 59161)
15/07/26 13:59:59 INFO BlockManagerMaster: Registered BlockManager
Welcome to
```



version 1.3.1

```
Using Python version 2.7.5 (default, Mar 9 2014 22:15:05)
SparkContext available as sc, HiveContext available as sqlContext.
>>> 
```

\$ bin/pyspark

```
15/07/26 13:59:26 INFO HttpServer: Starting HTTP Server
15/07/26 13:59:26 INFO Server: jetty-8.y.z-SNAPSHOT
15/07/26 13:59:26 INFO AbstractConnector: Started SocketConnector@0.0.0.0:59158
15/07/26 13:59:26 INFO Utils: Successfully started service "HTTP class server" on port 59158.
Welcome to

version 1.3.1

Using Scala version 2.10.4 (Java HotSpot(TM) 64-Bit Server VM, Java 1.7.0_45)
Type in expressions to have them evaluated.
Type :help for more information.
15/07/26 13:59:29 WARN Utils: Your hostname, localhost resolves to a loopback address: 127.0.0.1, but we couldn't find any external IP address!
15/07/26 13:59:29 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
15/07/26 13:59:29 INFO SparkContext: Running Spark version 1.3.1
15/07/26 13:59:29 INFO SparkContext: Using org.apache.spark.repl.SparkREPL as the repl
15/07/26 13:59:29 INFO SecurityManager: Changing view acls acts to: tester
15/07/26 13:59:29 INFO SecurityManager: authentication disabled; ui acls disabled; users with view permissions: Set(tester); users with modify : Set()
15/07/26 13:59:29 INFO SecurityManager: SecurityManager: log4jRootCategory = INFO
15/07/26 13:59:29 INFO Remoting: Starting Remoting
15/07/26 13:59:29 INFO Remoting: Remoting started; listening on addresses :[akka.tcp://sparkDriver@localhost:59151]
15/07/26 13:59:29 INFO Remoting: Remoting successfully started service [sparkDriver] on port 59151.
15/07/26 13:59:29 INFO SparkContext: Registered SparkContext
15/07/26 13:59:29 INFO SparkEnv: Registering BlockManagerMaster
15/07/26 13:59:29 INFO DiskBlockManager: Created local directory at /var/folders/fs/7n_kjky94b5d0xs_wbkt258c0000gp/T/spark-87809d54-741c-463f-9365-71e88eff04ed/l
b01
15/07/26 13:59:29 INFO MemoryStore: MemoryStore started with capacity 265.4 MB
15/07/26 13:59:29 INFO HttpFileServer: HTTP File server directory is /var/folders/fs/7n_kjky94b5d0xs_wbkt258c0000gp/T/spark-961c5d98-f57c-42f5-9269-437b07c400ee,
2
15/07/26 13:59:29 INFO HttpServer: Starting HTTP Server
15/07/26 13:59:29 INFO Server: jetty-8.y.z-SNAPSHOT
15/07/26 13:59:29 INFO AbstractConnector: Started SelectChannelConnector@0.0.0.0:59152
15/07/26 13:59:29 INFO Utils: Successfully started service "HTTP file server" on port 59152.
15/07/26 13:59:29 INFO SparkEnv: Registering OutputCommitCoordinator
15/07/26 13:59:29 INFO AbstractConnector: Started SelectChannelConnector@0.0.0.0:4840
15/07/26 13:59:29 INFO Utils: Successfully started service "SparkUI" on port 4840.
15/07/26 13:59:29 INFO Executor: Starting executor ID driver on host localhost
15/07/26 13:59:29 INFO Executor: Starting executor ID driver on host localhost
15/07/26 13:59:29 INFO Executor: Using REPL class URL: http://127.0.0.1:59158
15/07/26 13:59:29 INFO Executor: Registered executor ID driver on host localhost
15/07/26 13:59:29 INFO BlockManagerMasterActor: Trying to register BlockManager
15/07/26 13:59:29 INFO BlockManagerMaster: Registered BlockManager [localhost:59153 with 265.4 MB RAM, BlockManagerId(<driver>, localhost, 59153)
15/07/26 13:59:29 INFO BlockManagerMaster: Registered BlockManager
15/07/26 13:59:30 INFO SparkILoop: Created spark context.
SparkContext available as sc
15/07/26 13:59:30 INFO SparkILoop: Created sql context (with Hive support)..
```

SQL context available as sqlContext.

scala>

# Spark - Shell

To quit from the shell

`scala> :quit`

# Read all files inside a directory

Spark can read the whole content of the directory

For that, **wholeTextFiles()** API of the spark context is used.

After which, the filename and the file content can be accessed seperately.

# Read all files inside a directory

```
val sparkContext = new SparkContext(config)

val allFilesRDD = sparkContext.wholeTextFiles("file:///D:/ac/data");
allFilesRDD.foreach(x => println(x._1 + x._2))
```

x.\_1 gives the filename

x.\_2 gives the file content

SparkReadFile.java

# Spark Running Modes

## local mode

- local : Runs spark locally without any parallelism
- local[k] : Runs spark locally with k threads. Usually k is the number of cores of the machine.
- local[\*] : Runs spark locally with many threads.

## Spark Cluster

```
SparkConf sparkConfig = new SparkConf()  
    .setAppName("ReadLogFile")  
    .setMaster("spark://Apples-MacBook-Pro.local:7077");
```

eg: spark://ec2-52-24-58-38.us-west-2.compute.amazonaws.com:7077

## Yarn (Yet Another Resource Negotiator)

## Mesos

# Simple map transformation

```
SparkConf sparkConfig = new SparkConf()
    .set("spark.local.dir", "/Users/tester/whitetiger")
    .setAppName("ReadCSVFile")
    .setMaster("local[8]");

JavaSparkContext javaSparkContext = new JavaSparkContext(sparkConfig);

JavaRDD<String> rdd = javaSparkContext.textFile("file:///Users/tester/ac/entitlement_view.csv");

JavaRDD<LicenseCountDataObject> lcRDD = rdd.map(new Function<String, LicenseCountDataObject>() {

    @Override
    public LicenseCountDataObject call(String input) throws Exception {
        String[] splitColumns = input.split(",");
        LicenseCountDataObject lcObject = new LicenseCountDataObject();
        lcObject.setId(splitColumns[0]);
        lcObject.setLicenseCount(splitColumns[3]);
        return lcObject;
    }
});

lcRDD.foreach(z -> System.out.println(z.getId()));
javaSparkContext.stop();
```

# Simple map transformation

```
class UNBeanConverterFunction implements Function<String,  
UNDataBean> {
```

```
/**  
 *  
 */
```

```
private static final long serialVersionUID = 1L;
```

```
@Override
```

```
public UNDataBean call(String input) throws Exception {
```

```
String[] splitColumns = input.split(",");
```

```
UNDataBean undata = new UNDataBean();
```

```
undata.setCountry(splitColumns[0]);
```

```
undata.setCommodityCode(splitColumns[2]);
```

```
undata.setCommodity(splitColumns[3]);
```

```
return undata;
```

```
}
```

JavaRDD<UNDataBean> undataBeanconverterRDD  
= rdd.map(new UNBeanConverterFunction());

# Simple map transformation

*The same functional program as a part of lambda expression.*

```
JavaRDD<UNDataBean> undataBeanRDD = rdd.map((each) -> {  
    String[] splitColumns = each.split(",");  
  
    UNDataBean undata = new UNDataBean();  
    undata.setCountry(splitColumns[0]);  
    undata.setCommodityCode(splitColumns[2]);  
    undata.setCommodity(splitColumns[3]);  
    return undata;  
});
```

# Simple map transformation - Scala

```
object SparkScalaMapTransformation {  
  
  def main(args: Array[String]) {  
  
    val config = new SparkConf().setAppName("ReadCSVFile").setMaster("local[8]")  
    val sc = new SparkContext(config)  
  
    val titanicText = sc.textFile("file:///Users/apple/titanic3.csv")  
  
    titanicText.map(new TitanicBeanConverter().titanicBeanConverter)  
      .foreach(x => println(x.name))  
  }  
  
  class TitanicBeanConverter extends Serializable {  
  
    def titanicBeanConverter(line : String) : TitanicBean = {  
      val titanicBean = new TitanicBean()  
      titanicBean.name = line.split(",")(2)  
  
      return titanicBean  
    }  
  }  
}
```

```
class TitanicBean {  
  var className = ""  
  var survived = ""  
  var name = ""  
  var sex = ""  
  var age = ""  
  var sibsp = ""  
  var parch = ""  
  var ticket = ""  
  var fare = ""  
  var cabin = ""  
  var embarked = ""  
  var boat = ""  
  var body = ""  
  var destination = ""  
}
```

# Simple map transformation - pyspark

```
from pyspark import SparkContext
from pyspark.conf import SparkConf

config = SparkConf()

config.setAppName("SPARK_READ_FILE_JOB")
config.setMaster("local[*]")
sc = SparkContext(conf=config)

def print_each_line(eachLine):
    print eachLine
    return

textFileRDD = sc.textFile("/home/dharshekthvel/history_1.txt")
mappedRDD = textFileRDD.map(lambda x : len(x))

mappedRDD.foreach(print_each_line)
```

# Lazy Evaluation of RDD

RDD's are lazily evaluated, emphasizing the fact that RDD's will get evaluated by spark only when an action is encountered.

If you see a log as shown below, then it means that Spark has done a action on the RDD.

```
15/07/26 12:08:08 INFO HadoopRDD: Input split: file:/Users/tester/ac/entitlement_view.csv:106634+106635
```

```
15/07/26 12:08:08 INFO HadoopRDD: Input split: file:/Users/tester/ac/entitlement_view.csv:0+106634
```

# Different ways of creating an RDD

## [] Loading from an external dataset

```
javaSparkContext.textFile("file:///Users/apple/titanic3.csv")
    .foreach(z -> System.out.println(z));

javaSparkContext.textFile("hdfs://localhost:9000/data/titanic3.csv")
    .foreach(z -> System.out.println(z));
```

## [] Parallelize from the Spark Context by giving an input list

```
JavaSparkContext javaSparkContext = new JavaSparkContext(sparkConfig);
List<LatLong> locationList = new ListSource().retrieveList();
JavaRDD<LatLong> locRDD = javaSparkContext.parallelize(locationList);
```

## [] Creating an RDD from another RDD.

# Different ways of creating an RDD

- [] Use makeRDD of sparkContext

```
val supportVectorList = List(  
    SVMSupportVector("mesh", "12", 1),  
    SVMSupportVector("maille", "15", 2),  
    SVMSupportVector("referential", "33", 2))  
  
val rdd = sparkContext.makeRDD(supportVectorList)
```

- [] A Range RDD can be created from SparkContext

```
val rangeRDD = sparkContext.range(1, 100, 2, 10)
```

1 to 100 by 2, with 10 partitions

# Different ways of creating an RDD

- [] Use hadoopFile() to build your own custom format

```
val.hadoopFileRDD = sparkContext.hadoopFile[LongWritable,Text,TextInputFormat]("file:///home/dharshekthvel/ac/docs/file.txt")
.map { case (x,y) => y.toString }
```

```
hadoopFileRDD.foreach(x => println(x))
```

# Multiple spark context

```
val sparkSession = SparkSession .builder()
    .appName("SparkJOB")
    .master("local")
    .config("spark.driver.allowMultipleContexts", "true")
    .getOrCreate
```

```
val sparkContext = sparkSession.sparkContext
```

```
val sparkSessionOne = SparkSession .builder()
    .appName("SparkJOBOne")
    .master("local")
    .config("spark.driver.allowMultipleContexts", "true")
    .getOrCreate
```

```
val sparkContextOne = sparkSession.sparkContext
```

# Dependency Handling with libraries and files

## [ ] Adding a File

```
sparkContext.addFile("path")
SparkFiles.get("path")
```

Downloads the file present at path to every node. The file can be accessed in the worker node via `SparkFiles.get("path")`

## [ ] Adding a Jar

```
sparkContext.addJar("path");
```

Adds the file at path as a JAR dependency for all tasks executed in the Spark Context.

# Spark 2.0 Features

DataFrames is the type alias for Dataset. Path is on for the unification of DataSets and DataFrames.

More performance oriented Accumulator API. Accumulator supports primitives.

Spark 2.0 has the Tungsten engine for performance optimization. The tungsten optimization is called as the "whole-stage code generation".

The tungsten code generation can be turned off using `SparkSession.config("spark.sqlcodegen.wholeStage", "false")`

Catalyst optimizer for general query optimization is also introduced.

Structured streaming has been introduced.

# Spark 2.0 Features

High optimization is done in Spark 2.0. As said, Spark 2.0 ships with a Tungsten engine. The engine is built upon the idea of modern compilers and MPP databases and applies them to data processing.

# maven dependencies

```
<dependency>
  <groupId>org.apache.spark</groupId>
  <artifactId>spark-core_2.10</artifactId>
  <version>1.4.0</version>
</dependency>

<dependency>
  <groupId>org.apache.spark</groupId>
  <artifactId>spark-streaming_2.10</artifactId>
  <version>1.4.0</version>
</dependency>

<dependency>
  <groupId>org.apache.spark</groupId>
  <artifactId>spark-sql_2.10</artifactId>
  <version>1.4.0</version>
</dependency>
```

**Spark Core**

**Spark Streaming**

**SparkSQL**

```
<dependency>
  <groupId>org.apache.spark</groupId>
  <artifactId>spark-mllib_2.10</artifactId>
  <version>1.4.0</version>
</dependency>

<dependency>
  <groupId>org.apache.spark</groupId>
  <artifactId>spark-streaming-twitter_2.10</artifactId>
  <version>1.5.0</version>
</dependency>

<dependency>
  <groupId>org.apache.spark</groupId>
  <artifactId>spark-streaming-kafka_2.10</artifactId>
  <version>1.4.0</version>
</dependency>
```

**Spark Machine Learning**

**Twitter Streaming**

**Kafka Streaming**

# sbt dependencies

```
name := "sparkscala9"

version := "1.0"

scalaVersion := "2.11.7"

libraryDependencies ++= Seq(
    "org.apache.spark" %% "spark-core" % "1.6.0",
    "org.apache.spark" %% "spark-sql" % "1.6.0",
    "org.apache.spark" %% "spark-streaming" % "1.6.0",
    "org.apache.spark" %% "spark-streaming-kafka" % "1.6.0"
)

resolvers += "Akka Repository" at "http://repo.akka.io/releases/"
```

# Spark

```
name := "sparkscala9"

version := "1.0"

scalaVersion := "2.11.7"

libraryDependencies += Seq(
    "org.apache.spark" %% "spark-core" % "1.6.0",
    "org.apache.spark" %% "spark-sql" % "1.6.0",
    "org.apache.spark" %% "spark-streaming" % "1.6.0",
    "org.apache.spark" %% "spark-streaming-kafka" % "1.6.0"
)

resolvers += "Akka Repository" at "http://repo.akka.io/releases/"
```

# Spark Standalone Cluster

[1] Start the master

```
$ /sbin/start-master.sh
```

[2] Start the worker

```
$ /bin/spark-class
```

```
org.apache.spark.deploy.worker.Worker  
spark://SCHMAC-TESTER-4.local:7077
```

# Start and Stop a Spark standalone cluster (Masters and Workers)

To start a master

```
$ /sbin/start-master.sh
```

After starting the master the below are the successful logs after the start of the master.

```
15/07/24 12:41:43 INFO Remoting: Starting remoting
```

```
15/07/24 12:41:44 INFO Remoting: Remoting started; listening on addresses  
:[akka.tcp://sparkMaster@ec2-52-24-58-38.us-west-2.compute.amazonaws.com:7077]
```

```
15/07/24 12:41:44 INFO Utils: Successfully started service 'sparkMaster' on port 7077.
```

```
15/07/24 12:41:44 INFO Utils: Successfully started service on port 6066.
```

```
15/07/24 12:41:44 INFO StandaloneRestServer: Started REST server for submitting applications on port  
6066
```

# Start and Stop a Spark standalone cluster (Masters and Workers)

To start a worker

\$ /bin/spark-class

org.apache.spark.deploy.worker.Worker spark://SCHMAC-TESTER-4.local:7077

\$ cd sbin/  
\$ ./start-slave.sh spark://Dharshekths-MacBook-Pro.local:7077

\$ ./spark-class org.apache.spark.deploy.worker.Worker  
spark://ec2-52-24-58-38.us-west-2.compute.amazonaws.com:7077

The below would be the successful log messages when the worker is added to the master.

15/07/24 13:09:04 INFO Slf4jLogger: Slf4jLogger started

15/07/24 13:09:04 INFO Remoting: Starting remoting

15/07/24 13:09:05 INFO Remoting: Remoting started; listening on addresses :[akka.tcp://sparkWorker@172.31.40.73:47537]

15/07/24 13:09:05 INFO Utils: Successfully started service 'sparkWorker' on port 47537.

15/07/24 13:09:05 INFO Worker: Starting Spark worker 172.31.40.73:47537 with 2 cores, 6.3 GB RAM

15/07/24 13:09:05 INFO Worker: Running Spark version 1.4.0

15/07/24 13:09:05 INFO Worker: Spark home: /home/ubuntu/inferneon/spark14

15/07/24 13:09:05 INFO Utils: Successfully started service 'WorkerUI' on port 8081.

15/07/24 13:09:05 INFO WorkerWebUI: Started WorkerWebUI at http://172.31.40.73:8081

15/07/24 13:09:05 INFO Worker: Connecting to akka.tcp://sparkMaster@ec2-52-24-58-38.us-west-2.compute.amazonaws.com:7077/user/Master...

15/07/24 13:09:05 INFO Worker: Successfully registered with master spark://ec2-52-24-58-38.us-west-2.compute.amazonaws.com:7077

# Start and Stop a Spark standalone cluster (Masters and Workers)

To stop a master

```
$ /sbin/stop-master.sh
```

# Start and Stop a Spark standalone cluster (Masters and Workers)

To start a cluster, you can also do

**\$ /sbin/start-all.sh**

The above will start master and all nodes listed in conf/slaves config file.

To stop a cluster,

**\$ /sbin/stop-all.sh**

# Spark Submit

To run spark

```
/spark_installation/bin/spark-submit
```

```
./spark-submit
```

```
--name THIS_IS_MY_APP_NAME  
--master local[*]  
--deploy-mode client          (client or cluster)  
--driver-memory 1G            (Default: 512M)  
--class com.dmac.analytics.spark.SparkSubmitJob  
~/sparktrainingjava-1.0-SNAPSHOT.jar
```

```
./spark-submit
```

```
--properties-file ~/spark_qa.properties  
--class com.dmac.analytics.spark.SparkInternals  
~/sparktrainingjava-1.0-SNAPSHOT.jar
```

```
./spark-submit --help
```

chinnasamyad@gmail.com

Check out help for more options

# Spark Submit

spark\_qa.properties

spark.master	local
spark.app.name	MY OWN APP NAME
spark.ui.port	1234

The screenshot shows the Spark 1.4.0 web interface with the 'Environment' tab selected. The 'Runtime Information' section lists Java Home, Java Version, and Scala Version. Below it, a box highlights the 'Spark Properties' section, which contains entries for spark.app.id, spark.app.name, spark.driver.host, spark.driver.port, spark.executor.id, spark.externalBlockStore.folderName, spark.fileserver.uri, spark.jars, spark.master, spark.scheduler.mode, and spark.ui.port. The 'spark.ui.port' entry in this list is highlighted with a blue box and has a blue arrow pointing from it to the 'spark.ui.port' entry in the 'spark\_qa.properties' file shown on the left.

Name
Java Home
Java Version
Scala Version

Name
spark.app.id
spark.app.name
spark.driver.host
spark.driver.port
spark.executor.id
spark.externalBlockStore.folderName
spark.fileserver.uri
spark.jars
spark.master
spark.scheduler.mode
spark.ui.port

Name
SPARK_SUBMIT
awt.toolkit

# Spark UI

local mode                    <http://localhost:4040/>

./spark-shell

standalone mode              <http://bdsa:8080/>

Starts the web-ui port in 9999 and spark in 8088

./start-master.sh --port 8088 --webui-port 9999

Default is 7077 and 8080

# Spark UI

← → ⌂ ⓘ dharshkths-macbook-pro.local:8080/#running-app ⋮



## Spark Master at spark://Dharshkths-MacBook-Pro.local:7077

**URL:** spark://Dharshkths-MacBook-Pro.local:7077

**REST URL:** spark://Dharshkths-MacBook-Pro.local:6066 (cluster mode)

**Alive Workers:** 1

**Cores in use:** 8 Total, 0 Used

**Memory in use:** 15.0 GB Total, 0.0 B Used

**Applications:** 0 [Running](#), 0 [Completed](#)

**Drivers:** 0 Running, 0 Completed

**Status:** ALIVE

### Workers

Worker Id	Address	State	Cores	Memory
<a href="#">worker-20170203104740-192.168.0.3-51005</a>	192.168.0.3:51005	ALIVE	8 (0 Used)	15.0 GB (0.0 B Used)

### Running Applications

Application ID	Name	Cores	Memory per Node	Submitted Time	User	State	Duration

### Completed Applications

Application ID	Name	Cores	Memory per Node	Submitted Time	User	State	Duration

# Spark UI

Spark 1.4.0

Jobs Stages Storage Environment Executors Streaming

ReadTwitter application UI

## Stages for All Jobs

Active Stages: 1

Completed Stages: 15

### Active Stages (1)

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
0	start at TwitterStreaming.java:49	+details (kill)	2015/09/18 18:16:51	2.1 min	0/1			

### Completed Stages (15)

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
15	print at TwitterStreaming.java:47	+details	2015/09/18 18:18:50	3 ms	1/1			
14	print at TwitterStreaming.java:47	+details	2015/09/18 18:18:40	3 ms	1/1			
13	print at TwitterStreaming.java:47	+details	2015/09/18 18:18:30	3 ms	1/1			
12	print at TwitterStreaming.java:47	+details	2015/09/18 18:18:20	2 ms	1/1			
11	print at TwitterStreaming.java:47	+details	2015/09/18 18:18:10	4 ms	1/1			
10	print at TwitterStreaming.java:47	+details	2015/09/18 18:18:00	4 ms	1/1			
9	print at TwitterStreaming.java:47	+details	2015/09/18 18:17:50	3 ms	1/1			
8	print at TwitterStreaming.java:47	+details	2015/09/18 18:17:40	4 ms	1/1			
7	print at TwitterStreaming.java:47	+details	2015/09/18 18:17:30	3 ms	1/1			
6	print at TwitterStreaming.java:47	+details	2015/09/18 18:17:30	3 ms	1/1			
5	print at TwitterStreaming.java:47	+details	2015/09/18 18:17:20	6 ms	4/4			
4	print at TwitterStreaming.java:47	+details	2015/09/18 18:17:20	3 ms	1/1			
3	print at TwitterStreaming.java:47	+details	2015/09/18 18:17:10	6 ms	1/1			
2	print at TwitterStreaming.java:47	+details	2015/09/18 18:17:00	15 ms	4/4			
1	print at TwitterStreaming.java:47	+details	2015/09/18 18:17:00	18 ms	1/1			

# Spark UI



Jobs Stages Storage Environment

Executors

SQL

Spark shell application UI

## Executors

### Summary

	RDD Blocks	Storage Memory	Disk Used	Cores	Active Tasks	Failed Tasks	Complete Tasks	Total Tasks	Task Time (GC Time)	Input	Shuffle Read	Shuffle Write
Active(1)	12	280.9 KB / 384.1 MB	0.0 B	8	0	0	170	170	21 s (1 s)	7.4 GB	0.0 B	0.0 B
Dead(0)	0	0.0 B / 0.0 B	0.0 B	0	0	0	0	0	0 ms (0 ms)	0.0 B	0.0 B	0.0 B
Total(1)	12	280.9 KB / 384.1 MB	0.0 B	8	0	0	170	170	21 s (1 s)	7.4 GB	0.0 B	0.0 B

### Executors

Show 20 entries

Search:

Executor ID	Address	Status	RDD Blocks	Storage Memory	Disk Used	Cores	Active Tasks	Failed Tasks	Complete Tasks	Total Tasks	Task Time (GC Time)	Input	Shuffle Read	Shuffle Write	Thread Dump
driver	192.168.0.3:49818	Active	12	280.9 KB / 384.1 MB	0.0 B	8	0	0	170	170	21 s (1 s)	7.4 GB	0.0 B	0.0 B	<a href="#">Thread Dump</a>

Showing 1 to 1 of 1 entries

[Previous](#) [1](#) [Next](#)

# Spark - REST outbounds

<http://localhost:4040/api/v1> - Interfaces are bounded out at /api/v1

<http://localhost:4040/api/v1/applications>

```
[ {  
    "id" : "local-1460702558532",  
    "name" : "ReadLogFile",  
    "attempts" : [ {  
        "startTime" : "2016-04-15T06:42:32.743GMT",  
        "endTime" : "1969-12-31T23:59:59.999GMT",  
        "sparkUser" : "",  
        "completed" : false  
    } ]  
} ]
```

# Spark - REST outbound

/applications

---

/applications/[app-id]/jobs

---

/applications/[app-id]/jobs/[job-id]

---

/applications/[app-id]/stages

---

/applications/[app-id]/stages/[stage-id]

---

/applications/[app-id]/stages/[stage-id]/[stage-attempt-id]

---

/applications/[app-id]/stages/[stage-id]/[stage-attempt-id]/taskSummary

---

/applications/[app-id]/stages/[stage-id]/[stage-attempt-id]/taskList

---

/applications/[app-id]/executors

---

/applications/[app-id]/storage/rdd

---

/applications/[app-id]/**storage**/rdd/[rdd-id]

---

/applications/[app-id]/logs

# Partitions - The core of RDD

An RDD is stored in the memory as partition.

Partition is the logical division of data and helps in parallelism.

# Partitions - The core of RDD

The number of partitions can be obtained from `rdd.partitions.length`

```
val textFile = sparkContext.textFile("file:///D:/ac/data/sample.csv")
```

```
println(textFile.partitions.length)
```

or

```
textFile.getNumPartitions
```

# Transformations and Actions

Transformation and Actions are the two types of operations supported by the RDD.

Transformation returns a new RDD

map(), filter(), textFile(), distinct(), flatMap(), mapPartitions(), mapPartitionsWithIndex(), sample(), union(), intersection(), groupByKey(), reduceByKey(), aggregateByKey(), sortByKey(), join(), cogroup(), cartesian(), pipe(), coalesce(), repartition(), repartitionAndSortWithinPartitions().

Actions return a result to the driver program

count(), first(), collect(), foreach(), reduce(), count(), take(), takeSample(), takeOrdered(), saveAsTextFile(), saveAsSequenceFile(), saveAsObjectFile(), countByKey().

# map and flatMap transformation

map transformation passes each element to the function and returns a new RDD.

flatMap transformation flattens the data to an iterable data set.

Used when making a dataset flatten on nested datasets.

# map and flatMap transformation

```
JavaRDD<String> rdd = javaSparkContext.textFile("file:///Users/tester/ac/entitlement_view.csv");
JavaRDD<LicenseCountDataObject> lcRDD = rdd.map(new Function<String, LicenseCountDataObject>() {
    @Override
    public LicenseCountDataObject call(String input) throws Exception {
        String[] splitColumns = input.split(",");
        LicenseCountDataObject lcObject = new LicenseCountDataObject();
        lcObject.setId(splitColumns[0]);
        lcObject.setLicenseCount(splitColumns[3]);
        return lcObject;
    }
});
```

```
JavaRDD<String> textRDD = javaSparkContext.textFile("file:///Users/apple/simple_text_file.txt");
JavaRDD<String> wordsFlattenedRDD = textRDD.flatMap(new FlatMapFunction<String, String>() {

    @Override
    public Iterable<String> call(String input) throws Exception {
        return Arrays.asList(input.split(" "));
    }
});
```

# map transformation - Scala

```
val config = new SparkConf()
config.setAppName("Transformations").setMaster("local[4]")

val sc = new SparkContext(config)

val uidRDD = sc.textFile("D:\\ac\\data\\legacy_uid.csv")
    .map(eachLine => Uid(eachLine))

case class Uid(uidNumber : String)
```

SparkTransformations.scala

# distinct() transformation

```
JavaRDD<String> locRDD = javaSparkContext.parallelize(locationList);
```

```
JavaRDD<String> distinctRDD = locRDD.distinct();
```

```
distinctRDD.foreach((outParam) -> System.out.println(outParam));
```

# cartesian and pipe transformation

A cartesian operation returns a PairRDD of the two types of rdd's that has to be combined.

The pipe operation executes a linux shell command. The output of the linux command is obtained as a JavaRDD<String>.

# cartesian and pipe

```
JavaPairRDD<String, String> countryAndCommodity = countryRDD.cartesian(commodityRDD);  
countryAndCommodity.foreach(param -> System.out.println(param._1 + " " + param._2));
```

```
JavaRDD<String> rdd = javaSparkContext.textFile("file:///Users/apple/undata1.csv");  
rdd.pipe("ls -l").foreach(param -> System.out.println(param));
```

# sample

# transformation

```
javaSparkContext.textFile("file:///Users/apple/titanic3.csv")
    .sample(false, 0.1)
    .foreach(param -> System.out.println(param));
```

false/true - tends to withReplacement factor.

When a population element can be selected more than one time, we are sampling with replacement.

When a population element can be selected only one time, we are sampling without replacement.

# sample

# transformation

withReplacementFactor it uses **Poisson Sampler Algorithm.**

withoutReplacementFactor it uses **Bernoulli Sampler Algorithm.**

sample() returns a PartitionwiseSampledRDD as the result.

# union and intersection transformation

```
JavaRDD<String> indiaRDD = javaSparkContext.textFile("file:///Users/apple/undata1.csv")
    .filter(param -> param.split(",")[0].toString().equals("\\"India\"));
```

```
JavaRDD<String> hungaryRDD = javaSparkContext.textFile("file:///Users/apple/undata1.csv")
    .filter(param -> param.split(",")[0].toString().equals("\\"Hungary\"));
```

```
indiaRDD.union(hungaryRDD).foreach(param -> System.out.println(param));
```

```
JavaRDD<String> indiaRDDCode = javaSparkContext.textFile("file:///Users/apple/undata1.csv")
    .map(param -> param.split(",")[2].toString());
```

```
JavaRDD<String> hungaryRDDCode = javaSparkContext.textFile("file:///Users/apple/undata1.csv")
    .map(param -> param.split(",")[2].toString());
```

```
indiaRDDCode.intersection(hungaryRDDCode).foreach(param -> System.out.println(param));
```

# repartition() and coalesce() transformation

repartition: Reshuffles the data in the RDD to create more or fewer(less) number of partitions.

coalesce(n): Decrease the number of partitions to n.

After filtering a large dataset then coalesce may be useful to reduce the number of partitions. The difference between a repartition and coalesce is that coalesce avoids a full shuffle.

If you are decreasing the number of partitions, then use coalesce. If you are increasing the number of partitions then use repartition.

# repartition() and coalesce() transformation

```
JavaRDD<String> rdd = javaSparkContext.textFile("file:///Users/apple/undata1.csv");
    JavaRDD<String> countryRDD = rdd.map(new Function<String, String>() {

        @Override
        public String call(String input) throws Exception {

            String[] splitColumns = input.split(",");
            return splitColumns[0];

        }
    });
}
```

```
JavaRDD<String> repartitionedRDD = countryRDD.repartition(100);
```

```
JavaRDD<String> coalesceRDD = countryRDD.coalesce(5);
```

# repartition() and coalesce() transformation

```
def repartition(numPartition) {  
    coalesce(numPartitions, shuffle = true)  
}
```

```
def coalesce(numPartitions: Int, shuffle: Boolean = false)
```

Internally repartition calls in coalesce with a shuffle true.

# zip() transformation

```
JavaSparkContext javaSparkContext = new JavaSparkContext(sparkConfig);

JavaRDD<String> first = javaSparkContext.textFile("file:///Users/apple/titanic3.csv");
JavaRDD<String> second = javaSparkContext.textFile("file:///Users/apple/titanic3.csv");

JavaPairRDD<String, String> zippedRDD = first.zip(second);
zippedRDD.foreach(param -> System.out.println(param._1 + " *** " + param._2));
```

# zip() transformation

```
3,0,"Lockyer, Mr. Edward","male",,0,0,"1222",7.8792,, "S",,"153", *** 3,0,"Lockyer, Mr.  
Edward","male",,0,0,"1222",7.8792,, "S",,"153",  
3,0,"Lovell, Mr. John Hall (""Henry""),"male",20.5,0,0,"A/5 21173",7.2500,, "S",,, *** 3,0,"Lovell, Mr. John Hall  
(""Henry""),"male",20.5,0,0,"A/5 21173",7.2500,, "S",,,  
3,1,"Lulic, Mr. Nikola","male",27,0,0,"315098",8.6625,, "S",,"15",, *** 3,1,"Lulic, Mr.  
Nikola","male",27,0,0,"315098",8.6625,, "S",,"15",,  
3,0,"Lundahl, Mr. Johan Svensson","male",51,0,0,"347743",7.0542,, "S",,, *** 3,0,"Lundahl, Mr. Johan  
Svensson","male",51,0,0,"347743",7.0542,, "S",,,  
3,1,"Lundin, Miss. Olga Elida","female",23,0,0,"347469",7.8542,, "S",,"10",, *** 3,1,"Lundin, Miss. Olga  
Elida","female",23,0,0,"347469",7.8542,, "S",,"10",,  
3,1,"Lundstrom, Mr. Thure Edvin","male",32,0,0,"350403",7.5792,, "S",,"15",, *** 3,1,"Lundstrom, Mr. Thure  
Edvin","male",32,0,0,"350403",7.5792,, "S",,"15",,  
3,0,"Lyntakoff, Mr. Stanko","male",,0,0,"349235",7.8958,, "S",,, *** 3,0,"Lyntakoff, Mr. Stanko","male",,0,0,"349235",7.8958,, "S",,,
```

# zip() transformation

```
3,0,"Lockyer, Mr. Edward","male",,0,0,"1222",7.8792,, "S",,"153", *** 3,0,"Lockyer, Mr.  
Edward","male",,0,0,"1222",7.8792,, "S",,"153",  
3,0,"Lovell, Mr. John Hall (""Henry""),"male",20.5,0,0,"A/5 21173",7.2500,, "S",,, *** 3,0,"Lovell, Mr. John Hall  
(""Henry""),"male",20.5,0,0,"A/5 21173",7.2500,, "S",,,  
3,1,"Lulic, Mr. Nikola","male",27,0,0,"315098",8.6625,, "S",,"15",, *** 3,1,"Lulic, Mr.  
Nikola","male",27,0,0,"315098",8.6625,, "S",,"15",,  
3,0,"Lundahl, Mr. Johan Svensson","male",51,0,0,"347743",7.0542,, "S",,, *** 3,0,"Lundahl, Mr. Johan  
Svensson","male",51,0,0,"347743",7.0542,, "S",,,  
3,1,"Lundin, Miss. Olga Elida","female",23,0,0,"347469",7.8542,, "S",,"10",, *** 3,1,"Lundin, Miss. Olga  
Elida","female",23,0,0,"347469",7.8542,, "S",,"10",,  
3,1,"Lundstrom, Mr. Thure Edvin","male",32,0,0,"350403",7.5792,, "S",,"15",, *** 3,1,"Lundstrom, Mr. Thure  
Edvin","male",32,0,0,"350403",7.5792,, "S",,"15",,  
3,0,"Lyntakoff, Mr. Stanko","male",,0,0,"349235",7.8958,, "S",,, *** 3,0,"Lyntakoff, Mr. Stanko","male",,0,0,"349235",7.8958,, "S",,,
```

# glom() transformation

Returns an RDD by fetching all elements within each partition to an array.

or in other words

Returns an RDD which contain array of elements.

Used mostly on matrix manipulations.

Used heavily on statistics and machine learning.

# glom() transformation

```
sc.textFile("D:\\ac\\data\\sample.csv")
    .glom()
    .foreach(eachArray =>
        eachArray.foreach(line => println(line)) )
```

SparkGlom.scala

# take, first, count, collect

# Actions

take(n) returns array with first n elements

first returns first element. Nothing but take(1)

count returns the count.

collect returns the result to the driver program. Always return a small amount of data.

foreach() iterates on an RDD.

# take and collect action

```
JavaRDD<String> rdd = javaSparkContext.textFile("file:///Users/apple/undata1.csv");
List<String> firstFiveRows = rdd.take(5);
firstFiveRows.forEach(z -> System.out.println(z));
```



Remember, take()  
and collect() return  
list of the objects.  
They don't return  
RDD's.



```
JavaRDD<String> rdd = javaSparkContext.textFile("file:///Users/apple/undata1.csv");
rdd.collect().forEach(z -> System.out.println(z));
```

# first and count action

```
JavaRDD<String> rdd = javaSparkContext.textFile("file:///Users/apple/undata1.csv");
```

```
String firstRow = rdd.first();  
System.out.println(firstRow);
```

```
List<String> firstOneRow = rdd.take(1);  
firstOneRow.forEach(z -> System.out.println(z));
```

```
JavaRDD<String> rdd = javaSparkContext.textFile("file:///Users/apple/undata1.csv");  
long totalCount = rdd.count();  
System.out.println(totalCount);
```

# foreach action

```
javaSparkContext.textFile("file:///Users/apple/titanic3.csv")
    .foreach(z -> System.out.println(z));
```

```
javaSparkContext.textFile("file:///Users/apple/titanic3.csv")
    .foreach(System.out::println);
```

# saveAsTextFile() and saveAsObjectFile() Action

## saveAsTextFile()

Writes data to the directory of file system. Spark uses `toString()` method on each object and stores it to disk.

```
rdd.saveAsTextFile("/Users/apple/saver002");
```

## saveAsObjectFile()

Write the element as an object file to the file system using simple java serialization. The file saved can be loaded using `SparkContext.objectFile()`.

```
rdd.saveAsObjectFile("/Users/apple/saverzz");
```

```
| Apples-MacBook-Pro:~ apple$ ls saverzz/  
| _SUCCESS          part-00000      part-00001
```

# saveAsTextFile() and saveAsObjectFile() Action

After saving the data to the file system, the data can be loaded to spark by SparkContext.objectFile() as shown below.

```
JavaSparkContext javaSparkContext = new JavaSparkContext(sparkConfig);
JavaRDD<UNDataBean> lcRDD = javaSparkContext.objectFile("/Users/apple/saverzz");
```

# saveAsSequenceFile() Action

Writes the data as a Hadoop SequenceFile.

A Hadoop SequenceFile is a binary key value pair based file.  
saveAsSequenceFile() is supported only on a pair rdd.

```
val sparkSession = SparkSession .builder()  
    .appName("SparkJOB")  
    .master("local")  
    .getOrCreate
```

```
val sparkContext = sparkSession.sparkContext
```

```
sparkContext.textFile("file:///home/aad-pc1/insurance.csv")  
    .map(x => (x.split(",")(0),x.split(",")(1)))  
    .saveAsSequenceFile("file:///home/aad-pc1/insurance_sequence.c
```

```
@InterfaceAudience.Public  
@InterfaceStability.Stable  
public class SequenceFile  
extends Object
```

SequenceFiles are flat files consisting of binary key/value pairs.  
SequenceFile provides SequenceFile.Writer, SequenceFile.Reader and  
SequenceFile.Sorger classes for writing, reading and sorting respectively.

There are three SequenceFile Writers based on the SequenceFile.CompressionType  
used to compress key/value pairs:

1. Writer : Uncompressed records.
2. RecordCompressWriter : Record-compressed files, only compress values.
3. BlockCompressWriter : Block-compressed files, both keys & values are  
collected in 'blocks' separately and compressed. The size of the 'block' is  
configurable.

The actual compression algorithm used to compress key and/or values can be specified  
by using the appropriate [CompressionCodec](#).

The recommended way is to use the static createWriter methods provided by the  
SequenceFile to chose the preferred format.

The SequenceFile.Reader acts as the bridge and can read any of the above  
SequenceFile formats.

# saveAsSequenceFile() Action

```
aad-pc1@AAD-PC1: ~/Insurance_sequence.csv
[1] 119736 FL 00000
[2] 448994 FL 00000
[3] 206893 FL 00000
[4] 333743 FL 00000
[5] 172534 FL 00000
[6] 785275 FL 00000
[7] 999592 FL 00000
[8] 223486 FL 00000
[9] 433512 FL 00000
[10] 142071 FL 00000
[11] 253816 FL 00000
[12] 894922 FL 00000
[13] 422834 FL 00000
[14] 582721 FL 00000
[15] 842700 FL 00000
[16] 874333 FL 00000
[17] 580146 FL 00000
[18] 456149 FL 00000
[19] 767862 FL 00000
[20] 353022 FL 00000
[21] 367814 FL 00000
[22] 671392 FL 00000
[23] 772887 FL 00000
[24] 983122 FL 00000
[25] 934215 FL 00000
[26] 385951 FL 00000
[27] 716332 FL 00000
[28] 751262 FL 00000
[29] 633663 FL 00000
[30] 105851 FL 00000
[31] 710400 FL 00000
[32] 703061 FL 00000
[33] 352792 FL 00000
[34] 717603 FL 00000
[35] 937659 FL 00000
[36] 294022 FL 00000
[37] 410500 FL 00000
[38] 524433 FL 00000
[39] 779298 FL 00000
[40] 491831 FL 00000
[41] 814637 FL 00000
[42] 737515 FL 00000
[43] 222653 FL 00000
[44] 788543 FL 00000
[45] 691681 FL 00000
[46] 368807 FL 00000
[47] 174002 FL 00000
[48] 198760 FL 00000
[49] 831395 FL 00000
[50] 305694 FL 00000
[51] 515722 FL 00000
[52] 415582 FL 00000
[53] 783533 FL 00000
[54] 640802 FL 00000
[55] 403866 FL 00000
[56] 282788 FL 00000
"part-00000" [neo4j][converted] 36635L, 670317C
```

# StatCounter - Statistical info on Numerical Data

```
val rdd = sparkContext.parallelize(List(100,  
    200,  
    300,  
    400,  
    500))
```

The statcounter acts on a RDD and gives numerical computation on the data.

```
val max = rdd.stats().max  
val min = rdd.stats().min  
val mean = rdd.stats().mean  
val variance = rdd.stats().variance  
  
println(min + " " + max + " " + mean + " " + variance)
```

The data on the RDD should be numerical data for StatCounter to be acted upon.

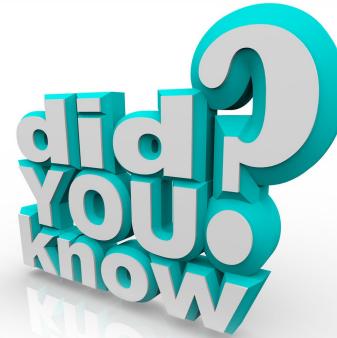
*org.apache.spark.util.StatCounter*

# Spark Packages

<http://spark-packages.org/>



Contains all open source libraries which are not shipped as the part of official distribution.



# PairRDD - Creation

- map()
- mapToPair()
- keyBy()
- flatMapValues()

# PairRDD - Operations

- reduceByKey()
- countByKey()
- groupByKey()
- sortByKey()

# PairRDD - Creation map()

- Constructs a two component tuples.

```
val sparkSession = SparkSession.builder()
    .appName("SparkJOB")
    .master("local")
    .getOrCreate

val sparkContext = sparkSession.sparkContext

val pairRDD = sparkContext.textFile("file:///home/aad-pc1/insurance.csv")
    .map(x => (x.split(",")(0),x.split(",")(1)))
    .foreach(x => println(x._1 + " " + x._2))
```

x. 1 will have the code

x. 2 will have the country code

chinnasamyad@gmail.com



# PairRDD - Creation

# keyBy()

- Constructs a two component tuples.
  - First key has the result of the function logic written inside the keyBy and the value has the original data

```
val sparkSession = SparkSession.builder()
    .appName("SparkJOB")
    .master("local")
    .getOrCreate

val sparkContext = sparkSession.sparkContext

sparkContext.textFile("file:///home/aad-pc1/insurance.csv")
    .keyBy((x) => x.split(",")(2))
    .map(x => println(x._1) )
    .collect()
```

- x.\_1 will have the country name
- x.\_2 will have the entire data.

# PairRDD - Actions

1. lookup()
2. collectAsMap()
3. countByKey()

```
val out = sparkContext.textFile("file:///home/aad-pc1/insurance.csv")
    .map(x => (x.split(",")(0),x.split(",")(1)))
    .lookup("742303")
```

```
val out1 = sparkContext.textFile("file:///home/aad-pc1/insurance.csv")
    .map(x => (x.split(",")(0),x.split(",")(1)))
    .collectAsMap()
```

```
val out2 = sparkContext.textFile("file:///home/aad-pc1/insurance.csv")
    .map(x => (x.split(",")(0),x.split(",")(1)))
    .countByKey()
```

# PairRDD - Creation

## mapToPair()

```
JavaRDD<LatLong> locRDD = javaSparkContext.parallelize(locationList);
```

```
JavaPairRDD<Integer, LatLong> pairRDD = locRDD.mapToPair(new PairFunction<LatLong, Integer, LatLong>() {
```

```
    @Override  
    public Tuple2<Integer, LatLong> call(LatLong latLong) throws Exception {  
        return new Tuple2<>(new Integer(latLong.getId()), latLong);  
    }  
});
```

```
pairRDD.foreach((z) -> System.out.println(z._1.intValue() + " - " + z._2.getName()));
```

# PairRDD - Creation

## keyBy()

```
JavaRDD<LatLong> locRDD = javaSparkContext.parallelize(locationList);
```

```
JavaPairRDD<Integer, LatLong> pairRDD_KeyBy_ = locRDD.keyBy(new Function<LatLong, Integer>() {
```

```
    @Override
```

```
    public Integer call(LatLong v1) throws Exception {
```

```
        return new Integer(v1.getId());
```

```
    }
```

```
});
```

```
pairRDD_KeyBy_.foreach((z) -> System.out.println(z._1.intValue() + " --- " + z._2.getName()));
```

# PairRDD - Creation

## flatMapValues()

flatMapValues() transformation:

The flatMapValues can be performed over only on PairRDD's

```
JavaPairRDD<Integer, String> flatMapValuesRDD = pairRDD
    .flatMapValues(new Function<LatLong, Iterable<String>>())
{
    @Override
    public Iterable<String> call(LatLong latLong) throws Exception {
        return Arrays.asList(latLong.getLatitude(), latLong.getLongitude(), latLong.getName());
    }
});
```

# PairRDD - Operations

## reduceByKey()

```
JavaPairRDD<String, Integer> reduceByKeyRDD =  
  
    javaSparkContext.parallelize(listOfNames)  
        .mapToPair  
            (param -> new Tuple2<>(String.valueOf(param.charAt(0)), new Integer(1)))  
        .reduceByKey  
            ((param1, param2) -> new Integer(param1.intValue() + param2.intValue()));
```

# PairRDD - Operations

## groupByKey()

```
JavaRDD<String> textRDD = javaSparkContext.textFile("file:///Users/apple/simple_text_file.txt");

textRDD.flatMap(param -> Arrays.asList(param.split(" ")))
    .mapToPair(param -> new Tuple2<String, Integer>(param, 1))
    .groupByKey()
    .foreach((words) -> System.out.println(words._1 + " " + words._2));
```

MLlib [1]

for [1, 1, 1]

computing [1]

an [1]

Scala, [1]

machine [1]

engine [1]

and [1, 1, 1, 1, 1]

supports [1, 1]

# PairRDD - Operations

## sortByKey()

```
JavaRDD<String> textRDD = javaSparkContext.textFile("file:///Users/apple/simple_text_file.txt");
```

```
textRDD.flatMap(param -> Arrays.asList(param.split(" ")))
    .mapToPair(param -> new Tuple2<String, Integer>(param, 1))
    .sortByKey()
    .foreach((words) -> System.out.println(words._1 + " " + words._2));
```

fast 1  
APIs 1  
for 1  
Apache 1  
for 1  
GraphX 1  
for 1

# PairRDD - Operations

## countByKey()

```
JavaRDD<String> textRDD = javaSparkContext.textFile("file:///Users/apple/simple_text_file.txt");

Set<Entry<String, Object>> values = textRDD.flatMap(param -> Arrays.asList(param.split(" ")))
    .mapToPair(param -> new Tuple2<String, Integer>(param, 1))
    .countByKey()
    .entrySet();

values.forEach(param -> System.out.println(param.getKey() + " " + param.getValue()));
```

Note: countByKey() is an action.

# PairRDD - Operations

## countByKey()

countByKey internally calls reduceByKey and then collect and converts to a map.

*reduceByKey(\_ + \_).collect().toMap*

So countByKey should not be used for large datasets. Since this API will load everything to the driver's memory.

Performance Improvement

# PairRDD - Operations

## countByKey()

Instead of countByKey() use

```
rdd.mapValues(_ => 1L).reduceByKey(_ + _)
```

for large datasets.

Performance Improvement

# Word count in Spark

```
SparkConf sparkConfig = new SparkConf()
    .setAppName("WordCountSparkExample")
    .setMaster("local[8]");

JavaSparkContext javaSparkContext = new JavaSparkContext(sparkConfig);

JavaRDD<String> textRDD = javaSparkContext.textFile("file:///Users/apple/simple_text_file.txt");

JavaRDD<String> wordsFlattenedRDD = textRDD.flatMap(new FlatMapFunction<String, String>() {

    @Override
    public Iterable<String> call(String input) throws Exception {
        return Arrays.asList(input.split(" "));
    }
});
```

# Word count in Spark

```
JavaPairRDD<String, Integer> pairWordMapperRDD = wordsFlattenedRDD.mapToPair(new PairFunction<String, String, Integer>() {  
  
    @Override  
    public Tuple2<String, Integer> call(String inputWord) throws Exception {  
  
        return new Tuple2<String, Integer>(inputWord, 1);  
    }  
  
});  
  
JavaPairRDD<String, Integer> countedWords = pairWordMapperRDD.reduceByKey(new  
Function2<Integer, Integer, Integer>() {  
  
    @Override  
    public Integer call(Integer arg0, Integer arg1) throws Exception {  
        return new Integer(arg0 + arg1);  
    }  
});  
  
countedWords.foreach((words) -> System.out.println(words._1 + words._2));
```

# Word count in Spark

```
textRDD.flatMap(param -> Arrays.asList(param.split(" ")))  
    .mapToPair(param -> new Tuple2<String, Integer>(param, 1))  
    .reduceByKey((integerParam1, integerParam2) -> new  
                Integer(integerParam1 + integerParam2))  
    .foreach((words) -> System.out.println(words._1 + " " + words._2));
```

# Word count in Spark - Scala

```
object WordCountSpark extends App {  
  
    val config = new SparkConf  
    config.setAppName("WordCount")  
    config.setMaster("local[4]")  
  
    val sc = new SparkContext(config)  
  
    sc.textFile("D:/ac/data/Simple_Text_File.txt").flatMap(line => line.split(" "))  
        .map(eachWord => (eachWord,1))  
        .reduceByKey((parameterA,parameterB) => parameterA + parameterB)  
        .foreach(wordcount => println(wordcount._1 + wordcount._2))  
  
    sc.stop()  
  
}
```

# Spark Job History Server

# Spark Job History Server

To start the job history server use the below command,

*sbin/start-history-server.sh*

```
SparkConf sparkConfig = new SparkConf()
    .set("spark.eventLog.enabled", "true")
    .set("spark.eventLog.dir", "file:///Users/apple/spark-events")
        // Default is /tmp/spark-events

    .setAppName("ReadUNData")
    .setMaster("local[8]");
```

Note: You need to create the spark-events directory manually.

# Spark Job History Server

When the job history server is started, you should see it in the process status

```
Apples-MacBook-Pro:sbin apple$ ./start-history-server.sh
starting org.apache.spark.deploy.history.HistoryServer, logging to /Users/apple/spark14/sbin/../logs/spark-apple-org.apache.spark.deploy.history.HistoryServer-1
Apples-MacBook-Pro.local.out
Apples-MacBook-Pro:sbin apple$ ps -ef | grep spark
 501 2135 144  0 11:08AM ??        1:34.37 /Users/apple/sparktraining/Eclipse.app/Contents/MacOS/eclipse
 501 2224     1  0 11:13AM ttys001    0:02.44 /Library/Java/JavaVirtualMachines/jdk1.8.0_60.jdk/Contents/Home/jre/bin/java -cp /Users/apple/spark14/sbin/../co
nf:/Users/apple/spark14/lib/spark-assembly-1.4.0-hadoop2.6.0.jar:/Users/apple/spark14/lib/datanucleus-api-ido-3.2.6.jar:/Users/apple/spark14/lib/datanucleus-cor
e-3.2.10.jar:/Users/apple/spark14/lib/datanucleus-rdbms-3.2.9.jar -Xms512m -Xmx512m org.apache.spark.deploy.history.HistoryServer
```

# Spark Job History Server

Job history server starts in 18080 port by default.

The screenshot shows a web browser window with the URL `localhost:18080` in the address bar. The page title is **History Server**, with the **Spark 1.4.0** logo to its left. Below the title, it says **Event log directory: file:/tmp/spark-events**. A table header indicates **Showing 1-1 of 1** applications. The table has columns: App ID, App Name, Started, Completed, Duration, Spark User, and Last Updated. One row is shown: **App ID** is `local-1443159738356`, **App Name** is `ReadUNData`, **Started** is `2015/09/25 11:12:17`, **Completed** is `2015/09/25 11:12:19`, **Duration** is `2 s`, **Spark User** is `apple`, and **Last Updated** is `2015/09/25 11:12:19`. At the bottom left, there is a link **Show incomplete applications**.

App ID	App Name	Started	Completed	Duration	Spark User	Last Updated
local-1443159738356	ReadUNData	2015/09/25 11:12:17	2015/09/25 11:12:19	2 s	apple	2015/09/25 11:12:19

# RDD Lineage

RDD's follow a parent child relationship whereby the child knows its parent.

Each action on the RDD, re-executes the transformation starting from the base.

RDD Lineage helps in fault tolerant behavior of the RDD.

# RDD Checkpointing

RDD lineage provides fault tolerance but also can cause problems when the lineage gets long.

Checkpointing saves the data in the disk and solves in RDD lineage issues.

Since the recovery of data can be expensive, checkpointing saves the data to HDFS.

Checkpointing has to be done before any actions on the RDD is taken.

In spark streaming, checkpointing helps in fault tolerant streaming.

# RDD Checkpointing

```
JavaSparkContext javaSparkContext = new JavaSparkContext(sparkConfig);
javaSparkContext.setCheckpointDir("/Users/apple/checkpointdir");
```

```
JavaRDD<String> locRDD = javaSparkContext.textFile("file:///Users/apple/titanic3.csv");
```

```
JavaRDD<String> locRDDUnion =
    locRDD.union(locRDD).union(locRDD).union(locRDD).union(locRDD).union(locRDD).union(
    locRDD).union(locRDD).union(locRDD).union(locRDD).union(locRDD).union(locRDD).union(locRDD).;
```

```
locRDDUnion.checkpoint();
```

```
locRDDUnion.foreach((z) -> System.out.println(z));
```

# RDD Checkpointing

Use the `isCheckpointed()` method to check if an RDD is checkpointed

```
boolean islocRDDCheckpointed = locRDDUnion.isCheckpointed()
```

```
if (islocRDDCheckpointed) {
```

```
....
```

```
....
```

```
}
```

```
else {
```

```
....
```

```
....
```

```
}
```

# When to do RDD Checkpointing

The computation takes a long time.

The RDD lineage chain is long.

The code complexity depends on too many RDD's.

# Speculative Execution in Spark

It is an optimization technique adopted by the mapreduce framework.

When a JobTracker finds out that a task takes too much of time to execute, it can start additional instance of the same task.

This process of execution is called **speculative execution**.

Speculative execution ensures that a slowness in a machine will not slow down a task.

# Speculative Execution in Spark

By default it is false.

You make it enabled by *spark.speculation* to true.

# Accumulators

Accumulators provides a simple way of aggregating values from the worker nodes to the driver program. Accumulators are used for counting in data.

An accumulator is created using a spark-context.

A worker adds a value to the accumulator using the add() method and the driver program calls the accumulator to access its final value.

Worker nodes always write to accumulators and finally the accumulator value is accessed by the driver program.

# Accumulators

```
Accumulator<Integer> survivedAccumulator = javaSparkContext.accumulator(0, "survived_accumulator");
Accumulator<Integer> deadAccumulator = javaSparkContext.accumulator(0, "dead_accumulator");
```

```
long totalCount = javaSparkContext.textFile("file:///Users/koteshwar/titanic3.csv")
    .map(new Function<String, String>() {
```

```
        @Override
        public String call(String input) throws Exception {
```

```
            String[] splittedInput = input.split(",");
```

```
            if (splittedInput[1].equals("1"))
                survivedAccumulator.add(new Integer(1));
            else
                deadAccumulator.add(new Integer(1));
```

```
        return splittedInput[1];
    }
```

```
}).count();
```

```
System.out.println("Survived People - " + survivedAccumulator.value());
System.out.println("People Dead - " + deadAccumulator.value());
System.out.println("Total Count - " + totalCount);
```

# Accumulator value cannot be accessed inside a worker

```
long totalCount = javaSparkContext.textFile("file:///Users/koteshwar/titanic3.csv")
    .map(new Function<String, String>() {

        @Override
        public String call(String input) throws Exception {

            System.out.println(survivedAccumulator.value());
            .....
            .....
        }
    }
```

```
java.lang.UnsupportedOperationException: Can't read accumulator value in task
    at org.apache.spark.Accumulable.value(Accumulators.scala:97)
```

# Custom Accumulator

Implement the accumulator extending the Accumulator Param

```
class UIDAccumulator implements AccumulatorParam<UIDBean> {  
  
    public UIDBean addInPlace(UIDBean first, UIDBean second) {  
  
        UIDBean bean = new UIDBean();  
        bean.setUid(first.getUid() + " - " + second.getUid());  
  
        return bean;  
    }  
  
    public UIDBean zero(UIDBean arg0) {  
        return new UIDBean();  
    }  
  
    public UIDBean addAccumulator(UIDBean first, UIDBean second) {  
  
        UIDBean bean = new UIDBean();  
        bean.setUid(first.getUid() + " - " + second.getUid());  
  
        return bean;  
    } }
```

Java

# Custom Accumulator

## Second : define the accumulator

```
Accumulator<UIDBean> successAccumulator = javaSparkContext.accumulator(  
    new UIDBean(),  
    "success_uid_accumulator",  
    new UIDAccumulator());
```

Java

```
Accumulator<UIDBean> failureAccumulator = javaSparkContext.accumulator(  
    new UIDBean(),  
    "failure_uid_accumulator",  
    new UIDAccumulator());
```

# Custom Accumulator

```
javaSparkContext.textFile("file:///D:/ac/data/legacy uid.csv")
    .map (new Function<String, String>() {
        @Override
        public String call(String input) throws Exception {
            UIDBean uidBean = new UIDBean();
            uidBean.setUid(input);
            if (input.startsWith("28336"))
                successAccumulator.add(uidBean);
            else
                failureAccumulator.add(uidBean);
            return input;
        }
    }).count();
```

```
System.out.println("Successful Aadhaars - " + successAccumulator.value().getUid());
```

# Custom Accumulator

```
class UIDBean implements Serializable {  
  
    private String uid = "";  
  
    public String getUid() {  
        return uid;  
    }  
  
    public void setUid(String uid) {  
        this.uid = uid;  
    }  
  
}
```

Java

# Custom Accumulator

SCALA

```
implicit object AuthSuccessAccumulator extends AccumulatorParam[AuthSuccess] {  
    override def addInPlace(r1: AuthSuccess, r2: AuthSuccess): AuthSuccess =  
        AuthSuccess(r1.uid + " - " + r2.uid)  
    override def zero(initialValue: AuthSuccess): AuthSuccess = AuthSuccess("")  
}
```

# Custom Accumulator

```
object CustomAccumulator extends App {  
  
    val config = new SparkConf  
    config.setAppName("Accumulator")  
    config.setMaster("local[4]")  
  
    val context = new SparkContext(config)  
  
    val successAccumulator = context.accumulator(AuthSuccess("1000"),  
    "SUCCESS_ACCUMULATOR")  
  
    context.textFile("D:/ac/data/legacy_uid.csv").map(x =>  
  
        if (x.startsWith("28336"))  
            successAccumulator.add(AuthSuccess(x))  
  
    ).collect()  
  
    println(successAccumulator.value)  
}  
chinnasamyad@gmail.com
```

SCALA

# Broadcast Variables

Broadcast variable allows a program to send in read-only value objects to all worker nodes.

A broadcast variable is an object of type `org.apache.spark.broadcast.Broadcast` which wraps a custom object.

The broadcast-variable allows spark to efficiently transfer data to all worker nodes.

Broadcast variables minimizes the data transfer over the network. The value is sent to each node only once.

# Broadcast Variables

```
import org.apache.spark.broadcast.Broadcast;

Broadcast<TitanicBean> bc = javaSparkContext.broadcast(new TitanicBean("1", "1", "William"));

long totalCount = javaSparkContext.textFile("file:///Users/koteshwar/titanic3.csv")
    .map(new Function<String, String>() {

        @Override
        public String call(String input) throws Exception {
            String[] splittedInput = input.split(",");
            TitanicBean titanicBean = bc.getValue();

            System.out.println(titanicBean.getName());
            return splittedInput[1];
        }
    }).count();
```

# Spark Internals

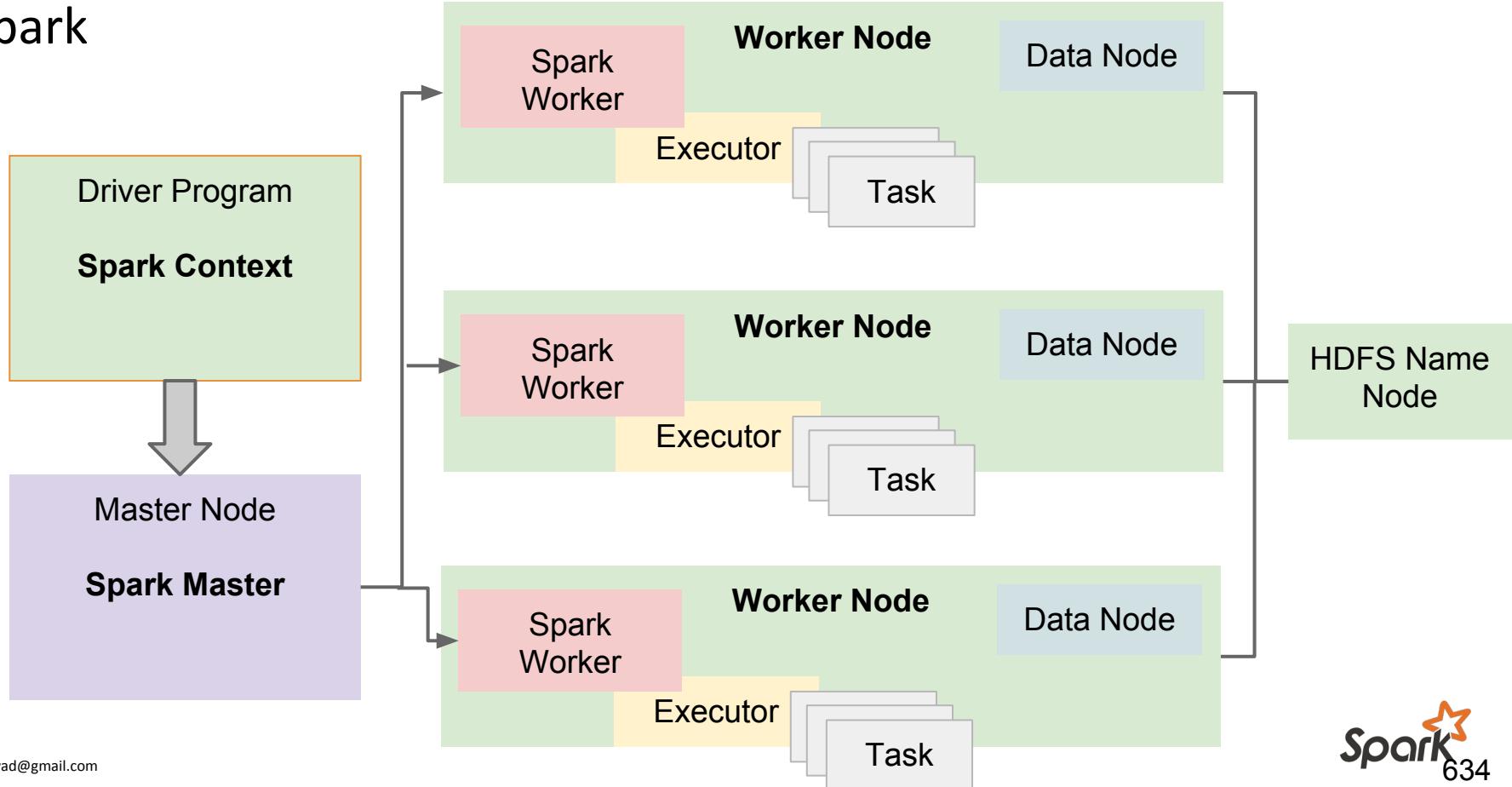
Driver Program: The main program which creates the Spark Context (sc)

Master: Spark cluster runs in the master node. Spark comes with its own cluster manager or can run on Mesos/YARN.

Workers: A node which runs application code on the cluster

Executors: Executor is the program that is launched on the workers when a job is submitted. Each application has its own executors.

# The Driver program, the master, the workers and the tasks of spark

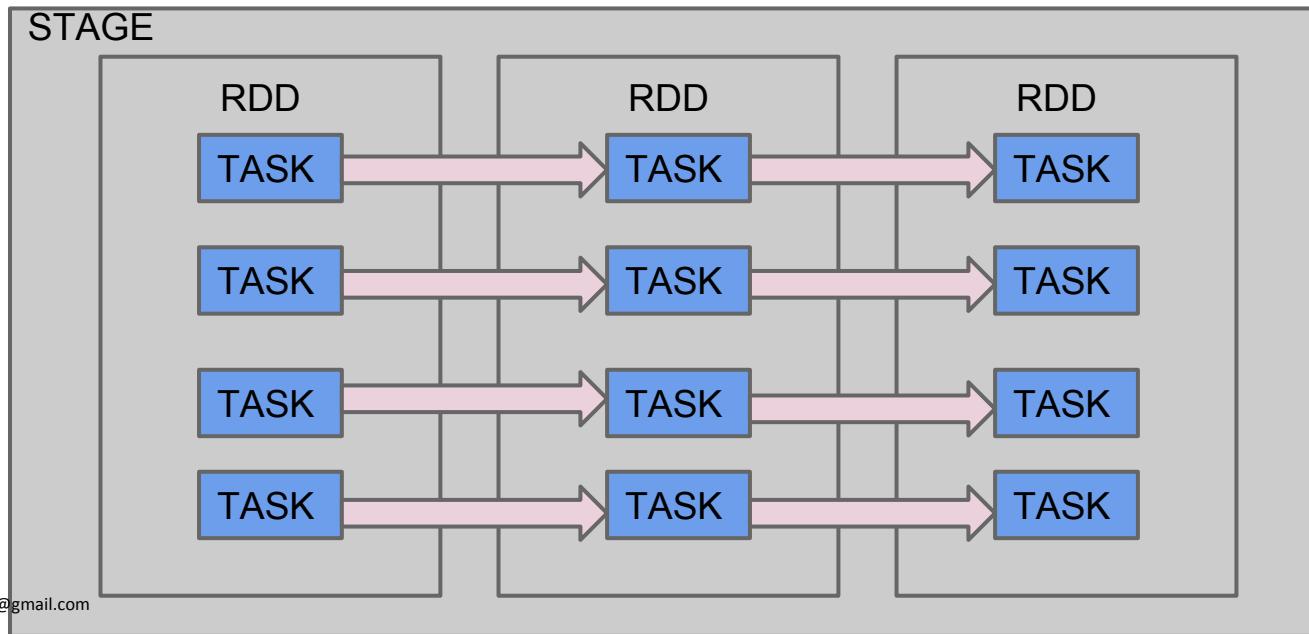


# Spark Internals - Job, Stage and Task

Job a set of tasks executed as a result of an action

Stage a set of tasks in a job that can be executed in parallel

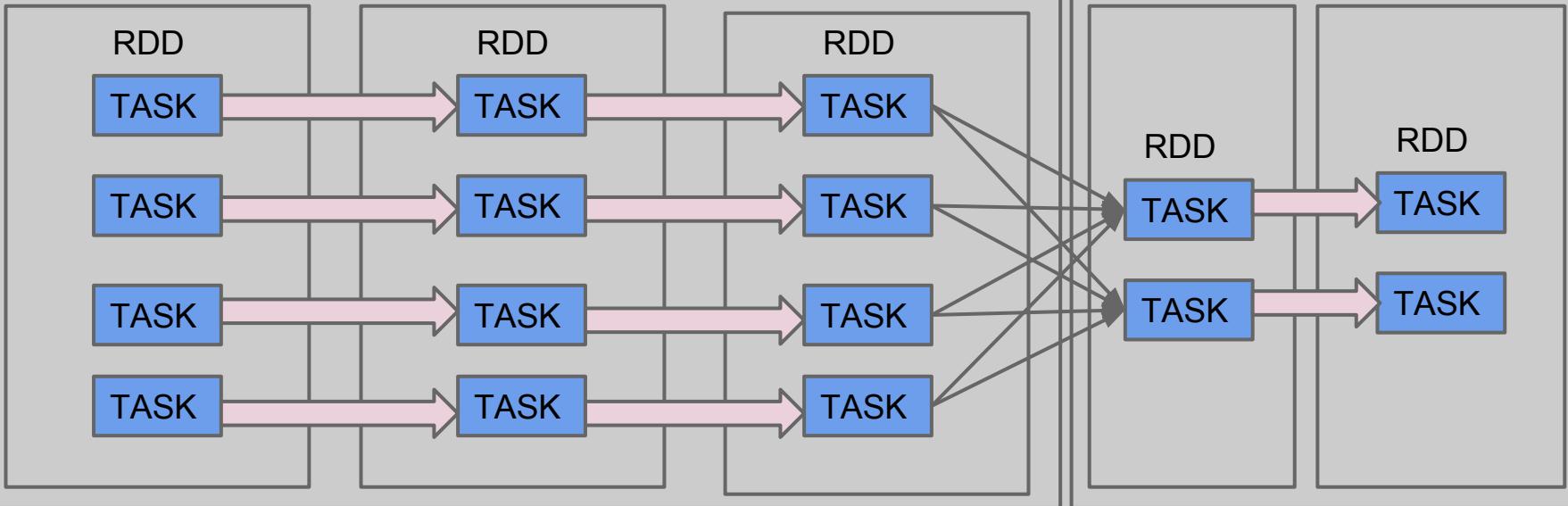
Task an individual unit of work sent to one executor



# Spark Internals - Job, Stage and Task

JOB

STAGE



# Spark Internals - Job, Stage and Task

RDD's are stored in memory of the Spark Executor JVM's

RDD operations are executed on partitions in parallel.

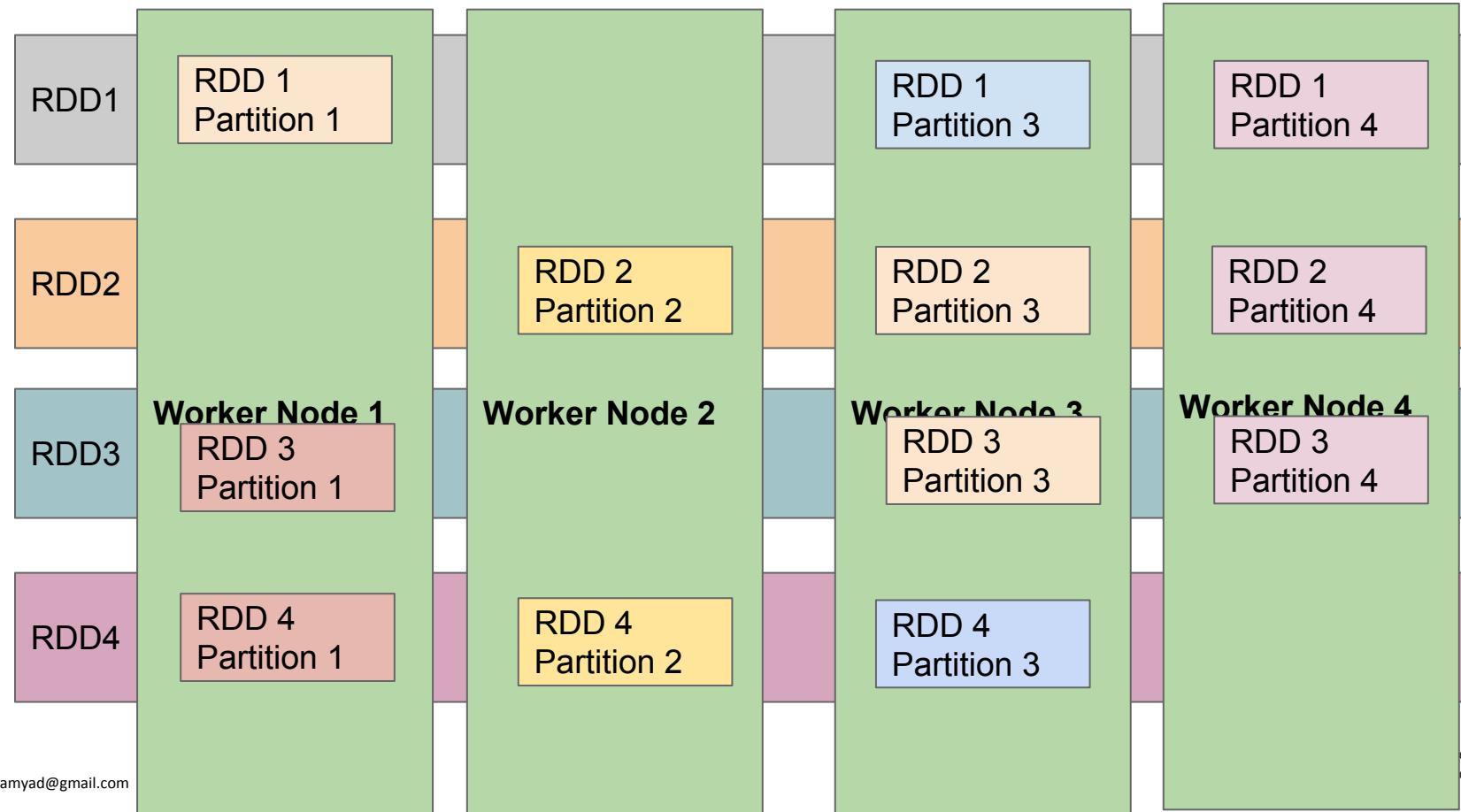
Operations that depend on the same partition are pipelined together in stages.

Operations that depend on multiple partitions are executed in separate stages.

Number of Tasks = Number of Partitions

Tasks within a stage are pipelined together.

# Spark Partition

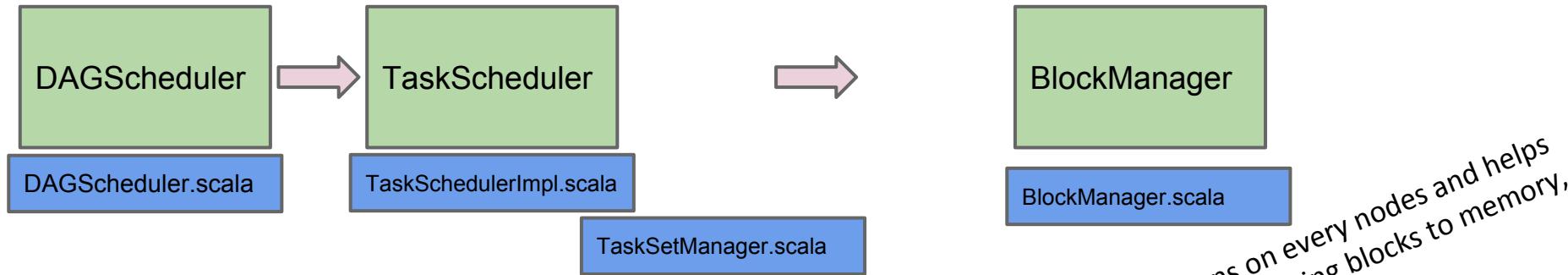


# Spark Internals - Job, Stage and Task

By default, spark creates one partition for each block of file.

Here, block meaning to one HDFS block. (ie., 64 MB by default).

# Spark Internals - DAG



DAGScheduler computes a DAG of stages for each job.

Number of times that a task will be attempted in a stage can be configured on `spark.task.maxFailures`, default is 4.

BlockManager runs on every nodes and helps in retrieving and persisting blocks to memory, disk or tachyon.

# Spark Internals

Spark 1.4.0

Jobs Stages Storage Environment Executors

## Details for Job 0

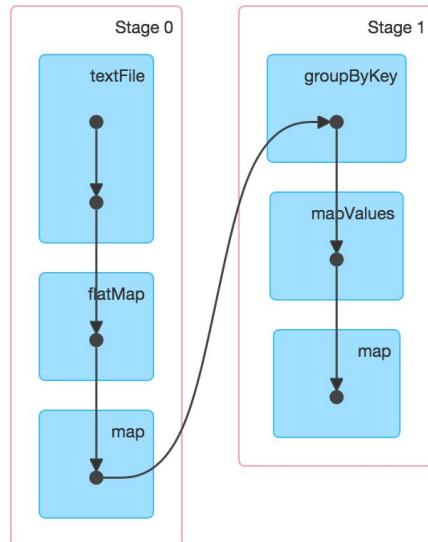
Status: RUNNING

Active Stages: 1

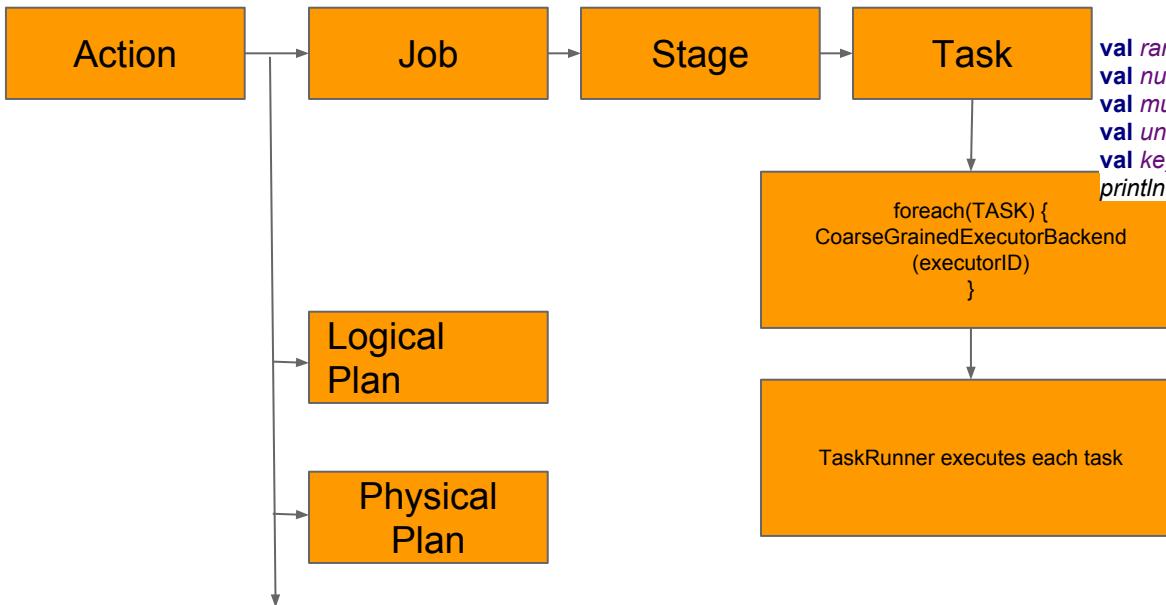
Completed Stages: 1

▶ Event Timeline

▼ DAG Visualization



# Spark Internals



```
(2) MapPartitionsRDD[4] at map at SparkScalaSingle.scala:373 []
| UnionRDD[3] at union at SparkScalaSingle.scala:370 []
| ParallelCollectionRDD[1] at parallelize at SparkScalaSingle.scala:364 []
| MapPartitionsRDD[2] at map at SparkScalaSingle.scala:367 []
| ParallelCollectionRDD[1] at parallelize at SparkScalaSingle.scala:364 []
```

```
val rangeOfNumbers = 1 to 100
val numberRDD = sparkContext.parallelize(rangeOfNumbers)
val multipliedRDD = numberRDD.map(x => x*2)
val unionRDD = numberRDD.union(multipliedRDD)
val keyValueUnionRDD = unionRDD.map(x=> (x,x))
println(keyValueUnionRDD.toDebugString)
```

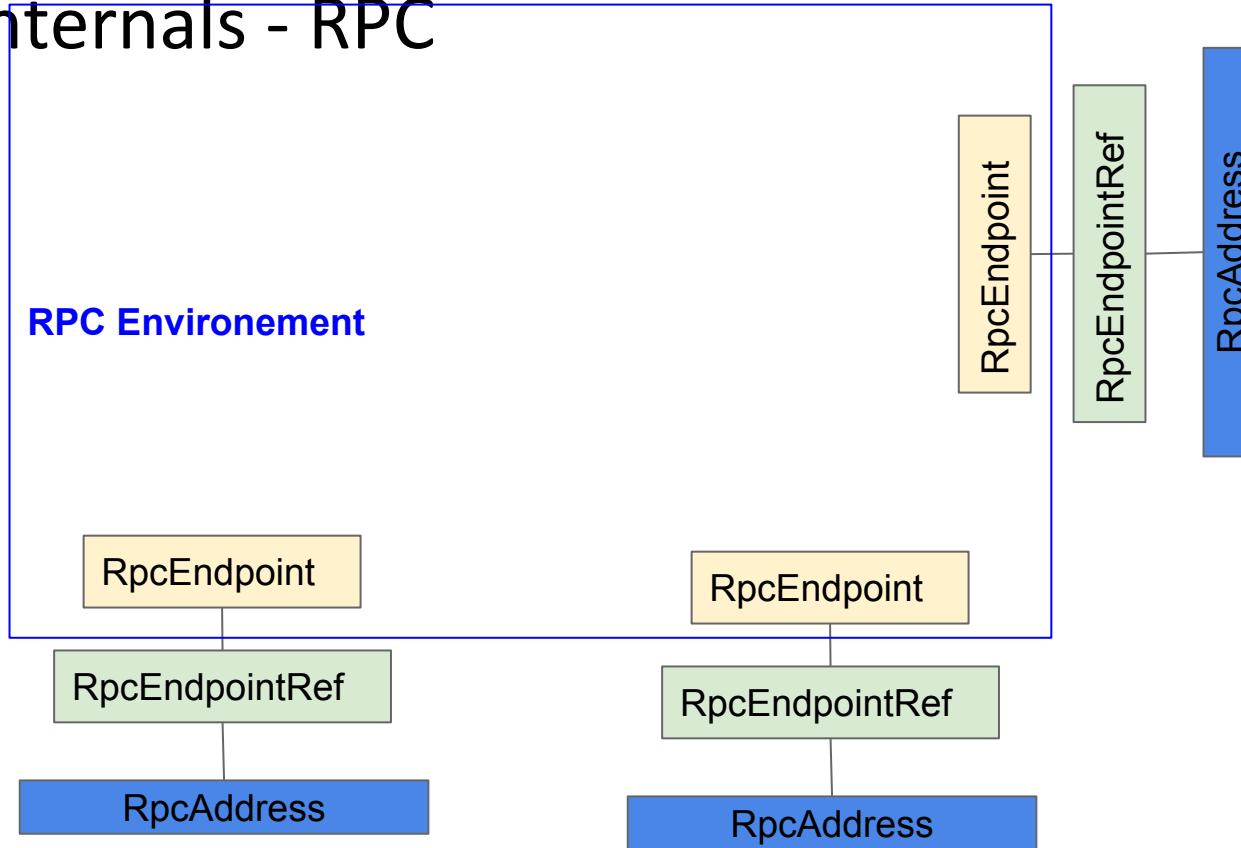
# Spark Internals

The screenshot shows the Spark UI interface for a "ReadFile application UI". The top navigation bar includes links for Jobs (which is active), Stages, Storage, Environment, and Executors. The main content area displays "Details for Job 0". Key metrics shown are Status: RUNNING, Active Stages: 1, and Completed Stages: 1. There are links for Event Timeline and DAG Visualization. Two tables are present: "Active Stages (1)" and "Completed Stages (1)". The "Active Stages" table has one row with Stage Id 1 and Description "count at SparkInternals.java:43". The "Completed Stages" table has one row with Stage Id 0 and Description "mapToPair at SparkInternals.java:26". Both tables include columns for Stage Id, Description, +details, Submitted (2015/09/20 10:04:12), Duration (1.2 min for active, 97 ms for completed), Tasks: Succeeded/Total (0/2 for active, 2/2 for completed), Input (1063.0 B), Output, Shuffle Read, and Shuffle Write.

Stage Id	Description	+details	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
1	count at SparkInternals.java:43	(kill)	2015/09/20 10:04:12	1.2 min	0/2			1063.0 B	

Stage Id	Description	+details	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
0	mapToPair at SparkInternals.java:26		2015/09/20 10:04:12	97 ms	2/2	407.0 B			1063.0 B

# Spark Internals - RPC



# Spark Internals - RPC

Two RPC Implementations of Rpc is supported by spark

**Netty**

**Akka**

**Netty - org.apache.spark.rpc.netty.NettyRpcEnvFactory**

**Akka - org.apache.spark.rpc.akka.AkkaRpcEnvFactory**

This can be set in the config `spark.rpc` (either `netty` or `akka`, default is `netty`)

`--config spark.rpc = netty`

# Custom RDD

Creating our own RDD by adding custom functionalities to it.

One way of extending Spark API

To implement extend the RDD class and override  
`getPartitions` and `compute()` methods.

# Custom RDD

```
class LegitimateBiometricRDD(biometricBeanRDD : RDD[BiometricBean]) extends  
RDD[BiometricBean](biometricBeanRDD) {  
  
override def compute(split: Partition, context: TaskContext): Iterator[BiometricBean]  
= {  
  
    firstParent[BiometricBean].iterator(split, context).map(eachRecord => {  
        val fuzzyScore = eachRecord.fuzzyScore  
  
        if (fuzzyScore > 6)  
            BiometricBean(eachRecord.irisTemplate, eachRecord.handTemplate,  
"LEGITIMATE BIOMETRIC", 10)  
        else  
            BiometricBean(eachRecord.irisTemplate, eachRecord.handTemplate,  
"BIO_METRIC SCORE INSUFFICIENT", 0)  
  
    })  
}  
  
override protected def getPartitions: Array[Partition] = {  
    firstParent[BiometricBean].partitions  
}  
}  
chinnasamyad@gmail.com
```

CustomRDD.scala



# Custom RDD

```
val sc = new SparkContext(config)
val biometricRDD = sc.parallelize(dataSource)

/*
  Using LegitimateBiometricRDD
*/
val legitimateBiometricRDD = new LegitimateBiometricRDD(biometricRDD)
legitimateBiometricRDD.foreach(biometricRecord =>
  println(biometricRecord.fuzzyScore + " - " + biometricRecord.errorCode))

case class BiometricBean(irisTemplate : String,
                        handTemplate : String,
                        errorCode : String,
                        fuzzyScore : Int)
```

CustomRDD.scala

# Spark RDD

## Logical Plan

Logical plan is nothing but the Data Dependency Graph. `rdd.toDebugString()` returns the logical plan.

## Physical Plan

Physical plan is the Directed Acyclic Graph.

After the physical plan, concrete tasks are generated.

# Spark RDD - Logical Plan

```
JavaPairRDD<String, Integer> reducedRDD =  
    textRDD.flatMap(param -> Arrays.asList(param.split(" ")))  
        .mapToPair(param -> new Tuple2<String, Integer>(param, 1))  
        .reduceByKey((integerParam1, integerParam2) -> new Integer(integerParam1 +  
integerParam2));  
  
System.out.println(reducedRDD.toDebugString());
```

*Output:*

```
(2) ShuffledRDD[4] at reduceByKey at WordCountUsingSpark.java:71 []  
+- (2) MapPartitionsRDD[3] at mapToPair at WordCountUsingSpark.java:70 []  
|  MapPartitionsRDD[2] at flatMap at WordCountUsingSpark.java:69 []  
|  MapPartitionsRDD[1] at textFile at WordCountUsingSpark.java:26 []  
|  file:///Users/apple/simple_text_file.txt HadoopRDD[0] at textFile at WordCountUsingSpark.java:26 []
```

# Spark Memory Model

`spark.storage.memoryFraction`

`spark.storage.unrollFraction`

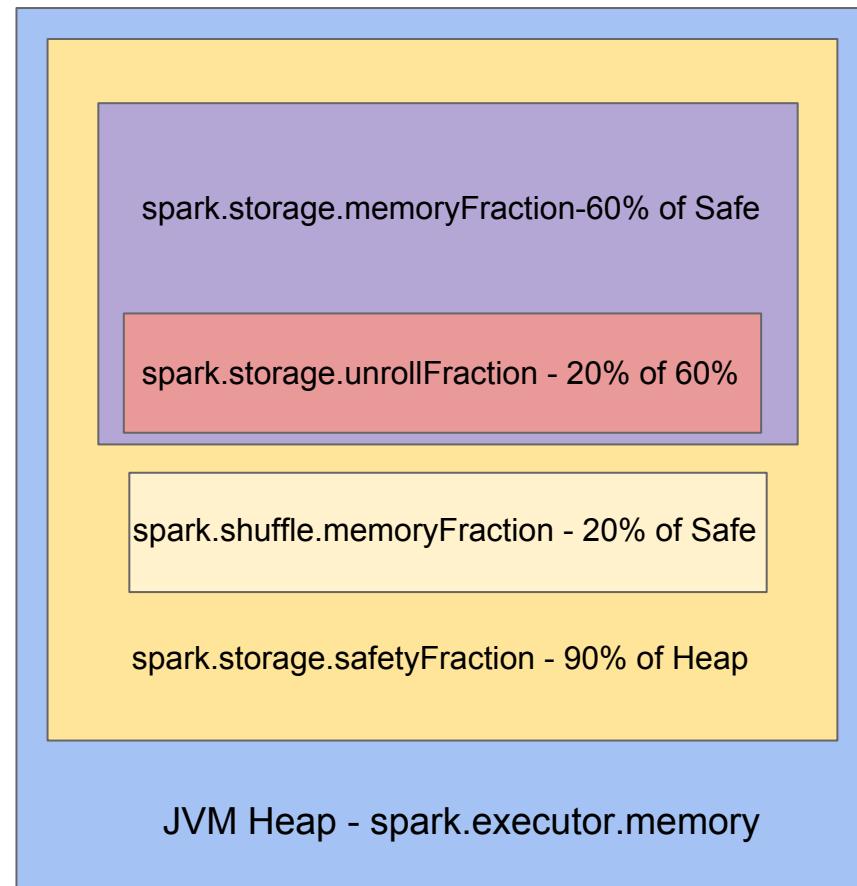
Serialize/Deserialize objects  
to disk when they don't fit in memory.

`spark.shuffle.memoryFraction`

Storing in-memory shuffle.

`spark.storage.safetyFraction`

`spark.executor.memory`

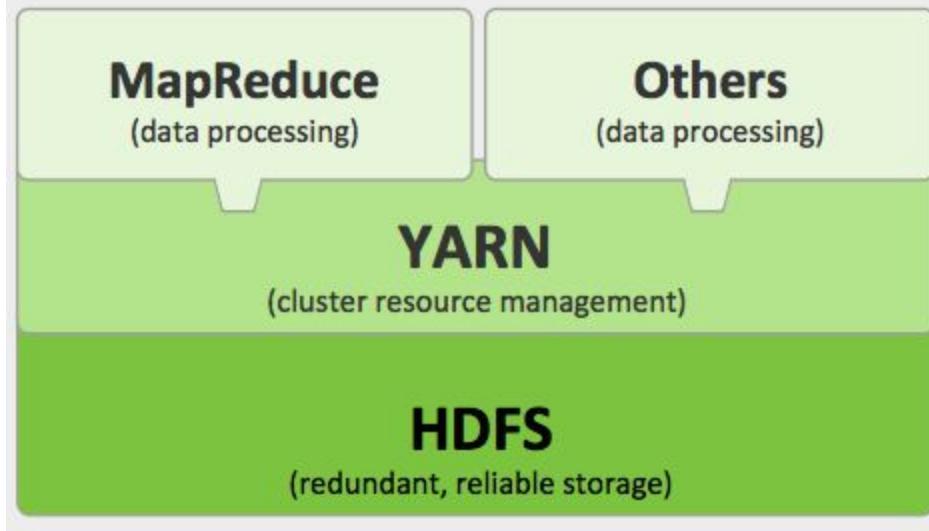


# Spark and Hadoop Difference

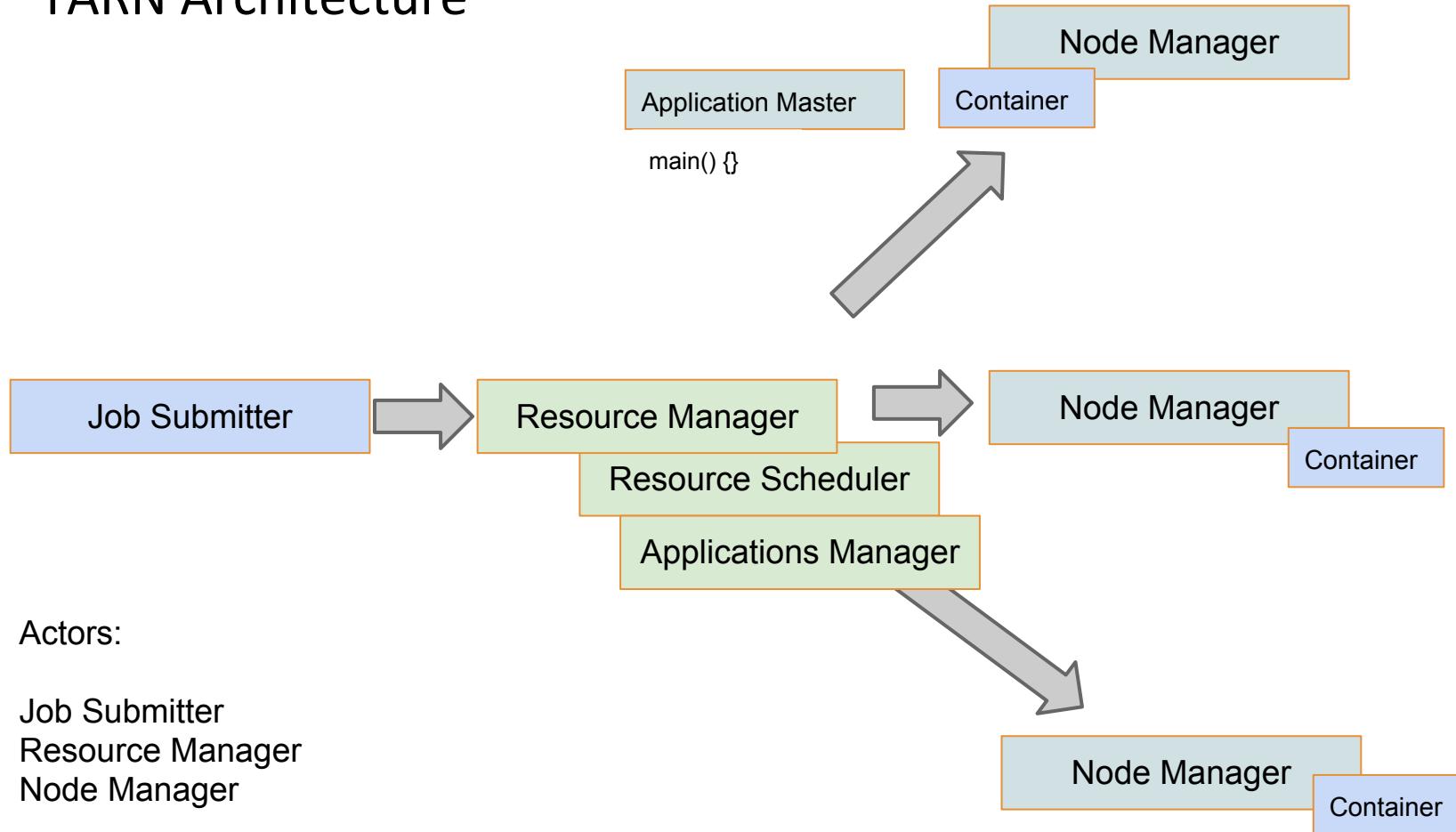
Hadoop - MR	Spark
The dataflow is fixed and predefined. We just fill in the code for mapper() and reducer().	In Spark, the dataflow can be complicated and coding is easy since the plumbing of the code is hidden from the programmer.
No plumbing of code.	Plumbing of the code. So programmer can concentrate on the business logic.



# HADOOP 2.0



# YARN Architecture



# YARN Architecture

- [1] Client submits an application to the Resource Manager
- [2] Resource manager assigns a container by contacting the Node manager
- [3] The node manager launches the container
- [4] And at last the node manager executes the Application Master

# YARN Architecture

```
localhost:hadoop tester$ cat mapred-site.xml
<?xml version="1.0"?>
<!DOCTYPE configuration SYSTEM "http://www.apache.org/xml/ns/policy.dtd">
<!--
    Licensed under the Apache License, Version 2.0 (the "License");
    you may not use this file except in compliance with the License.
    You may obtain a copy of the License at

        http://www.apache.org/licenses/LICENSE-2.0

    Unless required by applicable law or agreed to in writing, software
    distributed under the License is distributed on an "AS IS" BASIS,
    WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
    See the License for the specific language governing permissions and
    limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->
<configuration>
    <property>
        <name>mapreduce.framework.name</name>
        <value>yarn</value>
    </property>

    <property>
        <name>mapreduce.jobhistory.address</name>
        <value>localhost:10020</value>
        <description>Host and port for Job History Server (default 0.0.0.0:10020)</description>
    </property>
</configuration>
localhost:hadoop tester$ cat yarn-site.xml
<?xml version="1.0"?>
<!--
    Licensed under the Apache License, Version 2.0 (the "License");
    you may not use this file except in compliance with the License.
    You may obtain a copy of the License at

        http://www.apache.org/licenses/LICENSE-2.0

    Unless required by applicable law or agreed to in writing, software
    distributed under the License is distributed on an "AS IS" BASIS,
    WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
    See the License for the specific language governing permissions and
    limitations under the License. See accompanying LICENSE file.
-->
<configuration>
    <property>
        <name>yarn.nodemanager.aux-services</name>
        <value>mapreduce_shuffle</value>
    </property>
</configuration>
```

# Storage Levels

- |                     |  |
|---------------------|--|
| DISK_ONLY           | - Store the RDD only on disk   |
| MEMORY_ONLY         | - Default mode. The RDD is stored in memory. If the some of the RDD doesn't fit in memory, they are computed each time whenever they are needed. |
| MEMORY_ONLY_SER     | - Serialize data on memory.  |
| MEMORY_AND_DISK     | - Stores the partition on disk if it does not fit in memory. Also called as spilling (data spills to disk).                                      |
| MEMORY_AND_DISK_SER | - Similar to MEMORY_ONLY_SER but spills data to the disk if the data doesn't fit to the memory.  |

# Storage Levels

There are other storage levels (with a numeric 2) specified below. Those represent the same behavior as said above but stores the partitions in two nodes (giving the spark a replication factor of 2).

DISK\_ONLY\_2

MEMORY\_ONLY\_2

MEMORY\_ONLY\_SER\_2

MEMORY\_AND\_DISK\_2

MEMORY\_AND\_DISK\_SER\_2

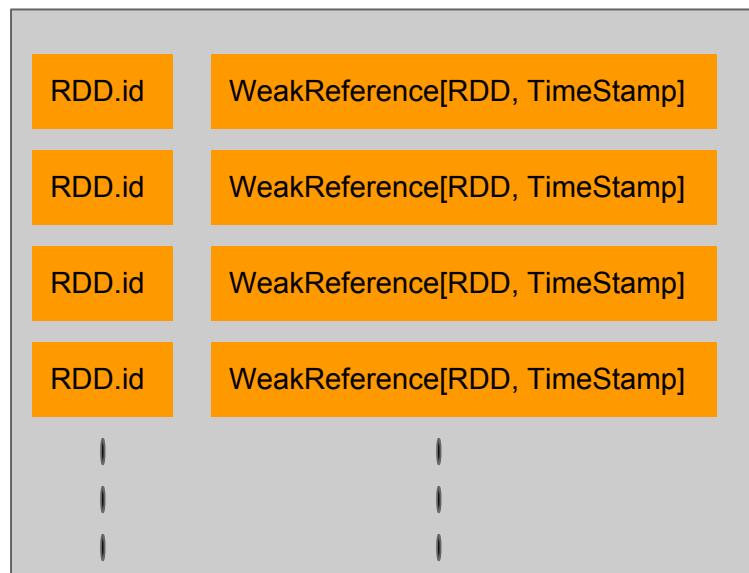
# Persist - Unpersist - cache

```
JavaRDD<String> persistedRDD = rdd.persist(StorageLevel.DISK_ONLY());  
<<..... do complex stuff with persistedRDD.....>>  
persistedRDD.unpersist();
```

```
// Caching is nothing but the StorageLevel.MEMORY_ONLY()  
JavaRDD<String> cachedRDD = rdd.cache();
```

# Persist - Unpersist - cache

When you persist an RDD, spark uses a hashtable to persist in data.



`TimeStampedWeakValueHashMap.scala`

`TimeStampedHashMap.scala`

`TimeStampedValue[V](  
  value: V, timestamp: Long)`

`ConcurrentHashMap.scala`

# Persist - Unpersist - cache

When you persist an RDD, spark uses a hashtable to persist in data.

**TimeStampedWeakValueHashMap.scala**

**TimeStampedHashMap.scala**

TimeStampedValue[V](value: V, timestamp: Long)

**ConcurrentHashMap.scala**

The TimeStampedWeakValueHashMap is the entity

**MetadataCleaner.scala**

# Persist - Unpersist - cache

The TimeStampedWeakValueHashMap is the entity holding references of TimeStampedHashMap. These references are of WeakReference.

The TimeStampedHashMap inherently uses ConcurrentHashMap of TimeStampedValue case object.

The TimeStampedValue contains the data along with the timestamp value in long.

ConcurrentHashMap does not lock the entire map when reading and writing to it. Also it does not throw concurrent modification exception.

# Shuffling

Shuffling is a process by which, the like key elements are brought together by the spark.

Certain API's in spark triggers shuffling. Operations that trigger shuffling are:

repartition(),

coalesce(),

groupByKey(),

reduceByKey(),

cogroup(),

join().

# JDBC RDD

```
JavaSparkContext javaSparkContext = new JavaSparkContext(sparkConfig);

JdbcRDD<Object[]> jdbcRDD =
    new JdbcRDD<>(javaSparkContext.sc(), dbConnection,
                    "select * from employees where emp_no >= ? and emp_no <= ?",
                    10001,
                    49999,
                    10,
                    new                                         MapResult(),
                    ClassManifestFactory$.MODULE$.fromClass(Object[].class));

jdbcRDD.foreach(null);
```

# JDBC RDD

```
static class MapResult extends AbstractFunction1<ResultSet, Object[]> implements Serializable {  
  
    public Object[] apply(ResultSet row) {  
        return JdbcRDD.resultSetToObjectArray(row);  
    }  
}
```

```
static class DbConnection extends AbstractFunction0<Connection> implements Serializable {
```

```
    @Override  
    public Connection apply() {  
        return Connection();  
    }
```



# Spark Streaming

# Spark Streaming

Stream processing of spark allows in scalable, high-throughput, fault-tolerant stream processing of live data streams. By this data is ingested from streaming sources and processed data is saved in file systems.

*StreamingContext* is the entry point for Spark Streaming applications.  
Equivalent to *SparkContext* in core Spark.

# Spark Streaming



# Spark Streaming

Spark uses DStreams (Discretized Streams) as the higher level of abstraction to read data from real time sources.

A streaming context is created in order to process DStreams.

```
JavaStreamingContext jsc = new JavaStreamingContext(sparkConfig, Durations.seconds(10));
```

# Spark Streaming

```
// Start the execution of streams  
jsc.start();
```

```
// Waits on the streaming thread  
jsc.awaitTermination();
```

```
// Stop the streaming process  
jsc.stop();
```

# Spark Streaming

Socket integration

Kafka integration

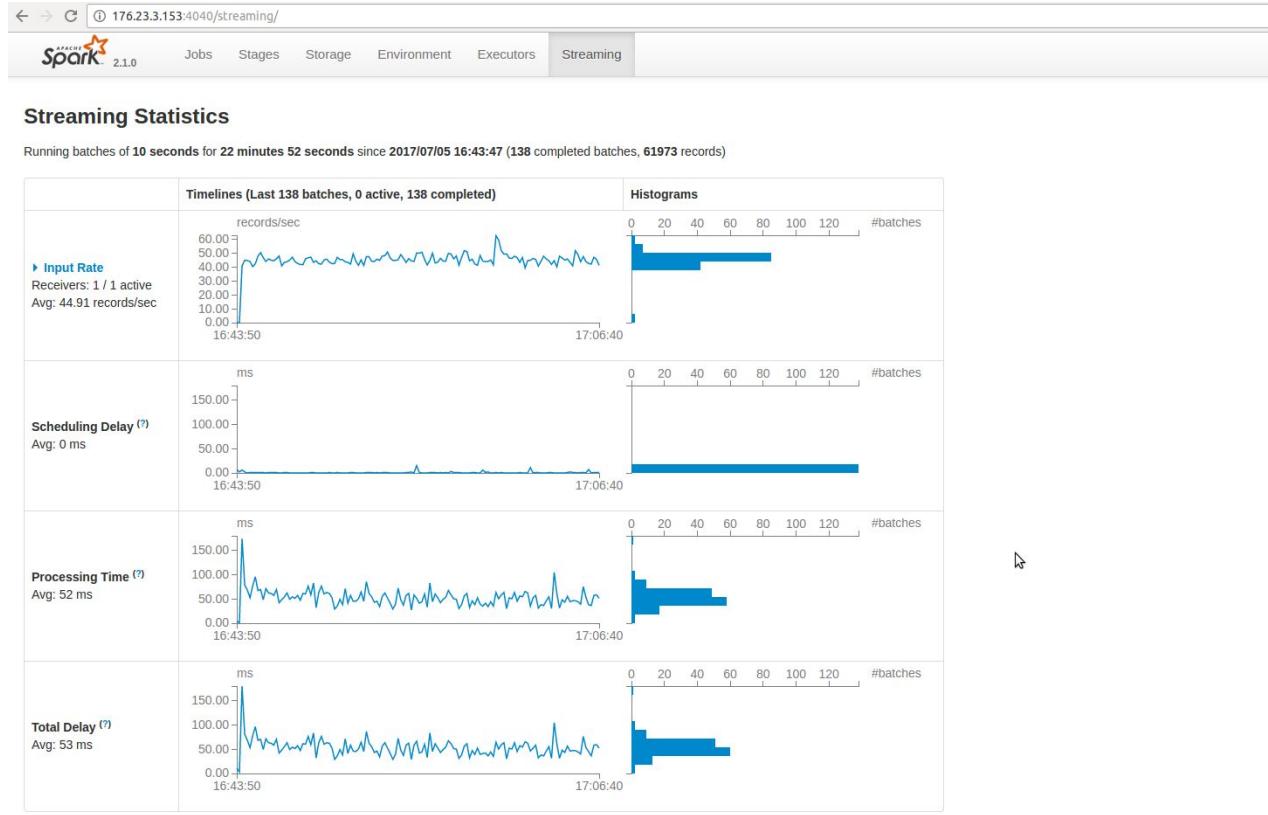
Flume integration

Twitter integration

Kinesis integration

Custom receiver integration

# Spark Streaming UI



chinnasamyad@gmail.com

# Spark Streaming - Reading from socket

```
JavaStreamingContext jsc = new JavaStreamingContext(sparkConfig, Durations.seconds(10));
JavaReceiverInputDStream<String> streamOfLines = jsc.socketTextStream("localhost", 5555);

JavaDStream<String> dStream = streamOfLines.flatMap(z -> Arrays.asList(z.split(" ")));

JavaPairDStream<String, Integer> pairDStream = dStream.mapToPair(
    new PairFunction<String, String, Integer>() {

        @Override
        public Tuple2<String, Integer> call(String input) throws Exception {
            int lengthOfString = input.length();
            return new Tuple2<String, Integer>(input, new Integer(lengthOfString));
        }
    });
pairDStream.print();
jsc.start();
jsc.awaitTermination();
```

# Spark Streaming - Reading from socket

```
val config = new SparkConf  
config.setMaster("local[4]")  
config.setAppName("StreamingSocket")  
  
val streamingContext = new StreamingContext(config, Duration(10))  
val dStream =  
    streamingContext.socketTextStream("localhost",  
        4444, StorageLevel.MEMORY_ONLY)  
  
dStream.print()  
  
streamingContext.start  
streamingContext.awaitTermination
```

# Spark Streaming - Reading from socket

```
$ nc -lk <port_no>
```

```
$ nc -lk 5555
```

The above utility can send data from the command line.

# Spark Streaming - Kafka

```
SparkConf sparkConfig = new SparkConf()  
    .setAppName("KafkaStreaming")  
    .setMaster("local[8]");  
  
JavaStreamingContext jsc = new JavaStreamingContext(sparkConfig, Durations.seconds(5));  
  
Map<String, Integer> params = new HashMap();  
params.put("Lohith_Topic", 1);  
  
JavaPairReceiverInputDStream<String, String> receive =  
    KafkaUtils.createStream(jsc, "localhost:2181", "GROUP-ID", params);  
  
receive.print();  
jsc.start();  
jsc.awaitTermination();
```

# Spark Streaming - Kafka

```
val conf = new SparkConf()
conf.setAppName(HASHCONFIG.APP_STREAMING_NAME)
conf.setMaster(HASHCONFIG.MASTER_URL)
conf.set("es.index.auto.create", "true")

val sparkContext = new SparkContext(conf)
val streamingContext = new StreamingContext(sparkContext, Seconds(1))

val kafkaParams = Map[String, String]("metadata.broker.list" -> "localhost:9092")
val messages = KafkaUtils.createDirectStream[String, String](StringDecoder, StringDecoder)(streamingContext, kafkaParams, Set("HASH"))

messages.print

// val microbatches = mutable.Queue(messages);
// streamingContext.queueStream(microbatches).saveToEs("hash360index/fields")

/*
 * ("custom-index-{date}/customtype")
 * ("custom-index-{date:{YYYY.mm.dd}}/customtype")
 */
messages.foreachRDD((x,y) => EsSpark.saveToEs(x, "hash360index/fields"))

streamingContext.start()
streamingContext.awaitTermination()
chinnasamyad@gmail.com
```

```
object HASHCONFIG extends Enumeration {
  type HASHCONFIG = Value
  var APP_STREAMING_NAME = "Hash360StreamingJOB"
  var MASTER_URL = "local[*]"
  var KAFKA_BROKER = "localhost:9092"
}
```

The Kafka old API

# Spark Streaming - Kafka

```
<dependency>
  <groupId>org.apache.spark</groupId>
  <artifactId>spark-streaming_2.11</artifactId>
  <version>1.6.0</version>
</dependency>

<dependency>
  <groupId>org.apache.spark</groupId>
  <artifactId>spark-streaming-kafka_2.11</artifactId>
  <version>1.6.0</version>
</dependency>

<dependency>
  <groupId>org.elasticsearch</groupId>
  <artifactId>elasticsearch-spark-13_2.11</artifactId>
  <version>5.0.0</version>
</dependency>
```

The Kafka old API

# Spark Streaming - Kafka

```
val sparkConf = new SparkConf()
sparkConf.setAppName("StreamingJOB").setMaster("local[*]")
val sparkContext1 = new SparkContext(sparkConf)
```

```
val streamingContext = new StreamingContext(sparkContext1, Seconds(2))
```

```
import org.apache.kafka.clients.consumer.ConsumerRecord
import org.apache.kafka.common.serialization.StringDeserializer
import org.apache.spark.streaming.kafka010._
import org.apache.spark.streaming.kafka010.LocationStrategies.PreferConsistent
import org.apache.spark.streaming.kafka010.ConsumerStrategies.Subscribe
```

The Kafka new API

# Spark Streaming - Kafka

```
val kafkaParams = Map[String, Object](  
    "bootstrap.servers" -> "localhost:9092",  
    "key.deserializer" -> classOf[StringDeserializer],  
    "value.deserializer" -> classOf[StringDeserializer],  
    "group.id" -> "GROUP-ID",  
    "auto.offset.reset" -> "latest",  
    "enable.auto.commit" -> (false: java.lang.Boolean)  
)
```

```
val topics = Array("DMAC")  
val stream = KafkaUtils.createDirectStream[String, String](  
    streamingContext,  
    PreferConsistent,  
    Subscribe[String, String](topics, kafkaParams)  
)
```

```
stream.map(record => (record.key, record.value)).print()
```

The Kafka new API

```
streamingContext.start()
```

```
streamingContext.awaitTermination()
```

# Spark Streaming - Kafka

```
Dharshekths-MacBook-Pro:bin dharshekthvel$ ./kafka-console-producer.sh --broker-list localhost:9092 --topic DMAC
```

Use the kafka console producer command to publish to the topic.

The Kafka new API

# Spark Streaming - Twitter

```
object SparkTwitterConnector extends App {  
  
    val conf = new SparkConf()  
    conf.setAppName(HASHCONFIG.APP_STREAMING_NAME)  
    conf.setMaster(HASHCONFIG.MASTER_URL)  
    conf.set("es.index.auto.create", "true")  
    conf.set("es.nodes", "localhost:9200")  
  
    val sparkContext = new SparkContext(conf)  
    sparkContext.setLogLevel("WARN")  
  
    val streamingContext = new StreamingContext(sparkContext, Seconds(180))  

```

```
System.setProperty("twitter4j.oauth.consumerKey", "H7gCcVkw7HI8C4hbgEo62h6zF")  
System.setProperty("twitter4j.oauth.consumerSecret",  
"yGgJ8IT5FKoNcrMWCE31tmQqtkzPiSnhIX9XuXIFP4P54XVdxg")  
System.setProperty("twitter4j.oauth.accessToken",  
"837928192170209280-v0p7XGPdehsV0VSTouN6YisQReaEQGt")  
System.setProperty("twitter4j.oauth.accessTokenSecret",  
"6HnqRgWUyQWPSUfKKpthSP7jXUBFfU7hVop6vlfJxUQpL")
```

```
val stream = TwitterUtils.createStream(streamingContext, None)  
//val tags = stream.flatMap { status => status.getHashtagEntities.map(_.getText) }  
  
chinni: val tweets = stream.filter {t =>  
    val tags = t.getText.split(" ").map(_.toLowerCase)
```

# Spark Streaming - Twitter

```
<dependency>
  <groupId>org.elasticsearch</groupId>
  <artifactId>elasticsearch-spark-13_2.11</artifactId>
  <version>5.0.0</version>
</dependency>

<dependency>
  <groupId>org.apache.spark</groupId>
  <artifactId>spark-streaming-twitter_2.10</artifactId>
  <version>1.6.0</version>
</dependency>

<dependency>
  <groupId>org.apache.spark</groupId>
  <artifactId>spark-streaming_2.11</artifactId>
  <version>1.6.0</version>
</dependency>
```

# Spark Streaming - Twitter



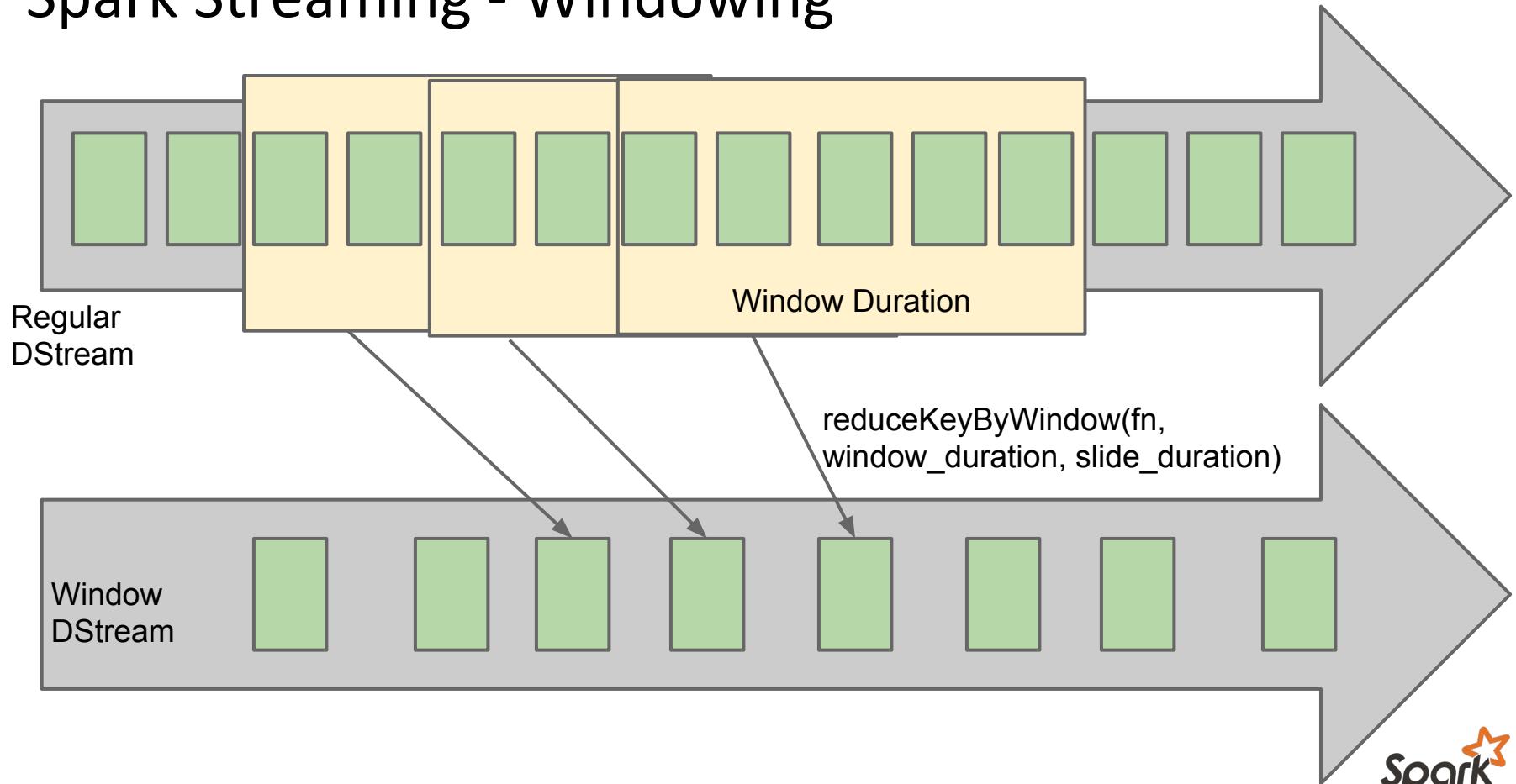
Spark 2.0 - Twitter, Akka, MQTT, ZeroMQ has been moved to Apache Bahir, so that development can be fast in the community.

```
<dependency>
  <groupId>org.apache.bahir</groupId>
  <artifactId>spark-streaming-twitter_2.11</artifactId>
  <version>2.1.0</version>
</dependency>
```

# Spark Streaming - Twitter



# Spark Streaming - Windowing



# Spark Streaming - Windowing

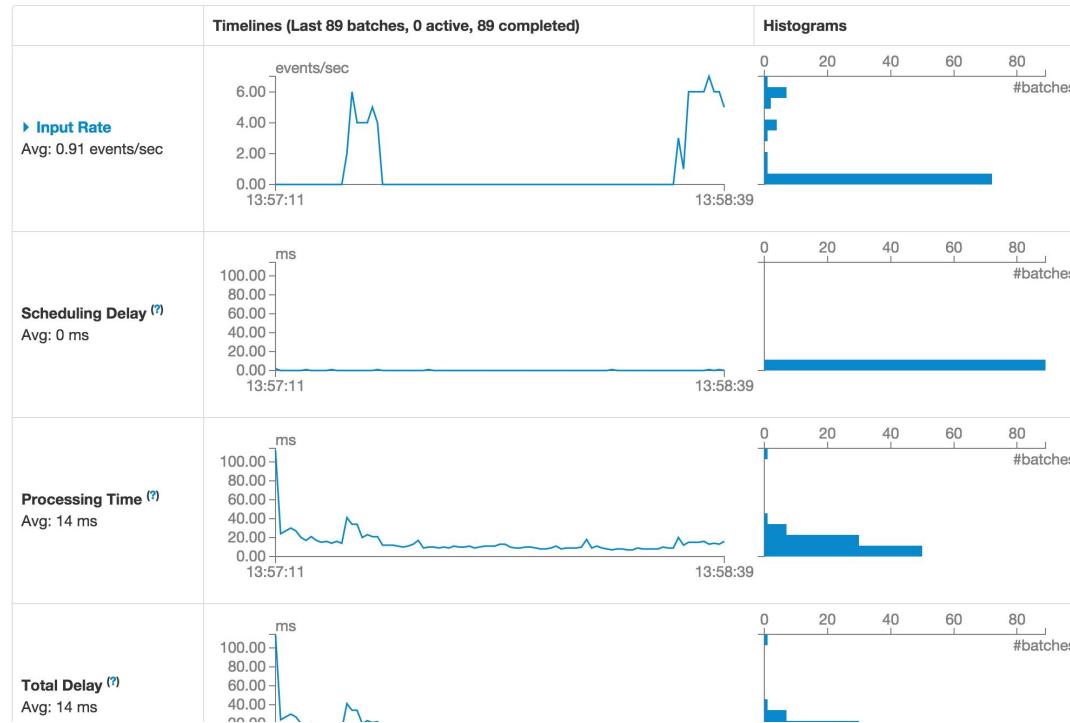
```
JavaStreamingContext jsc = new JavaStreamingContext(sparkConfig, Durations.seconds(5));  
  
JavaReceiverInputDStream<String> streamOfLines = jsc.socketTextStream("localhost", 5555);  
  
streamOfLines.reduceByWindow(new Function2<String, String, String>() {  
  
    @Override  
    public String call(String arg0, String arg1) throws Exception {  
        return arg1.concat(arg0);  
    }  
}, Durations.seconds(15), Durations.seconds(5)).foreach(new Function<JavaRDD<String>, Void>() {  
  
    @Override  
    public Void call(JavaRDD<String> arg0) throws Exception {  
        arg0.foreach(param -> System.out.println("\n\n\n\n\n\n\n\nOUTPUT - " + param));  
        return null;  
    }  
});  
jsc.start();  
jsc.awaitTermination();
```

# Spark Streaming - UI



## Streaming Statistics

Running batches of 1 second for 1 minute 30 seconds since 2015/09/20 13:57:09 (89 completed batches, 81 records)



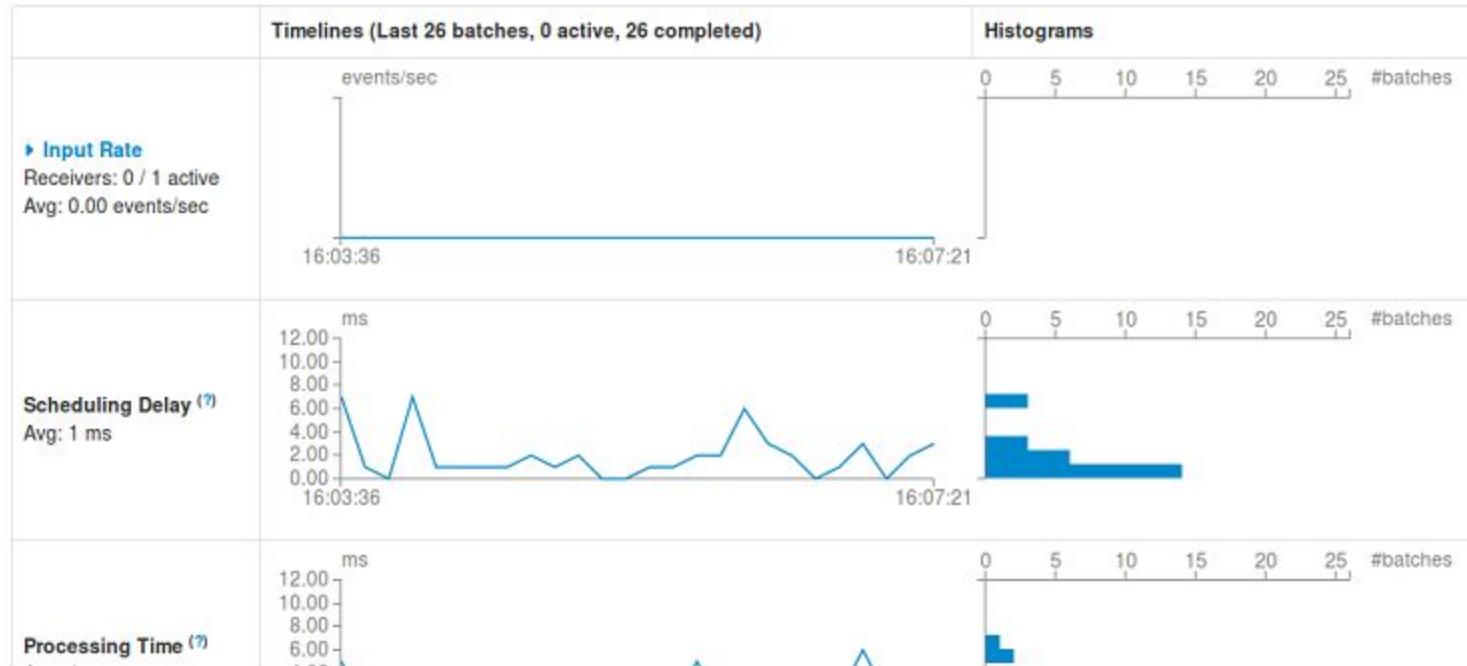
# Spark Streaming - UI

localhost:4040/streaming/ Search

Spark 1.6.0 Jobs Stages Storage Environment Executors Streaming ReadingFile application UI

## Streaming Statistics

Running batches of 9 seconds for 3 minutes 55 seconds since 2016/02/02 16:03:28 (26 completed batches, 0 records)



# Spark Streaming - UI



## Active Batches (0)

Batch Time	Input Size	Scheduling Delay (ms)	Processing Time (ms)	Output Ops: Succeeded/Total	Status
------------	------------	-----------------------	----------------------	-----------------------------	--------

## Completed Batches (last 26 out of 26)

Batch Time	Input Size	Scheduling Delay (ms)	Processing Time (ms)	Total Delay (ms)	Output Ops: Succeeded/Total
2016/02/02 16:07:21	0 events	3 ms	2 ms	5 ms	1/1
2016/02/02 16:07:12	0 events	2 ms	2 ms	4 ms	1/1
2016/02/02 16:07:03	0 events	0 ms	2 ms	2 ms	1/1

# Spark Streaming - Connecting to other sources

```
JavaStreamingContext jsc = new JavaStreamingContext(config, Durations.seconds(10));
JavaDStream<Status> tweets = TwitterUtils.createStream(jsc);

JavaDStream<String> statuses = tweets.map(
    new Function<Status, String>() {
        public String call(Status status) {
            String tweets = status.getText();
            if (tweets.equals("LOHITHAAA"))
                return status.getText();
            return "";
        }
    });
statuses.print();
```

**FlumeUtils.createStream()**

**KinesisUtils.createStream**

# Spark Custom Receiver

```
public class MyOwnCustomReceiver extends Receiver<TitanicBean> {

    public MyOwnCustomReceiver(StorageLevel storageLevel) {
        super(StorageLevel.MEMORY_ONLY());
    }

    @Override
    public void onStart() {
        new Thread() {
            @Override public void run() {
                <<listen_to_a_socket>>
            }
        }.start();
    }

    @Override
    public void onStop() {
    }
}
```

# Spark Custom Receiver

```
SparkConf sparkConfig = new SparkConf()  
    .setAppName("SparkStreaming")  
    .setMaster("local[5]");  
  
JavaStreamingContext jsc = new JavaStreamingContext(sparkConfig, Durations.seconds(10));  
  
JavaReceiverInputDStream<String> streamOfLines  
    = jsc.receiverStream(new MyOwnCustomReceiver(StorageLevel.MEMORY_ONLY()));
```

# Back pressure

In spark streaming, there can be at times where there are tons of data that are popping in.

You can see that there is a sudden spike of data when we are listening on a twitter or on a particular socket.

In order to handle to such a huge volume of data ingestion the back pressure algorithm had been introduced in spark.

Spark back pressure can be enabled through the config setting  
`conf.set("spark.streaming.backpressure.enabled","true")`.

# Custom Partitioning - Spark

By default, spark uses HashPartitioning for manipulating data in memory.

Let's write a custom partitioner for partitioning the data in memory.

For to do a custom partitioning, you need to have a PairRDD data structure.

# Custom Partitioning - Spark

The below Partitioner partitions the data based on the year, which is a Int field. The usage is shown on the next slide.

```
class YearPartitioner extends Partitioner {  
  
    override def numPartitions: Int = {  
        15  
    }  
  
    override def getPartition(key: Any): Int = {  
        val k = key.asInstanceOf[Int]  
        if (k > 2000)  
            10  
        else if (k > 1900 && k < 2000)  
            11  
        else  
            12  
    }  
}
```

YearPartitioner.scala

chinnasamyad@gmail.com

# Custom Partitioning - Spark

```
-- Our usual stuffs. Create the data ---  
val sparkContext = new SparkContext(sparkConfig)  
val list = List(1923, 1800, 2012, 1834, 1987, 2014);  
val listRDD = sparkContext.parallelize(list)  
  
-- Create the Pair RDD for partitioning --  
val pairRDD = listRDD.map(x => (x,0))  
  
val partitionedData = pairRDD.partitionBy(new YearPartitioner)  
  
// partitionedData.mapPartitions(dataIterator =>  
//     dataIterator.map(dataInfo => println(dataInfo)))  
//     .collect()  
  
val mapPartitionIndexRDD = partitionedData.mapPartitionsWithIndex((partitionIndex, dataIterator) =>  
    dataIterator.map(dataInfo => "Partition is - " + partitionIndex + " - Data = " + dataInfo))  
  
mapPartitionIndexRDD.saveAsTextFile("/home/dharshekthvel/dele199")  
  
print("Partitioner " + partitionedData.partitioner)
```

YearPartitioner.scala

chinnasamyad@gmail.com

# Spark SQL

SQL Query Engine for Spark

# Spark SQL

SparkSQL is the query engine querying structured data sitting on top of an abstraction called DataFrames.

```
SparkConf sparkConfig = new SparkConf()  
    .setAppName("DataFrames")  
    .setMaster("local[5]");
```

```
JavaSparkContext javaSparkContext = new JavaSparkContext(sparkConfig);
```

```
SQLContext sc = new SQLContext(javaSparkContext);  
DataFrame df = sc.read().json("file:///Users/tester/ac/ycy.json");  
df.printSchema();
```

# Spark SQL

SparkSQL is the query engine querying structured data sitting on top of an abstraction called DataFrames.

```
val sparkSession = SparkSession.builder()  
    .appName("SQLJob")  
    .master("local[*]")  
    .getOrCreate()
```

```
val salesCSV = sparkSession.sqlContext.read.option("header","true")  
    .csv("/home/dharshekthvel/Desktop/ac/code/scalatrainingintellij/data/sales.csv")
```

Spark 2.0

# Spark SQL

DataFrame is the abstraction for SparkSQL. Internally it is composed of data organized as named columns.

In the spark source, refer DataFrame.scala for more insight.

# Spark SQL

A Dataframe is a distributed collection of data organized into columns.  
Nothing but a table of a relational database.

With Spark 2.0, DataFrame is a synonym of Dataset[Row].  
The Dataset is more recommended.

Dataset store data in a optimized binary format, in heap memory, to avoid the cost of deserialization and garbage collection. Spark creates its own optimized byte-code for accessing the data directly.

# Spark SQL

## Dataset

A dataset is a strongly typed immutable collection of objects that are mapped to a relational schema.

Dataset are encoded objects with the tungsten optimizer of spark which helps in serialization and memory efficiency.

Dataframes are nothing but datasets organized as named columns.

# Spark SQL

Spark 2.0

SparkSQL from Spark 2.0 supports majority of the queries.  
It can run 99 queries of the TPC-DS specification.

In Spark 2.0, there are significant changes, there is no backward compatibility.  
You need to migrate the code from Spark 1.x to Spark 2.x

Subquery support has been added in Spark 2.0

Transaction Processing Performance Council (TPC) sets a benchmark for SQL engines.  
Visit [www.tpc.org](http://www(tpc.org)

# Spark SQL

Creation of dataset. Dataset are created by two ways:

```
[1] val dataset = sparkSession.createDataset(List("IN", "US", "HK"))
```

```
[2] val csvDataSetText =
sparkSession.read.text("hdfs://localhost:9000/covtype.info").as[String]
csvDataSetText.foreach(x => println(x))
```

# Spark SQL

```
SparkConf sparkConfig = new SparkConf()  
    .setAppName("ReadLogFile")  
    .setMaster("local[5]");
```

```
JavaSparkContext javaSparkContext = new JavaSparkContext(sparkConfig);  
  
SQLContext sqlContext = new SQLContext(javaSparkContext);  
DataFrame df = sqlContext.read().json("file:///Users/koteshwar/world_bank.json");  
  
df.show();
```

# Spark SQL

approvalfy	board_approval_month	boardapprovaldate	borrower	closingdate	country_namecode	countrycode	countryname	countryshortname	doctylenav
1999I	November	2013-11-12T00:00:00Z	FEDERAL DEMOCRATI...	2018-07-07T00:00:00Z	Federal Democrati...	ETI	Federal Democrati...	Ethiopia	Project Informati...
2015I	November	2013-11-04T00:00:00Z	GOVERNMENT OF TUN...	null	Republic of Tunis...	TNI	Republic of Tunisia	Tunisia	Project Informati...
2014I	November	2013-11-01T00:00:00Z	MINISTRY OF FINAN...	null	Tuvalu\$!TVI	TVI	Tuvalu	Tuvalu	Resettlement Plan...
2014I	October	2013-10-31T00:00:00Z	MIN. OF PLANNING ...	null	Republic of Yemen...	RYI	Republic of Yemen Yemen, Republic of	Yemen	Procurement Plan...
2014I	October	2013-10-31T00:00:00Z	MINISTRY OF FINANCE	2019-04-30T00:00:00Z	Kingdom of Lesotho...	LSI	Kingdom of Lesotho	Lesotho	Project Informati...
2014I	October	2013-10-31T00:00:00Z	REPUBLIC OF KENYA	null	Republic of Kenya...	KEI	Republic of Kenya	Kenya	Integrated Safegu...
2014I	October	2013-10-29T00:00:00Z	GOVERNMENT OF INDIA	2019-06-30T00:00:00Z	Republic of India...	INI	Republic of India	India	Project Appraisal...
2014I	October	2013-10-29T00:00:00Z	PEOPLE'S REPUBLIC...	null	People's Republic...	CNI	People's Republic...	China	Project Appraisal...
2014I	October	2013-10-29T00:00:00Z	THE GOVERNMENT OF...	2018-12-31T00:00:00Z	Republic of India...	INI	Republic of India	India	Project Appraisal...
2014I	October	2013-10-29T00:00:00Z	THE KINGDOM OF MO...	2014-12-31T00:00:00Z	Kingdom of Morocco	MAI	Kingdom of Morocco	Morocco	Program Document...
2014I	October	2013-10-25T00:00:00Z	GOVERNMENT OF SOU...	null	Republic of South...	SSI	Republic of South...	South	Sudan Project Paper, Pro...
2014I	October	2013-10-25T00:00:00Z	null	2017-12-31T00:00:00Z	Republic of India...	INI	Republic of India	India	Project Appraisal...
2014I	October	2013-10-24T00:00:00Z	GOVERNMENT OF GHANA	2019-06-30T00:00:00Z	Republic of Ghana...	GHI	Republic of Ghana	Ghana	Project Appraisal...
2014I	October	2013-10-22T00:00:00Z	GOVERNMENT OF TIM...	null	Democratic Republ...	TPD	Democratic Republ...	Timor-Leste	Integrated Safegu...
2014I	October	2013-10-22T00:00:00Z	GOVERNMENT OF JORDANI	null	Hashemite Kingdom...	JOI	Hashemite Kingdom...	Jordan	null
2014I	October	2013-10-17T00:00:00Z	MINISTRY OF FINANCE	2019-04-30T00:00:00Z	Samoa\$!WSI	WSI	Samoan	Samoa	Environmental Ass...
2014I	October	2013-10-17T00:00:00Z	MINISTRY OF FINANCE	2015-12-31T00:00:00Z	Samoa\$!WSI	WSI	Samoan	Samoa	Project Appraisal...
2014I	October	2013-10-16T00:00:00Z	MINISTRY OF FINAN...	null	Republic of Madag...	MGI	Republic of Madag...	Madagascar	Integrated Safegu...
2014I	October	2013-10-16T00:00:00Z	ROYAL GOVERNMENT ...	null	Kingdom of Cambod...	KHI	Kingdom of Cambod...	Cambodia	Project Informati...
2014I	October	2013-10-10T00:00:00Z	MINISTRY OF FINANCE	null	Kingdom of Morocco...	MAI	Kingdom of Morocco	Morocco	Project Informati...

# Spark SQL - select variants

```
df.select("borrower").show();
```

```
df.select("borrower", "country_namecode").show();
```

```
df.select(df.col("borrower"), df.col("country_namecode")).show();
```

# Spark SQL - select variants

```
df.select(df.col("borrower"), df.col("grantamt").plus(1)).show();
```

```
df.select(df.col("borrower"), df.col("grantamt").gt(15000000)).show();
```

# Spark SQL - operations

```
df.groupBy("countryshortname").count().show();
```

# DataFrames - Writing the output

```
JavaSparkContext javaSparkContext = new JavaSparkContext(sparkConfig);

    SQLContext sqlContext = new SQLContext(javaSparkContext);
    DataFrame df =
sqlContext.read().json("file:///Users/apple/world_bank.json");

    df.select("borrower",
"country_namecode").write().json("file:///Users/apple/out");
```

# DataFrames - Writing the output

```
Apples-MacBook-Pro:out apple$ ls
_SUCCESS          part-00000      part-00001
Apples-MacBook-Pro:out apple$ 
Apples-MacBook-Pro:out apple$ 
Apples-MacBook-Pro:out apple$ 
Apples-MacBook-Pro:out apple$ cat part-00001
{"borrower":"GOVERNMENT OF COMOROS","country_namecode":"Union of the Comoros(!$!KM"}
 {"borrower":"GOVERNMENT OF SRI LANKA","country_namecode":"Democratic Socialist Republic of Sri Lan(!$!LK"}
 {"borrower":"HALK BANK; VAKIF BANK; ZIRAAT BANK","country_namecode":"Republic of Turkey(!$!TR"}
 {"borrower":"GOVERNMENT OF MOROCCO","country_namecode":"Kingdom of Morocco(!$!MA"}
 {"borrower":"GOVERNMENT OF MAURITIUS","country_namecode":"Republic of Mauritius(!$!MU"}
 {"borrower":"REPUBLIC OF ARMENIA","country_namecode":"Republic of Armenia(!$!AM"}
 {"borrower":"HALKBANK,VAKIFBANK,ZIRAATBANK","country_namecode":"Republic of Turkey(!$!TR"}
 {"borrower":"GOVERNMENT OF MAURITIUS","country_namecode":"Republic of Mauritius(!$!MU"}
 {"borrower":"UNITED REPUBLIC OF TANZANIA","country_namecode":"United Republic of Tanzania(!$!TZ"}
```

# Spark SQL - Writing a Select Query on View

```
val jsonDataFrame = sparkSession.read.json("examples/src/main/resources/people.json")  
  
val csvDataFrame =  
sparkSession.read.csv("/Users/dharshekthvel/ac/training/sparktraining/data/undata1.csv")  
  
csvDataFrame.createTempView("DATATABLE")  
csvDataFrame.createOrReplaceTempView("MYOWNTABLE")  
csvDataFrame.createGlobalTempView("STRUCTUREDTABLE")  
  
sparkSession.sql("SELECT _c0,_c1 FROM DATATABLE").show(100)
```

# Spark SQL - Select Query

Temp views in Spark are session scoped and terminates when the session terminates.

View that has to be shared among all sessions are created under createGlobalTempView

Global tables are accessed with global\_temp table.

```
csvDataFrame.createGlobalTempView("STRUCTURETABLE")
```

```
sparkSession.sql("SELECT _c0,_c1 FROM global_temp.STRUCTURETABLE").show(100)
```

# Spark SQL - Select Query

```
JavaSparkContext javaSparkContext = new JavaSparkContext(sparkConfig);  
  
SQLContext sqlContext = new SQLContext(javaSparkContext);  
  
DataFrame df = sqlContext.read().json("file:///Users/apple/world_bank.json");  
df.registerTempTable("WORLD_BANK");  
  
DataFrame borrower_country = sqlContext.sql("select borrower, country_namecode from WORLD_BANK");  
  
borrower_country.show();
```

The registerTempTable is deprecated from 2.0, use createTempView

# Spark SQL - Writing a UDF

```
/**  
 * borrowerConverter is the User Defined Function (UDF)  
 */  
UDF1 borrowerConverter = new UDF1<String, String>() {  
  
    @Override  
    public String call(String input) throws Exception {  
  
        if (input == null)  
            return "_EMPTY_";  
        else if (input.equals("MINISTRY OF FINANCE"))  
            return "FINANCE";  
        else  
            return input;  
    }  
};
```

# Spark SQL - Writing a UDF

```
JavaSparkContext javaSparkContext = new JavaSparkContext(sparkConfig);

SQLContext sqlContext = new SQLContext(javaSparkContext);
DataFrame df = sqlContext.read().json("file:///C:/ac/spark/code/sparktraining/data/world_bank.json");
df.registerTempTable("WORLD_BANK");

sqlContext.udf().register("borrowerConverter", borrowerConverter, DataTypes.StringType);

DataFrame borrowerCountry = sqlContext.sql("select borrowerConverter(borrower), country_namecode
                                             from WORLD_BANK");

borrowerCountry.show();

javaSparkContext.close();
```

# Spark SQL - JDBC

```
val sparkContext = new SparkContext(config)

val sqlContext = new SQLContext(sparkContext)

val df = sqlContext.read.format("jdbc").options(
    Map("url" -> "jdbc:mysql://127.0.0.1:3306/slz_core",
        "user" -> "slz02",
        "password" -> "slz02@123",
        "dbtable" -> "regions"))
    .load()

df.select("name").show()

sparkContext.stop()
```

Don't use JDBCRDD. JDBCRDD is not recommended.

# Spark SQL - Saving Data...

```
val requiredDataFrame = sparkSession.sql("SELECT _c0,_c1 FROM global_temp.STRUCTUREDTABLE")  
  
requiredDataFrame.write.csv("/Users/dharshekthvel/ac/out.csv")  
  
requiredDataFrame.write.parquet("/Users/dharshekthvel/ac/out.parquet")  
  
requiredDataFrame.write.json("/Users/dharshekthvel/ac/out.json")
```

Any DataFrame can be saved as a csv, parquet, or json. The data gets saved in a directory as a partition.

```
/Users/dharshekthvel/ac/out.csv  
[Dharshekths-MacBook-Pro:out.csv dharshekthvel$ ls  
_SUCCESS  
part-00000-93c2d6ed-1512-4609-a20c-b3dad78719f0.csv  
Dharshekths-MacBook-Pro:out.csv dharshekthvel$ ]
```

# Spark SQL - Know your data

```
val sparkSession = SparkSession.builder().appName("SQLJob").master("local[*]").getOrCreate()
```

```
import sparkSession.implicits._
```

```
val electionCSV = sparkSession.sqlContext.read.option("header", "true")  
.csv("/home/dharshekthvel/Desktop/ac/code/scalatrainingintellij/data/electiondata.csv")
```

```
electionCSV.describe("CQ").show()
```

```
electionCSV.describe("REP_NAME").show()
```

summary	CQ
count	534702
mean	309.86206896551727
stddev	182.16032657046628
min	10
max	99

summary	REP_NAME
count	534702
mean	null
stddev	null
min	ABERCROM
max	YOUNG

Statistical Exploration

# Spark SQL - Change your schema

```
val toDouble = udf[Double, String](_.toDouble)
```

Change your schema-datatype using the UDF function.

```
salesCSV.show(100)  
salesCSV.printSchema()
```

```
val salesDF = salesCSV.toDF()
```

```
val salesModifiedSchema = salesCSV.withColumn("Revenue", toDouble(salesDF.col("Revenue")))
```

```
salesModifiedSchema.printSchema()
```

*One of the common operations done. In the above code, the column Revenue is converted from String to Double. This can be viewed by using the printSchema() before and after the schema modification.*

# Dataset API

# Dataset API

Introduced in Spark 1.6, Dataset is a distributed collection of data.

Dataset API's are infact RDD's which use specific encoders to serialize the object rather than using java serialization or kryo serialization.

Optimized API's in computation.

A Dataframe is a Dataset organized into named columns.

Always use this. All futuristic API's of Spark will be using dataset's heavily.

# Dataset API

A Dataset API executes much faster than a native RDD.

Datasets uses low memory footprint.

Datasets uses encoders which are highly optimized and uses runtime code generation to build custom bytecode for serialization/deserialization.

# Dataset API

```
List<String> data = Arrays.asList("Spark", "is", "doing", "awesome");

SparkConf sparkConfig = new SparkConf()
    .setAppName("DatasetAPI")
    .setMaster("local[8]");

JavaSparkContext javaSparkContext = new JavaSparkContext(sparkConfig);

SQLContext sqlContext = new SQLContext(javaSparkContext);

Dataset<String> dataset = sqlContext.createDataset(data, Encoders.STRING());

dataset.foreach(eachElement -> System.out.println(eachElement));
```

# Dataset API

```
val config = new SparkConf()

config.setMaster("local")
config.setAppName("DatasetElucidation")

val sc = new SparkContext(config)

val sqlContext = new SQLContext(sc)

val list = List("A", "Simple", "Spark", "List")

import sqlContext.implicits._

val lines = sqlContext.createDataset(list).as[String]

lines.foreach(x => println(x))
```

# Dataset API

```
val sparkSession = SparkSession.builder()
    .appName("SparkJOB").master("local[*]")
    .getOrCreate()

import sparkSession.implicits._

val dataset = sparkSession.createDataset(List(
    "INVESTMENT_BANK_TIMELINE_01",
    "INVESTMENT_BANK_TIMELINE_02",
    "INVESTMENT_BANK_TIMELINE_003"))

val lengthDataset = dataset.map(each => each.length)

lengthDataset.foreach(x => println(x))
```

# Dataset API

## Spark 1.X

Dataframe is structured abstraction over the RDD.

## Spark 2.X

From Spark 2.x, the dataframe is an alias of Dataset[Row].

Dataset supports Structured Querying using DSL and SQL. Also it supports functional API's.

# Dataset API - Advantages

Gives high performance throughput.

# Catalog API - Introduced in Spark 2.0

Dataset and Dataframe supports structured data analysis in Spark.

The Catalog API is used for accessing metadata on the SQL Context of Spark.

Catalog is available in SparkSession

# Catalog API - Introduced in Spark 2.0

```
sparkSession.catalog.refreshTable("")
```

```
sparkSession.catalog.cacheTable("")
```

```
sparkSession.catalog.clearCache()
```

```
sparkSession.catalog.dropTempView("")
```

```
sparkSession.catalog.dropGlobalTempView("")
```

```
sparkSession.catalog.createExternalTable("", "")
```

# Apache Parquet

# Parquet

Parquet is the file format for storing data in a columnar format.

Spark SQL heavily embraces Parquet format for operating on SQL.

Use Parquet file format when and wherever possible with SQL.  
Because parquet performs much better than CSV or JSON.

The screenshot shows the official Apache Parquet website at <https://parquet.apache.org>. The header includes the Apache logo, the Parquet logo (a blue diamond shape with white lines), and navigation links for Documentation, Download, Presentations, Adopters, and Get Involved. Below the header, a breadcrumb trail shows 'Apache Software Foundation' and 'Apache Parquet'. A main text box states: 'Apache Parquet is a [columnar storage](#) format available to any project in the Hadoop ecosystem, regardless of the choice of data processing framework, data model or programming language.'

# Parquet

Parquet is a columnar storage of data and so aggregation queries are faster.

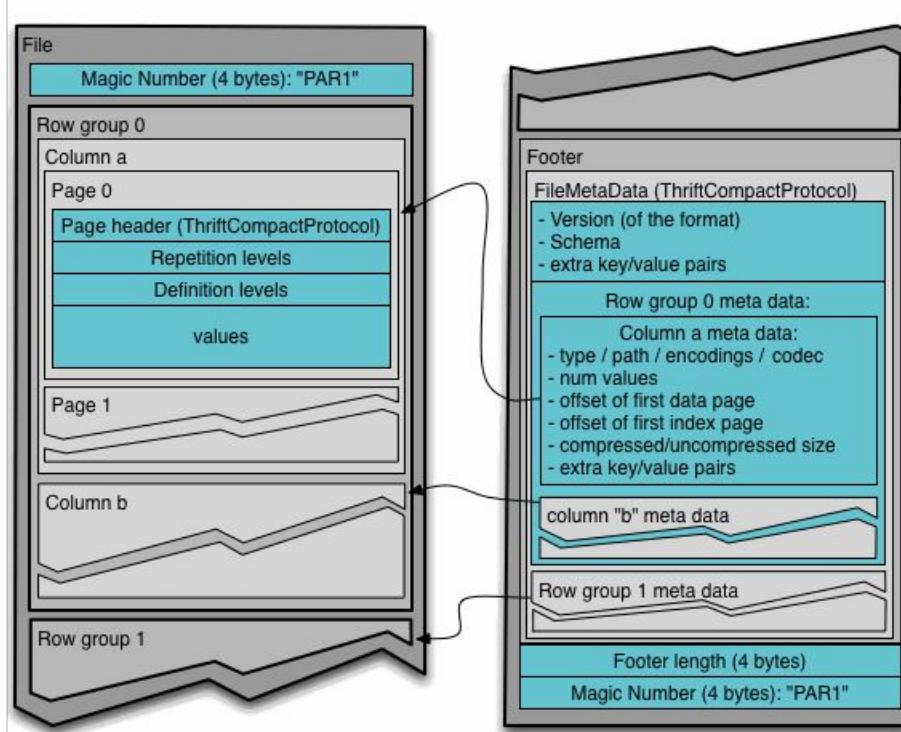
The required columns are read from the disk.

Data has the schema.

Parquet supports compression in data storage.

Parquet default compression format is Snappy.

# Parquet



Source : <https://parquet.apache.org/documentation/latest/>

# Parquet

# Parquet

```
val sparkSession = SparkSession.builder().appName("ParqueJOB")
    .master("local[*]").getOrCreate()
```

```
val salesCSV = sparkSession.sqlContext.read.option("header","true")
    .csv("/home/dharshekthvel/Desktop/ac/code/scalatrainingintellij/data/salessimple.csv")
```

```
salesCSV.write
    //.option("compression","snappy")
    .option("compression","gzip")
    .partitionBy("CookingGear","CampingEquipment")
    .parquet("/home/dharshekthvel/Desktop/ac/code/scalatrainingintellij/data/sales_partition_gzip.parquet")
```

# Parquet

```
val sparkSession = SparkSession.builder().appName("ParqueJOB")
    .master("local[*]").getOrCreate()
```

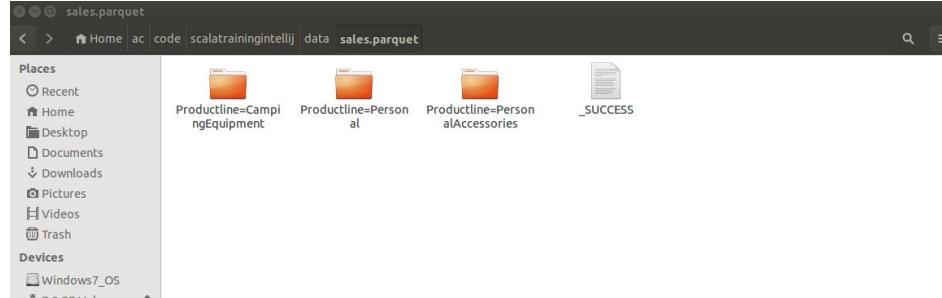
```
val salesCSV = sparkSession.sqlContext.read.option("header","true")
    .csv("/home/dharshekthvel/Desktop/ac/code/scalatrainingintellij/data/salessimple.csv")
```

```
salesCSV.write
    //.option("compression","snappy")
    .option("compression","gzip")
    .partitionBy("CookingGear","CampingEquipment")
    .parquet("/home/dharshekthvel/Desktop/ac/code/scalatrainingintellij/data/sales_partition_gzip.parquet")
```

# Parquet

```
val salesCSV = sparkSession.sqlContext.read.option("header","true")  
.csv("/home/dharshekthvel/Desktop/ac/code/scalatrainingintellij/data/salessimple.csv")
```

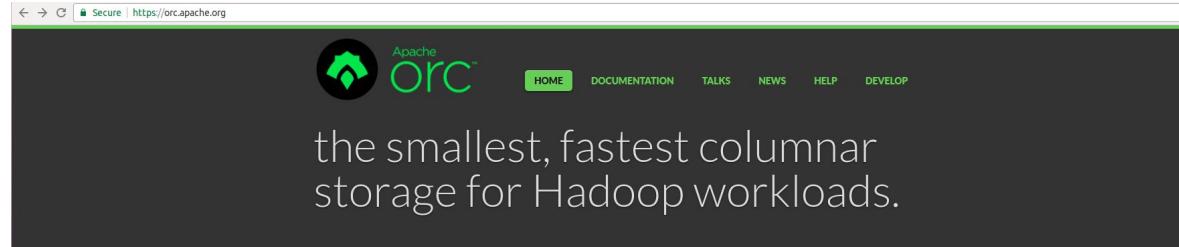
```
salesCSV.write.partitionBy("Productline")  
.parquet("/home/dharshekthvel/Desktop/ac/code/scalatrainingintellij/data/sales.parquet")
```



# Apache ORC

# Apache ORC

Apache ORC is the Optimized Row Columnar format for the Big Data.



# Apache ORC

```
val salesCSV = sparkSession.sqlContext.read.option("header","true")
.csv("/home/dharshekthvel/Desktop/ac/code/scalatrainingintellij/data/salessimple.csv")

salesCSV.write.orc("/home/dharshekthvel/Desktop/ac/code/scalatrainingintellij/data/saless
_partitions.orc")
```

# Apache ORC

part-00000-3ac92b1d...59e8d62c7.snappy.orc x

ORC [ 681 B ]

[ 681 B ] P [ 681 B ] \00\00

\00\00\00\00\00 " "

2004 2010 P [ 681 B ] \00\00

6

( 00\00,0 / )")

CampingEquipment PersonalAccessories | P [ 681 B ] \00\00

%

\00\00\00\00\00 " "

CookingGear Watches |>P [ 681 B ] \00\00

.

\00\00\00\00\00 % "

MaxGizmo TrailChefWaterBag | tP [ 681 B ] \00\00

\00\00\00 " "

Mail Web | P [ 681 B ] \00\00

(

\00\00,0 ! "

Australia UnitedStates | HP [ 681 B ] \00\00

%

\00\00,0 315044.33 | 315044.33 | HP [ 681 B ] \00\00

%

\00\00,0 437477.15 | 437477.15 | HP [ 681 B ] \00\00

%

\00\00,0 158371.76 | 158371.76 | HP [ 681 B ] \00\00

\00\00\00 " "

66385 | 66385 | P [ 681 B ] \00\00

.

\00\00\00 " "

2.55285714

2.55285714 | PP [ 681 B ] \00\00

\00\00\00 " "

6.59 | 6.59 | P [ 681 B ] \00\00

%

\00\00\00 " "

156672.57 | 156672.57 | HP [ 681 B ] \00\00

.

\00\00\00 " "

5.19571429

5.19571429 | PP [ 681 B ] \00\00 2004200820072010 | \00\00 | \00\00B | \00\00N | | | P [ 681 B ] + CampingEquipmentPersonal | (Accesso

# Spark Performance Tuning

Keep your memory always occupied and make it thinking about something.

# Spark Performance Tuning

- ~ Try to avoid sending large amounts of data to the driver program. For eg., the collect() operation sends in data to the driver program.
- ~ Process data in the workers not in the driver. Instead store it in the hdfs.
- ~ Store large results in the hdfs
- ~ Use kryo serialization
- ~ Use accumulators and shared variables to share data.
- ~ Use Alluxio. It is now the top memory oriented storage.

# Spark Performance Tuning

`spark.dynamicAllocation.enabled` By default false. Make it to true. Based on varying loads, enable the dynamic allocation to true.

Too much memory for the executor will increase overhead in Executor memory resulting in garbage collection.

# Spark Performance Tuning

The default form of serialization, which is the java serialization and it is slow.  
Kryo serializer solves in the performance issues of the basic serialization.

In order to enable kryo serialization, the "spark.serializer" configuration should be set to "org.apache.spark.serializer.KryoSerializer" in spark configuration as shown below.

```
SparkConf sparkConfig = new SparkConf()  
    .set("spark.serializer", "org.apache.spark.serializer.KryoSerializer")  
    .registerKryoClasses(new Class[]{TitanicBean.class})  
    .setAppName("ReadFile")  
    .setMaster("local[8]");
```

# Spark Performance Tuning

If an RDD, will be used multiple times, persist to avoid recomputation. After done, unpersist the data whenever not required.

```
JavaRDD<String> persistedRDD = rdd.persist(StorageLevel.DISK_ONLY());
```

```
<<some operation on persistedRDD>>
```

```
persistedRDD.unpersist();
```

# Compress your broadcast data

`spark.broadcast.compress = true`

By default it is true. But make it false, when you are using too much broadcasting.  
You can minimize the cost to decompress.

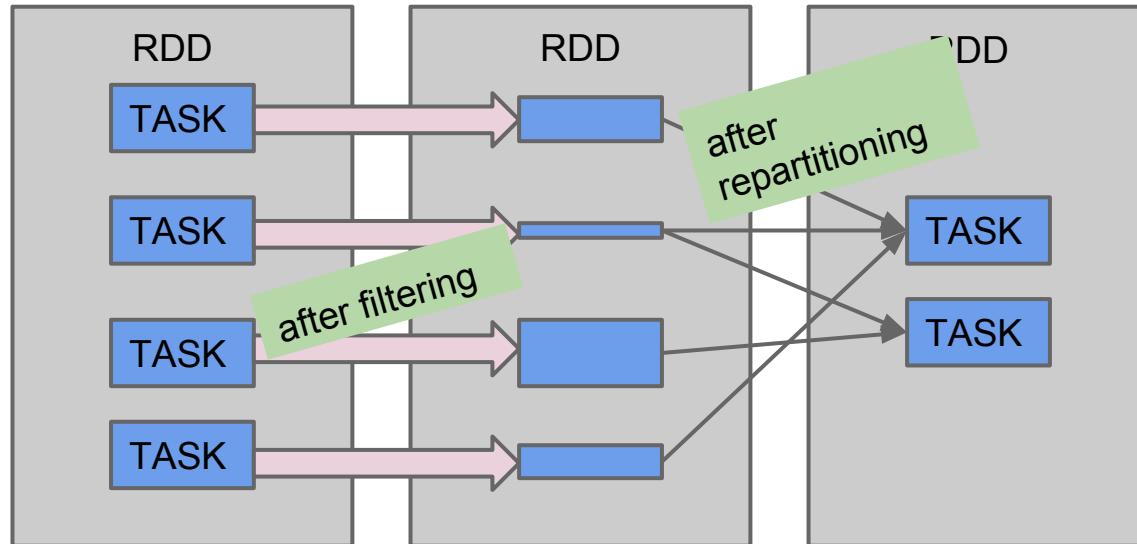
Varies from use case to use case.

# Spark Performance Tuning

Filter can result in data with small partitions. So repartition() it.

```
JavaRDD<String> rdd = javaSparkContext.textFile("file:///Users/apple/undata1.csv");
```

```
rdd.filter(param -> param.split(",")[0].equals("\\"India\\\""))
.repartition(2)
.foreach(System.out::println);
```



chinnasamyad@gmail.com

# Spark Performance Tuning

Don't pass in large amounts of data into function transformations.

```
Map<Integer, String> hugeHashMap = new HashMap<>();  
hugeHashMap.put(1, "Turing");  
hugeHashMap.put(...)  
  
rdd.map(new Function<String, String>() {  
  
    @Override  
    public String call(String arg0) throws Exception {  
        System.out.println(hugeHashMap.get(1));  
        .....  
    }  
  
}).count();
```

# Spark Performance Tuning

Don't pass in large amounts of data into parallel functions,  
Instead  
use broadcast or parallelize to a RDD.

```
javaSparkContext.parallelizePairs(list)
```

```
javaSparkContext.parallelize(list)
```

# Spark Performance Tuning

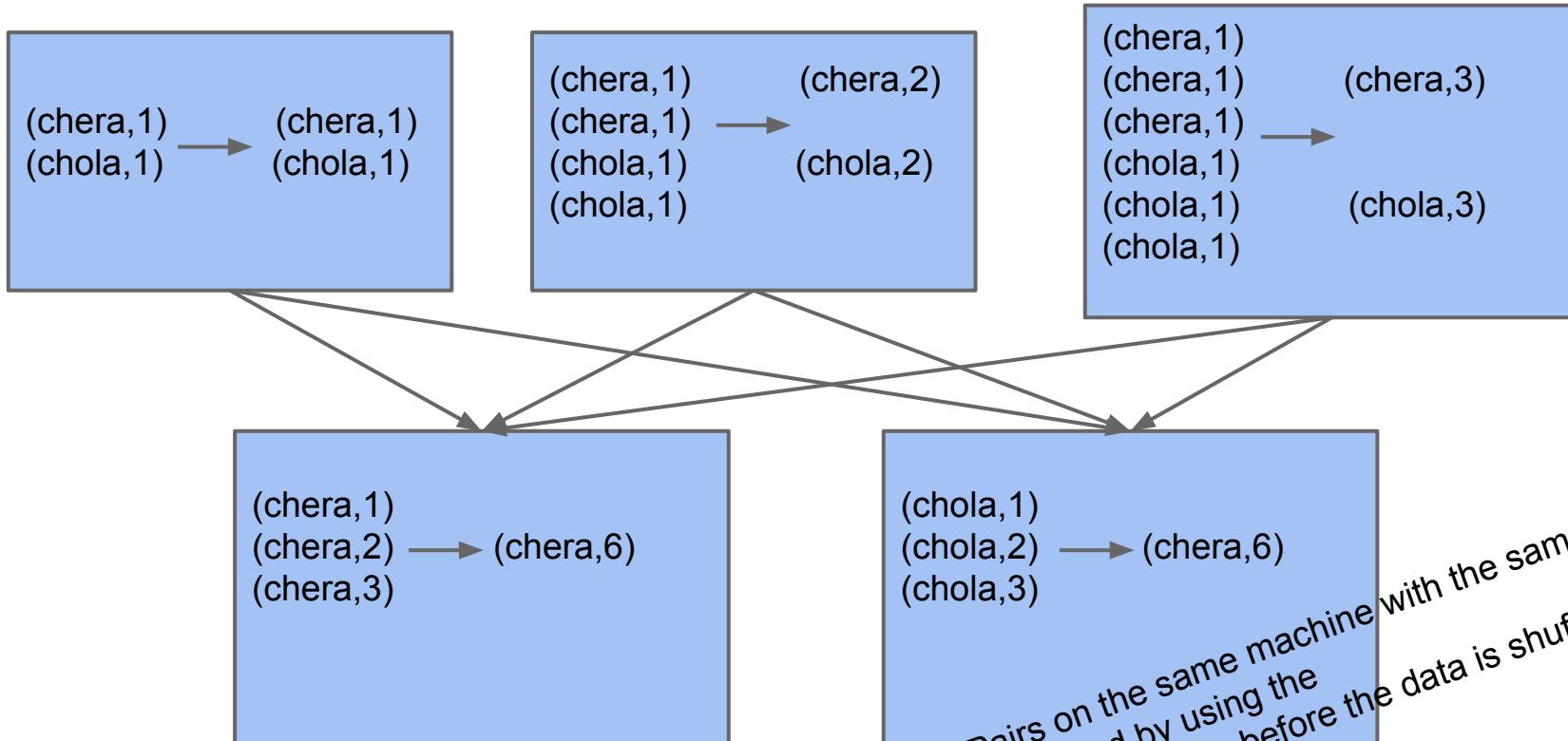
Be cautious when using `groupByKey()` instead use `reduceByKey()`.

`reduceByKey()` - Combines the output with a common key on each partition before shuffling. The pairs on the same machine with the same key are combined before shuffling happens.

`groupByKey()` - in case of `groupByKey()` all key-value pairs are shuffled around with lot of unnecessary data being transferred over the network.

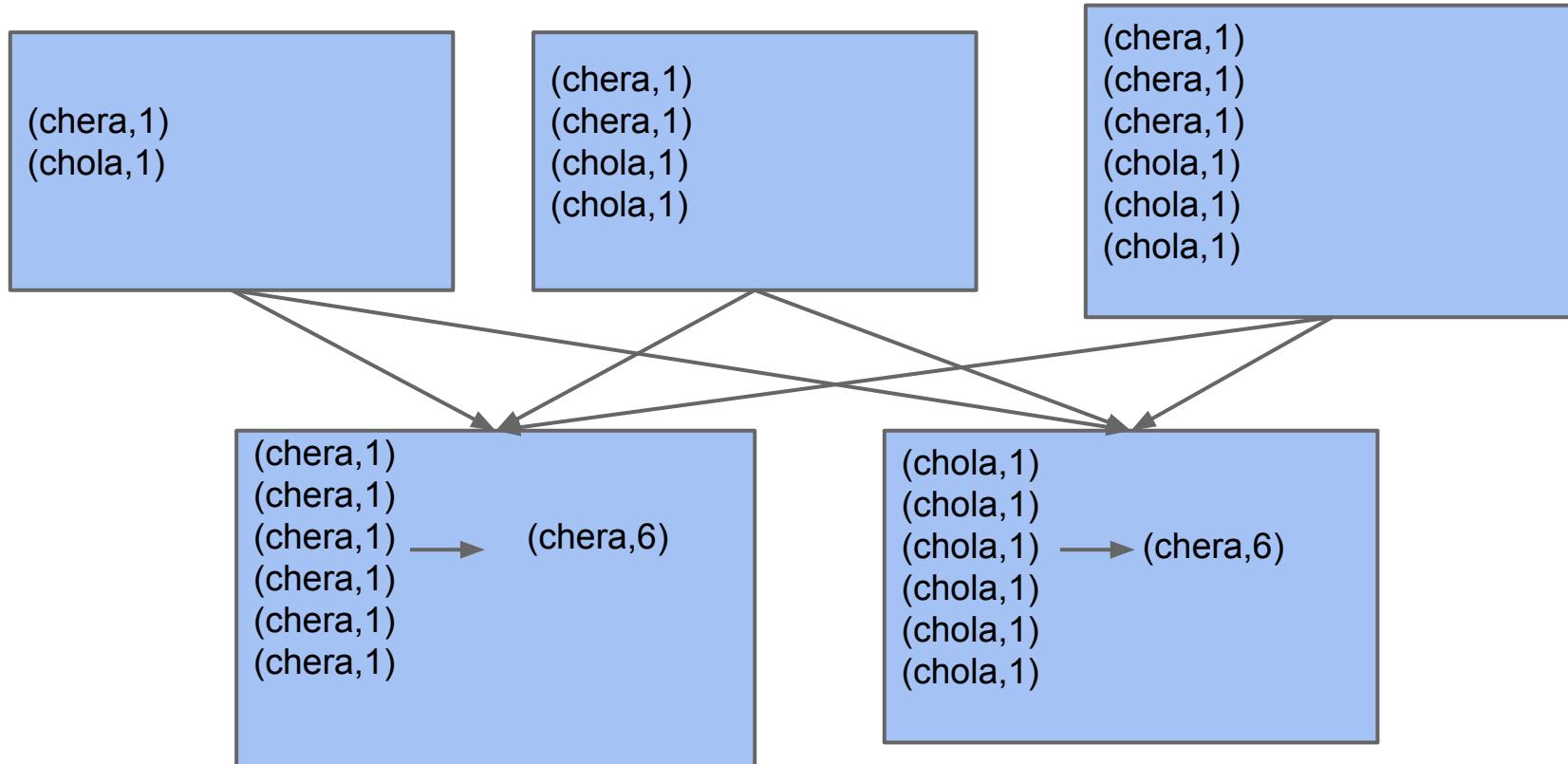
# Spark Performance Tuning

## reduceByKey()



# Spark Performance Tuning

## groupByKey()



# Spark Performance Tuning

## Stages for All Jobs

Active Stages: 1

Completed Stages: 1

### Active Stages (1)

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
1	count at SparkInternals.java:43	+details (kill)	2015/09/20 12:56:54	1.8 min	0/2		1063.0 B	

### Completed Stages (1)

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffle Write
0	mapToPair at SparkInternals.java:26	+details	2015/09/20 12:56:54	0.1 s	2/2	407.0 B		1063.0 B

### Details for Stage 1 (Attempt 0)

Total Time Across All Tasks: 0 ms

Shuffle Read: 1063.0 B / 58

- ▶ DAG Visualization
- ▶ Show Additional Metrics
- ▶ Event Timeline

### Summary Metrics for 0 Completed Tasks

No tasks have reported metrics yet.

### Aggregated Metrics by Executor

Executor ID	Address	Task Time	Total Tasks	Failed Tasks	Succeeded Tasks	Shuffle Read Size / Records
driver	localhost:58034	0 ms	0	0	0	1063.0 B / 58

### Tasks

Index	ID	Attempt	Status	Locality Level	Executor ID / Host	Launch Time	Duration	GC Time	Shuffle Read Size / Records	Errors
0	2	0	RUNNING	PROCESS_LOCAL	driver / localhost	2015/09/20 12:56:54	2.4 min	0 s	523.0 B / 26	
1	3	0	RUNNING	PROCESS_LOCAL	driver / localhost	2015/09/20 12:56:54	2.4 min	0 s	540.0 B / 32	

# Spark Performance Tuning

Total Time Across All Tasks: 92 ms

Input Size / Records: 407.0 B / 29

Shuffle Write: 1063.0 B / 58

- ▶ DAG Visualization
- ▶ Show Additional Metrics
- ▶ Event Timeline

## Summary Metrics for 2 Completed Tasks

Metric	Min	25th percentile	Median	75th percentile	Max
Duration	46 ms	46 ms	46 ms	46 ms	46 ms
GC Time	0 ms	0 ms	0 ms	0 ms	0 ms
Input Size / Records	203.0 B / 14	203.0 B / 14	204.0 B / 15	204.0 B / 15	204.0 B / 15
Shuffle Write Size / Records	417.0 B / 19	417.0 B / 19	646.0 B / 39	646.0 B / 39	646.0 B / 39

Task Duration should be even

## Aggregated Metrics by Executor

Executor ID	Address	Task Time	Total Tasks	Failed Tasks	Succeeded Tasks	Input Size / Records	Shuffle Write Size / Records
driver	localhost:58034	0.2 s	2	0	2	407.0 B / 29	1063.0 B / 58

## Tasks

Index	ID	Attempt	Status	Locality Level	Executor ID / Host	Launch Time	Duration	GC Time	Input Size / Records	Write Time	Shuffle Write Size / Records	Errors
0	0	0	SUCCESS	PROCESS_LOCAL	driver / localhost	2015/09/20 12:56:54	46 ms		203.0 B (hadoop) / 14	11 ms	646.0 B / 39	
1	1	0	SUCCESS	PROCESS_LOCAL	driver / localhost	2015/09/20 12:56:54	46 ms		204.0 B (hadoop) / 15	11 ms	417.0 B / 19	

# Spark Performance Tuning

Total Time Across All Tasks: 92 ms

Input Size / Records: 407.0 B / 29

Shuffle Write: 1063.0 B / 58

- ▶ DAG Visualization
- ▶ Show Additional Metrics
- ▶ Event Timeline

## Summary Metrics for 2 Completed Tasks

Metric	Min	25th percentile	Median	75th percentile	Max
Duration	46 ms	46 ms	46 ms	46 ms	46 ms
GC Time	0 ms	0 ms	0 ms	0 ms	0 ms
Input Size / Records	203.0 B / 14	203.0 B / 14	204.0 B / 15	204.0 B / 15	204.0 B / 15
Shuffle Write Size / Records	417.0 B / 19	417.0 B / 19	646.0 B / 39	646.0 B / 39	646.0 B / 39

Task Duration should be even

## Aggregated Metrics by Executor

Executor ID	Address	Task Time	Total Tasks	Failed Tasks	Succeeded Tasks	Input Size / Records	Shuffle Write Size / Records
driver	localhost:58034	0.2 s	2	0	2	407.0 B / 29	1063.0 B / 58

## Tasks

Index	ID	Attempt	Status	Locality Level	Executor ID / Host	Launch Time	Duration	GC Time	Input Size / Records	Write Time	Shuffle Write Size / Records	Errors
0	0	0	SUCCESS	PROCESS_LOCAL	driver / localhost	2015/09/20 12:56:54	46 ms		203.0 B (hadoop) / 14	11 ms	46.0 B / 39	
1	1	0	SUCCESS	PROCESS_LOCAL	driver / localhost	2015/09/20 12:56:54	46 ms		204.0 B (hadoop) / 15	11 ms	417.0 B / 19	

Write time should not be big

# Spark Data Locality

Data locality is nothing but the distance between the code and the data. Types of data locality are:

PROCESS\_LOCAL

NODE\_LOCAL

NODE\_PREF

RACK\_LOCAL

ANY

# Spark Data Locality

spark.locality.wait

spark.locality.process

spark.locality.wait.node

spark.locality.wait.rack

# Spark Best Practice

```
val config = new SparkConf()
config.setMaster("local")
config.setAppName("DatasetElucidation")

val sc = new SparkContext(config)

require(sc.version.replace(".", "").toInt >= 160,
        "Spark version should be 1.6+ or greater")
```

Always check for the spark version using the require, because the API use should be in compatible with the spark version installation.

# Connecting to cassandra

```
<dependency>
    <groupId>com.datastax.spark</groupId>
    <artifactId>spark-cassandra-connector_2.10</artifactId>
    <version>1.6.0-M1</version>
</dependency>
```

# Connecting to cassandra

```
SparkConf conf = new SparkConf();
    conf.setAppName("SparkCassandra");
    conf.setMaster("local[8]");
    conf.set("spark.cassandra.connection.host", "localhost");

CassandraConnector connector = CassandraConnector.apply(conf);
Session session = connector.openSession();
session.execute("CREATE KEYSPACE uidai WITH replication = {'class': 'SimpleStrategy',
                                                    'replication_factor': 3}");

session.execute("CREATE TABLE uidai.resident_detail (id INT PRIMARY KEY, name TEXT)");
session.close();
```

# Connecting to Elastic Search

```
import org.elasticsearch.spark._

val supportVectorList = List(SVMSupportVector("mesh", "12", 1),
                             SVMSupportVector("maille", "15", 2),
                             SVMSupportVector("referential", "33", 2))

// pairrdd.saveToEs("spark2/data")
val rdd = sparkContext.makeRDD(supportVectorList)

EsSpark.saveToEs(rdd, "spark99/svmIndex")
// rdd.saveToEs("spark99/svmIndex")
```

# Connecting to Elastic Search

```
val supportVectorList = List(SVMSupportVector("mesh", "12", 1),  
    SVMSupportVector("maille", "15", 2),  
    SVMSupportVector("referential", "33", 2))  
  
val rdd = sparkContext.makeRDD(supportVectorList)
```

Spark writes only a RDD to elastic search

That RDD should only be a case class or a Map. If any other object form has to be written then the ValueWriter class should be implemented.

# Connecting to Elastic Search

```
<dependency>
<groupId>org.elasticsearch</groupId>
<artifactId>elasticsearch-spark-20_2.11</artifactId>
<version>5.2.0</version>
</dependency>
```

# Connecting to ElasticSearch

localhost:9200/\_search

Apps Difference between... Using JConsole - J... Spring Interview Qu... Hibernate Interview... hadoop Hibernate Interview... 201 Core Java Inter... Real Estate Forum... Other Bookmarks

```
{"took":2,"timed_out":false,"_shards":{"total":30,"successful":30,"failed":0},"hits":{"total":11,"max_score":1.0,"hits":[{"_index":"spark3","_type":"dd","_id":"AVpCVASKKeibdPl0xbpQ","_score":1.0,"_source":{"departure":"OTP","arrival":"SFO"}}, {"_index":"spark99","_type":"svmIndex","_id":"AVpEvUoIKEibdPl0xbpn","_score":1.0,"_source":{"name":"mesh","feature":12,"id":1}}, {"_index":"spark99","_type":"svmIndex","_id":"AVpEvUoIKEibdPl0xbpo","_score":1.0,"_source":{"name":"maille","feature":15,"id":2}}, {"_index":"ss","_type": "dd", "_id": "AVpCY9W-KeibdPl0xbpg", "score": 1.0, "source": {"departure": "OTP", "arrival": "SFO"}}, {"_index": "spark3", "type": "dd", "id": "AVpCVASKKeibdPl0xbpR", "score": 1.0, "source": {"name": "mesh", "feature": 12, "id": 1}}, {"_index": "MUC", "arrival": "OTP"}, {"_index": "spark99", "type": "svmIndex", "id": "AVpEvGJ3KeibdPl0xbpk", "score": 1.0, "source": {"name": "mesh", "feature": 12, "id": 1}}, {"_index": "spark99", "type": "svmIndex", "id": "AVpEvGJ3KeibdPl0xbpm", "score": 1.0, "source": {"name": "referential", "feature": 33, "id": 2}}, {"_index": "spark3", "type": "ss", "id": "AVpCT-NwKeibdPl0xbpM", "score": 1.0, "source": {"departure": "MUC", "arrival": "OTP"}}, {"_index": "spark3", "type": "ss", "id": "AVpCT-NwKeibdPl0xbpl", "score": 1.0, "source": {"departure": "OTP", "arrival": "SFO"}}, {"_index": "spark99", "type": "svmIndex", "id": "AVpEvGJ3KeibdPl0xbpl", "score": 1.0, "source": {"name": "maille", "feature": 15, "id": 2}}]}}}
```



chinna samyad@gmail.com



# Avro Encoding

```
public byte[] encode(MyMessage myMessage) {  
  
    final ByteArrayOutputStream baos = new ByteArrayOutputStream();  
    final DatumWriter<MyMessage> writer = new ReflectDatumWriter(MyMessage.class);  
  
    try (DataFileWriter<MyMessage> out = new DataFileWriter(writer))  
    {  
  
        out.setCodec(CodecFactory.bzip2Codec()).create(ReflectData.get().getSchema(MyMessage.class), baos);  
        out.append(myMessage);  
    }  
    catch (final IOException exception) {  
  
    }  
  
    return baos.toByteArray();  
}
```

# Avro Decoding

```
public MyMessage decode(byte[] bytes) {  
  
    MyMessage myMessage = null;  
  
    final ByteArrayInputStream bais = new ByteArrayInputStream (bytes);  
  
    final DatumReader<MyMessage> reader = new  
        ReflectDatumReader<IPROMessage>(ReflectData.get().getSchema(MyMessage.class));  
  
    try (final DataFileStream<MyMessage> dfs = new DataFileStream<MyMessage> (bais, reader);)  
    {  
        myMessage = dfs.next (myMessage);  
  
    } catch (final IOException exception) {  
  
    }  
  
    return myMessage;  
}
```

# Structured Streaming

# Structured Streaming

Introduced in Spark 2.0. From Spark 2.2, it has been a stable component.

Models stream as an *infinite table* rather than discretized stream of data.

Spark 1.x streaming was based upon *RDD* based stream processing,  
but Spark 2.0 is based upon *Dataset* based streaming.

# Structured Streaming

Structured Streaming is a fast and fault tolerant streaming which guarantees end-to-end exactly once stream processing.

Structured streaming is a fault tolerant engine built on top of Spark SQL engine.

The Dataset API's are used heavily to process it.

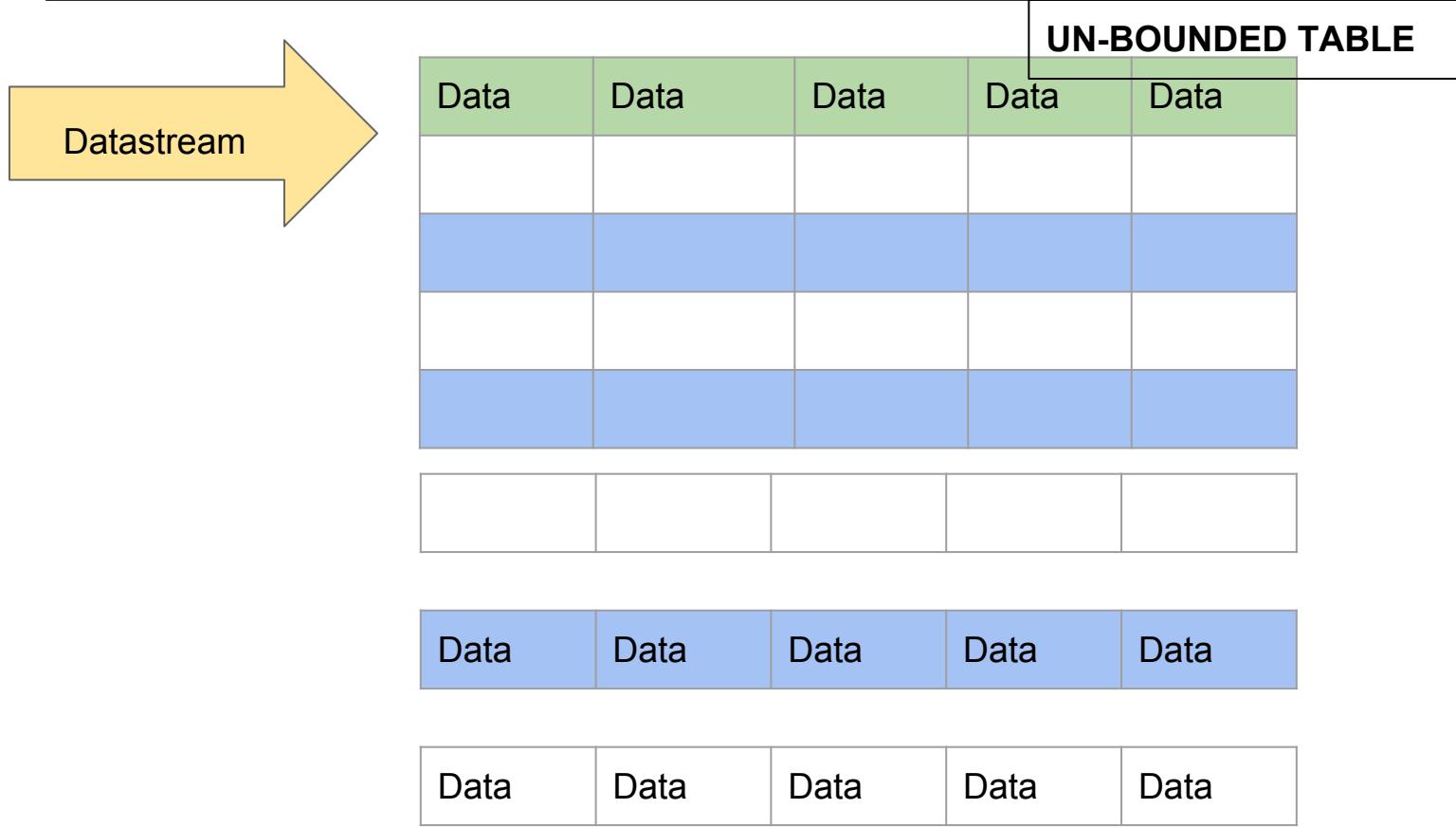
# Structured Streaming

Structured Streaming uses the approach of computing on streaming as considering it as not streams but as tables, computing through DataFrame/Dataset API.

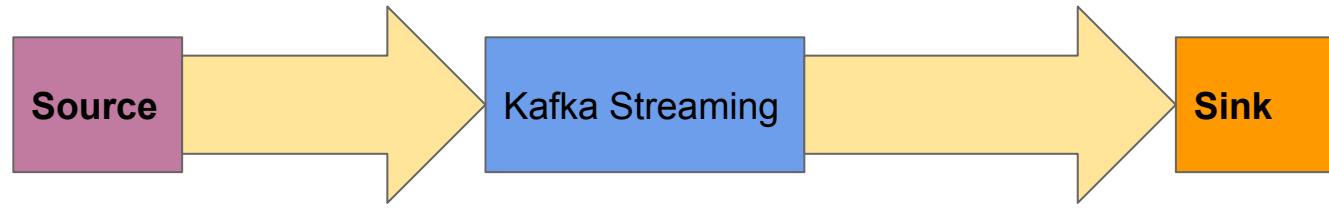
Structured Streaming utilizes Catalyst optimizer to compute on the infinite incremental execution of the data.

When viewed deeply, the streaming can be simplified as an *infinite table*.

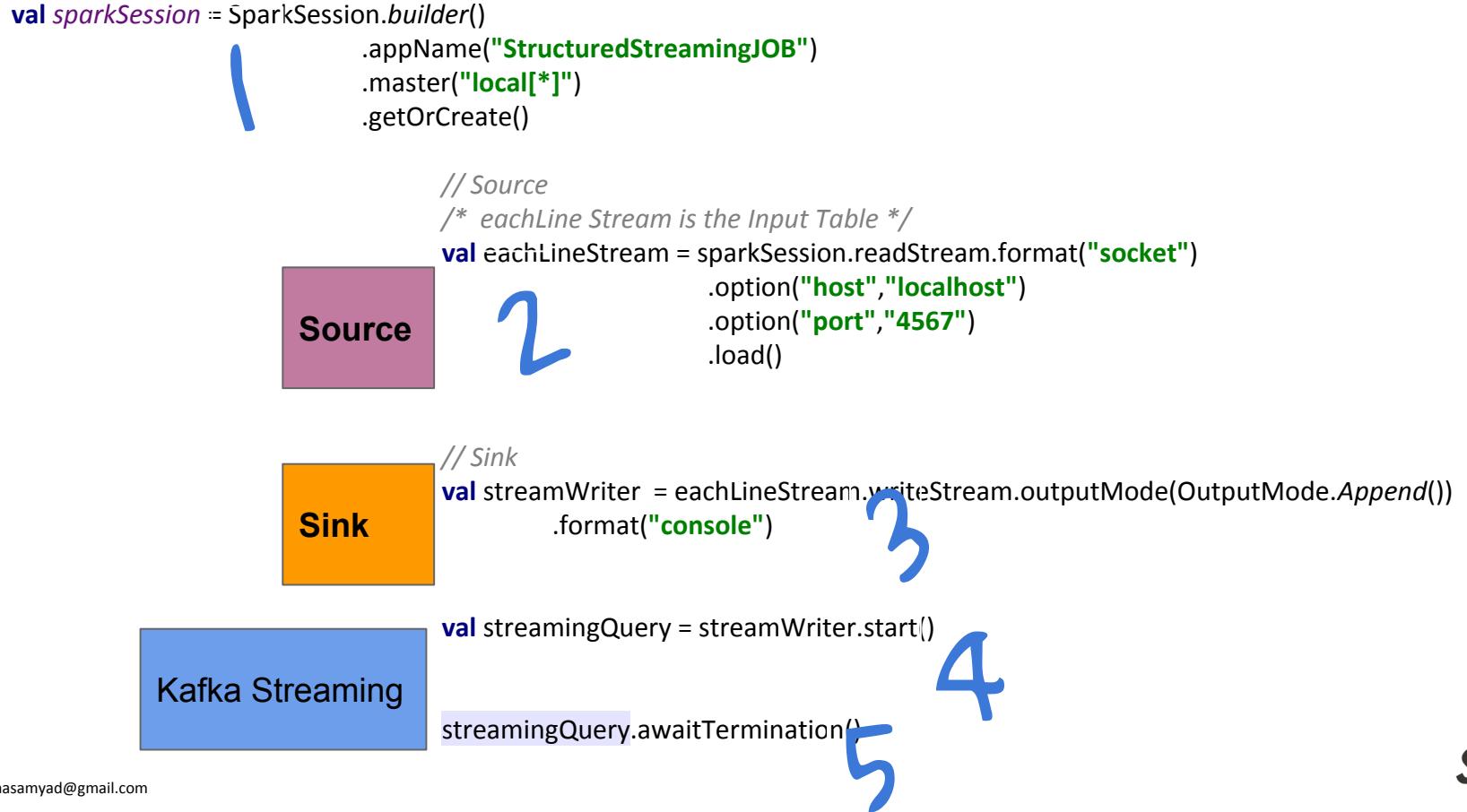
# Structured Streaming



# Structured Streaming



# Structured Streaming



# Structured Streaming

```
val sparkSession = SparkSession.builder()
    .appName("StructuredStreamingMultiplierJOB")
    .master("local[*]")
    .getOrCreate()

/* eachLine Stream is the Input Table */
val eachLineStream = sparkSession.readStream.format("socket")
    .option("host", "localhost")
    .option("port", "4567")
    .load()

import sparkSession.implicits._

// Result Table
val eachRow = eachLineStream.as[String].map(Integer.parseInt(_) * 3)

val finalReckoner = eachRow.writeStream.outputMode("append").format("console").start()

finalReckoner.awaitTermination()
```

# Structured Streaming

## Output mode

**Complete Mode** - The entire Result Table will be flushed (written) to the external storage.

**Append Mode** - The new rows appended to the Result Table since the last trigger will be written to the external storage.

**Update Mode** - The rows that were updated in Result Table since the last trigger will be written to the external storage.

# Tachyon

<http://tachyon-project.org/>



**TACHYON**

# What is tachyon ?

A distributed storage system at memory speed across cluster frameworks.

# Configuring tachyon

[1] Create the tachyon-env.sh from the template in the conf directory

```
cp /tachyon071/conf/tachyon-env.sh.template /tachyon071/conf/tachyon-env.sh
```

[2] Format the tachyon.

```
./bin/tachyon format
```

[3] Start the tachyon

```
./tachyon-start.sh local
```

[4] Stop tachyon

```
./bin/tachyon-stop.sh
```

# Data in tachyon

Like HDFS we also have tachyon command line

./tachyon tfs ls /

List

./tachyon tfs mkdir /chin

Make Directory

./tachyon tfs copyFromLocal /Users/apple/titanic3.csv /

Copy File

./tachyon tfs cat /chin/titanic3.csv

Display file

# Tachyon Configurations

By default tachyon runs in port 19998 and tachyon UI starts in port 19999.

# Tachyon Advantages

Multiple executors share the same pool of memory in tachyon

By the above advantage, cached data is not lost when individual executors crash.

Garbage collection cost is reduced.

# tachyon

When a grep on tachyon is done, it should show a TachyonMaster and TachyonWorker running as shown below.

```
Apples-MacBook-Pro:bin apple$ ps -ef | grep tachyon
 501 26958      1  0  4:00PM ??        0:01.11 /Library/Java/JavaVirtualMachines/jdk1.8.0_60.jdk/Contents/Home/bin/java -cp /Users/apple/tachyon071/conf/:/Users/apple/tachyon071/assembly/target/tachyon-assemblies-0.7.1-jar-with-dependencies.jar -Dtachyon.home=/Users/apple/tachyon071 -Dtachyon.logs.dir=/Users/apple/tachyon071/logs -Dtachyon.logger.type=W
0RKER_LOGGER -Dlog4j.configuration=file:/Users/apple/tachyon071/conf/log4j.properties -Djava.security.krb5.realm= -Djava.security.krb5.kdc= -Dlog4j.configuration=file:/Users/apple
/tachyon071/conf/log4j.properties -Dtachyon.debug=false -Dtachyon.worker.tieredstore.level.max=1 -Dtachyon.worker.tieredstore.level0.alias=MEM -Dtachyon.worker.tieredstore.level0.
dirs.path=/Volumes/ramdisk -Dtachyon.worker.tieredstore.level0.dirs.quota=1GB -Dtachyon.underfs.address=/Users/apple/tachyon071/underFSStorage -Dtachyon.underfs.hdfs.impl=org.apac
he.hadoop.hdfs.DistributedFileSystem -Dtachyon.data.folder=/Users/apple/tachyon071/underFSStorage/tmp/tachyon/data -Dtachyon.worker.max.worker.threads=2048 -Dtachyon.workers.fold
er=/Users/apple/tachyon071/underFSStorage/tmp/tachyon/workers -Dtachyon.worker.memory.size=1GB -Dtachyon.worker.data.folder=/tachyonworker/ -Dtachyon.master.max.worker.threads=2048
-Dtachyon.master.worker.timeout.ms=60000 -Dtachyon.master.hostname=localhost -Dtachyon.master.journal.folder=/Users/apple/tachyon071/journal/ -Dorg.apache.jasper.compiler.disable
jsr199=true -Djava.net.preferIPv4Stack=true tachyon.worker.TachyonWorker
 501 24609      1  0  3:23PM ttys000    0:07.68 /Library/Java/JavaVirtualMachines/jdk1.8.0_60.jdk/Contents/Home/bin/java -cp /Users/apple/tachyon071/conf/:/Users/apple/tachyon07
1/assembly/target/tachyon-assemblies-0.7.1-jar-with-dependencies.jar -Dtachyon.home=/Users/apple/tachyon071 -Dtachyon.logs.dir=/Users/apple/tachyon071/logs -Dtachyon.logger.type=M
ASTER_LOGGER -Dlog4j.configuration=file:/Users/apple/tachyon071/conf/log4j.properties -Djava.security.krb5.realm= -Djava.security.krb5.kdc= -Dlog4j.configuration=file:/Users/apple
/tachyon071/conf/log4j.properties -Dtachyon.debug=false -Dtachyon.worker.tieredstore.level.max=1 -Dtachyon.worker.tieredstore.level0.alias=MEM -Dtachyon.worker.tieredstore.level0.
dirs.path=/Volumes/ramdisk -Dtachyon.worker.tieredstore.level0.dirs.quota=1GB -Dtachyon.underfs.address=/Users/apple/tachyon071/underFSStorage -Dtachyon.underfs.hdfs.impl=org.apac
he.hadoop.hdfs.DistributedFileSystem -Dtachyon.data.folder=/Users/apple/tachyon071/underFSStorage/tmp/tachyon/data -Dtachyon.worker.max.worker.threads=2048 -Dtachyon.workers.fold
er=/Users/apple/tachyon071/underFSStorage/tmp/tachyon/workers -Dtachyon.worker.memory.size=1GB -Dtachyon.worker.data.folder=/tachyonworker/ -Dtachyon.master.max.worker.threads=2048
-Dtachyon.master.worker.timeout.ms=60000 -Dtachyon.master.hostname=localhost -Dtachyon.master.journal.folder=/Users/apple/tachyon071/journal/ -Dorg.apache.jasper.compiler.disable
jsr199=true -Djava.net.preferIPv4Stack=true tachyon.master.TachyonMaster
```

# tachyon UI

localhost:19999/home

TACHYON Overview Browse File System System Configuration Workers In-Memory Files Log Files Enable Auto-Refresh

### Tachyon Summary

Master Address:	localhost/127.0.0.1:19998
Started:	09-29-2015 15:23:54:904
Uptime:	0 day(s), 0 hour(s), 0 minute(s), and 11 second(s)
Version:	0.7.1
Running Workers:	1

### Cluster Usage Summary

Workers Capacity:	1024.00 MB
Workers Free / Used:	1024.00 MB / 0.00 B
UnderFS Capacity:	465.60 GB
UnderFS Free / Used:	431.22 GB / 34.38 GB

### Storage Usage Summary

Storage Alias	Space Capacity	Space Used	Space Usage
MEM	1024.00 MB	0.00 B	100%Free

Tachyon is an [open source](#) project developed at the UC Berkeley [AMPLab](#).

# tachyon UI

localhost:19999/configuration

TACHYON Overview Browse File System **System Configuration** Workers In-Memory Files Log Files Enable Auto-Refresh

### Tachyon Configuration

tachyon.async.enabled	false
tachyon.data.folder	/Users/apple/tachyon071/underFSStorage/tmp/tachyon/data
tachyon.debug	false
tachyon.home	/Users/apple/tachyon071
tachyon.host.resolution.timeout.ms	5000
tachyon.logger.type	MASTER_LOGGER
tachyon.logs.dir	/Users/apple/tachyon071/logs
tachyon.master.address	tachyon://localhost:19998
tachyon.master.format.file_prefix	_format_
tachyon.master.heartbeat.interval.ms	1000
tachyon.master.hostname	localhost
tachyon.master.journal.folder	/Users/apple/tachyon071/journal/
tachyon.master.max.worker.threads	2048
tachyon.master.min.worker.threads	8
tachyon.master.port	19998



## What is alluxio ?

A distributed storage system at memory speed across cluster frameworks. ie., it is a virtual memory distributed storage system.

Formerly tachyon. Baidu uses alluxio.

Alluxio is a intermediate storage guy in between hadoop frameworks. It can act as a intermediary between HBASE, MR, Spark etc.,

## Why alluxio ?

Spark context might crash, so computing has to restart from beginning. Use alluxio !!!

Share data between spark jobs and also between many computing engines.

An added flavor for memory speed storage and computation.

# Prepare alluxio first

Do bootstrap to prepare for the default configuration.

```
Dharshekths-MacBook-Pro:bin dharshekthvel$ ./alluxio bootstrapConf localhost  
/Users/dharshekthvel/ac/alluxio14/bin/..../conf/alluxio-env.sh is created.  
Dharshekths-MacBook-Pro:bin dharshekthvel$ █
```

Do format to prepare auxillo for storage.

```
Dharshekths-MacBook-Pro:bin dharshekthvel$ ./alluxio format  
Waiting for tasks to finish...  
All tasks finished, please analyze the log at /Users/dharshekthvel/ac/alluxio14/bin/..../logs/task.log.  
Formatting Alluxio Master @ Dharshekths-MacBook-Pro.local  
Dharshekths-MacBook-Pro:bin dharshekthvel$ █
```

# Start alluxio server

```
[Dharshekths-MacBook-Pro:bin dharshekthvel$ ./alluxio-start.sh local
Waiting for tasks to finish...
All tasks finished, please analyze the log at /Users/dharshekthvel/ac/alluxio14/bin/..../logs/task.log.
Waiting for tasks to finish...
All tasks finished, please analyze the log at /Users/dharshekthvel/ac/alluxio14/bin/..../logs/task.log.
Killed 0 processes on Dharshekths-MacBook-Pro.local
Killed 0 processes on Dharshekths-MacBook-Pro.local
Starting master @ Dharshekths-MacBook-Pro.local. Logging to /Users/dharshekthvel/ac/alluxio14/logs
Formatting RamFS: ramdisk 22617129 sectors (10922mb).
Started erase on disk3
Unmounting disk
Erasing
Initialized /dev/rdisk3 as a 11 GB case-insensitive HFS Plus volume
Mounting disk
Finished erase on disk3 ramdisk
Starting worker @ Dharshekths-MacBook-Pro.local. Logging to /Users/dharshekthvel/ac/alluxio14/logs
Starting proxy @ Dharshekths-MacBook-Pro.local. Logging to /Users/dharshekthvel/ac/alluxio14/logs
Dharshekths-MacBook-Pro:bin dharshekthvel$ ]
```

# Using alluxio shell to transfer data

```
dharshekthvel@dharshekthvel:~/ac/bin/alluxio-1.5.0-hadoop-2.8/bin$ ./alluxio fs copyFromLocal ~/ac/baahubali.mp4 /  
Copied file:///home/dharshekthvel/ac/baahubali.mp4 to /
```

*After copying the file. Persist the file to the file system.*

```
dharshekthvel@dharshekthvel:~/ac/bin/alluxio-1.5.0-hadoop-2.8/bin$ ./alluxio fs persist /baahubali.mp4  
persisted file /baahubali.mp4 with size 15131188
```

The file would be available in the local storage.

```
dharshekthvel@dharshekthvel:~/ac/bin/alluxio-1.5.0-hadoop-2.8/bin$ ls ..underFSStorage/  
baahubali.mp4
```

# Using alluxio shell to transfer data

```
[Dharshekths-MacBook-Pro:bin dharshekthvel$ ./alluxio fs ls /
[Dharshekths-MacBook-Pro:bin dharshekthvel$ ./alluxio fs mkdir /dhara
Successfully created directory /dhara
[Dharshekths-MacBook-Pro:bin dharshekthvel$ ./alluxio fs ls /
drwxr-xr-x    dharshekthvel  staff      0.00B  02-03-2017 10:00:34:152  Directory      /dhara
Dharshekths-MacBook-Pro:bin dharshekthvel$ █
```

```
[Dharshekths-MacBook-Pro:bin dharshekthvel$ ./alluxio fs persist /dhara
```

# Alluxio UI

Alluxio starts a UI by default at 19999

← → ⌂ ⓘ localhost:19999/home

 ALLUXIO Overview Browse Configuration Workers In-Memory Data Logs Metrics Enable Auto-Refresh

### Alluxio Summary

Master Address:	localhost/127.0.0.1:19998
Started:	02-03-2017 09:59:39:500
Uptime:	0 day(s), 0 hour(s), 3 minute(s), and 27 second(s)
Version:	1.4.0
Running Workers:	1
Startup Consistency Check:	COMPLETE

### Cluster Usage Summary

Workers Capacity:	10.67GB
Workers Free / Used:	10.67GB / 0.00B
UnderFS Capacity:	232.63GB
UnderFS Free / Used:	26.05GB / 206.57GB

### Storage Usage Summary

Storage Alias	Space Capacity	Space Used	Space Usage
MEM	10.67GB	0.00B	100%Free

[Project Website](#) | [User Mailing List](#) | [User Survey](#) | [Resources](#)

# Alluxio UI

Alluxio starts a UI by default at 19999

localhost:19999/browse?path=%2F&offset=0&limit=1

ALLUXIO Overview Browse Configuration Workers In-Memory Data Logs Metrics Enable Auto-Refresh

File Name	Size	Block Size	In-Memory	Mode	Owner	Group	Persistence State	Pin	Creation Time	Modification Time
baahubali.mp4	14.43MB	512.00MB	100%	-rw-r--r--	dharshekthvel	dharshekthvel	PERSISTED	NO	06-22-2017 11:11:14:800	06-22-2017 11:12:49:564

[View Settings](#)

# Stop alluxio server

```
Dharshekths-MacBook-Pro:bin dharshekthvel$ ./alluxio-stop.sh all
Waiting for tasks to finish...
All tasks finished, please analyze the log at /Users/dharshekthvel/ac/alluxio14/bin/..../logs/task.log.
Waiting for tasks to finish...
All tasks finished, please analyze the log at /Users/dharshekthvel/ac/alluxio14/bin/..../logs/task.log.
Killed 1 processes on Dharshekths-MacBook-Pro.local
Killed 1 processes on Dharshekths-MacBook-Pro.local
Dharshekths-MacBook-Pro:bin dharshekthvel$ █
```

# Underlying storage should be HDFS

In the conf directory of alluxio, copy the alluxio-site.properties.template to alluxio-site.properties and add the HDFS namenode URI to the alluxio-site.properties config file.

```
dharshekthvel@dharshekthvel:~/ac/bin/alluxio-1.5.0-hadoop-2.8/conf$ ls  
alluxio-env.sh      alluxio-site.properties      core-site.xml.template  masters          workers  
alluxio-env.sh.template  alluxio-site.properties.template  log4j.properties    metrics.properties.template
```

```
dharshekthvel@dharshekthvel:~/ac/bin/alluxio-1.5.0-hadoop-2.8/conf$ cp alluxio-site.properties.template alluxio-site.properties
```

```
dharshekthvel@dharshekthvel:~/ac/bin/alluxio-1.5.0-hadoop-2.8/conf$ ls  
alluxio-env.sh      alluxio-site.properties      core-site.xml.template  masters          workers  
alluxio-env.sh.template  alluxio-site.properties.template  log4j.properties    metrics.properties.template  
#  
# The Alluxio Open Foundation licenses this work under the Apache License, version 2.0  
# (the "License"). You may not use this work except in compliance with the License, which is  
# available at www.apache.org/licenses/LICENSE-2.0.  
#  
# This software is distributed on an "AS IS" basis, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND,  
# either express or implied, as more fully set forth in the License.  
#  
# See the NOTICE file distributed with this work for information regarding copyright ownership.  
#  
# Site specific configuration properties for Alluxio  
# Details about all configuration properties http://www.alluxio.org/documentation/en/Configuration-Settings.html  
# Common properties  
# alluxio.master.hostname=localhost  
# alluxio.underfs.address=${alluxio.work.dir}/underFSStorage  
# Security properties  
# alluxio.security.authorization.permission.enabled=true  
# alluxio.security.authentication.type=SIMPLE  
# Worker properties  
# alluxio.worker.memory.size=1GB  
# alluxio.worker.tieredstore.levels=1  
# alluxio.worker.tieredstore.level0.alias=MEM  
# alluxio.worker.tieredstore.level0.dirs.path=/mnt/ramdisk  
# User properties  
# alluxio.user.file.readtype.default=CACHE_PROMOTE  
# alluxio.user.file.writetype.default=MUST_CACHE  
alluxio.underfs.address=hdfs://localhost:9080  
dharshekthvel@dharshekthvel:~/ac/bin/alluxio-1.5.0-hadoop-2.8/conf$
```

# Alluxio Load Data from Underlying Storage

```
dharshekthvel@dharshekthvel:~/ac/bin/alluxio-1.5.0-hadoop-2.8/bin$ ./alluxio fs load /GateNLP.pdf  
/GateNLP.pdf loaded
```

In the screen-shot, you can see the In-memory being 100%. By which the GATENLP.pdf is loaded in memory.

The screenshot shows the Alluxio UI interface. At the top, there is a navigation bar with tabs: Overview, Browse (which is selected), Configuration, Workers, In-Memory Data, Logs, Metrics, and Enable Auto-Refresh. Below the navigation bar is a search bar with the placeholder text 'root' and a 'Go' button. The main area displays a table of files under the 'root' directory. The columns in the table are: File Name, Size, Block Size, In-Memory, Mode, Owner, Group, Persistence State, Pin, Creation Time, and Modification Time. The table contains the following data:

File Name	Size	Block Size	In-Memory	Mode	Owner	Group	Persistence State	Pin	Creation Time	Modification Time
GateNLP.pdf	5.07MB	128.00MB	100%	-rw-r--r--	dharshekthvel	supergroup	PERSISTED	NO	06-22-2017 11:26:48:521	06-22-2017 11:26:48:524
IMG-20161219-WA0030.jpg	100.29KB	128.00MB	0%	-rw-r--r--	dharshekthvel	supergroup	PERSISTED	NO	06-22-2017 11:26:48:529	06-22-2017 11:26:48:529
Test.tar.gz	63.71KB	128.00MB	0%	-rw-r--r--	dharshekthvel	supergroup	PERSISTED	NO	06-22-2017 11:26:48:531	06-22-2017 11:26:48:531
a.txt	3.00B	128.00MB	100%	-rw-r--r--	dharshekthvel	supergroup	PERSISTED	NO	06-22-2017 11:26:48:534	06-22-2017 11:26:48:534
arav				drwxr-xr-x	dharshekthvel	supergroup	PERSISTED	NO	06-22-2017 11:26:48:535	06-22-2017 11:26:48:535
baahubali.mp4	14.43MB	512.00MB	0%	-rw-r--r--	dharshekthvel	dharshekthvel	PERSISTED	NO	06-22-2017 11:11:14:800	06-22-2017 11:12:49:564
chinnasamy				drwxr-xr-x	dharshekthvel	supergroup	PERSISTED	NO	06-22-2017 11:26:48:541	06-22-2017 11:26:48:541
county.pkl	33.00B	512.00MB	100%	-rw-r--r--	dharshekthvel	dharshekthvel	PERSISTED	NO	06-22-2017 11:37:12:636	06-22-2017 11:37:29:504
covtype.info	14.27KB	128.00MB	0%	-rw-r--r--	dharshekthvel	supergroup	PERSISTED	NO	06-22-2017 11:26:48:544	06-22-2017 11:26:48:545
history_1.txt	21.59KB	128.00MB	0%	-rw-r--r--	dharshekthvel	supergroup	PERSISTED	NO	06-22-2017 11:26:48:547	06-22-2017 11:26:48:547
mapred.cmd	6.16KB	128.00MB	0%	-rw-r--r--	dharshekthvel	supergroup	PERSISTED	NO	06-22-2017 11:26:48:549	06-22-2017 11:26:48:549

Below the table, there is a 'View Settings' section with two input fields: 'Number of items per page:' (current value is 20) and 'Maximum number of pages to show in pagination component:' (current value is 10). There is also a 'Update' button.

L<sup>↔</sup>VY

# LIVY

Open source rest service for Apache Spark.

<http://livy.io>

Livy by default starts in port 8998

# LIVY

Livy needs SPARK\_HOME to be set. Ensure that the SPARK\_HOME variable is set.

Goto `livy_installation/bin` directory,  
Start the server using `./livy-server`

```
ekniven@dharshekthvel:~/home/dharshekthvel/ac/bin/livy-server030/bin$ ls
dharshekthvel@dharshekthvel:/home/dharshekthvel/ac/bin/livy-server030/bin
dharshekthvel@dharshekthvel:~/ac/bin/livy-server030/bin$ ./livy-server
dharshekthvel@dharshekthvel:~/ac/bin/livy-server030/bin$ ./livy-server
17/03/11 13:52:39 INFO StateStore$: Using BlackholeStateStore for recovery.
17/03/11 13:52:39 INFO BatchSessionManager: Recovered 0 batch sessions. Next session id: 0
17/03/11 13:52:39 INFO InteractiveSessionManager: Recovered 0 interactive sessions. Next session id: 0
17/03/11 13:52:39 INFO InteractiveSessionManager: Heartbeat watchdog thread started.
17/03/11 13:52:39 INFO WebServer: Starting server on http://dharshekthvel:8998
```

# LIVY

The screenshot shows the Advanced REST client interface. On the left, there is a sidebar with sections for ARC, HTTP request, Socket, History, Saved, and Projects. The main area is titled "Request" and shows a URL input field with "http://localhost:8998/metrics". Below it, a method selector has "GET" selected. Under "Raw headers", there is an empty text area. To the right are tabs for "Headers form", "Headers sets", and "Variables". At the bottom of the request section is a "SEND" button. The response section shows a green status bar with "200 OK" and "7.00 ms". Below this are "Raw" and "JSON" tabs, with "JSON" currently active. The JSON response body is displayed as:

```
{  
  "version": "3.0.0",  
  "gauges": {},  
  "counters": {},  
  "histograms": {},  
  "meters": {},  
  "timers": {}  
}
```

At the bottom of the client window, there are several small icons and a status bar that says "Selected environment: default".



## Livy REST CALLS

<http://localhost:8998/sessions>

```
{  
    "from": 0,  
    "total": 0,  
    "sessions": [],  
}
```

<http://localhost:8998/metrics>

```
{  
    "version": "3.0.0",  
    "gauges": {},  
    "counters": {},  
    "histograms": {},  
    "meters": {},  
    "timers": {}  
}
```

<http://localhost:8998/ping> - Returns PONG

chinna.samyad@gmail.com



## Livy REST CALLS

POST : <http://localhost:8998/sessions>

```
{"kind":"spark",
"proxyUser":"",
"jars":[],
"pyFiles":[],
"files":[],
"driverMemory":"1G",
"driverCores":8,
"executorMemory":"",
"executorCores":4,
"numExecutors":2,
"archives":[],
"queue":"",
"name":"SparkJOBSession",
"conf":{},
"heartbeatTimeoutInSecond":10}
```

# Spark ML

Machine Learning

# Types of learning

Supervised Learning

Unsupervised Learning

# Spark ML - Linear Regression

A learning algorithm to learn relation between dependent variable (y) with one or more explanatory variable (x) which are connected by the linear relation.

The hypothesis for the linear regression is as follows

$$h(x) = c_1.(x_1) + c_2.(x_2) + c_3.(x_3) + \dots$$

$x_1, x_2, x_3, \dots$  are the explanatory variables.  $h(x)$  is the hypothesis.

Linear regression goal is to learn  $c_1, c_2, c_3, \dots$

# Spark ML - Decision Trees

Decision tree is a flowchart like structure, where any node is a if-check on the attribute formed from the data.

Decision trees are used in predicting a target variable based on the input data.

# Spark ML - Decision Trees

```
import org.apache.spark.mllib.tree.DecisionTree;
import org.apache.spark.mllib.tree.model.DecisionTreeModel;
```

```
DecisionTreeModel dtm =
    DecisionTree.trainClassifier(trainingData, 2, categoricalFeaturesInfo, impurity, maxDepth, maxBins);

dtm.predict(features);
```

# Spark Future

Project Tungsten - Memory and CPU efficiency tuning

Spark and R - Already done. But more robust API's.

Off heap memory management - Tachyon

# Spark Future - Road Ahead

SuccintRDD - Querying on the compressed Data.

<https://github.com/amplab/succinct>

Velox Model Server - Predictions on data.

<https://github.com/amplab/velox-modelserver>

# Project

1. First spark batch job, let it read the dataset and publish to a queue
2. The second streaming job, read from the queue and write it to cassandra.
3. Use the agriculture dataset

# Spark Future - Road Ahead

SuccintRDD - Querying on the compressed Data

Velox Model Server

# Apache Zeppelin



---

Apache Zeppelin

## Zeppelin

Zeppelin is a visualization tool with various support for interpreters.

To start zeppelin:

[1] First configure the `conf/zeppelin-site.xml` in `conf` directory. There is a template file available. Rename it to `zeppelin-site.xml` and use the defaults.

[2] `dharshekthvel@dharshekthvel:~/ac/bin/zeppelin070/bin$ ./zeppelin.sh start`

[3] By default, zeppelin starts at port 8080.

# Zeppelin UI

The screenshot shows the Zeppelin UI homepage at [localhost:6060/#/](http://localhost:6060/#/). The page has a blue header with the Zeppelin logo, a search bar, and a user dropdown. The main content area features a large, stylized graphic of a hot air balloon on the right. On the left, there's a "Welcome to Zeppelin!" message, a brief introduction about Zeppelin, and a sidebar with "Notebook" and "Job" tabs, along with links for "Import note", "Create new note", "Filter", and "Zeppelin Tutorial". A "Help" section includes a link to the documentation, and a "Community" section encourages contributions with links to the mailing list, issues tracking, and GitHub.

Welcome to Zeppelin!

Zeppelin is web-based notebook that enables interactive data analytics.  
You can make beautiful data-driven, interactive, collaborative document with SQL, code and even more!

Notebook

- Import note
- Create new note

Filter

Zeppelin Tutorial

Help

Get started with [Zeppelin documentation](#)

Community

Please feel free to help us to improve Zeppelin,  
Any contribution are welcome!

Mailing list

Issues tracking

Github

Zeppelin

# Apache Flink



# Apache Flink

A set of robust API's for  
Iterative algorithms, Streaming for Big Data

The underlying core engine is different from spark.

But the programming model is similar.

# Apache Flink

A set of robust API's for  
Iterative algorithms, Streaming for Big Data

The underlying core engine is different from spark.

But the programming model is similar.



# Cassandra

# Cassandra

First, start the server

\$ install\_folder/bin/cassandra

\$ Then start the cqlsh

## Start the cqlsh

```
Program Files\DataStax-DDC\apache-cassandra\bin>cqlsh
Connected to Test Cluster at 127.0.0.1:9042.
lsh 5.0.1 : Cassandra 3.3.0 | CQL spec 3.4.0 | Native protocol v4
: HELP for help.
: NING: pyreadline dependency missing. Install to enable tab completion.
sh> _
```

# Cassandra

```
cqlsh help  
Documented shell commands:  
=====  
CAPTURE    CLS          COPY      DESCRIBE   EXPAND    LOGIN     SERIAL    SOURCE    UNICODE  
CLEAR      CONSISTENCY  DESC      EXIT       HELP      PAGING    SHOW      TRACING  
  
CQL help topics:  
=====  
AGGREGATES           CREATE_KEYSPACE        DROP_TRIGGER      TEXT  
ALTER_KEYSPACE        CREATE_MATERIALIZED_VIEW  DROP_TYPE        TIME  
ALTER_MATERIALIZED_VIEW CREATE_ROLE          DROP_USER        TIMESTAMP  
ALTER_TABLE           CREATE_TABLE          FUNCTIONS        TRUNCATE  
ALTER_TYPE            CREATE_TRIGGER        GRANT          TYPES  
ALTER_USER            CREATE_TYPE          INSERT          UPDATE  
APPLY                 CREATE_USER          INSERT_JSON      USE  
ASCII                DATE                  INT             UUID  
BATCH                DELETE                JSON  
BEGIN                DROP_AGGREGATE        KEYWORDS  
BLOB                 DROP_COLUMNFAMILY      LIST_PERMISSIONS  
BOOLEAN              DROP_FUNCTION         LIST_ROLES  
COUNTER              DROP_INDEX           LIST_USERS  
CREATE_AGGREGATE      DROP_KEYSPACE        PERMISSIONS  
CREATE_COLUMNFAMILY    DROP_MATERIALIZED_VIEW  REVOKE  
CREATE_FUNCTION       DROP_ROLE             SELECT  
CREATE_INDEX          DROP_TABLE            SELECT_JSON
```

cqlsh> \_

# Cassandra - Shell commands

Captures all the commands

```
cqlsh> CAPTURE 'out_capture.txt';
```

Gives in the detail about the cluster

```
cqlsh> describe cluster;
```

Lists all the keyspaces

```
cqlsh> describe keyspaces;
```

List all the tables

```
cqlsh> describe tables;
```

# Cassandra - Shell Commands

## Create table

```
cqlsh> create table <<keyspace>>. <<table>>(uid int PRIMARY KEY, otp text);  
cqlsh> create table uidai.otp(uid int PRIMARY KEY, otp text);
```

## Insert Query

```
cqlsh> insert into uidai.otp (uid, otp) VALUES (12345536, '3456');
```

## Select Query

```
cqlsh>select * from uidai.otp;
```

# Cassandra - Core components

**Commit Log, Memtable, SSTable, Bloom Filter** are the basic core components of cassandra.

**Commitlog:** It is a crash recovery mechanism which maintains every write.

**Memtable:** After commit log, data is written in memtable. Memtable is a memory oriented entity.

## Cassandra - Core components

SSTable: Data is flushed to the SSTable after the memtable has reached its fullest threshold.

Bloom filters: An algorithmic way to check whether the element is present or not.

Bloom filters are used check on the SSTable whether the data is present for a particular row or not.

# Cassandra



# Apache Zookeeper

# What is Zookeeper

A distributed system gives a single coherent view of many systems which are geographically distributed.

Zookeeper - a distributed,  
Co-ordination service for distributed computing.

A Base manager. Taking care of synchronization, grouping, naming, configs, race conditions, deadlock, maintenance etc.,

So higher order services can be built on top of it.

Typically used in a place where there are many reads and few writes. An ideal candidate for Big-Data based applications.



# What is Zookeeper

Zookeeper is a co-ordination, distributed service taking care of

- Status information

- Configuration

- Location Information

Zookeeper allows you to worry more on the application services built rather than on low level distributed framework.

Built by Yahoo for its in-house applications and later became opensource.



# Zookeeper Architecture

1. Zookeeper data are stored internally and are called as *znode*.
2. Collection of Zookeeper servers are called *ensemble*.
3. Zookeeper can handle failures till its majority of the servers are up.  
eg: On a 4 node cluster, a 1 node failure is tolerated. And on a 5 node cluster, 2 node failures are tolerated. Because on 5, 3 is a Majority.
4. Client write is sequential and also consistent.
5. Ephemeral Node: These nodes exist as long as the session exist. When the session ends, this ephemeral node is also removed.



# Zookeeper Architecture

Zookeeper is a Apache open source which takes care of managing the cluster.

Structure of storage is simple as that of a unix tree structure, called by *shared hierarchical namespace*, which contains data registers called by name **znodes**.

Znodes are typically files and directories and can be accessed through zookeeper cli.



# Zookeeper Architecture

Zookeeper data is kept in memory for high-throughput, high-performance and low latency.

Zookeeper data is replicated into the cluster of nodes called ensemble.

Zookeeper follows data ordering strictly.

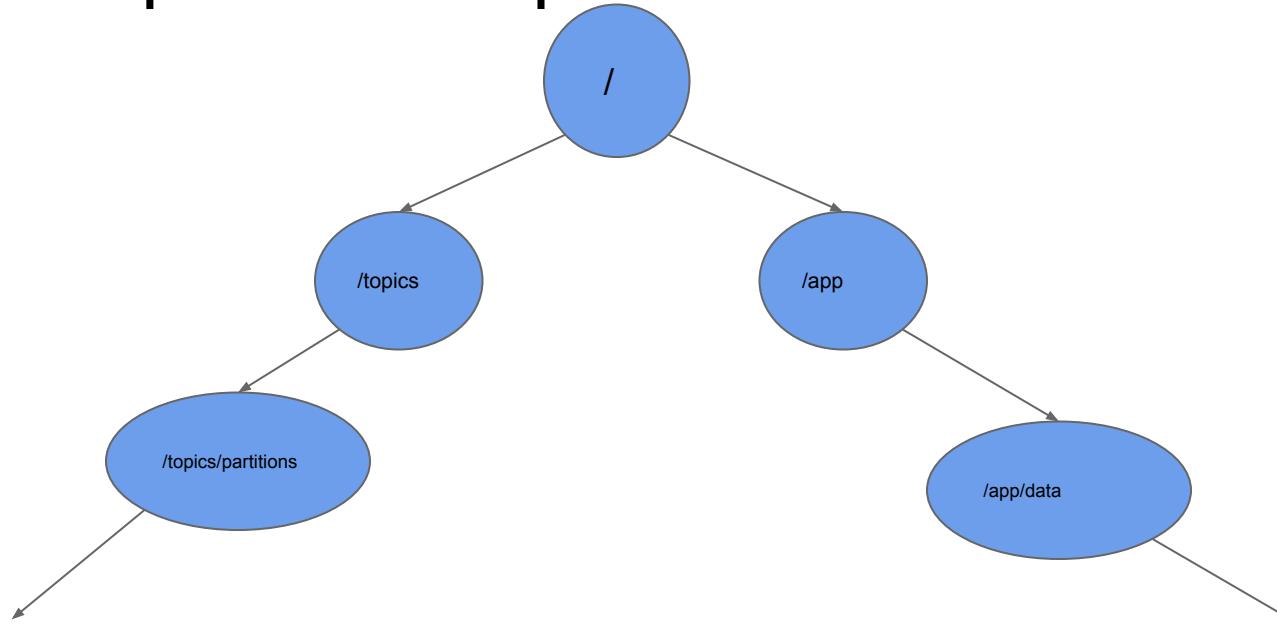
Zookeeper holds data as filesystem. And data can have children.

Zookeeper data transfer follows atomicity principle. i.e., Either all will succeed or will fail completely.

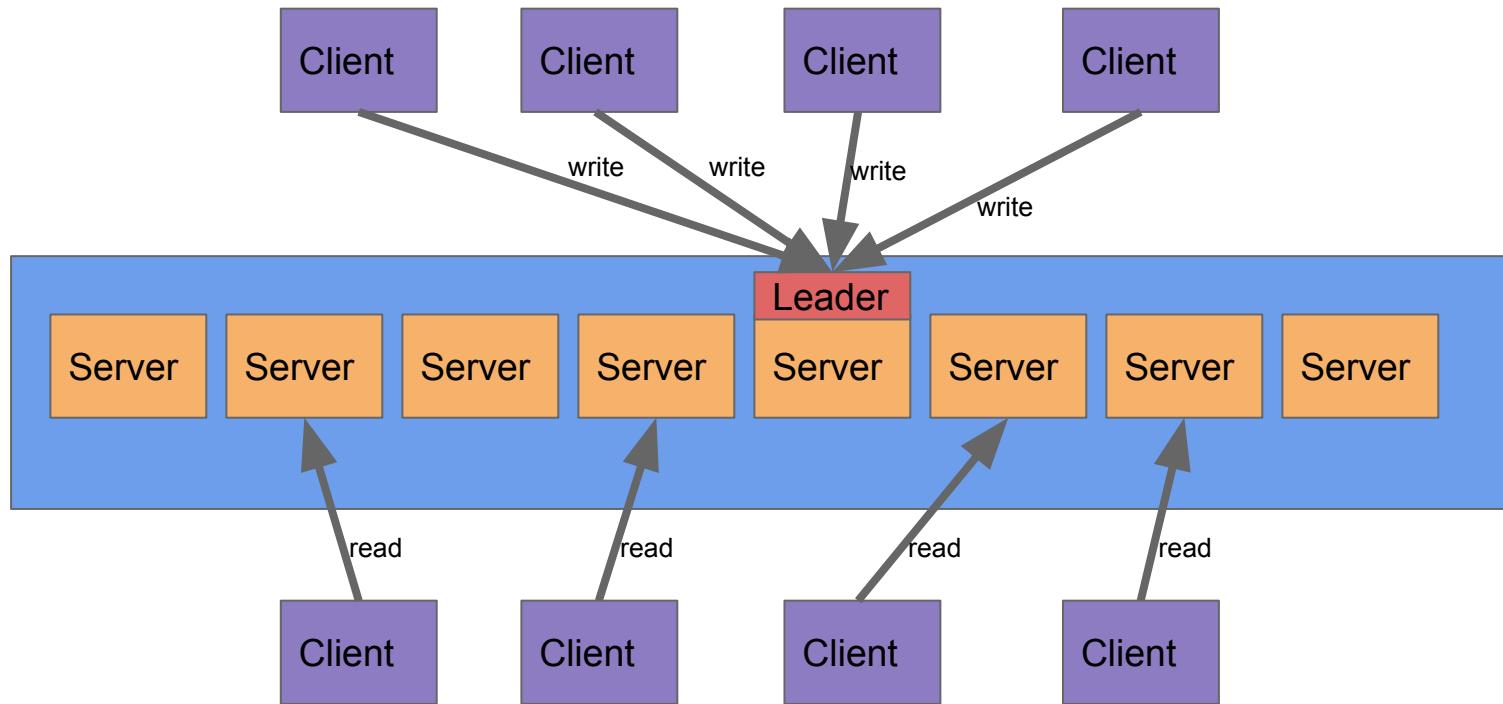
High reliable data across the nodes. This is guaranteed through znodes.



# Zookeeper Namespace



# Zookeeper Architecture



# Zookeeper Architecture - Order of execution

- [1] When the client connects to the zookeeper server, a unique session id is generated and assigned to the client.
- [2] When the time-out of the client occurs or if there is no heartbeat received, then the client is considered dead and the ephemeral nodes created by the client is deleted.

## [3] Read

A client directly requests the server in the zookeeper ensemble and the node directly retrieves the data from its own database. That's the reason, reads are faster in zookeeper.

## [4] Write

In write process, the client sends the data and the znode path to the server,  
The write request is sent to the leader,  
the leader then sends the re-request to the followers for the write.  
If only a majority of the servers ( $\frac{4}{6}$ ) write, then there is success response sent back to the client. Otherwise the write will fail.  
This majority of voting for write is called ***quorum***.



# Zookeeper Architecture - Order of execution

## Write

Write process is handled by the leader node. The leader forwards the write request to all the znodes and waits for answers from the znodes. If half of the znodes reply, then the write process is complete.

**Read**      Reads are performed internally by a specific connected znode, so there is no need to interact with the cluster.

**Leader**      The znode that is responsible for processing write requests.

**Follower**      The followers receive write requests from the clients and forward them to the leader znode.



# Zookeeper Architecture - Znode types

Persistent Znode : This znode type is persistent and is the default type.

Ephemeral Znode : This type of znode gets deleted when the session of the client expires.

Sequential Znode : Can either be persistent or ephemeral. Adds a increasing unique counter at the end of the node.

CreateMode.java

```
CreateMode {  
    PERSISTENT(0, false, false),  
    PERSISTENT_SEQUENTIAL(2, false, true),  
    EPHEMERAL(1, true, false),  
    EPHEMERAL_SEQUENTIAL(3, true, true);  
}
```



# Zookeeper CLI

Zookeeper provides a Call-Level-Interface ( a command-line client) which interacts with the zookeeper.

To connect to the zookeeper use the below command.

```
$ ./zookeeper-shell.sh localhost:2181
```



# Zookeeper CLI

Create a znode and insert data into it.

```
$ create /bus_intro_csv "Data ingested into crocodile"
```

To view all the znodes.

```
$ ls /
```

Get the data about a znode.

```
$ get /bus_intro_csv
```

To modify the data

```
$ set /bus_intro_csv "Modified Data"
```

Delete a znode.

```
$ delete /bus_intro_csv
```

To exit

```
$ quit
```



# Zookeeper CLI

Close the connection

```
$ close
```

To connect

```
$ connect localhost:2181
```



# Zookeeper CLI

## Kafka Topic

The details about the kafka topic is stored in the following place:

```
$ ls /brokers/topics/BULK_LOAD/partitions/0/state
```



# Zookeeper Java Code - Create Znode

```
ZooKeeper zookeeper = new ZooKeeper("localhost", 5000, new Watcher() {  
    @Override  
    public void process(WatchedEvent watchedEvent) {  
        }  
    });  
  
byte[] data = "My first zookeeper app".getBytes();  
  
zookeeper.create("/ARAVINDH-TOPIC", data, ZooDefs.Ids.OPEN_ACL_UNSAFE, CreateMode.PERSISTENT);
```



# Zookeeper Projects

The following are the projects that are built on top of zookeeper.

KAFKA

HBASE

SOLR

YAHOO





# Apache Kafka

A high-throughput distributed messaging system.

# Why do we need high-throughput messaging

- [#] Too many events coming on a short interval of time. Put in a queue and process it later.
- [#] Era of data, so we need a high-throughput messaging system rather than a conventional messaging.

# Apache Kafka

Kafka is a high throughput,  
partitioned,  
messaging system from LinkedIn.

An unified platform for handling all real-time data that LinkedIn has.

# Apache Kafka

Kafka is a persistent distributed log.

Distributed log meaning *spread across the cluster*.

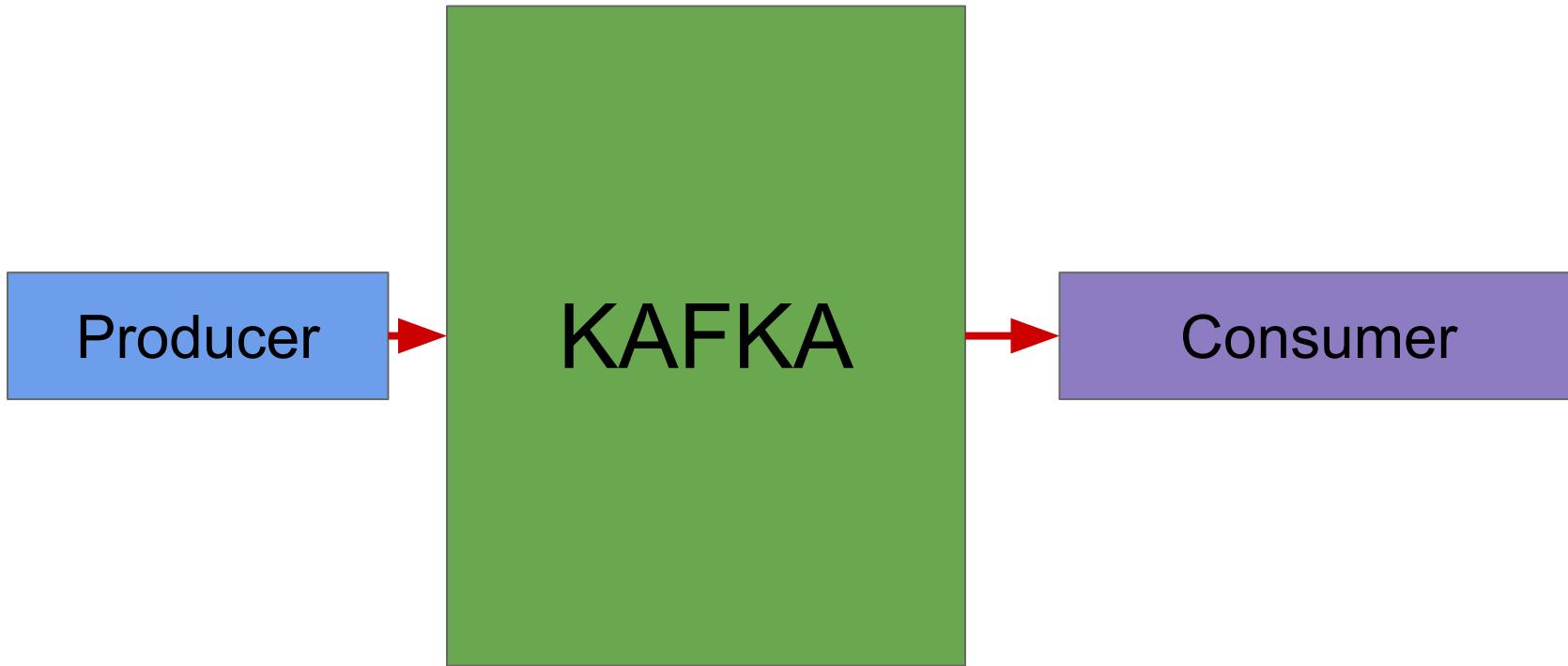
Log in the sense, *has a append only sequence of messages, bounded length,*  
*allows random reads.*

# Apache Kafka

Mostly written in Java.

Also contains scala code.

# Kafka - Producer and Consumer



# Kafka advantages over other Queueing systems

- [1] Sequential disk access, which results in optimal disk utilization.
- [2] Zero copy, which saves CPU cycles.
- [3] Compression, which saves network bandwidth.
- [4] Growing community and traction in Big Data.

# The zero copy approach

Zero copy is a mechanism by which the kernel space writes the data to the socket without any user-space, thereby reducing CPU cycles and context switching, which increases fast data transfer.

# Kafka Installation

<https://kafka.apache.org/downloads>



HOME  
INTRODUCTION  
QUICKSTART  
USE CASES  
DOCUMENTATION  
PERFORMANCE  
POWERED BY  
PROJECT INFO  
ECOSYSTEM  
CLIENTS  
EVENTS  
CONTACT US  
APACHE

Download

## Download

0.10.2.1 is the latest release. The current stable version is 0.10.2.1.

You can verify your download by following these [procedures](#) and using these [KEYS](#).

### 0.10.2.1

- [Release Notes](#)
- Source download: [kafka-0.10.2.1-src.tgz](#) (asc, md5)
- Binary downloads:
  - Scala 2.10 - [kafka\\_2.10-0.10.2.1.tgz](#) (asc, md5)
  - Scala 2.11 - [kafka\\_2.11-0.10.2.1.tgz](#) (asc, md5)
  - Scala 2.12 - [kafka\\_2.12-0.10.2.1.tgz](#) (asc, md5)

We add 2.12 to the supported Scala version. These different versions only matter if you are using Scala and you want a version built for the same Scala version you use. Otherwise any version should work (2.12 is recommended).

### 0.10.2.0

- [Release Notes](#)
- Source download: [kafka-0.10.2.0-src.tgz](#) (asc, md5)
- Binary downloads:
  - Scala 2.10 - [kafka\\_2.10-0.10.2.0.tgz](#) (asc, md5)

Goto

[https://www.apache.org/dyn/closer.cgi?path=/kafka/0.10.2.1/kafka\\_2.10-0.10.2.1.tgz](https://www.apache.org/dyn/closer.cgi?path=/kafka/0.10.2.1/kafka_2.10-0.10.2.1.tgz)

china\_emaild@gmail.com



# Kafka Installation

In order to start kafka, first start zookeeper and then start kafka

To start zookeeper, use the below command

**./zookeeper-server-start.sh ..//config/zookeeper.properties**

After starting zookeeper, then start kafka with the below command

**./kafka-server-start.sh ..//config/server.properties**

# Kafka Installation

**./zookeeper-server-start.sh ..//config/zookeeper.properties**



Zookeeper starts on port 2181 by default

```
[2017-06-04 13:56:15,030] INFO Server environment:java.library.path=/usr/java/packages/lib/amd64:/usr/lib64:/lib64:/lib:/usr/lib (org.apache.zookeeper.server.ZooKeeperServer)
[2017-06-04 13:56:15,030] INFO Server environment:java.io.tmpdir=/tmp (org.apache.zookeeper.server.ZooKeeperServer)
[2017-06-04 13:56:15,030] INFO Server environment:java.compiler=<NA> (org.apache.zookeeper.server.ZooKeeperServer)
[2017-06-04 13:56:15,030] INFO Server environment:os.name=Linux (org.apache.zookeeper.server.ZooKeeperServer)
[2017-06-04 13:56:15,030] INFO Server environment:os.arch=amd64 (org.apache.zookeeper.server.ZooKeeperServer)
[2017-06-04 13:56:15,030] INFO Server environment:os.version=4.4.0-31-generic (org.apache.zookeeper.server.ZooKeeperServer)
[2017-06-04 13:56:15,030] INFO Server environment:user.name=dharshekthvel (org.apache.zookeeper.server.ZooKeeperServer)
[2017-06-04 13:56:15,030] INFO Server environment:user.home=/home/dharshekthvel (org.apache.zookeeper.server.ZooKeeperServer)
[2017-06-04 13:56:15,030] INFO Server environment:user.dir=/home/dharshekthvel/ac/bin/kafka2110102/bin (org.apache.zookeeper.server.ZooKeeperServer)
[2017-06-04 13:56:15,039] INFO tickTime set to 3000 (org.apache.zookeeper.server.ZooKeeperServer)
[2017-06-04 13:56:15,039] INFO minSessionTimeout set to -1 (org.apache.zookeeper.server.ZooKeeperServer)
[2017-06-04 13:56:15,039] INFO maxSessionTimeout set to -1 (org.apache.zookeeper.server.ZooKeeperServer)
[2017-06-04 13:56:15,055] INFO binding to port 0.0.0.0/0.0.0.0:2181 (org.apache.zookeeper.server.NIOServerCnxnFactory)
[2017-06-04 13:56:25,275] INFO Accepted socket connection from /127.0.0.1:54872 (org.apache.zookeeper.server.NIOServerCnxnFactory)
[2017-06-04 13:56:25,281] INFO Processing ruok command from /127.0.0.1:54872 (org.apache.zookeeper.server.NIOServerCnxn)
[2017-06-04 13:56:25,283] INFO Closed socket connection for client /127.0.0.1:54872 (no session established for client) (org.apache.zookeeper.server.NIOServerCnxn)
```

# Kafka Installation

## ./kafka-server-start.sh ..//config/server.properties

```
[2017-06-04 14:00:29,160] INFO Client environment:user.name=dharshekthvel (org.apache.zookeeper.ZooKeeper)
[2017-06-04 14:00:29,160] INFO Client environment:user.home=/home/dharshekthvel (org.apache.zookeeper.ZooKeeper)
[2017-06-04 14:00:29,160] INFO Client environment:user.dir=/home/dharshekthvel/ac/bin/kafka2110102/bin (org.apache.zookeeper.ZooKeeper)
[2017-06-04 14:00:29,161] INFO Initiating client connection, connectString=localhost:2181 sessionTimeout=6000 watcher=org.I0Itec.zkclient.ZkClient@333291e3 (org.apache.zookeeper.ClientCnxn)
[2017-06-04 14:00:29,172] INFO Waiting for keeper state SyncConnected (org.I0Itec.zkclient.ZkClient)
[2017-06-04 14:00:29,173] INFO Opening socket connection to server localhost/127.0.0.1:2181. Will not attempt to authenticate using SASL (unknown error) (org.apache.zookeeper.ClientCnxn)
[2017-06-04 14:00:29,178] INFO Socket connection established to localhost/127.0.0.1:2181, initiating session (org.apache.zookeeper.ClientCnxn)
[2017-06-04 14:00:29,193] INFO Session establishment complete on server localhost/127.0.0.1:2181, sessionid = 0x15c7236389d0000, negotiated timeout = 6000 (org.apache.zookeeper.ClientCnxn)
[2017-06-04 14:00:29,195] INFO zookeeper state changed (SyncConnected) (org.I0Itec.zkclient.ZkClient)
[2017-06-04 14:00:29,318] INFO Cluster ID = z6a_aOAsSTqek6hewZz1tQ (kafka.server.KafkaServer)
[2017-06-04 14:00:29,322] WARN No meta.properties file under dir /tmp/kafka-logs/meta.properties (kafka.server.BrokerMetadataCheckpoint)
[2017-06-04 14:00:29,338] INFO [ThrottledRequestReaper-Fetch], Starting (kafka.server.ClientQuotaManager$ThrottledRequestReaper)
[2017-06-04 14:00:29,340] INFO [ThrottledRequestReaper-Producer], Starting (kafka.server.ClientQuotaManager$ThrottledRequestReaper)
[2017-06-04 14:00:29,365] INFO Log directory '/tmp/kafka-logs' not found, creating it. (kafka.log.LogManager)
[2017-06-04 14:00:29,376] INFO Loading logs. (kafka.log.LogManager)
[2017-06-04 14:00:29,389] INFO Logs loading complete in 13 ms. (kafka.log.LogManager)
[2017-06-04 14:00:29,416] INFO Starting log cleanup with a period of 300000 ms. (kafka.log.LogManager)
[2017-06-04 14:00:29,417] INFO Starting log flusher with a default period of 9223372036854775807 ms. (kafka.log.LogManager)
[2017-06-04 14:00:29,449] INFO Awaiting socket connections on 0.0.0.0:9092. (kafka.network.Acceptor)
[2017-06-04 14:00:29,451] INFO [Socket Server on Broker 0], Started 1 acceptor threads (kafka.network.SocketServer)
[2017-06-04 14:00:29,468] INFO [ExpirationReaper-0], Starting (kafka.server.DelayedOperationPurgatory$ExpiredOperationReaper)
[2017-06-04 14:00:29,469] INFO [ExpirationReaper-0], Starting (kafka.server.DelayedOperationPurgatory$ExpiredOperationReaper)
[2017-06-04 14:00:29,497] INFO Creating /controller (is it secure? false) (kafka.utils.ZKCheckedEphemeral)
[2017-06-04 14:00:29,502] INFO Result of znode creation is: OK (kafka.utils.ZKCheckedEphemeral)
[2017-06-04 14:00:29,502] INFO 0 successfully elected as leader (kafka.server.ZookeeperLeaderElector)
[2017-06-04 14:00:29,570] INFO [ExpirationReaper-0], Starting (kafka.server.DelayedOperationPurgatory$ExpiredOperationReaper)
[2017-06-04 14:00:29,574] INFO [ExpirationReaper-0], Starting (kafka.server.DelayedOperationPurgatory$ExpiredOperationReaper)
[2017-06-04 14:00:29,575] INFO [ExpirationReaper-0], Starting (kafka.server.DelayedOperationPurgatory$ExpiredOperationReaper)
[2017-06-04 14:00:29,588] INFO [GroupCoordinator 0]: Starting up. (kafka.coordinator.GroupCoordinator)
[2017-06-04 14:00:29,589] INFO [GroupCoordinator 0]: Startup complete. (kafka.coordinator.GroupCoordinator)
[2017-06-04 14:00:29,590] INFO [Group Metadata Manager on Broker 0]: Removed 0 expired offsets in 1 milliseconds. (kafka.coordinator.GroupMetadataManager)
[2017-06-04 14:00:29,608] INFO Will not load MX4J, mx4j-tools.jar is not in the classpath (kafka.utils.Mx4jLoader$)
[2017-06-04 14:00:29,624] INFO New leader is 0 (kafka.server.ZookeeperLeaderElector$LeaderChangeListener)
[2017-06-04 14:00:29,634] INFO Creating /brokers/ids/0 (is it secure? false) (kafka.utils.ZKCheckedEphemeral)
[2017-06-04 14:00:29,638] INFO Result of znode creation is: OK (kafka.utils.ZKCheckedEphemeral)
[2017-06-04 14:00:29,639] INFO Registered broker 0 at path /brokers/ids/0 with addresses:EndPoint(dharshekthvel,9092,ListenerName(PLAINTEXT),PLAINTEXT) (kafka.utils.ZkUtils)
[2017-06-04 14:00:29,640] WARN No meta.properties file under dir /tmp/kafka-logs/meta.properties (kafka.server.BrokerMetadataCheckpoint)
[2017-06-04 14:00:29,657] INFO Kafka version : 0.10.2.0 (org.apache.kafka.common.utils.AppInfoParser)
[2017-06-04 14:00:29,657] INFO Kafka commitId: 576d93a8dc0cf421 (org.apache.kafka.common.utils.AppInfoParser)
[2017-06-04 14:00:29,658] INFO [Kafka Server 0], started (kafka.server.KafkaServer)
```

Kafka starts on port 9092 by default



# Role of Zookeeper in Kafka

## [1] Election of Controller

The controller is a broker which takes the responsibility of electing the leader and follower for each of the partition.

## [2] Cluster management

Which part of the node (broker) is alive.

## [3] ACL for each topics.

## [4] What are the topics. How many partitions each topic has. The state of each partition.

# Zookeeper role in kafka

Partition states

Broker registration

Zookeeper knows what all the nodes of kafka are forming the kafka cluster.

Consumer registration and subscription



# Messaging guarantees

Messages sent by a producer to a topic are stored in the log by order.

The consumer also sees (consumes) the record, in the same order by which the records are written.

Kafka tolerates up to  $N-1$  broker failures which has  $N$  replication.

# What's next

- [\*] Producer and Consumer using console.
- [\*] Producer and Consumer through Programmatic Approach
- [\*] How to tune in parameters for maximum throughput

# Producer and Consumer - Using console

Create a topic:

```
$ ./kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic Lohith_Topic
```

Console Producer:

```
./kafka-console-producer.sh --broker-list localhost:9092 --topic Lohith_Topic
```

*The console producer with key and value:*

```
./kafka-console-producer.sh --broker-list localhost:9092 --topic Squad-Topic --property parse.key=true  
--property key.separator=:
```

Console Consumer:

```
./kafka-console-consumer.sh --zookeeper localhost:9092 --topic Lohith_Topic --from-beginning
```

(or)

```
./kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic Lohith_Topic --from-beginning
```



# Terminologies

Brokers

Producer

Consumer

Consumer Groups and Consumers

Topic

Partition

Segments

Offset

# Terminologies

- |           |   |
|-----------|---|
| Brokers   | - Individual nodes in kafka are called brokers.                           |
| Producer  | - Producers write data to brokers.  |
| Consumer  | - Consumers read data from brokers.                                       |
| Topic     | - Data is stored in topics.   |
| Partition | - Topics are split into partitions, which are distributed and replicated. |
| Offset    | - Pointer by which Producer and Consumer keep track of data.              |



# Producer, Consumer, Topics, Partition

**Producer** writes data to brokers.

**Consumer** reads data from brokers.

Data is stored in **topics**.

**Topics** are split into **partitions**, which are **replicated**.

Topics consists of partitions. A partition contains **ordered** and **immutable** sequence of messages.

# Producer

Producer publish data to *topic*.

Topic is made of *partitions*.

Producer is responsible for assigning data to a particular partition.

You can create any number of partitions to a topic.

By default kafka assigns data to partition based on round robin.

You can also write a custom partitioner for storing data to partition.

# Scala Producer

[1] Create the KafkaProducer.

[2] Create the ProducerRecord

```
val data = new ProducerRecord[String, String]("TOPIC_NAME", "key", "value")
```

[3] Using producer, send() the data

[4] producer flush()

Kafka ensures that all previously sent messages has been completed. This is done using flush().

[5] Close the producer

# Scala Producer - Asynchronous

```
val props = new Properties()
props.put("bootstrap.servers", "localhost:9092")
props.put("client.id", "SLZ_CHAMBER")
props.put("key.serializer", "org.apache.kafka.common.serialization.StringSerializer")
props.put("value.serializer", "org.apache.kafka.common.serialization.StringSerializer")
props.put("linger.ms", "1")
props.put("batch.size", "445")
props.put("compression.codec", "1")
```

```
val producer = new KafkaProducer[String, String](props)

val data = new ProducerRecord[String, String]("SLZCHAMBER", "key_1", "MAY-17-Revolution-1")

// Just a call to send will be non-blocking
producer.send(data)

producer.close()
```

# Scala Producer - Asynchronous - Blocking

```
val producer = new KafkaProducer[String, String](props)  
val data = new ProducerRecord[String, String]("SLZCHAMBER", "key_1", "MAY-17-Revolution-1")
```

```
// Calling send() with get will be a blocking.  
val metadata = producer.send(data).get()
```

```
println(metadata.offset())  
println(metadata.partition())  
println(metadata.topic())  
println(metadata.serializedKeySize())  
println(metadata.serializedValueSize())
```

```
producer.close()
```

# Scala Producer - Send data to particular partition

You can specify the partition to send in data. Here the data is sent to the partition 0.

```
val producer = new KafkaProducer[String, String](props)  
  
// Send data to a particular partition  
val data = new ProducerRecord[String, String]("BINTELLIGENCE", 0, "key_12", "Richard Feynman 101 - thank you for your physics")  
  
producer.send(data)
```

# Scala Producer - Producer Record

Ways to create ProducerRecord. With variants of Key, Value, Topic, Partitions and Timestamp.

```
val data1 = new ProducerRecord[String, String]("TOPIC", "KEY", "VALUE")
val data2 = new ProducerRecord[String, String]("TOPIC", 100, "KEY", "VALUE")
val data3 = new ProducerRecord[String, String]("TOPIC", "VALUE")
val data4 = new ProducerRecord[String, String]("TOPIC", 100, today.getTime, "KEY", "VALUE")
```

```
val today = new java.util.Date();
today.getTime;
```

# Scala Producer - With callback

```
val producer = new KafkaProducer[String, String](props)

// Send data to a particular partition
val data = new ProducerRecord[String, String]("BINTELLIGENCE", 0, "key_12", "Richard Feynman 101 - thank you for your physics")

producer.send(data, oncallback)
```

Note: Callbacks sent to the same partition are guaranteed to execute in the same order

```
val oncallback = new Callback {
  override def onCompletion(recordMetadata: RecordMetadata, e: Exception) = {
    println(recordMetadata.offset())
    println(recordMetadata.checksum())
    println(recordMetadata.topic())
    println(recordMetadata.partition())
  }
}
```

# Scala Producer

client.id                      Logical application name  
The Application name of the producer.

```
props.put("key.serializer", "org.apache.kafka.common.serialization.StringSerializer");  
props.put("value.serializer", "org.apache.kafka.common.serialization.StringSerializer");  
Serializer has to be given at the time of producing.  
Deserializer has to be given at the time of consuming.
```

# Kafka Producer

Kafka producer writes data to Commit Log

Producers always appends to the file of commit log.

# Producer - Custom Partitioner

```
class KafkaUserCustomPartitioner extends Partitioner {  
  
    override def close(): Unit = {}  
  
    override def configure(map: util.Map[String, _]): Unit = {}  
  
    override def partition(topic: String, key: scala.Any, keybytes: Array[Byte], value: scala.Any,  
                          valuebytes: Array[Byte], cluster: Cluster): Int =  
    {  
  
        val key_ = key.asInstanceOf[String]  
        if (key_.startsWith("M"))  
            1  
        else if (key_.startsWith("C"))  
            6  
        else  
            9  
    }  
}  
KafkaUserCustomPartitioner.scala
```

In your properties use the custom partitioner class.

```
props.put("partitioner.class",  
         "com.dmac.kafka.KafkaUserCustomPartitioner");
```

# Producer - Custom Partitioner

```
val props = new Properties()
props.put("bootstrap.servers", "localhost:9092")
props.put("client.id", "Mesh_Group")
props.put("key.serializer", "org.apache.kafka.common.serialization.StringSerializer")
props.put("value.serializer", "org.apache.kafka.common.serialization.StringSerializer")
props.put("linger.ms", "1")
props.put("batch.size", "1")
props.put("compression.codec", "1")
props.put("partitioner.class", "com.dmac.kafka.KafkaUserCustomPartitioner");
```

```
val producer = new KafkaProducer[String, String](props)
```

```
for (i <- 1 to 100) {
    val key = "Mesh_key".concat(i.toString)
    val value = "MESH_VALUE_".concat(i.toString)

    val data = new ProducerRecord[String, String]("MESH_TOPIC",
                                                key,
                                                value)

    producer.send(data)
}

producer.close()
```

UsingCustomKafkaPartitioner.scala

# Producer - Custom Partitioner

Run the below command, to check if the data is actually stored in that partition.

```
./kafka-run-class.sh kafka.tools.DumpLogSegments --deep-iteration --print-data-log --files  
/tmp/dharshekth-kafka/MESH_TOPIC-1/000000000000000000000000.log
```

```
dharshekthvel@dharshekthvel:~/ac/bin/kafka21101100/bin$ ./kafka-run-class.sh kafka.tools.DumpLogSegments --deep-iteration --print-data-log --files  
/tmp/kafka-logs/VEH-TOPIC-2/000000000000000000000000.log  
Dumping /tmp/kafka-logs/VEH-TOPIC-2/000000000000000000000000.log
```

```
Starting offset: 0  
offset: 0 position: 0 CreateTime: 1503634047611 isValid: true keysize: 6 valuesize: 27 magic: 2 compresscodec: NONE producerId: -1 sequence: -1 isTransactional: false headerKeys: [] key: key_12 payload: data ingested to partition 2  
  
offset: 1 position: 101 CreateTime: 1503634104503 isValid: true keysize: 6 valuesize: 27 magic: 2 compresscodec: NONE producerId: -1 sequence: -1 isTransactional: false headerKeys: [] key: key_12 payload: data ingested to partition 2
```

*payload will contain the actual message*

# Kafka Producer - Metrics

```
KafkaProducer producer = new KafkaProducer<String, String>(props);  
  
ProducerRecord data = new ProducerRecord<String, String>("BHAS-TOPIC", "seektoend", "time-line-at-84765672464");  
  
producer.send(data);  
  
Map<MetricName, Metric> metrics = producer.metrics();  
metrics.forEach(new IteratorMetricClass());
```

```
class IteratorMetricClass implements BiConsumer<MetricName, Metric> {  
    @Override  
    public void accept(MetricName metricName, Metric metric) {  
        System.out.println(metric.metricName().description() + " - " + metric.value() + " - " + metricName.name());  
    }  
}
```

# Kafka Producer - Metrics

The average number of records sent per second. - 0.0 - record-send-rate

The average time in ms record batches spent in the record accumulator. - 0.0 - record-queue-time-avg

The average length of time the I/O thread spent waiting for a socket ready for reads or writes in nanoseconds. - 2038974.625 - io-wait-time-ns-avg

The rate of record batch split - 0.0 - batch-split-rate

The average number of outgoing bytes sent per second to all servers. - 1.630561378989052 - outgoing-byte-rate

The average number of network operations (reads or writes) on all connections per second. - 0.13310705134604506 - network-io-rate

New connections established per second in the window. - 0.03324578609661225 - connection-creation-rate

Connections closed per second in the window. - 0.0 - connection-close-rate

The maximum size of any request sent in the window. - 31.0 - request-size-max

- 10.915141430948418 - incoming-byte-rate

Responses received sent per second. - 0.06655574043261231 - response-rate

The maximum size of any request sent in the window. - -Infinity - request-size-max

The average size of all requests in the window.. - 24.5 - request-size-avg

The average length of time for I/O per select call in nanoseconds. - 444352.71428571426 - io-time-ns-avg

- 0.0 - outgoing-byte-rate

total number of registered metrics - 55.0 - count

- 0.0 - request-latency-avg

The average size of all requests in the window.. - 0.0 - request-size-avg

The current number of active connections. - 2.0 - connection-count

The average number of records per request. - 0.0 - records-per-request-avg

The number of user threads blocked waiting for buffer memory to enqueue their records - 0.0 - waiting-threads

The average per-second number of record sends that are dropped due to buffer exhaustion - 0.0 - buffer-exhausted-rate

The average number of responses received per second. - 0.06655574043261231 - response-rate

The maximum size of any request sent in the window. - 31.0 - request-size-max

The average per-second number of retried record sends - 0.0 - record-retry-rate

The maximum record size - -Infinity - record-size-max

The average number of requests sent per second. - 0.0333333333333333 - request-rate

Number of times the I/O layer checked for new I/O to perform per second - 0.29876510423582525 - select-rate

The total amount of buffer memory that is not being used (either unallocated or in the free list). - 3.3553987E7 - buffer-available-bytes

The maximum time in ms record batches spent in the record accumulator. - -Infinity - record-queue-time-max

.....

.....



# Producer - Topics - Partition

RECAP

**Producer** writes data to brokers.

Data is stored in **topics**.

**Topics** are split into **partitions**, which are **replicated**.

# Topic

Create a topic:

```
$ ./kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 2 --partitions 10 --topic Lohith_Topic
```

Replication factor can either be equal or less than the number of brokers.

My broker was only one, but if I give in the replication factor as 2, I get the below message.

Error while executing topic command : replication factor: 2 larger than available brokers: 1

```
[2017-06-20 11:38:39,607] ERROR org.apache.kafka.common.errors.InvalidReplicationFactorException: replication factor: 2 larger than available brokers: 1  
(kafka.admin.TopicCommand$)
```



# Topic operations

## Listing the topic

```
$ ./kafka-topics.sh --list --zookeeper localhost:2181
```

## Deleting a topic

```
$ ./kafka-topics.sh --zookeeper localhost:2181 --delete --topic CHOLA
```

But this will have impact only when **delete.topic.enable** is set to true



# Topic - auto.create = false

In the server.properties if you make auto.create.topics.enable to false. Then you need to create a topic manually.

auto.create.topics.enable=false

By default the auto.create.topics.enable is true.

# Topic <-> Partition



# Partition

Topic partition is the key-unit of parallelism in kafka.

Topic partition not only supports on scaling but also in storage and operations.

# Partitions

```
./kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 100 --topic CHEGUERA
```

100 Partitions are created for the topic CHEGUERA.

100 directory logs are created for CHEGUERA topic.

It's created in /tmp/dharshekth-kafka directory.

Default is /tmp/kafka-logs.

To change the location, goto config/server.properties

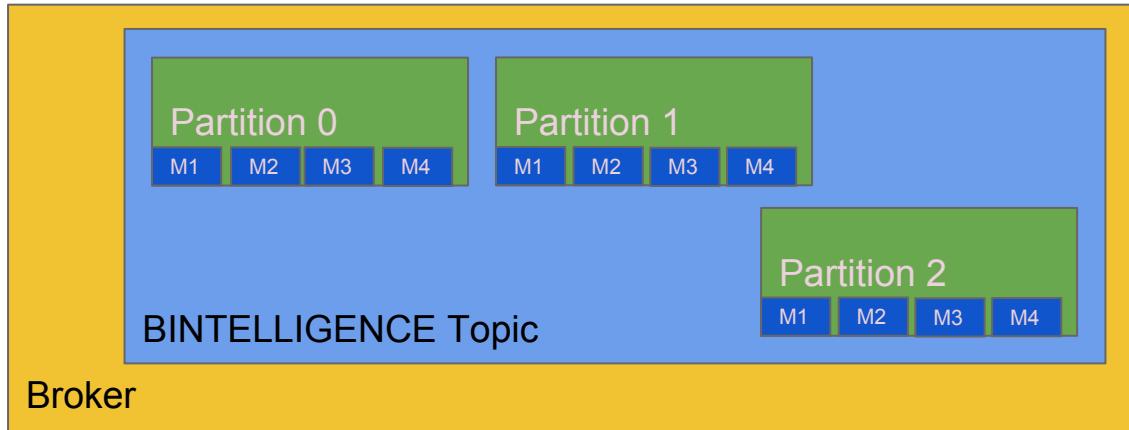
and change the **log.dirs** value.

**log.dirs=/tmp/dharshekth-kafka**

```
CHEGUERA-33/ CHEGUERA-70/ CHOLA-17/ CHOLA-54/ CHOLA-91/  
CHEGUERA-34/ CHEGUERA-71/ CHOLA-18/ CHOLA-55/ CHOLA-92/  
CHEGUERA-35/ CHEGUERA-72/ CHOLA-19/ CHOLA-56/ CHOLA-93/  
CHEGUERA-36/ CHEGUERA-73/ CHOLA-20/ CHOLA-57/ CHOLA-94/  
CHEGUERA-37/ CHEGUERA-74/ CHOLA-20/ CHOLA-58/ CHOLA-95/  
CHEGUERA-38/ CHEGUERA-75/ CHOLA-21/ CHOLA-59/ CHOLA-96/  
CHEGUERA-39/ CHEGUERA-76/ CHOLA-22/ CHOLA-60/ CHOLA-97/  
CHEGUERA-4/ CHEGUERA-77/ CHOLA-23/ CHOLA-61/ CHOLA-98/  
CHEGUERA-40/ CHEGUERA-78/ CHOLA-24/ CHOLA-62/ CHOLA-99/  
CHEGUERA-41/ CHEGUERA-79/ CHOLA-25/ CHOLA-63/ cleaner-offset-checkpoint  
CHEGUERA-42/ CHEGUERA-80/ CHOLA-26/ CHOLA-64/ lock  
CHEGUERA-43/ CHEGUERA-81/ CHOLA-27/ CHOLA-65/ meta.properties  
CHEGUERA-44/ CHEGUERA-82/ CHOLA-28/ CHOLA-66/ recovery-point-offset-checkpoint  
CHEGUERA-45/ dhrshekthvel@dharshekthvel:~/ac/bin/kafka2110102/bin$ clear replication-offset-checkpoint  
dhrshekthvel@dharshekthvel:~/ac/bin/kafka2110102/bin$ ls /tmp/dharshekth-kafka/  
CHEGUERA-0 CHEGUERA-22 CHEGUERA-36 CHEGUERA-5 CHEGUERA-63 CHEGUERA-77 CHEGUERA-90 CHOLA-13 CHOLA-27 CHOLA-40 CHOLA-54 CHOLA-68 CHOLA-81 CHOLA-95  
CHEGUERA-1 CHEGUERA-23 CHEGUERA-37 CHEGUERA-50 CHEGUERA-64 CHEGUERA-78 CHEGUERA-91 CHOLA-14 CHOLA-28 CHOLA-41 CHOLA-55 CHOLA-69 CHOLA-82 CHOLA-96  
CHEGUERA-2 CHEGUERA-24 CHEGUERA-38 CHEGUERA-51 CHEGUERA-65 CHEGUERA-79 CHEGUERA-92 CHOLA-15 CHOLA-29 CHOLA-42 CHOLA-56 CHOLA-70 CHOLA-83 CHOLA-97  
CHEGUERA-3 CHEGUERA-25 CHEGUERA-39 CHEGUERA-52 CHEGUERA-66 CHEGUERA-80 CHEGUERA-93 CHOLA-16 CHOLA-3 CHOLA-43 CHOLA-57 CHOLA-78 CHOLA-84 CHOLA-98  
CHEGUERA-4 CHEGUERA-26 CHEGUERA-40 CHEGUERA-53 CHEGUERA-67 CHEGUERA-80 CHEGUERA-94 CHOLA-17 CHOLA-30 CHOLA-44 CHOLA-58 CHOLA-71 CHOLA-85 CHOLA-99  
CHEGUERA-5 CHEGUERA-27 CHEGUERA-41 CHEGUERA-54 CHEGUERA-68 CHEGUERA-81 CHEGUERA-95 CHOLA-18 CHOLA-31 CHOLA-45 CHOLA-59 CHOLA-72 CHOLA-86 cleaner-offset-checkpoint  
CHEGUERA-6 CHEGUERA-28 CHEGUERA-42 CHEGUERA-55 CHEGUERA-69 CHEGUERA-82 CHEGUERA-96 CHOLA-19 CHOLA-32 CHOLA-46 CHOLA-60 CHOLA-73 CHOLA-87 meta.properties  
CHEGUERA-7 CHEGUERA-29 CHEGUERA-43 CHEGUERA-56 CHEGUERA-70 CHEGUERA-83 CHEGUERA-97 CHOLA-20 CHOLA-33 CHOLA-47 CHOLA-61 CHOLA-74 CHOLA-88 recovery-point-offset-checkpoint  
CHEGUERA-8 CHEGUERA-30 CHEGUERA-44 CHEGUERA-57 CHEGUERA-71 CHEGUERA-84 CHEGUERA-98 CHOLA-21 CHOLA-34 CHOLA-48 CHOLA-61 CHOLA-75 CHOLA-89 replication-offset-checkpoint  
CHEGUERA-9 CHEGUERA-31 CHEGUERA-45 CHEGUERA-59 CHEGUERA-72 CHEGUERA-85 CHEGUERA-99 CHOLA-22 CHOLA-36 CHOLA-5 CHOLA-63 CHOLA-77 CHOLA-90  
CHEGUERA-10 CHEGUERA-32 CHEGUERA-46 CHEGUERA-60 CHEGUERA-73 CHEGUERA-86 CHOLA-23 CHOLA-37 CHOLA-59 CHOLA-64 CHOLA-78 CHOLA-91  
CHEGUERA-11 CHEGUERA-33 CHEGUERA-47 CHEGUERA-61 CHEGUERA-74 CHEGUERA-87 CHEGUERA-90 CHOLA-24 CHOLA-38 CHOLA-60 CHOLA-79 CHOLA-82 CHOLA-93  
CHEGUERA-12 CHEGUERA-34 CHEGUERA-48 CHEGUERA-62 CHEGUERA-75 CHEGUERA-89 CHOLA-11 CHOLA-25 CHOLA-39 CHOLA-52 CHOLA-66 CHOLA-8 CHOLA-94  
CHEGUERA-13 CHEGUERA-35 CHEGUERA-49 CHEGUERA-63 CHEGUERA-76 CHEGUERA-90 CHOLA-12 CHOLA-26 CHOLA-4 CHOLA-53 CHOLA-67 CHOLA-80 CHOLA-95  
dhrshekthvel@dharshekthvel:~/ac/bin/kafka2110102/bin$
```



# Kafka Architecture



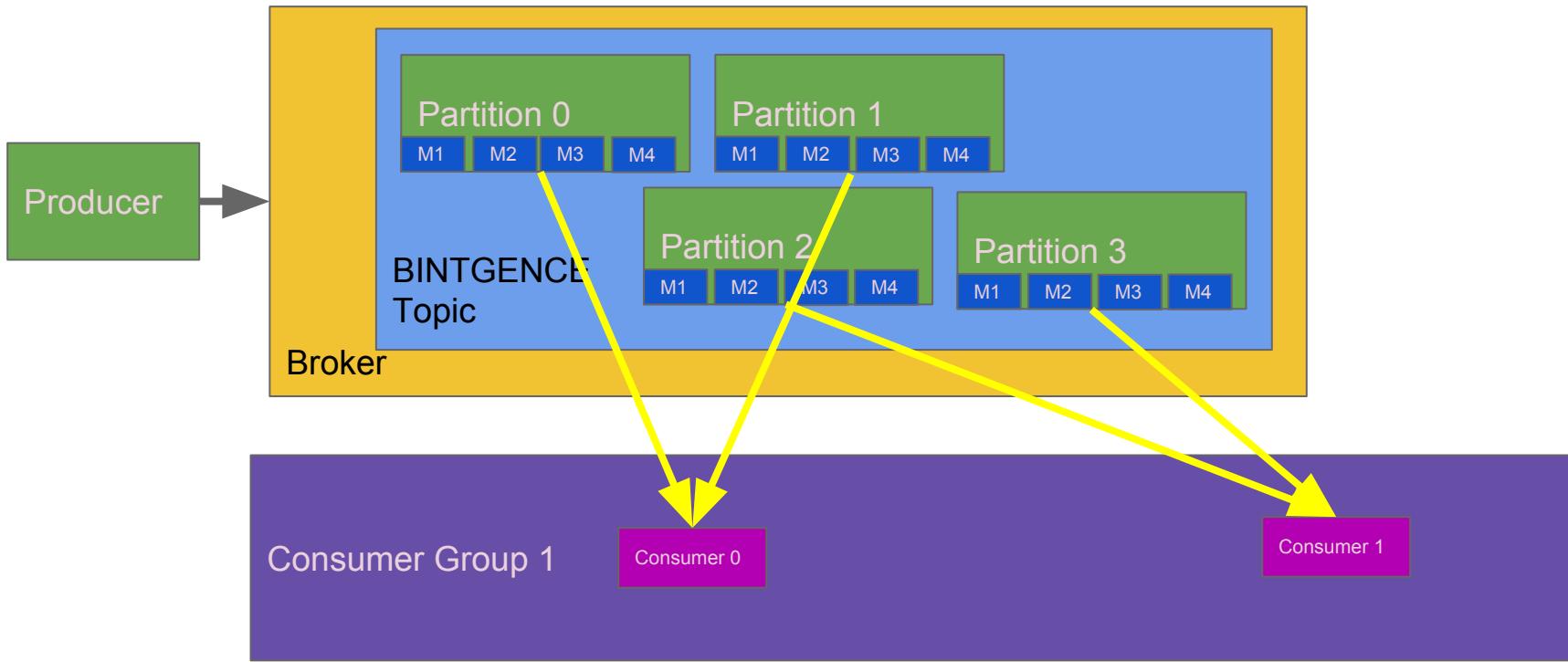
A Topic is stored internally as a partition.

You can specify the number of partitions and replications at the time of creating the topic.

Partitions are stored in the local storage of each of the brokers.

# Kafka Architecture

Data in Partition 0 and 1 will be consumed by Consumer 0 and data in Partition 2 and 3 will be consumed by Consumer 1.



# Kafka Architecture



# Partition - Leader and Followers

Data is written to the leader partition and all the followers passively copy from the leader.

Leader is the one who handles the reads/writes for a particular partition.

# Partition

Each partition has a leader.

The producer writes to the leader and the followers just copy data from the leaders log.

A message is said to be committed when the leader writes the data to itself and also replicates the data to its followers (the in-sync replicas (ISR)).

*Configs:*

*replica.lag.max.messages*  
*replica.lag.time.max.ms*

# Partition - Leader and Followers

Each broker holds many partitions.

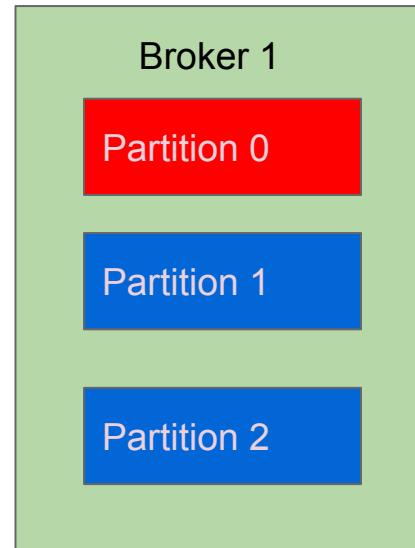
Each partition can be a **leader** or a **replica**.

All writes and reads to a topic go through leader and the leader coordinates updating replicas with new data.

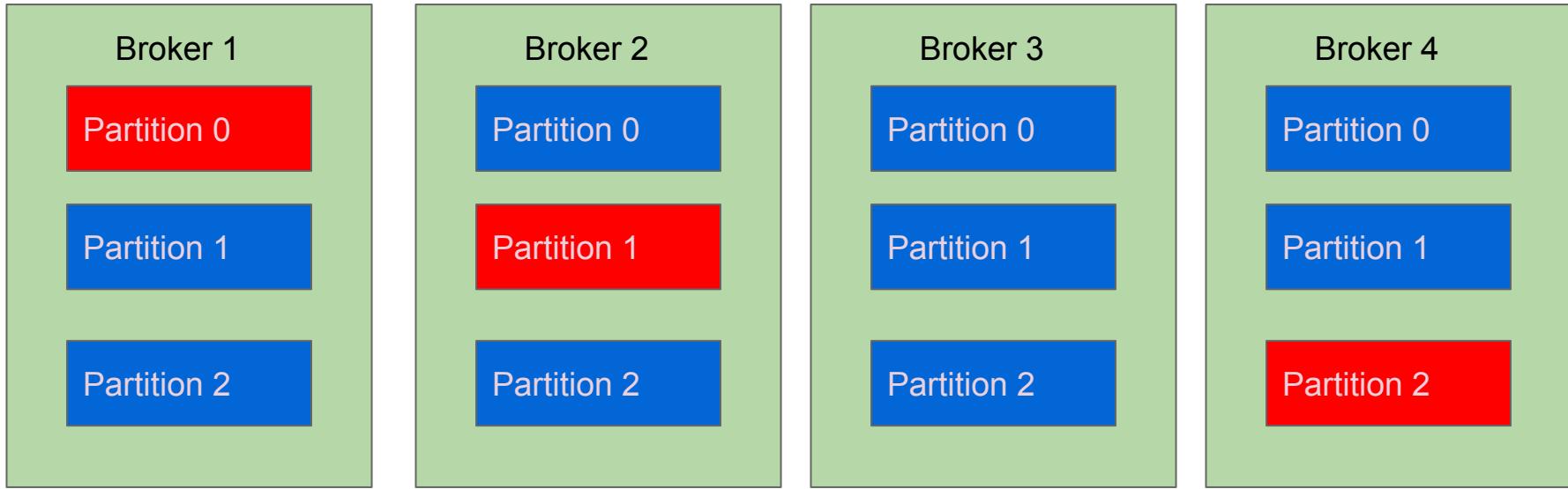
If a leader fails, a replica takes as a new leader.

Partition is a Write-a-head-log.

Write-a-head-log guarantees Atomicity and Durability of ACID property.



# Partition



Leader

Follower

# Partition

*replica.lag.max.messages=4*

The max messages that a leader waits on to take a follower out of ISR  
Used for slow replicas.

*Replica.lag.time.max.ms=100*

The max time that a leader waits on to take a follower out of ISR  
Used for dead replicas.

# Messaging Guarantees

When a producer sends a message to a particular topic, the partition appends the message in the order of how it arrives.

The consumer consumes the messages in the order they are stored in the partition.



# Partitions

*Description about partitions*

**./kafka-topics.sh --describe --zookeeper localhost:2181 --topic CHE**

Topic:CHE	PartitionCount:20	ReplicationFactor:2	Configs:
Topic: CHE	Partition: 0	Leader: 1 Replicas: 1,0	Isr: 1,0
Topic: CHE	Partition: 1	Leader: 0 Replicas: 0,1	Isr: 0,1
Topic: CHE	Partition: 2	Leader: 1 Replicas: 1,0	Isr: 1,0
Topic: CHE	Partition: 3	Leader: 0 Replicas: 0,1	Isr: 0,1
Topic: CHE	Partition: 4	Leader: 1 Replicas: 1,0	Isr: 1,0
Topic: CHE	Partition: 5	Leader: 0 Replicas: 0,1	Isr: 0,1
Topic: CHE	Partition: 6	Leader: 1 Replicas: 1,0	Isr: 1,0
Topic: CHE	Partition: 7	Leader: 0 Replicas: 0,1	Isr: 0,1
Topic: CHE	Partition: 8	Leader: 1 Replicas: 1,0	Isr: 1,0
Topic: CHE	Partition: 9	Leader: 0 Replicas: 0,1	Isr: 0,1
Topic: CHE	Partition: 10	Leader: 1 Replicas: 1,0	Isr: 1,0
Topic: CHE	Partition: 11	Leader: 0 Replicas: 0,1	Isr: 0,1
Topic: CHE	Partition: 12	Leader: 1 Replicas: 1,0	Isr: 1,0
Topic: CHE	Partition: 13	Leader: 0 Replicas: 0,1	Isr: 0,1
Topic: CHE	Partition: 14	Leader: 1 Replicas: 1,0	Isr: 1,0
Topic: CHE	Partition: 15	Leader: 0 Replicas: 0,1	Isr: 0,1
Topic: CHE	Partition: 16	Leader: 1 Replicas: 1,0	Isr: 1,0
Topic: CHE	Partition: 17	Leader: 0 Replicas: 0,1	Isr: 0,1
Topic: CHE	Partition: 18	Leader: 1 Replicas: 1,0	Isr: 1,0
Topic: CHE	Partition: 19	Leader: 0 Replicas: 0,1	Isr: 0,1



# Analyze the data in Partition

Run the below command to check on the data written inside partition

```
./kafka-run-class.sh kafka.tools.DumpLogSegments --deep-iteration --print-data-log --files  
/tmp/dharshekth-kafka/SLZCHAMBER-1/000000000000000000000000.log
```

# Kafka Controller

One of the broker servers as the kafka controller.

Kafka controller takes the responsibility of assigning partitions and managing replicas.

# Serializer and Deserializer

For a **producer**, we give in a Serializer in the properties.

```
props.put("key.serializer", "org.apache.kafka.common.serialization.StringSerializer")
props.put("value.serializer", "org.apache.kafka.common.serialization.StringSerializer")
```

For a **consumer**, we give in a Deserializer in the properties.

```
props.put("key.deserializer", "org.apache.kafka.common.serialization.StringDeserializer");
props.put("value.deserializer", "org.apache.kafka.common.serialization.StringDeserializer");
```

# Partition <-> Segments

# Partition - Segments

Partition is immutable,  
sequence of messages of data,  
stored sequentially.

Partition is stored as segments. (or) Partition is split as segments.

When kafka writes to partition, it writes to segment (active segment)

In disk, partition is a directory and each segment is a log file and a index file.

Log file - Where message is stored.

Index file - Contains the index to find data

# Partition, Segments

Data is stored in topics as Partitions

The data order is preserved inside the partitions.

Partitions store data in disk as segments.

```
dharshekthvel@dharshekthvel:~/ac/bin/kafka2110102/bin$ tree /tmp/dharshekth-kafka/TIGER-0
/tmp/dharshekth-kafka/TIGER-0
|-- 00000000000000000000.index
|-- 00000000000000000000.log
|-- 00000000000000000000.timeindex
|-- 00000000000000000003.index
|-- 00000000000000000003.log
|-- 00000000000000000003.timeindex
|-- 00000000000000000006.index
|-- 00000000000000000006.log
-- 00000000000000000006.timeindex

0 directories, 9 files
dharshekthvel@dharshekthvel:~/ac/bin/kafka2110102/bin$
```

In the server.properties. Change the *log.segment.bytes*.

# The maximum size of a log segment file. When this size is reached a new log segment will be created.

```
#log.segment.bytes=1073741824
log.segment.bytes=200
```

# Partition - Segments

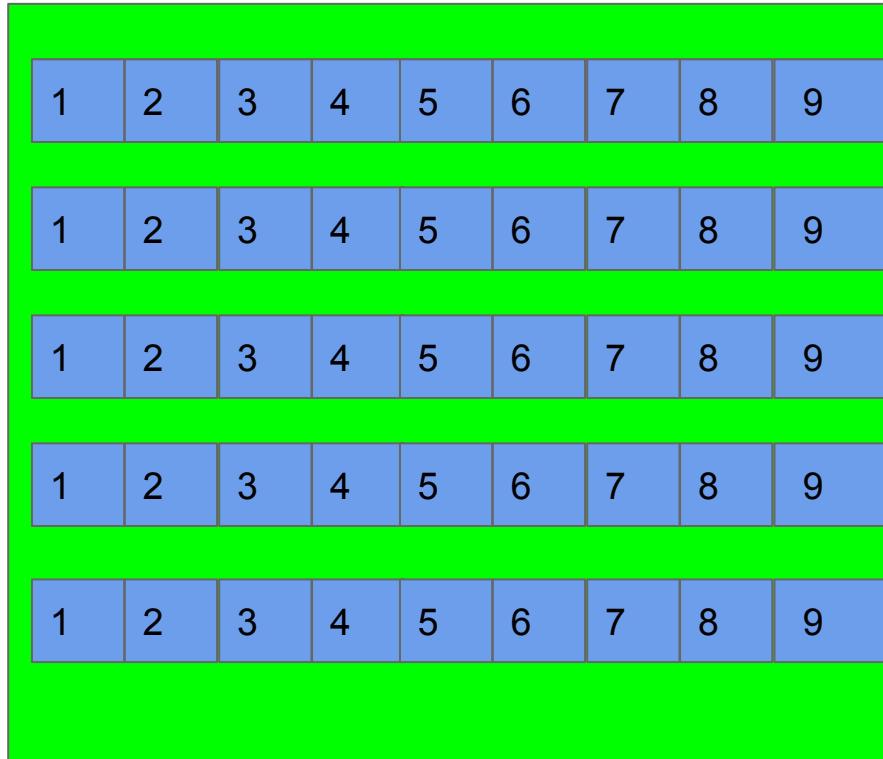
To find the content of the log file and index file, execute the below command.

```
./kafka-run-class.sh kafka.tools.DumpLogSegments --deep-iteration --print-data-log --files  
/tmp/dharshekth-kafka/CHE-18/00000000000000000000000000000000.log
```

```
./kafka-run-class.sh kafka.tools.DumpLogSegments --deep-iteration --print-data-log --files  
/tmp/dharshekth-kafka/CHE-18/00000000000000000000000000000000.index
```

```
./kafka-run-class.sh kafka.tools.DumpLogSegments --deep-iteration --print-data-log --files  
/tmp/dharshekth-kafka/CHE-18/00000000000000000000000000000000.timeindex
```

# Producer - Consumers - Partition



# Kafka Producer - Java

```
Properties props = new Properties();
```

```
props.put("zk.connect", "localhost:2181");
props.put("serializer.class", "kafka.serializer.StringEncoder");
props.put("metadata.broker.list", "localhost:9092");
```

```
ProducerConfig config = new ProducerConfig(props);
```

```
Producer<String, String> producer = new Producer<String, String>(config);
```

```
producer.send(new KeyedMessage<String, String>("Lohith_Topic", "BUSINESS-MESSAGE"));
```

```
producer.close();
```



# Kafka Producer - Acknowledgement

`props.put("request.required.acks", "0")` - no acknowledgement

`props.put("request.required.acks", "1")` -

producer gets an acknowledgement after the leader gets the data.

`props.put("request.required.acks", "-1")` -

producer gets an acknowledgement after all the ISR (In Sync Replicas) have received the data.

# Kafka data retention

Kafka by default keeps the data in the topic upto 7 days. This is present in the configuration *retention.ms*. *The default value of the retention.ms is 604800000.*

*The above value can be overridden to any value of choice.*

# Kafka Consumer

# Kafka Consumer

A client which consumes record from the Kafka Cluster, typically from a topic.

Consumers use TCP connections to communicate data.

Consumers are not thread-safe.

# Consumer

```
val props = new Properties();
props.put("bootstrap.servers", "localhost:9092");
props.put("group.id", "SLZ-ZONE-GROUP");          // Give a group name
props.put("enable.auto.commit", "true");
props.put("auto.commit.interval.ms", "1000");
props.put("session.timeout.ms", "30000");
props.put("key.deserializer", "org.apache.kafka.common.serialization.StringDeserializer");
props.put("value.deserializer", "org.apache.kafka.common.serialization.StringDeserializer");

val consumer = new KafkaConsumer[String, String](props);
consumer.subscribe(util.Collections.singletonList("TIGER"));           // Subscribe the consumer to a TOPIC

while(true) {
  val records=consumer.poll(0)

  for (record<-records.asScala){
    println(record)
  }
}
```

# Consumers

To list all the consumers, use the command below.

```
./kafka-consumer-groups.sh --bootstrap-server localhost:9092 --list
```

```
./kafka-consumer-groups.sh --describe --group EWS-TOPIC-GROUP --bootstrap-server localhost:9092
```

# Consumers - Broadcast

There can be multiple consumers in a *Consumer-Group*.

Multiple *Consumer-Group* can be subscribed to a topic.

When a producer sends message to a topic, all *Consumer-Group* receives messages.

But only one consumer in a *Consumer-Group* will receive message and processes it.

In order to achieve broadcast, have multiple *Consumer-Group*, so all groups will receive messages.

If you want all your consumers to receive messages, assign them to different consumer groups.

Each message goes to all the consumer groups, but within a *Consumer-Group* it goes to only one consumer.

# Consumers - Broadcast

Queue and Topic implementation through Consumer-Broadcast

# Kafka Consumer - Java

```
Properties props = new Properties();
props.put("bootstrap.servers", "localhost:9092");

// When changed the consumer group, it behaves as a TOPIC or QUEUE
props.put("group.id", "ECHELON-CONSUMER2");

props.put("enable.auto.commit", "false");
props.put("auto.commit.interval.ms", "1000");
props.put("session.timeout.ms", "30000");
props.put("key.deserializer", "org.apache.kafka.common.serialization.StringDeserializer");
props.put("value.deserializer", "org.apache.kafka.common.serialization.StringDeserializer");
```

Declare the properties first

# Kafka Consumer - Java

```
KafkaConsumer consumer = new KafkaConsumer<>(props);
consumer.subscribe(Arrays.asList("ECHELON-TOPIC"));

while(true) {

    ConsumerRecords<String, String> records = consumer.poll(1);

    for (ConsumerRecord<String, String> record : records) {

        Set<TopicPartition> partitions = consumer.assignment();

        String formattedText = String.format("Key = %s - Value is %s - Offset is %s - Partition is %s - Partitions size = %s ", record.key(),
                                             record.value(),
                                             record.offset(),
                                             record.partition(),
                                             partitions.size());

        System.out.println(formattedText);

    }
}
```

chinnasanyad@gmail.com



# Kafka Consumer - Topic Routing

```
KafkaConsumer consumer = new KafkaConsumer<>(props);
consumer.subscribe(Arrays.asList("ECHELON-TOPIC"));

while(true) {

    ConsumerRecords<String, String> records = consumer.poll(1);

    for (ConsumerRecord<String, String> record : records) {

        // Unsubscribe the consumer from listening to current topic and subscribe to a new topic
        if (record.key().equals("unsubscribe")) {
            consumer.unsubscribe();
            consumer.subscribe(Arrays.asList(record.value()));
        }
    }

}
```

# Kafka Consumer - Clean Exit

```
KafkaConsumer consumer = new KafkaConsumer<>(props);
consumer.subscribe(Arrays.asList("ECHELON-TOPIC"));

while(true) {

    ConsumerRecords<String, String> records = consumer.poll(1);
    boolean shouldConsumerBeClosed = false;

    for (ConsumerRecord<String, String> record : records) {

        // Closing and coming out clean from a consumer
        if (record.key().equals("exit"))
            shouldConsumerBeClosed = true;

    }

    if (shouldConsumerBeClosed) {
        consumer.close();
        break;
    }

}
```



# Kafka Consumer - Consume from a specific partition

Consumer can consume from a specific partition. The below code will consume from a specific partition 10.

```
KafkaConsumer consumer = new KafkaConsumer<String, String>(props);  
  
TopicPartition partition2 = new TopicPartition("VEH-TOPIC", 10);  
consumer.assign(Arrays.asList(partition2));  
  
while (true) {  
    ConsumerRecords records = consumer.poll(1);  
  
    records.forEach(new ForEachDataIteration());  
}  
}
```



# Kafka Consumer - Multiple Consumers

```
Properties props = new Properties();
props.put("bootstrap.servers", "localhost:9092");

props.put("group.id", "ECHELON-CONSUMER-GROUP");

props.put("enable.auto.commit", "false");
props.put("auto.commit.interval.ms", "1000");
props.put("session.timeout.ms", "30000");
props.put("key.deserializer", "org.apache.kafka.common.serialization.StringDeserializer");
props.put("value.deserializer", "org.apache.kafka.common.serialization.StringDeserializer");
```

**IMPORTANT**

To create multiple consumers. Run the code under the **Same** group. Here the group is **ECHELON-CONSUMER-GROUP**.



# Kafka Consumer - Topic and Queue

```
Properties props = new Properties();
props.put("bootstrap.servers", "localhost:9092");

// When changed the consumer group, it behaves as a TOPIC or QUEUE
props.put("group.id", "ECHELON-CONSUMER-GROUP");

props.put("enable.auto.commit", "false");
props.put("auto.commit.interval.ms", "1000");
props.put("session.timeout.ms", "30000");
props.put("key.deserializer", "org.apache.kafka.common.serialization.StringDeserializer");
props.put("value.deserializer", "org.apache.kafka.common.serialization.StringDeserializer");
```

**IMPORTANT**

When the group is **changed**,  
the consumer will behave as a topic else  
it will be a queue. Here the group is  
**ECHELON-CONSUMER-GROUP.**

# Kafka Consumer - seekToBeginning

```
Set<TopicPartition> partitions = consumer.assignment();
```

```
if (record.key().equals("seektobeginning"))
    consumer.seekToBeginning(partitions);
```

# Kafka Consumer - seekToEnd

```
Set<TopicPartition> partitions = consumer.assignment();
```

```
if (record.key().equals("seektoend"))
    consumer.seekToEnd(partitions);
```

# Kafka Consumer - Commit your offset

Consumer consuming the offset can commit the offset automatically or programmatically

```
val consumer = new KafkaConsumer[String, String](props)

// Consumer consuming from a Particular Topic
consumer.subscribe(Arrays.asList("AWS"))

while (true) {
    val records = consumer.poll(1)

    records.forEach(new ForEacher)

    // Non-Blocking commit.
    consumer.commitAsync()

    // Blocking commit until the offsets has been successfully committed.
    consumer.commitSync()

}
```

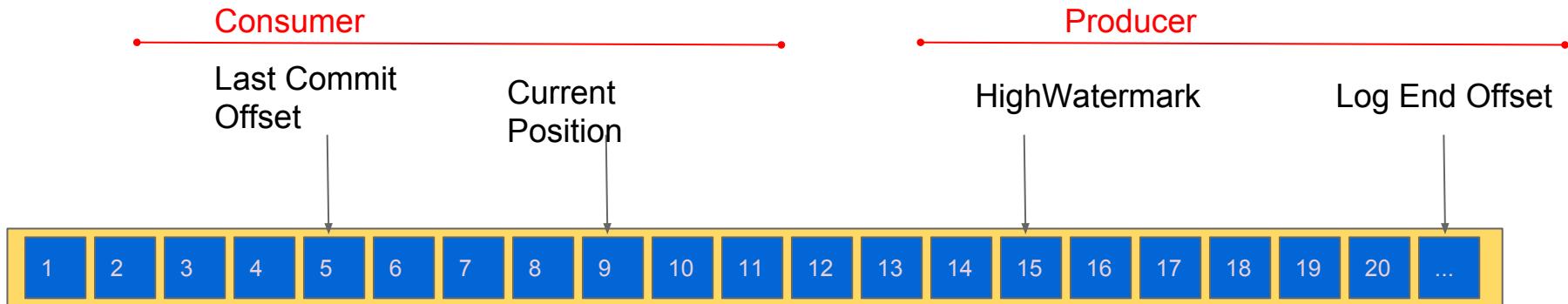
# Message - Offset

# Offset

Kafka maintains a numerical pointer to each of partition called by the name **offset**.

Consumer identify the data in the segment using a pointer call Offset. (or) The position of the consumer is known by looking at the partition offset.

# Partition - Offset



# Kafka Partition - Offset

ISR      In Sync Replica

LCO      Last Committed Offset

LEO      Log End Offset

HW      High Watermark

CP      Current Position

Last Committed Offset - It is the last offset that is committed by a consumer.

Current Position - It is the current position of the consumer where it reads messages.

Log End Offset (LEO) is the offset that a producer writes to the partition.

High Watermark (HW) is the offset of messages that are fully replicated.

Log End Offset is always higher than HW.

Consumers can only consume messages up to high water mark, meaning  $CP < HW$



# Kafka Partition - Offset

ISR      In Sync Replica

**LCO**      **Last Committed Offset**

*The LCO is the committed offset that has been saved. If the consumer process fails and restart, the consumer will restart its consuming from the LCO.*

LEO      Log End Offset

HW      High Watermark

CP      Current Position

# Offset

The offset acts a unique identifier for a particular partition.

The offset progresses forward each time when the consumer receives call through poll() method.

# OFFSET

## Finding the Offset for a partition

```
$ ./kafka-consumer-groups.sh --bootstrap-server localhost:9092 --describe --group CHE-GROUP
```

```
$ ./kafka-console-consumer.sh --consumer.config /tmp/consumer.config --formatter "kafka.coordinator.GroupMetadataManager\$OffsetsMessageFormatter" --bootstrap-server localhost:9092 --topic __consumer_offsets
```

# Kafka Partition Data Replication

The High Watermark is updated by calculating,

*Minimum LEO across all the ISR of the partition.*

```
/*
 * Check and maybe increment the high watermark of the partition;
 * this function can be triggered when
 *
 * 1. Partition ISR changed
 * 2. Any replica's LEO changed
 *
 * Returns true if the HW was incremented, and false otherwise.
 * Note There is no need to acquire the leaderIsrUpdate lock here
 * since all callers of this private API acquire that lock
 */
private def maybeIncrementLeaderHW(leaderReplica: Replica): Boolean = {
  val allLogEndOffsets = inSyncReplicas.map(_.logEndOffset)
  val newHighWatermark = allLogEndOffsets.min(new
    LogOffsetMetadata.OffsetOrdering)
  val oldHighWatermark = leaderReplica.highWatermark
  if (oldHighWatermark.messageOffset <
    newHighWatermark.messageOffset ||
    oldHighWatermark.onOlderSegment(newHighWatermark)) {
    leaderReplica.highWatermark = newHighWatermark
    debug("High watermark for partition [%s,%d] updated to
      %s".format(topic, partitionId, newHighWatermark))
    True
  } else {
    debug("Skipping update high watermark since Old hw %s is larger than new hw %
      for partition [%s,%d]. All leo's are %s"
      .format(oldHighWatermark, newHighWatermark, topic, partitionId,
        allLogEndOffsets.mkString(",")))
    false
  }
}
```



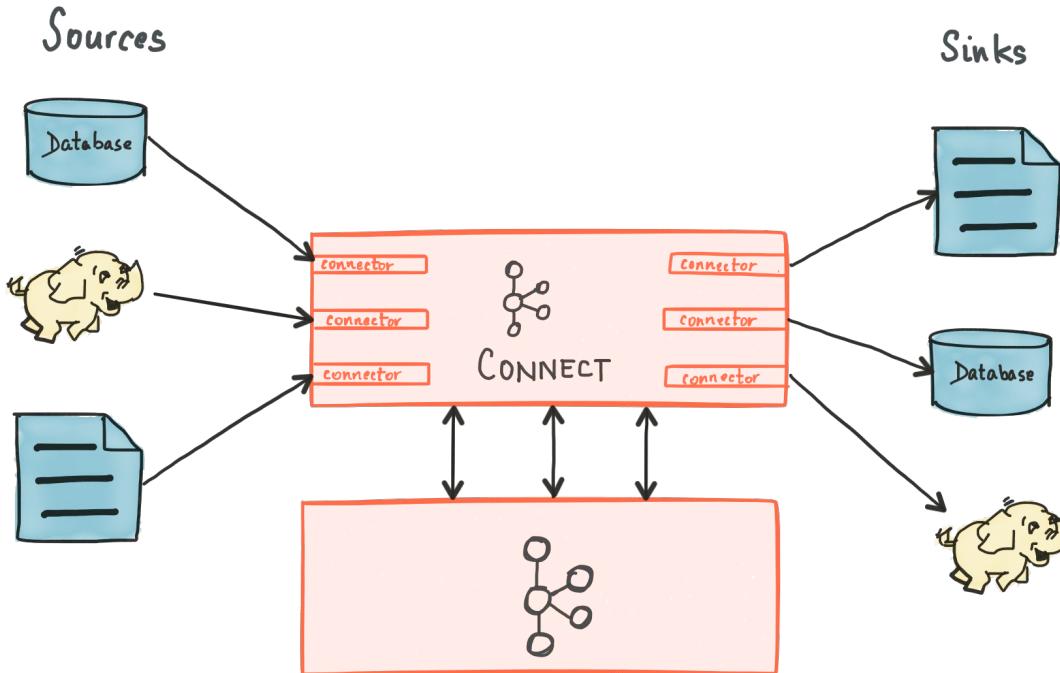
# Kafka Installation

Use console producer to produce messages. The below command produces messages from the console.

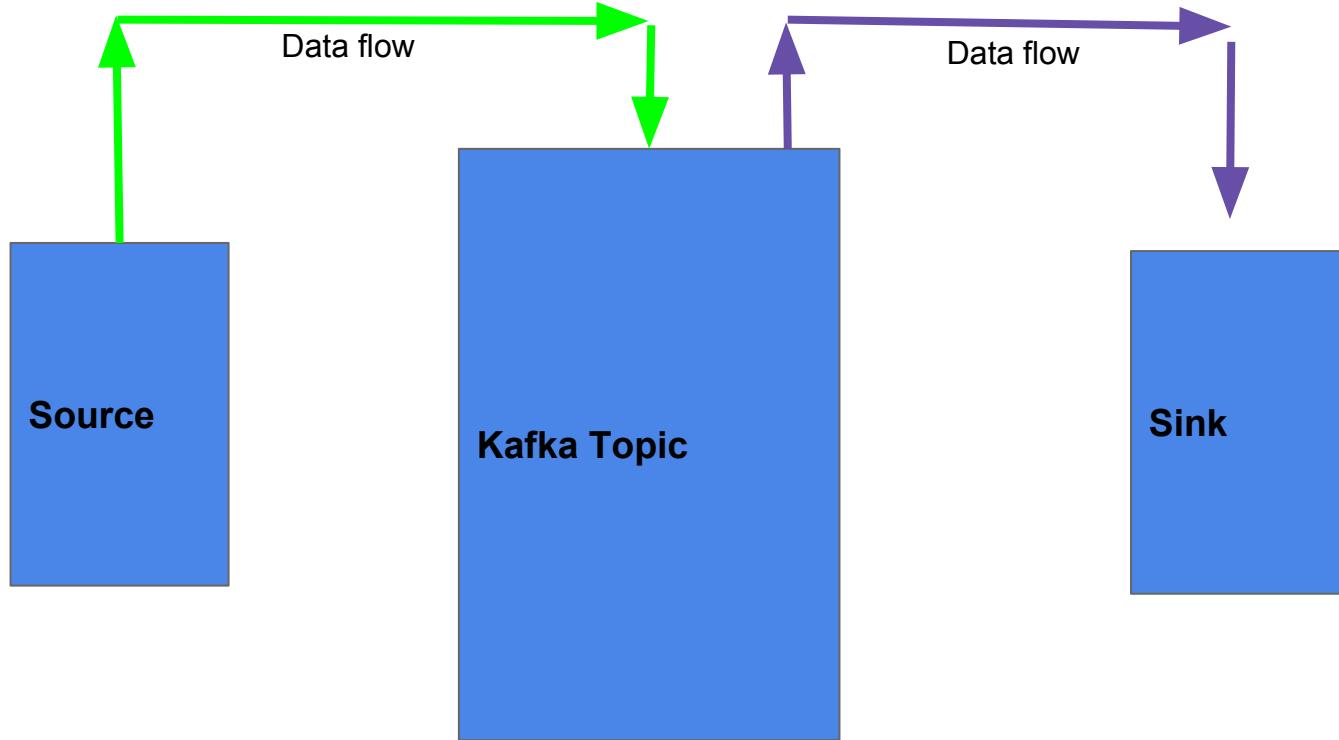
```
./kafka-console-producer.sh --broker-list localhost:9092 --topic Lohith_Topic
```

# Kafka Connect

# Kafka Connect



# Kafka Connect



# Kafka Connect

Apache Kafka Connect supports quick define connectors that move large collections of data from other systems to kafka and from kafka to other systems.

To integrate with systems through ease of development for Kafka.

# Kafka Connect

Standalone mode

`kafka/bin/connect-standalone.sh`

Distributed mode

`kafka/bin/connect-distributed.sh`

# Kafka Connect

./connect-standalone.sh

..../config/connect-standalone.properties

..../config/connect-file-source.properties

..../config/connect-file-sink.properties

# Kafka Connect - Source - File to Kafka Topic

```
$ ./connect-standalone.sh ..../config/connect-standalone.properties ..../config/connect-file-source.properties
```

## ***connect-file-source.properties***

```
name=INTELLIGENCE-SOURCE  
connector.class=FileStreamSource  
tasks.max=1  
file=intellisource.txt  
topic=BINTELLIGENCE
```

## ***connect-standalone.properties***

```
bootstrap.servers=localhost:9092  
  
#key.converter=org.apache.kafka.connect.json.JsonConverter  
#value.converter=org.apache.kafka.connect.json.JsonConverter  
  
key.converter=org.apache.kafka.connect.storage.StringConverter  
value.converter=org.apache.kafka.connect.storage.StringConverter  
  
key.converter.schemas.enable=true  
value.converter.schemas.enable=true
```

# Kafka Connect - Sink - Kafka Topic to HDFS

Download the confluent.

In the bin directory of the confluent, give in the below command. Both the config files are there in etc/ directory of the confluent.

```
$ ./connect-standalone .../etc/kafka/connect-standalone.properties .../etc/kafka-connect-hdfs/quickstart-hdfs.properties
```

## ***connect-file-source.properties***

```
name=hdfs-sink  
connector.class=io.confluent.connect.hdfs.HdfsSinkConnector  
tasks.max=1  
topics=HDFS-TOPICONE  
hdfs.url=hdfs://localhost:9000/hdfs kafka  
flush.size=3  
schemas.enable=false
```

## ***connect-standalone.properties***

```
bootstrap.servers=localhost:9092  
  
key.converter.schemas.enable=false  
value.converter.schemas.enable=false  
  
key.converter=org.apache.kafka.connect.storage.StringConverter  
value.converter=org.apache.kafka.connect.storage.StringConverter  
  
internal.key.converter=org.apache.kafka.connect.storage.StringConve  
rter  
internal.value.converter=org.apache.kafka.connect.storage.StringCon  
verter  
internal.key.converter.schemas.enable=false  
internal.value.converter.schemas.enable=false
```

# Kafka Compression

# Kafka supports data compression

Data is compressed by the producer.

Data is written in a compressed format on the server.

Data is decompressed by the consumer.

Data compression improves consumer throughput at the cost of decompression.

# Kafka supports the following compression algos

GZIP Compression

Snappy Compression

LZ4 Compression

# Kafka config for compression

## compression.codec

Specify which algorithm to be used by producer.  
Default is 0.

## compressed.topics

Compression on a topic level.  
Default is empty.

# Kafka config for compression

## compression.codec

- 0 default
- 1 GZIP compression
- 2 Snappy compression
- 3 LZ4 compression

## compressed.topics

“topic\_ONE, topic\_TWO”

# Kafka Streams



KAFKA STREAMS:  
*Stream processing made simple*

# Kafka Streams

- 1. Processor API - Low Level API
- 2. Kafka Streams DSL - High Level API

# Processor API

1. Create Streams Config

```
StreamsConfig config = new StreamsConfig(settings);
```

2. Build the TopologyBuilder

```
TopologyBuilder builder = new TopologyBuilder();
```

3. Create the KafkaStreams

```
KafkaStreams streams = new KafkaStreams(builder, config);
```

4. Start the streams

```
streams.start();
```

# Processor API - Topology Builder

1. Source
2. Processor
3. Sink
4. State Store

```
TopologyBuilder builder = new TopologyBuilder();
builder.addSource("Source", "BINTELLIGENCE")

.addProcessor("Process", new DataProcessSupplier(), "Source")

.addStateStore(dataStore, "Process")

.addSink("Sink", "P-INTELLIGENCE", "Process");
```

# Processor API - Topology Builder - Processor

```
public class DataProcessSupplier implements ProcessorSupplier<String, String> {
    @Override
    public Processor<String, String> get() {
        return new DataProcessor();
    }
}

public class DataProcessor implements Processor<String, String> {
    @Override
    public void init(ProcessorContext processorContext) {
    }

    @Override
    public void process(String key, String value) {
        System.out.println(key + value);
    }

    @Override
    public void punctuate(long l) {
    }

    @Override
    public void close() {
    }
}
```

# Processor API

The topology is nothing but a Acyclic graph of sources, processors and sinks.

Internally, it uses List, Set and Map as the basic data structure to construct the topology.

# Building a complex topology

```
TopologyBuilder builder = new TopologyBuilder();

builder.addSource("Source", "DATASOURCE-TOPIC")
    .addProcessor("CAPITAL-PROCESSOR",new DataProcessSupplier(), "Source")
    .addProcessor("CAPITAL-LENGTH-PROCESSOR",new CapitalToLengthSupplier(), "CAPITAL-PROCESSOR")
    .addStateStore(dataStore, "CAPITAL-PROCESSOR")
    .addStateStore(lengthStore, "CAPITAL-LENGTH-PROCESSOR")
    .addSink("DataSink","DATA-OCEAN-TOPIC", "CAPITAL-PROCESSOR")
    .addSink("LengthSink","DATA-LENGTH-TOPIC", "CAPITAL-LENGTH-PROCESSOR");
```

```
KafkaStreams streams = new KafkaStreams(builder, config);
streams.start();
```

# Building a complex topology

```
class CapitalToLengthSupplier implements ProcessorSupplier<String, String> {  
    @Override  
    public Processor<String, String> get() {  
        return new LetterToLengthProcessor();  
    }  
}
```

```
class LetterToLengthProcessor implements Processor<String, String> {  
  
    ProcessorContext processorContext = null;  
    private KeyValueStore<String, String> kvStore = null;  
  
    @Override  
    public void init(ProcessorContext _processorContext) {  
  
        this.processorContext = _processorContext;  
        kvStore = (KeyValueStore)  
        _processorContext.getStateStore("LENGTH_STORE");  
  
        // call the punctuate  
        this.processorContext.schedule(1000);  
    }  
  
    @Override  
    public void process(String key, String value) {  
  
        // Do complex processing and forward it to next topic  
        processorContext.forward(Integer.toString(key.length()),  
        Integer.toString(value.length()));  
        kvStore.put(Integer.toString(key.length()), Integer.toString(value.length()));  
    }  
  
    @Override  
    public void punctuate(long l) {  
        processorContext.commit();  
    }  
  
    @Override  
    public void close() {  
  
        kvStore.close();  
    }  
}
```

# Kafka Streams DSL

KStream

KTable

GlobalKTable

KGroupedStream

# Kafka Streams DSL

KStream - A record stream. Data is INSERT here.

KTable - Changelog stream. Date is UPDATE here based on key. If key not found, then INSERT.

GlobalKTable - Changelog stream.

# Kafka Streams DSL

1. Create the KStreamBuilder

```
val builder = new KStreamBuilder
```

2. Connect to a topic and create a stream

```
val users : KStream[String, String] = builder.stream(Serdes.String(), Serdes.String(), "BINT")
```

3. Code one or more transformations and one terminal operations.

```
users.foreach(new Printer)
```

4. Create kafka stream and start the streaming

```
val stream = new KafkaStreams(builder, props)  
stream.start()
```

# Kafka Streams - Transformation operations

map()

groupBy()

mapValues()

groupByKey()

filter()

join()

filterNot()

flatMap()

flatMapValues()

# Kafka Streams - Terminal Operations

foreach()

print()

writeAsText("")

peek()

# Kafka Streams - map()

```
KStreamBuilder builder = new KStreamBuilder();  
  
KStream data = builder.stream(Serdes.String(), Serdes.String(), "Squad-Topic");  
  
//KStream mappedValuesData = data.mapValues(each -> each.toString().toUpperCase());  
KStream mappedData = data.map(((key, value) -> new KeyValue<>(key.toString().toUpperCase(),  
value.toString().toUpperCase())));
```

# Kafka Streams - mapValues()

```
KStream data = builder.stream(Serdes.String(), Serdes.String(), "Squad-Topic");  
KStream mappedValuesData = data.mapValues(each -> each.toString().toUpperCase());
```

# Kafka Streams - filter()

```
KStream filteredData = mappedData.filter((key, value) -> key.toString().startsWith("A"));
```

# Kafka Streams - filterNot()

```
KStream inverseFilteredData = mappedData.filterNot((key, value) -> key.toString().startsWith("A"));
```

# Kafka Streams - foreach()

```
mappedData.foreach((key,value) -> System.out.println("The key is " + key + " and value is " + value));
```

# Kafka Streams - peek()

```
mappedData.peek((key,value) -> System.out.println("The key is " + key + " and value is " + value));
```

# Kafka Streams - branch

```
KStream<String, String>[] branches = mappedData.branch((key, value) ->
    value.toString().startsWith("MESH"),
    (key, value) -> value.toString().startsWith("CONTRACT"),
    (key, value) -> true
);
```

```
KStream meshData = branches[0];
KStream contractData = branches[1];
KStream allOtherData = branches[2];
```

# Kafka Streams - Data transfer to other topic to

```
mappedData.foreach((key,value) -> System.out.println("The key is " + key + " and value is " + value));
```

```
mappedData.to("Squad-SVM-Model");
```

# Kafka Streams - Data transfer to other topic through

```
mappedData.through("Squad-SVM-Model")
    .map(((key, value) -> new KeyValue<>(key.toString().length(), value.toString().length())))
    .print();
```

# Kafka Streams - groupByKey()

```
val builder = new KStreamBuilder  
val data = builder.stream(Serdes.String, Serdes.String, "BDAS")  
  
val groupedStream:KGroupedStream[String, String] = data.groupByKey()
```

groupByKey() returns a KGroupedStream.

Only in a KGroupedStream datastructure you can perform a reduce() operation.

# Kafka Streams - count()

```
val groupedStream:KGroupedStream[String, String] = data.groupByKey()
```

```
val countAgg = groupedStream.count()
```

count() operation is available on a grouped stream only.

Remember: The count() transformation always return a KTable.

# Kafka Streams - Exception Handling

To catch an unexpected exception, the `UncaughtExceptionHandler` has to be coded and logged.

```
KafkaStreams stream = new KafkaStreams(builder, props);

stream.setUncaughtExceptionHandler(new Thread.UncaughtExceptionHandler() {
    @Override
    public void uncaughtException(Thread t, Throwable e) {
        System.out.println(e.getMessage());
    }
});
```

After all the computations, close the stream.  
`stream.close();`

# Kafka Streams - KTable

```
KStreamBuilder builder = new KStreamBuilder();
```

```
KTable data = builder.table(Serdes.String(), Serdes.String(), "KTABLE-TOPIC");
```

```
data.toStream().foreach((x,y) -> System.out.println("Key = " + x + " Value" + y));
```

```
Properties props = new Properties();
props.put(StreamsConfig.APPLICATION_ID_CONFIG, "KTABLEJOB");
props.put(StreamsConfig.BOOTSTRAP_SERVERS_CONFIG, "localhost:9092");
props.put(StreamsConfig.DEFAULT_VALUE_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName());
props.put(StreamsConfig.DEFAULT_KEY_SERDE_CLASS_CONFIG, Serdes.String().getClass().getName());
props.put(StreamsConfig.CACHE_MAX_BYTES_BUFFERING_CONFIG, 1000L);
```

```
KafkaStreams stream = new KafkaStreams(builder, props);
```

```
stream.start();
```

# Kafka Streams - KTable - Best Practice

```
KStreamBuilder builder = new KStreamBuilder();  
  
KTable data = builder.table(Serdes.String(), Serdes.String(), "KTABLE-TOPIC");  
  
data.toStream().foreach((x,y) -> System.out.println("Key = " + x + " Value" + y));
```

Always do a toStream() and do the data manipulations.

# Kafka Streams - KTable - Transformations

```
KStreamBuilder builder = new KStreamBuilder();
```

```
KTable data = builder.table(Serdes.String(), Serdes.String(), "KTABLE-TOPIC");
```

```
data.filter()
```

```
data.filterNot()
```

```
data.groupBy()
```

```
data.join()
```

```
data.leftJoin()
```

```
data.outerJoin()
```

```
data.through()
```

```
data.to();
```

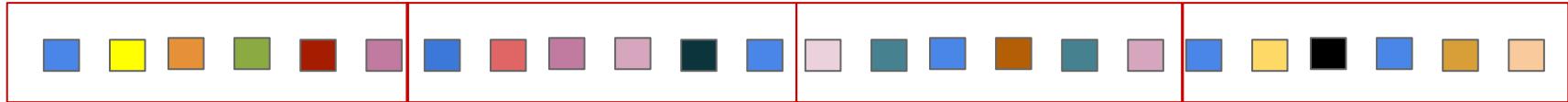
The following are the transformation supported by the KTable. Not all KStream transformation is supported by KTable.

# Kafka Streams - Windowing

Tumbling time window

Hopping time window

## Kafka Streams - Tumbling time window



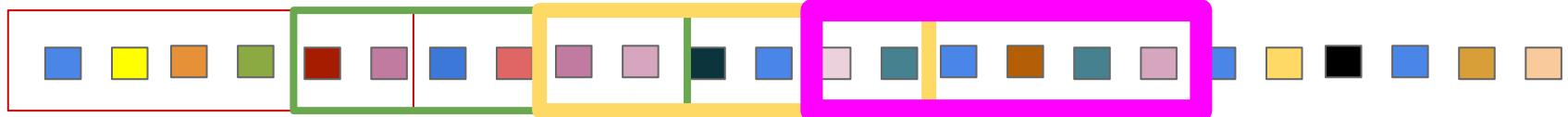
```
alerts.map[String, String](new MapData)
    .groupByKey()

    .count(TimeWindows.of(TimeUnit.SECONDS.toMillis(60)), "alert-store")

    .toStream

    .foreach(new PrinterAllData)
```

## Kafka Streams - Hoping time window



```
alerts.map[String, String](new MapData)
    .groupByKey()
    .count(TimeWindows.of(TimeUnit.SECONDS.toMillis(60)))
        .advanceBy(TimeUnit.SECONDS.toMillis(30)), "alert-store")

    .toStream

    .foreach(new PrinterAllData)
```

# Kafka ACL

## Access Control List

# Kafka ACL

First enable the ACL for kafka

[1] In the server.properties, add the following below two lines:

```
authorizer.class.name=kafka.security.auth.SimpleAclAuthorizer  
super.users=User:dharshekthvel
```

Then restart the zookeeper and kafka.

# Kafka ACL - Commands

```
authorizer.class.name=kafka.security.auth.SimpleAclAuthorizer
```

```
./kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic HULKER
```

```
./kafka-acls.sh --authorizer-properties zookeeper.connect=localhost:2181 --add --allow-principal User:* --operation ALL --topic BASHAS
```

```
./kafka-acls.sh --authorizer-properties zookeeper.connect=localhost:2181 --add --allow-principal User:* --operation WRITE --topic HULK
```

```
./kafka-acls.sh --authorizer-properties zookeeper.connect=localhost:2181 --add --allow-principal User:* --producer --topic HULKER
```

```
./kafka-acls.sh --authorizer-properties zookeeper.connect=localhost:2181 --authorizer kafka.security.auth.SimpleAclAuthorizer --add  
--allow-principal User:* --producer --topic BASHAS
```

```
./kafka-acls.sh --authorizer-properties zookeeper.connect=localhost:2181 --authorizer kafka.security.auth.SimpleAclAuthorizer --add  
--allow-principal User:* --consumer --topic BASHAS --group *
```

```
./kafka-acls.sh --authorizer-properties zookeeper.connect=localhost:2181 --add --deny-principal User:* --operation ALL --topic DATAS
```

```
./kafka-acls.sh --authorizer-properties zookeeper.connect=localhost:2181 --list --topic FARM-TOPIC
```

# Kafka ACL - To turn debugging on

```
log4j.logger.kafka.authorizer.logger=DEBUG, authorizerAppender  
log4j.additivity.kafka.authorizer.logger=false
```

```
log4j.appendер.authorizerAppender=org.apache.log4j.DailyRollingFileAppender  
log4j.appendер.authorizerAppender.DatePattern='yyyy-MM-dd-HH  
log4j.appendер.authorizerAppender.File=${kafka.logs.dir}/kafka-authorizer.log  
log4j.appendер.authorizerAppender.layout=org.apache.log4j.PatternLayout  
log4j.appendер.authorizerAppender.layout.ConversionPattern=[%d] %p %m (%c)%n
```

# Kafka - Zookeeper

# Kafka - Zookeeper

Let's deep dive into how kafka interacts with zookeeper.

[1] Lets create a topic with name BULK\_LOADER with 100 partitions.

```
./kafka-topics.sh --create --zookeeper localhost:2181 --replication-factor 1 --partitions 100 --topic BULK_LOADER
```

[2] Kick on the zookeeper cli and put a ls on the below path:

In the zookeeper-CLI, give in the below command.

```
$ ls /brokers/topics/BULK_LOADER/partitions  
You will see 100 partitions on it.
```

```
$ ls /brokers/topics/BULK_LOADER/partitions/85/state
```

[3] Top list the brokers attached to the zookeeper

```
$ ls /brokers/ids
```

# ELK Stack

Elasticsearch

Logstash

Kibana



# Elasticsearch

It offers a competitive edge over splunk



# Elasticsearch Installation

[1] Download ElasticSearch 1.7.2 from

<https://www.elastic.co/downloads/elasticsearch>

[2] Download JDK and set the JAVA\_HOME variable

[3] Extract the elastic.elasticsearch-1.7.2.zip file.

[4] After the extraction, go inside the *elasticsearch/bin* directory and run the *elasticsearch.bat* to start the elasticsearch.

# Elasticsearch Installation

```
ca| Elasticsearch 1.7.2

D:\elasticsearch172\bin>dir
Volume in drive D has no label.
Volume Serial Number is 264F-D60B

Directory of D:\elasticsearch172\bin

10/13/2015  02:40 PM    <DIR>          .
10/13/2015  02:40 PM    <DIR>          ..
10/13/2015  02:40 PM           8,114 elasticsearch
10/13/2015  02:40 PM        104,448 elasticsearch-service-mgr.exe
10/13/2015  02:40 PM        103,936 elasticsearch-service-x64.exe
10/13/2015  02:40 PM        80,896 elasticsearch-service-x86.exe
10/13/2015  02:40 PM           901 elasticsearch.bat
10/13/2015  02:40 PM           2,797 elasticsearch.in.bat
10/13/2015  02:40 PM           2,170 elasticsearch.in.sh
10/13/2015  02:40 PM           2,523 plugin
10/13/2015  02:40 PM           482 plugin.bat
10/13/2015  02:40 PM           6,210 service.bat
                           10 File(s)   312,477 bytes
                           2 Dir(s)  49,467,441,152 bytes free

D:\elasticsearch172\bin>elasticsearch.bat
[2015-10-13 15:03:49,825][INFO ][node] [Hydro] version[1.7.
[2015-10-13 15:03:49,826][INFO ][node] [Hydro] initializing
[2015-10-13 15:03:50,154][INFO ][plugins] [Hydro] loaded [1], s
ites []
[2015-10-13 15:03:50,328][INFO ][env] [Hydro] using [1] da
ta paths, mounts [[(D:)]], net usable_space [46gb], net total_space [50.7gb], ty
pes [NTFS]
[2015-10-13 15:03:58,486][INFO ][node] [Hydro] initialized
[2015-10-13 15:03:58,487][INFO ][node] [Hydro] starting ...
[2015-10-13 15:03:58,935][INFO ][transport] [Hydro] bound_addres
s {inet[/0:0:0:0:0:0:9300]}, publish_address {inet[/10.15.10.137:9300]}
[2015-10-13 15:03:59,376][INFO ][discovery] [Hydro] elasticsearc
h/UOWFiaLgQX-1ibBSNQcr5A
[2015-10-13 15:04:03,238][INFO ][cluster.service] [Hydro] new_master [
Hydro][UOWFiaLgQX-1ibBSNQcr5A][2400002720-LT33][inet[/10.15.10.137:9300]], reaso
n: zen-disco-join {elected_as_master}
[2015-10-13 15:04:03,373][INFO ][gateway] [Hydro] recovered [0]
[indices into cluster_state]
[2015-10-13 15:04:03,385][INFO ][http] [Hydro] bound_addres
s {inet[/0:0:0:0:0:0:9200]}, publish_address {inet[/10.15.10.137:9200]}
[2015-10-13 15:04:03,386][INFO ][node] [Hydro] started
```

# Elasticsearch Installation

```
bin — java -Xms256m -Xmx1g -Djava.awt.headless=true -XX:+UseParNewGC -XX:+UseConcMarkSweepGC -XX:CMSInitiatingOccupancyFraction=75 -XX:+UseCMSInitiatingOccupancyFraction=75  
elasticsearch-service-mgr.exe  elasticsearch.in.sh  
elasticsearch-service-x64.exe  plugin  
elasticsearch-service-x86.exe  plugin.bat  
elasticsearch.bat            service.bat  
[Dharshekthi-MacBook-Pro:bin dharshekthi]$ ./elasticsearch  
[2017-02-15 19:51:41,727][INFO ][node              ] [Screech] version[2.4.1], pid[2124], build[c67dc32/2016-09-27T18:57:55Z]  
[2017-02-15 19:51:41,729][INFO ][node              ] [Screech] initializing ...  
[2017-02-15 19:51:42,241][INFO ][plugins           ] [Screech] modules [reindex, lang-expression, lang-groovy], plugins [], sites []  
[2017-02-15 19:51:42,268][INFO ][env               ] [Screech] using [1] data paths, mounts [[(/dev/disk1)]], net usable_space [19.8gb], net total_space [232.6gb]  
b], spins? [unknown], types [hfs]  
[2017-02-15 19:51:42,268][INFO ][env               ] [Screech] heap size [989.8mb], compressed ordinary object pointers [true]  
[2017-02-15 19:51:42,269][WARN ][env               ] [Screech] max file descriptors [10240] for elasticsearch process likely too low, consider increasing to at least [65536]  
[2017-02-15 19:51:43,758][INFO ][node              ] [Screech] initialized  
[2017-02-15 19:51:43,758][INFO ][node              ] [Screech] starting ...  
[2017-02-15 19:51:43,825][INFO ][transport         ] [Screech] publish_address {127.0.0.1:9300}, bound_addresses {[fe80::1]:9300}, {[::]:9300}, {127.0.0.1:9300}  
[2017-02-15 19:51:43,830][INFO ][discovery        ] [Screech] elasticsearch/Q3QMOMSJQIOnCIGp2Xaufw  
[2017-02-15 19:51:46,872][INFO ][cluster.service  ] [Screech] new_master {Screech}{Q3QMOMSJQIOnCIGp2Xaufw}{127.0.0.1}{127.0.0.1:9300}, reason: zen-disco-join(elected_as_master, [0] joins received)  
[2017-02-15 19:51:46,888][INFO ][http              ] [Screech] publish_address {127.0.0.1:9200}, bound_addresses {[fe80::1]:9200}, {[::]:9200}, {127.0.0.1:9200}  
[2017-02-15 19:51:46,888][INFO ][node              ] [Screech] started  
[2017-02-15 19:51:46,900][INFO ][gateway          ] [Screech] recovered [0] indices into cluster_state  
[2017-02-15 19:52:16,889][WARN ][cluster.routing.allocation.decider] [Screech] high disk watermark [90%] exceeded on [Q3QMOMSJQIOnCIGp2Xaufw] [Screech] [/Users/dharshekthivel/ac/elasticsearch241/data/elasticsearch/nodes/0] free: 19.8gb[8.5%], shards will be relocated away from this node  
[2017-02-15 19:52:16,889][INFO ][cluster.routing.allocation.decider] [Screech] rerouting shards: [high disk watermark exceeded on one or more nodes]  
[2017-02-15 19:52:46,900][WARN ][cluster.routing.allocation.decider] [Screech] high disk watermark [90%] exceeded on [Q3QMOMSJQIOnCIGp2Xaufw] [Screech] [/Users/dharshekthivel/ac/elasticsearch241/data/elasticsearch/nodes/0] free: 19.8gb[8.5%], shards will be relocated away from this node  
[2017-02-15 19:53:16,904][WARN ][cluster.routing.allocation.decider] [Screech] high disk watermark [90%] exceeded on [Q3QMOMSJQIOnCIGp2Xaufw] [Screech] [/Users/dharshekthivel/ac/elasticsearch241/data/elasticsearch/nodes/0] free: 19.8gb[8.5%], shards will be relocated away from this node  
[2017-02-15 19:53:16,904][INFO ][cluster.routing.allocation.decider] [Screech] rerouting shards: [high disk watermark exceeded on one or more nodes]  
[2017-02-15 19:53:46,910][WARN ][cluster.routing.allocation.decider] [Screech] high disk watermark [90%] exceeded on [Q3QMOMSJQIOnCIGp2Xaufw] [Screech] [/Users/dharshekthivel/ac/elasticsearch241/data/elasticsearch/nodes/0] free: 19.8gb[8.5%], shards will be relocated away from this node  
[2017-02-15 19:54:16,915][WARN ][cluster.routing.allocation.decider] [Screech] high disk watermark [90%] exceeded on [Q3QMOMSJQIOnCIGp2Xaufw] [Screech] [/Users/dharshekthivel/ac/elasticsearch241/data/elasticsearch/nodes/0] free: 19.8gb[8.5%], shards will be relocated away from this node  
[2017-02-15 19:54:16,915][INFO ][cluster.routing.allocation.decider] [Screech] rerouting shards: [high disk watermark exceeded on one or more nodes]  
[2017-02-15 19:54:46,919][WARN ][cluster.routing.allocation.decider] [Screech] high disk watermark [90%] exceeded on [Q3QMOMSJQIOnCIGp2Xaufw] [Screech] [/Users/dharshekthivel/ac/elasticsearch241/data/elasticsearch/nodes/0] free: 19.8gb[8.5%], shards will be relocated away from this node  
[2017-02-15 19:55:16,920][WARN ][cluster.routing.allocation.decider] [Screech] high disk watermark [90%] exceeded on [Q3QMOMSJQIOnCIGp2Xaufw] [Screech] [/Users/dharshekthivel/ac/elasticsearch241/data/elasticsearch/nodes/0] free: 19.8gb[8.5%], shards will be relocated away from this node  
[2017-02-15 19:55:16,920][INFO ][cluster.routing.allocation.decider] [Screech] rerouting shards: [high disk watermark exceeded on one or more nodes]
```



# Elasticsearch Installation

Elastic Search starts on 9200 and when hit the port should return the below JSON

<http://localhost:9200/>

```
{  
  "status" : 200,  
  "name" : "Hydro",  
  "cluster_name" : "elasticsearch",  
  "version" : {  
    "number" : "1.7.2",  
    "build_hash" : "e43676b1385b8125d647f593f7202acbd816e8ec",  
    "build_timestamp" : "2015-09-14T09:49:53Z",  
    "build_snapshot" : false,  
    "lucene_version" : "4.10.4"  
  },  
  "tagline" : "You Know, for Search"  
}
```

# Elastic Search Commands

To clear the cache. Here the logstash-gpuz is the index.

[http://localhost:9200/logstash-gpuz\\*/\\_cache/clear](http://localhost:9200/logstash-gpuz*/_cache/clear)

To get a list of all indexes

[http://localhost:9200/\\_cat/indices?v](http://localhost:9200/_cat/indices?v)

To refresh

[http://localhost:9200/\\_refresh](http://localhost:9200/_refresh)

# Elastic Search Commands

To get the indices

```
$ curl http://localhost:9200/\_cat/indices?v
```

To delete an index

```
$ curl -XDELETE http://localhost:9200/uidai-\*
```

```
$ curl -XPUT 'http://localhost:9200/elephant-2015.12.02/_close'
```

```
$ curl -XPUT 'http://localhost:9200/elephant-2015.12.02/_settings' -d '{ "index" : {  
    "index.refresh_interval" : -1 } }'
```

```
$ curl -XPUT 'http://localhost:9200/elephant-2015.12.02/_open'
```

# Elastic Search - Create Index

```
import requests  
  
# Create index  
  
url = 'http://localhost:9200/index102'  
response = requests.put(url)  
  
print response.text
```

*To create a index, use the put API to put in the elastic search.*

# Elastic Search - Geo Index

```
{"mappings": {  
  "bank": {  
    "properties": {  
      "name": {  
        "type": "string"  
      },  
      "id": {  
        "type": "string"  
      },  
      "location": {  
        "type": "geo_point"  
      }  
    }  
  }  
}
```

The screenshot shows the Advanced REST client (ARC) interface. The left sidebar has 'ARC' selected. Under 'HTTP request', the method is set to POST, Content-Type is application/json, and the raw payload contains the provided JSON mapping. The URL is http://localhost:9200/bexpress-geo/\_geo.

```
POST /bexpress-geo/_geo  
Content-Type: application/json  
  
{"mappings": {  
  "bank": {  
    "properties": {  
      "name": {  
        "type": "string"  
      },  
      "id": {  
        "type": "string"  
      },  
      "location": {  
        "type": "geo_point"  
      }  
    }  
  }  
}
```

# ElasticSearch - Geo Index

```
{  
  "banks": {  
    "properties": {  
      "name": {  
        "type": "string"  
      },  
      "id": {  
        "type": "string"  
      },  
      "location": {  
        "type": "geo_point"  
      }  
    }  
  }  
}
```

The screenshot shows the Advanced REST client (ARC) interface. The left sidebar has tabs for ARC, Socket, History, Saved, and Projects. The main area is titled 'Request' with the URL `http://localhost:9200/bgeo/banks/_mapping`. The method is set to PUT. The 'Headers form' tab is selected, showing the Content-Type header set to application/json. The 'Raw payload' tab is selected, displaying the JSON mapping definition. The bottom status bar indicates 'Selected environment: default'.

```
HTTP request  
> http://localhost:9200/bgeo/banks/_mapping  
Socket  
History  
Saved  
Projects  
Request  
Use XHR  
⋮  
HTTP request  
Method: PUT  
Content-Type: application/json  
Headers form  
Headers sets  
Variables  
Content-Type: application/json  
ADD HEADER  
Raw payload  
Data form  
Files  
Raw payload  
{  
  "banks": {  
    "properties": {  
      "name": {  
        "type": "string"  
      },  
      "id": {  
        "type": "string"  
      },  
      "location": {  
        "type": "geo_point"  
      }  
    }  
  }  
30 bytes  
Selected environment: default
```

# ElasticSearch - Geo Index

For the creation and population of GEO Index,

1. First create the index
2. Then create the mappings.
3. Then feed the data.

```
# Create index
```

```
url = 'http://localhost:9200/index102'  
response = requests.put(url)  
  
print response.text
```

# ElasticSearch - Geo Index

For the creation and population of GEO Index,

1. First create the index
2. Then create the mappings.
3. Then feed the data.

# Create the mapping for the geo-index

```
#To create a GEO INDEX

url = 'http://localhost:9200/index102/banks/_mapping'

data = """"

{
  "banks": {
    "properties": {
      "name": {
        "type": "string"
      },
      "id": {
        "type": "string"
      },
      "location": {
        "type": "geo_point"
      }
    }
  }
"""

response = requests.put(url, data=data)
print response.text
```

# ElasticSearch - Geo Index

For the creation and population of GEO Index,

1. First create the index
2. Then create the mappings.
3. Then feed the data.

#### Feed the data

```
data = ""  
{  
    "name": "Goldman Sachs",  
    "location": {  
        "lat": 60.586967,  
        "lon": 15.422058  
  
    }  
}  
...  
  
range_of_counter = range(4000, 5000)  
  
for i in range_of_counter:  
    url = 'http://localhost:9200/index102/banks/%d?pretty' % i  
    response = requests.post(url, data=data)  
    print(" The response is - %s" % response.text)
```

# Logstash Installation

Download Logstash from <https://www.elastic.co/downloads/logstash>

Download the logstash 1.5.4 from the above link and extract it.

# Kibana Installation - Kibana 4.1.2

- [1] Download kibana from <https://www.elastic.co/downloads/kibana>
- [2] Extract the kibana-4.1.2-windows.zip file.
- [3] Go inside the *kibana/bin* directory and run the *kibana.bat* to start kibana.
- [4] Kibana should start by default in port 5601. You should be able to knock <http://localhost:5601/> and kibana should be up and running.

# Kibana Installation - Kibana 4.1.2

```
D:\kibana412>cd bin
D:\kibana412\bin>dir
 Volume in drive D has no label.
 Volume Serial Number is 264F-D60B

 Directory of D:\kibana412\bin

09/08/2015  01:12 PM    <DIR>
09/08/2015  01:12 PM    <DIR>
09/08/2015  01:12 PM                540 kibana
09/08/2015  01:12 PM            295 kibana.bat
              2 File(s)           835 bytes
              2 Dir(s)   49,435,271,168 bytes free

D:\kibana412\bin>kibana.bat
{"name":"Kibana","hostname":"240.002720-LT33","pid":6376,"level":30,"msg":"No existing kibana index found","time":"2015-10-13T09:49:47.679Z","v":0}
{"name":"Kibana","hostname":"240.002720-LT33","pid":6376,"level":30,"msg":"Listing on 0.0.0.0:5601","time":"2015-10-13T09:49:47.753Z","v":0}
```

The screenshot shows the Kibana Settings page at [localhost:5601/#/settings/indices/\\_g\(\)](http://localhost:5601/#/settings/indices/_g()). The top navigation bar includes links for Discover, Visualize, Dashboard, and Settings. The main content area is titled "Configure an index pattern". A warning message states: "Warning No default index pattern. You must select or create one to continue." Below this, a note says: "In order to use Kibana you must configure at least one index pattern. Index patterns are used to identify the Elasticsearch index to run search and analytics against. To configure fields." There are two checkboxes at the bottom: "Index contains time-based events" (checked) and "Use event times to create index names".

localhost:5601/#/settings/indices/\_g()

kibana

Discover   Visualize   Dashboard   Settings

Indices   Advanced   Objects   About

Index Patterns

Warning No default index pattern. You must select or create one to continue.

## Configure an index pattern

In order to use Kibana you must configure at least one index pattern. Index patterns are used to identify the Elasticsearch index to run search and analytics against. To configure fields.

Index contains time-based events

Use event times to create index names

# Kibana Installation - Kibana 4.1.2

The screenshot shows two parts of the Kibana setup process:

- Top Part:** A browser window displaying the Elasticsearch `_cat/indices?v` endpoint. The output is a table of indices:

health	status	index	pri	rep	docs.count	docs.deleted	store.size	pri.store.size
yellow	open	.kibana	1	1	10	1	50.1kb	50.1kb
yellow	open	uidai-auth-alpha-2015.10.29	5	1	791	0	2.4mb	2.4mb
yellow	open	uidai-auth-alpha-2015.10.28	5	1	6049	0	9.7mb	9.7mb

An orange arrow points from the URL bar to the table, and a callout box states: "All the list of indices can be obtained from the elastic search URL".

- Bottom Part:** The Kibana configuration interface at `localhost:5601/#/settings/indices/?_g=(filters:!((meta:(disabled:!f,index:'uidai-auth-alpha*',key:error_code_events.count,negate:!f,value:'123')))`.

  - The sidebar shows an index pattern: `uidai-auth-alpha*`.
  - The main area is titled "Configure an index pattern". It includes:
    - A note: "In order to use Kibana you must configure at least one index pattern. Index patterns are used to identify the Elasticsearch index to run search and analytics against."
    - Checkboxes:
      - Index contains time-based events
      - Use event times to create index names
    - A field labeled "Index name or pattern" containing the value `logstash-*`, which is highlighted with a yellow background.
    - A message at the bottom: "Unable to fetch mapping. Do you have indices matching the pattern?"

A red arrow points from the "Index name or pattern" field to the yellow highlight, and a callout box says: "Give the proper index name".

# Logstash and ElasticSearch Connection

To check on whether the input.conf is in coherent to the standard use the below command. The below command would parse the input.conf and show the parsing result. The below --configtest is just a configuration test.

```
D:\logstash154\bin>logstash.bat -f D:/input.conf --configtest
```

Start the log stash by using the below command.

```
D:\logstash154\bin>logstash.bat -f input.conf
```

The input.conf should be in the bin folder and would look like as shown below. The input is a file log and the output is fed to the elastic search.

```
input {  
    stdin {}  
    file { path => "D:/log001.txt" }  
}  
output {  
    stdout {}  
    elasticsearch_http { host => "localhost" index => "logstash-gpuz-%{+YYYY.MM.dd}" }  
}
```

# Logstash filter - hello world

A simple filter using grok

```
2015-10-14 11:06:51,205 ["http-bio-9000"-exec-2] INFO [authcommon.helpers.Auditor] - Time taken to save Hbase Audit Write is: 14 ms
2015-10-14 11:06:51,205 ["http-bio-9000"-exec-2] INFO [authcommon.helpers.NotificationConstructor] - ** No email or phone for notification purposes. Skipping notification for 2bd8e85e90da4a879372ef323ba443ce
2015-10-14 11:06:51,213 ["http-bio-9000"-exec-2] INFO [impl.messaging.RabbitMQMessagePublisherImpl] - Message of count index : 21 published into queue : genericBiNotificationQueue
2015-10-14 11:06:51,214 ["http-bio-9000"-exec-2] INFO [authcommon.helpers.Auditor] - Time taken to publish in BI notification queue:8 ms
2015-10-14 11:06:51,214 ["http-bio-9000"-exec-2] INFO [web.rest.AuthenticationResource] - Total Auth Response Time: 304 ms
```

```
filter {
  grok {
    match => { "message" => "%{TIMESTAMP_ISO8601:stamp} \[" %{GREEDYDATA:message}" }
    add_tag => "to_%{message}"
  }
}
```

# Logstash filter - csv

```
filter {
    csv {
        columns => ["auth_code","subreq_id","aua","sa","asa","ver"]
        separator => ","
    }
    mutate {
        convert => {"asa" => "integer"}
    }
}
```

# Logstash

grok() - grok converts unstructured data to structured data.

# Elasticsearch Delete

Advanced Rest Client [Unnamed] Save Open

Request  1

Socket Projects Saved History Settings About

Raw Form Headers 2

Raw Form Files (0) Payload 3

Encode payload Decode payload

application/x-www-form-urlencoded Set "Content-Type" header to overwrite this value.

Status **200 OK** Loading time: 262 ms

Request headers User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2490.71 Safari/537.36  
Origin: chrome-extension://hgmloofddfdnphgcellkdfbjeloo  
Content-Type: application/x-www-form-urlencoded  
Accept: \*/\*  
Accept-Encoding: gzip, deflate, sdch  
Accept-Language: en-US,en;q=0.8

Clear Send



# Performance Tuning in ELK Stack

Fine tune your elastic search  
memory components

In the  
`/elasticsearch/bin/elasticsearch.in.sh`  
file set the **ES\_MIN\_MEM** and  
**ES\_MAX\_MEM** values to an optimal  
level. **ES\_MAX\_MEM** can have a  
60% of your RAM.

```
#!/bin/sh

# check in case a user was using this mechanism
if [ "x$ES_CLASSPATH" != "x" ]; then
    cat >&2 << EOF
Error: Don't modify the classpath with ES_CLASSPATH. Best is to add
additional elements via the plugin mechanism, or if code must really be
added to the main classpath, add jars to lib/ (unsupported).
EOF
    exit 1
fi

ES_CLASSPATH="$ES_HOME/lib/elasticsearch-2.0.0.jar:$ES_HOME/lib/*"

if [ "x$ES_MIN_MEM" = "x" ]; then
    ES_MIN_MEM=5g
fi
if [ "x$ES_MAX_MEM" = "x" ]; then
    ES_MAX_MEM=70g
fi
if [ "x$ES_HEAP_SIZE" != "x" ]; then
    ES_MIN_MEM=$ES_HEAP_SIZE
    ES_MAX_MEM=$ES_HEAP_SIZE
fi

# min and max heap sizes should be set to the same value to avoid
# stop-the-world GC pauses during resize, and so that we can lock the
# heap in memory on startup to prevent any of it from being swapped
# out.
JAVA_OPTS="$JAVA_OPTS -Xms${ES_MIN_MEM}"
JAVA_OPTS="$JAVA_OPTS -Xmx${ES_MAX_MEM}"
```

# Performance Tuning in ELK Stack

Fine tune your elastic search  
memory components

Set the heap size of the elastic search to 60% of your memory. In order to set the heap size, set the value to the ES\_HEAP\_SIZE variable.

```
$ export ES_HEAP_SIZE=80g
```

# Performance Tuning in ELK Stack

When starting logstash  
specify the number of cores of  
your CPU's using the -w

```
$ logstash -w 24 -f input.conf
```

# X-Pack

X-Pack is an elastic extension that bundles security, alerting, monitoring, reporting and graph capabilities.

To install x-pack execute the below command:

```
$ bin/elasticsearch-plugin install x-pack
```



# Apache Solr

Apache Solr is the open source enterprise search platform built on top of Apache Lucene.

Latest Version - 5.4

# Apache Solr

Hibernate Search,  
Solr,  
Elastic Search all form the inverted search index on top of  
lucene library.

# Solr - Start the server.

To start the server, go to bin folder and give `$ solr start`

`$ solr_installation/bin/ solr start`

The screenshot shows a Windows command prompt window titled 'C:\Windows\system32\cmd.exe'. The window displays the output of a 'dir' command run from the directory 'C:\solr541\bin'. The output lists several files and directories, including 'install\_solr\_service.sh', 'oom\_solr.sh', 'post', 'solr', 'solr.cmd', 'solr.in.cmd', 'solr.in.sh', and 'solr.in.sh'. It also shows the total size of the files and free disk space. At the bottom of the window, the command '`>solr start`' is typed, followed by the server's response: 'Waiting up to 30 to see Solr running on port 8983' and 'Started Solr server on port 8983. Happy searching!'. The entire window is contained within a red rectangular border.

```
C:\solr541\bin>dir
Volume in drive C has no label.
Volume Serial Number is 14BC-5505

Directory of C:\solr541\bin

19-02-2016  11:11    <DIR>
19-02-2016  11:11    <DIR>
19-02-2016  11:11    <DIR>          init.d
19-02-2016  11:11            10,647 install_solr_service.sh
19-02-2016  11:11            1,255 oom_solr.sh
19-02-2016  11:11            7,754 post
19-02-2016  11:11            47,969 solr
19-02-2016  11:11            43,203 solr.cmd
19-02-2016  11:11            4,487 solr.in.cmd
19-02-2016  11:11            4,953 solr.in.sh
                           ? File(s)      120,268 bytes
                           3 Dir(s)   653,920,395,264 bytes free

C:\solr541\bin>>solr start
Waiting up to 30 to see Solr running on port 8983
Started Solr server on port 8983. Happy searching!
C:\solr541\bin>
```

## Solr - Start the server.

After you issue the command, solr start, solr by default starts in 8983 port.

Hit on <http://localhost:8983/solr/#/> to see the admin UI of the solr start.



Dashboard

Logging

Core Admin

Java Properties

Thread Dump

No cores available

Go and create one

### Instance

Start 6 minutes ago

### Versions

solr-spec	5.4.1
solr-impl	5.4.1 1725212 - jpountz - 2016-01-18 11:51:45
lucene-spec	5.4.1
lucene-impl	5.4.1 1725212 - jpountz - 2016-01-18 11:44:59

### JVM

Runtime	Oracle Corporation Java HotSpot(TM) 64-Bit Server VM (1.8.0_60 25.60-b23)
Processors	4
Args	<pre>-DSTOP.KEY=solrrocks -DSTOP.PORT=7983 -Djava.io.tmpdir=C:\solr541\server\tmp -Djetty.home=C:\solr541\server -Djetty.port=8983 -Dlog4j.configuration=file:C:\solr541\server\resources\log4j.properties -Dsolr.install.dir=C:\solr541 -Dsolr.solr.home=C:\solr541\server\solr -Duser.timezone=UTC -XX:+CMSParallelRemarkEnabled -XX:+CMSScavengeBeforeRemark -XX:+ParallelRefProcEnabled</pre>

### System

Physical Memory 72.2%

Swap Space 37.1%

### JVM-Memory

## Create a document collection

```
C:\solr541\bin>solr create -c uidai-demographic
```

Copying configuration to new core instance directory:

```
C:\solr541\server\solr\uidai-demographic
```

Creating new core 'uidai-demographic' using command:

```
http://localhost:8983/solr/admin/cores?action=CREATE&name=uidai-demographic&instanceDir=uidai-demographic
```

```
{
  "responseHeader":{
    "status":0,
    "QTime":2512},
  "core":"uidai-demographic"}
```

## Create a document collection

```
C:\solr541\bin>solr create -c uidai-demographic
```

Copying configuration to new core instance directory:

```
C:\solr541\server\solr\uidai-demographic
```

Creating new core 'uidai-demographic' using command:

```
http://localhost:8983/solr/admin/cores?action=CREATE&name=uidai-demographic&instanceDir=uidai-demographic
```

```
{
  "responseHeader":{
    "status":0,
    "QTime":2512},
  "core":"uidai-demographic"}
```

## Write document to Solr

```
HttpSolrClient client = new HttpSolrClient("http://localhost:8983/solr/uidai-demographic");
```

```
SolrInputDocument input = new SolrInputDocument();
input.addField( "uid", "764567345689");
input.addField( "address", "rnnpuram");
input.addField( "state", "tamil nadu" );
```

```
Collection<SolrInputDocument> docs = new ArrayList<SolrInputDocument>();
docs.add(input);
```

```
try {
    client.add(docs);
    client.commit();
    client.close();
} catch (SolrServerException | IOException e) {
    System.out.println(e.getMessage());
}
```

## Write document to Solr

```
import org.apache.solr.client.solrj.SolrQuery;  
import org.apache.solr.client.solrj.SolrServerException;  
import org.apache.solr.client.solrj.impl.HttpSolrClient;  
import org.apache.solr.client.solrj.response.QueryResponse;  
import org.apache.solr.common.SolrDocumentList;  
import org.apache.solr.common.SolrInputDocument;
```

## Write document to Solr

```
import org.apache.solr.client.solrj.SolrQuery;  
import org.apache.solr.client.solrj.SolrServerException;  
import org.apache.solr.client.solrj.impl.HttpSolrClient;  
import org.apache.solr.client.solrj.response.QueryResponse;  
import org.apache.solr.common.SolrDocumentList;  
import org.apache.solr.common.SolrInputDocument;
```

# Querying on to Solr

```
HttpSolrClient client = new HttpSolrClient("http://localhost:8983/solr/uidai-demographic");

SolrQuery query = new SolrQuery();
query.setQuery("uid=764567345689");

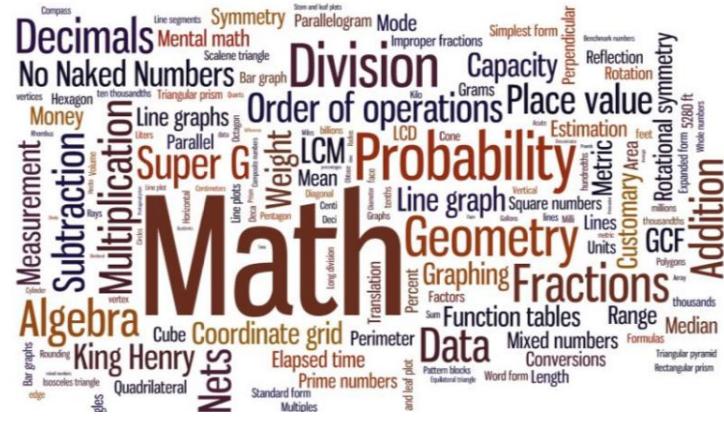
query.setFields("uid","address","state");
    query.setStart(0);
    query.set("defType", "edismax");
try {
    QueryResponse response = client.query(query);
    SolrDocumentList results = response.getResults();

    System.out.println(results.size());
    for (int i = 0; i < results.size(); ++i) {
        System.out.println(results.get(i).getFieldValue("state"));
    }

    client.close();
} catch (SolrServerException | IOException e) {
    e.printStackTrace();
}
```

# Inverted Index

Solr search works on the principle of inverted index data structure.



$$\begin{aligned} \cos x &= \sum_{k=0}^{\infty} C_{n+k} \frac{\sin^{n+k} x}{(n+k)!} \quad C_n = C_{n+k} \delta_{n+k} \\ \int \cos x dx &= \int dt = \arctg t - C \quad (t=\sin x) \\ \int \cos^2 x dx &= \int dt = \arctg t - C \quad (t=\sin x) \\ \int \cos x dx &= \int \frac{dt}{t} = \ln|t| + C \quad (t=\sin x) \\ \int \cos x dx &= \int \frac{dt}{t} = \ln|\sin x| + C \end{aligned}$$

# Machine Learning

$$\begin{aligned} y^2 &= \frac{(x+15)}{(x^2+138)} \quad a \cdot \operatorname{sh} x = \frac{dt}{1-t^2} \\ .84\% e = M \cdot l \cdot D \quad (-1 < t < 1) & \\ \cos \alpha = 45^\circ \quad \cos \beta = 84^\circ &= \sin \alpha + \beta \\ \sum_{k=0}^{\infty} (z-z_0)^n \quad C_n^0 \cdot C_{n+k}^{n+k} = 1 & \int dx = \frac{dt}{1-t^2}, \quad \frac{1}{\cos(u/2)} \\ \sum_{k=0}^{\infty} (t-t_0)^n \quad C_n^1 & \end{aligned}$$

# Supervised Learning

Supervised is by which the model is rendered by the data in hand.

# DBScan

DBScan for anomaly detection.

DBScan with gower's distance calculation performs much better in the anomaly detection.

# Natural Language Processing

## NLP

# Nltk.org

In the python prompt,

```
>> import nltk  
>> nltk.download()
```

Will start the downloader of the nltk data.

NLTK Downloader

Collections	Corpora	Models	All Packages	
Identifier		Name	Size	Status
all		All packages	n/a	partial
all-corpora		All the corpora	n/a	partial
book		Everything used in the NLTK Book	n/a	partial

Cancel Refresh

Server Index: [https://raw.githubusercontent.com/nltk/nltk\\_data/gh-pages/index.xml](https://raw.githubusercontent.com/nltk/nltk_data/gh-pages/index.xml)

Download Directory: /Users/dharshekthvel/nltk\_data

Downloading package u'unicode\_samples'





Gensim is a good NLP tool for finding similarity between phrases and documents

<https://radimrehurek.com/gensim/tut3.html>

# **GATE**

## General Architecture for Text Engineering

Names are a { bag of words } in GATE.

**To add a list of names, add the names to the following lst file.**

`/home/dharshekthvel/ac/code/ResumeParser-master/GATEFiles/plugins/ANNIE/resources/gazetteer/person_male.lst`

**To add a list of months, add to the following file**

`/home/dharshekthvel/ac/code/ResumeParser-master/GATEFiles/plugins/ANNIE/resources/gazetteer/months.lst`

**To add Organizations, add to the below place holder**

`/home/dharshekthvel/ac/code/ResumeParser-master/GATEFiles/plugins/ANNIE/resources/gazetteer/organization.lst`

# ANNIE

Tokenizer => Sentence Splitter => POS Tagger => Gazetteer => JAPE Transducer

Tokenizer: Splits the text into tokens such as numbers, punctuation, words.

Sentence Splitter: Segments the text into sentences. This phase is needed for POS tagging.

POS tagger: This phase produces a parts-of-speech tag as an annotation to each word.

Gazetteer: The Gazetteer list is the lookup of the entities. The list is stored in various files.

# JAPE - Java Annotation Patterns Engine

chinnasamyad@gmail.com



# JAPE - Java Annotation Patterns Engine

chinnasamyad@gmail.com

# JAPE - Java Annotation Patterns Engine

chinnasamyad@gmail.com

# JAPE - Java Annotation Patterns Engine

chinnasamyad@gmail.com

# Twitter



## Twitter Apps

Please [sign in](#) with your Twitter Account to create and maintain Twitter Apps.

Goto <https://apps.twitter.com>





# Create an application

## Application Details

**Name \***

Your application name. This is used to attribute the source of a tweet and in user-facing authorization screens. 32 characters max.

**Description \***

Your application description, which will be shown in user-facing authorization screens. Between 10 and 200 characters max.

**Website \***

Your application's publicly accessible home page, where users can go to download, make use of, or find out more information about your application. This fully-qualified URL is used in the source attribution for tweets created by your application and will be shown in user-facing authorization screens.  
(If you don't have a URL yet, just put a placeholder here but remember to change it later.)

**Callback URL**

Where should we return after successfully authenticating? OAuth 1.0a applications should explicitly specify their oauth\_callback URL on the request token step, regardless of the value given here. To restrict your application from using callbacks, leave this field blank.

## Developer Agreement

Yes, I have read and agree to the [Twitter Developer Agreement](#).

# Create an application

## Application Details

**Name \***

Your application name. This is used to attribute the source of a tweet and in user-facing authorization screens. 32 characters max.

**Description \***

Your application description, which will be shown in user-facing authorization screens. Between 10 and 200 characters max.

**Website \***

Your application's publicly accessible home page, where users can go to download, make use of, or find out more information about your application. This fully-qualified URL is used in the source attribution for tweets created by your application and will be shown in user-facing authorization screens.  
(If you don't have a URL yet, just put a placeholder here but remember to change it later.)

**Callback URL**

Where should we return after successfully authenticating? [OAuth 1.0a](#) applications should explicitly specify their oauth\_callback URL on the request token step, regardless of the value given here. To restrict your application from using callbacks, leave this field blank.

## Developer Agreement

Yes, I have read and agree to the [Twitter Developer Agreement](#).

# HThreeSixty

[Test OAuth](#)

Details

Settings

Keys and Access Tokens

Permissions



HThreeSixty\_Project

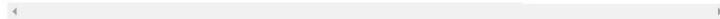
<http://yahoo.com>

## Organization

*Information about the organization or company associated with your application. This information is optional.*

Organization      None

Organization website      None



## Application Settings

*Your application's Consumer Key and Secret are used to [authenticate](#) requests to the Twitter Platform.*

Access level      Read and write ([modify app permissions](#))

Consumer Key (API Key)      H7gCcVkw7HI8C4hbgEo62h6zF ([manage keys and access tokens](#))

Callback URL      None

Callback URL Locked      No

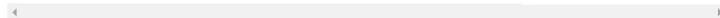
Sign in with Twitter      Yes

App-only authentication      <https://api.twitter.com/oauth2/token>

Request token URL      [https://api.twitter.com/oauth/request\\_token](https://api.twitter.com/oauth/request_token)

Authorize URL      <https://api.twitter.com/oauth/authorize>

Access token URL      [https://api.twitter.com/oauth/access\\_token](https://api.twitter.com/oauth/access_token)



## Application Settings

Keep the "Consumer Secret" a secret. This key should never be human-readable in your application.

Consumer Key (API Key) H7gCcVkw7Hl8C4hbgEo62h6zF

Consumer Secret (API Secret) yGgJ8lT5FKoNcrMWCE31tmQqtKzPiSnhiX9XuXlFP4P54XVdxg

Access Level Read and write ([modify app permissions](#))

Owner chinnasamydhara

Owner ID 837928192170209280

### Application Actions

[Regenerate Consumer Key and Secret](#)

[Change App Permissions](#)

## Your Access Token

This access token can be used to make API requests on your own account's behalf. Do not share your access token secret with anyone.

Access Token 837928192170209280-vOp7XGPdehsV0VSTouN6YisQReaEQGt

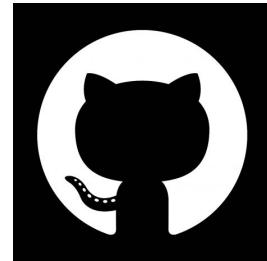
Access Token Secret 6HnqRgWUyQWPSUfKKpthSP7jXUBFfU7hVop6vlfJxUQpL

Access Level Read and write

Owner chinnasamydhara

Owner ID 837928192170209280

Create your access token and you should have  
your access tokens.



github

## Common git commands

```
$ git diff HEAD --name-only
```

```
$ git branch spark-scala-branch
```

```
$ git checkout spark-scala-branch
```

```
$ git commit -m "Comments" full/path/to/file/file.scala
```

```
$ git push -u origin spark-scala-branch
```

```
// To push directly to master
```

```
$ git push origin master
```



## Videos

<https://github.com/achinnasamy/videos>

1. *roadshow\_news.mp4*: The video contains news feeds of RSS parsed and written to elastic search and kibana is visualization of the elastic search to a data table. Tech: Kibana (Data Table), Elastic Search, Java, RSS Feed Parsing.
2. *roadshow\_twitter.mkv*: The video contains a Spark streaming hitting twitter and taking twitter feeds and ingesting to elastic search. Kibana visualization with various graphs with a Dashboard is performed on the elastic search index. Tech: Kibana, Spark Twitter Streaming, Elastic Search, Many visualization graphs on Kibana.
3. [HDFS\\_Installation\\_Configuring.mkv](#) The video contains, setting up of hdfs in a single node cluster. All issues that you encounter in a setup is shown in the video.





chinna samyad@gmail.com



# Add java to your classpath

[1] *Setting up PATH in environment, so that it is available every time when a PC starts:*

In linux, to add java to your path goto \$ /etc/environment and java bin folder to it.

```
PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/home/dharshekthvel/ac/bin/jdk8  
/bin"
```

[2] *Setting up JAVA\_HOME in profile:*

Also add the following entry to /etc/profile

```
export JAVA_HOME=/home/dharshekthvel/ac/bin/jdk8
```

```
export PATH=$JAVA_HOME/bin:$PATH
```

**To find the complete list of files containing the word ordinal, use the below command.**

```
$ grep -rnw '/home/dharshekthvel/ac/code' -e "ordinal"
```

**Check what all hadoop process is running**

```
$ ps -ef | grep hadoop
```

**Check what linux you are running**

```
$ uname -a
```

```
$ lsb_release -a
```

**To get the info of the complete hardware in MAC:**

```
$ system_profiler SPHardwareDataType
```

**To get the no of cores of the CPU:**

In linux:

```
$ lscpu
```

Or

```
$ sysctl --all | grep cpu
```

**In mac:**

```
$ sysctl -n hw.cpu  
$ sysctl -a | grep cpu
```

**To get the no of cores of the CPU:**

In linux:

```
$ lscpu
```

CPU(s): 8

On-line CPU(s) list: 0-7

Thread(s) per core: 2

Core(s) per socket: 4

Socket(s): 1

*The CPU depicts the number of threads.*

CPU(s): 8

Total number of threads is = CPU(s): which is 8

which is calculated using CPU(s): Thread(s) per core x Core(s) per socket x Socket(s) which is  $2 \times 4 \times 1 = 8$  which is nothing by CPU(s)

# Netcat utility

Producer and consumer using netcat utility.

```
$ nc 127.0.0.1 4567
```

The above command sends data to 4567.

```
$ nc -l 4567
```

The above command receives data from port 4567



# Setting up ambari server

1. Log in to your host as root.
2. wget -nv http://public-repo-1.hortonworks.com/ambari/ubuntu14/2.x/updates/2.4.2.0/ambari.list -O /etc/apt/sources.list.d/ambari.list
3. apt-key adv --recv-keys --keyserver keyserver.ubuntu.com B9733A7A07513CAD
4. apt-get update
5. Confirm whether the ambari server installation is present.  
    apt-cache showpkg ambari-server  
    apt-cache showpkg ambari-agent  
    apt-cache showpkg ambari-metrics-assembly
6. Install the ambari server  
    apt-get install ambari-server

# Setting up ambari server

7.     \$ ambari-server setup
8.     \$ ambari-server start

Install ambari agent on each of the hosts.

9.     \$ sudo apt-get install ambari-agent

For change in the config of ambari-agent, change the below config file.  
vi /etc/ambari-agent/conf/ambari-agent.ini

Then start the ambari agent:

\$ ambari-agent start

Big Data - Google 1. Log In to Apache Ambari stanford nlp find online proxy brows New Tab

localhost:8080/#/login

Ambari

Sign in

Username  
admin

Password  
\*\*\*\*\*

Sign in

Big Data - Google ▾ 1. Log In to Apache ▾ Ambari - Cluster Installs ▾ stanford nlp find ▾ online proxy browser ▾ New Tab

localhost:8080/#/installer/step0

Ambari

admin

CLUSTER INSTALL WIZARD

Get Started

Select Version

Install Options

Confirm Hosts

Choose Services

Assign Masters

Assign Slaves and Clients

Customize Services

Review

Install, Start and Test

Summary

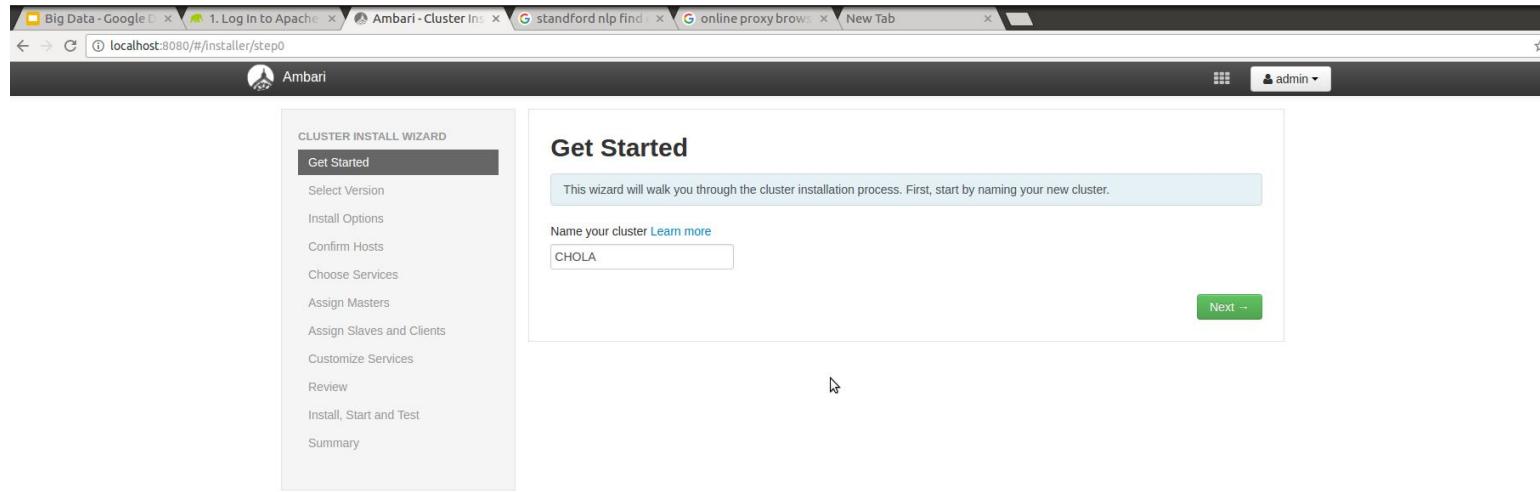
Get Started

This wizard will walk you through the cluster installation process. First, start by naming your new cluster.

Name your cluster [Learn more](#)

CHOLA

Next →



Licensed under the Apache License, Version 2.0.  
See third-party tools/resources that Ambari uses and their respective authors

Big Data - Google ... x 1. Log In to Apache ... x Ambari - Cluster In... x standford nlp find x Google online proxy brows... x New Tab

localhost:8080/#/installer/step2

Ambari

admin

CLUSTER INSTALL WIZARD

Get Started

Select Version

**Install Options**

Confirm Hosts

Choose Services

Assign Masters

Assign Slaves and Clients

Customize Services

Review

Install, Start and Test

Summary

## Install Options

Enter the list of hosts to be included in the cluster and provide your SSH key.

### Target Hosts

Enter a list of hosts using the Fully Qualified Domain Name (FQDN), one per line. Or use Pattern Expressions

host names

### Host Registration Information

Provide your [SSH Private Key](#) to automatically register hosts

[Choose file](#) No file chosen

ssh private key

SSH User Account

SSH Port Number

Perform [manual registration](#) on hosts and do not use SSH

[Back](#) [Register and Confirm](#)

Big Data - Google | 1. Log in to Apache | Ambari - Cluster In | stanford nlp find | online proxy brows | New Tab

localhost:8080/#/installer/step3

Ambari

admin

CLUSTER INSTALL WIZARD

- Get Started
- Select Version
- Install Options
- Confirm Hosts**
- Choose Services
- Assign Masters
- Assign Slaves and Clients
- Customize Services
- Review
- Install, Start and Test
- Summary

## Confirm Hosts

Registering your hosts.  
Please confirm the host list and remove any hosts that you do not want to include in the cluster.

Show:	All (1)	Installing (0)	Registering (1)	Success (0)	Fail (0)
Host	Progress	Status	Action		
localhost	<div style="width: 100%;"> </div>	Registering	<input type="button" value="Remove"/>		

Show: 25 1 - 1 of 1

Back Next



Licensed under the Apache License, Version 2.0.  
See third-party tools/resources that Ambari uses and their respective authors

Big Data - Google S × Hortonworks - Go × Inbox (1,440) - chin × Ambari - Cluster In × 2. Install the Ambari

localhost:8080#/installer/step3

Ambari admin

CLUSTER INSTALL WIZARD

- Get Started
- Select Version
- Install Options
- Confirm Hosts**
- Choose Services
- Assign Masters
- Assign Slaves and Clients
- Customize Services
- Review
- Install, Start and Test
- Summary

## Confirm Hosts

Registering your hosts.  
Please confirm the host list and remove any hosts that you do not want to include in the cluster.

Show: All (1)   Installing (0)   Registering (0)   Success (1)   Fail (0)				
<input type="checkbox"/>	Host	Progress	Status	Action
<input type="checkbox"/>	dhrshekhyvel	<div style="width: 100%;"> </div>	Success	<a href="#">Remove</a>

Show: 25 ▾ 1 - 1 of 1 ⏪ ⏴ ⏵ ⏩ ⏫

Please wait while the hosts are being checked for potential problems...

Back Next →

Licensed under the Apache License, Version 2.0.  
See third-party tools/resources that Ambari uses and their respective authors

Big Data - Google | hortonworks - Go | M Inbox (1,440) - chin | Ambari - Cluster Info | 2. Install the Ambari

localhost:8080/#/installer/step4

Ambari

admin

CLUSTER INSTALL WIZARD

Get Started

Select Version

Install Options

Confirm Hosts

**Choose Services**

Assign Masters

Assign Slaves and Clients

Customize Services

Review

Install, Start and Test

Summary

## Choose Services

Choose which services you want to install on your cluster.

Service	Version	Description
<input checked="" type="checkbox"/> HDFS	2.7.3	Apache Hadoop Distributed File System
<input checked="" type="checkbox"/> YARN + MapReduce2	2.7.3	Apache Hadoop NextGen MapReduce (YARN)
<input checked="" type="checkbox"/> Tez	0.7.0	Tez is the next generation Hadoop Query Processing framework written on top of YARN.
<input checked="" type="checkbox"/> Hive	1.2.1000	Data warehouse system for ad-hoc queries & analysis of large datasets and table & storage management service
<input checked="" type="checkbox"/> HBase	1.1.2	A Non-relational distributed database, plus Phoenix, a high performance SQL layer for low latency applications.
<input checked="" type="checkbox"/> Pig	0.16.0	Scripting platform for analyzing large datasets
<input checked="" type="checkbox"/> Sqoop	1.4.6	Tool for transferring bulk data between Apache Hadoop and structured data stores such as relational databases
<input checked="" type="checkbox"/> Oozie	4.2.0	System for workflow coordination and execution of Apache Hadoop jobs. This also includes the installation of the optional Oozie Web Console which relies on and will install the ExtJS Library.
<input checked="" type="checkbox"/> ZooKeeper	3.4.6	Centralized service which provides highly reliable distributed coordination
<input checked="" type="checkbox"/> Falcon	0.10.0	Data management and processing platform
<input checked="" type="checkbox"/> Storm	1.0.1	Apache Hadoop Stream processing framework
<input checked="" type="checkbox"/> Flume	1.5.2	A distributed service for collecting, aggregating, and moving large amounts of streaming data into HDFS
<input checked="" type="checkbox"/> Accumulo	1.7.0	Robust, scalable, high performance distributed key/value store.
<input checked="" type="checkbox"/> Ambari Infra	0.1.0	Core shared service used by Ambari managed components.
<input checked="" type="checkbox"/> Ambari Metrics	0.1.0	A system for metrics collection that provides storage and retrieval capability for metrics collected from the cluster
<input checked="" type="checkbox"/> Atlas	0.7.0	Atlas Metadata and Governance platform

Big Data - Google... × Hortonworks - Go... × Inbox (1,440) - chin... × Ambari - Cluster In... × 2. Install the Amb... ×

localhost:8080/#installer/step5

Ambari

admin

## CLUSTER INSTALL WIZARD

Get Started  
Select Version  
Install Options  
Confirm Hosts  
Choose Services  
**Assign Masters**  
Assign Slaves and Clients  
Customize Services  
Review  
Install, Start and Test  
Summary

## Assign Masters

Assign master components to hosts you want to run them on.  
★ HiveServer2 and WebHCat Server will be hosted on the same host.

SNameNode: dharshkthvel (15.5 GB, 8 cores)

NameNode: dharshkthvel (15.5 GB, 8 cores)

ResourceManager: dharshkthvel (15.5 GB, 8 cores)

App Timeline Server: dharshkthvel (15.5 GB, 8 cores)

History Server: dharshkthvel (15.5 GB, 8 cores)

Hive Metastore: dharshkthvel (15.5 GB, 8 cores)

WebHCat Server: dharshkthvel\*

HiveServer2: dharshkthvel (15.5 GB, 8 cores)

HBase Master: dharshkthvel (15.5 GB, 8 cores)

Oozie Server: dharshkthvel (15.5 GB, 8 cores)

ZooKeeper Server: dharshkthvel (15.5 GB, 8 cores)

Falcon Server: dharshkthvel (15.5 GB, 8 cores)

Nimbus: dharshkthvel (15.5 GB, 8 cores)

DRPC Server: dharshkthvel (15.5 GB, 8 cores)

Storm UI Server: dharshkthvel (15.5 GB, 8 cores)

Accumulo GC: dharshkthvel (15.5 GB, 8 cores)

dharshkthvel (15.5 GB, 8 cores)

SNameNode  
NameNode  
ResourceManager  
App Timeline Server  
History Server  
Hive Metastore  
WebHCat Server  
HiveServer2  
HBase Master  
Oozie Server  
ZooKeeper Server  
Falcon Server  
Nimbus  
DRPC Server  
Storm UI Server  
Accumulo GC  
Accumulo Monitor  
Accumulo Tracer  
Accumulo Master  
Infra Solr Instance  
Grafana  
Metrics Collector  
Atlas Metadata Server  
Kafka Broker  
Knox Gateway  
Log Search Server  
Activity Analyzer  
Activity Explorer  
HST Server  
Spark History Server  
Spark2 History Server  
Zeppelin Notebook

Big Data - Google ... ▾ hortonworks - Go... ▾ Inbox (1,440) - chin... ▾ Ambari - Cluster In... ▾ 2. Install the Ambari ... ▾

localhost:8080/#/installer/step6

Ambari

admin

CLUSTER INSTALL WIZARD

Get Started

Select Version

Install Options

Confirm Hosts

Choose Services

Assign Masters

**Assign Slaves and Clients**

Customize Services

Review

Install, Start and Test

Summary

## Assign Slaves and Clients

Assign slave and client components to hosts you want to run them on.  
Hosts that are assigned master components are shown with .  
"Client" will install HDFS Client, YARN Client, MapReduce2 Client, Tez Client, HCat Client, Hive Client, HBase Client, Pig Client, Sqoop Client, Oozie Client, ZooKeeper Client, Falcon Client, Accumulo Client, Infra Solr Client, Atlas Metadata Client, Spark Client, Spark2 Client, Mahout Client and Slider Client.

Host	all   none	all   none	all   none	all   none	all   none	all   none
dharshekthivel*	<input checked="" type="checkbox"/> DataNode	<input checked="" type="checkbox"/> NFSGateway	<input checked="" type="checkbox"/> NodeManager	<input checked="" type="checkbox"/> RegionServer	<input checked="" type="checkbox"/> Phoenix Query Server	<input checked="" type="checkbox"/> Supervisor

Show: 25 1 - 1 of 1

← Back Next →

Licensed under the Apache License, Version 2.0.  
See third-party tools/resources that Ambari uses and their respective authors

Big Data - Google ... ▾ hortonworks - Go... ▾ Inbox (1,440) - chin... ▾ Ambari - Cluster In... ▾ 2. Install the Ambari ... ▾

localhost:8080/#installer/step7

Ambari

admin

CLUSTER INSTALL WIZARD

- Get Started
- Select Version
- Install Options
- Confirm Hosts
- Choose Services
- Assign Masters
- Assign Slaves and Clients
- Customize Services**
- Review
- Install, Start and Test
- Summary

## Customize Services

We have come up with recommended configurations for the services you selected. Customize them as you see fit.

HDFS YARN MapReduce2 Tez Hive HBase Pig Sqoop Oozie 1 ZooKeeper Falcon Storm Flume Accumulo 2 Ambari Infra Ambari Metrics 1 Atlas Kafka Knox 1 Log Search 1 SmartSense 1

Spark Spark2 Zeppelin Notebook Mahout Slider Misc

Group Default (1) Manage Config Groups Filter...

Settings Advanced

**Hive Metastore**

Database Password  Database Password `javax.jdbc.option.ConnectionPassword`  
password to use against metastore database  
For security purposes, password changes will not be shown in configuration version comparisons

⚠ Attention: Some configurations need your attention before you can proceed.  
Showing properties with issues Show all properties

Back Next

Licensed under the Apache License, Version 2.0.  
See third-party tools/resources that Ambari uses and their respective authors

Big Data - Google ... × G hortonworks - Go... × M Inbox (1,440) - chin... × Ambari - Cluster Ins... × 2. Install the Ambari ...

localhost:8080/#/installer/step7

Assign Slaves and Clients

Customize Services

Review

Install, Start and Test

Summary

Warning

Derby is not recommended for production use. With Derby, Oozie Server HA and concurrent connection support will not be available.

Cancel Proceed Anyway

Password for user 'admin'

... ...

Back Next

Licensed under the Apache License, Version 2.0.  
See third-party tools/resources that Ambari uses and their respective authors

Big Data - Google S × Hortonworks - Goog × Inbox (1,440) - chinna × Ambari - Cluster In × 2. Install the Ambari

localhost:8080/#installer/step8

Ambari

admin

CLUSTER INSTALL WIZARD

- Get Started
- Select Version
- Install Options
- Confirm Hosts
- Choose Services
- Assign Masters
- Assign Slaves and Clients
- Customize Services
- Review**
- Install, Start and Test
- Summary

## Review

Please review the configuration before installation

Admin Name : admin

Cluster Name : chola

Total Hosts : 1 (1 new)

Repositories:

- debian7 (HDP-2.5):  
http://public-repo-1.hortonworks.com/HDP/debian7/2.x/updates/2.5.3.0
- debian7 (HDP-UTILS-1.1.0.21):  
http://public-repo-1.hortonworks.com/HDP-UTILS-1.1.0.21/repos/debian7
- redhat6 (HDP-2.5):  
http://public-repo-1.hortonworks.com/HDP/centos6/2.x/updates/2.5.3.0
- redhat6 (HDP-UTILS-1.1.0.21):  
http://public-repo-1.hortonworks.com/HDP-UTILS-1.1.0.21/repos/centos6
- redhat7 (HDP-2.5):  
http://public-repo-1.hortonworks.com/HDP/centos7/2.x/updates/2.5.3.0
- redhat7 (HDP-UTILS-1.1.0.21):  
http://public-repo-1.hortonworks.com/HDP-UTILS-1.1.0.21/repos/centos7
- suse11 (HDP-2.5):  
http://public-repo-1.hortonworks.com/HDP/suse11sp3/2.x/updates/2.5.3.0

← Back Print Deploy →

Licensed under the Apache License, Version 2.0.  
See third-party tools/resources that Ambari uses and their respective authors

Big Data - Google S hortonworks - Googl Inbox (1,440) - chin... Ambari - Cluster Ins 2. Install the Ambari

localhost:8080#/installer/step8

Ambari

admin

CLUSTER INSTALL WIZARD

- Get Started
- Select Version
- Install Options
- Confirm Hosts
- Choose Services
- Assign Masters
- Assign Slaves and Clients
- Customize Services
- Review**
- Install, Start and Test
- Summary

Preparing to Deploy: 3 of 96 tasks completed.

Admin Name : admin

Cluster Name : chola

Total Hosts : 1 (1 new)

Repositories:

- debian7 (HDP-2.5):  
http://public-repo-1.hortonworks.com/HDP/debian7/x/updates/2.5.3.0
- debian7 (HDP-UTILS-1.1.0.21):  
http://public-repo-1.hortonworks.com/HDP-UTILS-1.1.0.21/repos/debian7
- redhat6 (HDP-2.5):  
http://public-repo-1.hortonworks.com/HDP/centos6/x/updates/2.5.3.0
- redhat6 (HDP-UTILS-1.1.0.21):  
http://public-repo-1.hortonworks.com/HDP-UTILS-1.1.0.21/repos/centos6
- redhat7 (HDP-2.5):  
http://public-repo-1.hortonworks.com/HDP/centos7/x/updates/2.5.3.0
- redhat7 (HDP-UTILS-1.1.0.21):  
http://public-repo-1.hortonworks.com/HDP-UTILS-1.1.0.21/repos/centos7
- suse11 (HDP-2.5):  
http://public-repo-1.hortonworks.com/HDP/suse11sp3/x/updates/2.5.3.0

← Back      Print      Deploy ...

Licensed under the Apache License, Version 2.0.  
See third-party tools/resources that Ambari uses and their respective authors

Big Data - Google S... hortonworks - Go... Inbox (1,440) - chin... Ambari - Cluster Ins... 2. Install the Ambari ... localhost:8080/#installer/step9

Ambari admin

CLUSTER INSTALL WIZARD

- Get Started
- Select Version
- Install Options
- Confirm Hosts
- Choose Services
- Assign Masters
- Assign Slaves and Clients
- Customize Services
- Review
- Install, Start and Test**
- Summary

## Install, Start and Test

Please wait while the selected services are installed and started.

3 % overall

Host	Status	Message
dharshkithvel	3%	Waiting to install Accumulo Client

1 of 1 hosts showing - [Show All](#)

Show: 25 1 - 1 of 1

Next →

Licensed under the Apache License, Version 2.0.  
See third-party tools/resources that Ambari uses and their respective authors



# OpenCV

<http://opencv.org/>



# OpenCV

<http://opencv.org/>

# B Express



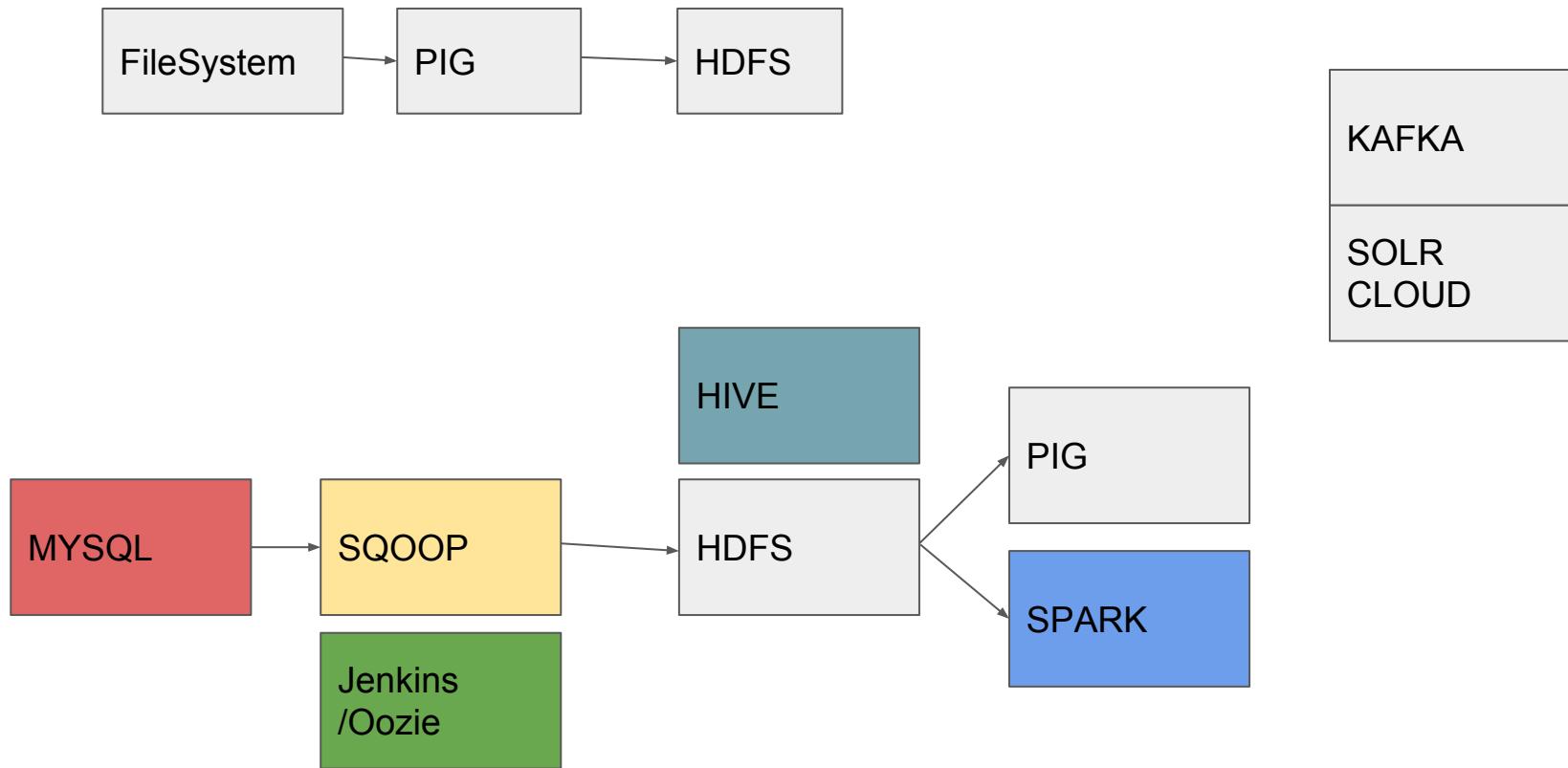
# Scope and work

- 1 Sentiment analysis of investments.
- 2 Sentiment analysis on new products.
- 3 Heat map analysis of net promoter score on new products.

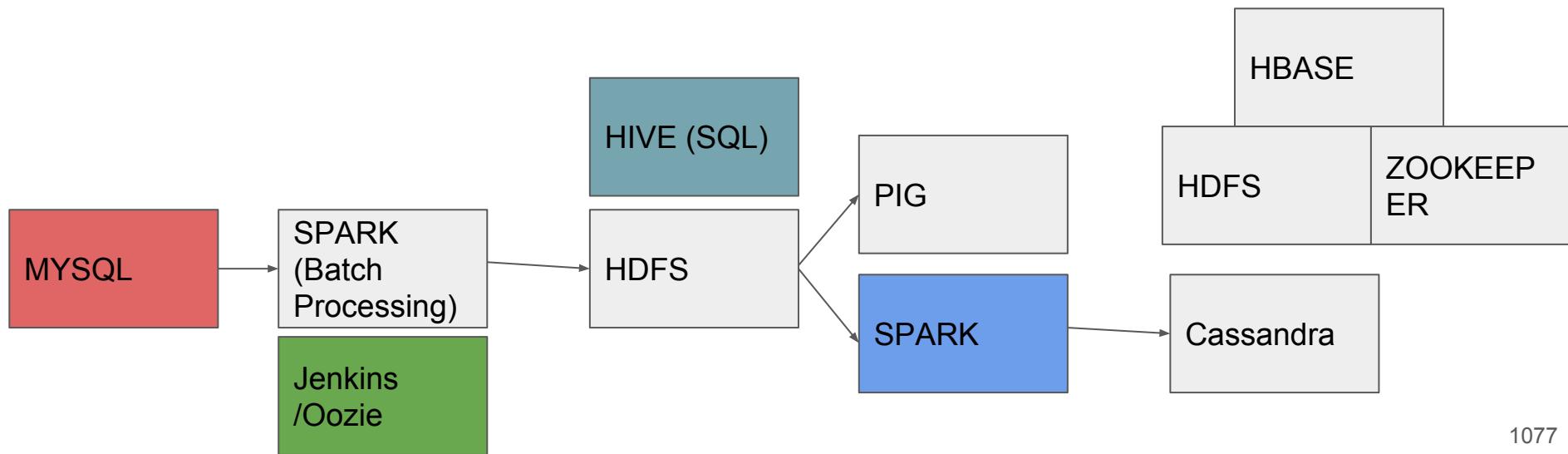
# Advantages to Bank

- 1 Quick analysis of Investments, reduces manual effort of analysing shares and commodities.
- 2 Assessment of new products success
- 3 Assessment of regional sentiments

## Big Data Architectures



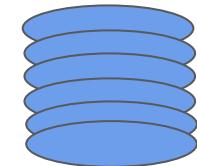
## Big Data Architectures



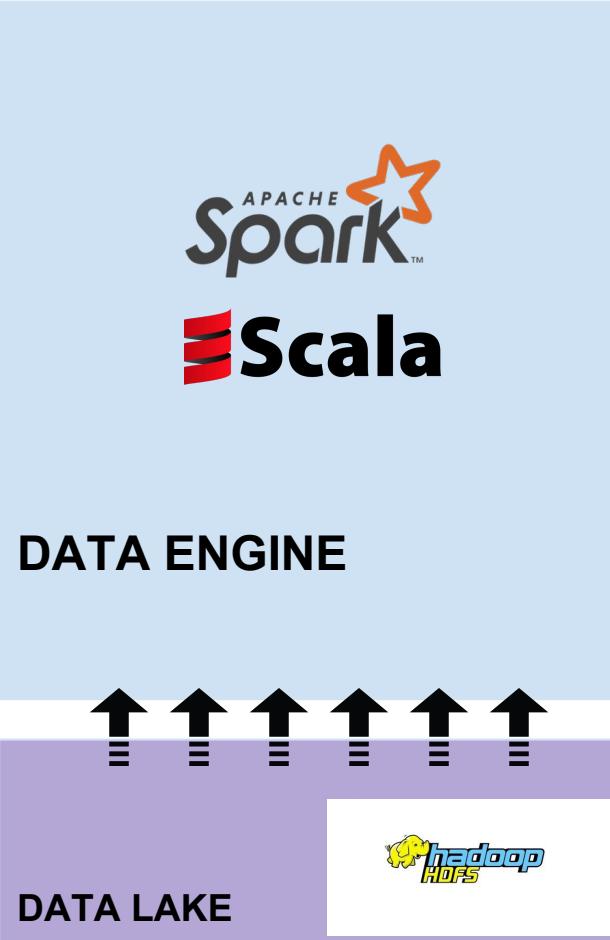
## Teradata

## Decommissioning

## Architecture



Teradata



Thank you

chinnasamyad@gmail.com