BITS Pilani
Pilani | Dubai | Goa | Hyderabad

DSECL ZG 522: Big Data Systems

Session 9: Hadoop Ecosystem – 2

Zookeeper, Sqoop, Flume, Oozie

Janardhanan PS

Professor

janardhanan.ps@wilp.bits-pilani.ac.in

Topics for today

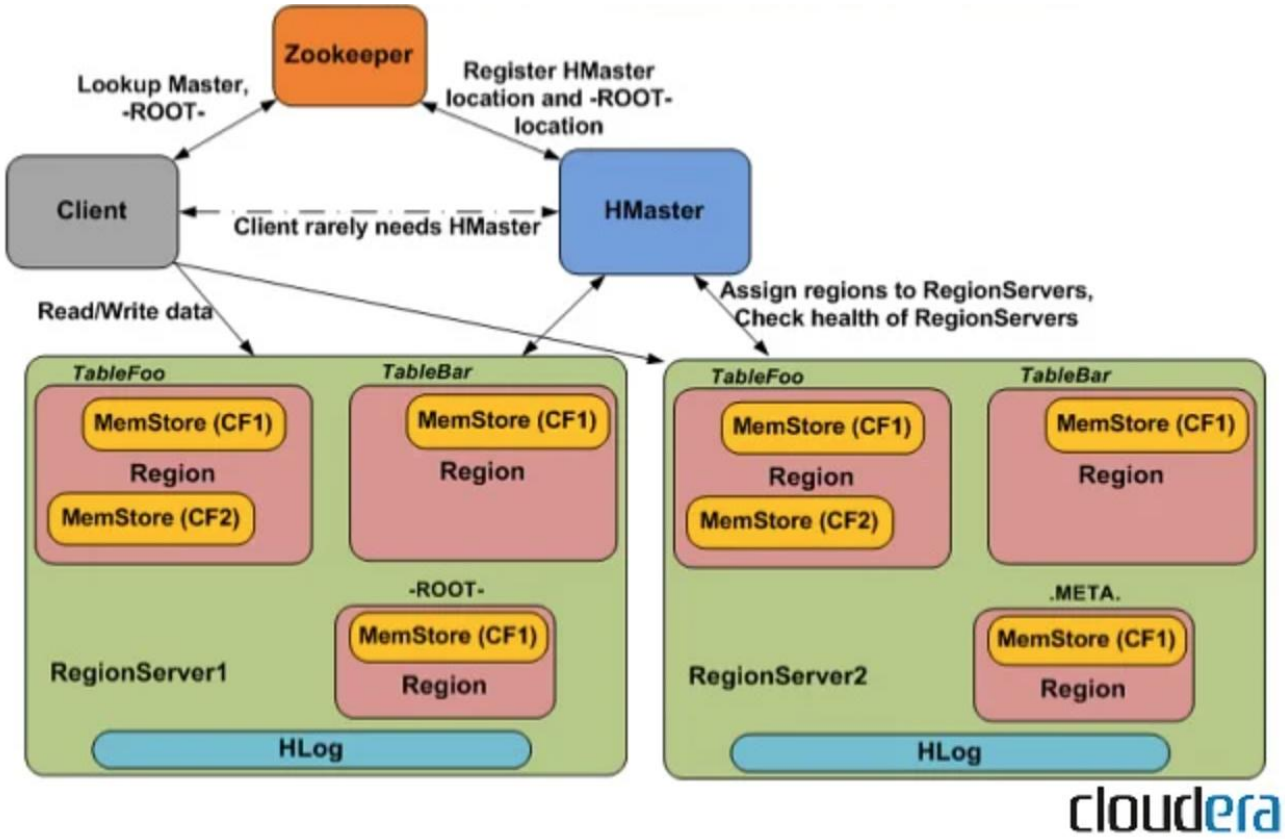
- Zookeeper - coordination and light meta-data storage in a distributed environment
- Sqoop - move data between RDBMS and Hadoop
- Flume - stream semi-structured data into Hadoop
- Oozie - scheduling and workflows

Topics for today

- **Zookeeper**
- Sqoop
- Flume
- Oozie

Zookeeper Introduction

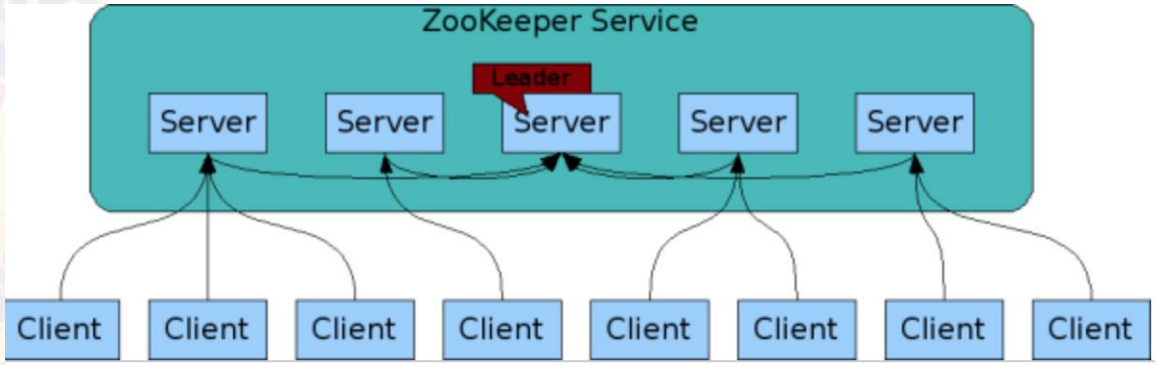
- Generic coordination service for distributed systems
- Lightweight store for meta-data, not a replacement of a distributed DB
- Data item updates seen across clients (connected to different nodes) are in same order - Total Ordering
  - ✓ Sequential consistency
- Data updates can be notified to clients
- Used for leader election - using unique node ID (refer to ring-based leader election algorithms)
- Example: Zookeeper for HBase cluster
  - ✓ Client needs to lookup current HMaster node
  - ✓ Track failures in cluster with heartbeat and pick a new HMaster
  - ✓ Maintain light weight cluster config data



https://www.elastic.co/blog/found-zookeeper-king-of-coordination

Zookeeper Architecture

- Built in-memory and as a Java application
  - ✓ Organized like an in-memory file system with dirs/files (called znodes)
- Zookeeper itself is clustered - Ensemble
- High performance, highly fault tolerant, strictly ordered access
  - ✓ It is a CP system as per CAP Theorem
  - ✓ Uses majority quorum within Ensemble for durability and avoid split-brain - so client gets no response if node is not in majority quorum
  - ✓ ZAB* protocol for consistency
- Clients connect on TCP session for request, response, heartbeats, events etc.
  - ✓ When a connection dies, client connects to another node or node detects client failure
- All client transactions are time stamped and ordered
- Typically meant for read-heavy workloads (10:1)

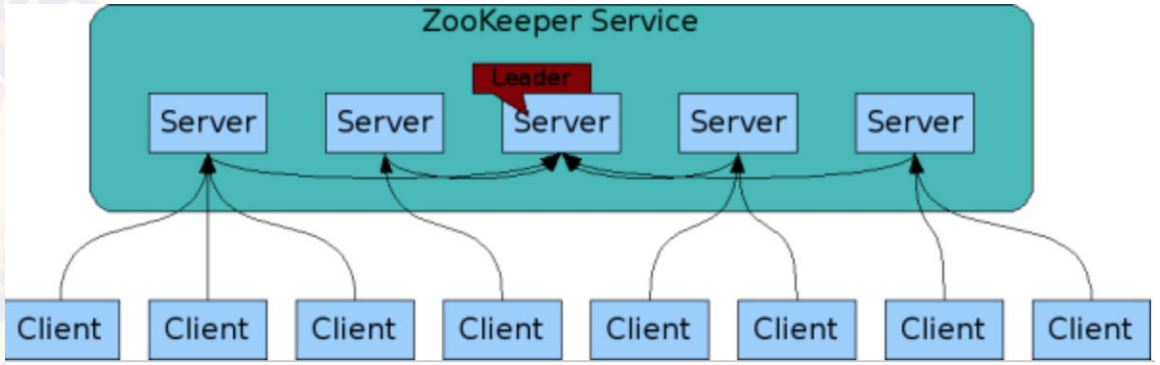


* Zookeeper Atomic Broadcast

https://zookeeper.apache.org/

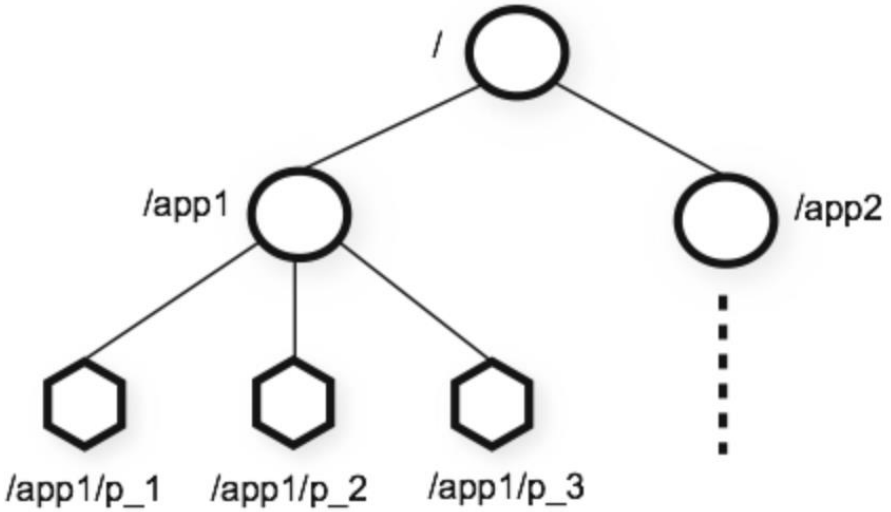
Reads / Writes

- Replicated in-memory DB
- Updates are logged on disk for recovery
- Reads are serviced from local copy, which ever server client connects to
- Memory copy is updated after writes are on disk
- Messaging layer makes sure all writes go to Leader first and on failure a new Leader is assigned
- All local replicas are kept in sync
- Server process - QuorumPeerMain



Data model

- Data is organised in a hierarchy with each node called a znode to hold some data
- Data can be status, config, location etc. (few KB)
- znode also contains a stat structure
  - ✓ version, ACL, timestamp
  - ✓ helps to order data and do access control
  - ✓ version updates on every write
- znodes can be ephemeral (only for a session), but these don't have children
- Clients can setup a watch / event for znode changes



Guarantees

- Sequential Consistency - Updates on a data item from a client will be applied in the order that they were sent and total order across all operations.
- Atomicity - Updates on a data item either succeed or fail. No partial results.
- Single System Image - A client will see the same view of the service regardless of the server that it connects to.
- Reliability - Once an update has been applied on a data item, it will persist from that time forward until a client overwrites the update.
- Timeliness - The clients view of the system is guaranteed to be up-to-date within a certain time bound.

Zookeeper command line interface

- echo stat | nc bdslab1 2181 | grep Mode – find out whether a node is Leader or Follower
- Sample 4lw commands
  - stat - list brief details for the server and connected clients
  - conf - serving configuration
  - cons - connection details for all clients
  - crst - reset connections
  - dump - lists the outstanding sessions and ephemeral nodes (Works on Leader Node)
  - envi - serving environment
  - ruok - I am OK or No response
  - srst - reset server statistics
  - srvr - lists full details of the server

Case study: Twitter

- Stores critical meta-data for coordination services, such as distributed locking, leader election etc.
- Stores a service registry for Twitter's service discovery
- Databases in Twitter (e.g. key-value store, image/video store), store their cluster topology
- Pub-sub messaging and compute platform uses it for leader election

https://zookeeper.apache.org/doc/current/zookeeperUseCases.html

Topics for today

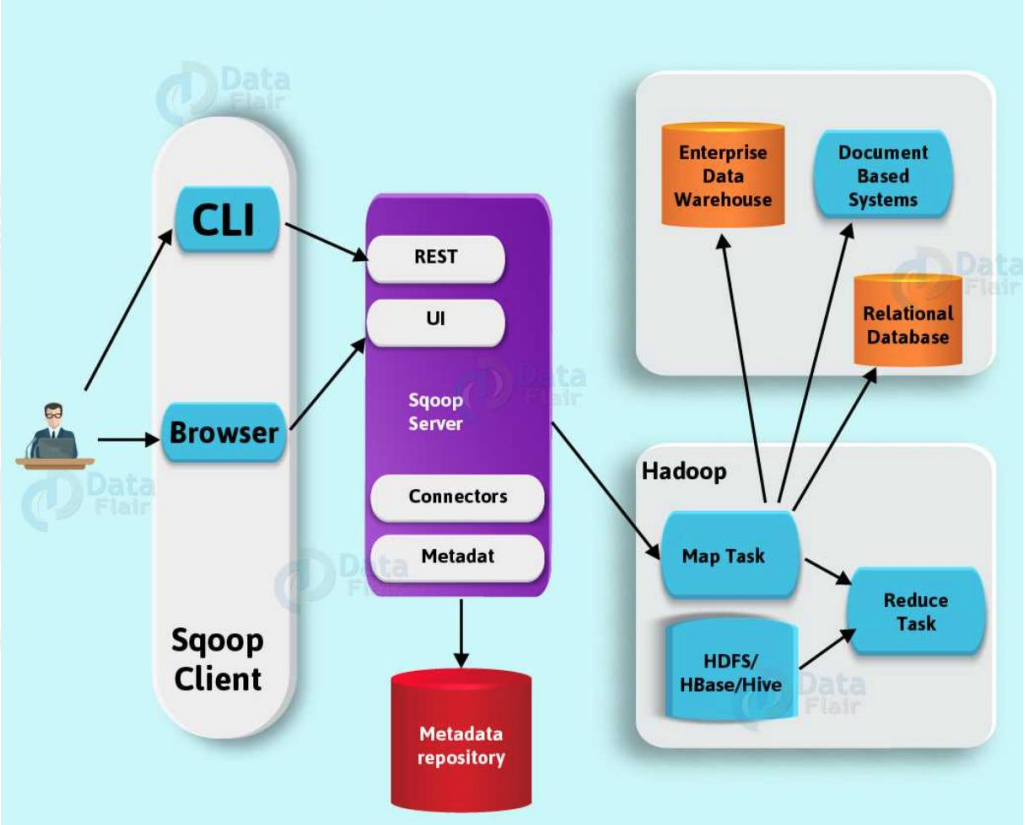
- Zookeeper
- **Sqoop**
- Flume
- Oozie

Sqoop Introduction

- "**Sq**l to Had**oop** & Hadoop to SQL"

- A tool in Hadoop ecosystem which is designed to transfer data between HDFS (Hadoop storage) and RDBMS
  - ✓ like MySQL, Oracle RDB, SQLite, Teradata, Netezza, Postgres etc.
- Efficiently transfers bulk data between Hadoop and external data stores such as enterprise data warehouses, relational databases, etc.

- Supports import and export operations
  - ✓ import data from relational DB such as MySQL, Oracle.
  - ✓ export data from HDFS to relational DB.



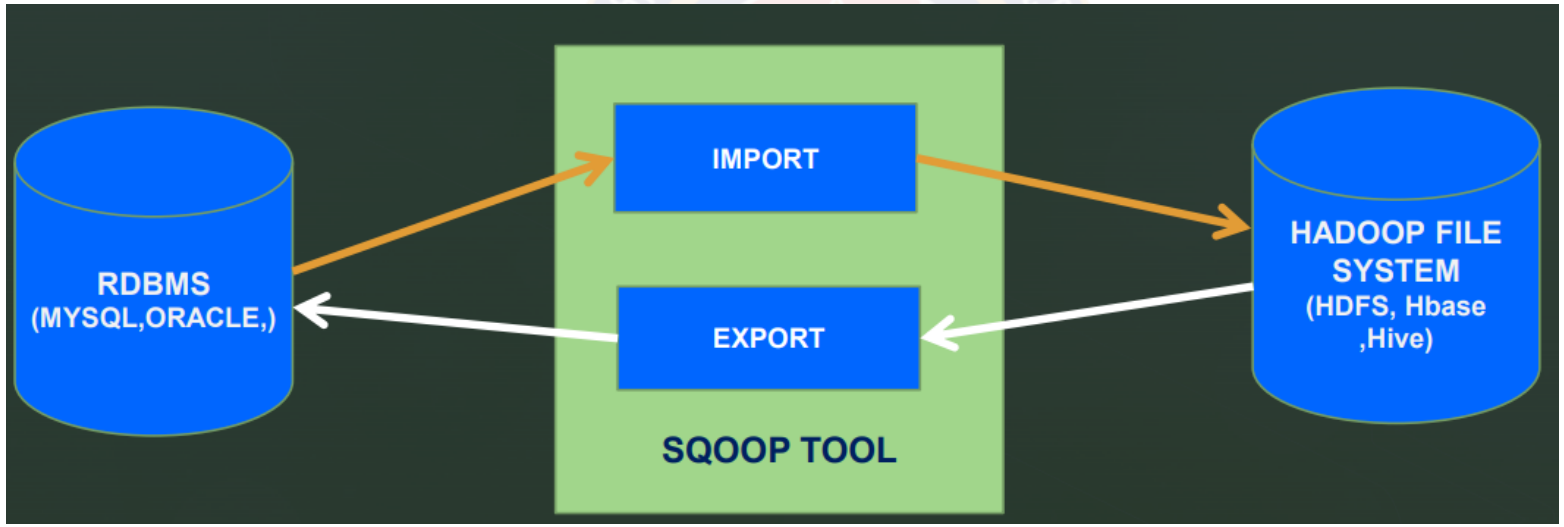
Why Sqoop?

- Big data developer's
  - ✓ work starts once the data is in Hadoop system like in HDFS, Hive or Hbase
  - ✓ Performs magical stuff to find all the golden information hidden on such a huge amount of data
  - ✓ Used to write to import and export data between Hadoop and RDBMS
- Hence a tool was needed !

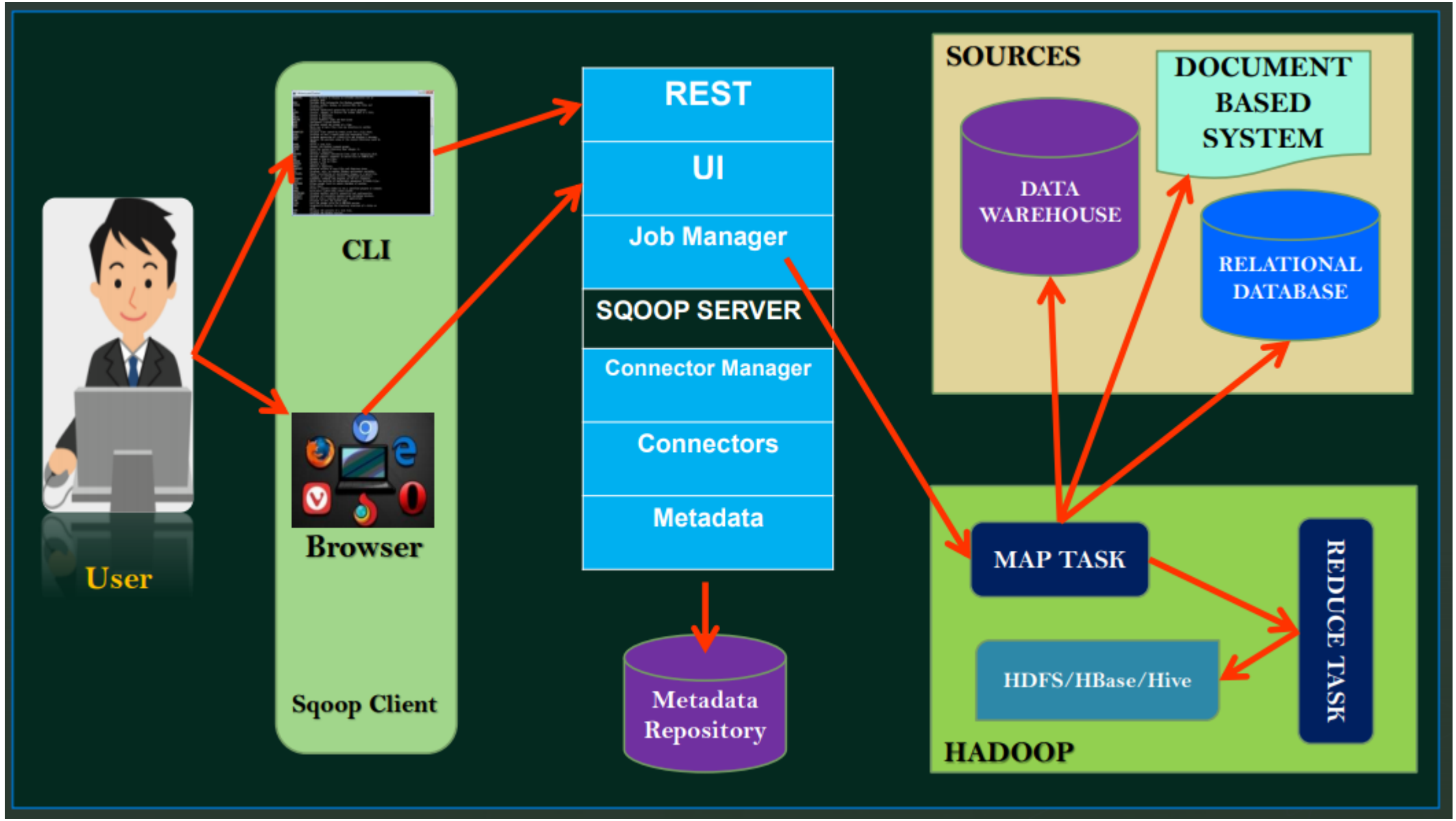
- Solution came in form of Sqoop
  - ✓ Sqoop uses the MapReduce mechanism for its operations like import and export work and work on a parallel mechanism as well as fault tolerance.
  - ✓ Developers just need to mention the source, destination and the rest of the work will be done by Sqoop
  - ✓ Filled the data transfer gap between relational databases and Hadoop system

Sqoop Architecture

- Source which is RDBMS like MySQL
- Destination like Hbase or HDFS or Hive
- Sqoop performs the operation for import and export



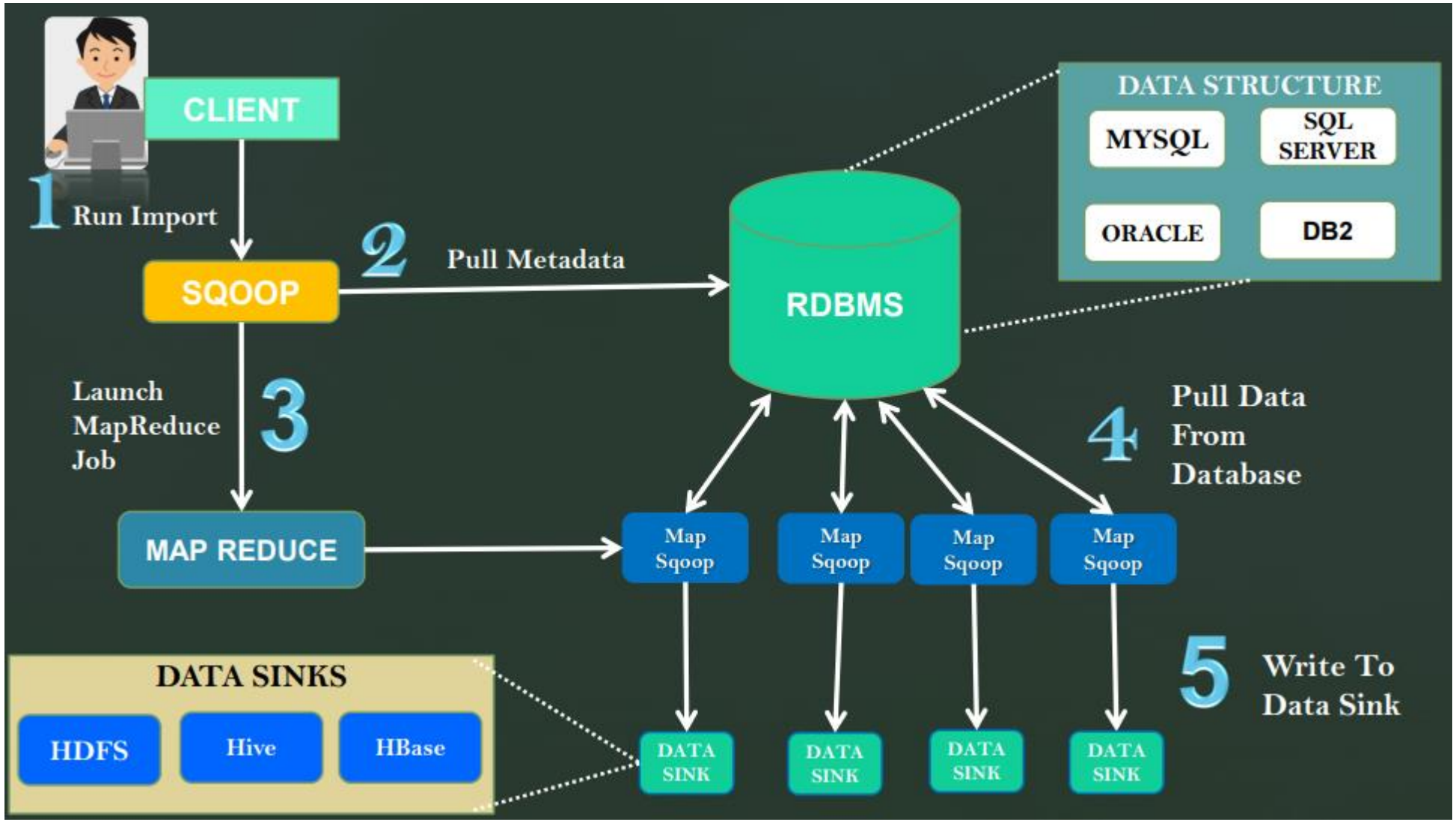
Sqoop 2 Architecture



Features of Sqoop

- Full Load
  - ✓ Can load the whole table by a single command
  - ✓ Can also load all the tables from a database using a single command
- Incremental Load
  - ✓ Also provides the facility of incremental load where you can load parts of table whenever it is updated
  - ✓ Supports two types of incremental imports: 1. Append 2. Last modified
- Parallel import/export
  - ✓ Uses YARN framework to import and export the data, which provides fault tolerance on top of parallelism
- Import results of SQL query
  - ✓ Can also import the result returned from an SQL query in HDFS
- Compression
  - ✓ Can compress data by using deflate(gzip) algorithm with –compress argument, or by specifying –compression-codec argument
  - ✓ Can also load compressed table in Apache Hive
- Connectors for all major RDBMS Databases
  - ✓ Provides connectors for multiple RDBMS databases, covering almost the entire circumference
- Load data directly into HIVE/HBase
  - ✓ Can load data directly into Apache Hive for analysis and also dump data in HBase, which is a NoSQL database

Five stage Sqoop Import Overview

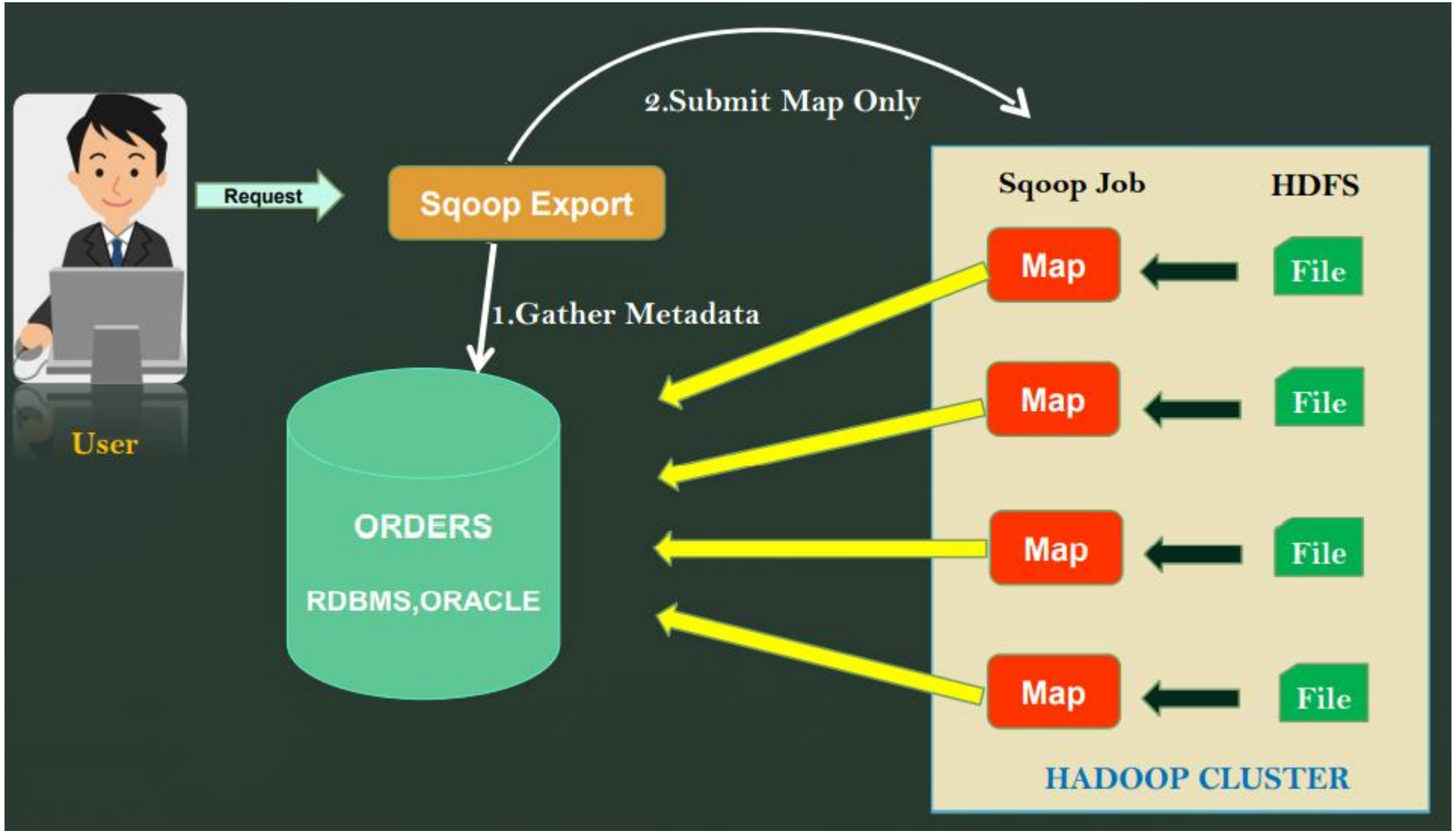


Importing Data using Sqoop

- SQOOP Import Command
  - ✓ Import individual tables from RDBMS to HDFS
  - ✓ Each row in a table is treated as records in HDFS
  - ✓ All record are stored as text data in text files or binary files

- Generic Syntax:
- Importing a Table into HDFS Syntax:
  - ✓ Takes JDBC url and connects to database
  - ✓ --table - Source table name to be imported
  - ✓ --username - Username to connect to database
  - ✓ --password - Password of the connecting user
  - ✓ --target-dir - Imports data to the specified directory

Exporting Data using Sqoop



Sqoop Export Data

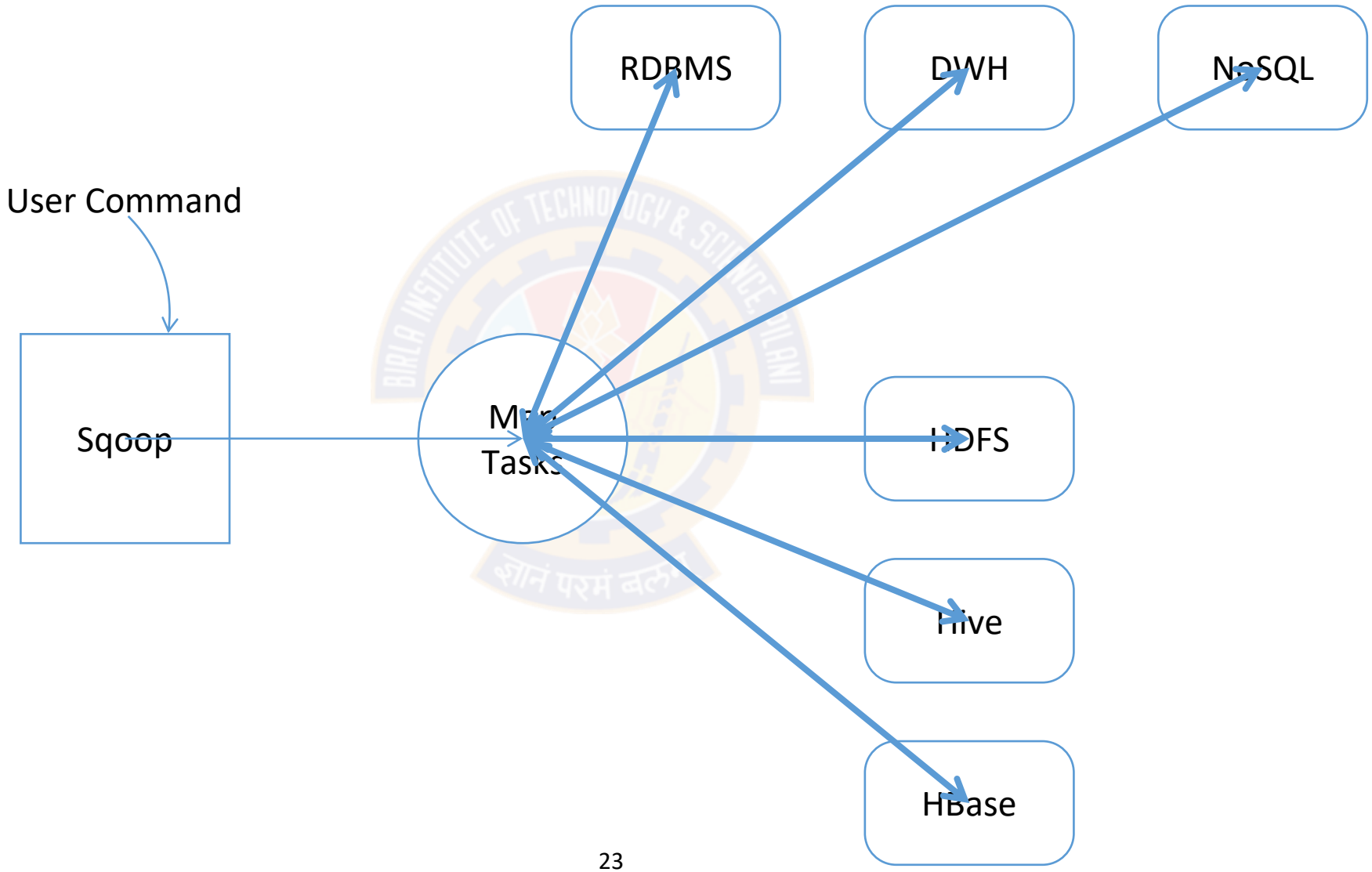
- SQOOP Export Command
  - ✓ Export a set of files from HDFS back to RDBMS
  - ✓ Files given an input to SQOOP contains records called as rows in table

- Generic Syntax:
  - ✓ Exporting a Table into RDBMS Syntax:
  - ✓ Takes JDBC url and connects to database
  - ✓ --table - Source table name to be exported
  - ✓ --username - Username to connect to database
  - ✓ --password - Password of the connecting user
  - ✓ --target-dir - Exports data from the specified directory

Limitations of Sqoop

- **Sqoop cannot be paused and resumed**
  - ✓an atomic step
  - ✓If failed, need to clear things up and start again
- Failures need special handling in case of partial import or export
- Sqoop Export performance also depends upon the hardware configuration (Memory, Hard disk) of RDBMS server
- For few databases Sqoop provides bulk connector which has faster performance
  - ✓Uses a JDBC connection to connect with RDBMS based on data stores, and this can be inefficient and less performance

Sqoop - Import / Export



Sqoop commands

```
$ sqoop import –connect jdbc:mysql://db.foo.com/corp –table EMPLOYEES \
–columns “employee_id,dept_id, first_name,last_name,job_title” -split-by dept_id
$ hadoop fs -ls EMPLOYEES
Found 2 items
-rw-r–r–   1 someuser somegrp    2913511 2021-04-27 16:40 /user/someuser/EMPLOYEES/part-m-00000
-rw-r–r–   1 someuser somegrp    1683938 2021-04-27 16:40 /user/someuser/EMPLOYEES/part-m-00001

$ sqoop export –connect jdbc:mysql://db.example.com/foo –table bar  –export-dir /results/bar_data
```

- Other commands : merge data, eval to run SQL etc.

Commands

https://docs.microsoft.com/en-us/azure/hdinsight/hadoop/hdinsight-use-sqoop
https://docs.microsoft.com/en-us/azure/hdinsight/hadoop/apache-hadoop-use-sqoop-mac-linux

```
create table mobiledata (clientid varchar(100), querytime varchar(100), market varchar(100), deviceplatform varchar(100),
devicemake varchar(100), devicemodel varchar(100), state varchar(100),
country varchar(100),
querydwelltime varchar(100),
sessionid               bigint,
sessionpagevieworder    bigint)
```

```
export serverConnect="jdbc:sqlserver://DBSERVER.database.windows.net:1433;user=USRNAME;password=PASSWORD"
export serverDbConnect="jdbc:sqlserver://DBSERVER.database.windows.net:1433;user=USRNAME;password=PASSWORD;database=DB_NAME"

sqoop list-databases --connect $serverConnect

sqoop list-tables --connect $serverDbConnect

sqoop export --connect $serverDbConnect \
-table mobiledata \
--hcatalog-table hivesampletable

sqoop eval --connect $serverDbConnect \
--query "SELECT COUNT(*) from dbo.mobiledata WITH (NOLOCK)"

sqoop eval --connect $serverDbConnect \
--query "SELECT TOP(10) * from dbo.mobiledata WITH (NOLOCK)"

sqoop import --connect $serverDbConnect \
--table mobiledata \
--target-dir 'wasb:///tutorials/usesqoop/importeddata' \
--fields-terminated-by '\t' \
--lines-terminated-by '\n' -m 1

hadoop fs -tail /tutorials/usesqoop/importeddata/part-m-00000

sqoop import --connect $serverDbConnect \
--table mobiledata \
--target-dir 'wasb:///tutorials/usesqoop/importeddata2' \
--fields-terminated-by '\t' \
--lines-terminated-by '\n' \
--create-hive-table \
--hive-table mobiledata_imported2 \
--hive-import -m 1

beeline -u 'jdbc:hive2://headnodehost:10001/;transportMode=http'

show tables;
describe mobiledata_imported2;
SELECT COUNT(*) FROM mobiledata_imported2;
SELECT * FROM mobiledata_imported2 LIMIT 10;
```

Topics for today

- Zookeeper
- Sqoop
- **Flume**
- Oozie

Flume Introduction (1)

- An e-retailer generates customer browse, buy logs on its website
- These logs need to be streamed into a Hadoop environment for analysis of customer behaviour by the data analytics team
- Flume
  - ✓ Moves logs from source application servers to HDFS files
  - ✓ Performs reliable buffering and message delivery for occasional data spikes
  - ✓ Scales with workload
  - ✓ Can be easily managed and customised
  - ✓ Parallel transfer support
  - ✓ Contextual routing

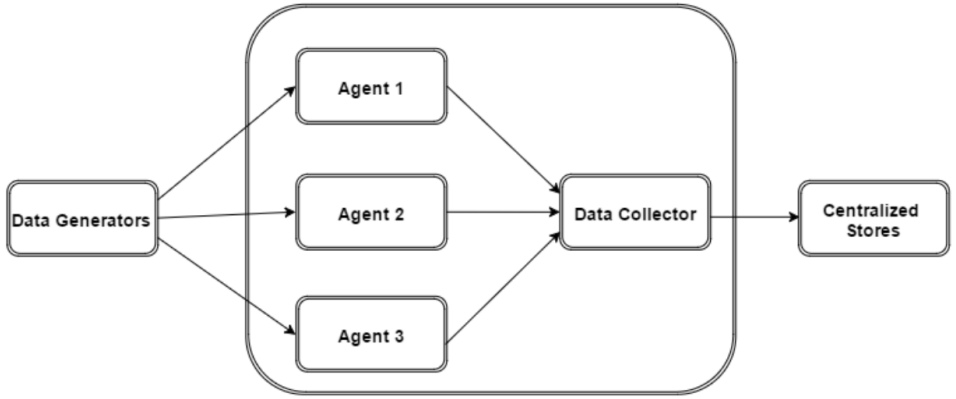
Flume Introduction (2)

- Why can't we simply do an HDFS 'put' ?
  - ✓ It is a one time command - we need to reliably stream data
  - ✓ If data is partially copied or if a failure happens in between the HDFS file is not closed and size is 0

- Solution: Apache Flume

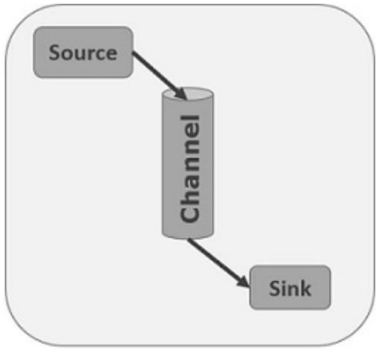
Architecture

- Agents move data from input to output - possible to other agents
  - ✓ Contain source, channel, sink
  - ✓ JVM process
  - ✓ Source: Receives data from generator or another agent, specific types (e.g. Avro source)
  - ✓ Sink: Consumes from channel and stores in Central store (e.g. HDFS, Hive, HBase sinks) or sends to another agent
  - ✓ Channel: "Transactional" transient data store to act as a bridge between "1 or more" sources and sinks (e.g. JDBC, memory, filesystem channels)
- Flume is event driven
  - ✓ Event: Basic unit of data transfer
- Reliable messaging: For each event, 2 transactions happen. One at sender and second at receiver. Sender commits after receiver commits.



Flume

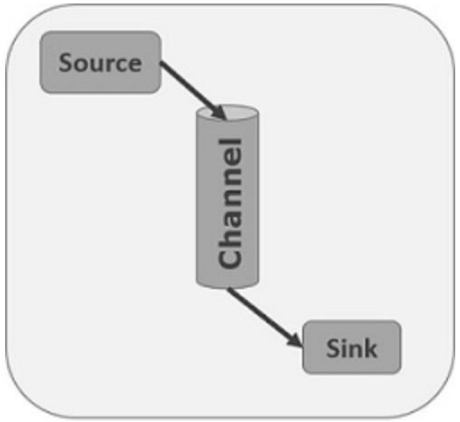
Flume event



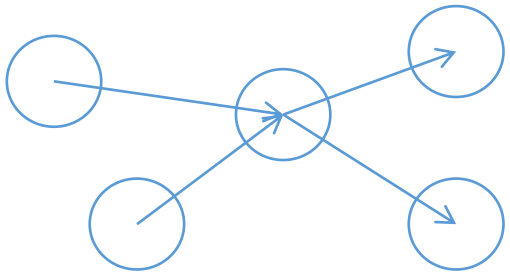
Flume Agent

Other components

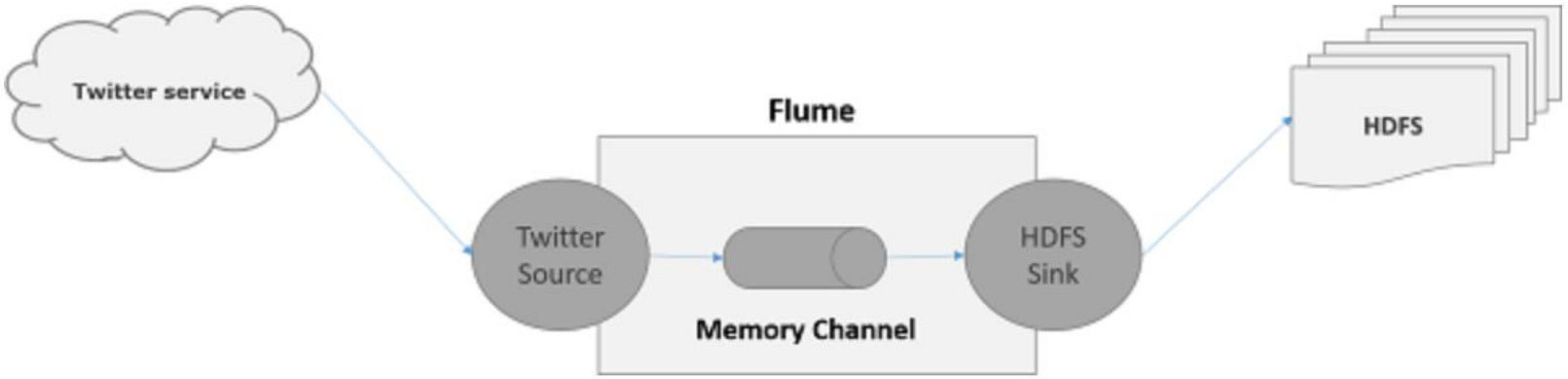
- Interceptors
  - Inspect / Alter events between source and channel
- Channel selectors
  - Select which channel to send the event on
    - Replicating default channel selector: Replicate event to all channels
    - Multiplexing selector : Select based on address in header
- Sink Processors
  - Select a sink in a group
  - Used for failover, load balance across multiple sinks in a channel
- Using these components Flume data flows can have various topologies
  - Multi-hop, Fan-in, Fan-out



Flume Agent



Example



- Create a Twitter API based app for data source
- Install a HDFS sink
- Setup Flume and configure it with the app as a source and HDFS as the sink

Sources , Channels and Sinks

SOURCES	CHANNELS	SINKS
Avro Source	Memory Channel	HDFS Sink
Thrift Source	JDBC Channel	Hive Sink
Exec Source	Kafka Channel	Logger Sink
JMS Source	File Channel	Avro Sink
Spooling Directory Source	Spillable Memory Channel	Thrift Sink
Twitter 1% firehose Source	Pseudo Transaction Channel	IRC Sink
Kafka Source	FILE channel	File Roll Sink
NetCat Source		Null Sink
Sequence Generator Source		HBaseSink
Syslog Sources		AsyncHBaseSink
Syslog TCP Source		MorphlineSolrSink
Multiport Syslog TCP Source		ElasticSearchSink
Syslog UDP Source		
HTTP Source		
Stress Source		
Legacy Sources		
Thrift Legacy Source		
Custom Source		
Scribe Source		

Example

To create a Twitter API based app for data source.

To Setup Flume and configure an agent to stream data from Twitter to HDFS

Steps:

1. Naming the components on the current agent.
    - ✓ TwitterAgent.sources = Twitter
    - ✓ TwitterAgent.channels = MemChannel
    - ✓ TwitterAgent.sinks = HDFS

2. Describing/Configuring the source

3. Describing/Configuring the sink

4. Describing/Configuring the channel

5. Binding the source and sink to the channel
    - ✓ TwitterAgent.sources.Twitter.channels = MemChannel
    - ✓ TwitterAgent.sinks.HDFS.channel = MemChannel

Simple test

- Install Flume
- Create a config for an agent
- Run flume
  - sudo flume-ng agent -c . -f flume.conf -n a1 -Dflume.root.logger=INFO,console
- Create logs at source
  - sudo sh -c 'echo "hello flume" > /opt/logs/flume/test.log'
- View output at the logger sink

```
# Specify the component name of the Agent
a1.sources = r1
a1.sinks = k1
a1.channels = c1

 # Specify Flume source (path to be monitored)
a1.sources.r1.type = spooldir
a1.sources.r1.spoolDir = /opt/logs/flume
 # Specify Flume sink
a1.sinks.k1.type = logger

 # Specify Flume channel
a1.channels.c1.type = memory
a1.channels.c1.capacity = 1000
a1.channels.c1.transactionCapacity = 100

 # Bind source and sink to channel
a1.sources.r1.channels = c1
a1.sinks.k1.channel = c1
```

https://www.programmersought.com/article/99104811160/

Sqoop vs Flume

	Sqoop	Flume
Data Flow	Various RDBMS, NoSQL, can map to Hive/HBase from RDBMS	Streaming data, e.g. logs
Loading type	Not event driven	Event driven
Source integration	Connector driven	Agent driven
Usage	Structured sources	Streaming systems with semi-structured data, e.g. twitter feed, web server logs
Performance and reliability	Parallel transfer with high utilisation	Reliable transfer with aggregations at low latency

Kafka vs Flume

	Kafka	Flume
Base function	Ingestion and processing of streaming data in real-time	Ingestion of streaming log data into a data store
Scale	More scalable	Less scalable
Data movement	Pull based (pub-sub system)	Push based
Recovery	Resilient to failures, can auto-recover	Events may be lost on agent failure
Flexibility	General purpose streaming systems	Tuned for Hadoop ecosystem

* Flume can have a ‘Kafka Channel’ which stores events in a Kafka cluster

Topics for today

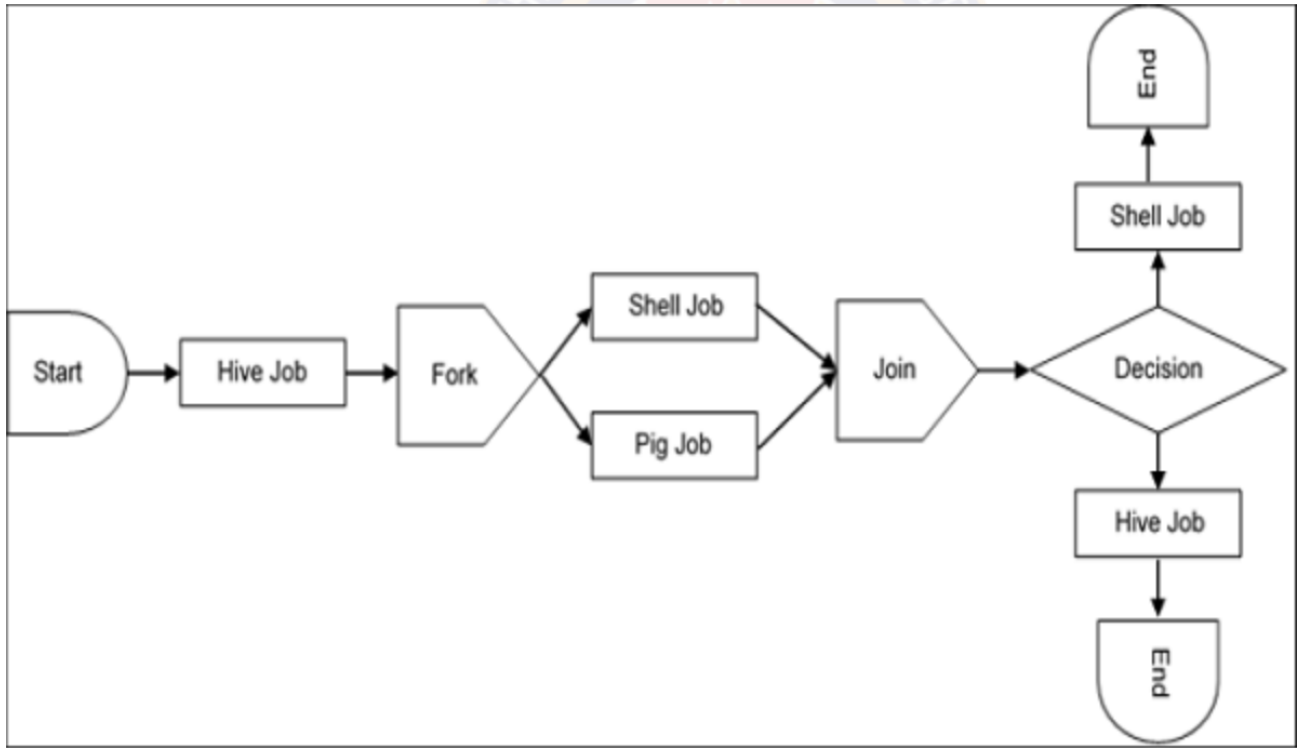
- Sqoop
- Flume
- Zookeeper
- **Oozie**

Introduction

- Scheduler for Hadoop jobs
- Sequence multiple jobs using a specification input
- Support for Hive, PIG, Sqoop, or Java and shell scripts
- Built in Java as a web-application
- Uses Hadoop execution engine to execute tasks
- So load-balancing, failover is done by Hadoop
- Tasks have a callback URL to tell Oozie when done or can poll the task status

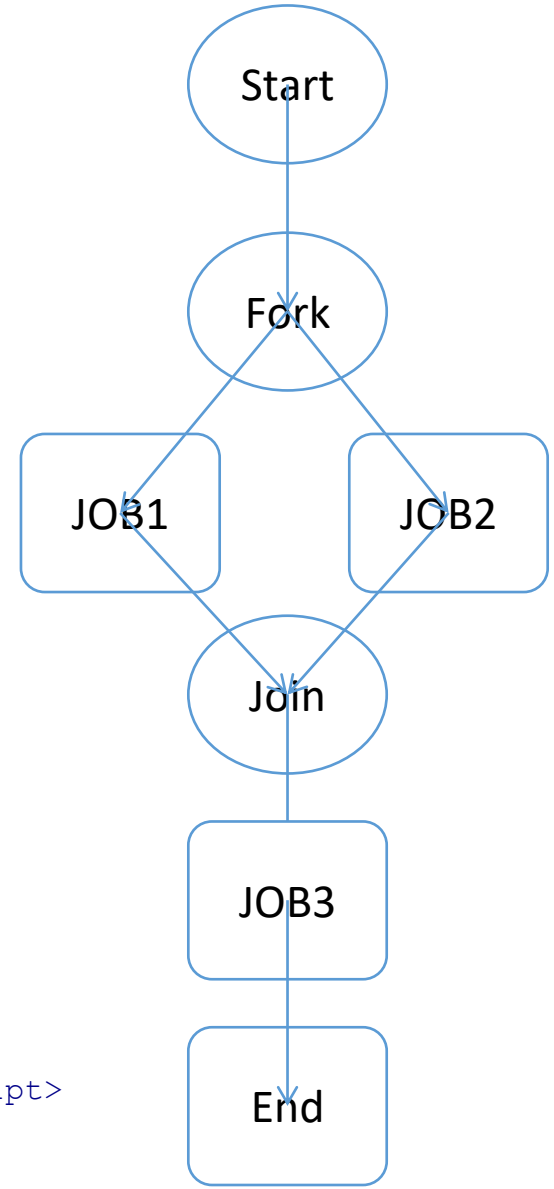
Types of jobs

- Workflow jobs: In a DAG form to specify sequence of actions
- Coordinator jobs: Triggered by time and data availability
- Bundle: Package of workflow and coordinator jobs



Workflow (1)

```
<workflow-app name = "simple-Workflow">
   <start to = "fork_node" />

   <fork name = "fork_node">
      <path start = "Job1"/>
      <path start = "Job2"/>
   </fork>

   <action name = "Job1">
      <parameters for running Job1 with arguments, settings, executable path>
      <ok to = "join_node" />
      <error to = "kill_job" />
   </action>

   <action name = "Job2">
      <parameters for running Job1 with arguments, settings, executable path>
      <ok to = "join_node" />
      <error to = "kill_job" />
   </action>

   <join name = "join_node" to = "Job3"/>

   <action name = "Job3">
      <parameters for running Job1 with arguments, settings, executable path>
     <ok to = "end" />
      <error to = "kill_job" />
   </action>

   <kill name = "kill_job">
      <message>Job failed</message>
   </kill>

   <end name = "end" />

</workflow-app>
```

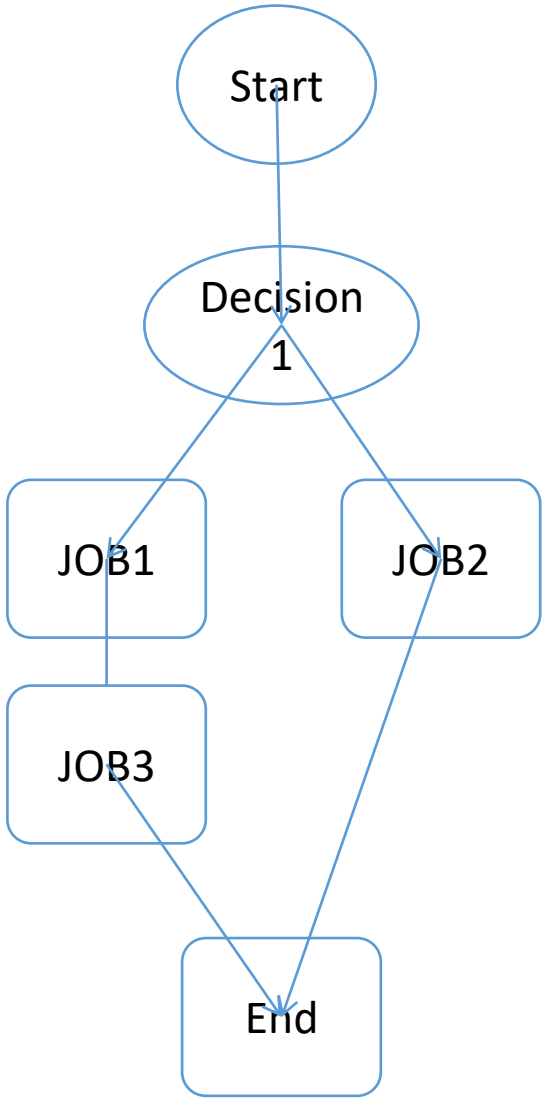


Example job spec :

```
<hive xmlns = "uri:oozie:hive-action:0.4">
 <job-tracker>xyz.com:8088</job-tracker>
 <name-node>hdfs://rootname</name-node>
 <script>hdfs_path_of_script/Copydata.hive</script>
 <param>database_name</param>
</hive>
```

Workflow (2)

```
<workflow-app xmlns = "uri:oozie:workflow:0.4" name = "simple-Workflow">
   <start to = "Decision1" />

   <decision name = "Decision1">
      <switch>
         <case to = "Job1">${fs:exists('/test/abc') eq 'false'}
            </case>
         <default to = "Job2" />
      </switch>
   </decision>

   <action name = "Job1">
      <action details>
      <ok to = "Job3" />
      <error to = "kill_job" />
   </action>

   <action name = "Job2">
      <action details>
      <ok to = "end" />
      <error to = "kill_job" />
   </action>

   <action name = "Job3">
      <action details>
      <ok to = "end" />
      <error to = "kill_job" />
   </action>

   <kill name = "kill_job">
      <message>Job failed</message>
   </kill>

   <end name = "end" />

</workflow-app>
```



Coordinator

Base workflow

```
<workflow-app name = "simple-Workflow">
   <start to = "Job1" />

   <action name = "Job1">
      <ok to = "end" />
      <error to = "kill_job" />
   </action>

   <kill name = "kill_job">
      <message>Job failed</message>
   </kill>
   <end name = "end" />

</workflow-app>
```

Coordinator to run the workflow

```
<coordinator-app name =
   "simple coordinator" frequency = "5 * * * *" start =
   "2021-00-18T01:00Z" end = "2025-12-31T00:00Z"" timezone
= "India">

   <controls>
      <timeout>1</timeout>
      <concurrency>1</concurrency>
      <execution>FIFO</execution>
      <throttle>1</throttle>
   </controls>

   <action>
      <workflow>
         <app-path>path to base workflow xml</app-path>
      </workflow>
   </action>

</coordinator-app>
```

Exercise

- Create 2 PIG jobs using geosales.csv data and run through Oozie. Try workflow and use a coordinator to run the workflow periodically.

Summary

- Zookeeper: Building distributed systems with coordination, light weight meta-data storage
- Sqoop: Moving structured data into and out of Hadoop
- Flume: Moving semi-structured data into Hadoop
- Oozie: Building workflows and scheduling of tasks written on various ecosystem technologies

Next Session:
NoSQL Databases