

<http://arnaud-nauwynck.github.io>

# Big Data – Part 3

## Hadoop Ecosystem

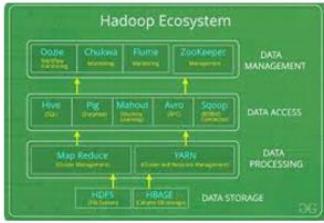
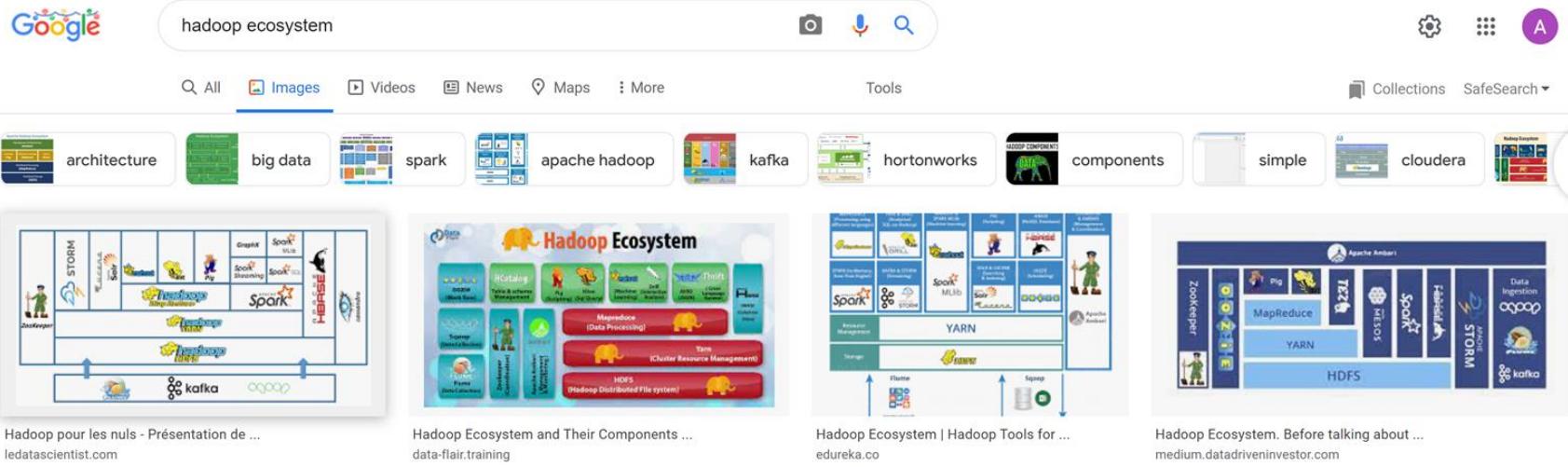
### Main Components Explained

[arnaud.nauwynck@gmail.com](mailto:arnaud.nauwynck@gmail.com)

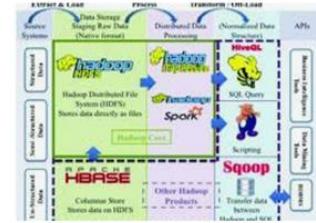
# Outline

- ZooKeeper
- Hdfs
- Yarn
- Oozie
- Next part 5
  - HiveMetaStore, Parquet, Spark

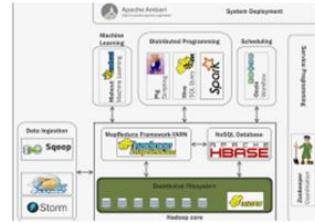
# Google image "Hadoop ecosystem"



Hadoop Ecosystem - GeeksforGeeks  
[geeksforgeeks.org](http://geeksforgeeks.org)



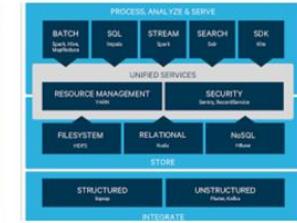
Apache Hadoop Ecosystem | Download ...  
[researchgate.net](http://researchgate.net)



The Hadoop ecosystem | Hadoop Essentials  
[subscription.packtpub.com](http://subscription.packtpub.com)



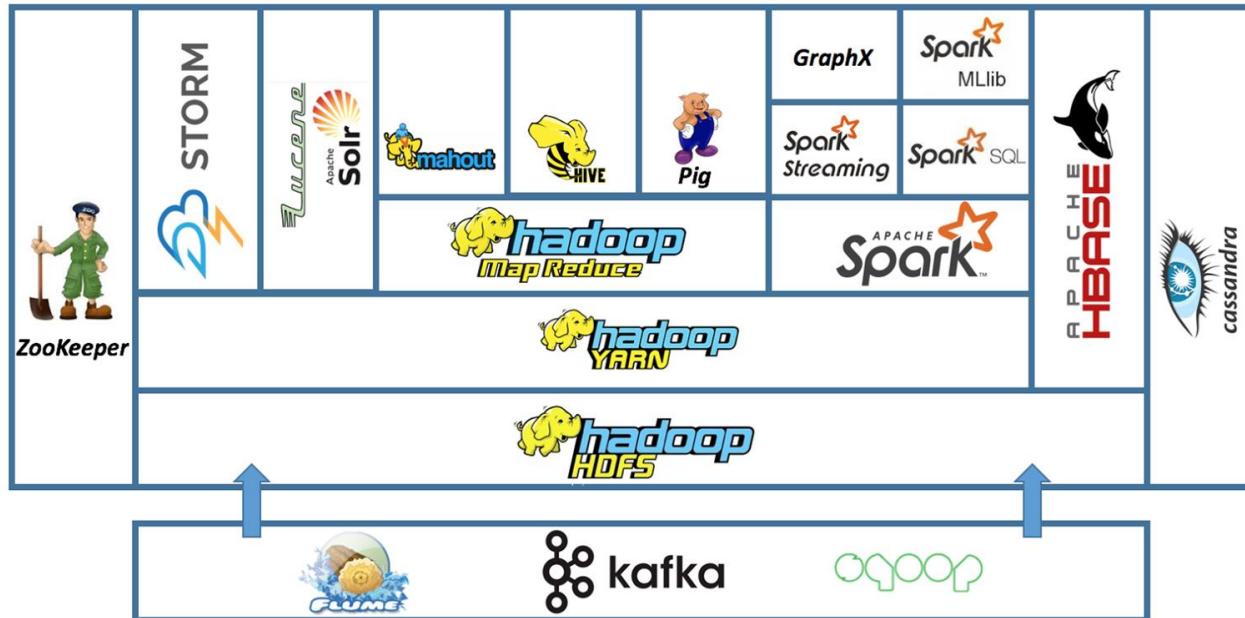
Overview of the Hadoop ecosystem ...  
[oreilly.com](http://oreilly.com)



Apache Hadoop open source ecosyste...  
[cloudera.com](http://cloudera.com)

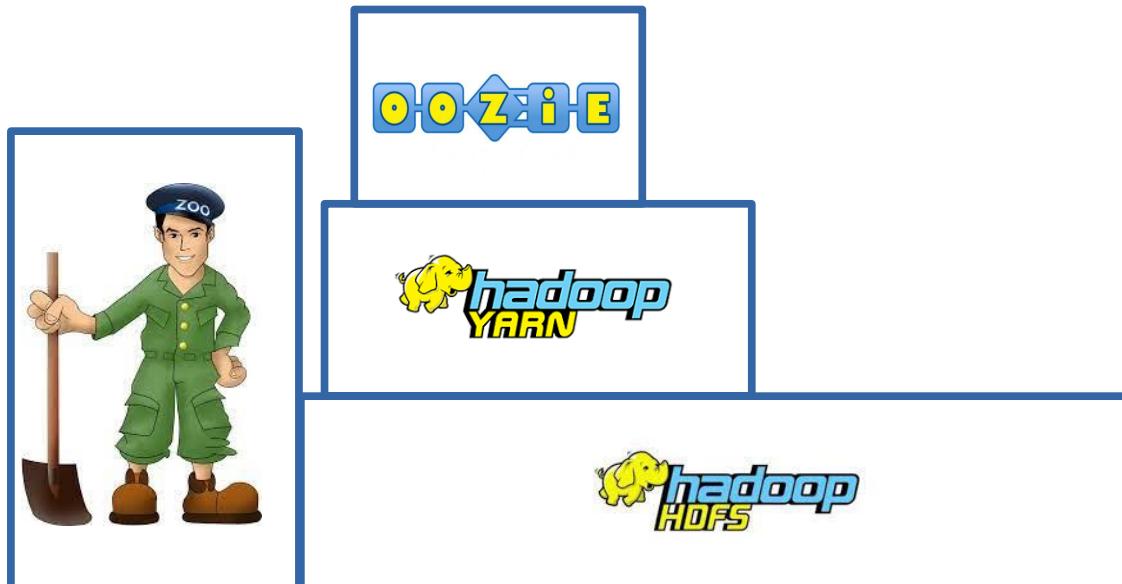


# Hadoop Ecosystem..

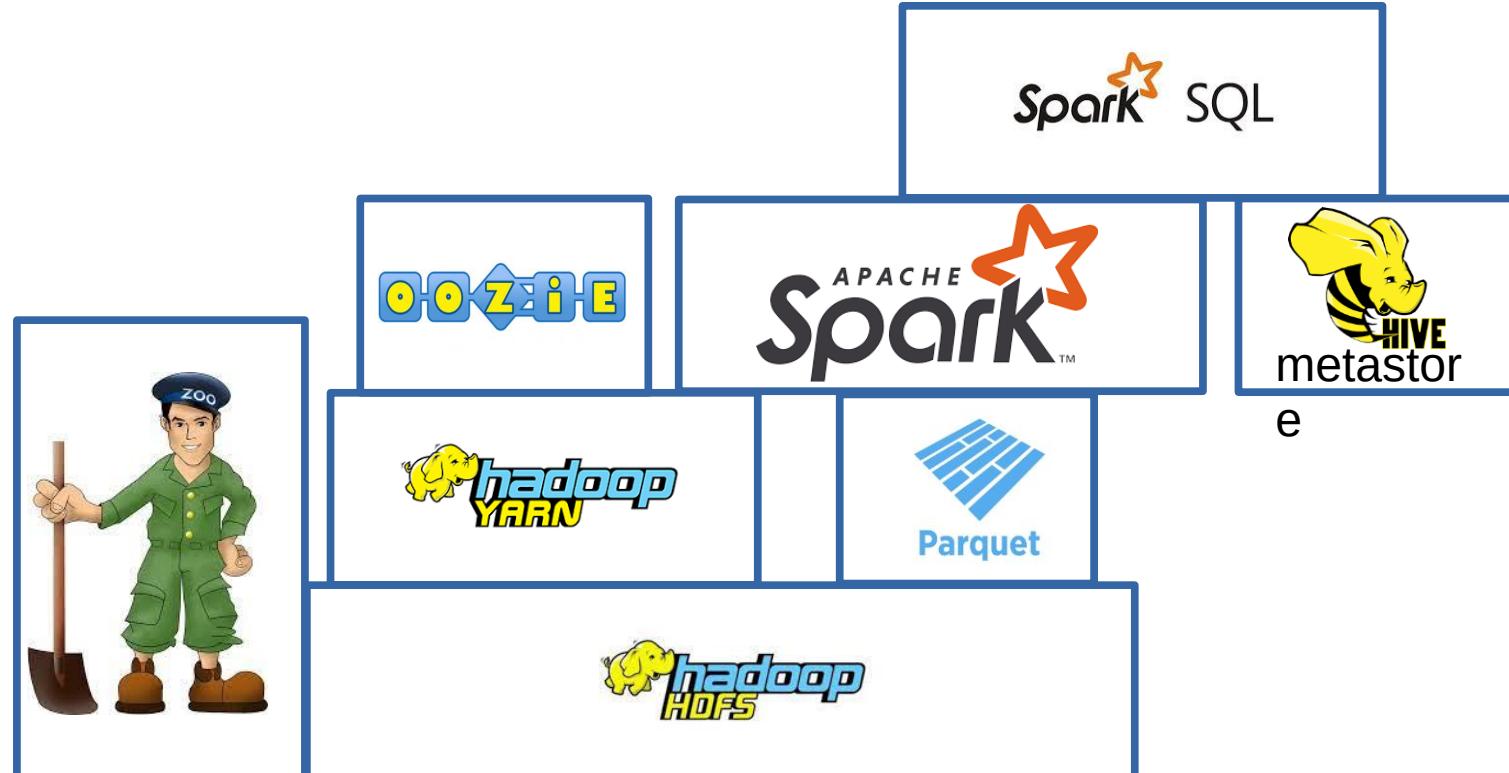


# Low-Level Focus

## ZooKeeper, HDFS, Yarn, Oozie



# Next Part4,5 ... High-Level Focus MetaStore, Parquet, Spark



# ZooKeeper



# <https://zookeeper.apache.org/>



Apache ZooKeeper™

Project ▾ Documentation ▾ Developers ▾ ASF ▾

## Welcome to Apache ZooKeeper™

Apache ZooKeeper is an effort to develop and maintain an open-source server which enables highly reliable distributed coordination.

### What is ZooKeeper?

ZooKeeper is a centralized service for maintaining configuration information, naming, providing distributed synchronization, and providing group services. All of these kinds of services are used in some form or another by distributed applications. Each time they are implemented there is a lot of work that goes into fixing the bugs and race conditions that are inevitable. Because of the difficulty of implementing these kinds of services, applications initially usually skimp on them, which make them brittle in the presence of change and difficult to manage. Even when done correctly, different implementations of these services lead to management complexity when the applications are deployed.

Learn more about ZooKeeper on the [ZooKeeper Wiki](#).

### Getting Started

Start by installing ZooKeeper on a single machine or a very small cluster.

1. Learn about ZooKeeper by reading the documentation.
2. Download ZooKeeper from the release page.

### Getting Involved

Apache ZooKeeper is an open source volunteer project under the Apache Software Foundation. We encourage you to learn about the project and contribute your expertise. Here are some starter links:

1. See our [How to Contribute to ZooKeeper](#) page.
2. Give us feedback: What can we do better?
3. Join the [mailing list](#): Meet the community.

Copyright © 2010-2020 The Apache Software Foundation, Licensed under the [Apache License, Version 2.0](#).

Apache ZooKeeper, ZooKeeper, Apache, the Apache feather logo, and the Apache ZooKeeper project logo are trademarks of The Apache Software Foundation.



# [https://en.wikipedia.org/wiki/Apache\\_ZooKeeper](https://en.wikipedia.org/wiki/Apache_ZooKeeper)



**WIKIPEDIA**  
The Free Encyclopedia

Main page  
Contents  
Current events  
Random article  
About Wikipedia  
Contact us  
Donate

Contribute  
Help  
Learn to edit  
Community portal  
Recent changes  
Upload file

Tools  
What links here  
Related changes  
Special pages  
Permanent link

Article [Talk](#)

Read [Edit](#) [View history](#)

Search Wikipedia



[Not logged in](#) [Talk](#) [Contributions](#) [Create account](#) [Log in](#)

## Apache ZooKeeper

From Wikipedia, the free encyclopedia

**Apache ZooKeeper** is an open-source server for highly reliable distributed coordination of cloud applications.<sup>[2]</sup> It is a project of the [Apache Software Foundation](#).

ZooKeeper is essentially a [service](#) for distributed systems offering a [hierarchical key-value store](#), which is used to provide a distributed [configuration service](#), [synchronization service](#), and [naming registry](#) for large distributed systems (see [Use cases](#)).<sup>[3]</sup> ZooKeeper was a sub-project of Hadoop but is now a [top-level Apache project](#) in its own right.

### Contents [hide]

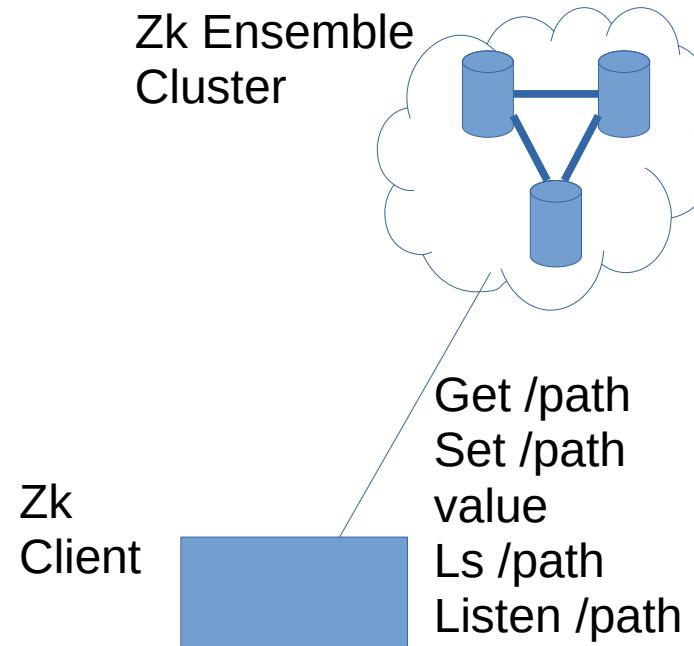
- [1 Overview](#)
- [2 Architecture](#)
- [3 Use cases](#)
- [4 Client libraries](#)
- [5 Apache projects using ZooKeeper](#)
- [6 See also](#)
- [7 References](#)
- [8 External links](#)

### Apache ZooKeeper



<b>Developer(s)</b>	Apache Software Foundation
<b>Stable release</b>	3.6.3 / April 13, 2021; 8 months ago <sup>[1]</sup>
<b>Repository</b>	<a href="#">ZooKeeper Repository</a>
<b>Written in</b>	Java
<b>Operating system</b>	Cross-platform
<b>Type</b>	Distributed computing
<b>License</b>	Apache License 2.0
<b>Website</b>	<a href="https://zookeeper.apache.org">zookeeper.apache.org</a>

# Hierarchical Key-Value Store



# Zk Shell

```
[zkshell: 0] help
```

ZooKeeper host:port cmd  
args

- get path [watch]
- ls path [watch]
- set path data [version]
- delquota [-n|-b] path
- quit
- printwatches on|off
- createpath data acl
- stat path [watch]
- listquota path
- history
- setAcl path acl
- getAcl path
- sync path
- redo cmdno
- addauth scheme auth
- delete path [version]

```
[zkshell] create /zk_test my_data1
```

Created /zk\_test

```
[zkshell] ls /
```

[zookeeper, zk\_test]

```
[zkshell] set /zk_test my_data2
```

```
[zkshell] get /zk_test
```

my\_data2  
cZxid = 5  
ctime = Fri Jun 05 13:57:06 PDT 2009  
mZxid = 5  
mtime = Fri Jun 05 13:57:06 PDT  
2009

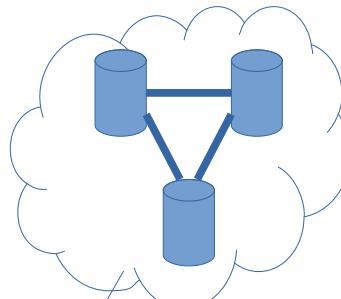
...

# Zk Session, EphemeralNode, Watch

ZooKeeper focus:

- Atomic updates / locks
- Inherit ACL per node
- Connection / Session  
(EphemeralNode for active session)
- Realtime Watch

# ZNode



Get key..  
Set  
key=value  
List key/\*  
Listen Key

Znode « / »

Znode « /data »

« /prop » = value

Znode « /servers »

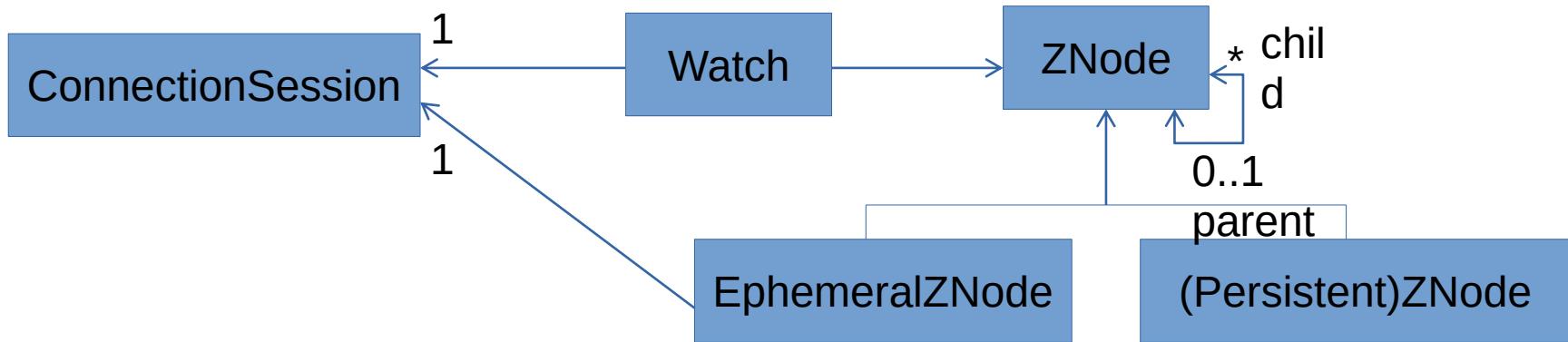
EphemeralZNode « node1»

EphemeralZNode « node2»

EphemeralZNode « node3»

Example for service  
discovery :  
Listen « /servers/\* »

# ZooKeeper Model



# Usages within Hadoop

## Use cases [ edit ]

Typical use cases for ZooKeeper are:

- Naming service
- Configuration management
- Data Synchronization
- Leader election
- Message queue
- Notification system

## Apache projects using ZooKeeper [ edit ]

- [Apache Hadoop](#)
  - [Apache Accumulo](#)
  - [Apache HBase](#)
  - [Apache Hive](#)
  - [Apache Kafka](#) ← Kafka 3 ... no more ZooKeeper
  - [Apache Solr](#)
  - [Apache Spark](#) ← ? support but no usage in Spark
  - [Apache NiFi](#)
  - [Apache Druid](#)
  - [Apache Helix](#)
- ← ? Wikipedia missing Apache BookKeeper  
( and Apache Pulsar)

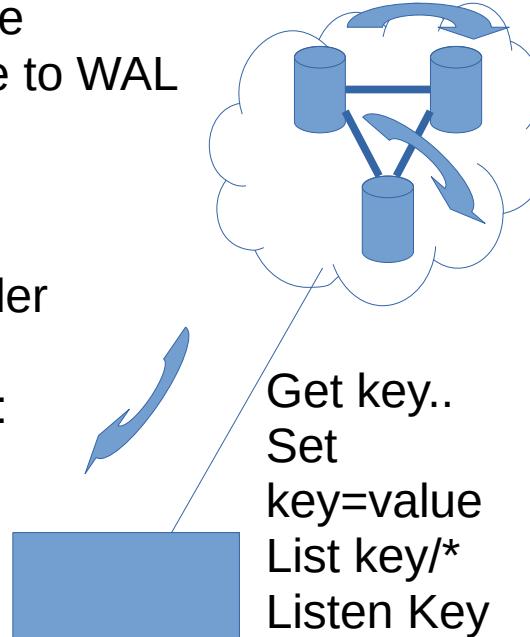
# Zk Quorum Write

1/ Forward  
write request  
to current Leader

Read requests:  
directly from  
nodes

2/ atomic  
change  
+ write to WAL

3/ broadcast change  
+ quorum for commit



# ZooKeeper : CP ... for small, mostly read data

Not a General Purpose Key-Value Database

All in memory ( + file + WAL ) !!

All operations are serialized, and wait Quorum

... Focus on consistency, Not “Available” in  
CAP

# Zk Alternatives

- Initially inspired by Google **Chubby**

- **Redis**

- <https://etcd.io/>

- (“A distributed, reliable key-value store for the most critical data of a distributed system”)  
... kernel of Kubernetes !

- **Gossip, Raft Protocols**

- <https://atomix.io/>

(“java library for fault tolerant distributed system”)

- **Curator** (library wrapper for low-level Zookeeper api)

# Zk Summary

- Extremely Robust & Mature
- store only critical data (small, mostly read)
- Low-level API, wrap in Curator
- Etcd better (Kubernetes choice)
- ... often: no need to use directly

# HDFS

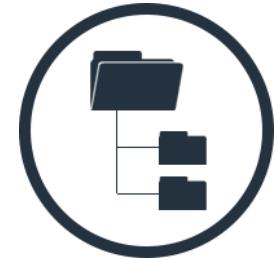
# Hadoop Distributed File System



# HDFS name explained

A **FileSystem** for directories and Files

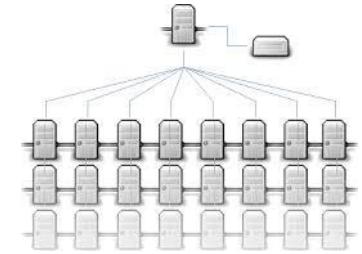
Like NTFS(windows), Ext4(linux), Ceph(distributed) ...



**Distributed**

works on cluster of  $\geq 100$  nodes  $\geq 1000$  disks

... to store Peta bytes



For/by **Hadoop**

API is also used by Spark or others..

implemented by AWS,Azure..

Abstract class  
**FileSystem**

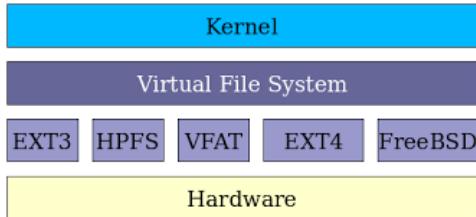
class  
**DistributedFileSystem**

# Abstract API / Concrete Implementation

```
package org.apache.hadoop.fs;

* An abstract base class for a fairly generic filesystem. It
@SuppressWarnings("DeprecatedIsStillUsed")
@interfaceAudience.Public
@interfaceStability.Stable
public abstract class FileSystem extends Configured
    implements Closeable, DelegationTokenIssuer, PathCapabilities {
```

Comparison with  
Linux



API  
Implementation

```
package org.apache.hadoop.hdfs;

public class DistributedFileSystem extends FileSystem
    implements KeyProviderTokenIssuer, BatchListingOperations {
```

# API

Standard operations  
mkdir, list, rm, rename...

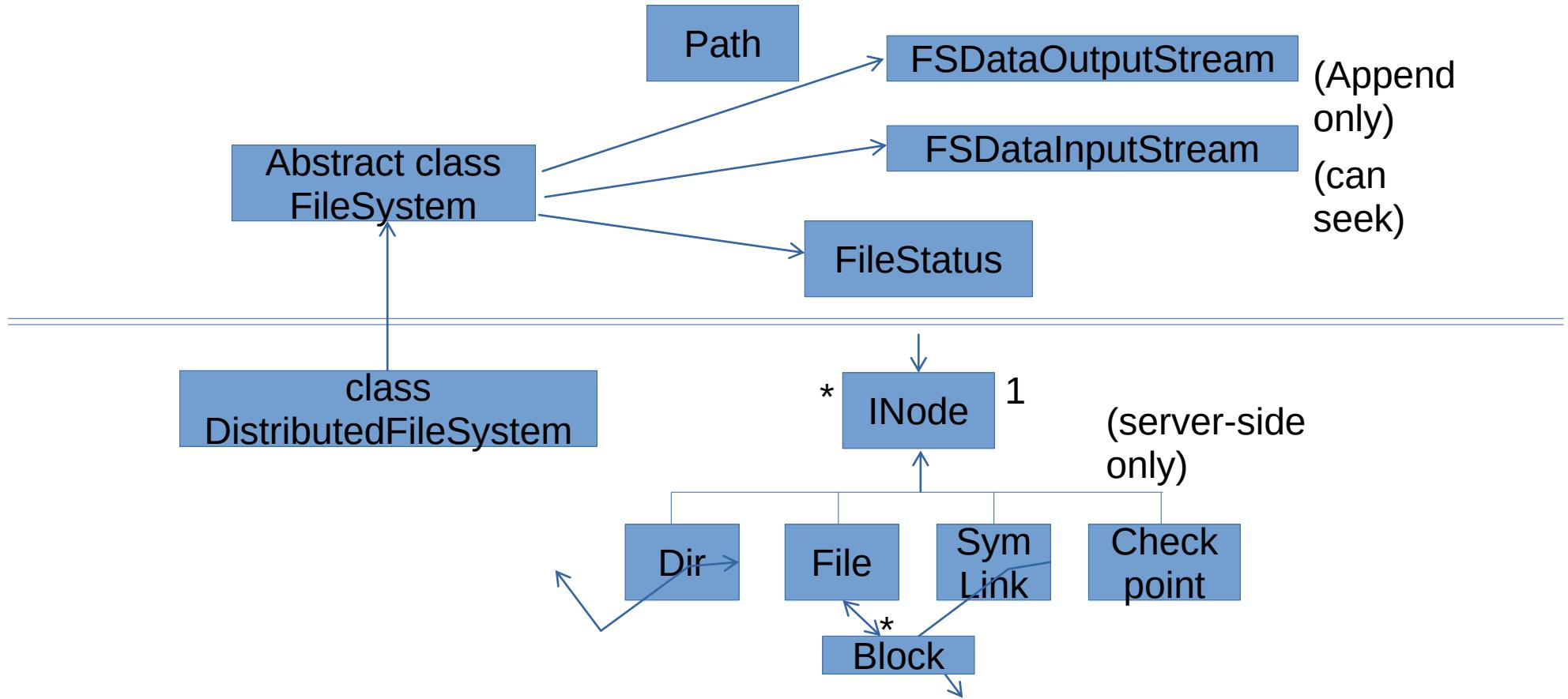
For file: **Append ONLY FileSystem**

To overwrite a single byte => overwrite all  
Only 1 exclusive writer

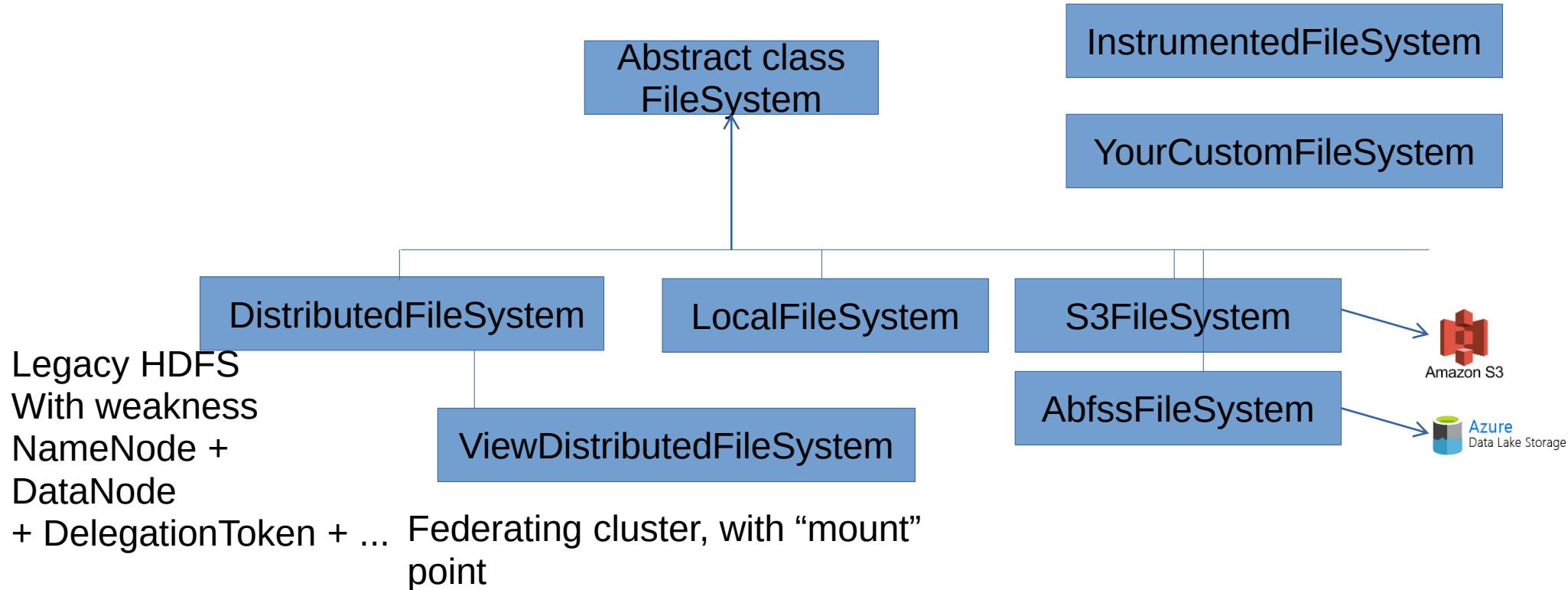
**Seek possible only in Read mode**

- `setConf(Configuration) : void`
- `getConf() : Configuration`
- `initialize(URI, Configuration) : void`
- `close() : void`
- `open(Path, int) : FSDataInputStream`
- `open(PathHandle, int) : FSDataInputStream`
- `create(Path, FsPermission, boolean, int, short, long) : FSDataOutputStream`
- `append(Path, int, Progressable) : FSDataOutputStream`
- `createFile(Path) : FSDataOutputStreamBuilder`
- `appendFile(Path) : FSDataOutputStreamBuilder`
- `createNonRecursive(Path, FsPermission, EnumSet<FsPermission>, Progressable) : FSDataOutputStream`
- `getUri() : URI`
- `setWorkingDirectory(Path) : void`
- `getWorkingDirectory() : Path`
- `rename(Path, Path) : boolean`
- `mkdirs(Path, FsPermission) : boolean`
- `delete(Path, boolean) : boolean`
- `listStatus(Path) : FileStatus[]`
- `getFileStatus(Path) : FileStatus`
- `addDelegationTokens(String, Credentials) : Token`
- `getScheme() : String`
- `getCanonicalServiceName() : String`
- `getName() : String`
- `makeQualified(Path) : Path`
- `getDelegationToken(String) : Token<?>`
- `getChildFileSystems() : FileSystem[]`
- `getAdditionalTokenIssuers() : DelegationToken`
- `getFileBlockLocations(FileStatus, long, long) : BlockLocation[]`
- `getFileBlockLocations(Path, long, long) : BlockLocation[]`
- `getServerDefaults() : FsServerDefaults`
- `getServerDefaults(Path) : FsServerDefaults`
- `resolvePath(Path) : Path`
- `createNewFile(Path) : boolean`
- `concat(Path, Path[]) : void`
- `getReplication(Path) : short`
- `setReplication(Path, short) : boolean`

# Api UML Classes



# Not only HDFS ...



```
$ hdfs dfs <cmd> <args..>
```

```
$ hdfs dfs -ls "/a/b"
```

```
$ hdfs dfs -mkdir "/a/b/c"
```

```
$ hdfs dfs -put localFile "/a/b/c/remoteFile"
```

```
$ hdfs dfs -get "/a/b/c/remoteFile" localFile
```

```
$ hdfs dfs -cp -r "/a/b" "/a/bCopy"
```

```
$ hdfs dfs -rm -r "/a/b" "/a/bCopy"
```

# Hadoop Implementation Limits

OK to store Peta bytes, with files > Giga octets

**BUT HDFS does not like “Small” Files ( “Too many” Files ) !!**

i.e. > 200 Millions... with NameNode jvm > -Xmx200G ...

**NameNode is THE bottleneck of HDFS**

# NameNode ...

**All metadata (“name”) operations starts with the NameNode**

NameNode maintains all metadata files in jvm memory !! **1 File ~ 4ko**

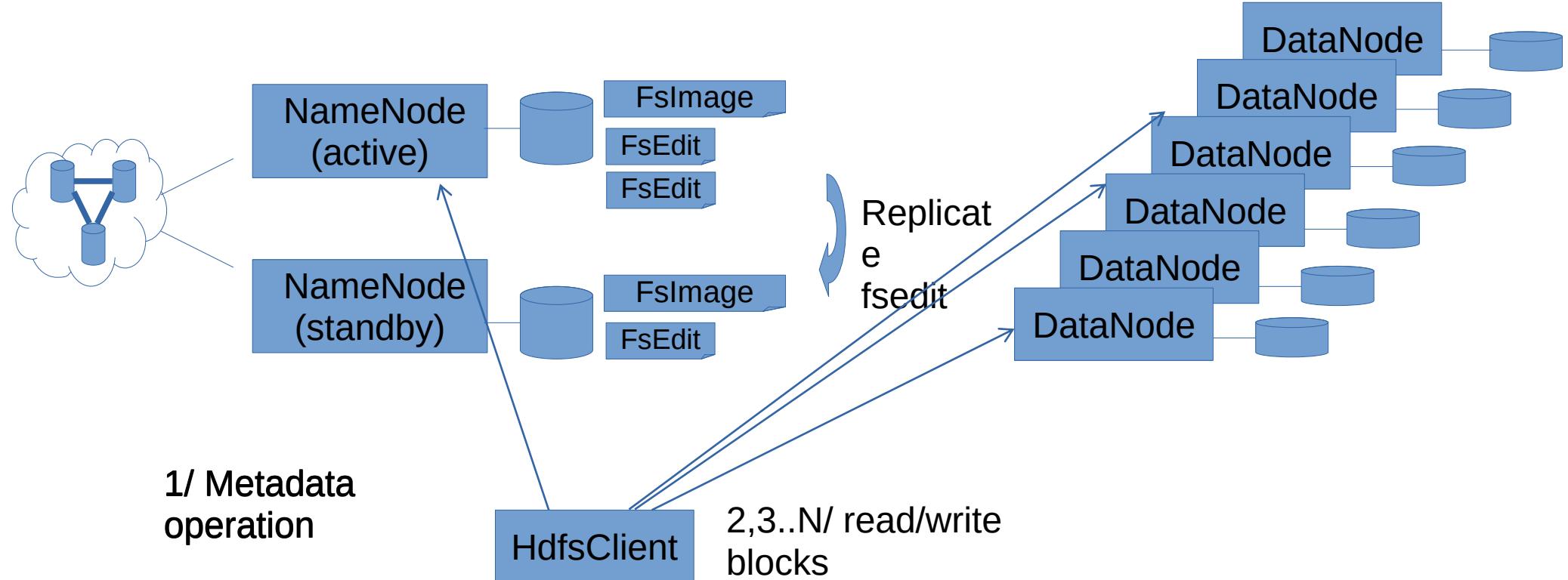
**Huge RAM + Disk needed**

**JVM needs GC Tuning**

The NameNode is **critical** (gc / corrupted file / crashing => system OFF )

NameNode manages kerberos delegation token

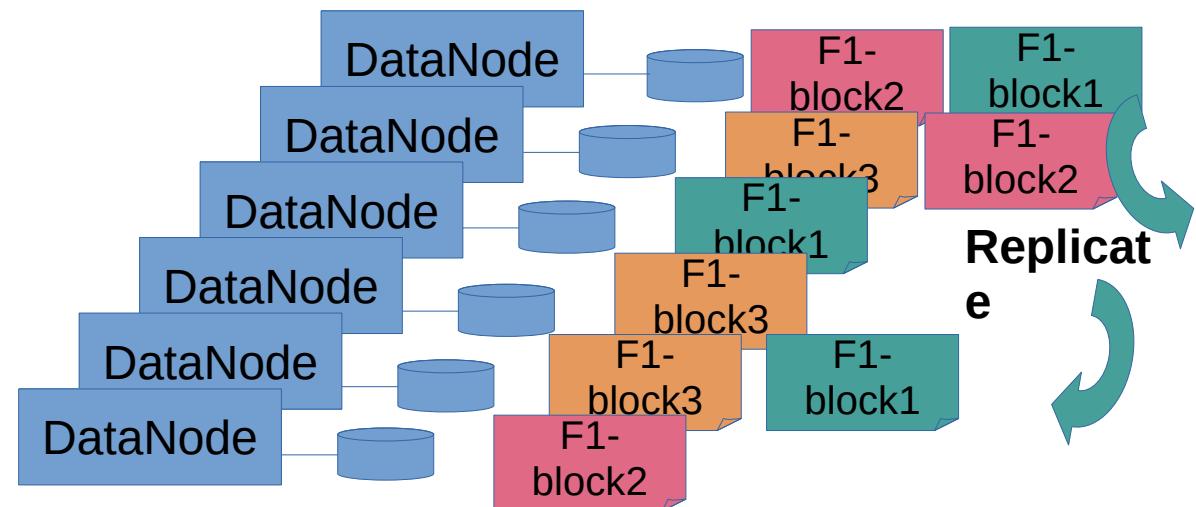
# HDFS = 2 NameNodes + N x DataNodes



# DataNode ... store blocks

## 1 File = N x Blocks (default: 128M)

Logical File  
metadata



Physical Block repartition on  
DataNodes

# RF (Replication Factor) = 3

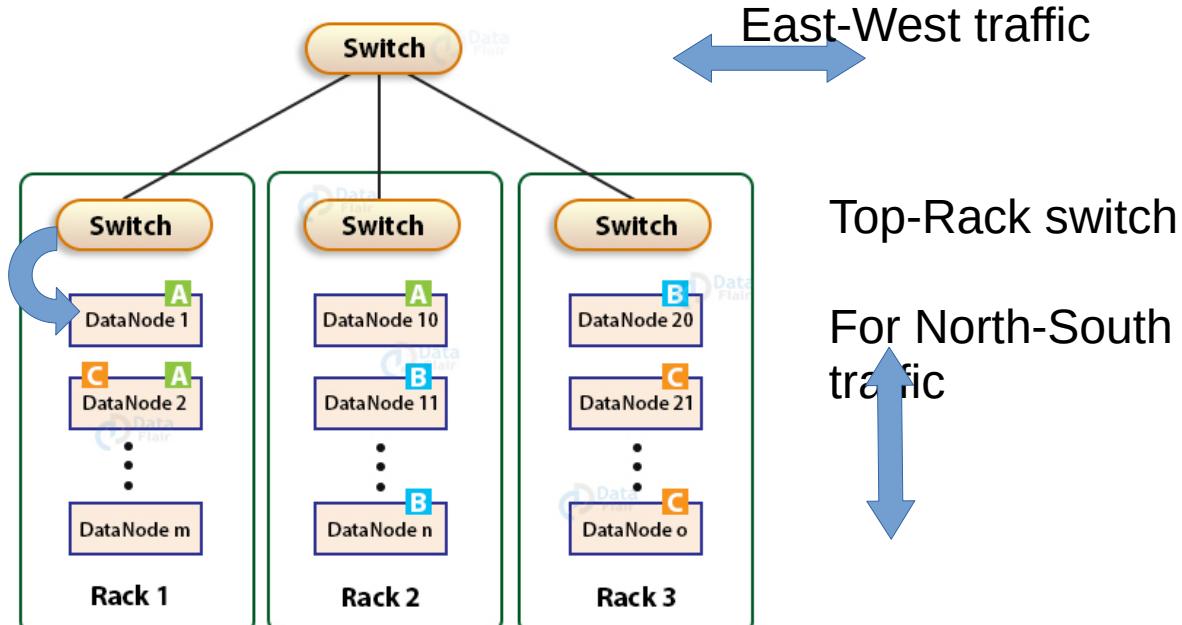
active NameNode manage  
replications:

When a DataNode “disappears” => all its blocks are lost (under replicated) =>  
replicate

When a DataNode “joins” cluster => it sends its “block-report”  
( at startup, NameNode is in “safe mode”... it waits enough block-reports to start)

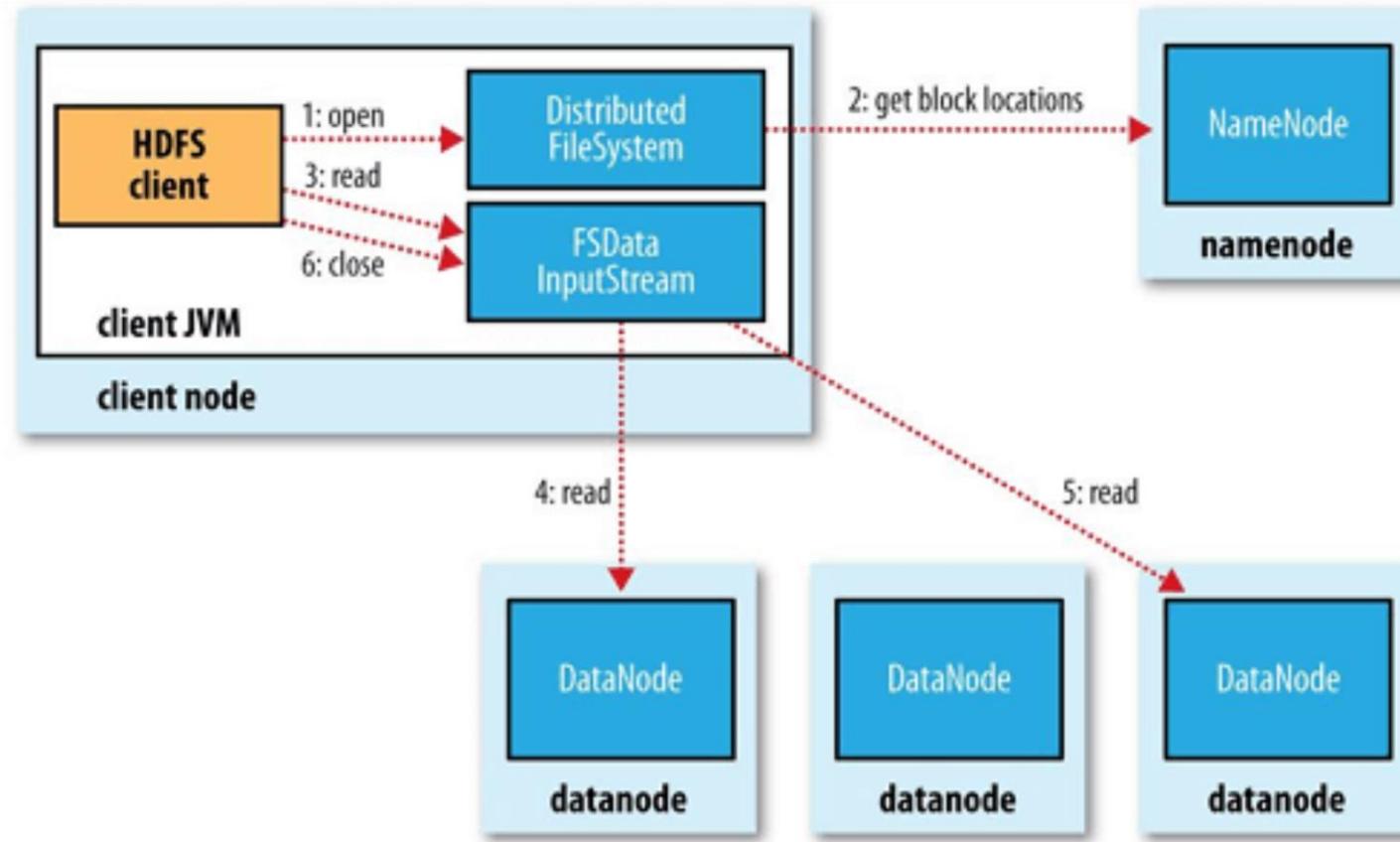
# RF ... Rack Aware

Can replicate  
1  
in same Rack  
(fast)

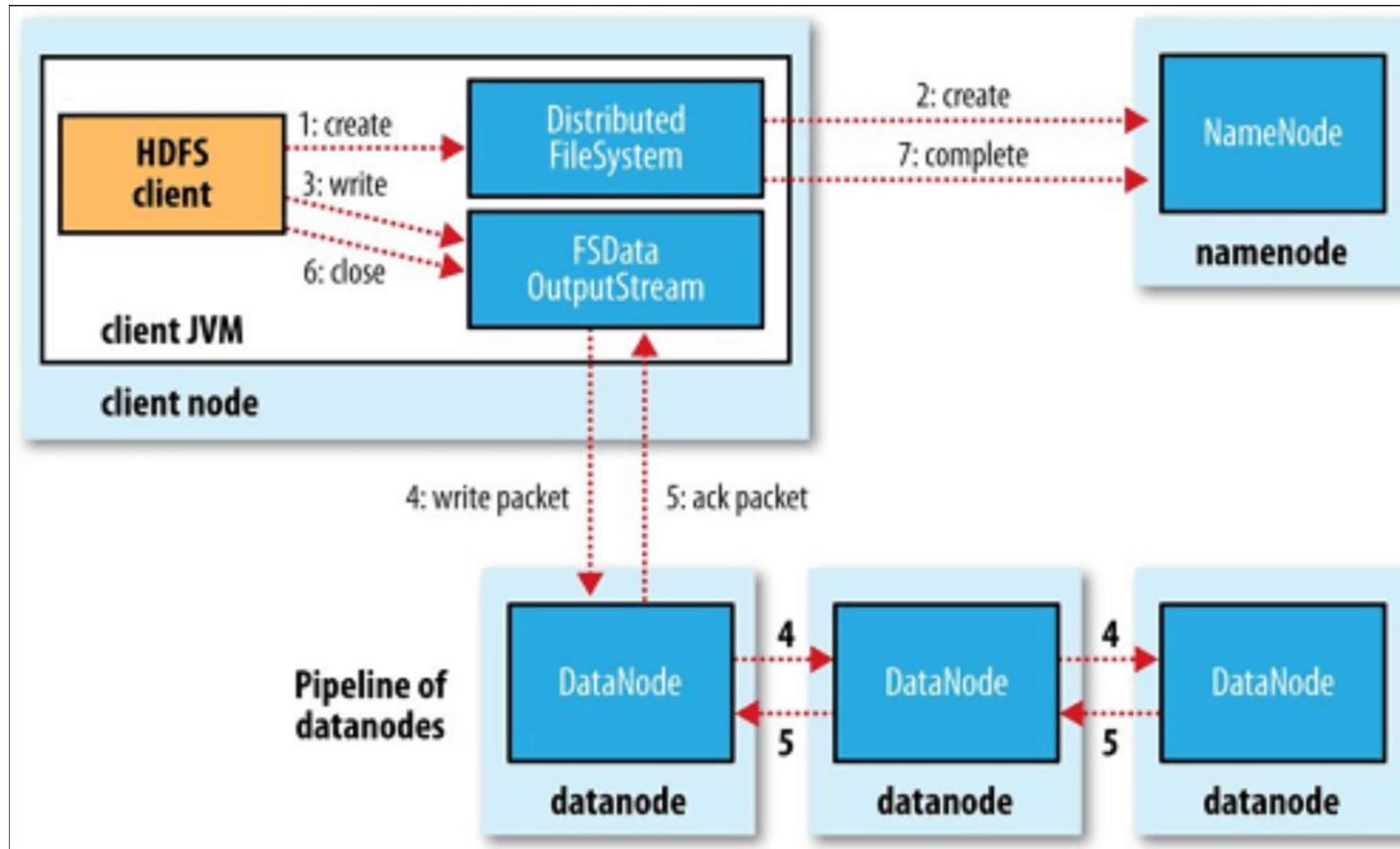


Must replicate  
 $\geq 1$   
in other Rack

# Reading a HDFS File



# Writing a HDFS File

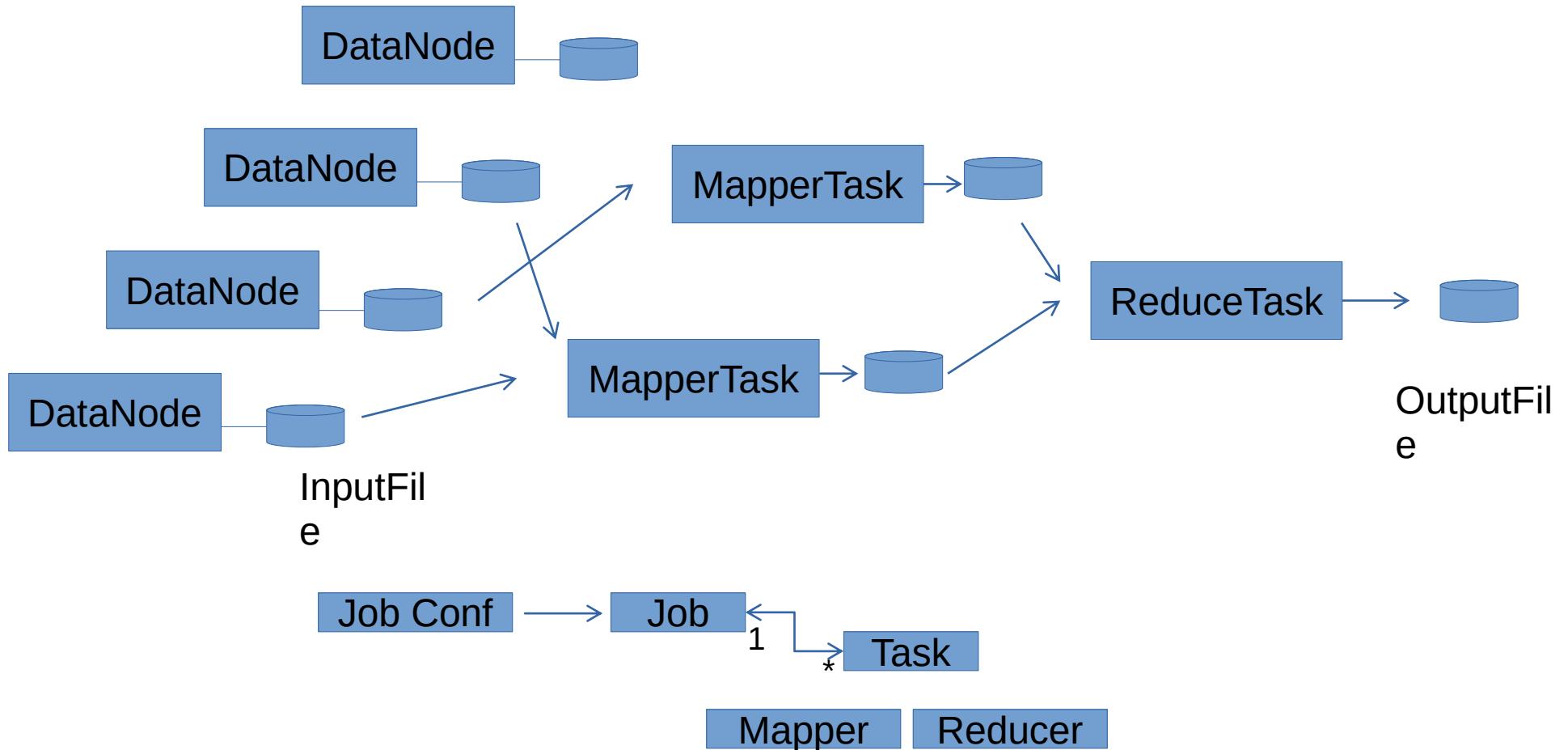


Default size ... Block(128M) => packet(64K) => 126 \* chunk (512b + checksum 4b)

# MapReduce



# MapReduce Principle



# MapReduce .. replaced by Yarn (then Kubernetes) + Spark

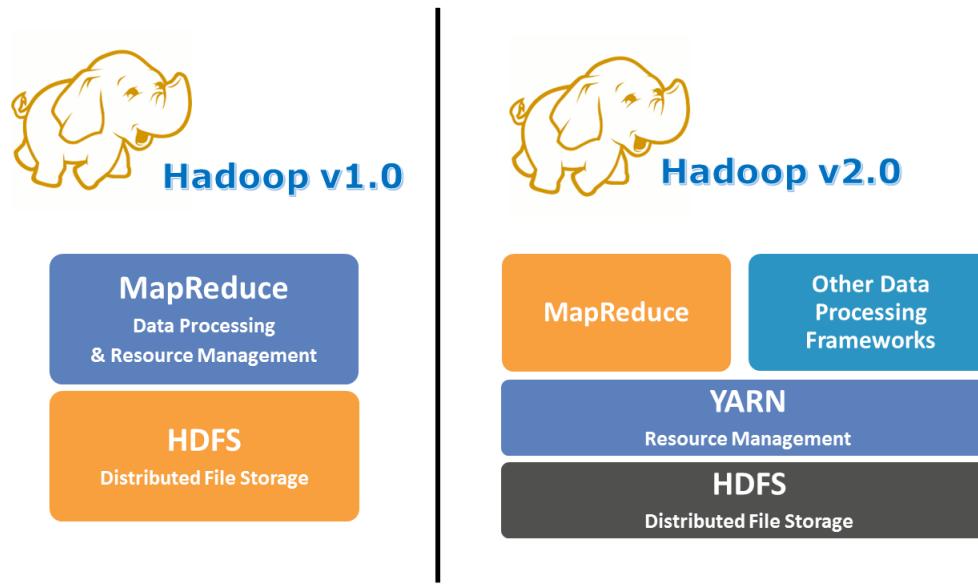
Origin = Google paper

Job resources launcher refactored in “Yarn”

MapReduce 100x slower than Spark (inefficient saving to disk)

=> Nobody use MapReduce anymore (except oozie, to launch shell commands)

# (legacy) Map Reduce ... Yarn



# Yarn



# Yarn: Yet Another Resource Negotiation

SSH

< MapReduce1 (no more used)

< Mesos (~ nobody uses)

< Yarn v2 (stuck to Hadoop)

< Kubernetes ( ... very Fashion since 2014 )

Google Borg / Omega (internal, inspiring Kubernetes)

# Spark clustering supports

None --master local[\*]

SSH (mode “standalone”)

--master **spark://IP:PORT**

< Mesos (since 1.0 ... **deprecated** in 3.2.0)

--master **mesos://host:5050**

< Yarn v2

--master **yarn**

< Kubernetes (**since 3.0**)

--master **k8s://https://<k8s-apiserver>**

}

Past



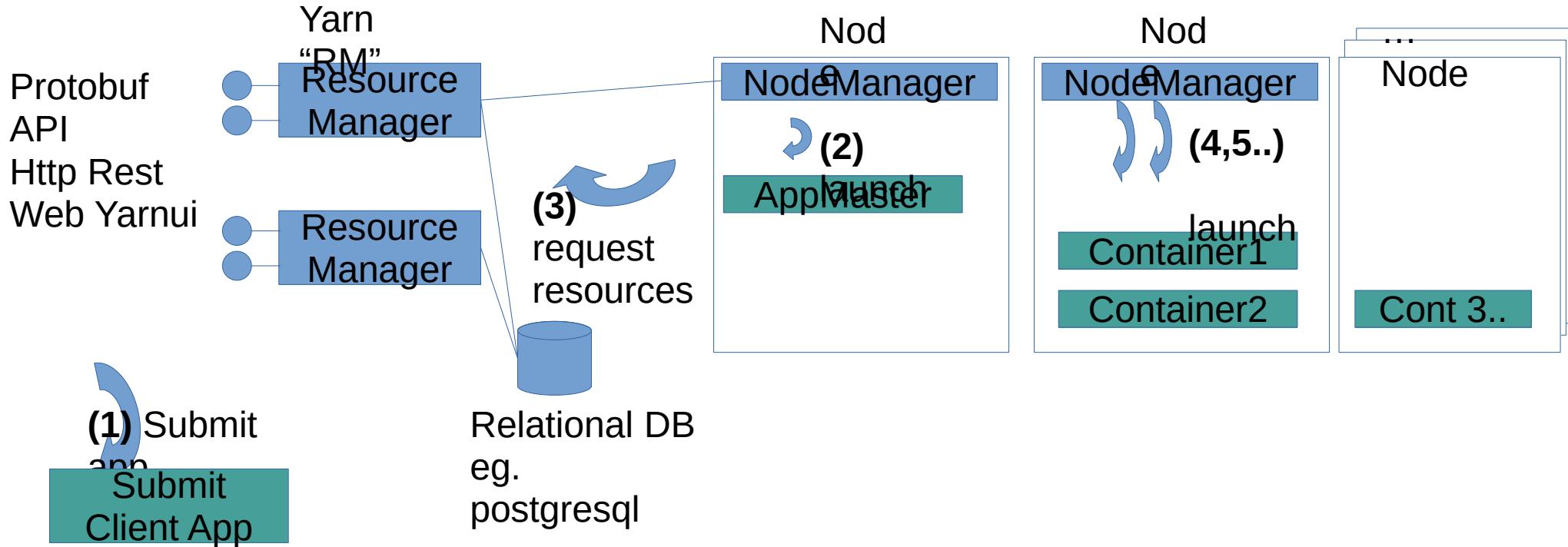
}

Present

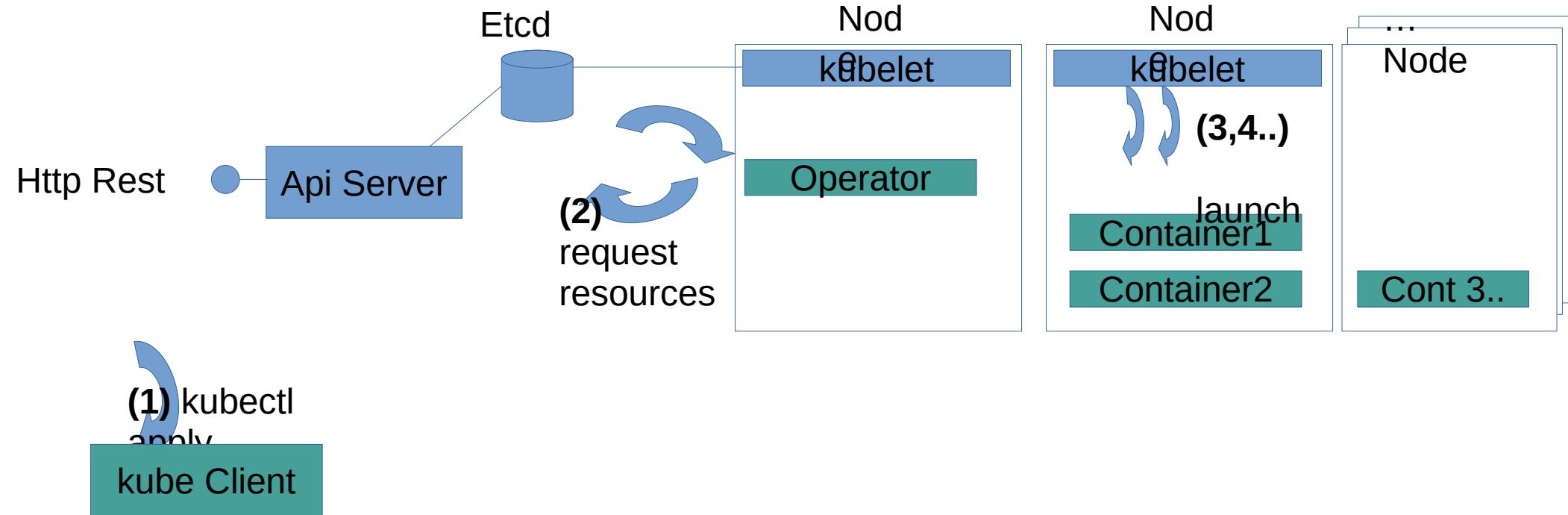
}

Future

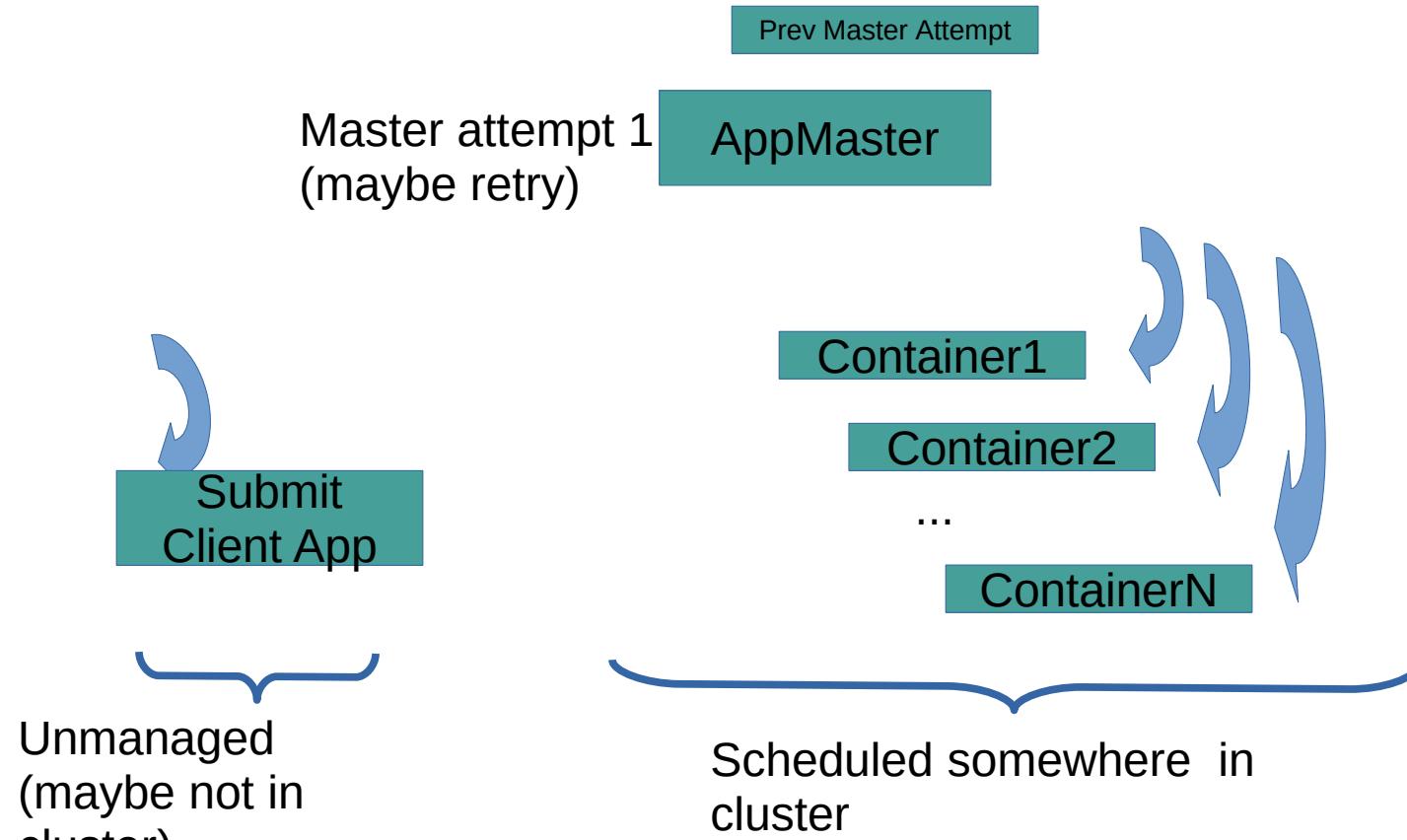
# Yarn Architecture



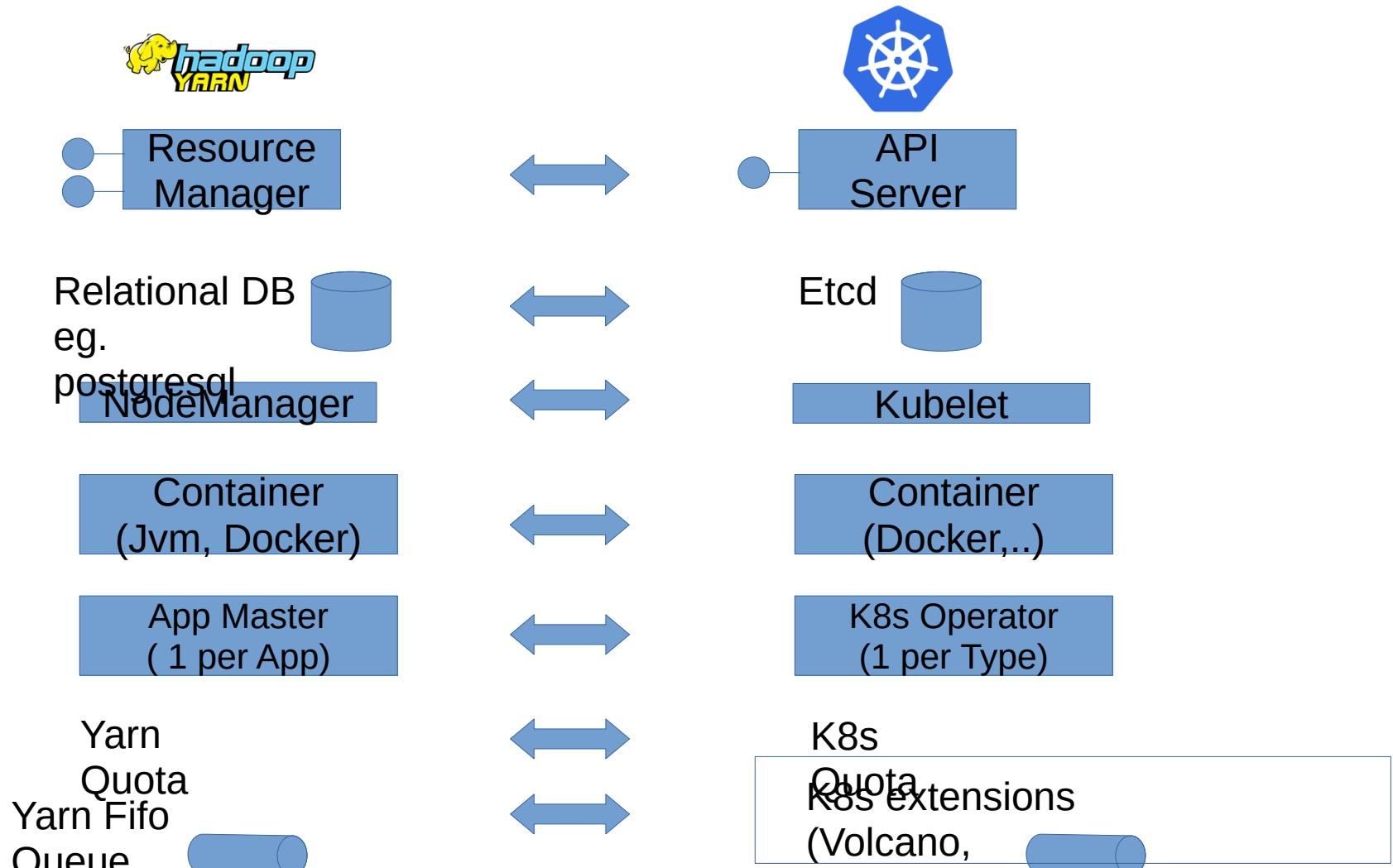
# Comparison ... Kubernetes



# Yarn app: 3..N Java process Client+Master+N Containers



# Comparison with Kubernetes



# Yarn Queue



## NEW, NEW\_SAVING, SUBMITTED, ACCEPTED, RUNNING Applications

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved	Active Nodes	Decommissioned Nodes	Lost Nodes
1	0	1	0	12	20.88 GB	102.38 GB	0 B	12	64	0	4	0	0

Scheduler Metrics

Scheduler Type	Scheduling Resource Type	Minimum Allocation	Maximum Alloc
Capacity Scheduler	[MEMORY]	<memory:32, vCores:1>	<memory:26208, vCores:16>

Application Queues

Legend: Capacity (Green), Used (Grey), Used (over capacity) (Orange), Max Capacity (Light Grey)

- root (Green bar, 20.4% used)
- q1 (Grey bar, 0.0% used)
- q1.q11 (Grey bar, 0.0% used)
- q1.q12 (Grey bar, 0.0% used)
- q2 (Grey bar, 102.0% used, orange background)
- default (Orange bar, 102.0% used)

Show 20+ entries

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Progress	Tracking
application_1494408864466_0001	hadoop	random-text-writer	MAPREDUCE	default	Wed May 10 18:07:39 +0800 2017	N/A	RUNNING	UNDEFINED	0%	Application

## Queue Hierarchy :

Child queues => sum to 100%

Capacity / Max Capacity  
... over-quota allowed

If Preemption enabled =>  
may kill app in over-quota

# Yarn Capacity Scheduler config

file: scheduler-conf.xml

```
<property>
  <name>yarn.scheduler.capacity.root.queues</name>
  <value>a,b,c</value>
</property>
<property>

<name>yarn.scheduler.capacity.{QueueXX}.{propertyYY}</name>
  <value>..</value>          property:
    .queues      .. child queue names
    .capacity     12.5           Queue capacity in %
    .maximum-capacity 15.0       Maximum queue capacity in %
    .minimum-user-limit-percent 100
    .user-limit-factor 1
    .maximum-allocation-mb     80G max memory for each container
    .maximum-allocation-vcores 32 max vcores for each container
    .weight        1.5
```

# Kubernetes Queue ... Volcano, UniKorn

Not built-in in Kubernetes !

Allocation spark-driver => will allocate N x spark-executors  
next

K8s built-in “pod scheduler” NOT adapted for BigData



<https://volcano.sh>  
/



<https://unikorn.apache.org>  
/

```
# kubectl create -f job.yaml
apiVersion: batch.volcano.sh/v1alpha1
kind: Job
metadata:
  name: job-1
spec:
  minAvailable: 1
  schedulerName: volcano
  queue: test
  policies:
    - event: PodEvicted
      action: RestartJob
  tasks:
    - replicas: 1
      name: ..
      policies:
        - event: TaskCompleted
          action: CompleteJob
  template:
    spec:
      containers:
        - ...
      resources:
        requests:
          cpu: 1
        limits:
          cpu: 1
    restartPolicy: Never
```

# Launching App in Yarn ...

## conf in java ... >100 lines of code

Architecture
Commands Reference
Capacity Scheduler
Fair Scheduler
ResourceManager Restart
ResourceManager HA
Resource Model
Node Labels
Web Application Proxy
Timeline Service V.2
Writing Yarn Applications
YARN Application Security
NodeManager
Running Applications in Docker Containers
Using Cgroups
Secure Containers
Reservation System
Graceful Decommission
Opportunistic Containers
YARN Federation
Shared Cache
Using GPU
Using FPGAs
Placement Constraints
YARN REST APIs
Introduction
Resource Manager
Node Manager
Timeline Server
Timeline Service V.2
YARN Service
Overview
QuickStart

### Writing a Simple YARN Application

#### Writing a simple Client

- The first step that a client needs to do is to initialize and start a YarnClient.

```
YarnClient yarnClient = YarnClient.createYarnClient();
yarnClient.init(conf);
yarnClient.start();
```

- Once a client is set up, the client needs to create an application.

```
YarnClientApplication app = yarnClient.createApp();
GetNewApplicationResponse appResponse = app.get
```

- The response from the YarnClientApplication for a new application is required so that to ensure that you can correctly set the details.

- The main crux of a client is to setup the ApplicationSubmissionContext:

- Application info: id, name

- Queue, priority info: Queue to which the application will be assigned

- User: The user submitting the application

- ContainerLaunchContext: The information defining the configuration needed to run the application such as the local resources.

```
// set the application submission context
ApplicationSubmissionContext appContext = app.getApplicationSubmissionContext();
ApplicationId appId = appContext.getApplicationId();

appContext.setKeepContainersAcrossRacks(true);
appContext.setApplicationName(appResponse.getApplicationId().getName());
appContext.setQueue(appResponse.getQueue());

// Set local resources for the application
// local files or archives as needed
// In this scenario, the jar file
Map<String, LocalResource> localResources = new HashMap<String, LocalResource>();
LOG.info("Copy App Master jar file");
// Copy the application master jar file
// Create a local resource to point to it
FileSystem fs = FileSystem.get(conf);
addToLocalResources(fs, appMasterLocalResources, null);

// Set the log4j properties if needed
if (!log4jPropFile.isEmpty()) {
    addToLocalResources(fs, log4jProperties, log4jPropFile);
}

// The shell script has to be made executable
// where it will be executed.
// To do this, we need to first copy it to the yarn framework.
// We do not need to set this as master as the application master
String hdfsShellScriptLocation = YarnConfiguration.YARN_AIRFLOW_SCRIPT_PATH;
long hdfsShellScriptLen = 0;
long hdfsShellScriptTimestamp = 0;
if (!shellScriptPath.isEmpty()) {
    Path shellSrc = new Path(shellScriptPath);
    String shellPathSuffix =
        appMasterId + "/" + appId.toStr();
    Path shellDst =
        new Path(fs.getHomeDirectory());
    fs.copyFromLocalFile(false, true, shellSrc, shellDst);
    hdfsShellScriptLocation = shellDst.toString();
    FileStatus shellFileStatus = fs.getFileStatus(shellDst);
    hdfsShellScriptLen = shellFileStatus.getLen();
    hdfsShellScriptTimestamp = shellFileStatus.getModificationTime();
}

// Set the necessary command
Vector<CharSequence> args =
    new Vector<CharSequence>(2);
args.add(appMasterId);
args.add(appId.toStr());

// Set java executable command
LOG.info("Setting up app master");
args.add(Environment.JAVA_HOME);
// Set Xmx based on am memory
args.add("-Xmx" + amMemory);
args.add("-Djava.class.name=" + appMasterMainClass);
// Set params for ApplicationMaster
args.add("--container-memory=" + containerMemory);
args.add("--container-vcores=" + containerVcores);
args.add("--num-containers=" + numContainers);
args.add("--priority=" + priority);

// Service data is a binary blob
// Not needed in this scenario
// amContainer.setServiceData(args);

// Setup security tokens
if (UserGroupInformation.isSecurityEnabled()) {
    // Note: Credentials class is marked as LimitedPrivate for HDFS and MapReduce
    Credentials credentials = new Credentials();
    String tokenRenewer = conf.get(YarnConfiguration.RM_PRINCIPAL);
    if (tokenRenewer == null || tokenRenewer.length() == 0) {
        throw new IOException("Can't get Master Kerberos principal for the RM to use as renewer");
    }

    // For now, only getting tokens for the default file-system.
    final Token[] tokens =
        fs.addDelegationTokens(tokenRenewer, credentials);
    if (tokens != null) {
        for (Token token : tokens) {
            LOG.info("Got dt for " + fs.getUri() + "; " + token);
        }
    }
}

DataOutputBuffer dob = new DataOutputBuffer();
credentials.writeTokenStorageToStream(dob);
ByteBuffer fsTokens = ByteBuffer.wrap(dob.getData(), 0, dob.getLength());
amContainer.setTokens(fsTokens);
}

appContext.setAMContainerSpec(amContainer);

// After the setup process is complete, the client is ready to submit the application with specified priority and queue.

// Set the priority for the application master
Priority pri = Priority.newInstance(amPriority);
appContext.setPriority(pri);

// Set the queue to which this application is to be submitted in the RM
appContext.setQueue(amQueue);

// Submit the application to the applications manager
// SubmitApplicationResponse submitResp = applicationsManager.submitApplication(appRequest);

yarnClient.submitApplication(appContext);
```

# Just the beginning... Need also ApplicationMaster code!

```
Map<String, String> envs = System.getenv();
String containerIdString =
    envs.get(ApplicationConstants.AM_CONTAINER_ID_ENV);
if (containerIdString == null) {
    // container id should always be set in the env by the framework
    throw new IllegalArgumentException(
        "ContainerId not set in the environment");
}
ContainerId containerId = ConverterUtils.toContainerId(containerIdString);
ApplicationAttemptId appAttemptID = containerId.getApplicationAttemptId();
```

- After an AM has initialized itself completely, we can start the two clients: one talk about those event handlers in detail later in this article.

```
AMRMClientAsync.CallbackHandler allocListener = new RMCallbackHandler();
amRMClient = AMRMClientAsync.createAMRMClientAsync(1000, allocListener);
amRMClient.init(conf);
amRMClient.start();

containerListener = createNMCallbackHandler();
nmClientAsync = new NMClientAsyncImpl(containerListener);
nmClientAsync.init(conf);
nmClientAsync.start();
```

- The AM has to emit heartbeats to the RM to keep it informed that the AM is alive and still running. This is done via `YarnConfiguration.RM_AM_EXPIRY_INTERVAL_MS` with the default being defined by `YarnConfiguration.DYNAMIC_ALLOCATION_ENABLED`. The ResourceManager to start heartbeating.

```
// Register self with ResourceManager
// This will start heartbeating to the RM
appMasterHostname = Netutils.getHostname();
RegisterApplicationMasterResponse response = amRMClient
    .registerApplicationMaster(appMasterHostname, appMasterRpcPort,
        appMasterTrackingUrl);
```

- Based on the task requirements, the AM can ask for a set of containers.

```
List<Container> previousAMRunningContainers =
    response.getContainersFromPreviousAttempts();
LOG.info("Received " + previousAMRunningContainers.size() +
    " previous AM's running containers on AM request");

int numTotalContainersToRequest =
    numTotalContainers - previousAMRunningContainers.size();
// Setup ask for containers from RM
// Send request for containers to RM
// Until we get our fully allocated quota, we keep
// requesting more containers
// Keep looping until all the containers are launched
// executed on them ( regardless of success/failure )
for (int i = 0; i < numTotalContainersToRequest; i++) {
    ContainerRequest containerAsk = setupContainerAsk();
    amRMClient.addContainerRequest(containerAsk);
}
```

```
// Set the necessary command to execute on the allocated container
Vector<CharSequence> vargs = new Vector<CharSequence>(5);

// Set executable command
vargs.add(shellCommand);
// Set shell script path
if (!scriptPath.isEmpty()) {
    vargs.add(Shell.WINDOWS ? ExecBatScriptStringPath :
        ExecShellStringPath);
}

// Set args for the shell command if any
vargs.add(shellArgs);
// Add log redirect params
vargs.add("1>" + ApplicationConstants.LOG_DIR_EXPANSION_VAR + "/stdout");
vargs.add("2>" + ApplicationConstants.LOG_DIR_EXPANSION_VAR + "/stderr");

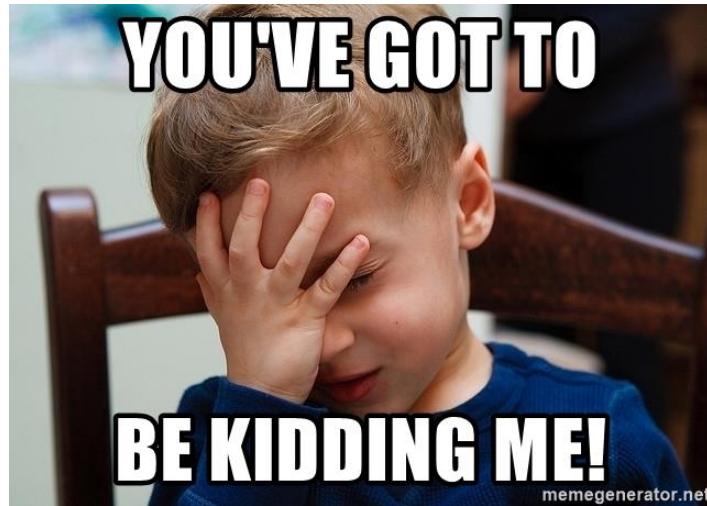
// Get final command
StringBuilder command = new StringBuilder();
for (CharSequence str : vargs) {
    command.append(str).append(" ");
}

List<String> commands = new ArrayList<String>();
commands.add(command.toString());

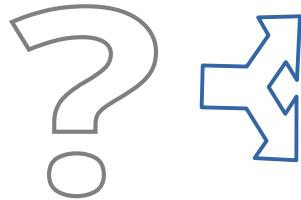
// Set up ContainerLaunchContext, setting local resource, environment,
// command and token for constructor.

// Note for tokens: Set up tokens for the container too. Today, for normal
// shell commands, the container in distribute-shell doesn't need any
// tokens. We are populating them mainly for NodeManagers to be able to
// download any files in the distributed file-system. The tokens are
// otherwise also useful in cases, for e.g., when one is running a
// "hadoop dfs" command inside the distributed shell.
ContainerLaunchContext ctx = ContainerLaunchContext.newInstance(
    localResources, shellEnv, commands, null, allTokens.duplicate(), null);
containerListener.addContainer(container.getId(), container);
nmClientAsync.startContainerAsync(container, ctx);
```

# Using Yarn api..



# Launching app on Yarn ?



Simple App  
(not  
distributed)

Distributed  
Hadoop  
App,  
not spark

→ Builtin:  
Spark-submit  
--master yarn →

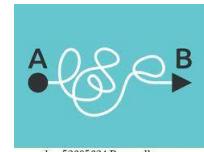


→ →



Declare in oozie  
workflow.xml  
<action> <shell>...

→ Reconsider !! →



Unpractical  
Config + Web  
UI



Maybe  
services  
- Multiple oozie

# Yarn Commands Line (not for launch)

```
$ yarn app -list
```

```
$ yarn app -status <applicationId>
```

```
$ yarn app -kill <applicationId>
```

```
$ yarn logs -applicationId <applicationId>
```

```
$ yarn logs -containerId <containerId>
```

# Yarn UI & Logs

As a developper, ops, admin, datascientist ...

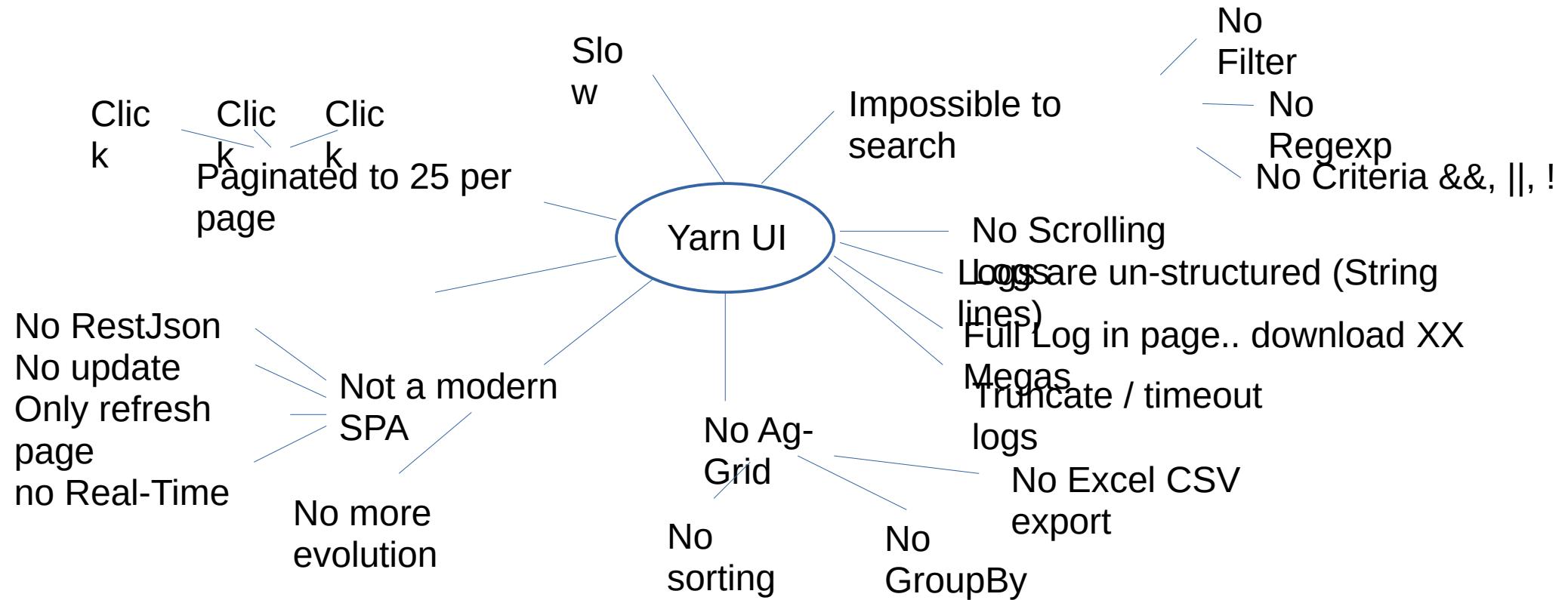
It is **necessary** to get logs to diagnose/launch/understand your job app  
**Web UI** “not mandatory”, but (would have) help (if practical)



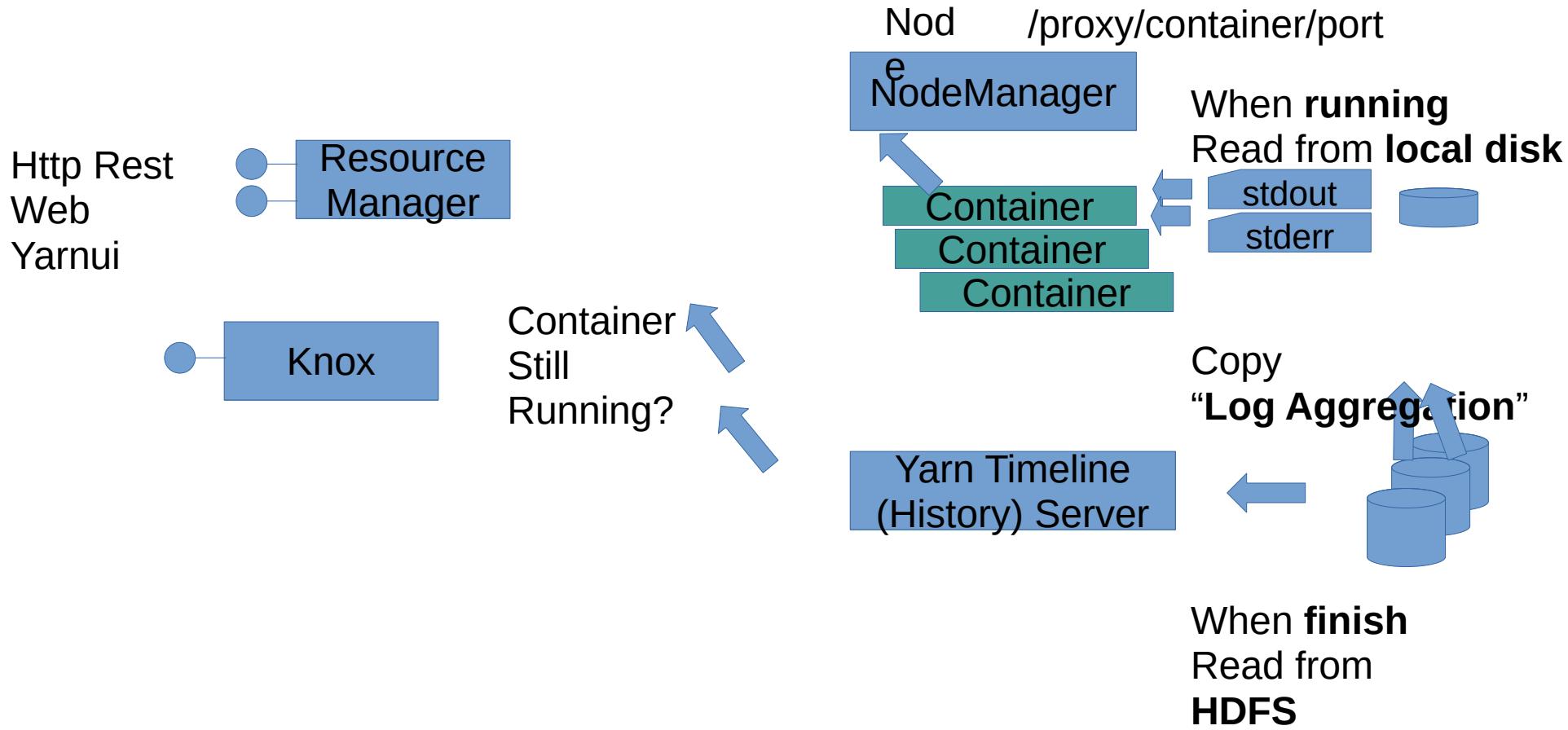
A screenshot of the Hadoop Yarn Web UI. At the top, there is a terminal window showing log output for a specific application. Below the terminal is a summary of the application's status. The main part of the screen is titled "All Applications" and displays a table of running applications with columns for ID, User, Name, Application Type, Queue, Start Time, Finish Time, State, Final Status, Running Containers, and Allocated CPU/Vcores. One row in the table is highlighted in yellow.



# Yarn UI ...



# Yarn Logs



# Oozie → Yarn UI → Yarn Logs

## 1/ Find Oozie workflow Id

## 2/ oozie-action

## 11/ stdout !!

The Hadoop logo consists of a yellow cartoon elephant icon followed by the word "hadoop" in a blue, lowercase, sans-serif font.

## Logs for

container 1428386215281 60485 01 00000

▪ <a href="#">ResourceManager</a>
▪ <a href="#">RM Home</a>
▪ <a href="#">NodeManager</a>
▪ <a href="#">Tools</a>

### **3/ Oozie shell action = “Mapper” of Map-Reduce in Yarn**

Action Info		Action Configuration
Name:	mr-node	
Type:	map-reduce	
Timestamp:		
Start Time:	Fri, 14 Oct 2011 07:56:16 GMT	
End Time:		
Status:	RUNNING	
Error Code:		
Error Message:		
External ID:	job_201110101107_0004	
External Status:	RUNNING	
Console URL:	<a href="http://localhost:5030/jobdetails.jsp?jobid=job_201110101107_0004">http://localhost:5030/jobdetails.jsp?jobid=job_201110101107_0004</a>	
Tracker URI:	localhost:9001	

**4|** Console URL =  
url in Yarn (Proxy or History)

The screenshot shows the Hadoop MapReduce Job History interface. At the top, it displays the job ID: job\_1381856870533\_0004. Below this, the job details are shown:

- Job Name:** Quicksort
- State:** RUNNING
- Progress:** 100%
- Started:** Tue Dec 11 13:29:29 EST 2012
- Ended:** (None)

On the right, there is a table titled "ApplicationMaster" with columns: Application Master, Start Time, Node ID, Log, and Block. The table shows one entry:

Application Master	Start Time	Node ID	Log	Block
Map 1	Tue Dec 11 13:29:29 EST 2012	(S0,B0)		

A red arrow points from the bottom left towards the "Block" column header.

## Example

71 spa

## 6/ Yarn (spark-submit client)

## 5/ Mapreduce (SHELL) ... call spark-submit --mode cluster

All Applications												
Cluster Metrics		All Applications										
Cluster	Nodes	Apps Submitted	Apps Running	Apps Completed	Containers Running	Memory Total	Memory Used	Memory Reserved	Vertices Used	Vertices Reserved	Active Nodes	Decon %
Above	0	568	0	0	0	0 B	0 B	32 GB	0 B	0	0	0
Below	0	0	0	0	0	0 B	0 B	0 B	0 B	0	0	0
Applications	0	0	0	0	0	0 B	0 B	0 B	0 B	0	0	0
NEW	0	0	0	0	0	0 B	0 B	0 B	0 B	0	0	0
SUBMITTED	0	0	0	0	0	0 B	0 B	0 B	0 B	0	0	0
PENDING	0	0	0	0	0	0 B	0 B	0 B	0 B	0	0	0
RUNNING	0	0	0	0	0	0 B	0 B	0 B	0 B	0	0	0
FAILED	0	0	0	0	0	0 B	0 B	0 B	0 B	0	0	0
SUCCEEDED	0	0	0	0	0	0 B	0 B	0 B	0 B	0	0	0
Scheduler	0	0	0	0	0	0 B	0 B	0 B	0 B	0	0	0
Tools	0	0	0	0	0	0 B	0 B	0 B	0 B	0	0	0
Show 20+ entries												
		ID	User	Name	Application Type	Queue	Start Time	Finish Time	State	Final Status	Running Containers	Allocated CPU/VCores
		application_1451372923882_0001	lma	infospark	SPARK	root/MRaa	Mon May 16 13:15:09 2016	Mon May 16 13:15:03 2016	FINISHED	SUCCEEDED	N/A	N/A
		application_1451372923882_0002	lma	infospark	SPARK	root/MRaa	Mon May 16 13:15:09 2016	Mon May 16 13:15:01 2016	FINISHED	SUCCEEDED	N/A	N/A
		application_1451372923882_0003	lma	infospark	SPARK	root/MRaa	Mon May 16 13:15:09 2016	Mon May 16 13:15:01 2016	FINISHED	SUCCEEDED	N/A	N/A
		application_1451372923882_0004	lma	infospark	SPARK	root/MRaa	Mon May 16 13:15:09 2016	Mon May 16 13:15:01 2016	FINISHED	SUCCEEDED	N/A	N/A
		application_1451372923882_0005	lma	infospark	SPARK	root/MRaa	Mon May 16 13:15:09 2016	Mon May 16 13:15:01 2016	FINISHED	SUCCEEDED	N/A	N/A
		application_1451372923882_0006	lma	infospark	SPARK	root/MRaa	Mon May 16 13:15:09 2016	Mon May 16 13:15:01 2016	FINISHED	SUCCEEDED	N/A	N/A
		application_1451372923882_0007	lma	infospark	SPARK	root/MRaa	Mon May 16 13:15:09 2016	Mon May 16 13:15:01 2016	FINISHED	SUCCEEDED	N/A	N/A
		application_1451372923882_0008	lma	infospark	SPARK	root/MRaa	Mon May 16 13:15:09 2016	Mon May 16 13:15:01 2016	FINISHED	SUCCEEDED	N/A	N/A
		application_1451372923882_0009	lma	infospark	SPARK	root/MRaa	Mon May 16 13:15:09 2016	Mon May 16 13:15:01 2016	FINISHED	SUCCEEDED	N/A	N/A
		application_1451372923882_0010	lma	infospark	SPARK	root/MRaa	Mon May 16 13:15:09 2016	Mon May 16 13:15:01 2016	FINISHED	SUCCEEDED	N/A	N/A
		application_1451372923882_0011	lma	infospark	SPARK	root/MRaa	Mon May 16 13:15:09 2016	Mon May 16 13:15:01 2016	FINISHED	SUCCEEDED	N/A	N/A
		application_1451372923882_0012	lma	infospark	SPARK	root/MRaa	Mon May 16 13:15:09 2016	Mon May 16 13:15:01 2016	FINISHED	SUCCEEDED	N/A	N/A
		application_1451372923882_0013	lma	infospark	SPARK	root/MRaa	Mon May 16 13:15:09 2016	Mon May 16 13:15:01 2016	FINISHED	SUCCEEDED	N/A	N/A
		application_1451372923882_0014	lma	infospark	SPARK	root/MRaa	Mon May 16 13:15:09 2016	Mon May 16 13:15:01 2016	FINISHED	SUCCEEDED	N/A	N/A
		application_1451372923882_0015	lma	infospark	SPARK	root/MRaa	Mon May 16 13:15:09 2016	Mon May 16 13:15:01 2016	FINISHED	SUCCEEDED	N/A	N/A
		application_1451372923882_0016	lma	infospark	SPARK	root/MRaa	Mon May 16 13:15:09 2016	Mon May 16 13:15:01 2016	FINISHED	SUCCEEDED	N/A	N/A
		application_1451372923882_0017	lma	infospark	SPARK	root/MRaa	Mon May 16 13:15:09 2016	Mon May 16 13:15:01 2016	FINISHED	SUCCEEDED	N/A	N/A
		application_1451372923882_0018	lma	infospark	SPARK	root/MRaa	Mon May 16 13:15:09 2016	Mon May 16 13:15:01 2016	FINISHED	SUCCEEDED	N/A	N/A
		application_1451372923882_0019	lma	infospark	SPARK	root/MRaa	Mon May 16 13:15:09 2016	Mon May 16 13:15:01 2016	FINISHED	SUCCEEDED	N/A	N/A
		application_1451372923882_0020	lma	infospark	SPARK	root/MRaa	Mon May 16 13:15:09 2016	Mon May 16 13:15:01 2016	FINISHED	SUCCEEDED	N/A	N/A

Application application 1455000259206 0001

The screenshot shows the Apache Ambari interface for managing Hadoop applications. On the left, a sidebar lists cluster status, node labels, and application types like New, Saving, Submitted, Accepted, Running, Finished, Failed, and Killed. The main area has tabs for Kill Application, Application Overview, and Application Metrics.

**Application Overview**

User:	root
Name:	select count(*) from server.logs(Stage-1)
Application Type:	MAPREDUCE
Application Tag:	
Final Status Reported by AM:	ACCEPTED: waiting for AM container to be allocated, launched and register with RM.
Application has not completed yet.	
Started:	Tue Feb 09 07:35:26 +0000 2016
Elapsed:	50sec
Tracking URL:	ApplicationMaster
Diagnostics:	

**Application Metrics**

Total Resource Preempted:	<memory:0, vCores:0>
Total Number of Non-AM Containers Preempted:	0
Total Number of AM Containers Preempted:	0
Resource Preempted from Current Attempt:	<memory:0, vCores:0>
Number of Non-AM Containers Preempted from Current Attempt:	0
Aggregate Resource Allocation:	0 MB-seconds, 0 vcore-seconds

**Logs**

Show:	zo_ entries	Attempts	Started	Node	Logs	Blacklisted Nodes
appattempt_1455000259206_0001_000001	0	0	Feb 9 07:35:26 +0000 2016	http://	Logs 0	0

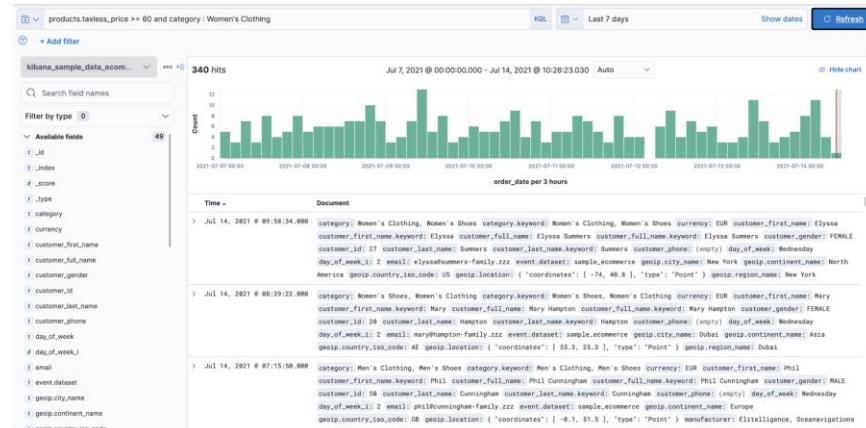
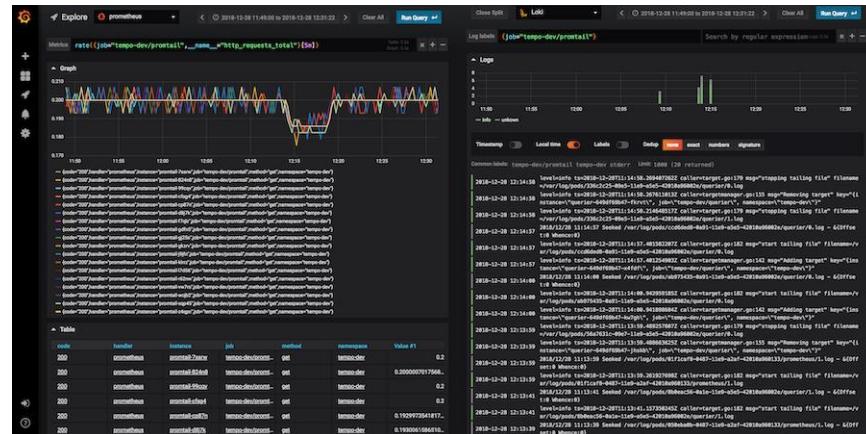
Showing 1 to 1 of 1 entries

First Previous 1 Next Last

## 8/ appmaster application attempt 1

## 9/ application (example: Reduce for shell)

# Yarn Log Alternatives...



# Yarn Summary / Alternative(s)

Mesos was deprecated

... Yarn is also (complex /

inefficient)



**kubernetes**

Queue  
S +



or



YUNIKORN

Log  
S +



or



Operator  
S +

# **Yarn Missing Features... Orchestrator + DataLineage**

**No “CRON” (periodic scheduling, like 2 \*/5 MON-FRI \* \* \*)**

**No simple packaging to launch**

**No trigger after data change**

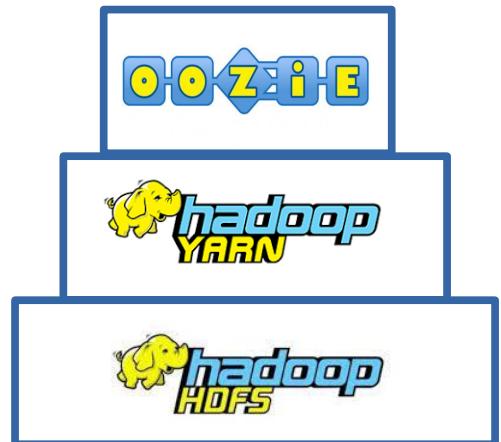
**No workflow between data / job**

# Apache Oozie



# Oozie Features simpler than / missing in Yarn

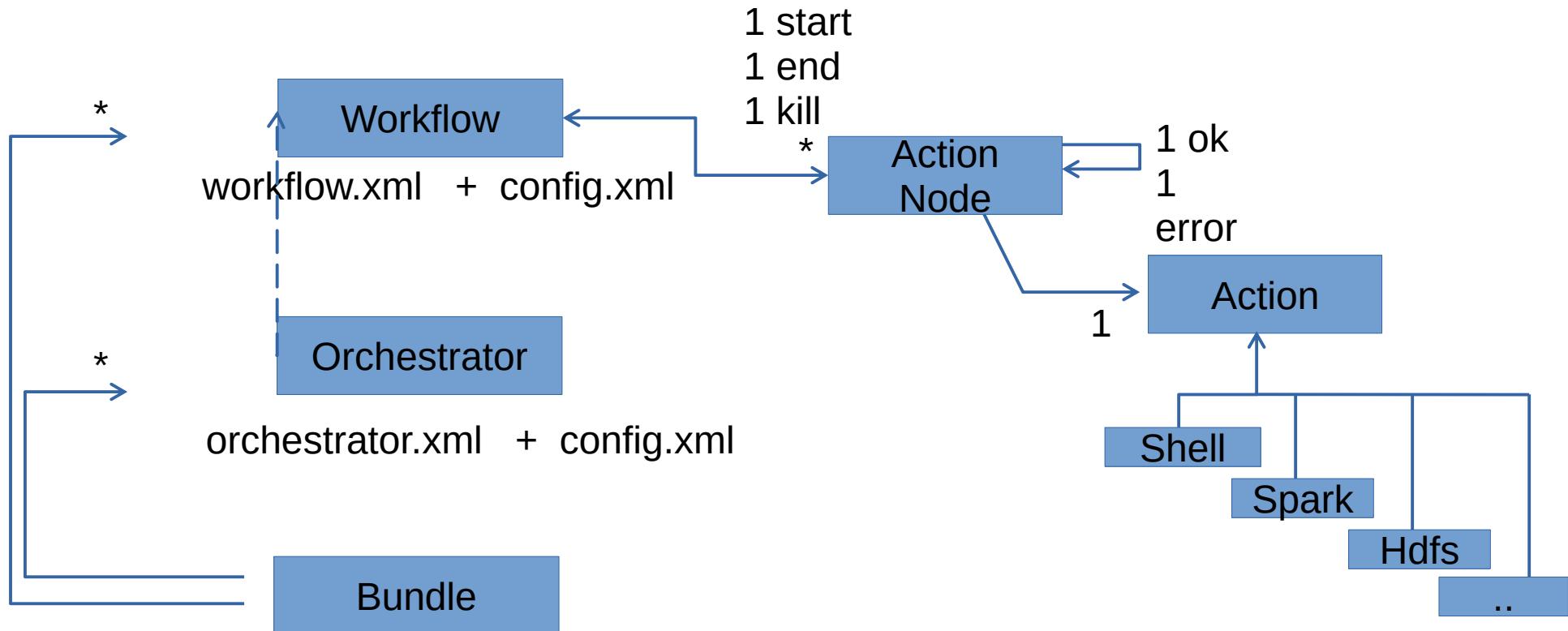
Orchestrator: “CRON” (periodic scheduling, like 2 \*/5 MON-FRI \* \* \*)



Workflow.xml    <action><shell> <command><arg><env>.. <file>  
**simple packaging to launch**

**workflow between action**

# Oozie Model



# workflow.xml

```
<workflow-app name="[WF-DEF-NAME]" xmlns="uri:oozie:workflow:0.3">
  <start to="[NODE-NAME]"/>
  <end name="end"/>
  <action name="[NODE-NAME]">
    <shell xmlns="uri:oozie:shell-action:0.1">
      <job-tracker>[JOB-TRACKER]</job-tracker>
      <name-node>[NAME-NODE]</name-node>
      <prepare>
        <delete path="[PATH]"/><mkdir path="[PATH]"/>
      </prepare>
      <job-xml>[SHELL SETTINGS FILE]</job-xml>
      <configuration>
        <property>
          <name>[PROPERTY-NAME]</name><value>[PROPERTY-VALUE]</value>
        </property>
      </configuration>
      <exec>[SHELL-COMMAND]</exec>
      <argument>[ARG-VALUE]</argument>
      ...
      <env-var>[VAR1=VALUE1]</env-var>
      ...
      <file>[FILE-PATH]</file>
      ...
    </shell>
    <ok to="[NODE-NAME]"/>
    <error to="[NODE-NAME]"/>
  </action>
```

# Config .xml / .properties

All  **\${variable}**  in coordinator.xml / workflow.xml are  
**replaced**

When launching

```
$ oozie job -config config.properties -start
```

or

```
$ curl -X POST http://oozie/job?action=start -d  
@config.properties
```

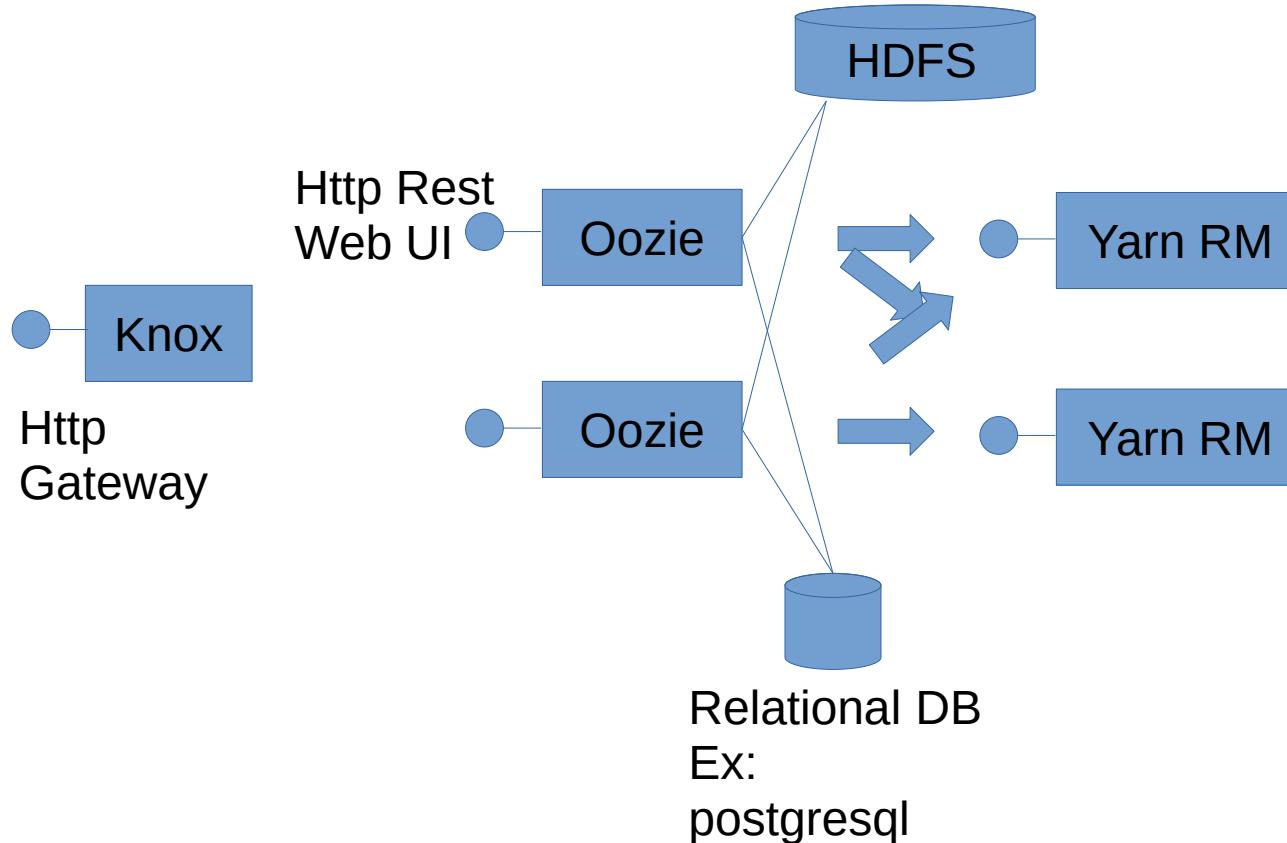
**oozie.wf.application.path= hdfs://a/b/workflow.xml**

user.name=john.smith

variable=xx

otherVariable=yy

# Oozie Architecture



# Oozie Command Line

```
$ oozie job -start -config <config.properties>  
$ oozie job -status <jobId>  
$ oozie job -kill <jobId>
```

```
$ oozie jobs -filter user=<U>\;name=<N>\;group=<G>\;status=<S>\;frequency=<F>\;startcreatedtime=<SC>\;endcreatedtime=<EC>\;sortBy=<SB> ...
```

```
$ oozie coordinator -start -config <config.properties>  
$ oozie coordinator -suspend <coordId>  
$ oozie coordinator -resume <coordId>  
$ oozie coordinator -kill <coordId>  
$ oozie coordinator -change <coordId> -value key=value
```

# Oozie Web UI

## Coordinator

The screenshot shows the Oozie Coordinator interface. At the top, there's a navigation bar with links for Documentation, Job Info, Job Definition, Job Configuration, Job Log, Job Error Log, Job Audit Log, and Job DAG. Below this, a detailed job summary is displayed:

Job Id:	0000001-170219172252198-oozie-oozi-W
Name:	OnPrem-HDFS-to-Databricks-ETL-job
App Path:	hdfs://sandbox.hortonworks.com:8020/user/apps/databricks/curl/shell/wf/
Run:	0
Status:	SUCCEEDED
User:	root
Group:	
Parent Coord:	0000000-170219172252198-oozie-oozi-C@1
Create Time:	Sun, 19 Feb 2017 17:26:17 GMT
Start Time:	Sun, 19 Feb 2017 17:26:17 GMT
Last Modified:	Sun, 19 Feb 2017 17:27:56 GMT
End Time:	Sun, 19 Feb 2017 17:27:56 GMT

Below the summary is a table titled "Actions" showing the execution details:

Action Id	Name	Type	Status
1 0000001-170219172252198-oozie-oozi-W@:start	:start	:START	OK
2 0000001-170219172252198-oozie-oozi-W@ACTION01-H...	ACTION01-HDFS-to-S3-File-Transfer	distcp	OK
3 0000001-170219172252198-oozie-oozi-W@ACTION02-R...	ACTION02-Run-ETL-in-Databricks	shell	OK
4 0000001-170219172252198-oozie-oozi-W@ACTION03-T...	ACTION03-Transfer-S3-to-OnPrem-HDFS	distcp	OK
5 0000001-170219172252198-oozie-oozi-W@ACTION04-C...	ACTION04-Create-WF-Success-Trigger	fs	OK
6 0000001-170219172252198-oozie-oozi-W@end	end	END	OK
7 0000001-170219172252198-oozie-oozi-W@sendEmailS...	sendEmailSuccess	email	OK

Coord executions  
= launched  
workflows

The screenshot shows the Oozie Workflow interface. At the top, there's a navigation bar with links for Documentation, Job Info, Job Definition, Job Configuration, Job Log, Job Error Log, Job Audit Log, and Job DAG. Below this, a detailed job summary is displayed:

Job Id:	0000001-170415112256550-oozie-serg-W
Name:	WorkflowWithSqoopAction
App Path:	hdfs://localhost:9000/oozieProject/workflowSqoopAction
Run:	3
Status:	FAILED
User:	sergio
Group:	
Parent Coord:	
Create Time:	Sat, 15 Apr 2017 12:14:30 GMT
Start Time:	Sat, 15 Apr 2017 12:49:59 GMT
Last Modified:	Sat, 15 Apr 2017 12:50:01 GMT
End Time:	Sat, 15 Apr 2017 12:50:01 GMT

Below the summary is a table titled "Actions" showing the execution details:

Action Id	Name	Type	Status
1 0000001-170415112256550-oozie-serg-W@:start	:start	:START	OK
2 0000001-170415112256550-oozie-serg-W@sqoopAction	sqoopAction	sqoop	FAILED

Workflow  
actions

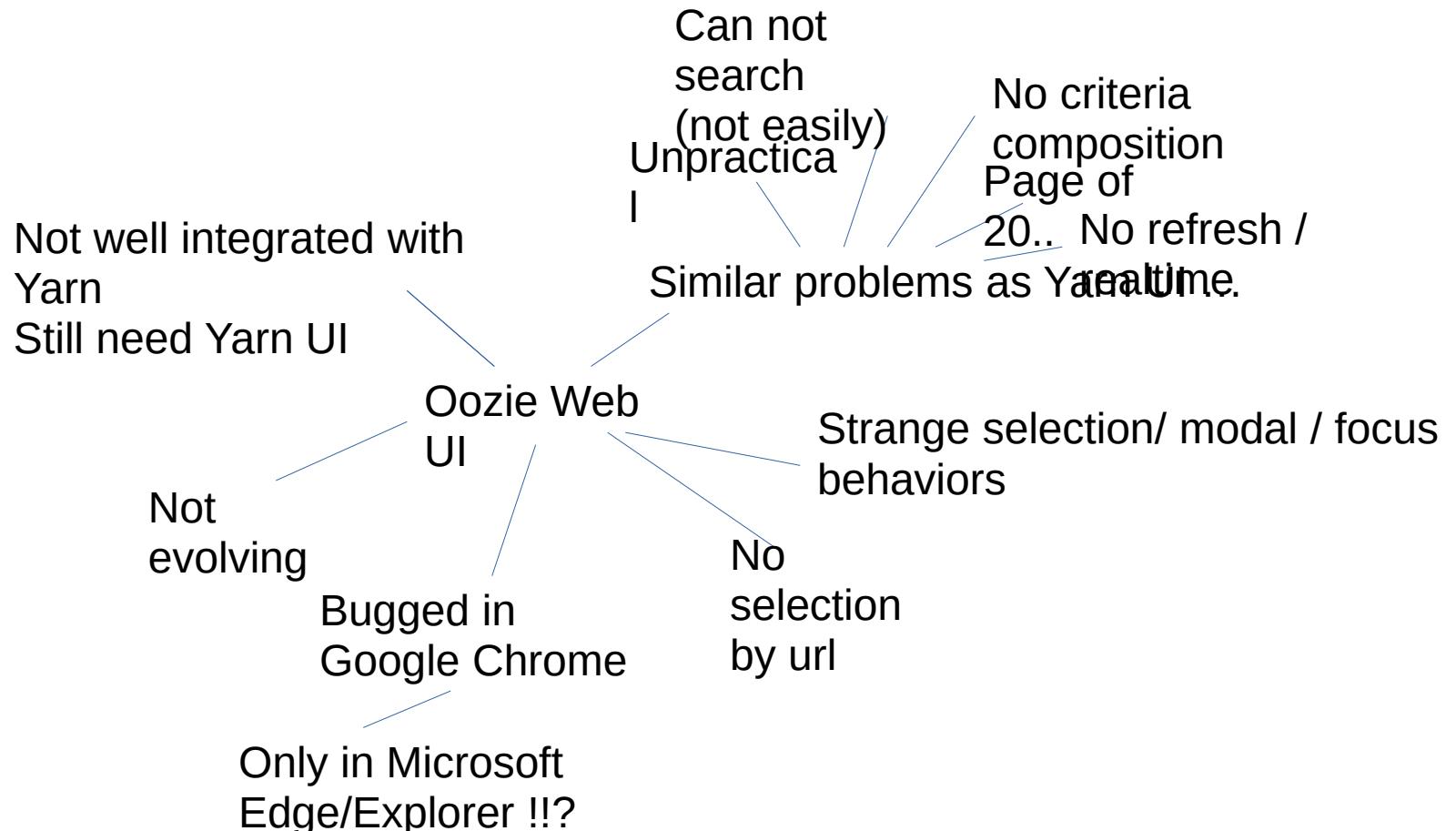
## Workflow

Action = yarn job

The screenshot shows the "Action" configuration dialog for a shell node. The action details are as follows:

Name:	shell-node
Type:	shell
Transition:	
Start Time:	Sun, 03 Apr 2016 19:37:15 GMT
End Time:	
Status:	RUNNING
Error Code:	
Error Message:	
External ID:	job_1459712194216_0002
External Status:	RUNNING
Console URL:	http://sandbox.hortonworks.com:8088/proxy/application_1459712194/
Tracker URI:	sandbox.hortonworks.com:8050

# Oozie Web UI ...



# Oozie Summary

OK to fill missing Yarn  
to simplify launching Yarn  
containers

Simple http Rest / Command line  
tool

Verbose yet simple xml +  
properties

Unpractical Web  
UI

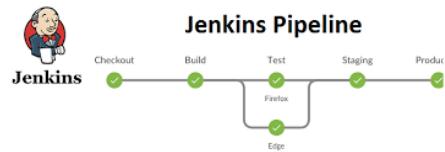
Still  
No Datalineage between data /  
job  
No trigger on data change

# Oozie Alternatives

**Livy** .... to start simple Spark actions



**Any Orchestrator** .... with plugin extension for Yarn support



Etc..

Etc..

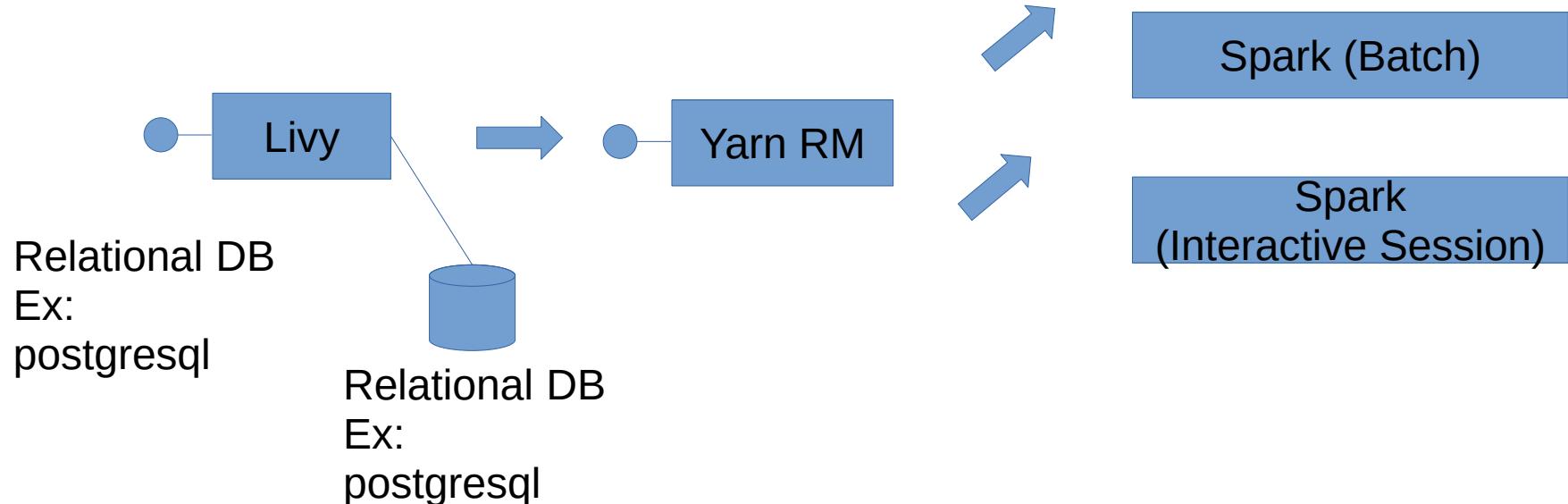
Etc..



# Apache Livy



# Livy Architecture + Features



# Livy sample

file:

app.json

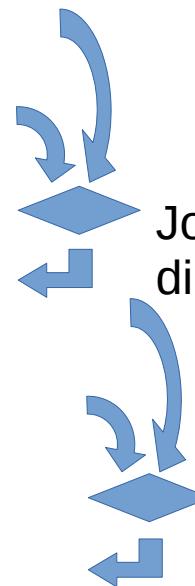
```
"file": "hdfs:///myapps/hello-spark.jar",
"className": "myapp.HelloSpark",
"args": ["hdfs:///inputs"],
"jars": [
  "hdfs:///myapps/mylib.jar"
],
"driverMemory": "1g",
"executorMemory": "5g",
"executorCores": 8,
"numExecutors": 2
}
```

```
$ curl -X POST -H 'Content-Type: application/json'
 \
  http://livy:8998/batches -d @app.json | jq '.'
batchId=..

$ curl http://livy:8998/batches | jq '.'

$ curl http://livy:8998/batches/${batchId}/log
```

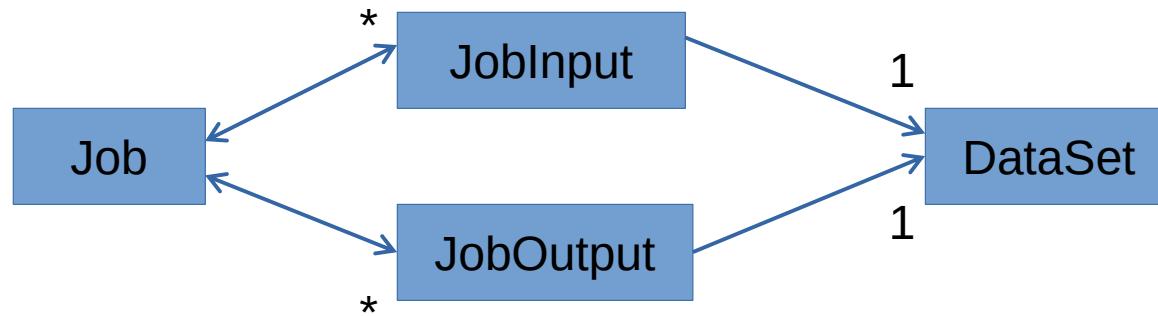
# Yarn+Oozie+Livy... Missing Feature: DataLineage, Data Trigger Orchestration



Job1 to compute data:  $\text{dir3} = \text{func1}(\text{dir1}, \text{dir2})$

Job2 to compute data:  $\text{dir5} = \text{func2}(\text{dir3}, \text{dir4})$

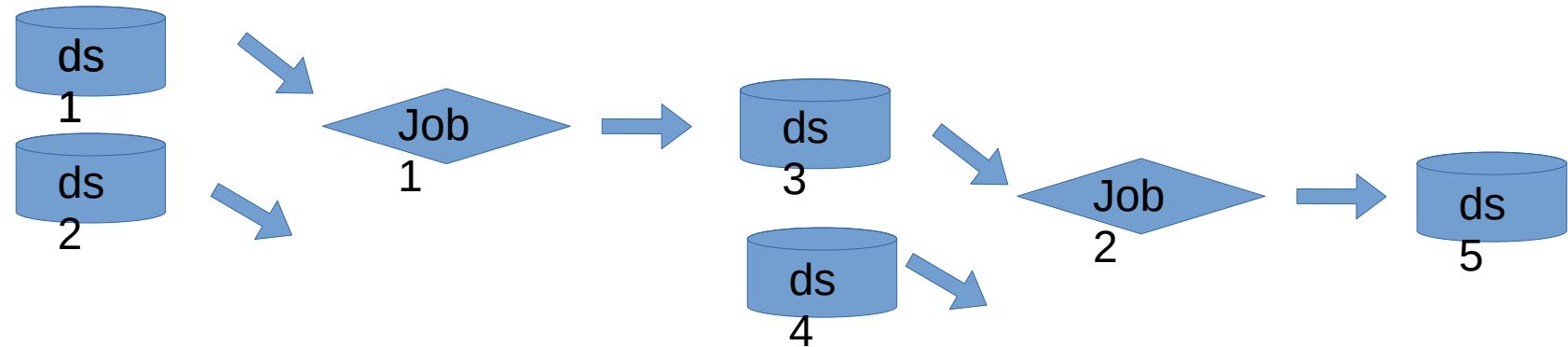
# Job (Input, Output) re-eval on input change



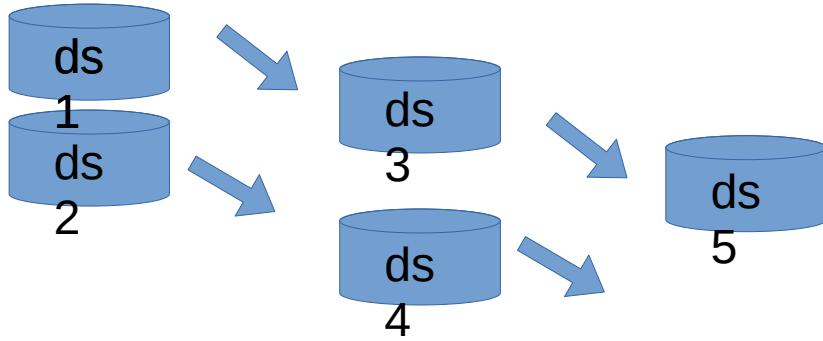
When detecting external change to DataSet => search for JobInputs => launch Jobs

When Job finish => JobOutput changed => launch successor Jobs for inputs

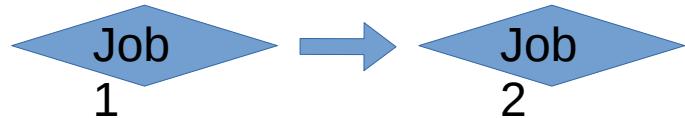
# DataLineage DataSet+Job



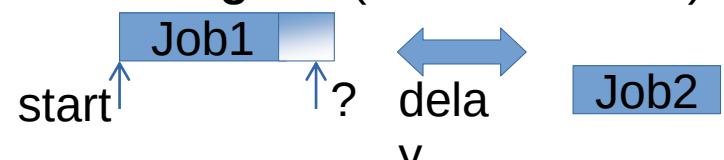
Sub graph : Data dependency



Sub graph : Job orchestration

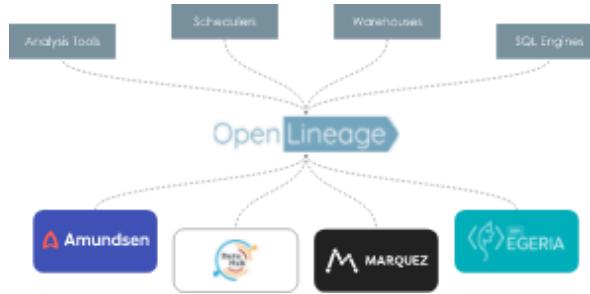


Gant Diagram (cron + duration)

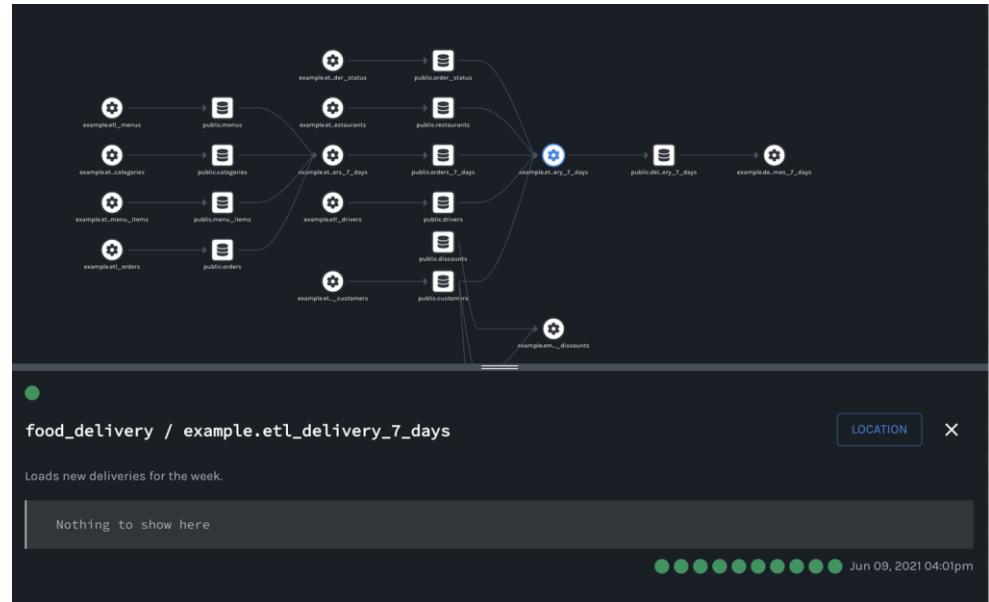


# Example DataLineage Spec / Implementation OpenLineage / Marquez

OpenLineage  
<https://openlineage.io/>



 MARQUEZ  
<https://marquezproject.github.io/>



```
$ curl -X POST http://openlineage/api/v1/lineage \
-H 'Content-Type: application/json' \
-d '{
  "eventType": "START",
  "eventTime": "2020-12-28T19:52:00.001+10:00",
  "run": { "runId": "d46e465b-d358-4d32-83d4-df660ff614dd" },
  "job": { "namespace": "my-namespace", "name": "my-job" },
  "inputs": [ { "namespace": "my-namespace", "name": "my-input" }]
}'
```

to be continued ...

Part5: High-Level focus  
HiveMetaStore, Parquet, Spark