

# Hadoop Ecosystem Main Components Explained

## Hadoop is Dead - Long life Spark+ Kubernetes

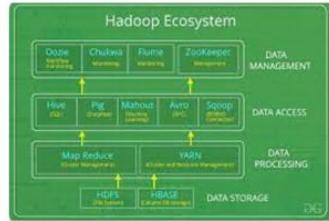
course 2024  
arnaud.nauwynck@gmail.com @All Rights reserved

this document:  
[http://github.com/arnaud-nauwynck/Presentations/big-data/  
4-hadoop-ecosystem](http://github.com/arnaud-nauwynck/Presentations/big-data/4-hadoop-ecosystem)

# Outline

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

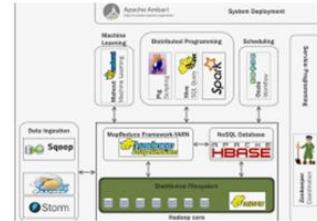
# Google image "Hadoop ecosystem"



Hadoop Ecosystem - GeeksforGeeks  
geeksforgeeks.org



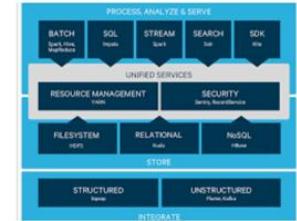
Apache Hadoop Ecosystem | Download .  
researchgate.net



The Hadoop ecosystem | Hadoop Essentials  
subscription.packtpub.com

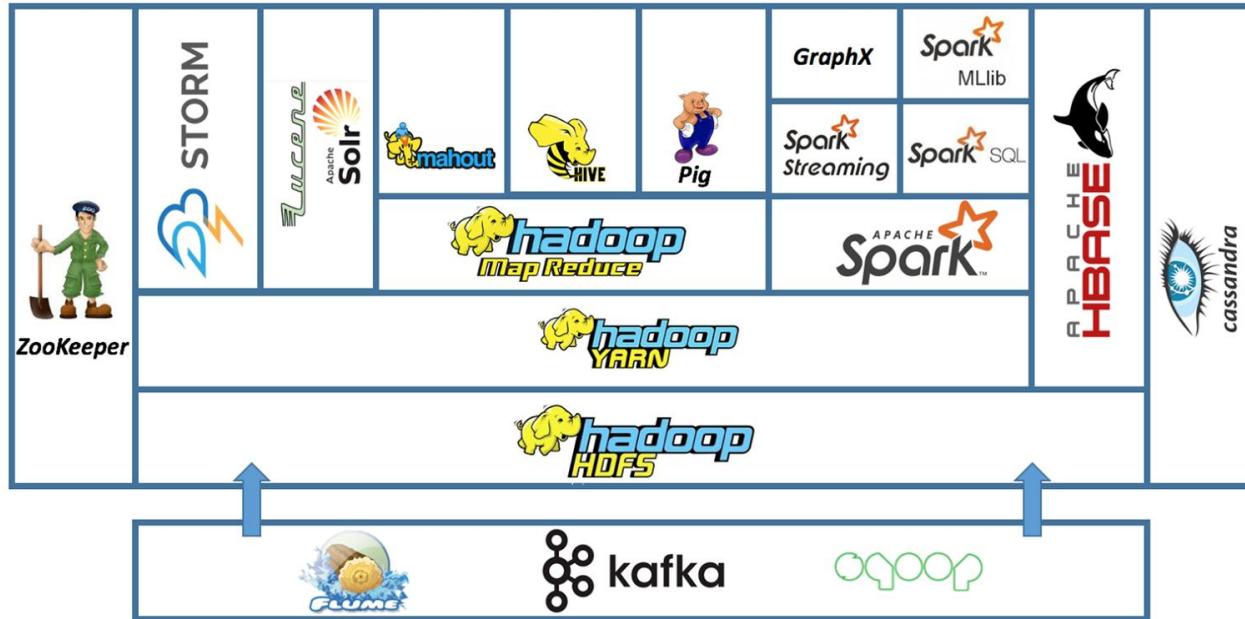


Overview of the Hadoop ecosystem ..  
oreilly.com



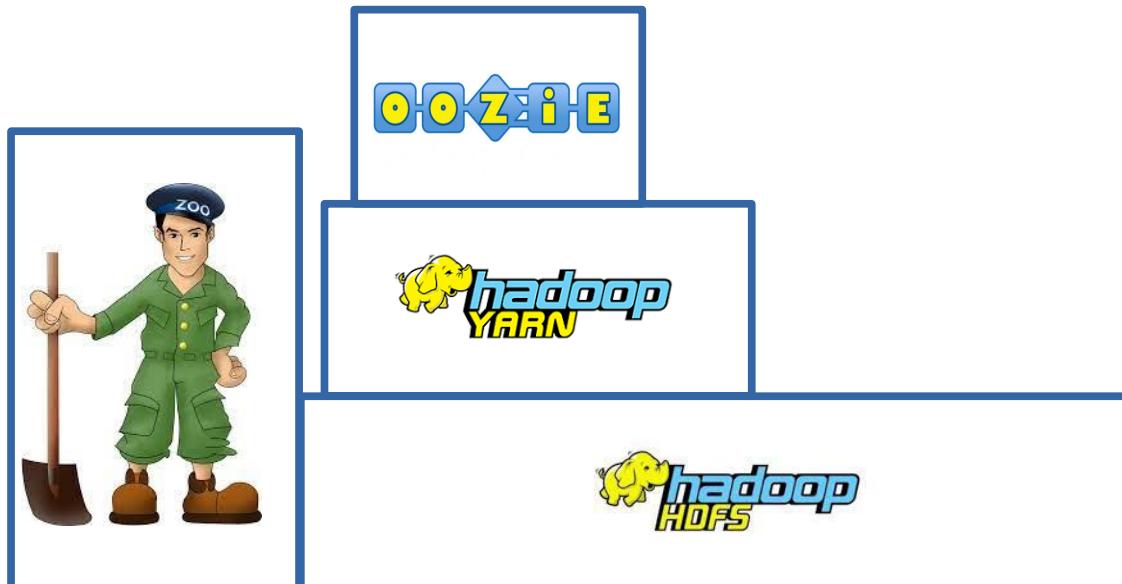
Apache Hadoop open source ecosyste...  
cloudera.com

# Hadoop Ecosystem..



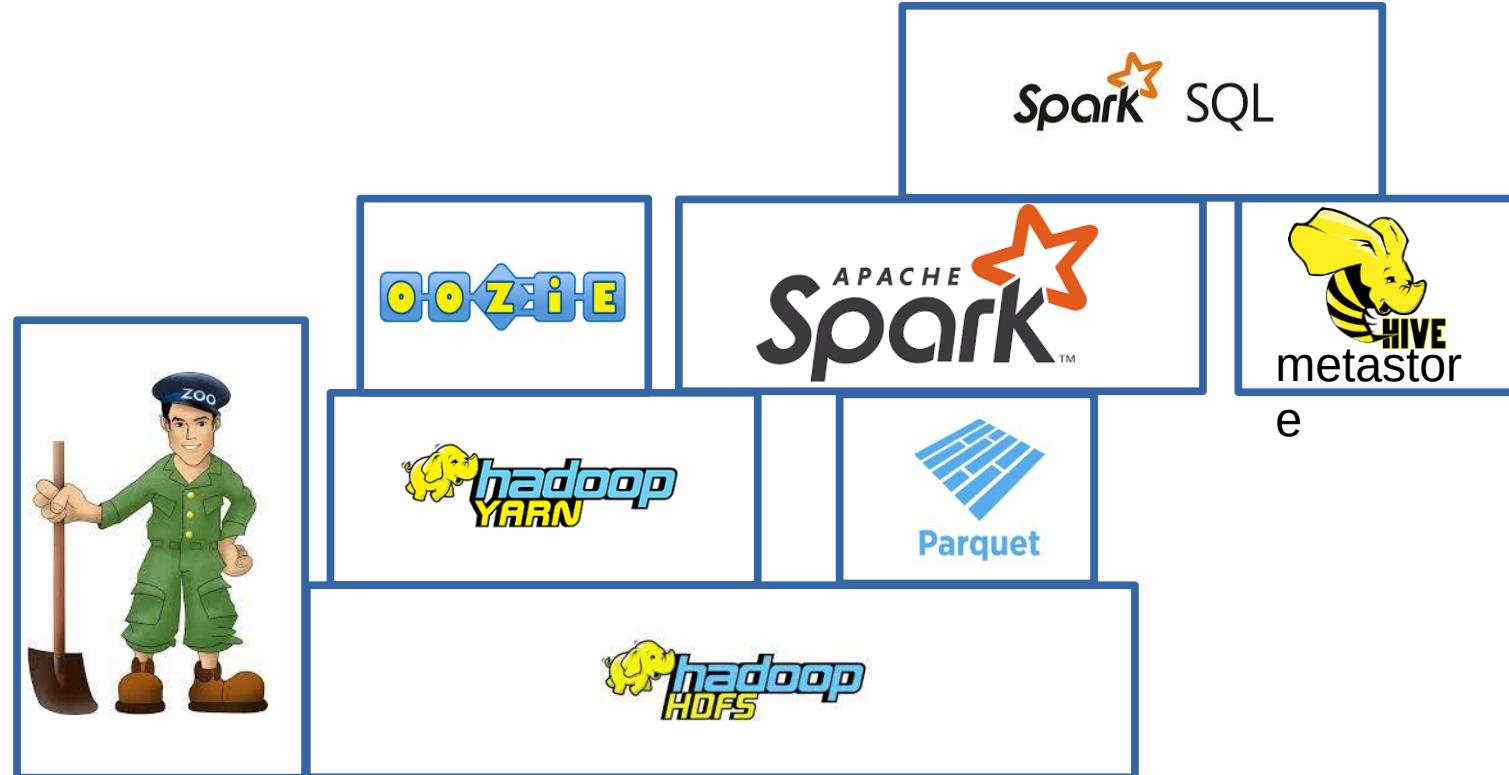
# Low-Level Focus

## 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]

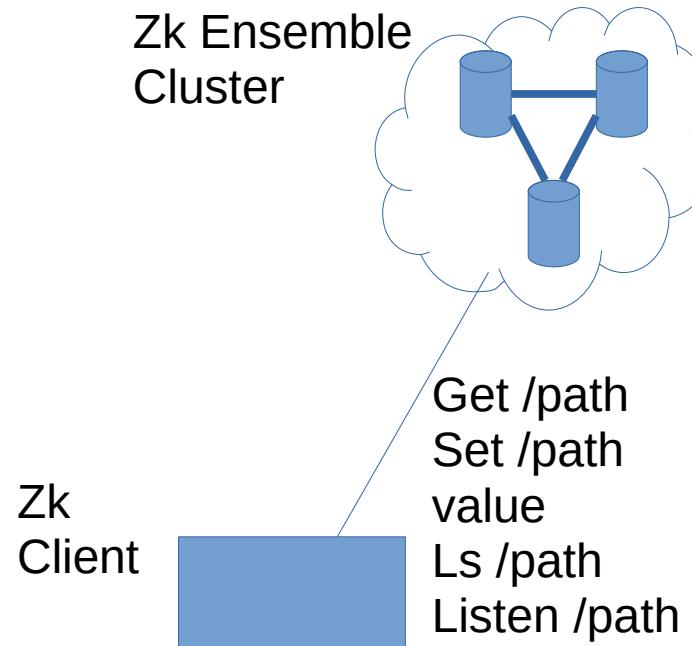
- 1 Overview
- 2 Architecture
- 3 Use cases
- 4 Client libraries
- 5 Apache projects using ZooKeeper
- 6 See also
- 7 References
- 8 External links

### Apache ZooKeeper



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

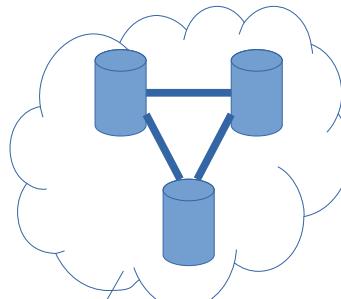
...

# 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 »

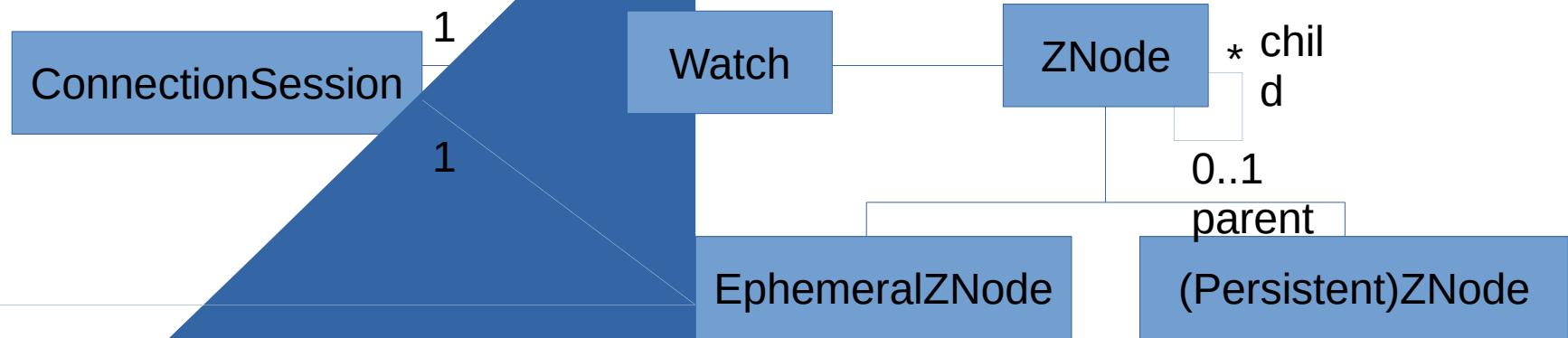
EphemeralZNode « node1»

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)

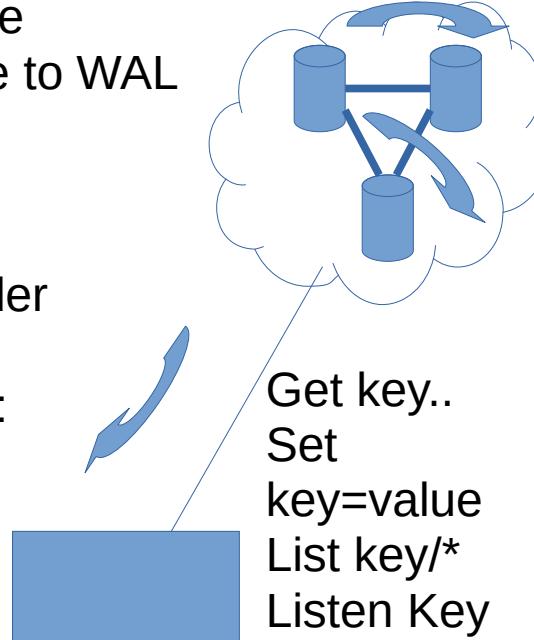
# Zk Quorum Write

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

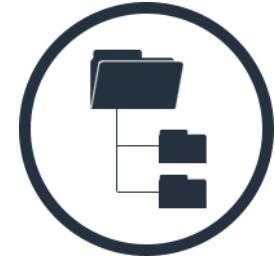
# Hadoop Distributed File System



# HDFS name explained

A **FileSystem** for directories and Files

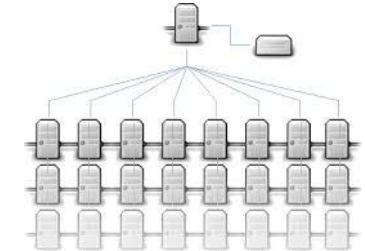
Like NTFS(windows), Ext4(linux), Ceph(distributed) ...



**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..

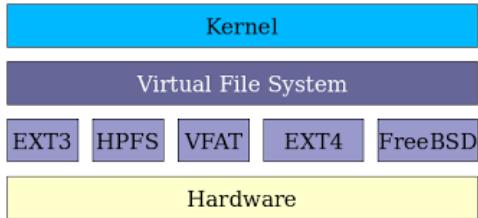
implemented by AWS,Azure..

Abstract class  
**FileSystem**

class  
**DistributedFileSystem**

# Abstract API / Concrete Implementation

## Comparison with Linux



API

Implementations

```
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 {
```

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

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

# API

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

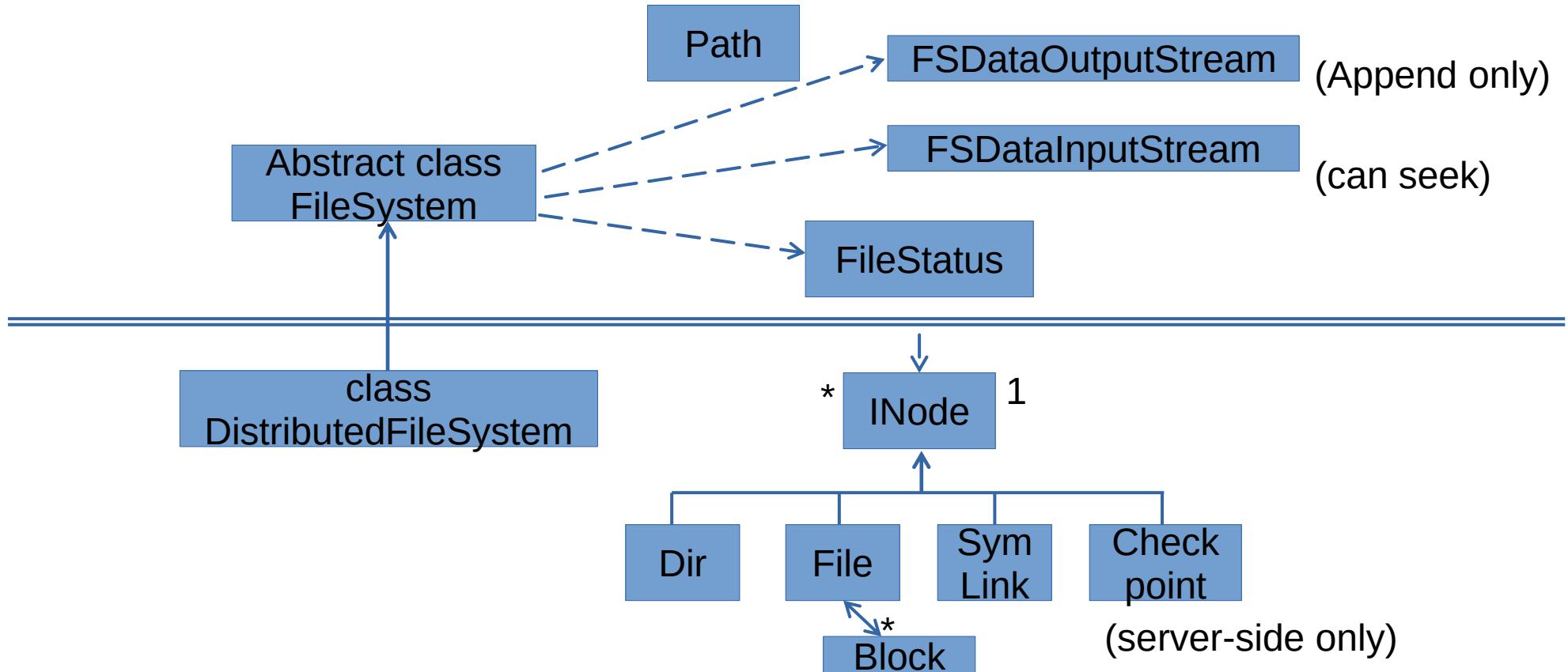
For file: **Append ONLY FileSystem**

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

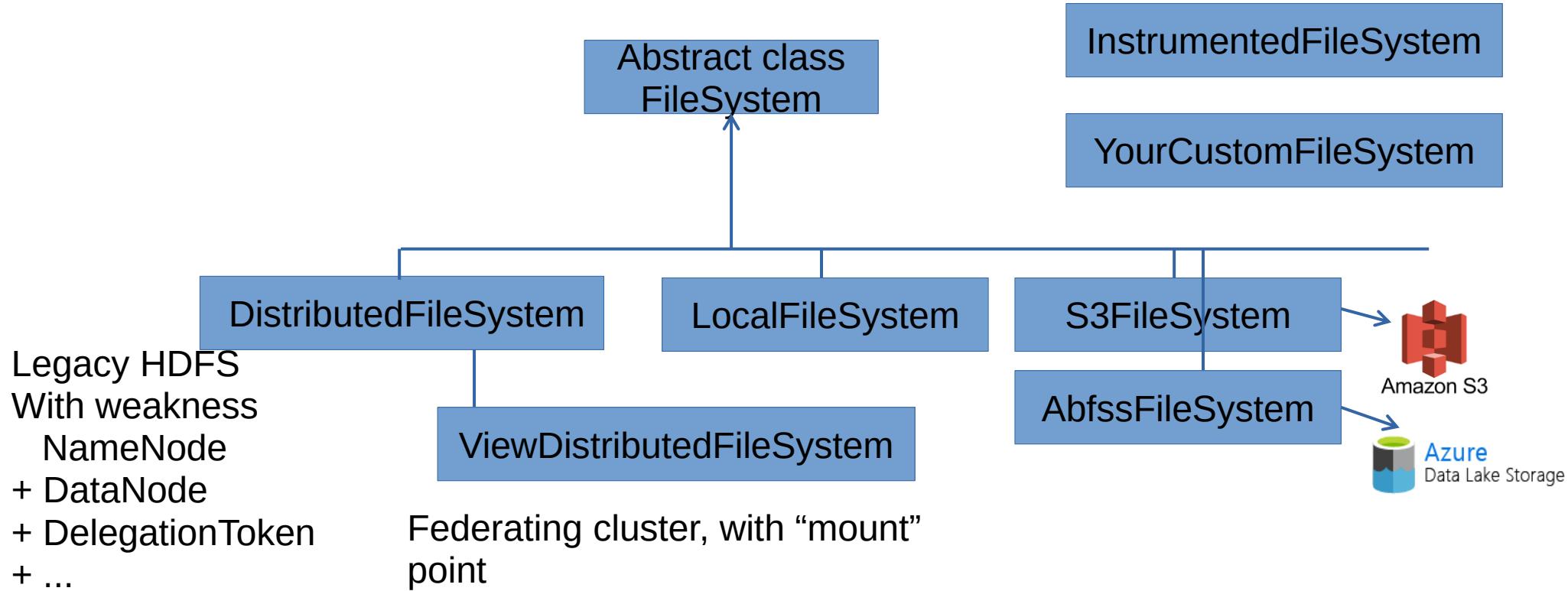
**Seek possible only in Read mode**

- `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**

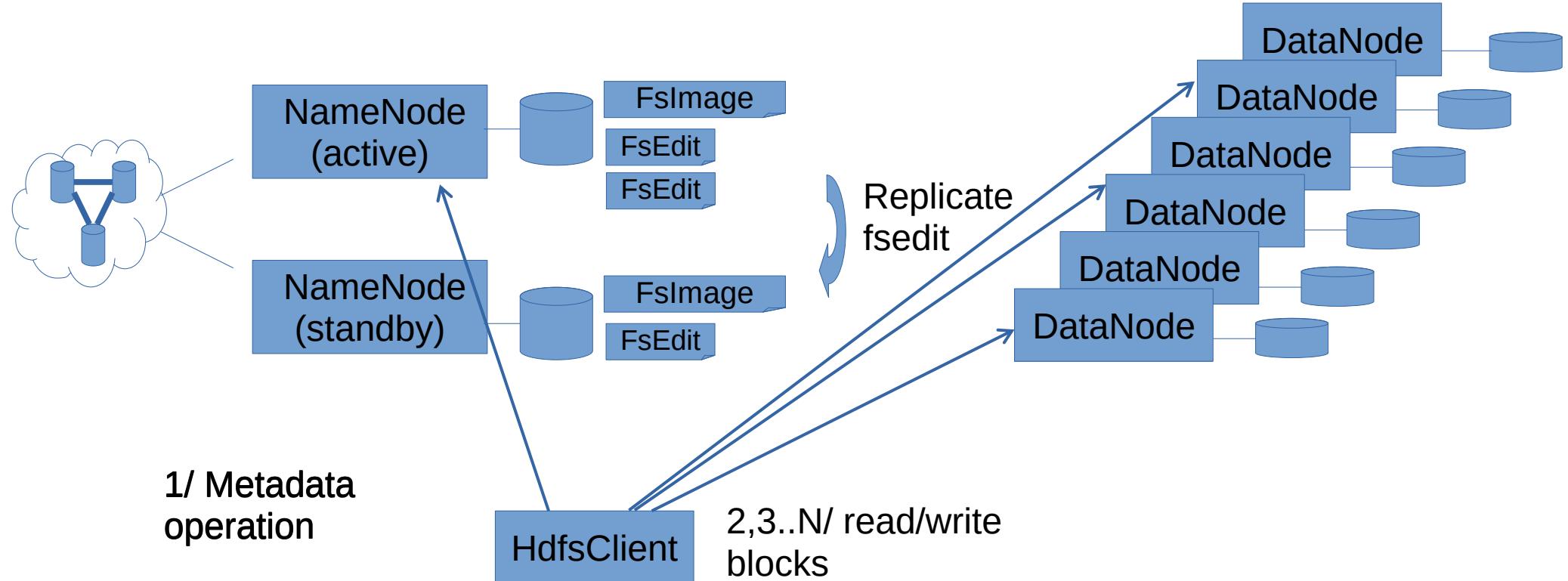
**Huge RAM + Disk needed**

**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

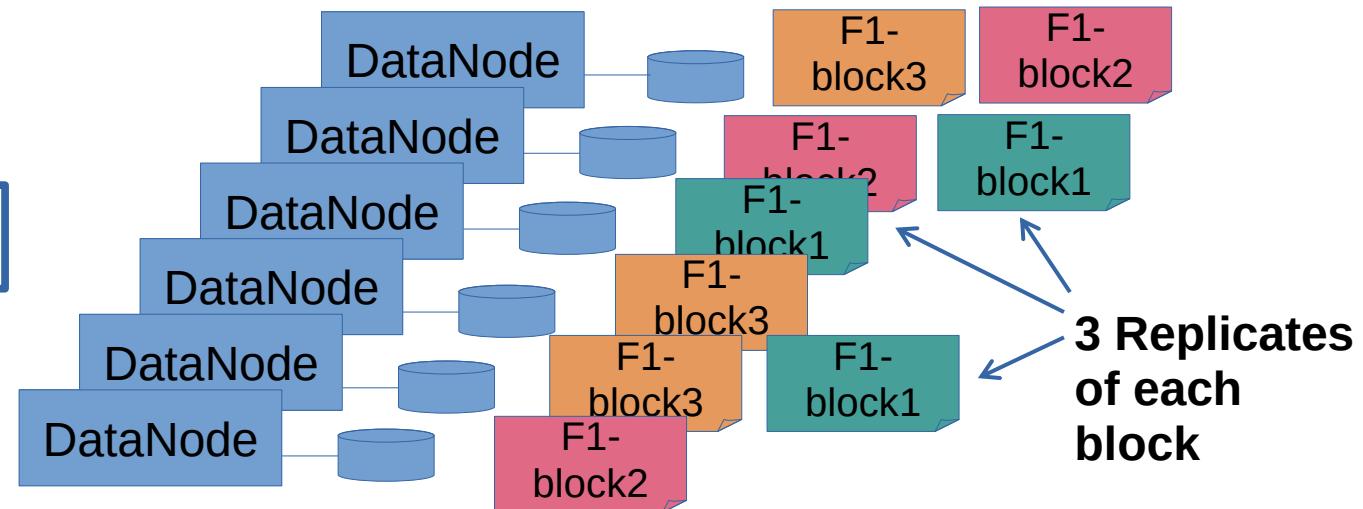


# DataNode ... store blocks

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

Logical File  
metadata

"F1" = 3 blocks



Physical Block repartition on DataNodes

# RF (Replication Factor) = 3

active NameNode manage replications:

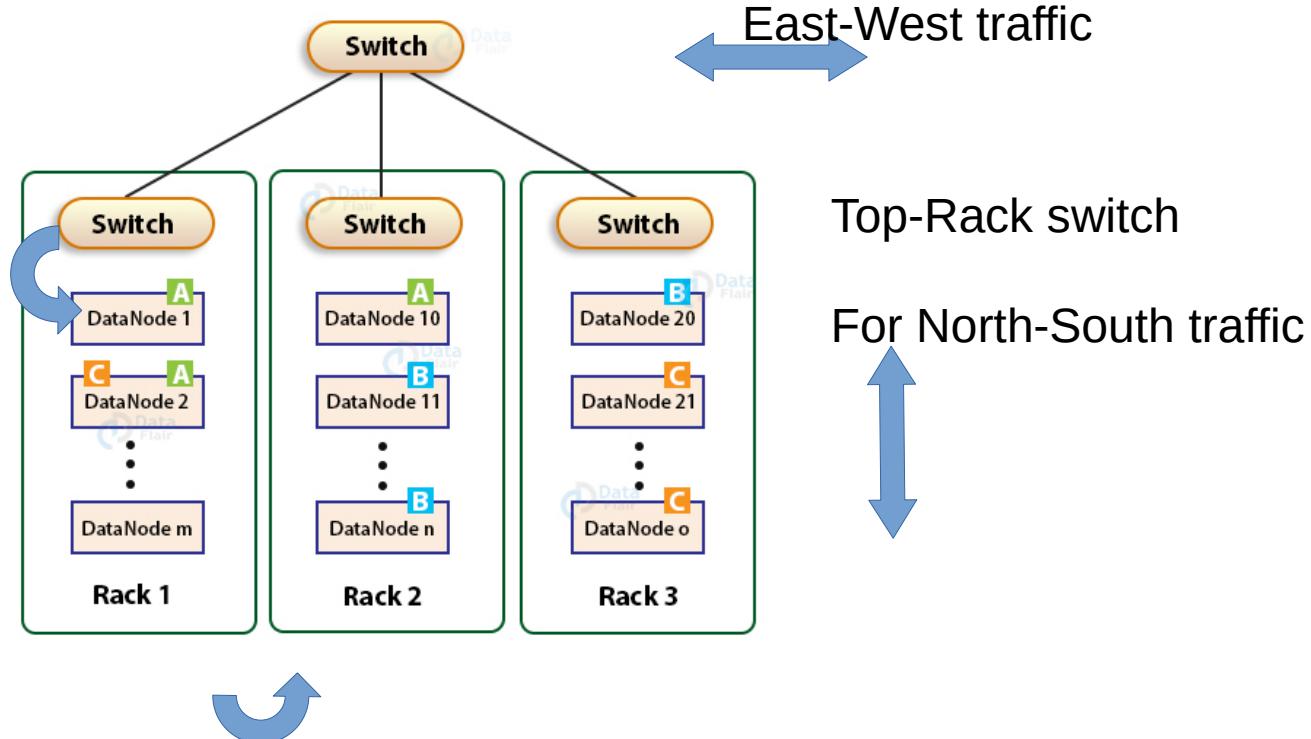
When a DataNode “disappears” => all its blocks are lost (blocks are under replicated)  
=> need re-replicate corresponding blocks

When a DataNode “joins” cluster => it sends its “block-report”  
=> blocks might be over-replicated => can be deleted

( at startup, NameNode is in “safe mode”...  
it waits enough block-reports to start useless replicating ! )

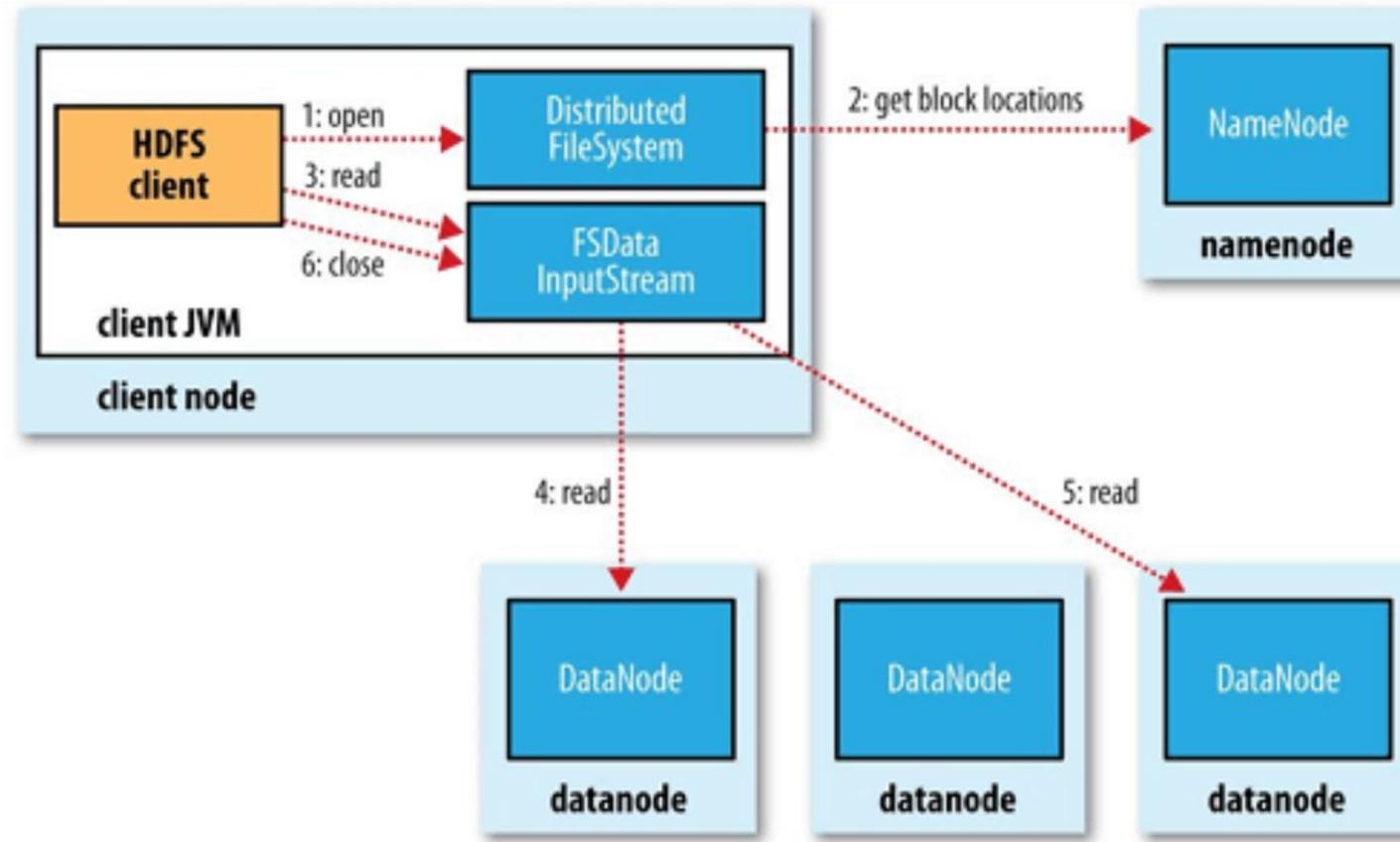
# RF ... Rack Aware

Can replicate 1  
in same Rack  
(fast)

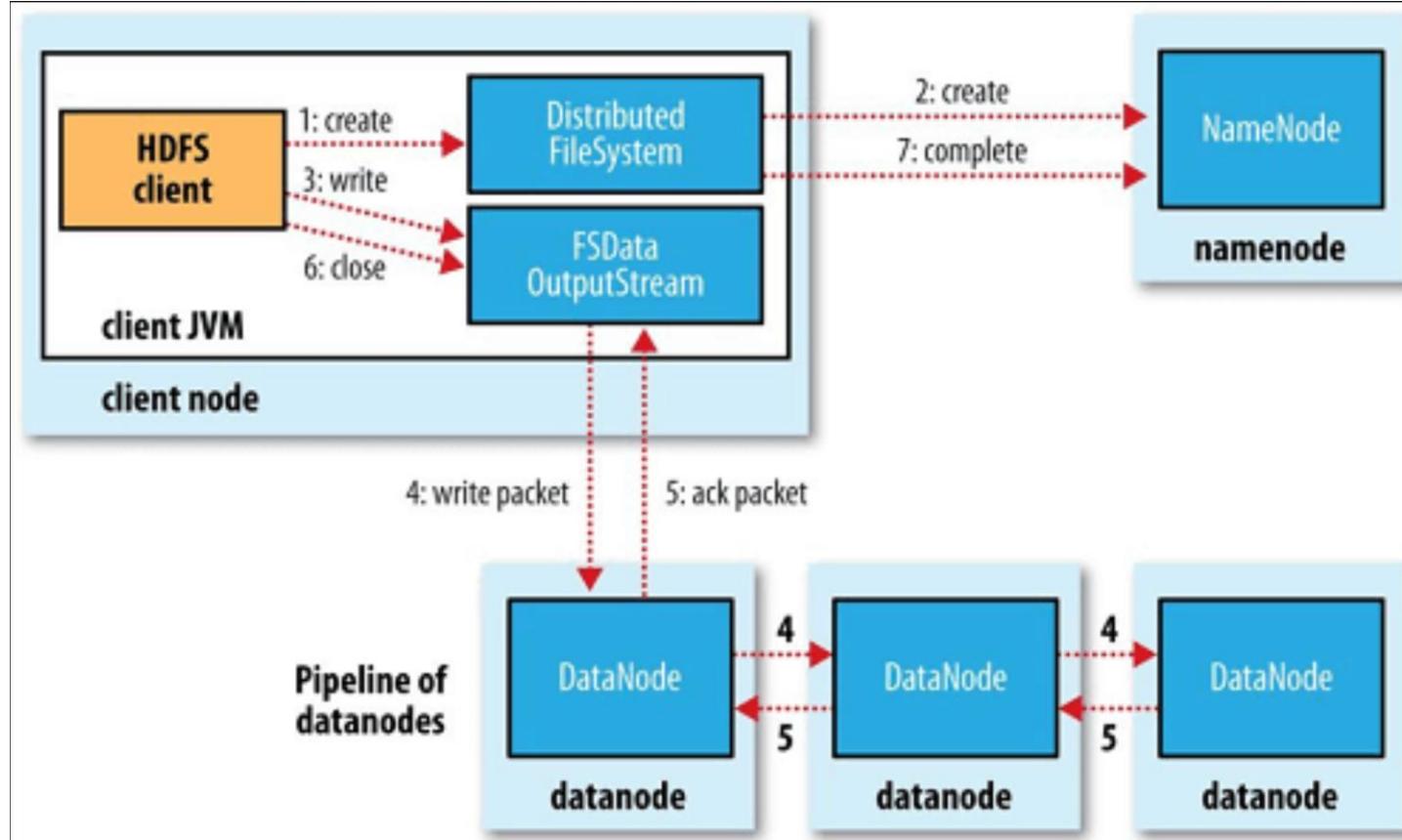


Must replicate  $\geq 1$   
in another Rack

# Reading a HDFS File



# Writing a HDFS File

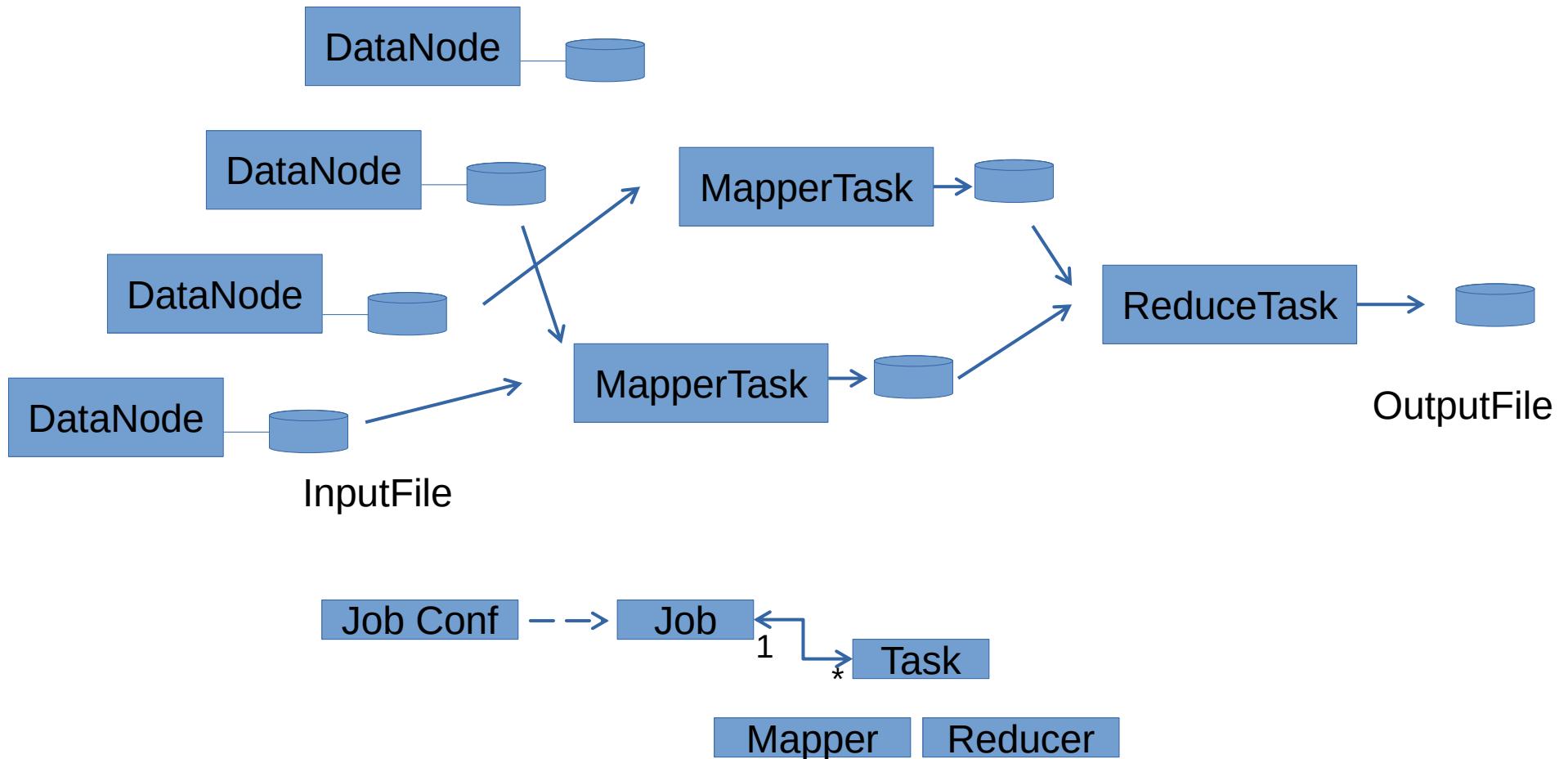


Default size ... Block(128M) => packet(64K) => 126 \* chunk (512b + checksum 4b)

# MapReduce



# MapReduce Principle



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

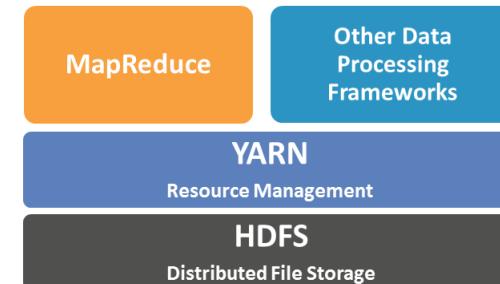
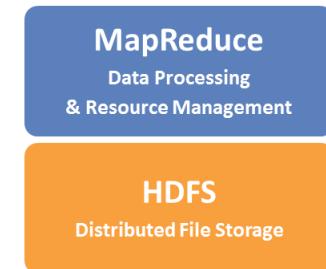
Origin = Google paper

Job resources launcher refactored in “Yarn”

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

=> 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



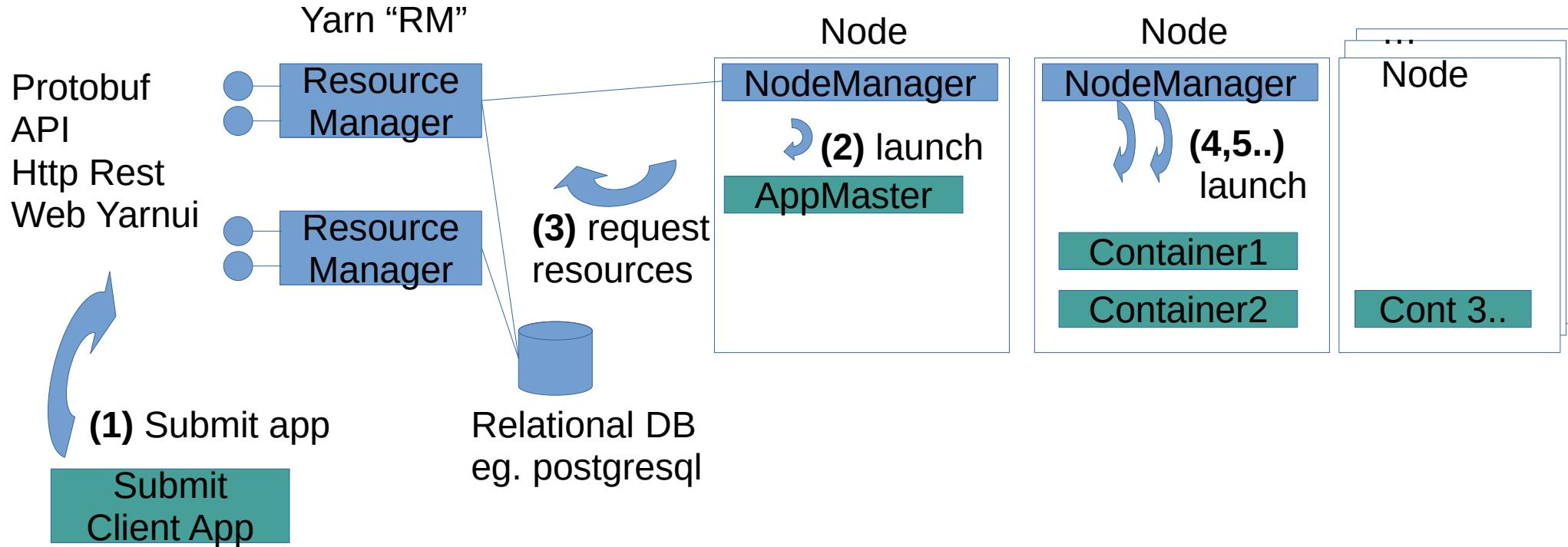
}

Present

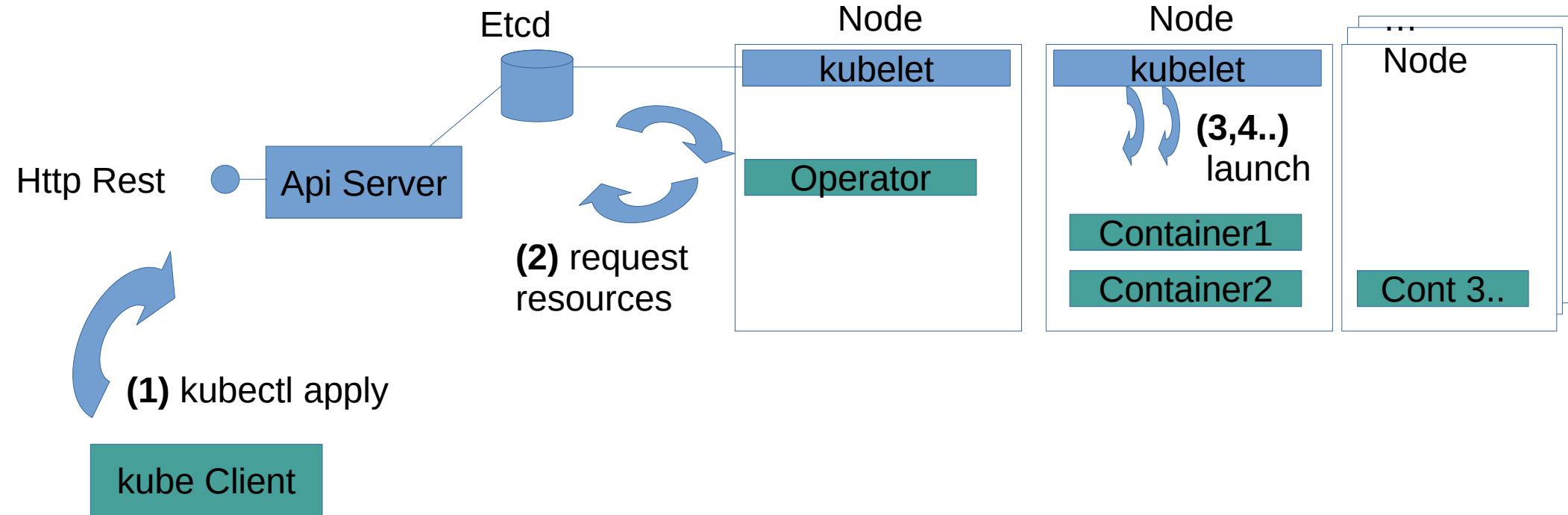
}

Future

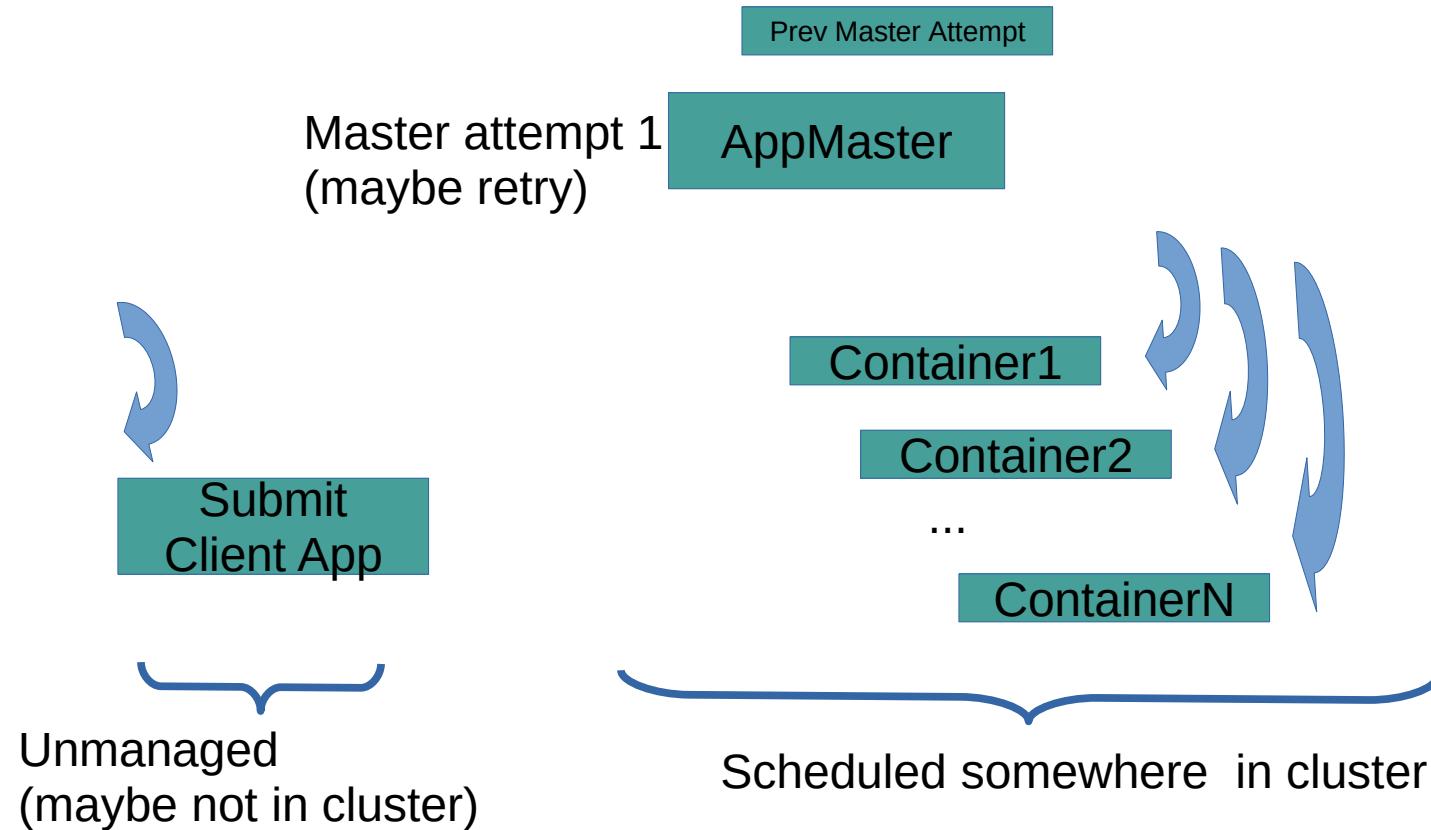
# Yarn Architecture



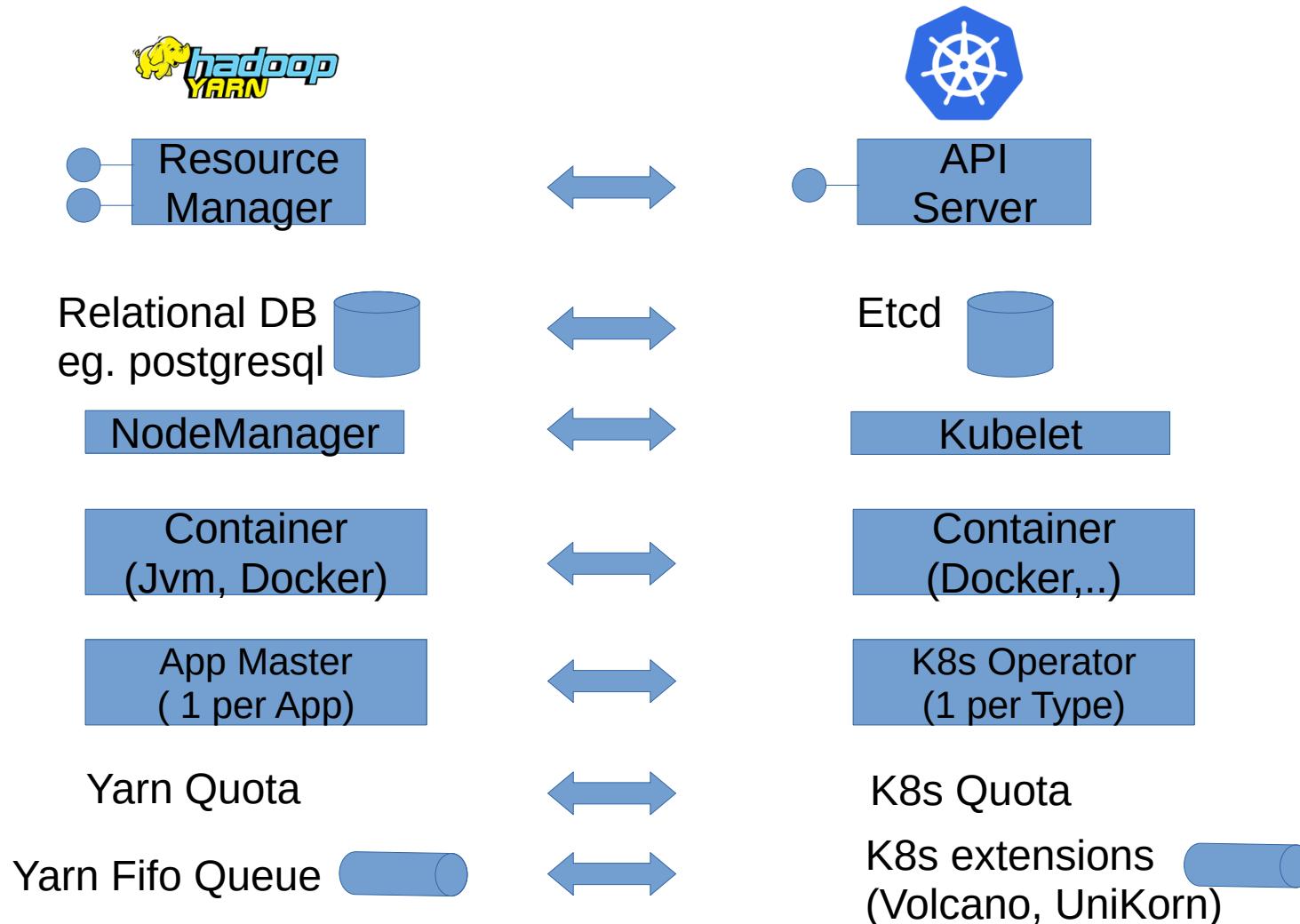
# Comparison ... Kubernetes



# 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 (Grey)

- root (Green bar)
- q1 (Grey bar)
- q1.q11 (Grey bar)
- q1.q12 (Grey bar)
- q2 (Grey bar)
- default (Orange bar)

Show 20+ entries

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus	Progress	Tracking
application_149408864466_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

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

## 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 job is required so that to ensure that you can correctly set the job details.

- The main

- #### **REFERENCES**

- Application filter tag name

- #### 三、影响土壤微生物活性的因素

- ContainerLaunchContext: The information defining the configuration needed to run the application such as the location (RECT).

# 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");

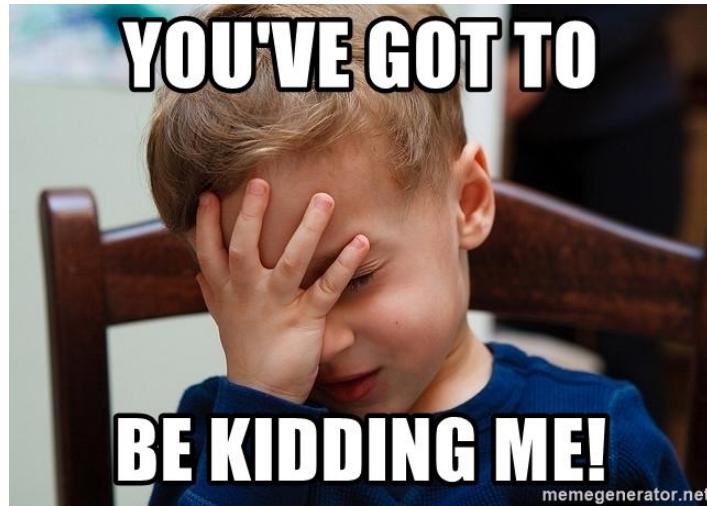
// Get final command
StringBuilder command = new StringBuilder();
for (CharSequence str : vargs) {
    command.append(str).append(" ");
}

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

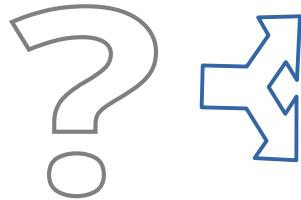
// Set up ContainerLaunchContext, setting local resource, environment,
// command and token for constructor.

// 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)

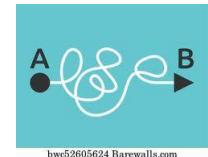
Distributed  
Hadoop  
App,  
not spark

→ Builtin:  
Spark-submit  
--master yarn  
→ A red push button with the word "easy" written on it in white.

→ The Oozie logo, which consists of several blue and yellow squares arranged in a diamond shape with the word "OOZIE" written across them.  
→ A teal square containing a white line that starts at point A and ends at point B, with arrows at both ends indicating a path or flow.

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

→ Reconsider !!  
Maybe  
services  
- Multiple oozie  
→ A black and white "U TURN PERMITTED" road sign.



Unpractical  
Config + Web UI

# Yarn Commands Line (not for launch)

```
$ yarn app -list
```

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

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

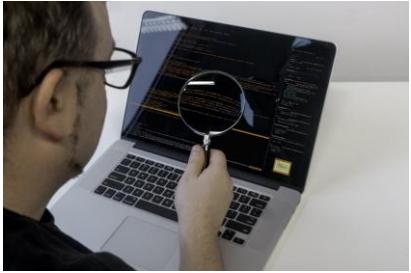
```
$ yarn logs -applicationId <applicationId>
```

```
$ 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 a dual-pane interface. The left pane shows the Hadoop Web UI's "All Applications" page with a table of application logs. The right pane shows a terminal window with log output from "yarn logs -applicationId application\_1550499008606\_0001".

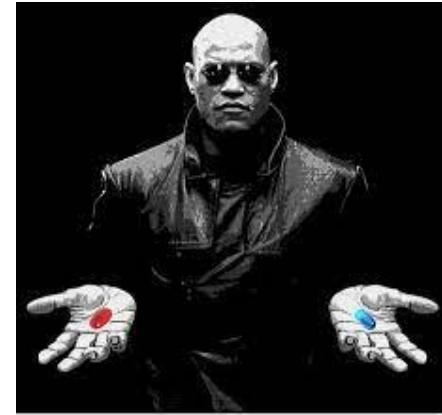
hduser@student11:~\$ yarn logs -applicationId application\_1550499008606\_0001  
19/02/18 14:23:50 INFO client.RMProxy: Connecting to ResourceManager at student11/10.42.0.21:8032

Container: container\_1550499008606\_0001\_01\_000001 on student11-x1\_37389\_1550499095956

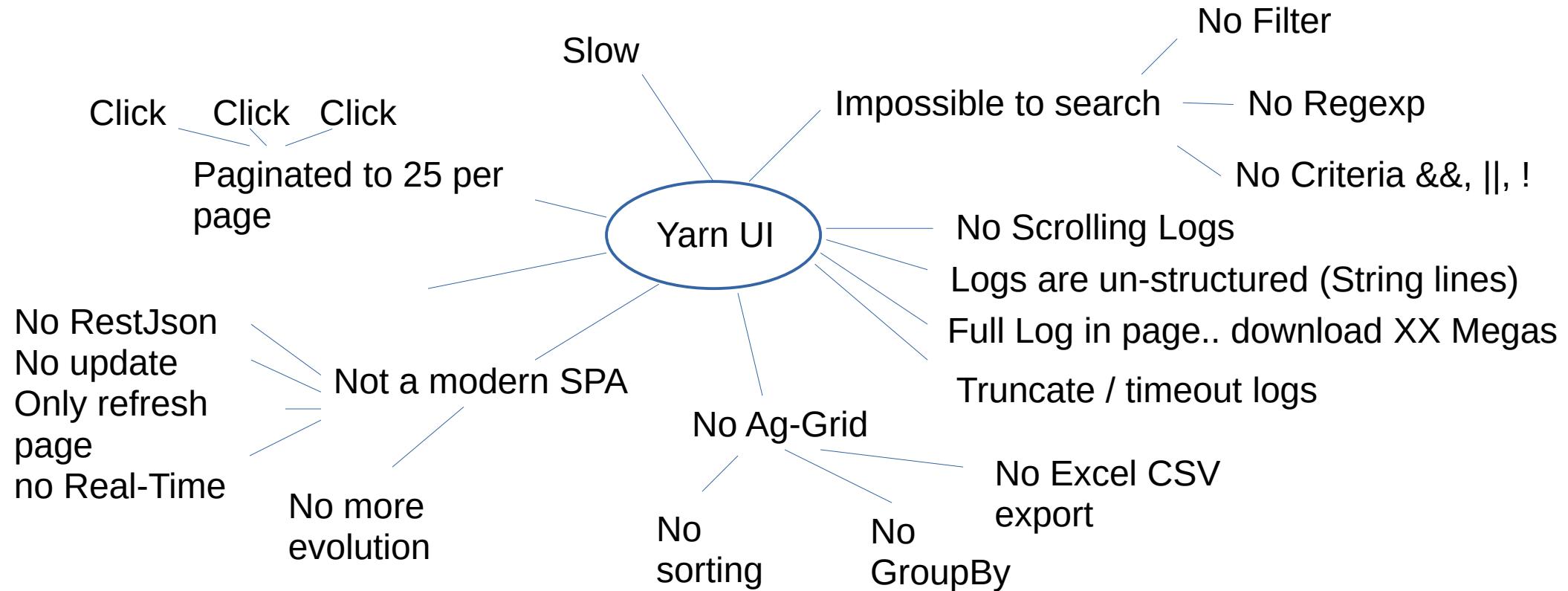
LogType:stderr  
Log Upload Time:Mon Feb 18 14:11:35 +0000 2019  
LogLength:553

hduser@student11:~\$

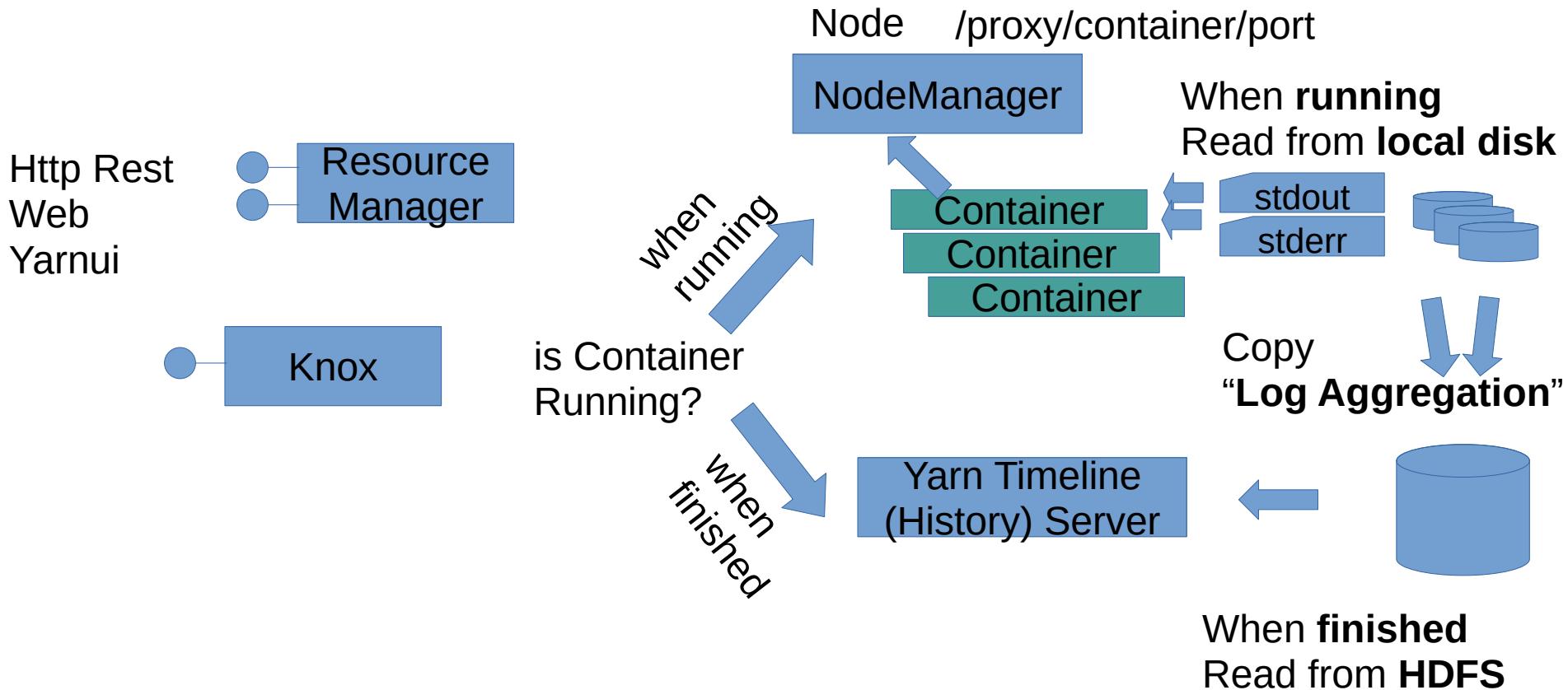
ID	User	Name	Application Type	Queue	StartTime	FinishTime	Status	FinalStatus	Running Containers	Allocated CPU/VCores
application_1463379223862_0568	hduser	infaspriko	SPARK	root.Idmua	Mon May 16 13:11:55 2016	Mon May 16 07:00 2016	FINISHED	SUCCEEDED	N/A	N/A
application_1463379223862_0567	hduser	infaspriko	SPARK	root.Idmua	Mon May 16 13:11:55 2016	Mon May 16 13:13:01 2016	FINISHED	SUCCEEDED	N/A	N/A
application_1463379223862_0566	hduser	infaspriko	SPARK	root.Idmua	Mon May 16 13:11:55 2016	Mon May 16 13:13:00 2016	FINISHED	SUCCEEDED	N/A	N/A
application_1463379223862_0565	hduser	infaspriko	SPARK	root.Idmua	Mon May 16 13:11:55 2016	Mon May 16 13:12:59 2016	FINISHED	SUCCEEDED	N/A	N/A
application_1463379223862_0564	hduser	infaspriko	SPARK	root.Idmua	Mon May 16 13:11:54 2016	Mon May 16 13:12:59 2016	FINISHED	SUCCEEDED	N/A	N/A



# Yarn UI ...



# Yarn Logs



# Oozie → Yarn UI → Yarn Logs

## 1/ Find Oozie workflow Id

Job Info

Job Name: My First Workflow  
Job Id: 0000007-150727083427440-oozie-ooz-W

Status: SUCCEEDED

User: hue

Create Time: Mon, 27 Jul 2015 11:06:58 GMT

Last Modified: Mon, 27 Jul 2015 11:06:59 GMT

End Time: Mon, 27 Jul 2015 11:06:59 GMT

## 2/ oozie-action

Action Info

Name: mr-node

Type: map-reduce

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: http://localhost:50030/jobdetails.jsp?jobid=job\_201110101107\_0004

Tracker URI: localhost:9001

## 4/ Console URL = url in Yarn (Proxy or History)

Logs for

container\_1428386215281\_60485\_01\_000002

File	Length
stdout	1508202 bytes
stderr	536 bytes
spark.out.1	53428982 bytes
spark.out.2	53428982 bytes
spark.out.3	524289821 bytes
spark.out.4	52428942 bytes
slider	536 bytes
stdout	1508202 bytes

## 10/ log files page

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

## Example

Shell → spark-submit

## 7/ spark

6/ Yarn (spark-submit client)

5/ Mapreduce (SHELL)

... call spark-submit --mode cluster

All Applications

ID	User	Name	Application Type	Queue	Start Time	FinishTime	State	FinalStatus	Running Containers	Allocated CPU/Vcores	Memory Used	Memory Pending	Memory Reserve
application_1455000259206_0001	dr.who	dr.who	SPARK	default	Mon May 16 13:11:59 2016	Mon May 16 13:13:03	FINISHED	SUCCEEDED	NA	NA	0	0	0
application_1455000259206_0002	dr.who	dr.who	SPARK	default	Mon May 16 13:11:59 2016	Mon May 16 13:13:03	FINISHED	SUCCEEDED	NA	NA	0	0	0
application_1455000259206_0003	dr.who	dr.who	SPARK	default	Mon May 16 13:11:59 2016	Mon May 16 13:13:03	FINISHED	SUCCEEDED	NA	NA	0	0	0
application_1455000259206_0004	dr.who	dr.who	SPARK	default	Mon May 16 13:11:59 2016	Mon May 16 13:13:03	FINISHED	SUCCEEDED	NA	NA	0	0	0
application_1455000259206_0005	dr.who	dr.who	SPARK	default	Mon May 16 13:11:59 2016	Mon May 16 13:13:03	FINISHED	SUCCEEDED	NA	NA	0	0	0
application_1455000259206_0006	dr.who	dr.who	SPARK	default	Mon May 16 13:11:59 2016	Mon May 16 13:13:03	FINISHED	SUCCEEDED	NA	NA	0	0	0

Application application\_1455000259206\_0001

Kill Application

User	Name	Application Type	FinalStatus Reported by AM
root	select count(*) from server_logs(Stage-1)	MAPREDUCE	ACCEPTED: waiting for AM container to be allocated, launched and register with RM.

Application Metrics

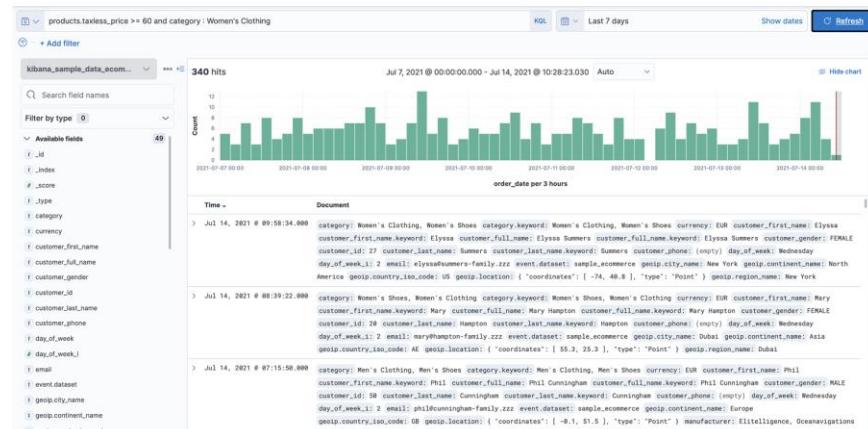
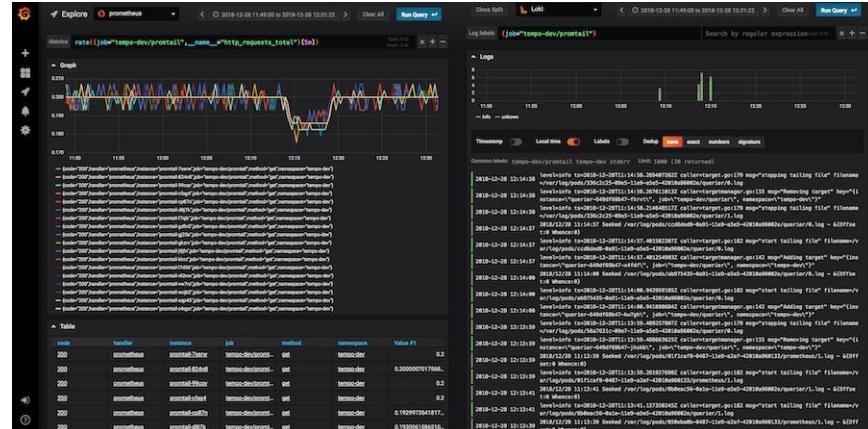
Total Resource Preempted	Total Number of Non-AM Containers Preempted	Total Number of AM Containers Preempted	Resource Preempted from Current Attempt	Number of Non-AM Containers Preempted from Current Attempt	Aggregate Resource Allocation
<memory:0, vCores:0>	0	0	<memory:0, vCores:0>	0	0 MB-seconds, 0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

Show: 20 entries

Attempt ID	Started	Node	Logs	Blacklisted Nodes
appattempt_1455000259206_0001_000001	Feb 9 07:35:26 +0000 2016	http://	0	

## 8/ appmaster application attempt 1

# Yarn Log Alternatives...



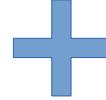
# Yarn Summary / Alternative(s)

Mesos was deprecated ... Yarn is also (complex / inefficient)



**kubernetes**

Queues



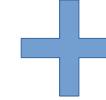
VOLCANO

or



YUNIKORN

Logs



Grafana loki

or



elasticsearch

Operators



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

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

**No simple packaging to launch**

**No trigger after data change**

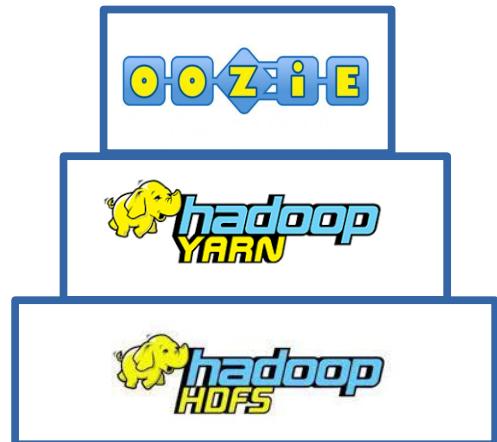
**No workflow between data / job**

# Apache Oozie



# Oozie Features simpler than / missing in Yarn

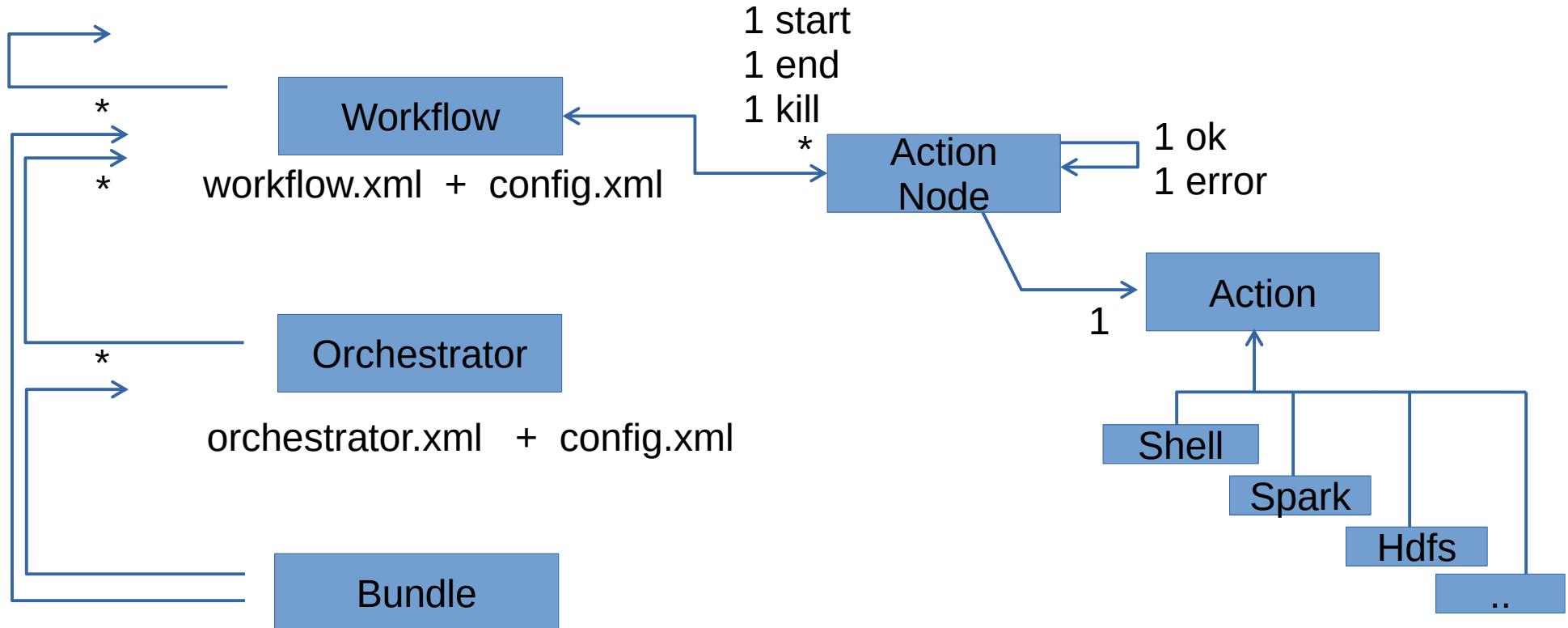
Orchestrator: “CRON” (periodic scheduling, like 2 \*/5 MON-FRI \* \* \*)



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
```

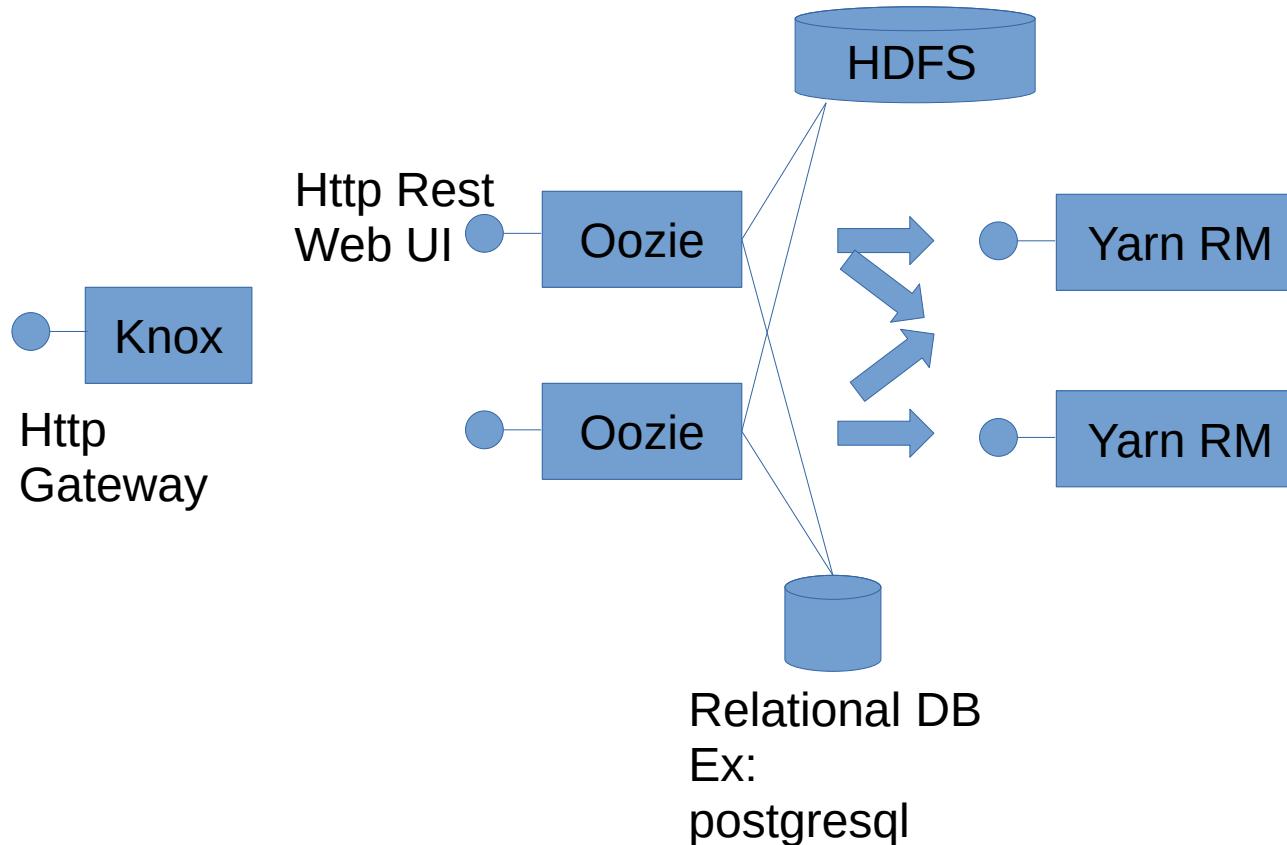
**oozie.wf.application.path= hdfs://a/b/workflow.xml**

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 for a job named "OnPrem-HDFS-to-Databricks-ETL-job". The job ID is 0000001-170219172252198-oozie-oozi-W. The status is SUCCEEDED. The Actions table lists several actions, with the first two circled in red.

Action Id	Name	Type	Status
1	0000001-170219172252198-oozie-oozi-W@: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	OK
7	0000001-170219172252198-oozie-oozi-W@emailS...	email	OK

Coord executions  
= launched  
workflows

The screenshot shows the Oozie Workflow interface for a workflow named "WorkflowWithSqoopAction". The job ID is 0000001-170415112256550-oozie-serg-W. The status is FAILED. The Actions table lists two actions, both circled in red.

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

Workflow  
actions

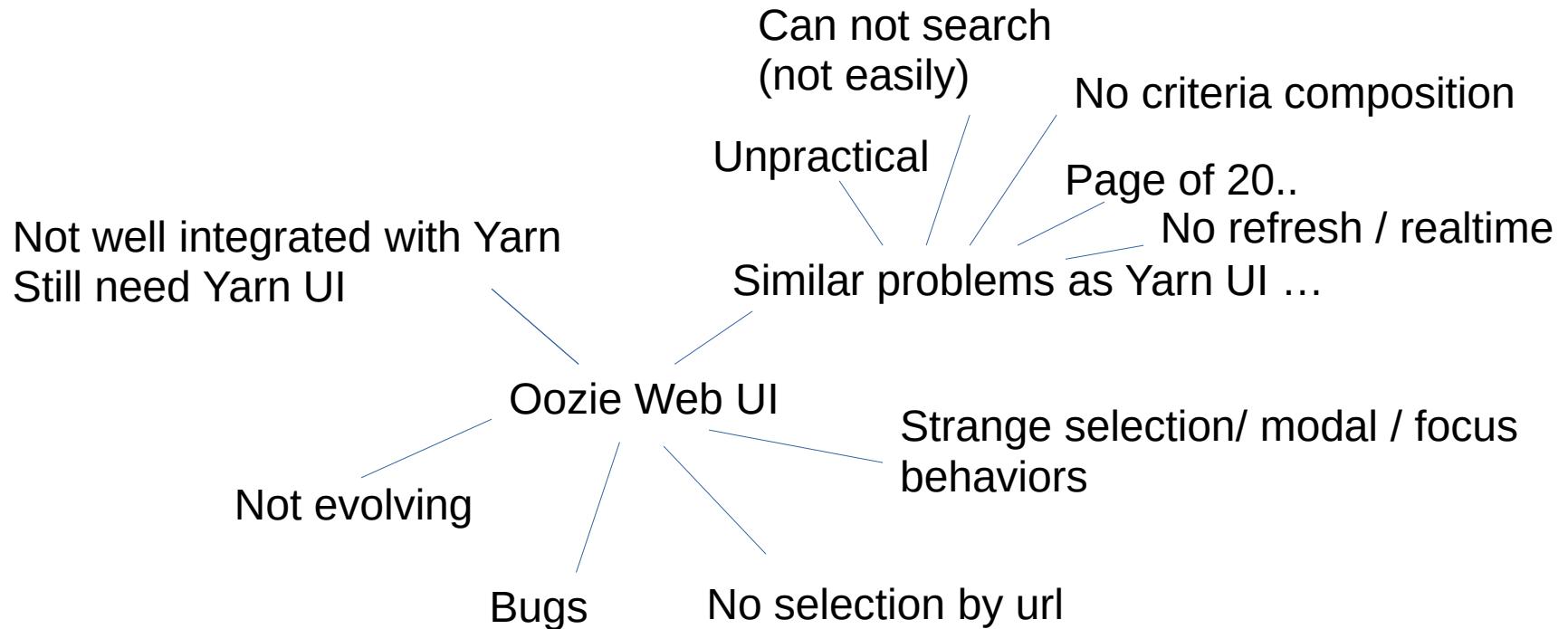
## Workflow

Action = yarn job

The screenshot shows the Action configuration interface for a shell action named "shell-node". The job ID is 0000051-160401125659728-oozie-oozi-W. The status is RUNNING. The configuration fields include Name: shell-node, Type: shell, Start Time: Sun, 03 Apr 2016 19:37:15 GMT, and Status: RUNNING.

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

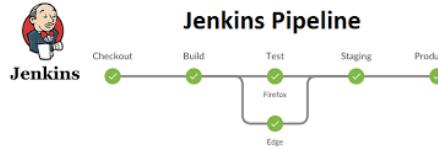
No Datalineage between data / job  
No trigger on data change

# Oozie Alternatives

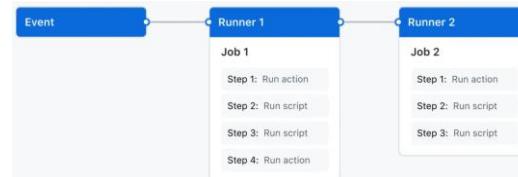
Livy .... to start simple Spark actions



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



GitHub Actions



Etc..

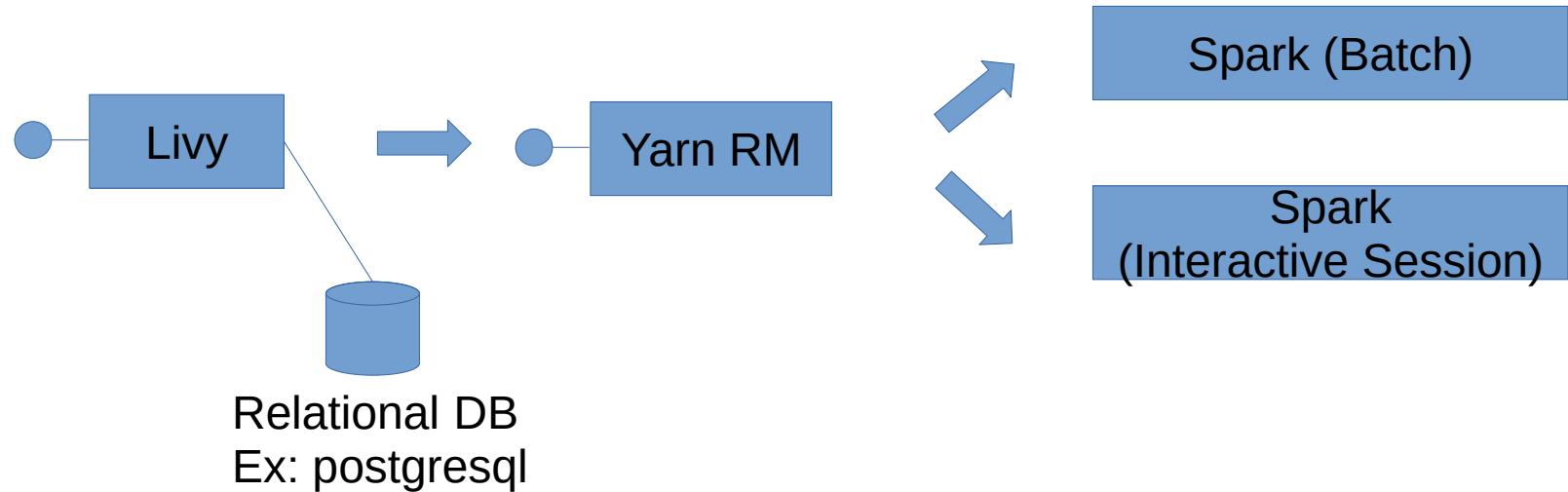
Etc..

Etc..

# Apache Livy



# Livy Architecture + Features



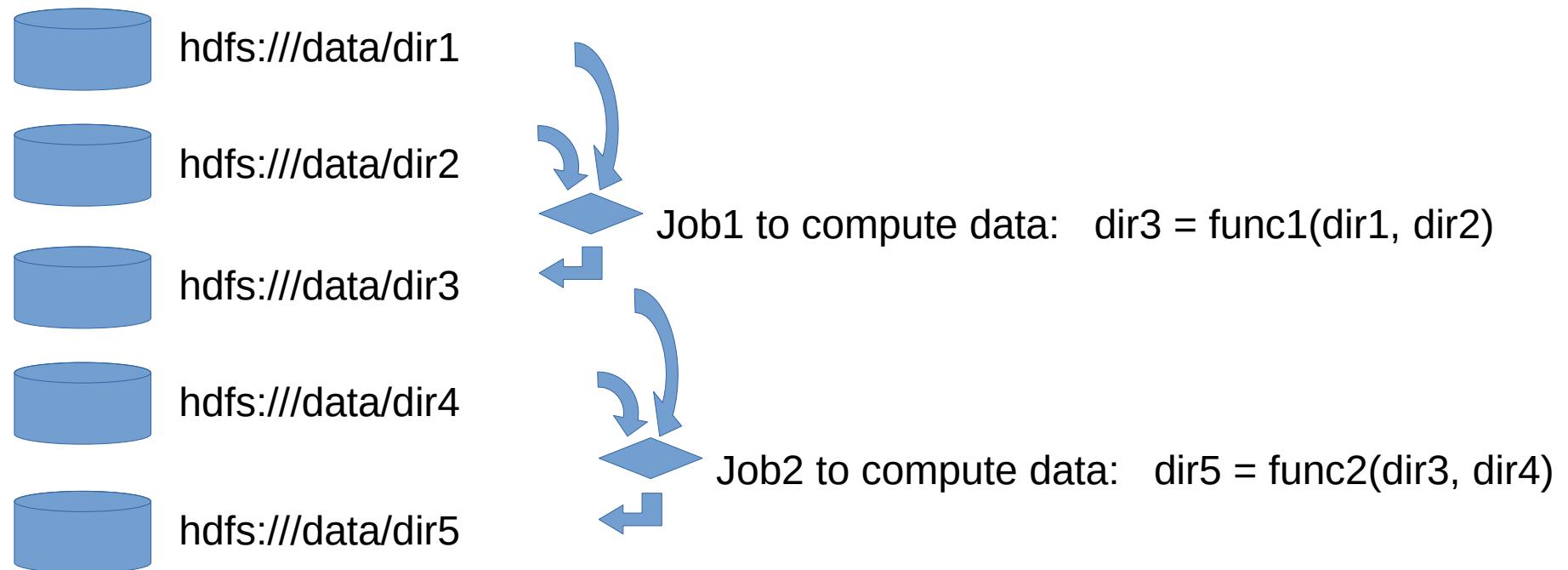
# Livy sample

file: app.json

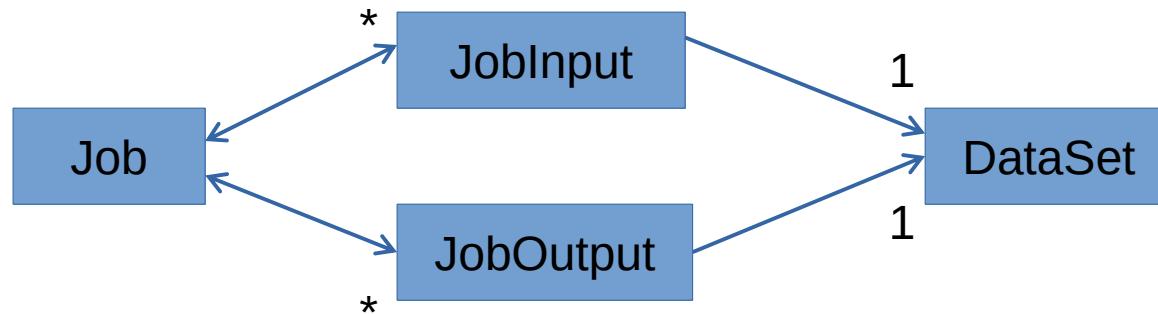
```
{  
  "file": "hdfs:///myapps/hello-  
spark.jar",  
  "className": "myapp.HelloSpark",  
  "args": ["hdfs:///inputs"],  
  "jars": [  
    "hdfs:///myapps/mylib.jar"  
  ],  
  "driverMemory": "1g",  
  "executorMemory": "5g",  
  "executorCores": 8,  
  "numExecutors": 2  
}
```

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

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



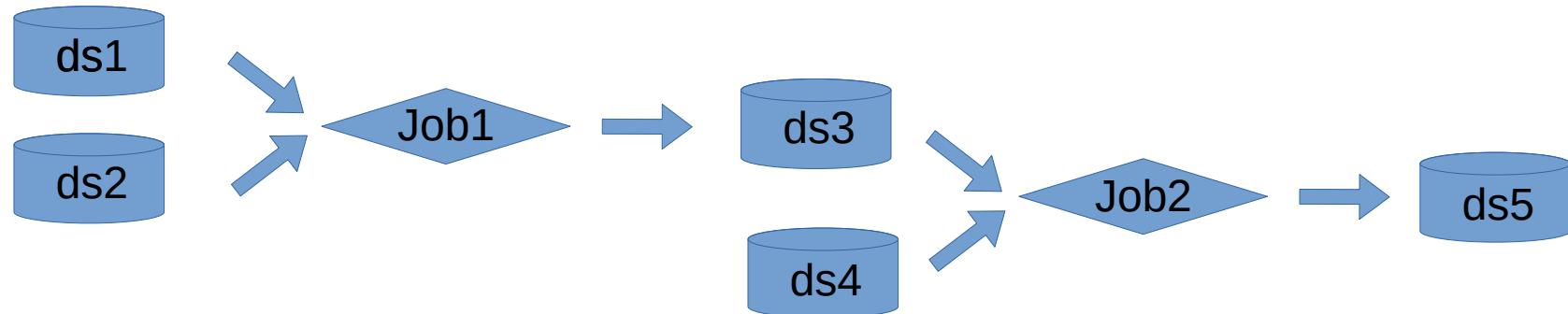
# Job (Input, Output) re-eval on input change



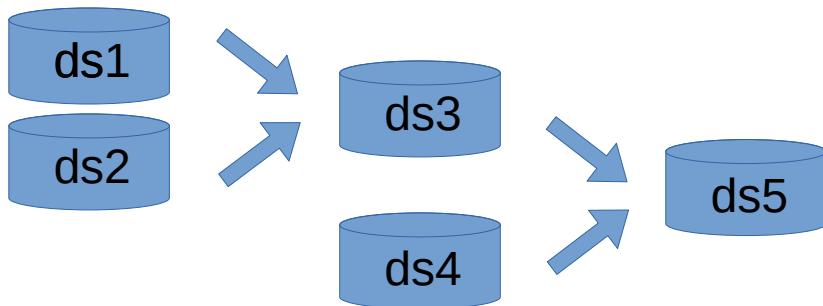
When detecting external change to DataSet => search for JobInputs => launch Jobs

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

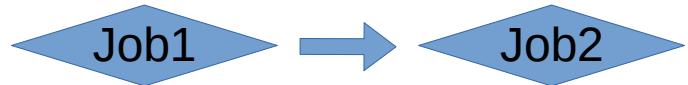
# DataLineage DataSet+Job



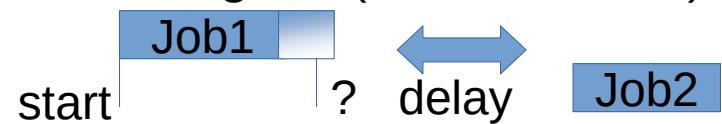
Sub graph : **Data dependency**



Sub graph : **Job orchestration**

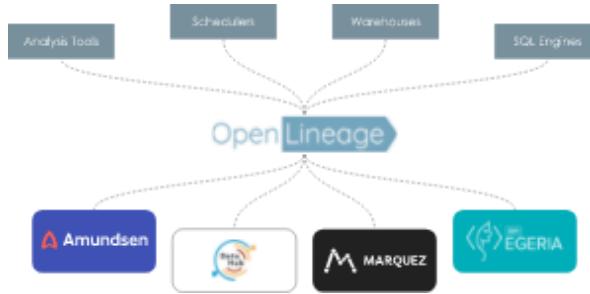


**Gant Diagram (cron + duration)**

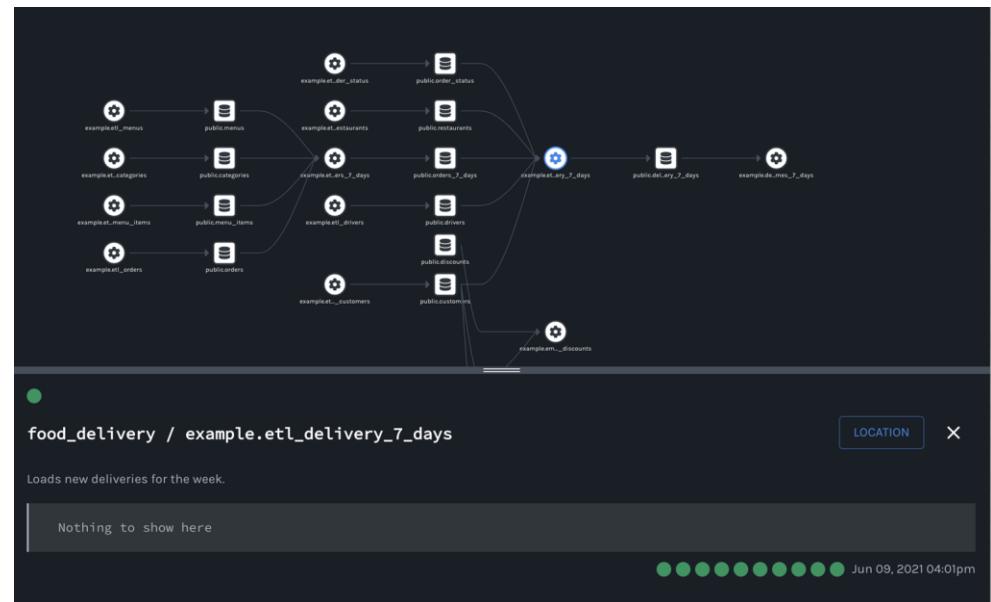


# Example DataLineage Spec / Implem OpenLineage / Marquez

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