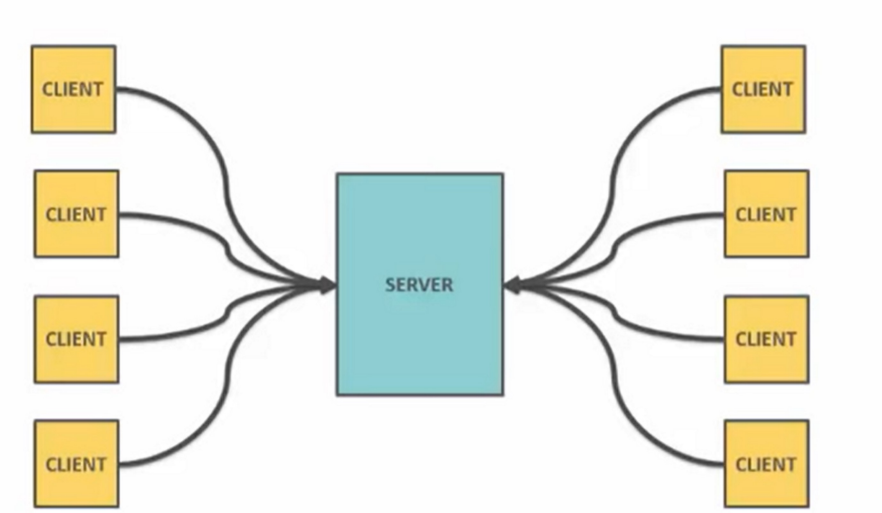
HADOOP

27 เมษายน 2565 9:41

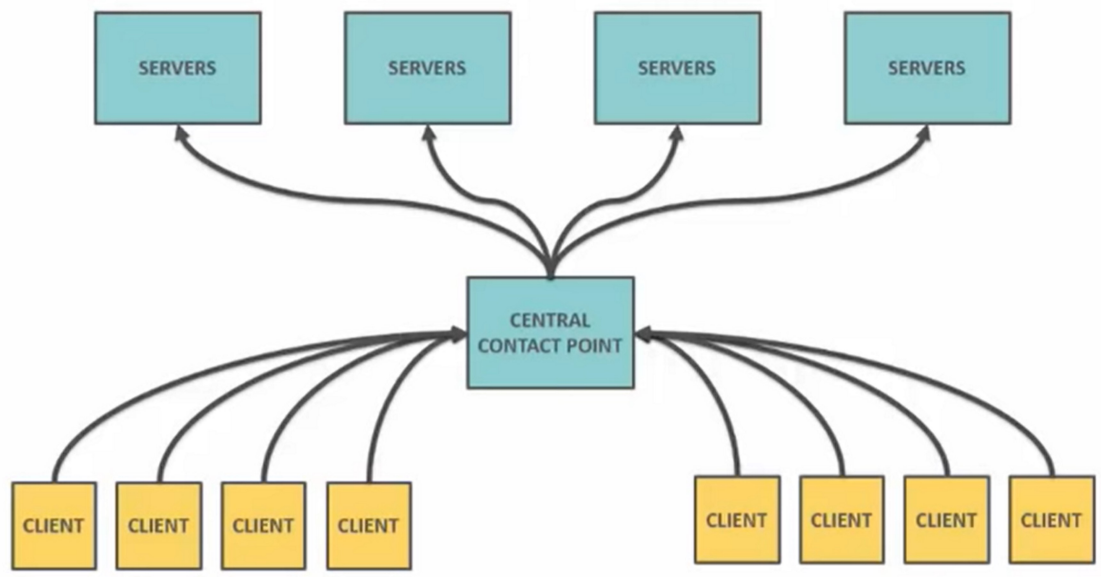
TABLE OF CONTENT

An Introduction to the Basics



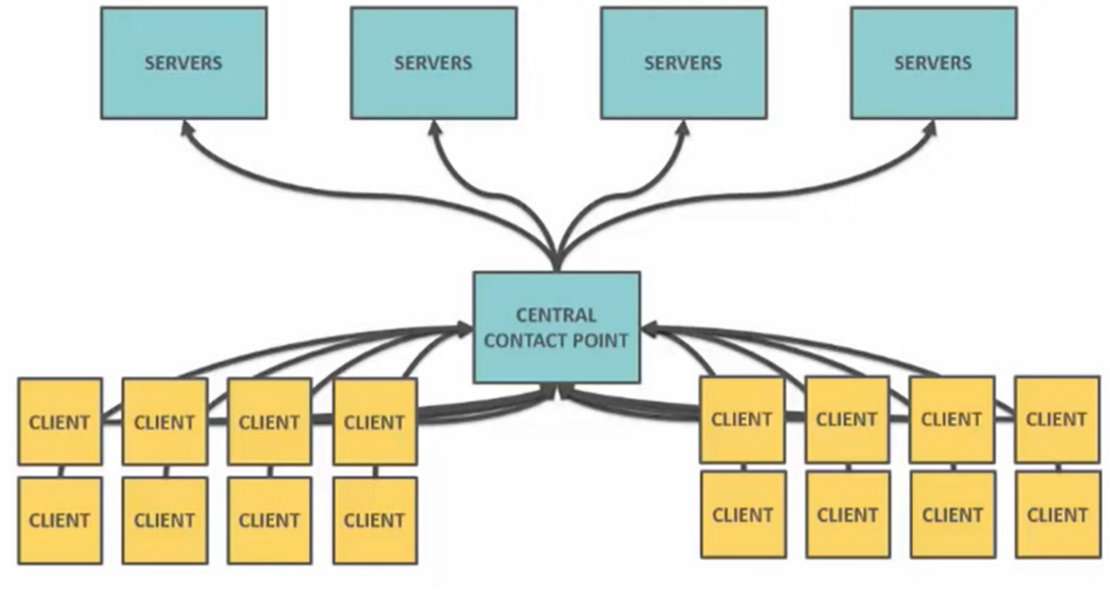
In the 80s big data is client server where the paradigm was client would come to the server and the server would provide data to them. But there's a limit to what backend server can do, at the end of the day this could leave you with unhappy clients and an overload server.

The most obvious solution is to get a bigger stronger server and upgrading hardware is still a good choice today but it's pretty expensive.



Another idea which came from the scientific computing space was to instead of making it one big machine why don't we do something like a storage area network or a distributed file system here there's a

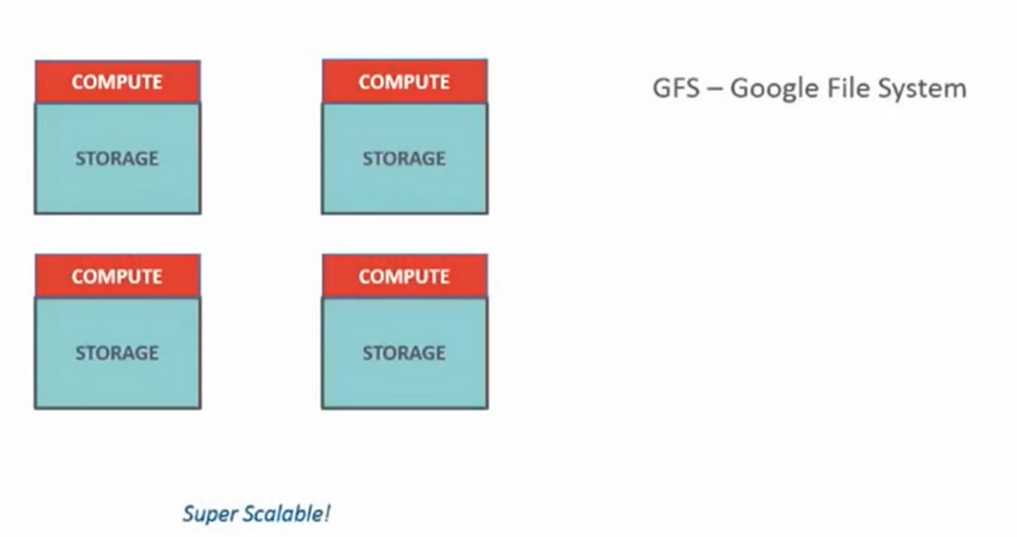
central point of contact to determine
which server you want to talk to and
then you have many back-end servers that
are shouldering the load this approach
will yield better scalability than just
upgrading your back-end machine



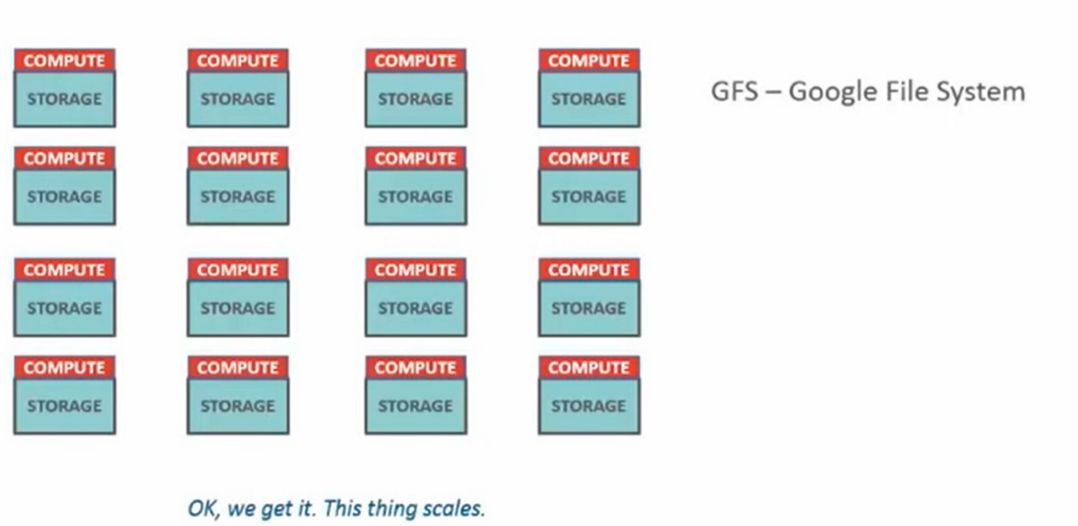
but you still have trouble when the clients can
overwhelm the servers. If you put a ton of servers in here then you get smoke out of your distributed file system and your servers will bottlenecked and unhappy. No matter how you try to carve this the separation between the backend owner of the data and the client-side
user of the data your clients can always overwhelm in the system. It's a fact of the architecture

So if put in your box on steroids can get expensive and if a
distributed file system can be overwhelmed what's the answer well to use another 80s reference let's go back
to the future.

Fast-forward to the
Internet age and Google needed to create
a file system to be able to take on a
formidable task indexing the web to
tackle this. They create something called
GFS the Google file system.



The idea with
GFS is that instead of having a giant
file storage appliance sitting in the
back end somewhere. you can instead use
industry standard Hardware on a large
scale and drive high-performance to the
sheer number of components.
Given that it's standard Hardware you expect
failure and you make it reliable through
redundancy and replication.
Now all Hardware fails at some point and if you
have thousands of nodes it's likely that
you will have node failure frequently.
If you just take the average time between
failure and apply that to thousands of
nodes it will happen pretty frequently.
But in this model it doesn't really
matter because your data doesn't live in
any one place.
It's replicated three times around the network to ensure
availability so the rest of the active nodes just keep marching along.
Even better this replication means the data is broken into chunks and spread
throughout many nodes so now there is no central storage to overwhelm.
This also provides scale as you add more nodes you
add more computing power and you're
adding more storage capacity so you have
a system that can scale rapidly and
without much trouble.



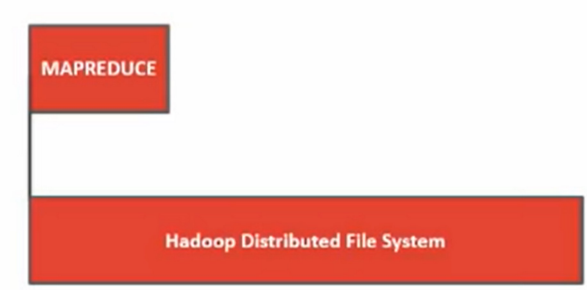
Google then paired
this storage system with a new computing
model called MapReduce the concept had
been around for a while but this was
really the first time it became an
industry changing event previously it
had been a research topic the idea is
that you take your task which is data
oriented and you chunk it up and
distribute it on the network such that
every piece of work is done within the
network by the machine that has the
piece of data that needs to be worked on.
So now not only does your storage scale
to compute implicitly but you need a lot
less network bandwidth because you're
not transferring massive amounts of data
Around.
The data lives where the data
will be used it's like your data works
from home.
So that's the story of Google.

where does Hadoop come into this?
Well Google was kind enough
to publish white papers on this
breakthrough out to the general public.
As you can imagine this created a ton of
buzz in the community and people started
to take notice.
Now of those taking note two people in particular mattered

Doug cutting and Mike calf Arella they
created their own version and Doug named
the project after his son's toy elephant
and the initial version of what we now
know as a dupe was created.
Since then Doug has moved on to the highest of
callings working here at Cloudera
now that we reviewed the history of data
management Hadoop let's dive it in to an
introduction to the core components of
the software.

Introduction to Core Hadoop

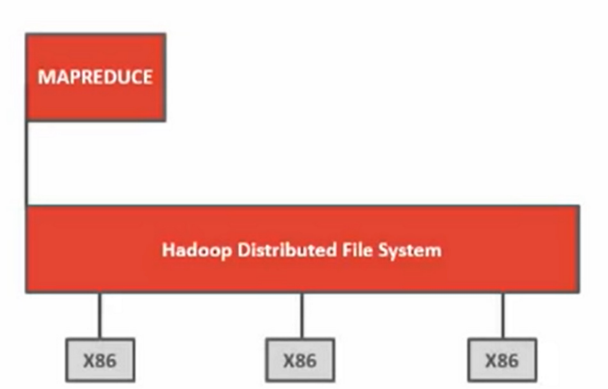
Now Hadoop can be a complicated topic sometimes it seems
like there are more animals and funny
names and you can shake a stick at.
Trying to make sense of it all at once
can be difficult and confusing but it
doesn't need to be if we start from the
ground up and build our understanding
step by step you'll see that it's more
straightforward than it seems at first
pass.
Let's get started with the foundation of Hadoop.



At its core Hadoop started out as a
distributed file system in a processing
paradigm.
That file system is called HDFS
the Hadoop distributed file system.
The processing paradigm is called MapReduce.

Let's start with HDFS
HDFS is a file system written in Java which sits on top
of the NATO filing system for whatever
OS you operate.

It's built on top of x86 standards which are very cost effective
when it comes to processing particularly
when compared to high performance
computing or HPC.
HPC certainly has its place it is a great technology but it's
not a prerequisite for a Hadoop
ecosystem when it comes to x86 there are
reference architectures for whatever
brand of server you'd like to use so
there's a lot of flexibility as we
discussed earlier Hadoop is built to
expect failures from these servers.

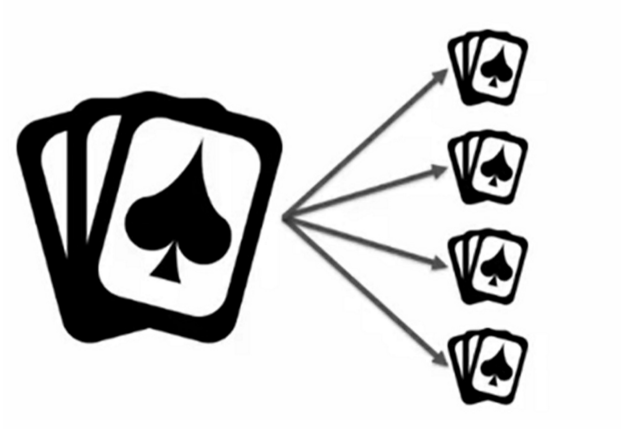


Generally the idea is that the data
comes in it lives on these servers and
you can push workloads to each of these
servers to live locally rather than
pulling it into a central location
that's a huge advantage.

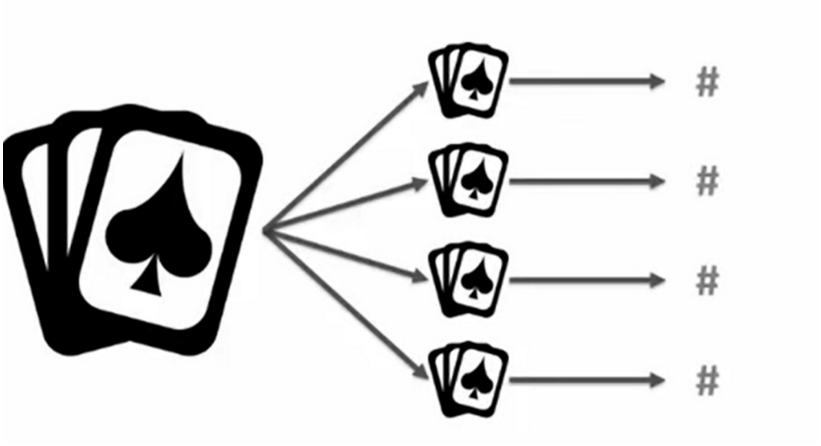
Now let's talk about MapReduce
MapReduce is a processing paradigm that pairs with HDFS
at its core Hadoop is just this
MapReduce and HDFS
MapReduce is a distributed computation
algorithm that pushes the compute down
to each of the x86 servers that we
discussed earlier
now Hadoop distributed file system
that's about as descriptive as a title
as it can get where the heck does
MapReduce come from
well it's the combination of a map
procedure in a reduced procedure.
The map procedure performs filtering and sorting
of the data and the reduced procedure

performed summary operations.

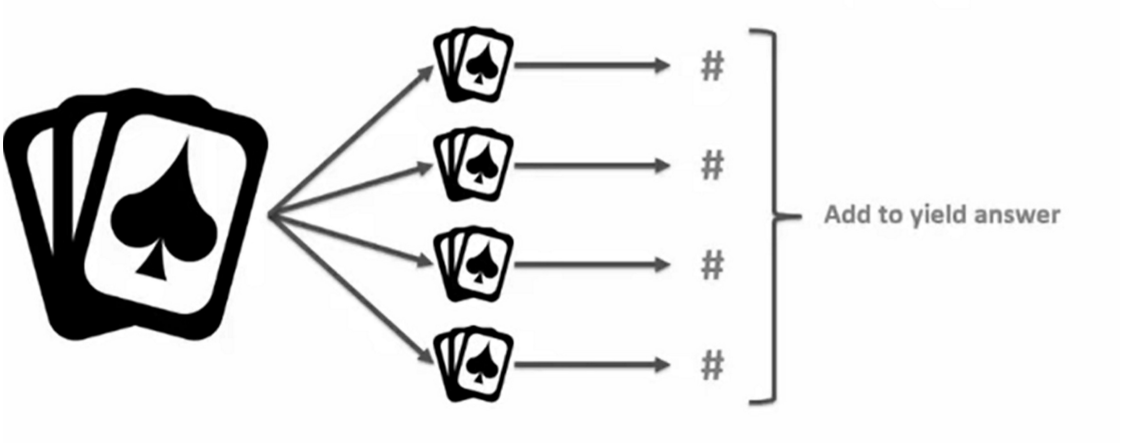
So in a simple example, say you want to get a sense of how many spades are in a deck of cards you never know what's in a deck that you already use some cards get lost some get found you know how that goes. There's a pretty straightforward way to count the spades you can go through the deck and count each of them as you come across them. However that can take a while you'll need to go through the entire deck on your own now unless you're playing solitaire chances are you have a few friends are sitting down to play with.



So if you divide that deck of cards up into four and then distribute the cards amongst your friends they can count the number of spades in the divided deck in a fraction of the time.



Then you can simply add the number of
spades you come across and you arrive at
the answer.



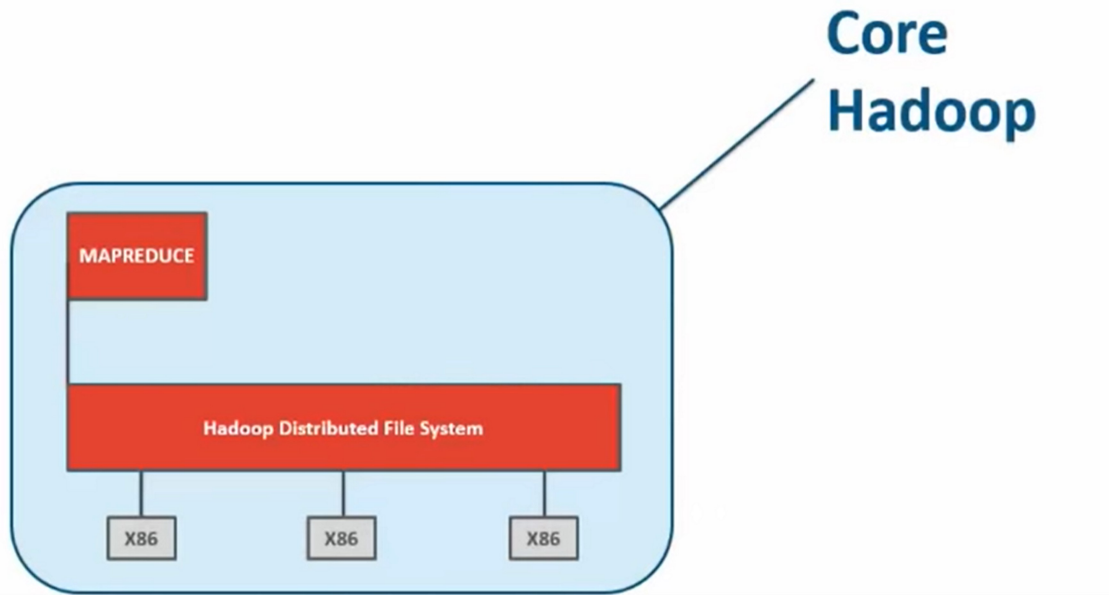
That's essentially what MapReduce does by telling each person to
count you're doing a mapping procedure.
By having them boil the cards down to a
number rather than just handing them
back to you they've done a reduce.
So you can think of each person as a computer
in this example and since they're
working on one job together we'll call
them a cluster.

So that's a pretty easy example but you might be thinking
hey I could have done that on my own
what do I need to do this for?

Well there are only so many cards in a deck so you're
right we call it big data for a reason
Hadoop is meant to tackle problems large
and scale say instead of cards you were
trying to gauge sentiment towards a
candidate for public office but looking
at the words most commonly associated

with her on Twitter the data set and
challenged become much more difficult
and that's where Hadoop to make a big
difference.

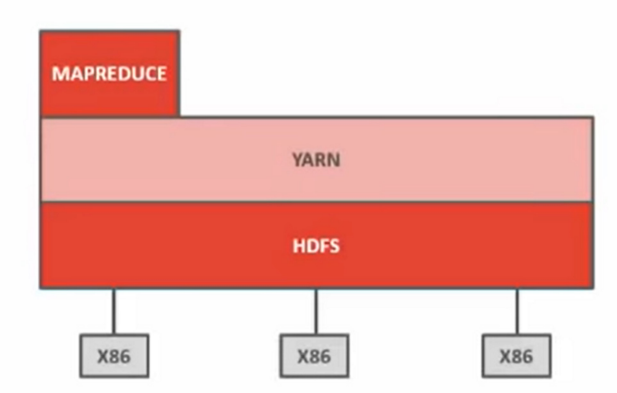
hey now look at that you now know what
the core of the hadoop is all about that
was easy.



But we're not going to stop just yet and
neither did the Hadoop community users
of the system knew that there were some
missing capabilities they needed to get
data into hadoop, get it out,
manage the resources, scheduled jobs and
even more.
As a result a broader open-source community got to work
building out a broader ecosystem and as
a result of this work Hadoop has much
more to offer.

First Steps into the Broader hadoop ecosystem

Let's now move on to the
next section of this session and take a
few steps into the broader Hadoop
ecosystem we'll take it step by step and
help you understand how the ecosystem
works together in a way that makes sense.

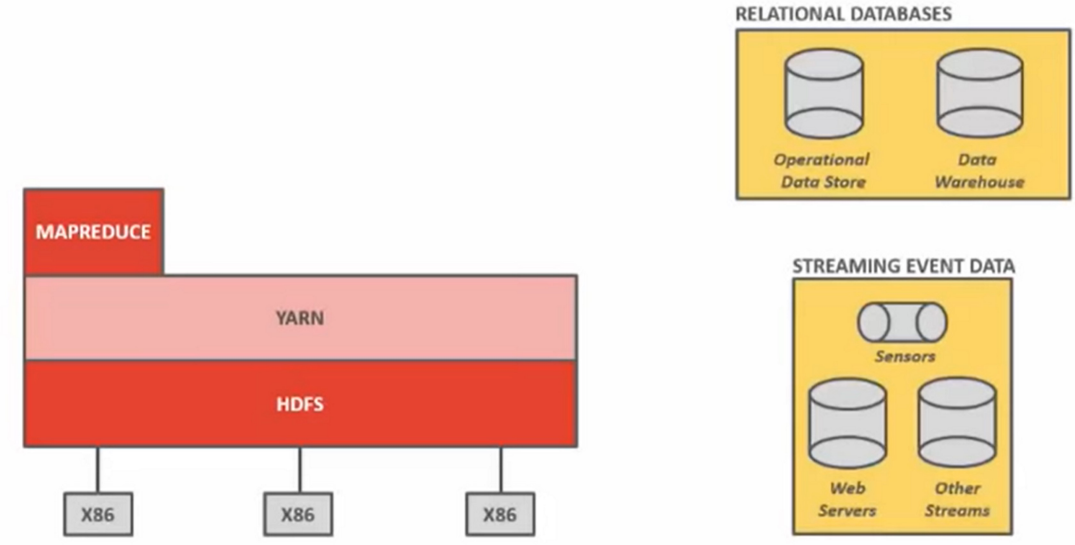


The first additional component will
discuss is yarn.
Yarn is a resource manager for different workloads that you
plug in on top of hadoop.
It manages compute resources and clusters and works
as scheduled users applications.

Again here we run into a strange name.
Why yarn? well very simply is an acronym
that stands for yet another resource
negotiator. So apparently Hadoop comes with a sense
of humor too.

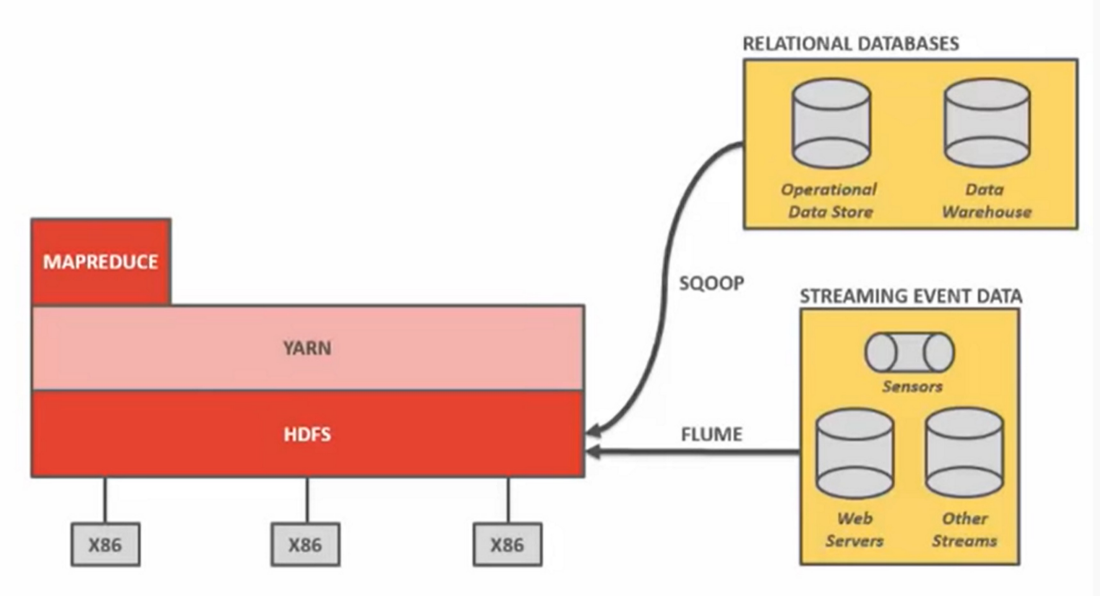
So thus far we've talked a lot about to do but we still haven't
talked about the data.
How do we get the data into the system to actually start
to process it and how do we work with
different kinds of data?

Well the good news is that tools have been developed
to help us out.



Now when it comes to relational databases we can pull data
from there into HDFS using a tool called
Sqoop.
Now boy that's another odd name isn't it and
it's spelt funny too well again there's
a method to the madness.
This one was arrived at by combining
sequel and Hadoop and you can think of
it as scooping data out of relational
databases.
In addition to pulling data from databases sqoop will also allow you
to push it out to them as well.
Sqoop can also compress data to help it make the
journey sqoop is used frequently in
Hadoop distributions but you can also
imagine any set of ETL, which stands for
extract transform load companies doing
the same thing with the graphical
Interface.

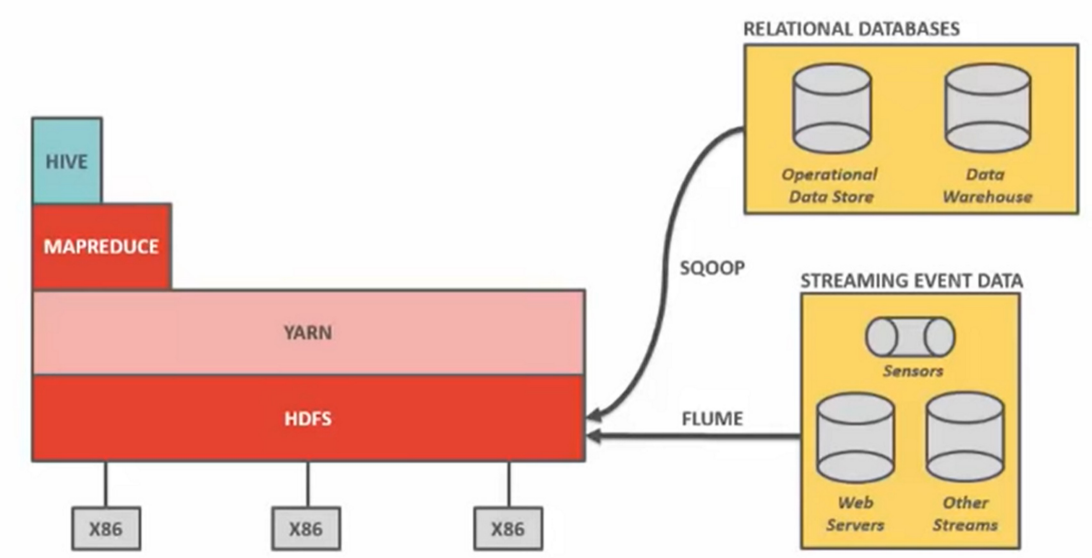
However, we won't focus on
those tools right now because we're
trying to stay within the parameters of
hadoop ecosystem.
Now we've discussed relational databases but what about
other things like web server logs
networking logs and sensor information.
These are less straightforward than a
relational database how do we handle
Them?



That's where flume comes into play

it is a massively distributable
framework for event based data.
With flume, we can now bring streaming event
data into HDFS again, why flume?

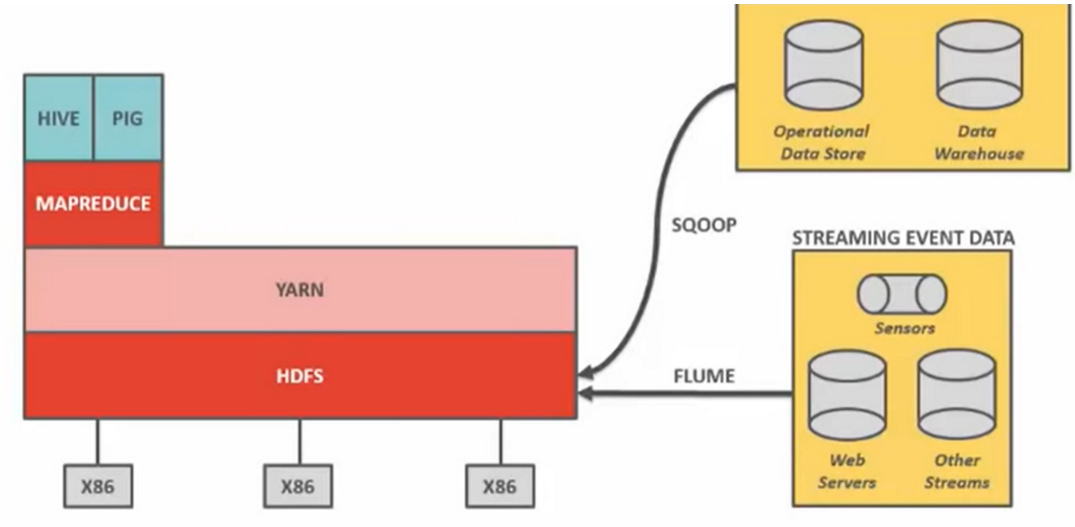
Well if you think about a physical flume for
logging it's a little river that routes
logs and wood to processing factories so
they can cut it up and ship wood.
The same way they do this for physical logs
we do the same for data logs.
So now we've got all the data loaded in we have
core Hadoop and yarn is a resource
manager to make sure things go well
what's next?



Let's talk about hive.
Having dfs and HDFS now let's say we want to do
some work on it we can write some Java,
Ruby, pearl to do some work.
However, people were getting tired of writing
thousands of lines of code when a few
lines of sequel would work so hive was
created to do just that.

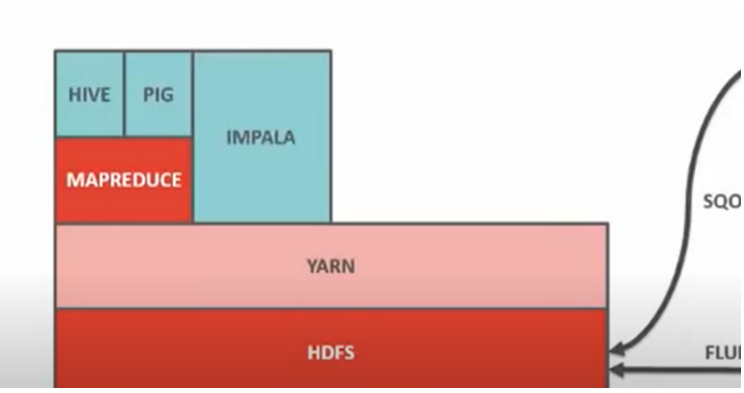
Hive allows you to run sequel that is
converted to MapReduce that can run
against HDFS.

So now you can get this massively powerful processing in terms
of sequel which is great for usability.



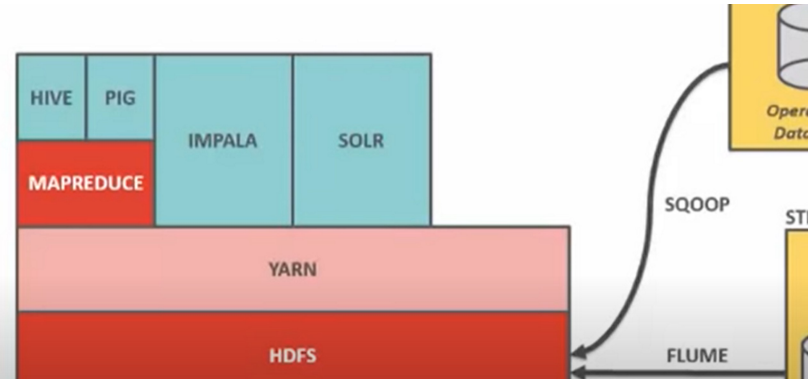
Apache Pig was created as a higher-level scripting language that allows you to create MapReduce programs to run against your data.

There's slightly analogous to Oracle's PLC equal.
You can think of Pig as having a massive appetite for data to make it easier to remember.
Now if we look at both Pig and hive they help us process massive amounts of data but aren't always the best for low latency sequel.
So if you want to hook reporting tools to things, that creates a little bit of an issue but as you might have guessed there's a solution to that.



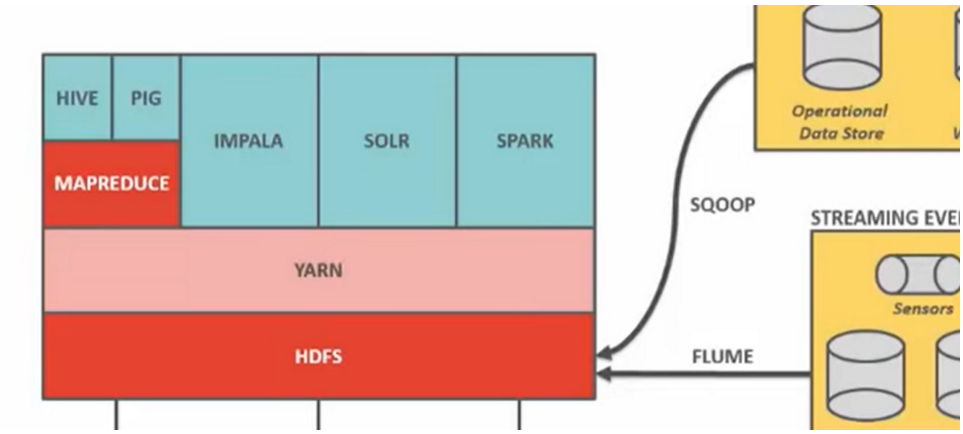
Impala is design is a low latency sequel engine that bypasses MapReduce.
This is along the same lines of sequel queries: for example a query to return 10 rows from a 1,000 row table and hive is going to take 20 to 30 seconds or so.

However using Impala it takes only
milliseconds.
This blazing speed is also how Impala came to be the name for this
component of the Hadoop ecosystem.



Now let's talk about search that's where
solar comes into play, which allows the
indexing of all your Hadoop data this
comes in handy if you'd like to search
or Google if you will, your Hadoop data.
With solar there is the flexibility to
route the data being brought in by sqoop
and flume directly into solar to do
indexing on-the-fly.
However you can also tell solar to index
that data in batches that's pretty neat
if you ask me.

There are software companies that integrate well with the
solar API that will allow you to do
searches on your data even if you don't
know it too well.



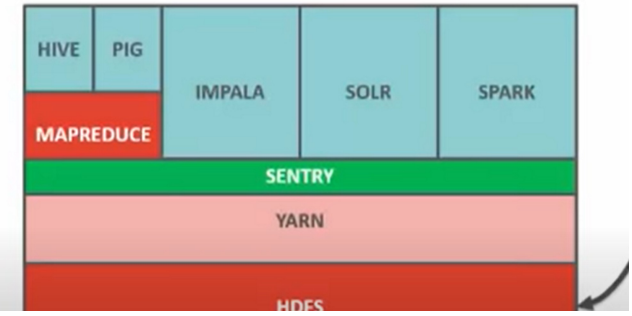
Now let's get to a rising star in the ecosystem Apache
spark.
Spark has generated a lot of buzz recently with the ecosystem and is

delivering on its promise.
This is a technology that may be able to
ultimately replace MapReduce down the
road in addition to providing real-time
streaming capabilities and machine
learning.

Now when it comes to MapReduce it uses specific technology
and a specific approach to breaking up calculations.

Spark on the other hand allows you to use a more traditional API
for analyzing data that doesn't make you think in terms of MapReduce.
It also leverages in-memory work more than MapReduce can.
This means if there are multiple phases or iterations on
calculations SPARK can do this in memory
rather than writing the disk between
each of the phases or iterations.
This makes for much faster speed.
As spark becomes more mature that will eventually
be the same ease-of-use tools for batch
Processing.

Now this is great we have hive Pig Impala solar spark and
many components of the ecosystem but how
can we do authentication?
We may not want Bob to see Mary's table or vice versa

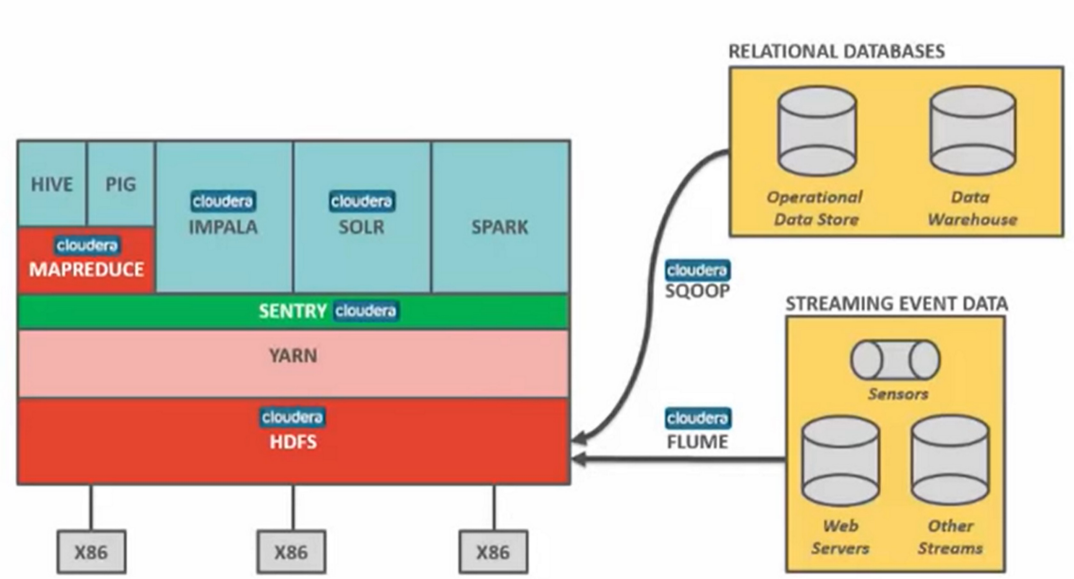


Sentry fills this gap and it creates
role based authentication which is
critical again, critical for security.

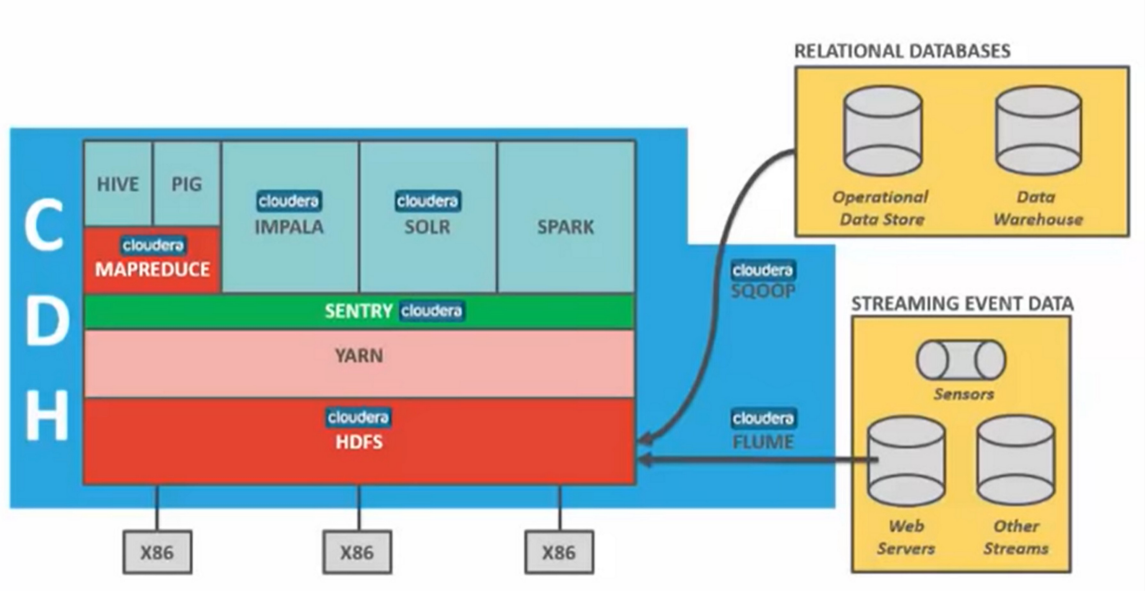
For this introduction will end with sentry
though there are many more components to
the ecosystem hopefully you're feeling
more comfortable with hadoop now and
have a good idea of how these components
come together to help you access data

like never before now that we've covered hadoop's core in a bit of its broader ecosystem.

Let's talk about how cloud era fits into this picture you know we know who do well, let's talk some specifics.

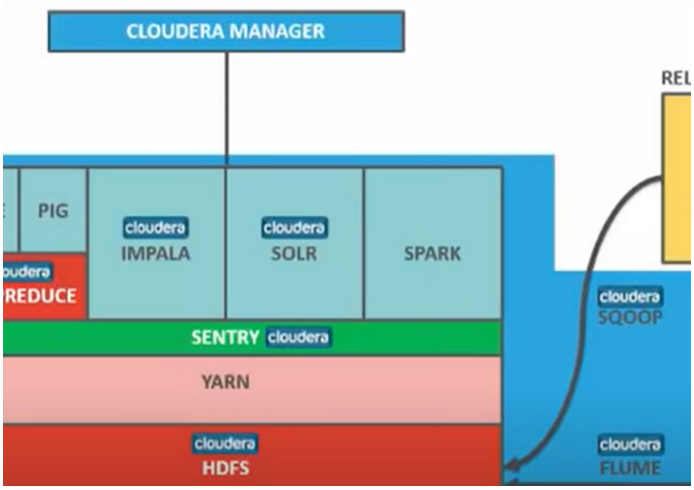


Looking at this ecosystem we all know that it's open source via the Apache foundation.
But even there the project's need to have an origin or inventor you'll see from the Cloudera logos which ones have been conceived by Cloudera there's current or former employees.
This leadership is unmatched within the open source community.
Now even with this great technology it can be pretty difficult to pull in software from open source and get it running all on your own.

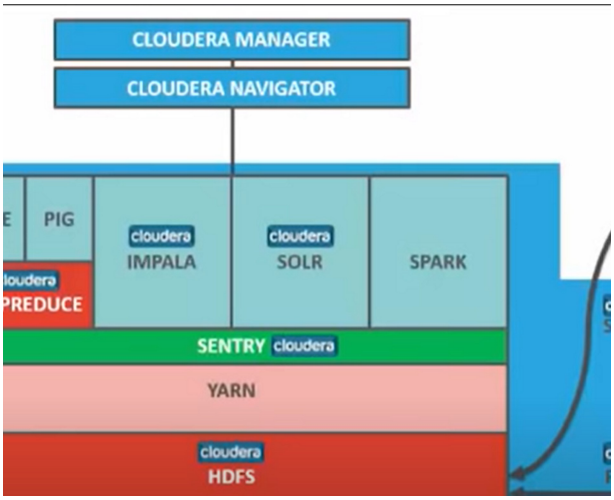


That's why Cloudera packages its own distribution of hadoop known as CDH it comes from cloud era but it's still 100% open source Apache software.
We make it easier through QA testing for quality and backwards compatibility among other things in addition to packaging the software cloud era also has a few value add Enterprise ready tools that make it much easier for you to drive value from your data.

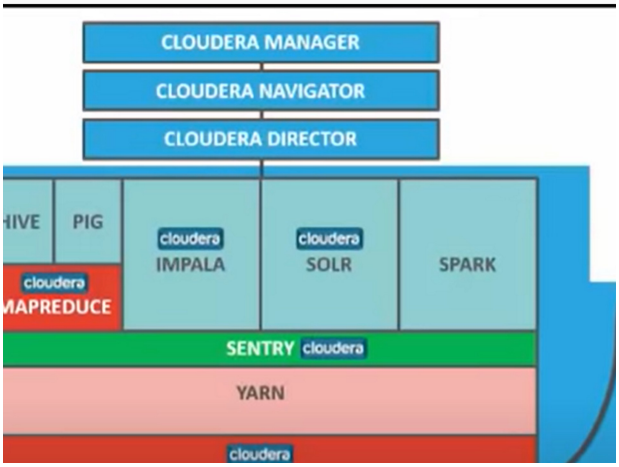
Let's take a quick tour of these before we end.



cloud era manager is a tool that manages the entire hadoop infrastructure for you.
Build into cloud era manager is the tool that enables you to configure all of the aforementioned products the deployment and management framework in cloud era manager is industry-leading.



Cloud era navigator is the only native end-to-end governance solution for apache hadoop based systems.
Through a single user interface it provides visibility for administrators data managers, data scientists, and analysts to secure govern and explore the large amounts of diverse data Atlanta hadoop.
Within navigator key trustee is a virtual safe deposit box for managing encryption keys, certificates, and passwords.
It provides software based key and certificate management that supports a variety of robust configurable and easy to implement policies governing access to the secure artifacts.
Cloud era Navigator is a part of cloud era enterprises comprehensive data security and governance offering and is a key part to meeting compliance and regulatory requirements.



Cloud era director will allow you to deploy hadoop to the cloud with best practices

determined by cloud era Hadoop experts.
Cloud era director partners tightly with
cloud era manager and in a somewhat
complex environment makes things easier
after all it's enough trouble to get
your head around cloud or hadoop when
they're on their own let alone when
they're working together.
We can make iteasy for you with the first portable
self-service tool for deploying and
managing hadoop in the cloud that brings
us to the end of today's education