



# DIGITAL TALENT SCHOLARSHIP 2019

Big Data Analytics



## Hadoop Administration dan HDFS

Oleh:

Fakultas Ilmu Komputer (Filkom) Universitas Brawijaya (UB)

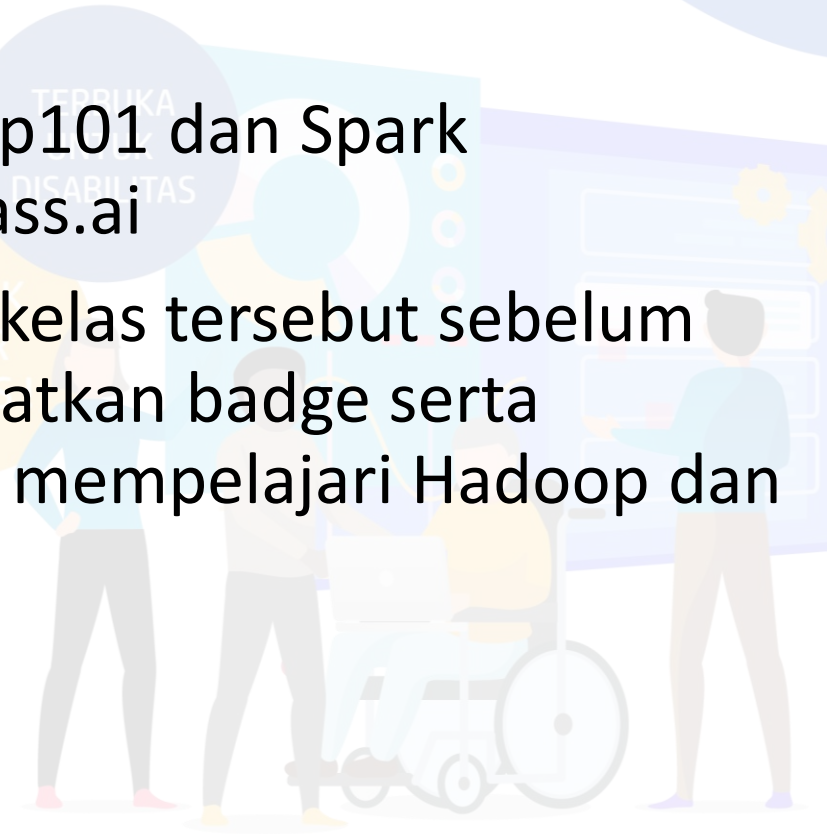
# Garis Besar

- Pendahuluan mengenai Hadoop
- Arsitektur Hadoop dan HDFS
- Hadoop Administration
- Komponen Hadoop



# Panduan

- Peserta enroll ke kelas Hadoop101 dan Spark Fundamental 1 di cognitiveclass.ai
- Peserta wajib menyelesaikan kelas tersebut sebelum materi pekan 25 dan mendapatkan badge serta sertifikat kelas → tanda telah mempelajari Hadoop dan Spark



# Pengenalan pada Hadoop

Big Data  
Hadoop



# Kenapa Hadoop?

- Dunia digital yang mengarah ke Big Data
- Mulanya, data masih bisa diproses oleh PC  
Mudah diproses, ukuran tidak terlalu besar  
Seiring waktu, data bertambah besar
- Data dan operasinya tidak bisa ditangani resource hardware yang ada
- 80% data yang tersebar di dunia digital/internet adalah unstructured data

TERBUKA  
UNTUK  
DISABILITAS

PELAYANAN

KELOMPOK

KELOMPOK

KELOMPOK

KELOMPOK

KELOMPOK

KELOMPOK

KELOMPOK

KELOMPOK

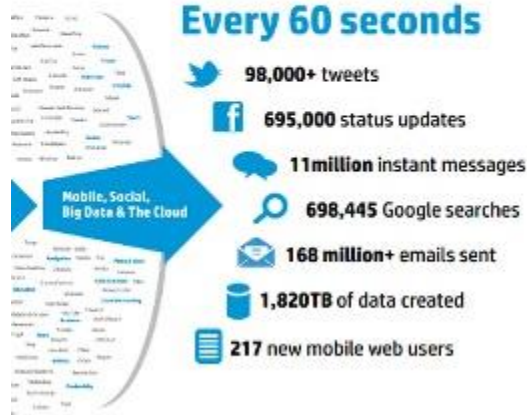
# Big Data

TERBUKA  
UNTUK

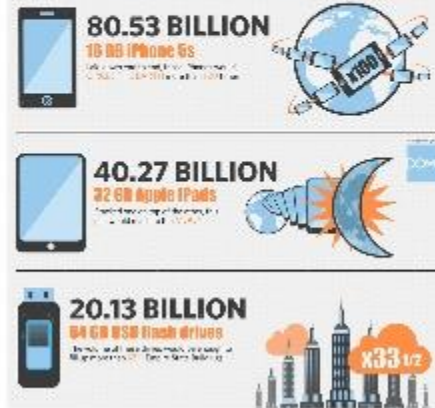
## CONTEXT: WHAT'S BIG DATA?

6

## HOW BIG IS BIG?



creates enough data to fill



in 1 year!

<https://www.ibm.com/cloud/2013/08/18/ibm-forecast-2013-data/>

# Big Data

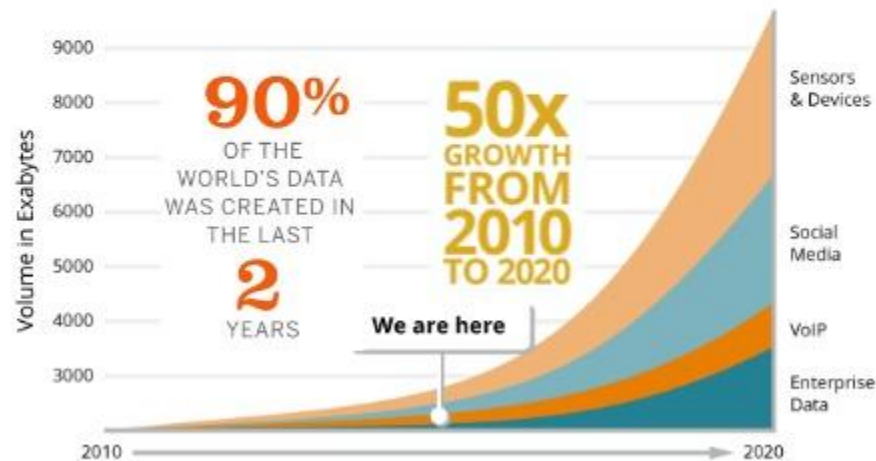
TERBUKA  
UNTUK

## CONTEXT: WHAT'S BIG DATA?

7

## BIG IN GROWTH, TOO.

1 exabyte (EB) = 1,000,000,000,000,000 bytes



Source: International Data Corporation (IDC), "Worldwide Datacenter Spending Forecast, 2014-2019," 2014.


Source: International Data Corporation (IDC), "Worldwide Datacenter Spending Forecast, 2014-2019," 2014.

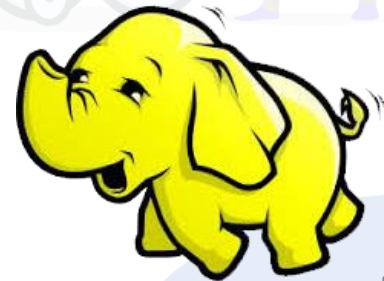




# Hadoop (1)



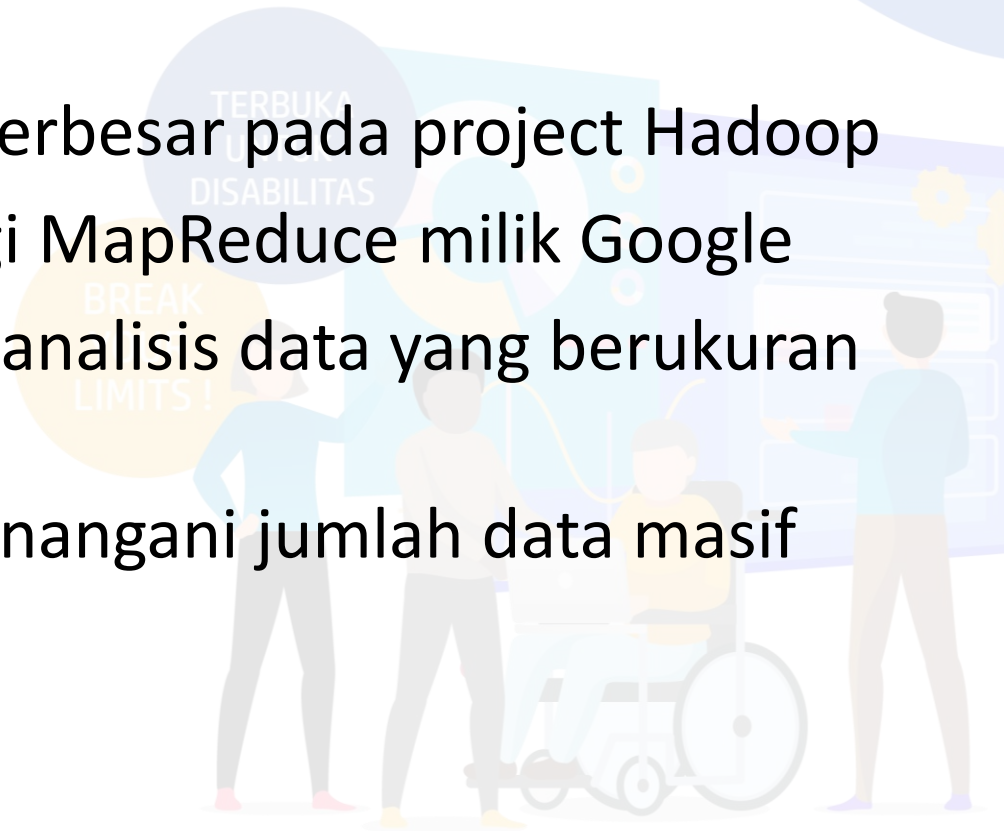
- Open source project dari Apache Foundation
- Sebuah framework yang dibuat menggunakan Java untuk distributed computing dan penyimpanan data yang reliabel dan skalabel
- Dikembangkan oleh Doug Cutting pada tahun 2005 untuk Nutch 
- Didanai oleh Yahoo dan pada tahun 2006 dihibahkan kepada Apache





# Hadoop (2)

- Yahoo masih pendana terbesar pada project Hadoop
- Menggunakan teknologi MapReduce milik Google
- Solusi pemrosesan dan analisis data yang berukuran sangat besar
- Dioptimalkan untuk menangani jumlah data masif
  - Structured
  - Unstructured
  - Semi Structured



# Hadoop (3)

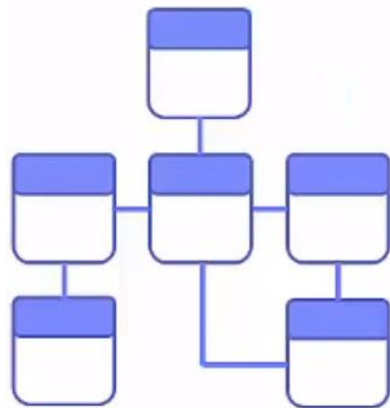
- Menggunakan resource hardware yang moderat
- Performa cepat dengan menggunakan paralel processing
  - \*Catatan: Hadoop menggunakan batch operation, untuk ukuran data yang masif → response time lambat → tidak bisa update instan, namun bisa menambahkan data
- Case: Jika data tidak konsisten, maka Hadoop mereplikasi data ke semua komputer yang ada dan jika 1 mati/hilang, data akan diproses di komputer lainnya

# Hadoop pada Proses

- Tidak cocok untuk OnLine Transaction Processing (OTLP)  
<data diakses secara random pada structured data seperti relational database>
- Tidak cocok untuk OnLine Analytical Processing (OLAP) atau Decision Support System (DSS)  
<data diakses secara sekuensial pada structured data seperti relational database untuk menghasilkan laporan business intelligence>

# Hadoop pada Proses (Lanjutan)

- Komplemen (Pelengkap) dari OTLP dan OLAP, bukan pengganti relational database

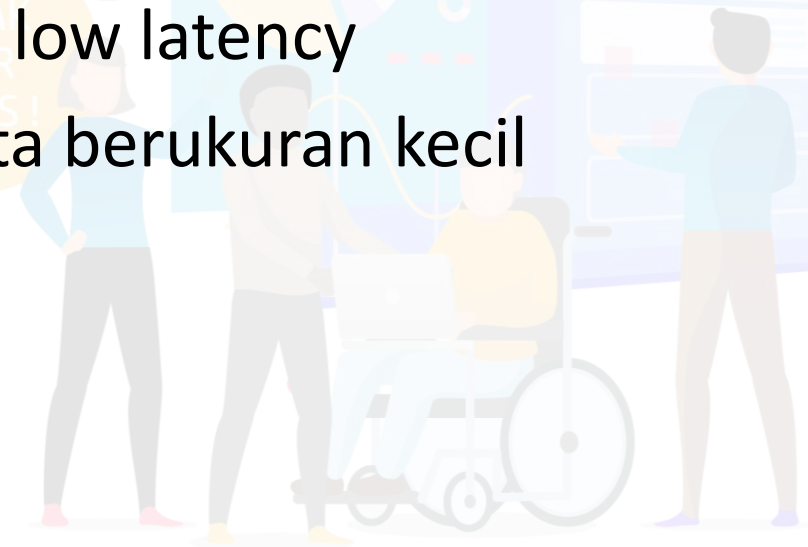


≠



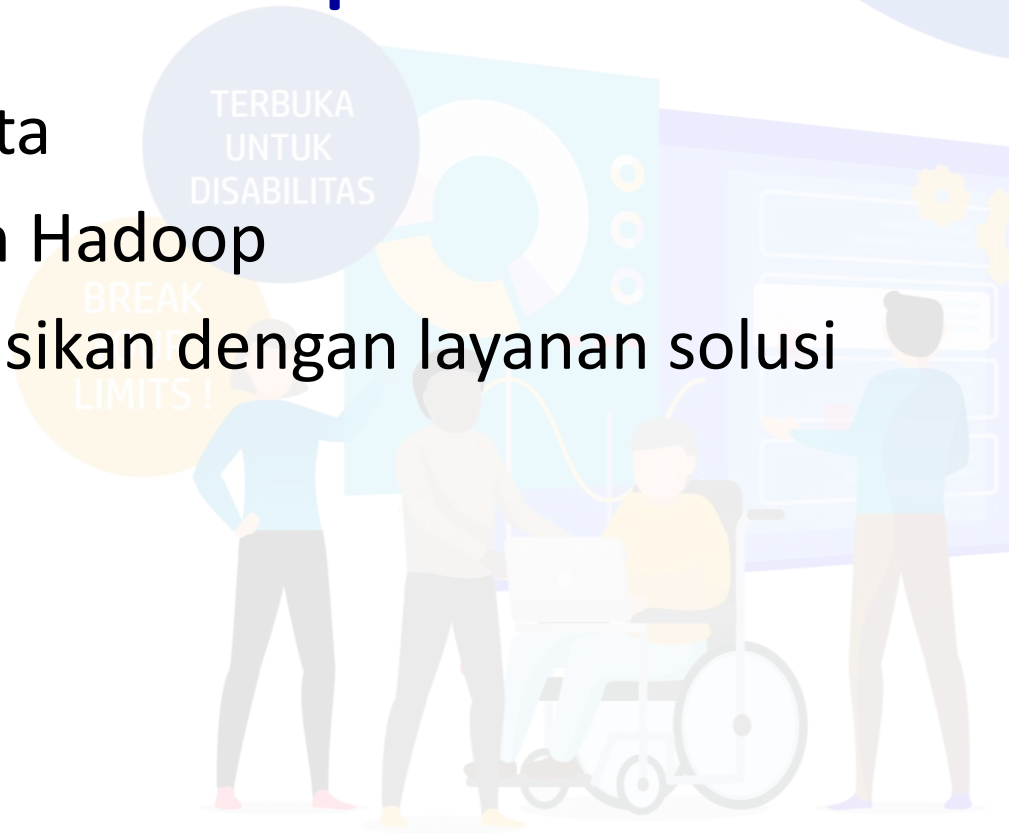
# Hadoop pada Data

- Tidak cocok untuk data yang saling berketerkaitan (tidak bisa diparalelisasikan karena tidak independen)
- Tidak cocok untuk akses data low latency
- Tidak cocok untuk banyak data berukuran kecil



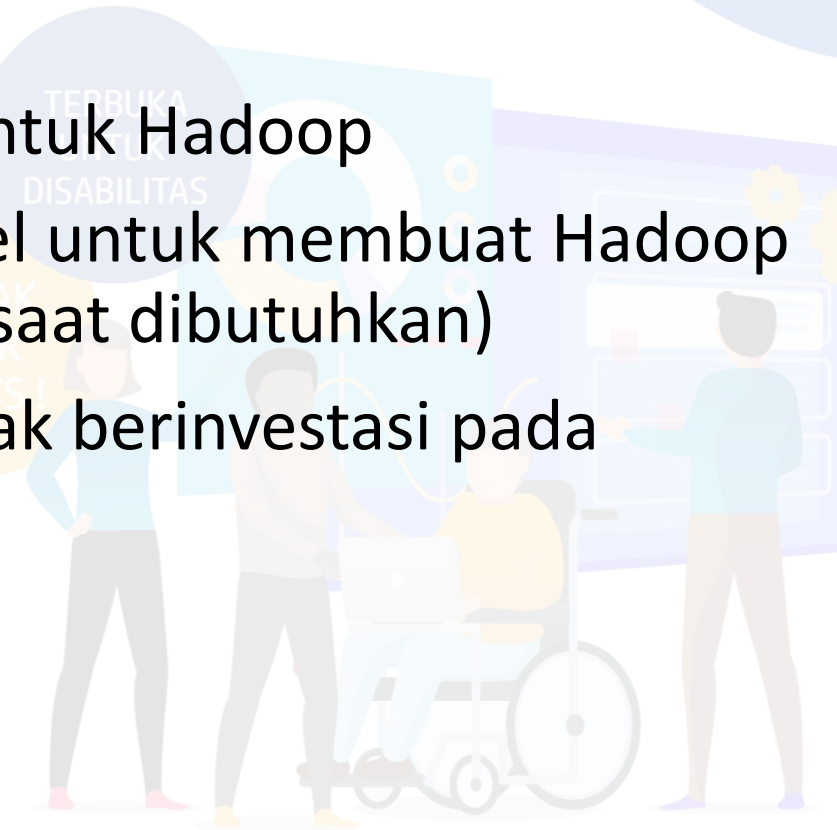
# Big Data dengan Hadoop?

- Salah satu solusi Big Data
- Banyak solusi lain selain Hadoop
- Hadoop dapat diintegrasikan dengan layanan solusi analitik lainnya



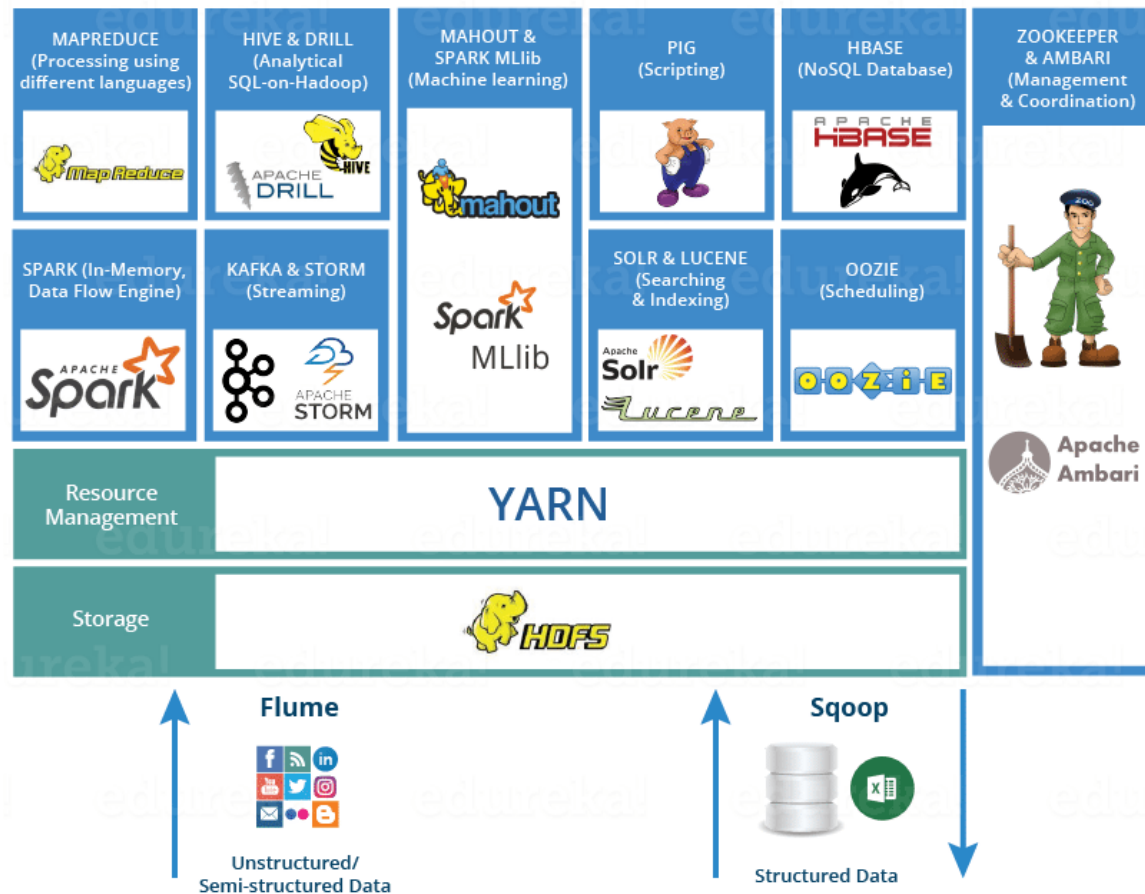
# Hadoop pada Cloud

- Pasangan yang sangat baik untuk Hadoop
- Fleksibel, simpel, dan skalabel untuk membuat Hadoop cluster (on-demand, dipakai saat dibutuhkan)
- Biaya lebih murah karena tidak berinvestasi pada hardware





# Hadoop Framework



# Project Hadoop (1)

- **Ambari™**: tool berbasis web untuk manajemen, provisioning, dan monitoring Hadoop
- **Avro™**: sistem sterilisasi data
- **Cassandra™**: database multi-master database skalabel tanpa kesalahan
- **Chukwa™**: sistem koleksi data untuk mengelola distributed system yang besar
- **HBase™**: distributed database skalabel mendukung structured data untuk tabel besar

# Project Hadoop (2)

- **HDFS™** (Hadoop Distributed File System): sistem file terdistribusi yang menyediakan akses throughput tinggi untuk data aplikasi
- **Hive™**: infrastruktur data warehouse untuk peringkasan data dan adhoc query
- **Mahout™**: library machine learning dan data mining skalabel
- **MapReduce**: sistem berbasis YARN untuk memparalelisasikan dataset besar
- **Ozone**: sebuah object store untuk Hadoop

# Project Hadoop (3)

- **Pig<sup>TM</sup>**: bahasa data-flow tingkat tinggi dan framework eksekusi untuk komputasi paralel
- **Spark<sup>TM</sup>**: compute engine cepat dan umum untuk data Hadoop Menyediakan programming model yang simpel dan ekspresif serta mendukung secara luas aplikasi, ETL (extract, transform, load), machine learning, stream processing, dan komputasi graph
- **Submarine**: engine machine learning untuk Hadoop

TERBUKA  
UNTUK  
DISABILITAS

# Project Hadoop (4)

- **Tez<sup>TM</sup>**: framework programming dengan generalized data-flow, berbasis YARN, dan menyediakan engine yang powerful dan fleksibel untuk eksekusi DAG (Directed Acyclic Graph) tasks untuk memproses data untuk batch dan interactive usecase  
Tex diadopsi oleh Hive, Pig, dll untuk menggantikan MapReduce
- **YARN**: framework untuk job scheduling dan manajemen resource cluster
- **ZooKeeper<sup>TM</sup>**: layanan koordinasi performa tinggi untuk aplikasi terdistribusi

# Implementasi Hadoop (1)

## IBM Watson

- Supercomputer IBM yang mengalahkan 2 orang yang paling sering menang di acara Jeopardy
- Menggunakan data dari 200 juta halaman teks input
- Menggunakan Hadoop untuk distribusi workload ke memori



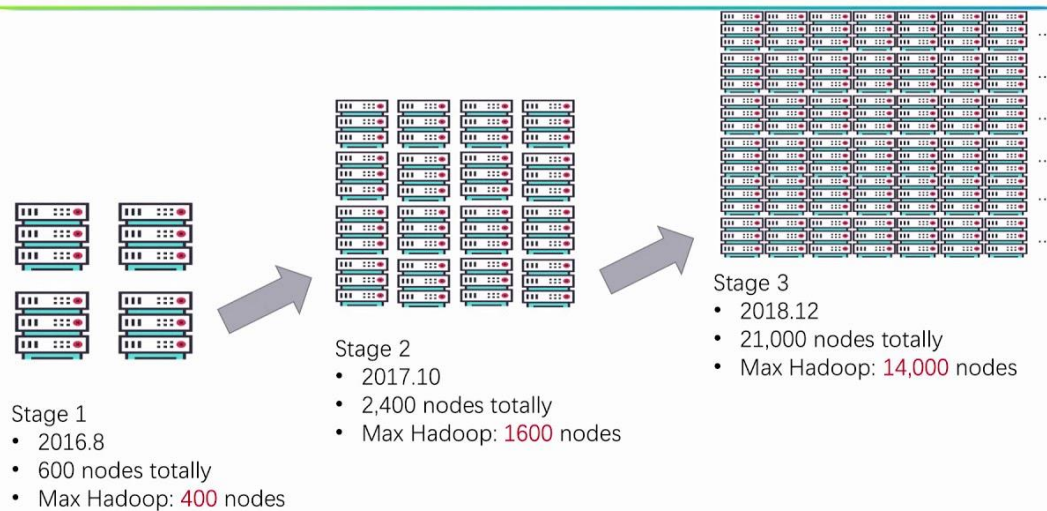


# Implementasi Hadoop (2)

## China Mobile

- Membuat Hadoop cluster untuk data mining dari Call Data Records (5-8 TB data setiap hari)
- Memproses data 10x lebih banyak dan 5x lebih cepat dari sistem lama tanpa Hadoop

### Brief History of CBA Project





# Implementasi Hadoop (3)

## New York Times

- Mengonversi semua berita dari tahun 1851-1922 ke dalam web dari (11 juta gambar  $\cong$  4TB hingga 1,5TB PDF)
- Diselesaikan oleh 1 orang dalam 24 jam pada 100-instance Amazon EC2

## The New York Times

- 1 topic
- 1 partition
- Contains every article published since 1851
- Multiple producers / consumers

Example for  
Stream / Table Duality

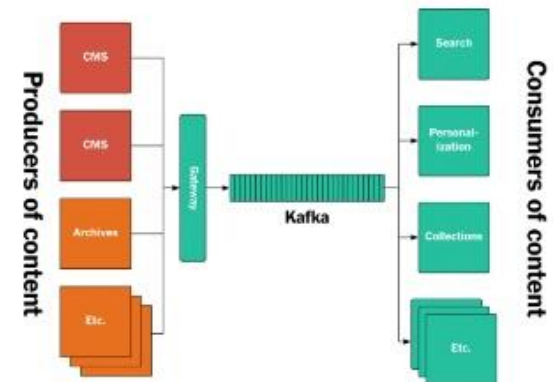
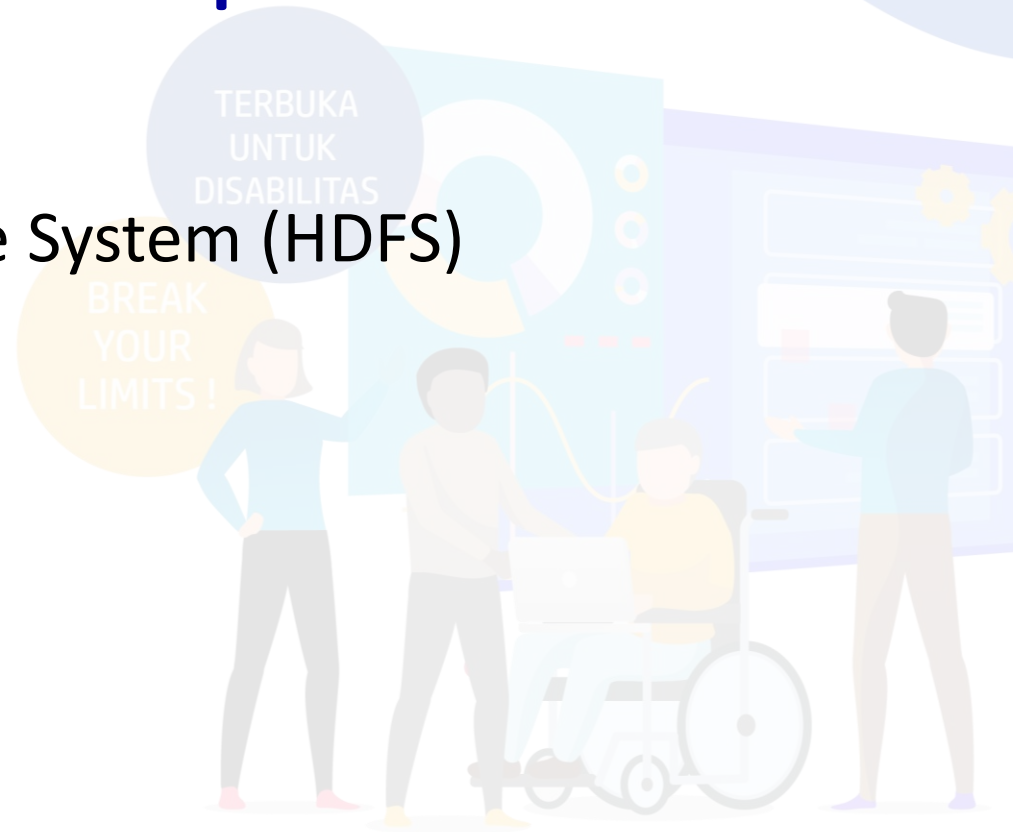


Figure 1: The new New York Times log/Kafka-based publishing architecture.  
<https://www.confluent.io/blog/publishing-apache-kafka-new-york-times/>

# Arsitektur Pre Hadoop 2.2

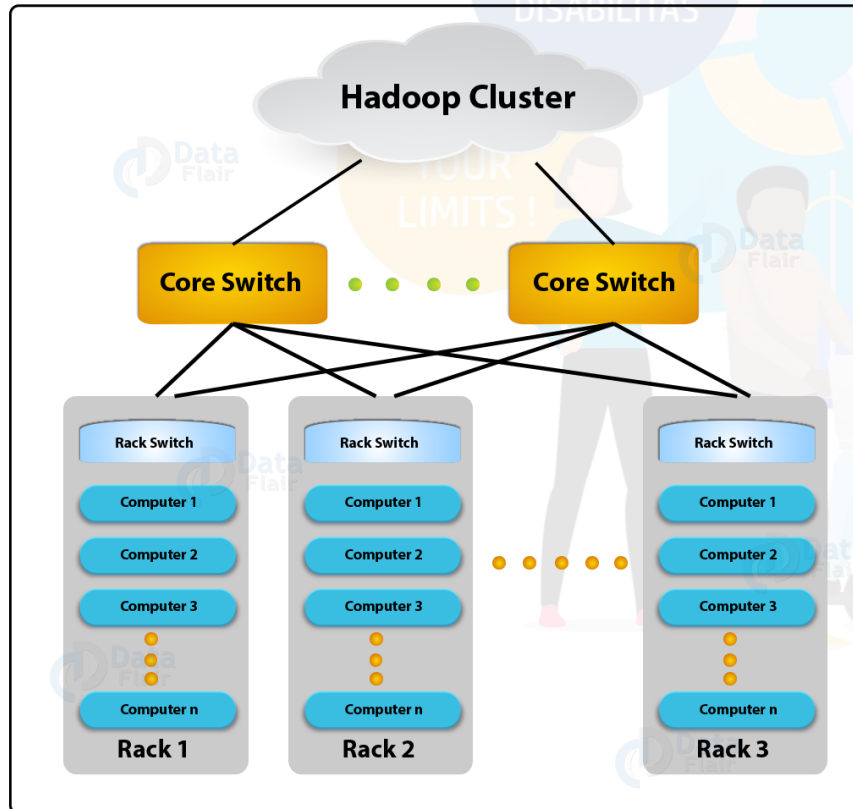
- Hadoop Cluster
- Hadoop Distributed File System (HDFS)
- Hadoop Blocks
- Hadoop Nodes



# Hadoop Cluster – Terminology Review



## Hadoop Cluster



# Arsitektur Pre Hadoop 2.2

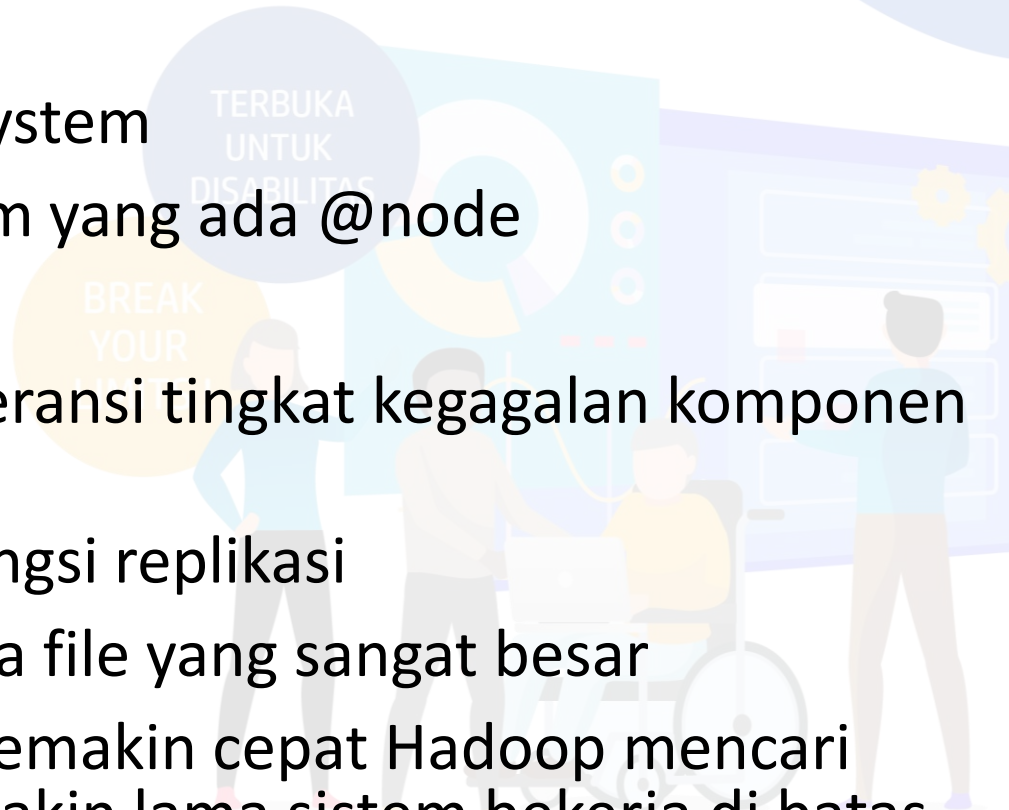
- Dua Komponen Utama
  - ✓ **Distributed File System**
    - Hadoop Distribution File System (HDFS)
  - ✓ **MapReduce Engine (Map Reduce V1)** \*dibahas pada sesi selanjutnya
    - framework untuk melakukan kalkulasi pada data di file system
    - Memiliki resource manager dan scheduler

TERBUKA  
UNTUK  
DISABILITAS

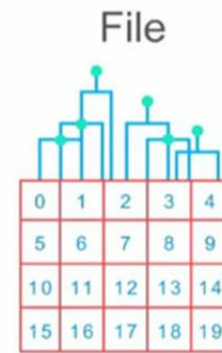
BREAK  
LIMITS!

# Hadoop Distributed File System (HDFS) (1)

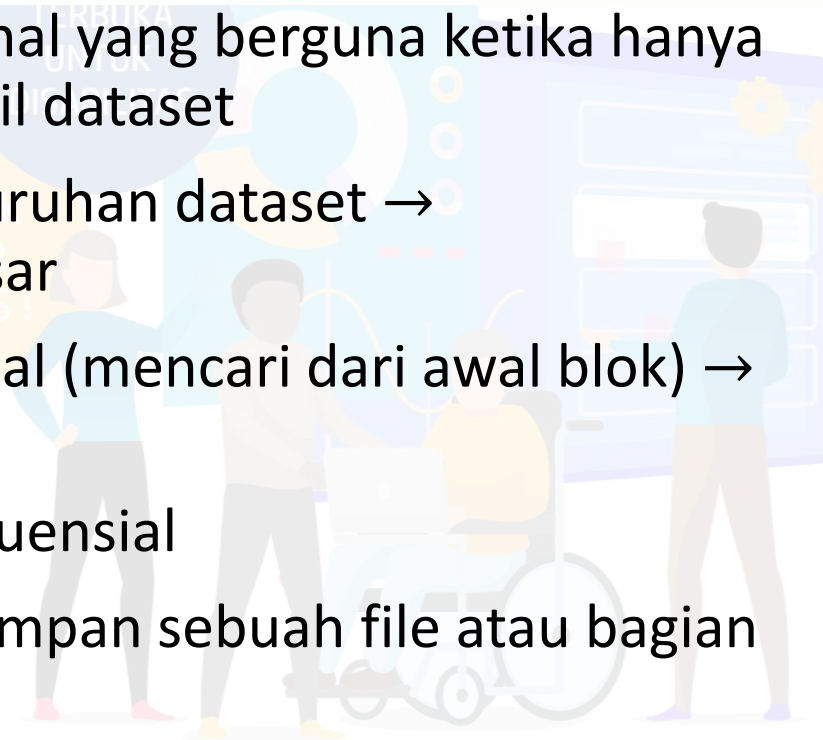
- Terinspirasi Google File System
- Berjalan di atas file system yang ada @node
  - Bukan POSIX
  - Didesain untuk menoleransi tingkat kegagalan komponen yang tinggi
    - ✓ Reliabel dengan fungsi replikasi
- Didesain untuk mengelola file yang sangat besar
  - Semakin besar file → semakin cepat Hadoop mencari data berikutnya → semakin lama sistem bekerja di batas kecepatan tertinggi



# HDFS (2)



- Pencarian lokasi data sangat mahal yang berguna ketika hanya untuk menganalisis sebagian kecil dataset
- Karena Hadoop bekerja di keseluruhan dataset → menggunakan file berukuran besar
- Tidak random access → sekuensial (mencari dari awal blok) → lebih sedikit pencarian data
- Cocok untuk data streaming/sekuensial
- Menggunakan blocks untuk menyimpan sebuah file atau bagian dari sebuah file



# HDFS File Blocks

- Tidak sama dengan file block pada OS
- Default ukuran Hadoop Block → 64MB ~128MB (rata-rata 128MB ~ lebih)
- Ukuran sebuah file bisa lebih besar dari 1 disk pada cluster – 1 file dibagi ke beberapa blok dan disebar ke beberapa node
- Jika sebagian file lebih kecil dari ukuran block → hanya ruang yang dibutuhkan yang digunakan
- Block bekerja baik dengan replikasi

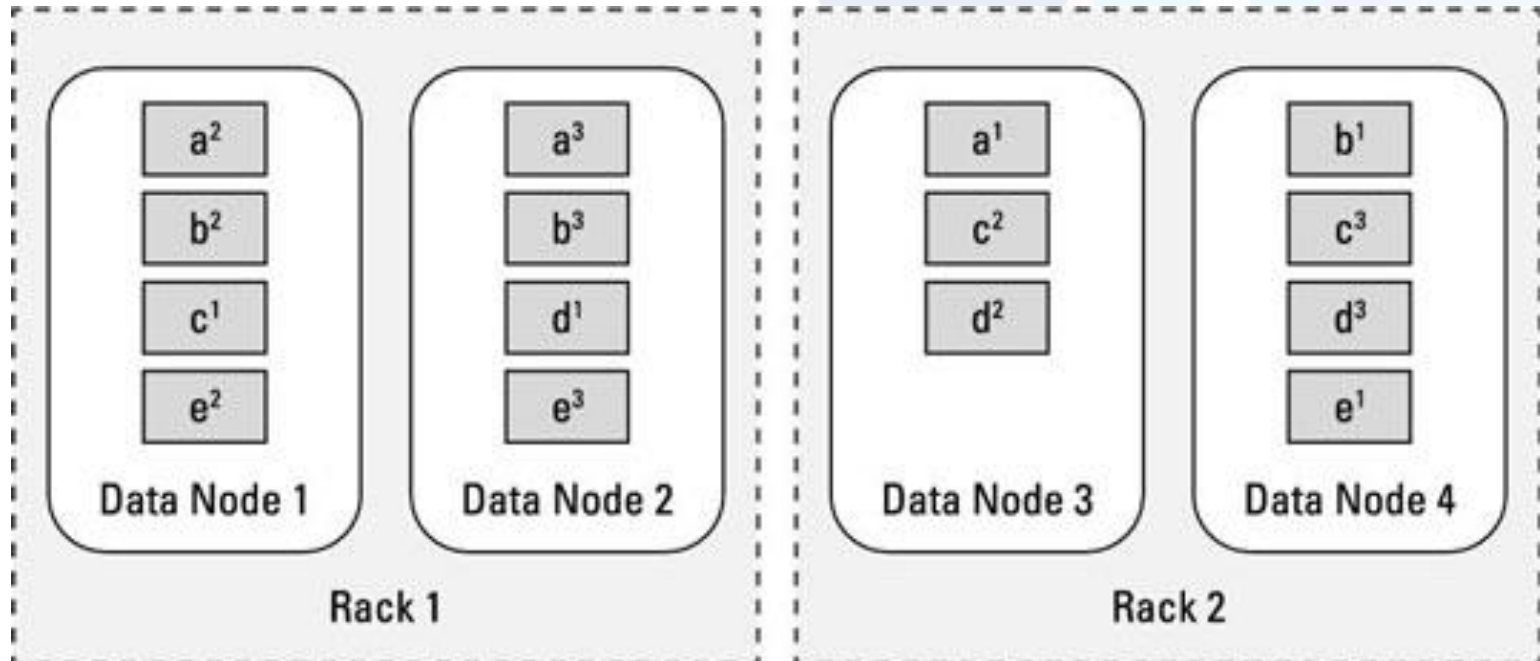
**Contoh sebuah file  
450MB**

128M B	128M B	128M B	66MB
-----------	-----------	-----------	------



# Hadoop Blocks dengan Replikasi

TERBUKA  
UNTUK  
DISABILITAS



# Framework MapReduce

\*dibahas lebih lanjut pada sesi berikutnya

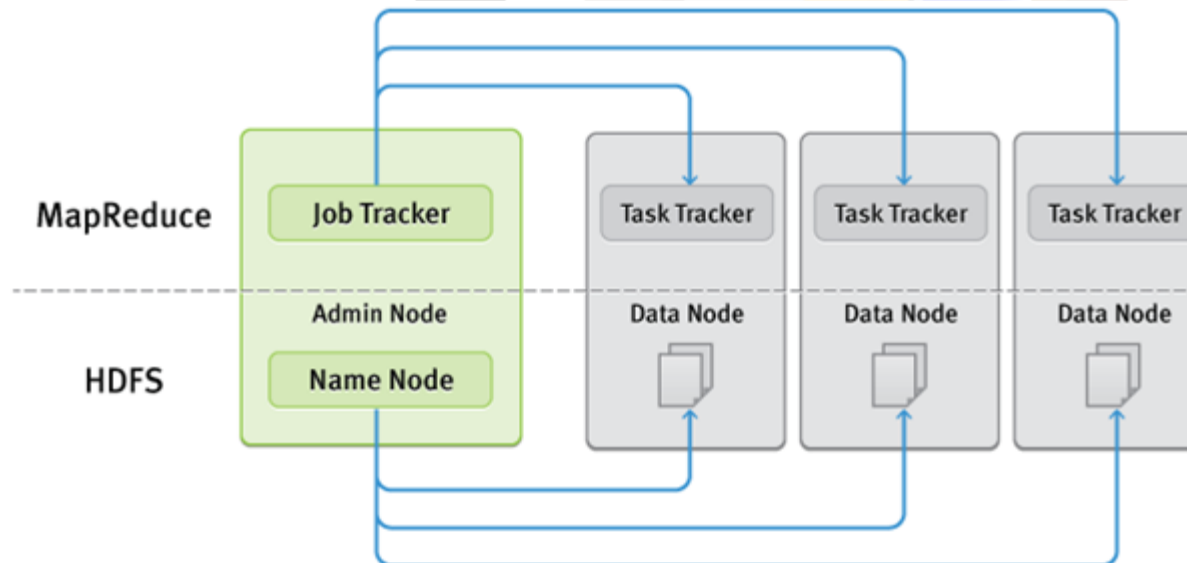
- Berbasis Google MapReduce
- Memproses dataset besar untuk beberapa jenis masalah yang dapat didistribusikan menggunakan banyak node
- Terdiri dari 2 transformasi: map dan reduce  
Dapat berjalan secara paralel ← distributed processing
  - Map: memetakan task-task yang berjalan secara paralel
  - Reduce: mengurangi task-task yang berjalan secara paralel

TERBUKA  
UNTUK  
DISABILITAS

YOUR

# Tipe-tipe Node pada Hadoop Cluster

- Dibagi menjadi 2, node HDFS atau MapReduce v1
  - HDFS → NameNode dan DataNode
  - MapReduce v1 → JobTracker dan TaskTracker
- Ada node HDFS lain:
  - Secondary NameNode
  - Checkpoint
  - Backup



# NameNode

- Hanya 1 per cluster
- Memiliki file → beberapa DataNode berisi block-block data
- Mengelola namespace sistem file dan metadata
- NameNode → single point of failure
  - sebaiknya direplikasi/mirror fisik
  - menggunakan hardware enterprise → max reliability
  - memiliki RAM yang besar ← menyimpan semua metadata filesystem

TERBUKA  
UNTUK  
DISABILITAS

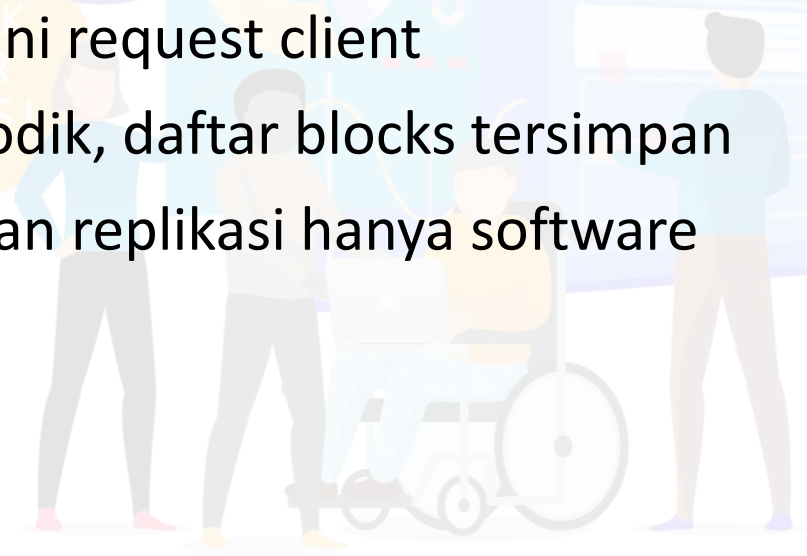
IDEAL  
YOUR  
UNIVERSITY

# DataNode

- Banyak per cluster
- Blocks dari file berbeda bisa disimpan → DataNode yang sama
- Mengelola blocks data dan melayani request client
- Update ke NameNode secara periodik, daftar blocks tersimpan
- Tidak perlu hardware enterprise dan replikasi hanya software

TERBUKA  
UNTUK  
DISABILITAS

BREAK  
YOUR  
LIMITS



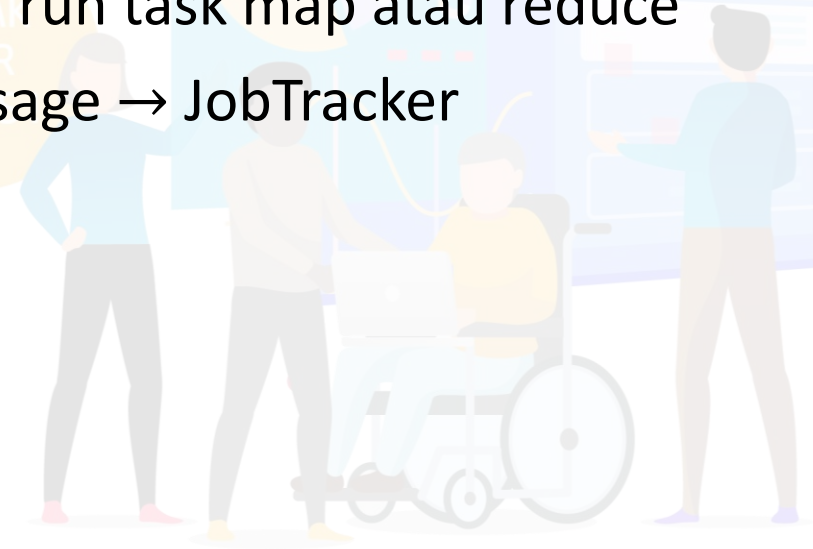
# JobTracker

- Mengelola MapReduce jobs
- Hanya 1 per cluster
- Menerima tugas dari client → scheduling Map dan Reduce → TaskTracker yang tepat (data disimpan <rack-aware manner>)
- Monitoring task gagal yang perlu direschedule pada TaskTracker berbeda

TERBUKA  
UNTUK  
DISABILITAS

# TaskTracker

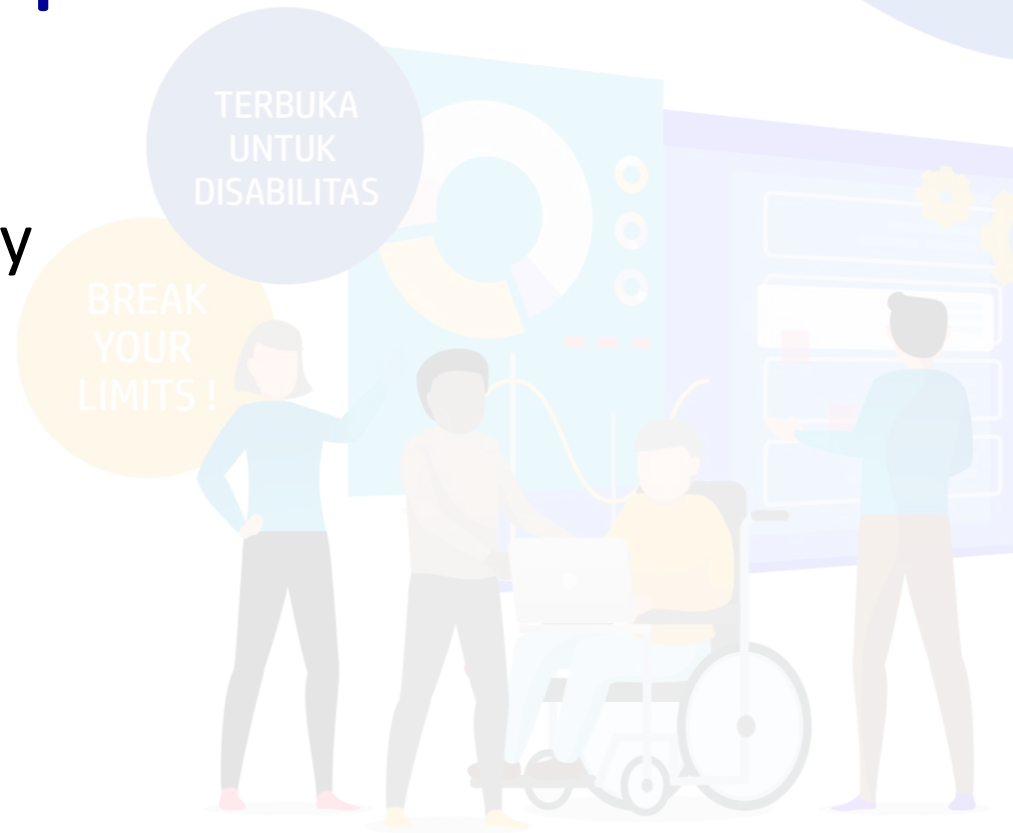
- Banyak TaskTracker per cluster → paralelisme task map dan reduce
- Tiap TaskTracker → call Java VM → run task map atau reduce
  - Berkomunikasi → heartbeat message → JobTracker
  - Membaca blocks dari DataNode





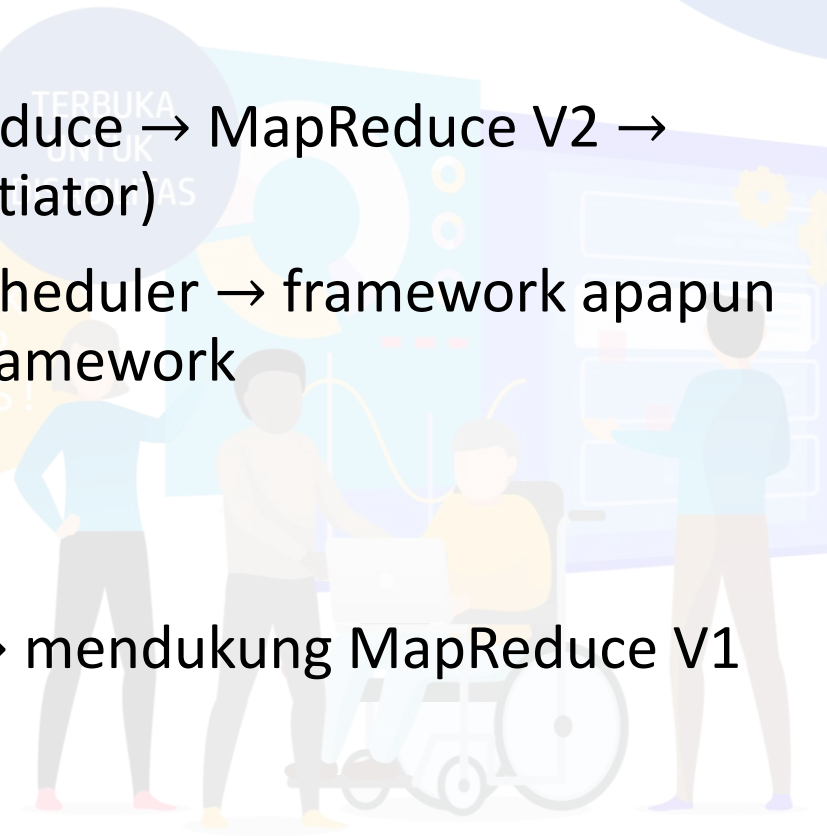
# Arsitektur Hadoop 2.2

- YARN
- Hadoop High Availability
- Topology Awareness
- Proses HDFS



# Arsitektur Hadoop 2.2

- Perubahan arsitektur pada MapReduce → MapReduce V2 → YARN (Yet Another Resource Negotiator)
- Resource manager dan external scheduler → framework apapun ↑ agar tidak ada kompetisi antar framework
- DataNodes masih eksis
- JobTracker dan TaskTracker hilang
- Tidak perlu Hadoop 2.2 ← **YARN** → mendukung MapReduce V1



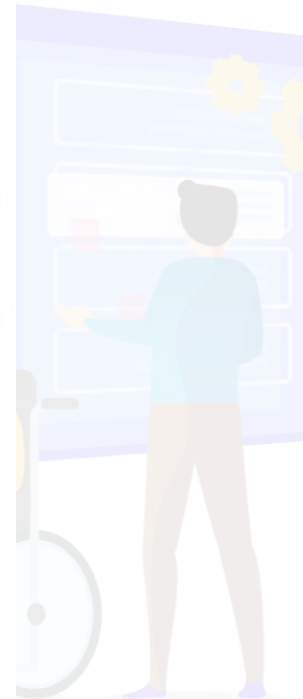
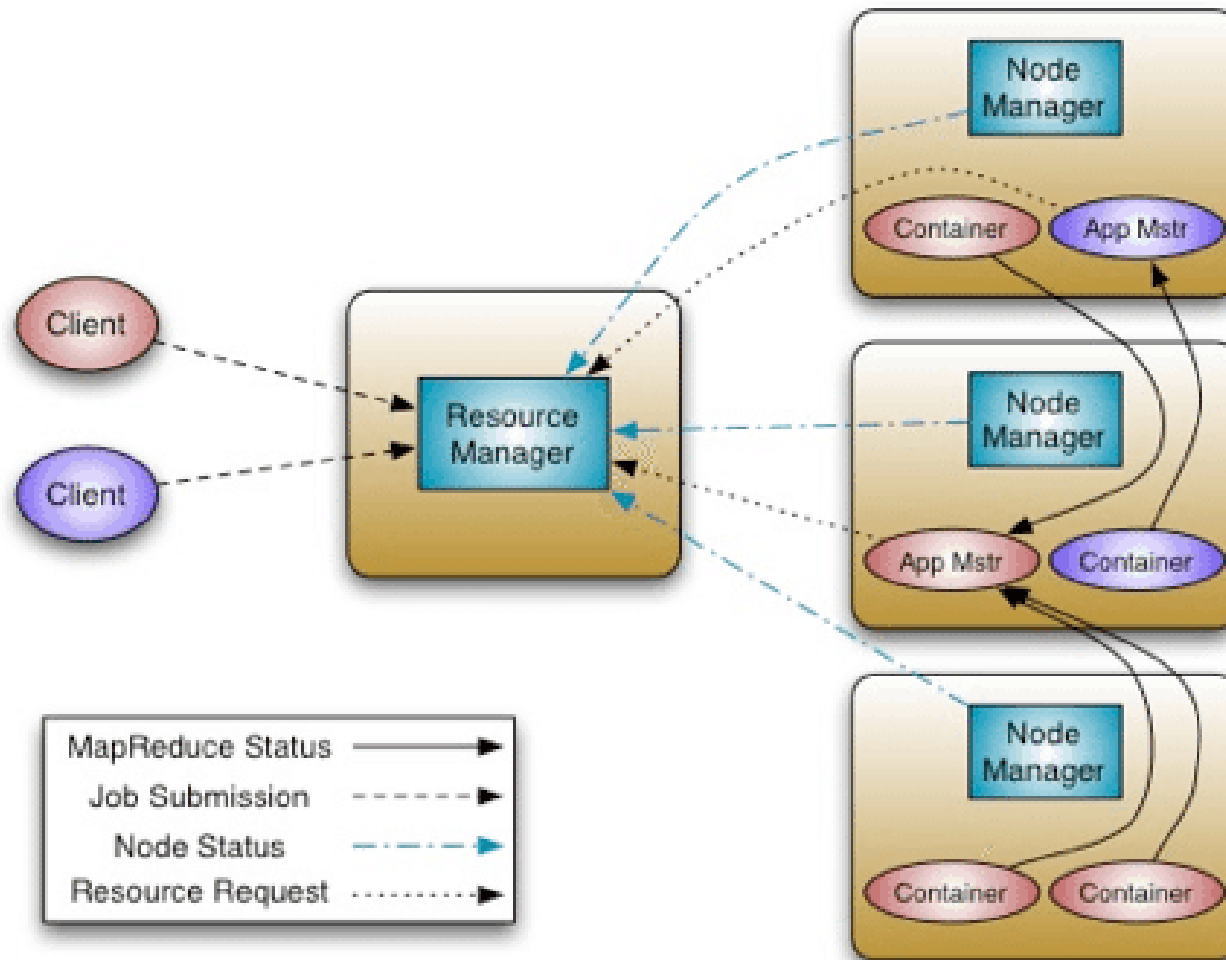
# YARN (1)

- Dua ide utama
  - Menyediakan generic scheduling dan manajemen resource
    - ✓ Tidak hanya MapReduce
    - ✓ Tidak hanya batch processing
  - Menyediakan scheduling dan manajemen workload yang lebih efisien
    - ✓ Tidak perlu define map slot dan reduce slot tiap node
      - ❖ MapReduce V1 membutuhkan jumlah slot map dan reduce
        - ↑ tiap node cluster bervariasi ← membatasi jumlah task

TERBUKA  
UNTUK  
DISABILITAS

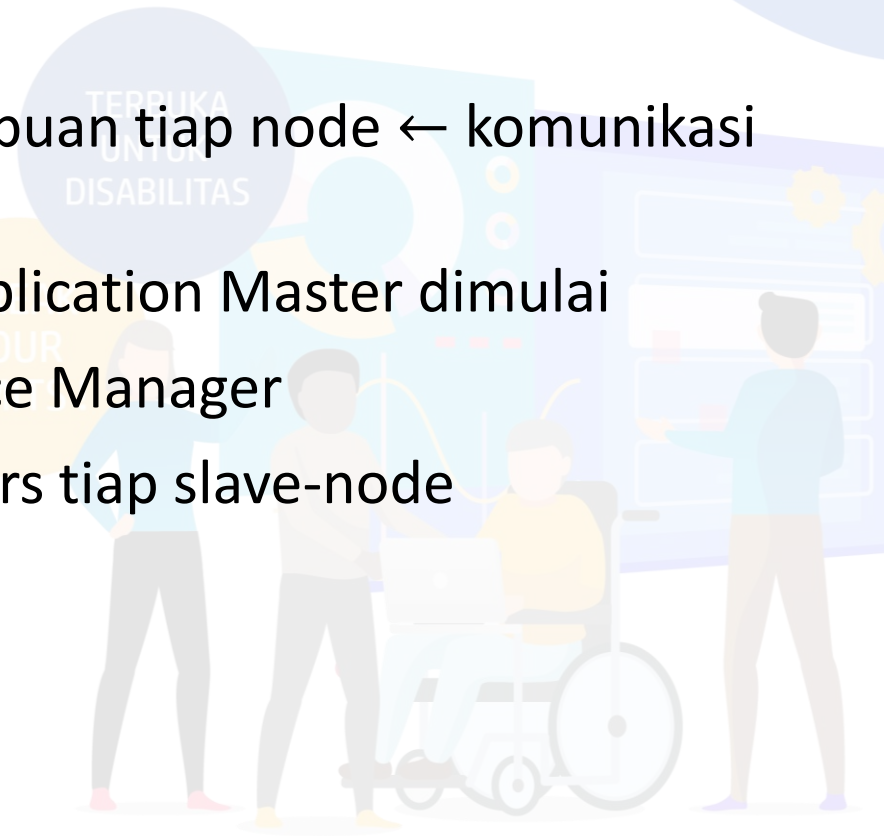
BREAK  
YOUR  
LIMITS!

# YARN Overview Diagram



## YARN (2)

- Resource Manager tahu kemampuan tiap node ← komunikasi NodeManager
  - Ketika aplikasi dipanggil → Application Master dimulai
  - Bertanggung jawab jawab → Resource Manager
  - Resource dan task → Containers tiap slave-node
  - Tidak lagi “one size fits all”

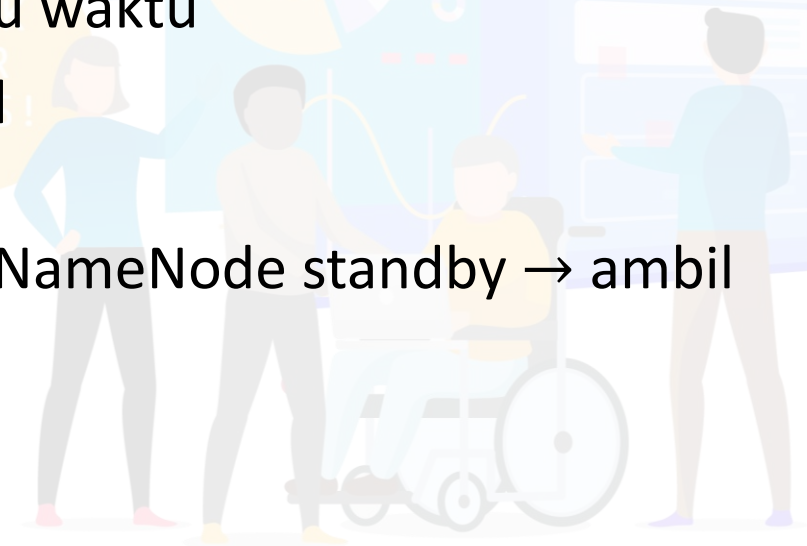


# NameNode pada YARN

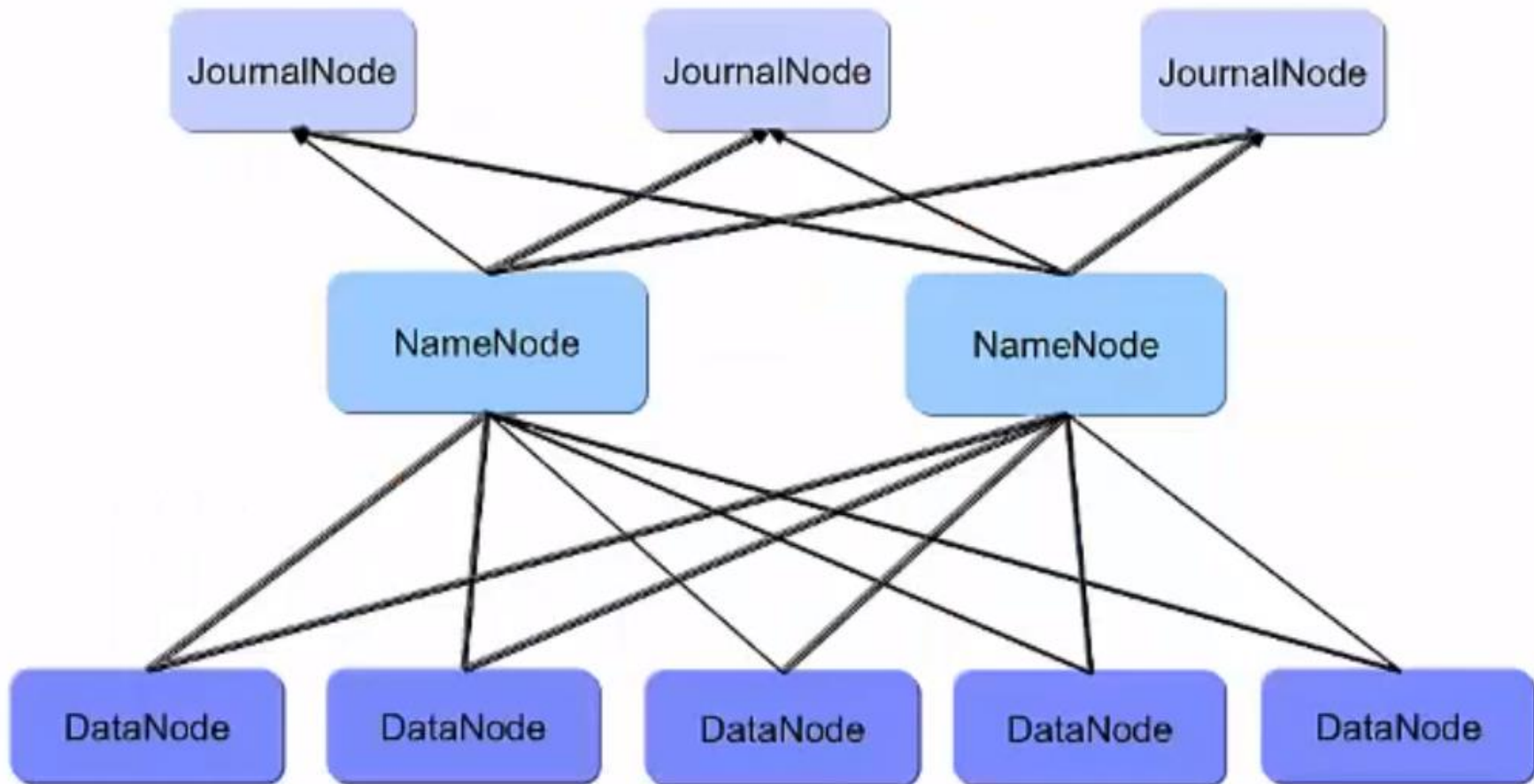
- Mendukung high availability
- Sekarang, ada 2 NameNode; 1 aktif dan 1 standby  
Hanya 1 NameNode aktif pada satu waktu
- JournalNode, minimal 3 atau ganjil !  
Menentukan NameNode aktif
- Jika NameNode aktif gagal/down, NameNode standby → ambil alih

TERBUKA  
UNTUK  
DISABILITAS

YOUR



# Hadoop High Availability



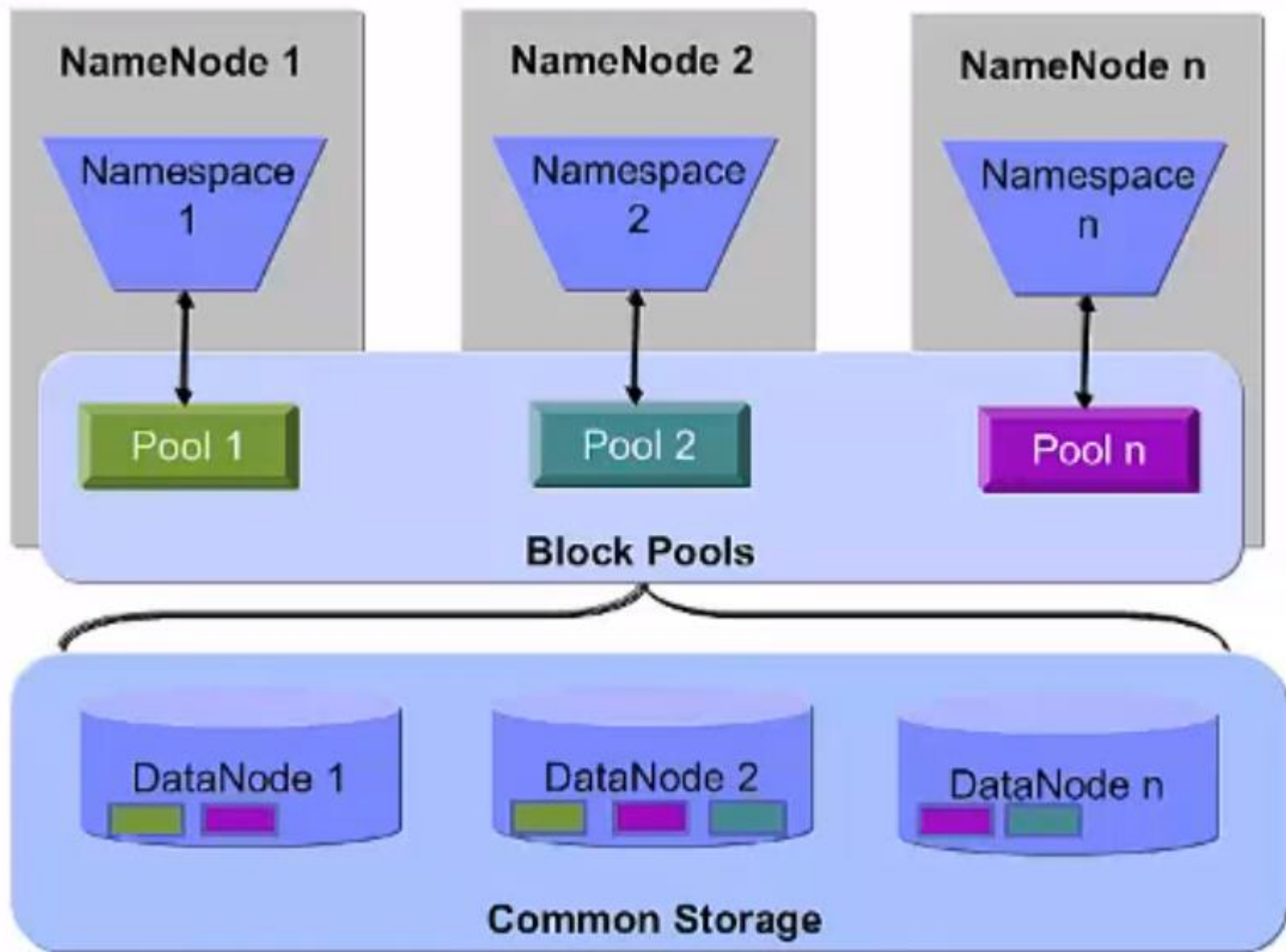


# NameNode Memory Limitation

- NameNode butuh banyak RAM untuk metadata semua data pada tiap cluster
- Terbatas ← model vertical growth
- Hadoop Federation → memperbolehkan model horizontal → memanfaatkan beberapa NameNode ← bekerja secara independen, berbagi semua DataNode
- Tiap NameNode punya namespace masing-masing, file sesuai namespace

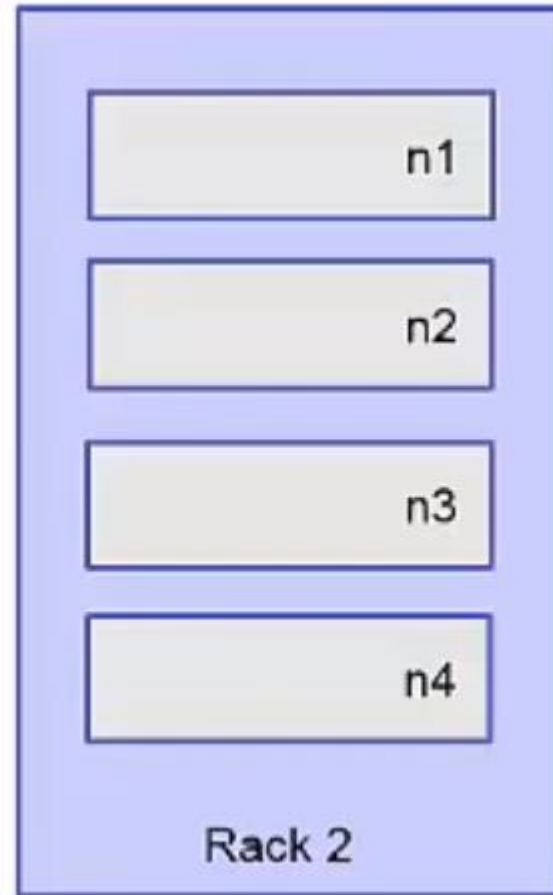
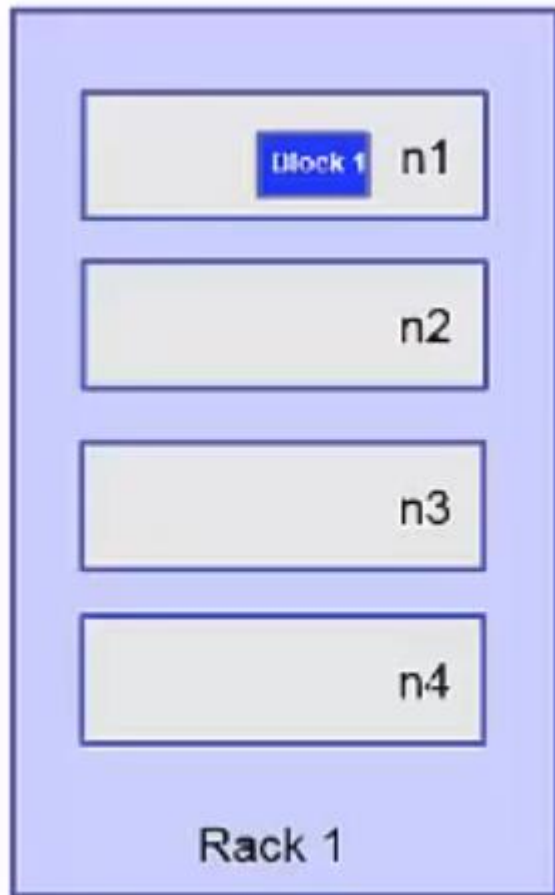
TERBUKA  
UNTUK  
DISABILITAS

TAKE  
YOUR

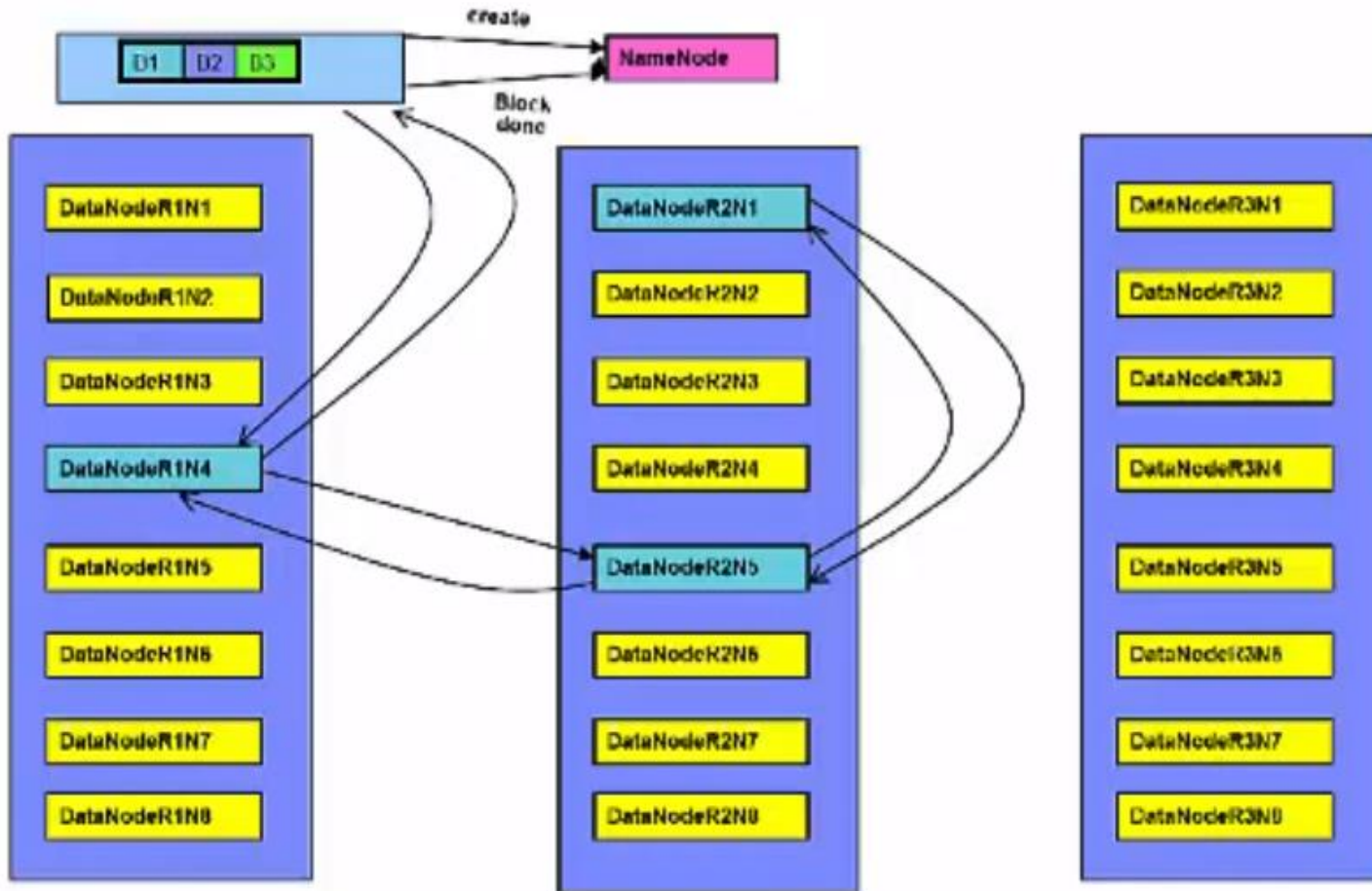


# Topology Awareness

- Hadoop aware akan topologi jaringan
- Menempatkan task sedekat mungkin dengan data → max bandwidth
- Ketika menentukan TaskTracer yang akan menerima MapTask untuk membaca data,
  - Opsi terbaik, pilih TaskTracer pada node sama
  - Jika tidak bisa, pilih pada node lain di rak yang sama
  - Opsi terburuk, diproses oleh node di rak berbeda
- Ketika rack-awareness menyala pada cluster → Hadoop akan selalu memilih node bandwidth tertinggi



# Replikasi HDFS (1)



# Replikasi HDFS (2)

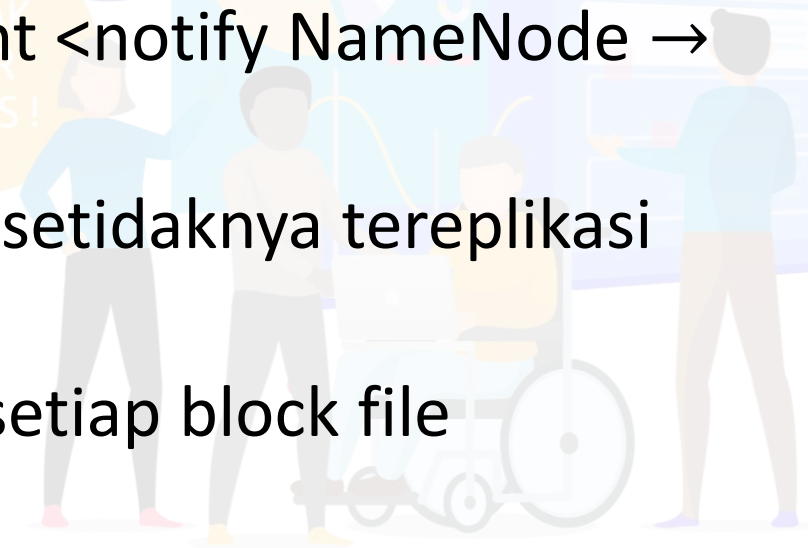
- Client request “create” → NameNode
- NameNode cek eksis? Jika tidak, → client permitted write file
- NameNode menentukan lokasi block pertama DataNode
- Jika client = @DataNode → pilih DataNode tersebut  
Jika tidak, pilih DataNode random
- Default, data direplikasi ke 2 tempat lain di cluster
- Pipeline terbentuk ke tiga DataNode
- DataNode kedua → random di rak yang berbeda (redundancy)
- DataNode ketiga → random di rak yang sama DataNode kedua, node berbeda



# Replikasi HDFS (3)

- Data disalurkan dari DataNode 2 ke 3
- Untuk menjamin write sukses, ACK dikirim dari 3 ke 2, dari 2 ke 1, dan dari 1 ke client <notify NameNode → proses write selesai>
- NameNode mengecek block, setidaknya tereplikasi sebelum merespon
- Proses replikasi terjadi pada setiap block file

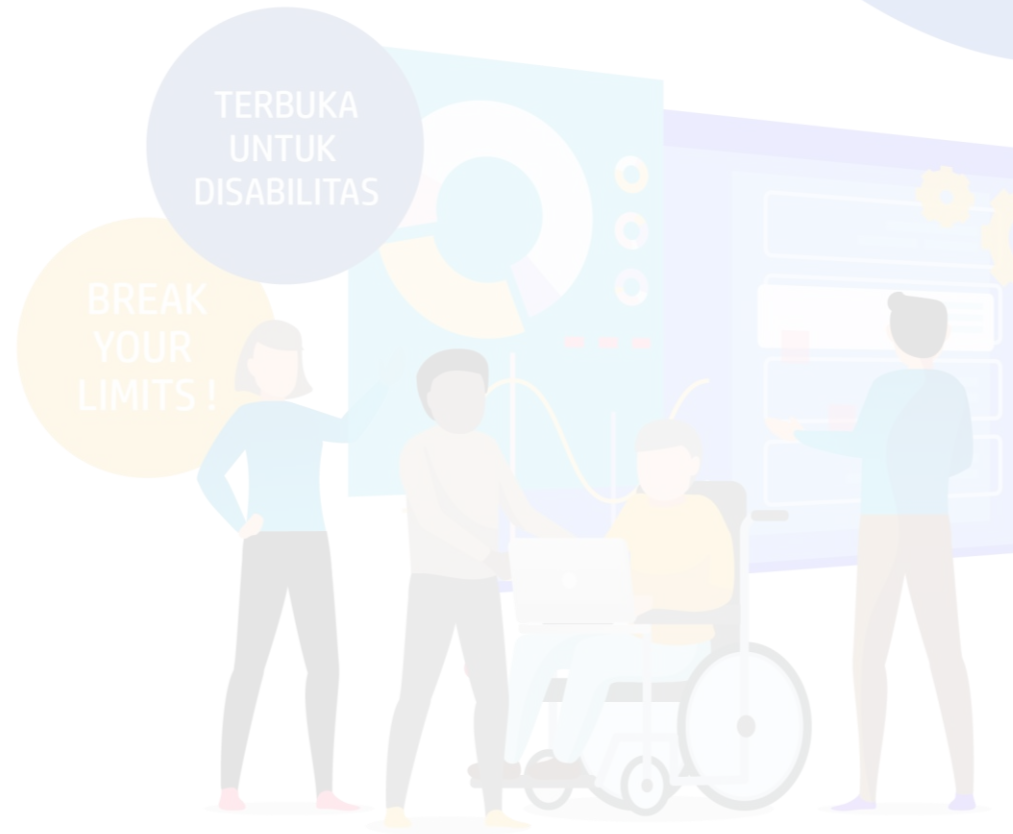
TERBUKA  
UNTUK  
DISABILITAS  
PELUANG  
BUAT  
MELAKUKAKAN  
IDEAL  
YOUR  
LIMITS!





# HDFS Command Line

- Memanggil HDFS shell
- Daftar command HDFS
- Ambari console



# HDFS file command interface

- Panggil FileSystem (FS) shell  
`hdfs dfs <args>`
- Contoh command daftar isi direktori saat ini di HDFS  
`hdfs dfs -ls`
- FS shell command path URI (Uniform Resource Identifier)  
↑ argumen  
`scheme://authority/path`
- Scheme: `hdfs` `hdfs` , local filesystem `file`  
`hdfs dfs -cp` (contoh command copy dari file ke HDFS)  
`file:///sampleData/spark/myfile.txt`  
`hdfs://rvm.svl.ibm.com:8020/user/spark/test/myfile.txt`
- Scheme dan authority optional, default dari *core-site.xml* conf file
- Mayoritas command FS shell mirip command UNIX

TERBUKA  
UNTUK  
DISABILITAS

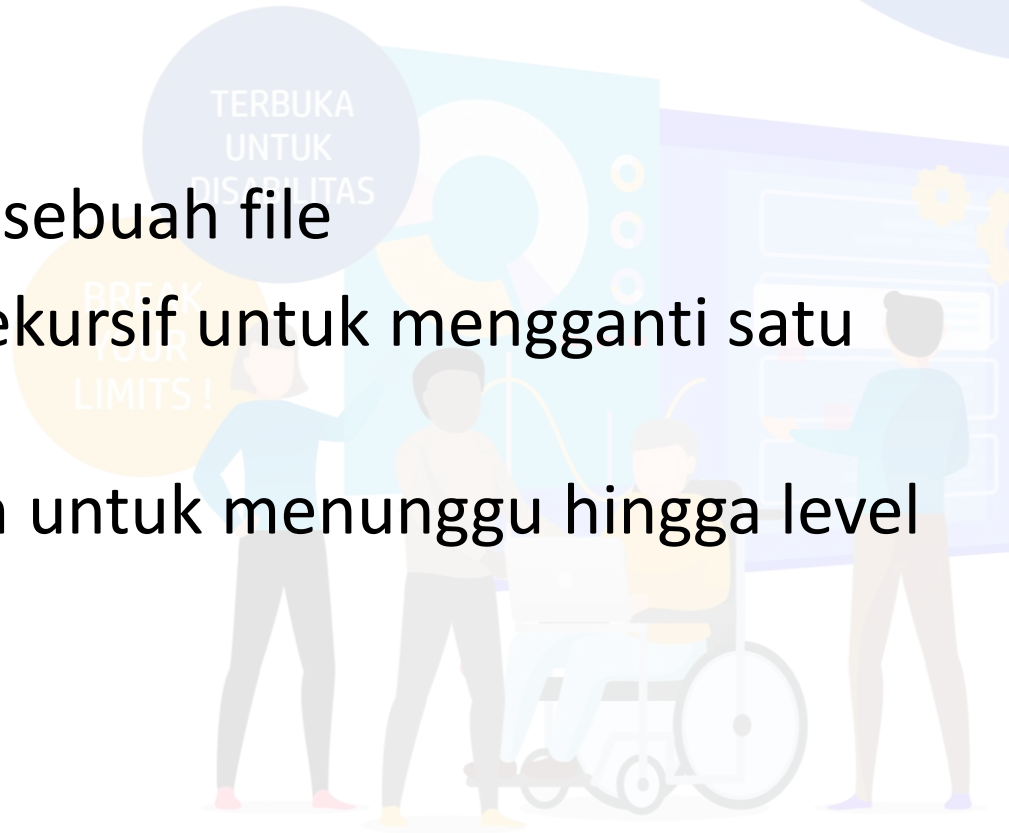
BREAK  
YOUR  
LIMIT

# HDFS file command interface

- Meski bukan POSIX compliant, ada beberapa command seperti POSIX  
cat, chgrp, chmod, chown, cp, dua, ls, mkdir, mv, rm, stat, tail
- Command spesifik HDFS  
copyFromLocal, copyToLocal, get, getmerge, put, setrep
  - copyFromLocal / put  
copy file dari local FS ke HDFS
  - copyToLocal / get  
copy file dari HDFS ke local
  - getMerge  
ambil semua file di direktori yang match dengan pola sumber  
gabung dan mengurutkan ke 1 file di local FS

### ➤ setRep

- ✓ set faktor replikasi sebuah file
- ✓ dapat dieksekusi rekursif untuk mengganti satu tree
- ✓ dapat dispesifikkan untuk menunggu hingga level replikasi terpenuhi





DIGITAL  
TALENT  
SCHOLARSHIP

# Ambari console



Ambari

HDPTTEST

0 ops

7 alerts

Dashboard

Services

Hosts 2

Alerts

Admin



admin



HDFS



MapReduce2



YARN



Tez



Hive



HBase



Pig



Sqoop



Oozie



ZooKeeper



Falcon



Storm



Flume



Ambari Metrics



Atlas



Flink



Kafka



Knox



NiFi



Ranger



Spark



Zeppelin Notebook

Actions

Metrics

Heatmaps

Config History

Metric Actions

Last 1 hour

HDFS Disk Usage



DataNodes Live

2/2

HDFS Links

NameNode  
Secondary NameNode  
2 DataNodes

More...

Memory Usage

No Data Available

Network Usage

No Data Available

CPU Usage

No Data Available

Cluster Load

No Data Available

NameNode Heap



NameNode RPC

0.22 ms

NameNode CPU WIO

n/a

NameNode Uptime

3.0 d

ResourceManager  
Heap



ResourceManager  
Uptime

3.0 d

NodeManagers Live

2/2

YARN Memory

n/a

YARN Links

ResourceManager  
2 NodeManagers

HBase Links

HBase Master  
2 RegionServers  
Master Web UI

HBase Master Heap



HBase Ave Load

0

Region In Transition

0

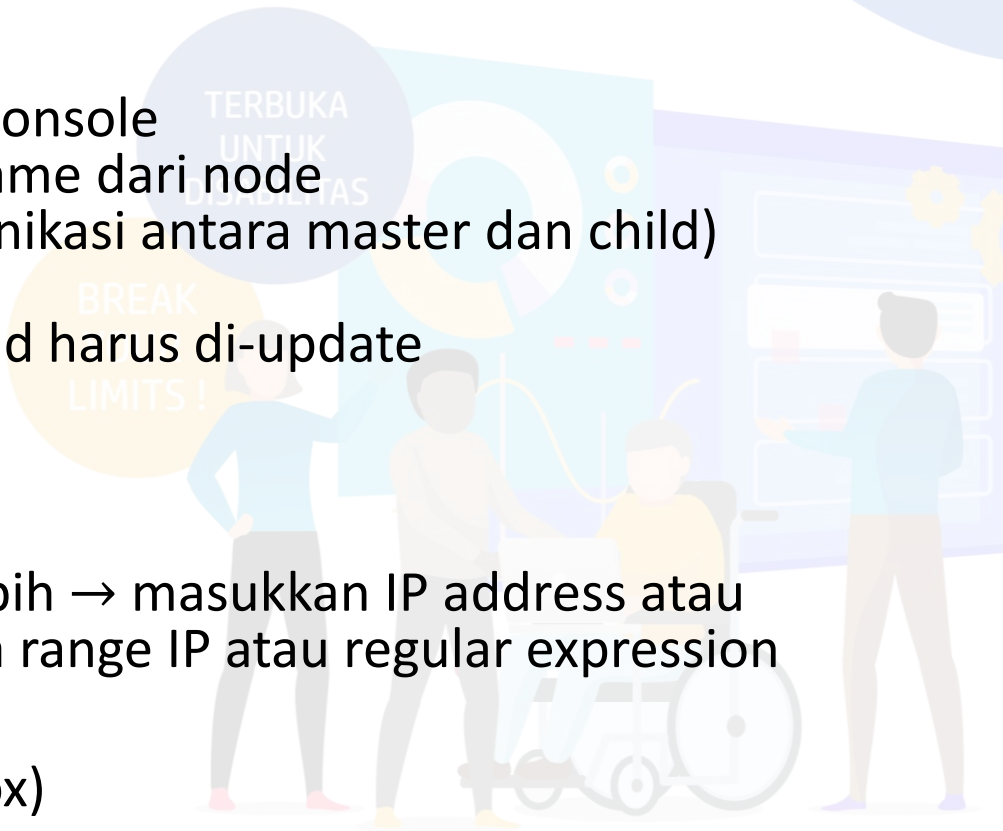
# Hadoop Administration

- Add / remove node dari cluster
- Monitoring cluster health
- Start / stop komponen/service
- Konfigurasi Hadoop
- Setting topologi rack



# Add/Remove Node dari Cluster

- Dapat dilakukan dari Ambari Console
  - butuh IP address atau hostname dari node
  - node harus reachable (komunikasi antara master dan child)
- \* /etc/hosts di master dan child harus di-update
- ✓ Ambari console → tab Hosts
- ✓ Actions → Add New Hosts
- ✓ Dialog tambah 1 node atau lebih → masukkan IP address atau hostname atau keduanya (bisa range IP atau regular expression hostname)
- ✓ Add Multiple Service (checkbox)
  - \*service dapat di-remove
- Remove node, matikan service terlebih dahulu





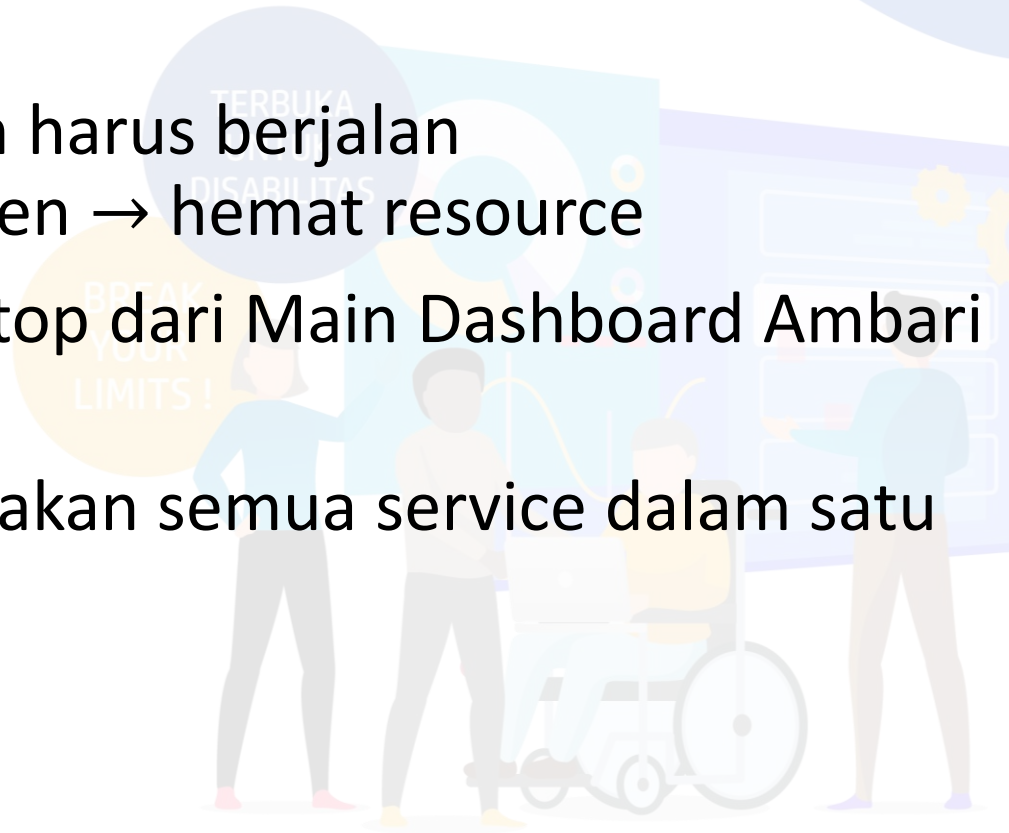
# Monitoring Cluster Health

- Dapat dilakukan melalui Ambari Console Monitor node dan service yang ada
- Disk Space
  - DFS Disk Check dengan DFS Report (`hdfs dfsadmin -report`)
    - mengetahui low storage
    - dapat dilihat di Ambari Console



# Start/Stop Komponen

- Tidak semua komponen harus berjalan  
Stop beberapa komponen → hemat resource
- Service dapat di-start/stop dari Main Dashboard Ambari Console
- Sebaiknya tidak menyalakan semua service dalam satu waktu



# Configuration files

- `hadoop-env.sh` – environment variable untuk run Hadoop
- `core-site.xml` – configuration setting Hadoop Core, I/O setting HDFS dan MapReduce
- `hdfs-site.xml` – configuration setting HDFS daemon: NameNode, Secondary NameNode, DataNode
- `mapred-site.xml` – configuration setting MapReduce daemon: JobTracker, TaskTracker
- `Masters` – Daftar mesin yang run Secondary NameNode
- `slaves` – Daftar mesin yang run DataNode dan TaskTracker
- `hadoop-metrics.properties` – kontrol metric pada Hadoop
- `log4j.properties` – system logfiles, NameNode audit log, task log untuk TaskTracker child proses

# hadoop-env.sh settings

- Sebagian besar variabel → default tidak diset
- Hanya export JAVA\_HOME harus diset ke Java SDK
- HADOOP\_HOME – berisi node dan config file  
/usr/iop/current/hadoop-client
- HADOOP\_LOG\_DIR – menjaga log /var/log/Hadoop/\$USER
- HADOOP\_HEAPSIZE – digunakan JVM untuk tiap daemon  
NameNode - HADOOP\_NAMENODE\_OPTS  
DataNode - HADOOP\_DATANODE\_OPTS  
Secondary NameNode - HADOOP\_SECONDARYNAMENODE\_OPTS  
JobTracker - HADOOP\_JOBTRACKER\_OPTS  
TaskTracker - HADOOP\_TASKTRACKER\_OPTS
- Environment variable lain - HADOOP\_CLASSPATH, HADOOP\_PID\_DIR

# core-site.xml setting

- `fs.defaultfs` – nama default filesystem. URI dengan scheme dan authority menentukan implementasi FileSystem. Scheme menentukan config property (`fs.SCHEME.impl`) penamaan class implementasi FileSystem. Authority menentukan host, port, dll FileSystem. Default file:///
- `hadoop.tmp.dir` – base untuk direktori sementara lainnya `/tmp/hadoop-${user.name}`
- `fs.trash.interval` – jumlah menit antara trash checkpoint. Jika 0, trash feature disabled (default). Jika > 0, file terhapus akan dimasukkan ke .trash pada direktori home milik user
- `io.file.buffer.size` – ukuran buffer untuk sequence file. Kelipatan hardware page size (4096 pada x86), menentukan buffer saat read dan write

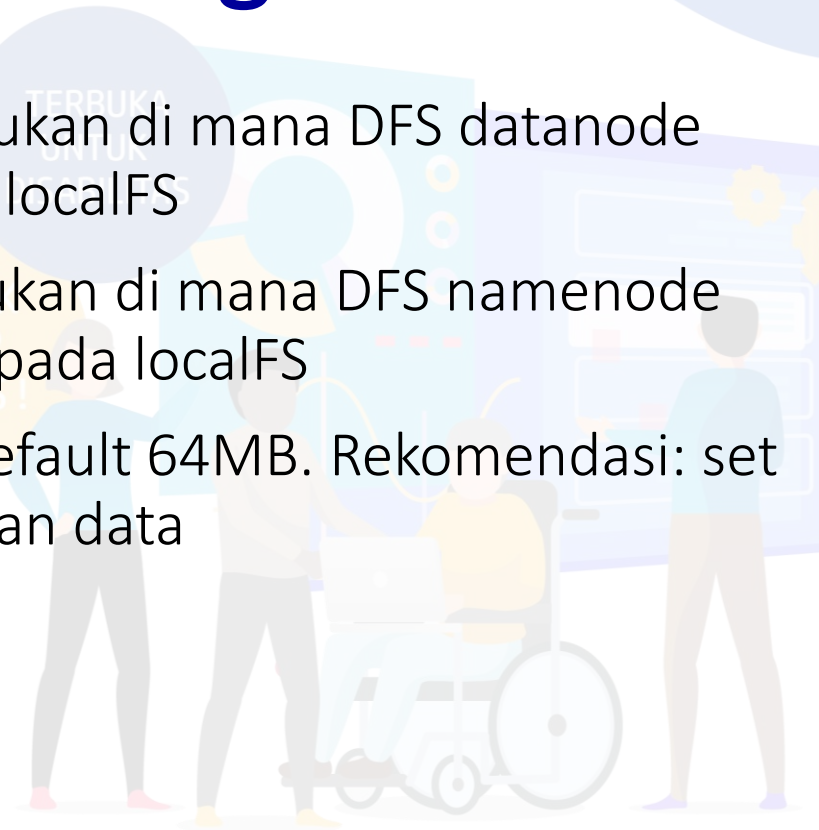
# core-site.xml setting

- `hadoop.rpc.socket.factory.class.default` – default SocketFactory untuk digunakan. Parameter untuk diformat sebagai `package.FactoryClassName`
- `hadoop.rpc.socket.factory.class.clientprotocol` – SocketFactory untuk koneksi ke DFS. Jika null atau kosong, gunakan `hadoop.rpc.socket.class.default`. Digunakan juga oleh DFSClient untuk membuat socket ke DataNode

\*biarkan kedua parameter tersebut kosong, tandai FINAL

# hdfs-site.xml setting

- `dfs.datanode.data.dir` – menentukan di mana DFS datanode harus menyimpan blocks-nya pada localFS
- `dfs.namenode.name.dir` - menentukan di mana DFS namenode harus menyimpan name table-nya pada localFS
- `dfs.blocksize` – HDFS blocksize, default 64MB. Rekomendasi: set ke 128MB atau sesuai dengan ukuran data





# mapred-site.xml configuration (1)

- `mapreduce.jobtracker.hosts` – menamai file yang berisi daftar nodes yang terhubung dengan jobtracker. Jika value-nya kosong, semua host diizinkan
- `mapreduce.jobtracker.hosts.exclude` – menamai file yang berisi daftar hosts yang harus di-exclude oleh jobtracker. Jika value-nya kosong, tidak ada host yang di-exclude
- `mapreduce.job.maxtaskfailures.per.tracker` – jumlah task-failure pada tasktracker job yang diberikan setelah task baru mana yang tidak di-assign ke job. Default 3.
- `mapreduce.jobtracker.tasktracker.maxblacklists` – jumlah blacklist untuk TaskTracker oleh berbagai job yang di-blacklisted di semua job. Tracker akan diberikan task kemudian (setelah 1 hari). Tracker akan menjadi healthy tracker setelah restart. Default 4.

# mapred-site.xml configuration (2)

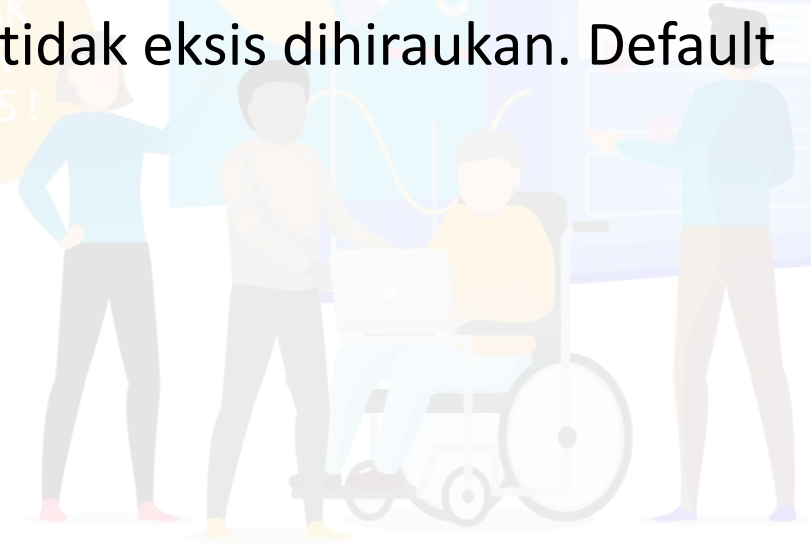
- `mapreduce.job.reduces` – jumlah default reduce task per job. Umumnya diset ke 99% ke kapasitas reduce cluster, jadi jika sebuah node gagal/down, makan reduce masih bisa dieksekusi dalam 1 gelombang. Dihiraukan ketika `mapred.job.tracker` “local”. Default 1. Rekomendasi diset 90%.
- `mapreduce.map.speculative` – Jika TRUE, multiple instance dari beberapa map task akan dieksekusi paralel. Default TRUE.
- `mapreduce.reduce.speculative` – Jika TRUE, multiple instance dari beberapa reduce task akan dieksekusi paralel. Default TRUE. Rekomendasi FALSE.
- `mapreduce.tasktracker.map.tasks.maximum` - Jumlah max map tasks yang akan dijalankan secara simultan oleh TaskTracker. Default 2. Rekomendasi set relevan ke jumlah CPU dan memori di tiap DataNode

# mapred-site.xml configuration (3)

- `mapreduce.tasktracker.reduce.tasks.maximum` - Jumlah max reduce tasks yang akan dijalankan secara simultan oleh TaskTracker. Default 2. Rekomendasi set relevan ke jumlah CPU dan memori di tiap DataNode
- `mapreduce.jobtracker.taskscheduler` – Class bertanggung jawab untuk scheduling task. Default ke FIFO scheduler. Rekomendasi menggunakan Job Queue Task – `org.apache.hadoop.mapred.JobQueueTaskScheduler`
- `mapreduce.jobtracker.restart.recover` – Recover job gagal ketika JobTracker restart. Untuk cluster produksi direkomendasikan diset TRUE.

# mapred-site.xml configuration (4)

- `mapreduce.cluster.local.dir` – Direktori lokal di mana MapReduce menyimpan file data intermediate. Bisa berbentuk daftar terpisah dengan koma, direktori di divais berbeda untuk menyebar I/O disk. Direktori yang tidak eksis dihiraukan. Default `${hadoop.tmp.dir}/mapred/local`



# Konfigurasi Hadoop - Contoh

- Stop Service sebelum melakukan perubahan
- Ganti ke direktori conf, hdfs-site.xml:  
`cd /usr/iop/current/hadoop-client/conf`  
`vi hdfs-site.xml`

```
<!--Fri Jan  8 12:06:20 2016-->
<configuration>

  <property>
    <name>dfs.block.access.token.enable</name>
    <value>true</value>
  </property>

  <property>
    <name>dfs.blockreport.initialDelay</name>
    <value>120</value>
  </property>

  <property>
    <name>dfs.blocksize</name>
    <value>134217728</value>
  </property>

  <property>
    <name>dfs.client.file-block-storage-locations.timeout.millis</name>
    <value>3000</value>
  </property>
```

# Setting Topologi Rack (Rack Awareness)

- Dapat didefinisikan dengan script, spesifik node di rak mana
- Script di `topology.script.file.name` property di `core-site.xml`  
Contoh:  

```
<property>  
    <name>topology.script.file.name</name>  
    <value>/opt/ibm/biginsights/hadoop-conf/rack-aware.sh</value>  
</property>
```
- Network topology script menerima argumen 1 atau lebih IP address node-nya di cluster  
Return stdout daftar nama rack, 1 untuk setiap input  
Urutan input dan output harus konsisten

\*contoh:

[http://wiki.apache.org/hadoop/topology\\_rack\\_awareness\\_scripts](http://wiki.apache.org/hadoop/topology_rack_awareness_scripts)



# Aktivitas Kelas

- Instalasi Hadoop
- Setup Single Node
- Setup Cluster
- Menjalankan perintah dasar Hadoop
- Meletakkan dan mengambil data dari HDFS
- Memasukkan data dari database ke HDFS





# Referensi

- <https://courses.cognitiveclass.ai/courses/course-v1:BigDataUniversity+BD0111EN+2016/>
- <https://www.ibm.com/analytics/hadoop/hdfs>
- <https://hadoop.apache.org/docs/r2.7.2/hadoop-project-dist/hadoop-hdfs/HdfsUserGuide.html>



# DIGITAL TALENT SCHOLARSHIP 2019

Big Data Analytics



## Terimakasih

Oleh:

Fakultas Ilmu Komputer (Filkom) Universitas Brawijaya (UB)