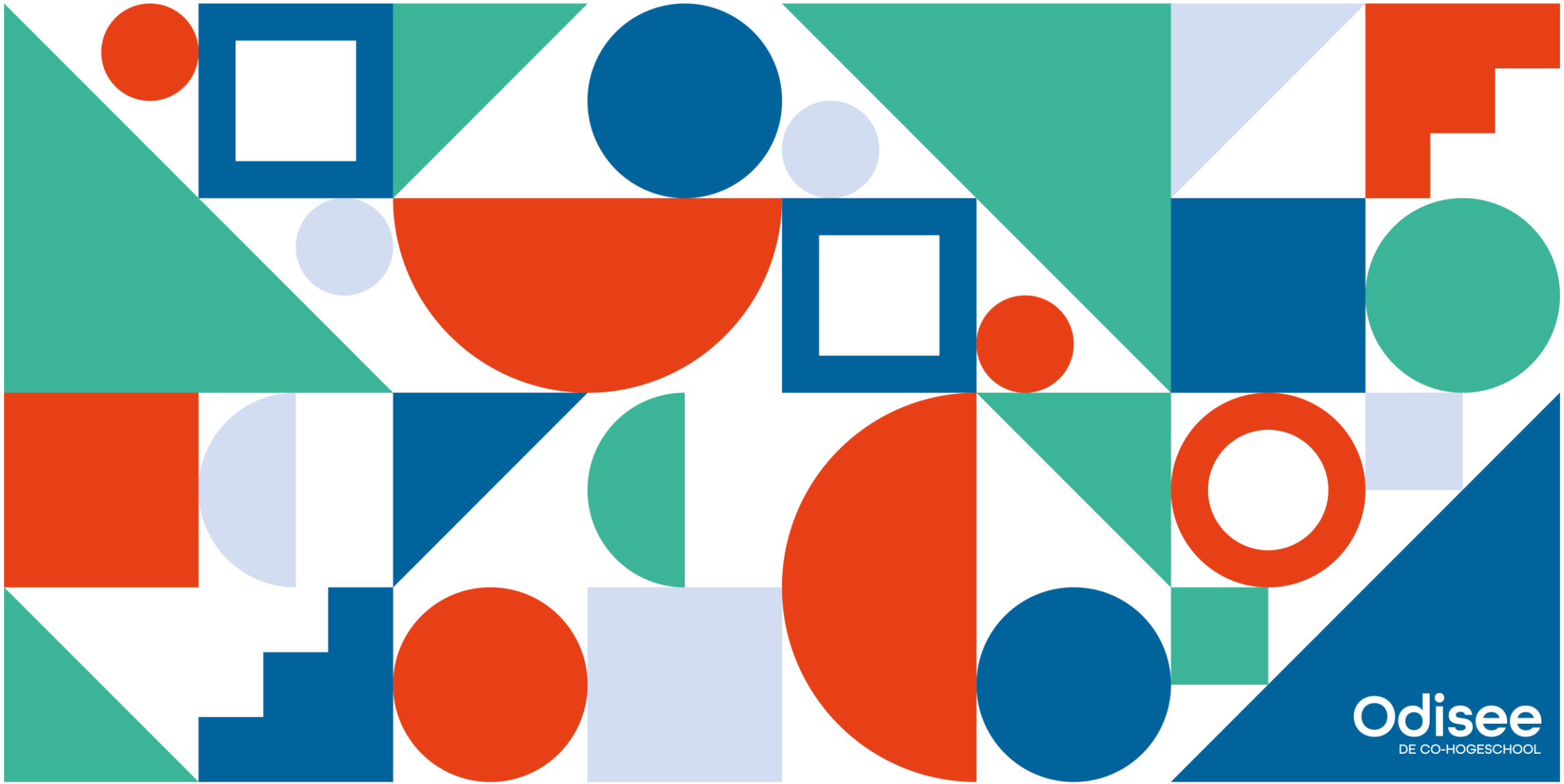

Odisee
DE CO-HOGESCHOOL

Big Data – Distributed storage

Jens Baetens

Structuur

- Hadoop ecosysteem
- Containers
  - Docker
- HDFS

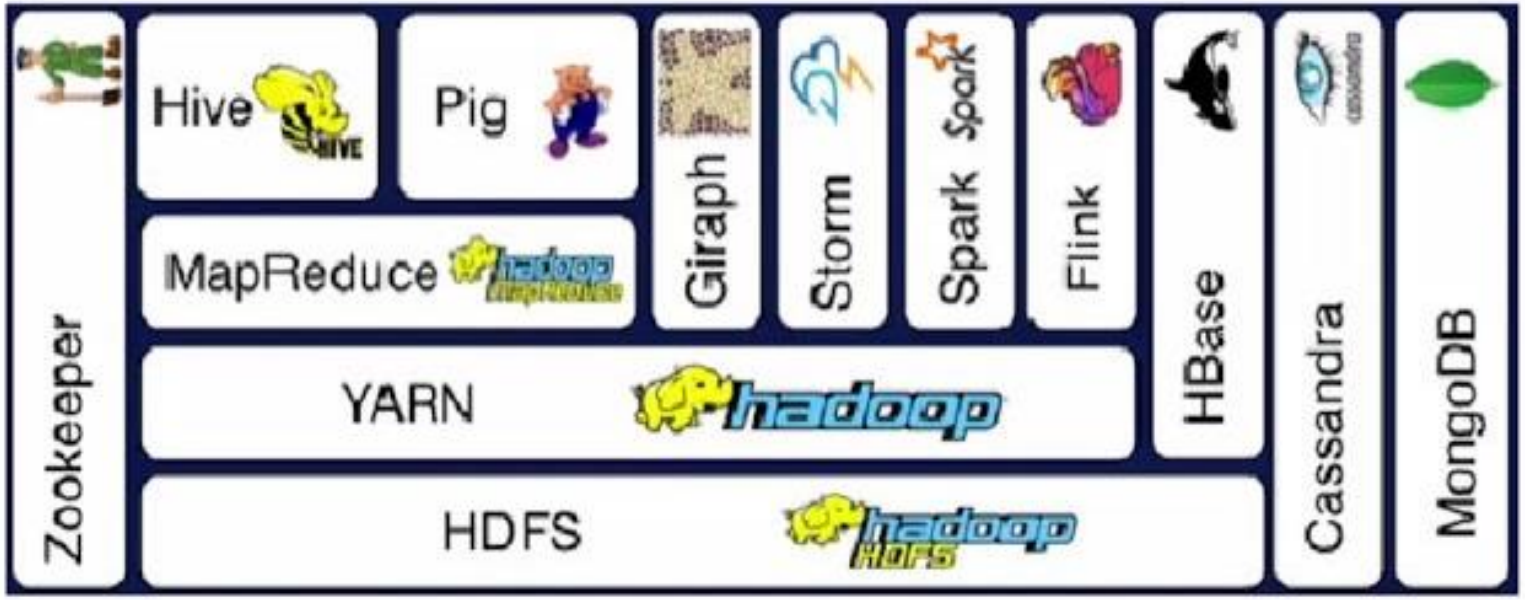
Hadoop Ecosystem

Hadoop

- Gebaseerd op Google File System (2003)
- Ontwikkeld door Apache
- Open source
- Uitgegroeid tot omgeving met veel verschillende applicaties

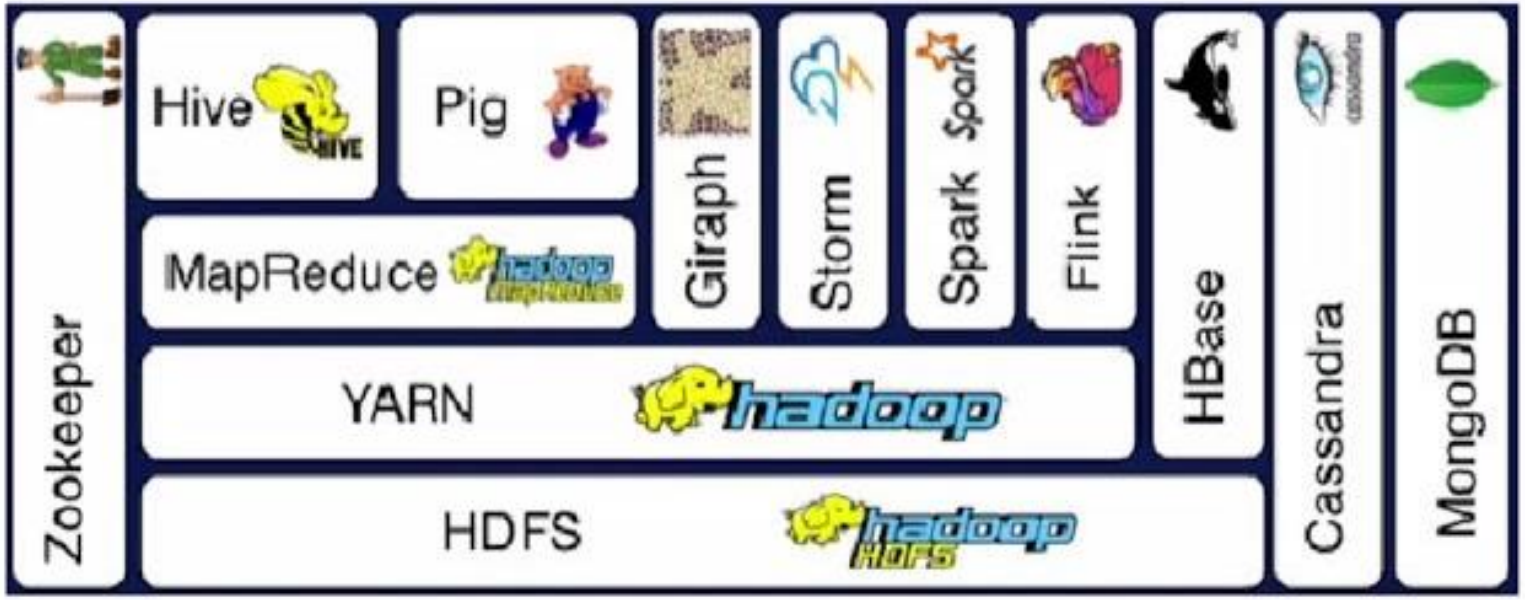
Hadoop Ecosystem

- HDFS – core functionality
- Distributed File System
- Op HDD



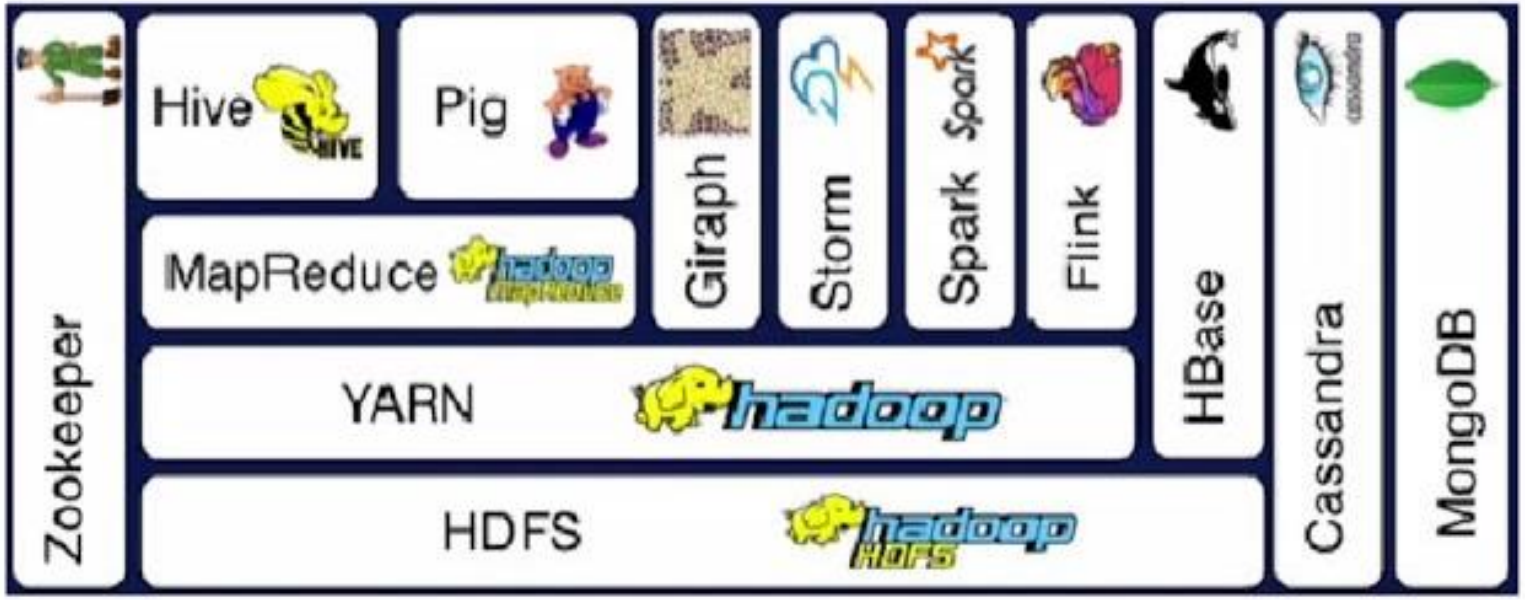
Hadoop Ecosystem

- YARN – Yet Another Resource Negotiator
- Beheer van computing power
- Welke code op welke node



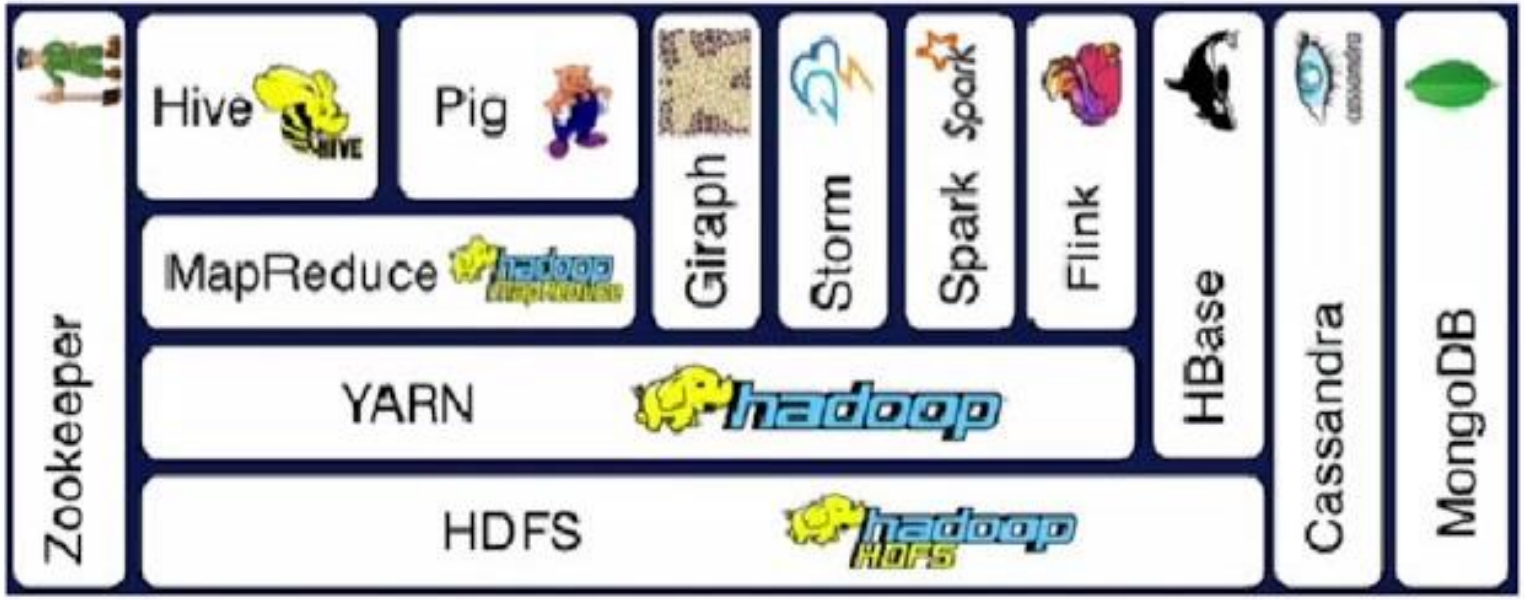
Hadoop Ecosystem

- MapReduce
- Distributed Computing
- Ontwikkeld door Google
- 2 fases
  - Mapping (Divergeren)
  - Reduce (Convergeren)



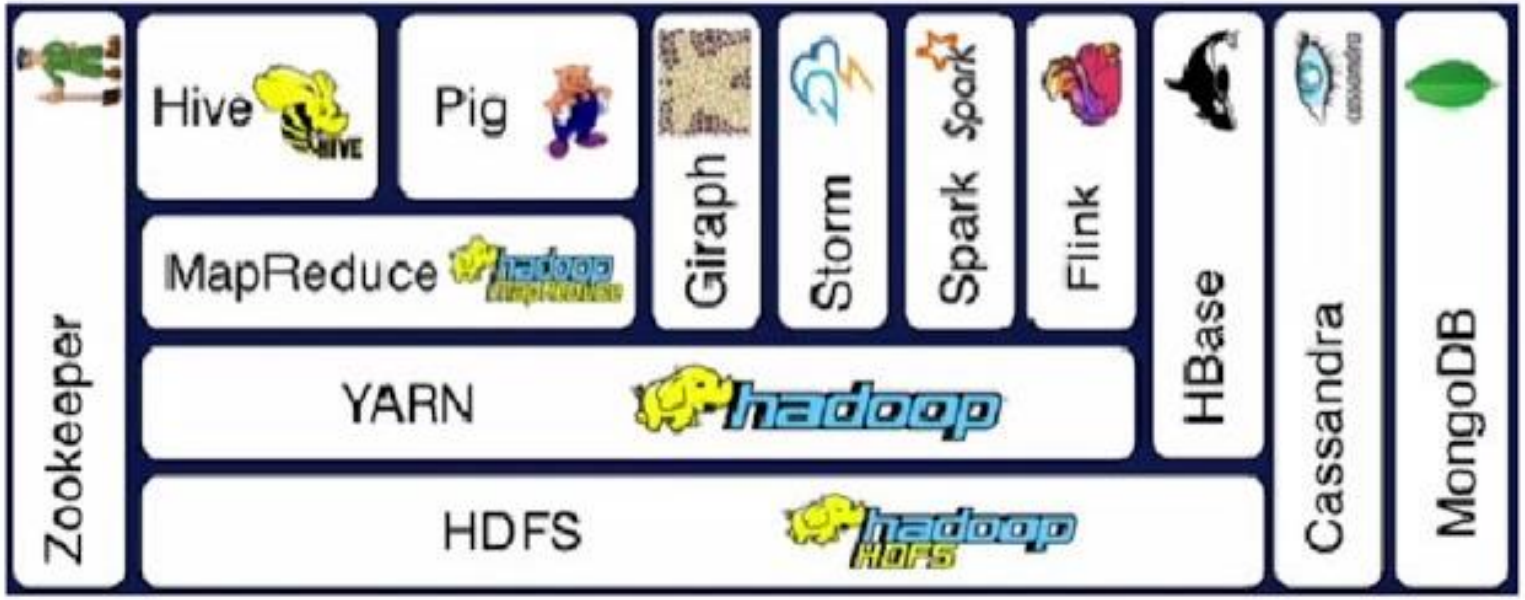
Hadoop Ecosystem

- Zookeeper
- Beheren van alle applicaties die lopen op de verschillende nodes



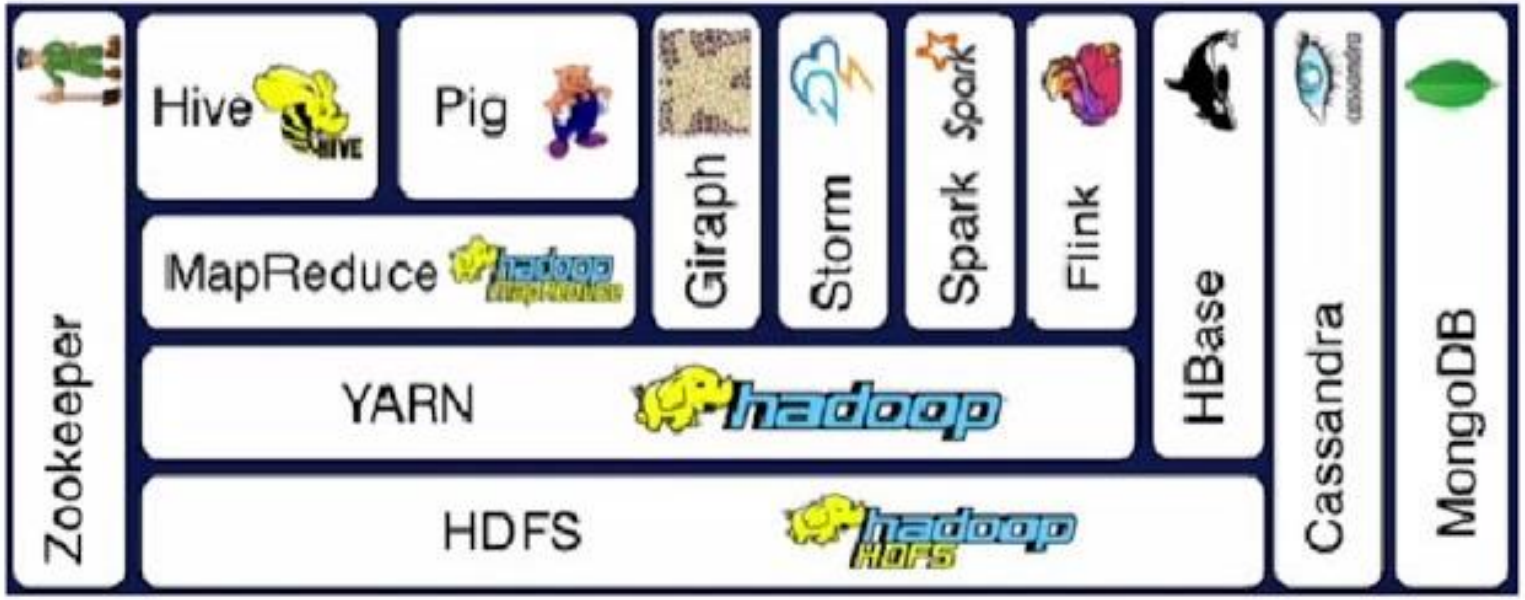
Hadoop Ecosystem

- Hive
- Distributed Datawarehouse
- Sql-like
- Queries via MapReduce



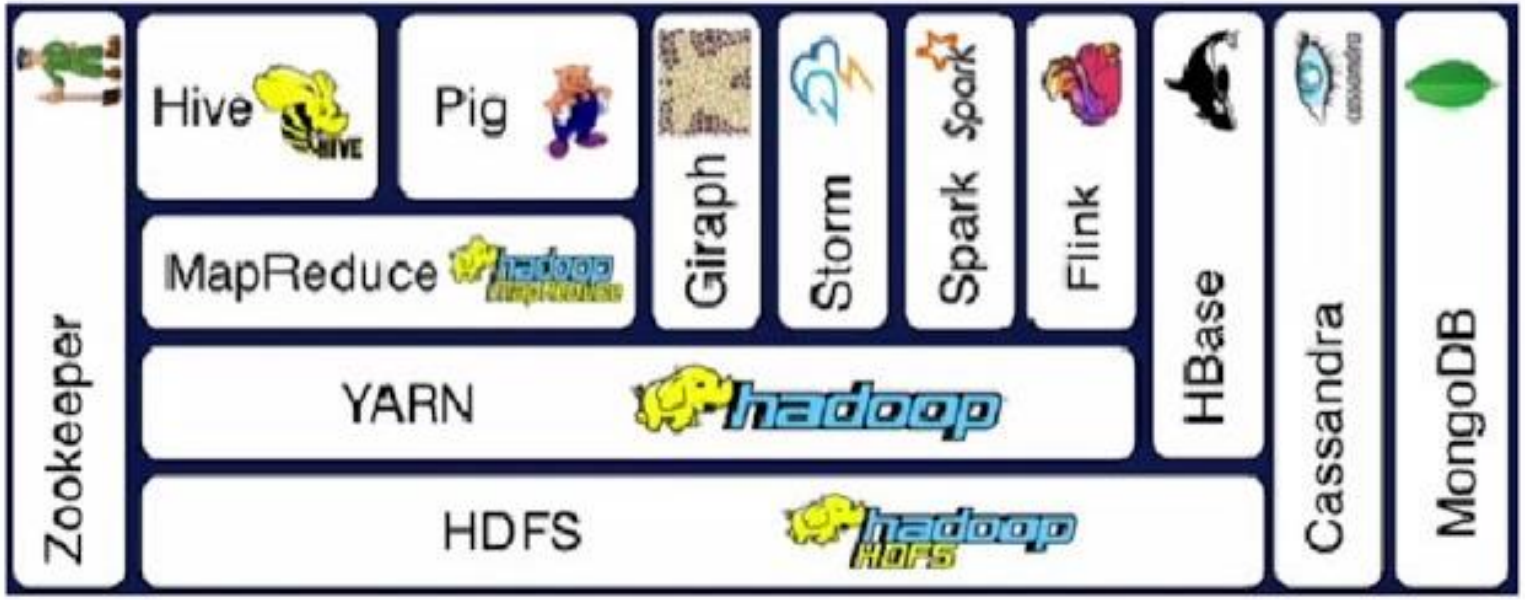
Hadoop Ecosystem

- Pig
- Data analysis
- Using MapReduce/Spark/…
- Taal: Pig Latin



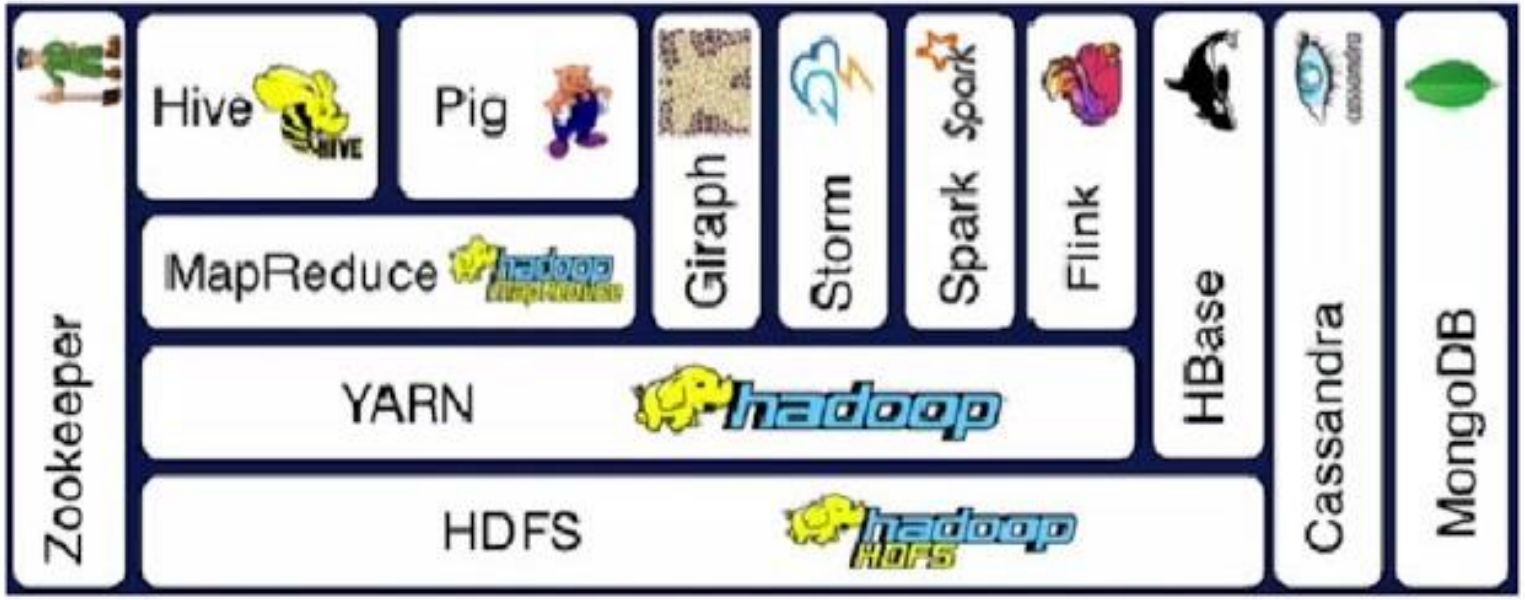
Hadoop Ecosystem

- Giraph
- Bestuderen van een graaf
- Social graph
  - Facebook
  - Twitter
  - …
- Gebruikt geen mapreduce



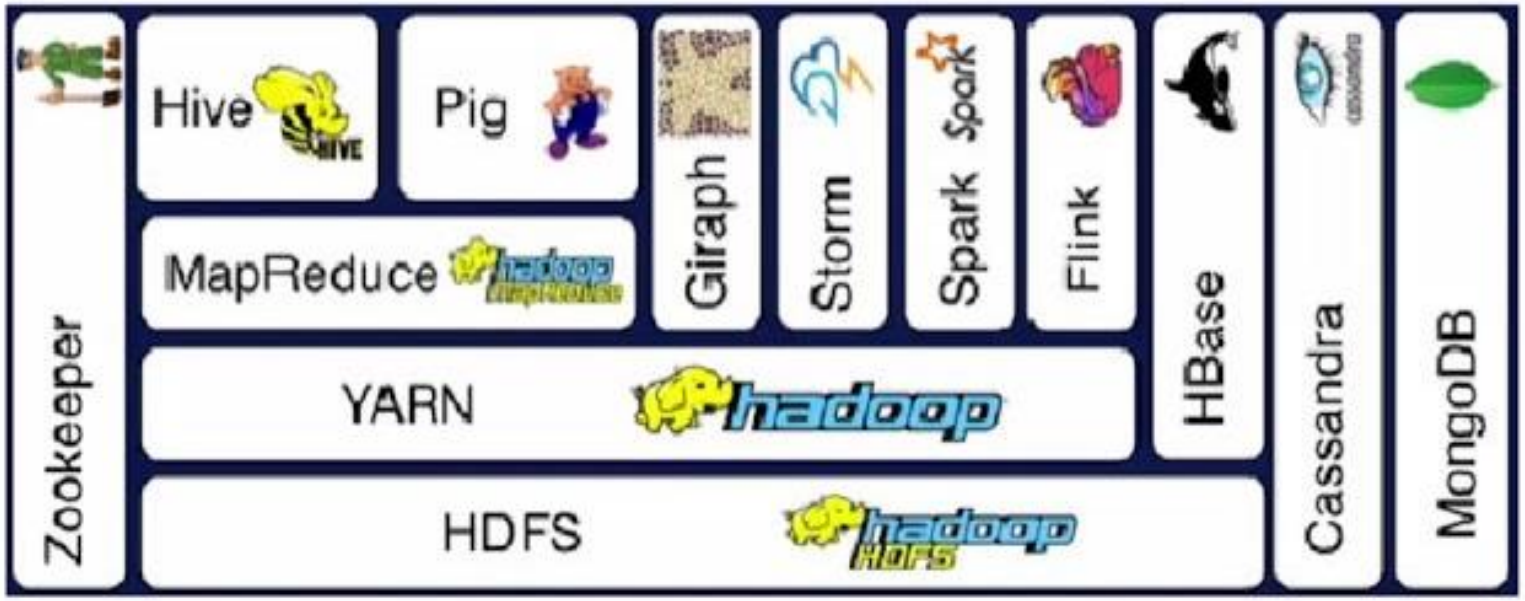
Hadoop Ecosystem

- Storm / Flink
- Verwerken van data streams – continue inkomende datastromen
  - Classificeren
  - Opslaan
  - …



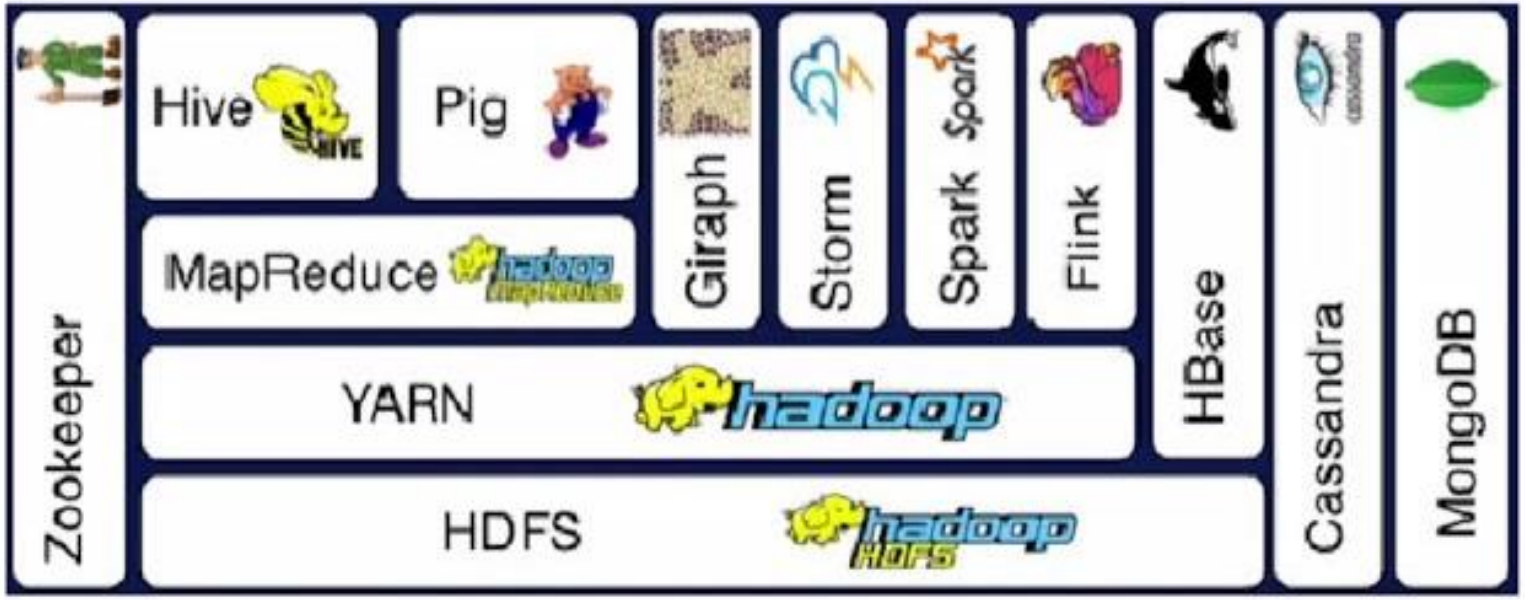
Hadoop Ecosystem

- Spark
- Alternatief voor MapReduce
- Computing in Ram
- Op Hadoop/Cloud/…
- Gebruikt voor
  - SQL (Spark SQL)
  - Streaming (Spark Streaming)
  - Machine Learning (MLlib)
  - Graph analysis (GraphX)



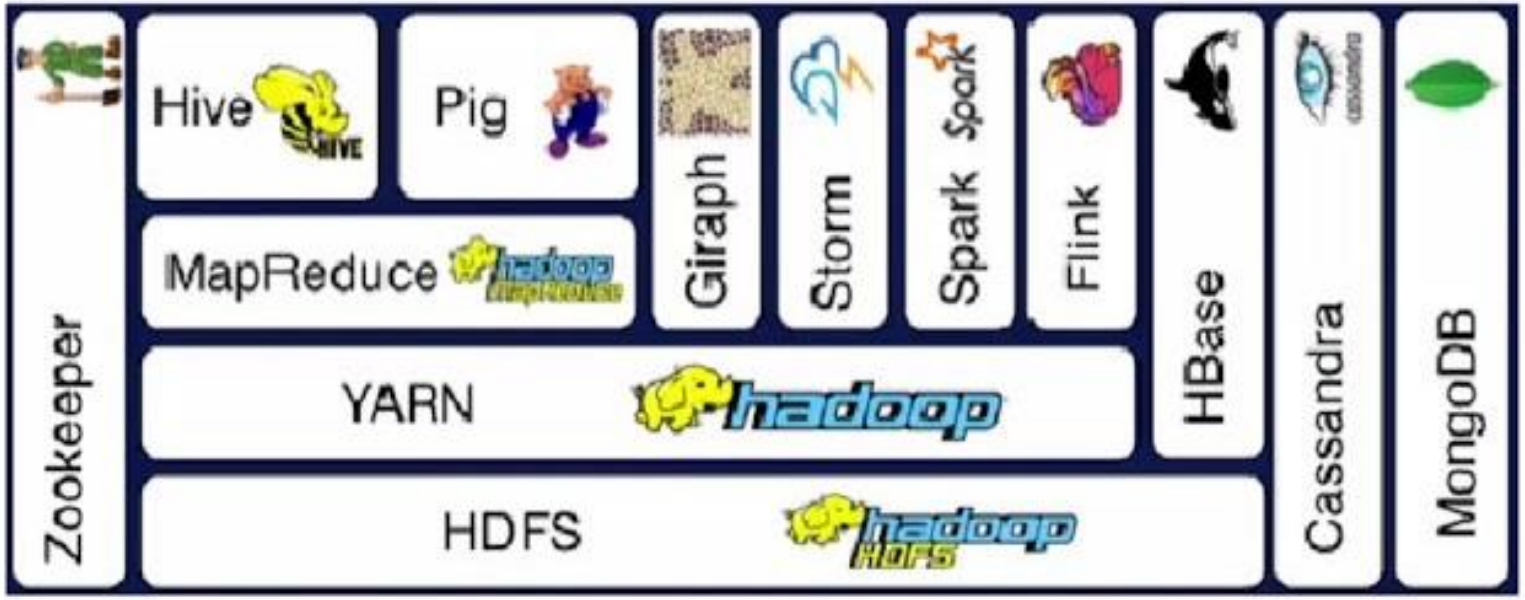
Hadoop Ecosystem

- HBase
- Distributed NoSQL Database
- Geen SQL maar in JAVA



Hadoop Ecosystem

- Cassandra / Mongo DB
- Maken geen gebruik van HDFS
- NoSql databases
- Stand-alone solutions

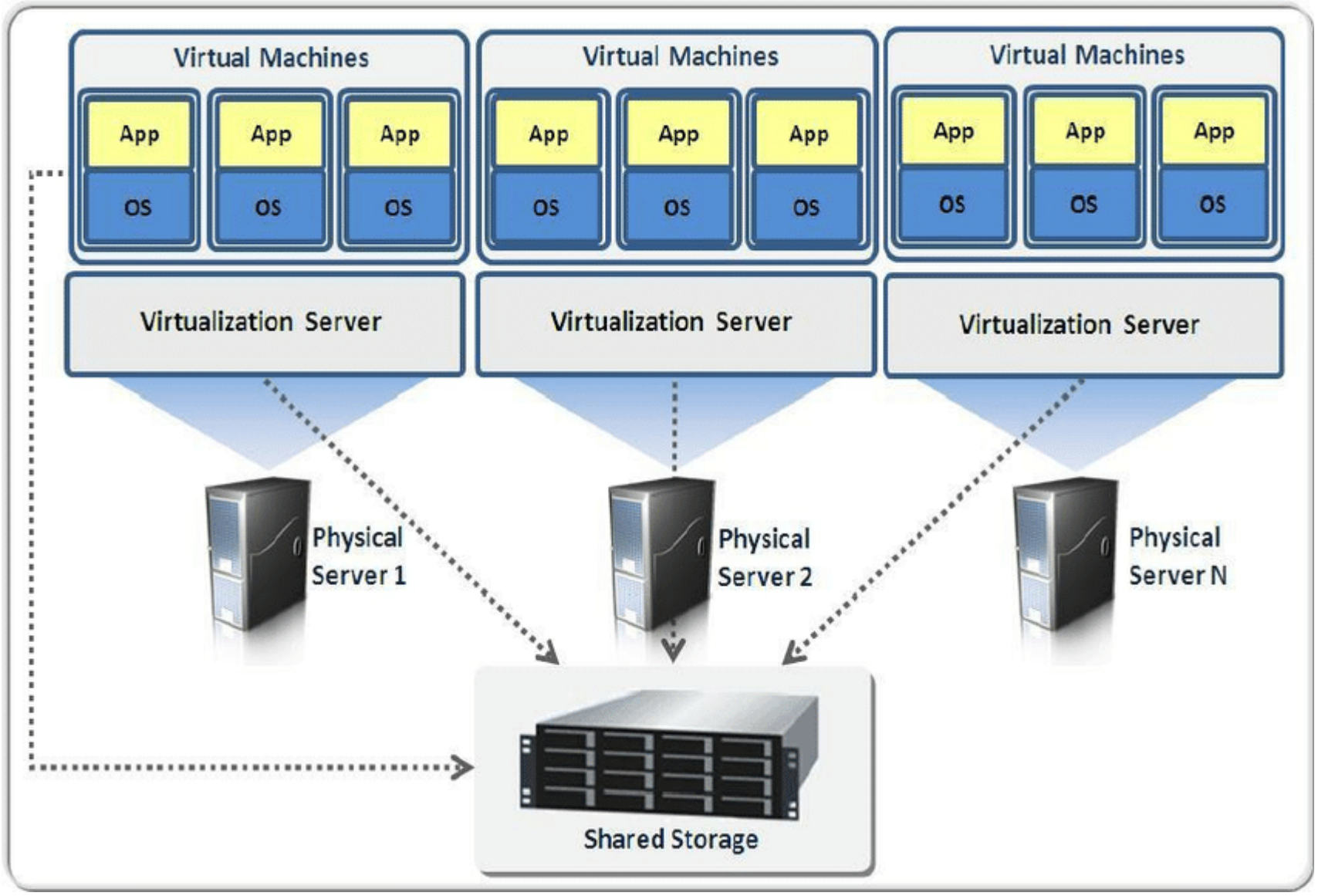


Containers

Virtualisatie

- Het virtueel maken van een fysiek toestel
  - Kan zowel op niveau van server, netwerk, opslagmedium, …
    - In deze cursus vooral het niveau van server relevant
  - Voordelen:
    - Efficiëntie
    - Flexibiliteit
    - Disaster recovery

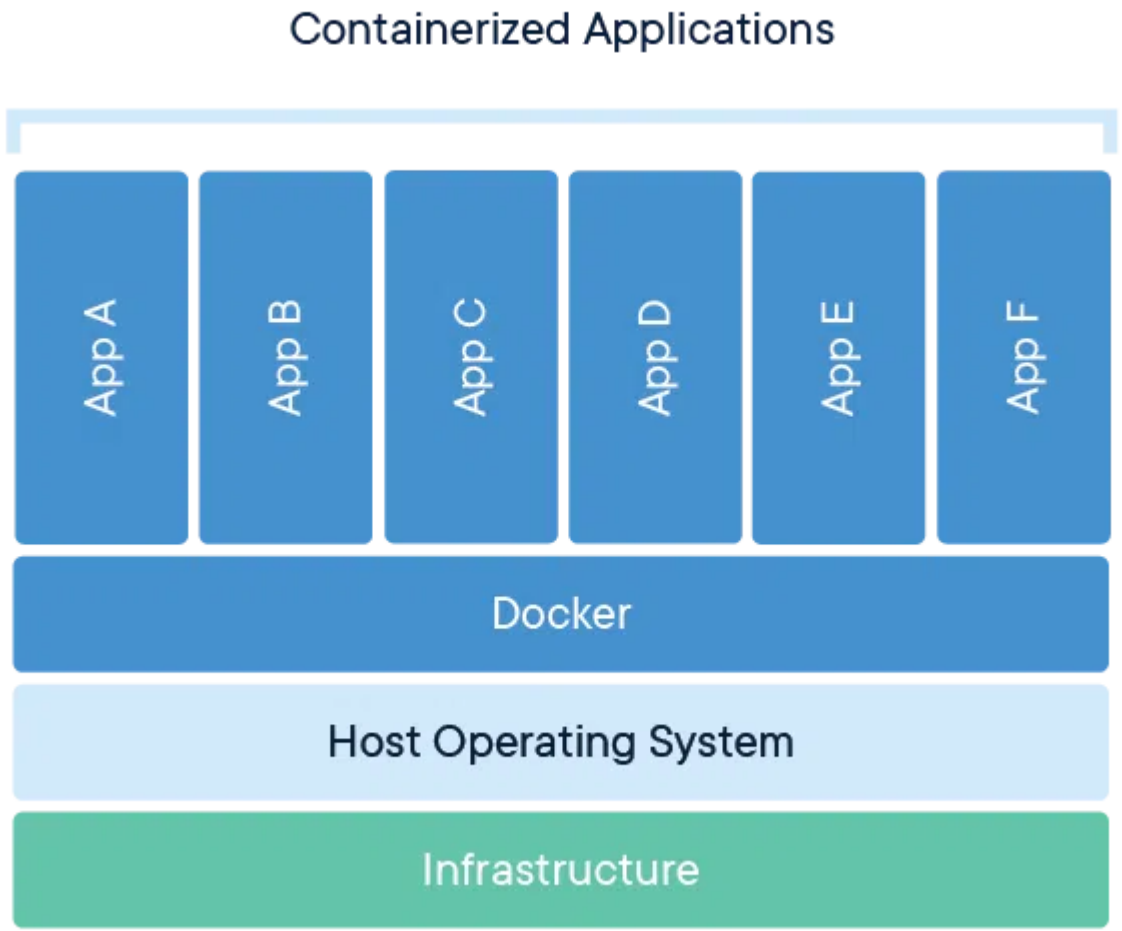
Server virtualisatie

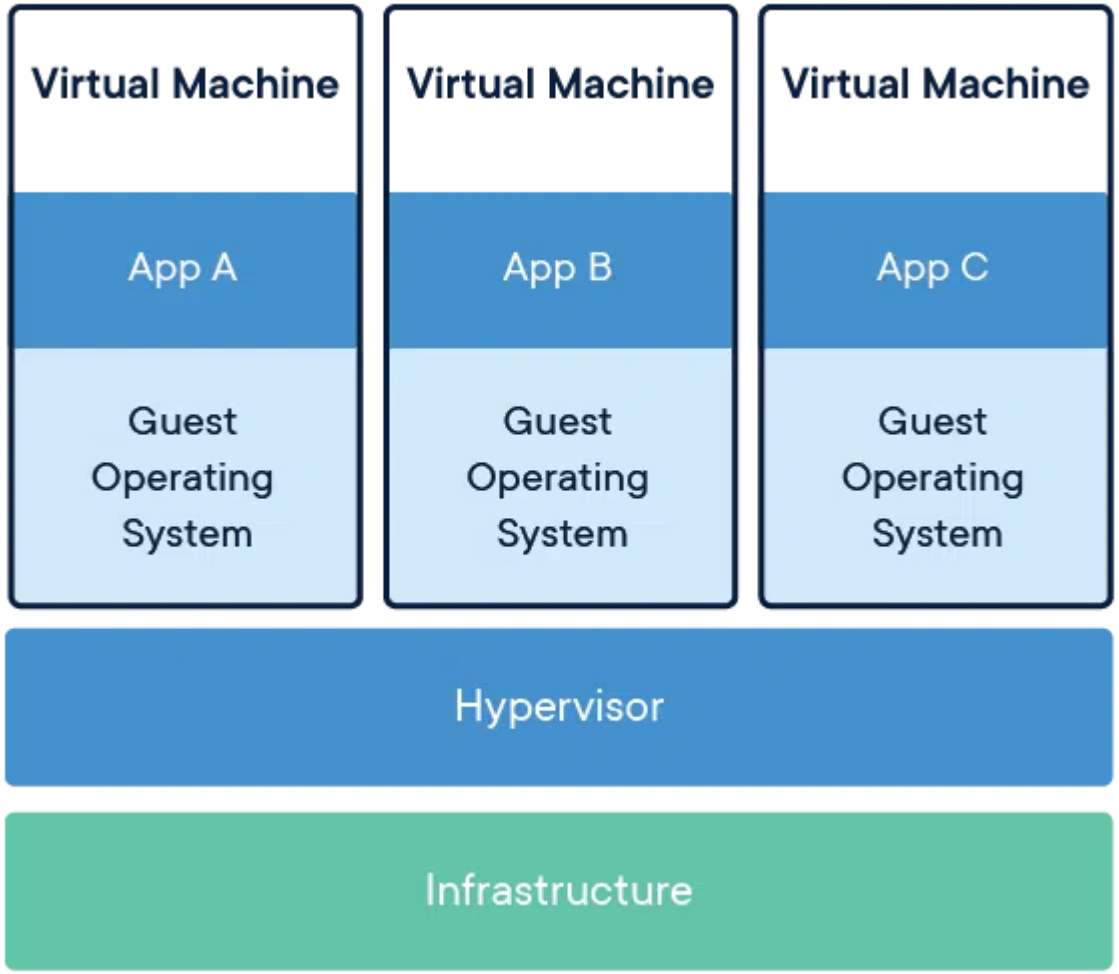


Containers

- Abstractie van de application layer
  - Groepeert applicatie code en afhankelijkheden
  - Alle containers delen de OS-kernel maar runnen onafhankelijk
- Containers worden beheerd door container runtimes ipv hypervisor
- Soorten software om containers te beheren:
  - Docker
  - Kubernetes
- Standaard voor containers ontwikkeld door Docker

Containers vs Virtuele machines



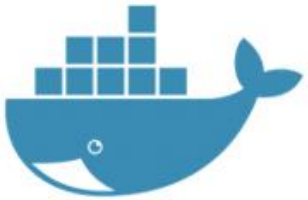


Voordelen van containers

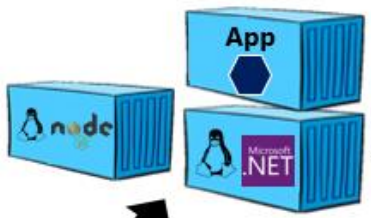
- Zelfde voordelen als VM's maar
  - Minder groot (MB's per image ipv 10 GB of meer)
  - Starten sneller (omdat ze minder groot zijn)
  - Schaalbaarder

Docker terminologie

Basic taxonomy in Docker



A **Registry**
Stores many static images



Images
Static, persisted container image

Container
Image-instance running an app process (service/web)

On-premises
('n' private organizations)

- **Hosted Docker Registry**
- **Docker Trusted Registry on-prem.**

Public Cloud
(specific vendors)

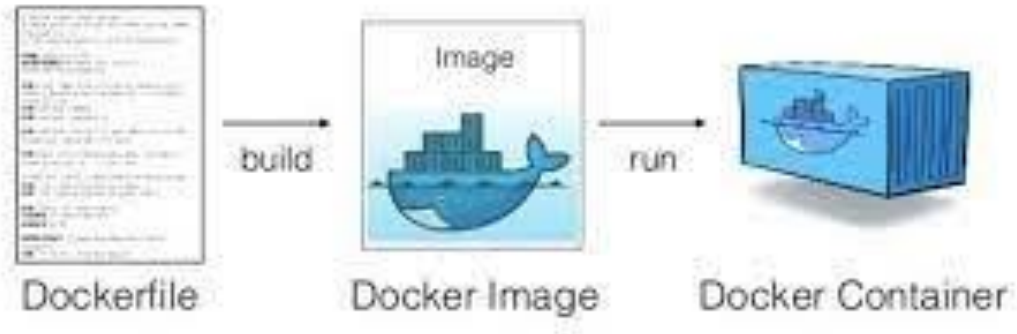
- **Docker Hub Registry**
- Docker Trusted Registry on-cloud
- **Azure Container Registry**
- AWS Container Registry
- Google Container Registry
- Quay Registry
- Other Cloud

Basic taxonomy

- Registry
  - Repository voor bewaren en verdelen van container images
  - Bvb: Docker hub
- Container image
  - Een lichtgewicht, standalone en uitvoerbare software package
  - Bevat applicatiecode, dependencies maar geen OS
- Container
  - Een container image dat uitgevoerd wordt
- Volume
  - Een manier om data te bewaren buiten de container of data te delen tussen host en/of containers

Docker file

- Een set van instructies om een container image te bouwen
  - Voorbeeld: dockerfiles voor onze containers
  - Belangrijke commando's:
    - FROM: van welke image te starten (uit een repository)
    - RUN: execute command in shell
    - ADD/COPY: Copy file from build directory to image
    - EXPOSE: open netwerkpoort
    - CMD/ENTRYPOINT: wat er gestart moet worden
    - ENV: set environment variabele
    - ARG: set build-time variabelen

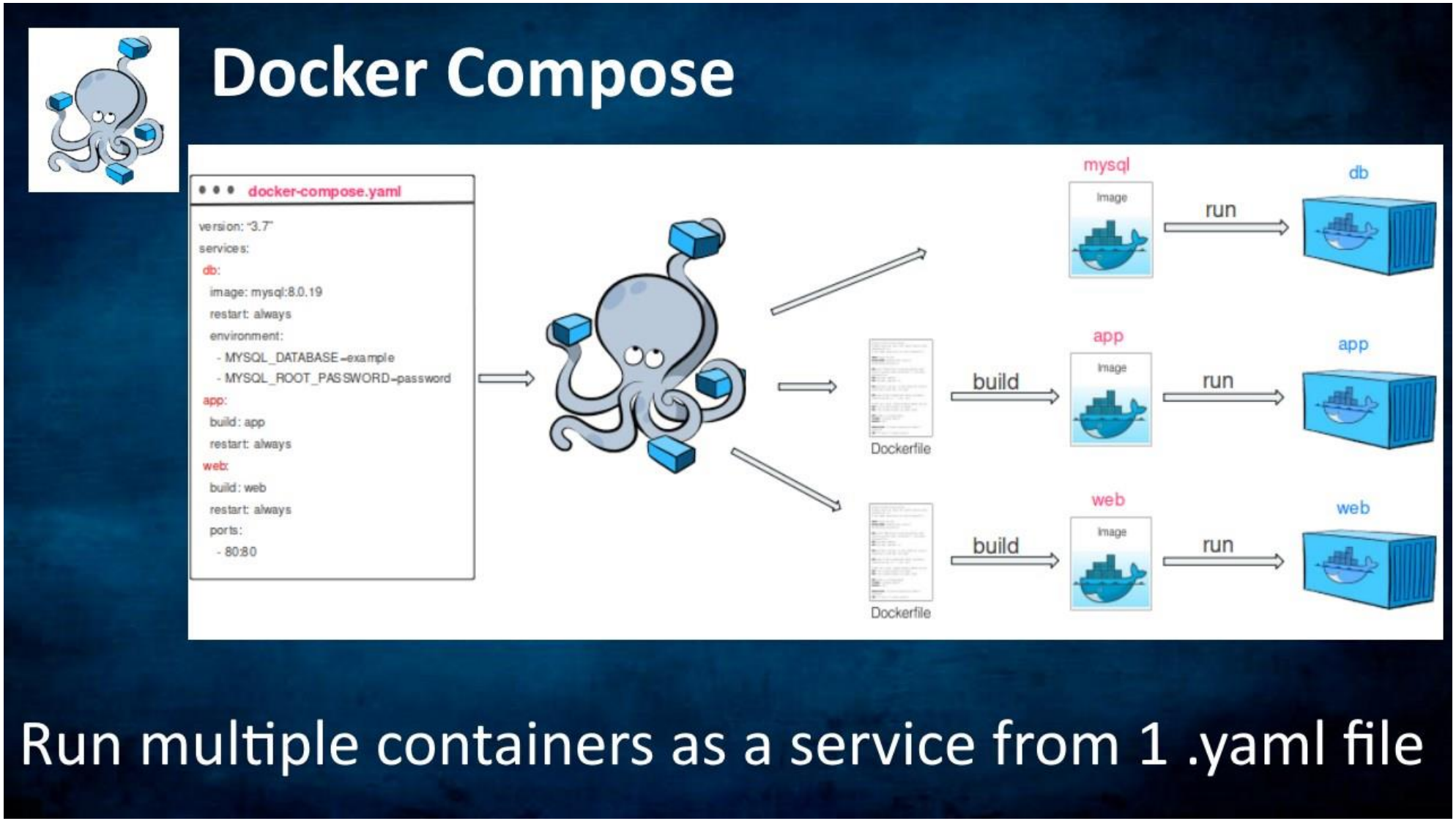


Docker compose

- Een software applicatie kan uit meerdere containers bestaan
  - Website bestaat uit een server en een database (2 containers bvb)
  - Onze cluster bestaat uit onder andere een namenode en 4 datanodes
- Is een tool om multi-container applications te definieren en te starten
  - Maakt gebruik van een yaml file

- Ter voorbeeld: zie compose hadoop

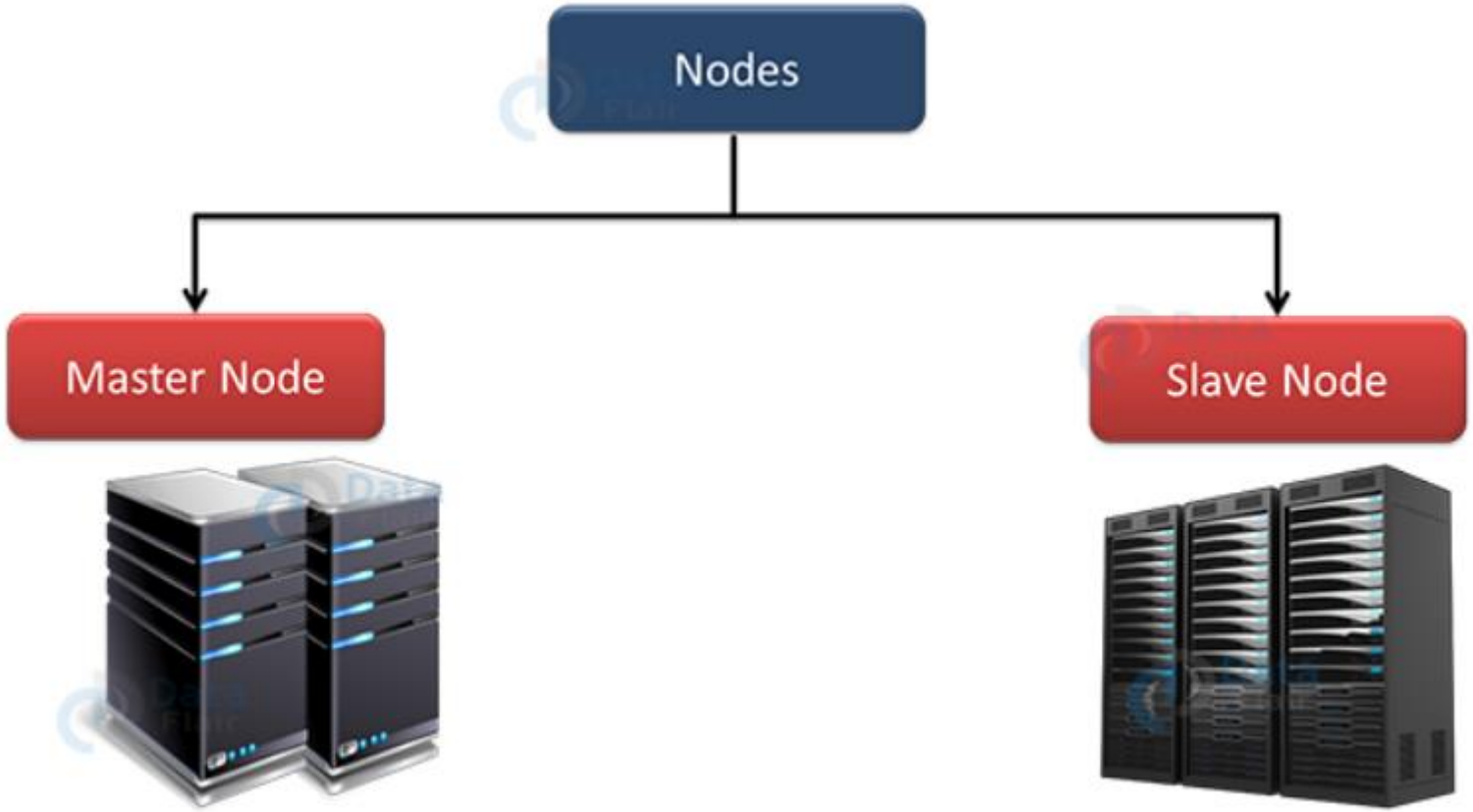
Docker compose

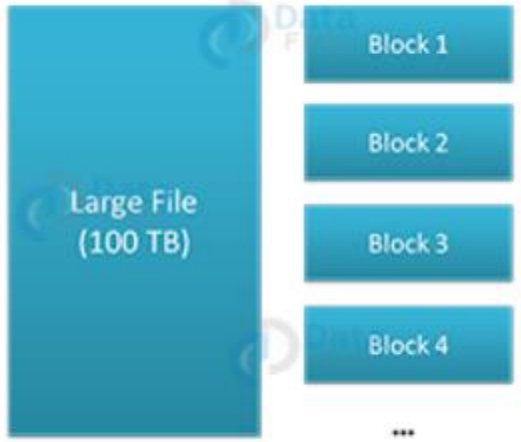


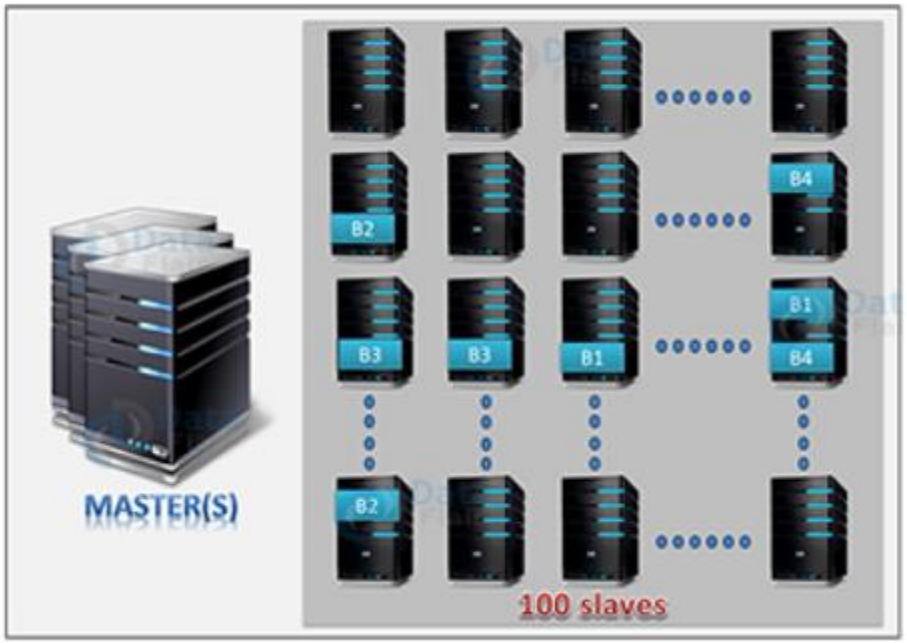
HDFS

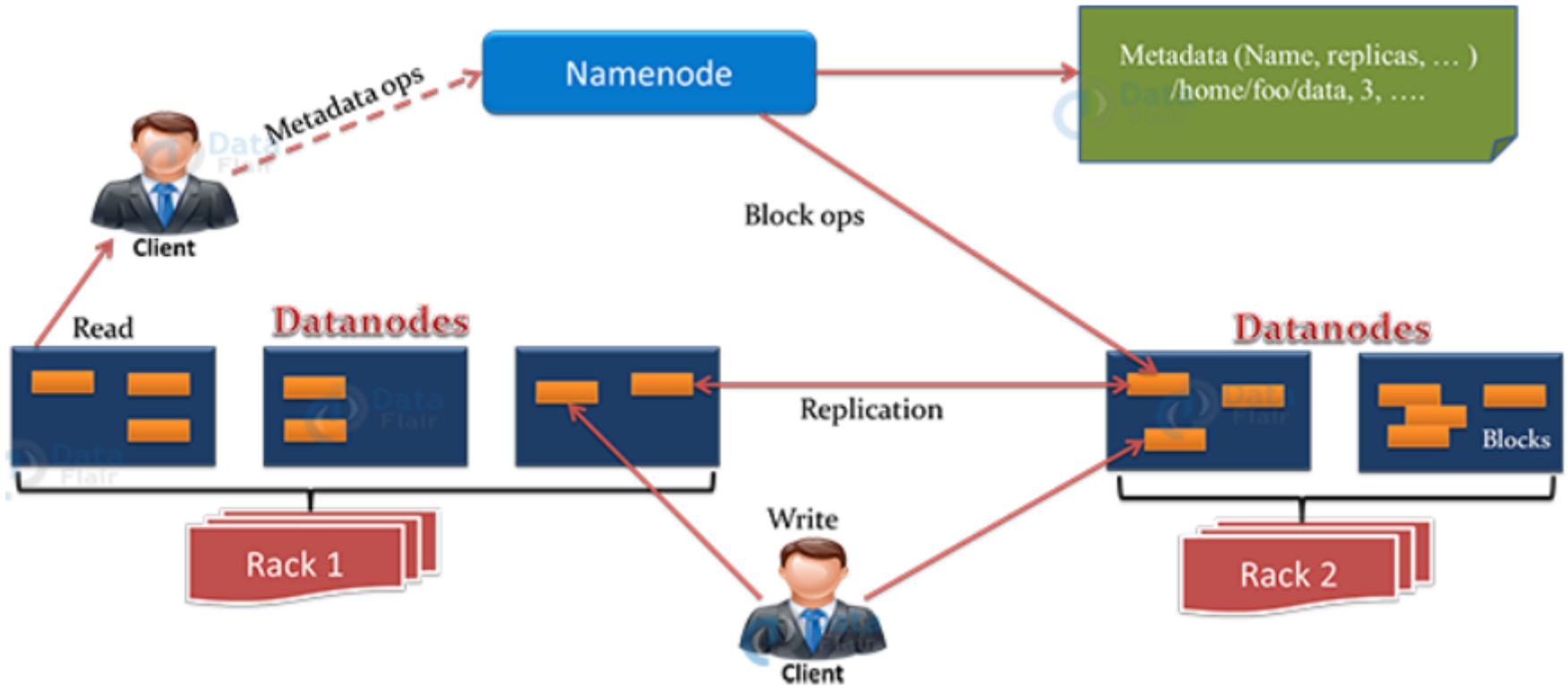
Wat is HDFS?

- Distributed File Storage
- Cluster van commodity hardware
- Fault Tolerance door replicatie van files
  - Verschillende racks, datacenters, continenten
- Scalable
  - Extra nodes kunnen eenvoudig toegevoegd worden
- Parallelle data access


Nodes
Master Node
Slave Node


Large File
(100 TB)
Block 1
Block 2
Block 3
Block 4
...


MASTER(S)
B2
B4
B3
B3
B1
B1
B4
B2
100 slaves


Namenode
Metadata (Name, replicas, …)
/home/foo/data, 3, ….
Metadata ops
Client
Block ops
Read
Datanodes
Datanodes
Replication
Blocks
Write
Rack 1
Rack 2
Client

Features – Distributed Storage

- Onderverdeel files in kleinere delen (Blocks)
- Verdeel de blokken over de nodes
- Repliceer de blokken het gewenste aantal keren (minstens 1 op een andere rack)

Features - Blocks

- Default block size is 128 MB
  - File van 150 MB wordt dus gesplitst in 128 MB en 22 MB
- Beheer van de blokken volledig door de namenode
- Voordeel van grotere block-sizes is dat
  - de file sneller ingelezen wordt
  - Map reduce voert functie uit per block dus niet te veel blocks gewenst.

Features - Replication

- Het aantal keer dat eenzelfde blok voorkomt over alle datanodes
- Dit verhoogt de beschikbaarheid van een blok omdat indien een node crashed, de data beschikbaar is op een andere node.

- Er wordt gepoogd minstens 1 replica op een andere node te plaatsen

- Default waarde is 3

Features – High Availability, Data Reliability en Fault Tolerance

- Datanode fails
  - Datanode stuurt heartbeat naar de namenode -> detecteren van crashed datanode
  - Datanode crashed tijdens opvragen gegevens -> vraag nieuwe locatie aan namenode

- Namenode fails
  - In de master-slave architectuur is de master een single point of failure
  - Vanaf Hadoop 2.0 is er een secondary namenode
- Consistency bij gebruik van meerdere namenodes vereist extra aandacht
  - Identieke gegevens in primary en secondary namenode
  - Wat bij terug online komen van primary namenode

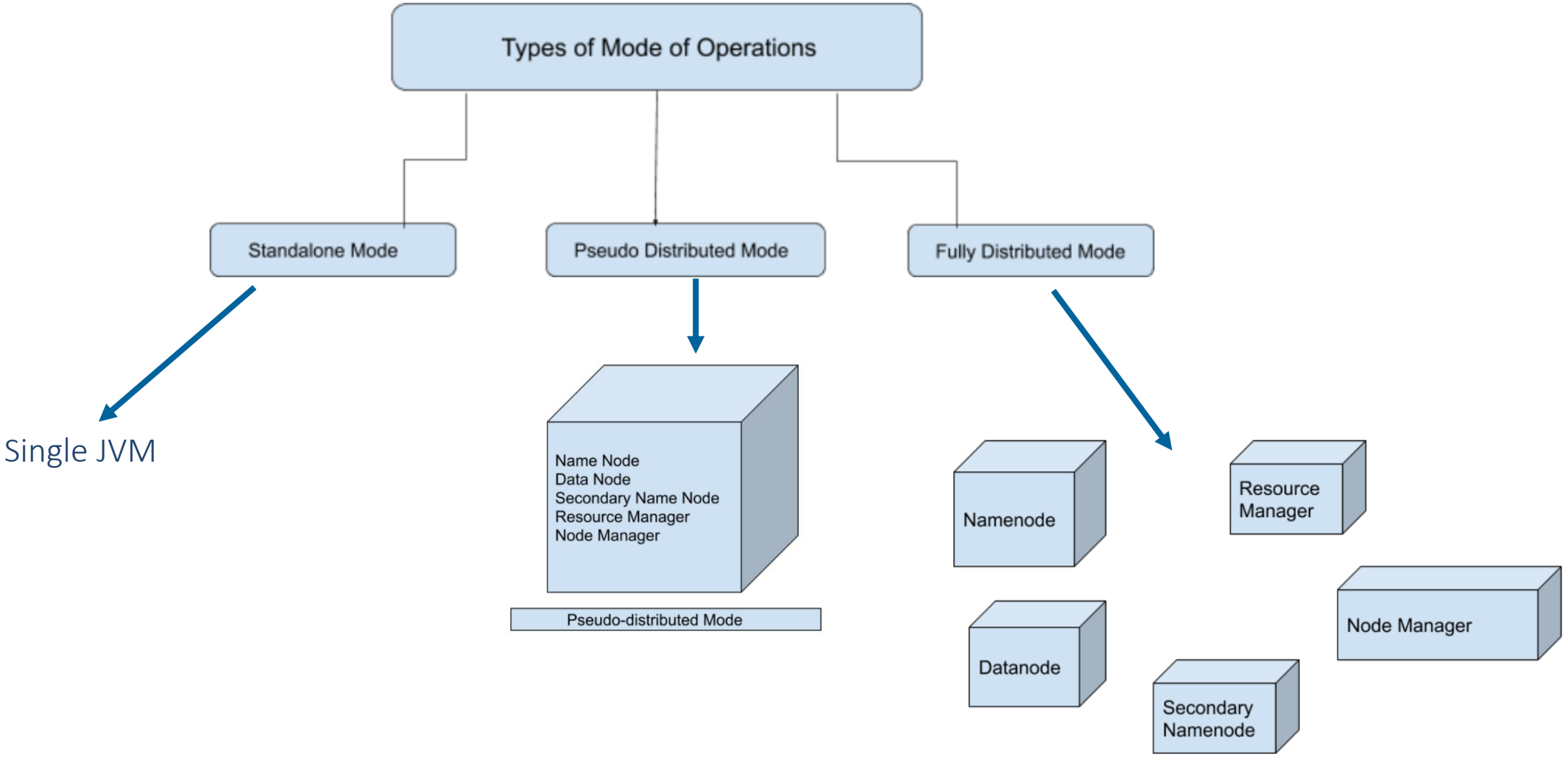
Features - Scalability

- Vertical scaling
  - Meer HDD’s in een node
  - Heeft downtime nodig (om HDD te installeren)
- Horizontal scaling
  - Extra noden toevoegen aan cluster

Features – High Throughput

- Throughput = Hoeveel werk dat gedaan wordt per seconde/minuut/…
- Data wordt parallel gelezen, het werk wordt verdeeld door de verschillende systemen

Hadoop working modes

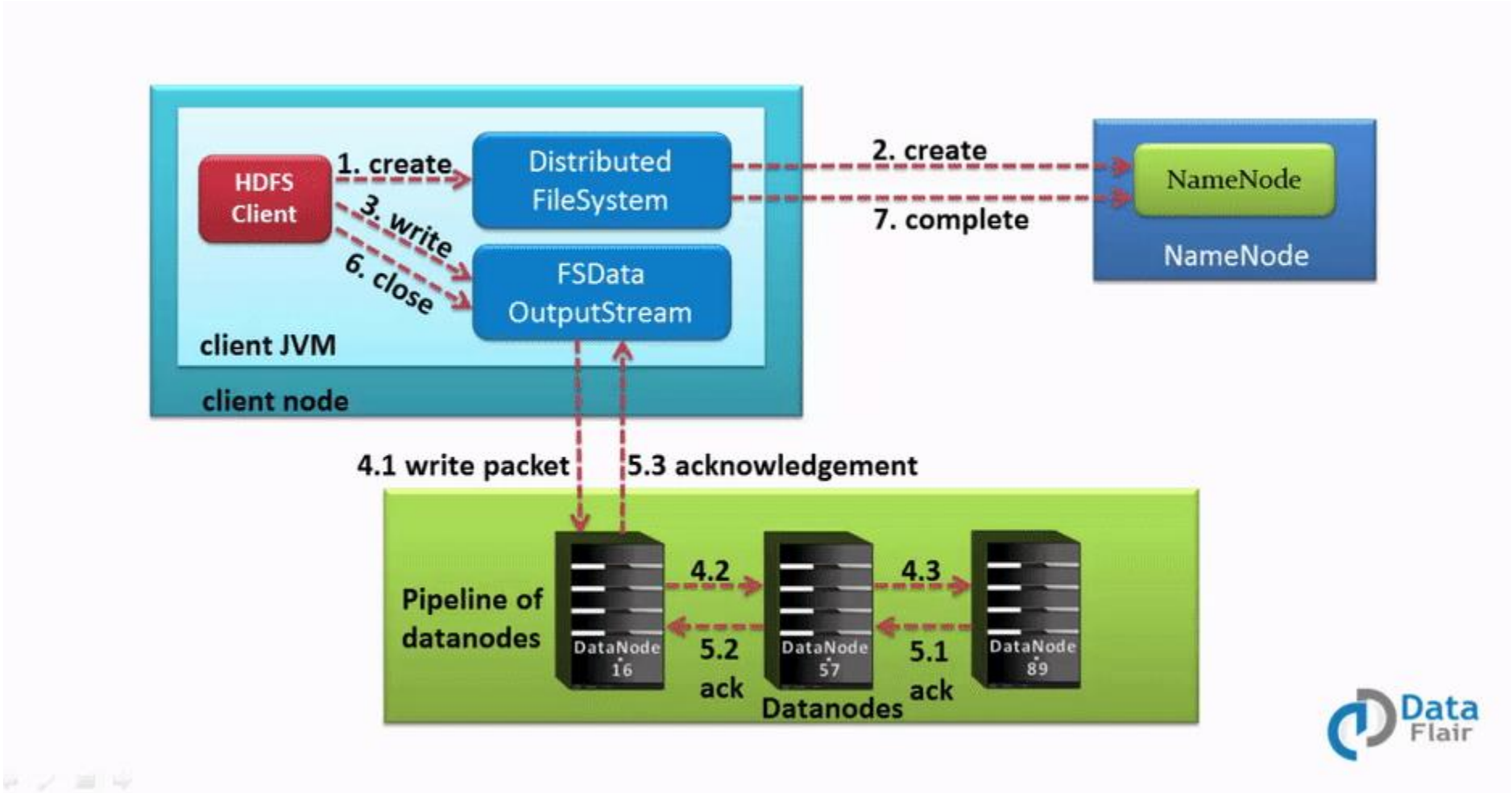


Leg de volgende termen uit

- Horizontal scaling
- Replication
- Block

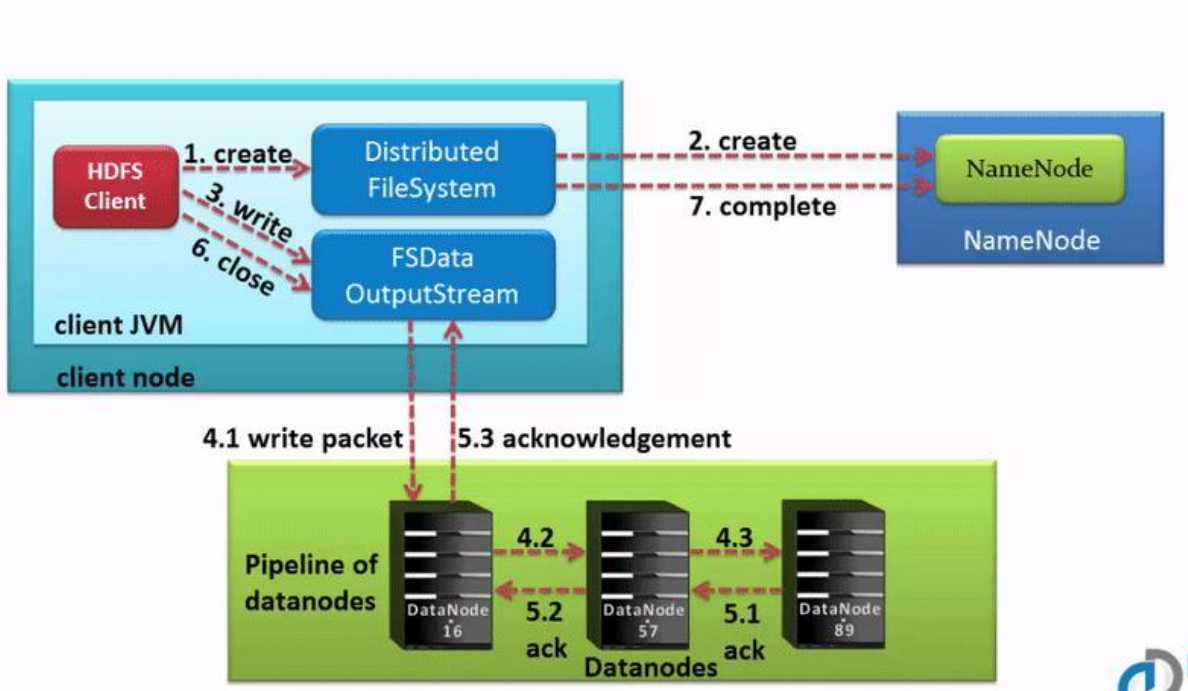
HDFS - Operations

Schrijven van data



Wat met data consistentie?

- Data wordt achter elkaar geschreven van de ene datanode naar de andere
  - Lezen van data kan een willekeurige datanode kiezen waar de node op staat
  - Kan je dan nog de oude data lezen?
    - Ja
- Eventually consistent
  - Veel voorkomend concept
  - Ooit komt het in orde



Hoe bewerken van een hdfs?

- Ofwel via commandline, communicatie met hdfs geconfigureerd in xml-files
  - https://www.informit.com/articles/article.aspx?p=2755708&seqNum=4
  - hadoop/etc/hadoop/core-site.xml
  - hadoop/etc/Hadoop/hdfs-site.xml
- Maak gebruik van libraries om met een HDFS te communiceren
  - Pydoop
  - Snakebite
  - Hdfs

```
from snakebite.client import Client

client = Client('localhost', 9000)
for x in client.ls(['/']):
    print x
```

Pydoop

- https://crs4.github.io/pydoop/api_docs/hdfs_api.html#hdfs-api
- Aanmaken van een connectie:
  - import pydoop.hdfs as hdfs
  - client = hdfs.hdfs(host='localhost', port=9000)
  - client.capacity()

HdfsCLI – python bindings

- API: https://hdfscli.readthedocs.io/en/latest/
- Bevat functies voor
  - te lezen
  - te schrijven
  - te verkennen
  - te manipuleren

Aanmaken van bestanden en folders

CLI

- Exists: hdfs dfs -test -d hdfs_path
- Mkdir: hdfs dfs -mkdir -p /bigdata/03_HDFS
- hadoop fs -put /path/in/linux /hdfs/path
- hadoop fs -get /hdfs/path /path/in/linux

Pydoop

- client.makedirs('HDFS’)
- client.upload('HDFS/ulysses2.txt', 'ulysses.txt')
- client.write('HDFS/test.txt', 'test2.txt')

Bekijken van het filesysteem

- hdfs dfs -ls command
- hdfs dfs –usage
- hdfs dfs -cat <hdfs_filename> | head -n 5

- print(client.acl_status("HDFS"))
- print(client.checksum('HDFS/test.txt'))
- print(client.content('HDFS/test.txt'))
- print(client.list('HDFS'))
- print(client.walk('/'))

Aanpassen van het filesysteem

- hadoop fs -mv oldname newname
- hdfs dfs -rm path (-r voor folders)
- hdfs dfs –setrep –w 3 /tmp/logs/file.txt

- client.rename(“old", “new")
- client.delete("HDFS/test.txt")
- client.set_replication("davinci.txt", 5)
- client.set_permission(“file”, 755)

Welke processen zijn er allemaal nodig voor een HDFS?

- Namenode
  - manages the file system namespace and regulates access to files by clients
  - http://localhost:9870
- SecondaryNamenode
  - Backup incase primary node fails
  - http://localhost:9868

Welke processen zijn er allemaal nodig voor een HDFS?

- Datanode
  - usually, one per node in the cluster
  - manage storage attached to the nodes that they run on
  - http://localhost:9864

Welke processen zijn er allemaal nodig voor een HDFS?

- Nodemanager
  - responsible for launching and managing containers on a node. Containers execute tasks as specified by the AppMaster.
  - http://localhost:8042
- Resourcemanager
  - responsible for tracking the resources in a cluster, and scheduling applications
  - http://localhost:8088

- Lijst van alle gebruikte poorten: https://kontext.tech/article/265/default-ports-used-by-hadoop-services-hdfs-mapreduce-yarn

https://youtu.be/4Gfl0WuONMY