CRATE

UNDERSTANDING DATABASES FOR DISTRIBUTED DOCKER APPLICATIONS

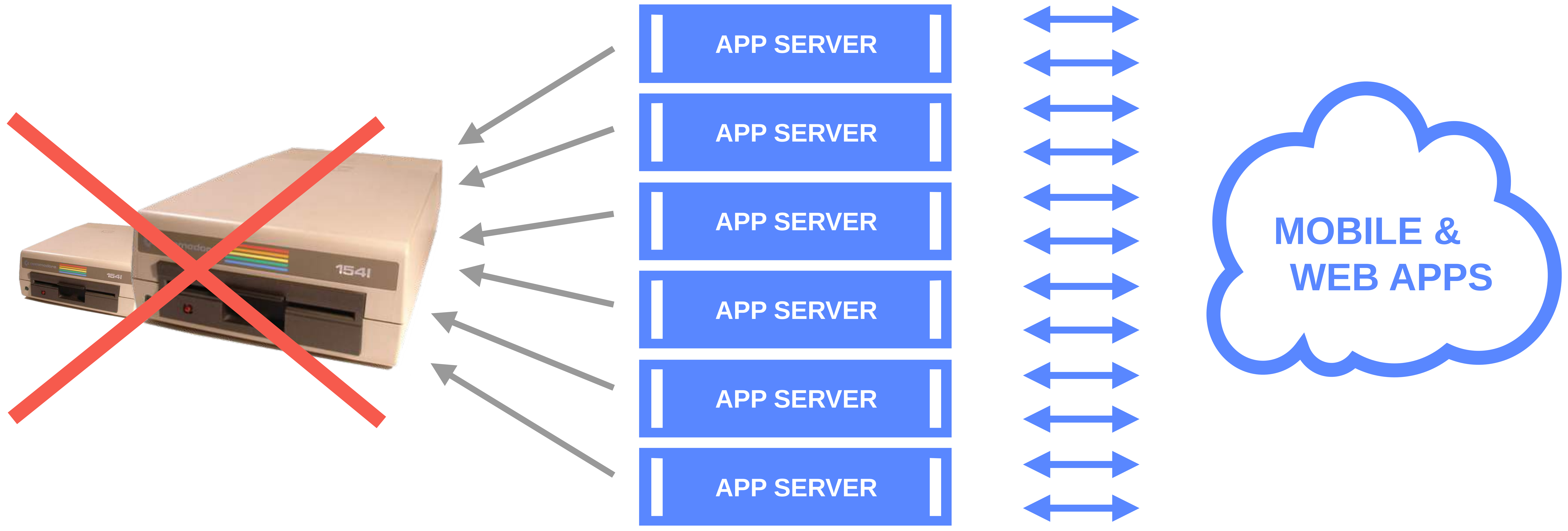
Berlin Buzzwords, June 1st

ABOUT ME

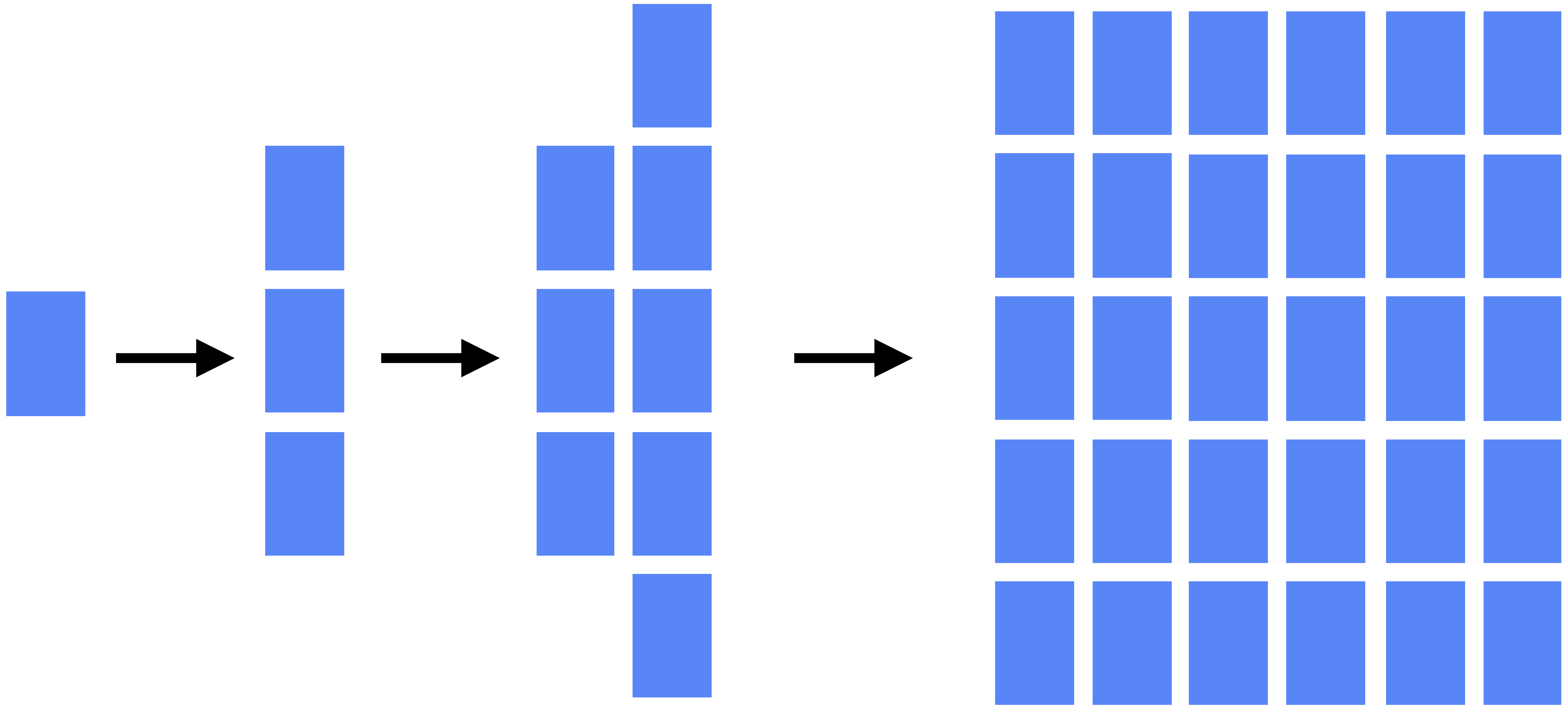
- London, Melbourne, Leipzig, Berlin
- Developer, Designer, Writer, Musician
- Developer Advocate

chris@crate.io
@chrischinch
@crateio

THE SINGLE NODE DATABASE BACKEND IS DEAD



THE FUTURE OF DATABASES IS DISTRIBUTED

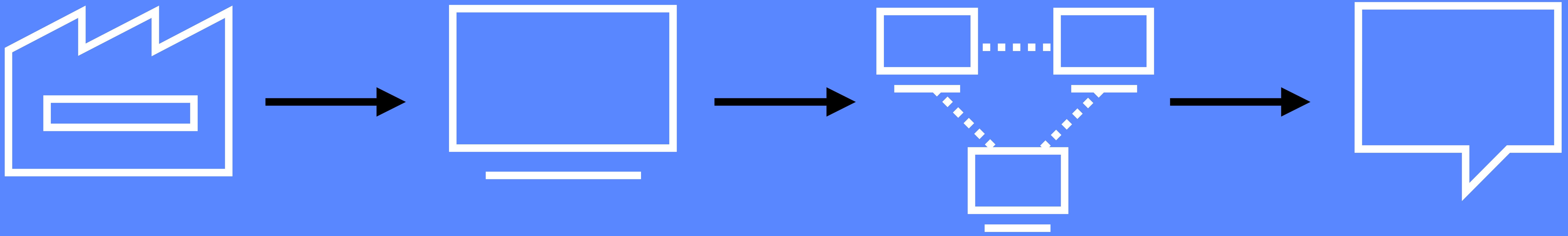


Google

twitter

facebook

MICROSERVICES & DOCKER



Datacenter Snowflakes
- Deploy in months
- Live for years

Virtualized and Cloud
- Deploy in minutes
- Live for weeks

Docker Containers
- Deploy in seconds
- Live for minutes/hours

Compute Service
- Deploy in milliseconds
- Live for seconds

Speed enables and encourages new micro service architectures

NOW: ZOO OF TECHNOLOGIES

Relational Database

MySQL

Document Database

mongoDB

Powerful Search

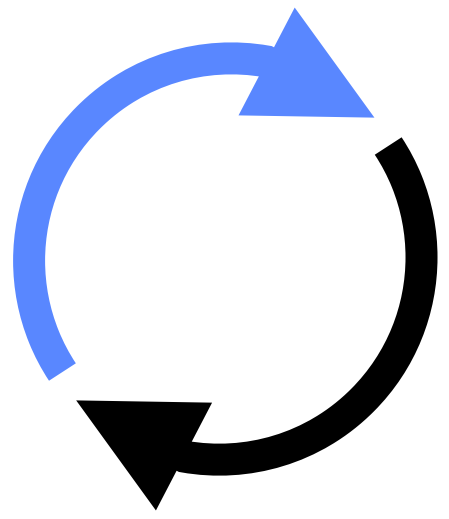
elasticsearch

Apache Solr

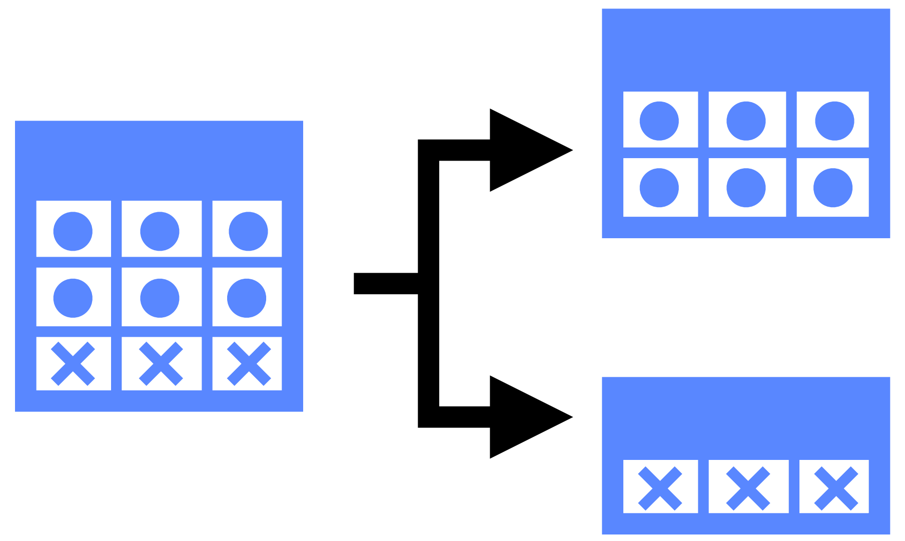
Blob Store

GridFS
HDFS

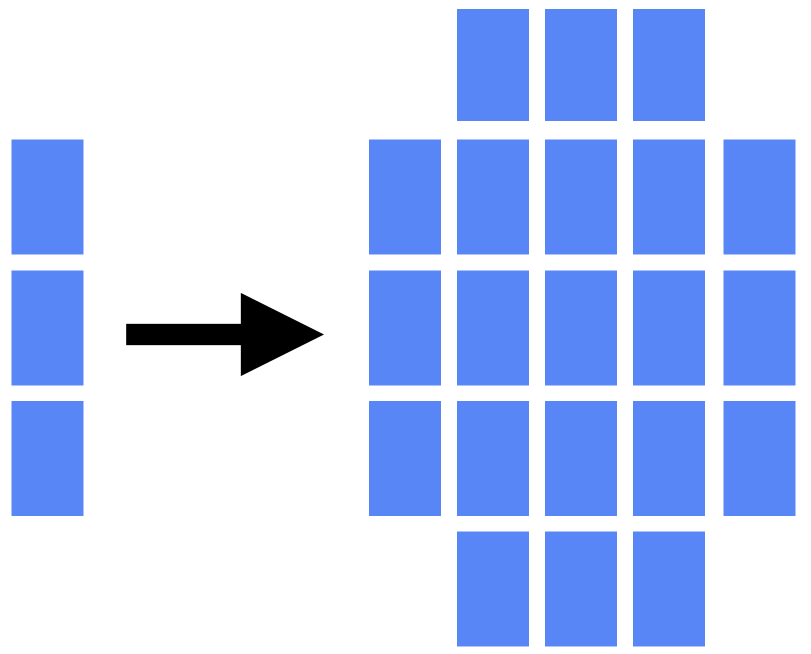
sync



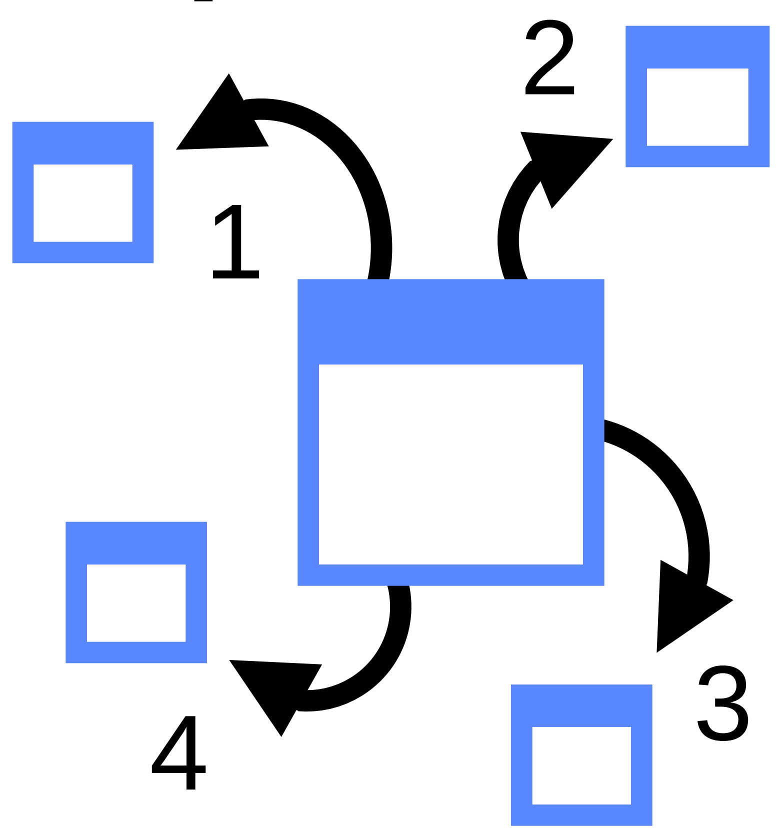
sharding



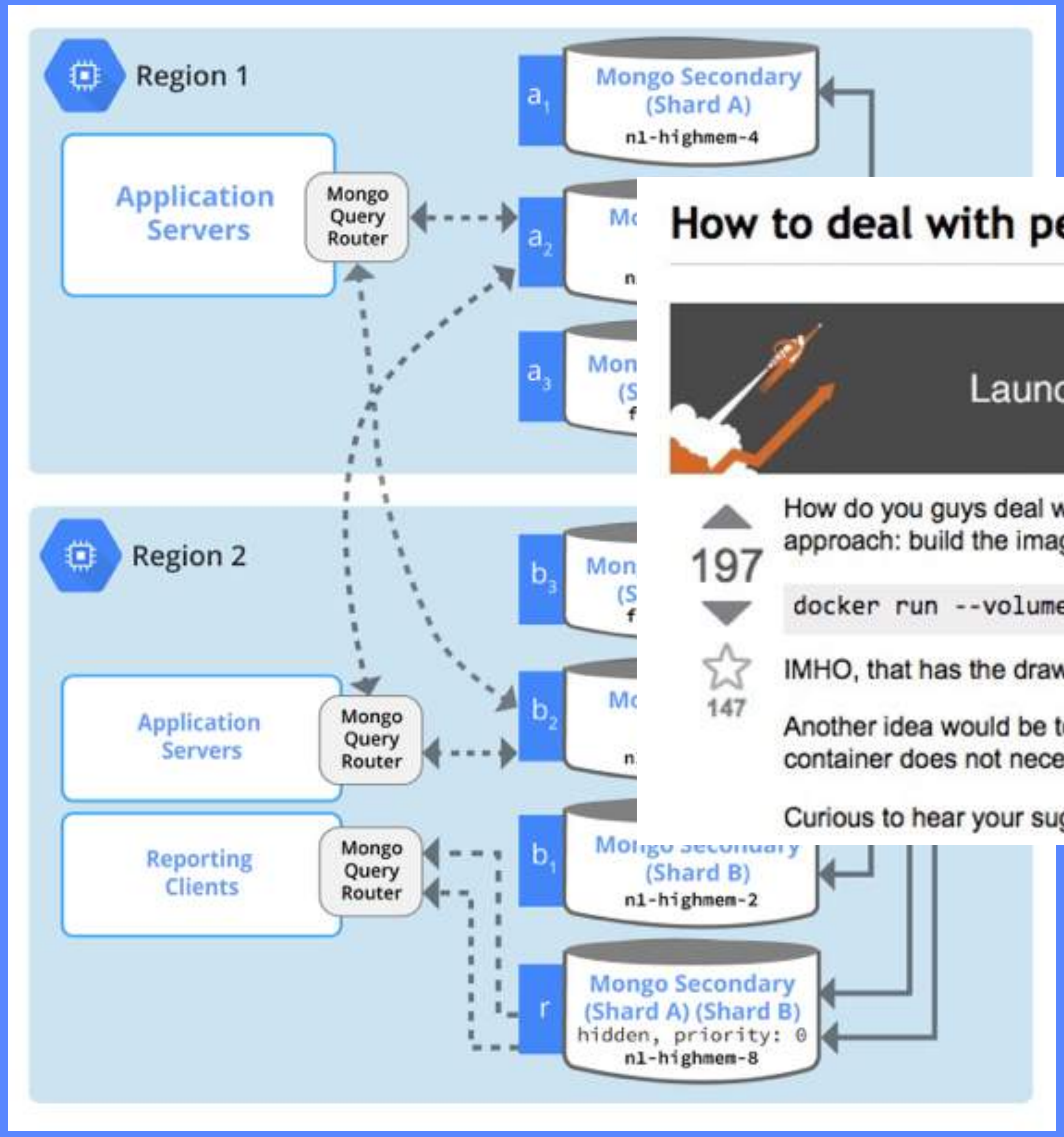
scaling

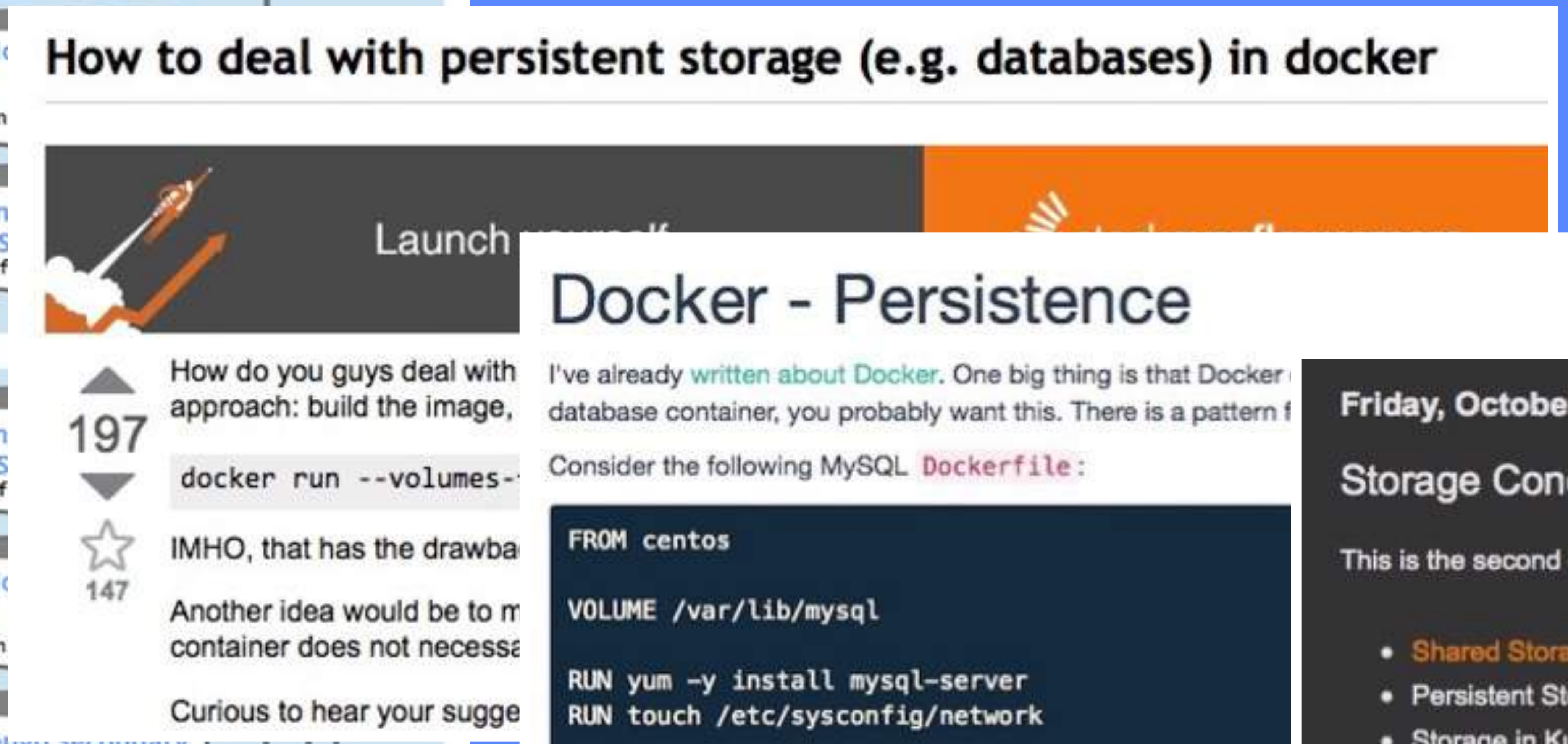


replication



TRADITIONAL DBS ON DOCKER




How to deal with persistent storage (e.g. databases) in docker

Launch

197

147

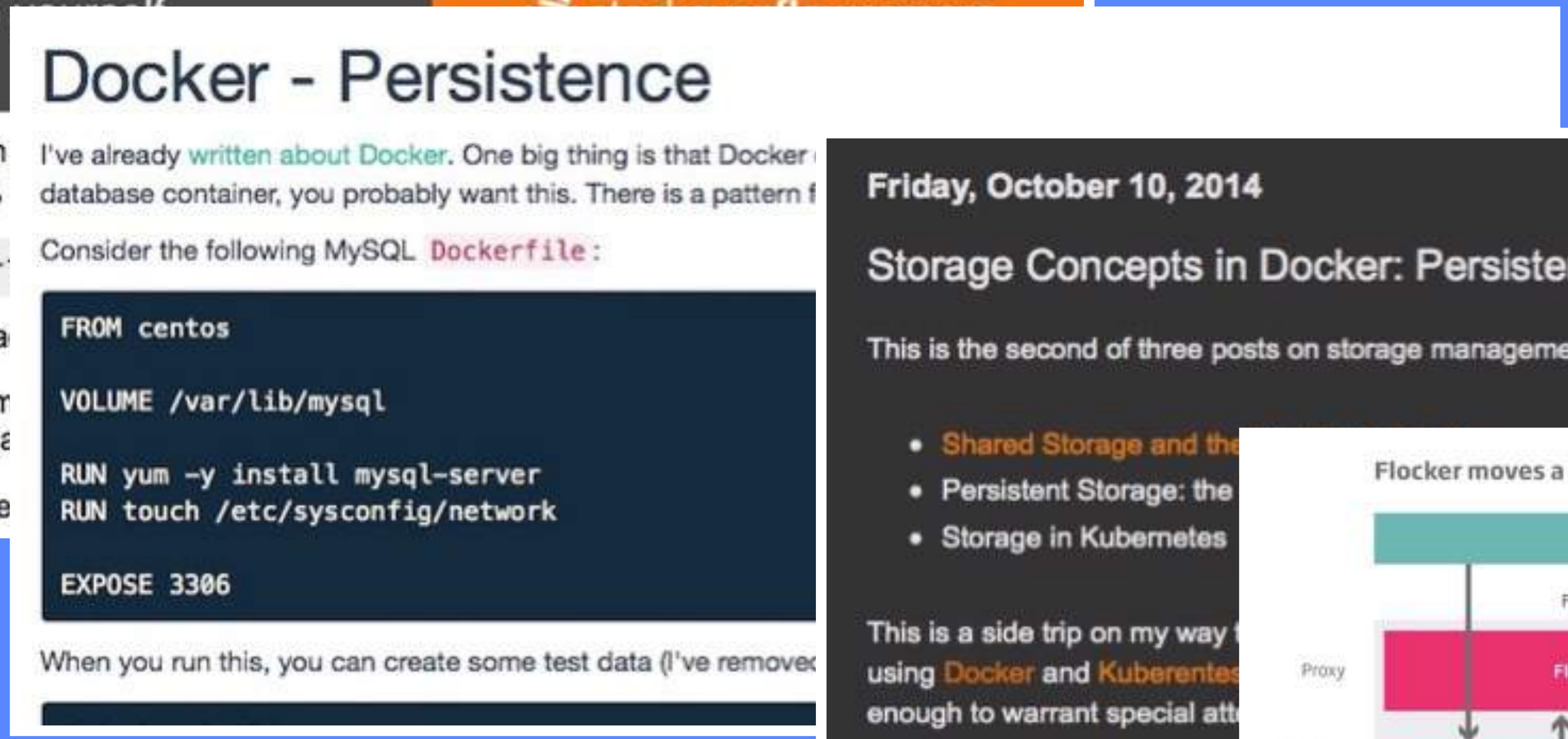
How do you guys deal with
approach: build the image,

```
docker run --volumes-
```

IMHO, that has the drawba

Another idea would be to m
container does not necessa

Curious to hear your sugge


Docker - Persistence

I've already written about Docker. One big thing is that Docker
database container, you probably want this. There is a pattern f

Consider the following MySQL `Dockerfile`:

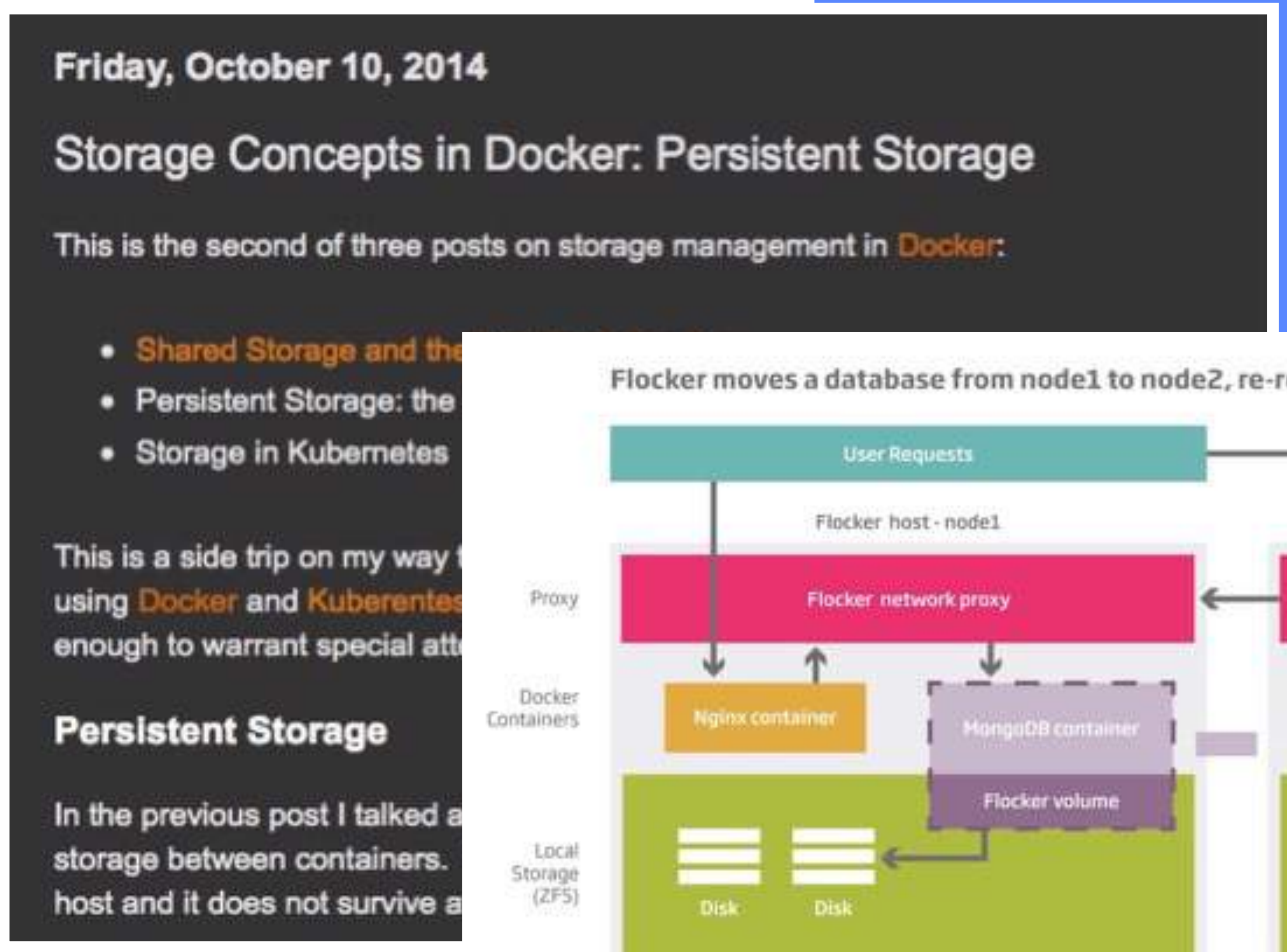
```
FROM centos

VOLUME /var/lib/mysql

RUN yum -y install mysql-server
RUN touch /etc/sysconfig/network

EXPOSE 3306
```

When you run this, you can create some test data (I've removed


Friday, October 10, 2014

Storage Concepts in Docker: Persistent Storage

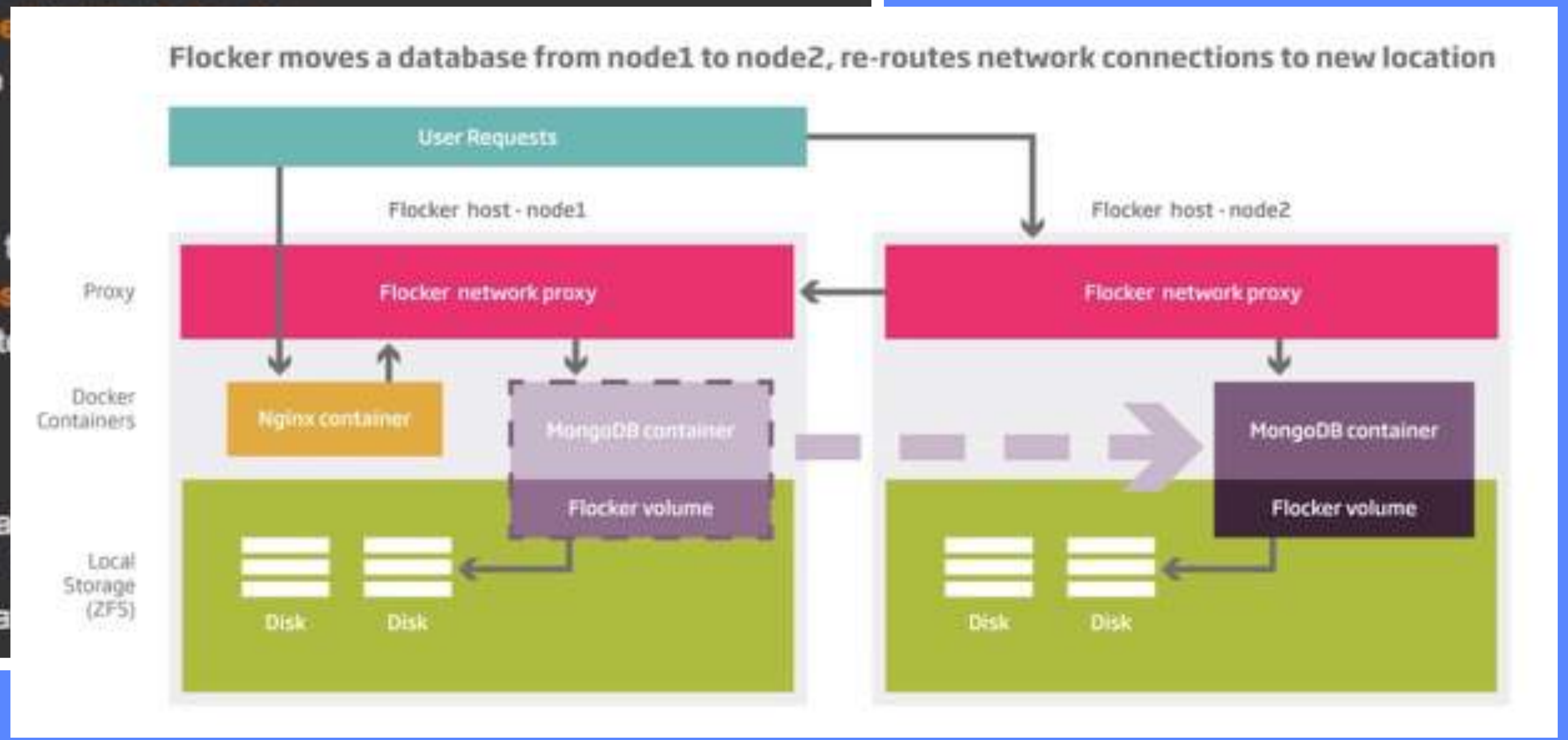
This is the second of three posts on storage management in Docker:

- Shared Storage and the
- Persistent Storage: the
- Storage in Kubernetes

This is a side trip on my way
using Docker and Kuberentes
enough to warrant special att

Persistent Storage

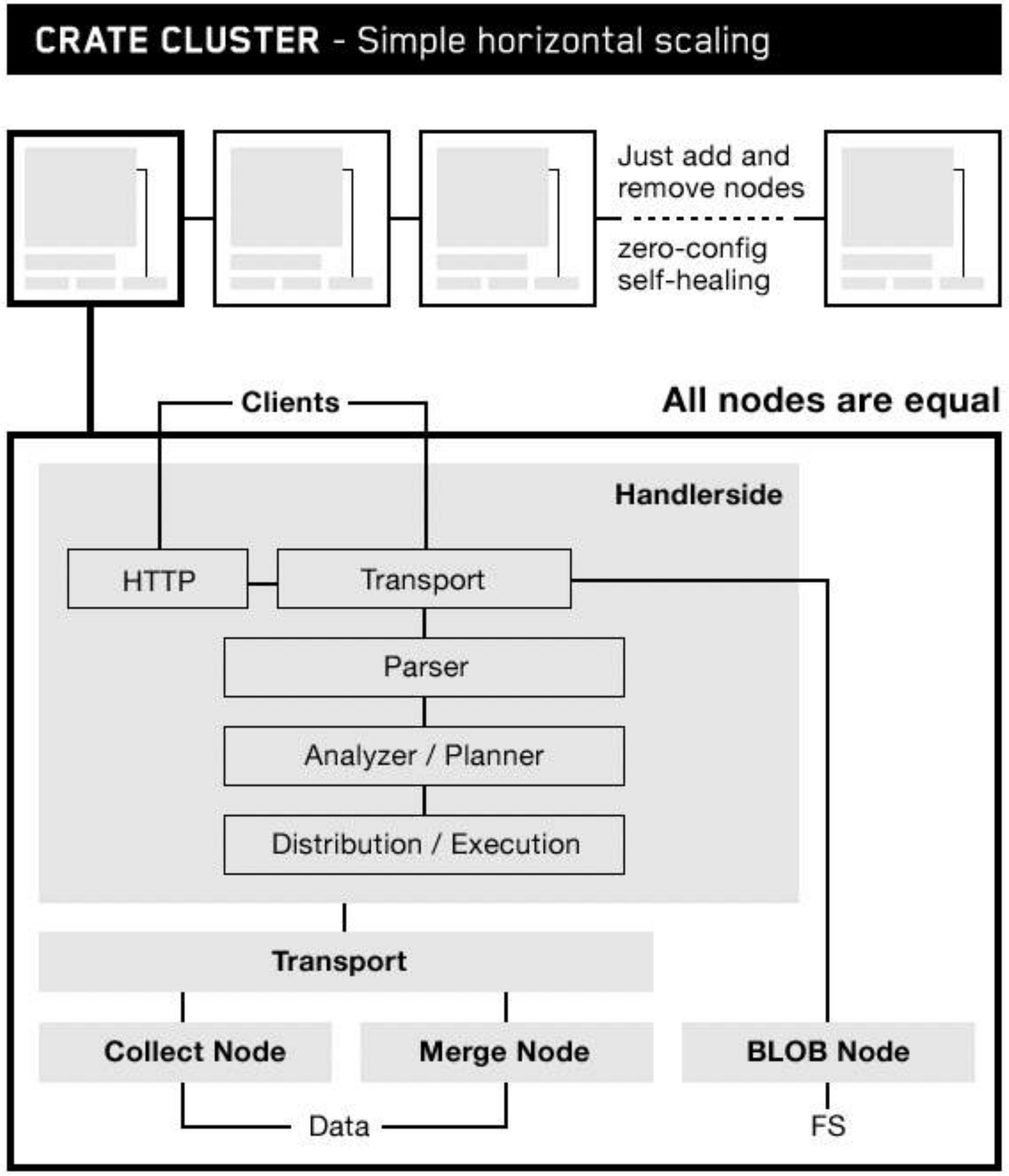
In the previous post I talked a
storage between containers.
host and it does not survive a

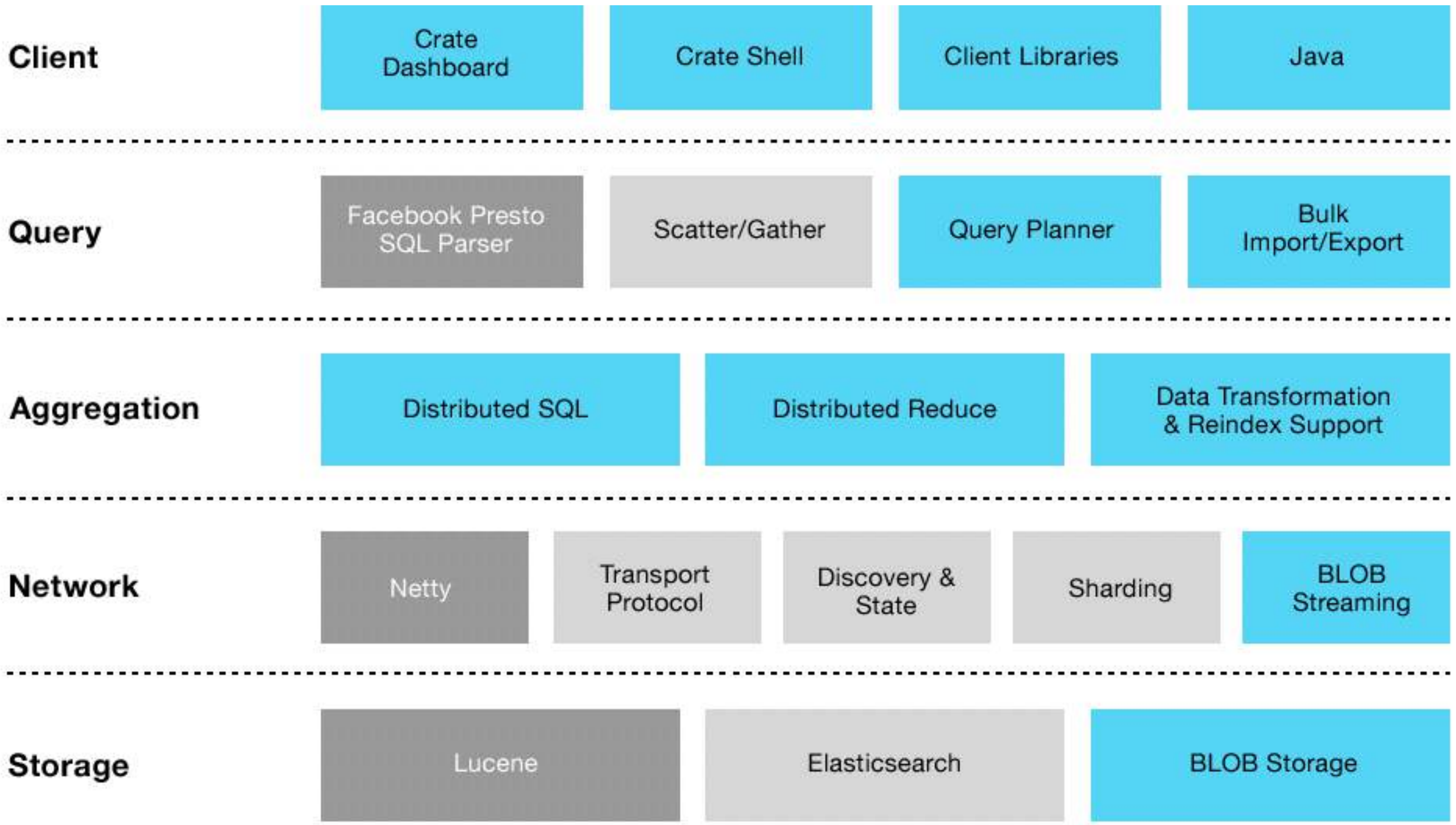


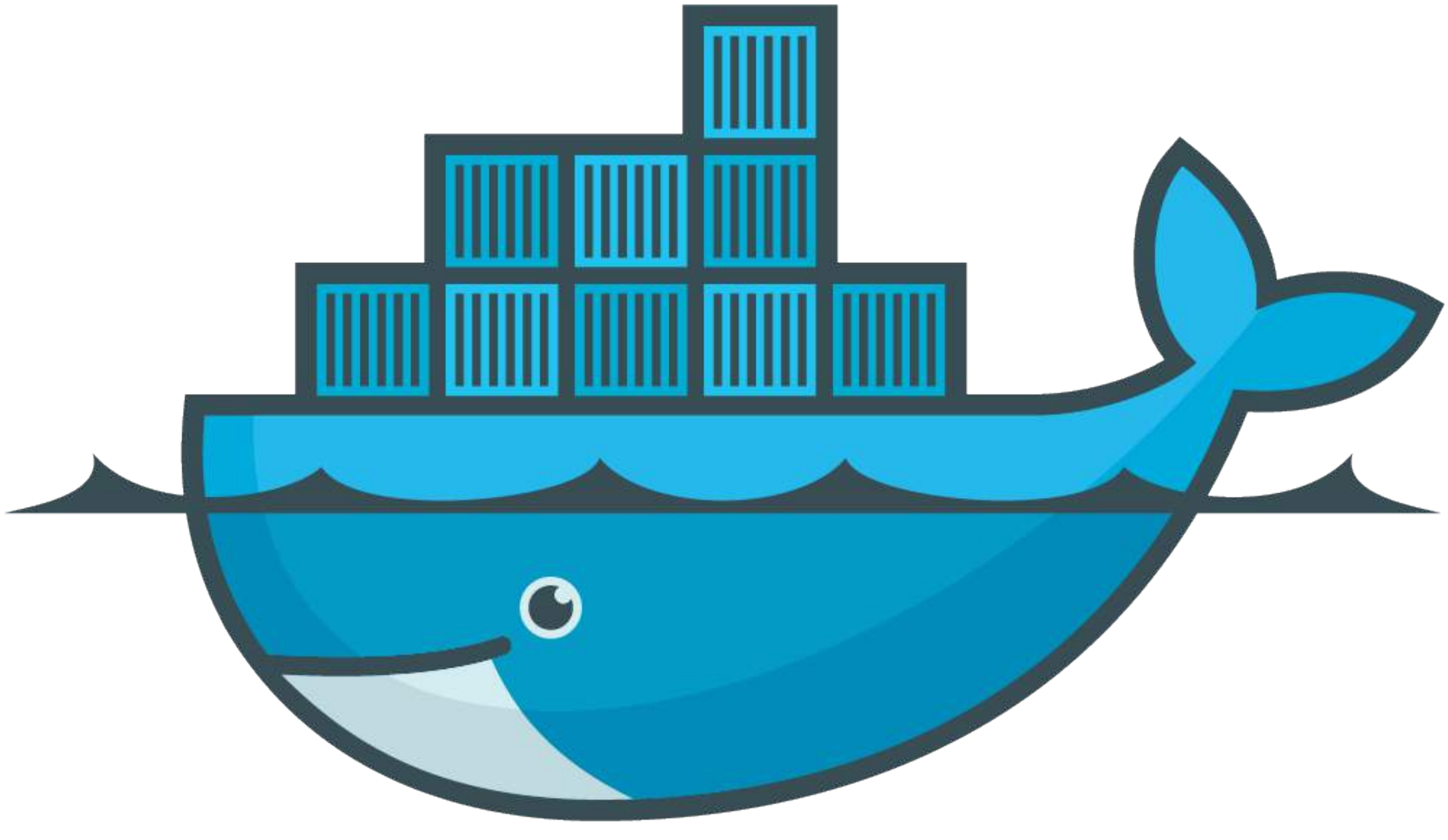
CRATE

CRATE IS...

- built on a **NoSQL architecture**
- a **distributed SQL** Database (supporting semi-structured records)
- extremely **simple** to install/operate
- superfast, powerful **search**
- horizontally scaling, **elastic**, resilient
- eventual-consistent, high concurrency




Client
Crate Dashboard
Crate Shell
Client Libraries
Java
Query
Facebook Presto SQL Parser
Scatter/Gather
Query Planner
Bulk Import/Export
Aggregation
Distributed SQL
Distributed Reduce
Data Transformation & Reindex Support
Network
Netty
Transport Protocol
Discovery & State
Sharding
BLOB Streaming
Storage
Lucene
Elasticsearch
BLOB Storage



```
> docker pull crate:latest
> docker run -d -P crate
> docker run -d -P crate
> docker run -d -p 4200:4200 -p 4300:4300 crate
> docker run -d -p 4200:4200 -p 4300:4300
    --volume /data:/data
    --env CRATE_HEAP_SIZE=8g
    crate crate
    -Des.path.data="/data/data1,/data/data2"
    -Des.multicast.enabled=false
    -Des.network.publish_host=$PRIVATE_IP
    -Des.discovery.zen.ping.unicast.hosts=$HOSTS
```

FOR EVERY DEVELOPER

- data is as **easy** to scale as the application
- Familiar, standard SQL Syntax.
- **Automatic** Configuration, Sharding and Replication
- Support for tables, **semi-structured** records and binary data and **search**.

open source initiative

python™ php

node JS™

Java Ruby

CLIENT DEMO

CRATE

COMPOSE

EXAMPLE

```
crate:
  image: crate
  ports:
    - "4200:4200"
    - "4300:4300"
  volumes:
    - /mnt/data/crate:/data
  environment:
    CRATE_HEAP_SIZE: 16g
  command: crate -Des.cluster.name=my-crate -Des.node.name=crate-1 -
Des.network.publish_host=cratedemo.dev

node:
  build: .
  ports:
    - "8000:8000"
  links:
    - crate
```

CRATE

MACHINE & SWARM

```
> docker-machine create
    --driver virtualbox
    staging

> eval "$(docker-machine env staging)"

> docker run -d -p 4200:4200 -p 4300:4300 crate
> docker-machine create
    -d virtualbox
    env-crate
> $(docker-machine env env-crate)
> docker run swarm create
> echo "export TOKEN=xx" >> .bash_profile
> source .bash_profile
> docker-machine create
    --driver virtualbox
    --swarm
    --swarm-master
    --swarm-discovery
    token://${TOKEN}
    crate-swarm
```

```
> docker-machine create
    --driver virtualbox
    --swarm
    --swarm-discovery
    token://yy
    crate-swarm-node1
```

```
> docker-machine ls
> $(docker-machine env --swarm crate-swarm)
> docker info
```

```
docker -H tcp://HOST:2376
     run -d -p 4200:4200 -p 4300:4300
     crate:latest crate
     -Des.cluster.name=crate-swarm
     -Des.multicast.enabled=false
     -Des.transport.publish_host=HOST
     -Des.discovery.zen.ping.unicast.hosts="HOSTS"
     -Des.discovery.zen.minimum_master_nodes=x
```

CRATE.IO/BLOG/CRATE-WITH-DOCKER-AND-WEAVE

USE CASES

WELL SUITED

- High volume, semistructured/ dynamic data
- Operational datastore for web applications that require powerful fulltext search
- Elastic datastore for dynamic startups
- Real time analytics and business intelligence

NOT WELL SUITED

- Systems that require strong consistency
- Systems that require transactions
- Strong relational data

CRATE

https://github.com/crate

http://stackoverflow.com/tags/crate/

IRC #crate on freenode

https://twitter.com/CrateIO

https://fb.me/cratedata

support@crate.io

https://crate.io

CRATE

ARCHITECTURE

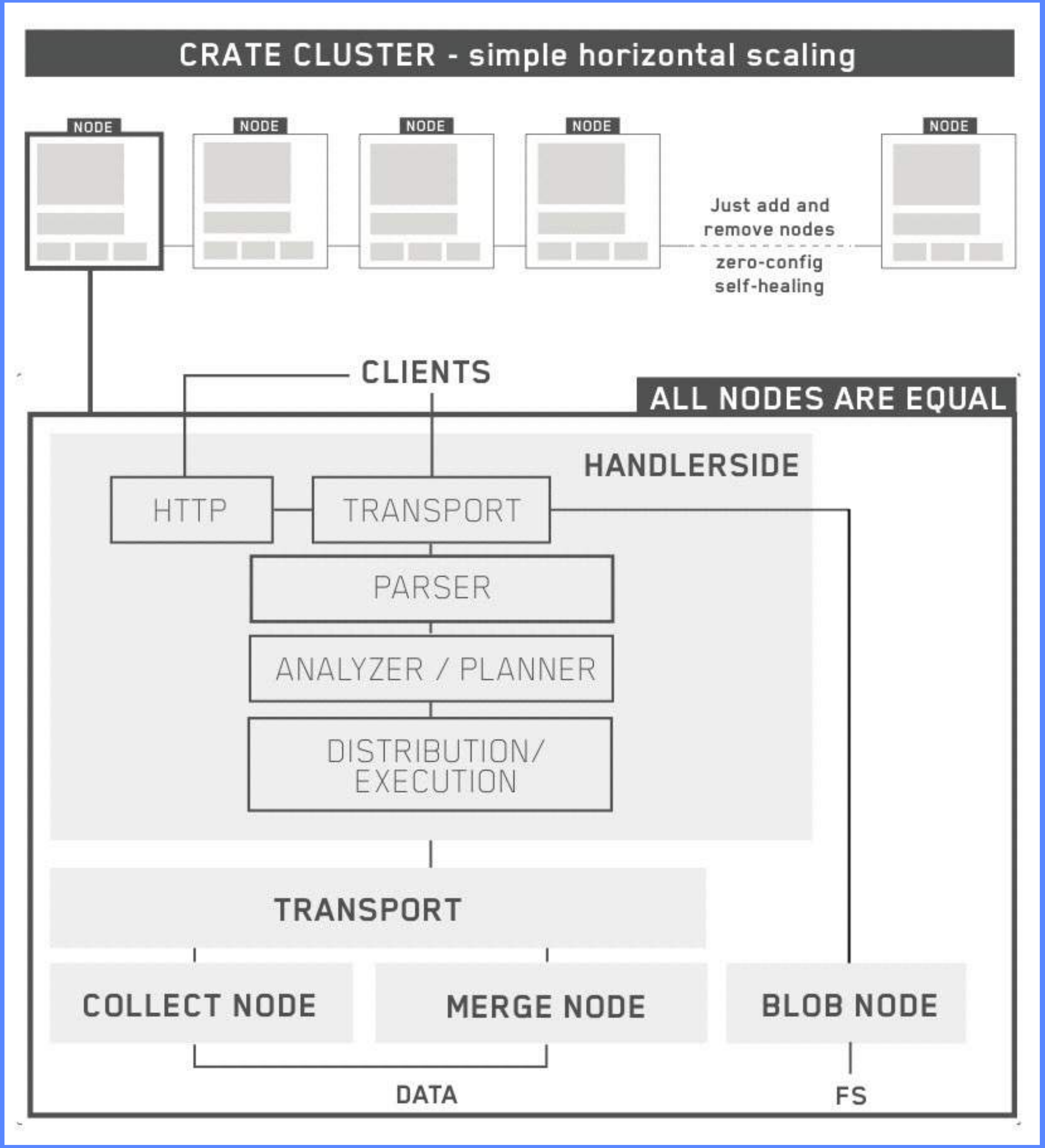
DISTRIBUTED REAL-TIME SQL

- Automatic Sharding, Partitioning, Replication
- Optimistic Concurrency Control, Read-After Write consistency
- Aggregations are superfast and executed truly distributed by realtime Map/Reduce.
- Crate uses Standard SQL* and can handle thousands of read/write connections per node

* equi-JOINs to be released soon

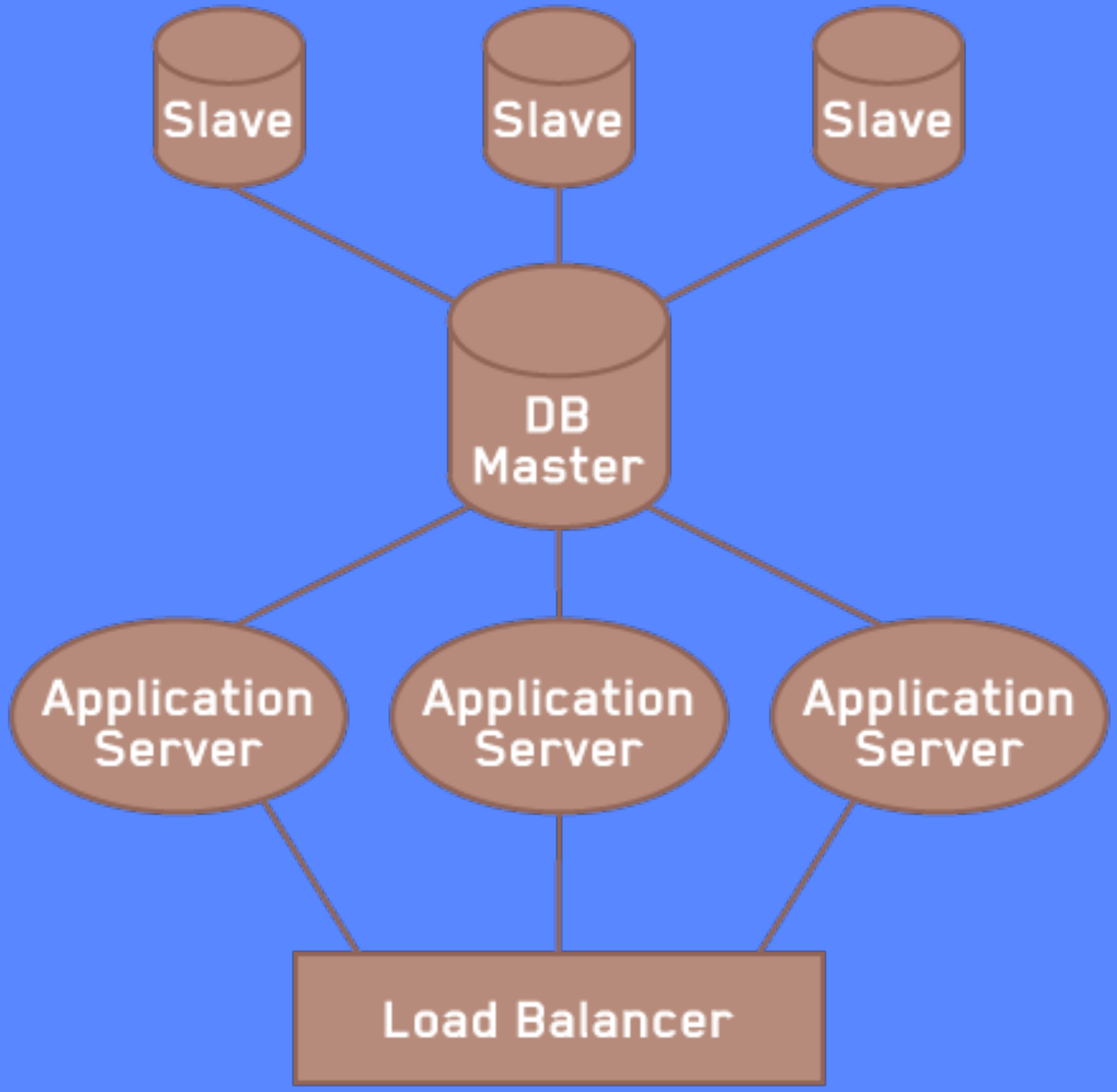
SHARED-NOTHING ARCHITECTURE

- Nodes do not share states
- All nodes are equal
- Each node is independent & self-sufficient
- Each node can perform every task



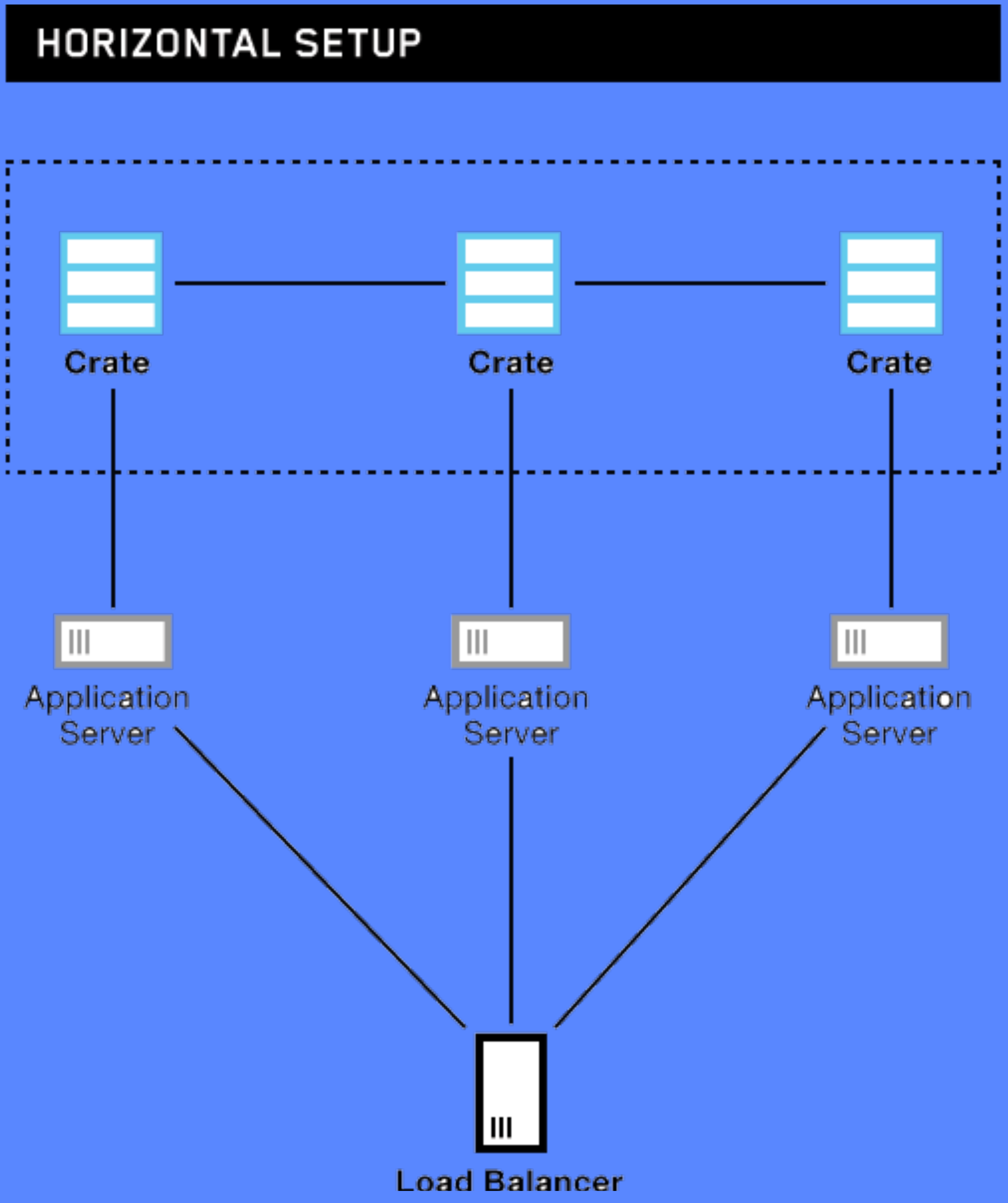
HORIZONTAL SCALING

- Quantity over Quality
- Increase amount of (smaller) nodes instead of scaling a single node
- Distributed/parallel computation power
- New way of building webservices



HORIZONTAL SCALING

- Quantity over Quality
- Increase amount of (smaller) nodes instead of scaling a single node
- Distributed/parallel computation power
- New way of building webservices


HORIZONTAL SETUP
Crate
Crate
Crate
Application Server
Application Server
Application Server
Load Balancer