Introduction to

MongoDB

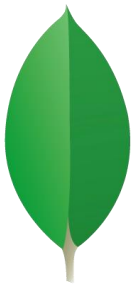
The Great Divide

Easy Flexible Scalable

What is MongoDB ?

•Scalable High-Performance Open-source, Document-orientated database.

- Built for Speed
- Rich Document based queries for Easy readability.
- Full Index Support for High Performance.
- Replication and Failover for High Availability.
- Auto Sharding for Easy Scalability.
- Map / Reduce for Aggregation.



Why use MongoDB?

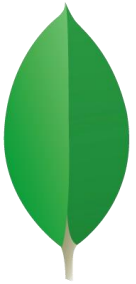
- SQL was invented in the 70's to store data.

- MongoDB stores documents (or) objects.

- Now-a-days, everyone works with objects (Python/Ruby/Java/etc.)

- And we need Databases to persist our objects. Then why not store objects directly ?

- Embedded documents and arrays reduce need for joins. No Joins and No-multi document transactions.



What is MongoDB great for?

- RDBMS replacement for Web Applications.

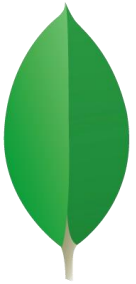
  Semi-structured Content Management.

  Real-time Analytics & High-Speed Logging.

- Caching and High Scalability

  Web 2.0, Media, SAAS, Gaming

  HealthCare, Finance, Telecom, Government



Not great for?

- Highly Transactional Applications.
- Problems requiring SQL.

Some Companies using MongoDB in Production

Let’s Dive in !

When I say

- Made up of Multiple Collections.
- Created on-the-fly when referenced for the first time.

When I say

- Schema-less, and contains Documents.

  Indexable by one/more keys.

- Created on-the-fly when referenced for the first time.

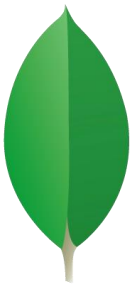
  Capped Collections: Fixed size, older records get dropped after reaching the limit.

When I say

- Stored in a Collection.
- Can have _id key – works like Primary keys in MySQL.
- Supported Relationships – Embedded (or) References.
- Document storage in BSON (Binary form of JSON).

Understanding the Document Model.

```
var p = {
'_id': '3432',
'author': DBRef('User', 2),
'title': 'Introduction to MongoDB',
'body': 'MongoDB is an open sources.. ',
'timestamp': Date('01-04-12'),
'tags': ['MongoDB', 'NoSQL'],
'comments': [{'author': DBRef('User', 4),
                'date': Date('02-04-12'),
                'text': 'Did you see.. ',
                'upvotes': 7, ... ]
}
> db.posts.save(p);
```



Secondary Indexes

Create Index on any field in the document

// 1 means ascending, -1 means descending

> db.posts.ensureIndex({'author': 1});

//Index Nested Documents

> db.posts.ensureIndex('comments.author': 1);

// Index on tags

> db.posts.ensureIndex({'tags': 1});

// Geo-spatial Index

> db.posts.ensureIndex({'author.location': '2d'});

What about Queries? So Simple

// find posts which has 'MongoDB' tag.

> db.posts.find({tags: 'MongoDB'});

// find posts by author's comments.

>db.posts.find({'comments.author': DBRef('User',2)}).count();

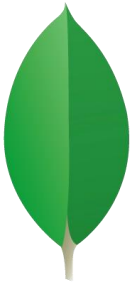
// find posts written after 31st March.

> db.posts.find({'timestamp': {'gte': Date('31-03-12')}});

// find posts written by authors around [22, 42]

> db.posts.find({'author.location': {'near':[22, 42]}});

$gt, $lt, $gte, $lte, $ne, $all, $in, $nin, count, limit, skip, group, etc...



Whatabout Updates? Atomic Operations makes it simple

```
db.posts.update({_id: '3432'},
{'title': 'Introduction to MongoDB (updated)',
'text': 'Updated text',
${addToSet: {'tags': 'webinar'}});
```

$set, $unset

$push, $pull, $pop, $addToSet

$inc, $decr, many more…

Where are my joins and transactions? !!!

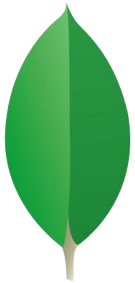
Some Cool features

- Geo-spatial Indexes for Geo-spatial queries.
  $near, $within_distance, Bound queries (circle, box)

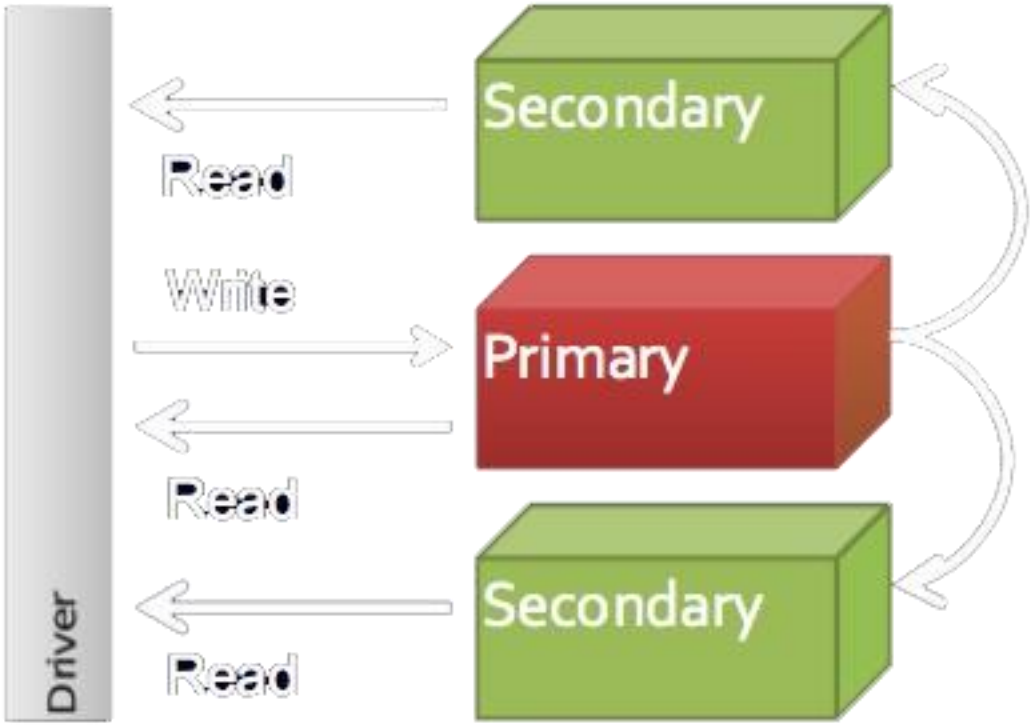
- GridFS
  Stores Large Binary Files.

- Map/Reduce
  GROUP BY in SQL, map/reduce in MongoDB.

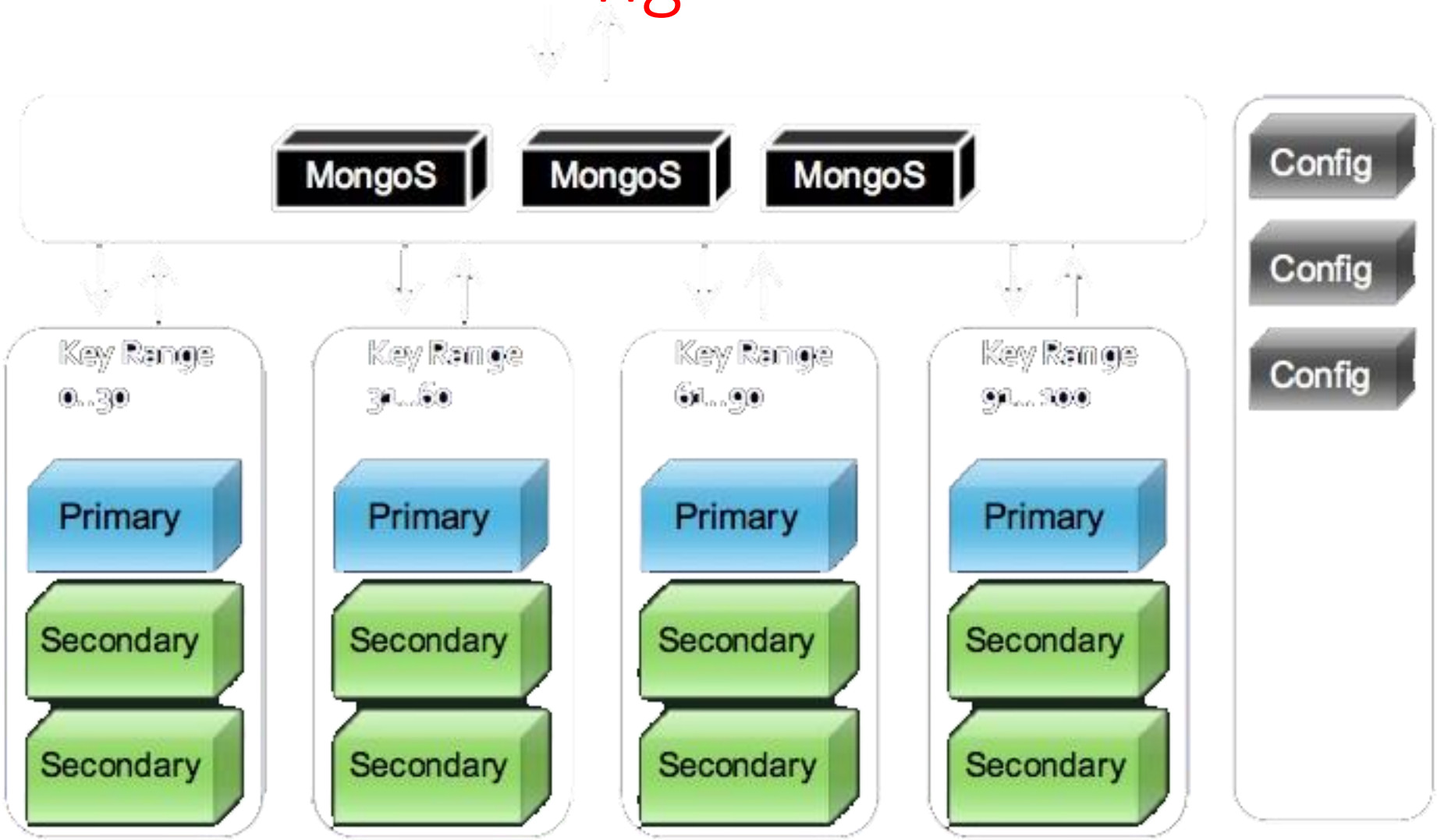
Deployment & Scaling



Replica Sets



Shardi ng


MongoS
MongoS
MongoS
Config
Config
Config
Key Range
0..30
Key Range
31..60
Key Range
61..90
Key Range
91..100
Primary
Secondary
Secondary
Primary
Secondary
Secondary
Primary
Secondary
Secondary
Primary
Secondary
Secondary