MONGODB 3.6 DEVELOPER WORKSHOP

Palo Alto, CA – January 25th, 2018

SPONSOR ANNOUNCEMENTS

O'REILLY CONFERENCES

Strata Data Conf March 2018

AI Conference April 2018

Velocity & Fluent Conf June 2018

OSCON July 2018

JupyterCon August 2018

AI Conference September 2018

Strata DATA CONFERENCE

March 6–8, 2018, San Jose, California

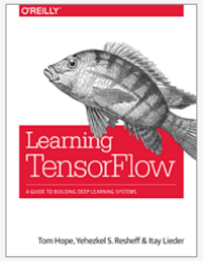
- All O'Reilly Conferences: http://bit.ly/ORConfs
- Use code **PCRAPHAEL** to get a 20% discount on your ticket
  - available for ANY O'Reilly 2018 conference, not just Strata

FREE O’REILLY BOOK OFFER

Choose one and download it at http://bit.ly/mugbooks

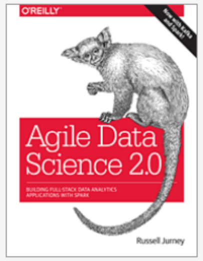
Data Science on the Google Cloud Platform

More Info >



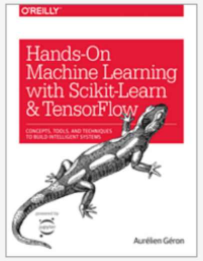
Learning TensorFlow

More Info >



Agile Data Science 2.0

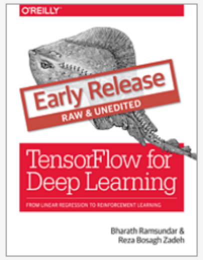
More Info >



Hands-On Machine Learning with Scikit-Learn and TensorFlow

More Info >

MUST READS!

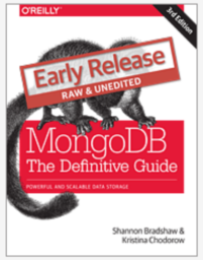

O'REILLY
Early Release
RAW & UNEDITED
TensorFlow for Deep Learning
Bharath Ramsundar & Reza Bosagh Zadeh

TensorFlow for Deep Learning : Early Release

More Info >

Designing Data-Intensive Applications

More Info >



MongoDB: The Definitive Guide : Early Release

More Info >

MongoDB Applied Design Patterns

More Info >

mongoDB.

MONGODB EVENTS

Coming up soon in 2018…

#MDBlocal

MONGODB.local

MONGODB.LOCAL SEATTLE 02/15/2018

HTTP://BIT.LY/MDBSEATTLE18

mongoDB

MONGODB WORLD'18

BUILD GIANT IDEAS

New York City, June 27-28

Use code **RaphaelLondner** for a 25% discount on the Super Duper Early Bird ticket ($199)

www.mongodb.com/world Twitter hashtag: #MDBW18

WORKSHOP DETAILS

LOGISTICS

Free Wifi: xgen_public
Password: mongodbatlas

Bathroom is on your right after the door and your right (again) at the green phone booth.

YOUR WORKSHOP HOSTS

Raphael Londner

Developer Advocate
MongoDB

@rlondner

Disclaimer

YOUR WORKSHOP HOSTS

Andrey Brindeyev

Technical Services Engineer
MongoDB

Disclaimer

YOUR WORKSHOP HOSTS

Justin LaBreck

Senior Consulting Engineer
MongoDB

Disclaimer

LET'S GET STARTED!

Workshop GitHub repository:

http://bit.ly/mdb36workshop

MongoDB Atlas Code: GOATLAS25

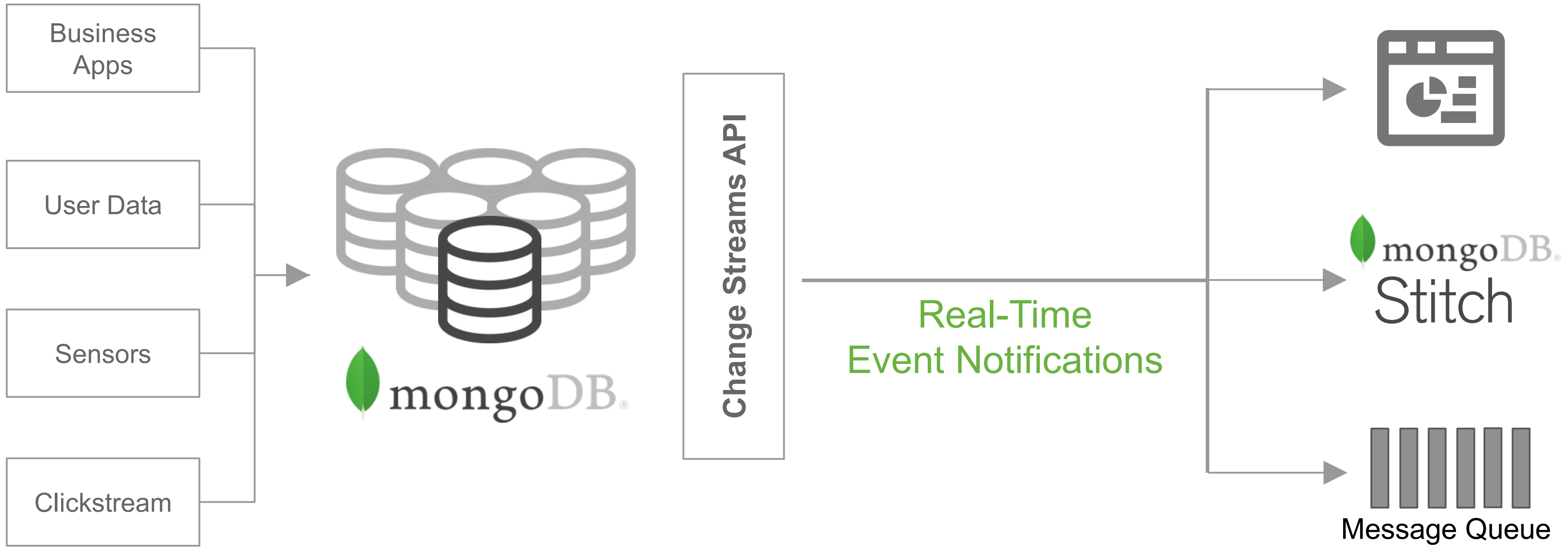
GETTING STARTED WITH MONGODB 3.6

- Download the latest release & review the Release Notes
- Read the What's New whitepaper
- Free M036 MongoDB University Training
- We can help: Major Version Upgrade Service

Speed to Develop

- Change Streams
- JSON Schema Validation
- Query Expressivity
- Fully Expressive Array Updates
- Retryable Writes
- Tunable Consistency

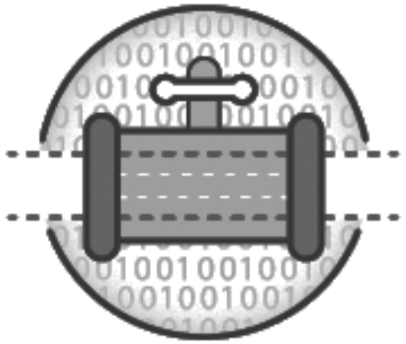
MONGODB CHANGE STREAMS



Enabling developers to build

reactive, real-time services

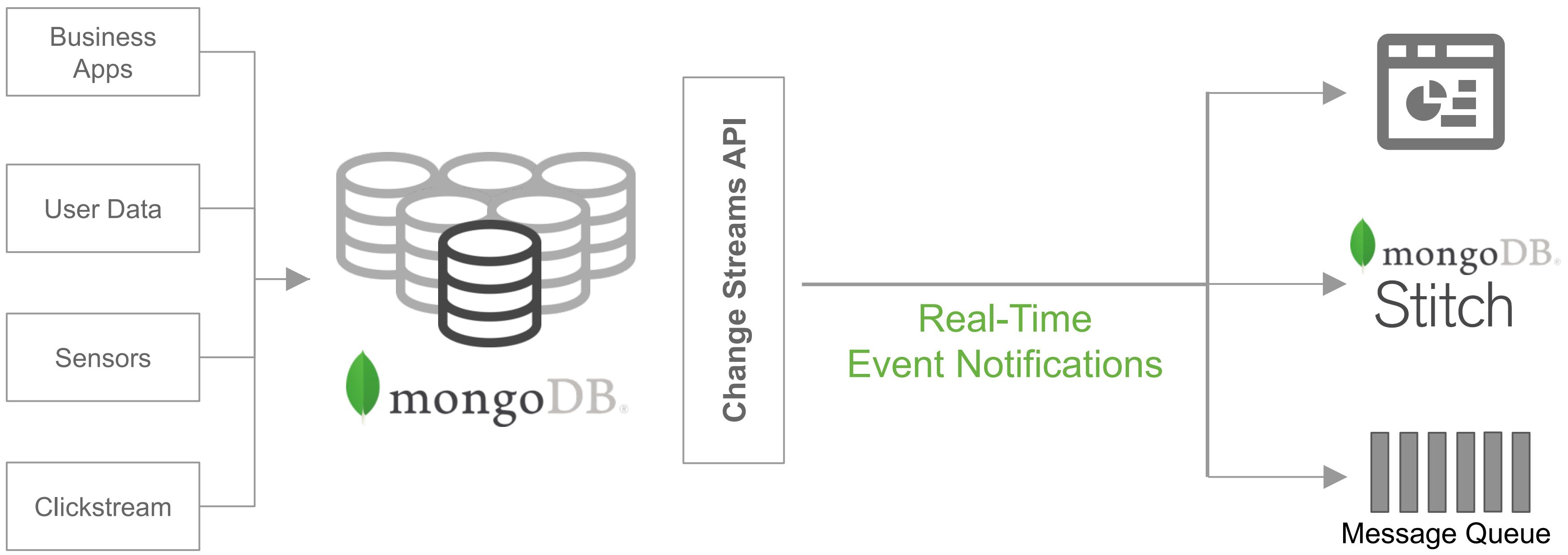
CHANGE STREAMS IMPLEMENTATION



Apps register for notifications via change streams API on top of MongoDB oplog

- Change streams are:
  - **Flexible:** deltas or the full document, filter on specific events only
  - **Consistent:** total ordering of data across shards
  - **Secure:** enforces collection's user access privileges
  - **Reliable:** only notifies once write committed on majority of replicas
  - **Resumable:** from node failure
  - **Concurrent:** up to 1,000 changes streams per MongoDB instance
  - **Familiar:** use regular MongoDB query language and drivers

CHANGE STREAMS USE CASES



- Refreshing trading apps as stock prices change
- Syncing changes across microservices
- Updating dashboards, analytics systems, search engines

- IoT data pipelines – e.g., generating alarms in response to connected asset failures
- Push new credit card transactions into ML models to recalculate risk
- Maintaining multiplayer game scoreboards

CHANGE STREAMS IN ACTION

```
// Select the collection to query.
 MongoCollection<Document> collection =
 database.getCollection("orders");

 // Create the change stream cursor.
  MongoCursor<Document> cursor =
 collection.watch().iterator();
```

SCHEMA VALIDATION IN ACTION

```
db.createCollection( "orders",
    {validator: {$jsonSchema:
        {
            properties:
            {line_items:
                {type: "array",
                    items:
                    {properties:
                        {title: {type: "string"},
                         price: {type: "number", minimum: 0.0} },
                        required: ["_id", "title", "price"],
                        additionalProperties: false}}},
            required: ["line_items"]}}}
)
```

SCHEMA VALIDATION

JSON Schema

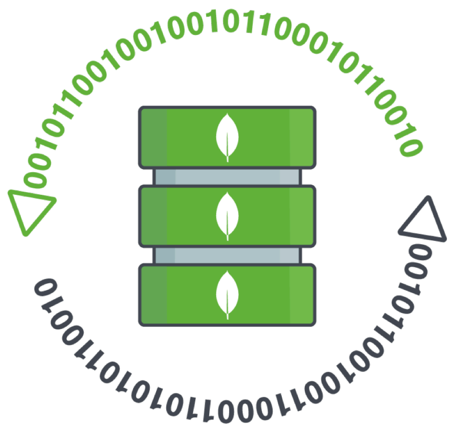
Enforces strict schema structure over a complete collection for data governance & quality

- Builds on document validation introduced by restricting new content that can be added to a document
- Enforces presence, type, and values for document content, including nested array
- Simplifies application logic

Tunable: enforce document structure, log warnings, or allow complete schema flexibility

Queryable: identify all existing documents that do not comply

MONGODB RETRYABLE WRITES



Write failure handling moved from the app to the database for transient network errors or primary elections

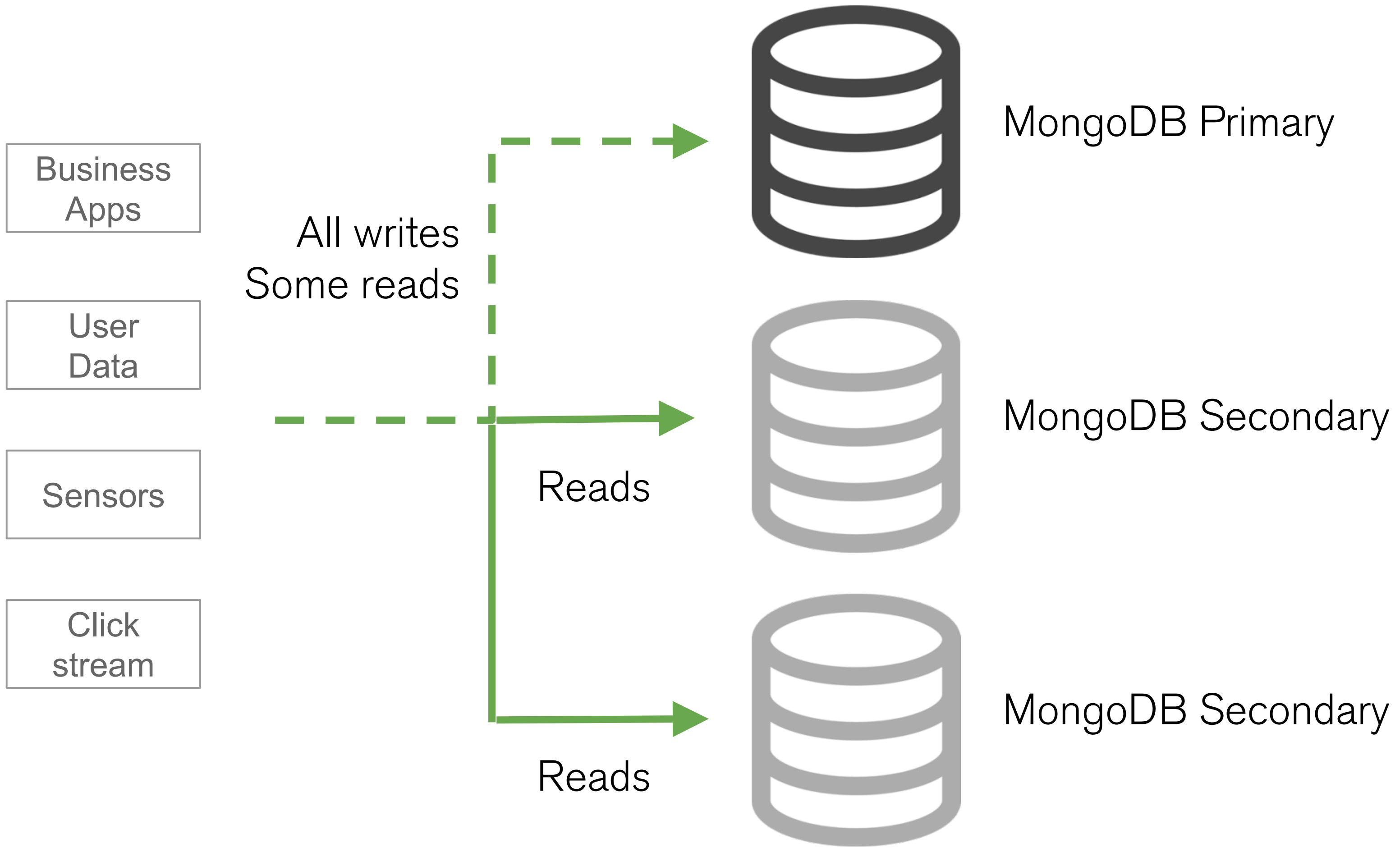
- Driver automatically retries failed write
- With a unique transaction identifier, server enforces exactly-once processing semantics

- Properties
  - Supports idempotent & non-idempotent operations, and errors caused by time-outs
- Delivers always-on, global availability of write operations
  - Overcomes the complexity imposed by multi-master, eventually consistent systems

RETRYABLE WRITES IN ACTION

```
uri = "mongodb://example.com:27017/?retryWrites=true"
client = MongoClient(uri)
database = client.database
collection = database.collection
```

TUNABLE CONSISTENCY: SCALING READS



TUNABLE CONSISTENCY CONTROLS

Balance data consistency with performance SLAs

- Developers have precise control over how queries are routed across the database cluster

- **Causal consistency:** guarantees monotonic, logically consistent reads from any replica node in the same user session
- **Sharded secondary reads:** Secondary replicas now chunk-aware, ensuring consistent reads even as data is being rebalanced across a sharded cluster

CAUSAL CONSISTENCY IN ACTION

```
//start client session, which is causally consistent by default
       try (ClientSession session =
             client.startSession(ClientSessionOptions.builder().build())) {

           //Run causally related operations within the session
           collection.insertOne(session, ... );
           collection.updateOne(session, ...);

           try (MongoCursor<Document> cursor =
                          collection.find(session).filter(...).iterator()) {
               while (cursor.hasNext()) {
                   Document cur = cursor.next();
               }
           }
```

QUERY EXPRESSIVITY & FULLY EXPRESSIVE ARRAY UPDATES

[ ]

- Use aggregation pipeline expressions within the MongoDB query language, using new `$expr` operator
  - SQL equivalent of `SELECT * FROM T1 WHERE a>b`
  - Example: find all customer accounts that have increased month on month spend by $200 or more
  - More expressive queries with less client-side code
- Atomically update multiple matching elements of an array in a single update command
  - Example: update all prices in an array by 20%
  - More flexible data modeling
  - Avoids document rewrites imposed by other databases

UPDATING ARRAYS: ALL ELEMENTS

```
orders:
{
  _id: 5,
  line_items : [
    { id: 123,
      title : "USB Battery",
      price: 15.0 },
    { id: 512,
      title : "Hip T-shirt",
      price : 45.0 }
  ],
  ...
}
```

```
db.orders.update(
  { _id: 5 },
  { $mul: {
      "line_items.$[].price":
      0.8
    }
  }
)
```

UPDATING ARRAYS: SOME ELEMENTS

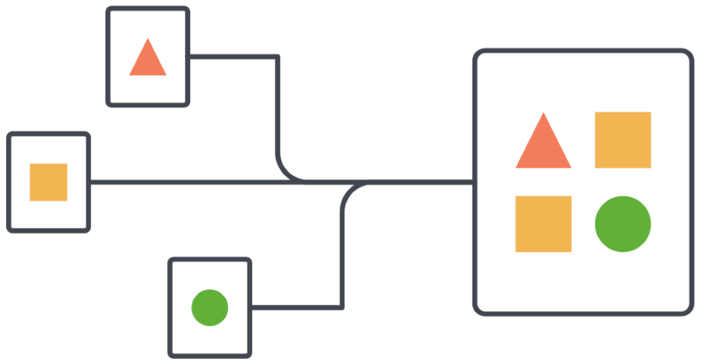
```
orders:
{
  _id: 5,
  line_items : [
    { id: 123,
      title : "USB Battery",
      price: 15.0 ,
      shipped: true},
    { id: 512,
      title : "Hip T-shirt",
      price : 45.0,
      shipped: false }
  ]
}
```

```
db.orders.update(
   { _id: 5 },
   { $mul: {

    "line_items.$[li].price":
    .8}},
    {arrayFilters:[

    {"li.shipped":{$ne:true}}
     ]}
)
```

RICHER AGGREGATION PIPELINE



- Expressive $lookup
  - Beyond Left Outer equi-join. Now supports non equi-joins & subqueries
  - Executed natively in the database, allowing more complex analytics queries with less code
- Timezone-aware aggregations
  - Enables multi-region analysis that are aware of region-specific timezones and working days when grouping data
- New expressions for richer transformations
  - Convert to and from objects to arrays of K-V pairs
  - Merge multiple objects into a single object
  - Remove fields from an object based on evaluation criteria

$LOOKUP IN 3.6

```
orders:
{
   ...
   line_items : [
     { id: 123,
       title : "USB Battery",
       price: 15.0 },
     { id: 512,
       title : "Hip T-shirt",
       price : 45.0 }
   ],
   ...
}
```

```
db.orders.aggregate([
    {$unwind: "$line_items"},
    {$lookup:{
        from: "reviews",
        let: {p_id:
"$line_items.id"},
        pipeline: [
        {$match: {$expr: {$eq:
["$p_id", "$$p_id"]}}},
        {$group: {
            _id: 1,
            rating: {$avg:"$rating"}
        }}
        ],
        as: "avgRating" }
    }
    ])
```

MONGODB 3.6

MOVE AT THE SPEED OF YOUR DATA

Speed to Develop

- Change Streams
- Retryable Writes
- Tunable Consistency
- Compass
- Query Expressivity & Fine-Grained Array Updates

Speed to Scale

- Ops Manager
- Schema Validation
- Extended Security Controls
- E2E Compression
- Multi-Tenancy Management

Speed to Insight

- BI Connector
- Richer Aggregation Pipeline
- R Driver