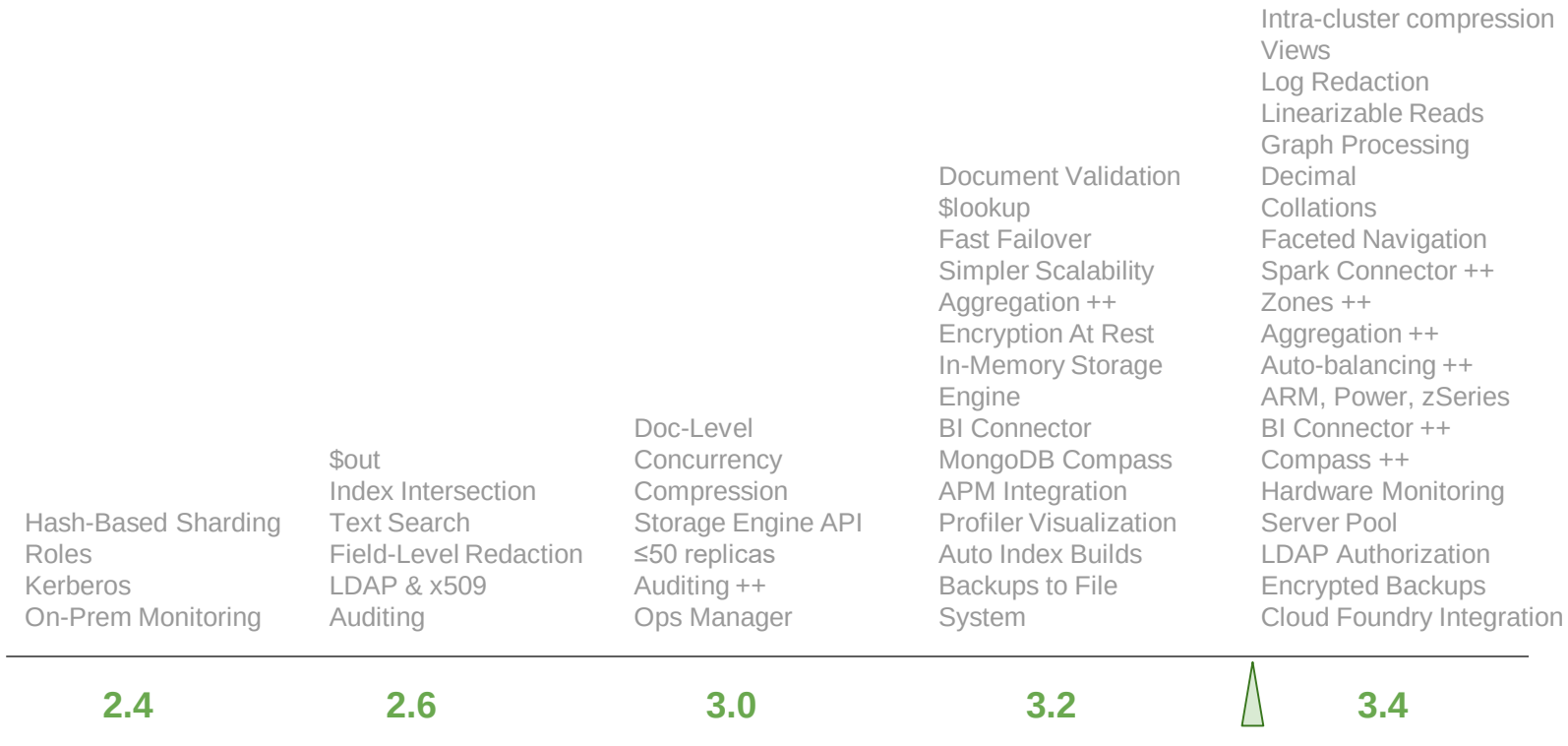
MongoDB 3.4

The Leading Non-Relational Database Evolved

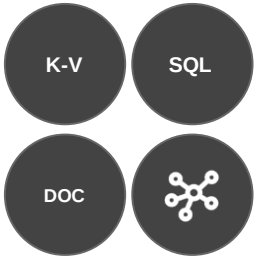
The Evolution of MongoDB



Headline Features by Release

mongoDB

MongoDB 3.4 Delivers



Multimodel Done Right

- Other vendors sell you multiple products
- MongoDB brings you multiple models in a single database

Mission Critical Apps

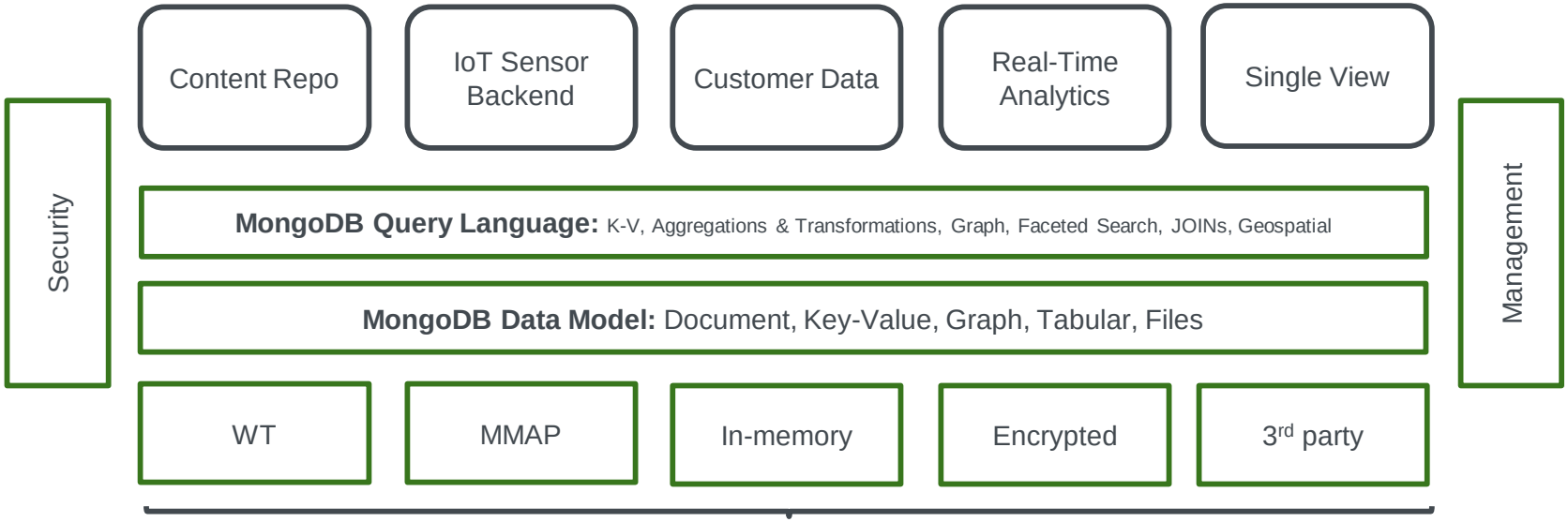
- Run larger clusters more efficiently, extend tunable consistency
- Extended security controls to address new threat classes

Modernized Tooling

- Native integration with enterprise standards
- Improve productivity of IT teams & data engineers

mongoDB

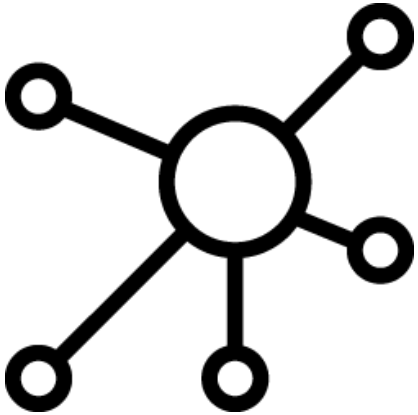
Multimodel Done Right



Available in MongoDB 3.4

mongoDB

Graph Processing



- Enables processing of graph & hierarchical data natively within MongoDB with $graphLookup operator
- Uncover indirect or transitive relationships in operational data
  - Recommendation engines, MDM, fraud models, social networks, etc.

Faceted Navigation

- Grouping data into related categories for intuitive exploration & discovery
  - Used in search and analytics applications
- New aggregation pipeline stages for faceting, bucketing & sorted counts across multiple dimensions
  - Eliminates requirement for external search engine

Collations

- Extend global reach of apps with collations, which allow proper text comparisons and sorting by applying language-specific rules
- MongoDB 3.4 adds support for 100+ different languages & locales throughout the query language and indexes
  - Over 2x as many as offered by most RDBMS

Decimal Data Type

Decimal128

- Support for the IEEE 754-2008 decimal128 type in server and drivers
  - Enables correct storage, comparing and sorting of decimal values
- Database stores exact values to eliminate rounding errors for high-precision calculations, complex financial & scientific apps

Advanced Analytics

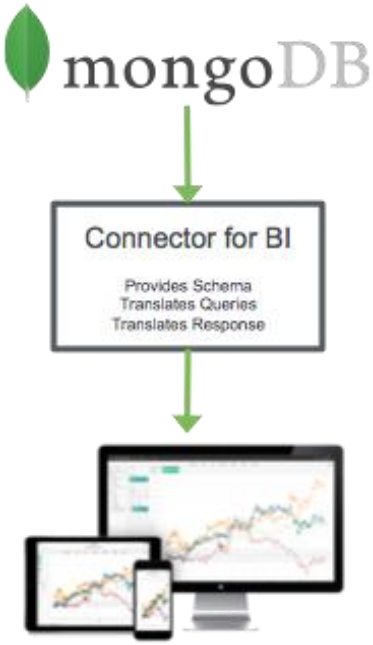
Aggregation Framework Enhancements

New Stages	Array Operators	String & Date Operators
`$graphLookup`	`$in`	`$indexOfBytes`
`$facet`	`$indexOfArray`	`$indexOfCP`
`$bucket`	`$range`	`$split`
`$bucketAuto`	`$reverseArray`	`$strLenBytes`
`$sortbyCount`	`$reduce`	`$strLenCP`
`$addFields`	`$zip`	`$substrBytes`
`$replaceRoot`		`$substrCP`
		`$isoDayOfWeek`
		`$isoWeek`
		`$isoWeekYear`

- Powerful data processing pipeline for analytics & transformations
- 25+ enhancements simplify app code
- Performance improvements with query optimizer moving `$match` stage earlier to use indexes

Advanced Analytics

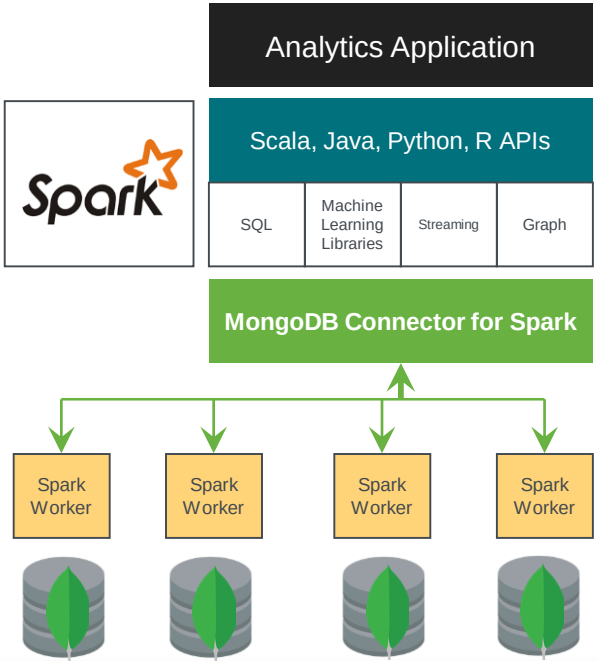
MongoDB Connector for BI



- Create powerful visualizations & analytics from SQL-based BI tooling
  - Auto-schema sampling
  - Eliminates ETL
- Higher performance with re-written SQL layer
  - More processing pushed down to the database
- Simplified installation and authentication

Advanced Analytics

MongoDB Connector for Apache Spark



- Native Scala connector, certified by Databricks
- Exposes all Spark APIs & libraries
- Efficient data filtering with predicate pushdown, secondary indexes, & in-database aggregations
- Locality awareness to reduce data movement
- Updated with Spark 2.0 support

"We reduced 100+ lines of integration code to just a single line after moving to the MongoDB Spark connector."
- Early Access Tester, Multi-National Banking Group

mongoDB

Mission-Critical Applications

Zones

Partition data across distributed clusters based on data locality policies

- Support distributed local writes
- Easily adhere to data sovereignty requirements
- Enable deployment patterns such as tiered storage

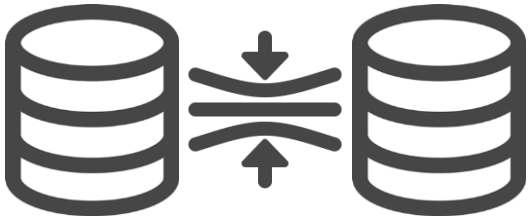
MongoDB Zones can now be configured visually from Ops Manager

Elastic Clusters

Seamless and elastic scalability in response to dynamic application demands

- Improved balancing of data across nodes with parallel data migrations
- Faster replica set synchronization with optimized initial sync process

Intra-Cluster Compression



With snappy compression algorithm, network traffic can be compressed by up to 70%

- Performance benefits in bandwidth-constrained environments
- Significantly reduce network costs

Improved Tunable Consistency

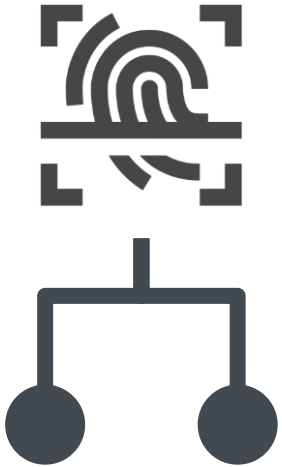
maxStalenessMS

- Choose how and when to route queries to secondary replicas
- Only read from replicas that are within a defined consistency window
- Improved data quality while scaling reads across secondaries

readConcern “linearizable” for the strongest consistency guarantees of any database

- Ensure that a node is the primary at the time of read
- Ensure that data returned will not be rolled back if another node is subsequently elected as primary

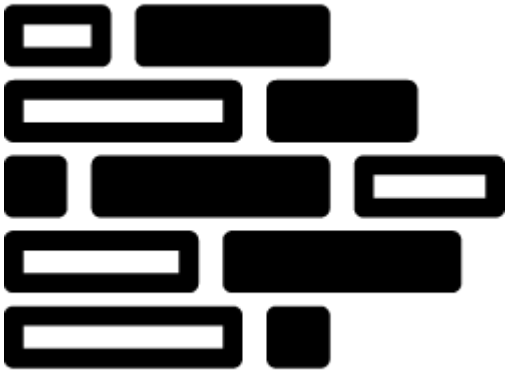
LDAP Authorization



LDAP authentication & authorization reduces administrative overhead & TCO

- User privileges can be managed centrally in LDAP and mapped to MongoDB roles without requiring duplication
- Native platform libraries to integrate with LDAP; no need for external dependencies and configurations; adds LDAP support for Windows

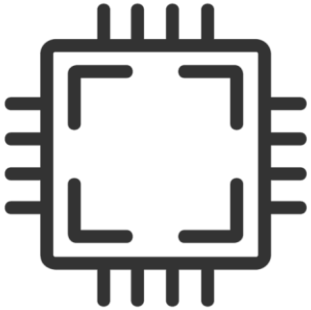
Read-Only Views



MongoDB 3.4 allows administrators to define dynamically generated views that expose a subset of data from the underlying collection

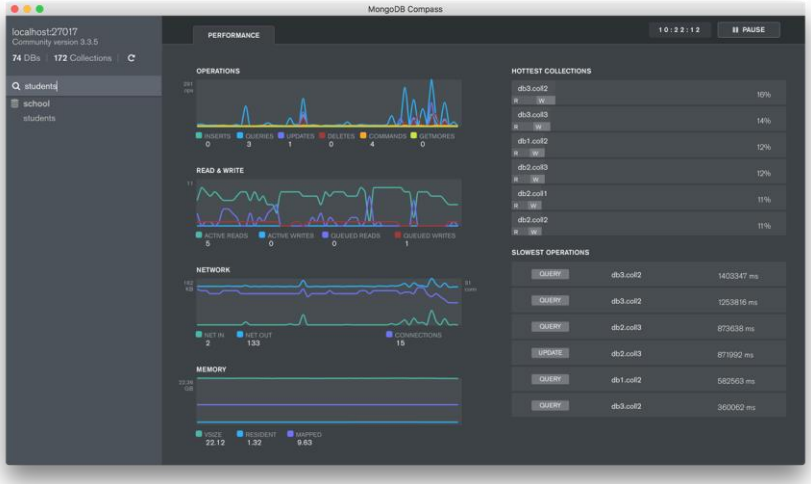
- Reduces risk of sensitive data exposure
- Views do not affect source collections
- Separately specified permissions levels
- Allows organizations to more easily meet compliance standards in regulated industries

Expanded Platform Support



MongoDB 3.4 supports the growing demand to run the database on a more diverse range of platforms

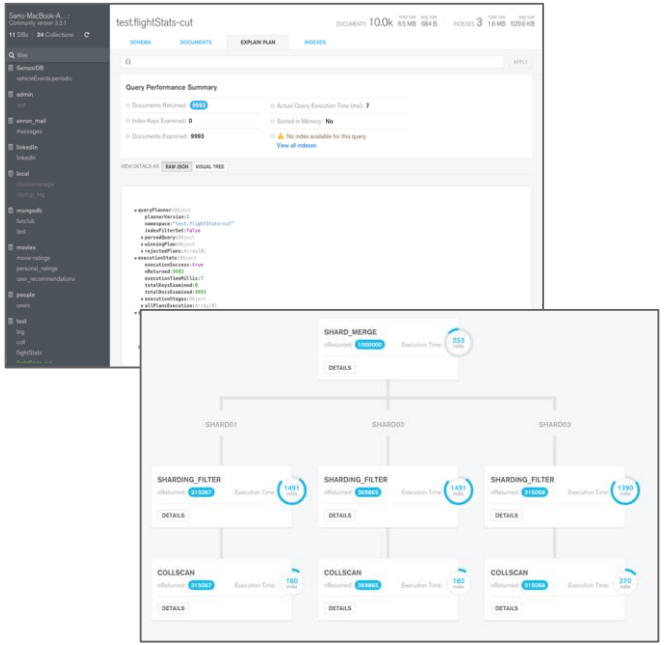
- **ARM v8-64 bit** support allows customers to take advantage of power-efficient servers being deployed into ultra dense data center racks
- **IBM Power8 and zSeries** support provides seamless migration for enterprises modernizing legacy workloads. Available for MongoDB Enterprise Server.



Modernized Tooling

mongoDB

MongoDB Compass

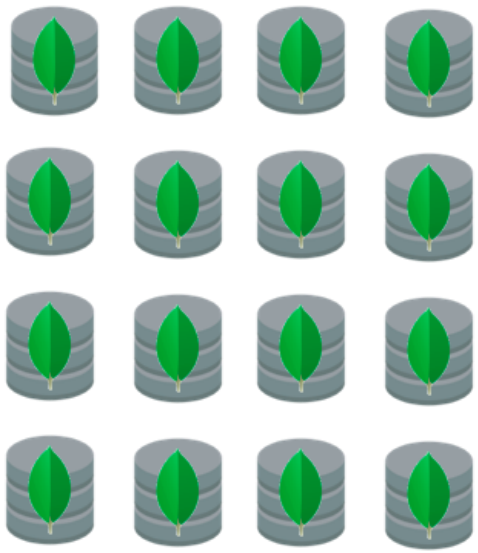


- Schema and query optimization
- MongoDB Compass enhancements
  - Modify documents
  - Create document validation rules
  - Optimize query performance with visual explain plans, index usage, and real-time statistics
- All controlled from a single intuitive and sophisticated GUI

mongoDB

Ops Manager

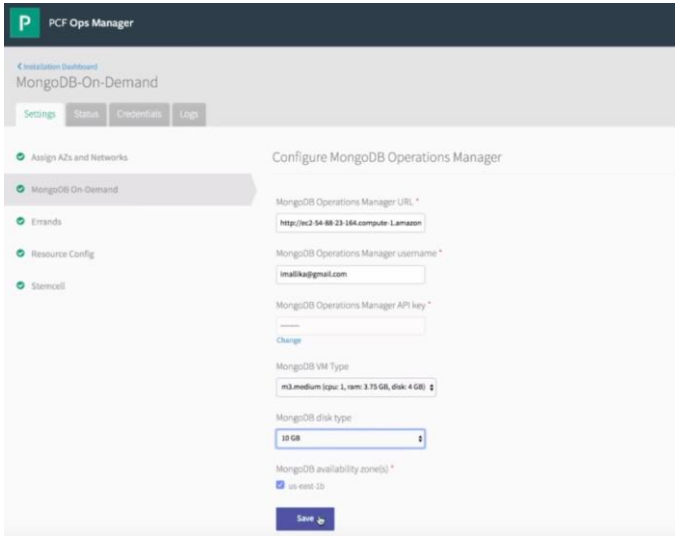
Server Pools



- Allocate and create pre-provisioned server pools
  - Ops Manager agent installed to pool via configuration management tools
- Server pools exposed to internal teams, ready for provisioning into local groups
- Allow administrators to create true, on demand database resources for private cloud environments

Ops Manager

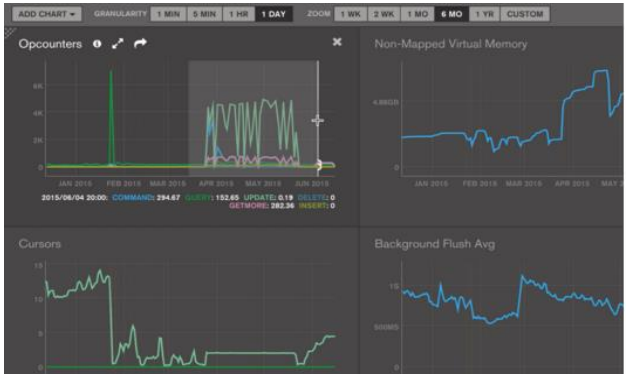
Cloud Foundry Integration



- MongoDB and Pivotal co-engineered Cloud Foundry integration
  - BOSH installs the Ops Manager agent
  - Calls the Ops Manager API to provision, configure, monitor & backup the cluster
- Enables users to integrate MongoDB as an on-demand DBaaS resource within Cloud Foundry platforms

Ops Manager

Higher Resolution Monitoring



- Finer grained telemetry data: collected every 10 seconds vs every 60 seconds
  - Configurable retention policies
- Simplified & extended management
  - Single agent to collect both database and hardware telemetry
  - Hardware metrics now collected for Windows & OSX hosts

Next Steps

- Read the What's New in MongoDB 3.4 Whitepaper
- Download the Release Candidate
- Register for the What's New in 3.4 Webinar

Summary

BBVA enjoys a position of leadership in the Spanish market, is the biggest financial institution in Mexico and has leading franchises in South America and the Sunbelt region in the United States.
With more than 50 million customers, each one having several accounts and cards, having a 360º view of the customer was a must for BBVA employees.
MongoDB provides the required performance and drives innovation offering a flexible platform for innovation and data analysis.

Company Background

BBVA is a global financial group founded in 1857. It has a customer-centric retail business model leveraged on innovation and technology. Customers are provided with a complete range of financial and non-financial products and services wherever the Group operates around the world.

Application Overview

BBVA branches requires an application that offered all information about a given customer and all his contracts (accounts, loans, cards, etc.)
This information will be used primary to offer new products and services based on customer profile.

That information originally lives in a mainframe environment, where reads are expensive due to the payment model (MIPS)

Once offloaded, BBVA had plans to use the information in other applications aligned with the bank spirit of become a digital bank

Why MongoDB?

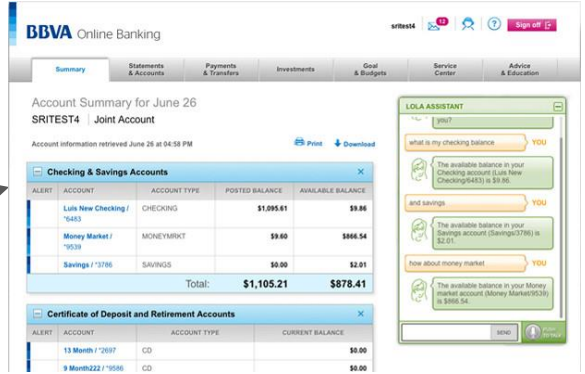
MongoDB flexible schema allows different types of movements (credit card, account, funds, etc.) to be stored in the same system and queried toghether.

Synchronisation from mainframe to MongoDB can happen during the alloted time thanks to the high write performance Wired Tiger offers.

MongoDB provides scale out capability to handle increased amount o data and increased amount of applications unsing the data.

Problem	Solution	Results
Took days to implement new functionality and business policies, inhibiting revenue growth	Built single view of customer on MongoDB – flexible and scalable app easy to adapt to new business needs	Cost avoidance of 10M$+
Branches needed an app providing single view of the customer and real time recommendations for new products and services	Super fast, ad hoc query capabilities (milliseconds), and real-time analytics thanks to MongoDB's Aggregation Framework	Application developed and deployed in less than 6 months. New business policies easily deployed and executed, bringing new revenue to the company
Multi-minute latency for accessing customer data stored in Teradata and Microstrategy	Can now leverage distributed infrastructure and commodity hardware for lower total cost of ownership and greater availability	Current capacity allows branches to load instantly all customer info in milliseconds, providing a great customer experience
	An access layer has been built to route the queries to MongoDB or Elastic Search depending on the query nature.	

BBVA Retail



Mainframe

Hourly Feeds

Deployment Details

Database

Total Data Volume: 15 TB
# of Documents: 7 billons
Growth: 1 billion documents / year
Queries per Second: 30.000

6 shards
Replica Set: P-S-S
MongoDB Version: 3.0.8
MongoDB Storage Engine(s): WT
OS: RHEL 6.4
Server Type:
- 256 GB RAM
- 4 cores
- 5x1.2GB 6G SAS 10K. RAID 10

Usage: Online Banking

The original usage of the solution was to provide BBVA employees a single view of customers.
Once the information was loaded in MongoDB they started to offer other services on top of it like online banking

Data in MongoDB is used to provide access to global position, historical information (currently the last 6 years) and personalized reporting. The users can query using any dimension of the data (amount, type, cathegory, dates, etc) or using free text search thanks to the presence of Elastic Search.

Usage: PFM

BBVA has been the first of the traditional Spanish banks to embrace the new fintech approach, leading the adoption of mobile apps.

Their Personal Finance Manager has been a great sucess, providing customers a better way to understand their finances, with agregated results by categories, saving goals, etc.

MongonDB aggregation framework is used to built the real time navigable reports provided by the tool.

Hands-on Session

Build a RESTful WS on top of MongoDB with Python Eve

17th September 2016

Session's goals

1. Python Eve installation and setup
2. MongoDB basic installation (WDTCS data set)
3. Expanding all to something more real (CRUD, HA, …)

Session's goals

1. Python Eve installation and setup
2. MongoDB basic installation (WDTCS data set)
3. Expanding all to something more real (CRUD, HA, …)

1. Python Eve installation and setup

- We'll go over the following steps:

1. Virtualenv setup for our API
2. Basic setup that we'll use along the session
3. RESTful WS test out with Postman

1.1 Virtualenv setup for our API

- It is as easy as:

```
Rauls-MacBook-Pro:~ rmarin$ mkdir pyday
Rauls-MacBook-Pro:~ rmarin$ virtualenv pyday
New python executable in /Users/rmarin/pyday/bin/python
Installing setuptools, pip, wheel...done.
Rauls-MacBook-Pro:~ rmarin$ source pyday/bin/activate
(pyday) Rauls-MacBook-Pro:~ rmarin$ pip install Eve

Collecting Eve
Collecting jinja2<3.0,>=2.7.2 (from Eve)
  Using cached Jinja2-2.8-py2.py3-none-any.whl
…
Successfully installed Eve-0.6.4 cerberus-0.9.2 events-0.2.1 flask-0.10.1 flask-
pymongo-0.4.1 itsdangerous-0.24 jinja2-2.8 markupsafe-0.23 pymongo-3.3.0 simplejson-3.8.2
werkzeug-0.11.3
```

1.2 Basic setup

- Let's create the base line for the next steps:

```
(pyday) Rauls-MacBook-Pro:~ rmarin$ cd pyday/
(pyday) Rauls-MacBook-Pro:pyday rmarin$ mkdir myapi
(pyday) Rauls-MacBook-Pro:pyday rmarin$ cd myapi/
(pyday) Rauls-MacBook-Pro:myapi rmarin$ vi app.py

from eve import Eve
app = Eve()

if __name__ == '__main__':
    app.run()

(pyday) Rauls-MacBook-Pro:myapi rmarin$ vi settings.py

DOMAIN = {'people': {}}
```

1.3 RESTful WS test out

- We're ready to go:

```
(pyday) Rauls-MacBook-Pro:myapi rmarin$ python app.py
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

- To test out our API will save us loads of hours and headaches:

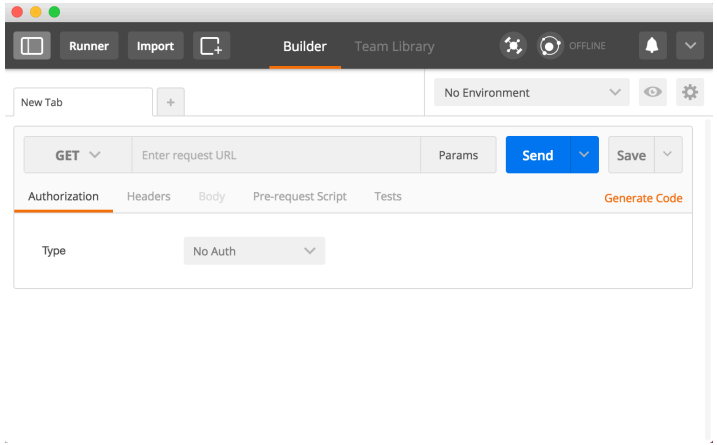
```
(pyday) Rauls-MacBook-Pro:myapi rmarin$  curl -i http://127.0.0.1:5000/
HTTP/1.0 200 OK
Content-Type: application/json
Content-Length: 60
Server: Eve/0.6.4 Werkzeug/0.11.3 Python/2.7.10
Date: Sat, 17 Sep 2016 07:26:54 GMT
```

1.3 RESTful WS test out (II)

- Python Eve (python-eve.org) is designed for human beings:

  let's test our RESTful API as human beings too

- Postman (www.getpostman.com) will help us in this matter:



Session's goals

1. Python Eve installation and setup
2. MongoDB basic installation (WDTCS data set)
3. Expanding all to something more real (CRUD, HA, …)

2. MongoDB basic installation

- We'll go over the following steps:

1. Spin up a standalone MongoDB instance
2. Restore the WDTCS data set

2.1 Spin up a MongoDB instance

- Download MongoDB 3.2.9 (.tgz/.zip) Community Server from:

  https://www.mongodb.com/download-center?jmp=nav#community

- Deploy it on your machine (ie. */opt/mongodb*):

```
(pyday) Rauls-MacBook-Pro:mongodb rmarin$ pwd
/opt/mongodb
(pyday) Rauls-MacBook-Pro:mongodb rmarin$ ls
GNU-AGPL-3.0            README                  bin
MPL-2                   THIRD-PARTY-NOTICES
```

2.1 Spin up a MongoDB instance (II)

- Create the folder structure for the MongoDB instance (ie. under the *pyday* folder):

```
(pyday) Rauls-MacBook-Pro:pyday rmarin$ mkdir -p rs/node1/data
(pyday) Rauls-MacBook-Pro:pyday rmarin$ cd rs/node1/
```

2.1 Spin up a MongoDB instance (III)

- Create the following configuration file under the previous folder:

```
(pyday) Rauls-MacBook-Pro:node1 rmarin$ vi mongod.conf
net:
  port: 27017
processManagement:
  fork: "true"
storage:
  dbPath: /Users/rmarin/pyday/rs/node1/data
  engine: wiredTiger
  journal:
    enabled: true
  wiredTiger:
    engineConfig:
      cacheSizeGB: 4
systemLog:
  destination: file
  logAppend: true
  path: /Users/rmarin/pyday/rs/node1/mongod.log
```

2.1 Spin up a MongoDB instance (IV)

- Start the MongoDB instance up:

```
(pyday) Rauls-MacBook-Pro:node1 rmarin$ /opt/mongodb/bin/mongod -f mongod.conf
about to fork child process, waiting until server is ready for connections.
forked process: 25240
child process started successfully, parent exiting
(pyday) Rauls-MacBook-Pro:node1 rmarin$ ls
data            mongod.conf     mongod.log
(pyday) Rauls-MacBook-Pro:node1 rmarin$ ls data
WiredTiger                              WiredTigerLAS.wt
diagnostic.data                         mongod.lock
WiredTiger.lock                         _mdb_catalog.wt
index-1--3316756531886542336.wt         sizeStorer.wt
WiredTiger.turtle                       collection-0--3316756531886542336.wt
index-3--3316756531886542336.wt         storage.bson
WiredTiger.wt                           collection-2--3316756531886542336.wt    journal
```

2.2 Restore the WDTCS data set

- Download the data set from the WDTCS's repo:

```
(pyday) Rauls-MacBook-Pro:node1 rmarin$ wget https://github.com/raulmarinperez/wdtcs/raw/
master/data/dump%202.zip
--2016-09-16 23:34:30--  https://github.com/raulmarinperez/wdtcs/raw/master/data/dump
%202.zip
…
HTTP request sent, awaiting response... 200 OK
Length: 19449335 (19M) [application/octet-stream]
Saving to: 'dump 2.zip'

dump 2.zip             100%[============================>]  18.55M  7.10MB/s   in 2.6s

2016-09-16 23:34:36 (7.10 MB/s) - 'dump 2.zip' saved [19449335/19449335]
```

2.2 Restore the WDTCS data set (I)

- Unzip the file containing the database dump:

```
(pyday) Rauls-MacBook-Pro:node1 rmarin$ unzip dump\ 2.zip
Archive:  dump 2.zip
   creating: dump/
   creating: dump/WDTCS/
  inflating: dump/WDTCS/containers.bson
  inflating: dump/WDTCS/containers.metadata.json
  inflating: dump/WDTCS/ports.bson
  inflating: dump/WDTCS/ports.metadata.json
  inflating: dump/WDTCS/ships.bson
  inflating: dump/WDTCS/ships.metadata.json
```

2.2 Restore the WDTCS data set (II)

- Restore the dump using the mongorestore tool:

```
(pyday) Rauls-MacBook-Pro:node1 rmarin$ mongorestore
2016-09-16T23:44:46.170+0200    using default 'dump' directory
2016-09-16T23:44:46.171+0200    preparing collections to restore from
…
2016-09-16T23:44:49.174+0200    [##########..............]  WDTCS.containers  70.7MB/164MB
(43.2%)
2016-09-16T23:44:52.174+0200    [####################...]  WDTCS.containers  149MB/164MB
(90.7%)
2016-09-16T23:44:52.717+0200    [########################]  WDTCS.containers  164MB/164MB
(100.0%)
2016-09-16T23:44:52.717+0200    restoring indexes for collection WDTCS.containers from
metadata

2016-09-16T23:45:01.372+0200    finished restoring WDTCS.containers (911249 documents)
2016-09-16T23:45:01.372+0200    done
```

2.2 Restore the WDTCS data set (III)

- Double check that the data set is available:

```
(pyday) Rauls-MacBook-Pro:node1 rmarin$ mongo
MongoDB shell version v3.3.11
connecting to: 127.0.0.1:27017/test

MongoDB server version: 3.3.11
> show dbs
WDTCS  0.033GB
local  0.000GB
> use WDTCS
switched to db WDTCS
> show collections
containers
ports
ships
```

Session's goals

1. Python Eve installation and setup
2. MongoDB basic installation (WDTCS data set)
3. Expanding all to something more real (CRUD, HA, …)

3. Expanding all to something more real

- We'll go over the following steps:

1. A more real settings.py
2. Address CRUD via the new RESTful API
3. Enable redundancy and data availability

3.1 A more real settings.py

```
MONGO_HOST = 'localhost'
MONGO_PORT = 27017
MONGO_DBNAME = 'WDTCS’

ports_schema = {
  '_id': {'type':'objectid'},
  'Ranking': {'type':'integer'},
  'Name': {'type':'string'},
  'Country': {'type':'string'}
}
ports = {
  'item_title': 'port',
  'schema': ports_schema
}
DOMAIN = {
  'ports': ports
}
```

3.2 Addressing CRUD via the new RESTful API

- List all ports (Read)

  http://127.0.0.1:5000/ports (GET)


```
{ "_updated": "Thu, 01 Jan 1970 00:00:00 GMT",
  "Ranking": 3,
  "Name": "Shenzhen",
  "Country": "China",
  "_links": { "self": { "href": "ports/56fda4e90a162d0f051f2ce7",
                        "title": "port" } },
  "_created": "Thu, 01 Jan 1970 00:00:00 GMT",
  "_id": "56fda4e90a162d0f051f2ce7",
  "_etag": "2ebc7ab98b18a263b7babe46b9de2eab0751f980"
}
```

3.2 Addressing CRUD via the new RESTful API (I)

- List Barcelona's port (Read):

  http://127.0.0.1:5000/ports/56fda4e90a162d0f051f2d33

- List Spanish ports (Read):

  http://127.0.0.1:5000/ports?where={"Country": "Spain"}

  We only consider Algeciras (30), Valencia (31) and Barcelona (77) so far

3.2 Addressing CRUD via the new RESTful API (II)

- Smart and elegant design for data integrity and concurrency control:

  http://python-eve.org/features.html#data-integrity-and-concurrency-control

  “Consumers are not allowed to edit (PATCH or PUT) or delete (DELETE) a resource unless they provide an up-to-date ETag for the resource they are attempting to edit. This prevents overwriting items with obsolete versions”

3.2 Addressing CRUD via the new RESTful API (III)

- Wouldn't be nice if Barcelona's port's ranking were 1? (Update):

  URL: http://127.0.0.1:5000/ports/56fda4e90a162d0f051f2d33 (POST)
  Content-type: *Body➔Raw➔JSON(application/json)*
  Content: *{ "Ranking": 1 }*

- This query rises an error:

```
{ "_status": "ERR",
  "_error": {
    "message": "The method is not allowed for the requested URL.",
    "code": 405
  } }
```

3.2 Addressing CRUD via the new RESTful API (IV)

- By default Python Eve provides a read-only RESTful API
- Enable writes by adding the following entry in the settings.py file:

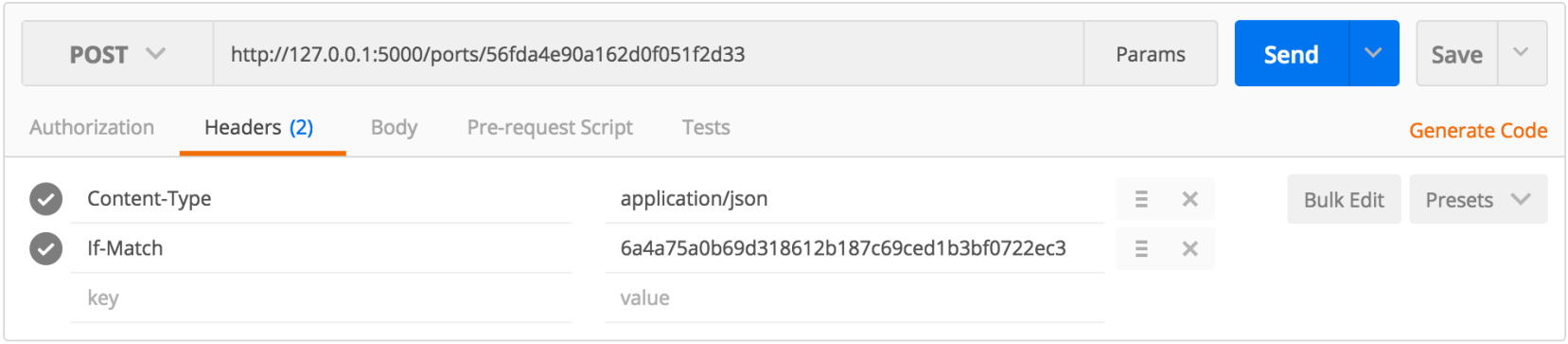
```
RESOURCE_METHODS = ['GET', 'POST', 'DELETE']
ITEM_METHODS = ['GET','PATCH','PUT','DELETE']
```

- Still rises an error:

```
{ "_status": "ERR",
  "_error": {
    "message": "An etag must be provided to edit a document",
    "code": 403
  } }
```

3.2 Addressing CRUD via the new RESTful API (V)

- You need to provide the _etag as a header entry (If-Match):



- Double check that Barcelona's port is now the best port in Spain:

http://127.0.0.1:5000/ports?where={"Country": "Spain"}&sort=[("Ranking", 1)]

3.2 Addressing CRUD via the new RESTful API (VI)

- Hey! I live in Málaga and I want Málaga's port to be in.
- Let's add a new port within the ports collection (Create):

  URL: http://127.0.0.1:5000/ports (POST)
  Content-type: *Body➔Raw➔JSON(application/json)*
  Content: `{ "Ranking": 100, "Name": "Malaga", "Country": "Spain" }`
- Let's do it for my partner in crime living in Madrid too (sounds weird though):

  URL: http://127.0.0.1:5000/ports (POST)
  Content-type: *Body➔Raw➔JSON(application/json)*
  Content: `{ "Ranking": 101, "Name": "Madrid", "Country": "Spain" }`

3.2 Addressing CRUD via the new RESTful API (VII)

- Wasn't a good idea to add "Madrid's port"
- Would be weird to see ships across rivers reaching out "El Manzanares"
- Let's remove the port from the ports collection (Delete):

URL: http://127.0.0.1:5000/ports/57dd0e7ab04cc5656c1ea0cf (DELETE)
Header: `If-Match: 383e5ae9ca3e52ccd2c76b9c4dcd79a7afecb78e`

3.3 Enabling redundancy and data availability

- So far so good! But what happens if MongoDB goes down (stop it)?

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<title>500 Internal Server Error</title>
<h1>Internal Server Error</h1>
<p>The server encountered an internal error and was unable to complete your request.  Either
the server is overloaded or there is an error in the application.</p>
```

- We need to go a step further ➔ to enable a MongoDB Replica Set
- Amend the following lines to the configuration file:

```
replication:
  oplogSizeMB: 100
  replSetName: wdtcs
```

3.3 Enabling redundancy and data availability (I)

- Let's expand the folders' structure to two additional nodes:

```
(pyday) Rauls-MacBook-Pro:pyday rmarin$ mkdir -p rs/node2/data
(pyday) Rauls-MacBook-Pro:pyday rmarin$ mkdir -p rs/node3/data
```

- Clone node1's configuration file for each new instance and adapt it:

```
net:
  port: 27018 (27019)
storage:
  dbPath: /Users/rmarin/pyday/rs/node2/data (node3)
  …
systemLog:
  path: /Users/rmarin/pyday/rs/node2/mongod.log (node3)
  …
```

3.3 Enabling redundancy and data availability (II)

- Start up all the MongoDB instances
- Connect to node1 and initiate the replica set:

```
(pyday) Rauls-MacBook-Pro:node1 rmarin$ mongo
MongoDB shell version v3.3.11
connecting to: 127.0.0.1:27017/test
MongoDB server version: 3.3.11
> rs.initiate()
{
        "info2" : "no configuration specified. Using a default configuration for the set",
        "me" : "Rauls-MacBook-Pro.local:27017",
        "ok" : 1
}
wdtcs:PRIMARY> rs.add("Rauls-MacBook-Pro.local:27018")
{ "ok" : 1 }
wdtcs:PRIMARY> rs.add("Rauls-MacBook-Pro.local:27019")
{ "ok" : 1 }
```

3.3 Enabling redundancy and data availability (III)

- The RESTful API should work again ➔ list all ports:

  http://127.0.0.1:5000/ports

- Take node1 down and try again ➔ there's still a primary but it's failing
- Our API is still bound to node1 ➔ change the following in settings.py:

```
MONGO_URI = 'mongodb://Rauls-MacBook-Pro.local:27017,Rauls-MacBook-Pro.local:27018,Rauls-MacBook-Pro.local:27019/WDTCS?replicaSet=wdtcs'
#MONGO_HOST = 'localhost'
#MONGO_PORT = 27017
#MONGO_DBNAME = 'WDTCS'
```

3.3 Enabling redundancy and data availability (IV)

- Take node2 down and try again ➔ there's still a secondary but it's failing
- By default pymongo only reads from primaries ➔ update the URI to:

```
MONGO_URI = 'mongodb://Rauls-MacBook-Pro.local:27017,Rauls-MacBook-Pro.local:27018,Rauls-
MacBook-Pro.local:27019/WDTCS?replicaSet=wdtcs&readPreference=primaryPreferred'
```

Questions?

Thank You!

```
{
  name        : "Rubén Terceño",
  title       : "Senior Solutions Architect",
  mail        : "ruben@mongodb.com",
  location    : "Madrid, ES”
}

{
  name        : "Raúl Marín",
  title       : "Senior Consulting Engineer",
  mail        : "raul.marin-perez@mongodb.com",
  location    : ”Málaga, ES”
}
```

mongoDB

Operational Excellence

A guided walkthrough on Ops Manager

MONGODB EUROPE16

FOR GIANT IDEAS

Rubén Terceño

@rubenTerceno
ruben@mongodb.com

Senior Solutions Architect, MongoDB

Ops Manager vs Cloud Manager

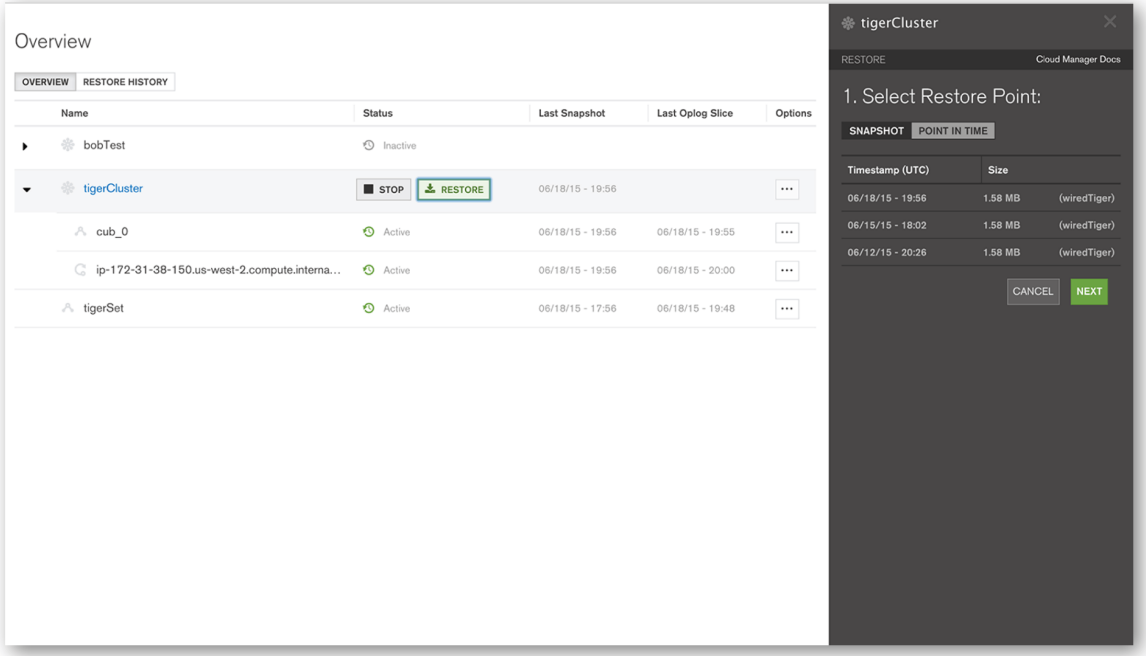
The Best Way to Run MongoDB

Ops Manager allows you *leverage and automate* the best practices we've learned from thousands of deployments in a comprehensive application that helps you run MongoDB safely and reliably.

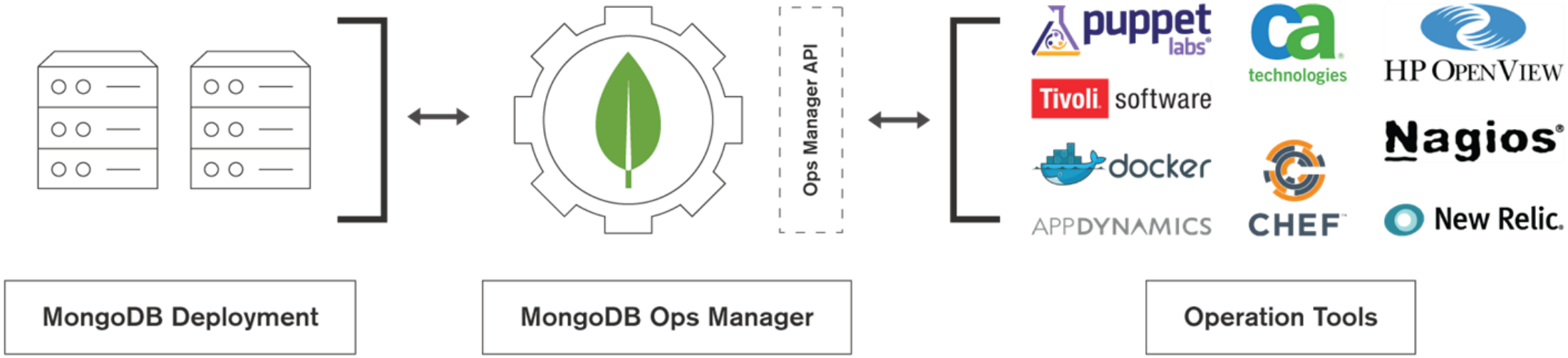
- 10x-20x more efficient operations
- Complete performance visibility
- Protection from data loss
- Performance optimization

Up to 95% Reduction in Operational Overhead

- Start from a global view of infrastructure: Integrations with Application Performance Monitoring platforms.
- Drill down: Visual query performance diagnostics, index recommendations. Dashboards and alerts on 100+ metrics.
- Then, deploy: Automated index builds. Single-click deployment, scaling & upgrades, admin tasks.
- Peace of mind: Backup and restore, with point-in-time recovery, support for sharded clusters.
- Refine: Partial indexes improve resource utilization.



REST API Allows Integration with Existing Infrastructure



How Ops Manager Helps You

Best Practices, Automated

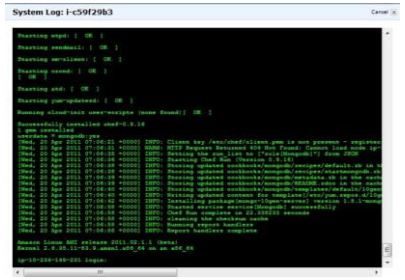
Cut Management Overhead

Meet SLAs

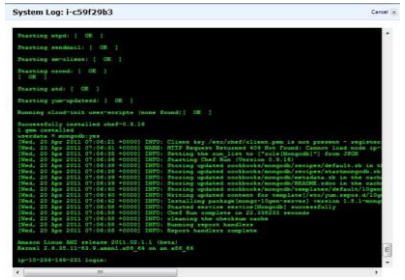
Scale Easily

Automation for Operational Excellence

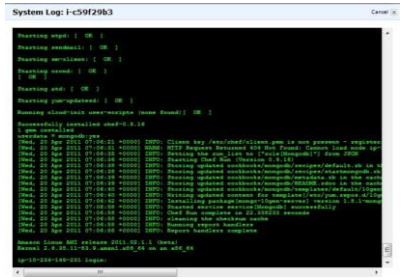
Without Ops Manager



- Install, Configure
  - 150+ steps

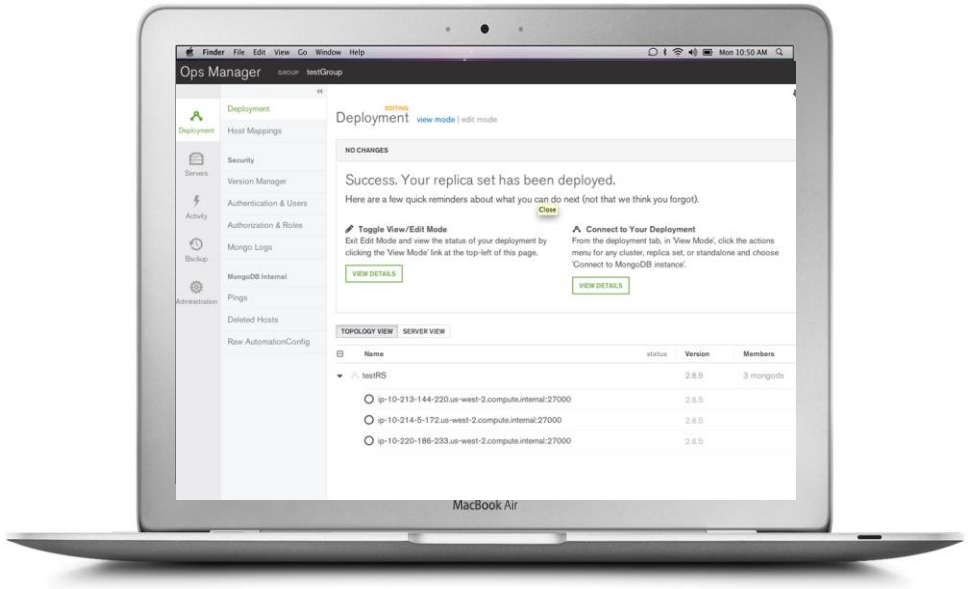


- Upgrades, downgrades
  - 100+ steps



- Scale out, move servers, resize oplog, etc.
  - 10-180+ steps

With Ops Manager



Ops Mgr (app)

or

Ops Manager API

Operations Tools

Ops Mgr (API)

150+ steps → 3 clicks

Easy

- Deployments. Including attaching to existing deployments.
- Hot Upgrades. Rolling, in the right order.
- Scale. Add/remove shards or replicas, with no downtime.
- Config and Management. Specify users, roles, Resize Oplog, etc.
- Automated Rolling Index Build. Reduce operational overhead risk.

Backup for Your Mission-Critical Data

Backup

- Point-In-Time Recovery.
  - Restore the correct version of your data with single operation granularity.
- Continuous, Incremental Backups.
  - Save backup storage while reducing your RPO to 0.
- Support for all MongoDB storage engines.
  - WT, MMAP, In Memory, Encrypted, …
- Automated restore process.
  - Minimize RTO and associated risks.
- Coherent backup of sharded clusters.
  - Configs, shards, replicas… No more headaches.

Monitoring Designed for MongoDB

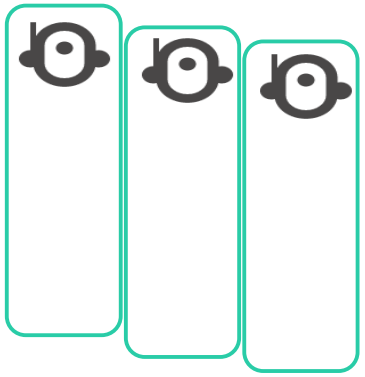
Monitoring

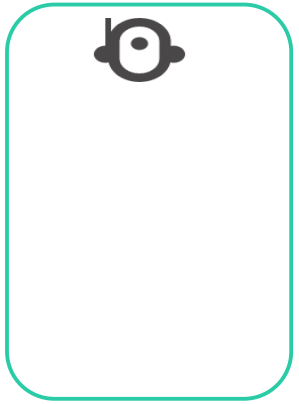
- Visibility Into Your Cluster. 100+ key database and systems health metrics
- Optimize performance, diagnose problems
- Custom, Metric-Based Alerting.
- Comprehensive API.
- Visual Query Profiler & Index suggestions.
  - Identifies the slow running queries with a click and generates index improvements.

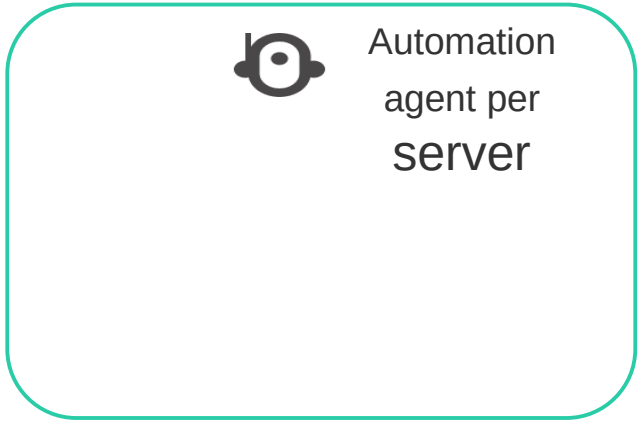
How it works

How Automation Works

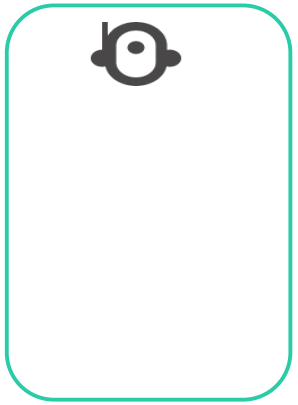
Ops Manager

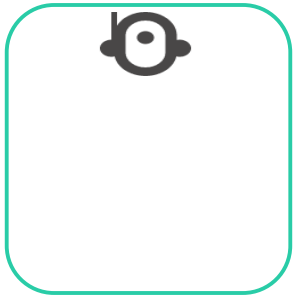




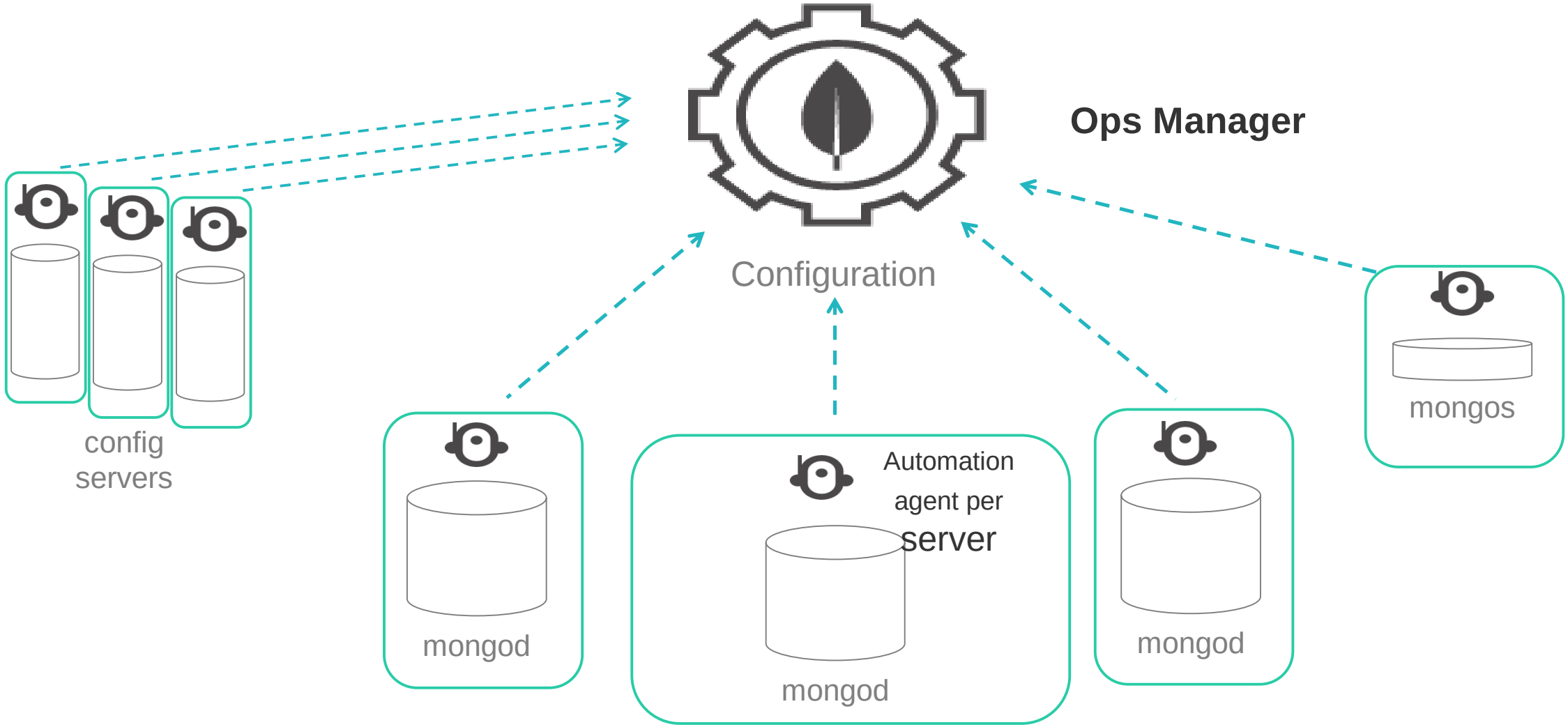


Automation agent per server

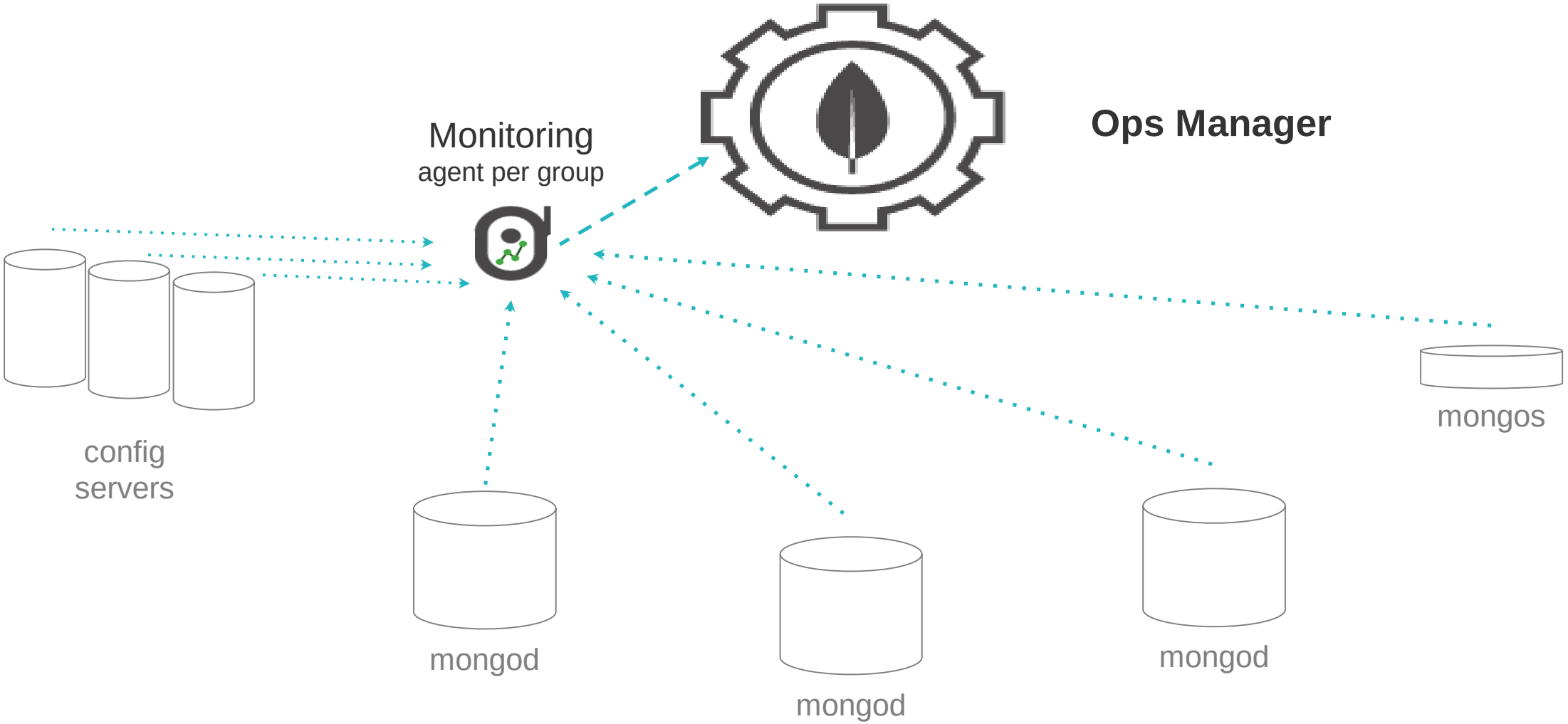




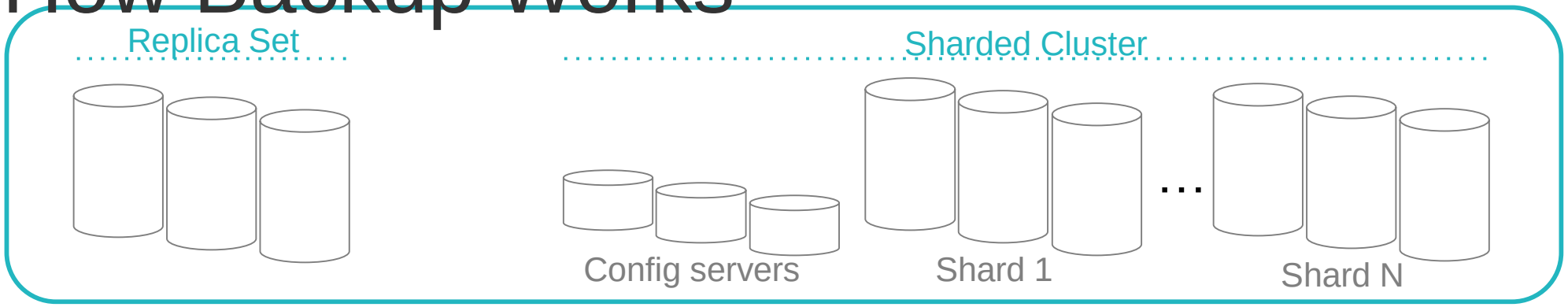
How Automation Works



How Monitoring Works

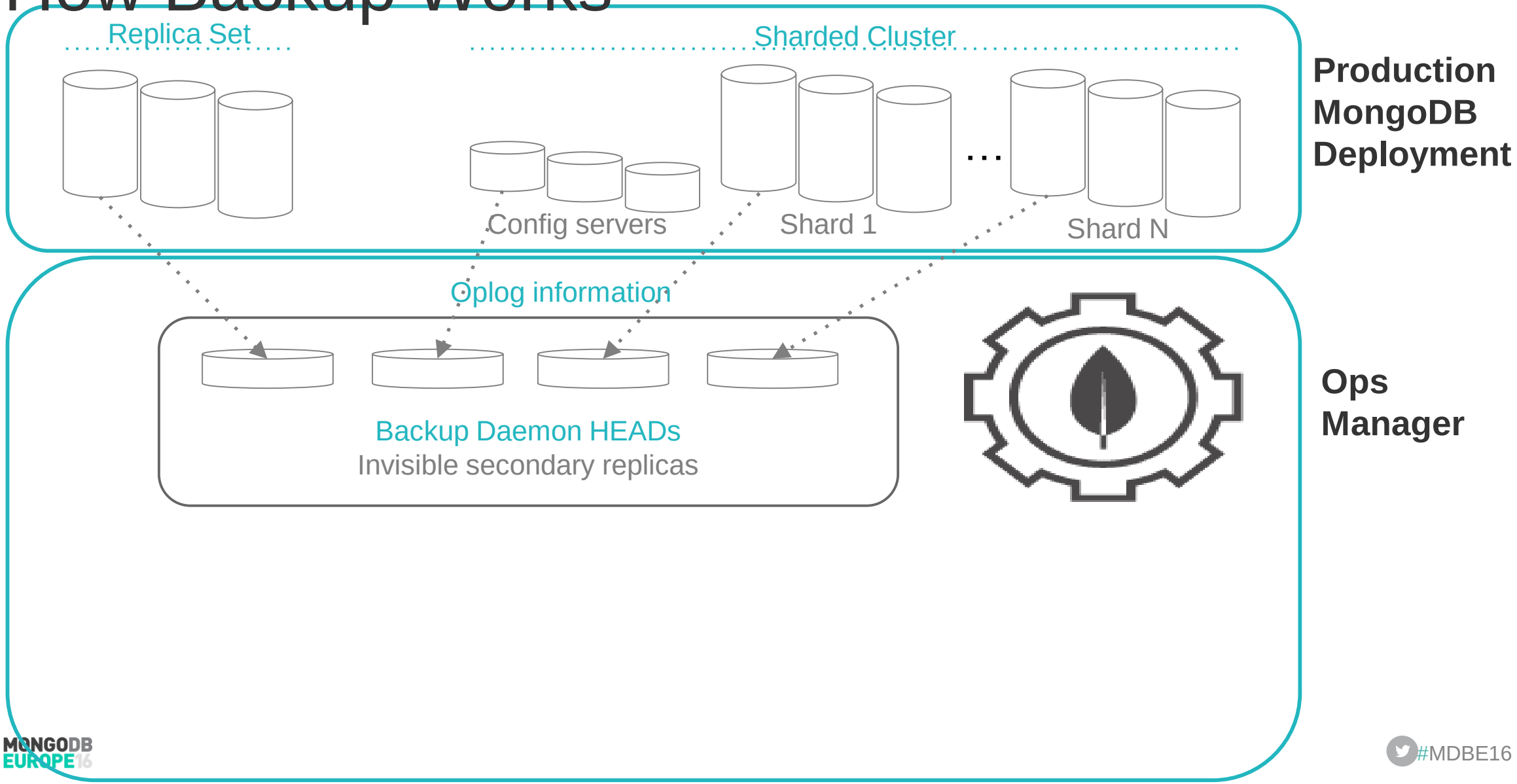


How Backup Works

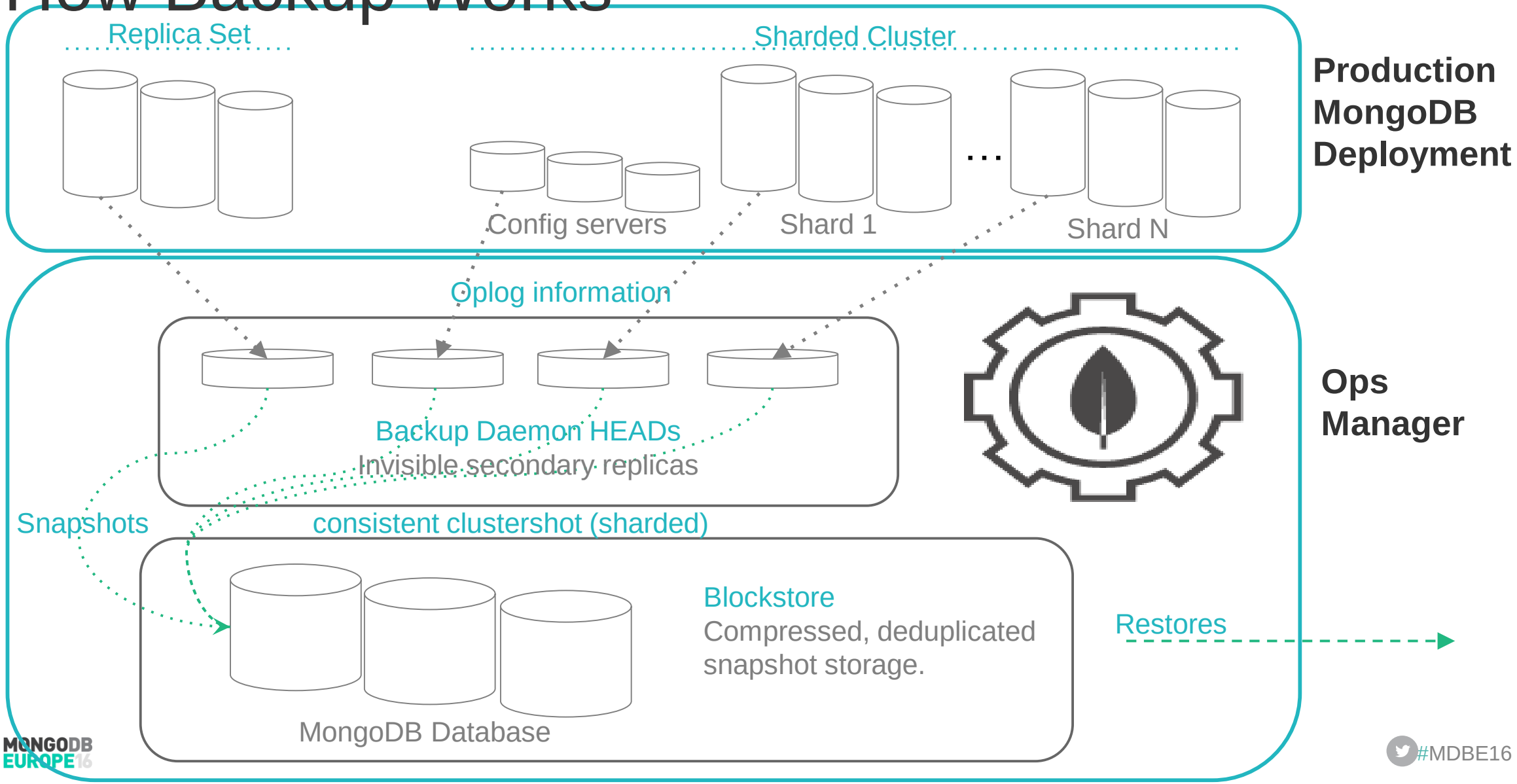


Production MongoDB Deployment

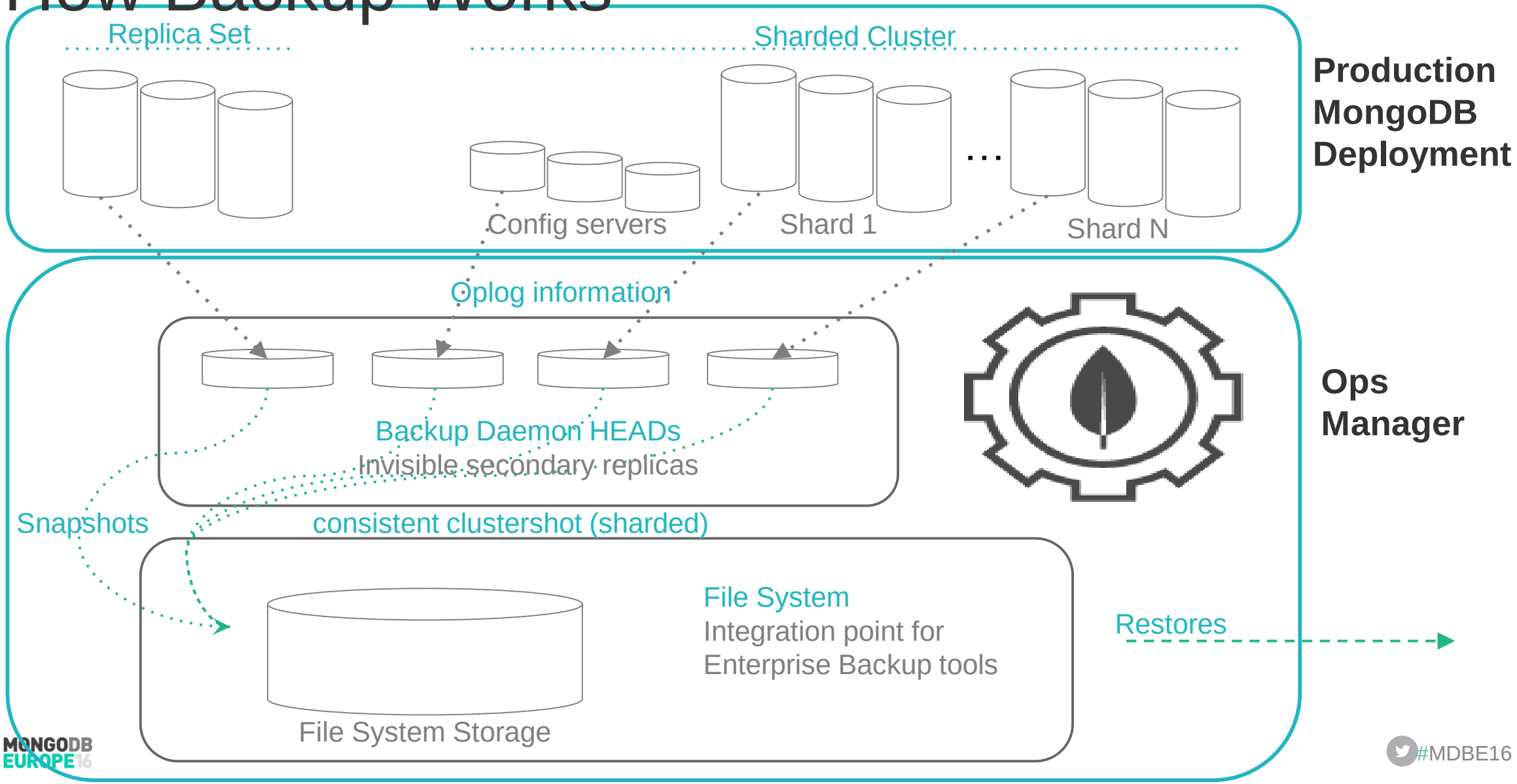
How Backup Works



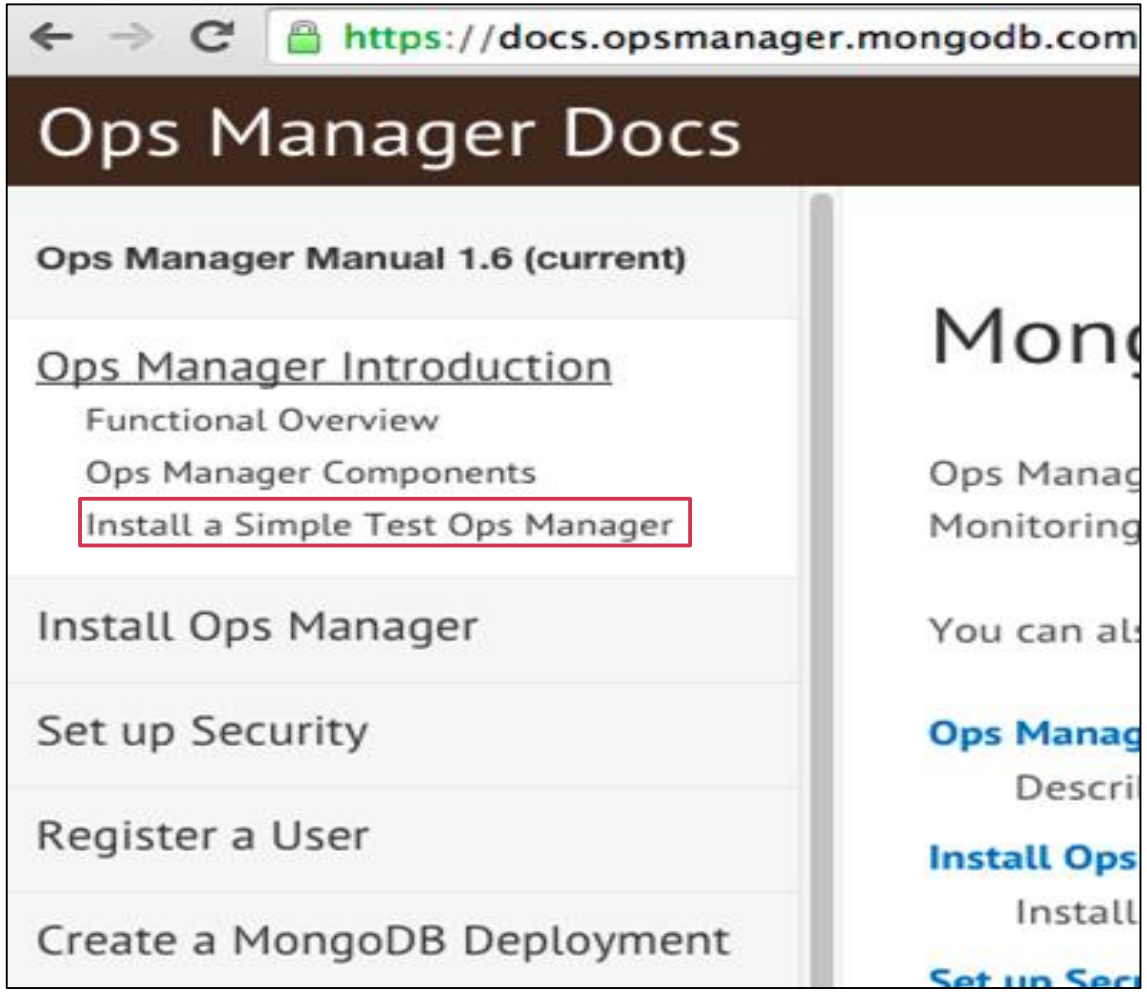
How Backup Works



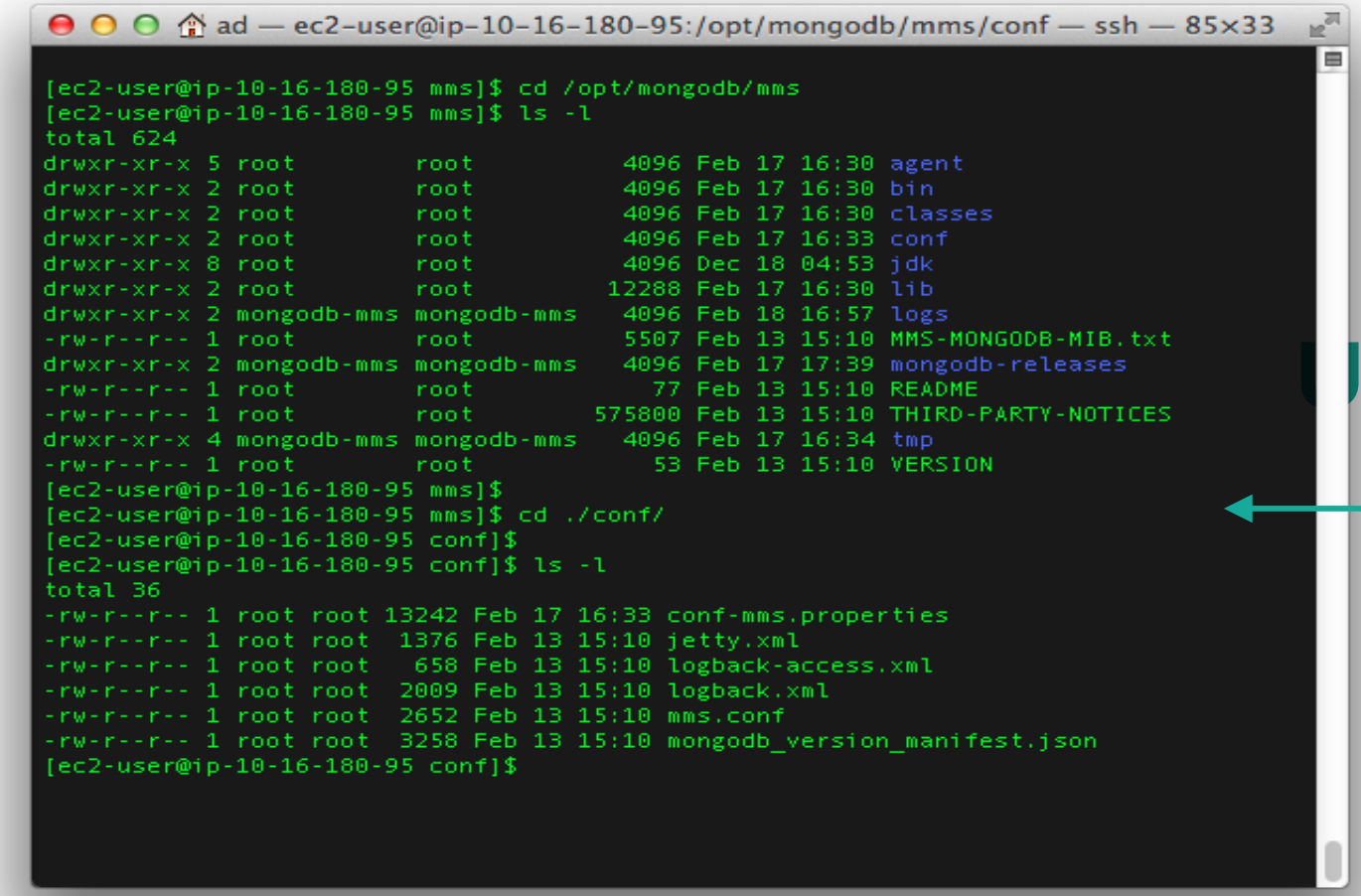
How Backup Works



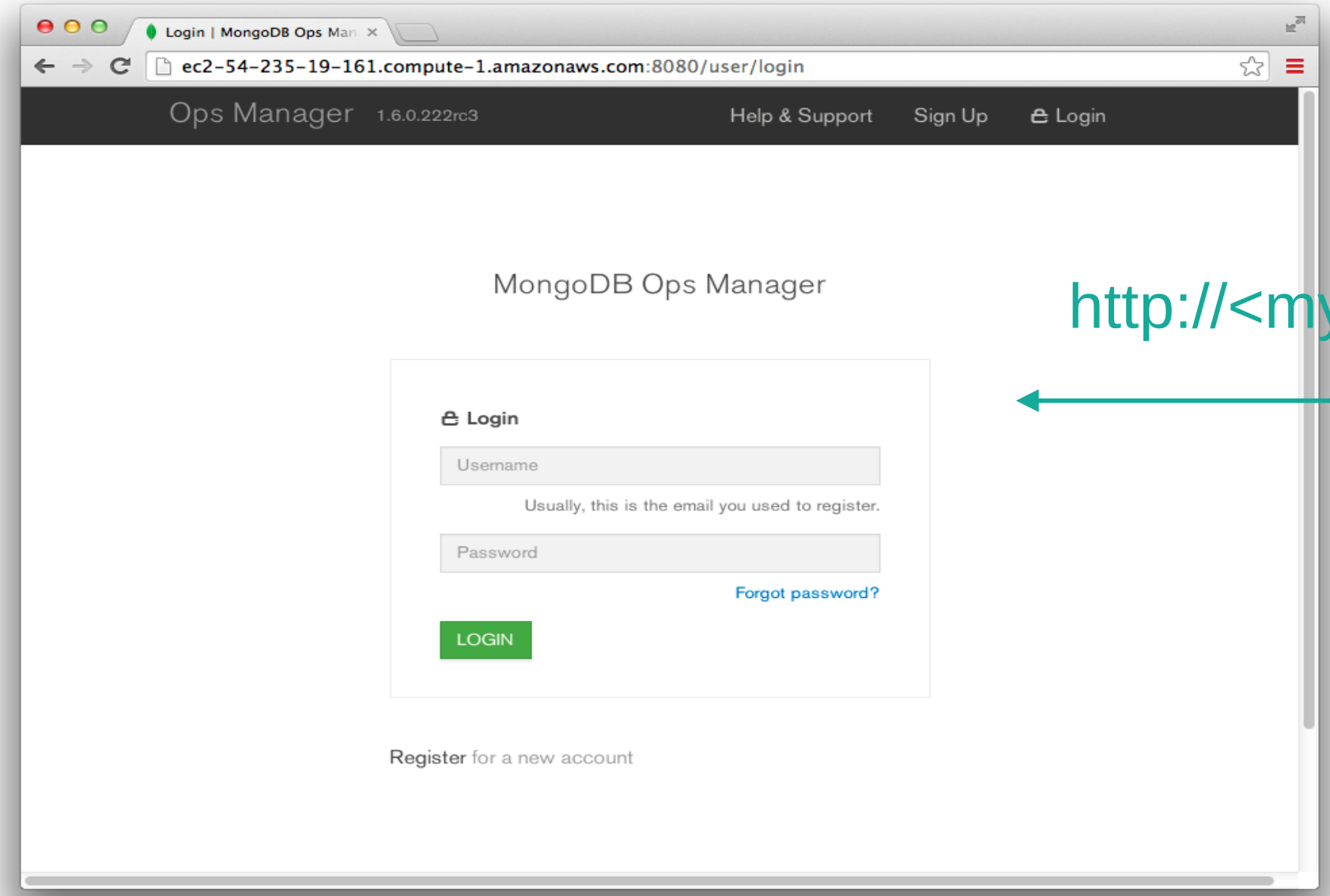
Usability Tour



Find installation tutorial in docs

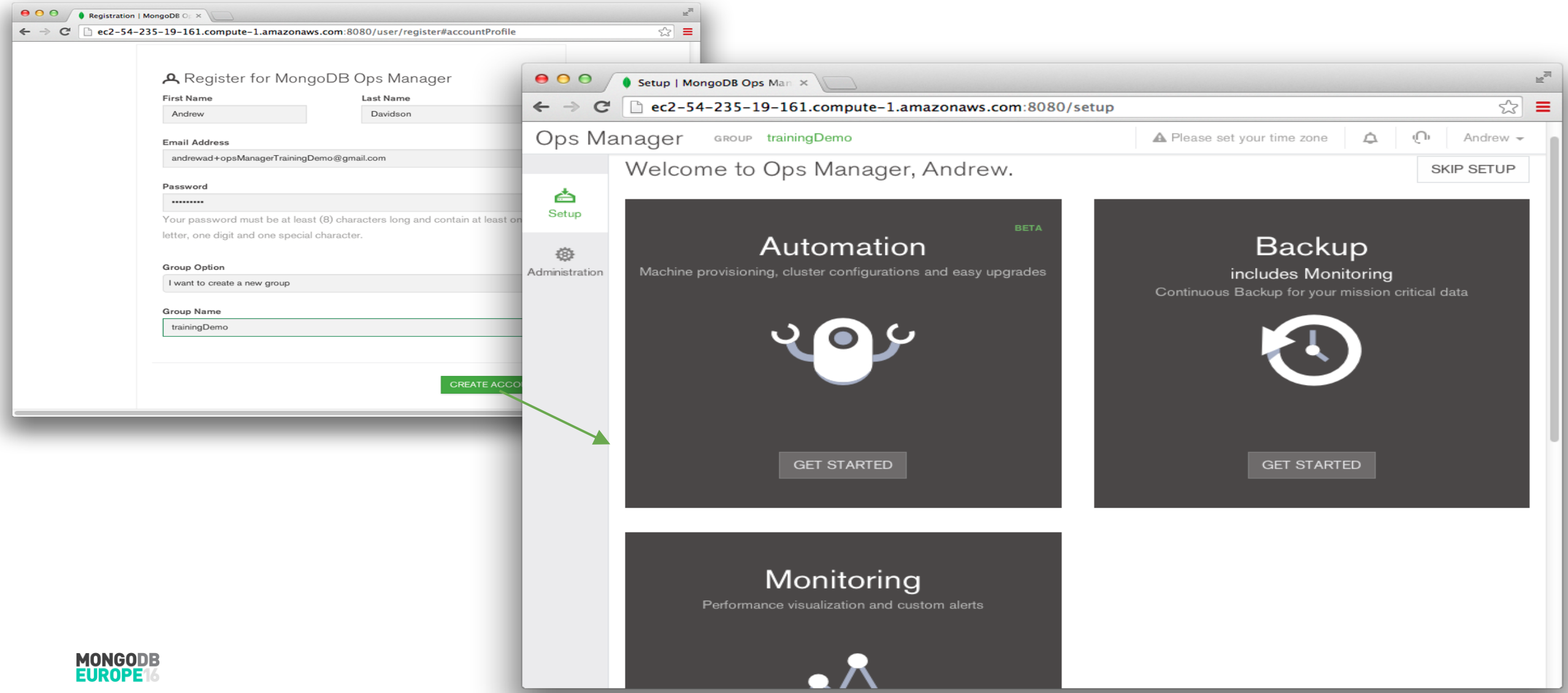


Use terminal to install

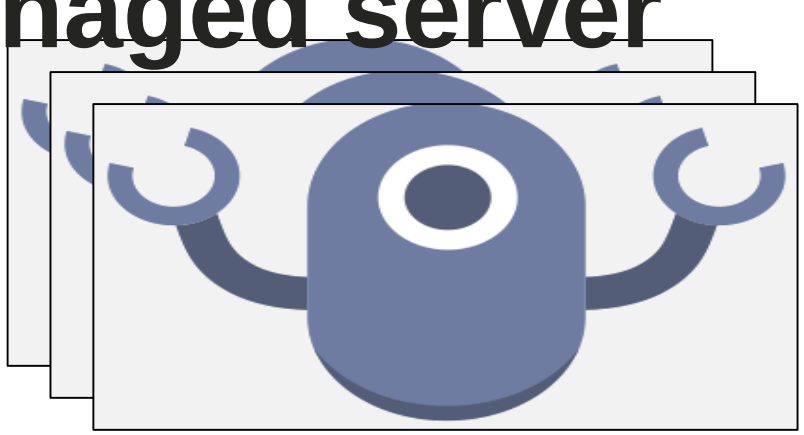


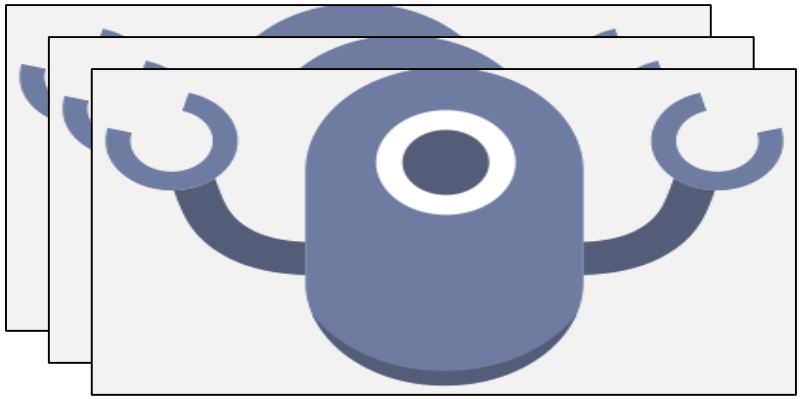
Browse to
http://<my-ops-mgr>:8080/

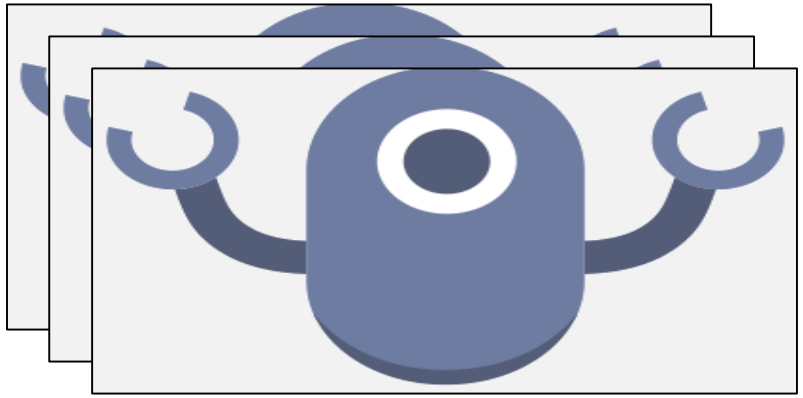
Register and create group

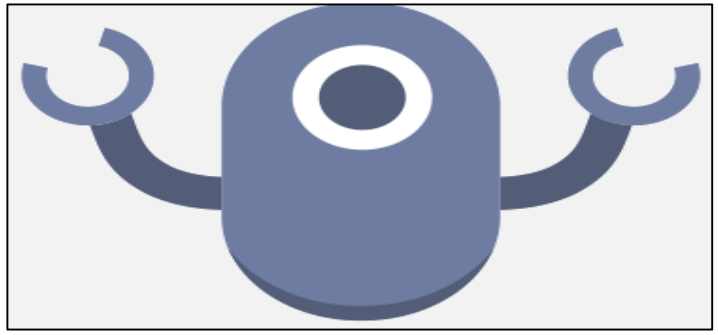


Install an Automation Agent on each managed server









Next Steps

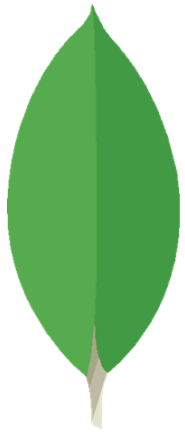
Ops Manager is F*** for Evaluation and Development

Try it today!
mongodb.com/download → Ops Manager
mongodb.com/cloud → Cloud Manager

Documentation
docs.opsmanager.mongodb.com

Operations Rapid Start

Overview	What's Included	Engagement Details
• Automated operations, monitoring and alerting, and safe backup and recovery for your deployment with optimal usage of MongoDB's management tools	• Introductory administrator training with detailed training materials • Setup and configuration of Ops Manager or Cloud Manager • Operations playbooks for administration, backup, and recovery	• An initial prep call to gather sizing requirements and make hardware recommendations • A MongoDB consulting engineer will spend at least 4 consecutive days working with your team. • Comprehensive written report within 10 business days



mongoDB

Questions?

MONGODB EUROPE 16

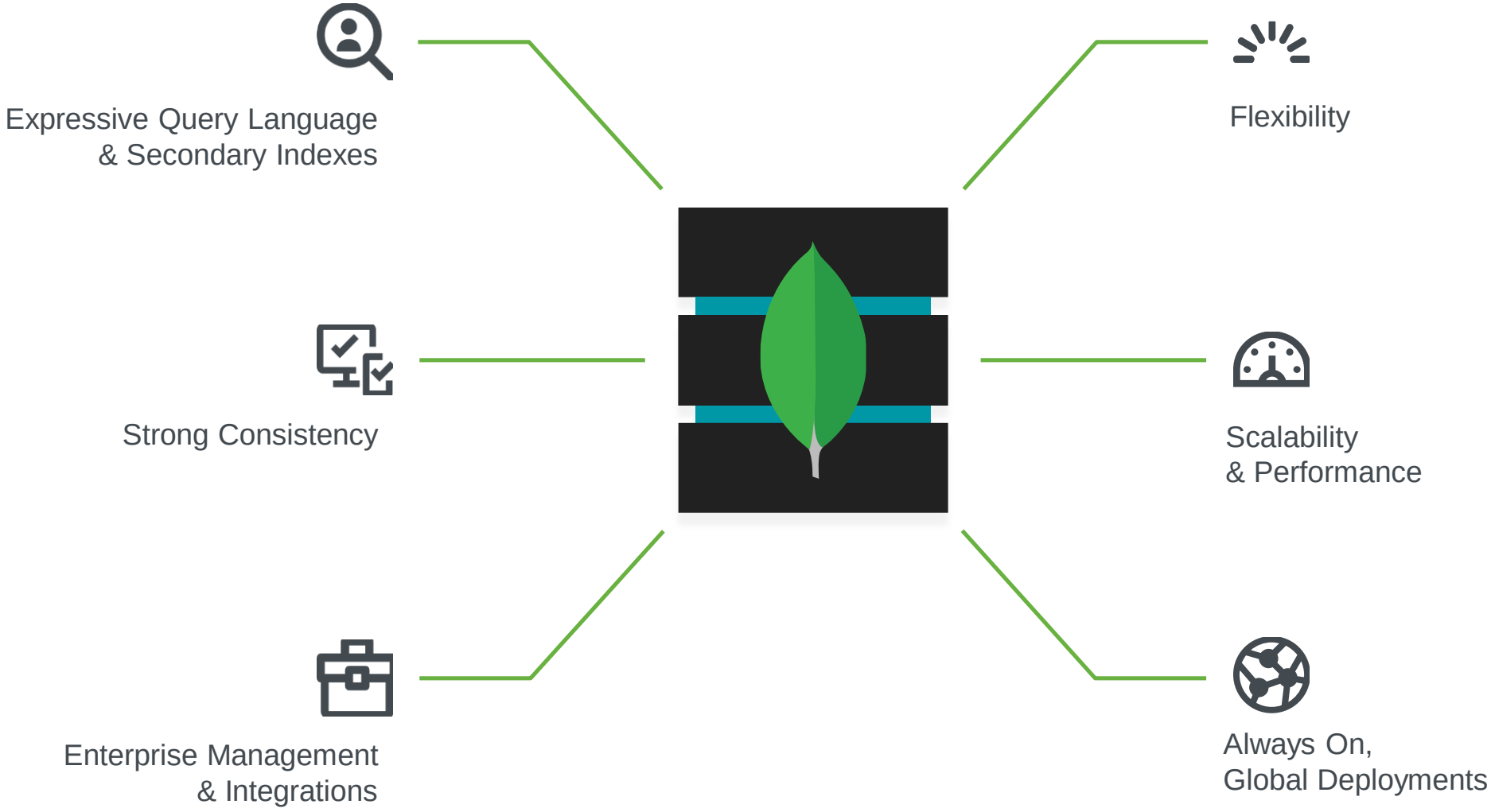
FOR GIANT IDEAS

MongoDB as a Service

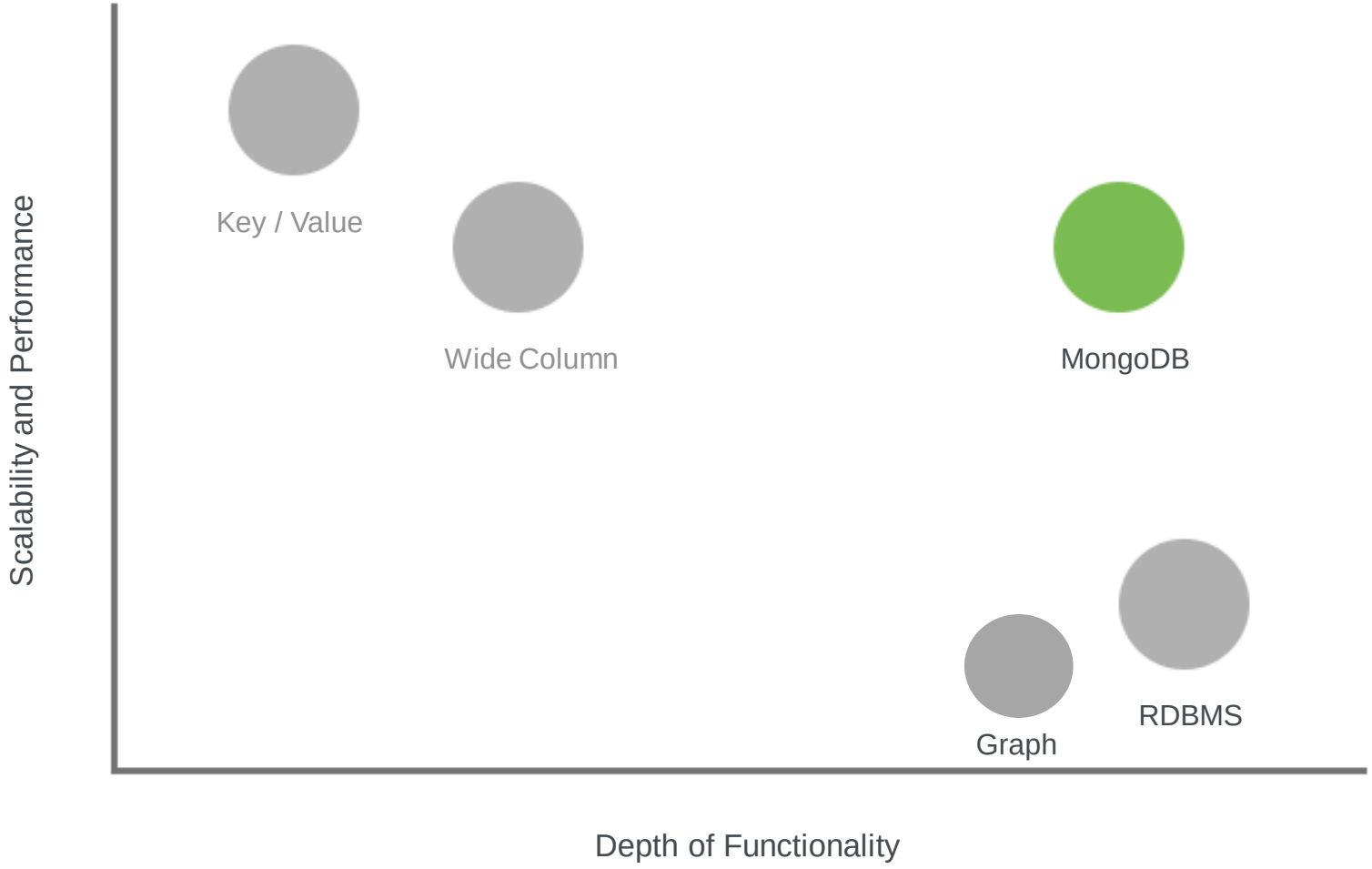
7th Sept 2016, v1.2

MongoDB Point of View

MongoDB Nexus Architecture

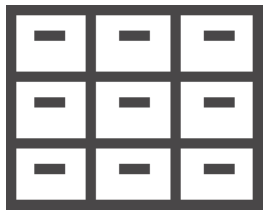


Operational Database Landscape

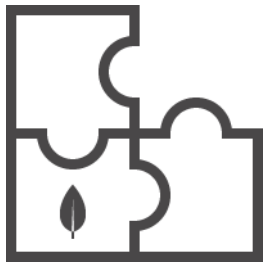


How we can help

Value of MongoDB as a Service



Standardize operations across multiple MongoDB deployments



Integrate MongoDB into your environment and tooling

Speed development with a fast and easy dev/test sandbox

Reduce costs, complexity and operations overhead while increasing value.

MongoDBaaS technology overview

PaaS / Configuration Management

- Tools: Puppet, Chef, CFEngine, Cloud Foundry, OpenShift…
- MongoDB (and other technologies) provisioning is ultimately driven by the chosen PaaS / configuration management tool
- The PaaS calls into Ops Manager as needed (via REST API) for the provisioning of MongoDB

Infrastructure: IaaS, Virtual or Physical

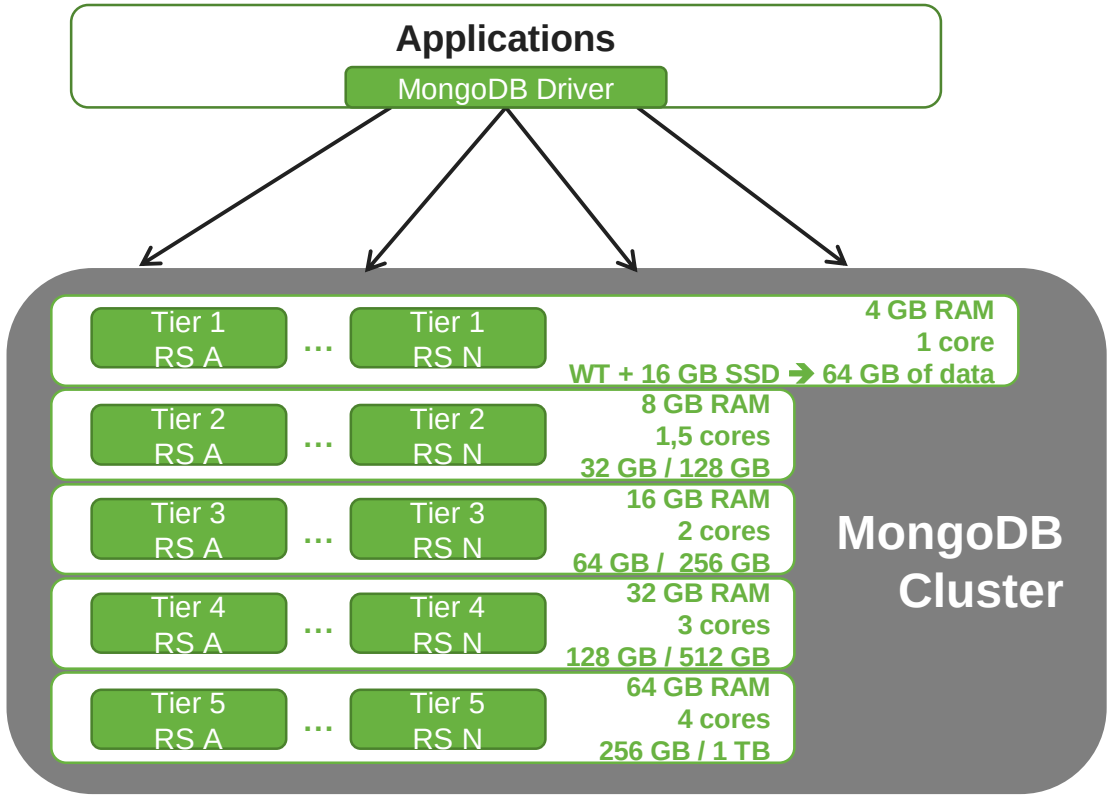
- VM hypervisor (Vmware, Vbox)
- Cloud provider (Azure, Amazon)
- Container manager (Docker, Drawbridge)
- Pool of physical servers

MongoDB Provisioning

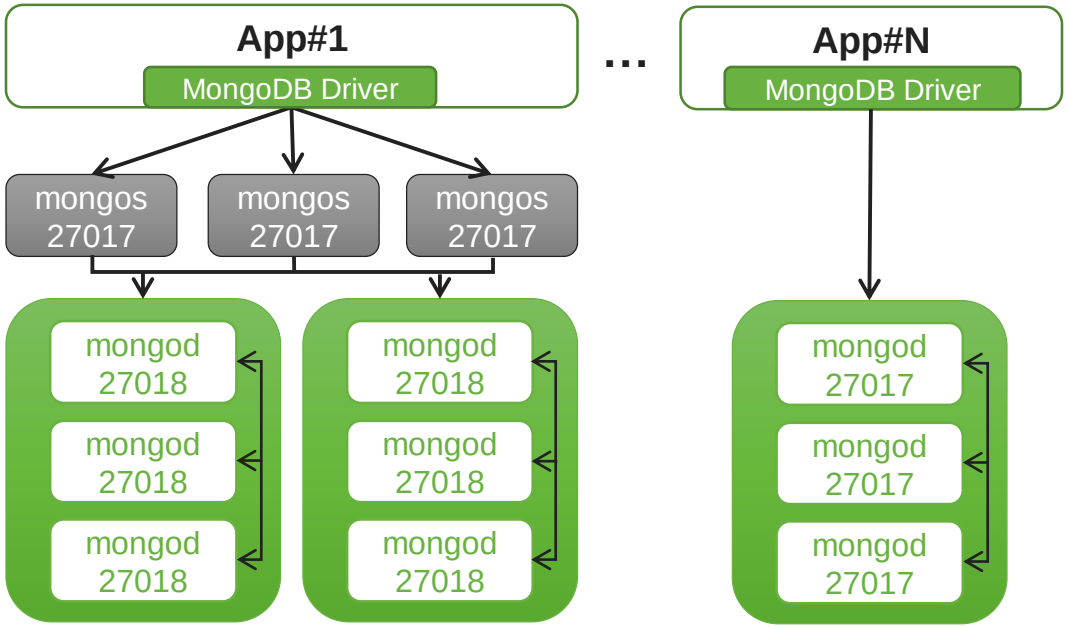
- Provisions MongoDB software on the underlying infrastructure / One click button
- Ensures security compliance and deployment best practices
- Provides monitoring and backup

Target Architecture

MongoDBaaS for most apps



Dedicated environments for other applications
(>2TB or strong need of dedicated resources)



Ops Manager: Automatisation, Monitoring & Backup/Restore

Technical Solution

MongoDB Enterprise Advanced

24 x 7 Support
(1 hour SLA)

Platform Certifications

Emergency Patches

Customer Success Program

On-Demand Online Training

MongoDB Ops Manager

- Monitoring & Alerting
- Query Optimization
- Automation & Configuration
- Backup & Recovery

REST API

MongoDB Compass

- Schema Visualization
- Data Exploration
- Ad-Hoc Queries

MongoDB Connector for BI

- Visualization
- Analysis
- Reporting

Authentication | Authorization | Auditing | Encryption (In Flight & at Rest)

MongoDB Enterprise Server

Commercial License
(No AGPL Copyleft Restrictions)

Indemnification

Limitation of Liability

Warranty

Zoom on Ops Manager

Monitoring

- Dozens of charts tracking key performance indicators
- Custom alerts that trigger when key metrics are out of range
- RESTful API to integrate with your existing APM tools

Optimization

- Visual displays of query and write latency
- Recommendations for new indexes to improve query performance
- One-click rollout of new indexes across your deployment; according to best practices and with no downtime

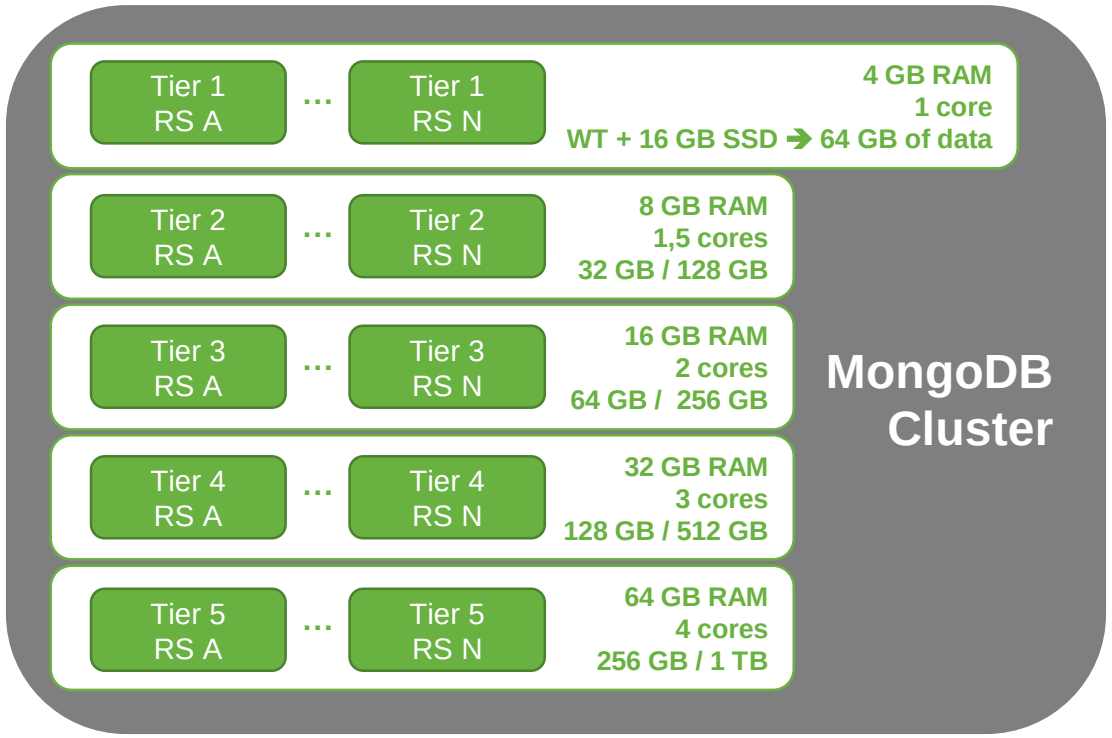
Automation

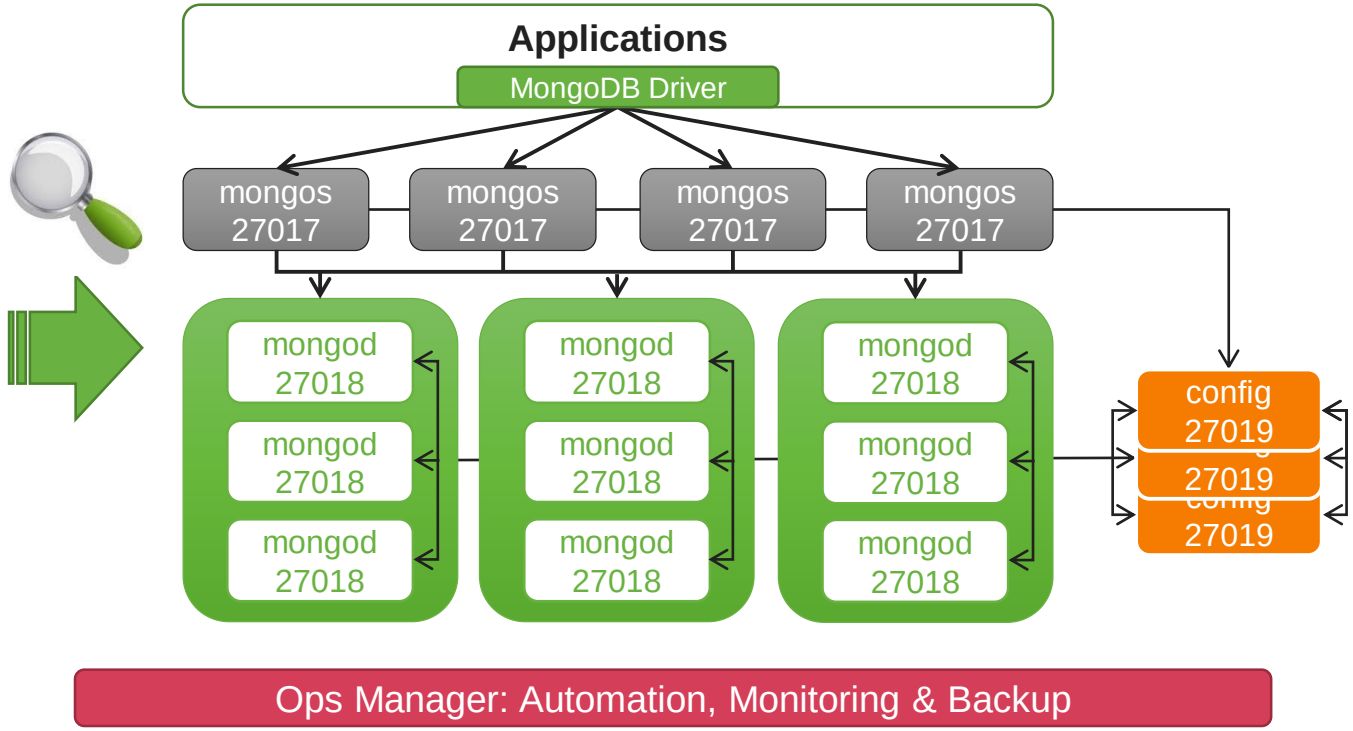
- Deploy, resize, and upgrade your deployments with just a few clicks
- Reduce the operational overhead of running MongoDB; enable your ops team to be 10-20x more efficient
- RESTful API to integrate with your enterprise orchestration tools

Backup

- Continuous backups to minimize your exposure to data loss
- Restore to precisely the moment you need with point-in-time recovery

MongoDBaaS Architecture Diagram





- One single MongoDB Cluster:
  - Each application has a dedicated RS (each one a shard)
  - Optionally, mongos can be included in the app server.
  - Easy to tune (different config per app/server)
  - Easy to uplift databases as they grow
  - No sharded collections (<2TB)
  - Not designed for apps that require dedicated resources

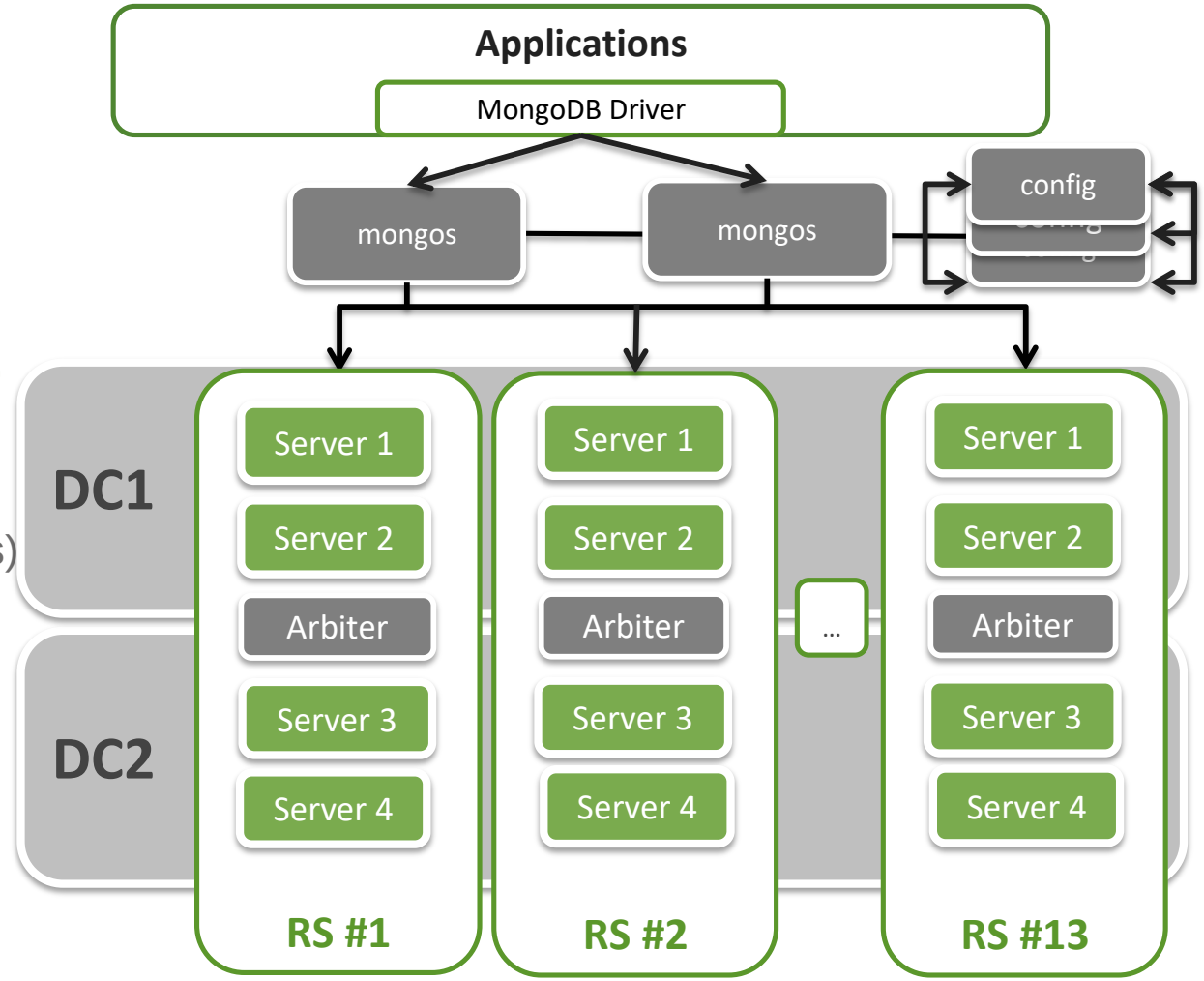
Ops Manager in command of operations:

- Servers are added using images (VMWare, Docker, etc.) including only the automation agent.
- Ops Manager deploys on the provided servers.
- All group policies (security, monitoring, backup, etc.) are applied automatically
- Dynamic allocation by layer based on resource need

MongoDBaaS

- MongoDB Version: 3.2 / Storage Engine: WiredTiger
- Data Server configuration (Server #1 to #4) - Physical:
  - Processor: 2 x 6 cores+HT @2.6 Ghz. (24 logical cores)
  - RAM: 256 GB RAM
  - Storage:
    - 4 x 240 GB SSD (Data + Journal + Oplog + Indexes)
    - 1 TB HDD (Logs, binaries, OS, …)
- Router Server configuration (mongos) – Virtual:
  - 1 x 1 core @2.5 Ghz cores / 4 GB RAM
- Config Server configuration - Virtual:
  - 0,5 x 1 core @2.5 Ghz / 2 GB RAM

- Configuration expected performance:
  - 180k op/sec & 2.5 TB of Data (Each instance specifications can be adjusted individually)
  - Up to 13 Applications
- Next Configuration to reach next limit (26 Apps / 5 TB data)
  - Double current configuration (Add servers #5 to #8 with same config)

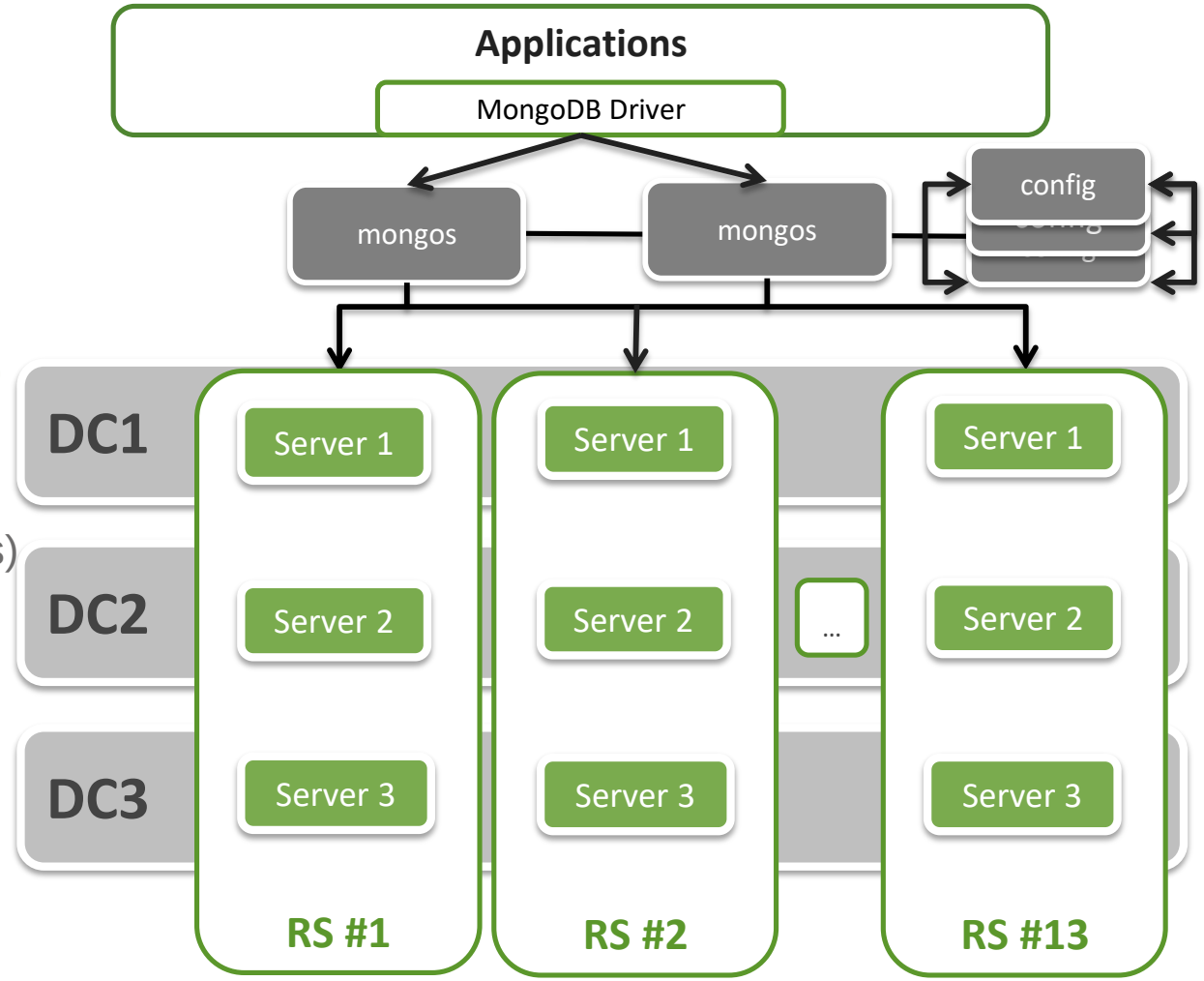


Type	# servers	RAM	Cores	SSD
Mongos	1	4	1	8
Config	1	1	0,5	8
Tier 1	2	4	1	16
Tier 2	5	8	1,5	32
Tier 3	3	16	2	64
Tier 4	2	32	3	128
Tier 5	1	64	4	256
Total	**13**	**229**	**27**	**912**

MongoDBaaS

- MongoDB Version: 3.2 / Storage Engine: WiredTiger
- Data Server configuration (Server #1 to #3) - Physical:
  - Processor: 2 x 6 cores+HT @2.6 Ghz. (24 logical cores)
  - RAM: 256 GB RAM
  - Storage:
    - 4 x 240 GB SSD (Data + Journal + Oplog + Indexes)
    - 1 TB HDD (Logs, binaries, OS, …)
- Router Server configuration (mongos) – Virtual:
  - 1 x 1 core @2.5 Ghz cores / 4 GB RAM
- Config Server configuration - Virtual:
  - 0,5 x 1 core @2.5 Ghz / 2 GB RAM

- Configuration expected performance:
  - 180k op/sec & 2.5 TB of Data (Each instance specifications can be adjusted individually)
  - Up to 13 Applications
- Next Configuration to reach next limit (26 Apps / 5 TB data)
  - Double current configuration (Add servers #4 to #6 with same config)



Type	# servers	RAM	Cores	SSD
Mongos	1	4	1	8
Config	1	1	0,5	8
Tier 1	2	4	1	16
Tier 2	5	8	1,5	32
Tier 3	3	16	2	64
Tier 4	2	32	3	128
Tier 5	1	64	4	256
Total	**13**	**229**	**27**	**912**

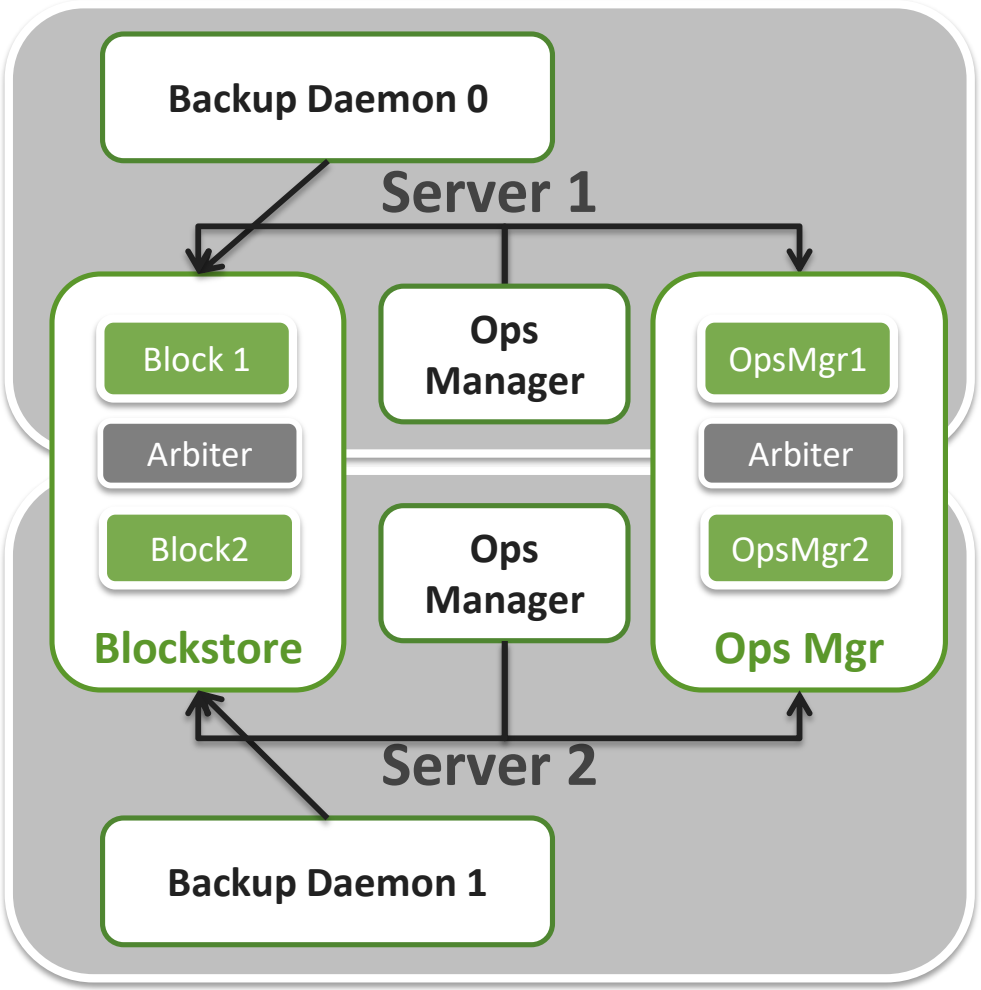
Ops Manager

- Ops Manager requires 2 databases + 1 application to be installed
  - Blockstore: Stores backup data and provides restores.
    - 8 cores @ 2.0 Ghz / 192 GB RAM / 10 TB HDD *
  - Ops Mgr: Stores monitoring, configuration & operational data
    - 2 cores @ 2.0 Ghz / 8 GB RAM / 200 GB HDD
  - Ops Manager: Web application. UI, REST API & profiling tools. Also controls backups daemons. Dual install
    - 2 cores @ 2.0 Ghz / 64 GB RAM / 2 TB HDD

- 2 Servers - Physical:
  - Processor: 2 x 6 cores+HT @2.6 Ghz.
  - RAM: 256 GB RAM
  - Storage: 12 TB HDD **
- Configuration expected performance:
  - Monitoring, automation and backup
    - Up to 160 nodes. Up to 6 TB of backed DB datafiles

*Final sizing to be done during Ops Optimization.

**Storage size dependant ond retention policies.



Evidence

Mobile Platform

Google provides a fast and flexible backend database for 375K+ mobile app developers, growing at 25x per year

Problem

Rigid data model: impossible to design a centralized schema that satisfies every application for DBaaS

Existing key-value database could not provide needed query flexibility

Existing relational databases could not scale to support workloads of tens of thousands of apps

Solution

Flexible data model supports diverse application requirements

Expressive query language and rich secondary indexes provide flexibility to support both ad-hoc and predefined queries

Elastic scalability supports thousands of applications running concurrently on the same platform

Results

Development and deployment times now up to 3x faster

Complex queries can be run with 80% less code and higher network efficiency than key-value database

MongoDB scales to support 25x forecast growth rate over the next 12 months

Database-as-a-Service

Migration from Oracle & Microsoft to create a consolidated "data fabric" reduces $m in cost, speeds application development & simplifies operations

Problem

High licensing costs from proprietary database and data grid technologies

Data duplication across systems with complex reconciliation controls

High operational complexity impacting service availability and speed of application delivery

Solution

Implemented a multi-tenant PaaS with shared data service based on MongoDB, accessed via a common API with message routing via Kafka

Standardized data structures for storage and communication based on JSON format

Multi-sharded, cross-data center deployment for scalability and availability

Results

$ millions in savings after migration from Coherence, Oracle database and Microsoft SQL Server

Develop new apps in days vs months

100% uptime with simplified platform architecture, higher utilization and reduced data center footprint

Database as a Service

Telecom equipment manufacturer provides a scalable database as a service for internal applications

Problem

Huawei had to support numerous internal applications and store associated big data, including logs and large attachments

Previous relational solutions (DB2 and MySQL) lacked horizontal scalability, delivered poor performance, especially in long-running analytical queries, and made it difficult to manage unstructured data

Solution

Huawei built the Huawei Application Engine (HAE) using MongoDB as the underlying database layer

MongoDB scaled horizontally to meet the customer's needs, and allowed them to distribute the cluster across data centers around the world

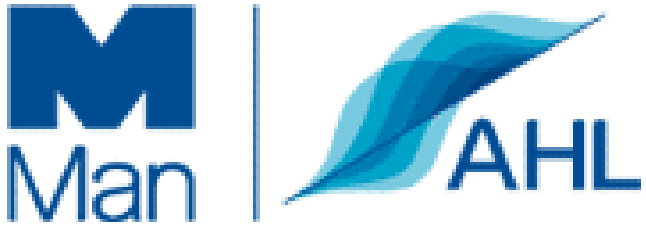
The HAE easily handles unstructured and variably structured data from Huawei's internal applications

Results

Accelerated app releases by allowing developers to focus on apps vs. ops

Replaced multiple technologies with a single MongoDB deployment, reducing costs and operational complexity

Improved concurrency and performance vs. previous RDBMS – with MongoDB 100K ops/sec; queries reduced from minutes to seconds



Single Platform for Financial Data

Quantitative investment manager with over $11B in assets under management invests heavily in new database

Problem

AHL needed new technologies to be more agile and gain competitive advantages in the systematic trading space

Proprietary systems in financial services tech, as well as relational databases, were too expensive and/or rigid

Solution

Built single platform for all financial data on MongoDB

Flexible data model and scalability were core to ability to put all data in single platform

Expressive query language, secondary indexes and strong consistency were core to ability to migrate core use cases to new platform

Results

100x faster to retrieve data

Tick Data: Quickly scaled to 250M ticks per second, a 25x improvement in tick throughput

Cut disk storage 60%, and realized 40% cost savings by using commodity SSDs

MongoDB-as-a-Service

With the help of MongoDB Professional Services, Swisscom built a MongoDB-as-a-Service deployment exposed both internally and externally

Problem

Swisscom sought to deploy a MongoDB Database-as-a-Service, as well as a Platform-as-a-Service built on MongoDB, to help its developers to be more efficient

Swisscom wanted to ensure high performance for the applications built on the Swisscom App Cloud, and train the the teams managing it to be MongoDB experts

Solution

Swisscom engaged MongoDB's professional services to train the operations team, set up and integrate MongoDB management tools, develop a backup and restore strategy, and optimize the first application to use App Cloud

MongoDB's consulting engineer worked with the teams over multiple months to architect a performant DBaaS

Results

First app built on Swisscom App Cloud launched in trial, quickly grew to 10k users, and is expected to grow 10x at full launch

Performance tuning kept average query time under a millisecond, with 50k new objects added per day

App Cloud was so successful it was exposed for use by external developers

Operations Rapid Start

Overview

- Automated operations, monitoring and alerting, and safe backup and recovery for your deployment with optimal usage of MongoDB's management tools

What's Included

- Introductory administrator training with detailed training materials
- Setup and configuration of Ops Manager or Cloud Manager
- Operations playbooks for administration, backup, and recovery

Engagement Details

- An initial prep call to gather sizing requirements and make hardware recommendations
- A MongoDB consulting engineer will spend at least 4 consecutive days working with your team, typically on site.
- Comprehensive written report within 10 business days

MongoDB Private Cloud Accelerator

Overview

- Standardize, centralize, and automate MongoDB operations for multiple applications across your organization

What's Included

- Introductory and advanced administrator training
- Cloud Manager or Ops Manager installed, configured, and integrated into your environment and tooling
- Custom playbooks for backup and restore, plus operations runbooks for your MongoDB private cloud
- Multiple applications evaluated, optimized, and onboarded to the private cloud

Engagement Details

- A MongoDB consulting engineer will spend 60 days working with your team, on a schedule to be agreed upon with MongoDB
- All 60 days of professional services are to be used within 1 year

mongoDB

Thank You

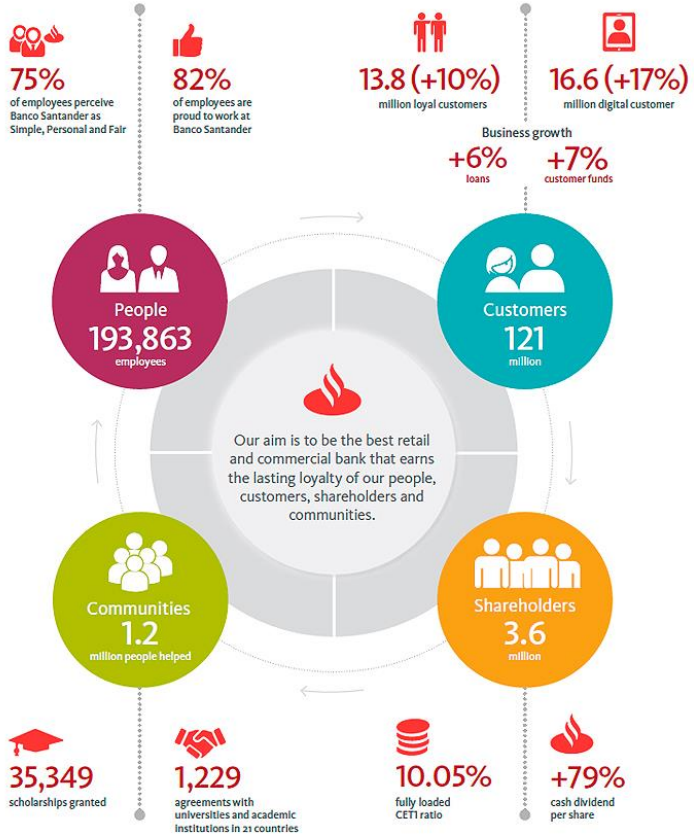
Rubén Terceño

Solutions Architect, MongoDB

ruben@mongodb.com

Summary

Santander maintained in 2015 attributable profit of 5,966 million € with a market capitalization of 65.792 million €.



Company Background

The Santander Group is a Spanish banking group centered on Banco Santander, S.A. As its name suggests, the company originated in Santander, Cantabria, Spain.

The group has expanded since 2000 through a number of acquisitions, with operations across Europe, Latin America, North America and Asia. Many subsidiaries, such as Abbey National, have been rebranded under the Santander name.

The company is a component of the Euro Stoxx 50 stock market index.In April 2013, Santander was ranked as 43rd in the Forbes Global 2000 list of the world's largest companies.

Company Organisation

Santander Spain is organized in three different companies from IT perpective: Isabn, in charge of development and innovation, Produban in charge of productions systems and Santander España, voice of business on the IT arena.

The problem

Isban has adopted MongoDB and homologated the solution one year ago, and now is a common choice for the laboratories and development teams.

Produban, in charge of productions sytems, is seen as inhibitor of the innovation due to their long procurement and deployment process.

Despite MongoDB is been used in several projects in production, each new development needs to be trated individually, and the variety of versions and deployment modes of MongoDB is becoming a problem.

The solution

In order to get production teams the best of breed we pushed for MongoDB Enterprise Advance for all projects arriving to production.

Ops Manager and Security features where key differentiators, but not all projects justify acquiring a license because either their size or criticism is small.

In order to accomodate all developments in the same platform and providing enterprice-class, homogeneus capabilitites to the production team we proposed to build up a MDBaaS platform.

Benefits

All new deployments will share the same infrastructure so the license costs will be diluted.

At the same time, thanks to virtualization, the different applications have access to dedicated resources.

The usage of local mongos process in the application servers simplifies the connectivity of the applications and allow simple database upgrades.

Ops Manager provides centralized monitoring, backup and operational excellence for a large number of applications.

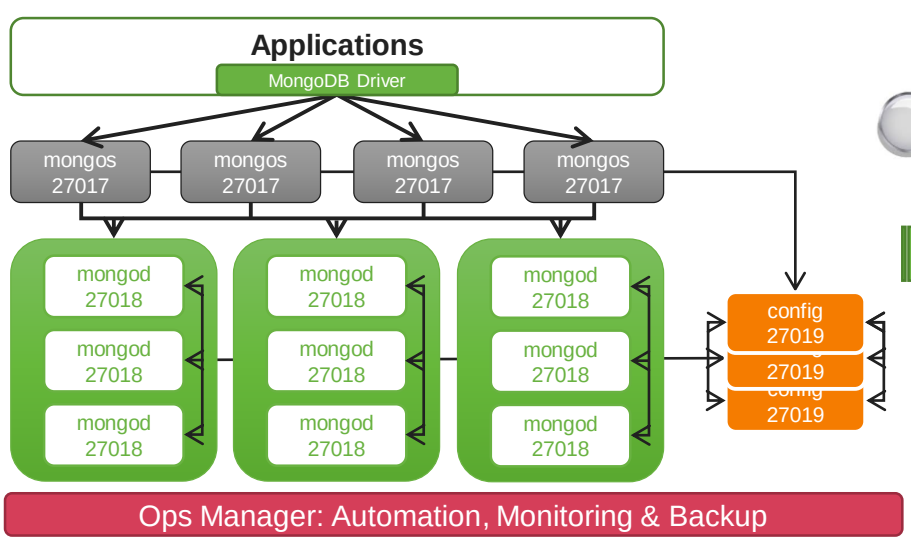
The solution allows starting small and grow organically as needed.

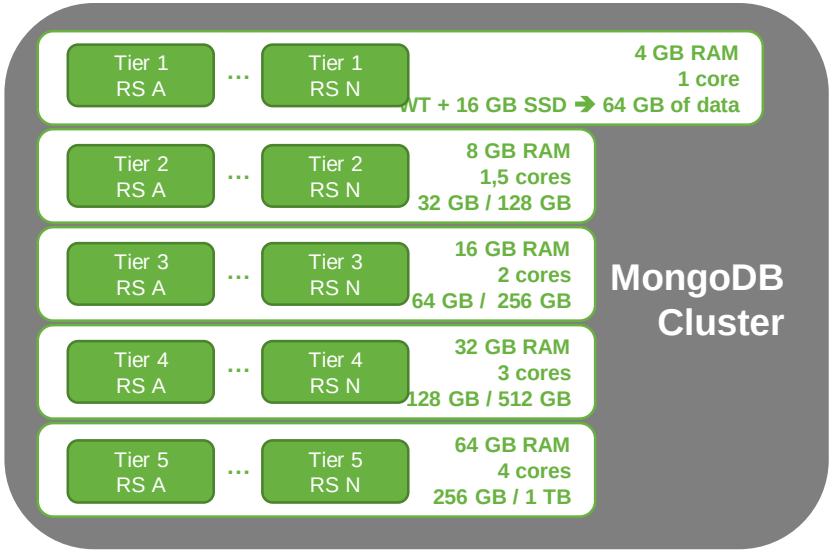
Results

New environments are provisioned in hours instead of weeks.

The whole project investment has been justified only with savings on the storage layer by using SSD local drives instead of the corporate SAN servers.

MongoDBaaS Architecture Diagram





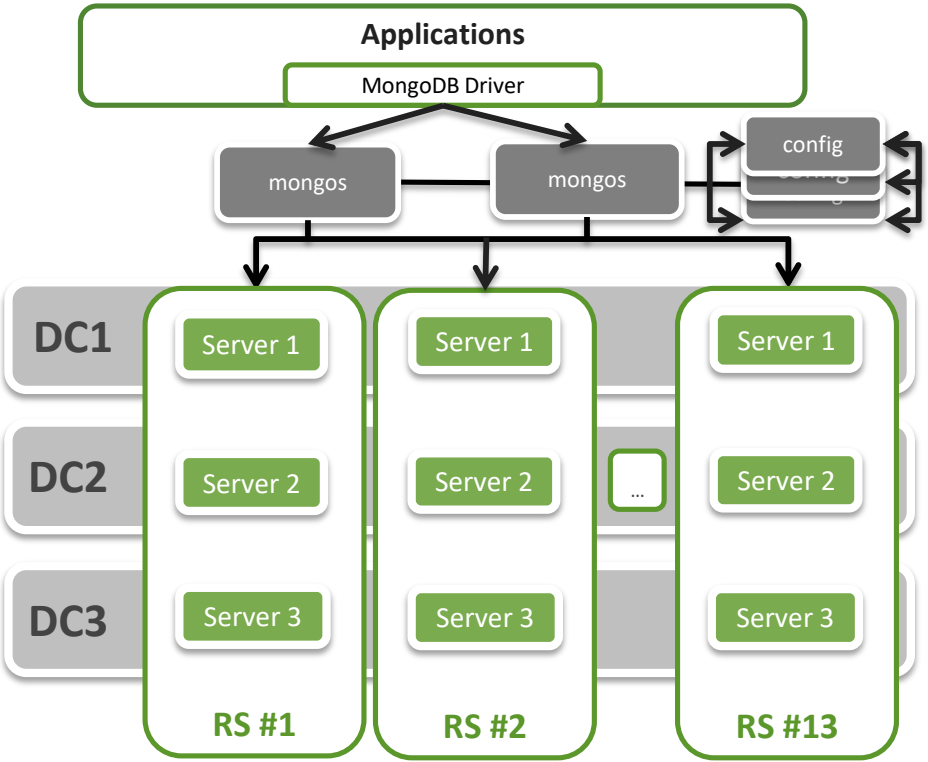
- One single MongoDB Cluster:
  - Each replicaset is a shard
  - Easy to administer
  - Easy to differentiate (i.e. config per app/server)
  - Easy to uplift databases as they grow
  - No sharded collections (<2TB)
  - Not designed for apps that require dedicated resources

- Ops Manager in command of operations:
  - Servers are images with the automation agent
  - Easier to implement corporate security policies
  - Multi tenancy at hardware level
  - Dynamic allocation by layer based on resource need

MongoDBaaS

- MongoDB Version: 3.2 / Storage Engine: WiredTiger
- Data Server configuration (Server #1 to #3) - Physical:
  - Processor: 2 x 6 cores+HT @2.6 Ghz. (24 logical cores)
  - RAM: 256 GB RAM
  - Storage:
    - 4 x 240GB SSD (Data + Journal + Oplog + Indexes)
    - 1 TB HDD (Logs, binaries, OS, …)
- Router Server configuration (mongos) – Virtual:
  - 1 x 1 core @2.5 Ghz cores / 4 GB RAM
- Config Server configuration - Virtual:
  - 0,5 x 1 core @2.5 Ghz / 2 GB RAM

- Configuration expected performance:
  - 180k op/sec & 2.5 TB of Data (Each instance specifications can be adjusted individually)
  - Up to 13 Applications
- Next Configuration to reach next limit (26 Apps / 5 TB data)
  - Double current configuration (Add servers #4 to #6 with same config)



Type	# servers	RAM	Cores	SSD
Mongos	1	4	1	8
Config	1	1	0,5	8
Tier 1	2	4	1	16
Tier 2	5	8	1,5	32
Tier 3	3	16	2	64
Tier 4	2	32	3	128
Tier 5	1	64	4	256
Total	**13**	**229**	**27**	**912**

Ops Manager

- Ops Manager requires 2 databases + 1 application to be installed
  - Blockstore: Stores backup data and provides restores.
    - 8 cores @ 2.0 Ghz / 192 GB RAM / 10 TB HDD *
  - Ops Mgr: Stores monitoring, configuration & operational data
    - 2 cores @ 2.0 Ghz / 8 GB RAM / 200 GB HDD
  - Ops Manager: Web application. UI, REST API & profiling tools. Also controls backups daemons. Dual install
    - 2 cores @ 2.0 Ghz / 64 GB RAM / 2 TB HDD

- 2 Servers - Physical:
  - Processor: 2 x 6 cores+HT @2.6 Ghz.
  - RAM: 256 GB RAM
  - Storage: 12 TB HDD **
- Configuration expected performance:
  - Monitoring, automation and backup
    - Up to 160 nodes. Up to 6 TB of backed DB datafiles

*Final sizing to be done during Ops Optimization.

**Storage size dependant ond retention policies.