Which Questions we should have

Jun Kim , Senior Solution Architect &
Database Specialist

Agenda

1. Polyglot persistence & Multi-model Database
2. NoSQL Vs RDBMS
3. Considerations & Questions ?
4. MongoDB is ..
5. Proof of Points

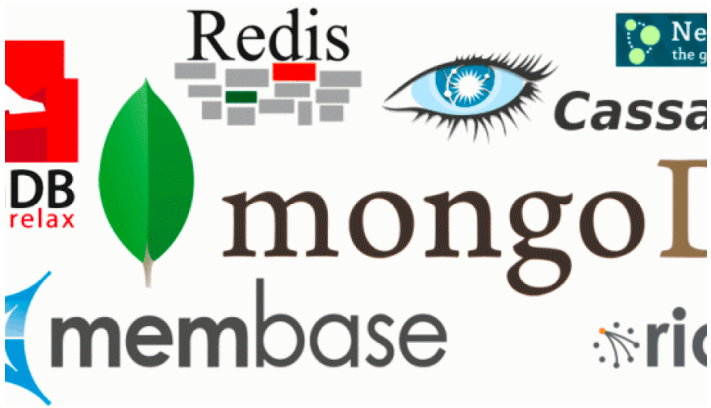
Polyglot persistence Vs Multi-model database

Polyglot persistence is the concept of using different data storage technologies to handle different data storage needs within a given software application.

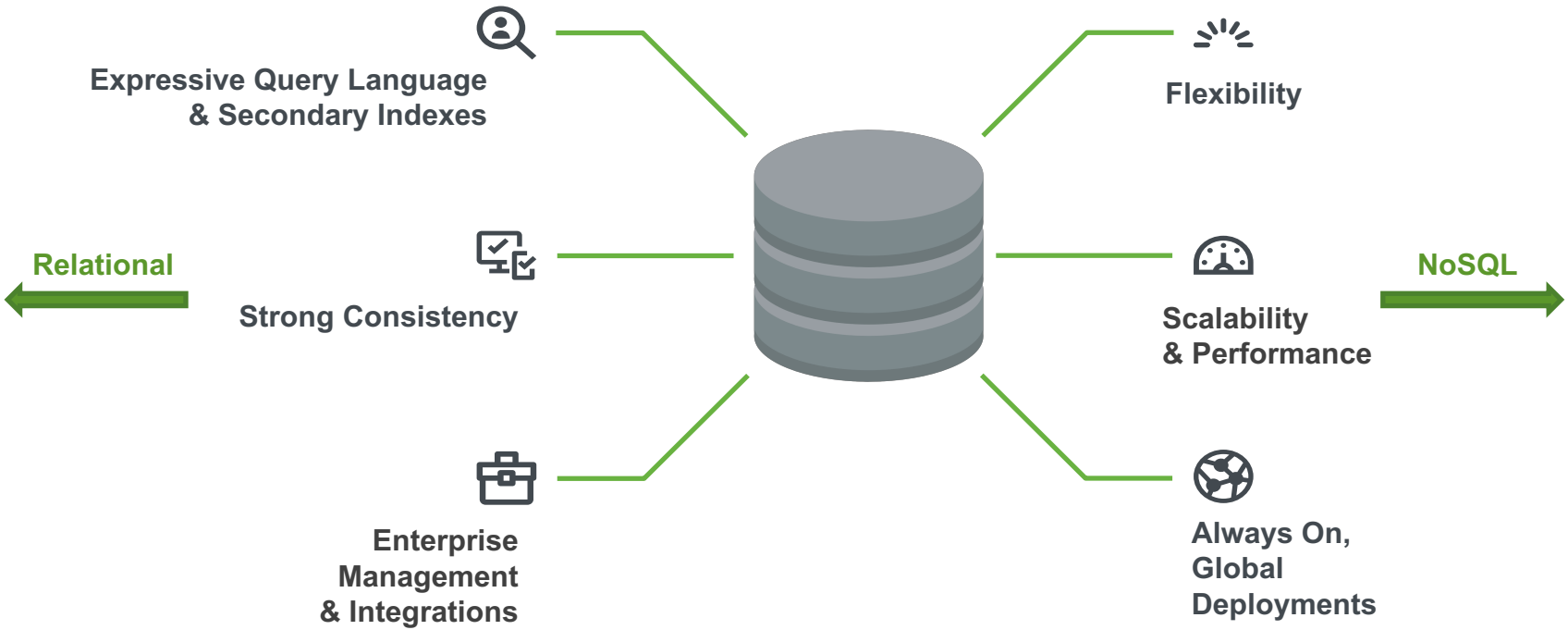
Most database management systems are organized around a single data model that determines how data can be organized, stored, and manipulated. In contrast, a **multi-model database** is designed to support multiple data models against a single, integrated backend. Document, graph, relational, and key-value models are examples of data models that may be supported by a multi-model database.

RDBMSs Vs NoSQLs

How To Use in the Market



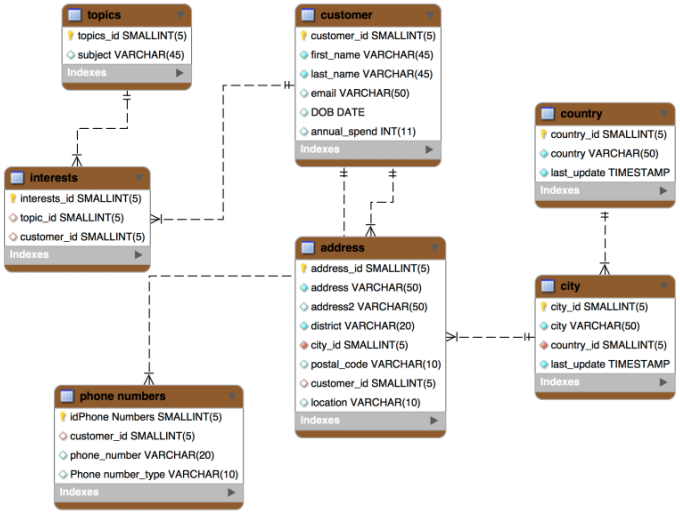
Strengths & Weakness



Q1 : Data Model

- All of these data models provide **schema flexibility**. → What do you mean flexibility ?
- The key-value and wide-column data model is opaque in the system - only the primary key can be queried. → is it really need ?
- The document data model has the **broadest applicability**. → Why ?
- The document data model is the **most natural and most productive** because it maps directly to objects in modern object-oriented languages. → more specific for productive ?
- The wide column model provides more granular access to data than the key value model, but less flexibility than the document data model. → why ?

What do you mean ?




```
_id: 12345678
name: Object
   first: "John"
   second: "Doe"
address: Array
   0: Object
      location: "work"
      address: Object
         number: 16
         street: "Hatfields"
         city: "London"
         postalCode: "SE1 8DJ"
   1: Object
      location: "home"
      address: Object
phone: Array
   0: Object
      location: "work"
      number: "+44-1234567890"
   1: Object
   2: Object
email: "john.doe@mongodb.com"
dob: 1966-07-30 01:00:00.000
interests: Array
   0: "Cycling"
   1: "IoT"
```

Tabular (Relational) Data Model

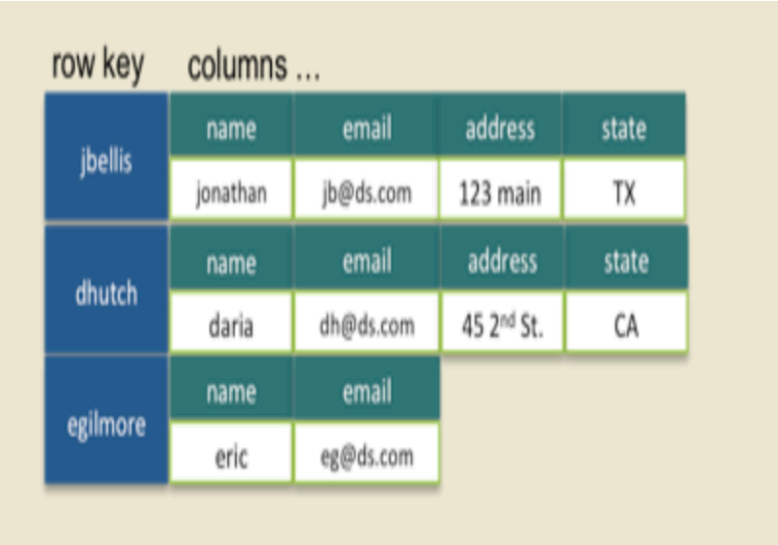
Related data split across multiple records and tables

Document Data Model

Related data contained in a single, rich document

mongoDB.

What do you mean ?




```
_id: 12345678
name: Object
  first: "John"
  second: "Doe"
address: Array
  0: Object
    location: "work"
    address: Object
      number: 16
      street: "Hatfields"
      city: "London"
      postalCode: "SE1 8DJ"
  1: Object
    location: "home"
    address: Object
phone: Array
  0: Object
    location: "work"
    number: "+44-1234567890"
  1: Object
  2: Object
email: "john.doe@mongodb.com"
dob: 1966-07-30 01:00:00.000
interests: Array
  0: "Cycling"
  1: "IoT"
```

Tabular (Relational) Data Model
Related data split across multiple records and tables

Document Data Model
Related data contained in a single, rich document

Q2 : Query Model

- The biggest difference between non-tabular databases lies in the ability to **query data efficiently**. → ambiguity

- Document databases provide the **richest query functionality**, which allows them to address a wide variety of operational and real-time analytics applications. → compare other data stores

- Key-value stores and wide column stores provide a single means of accessing data: by primary key. This can be fast, but they offer very **limited query functionality** and may impose additional development costs and application-level requirements to support anything more than basic query patterns. → it is not purpose for polygolot ?

- HA & Scales View of Points → utilize the resources ?

What do you mean ?

Expressive Queries	• Find anyone with phone # “1-212…” • Check if the person with number “555…” is on the “do not call” list
Geospatial	• Find the best offer for the customer at geo coordinates of 42nd St. and 6th Ave
Text Search	• Find all tweets that mention the firm within the last 2 days
Aggregation	• Count and sort number of customers by city, compute min, max, and average spend
Native Binary JSON support	• Add a phone number to Mark Smith’s • Update just 2 phone numbers out of 10 • Sort on the modified date
JOIN ($lookup)	• Query for all San Francisco residences, lookup their transactions, and sum the amount by person
Graph queries ($graphLookup)	• Query for all people within 3 degrees of separation from Mark

MongoDB

```
{       customer_id : 1,
        first_name : "Mark",
        last_name : "Smith",
        city : "San Francisco",
        phones: [{
            number : “1-212-777-
1212”,
            type : “work”
        },
        {
            number : “1-212-777-
1213”,
            type : “cell”
        }]
...
```

mongoDB.

What do you mean ?

Fully Indexable

Fully featured secondary indexes

Index Types

- Primary Index
  - Every Collection has a primary key index
- Compound Index
  - Index against multiple keys in the document
- MultiKey Index
  - Index into arrays
- Text Indexes
  - support for text searches
- GeoSpatial Indexes
  - 2d & 2dSphere indexes for spatial geometries
- Hashed Indexes
  - Hashed based values for sharding

Index Features

- TTL Indexes
  - Single Field indexes, when expired delete the document
- Unique Indexes
  - Ensures value is not duplicated
- Partial Indexes
  - Expression based indexes, allowing indexes on subsets of data
- Case Insensitive Indexes
  - supports text search using case insensitive search
- Sparse Indexes
  - Only index documents which have the given field

Q3 : Consistency and Transaction Model

- Most applications and development teams expect **strongly consistent** systems. → consistence is depends on the rules ?

- Different consistency models pose **different trade-offs** for applications in the areas of consistency and availability. → what is trade off ?

- MongoDB provides **tunable consistency**, defined at the query level. → what do you mean tunable ?

- Most non-tabular databases provide single record atomicity. This is sufficient for many applications, **but not for all**. MongoDB provides multi-document ACID guarantees, making it easier to address a complete **range of use-cases with a single data platform → data platform is ambiguity ?**

What do you mean ?

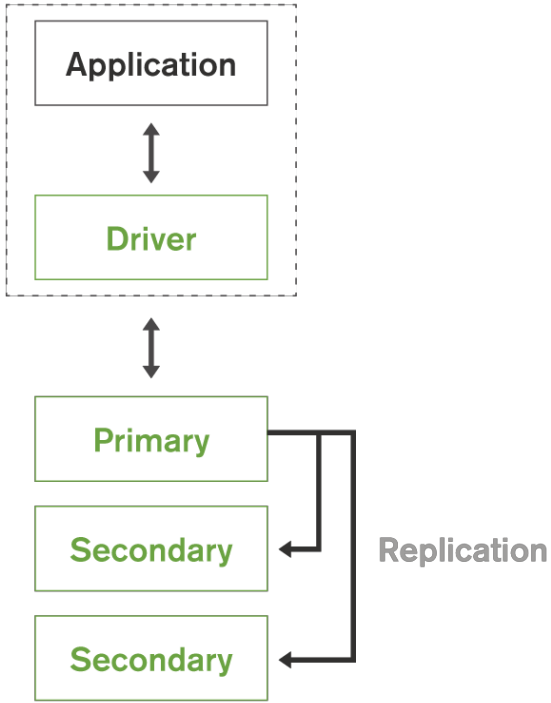
```
with client.start_session() as s:
    s.start_transaction()
    try:
        collection.insert_one(doc1, session=s)
        collection.insert_one(doc2, session=s)
        s.commit_transaction()
    except Exception:
        s.abort_transaction()
```

Natural for developers

- Idiomatic to the programming language
- Familiar to relational developers
- Simple

*Python. Syntax is subject to change

What do you mean ? High Availability



Replica Set – 2 to 50 copies

Self-healing shard

Data Center Aware

Addresses availability considerations:

High Availability

Disaster Recovery

Maintenance

Workload Isolation: operational & analytics

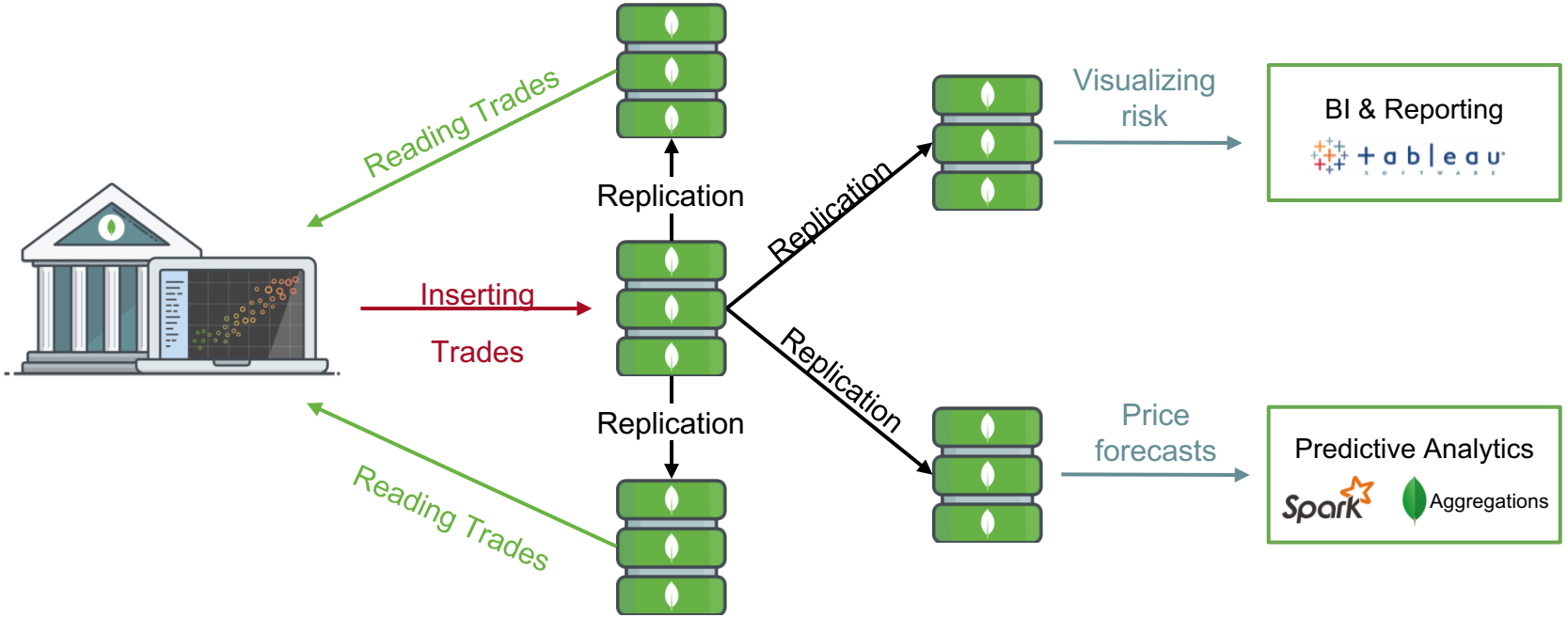
What do you mean ? Scaling Out

Three types: hash-based, range-based, location-aware

Increase or decrease capacity as you go

Automatic balancing

What do you mean ? Workload Isolation



Q4 : API

- The maturity and functionality of APIs **vary significantly** across non-relational products.

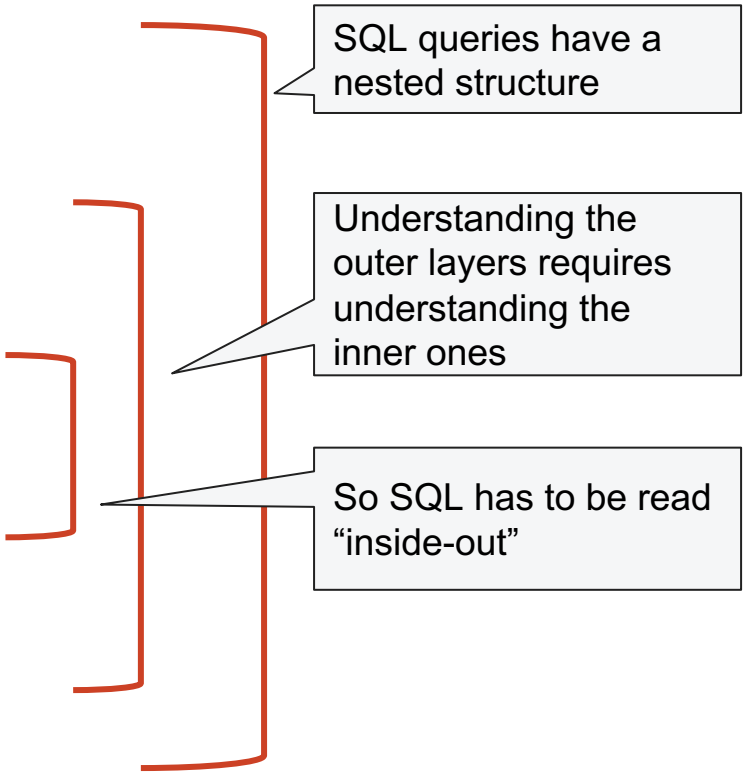
- **MongoDB's idiomatic drivers** minimize onboarding time for new developers and simplify application development.

- **Not all SQL is created equal**. Carefully evaluate the SQL-like APIs offered by non-relational databases to ensure they can meet the needs of your application and developers. → How do you think about that ?

- Is it really important to interact other system / env . ?

Example : SQL JOINs and aggregation

```
SELECT
    city,
    SUM(annual_spend) Total_Spend,
    AVG(annual_spend) Average_Spend,
    MAX(annual_spend) Max_Spend,
    COUNT(annual_spend) customers
FROM (
    SELECT t1.city, customer.annual_spend
    FROM customer
    LEFT JOIN (
        SELECT address.address_id, city.city,
              address.customer_id, address.location
        FROM address LEFT JOIN city
        ON address.city_id = city.city_id
        ) AS t1
    ON
    (customer.customer_id = t1.customer_id AND
            t1.location = "home")
) AS t2
GROUP BY city;
```



Example : Complex queries fast to create, optimize, & maintain

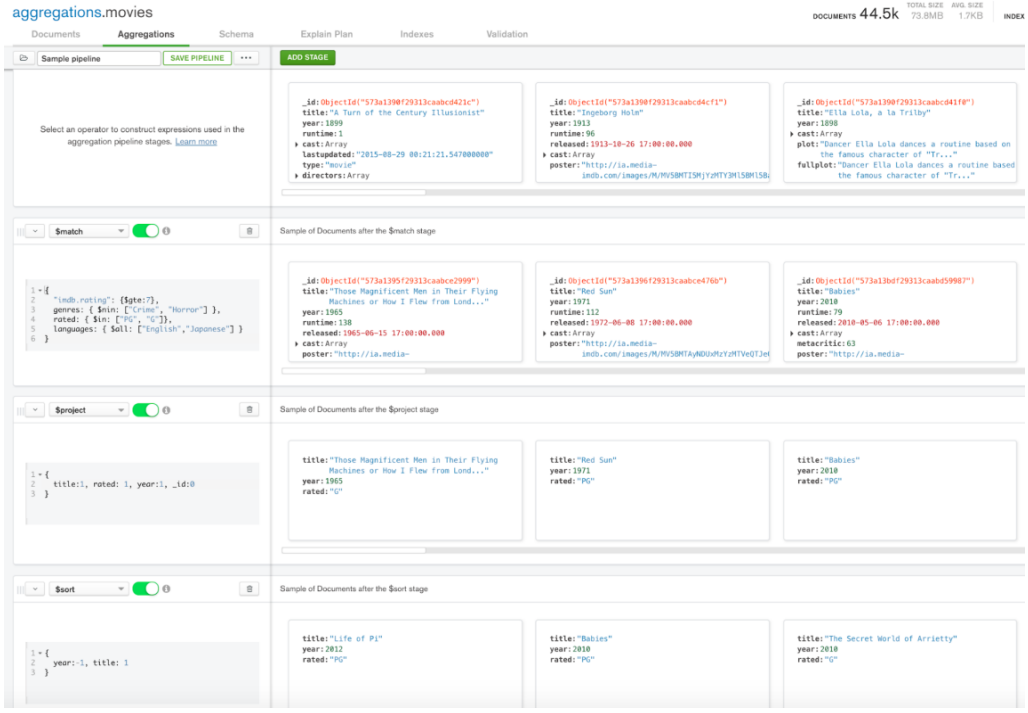
```
db.customers.aggregate([
        {
                $unwind: "$address",
        },
        {
                $match: {"address.location": "home"}
        },
        {
                $group: {
                        _id: "$address.city",
                        totalSpend: {$sum: "$annualSpend"},
                        averageSpend: {$avg: "$annualSpend"},
                        maximumSpend: {$max: "$annualSpend"},
                        customers: {$sum: 1}
                }
        }
])
```

These “phases” are distinct and easy to understand

They can be thought about in order… no nesting.

MongoDB’s aggregation framework has the flexibility you need to get value from your data, but without the complexity and fragility of SQL

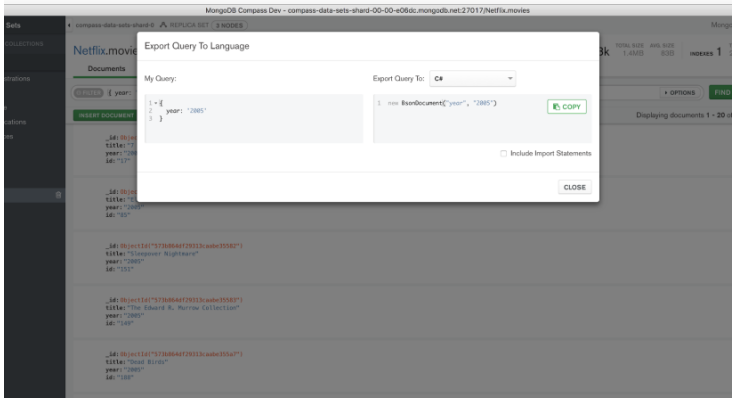
Tooling example



- Easily build aggregation pipelines for data processing and analytics
- Add new pipeline stages with code skeletons and auto-completion
- See a preview of documents as they are transformed in each stage
- Drag and drop to re-order pipeline stages
- Save pipelines in Compass or export to native code

Introduced in MongoDB Compass 1.14

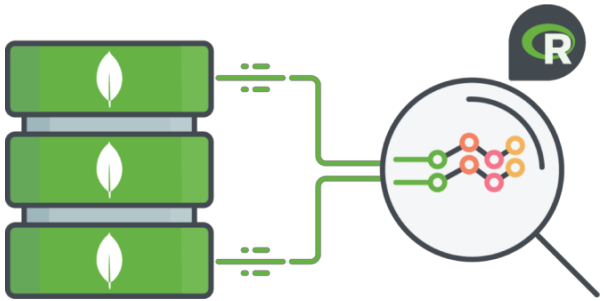
Tooling Example : Export to Language



- Simplify query creation: build a query in MongoDB Compass, then click to export it to native code
- Supports complex, multi-parameter queries and aggregation pipelines
- Initially supported: C#/.NET, Java, JavaScript, Python, and mongo shell syntax
  - More languages to come in future releases

Introduced in MongoDB Compass 1.15

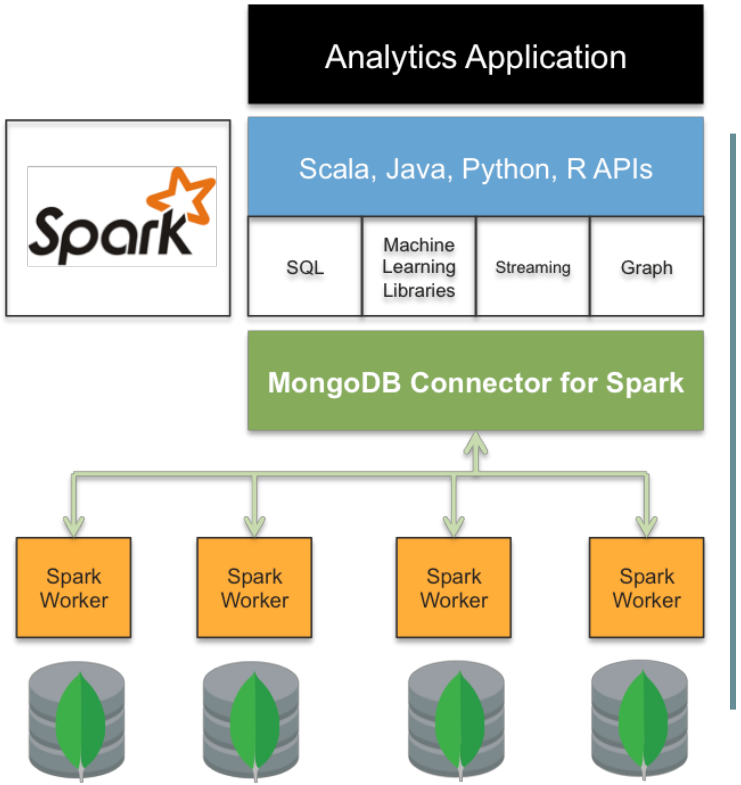
R Driver for MongoDB



Recommended MongoDB R driver for data scientists, developers & statisticians

- **Idiomatic,** native language access to the database
- **MongoDB read & write concerns** to control data consistency & durability
- **Data security** with enterprise authentication mechanisms
- **Advanced BSON data types,** e.g., Decimal 128 for high precision scientific & financial analysis

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

"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

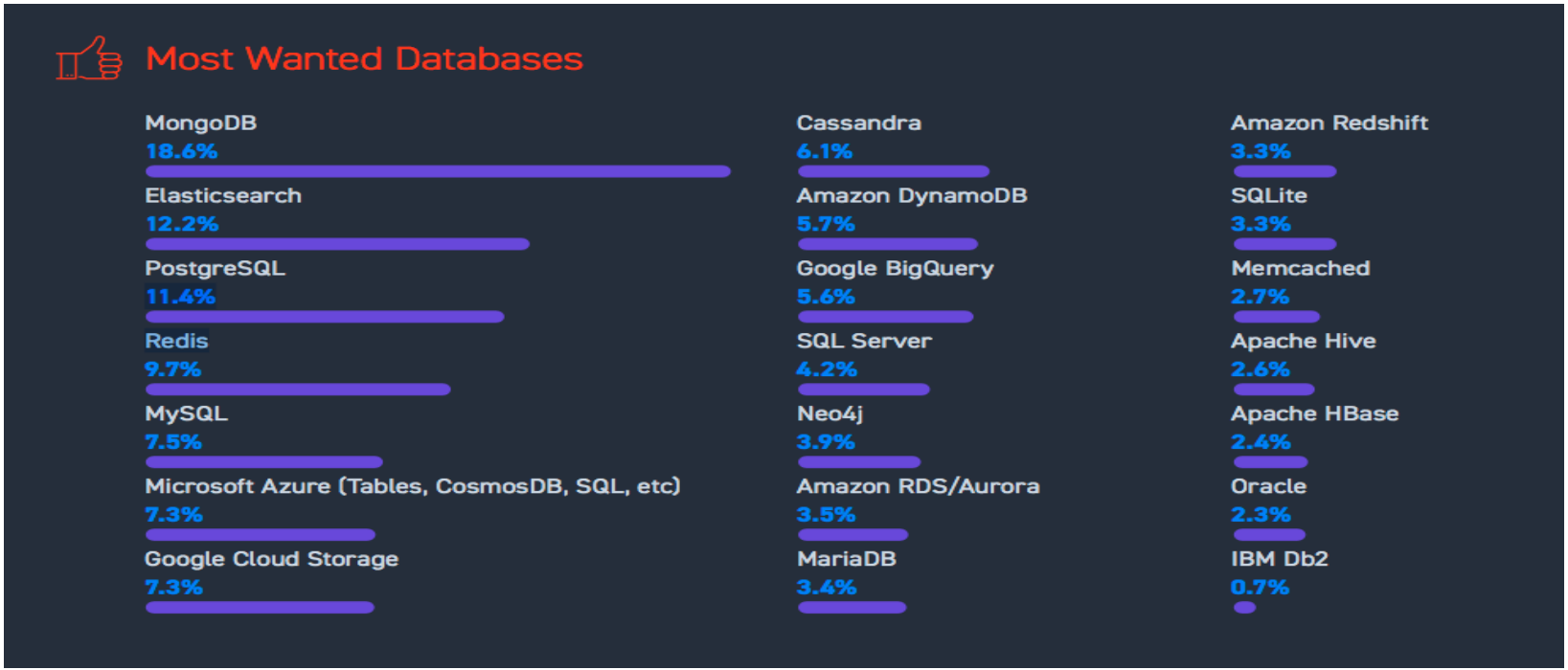
Q5 : Commercial Support, Community Strength, Freedom from Lock-In

- **Community size and commercial strength** is an important part of evaluating non-relational databases.

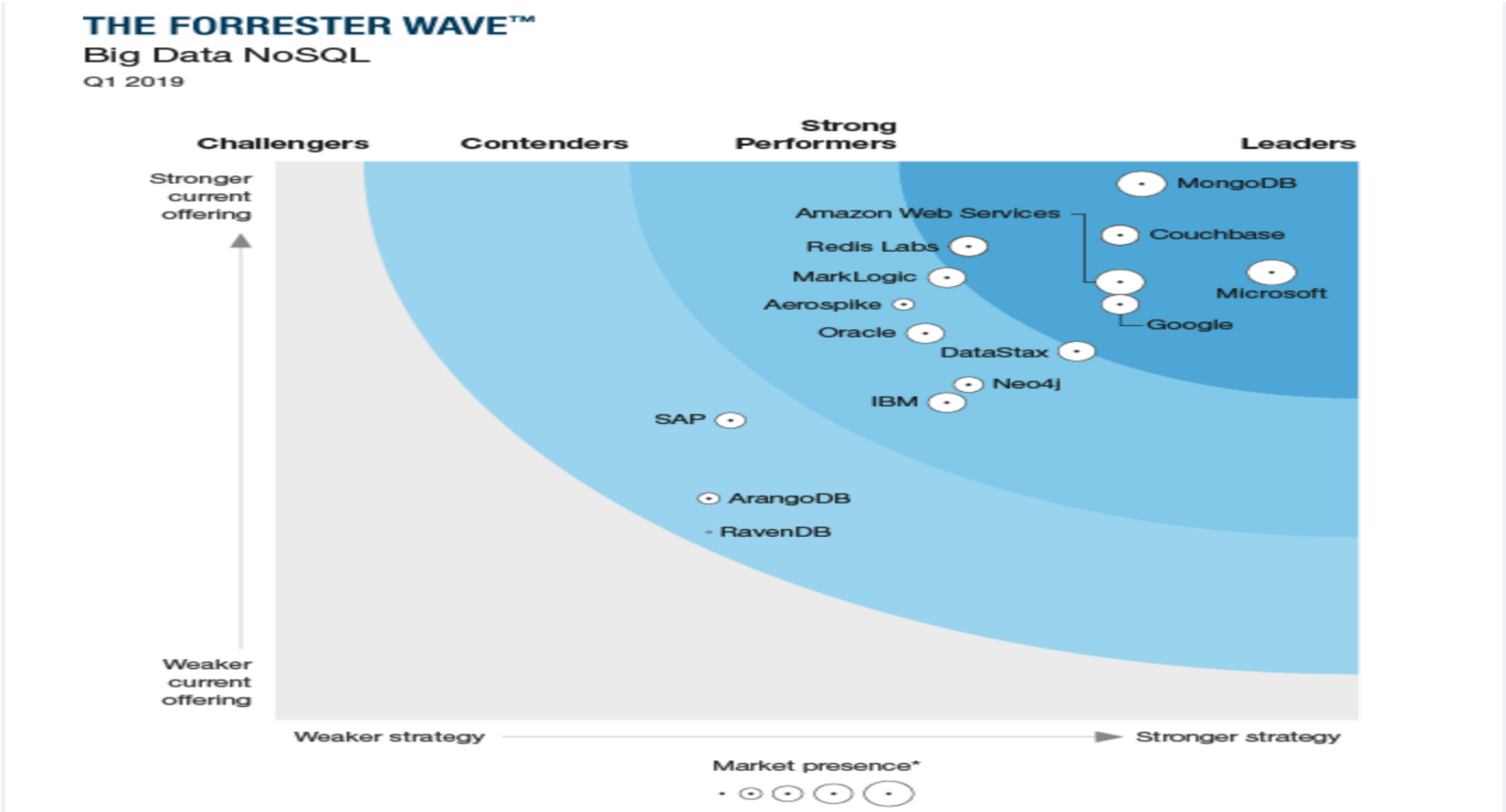
- MongoDB is one of the very few non-relational database providers to be a publicly traded company; it has the largest and most active community; support teams spread across the world providing 24x7 coverage; user-groups in most major cities; and extensive documentation.

- MongoDB is **available** to run on your own infrastructure, or as a fully managed cloud service on all of the leading public cloud platforms.

What do you mean ? Stack Overflow Developer Survey 2018 , 2019



The Forrester Wave™: Big Data NoSQL, Q1 2019



mongoDB.

The Largest Ecosystem

45,000,000+
MongoDB Downloads

850,000+
Online Education Students

100,000+
MongoDB Atlas Clusters

55,000+
MongoDB User Group Members

1,000+
Technology and Services Partners

5,700+
Customers Across All Industries

DB-Engines.com

346 systems in ranking, October 2018

Rank Oct 2018	Rank Sep 2018	Rank Oct 2017	DBMS	Database Model	Score Oct 2018	Score Sep 2018	Score Oct 2017
1.	1.	1.	Oracle	Relational DBMS	1319.27	+10.15	-29.54
2.	2.	2.	MySQL	Relational DBMS	1178.12	-2.36	-120.71
3.	3.	3.	Microsoft SQL Server	Relational DBMS	1058.33	+7.05	-151.99
4.	4.	4.	PostgreSQL	Relational DBMS	419.39	+12.97	+46.12
5.	5.	5.	MongoDB	Document store	363.19	+4.39	+33.79
6.	6.	6.	DB2	Relational DBMS	179.69	-1.38	-14.90
7.	↑ 8.	↑ 9.	Redis	Key-value store	145.29	+4.35	+23.24
8.	↓ 7.	↑ 10.	Elasticsearch	Search engine	142.33	-0.28	+22.09
9.	9.	↓ 7.	Microsoft Access	Relational DBMS	136.80	+3.41	+7.35
10.	10.	↓ 8.	Cassandra	Wide column store	123.39	+3.83	-1.40

Method of calculating the scores of the DB-Engines Ranking

The DB-Engines Ranking is a list of database management systems ranked by their current popularity. We measure the popularity of a system by using the following parameters:

- **Number of mentions of the system on websites**, measured as number of results in search engines queries. At the moment, we use Google, Bing and Yandex for this measurement. In order to count only relevant results, we are searching for <system name> together with the term database, e.g. "Oracle" and "database".
- **General interest in the system.** For this measurement, we use the frequency of searches in Google Trends.
- **Frequency of technical discussions about the system.** We use the number of related questions and the number of interested users on the well-known IT-related Q&A sites Stack Overflow and DBA Stack Exchange.
- **Number of job offers, in which the system is mentioned.** We use the number of offers on the leading job search engines Indeed and Simply Hired.
- **Number of profiles in professional networks, in which the system is mentioned.** We use the internationally most popular professional networks LinkedIn and Upwork.
- **Relevance in social networks.** We count the number of Twitter tweets, in which the system is mentioned.

Why bet on us?

Ease and flexibility of the document model led to massive early adoption. We are the **world's most popular modern database**.

First database company to go public in over 20 years. We've raised **over a half a billion dollars** to invest in our business.

Uniquely positioned as the **only company who can credibly offer a modern "general purpose" database**.

mongoDB

Appendix : MongoDB Atlas & Proof of Points

MongoDB in Database World

Intelligent Operational Data Platform

Best way to
work with data

Intelligently put data
where you need it

Freedom
to run anywhere

MongoDB Atlas — 글로벌 클라우드 매니즈드 데이터베이스 서비스

Self-service & elastic - 필요시 배포, 수정 및 업그레이드는 베스트 Practice기반으로 운영 자동화 제공 - 자동화 된 데이터베이스 유지 관리 몇 번의 클릭 또는 API 호출로 확대, 축소 또는 축소 구성의 자동화	**Global & cloud-agnostic** - AWS 여러 지역에서 사용 가능 - 어디에서나 읽기 / 쓰기가 가능한 글로벌 클러스터를 통한 배포 및 다중 리젼을 통한 고가용성 제공 - 클라우드 환경 기반에서 손쉽게 마이그레이션 가능	**Enterprise-grade security & SLAs** - 네트워크 격리, VPC 피어링, 종단 간 암호화 및 역할 기반 액세스 제어 - 암호화 키 관리, LDAP 통합, 상세 데이터베이스 감사 - SOC 2 / 개인 정보 보호/ HIPAA SLA로 안정성 보장
Comprehensive monitoring - 전 100 개 이상의 메트릭을 통한 KPI에 대한 alerting - 실시간 성능 조언 및 성능 관리자 - 모니터링 대시 보드와 통합 할 수 있는 API	**Managed backup** - 특정 시점의 데이터 복구 - 쿼리 가능한 백업 스냅 샷 - 샤딩환경내 일관된 스냅 샷 - 클라우드 데이터 이동성	**Stitch: Serverless platform services** - 백엔드 로직, 서비스 통합 및 API를위한 단순한 서버리스 기능 - 간단한 필드 레벨 액세스 규칙으로 보호되는 프론트 엔드에서의 데이터베이스 액세스 - 변경 사항에 실시간으로 반응하는 데이터베이스 및 인증 트리거

MongoDB Atlas를 사용할 수 있는 AWS Region

MongoDB Atlas is currently available in 16 AWS regions.

Create a **free tier cluster** by selecting a region with FREE TIER AVAILABLE and choosing the **M0** cluster tier below.

★ recommended region

NORTH AMERICA

- **N. Virginia** (us-east-1) ★ FREE TIER AVAILABLE
- **Ohio** (us-east-2) ★
- **N. California** (us-west-1)
- **Oregon** (us-west-2) ★ FREE TIER AVAILABLE
- **Montreal** (ca-central-1)

EUROPE

- **Stockholm** (eu-north-1) ★
- **Ireland** (eu-west-1) ★ FREE TIER AVAILABLE
- **London** (eu-west-2) ★
- **Paris** (eu-west-3) ★
- **Frankfurt** (eu-central-1) ★ FREE TIER AVAILABLE

SOUTH AMERICA

- **Sao Paulo** (sa-east-1)

AUSTRALIA

- **Sydney** (ap-southeast-2) ★

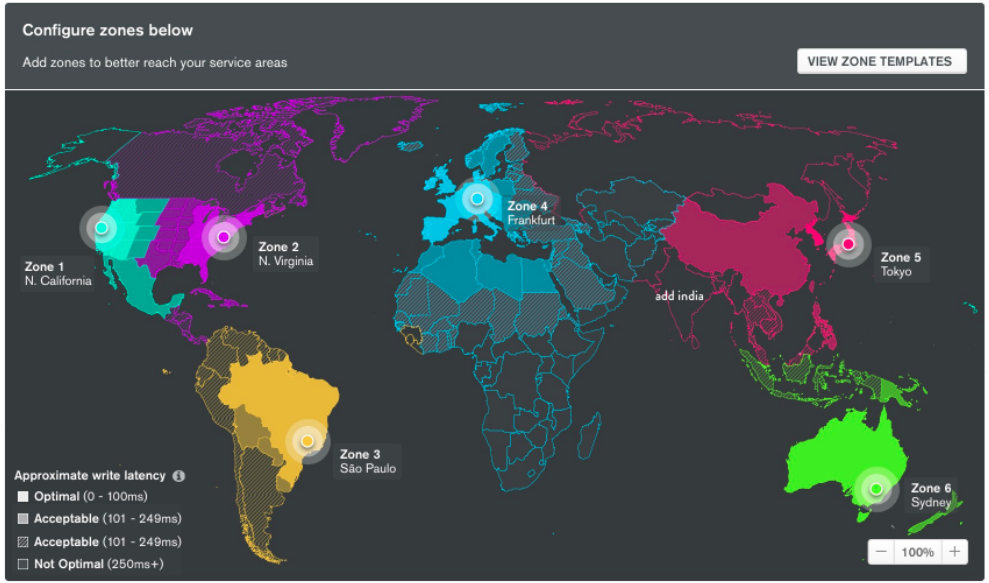
ASIA

- **Tokyo** (ap-northeast-1) ★
- **Seoul** (ap-northeast-2)
- **Singapore** (ap-southeast-1) ★ FREE TIER AVAILABLE
- **Mumbai** (ap-south-1) FREE TIER AVAILABLE

글로벌 클러스터

하나 이상의 클라우드 영역으로 구성되고, 지리적으로 분산 된 여러 영역에 걸쳐 완전 자동화 된 데이터베이스 배포

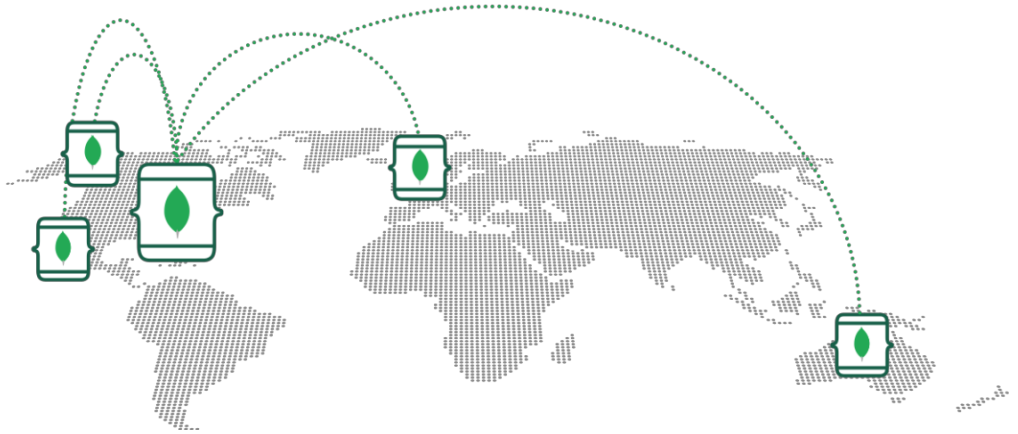
- 분산 응용 프로그램에 대해 한 자리 수 (밀리 초) 대기 시간을 제공하기 위해 로컬로 읽고 쓰기
- 사전 제작 된 영역 템플릿을 사용하여 손쉽게 배포하거나 사용하기 쉬운 시각적 인터페이스에서 클라우드 영역을 선택하여 자체 영역을 구축하십시오.



mongoDB.

다중 **Region**을 통한 **replication set** 구성

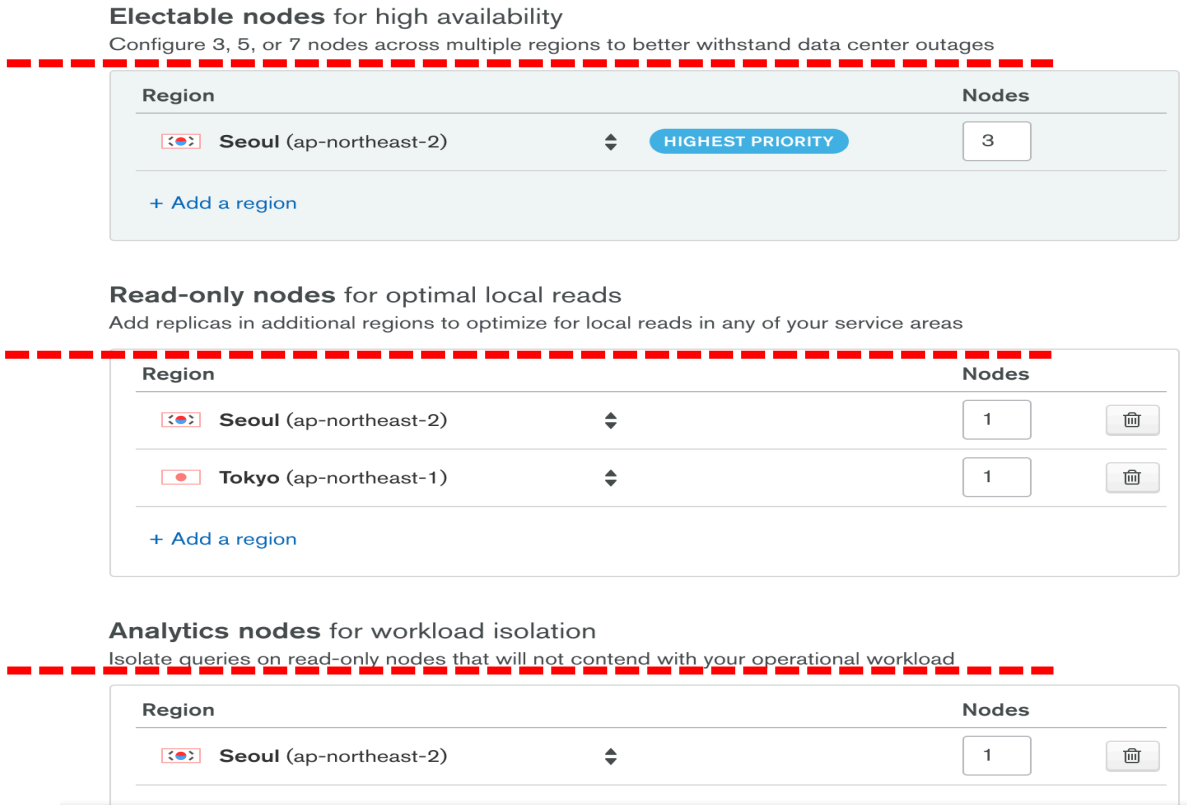
- 드물게 발생할 수 있는 전체 지역 장애에 대해 다중 리전 구성으로 고 가용성을 보장하십시오.
- Voting에 참여하지 않는 읽기 전용 노드 구성을 통해 (지리적으로 배포) 로컬 읽기 성능 향상할 수 있습니다.
- AWS Region 기반에서 구성,배포 가능 합니다.



mongoDB.

다중 **Region**을 통한 **replication set** 구성

- Electable Node에 대한 multi region 구성으로 single region에 대한 outage를 대비 할 수 있다.
- Read Only Node를 통해 (Local Node) 구성을 통해 latency 문제를 해결 할 수 있다.
- Analytic Node를 통해 (Local Node) 구성을 통해 latency 문제를 해결 할 수 있다.



mongoDB.

실시간 성능 모니터링

실시간 성능 패널은 클러스터에서 발생한 상황을 실시간으로 확인하고 문제를 신속하게 진단을 통해 탁월한 통찰력을 데이터베이스에 제공합니다.

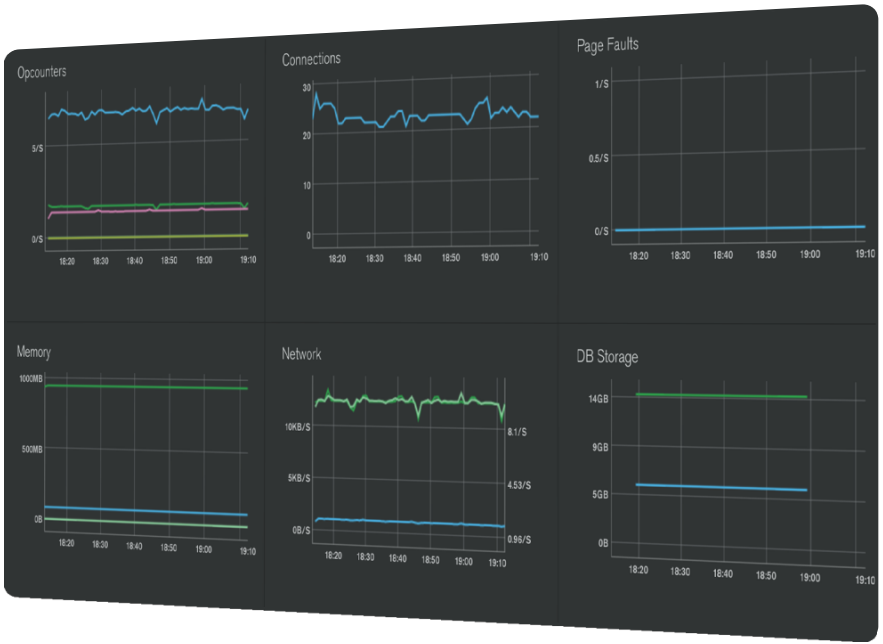
디스플레이 항목

- Operations
- Read/Writes
- Network In/Out
- Memory
- Hottest Collections
- Slowest Operations

mongoDB.

Fine-grained 모티터링 **&** 경보

- 모니터링 및 경고(alerting)는 클러스터 데이터베이스 및 서버 사용 상태에 대한 전체 메트릭을 제공합니다.
- 데이터베이스 작업 또는 서버 사용량이 클러스터 성능에 영향을 미치는 정의 된 임계 값에 도달하면 자동 알림
- 자동화 된 경고 기능과 MongoDB Atlas의 유연한 확장 및 축소 옵션을 결합하여 데이터베이스 지원 응용 프로그램을 항상 수행 할 수 있습니다.



Fine-grained 모티터링 & 경보

TOGGLE CHARTS

MongoDB Metrics

- Asserts
- Cache Activity
- Cache Usage
- Connections
- Cursors
+ DB Storage
- Document Metrics
+ Memory
- Network
- Opcounters
+ Opcounters - Repl
- Operation Execution Times
+ Oplog GB/Hour
- Page Faults
+ Query Executor
- Query Targeting
+ Queues
+ Replication Headroom
+ Replication Lag
+ Replication Oplog Window
- Scan And Order
- Tickets Available

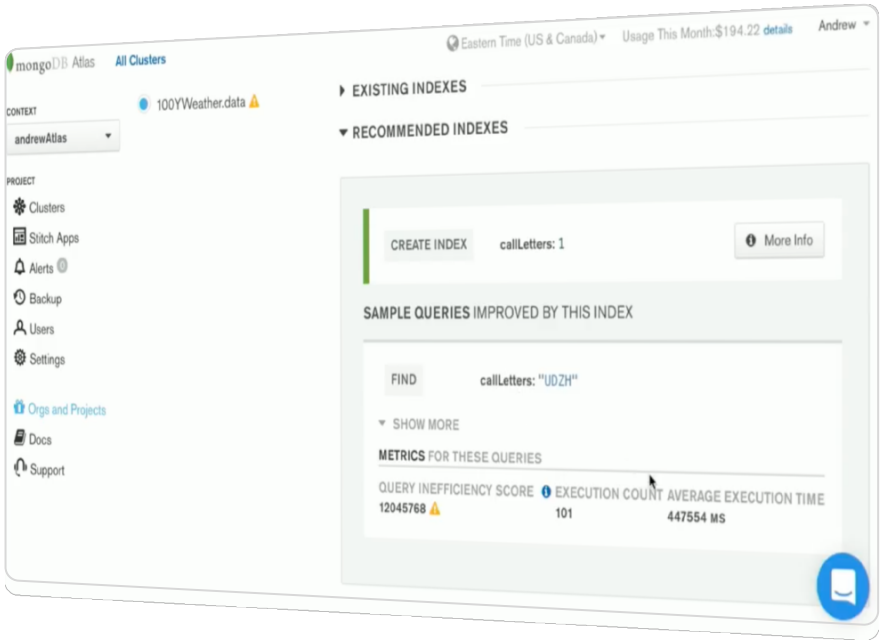
Hardware Metrics

+ Disk IOPS
+ Disk Latency
+ Disk Space Free
+ Disk Space Percent Free
+ Disk Space Used
+ Normalized Process CPU
+ Normalized System CPU
+ Process CPU
+ System CPU
- Util %

성능 어드바이저

MongoDB Atlas 클러스터에서 사용할 수 있는 Always-on Performance Advisor는 성능 향상에 도움이 되는 자동화 된 권장 사항을 제공합니다.

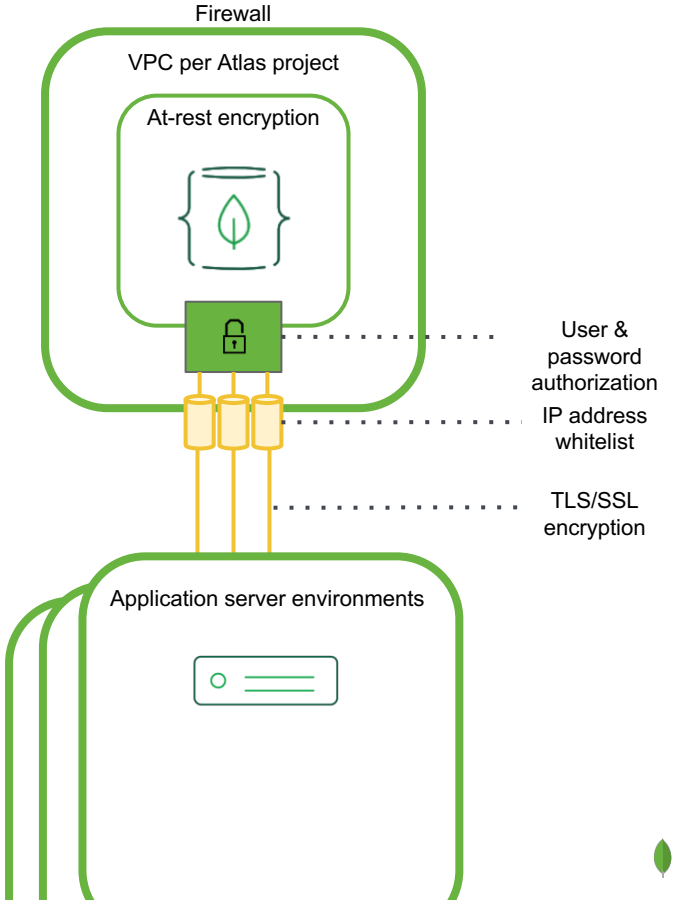
- 느리게 실행되는 쿼리에 대한 관련 통계를 보고 자동화 된 인덱스 제안 수신
- 클러스터의 기존 인덱스 보기
- The Performance Advisor는 성능 오버 헤드없이 언제든지 실행할 수 있습니다.



mongoDB.

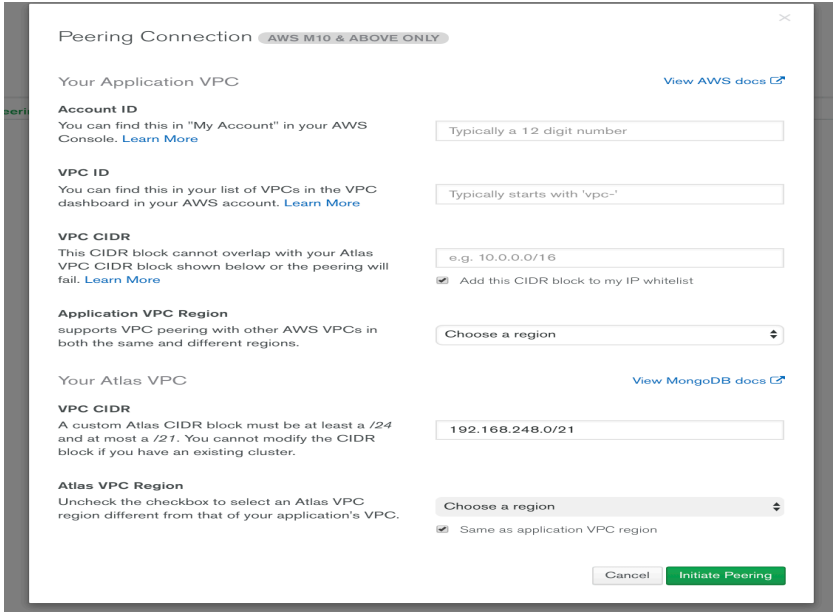
보안 : 고객 전용 VPC

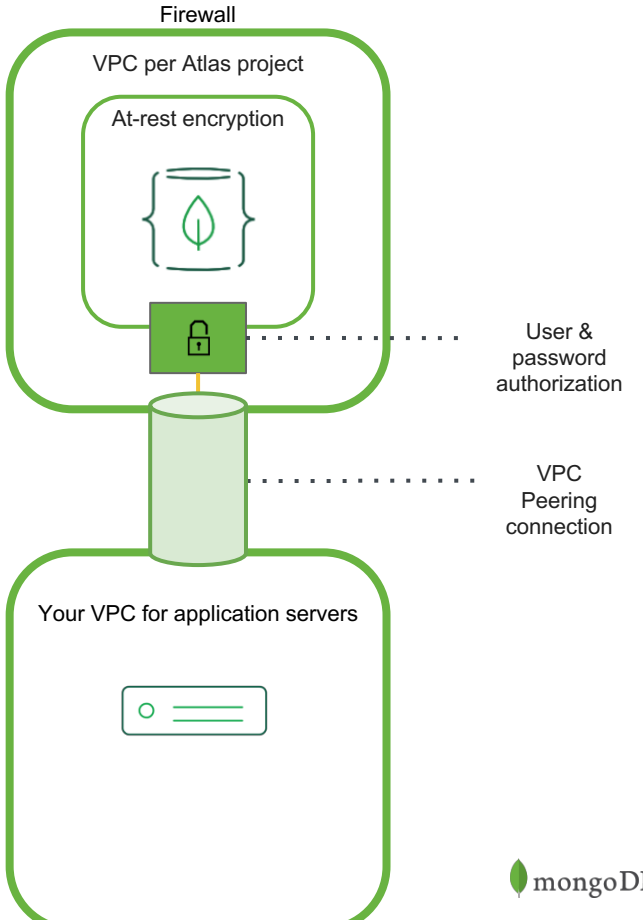
- 네트워크는 기본적으로 외부 와 연결 되어 있지 않습니다.
- IP 주소는 인바운드 네트워크 트래픽을 위해 명시적으로 허용 목록에 있어야합니다.
- 데이터 암호화 (in- flight) - 모든 네트워크 트래픽에 사용되는 TLS / SSL
- 데이터베이스에 연결하는 데 필요한 사용자 이름과 암호. 구성 가능한 권한
- 암호화 된 저장소(at rest) 볼륨에서의 데이터 암호화



보안 : VPC Peering

- **VPC Peering —** Atlas 클러스터가 포함 된 VPC를 사설 네트워크를 통해 연결하는 응용 프로그램 서버가 포함 된 VPC에 피어 투 피하십시오.





mongoDB.