|  |
| --- |
| TUTORIAL  MONGODB SHELL &  MONGODB - ACCESS |
|  |
| BNI  DGL – CLS  2022 |

Table of Contents

[Pengantar 3](#_Toc116652493)

[Fitur MongoDB 4](#_Toc116652494)

[MongoDB (mongosh) 5](#_Toc116652495)

[Helpers 5](#_Toc116652496)

[Show Databases 5](#_Toc116652497)

[Switch Database 5](#_Toc116652498)

[Show Collections 5](#_Toc116652499)

[Run JavaScript File 5](#_Toc116652500)

[CRUD 6](#_Toc116652501)

[Create 6](#_Toc116652502)

[Read 6](#_Toc116652503)

[Update 9](#_Toc116652504)

[Delete 10](#_Toc116652505)

[Databases and Collections Drop 11](#_Toc116652506)

[Other Collection Functions 11](#_Toc116652507)

[MongoDB Access 11](#_Toc116652508)

[Requirements 11](#_Toc116652509)

[Usage 11](#_Toc116652510)

[Operations 12](#_Toc116652511)

[list 13](#_Toc116652512)

[insert\_one 13](#_Toc116652513)

[insert\_many 14](#_Toc116652514)

[update\_one 14](#_Toc116652515)

[update\_many 14](#_Toc116652516)

[find\_one 15](#_Toc116652517)

[find\_many 15](#_Toc116652518)

[delete\_one 16](#_Toc116652519)

[delete\_many 16](#_Toc116652520)

# Pengantar

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| NoSQL adalah teknologi database baru yang menyediakan struktur data non-relasional. Dalam dokumen berikut ini, faktor kunci untuk memulai NoSQL dengan wadah database MongoDB akan dieksplorasi. Perangkat lunak lain juga akan digunakan bersamaan dengan MongoDB. Yaitu, MongoDBCompas, GUI resmi MongoDB. Di dalam MongoDBCompass akan dipergunakan fitur MongoDB Shell (mongosh) yaitu sebuah *environment* untuk berinteraksi dengan MongoDB. Dengan fitur ini digunakan untuk menguji *query* dan operasi secara langsung dengan database.    Figure 1 Fitur mongosh pada MongoDBCompas  Untuk membantu dalam *operations* pada mongoDB maka dibuat service mongoDB Access. Didalam mongoDB Acces data-data akan terenkripsi sebelum dikirim , sehingga lintas data terjamin aman. Aturan request pada mongoDB Acces akan dibahas pada dokumen ini.    Figure 2. Tampilan swagger UI di MongoDB Acces Fitur MongoDB  1. Replikasi: Fitur kumpulan replika MongoDB dikenal menyediakan ketersediaan tinggi. Dua atau lebih salinan data merupakan satu set replika. Sebuah replika-set bertindak sebagai replika primer atau sekunder. Replika sekunder menyimpan salinan data primer, menjaganya secara teratur, sebagai bagian dari sistem MongoDB yang direplikasi. Setiap kali replika utama berhenti, kumpulan replika secara otomatis menentukan sekunder mana yang harus menjadi primer dan melakukan pemilihan jika perlu. Replika sekunder juga dapat melayani operasi baca, tetapi data pada akhirnya hanya konsisten secara default. Untuk menyelesaikan pemilihan primer baru, tiga server mandiri harus ditambahkan sebagai server sekunder. 2. Pengindeksan: Bidang MongoDB dapat diindeks dengan indeks atau indeks primer dan sekunder. Indeks MongoDB menyimpan sebagian kecil dari kumpulan data dalam bentuk yang nyaman untuk dilintasi. Indeks menyimpan nilai bidang tertentu, atau kumpulan bidang, yang diurutkan berdasarkan nilainya. Di MongoDB, indeks membantu menyelesaikan *query* secara efisien dengan menyimpan sebagian kecil kumpulan data dalam bentuk yang ramah. Indeks MongoDB mirip dengan indeks basis data relasional yang khas. 3. Penyimpanan file: GridFS, yang menggunakan MongoDB sebagai sistem file, dapat digunakan untuk menyeimbangkan dan mereplikasi data di beberapa mesin. Sebuah file dapat disimpan di MongoDB sebagai sistem file grid. Ini memiliki fitur yang mirip dengan sistem file seperti load balancing dan replikasi data. 4. Agregasi: Agregasi pipeline. Menurut dokumentasi MongoDB, Agregasi Pipeline memberikan kinerja yang lebih baik untuk sebagian besar operasi agregasi. Kerangka kerja agregasi mencakup $lookup dan standar deviasi seperti operator statistik. 5. Sharding: Sharding adalah pemisahan data antar mesin. Sharding MongoDB memungkinkan untuk membagi koleksi di antara banyak mesin (pecahan), memungkinkannya untuk tumbuh melampaui batasan sumber daya.  MongoDB (mongosh) Ketikkan perintah-perintah dibawah ini pada menu **mongosh** di MongoDB Compass. HelpersShow Databases show dbs  db // prints the current database Switch Database use <database\_name> Show Collections show collections Run JavaScript File load("myScript.js") //if your file js exist CRUD Sebelumnya buat dahulu database **dbtest** di mongoDBCompas sebagai *dummy* pengetesan. Lalu pindah cursor database ke **dbtest** dengan mengetikan *command* berikut pada mongosh  use dbtest  Format umum dari syntax mongo shell biasanya seperti dibawah ini.  <*database name*>.<*collection name*>.<*operation name*>(<*content array*>) Create db.docs.insertOne({name: "Max"})  db.docs.insert([{name: "Max"}, {name:"Alex"}]) // ordered bulk insert  db.docs.insert([{name: "Max"}, {name:"Alex"}], {ordered: false}) // unordered bulk insert  db.docs.insert({date: ISODate()})  db.docs.insert({name: "Max"}, {"writeConcern": {"w": "majority", "wtimeout": 5000}}) ReadCommon db.docs.findOne() // returns a single document  db.docs.find() // returns a cursor - show 20 results - "it" to display more  db.docs.find().pretty()  db.docs.find({name: "Max", age: 32}) // implicit logical "AND".  db.docs.find({date: ISODate("2020-09-25T13:57:17.180Z")})  db.docs.distinct("name") // remove duplicate name Count db.docs.count({age: 32}) // estimation based on docsection metadata  db.docs.estimatedDocumentCount() // estimation based on docsection metadata  db.docs.countDocuments({age: 32}) // alias for an aggregation pipeline - accurate count Comparison db.docs.find({"year": {$gt: 1970}}) // greater than  db.docs.find({"year": {$gte: 1970}}) // greater than or equal  db.docs.find({"year": {$lt: 1970}}) // lower than  db.docs.find({"year": {$lte: 1970}}) // lower than or equal  db.docs.find({"year": {$ne: 1970}}) // not equal  db.docs.find({"year": {$in: [1958, 1959]}}) // value of a field equals any value in the specified array  db.docs.find({"year": {$nin: [1958, 1959]}}) // selects the documents whose field holds an array with no element equal to a value in the specified array Logical db.docs.find({name:{$not: {$eq: "Max"}}})  db.docs.find({$or: [{"year" : 1958}, {"year" : 1959}]})  db.docs.find({$nor: [{price: 1.99}, {sale: true}]})  db.docs.find({  $and: [  {$or: [{qty: {$lt :10}}, {qty :{$gt: 50}}]},  {$or: [{sale: true}, {price: {$lt: 5 }}]}  ]  }) Element db.docs.find({name: {$exists: true}})  db.docs.find({"zipCode": {$type: 2 }})  db.docs.find({"zipCode": {$type: "string"}})  // Aggregation Pipeline  db.docs.aggregate([  {$match: {status: "A"}},  {$group: {\_id: "$cust\_id", total: {$sum: "$amount"}}},  {$sort: {total: -1}}  ]) Text search with a "text" index db.docs.find({$text: {$search: "cake"}}, {score: {$meta: "textScore"}}).sort({score: {$meta: "textScore"}}) Regex db.docs.find({name: /^Max/}) // regex: starts by letter "M"  db.docs.find({name: /^Max$/i}) // regex case insensitive Array db.docs.find({tags: {$all: ["Realm", "Charts"]}})  db.docs.find({field: {$size: 2}}) // impossible to index - prefer storing the size of the array & update it  db.docs.find({results: {$elemMatch: {product: "xyz", score: {$gte: 8}}}}) Projections db.docs.find({"x": 1}, {"actors": 1}) // actors only + \_id  db.docs.find({"x": 1}, {"actors": 1, "\_id": 0}) // actors only  db.docs.find({"x": 1}, {"actors": 0, "summary": 0}) // all but except "actors" and "summary" field Sort, skip, limit db.docs.find({}).sort({"year": 1, "rating": -1}).skip(10).limit(3) Read Concern db.docs.find().readConcern("majority") UpdateCommon db.docs.update({"\_id": 1}, {"year": 2016}) // WARNING! Replaces the entire document  db.docs.update({"\_id": 1}, {$set: {"year": 2016, name: "Max"}})  db.docs.update({"\_id": 1}, {$unset: {"year": 1}})  db.docs.update({"\_id": 1}, {$rename: {"year": "date"} })  db.docs.update({"\_id": 1}, {$inc: {"year": 5}})  db.docs.update({"\_id": 1}, {$mul: {price: NumberDecimal("1.25"), qty: 2}})  db.docs.update({"\_id": 1}, {$min: {"imdb": 5}})  db.docs.update({"\_id": 1}, {$max: {"imdb": 8}})  db.docs.update({"\_id": 1}, {$currentDate: {"lastModified": true}})  db.docs.update({"\_id": 1}, {$currentDate: {"lastModified": {$type: "timestamp"}}}) Array db.docs.update({"\_id": 1}, {$push :{"array": 1}})  db.docs.update({"\_id": 1}, {$pull :{"array": 1}})  db.docs.update({"\_id": 1}, {$addToSet :{"array": 2}})  db.docs.update({"\_id": 1}, {$pop: {"array": 1}}) // last element  db.docs.update({"\_id": 1}, {$pop: {"array": -1}}) // first element  db.docs.update({"\_id": 1}, {$pullAll: {"array" :[3, 4, 5]}})  db.docs.update({"\_id": 1}, {$push: {scores: {$each: [90, 92, 85]}}})  db.docs.updateOne({"\_id": 1, "grades": 80}, {$set: {"grades.$": 82}})  db.docs.updateMany({}, {$inc: {"grades.$[]": 10}})  db.docs.update({}, {$set: {"grades.$[element]": 100}}, {multi: true, arrayFilters: [{"element": {$gte: 100}}]}) Update many db.docs.update({"year": 1999}, {$set: {"decade": "90's"}}, {"multi":true})  db.docs.updateMany({"year": 1999}, {$set: {"decade": "90's"}}) FindOneAndUpdate db.docs.findOneAndUpdate({"name": "Max"}, {$inc: {"points": 5}}, {returnNewDocument: true})  *Upsert*  db.docs.update({"\_id": 1}, {$set: {item: "apple"}, $setOnInsert: {defaultQty: 100}}, {upsert: true})  *Replace*  db.docs.replaceOne({"name": "Max"}, {"firstname": "Maxime", "surname": "Beugnet"})  *Save*  db.docs.save({"item": "book", "qty": 40})  *Write concern*  db.docs.update({}, {$set: {"x": 1}}, {"writeConcern": {"w": "majority", "wtimeout": 5000}}) Delete db.docs.remove({name: "Max"})  db.docs.remove({name: "Max"}, {justOne: true})  db.docs.remove({}) // WARNING! Deletes all the docs but not the docsection itself and its index definitions  db.docs.remove({name: "Max"}, {"writeConcern": {"w": "majority", "wtimeout": 5000}})  db.docs.findOneAndDelete({"name": "Max"}) Databases and Collections Drop db.docs.drop() // removes the collection and its index definitions  db.dropDatabase() // Warning this delete database Other Collection Functions db.docs.stats()  db.docs.storageSize()  db.docs.totalIndexSize()  db.docs.totalSize()  db.docs.validate({full: true})  db.docs.renameCollection("new\_coll", true) // 2nd parameter to drop the target collection if exists MongoDB AccessRequirements Berikut beberapa library yang harus di install pada aplikasi python:   * Cryptocode (mandatory) * Requests (optional bisa disesuaikan dengan library yang lain)   Kemudian untuk crytocode dibutuhkan kunci password hash sebagai dasar hashing string karakter. Usage Berikut syntax sederhana cara menggunakan mongoDB Acces service :  import cryptocode  import requests  password = 'PASSWORD\_CRYPTO\_FROM\_DB\_ACCESS'  db\_name = 'db'  query\_array = {  "collections" : "YOUR\_COLLECTION",  "operations" : "find\_one",  "doc\_array" : {"KOTA" : "KOTA MEDAN"}  }  query\_str = json.dumps(query\_array)  encodedQuery = cryptocode.encrypt(query\_str, password)  encodedDbname = cryptocode.encrypt(db\_name, password)  # encodedQuery = 'nI5Ft4C2L+zRj1jlqB84qEj18T+4+RBqJz4WQotuxKWaeI8QyDEgfraAqrmUAiRtC+D2tOtC6JUPr9Oik/cKp2G+xAjZ70U7OeZhqybS1hETCSOWw0pDaw==\*siLhoV2yOzfPncIzEm1sKg==\*PEj6Xu8vtETTzFT6yDfnZA==\*vZkOyZbxFC4nbvwy06i6MQ=='  # encodedDbname = 'mUImPJD3t+Fl\*U8xs63kfYk+15+ff1wbt+Q==\*bW7XqZja3zGIn1QHSRED3A==\*UPiT7SSHf8w8zH1PLl1IFQ=='  request\_array\_body = {  "SERVER" : "(INTERNAL/GKE)"  "CHANNEL\_ID" : "API",  "DB\_NAME" : encodedDbname,  "QUERY" : encodedQuery  }  # Encrypted request  response\_encrypted = requests.post(<db\_access>, data=request\_array\_body)  # Decrypted response  response\_decrypted = cryptocode.decrypt(response\_encrypted, password)  print(response\_decrypted) Operations Berikut operation-operation pada service MongoDB Access :   1. list 2. insert\_one 3. insert\_many 4. update\_one 5. update\_many 6. find\_one 7. find\_many 8. delete\_one 9. delete\_many 10. pipeline  list  |  |  | | --- | --- | | Normal | Encryption | | {  "SERVER": "GKE",  "CHANNEL\_ID": "API",  "DB\_NAME": "dbtest",  "QUERY": {  "collections": "",  "operations": "list",  "doc\_array": ""  }  } | {  "SERVER": "GKE",  "CHANNEL\_ID": "API",  "DB\_NAME": "cbeqkrs9BA==\*AHdtjZ93Iy586fOQa3i2BA==\*xlEXA6ldEaWPWedjGKhkUQ==\*4zDquU/YLvkh1KOsnskfHQ==",  "QUERY": "hk6BMOzZAhukfSvpTm5g4saC8DJ+0+ArVt1HeCeFjnKor6rvccutpuiVfXskzd1VVaOd04ehdOUWHQ==\*kQlsl+REbM+2STlQ4F+xaw==\*ZCjmV/WqXBjFfkAuJWHyFQ==\*zDzB0uDXdrXDg223dG43aw=="  } |  insert\_one  |  |  | | --- | --- | | Normal | Encryption | | {  "SERVER": "GKE",  "CHANNEL\_ID": "API",  "DB\_NAME": "dbtest",  "QUERY": {  "collections": "docs",  "operations": "insert\_one",  "doc\_array": {"name": "Max"}  }  } | {  "SERVER": "GKE",  "CHANNEL\_ID": "API",  "DB\_NAME": "7jeBt2Ur\*VMGeQlhEboE56AJc2kQ9IA==\*0trvuxDDkxbj+wXBZhXhjA==\*gpllAjktCE3I0mIF4HBgFg==",  "QUERY": "ebHYnioFDonvJUZ8hqM6PSHKC0l2YfM2jNJ25szZnU5UGKFuFLdYxXvwT8OssXKm5/QbCFtjI+P+SR+nzyIc/Fn7nb7vKvijiX05DzeYGcoH\*jxPumRnEBe/2engxIZmraA==\*nOXN5fAoCSTs3NDWVUr8mA==\*Oj/owcHt3za6afWc4t03bQ=="  } |  insert\_many  |  |  | | --- | --- | | Normal | Encryption | | {  "SERVER": "GKE",  "CHANNEL\_ID": "API",  "DB\_NAME": "dbtest",  "QUERY": {  "collections": "docs",  "operations": "insert\_many",  "doc\_array": [{"name": "Max"}, {"name":"Alex"}]  }  } | {  "SERVER": "GKE",  "CHANNEL\_ID": "API",  "DB\_NAME": "+uVc+/28\*xTGDO4H4CFzv8q5Id3wXVw==\*6Mvxg0LaKtg/KeUPC+mNBw==\*hwU0clMbZwbLBKu1OFdKdw==",  "QUERY": "v2OWLfWzJJ0q4oEdVYfdmThI298iLvQ51iH8pHzYDFQbA8+qn73SvDVI+0dCVtINASIttfEfe75M/QUD9Ef4ZT12dFQWFxF7FgGOQWih+h/APIQcz9Ew/hsHTJC8aQzQ1MEJl9He\*DOtC8OLIqWq8j09L+oZAjA==\*+uUeGomdFtt44EfykzCCAg==\*wi4mOGgYg9TYkbr3ppag9Q=="  } |  update\_one  |  |  | | --- | --- | | Normal | Encryption | | {  "SERVER": "GKE",  "CHANNEL\_ID": "API",  "DB\_NAME": "dbtest",  "QUERY": {  "collections": "docs",  "operations": "update\_one",  "doc\_array": [{"WILAYAH": "1"}, {"$set":{"KODE\_POS":"0000000000"}}]  }  } | {  "SERVER": "GKE",  "CHANNEL\_ID": "API",  "DB\_NAME": "IDeaXbzl\*+ZUHNRvRasOI8GKvvwPZFQ==\*YP/tyfcUh1otoqOlo+Imsw==\*t06jchZYfWkmYbsW/xSA7w==",  "QUERY": "77xvVeo3b3n1GLQATRB4F1l7Yb2QoUFThJpjTdcwUxlZ6lXAzUL91d8WuB7wpLUu5ZOGfOIR7D3xqwbQlBeFMV1ZaFYR3QFpAXmNsk/50YOEU8GWPI7H0KnNVmMeNiDl3HvTjhDENLmGfQGh57ppnzmGW70Vi7u4eT8=\*INqkcOgB5xMELhdNBpF4Dw==\*04iul9vdV5zVyfeXPLfaaQ==\*qaX5MpteQvsAv2m6OSGXrQ=="  } |  update\_many  |  |  | | --- | --- | | Normal | Encryption | | {  "SERVER": "GKE",  "CHANNEL\_ID": "API",  "DB\_NAME": "dbtest",  "QUERY": {  "collections": "docs",  "operations": "update\_many",  "doc\_array":[{"WILAYAH": "1"}, {"$set":{"KODE\_POS":"0000000000"}}]  }  } | {  "SERVER": "GKE",  "CHANNEL\_ID": "API",  "DB\_NAME": "nUjhXGlz\*FVm1RVzNyIalAXBVueDbSQ==\*6jiqAsQPVW7szukCKxoKOw==\*Xzop1k3mV5yYdjgEDrBVlw==",  "QUERY": "TA+Fy5tHABT6Zjv1zWgvd8hiXpXvch/QB4X9d6k426+5yRGhMcthfaK1nZtvGyiyOXAsq3d6YjjbYuJaJdppZwzpLLhudGrf/DJOJOXwKyKKHTz+fOZAUfD32/DlA8rVagBFUqvEBxz1i8vNePIe+ZUZvEkLp9Gm8AJv\*MRx7DWchbSCY1hi1MiavFA==\*a5zhGvh6Zr0fqbYbENd/EA==\*8b8v7iaURll5/PGUm1nmHA=="  } |  find\_one  |  |  | | --- | --- | | Normal | Encryption | | {  "SERVER": "GKE",  "CHANNEL\_ID": "API",  "DB\_NAME": "dbtest",  "QUERY": {  "collections": "docs",  "operations": "find\_one",  "doc\_array":{"KOTA" : "KOTA MEDAN"}  }  } | {  "SERVER": "GKE",  "CHANNEL\_ID": "API",  "DB\_NAME": "X537N32p\*hylDUJ7c7LqBymceIUTaDQ==\*8DiA7/m0fu7oclODipOZCw==\*hNfFJkWPC/JBoHLWUirhDg==",  "QUERY": "hQB/hAoOnrKPYMdHyXGtUyR2G/54o4NX1ErVJbhC8qhEjj9MzBfYVTnFhJeg1hgiOqMB8eByn3a8PuxrCJIPe8iM9ZVFJqTqOlvmBwEw77wMitgrSW0=\*3Fhvjz1XzuG5S0iF1oo2wA==\*yKkeFpRX2PX7JT0CZ6vQGg==\*V0ukt1djVOJDrqfHdAqBxQ=="  } |  find\_many  |  |  | | --- | --- | | Normal | Encryption | | {  "SERVER": "GKE",  "CHANNEL\_ID": "API",  "DB\_NAME": "dbtest",  "QUERY": {  "collections": "docs",  "operations": "find\_many",  "doc\_array":{"KOTA" : "KOTA MEDAN"}  }  } | {  "SERVER": "GKE",  "CHANNEL\_ID": "API",  "DB\_NAME": "W8ZO6guK\*pfNAfqbfAOeMQUgkkyRM0g==\*Z1XlYhNx9RsDjf4NgXkFjg==\*D/iKH+6zk0W0pgbWUs6nMw==",  "QUERY": "qU/0ejS7S7UQWAeoufbpiLccqdMd+3c+R+8c4th4cR3Y2WPT7XhrUDbDUSnHOun5B0w6z34rRbFj6/UWSaea6jbsIb5Cr2w+l5UXOLS7YQW++R10/9Ab\*JzUGzUO1DlbHH1u73HOkvA==\*li4fLa/ZpJKfNvm++Sh3Tg==\*e1hVZNVbR10GVbV1XlXSiA=="  } |  delete\_one  |  |  | | --- | --- | | Normal | Encryption | | {  "SERVER": "GKE",  "CHANNEL\_ID": "API",  "DB\_NAME": "dbtest",  "QUERY": {  "collections": "docs",  "operations": "delete\_one",  "doc\_array":{"KOTA" : "KOTA MEDAN"}  }  } | {  "SERVER": "GKE",  "CHANNEL\_ID": "API",  "DB\_NAME": "AfntY1UH\*otZD4o3a7F8aMSAxLa67jg==\*yM5y0sdrtJtmzhEoikXQRw==\*CucljqEOrWcWLAW/qcvteQ==",  "QUERY": "wYNrC+NQaXn15BiCwDuz0lp5DnoZyUly8MyefLo5lnQChlhitg3Ewf4ynm602EJ5FHOGNmnDdN3pDHFXotnQBj3KexXf9Puzd83yVRuDqYeG/g1/BOh1Ww==\*s4E4++ALkJxkT7cYuSyIXA==\*lx7FSyePtsOaO+7xF9/t/Q==\*fTaK6lTvobYk76l5wTBAmw=="  } |  delete\_many  |  |  | | --- | --- | | Normal | Encryption | | {  "SERVER": "GKE",  "CHANNEL\_ID": "API",  "DB\_NAME": "dbtest",  "QUERY": {  "collections": "docs",  "operations": "delete\_many",  "doc\_array":{"KOTA" : "KOTA MEDAN"}  }  } | {  "SERVER": "GKE",  "CHANNEL\_ID": "API",  "DB\_NAME": "YN0UmRLB\*5zivVD0OMm9n1edLDFaCUQ==\*+FSbmMfNrTp3W9FxF/HOqg==\*jIG0Kr3ltZN0X/cLTf3P+g==",  "QUERY": "8zd5lq3fDocZN7hLadw+Sk2lf+w+3Y1zFa5FuRqeTLhxbgrq/NhRZnoveV0He9jN4mSWvR7jHwWAosXuc76HRi5/QFjpZZqr+zP0VF/nZeRnpioMG+B1fZU=\*KTyvbh4ClKM9X5fnfJHs4g==\*xtIsxUqSfBQGKso6Ej7tYA==\*qVwYVeoxy/dc7lnGcBoM9Q=="  } |  pipeline  |  |  | | --- | --- | | Normal | Encryption | | {  "SERVER": "GKE",  "CHANNEL\_ID": "API",  "DB\_NAME": "list\_area",  "QUERY": {  "collections": "outlet",  "operations": "pipeline",  "doc\_array": [{  "$match": {  "NAMA\_OUTLET": "WILAYAH 01 - MEDAN"  }  }]  }  } | {  "SERVER": "GKE",  "CHANNEL\_ID": "API",  "DB\_NAME": "zfhRXJcnc/bk\*/GtSwCV5CE+CzYSoHfLYGQ==\*1IggVbuw1AOQ40FRlqciTA==\*KHHFBFQ6DocA0yS1ln7Iww==",  "QUERY": "iyo5JdtGZvqZMg5wzv0MP9xChVZNnY6RJltNR0cIhRtduss+W2Teu5hPWY1h07TbHoSpG4Vw5PPJ9o2MVai/kRLQW34FEF2n22EwKUeR792IcYSLBxzciGLVYdBUhV2MQXW5RBuTwkCygGSD3h4QkmXA9xbE\*oiiBz7TlctzWq+pG5zKrcw==\*gc0MxUoZZB4ExVOohoFIVQ==\*vHfRC7GgwyhSFkDjGA3ZIA=="  } | | {  "SERVER": "GKE",  "CHANNEL\_ID": "API",  "DB\_NAME": "CLSWebsite",  "QUERY": {  "collections": "UserManagement",  "operations": "pipeline",  "doc\_array": [{  "$match": {  "ROLE": {"$ne": "ADMIN"}  }  }]  }  } | {  "SERVER": "GKE",  "CHANNEL\_ID": "API",  "DB\_NAME": "k1ikcjg1Jv7lYw==\*0OQbXY9Bs9fFoiNF7JkY9w==\*9dqV6CVzb9TQ3QG5iz7hAA==\*BLZbavWrtMnXdiZzuCQ++w==",  "QUERY": "kTANrSAcYT47Ko8Nwv9OcGFFzUhPkO89NtusDfM54rsSIqw6fQruAPYNYVoD4AtS9jblVmUX0mXksyFfvqhnPiCzPyi4N+H/nRfNpTRh4r9dutuFYdaoCLGRPG0bkbrA2DoawLq4/vQ0Piwyge7mOOf7\*0ZyD0qz6ILpgV1PMZuxv5w==\*/p1tU4uqhWgTP3rm4ds9Dg==\*Wjh5BK0k2bm0g61/AkNHFQ=="  } | | {  "SERVER": "GKE",  "CHANNEL\_ID": "API",  "DB\_NAME": "list\_area",  "QUERY": {  "collections": "outlet",  "operations": "pipeline",  "doc\_array": [{  "$match": {  "WILAYAH": "1"  }  },  { "$limit": 1 }  ]  }  } | {  "SERVER": "GKE",  "CHANNEL\_ID": "API",  "DB\_NAME": "/IQK2jirg/cw\*HVnfOUuUFYsh2p0UF78sSg==\*o0LjujLcIYxJ3JHbosNIXw==\*n0tTsmFdH0vyvvf0WAlPAg==",  "QUERY": "eouPEau0oZ84n1Vv3hMx+K/7NykR+E/FF8F6doEEOb3NW0cLaI7VEZgd2DW9VUDSHt0KUwLXzRCjXquAw5EkcZeG0SzihilRH1ep8Rux1RnCUkj5gOCsKi0nnkbmxqOsjtgcngPkDtAW9oF6C/yc\*i7xB9nbUPTJ5uwwPNmwc2g==\*RKjpMCP0WvbrYVR/9+4n0w==\*4SO5HmkSo4z+tHXlCstpTg=="  } | |
| Terima Kasih |