



MongoDB

Data Persistency

Leijzen Jonas (IAO3)

Inhoud

Inhoud	1
Wat is MongoDB	3
NoSQL	3
JSON Document store	3
Waarom MongoDB	3
Sharding	4
Wat is sharding	4
Voordelen van sharding	4
Shard-keys	4
Replica Sets	5
Wat zijn replica sets	5
Waarom replica sets	5
Onze opstelling	6
Schema	6
Config servers	6
Shards	6
Shard A	6
Shard B	7
Mongos router	7
Database client	7
Opstellen van de configuratie	8
Vereiste programma's:	8
Mappenstructuur	8
Config servers	8
Shard servers	8
Configuratie servers	9
Mongod instanties	9
Configuratie	9
Shards opstellen	11
Shard A	11
Mongod instanties	11
Configuratie	11
Shard B	12
Mongod instanties	12
Configuratie	12
Mongos	13
Server instantie	13
Database	13
Configuratie	14

Chuck Size	15
Databank vullen	16
Data verkrijgen	16
Data importeren	17
Sharding distribution	17

Wat is MongoDB

NoSQL

MongoDB is een NoSQL databank. Wanneer mensen de term "NoSQL-database" gebruiken, bedoelen ze meestal een niet-relatieve database. Sommigen zeggen dat de term "NoSQL" staat voor "no SQL", terwijl anderen zeggen dat het staat voor "not only SQL". Hoe dan ook, de meesten zijn het erover eens dat NoSQL-databases databases zijn die gegevens opslaan in een ander formaat dan relationele tabellen.

JSON Document store

Een normale sql server heeft databases met daarin tabellen. In tegenstelling heeft een MongoDB server databases met daarin collecties. In een tabel worden rijen opgeslagen. Als het over MongoDB gaat spreekt men eerder van objecten. Elk zo'n object is een JSON bestand. Net zoals met rijen in een tabel kunnen op objecten in een collecties ook uitgebreide queries worden uitgevoerd.

Waarom MongoDB

MongoDB is gebouwd op een scale-out architectuur die populair is geworden bij allerlei ontwikkelaars voor het ontwikkelen van schaalbare toepassingen met evoluerende gegevensschema's.

Als documentendatabase maakt MongoDB het ontwikkelaars gemakkelijk om gestructureerde of ongestructureerde gegevens op te slaan. Dit formaat komt direct overeen met native objecten in de meeste moderne programmeertalen, waardoor het een gemakkelijke keuze is voor ontwikkelaars, omdat ze niet hoeven na te denken over het normaliseren van gegevens. MongoDB kan ook hoge volumes aan en kan zowel verticaal als horizontaal schalen.

MongoDB is gebouwd voor mensen die internet- en bedrijfstoepassingen bouwen en die snel moeten evolueren en elegant moeten schalen. Bedrijven en ontwikkelingsteams van elke omvang gebruiken MongoDB om uiteenlopende redenen.

Sharding

Wat is sharding

Sharding is een methode om gegevens te verspreiden over meerdere machines. MongoDB gebruikt sharding voor het ondersteunen van implementaties met zeer grote datasets en hoge verwerkingscapaciteit. Shards van een database kunnen op verschillende servers/locaties bevinden om de belasting op individuele server te spreiden.

Databasesystemen met grote datasets of toepassingen met een hoge verwerkingscapaciteit kunnen de capaciteit van een enkele server overbelasten en voor vertraging in het systeem zorgen. Als er veel queries worden uitgevoerd kan bijvoorbeeld de CPU-capaciteit van de server uitgeput raken.

Voordelen van sharding

MongoDB verdeelt de lees- en schrijfbelasting over de shards in het sharded cluster, zodat elke shard een subset van de queries kan verwerken. Zowel lezen als schrijven kan horizontaal over het cluster worden geschaald door meer shards toe te voegen.

Voor queries die de shard-key bevatten, kan mongos de query richten op een specifieke shard of set van shards. Deze gerichte operaties zijn over het algemeen efficiënter dan het uitzenden naar elke shard in het cluster omdat niet elke shard moet worden aangesproken.

Shard keys

Bij het verdelen van de data tussen de shards moet er beslist worden welke data zich waar hoort te bevinden. Dit gebeurt met behulp van een shard-key. Deze shard-key moet variëren over de dataset zodat de data gelijk verdeeld kan worden over de shards. Als de shard-key veel voorkomt en queries kan die ook voor meer performantie zorgen, omdat niet alle shards aangesproken moeten worden.

Replica Sets

Wat zijn replica sets

Een replica set is een groep mongod-instanties die dezelfde gegevensset bevatten. Een verzameling replica's bevat verscheidene nodes. Van de nodes wordt één en slechts één lid beschouwd als het primaire node, terwijl de andere nodes worden beschouwd als secundaire nodes.

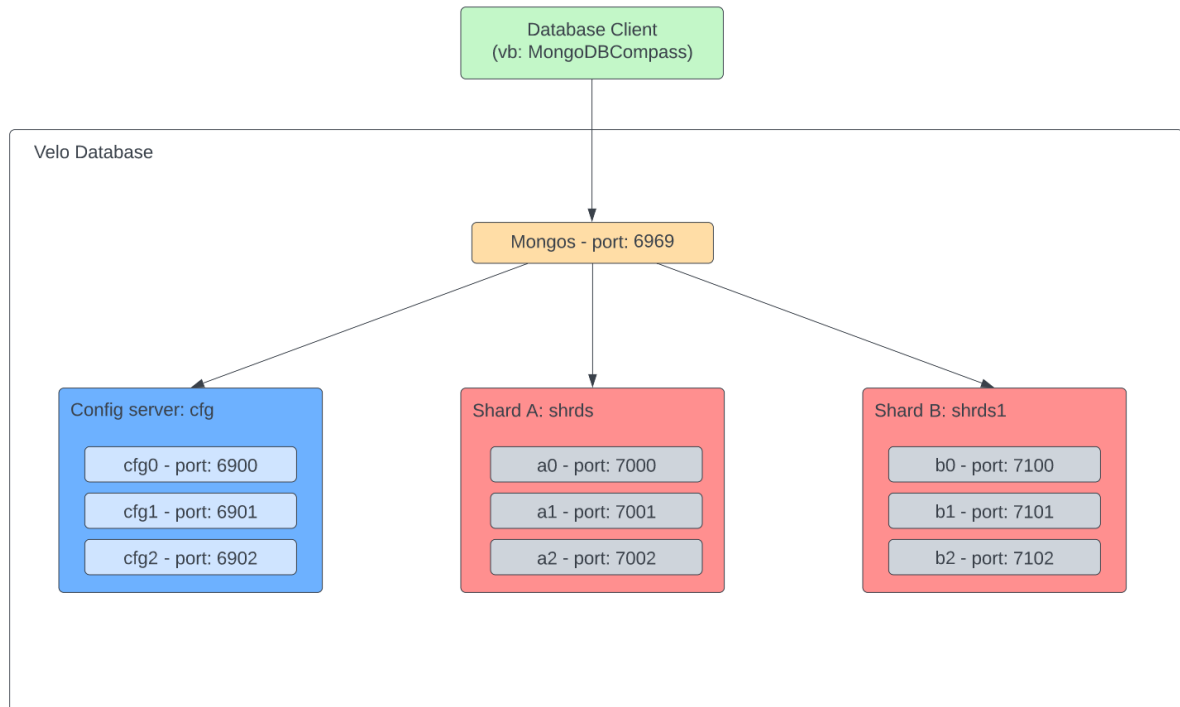
Waarom replica sets

Replica sets zorgen voor redundantie en hoge beschikbaarheid. Data kan corrupt gaan en hersteld kan worden met de andere kopieën van dezelfde data. Wanneer er veel queries tegelijk worden uitgevoerd kunnen deze ook verspreid worden over de meerdere replicaset.

Net zoals de shards kunnen aparte kopieën van een replica set zich op verschillende servers/locaties bevinden.

Onze opstelling

Schema



Config servers

Net zoals de shards kunnen config server ook deel uitmaken van een replica set. Zo is de kans op corruptie of verlies van de serverconfiguratie sterk verkleind.

In onze opstelling gebruiken we 3 mongod instanties met config servers met de poorten 6900, 6901, 6902 voor de server cfg0, cfg1 en cfg2. Samen vormen deze 3 config servers de replica set cfg

Shards

We maken in onze opstelling gebruik van 2 shards, dit is vooral om de werking ervan te kunnen demonstreren. Voor de selectie van de netwerkpoorten gebruiken we 70xx voor shard A en 71xx voor shard B.

Shard A

Dit is onze eerste shard van onze database. Voor de 3 mongod instanties gebruiken we hier de poorten 7000 voor a0, 7001 voor a1 en 7002 voor a2. Deze vormen samen de replicaset 'shrds'.

Shard B

Dit is de 2de shard van onze database. De replicaset van deze shard heet 'shards1'. De onderliggende mongod instanties zullen de poorten 7100 voor b0, 7101 voor b1 en 7102 voor b2 krijgen.

Mongos router

Dit is het centrale aanspreekpunt van onze database. Het is de tussenlaag tussen onze sharded cluster en de buitenwereld. Hier worden alle queries naar verstuurd en beantwoord. Hier zal ook de sharding configuratie ingesteld worden. In onze opstelling krijgt de mongos router de poort 6969.

Database client

Wij gebruiken de database client MongoDB Compass. Deze client is van de ontwikkelaars van MongoDB en heeft hierdoor een paar handige functies die we laten gaan gebruiken bij het opstellen van de database opstelling.

Opstellen van de configuratie

Nu gaan we onze configuratie opstellen en uitvoeren. We gebruiken Windows als besturingssysteem.

Vereiste programma's:

- MongoDB 6.0.2: <https://www.mongodb.com/try/download/community>
- MongoDBCompass 1.33.1: <https://www.mongodb.com/try/download/compass>
- Microsoft SQL Server Management Studio 18:
<https://learn.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver16>

PAS OP:

Voeg zeker ook de MongoDB commandos toe aan je PATH variable.
(standaard: C:\Program Files\MongoDB\Server\6.0\bin)

Mappenstructuur

Kies de locatie waar je de database wilt plaatsen. Alle commandos die volgen voor de rest van de configuratie worden met cmd vanuit deze locatie uitgevoerd.

Config servers

Voor de mapstructuur van de config servers voeren we volgend commando uit:

```
mkdir configs\cfg0 configs\cfg1 configs\cfg2
```

Deze zal een map configs aanmaken met daarin 3 mappen voor de individuele mongod instanties van de configservers.

Shard servers

De shard servers krijgen elks een aparte map. Alle mappen beginnen met de naam van de shard waar ze bijhoren (a of b). Daarna komt er getal (0, 1 of 2), dit getal duidt aan voor welke server uit de replicaset deze behoort.

We gebruiken volgend commando voor de shard mappen aan de maken:

```
mkdir shards\a0 shards\a1 shards\a2 shards\b0 shards\b1 shards\b2
```

Dit zal een map shards maken met daarin de mappen a0, a1, a2, b0, b1 en b2. Dit geeft elke mongod instantie een eigen map om uit te werken.

Configuratie servers

Mongod instanties

Voor de 3 configuratieservers in de replicaset moet telkens een mongod instantie gestart worden. Om deze te starten voeren we volgende commandos uit:

```
START "Config server 0 (cfg0)" mongod --configsvr --port 6900 --bind_ip localhost --replSet cfg --dbpath "configs/cfg0"

START "Config server 1 (cfg1)" mongod --configsvr --port 6901 --bind_ip localhost --replSet cfg --dbpath "configs/cfg1"

START "Config server 2 (cfg2)" mongod --configsvr --port 6902 --bind_ip localhost --replSet cfg --dbpath "configs/cfg2"
```

Dit zal 3 instanties van mongod starten die als configuratie server dienen. We geven ze elks een uniek poort (6900, 6901 en 6902), zeggen dat ze in de replicaset cfg zitten en de mappen die we hierboven hebben aangemaakt waar ze hun bestanden mogen opslaan.

Configuratie

Nu moeten we deze 3 servers met elkaar verbinden. Dit doen we door met mongosh (Mongo Shell) in cfg0 de configuratie aan te passen.

Open een connectie met de eerste mongod instantie (cfg0) met behulp van mongosh:

```
mongosh --host localhost --port 6900
```

Nu we verbonden zijn met cfg0 kunnen we de configuratie aanmaken:

```
rsconf = {
  _id: "cfg",
  members: [
    {
      _id: 0, host: "localhost:6900"
    },
    {
      _id: 1, host: "localhost:6901"
    },
    {
      _id: 2, host: "localhost:6902"
    }
  ]
}
```

Vervolgens moeten we de configuratie activeren op de server. Dit doen we met volgend commando:

```
rs.initiate(rsconf)
```

Eens we cfg0 hebben ingesteld worden de instellingen automatisch overgezet op cfg1 en cfg2. We kunnen de configuratie van cfg0 verlaten met `exit`.

Shards opstellen

Shard A

Mongod instanties

Voor de eerste shard gaan we nogmaals beginnen met het opstarten van drie mongod instanties voor het complete replicaset van shard A (shrds).

```
START "Shard server A0 (a0)" mongod --shardsvr --port 7000 --bind_ip localhost --replSet shrds --dbpath "shards\a0"
START "Shard server A1 (a1)" mongod --shardsvr --port 7001 --bind_ip localhost --replSet shrds --dbpath "shards\a1"
START "Shard server A2 (a2)" mongod --shardsvr --port 7002 --bind_ip localhost --replSet shrds --dbpath "shards\a2"
```

Net zoals bij de configuratie servers geven we de unieke poort, het replicaset en bestandslocatie mee.

Configuratie

Verbind met de eerste server om de replicaset te configureren.

```
mongosh --host localhost --port 7000
```

Als we met de server verbonden zijn uploaden we onze configuratie weer naar de server. Deze keer veranderen we wel de naam van de replicaset van cfg naar shrds, dit is naam van de replicaset voor shard A. Ook verwijzen we nu naar de nieuwe poorten van de mongod instanties hierboven.

```
rsconf = {
  _id: "shrds",
  members: [
    {
      _id: 0, host: "localhost:7000"
    },
    {
      _id: 1, host: "localhost:7001"
    },
    {
      _id: 2, host: "localhost:7002"
    }
  ]
}
rs.initiate(rsconf)
```

Als we verbonden zijn via mongosh met a0 kunnen we de status van replicaset controleren met `rs.status()`. Let hier vooral op de `ok: 1` om te verifiëren dat replicaset in orde is.

Shard B

Mongod instanties

Voor de shard server instanties gaan we hetzelfde doen als bij shard A, maar deze keer met andere poorten, andere bestandslocaties, andere procesnamen en een ander replicaset.

```
START "Shard server B0 (b0)" mongod --shardsvr --port 7100 --bind_ip
localhost --replSet shrds1 --dbpath "shards\b0"
START "Shard server B1 (b1)" mongod --shardsvr --port 7101 --bind_ip
localhost --replSet shrds1 --dbpath "shards\b1"
START "Shard server B2 (b2)" mongod --shardsvr --port 7102 --bind_ip
localhost --replSet shrds1 --dbpath "shards\b2"
```

Configuratie

De configuratie voor shard B verschilt slechts een klein beetje met die van shard A. We openen weer een mongosh connectie een van de mongod instanties:

```
mongosh --host localhost --port 7100
```

Nogmaals worden de replicaset en poorten veranderd.

```
rsconf = {
  _id: "shrds1",
  members: [
    {
      _id: 0, host: "localhost:7100"
    },
    {
      _id: 1, host: "localhost:7101"
    },
    {
      _id: 2, host: "localhost:7102"
    }
  ]
}

rs.initiate(rsconf)
```

Proficiat! De configuratie server en shard servers zijn volledig geconfigureerd, nu gaan we deze allemaal met elkaar verbinden.

Mongos

Server instantie

Zoals eerder vermeld is mongos de software die alles met elkaar verbindt en als uiteindelijk aanspreekpunt dient voor clients. Bij het opstarten van de mongos instantie moeten we de configuratie servers meegeven aan de hand van hun replicaset en poorten. We geven mongos ook de poort 6969.

Voor de mongos server te starten voeren we volgend commando uit:

```
START "Mongos server" mongos --configdb  
cfg/localhost:6900,localhost:6901,localhost:6902 --bind_ip localhost  
--port 6969
```

Nu draait de mongoDB server op onze machine.

Database

We hebben nu een mongodb server opgestart men de configuratie uit de replicaset 'cfg'. Voor we de mongos server instantie verder configureren moet eerst een database en een collectie aanmaken. Dit doen we met behulp van MongoDBCompass.

Maak een nieuwe connectie aan waar je de connectionstring van de server ingeeft. Hiermee kan je verbinden met de mongos instantie.

New Connection

Connect to a MongoDB deployment

URI ⓘ Edit Connection String ☒

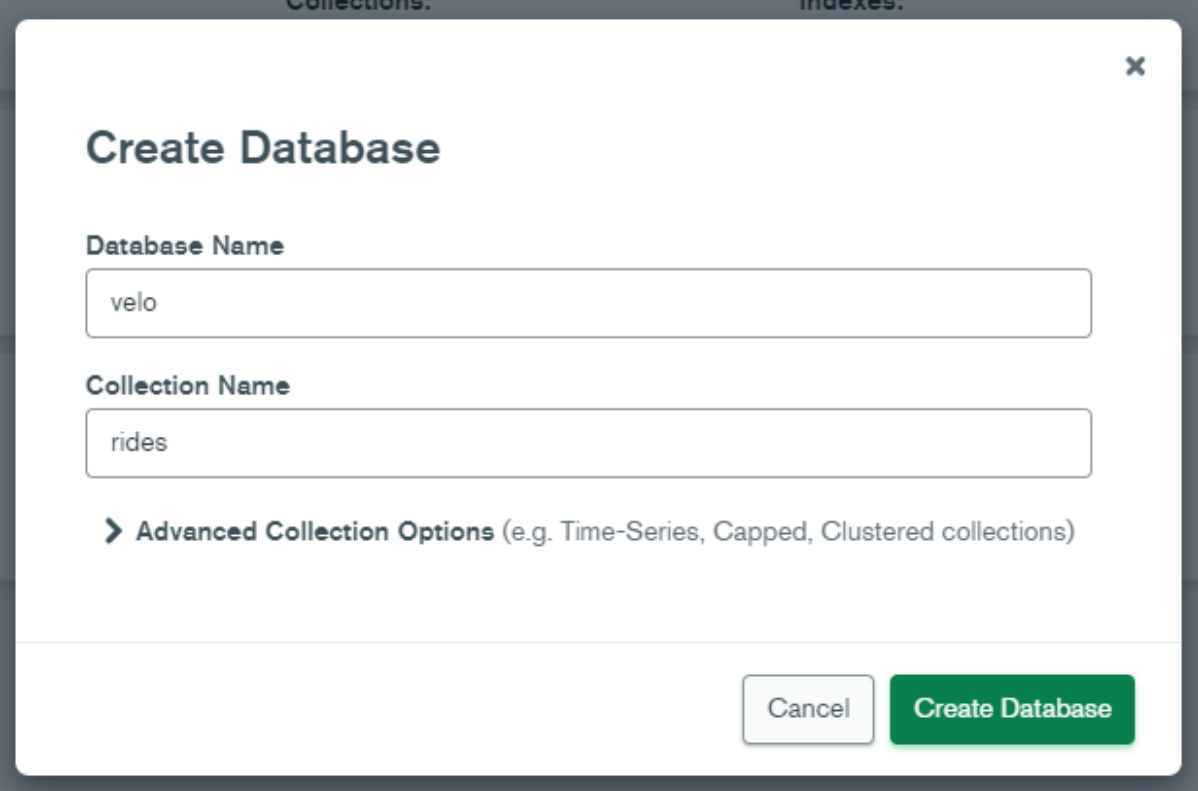
mongodb://localhost:6969

➤ Advanced Connection Options

Save Save & Connect Connect

FAVORITE

Navigeer vanboven aan het programma naar databases en maak een nieuwe database genaamd 'velo' aan. Hierin wordt er gewerkt met de collection rides.



Configuratie

Nu dat de juiste database en collection is aangemaakt kunnen we verder gaan met de configuratie. De volgende stap is om sharding aan te zetten en de shards toe te voegen aan de mongos server instantie.

We openen weer een connectie met mongosh om de server te configureren.

```
mongosh --host localhost --port 6969
```

De context moet veranderd worden van de standaard 'test' naar de 'velo' database die net aangemaakt is. Dit gebeurt met behulp van `use velo`. Merk op dat uitvoerlijn nu veranderd is.

Nu gaan we beide shards (A & B) toevoegen aan de mongos instantie. Voor beide shards geef je de replicaset en de unieke poort en host combinatie van individuele mongod instanties.

```
sh.addShard("shrd1/localhost:7000,localhost:7001,localhost:7002")
```

```
sh.addShard("shrd1/localhost:7100,localhost:7101,localhost:7102")
```

Om gebruik te maken van deze shards is het belangrijk dat we niet vergeten voor sharding aan te zetten om de server.

```
sh.enableSharding("rides")
```

De laatste stap voor de sharding te configureren is het instellen van een sharding key. We gebruiken voor deze opdracht `BeginStationId`. Deze key varieert veel over de data en kan vaak in een queries voorkomen. Configureer de sharding key met volgend commando:

```
sh.shardCollection("velo.rides", {"BeginStationId": 1})
```

Sharding is staat nu aan. De huidige configuratie kan perfect werken. Er wordt nog wel aanpassing doorgevoerd op de configuratie van de sharding.

Chunk Size

De standaard chunk size van een mongodb is 256MB. Voor een normale opstelling is dit een goede waarde, maar wij gaan deze waarde verminderen naar 32MB. Hiervoor moeten we zoals hiervoor eerst switchen van context. Doe dit met het `use` commando.

```
use config
```

Pas nu de chunksize aan met het volgende commando met het `updateOne` commando.

```
db.settings.updateOne(
  { _id: "chunksize" },
  { $set: { _id: "chunksize", value: 32 } },
  { upsert: true }
)
```

Verander de context terug naar de `velo` database en verifieer dat de instelling is aangepast.

```
use velo
```

```
sh.balancerCollectionStatus("velo.rides").chunkSize
```


Databank vullen

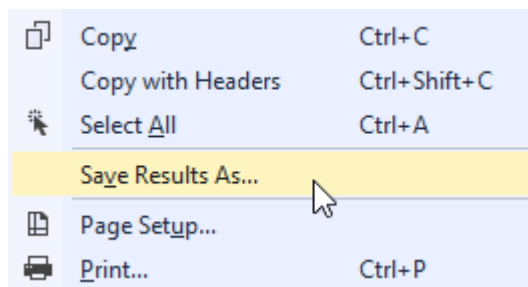
Data verkrijgen

Via MongoDBCompass kan je een csv importeren in de databank. Maak gebruik van Microsoft SQL Server Management Studio voor de bestaande data uit een tabel te halen en op te slaan in een csv.

Er is gebruikt gemaakt van volgende query op de sql server:

```
SELECT [RideId]
      ,[StartPoint]
      ,[EndPoint]
      ,[StartTime]
      ,[EndTime]
      ,R.[VehicleId]
      ,[SubscriptionId]
      ,[Startlockid]
      ,[EndLockId]
      ,L.StationId as 'BeginStationId'
FROM [velo_op].[dbo].[Rides] R
JOIN Locks L on r.Startlockid = l.LockId
```

Voor de query uit en slaag de output op in een csv met behulp van de 'Save Results As' function in SSMS:



Data importeren

Om de data te importeren in onze nieuwe database maken we nogmaals gebruik van MongoDBCompass.

Navigeer naar de Rides Collectie in de Velo database en selecteer 'add data' > 'import file'. Selecteer de geëxporteerde csv en selecteer csv. Verifieer dat de data die je inlaadt de correcte data is. Hier kan je ook de datatypen van je fields wijzigen.

Vervolgens wordt 'import' geselecteerd. MongoDBCompass zal nu de data uploaden naar de database. Als het process is afgelopen kan de shard verdeling geverifieerd worden. Hiermee kan er gekeken worden of de sharding effectief heeft gewerkt en de data verspreid is over meerdere shards.

Sharding distribution

De 'sharding distribution' kan bekenen worden met volgend commando:

```
db.rides.getShardDistribution ()
```

Hier kan de verdeling van de data in de shards bekeken worden. Als de data ongeveer 50/50 verdeeld is over de replicaset (shrds, shrds1) van shard A en shard B dan is de configuratie geslaagd.

Totals

```
{
  data: '1.22GiB',
  docs: 3737487,
  chunks: 39,
  'Shard shrds': [
    '51.52 % data',
    '51.45 % docs in cluster',
    '353B avg obj size on shard'
  ],
  'Shard shrds1': [
    '48.47 % data',
    '48.54 % docs in cluster',
    '352B avg obj size on shard'
  ]
}
```