MongoDB::the_good_stuff()

Dr. Roman Kutlak

Overview

- What is MongoDB
- Basics
- Dev/Python
- Extras
- Cloud

Slides / Code

- https://github.com/roman-kutlak/mongotest

What is MongoDB

MongoDB

- A document oriented NoSQL database
- Open Source - code available on github (C++)
- www.mongodb.com - MongoDB, Inc.
- Database, Ops Manager, Cloud Manager, Compass
- Documentation, Tutorials, MongoDB University courses

“MongoDB is a document database with the scalability and flexibility that you want with the querying and indexing that you need”

https://www.mongodb.com/what-is-mongodb

MongoDB

- Document database for JSON(-like) data
- Dev friendly
- Connectors for 10+ languages (C++, C#, Java, JS, Python)
- Database is distributed for robustness and scalability
- Latest version: 3.6
- www.mongodb.com

Basics

Install

- Docker
- Cloud (MongoDB Atlas)
- Download Community Server
- Package managers (Homebrew, apt, yum)

Connect

- Command line: >mongo (127.0.0.1:27017)
- Connect from your favourite language/library
- mongodb:// -- localhost/docker
- mongodb+srv:// -- MongoDB Atlas
- E.g., 'mongodb+srv://{username}:{password}@{host}/{database}?option1=value1&optionN=valueN'

Collections

- use database_name to select database; created on first insert or when creating collection
- Collection is like a table
- db.createCollection("log", { capped : true, size : 1024, max : 100 })
-
```
db.createCollection("students", {
   validator: {
      $jsonSchema: {...
```
- Read only views

CRUD: Create

```
db.inventory.insertOne({
  item: "canvas",
  qty: 100,
  tags: ["cotton"],
  size: { h: 28, w: 35.5, uom: "cm" }
})

{
  "acknowledged": true,
  "insertedId": ObjectId("5b09d98c2830327a92d41ad8")
}
```

CRUD: Retrieve

```
db.inventory.find({ status: "D" })

db.inventory.find({
      status: "A",
      $or: [{qty: {$lt: 30}},
             {item: /^p/ }]
})
```

CRUD: Update

```
db.inventory.updateOne(
   { item: "paper" },
   {
     $set: {"size.uom": "cm", status: "P"},
     $currentDate: { lastModified: true }
   }
)
```

```
{ "acknowledged" : true, "matchedCount" : 0,
"modifiedCount" : 0 }
```

CRUD: Delete

```
db.inventory.deleteOne( { status: "D" } )

db.inventory.deleteMany(
   { category: "cafe", status: "A" }
)

{ "acknowledged": true, "deletedCount": 0 }
```

Other Useful Functions

- `findOneAndDelete()`/ `findOneAndReplace()`/ `findOneAndUpdate()` -- do OP and return original doc
- `mapReduce()` -- Aggregate values: takes javascript functions
- `aggregate()` -- Aggregate values: "native" C++
- `wait()` -- Opens a change stream cursor
- updates have an "`upsert`" option

Dev/Python

Dev/Python

- pymongo -- official python driver
- mongoengine -- ORM framework on top of pymongo

Python (pymongo)

```
import pymongo

client = pymongo.MongoClient('mongodb+srv://
username:password@host/test')
db = client.test  # use database test
```

Python (pymongo)

```
print(list(db.person.find()))
> []

r = db.person.insert_one({"name": "Roman"})
print(r.inserted_id)
> 5b0b15e667c34f2164911b36

print(list(db.person.find({"name": "Roman"})))
> [{'_id': ObjectId('5b0b15e667c34f2164911b36'), 'name': 'Roman'}]

r = db.person.delete_one({"name": "Roman"})
print(r.deleted_count)
> 1
```

Python (mongoengine)

```python
from mongoengine import Document, StringField, connect


class Person(Document):
  name = StringField(max_length=256,
                     regex=r'^[A-Za-z].*')

  def __repr__(self):
    return f'<Person ({self.id}): {self.name}>'


connect('test', host='mongodb+srv://user:passd@host')
```

Python (mongoengine)

```
print(Person.objects.all())
> []

roman = Person(name='Roman')
roman.save()
print(roman.id)
> 5b0b171767c34f21ac26971e

print(Person.objects.filter(name='Roman'))
>  [<Person (5b0b171767c34f21ac26971e): Roman>]

roman.delete()
```

Extras

Extras

- GridFS
- Change Streams
- Retry-able Writes
- MongoDB Stitch
- Multi-document transactions in 4.0

pymongo/GridFS

```python
import os, gridfs, pymongo
from pprint import pprint

client = pymongo.MongoClient(connection_str)
db = client.test  # use database test
fs = gridfs.GridFS(db)
a = fs.put(b"hello world!",
           filename="test.txt",
           chunk_size=6)

# creates collections fs.files and fs.chunks
```

pymongo/GridFS

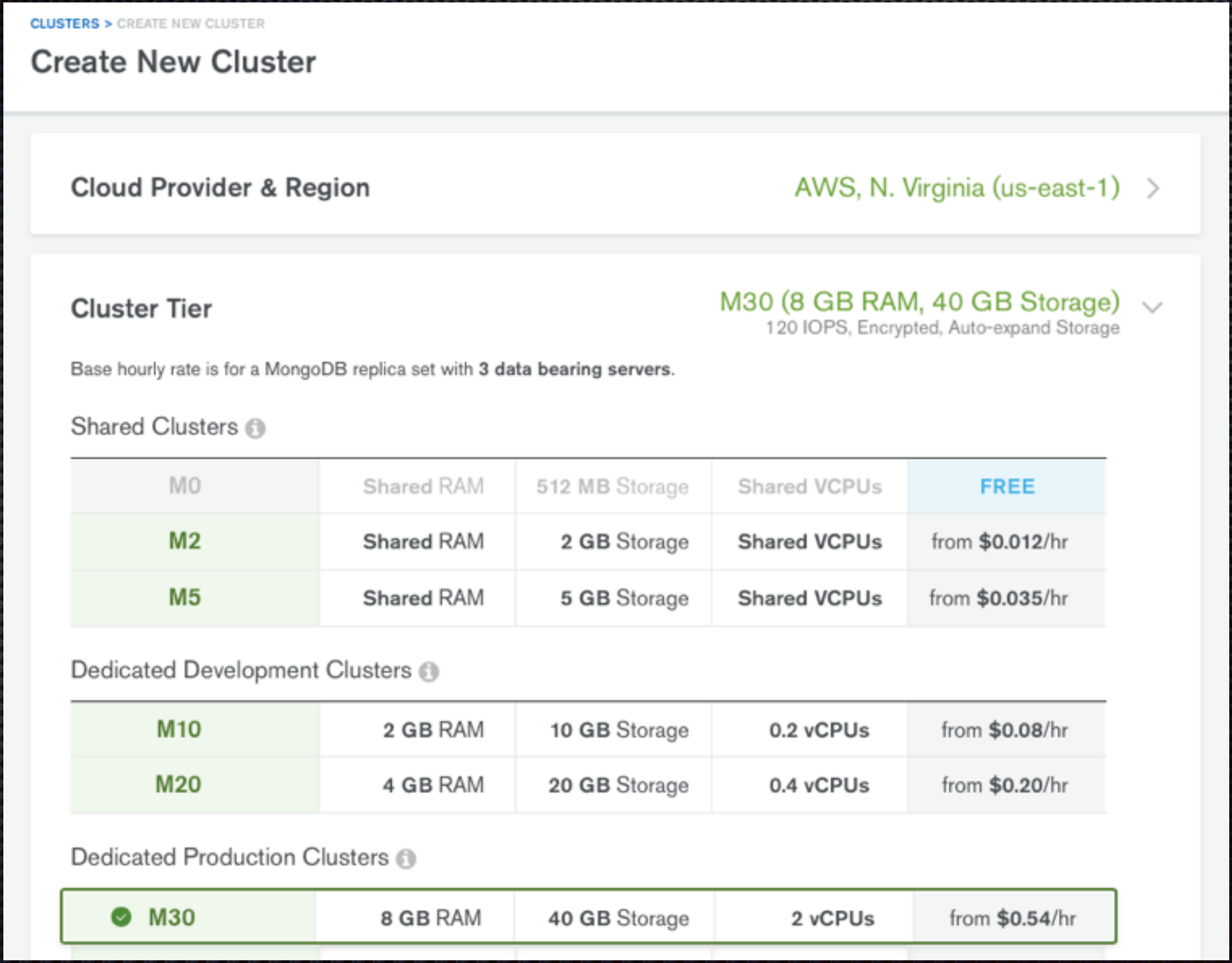
```
pprint(list(db.fs.files.find()))
[{'_id': ObjectId('5b0b1e1167c34f24dd9a543d'),
'filename' : 'test.txt', 'chunkSize' : 6,...

pprint(list(db.fs.chunks.find()))
[{'_id': ObjectId('5b0b1e1167c34f24dd9a543d'),
  'data': b'hello ',
  'files_id': ObjectId('5b0b1e0f67c34f24dd9a543c'),
  'n': 0},
 {'_id': ObjectId('5b0b1e1167c34f24dd9a543e'),
  'data': b'world!',
  'files_id': ObjectId('5b0b1e0f67c34f24dd9a543c'),
  'n': 1}]

print(fs.get(a).read())
b'hello world!'
```

pymongo/change stream

```
import pymongo

client = pymongo.MongoClient(connection_str)
db = client.test  # use database test
print('Tailing collection "logs"')
with db.logs.watch() as stream:
  for entry in stream:
    print('{}: {}'.format(entry['operationType'],
                          entry['fullDocument']))

# from mongo shell:
# db.logs.insertOne({message: "Message 1"})

Tailing collection "logs"
insert: {'_id': ObjectId('5b0b2e63b6cdb42883df9e2b'),
         'message': 'Message 1'}
```

MongoDB Atlas

MongoDB Atlas



CLUSTERS > CREATE NEW CLUSTER

Create New Cluster

Cloud Provider & Region — AWS, N. Virginia (us-east-1)

Cluster Tier — M30 (8 GB RAM, 40 GB Storage)
120 IOPS, Encrypted, Auto-expand Storage

Base hourly rate is for a MongoDB replica set with **3 data bearing servers.**

Shared Clusters

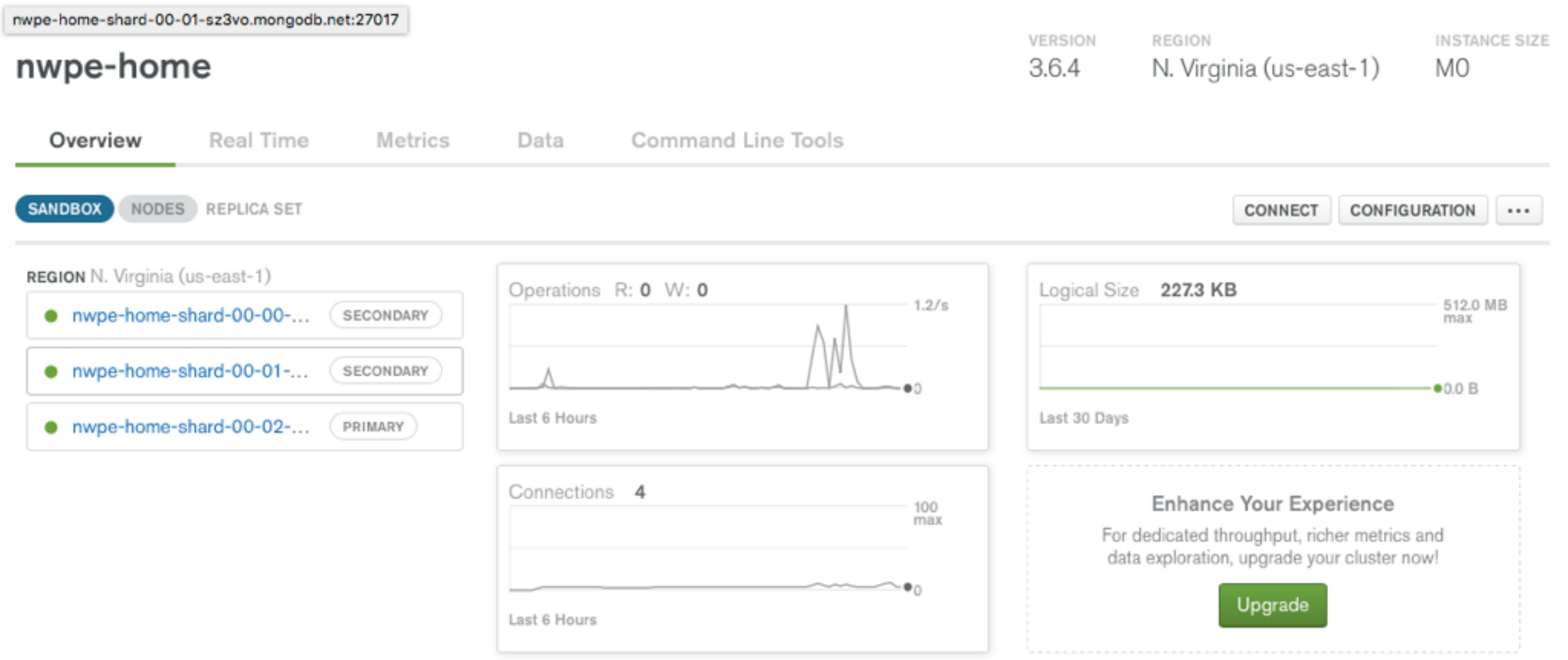
M0	Shared RAM	512 MB Storage	Shared VCPUs	FREE
M2	Shared RAM	2 GB Storage	Shared VCPUs	from $0.012/hr
M5	Shared RAM	5 GB Storage	Shared VCPUs	from $0.035/hr

Dedicated Development Clusters

M10	2 GB RAM	10 GB Storage	0.2 vCPUs	from $0.08/hr
M20	4 GB RAM	20 GB Storage	0.4 vCPUs	from $0.20/hr

Dedicated Production Clusters

✓ M30	8 GB RAM	40 GB Storage	2 vCPUs	from $0.54/hr



nwpe-home-shard-00-01-sz3vo.mongodb.net:27017

nwpe-home

VERSION 3.6.4

REGION N. Virginia (us-east-1)

INSTANCE SIZE M0

REGION N. Virginia (us-east-1)

- ● nwpe-home-shard-00-00-... SECONDARY
- ● nwpe-home-shard-00-01-... SECONDARY
- ● nwpe-home-shard-00-02-... PRIMARY

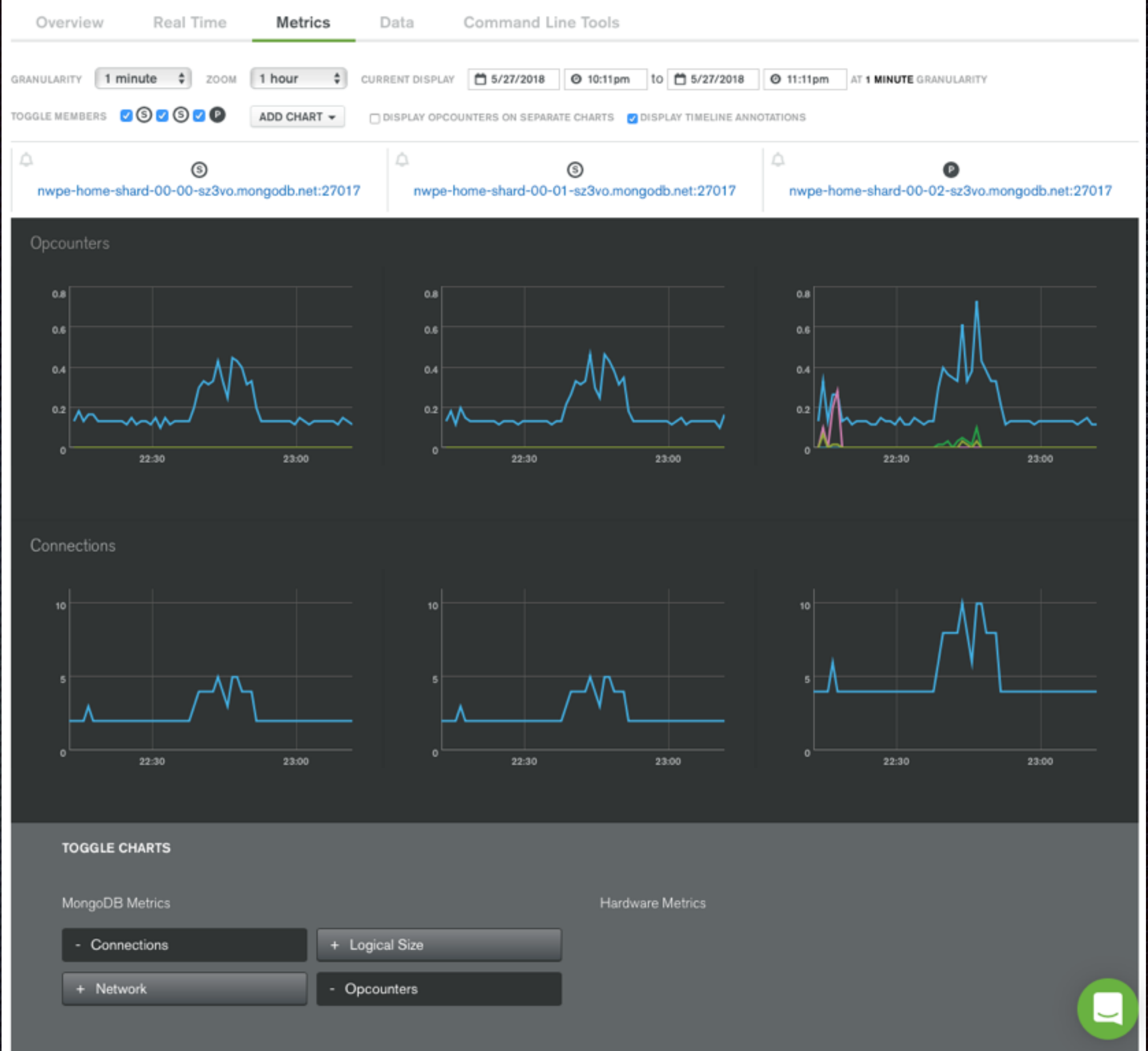
Operations R: 0 W: 0

Logical Size 227.3 KB

Connections 4

Enhance Your Experience

For dedicated throughput, richer metrics and data exploration, upgrade your cluster now!



Overview | Real Time | Metrics | Data | Command Line Tools

GRANULARITY 1 minute ZOOM 1 hour CURRENT DISPLAY 5/27/2018 10:11pm to 5/27/2018 11:11pm AT 1 MINUTE GRANULARITY

TOGGLE MEMBERS ☑ S ☑ S ☑ P   ADD CHART   ☐ DISPLAY OPCOUNTERS ON SEPARATE CHARTS ☑ DISPLAY TIMELINE ANNOTATIONS

S nwpe-home-shard-00-00-sz3vo.mongodb.net:27017
S nwpe-home-shard-00-01-sz3vo.mongodb.net:27017
P nwpe-home-shard-00-02-sz3vo.mongodb.net:27017

Opcounters

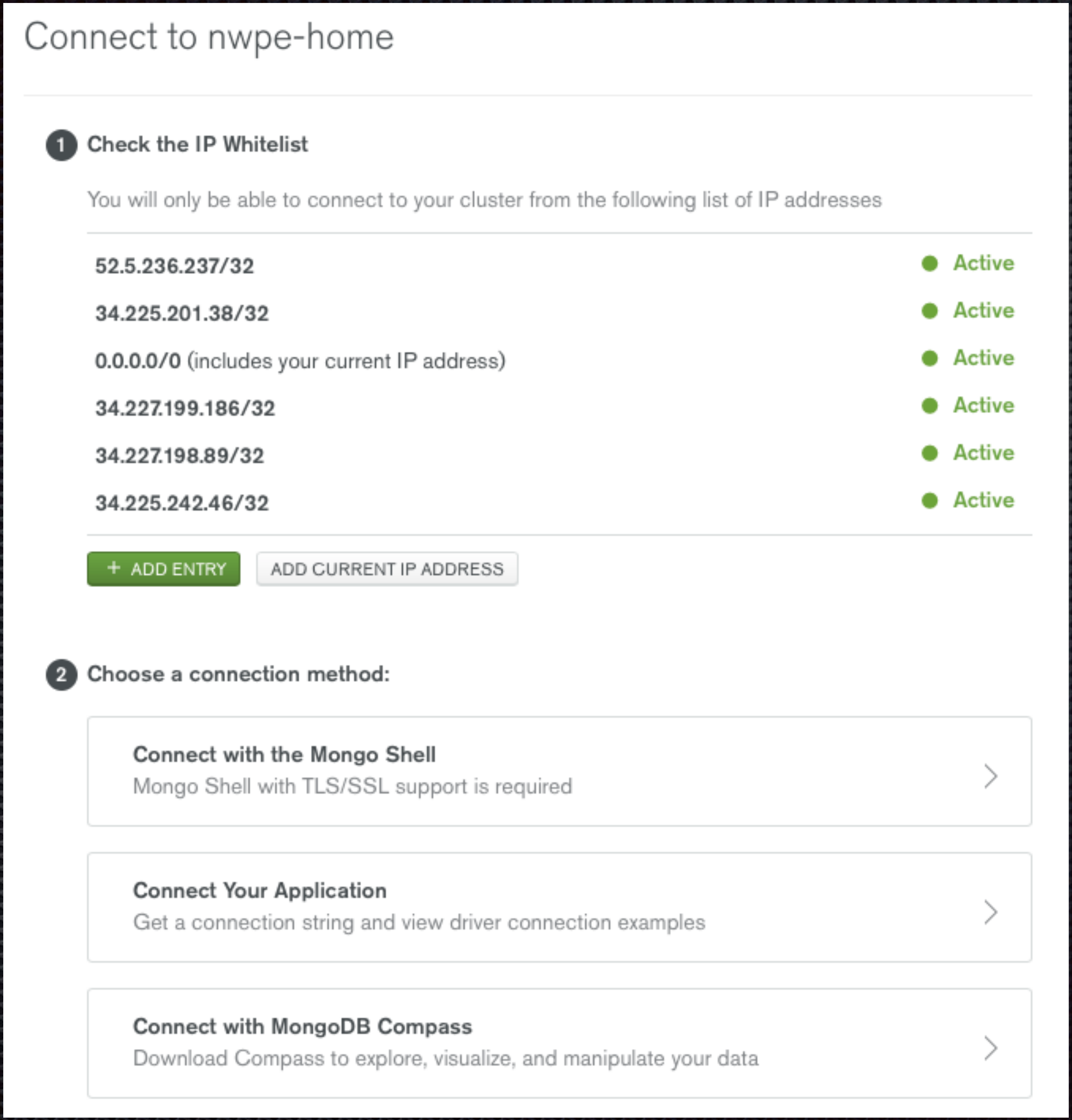
Connections

TOGGLE CHARTS

MongoDB Metrics

Hardware Metrics

- Connections
+ Logical Size
+ Network
- Opcounters



Connect to nwpe-home

1 Check the IP Whitelist

You will only be able to connect to your cluster from the following list of IP addresses

52.5.236.237/32	● Active
34.225.201.38/32	● Active
0.0.0.0/0 (includes your current IP address)	● Active
34.227.199.186/32	● Active
34.227.198.89/32	● Active
34.225.242.46/32	● Active

2 Choose a connection method:

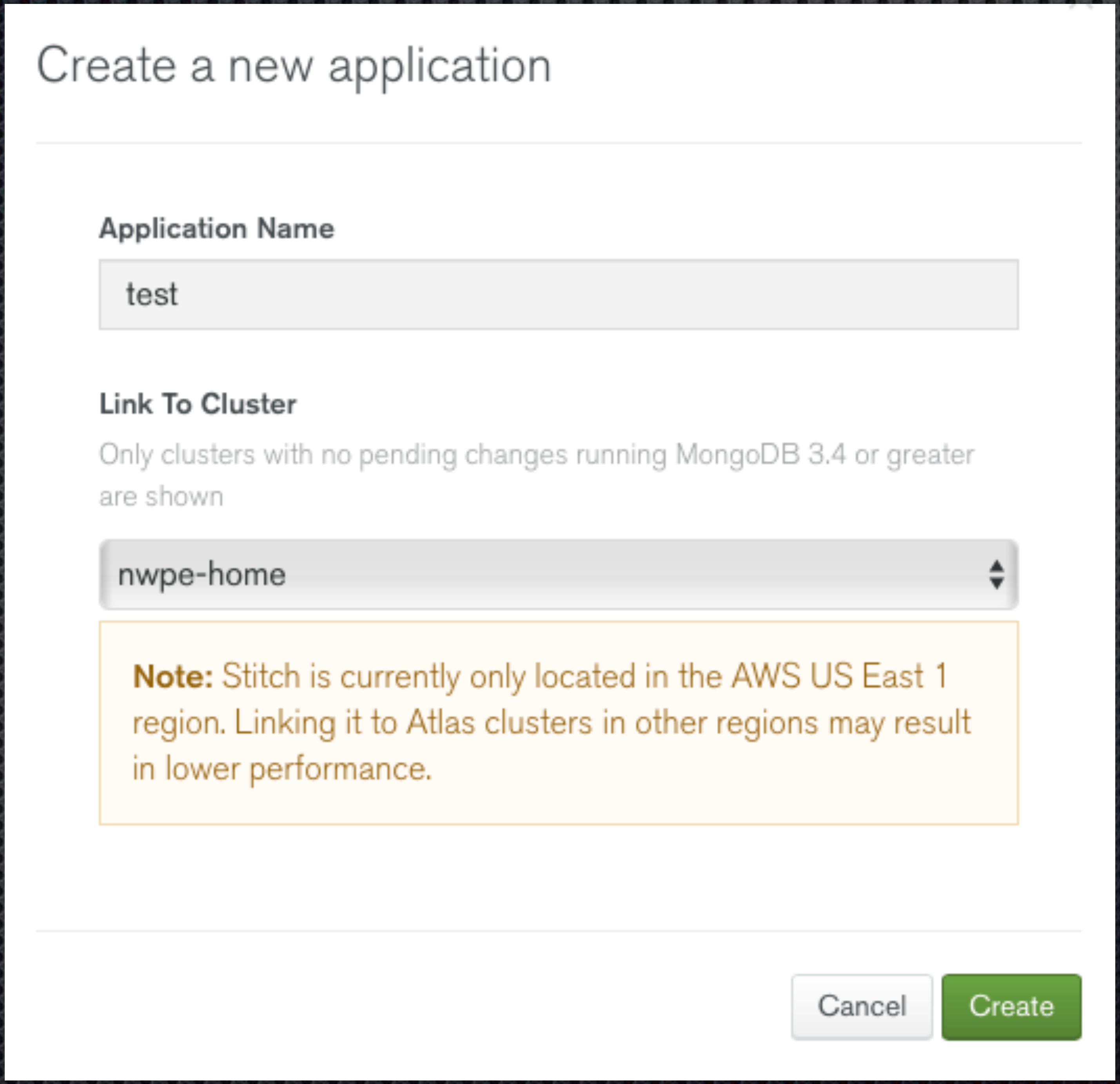
Connect with the Mongo Shell
Mongo Shell with TLS/SSL support is required

Connect Your Application
Get a connection string and view driver connection examples

Connect with MongoDB Compass
Download Compass to explore, visualize, and manipulate your data

MongoDB Stitch

MongoDB Stitch



Create a new application

Application Name

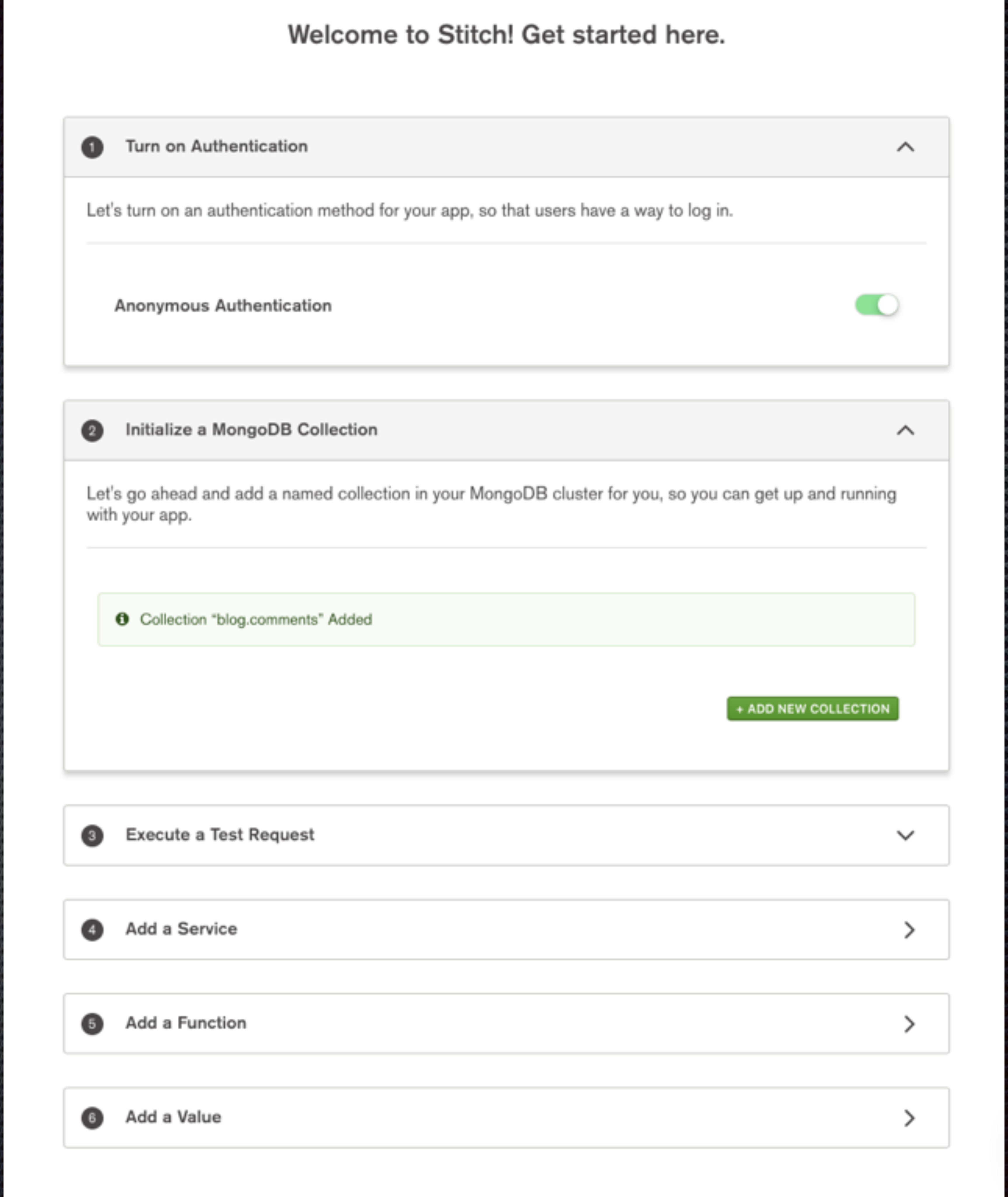
test

Link To Cluster

Only clusters with no pending changes running MongoDB 3.4 or greater are shown

nwpe-home

Note: Stitch is currently only located in the AWS US East 1 region. Linking it to Atlas clusters in other regions may result in lower performance.


Welcome to Stitch! Get started here.

1 Turn on Authentication

Let's turn on an authentication method for your app, so that users have a way to log in.

Anonymous Authentication

2 Initialize a MongoDB Collection

Let's go ahead and add a named collection in your MongoDB cluster for you, so you can get up and running with your app.

Collection "blog.comments" Added

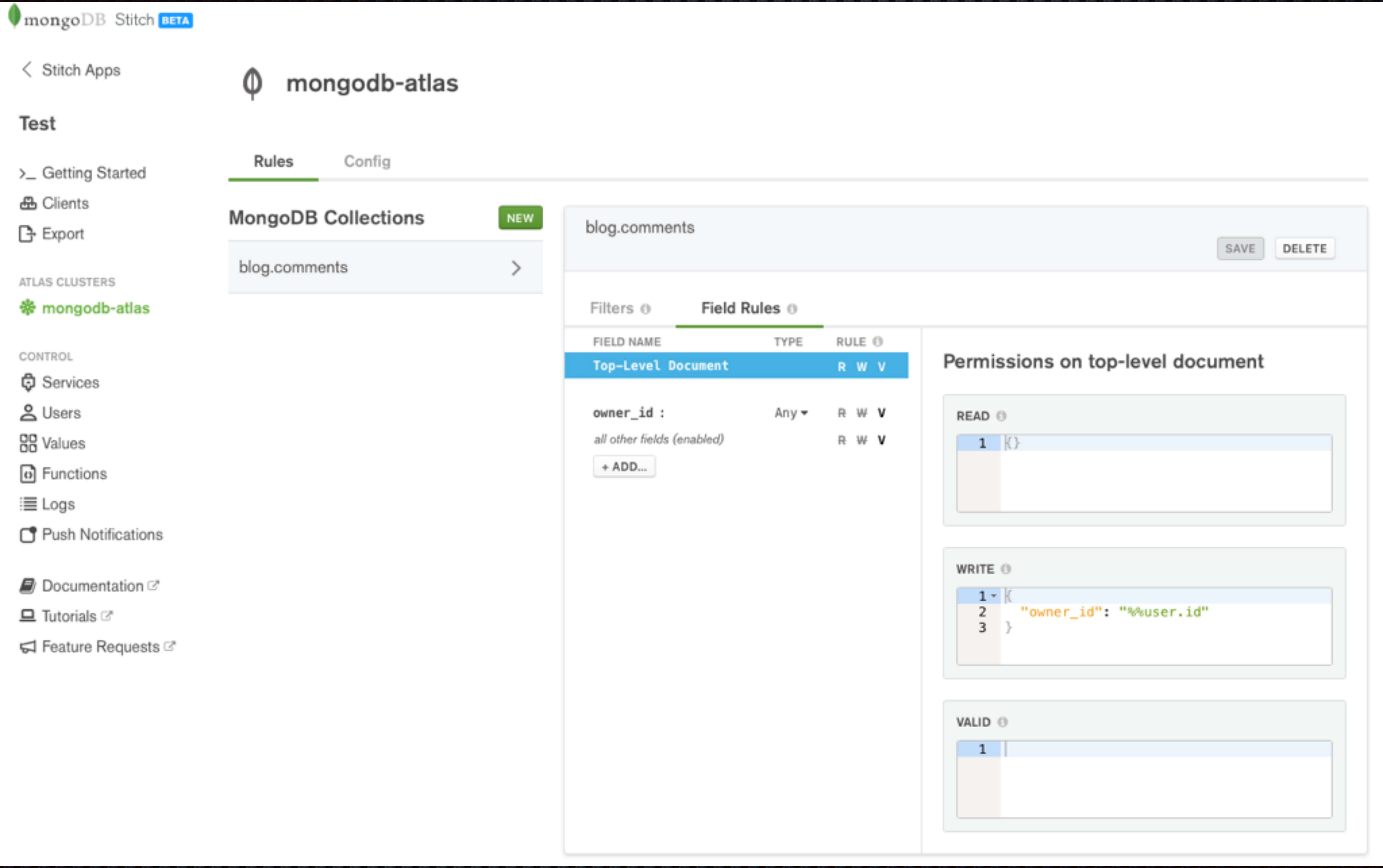
+ ADD NEW COLLECTION

3 Execute a Test Request

4 Add a Service

5 Add a Function

6 Add a Value



READ:

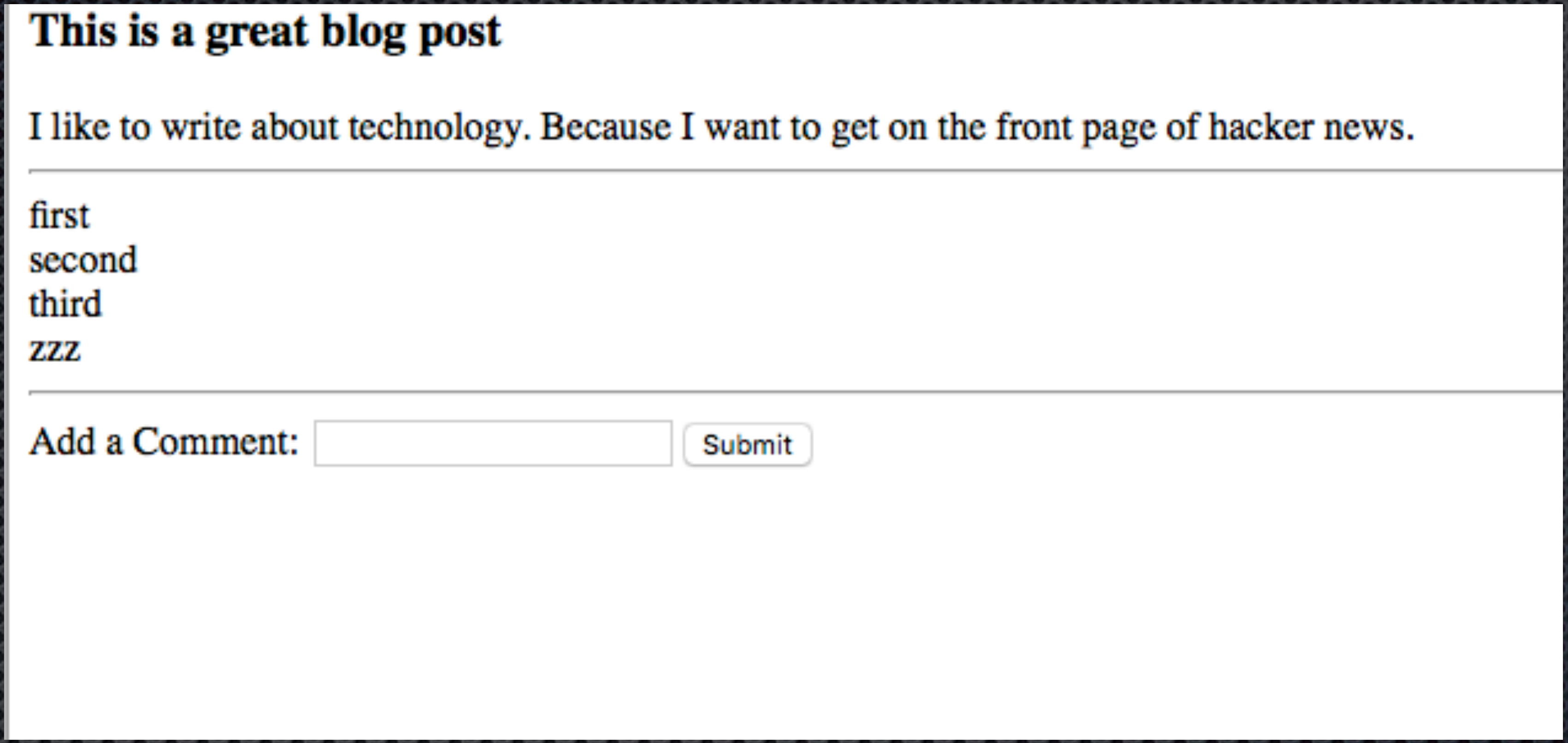
```
{}
```

WRITE:

```
{
  "owner_id": "%%user.id"
}
```

```
<html>
    <head>
        <script src="https://s3.amazonaws.com/stitch-sdks/js/library/v3/stable/stitch.min.js"></script>
        <script>
         const clientPromise = stitch.StitchClientFactory.create('test-sogwp');

         let client;
         let db;
         function displayCommentsOnLoad() {
             clientPromise.then(stitchClient => {
                 client = stitchClient;
                 db = client.service('mongodb', 'mongodb-atlas').db('blog');
                 return client.login().then(displayComments)
             });
         }

         function displayComments() {
             db.collection('comments').find({}).limit(1000).execute().then(docs => {
                 var html = docs.map(c => "<div>" + c.comment + "</div>").join("");
                 document.getElementById("comments").innerHTML = html;
             });
         }

         function addComment() {
             var foo = document.getElementById("new_comment");
             db.collection("comments").insertOne({owner_id : client.authedId(),
                                                  comment: foo.value}).then(displayComments);
             foo.value = "";
         }

        </script>
    </head>
    <body onLoad="displayCommentsOnLoad()">
        <h3>This is a great blog post</h3>
        <div id="content">
            I like to write about technology. Because I want to get on the front page of hacker news.
        </div>
        <hr>
        <div id="comments"></div>
        <hr>
        Add a Comment: <input id="new_comment"><input type="submit" onClick="addComment()">

    </body>
</html>
```

MongoDB Stitch



This is a great blog post

I like to write about technology. Because I want to get on the front page of hacker news.

first
second
third
zzz

Add a Comment:

Questions

Thank you