DATA COLLECTOR SERVICE

Αντωνίου Καλλιόπη 47441

Γεωργιλά Ιωάννα-Μαρία 47745

Κουτσοπούλου Αθηνά 47126

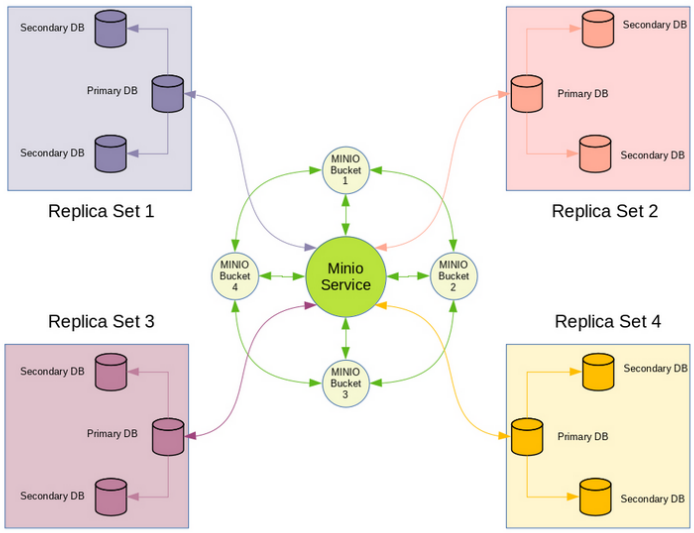
1. Σκοπός εργασίας

Η εργασία που επιλέχθηκε να πραγματοποιηθεί στα πλαίσια του εργαστήριου του μαθήματος Υπολογιστική Νέφους Υπηρεσιών, ήταν η δημιουργία και η ανάπτυξη δικτύου & μία βάσης δεδομένων τύπου NoSQL, και συγκεκριμένα η MongoDB με χρήση των εργαλείων της πλατφόρμας **swarmlab_hybrid**.

2. Αρχικοί στόχοι

Στο ξεκίνημα της ανάλυσης και ανάπτυξης της εργασίας μας θέσαμε αρχικούς στόχους :

- Δημιουργία δικτύου
- Δημιουργία Βάσης Δεδομένων
- Σχετική εκμάθηση λειτουργίας της MongoDB.



---

3. Πρώτη ενότητα

3.1. Βήματα δημιουργίας ενός δικτύου

- Συνδεόμαστε αρχικα στο Swarmalab-Hybrid.
- Στο Menu επιλέγουμε Private/Local έτσι ώστε να χρησιμοποιήσουμε ένα από τα εργαστήρια που παρέχονται.
- Κάνουμε Download Lab_Instance για το εργαστήριο hybrid_linux και στη συνέχεια επιλέγουμε Start Lan_Instance, αφού πρώτα κάνουμε start την υπηρεσία στο δίκτυο μας(./start.sh).
- Προσδιορίζουμε τον αριθμό των μηχανών που θέλουμε να έχει το δίκτυό μας ή/και την πόρτα που θέλουμε και πατάμε Up για να δημιουργηθούν.
- Αφού δούμε δεξιά στην κονσόλα ότι δημιουργήθηκαν επιλέγουμε στο μενού Container. Εκεί μπορούμε να δούμε και να διαχειριστούμε όλα τα containers που δημιουργήθηκαν.
- Για να συνδεθούμε σε κάποιο αυτά κάνουμε τα εξής βήματα:
  1. Στη στήλη Actions κάνουμε κλικ στο εικονίδιο του container που επιθυμούμε έτσι ώστε να εμφανιστούν τα χαρακτηριστικά του.
  2. Πατάμε Connect και εκτελούμε την εντολή που μας δείνει στο terminal.
  3. Τρέχοντας την εντολή ifconfig μπορούμε να δούμε το δίκτυο μας και με την παρακάτω εντολή μπορούμε να δούμε τα μέλη του.

     ```
     ./bin/swarmlab-nmap
     ```

3.1.1. Δημιουργία υπηρεσίας fluentd στον master για μετάδοση δεδομένων στους κόμβους

- Αλλάζω κατάλογο:

  ```
  cd courses/fluentd/
  ```

- Βλέπω τα αρχεία:

  ```
  ls -al
  ```

- Για να δω το configuration file του fluent:

  ```
  vim files/fluent.conf
  ls -al
  ```

3.2. Αυτόματη εγκατάσταση προγράμματος για τη συλλογή των δεδομένων

3.2.1. Το εργαλείο Fluentd

- Εγκαθιστούμε αυτόματα το ansible με την εντολή:

```
./fluentd.yml.sh
```

Το password που ζητείται είναι docker. SSH password: docker SUDO password: docker

- Γίνονται τα update και τα upgrade και μετά η εγκατάσταση των πακέτων από το yml file.Όταν τελειώσει μπορώ να μπω σε ενα container σε νέο τερματικό με ssh docker@ip . Δεν χρειάζονται κλειδιά.
- Για να δούμε αν τρέχει το fluentd τρέχουμε την εντολή:

```
ps aux
```

- Φτιάχνω ένα κατάλογο:

```
mkdir /var/log-in/cloud
```

- Τοποθετώ το αρχείο :

```
echo "hello world!" >> /var/log-in/cloud/hello
```

- Για να δω το αρχείο :

```
ls -al /tmp/mylog/
```

Θα χρησιμοποιήσουμε το nodejs για να φτιάξουμε ένα server με ένα πόρο post για να υποδέχεται το ερώτημα που έρχεται.

- Στο προηγούμενο τερματικό αλλάζω κατάλογο :

```
 cd ../nodeAppServer/
ls -al
```

3.3. OnEvent - Τοπική/Προσωρινή αποθήκευση των δεδομένων

Θα φτιάξουμε ένα application . Στο vim app.js χρησιμοποιούμε τα modules express και http και στη πόρτα 8000 θα φτιάξουμε τους πόρους get και post. Όταν το post θα πάρει κεντρικά μια μεταβλητή log θα τη διαβάσει και θα την επιστρέψει να την εμφανίσει. Στο vim app.js.sh εγκαθιστώ αρχικά το nodejs και το npm στη συνέχεια. Στο vim package.json εγκαθιστώ το express.

- Εγκαθιστώ το app:

```
./app.js.sh
password: docker
```

- Αφού γίνει η εγκατάσταση μπορώ να χρησιμοποιήσω το nodejs πατώντας:

```
node
(για να βγ πατάω 2 φορές ctrl+c)
```

- Βγαίνω και αντιγράφω την IP κάνοντας:

```
ifconfig
και τρέχω :
node app.js
```

Ξεκινάει να ακούει ο http server.

- Στο τερματικό του master εκτελώντας την παρακάτω εντολή :

```
curl http://172.21.0.2:8000/?log=hello
```

στέλνεται το μήνυμα που θα γράψουμε και θα εμφανιστεί στο άλλο τερματικό που είναι ο server(βάζουμε την ip του worker και μετά το log= βάζουμε το μήνυμα που θέλουμε).

- Στο άλλο τερματικό αφού βγω με ctrl+C με την εντολή:

```
set
```

βλεπουμε τις enviroment variables που έχουμε βάλει και δηλώνουν τον master.

- Βρίσκουμε το NODENAME και με την παρακάτω εντολή έχω τη διεύθυνση του master.:

```
echo $NODENAME
```

- Τρέχω ξανά τον server:

```
node app.js
```

- Στο τερματικό του άλλου μηχανήματος τρέχω:

```
curl http://hybrid-linux_master_1.hybrid-linux_hybrid-linux:8000/?log=hello
```

Αλλάζω δηλαδή την ip με την διεύθυνση του master και ετσι μεταφέρονται οι πληροφορίες σε όλα τα nodes.(όπου hybrid-linux_master_1.hybrid-linux_hybrid-linux είναι το nodename)

4. Δεύτερη ενότητα

4.1. MongoDB

4.2. Τι είναι η MongoDB

Η MongoDB ή αλλιώς Humongous Database που είναι το πλήρες όνομα της, είναι μία document-oriented (κειμενοστρεφής) βάση δεδομένων τύπου NoSQL που χρησιμοποιείται για αποθήκευση μεγάλου όγκου δεδομένων. Σε αντίθεση με τις κλασσικές σχεσιακές βάσης δεδομένων, που χρησιμοποιούν Πίνακες (tables) η MongoDB χρησιμοποιεί συλλογές (collections) και έγγραφα (documents). Τα έγγραφα αποτελούνται από ζευγάρια τιμών – κλειδιών (key-values) και αυτά συνθέτουν την βασική μονάδα δεδομένων στην MongoDB. Οι συλλογές περιέχουν σύνολα εγγράφων και λειτουργιών και είναι η αντιστοίχηση των πινάκων με μία σχεσιακή βάση δεδομένων.

4.2.1. Χαρακτηριστικά

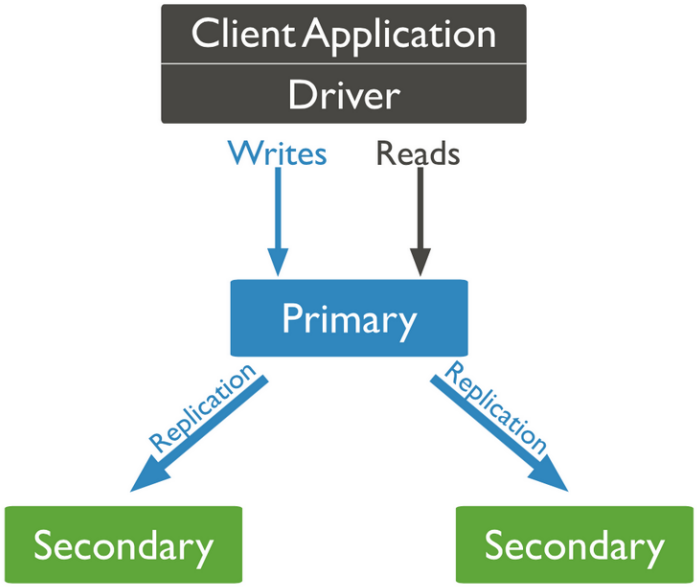
Μερικά από τα κύρια χαρακτηριστικά της είναι τα εξής:

- Κάθε βάση περιέχει collections που με την σειρά τους περιέχουν documents. Τα documents στο ίδιο collection μπορούν να έχουν διαφορετικό αριθμό πεδίων, διαφορετικό μέγεθος αλλά και περιεχόμενο μεταξύ τους.
- Τα documents συντάσσονται με την χρήση .json αρχείων και δεν χρειάζεται να έχουν από την αρχή κάποιο καθορισμένο Schema, αλλά ο κάθε προγραμματιστής μπορεί να δημιουργήσει τα πεδία που θέλει όταν θα τα χρειαστεί.

4.2.2. Γιατί MongoDB

Μερικοί από τους λόγους που κάποιος θα έπρεπε να αρχίσει να χρησιμοποιεί την MongoDB είναι οι παρακάτω:

- Document-oriented - Αφού η MongoDB είναι μία βάση δεδομένων τύπου NoSQL, και τα δεδομένα δεν αποθηκεύονται με σχεσιακό τρόπο αλλά με την χρήση documents της προσφέρει ευελιξία και καλύτερη προσαρμοστικότητα.
- Ad hoc queries – Η MongoDB υποστηρίζει αναζητήσεις σύμφωνα με κάποιο πεδίο, με εύρος τιμών καθώς και με χρήση Regular expression. Όλα τα queries (ερωτήματα) μπορούν να συνταχθούν έτσι ώστε να επιστρέφουν συγκεκριμένα μέσα από τα documents.
- Indexing – Προσφέρει δυνατότητα δημιουργίας ταξινόμησης από οποιοδήποτε πεδίο ενός document βελτιώνοντας σημαντικά την απόδοση στις αναζητήσεις.
- Load balancing – Η MongoDB χρησιμοποιεί την μέθοδο του Sharding (κατανομή των δεδομένων σε πολλαπλές μηχανές) για την οριζόντια κλιμάκωση της. Η MongoDB μπορεί να λειτουργεί σε πολλαπλά instances εξισορροπώντας το φορτίο και/ή αντιγράφοντας δεδομένα.
- Replication - Το MongoDB παρέχει υψηλή διαθεσιμότητα με σετ αντιγράφων. Ένα σύνολο αντιγράφων αποτελείται από δύο ή περισσότερα αντίγραφα των δεδομένων. Κάθε μέλος που αντιγράφεται μπορεί να ενεργεί στο ρόλο του πρωτεύοντος ή του δευτερεύοντος αντιγράφου ανά πάσα στιγμή. Όλες οι εγγραφές και οι αναγνώσεις γίνονται στο πρωτότυπο αντίγραφο από προεπιλογή. Τα δευτερεύοντα αντίγραφα διατηρούν ένα αντίγραφο των δεδομένων του πρωτεύοντος χρησιμοποιώντας ενσωματωμένη αναπαραγωγή. Όταν αποτυγχάνει ένα πρωτότυπο αντίγραφο, το σετ αντιγράφων διεξάγει αυτόματα μια διαδικασία εκλογής για να προσδιορίσει ποιο δευτερεύον θα πρέπει να γίνει το πρωτεύον. Οι δευτερεύοντες μπορούν προαιρετικά να προβάλλουν λειτουργίες ανάγνωσης, αλλά αυτά τα δεδομένα είναι τελικά τελικά συνεπή από προεπιλογή. Το replication είναι ένα απο τα ποιο σημαντικά στοιχεία και γι’αυτό αναλύεται περαιτέρω στην συνέχεια.



4.3. Replication in MongoDB

Η MongoDB σε αντίθεση με την MySQL προσφέρει την δυνατότητα, για αυτόματη ανάκαμψη για την περίπτωση που χαθεί η βάση, χωρίς όμως από την άλλη να χρειάζεται ιδιαίτερες γνώσεις για την εγκαθίδρυση του. Είναι η διαδικασία διάθεσης δεδομένων σε πολλούς διακομιστές δεδομένων. Η αναπαραγωγή δεδομένων διασφαλίζει την ασφάλεια δεδομένων όχι μόνο σε περίπτωση αποτυχίας ενός διακομιστή αλλά και σε περίπτωση αποτυχίας υλικού. Η αναπαραγωγή αποθηκεύει πολλά αντίγραφα δεδομένων σε διάφορους διακομιστές δεδομένων, επιτρέποντας στους χρήστες να ανακτούν δεδομένα όποτε απαιτείται. Το MongoDB υποστηρίζει την αναπαραγωγή με τη βοήθεια των Replica Sets. Αυτά τα Replica Sets είναι ένας συνδυασμός διαφόρων εμφανίσεων mongod, καθένα από τα οποία έχει έναν μόνο πρωτεύοντα κόμβο και πολλαπλούς δευτερεύοντες κόμβους. Ο δευτερεύων κόμβος αντιγράφει αυτόματα τις αλλαγές που έγιναν στον κύριο, διασφαλίζοντας ότι τα ίδια δεδομένα διατηρούνται σε όλους τους διακομιστές. Κατά τη διάρκεια μιας διαδικασίας ανακατεύθυνσης ή ανάκαμψης, ο εκλογικός μηχανισμός επιλέγει τον νέο κύριο κόμβο. Βασικά χαρακτηριστικά αναπαραγωγής:

Επεκτασιμότητα: Καθώς ο όγκος δεδομένων αυξάνει την πολυπλοκότητα της πρόσβασης σε δεδομένα και η εργασία με δεδομένα αυξάνεται επίσης. Με τη δυνατότητα αναπαραγωγής, είναι διαθέσιμα πολλά αντίγραφα δεδομένων, επιτρέποντας στους χρήστες όχι μόνο να αυξήσουν τα αποθεματικά δεδομένων τους, αλλά και να ανακτήσουν οποιαδήποτε προηγούμενη έκδοση σε περίπτωση σφαλμάτων ή αποτυχιών.

Απόδοση: Όταν τα δεδομένα είναι διαθέσιμα σε πολλούς υπολογιστές και διακομιστές, όχι μόνο διευκολύνει την πρόσβαση στα δεδομένα, αλλά διευκολύνει επίσης την ανάκτηση από απροσδόκητες και ξαφνικές αστοχίες. η αναπαραγωγή εξασφαλίζει διαθεσιμότητα και ασφάλεια δεδομένων ανά πάσα στιγμή.

Διαθεσιμότητα: Με την αναπαραγωγή σε ισχύ, δεν χρειάζεται να ανησυχείτε για αποτυχίες δεδομένων. Σε περιπτώσεις όπου η κύρια πηγή δεδομένων σας αποτυγχάνει, μπορείτε εύκολα να αποκτήσετε πρόσβαση στα ίδια ενημερωμένα δεδομένα από ένα δευτερεύον αποθεματικό. Αυτό προάγει ιδιαίτερα τη διαθεσιμότητα δεδομένων.

4.3.1. Δημιουργία Replica Set μέσω του Swarmlab Hybrid

- Στο Menu του Hybrid επιλέγουμε Private/Local → Storage έτσι ώστε να χρησιμοποιήσουμε το storage-mongo-replica που παρέχεται.
- Κάνουμε Download Lab_Instance και στη συνέχεια επιλέγουμε Start Lan_Instance
- Εκτελούμε την εντολή που μας δίνει για να συνδεθούμε.
- Το πρώτο που πρέπει να κάνουμε είναι να συνδέσουμε το δίκτυο που είναι οι βάσεις με το δίκτυο που είναι το fluentd.
- Στην ουσία έχουμε 2 σύννεφα, ένα με τη βάση mongo και ένα με τις μηχανές του Hybrid.
- Μπαίνουμε στον master εκτελώντας την εντολή που μας δίνει το connect στο Hybrid και αφού συνδεθούμε μπαίνουμε στον παρακάτω κατάλογο και περιμένουμε:

```
cd courses/fluentd
```

- Ανοίγουμε νέο τερματικό έτσι ώστε να συνδεθούμε σε έναν worker με:

```
ssh docker@ip (οπου ip η ip ενός από τους workers )
```

- Κάνουμε ping στη βάση swarmlabmongo1 και το αφήνουμε να τρέχει

```
ping swarmlabmongo1
```

- Στη συνέχεια, για να συνδέσουμε το fluent με τη βάση mongo, στο τερματικό του master τρέχουμε το config-update αρχείο του fluent με τους κωδικούς να είναι docker.

```
./fluentd-config-update.yml.sh
```

- Σε νέο τερματικό μπαίνουμε σε έναν worker με την εντολή που μας δίνει το connect στο Hybrid. Αφού συνδεθούμε γράφουμε αρκετές φορές:

```
echo "hello" >> /var/log-in/cloud/cloud.txt
```

- Με την παρακάτω εντολή βλέπουμε ότι δημιούργησε το αρχείο (buffer….):

```
ls -al /tmp/mylog/
```

- Αντιγράφουμε το αρχείο buffer και στη συνέχεια κάνουμε :

```
cat /tmp/mylog/όνομα_αρχειου που αντιγραψαμε.log
```

Σε αυτό το σημείο θα μας εμφανίσει το μήνυμα που γράψαμε.

- Επόμενο βήμα είναι το μήνυμα που εμφανίζει σε αυτό το σημείο να στέλνεται σε μια βάση. Σε άλλο τερματικό μπαίνουμε στο swarmlabmongo1 με την εντολη απο το *connect* και πατάμε **mongo** για να μπούμε. Φτιάχνουμε τη βάση:

```
use app_swarmlab
db.auth('app_swarmlab','app_swarmlab')
rs.status()
```

- Στο τερματικό του worker που γράψαμε εντολή echo ξανά τρέχουμε αρκετές φορές την εντολή αυτή μέχρι να περάσουν 20 δευτερόλεπτα (μέχρι δηλαδή να αδειάσει ο buffer).
- Τέλος, πάμε στη mongo βάση μας και εκτελούμε την παρακάτω εντολή έτσι ώστε να δούμε ότι τα μηνύματα έχουν καταγραφεί:

```
db.logs.find()
```

4.3.2. Key Features of Replica Set

Ορισμένα βασικά χαρακτηριστικά του Replica Set είναι τα εξής:

```
->   Το Replica Set είναι ένα συμπλεγμα N κομβων με διάφορους κομβους που φέρουν
δεδομένα και έναν κομβο arbiter.
->   Με το Replica Set, οποιοσδήποτε κομβος μπορεί να είναι ένας κυριος κομβος
διακομιστή.
->   Θα έχετε αυτοματη ανακατε□υνση και ανάκτηση με το Replica Set.
->   Οταν εκτελουμε οποιαδήποτε λειτουργία εγγραφής με το Replica Set, μεταβαίνει στον
κυριο κομβο του διακομιστή.
```

4.3.3. Mongo replicas

Η διαδικασία με την οποία η MongoDB πραγματοποιεί το replication είναι με την δημιουργία το λιγότερο 2 ακόμα MongoDB instances (είτε τοπικά είτε μέσω δικτύου), σύνδεση των instances μεταξύ τους και ορισμός Primary και Secondaries.

4.4. Δημιουργία Replica Set μέσω terminal

4.4.1. Δημιουργία Δικτύων

Ξεκινήσαμε με την δημιουργία κατάλληλων δικτυώσεων.

- Ένα δίκτυο για κάθε replica set (1 primary 2 secondary).
- Ένα δίκτυο που θα βρίσκονται όλα τα primary μαζί.

4.5. Δημιουργία Container

Δημιουργήθηκαν 9 container και ποιο συγκεκριμένα 3 Replica Set των 3 container έκαστος.
Για κάθε μέλος του Replica Set δόθηκαν οι κατάλλήλες ρυθμίσεις για την προετοιμασία ένταξης του στην ομάδα του.

4.5.1. Ονοματοδοσία

Σε κάθε container δόθηκε ένα μοναδικό όνομα κατάλληλο για την δική μας διευκόλυνση και ευκολία.

- Primary1
- secondary1_1
- secondary1_2
- Primary2
- secondary2_1
- secondary2_2
- Primary3
- secondary3_1
- secondary3_2

4.5.2. Ports

Σε κάθε container αντιστοιχήθηκε η port 27017 με μία port του εκάστοτε host που έτρεχε και για δική μας ευκολία έγινε πάλι ομαδοποίηση ανά replica set.

- Primary1 = 10001
- secondary1_1 = 10002
- secondary1_2 = 10003
- Primary2 = 20001
- κοκ.

4.5.3. Networks

Δόθηκαν οι κατάλληλες ρυθμίσεις για την ένταξη του στο ανάλογο δίκτυο. Έτσι το κάθε ReplicaSet που γίνεται deploy, καταλαβαίνουμε αν έχουν σηκωθεί τα Replica set και έπειτα "ενώνεται" το δίκτυο των Primary και το δίκτυο του Replica set), ενώ τα secondary μόνο στο δίκτυο της Replica που ακήκουν.

```
example:
    primary_network:
    replica_setX_network:
```

4.5.4. Εντολή έναρξης

Για εντολή έναρξης της βάσης δοκιμάστην διάφορες επιλογές όπως command, Entrypoint, αλλά ενώ κάποιες επιλογές εξυπηρετούσαν τον σκοπό τους ταυτόχρονα δημιουργούσαν πρόβλημα σε κάποιες άλλες. Έτσι καταλληξαμε στην χρήση την εντολής command με χρήση ορισμάτων μόνος ως προς την ονοματοδοδία και την ενεργοποίησει του Replica Set

```
command: ["mongod", "--replSet", "rep-setX"]
```

4.6. Replica Set files/commands

_ΒΗΜΑ 1: Δημιουργία docker-compose.yaml

```
Αρχείο docker-compose.yaml
```

_ΒΗΜΑ 2: Start the mongo databases

```
$ docker-compose up
or
$ docker-compose up -d
```

Για την ένωση των δικτύων ReplicaSet με τα container τους και τα secondary με τα primary των ReplicaSet.

```
$ ./deploy.sh
```

_ΒΗΜΑ 3: Exec into one of the mongos:

Για να δούμε τα περιεχόμενα/container που έχουν δημιουργηθεί.

```
$ docker ps
```

Εντολές:

```
$ docker exec -it replicaset_primary1_1 /bin/bash
$ docker exec -it replicaset_secondary1_1_1 /bin/bash
$ docker exec -it replicaset_secondary1_2_1 /bin/bash
$ docker exec -it replicaset_primary2_1 /bin/bash
$ docker exec -it replicaset_secondary2_1_1 /bin/bash
$ docker exec -it replicaset_secondary2_2_1 /bin/bash
$ docker exec -it replicaset_primary3_1 /bin/bash
$ docker exec -it replicaset_secondary3_1_1 /bin/bash
$ docker exec -it replicaset_secondary3_2_1 /bin/bash
```

_ΒΗΜΑ 4: Access mongo console

Εντολή:

```
mongo
```

_ΒΗΜΑ 5: Configure replica set by pasting the following

```
Αρχεία json:
rep-set0.js
rep-set1.js
rep-set2.js
```

_ΒΗΜΑ 6: Use Database

Για την εισαγωγή μας σε μία βάση δεδομένων, χρησιμοποιούμε την εξής εντολή:

```
$ use database_name
example: $ use name_db
```

όπου με την χρήση της εντολής μπορούμε να συνδεθούμε απευθείας σε αυτή κι αν δεν υπάρχει τη δημιουργεί αυτόματα.

_ΒΗΜΑ 7: Data Collection

Ωστόσο στο κάθε που έχουμε δημιουργήσει, κάνουμε προσθήκη δεδομένων,

```
$ db.databse_name.insert({"Name":"PROJECT","Title": "Cloud"})
```

Και με την εντολή :

```
$ db.databse_name.find()
```

Εμφανίζει όλες τις καταχωρήσεις/δεδομένα.

4.6.1. Ενεργοποίηση αναπαραγωγής στο MongoDB_*

Εντολή έναρξης:

```
$ rs.initiate ()
```

Έτσι μπορούμε να χρησιμοποιήσουμε την εντολή rs.initiate () για να ξεκινήσετε τη διαδικασία αναπαραγωγής. Το Mongo Shell θα αλλάξει τώρα την προτροπή του στο όνομα του replicaSet μας, replicaSet1.

4.6.2. Προσθήκη παρουσιών MongoDB σε Set Replica_*

Εντολή προσθήκης:

```
$ rs.add(<servername:port>)
```

For example:

```
rs.add('node-2:27017')
rs.add('node-3:27017')
```

4.6.3. Status

Για να ελέγξουμε την κατάσταση της αναπαραγωγής, χρησιμοποιούμε την εντολή κατάστασης ως εξής:

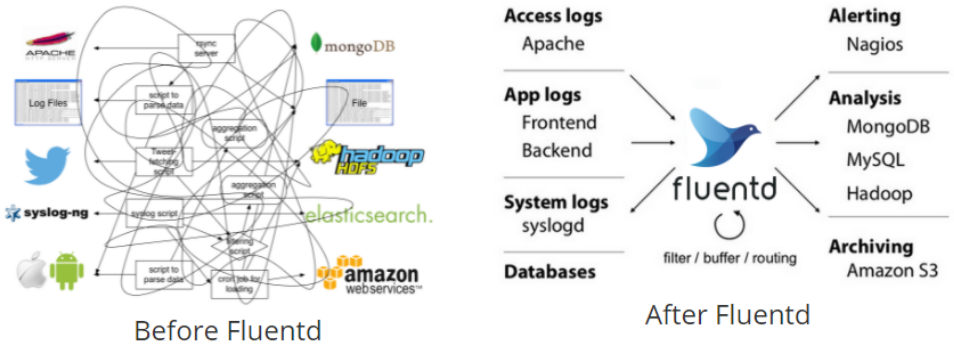
```
$ rs.status()
```

4.6.4. Connect to specific member

Εντολή :

```
$ docker exec -it fileprojectname_membername /bin/bash
$ mongo
```

4.7. Fluentd



4.7.1. Τι είναι το Fluentd?

Το Fluentd είναι ένας συλλέκτης δεδομένων ανοιχτού κώδικα, ο οποίος σας επιτρέπει να ενοποιήσετε τη συλλογή και την κατανάλωση δεδομένων για καλύτερη χρήση και κατανόηση των δεδομένων. Εδώ είναι τα βασικά χαρακτηριστικά:

- Unified Logging with JSON: Το Fluentd προσπαθεί να δομήσει δεδομένα όσο το δυνατόν περισσότερο JSON: αυτό επιτρέπει στο Fluentd να ενοποιήσει όλες τις πτυχές της επεξεργασίας δεδομένων καταγραφής: συλλογή, φιλτράρισμα, αποθήκευση και αποθήκευση αρχείων καταγραφής σε πολλές πηγές και προορισμούς (Unified Logging Layer). Η επεξεργασία δεδομένων κατάντη είναι πολύ πιο εύκολη με το JSON, καθώς έχει αρκετή δομή για να είναι προσβάσιμη διατηρώντας παράλληλα ευέλικτα σχήματα.
- Pluggable Architecture: Το Fluentd διαθέτει ένα ευέλικτο σύστημα προσθηκών που επιτρέπει στην κοινότητα να επεκτείνει τη λειτουργικότητά της. Τα 500+ πρόσθετα που συνεισφέρει η κοινότητα συνδέουν δεκάδες πηγές δεδομένων και εξόδους δεδομένων. Αξιοποιώντας τις προσθήκες, μπορείτε να αρχίσετε να χρησιμοποιείτε καλύτερα τα αρχεία καταγραφής σας αμέσως.
- Built-in Reliability: Το Fluentd υποστηρίζει την αποθήκευση μνήμης και αρχείων με βάση το αρχείο για την αποφυγή απώλειας δεδομένων μεταξύ κόμβων. Το Fluentd υποστηρίζει επίσης ισχυρό failover και μπορεί να ρυθμιστεί για υψηλή διαθεσιμότητα. 2.000+ εταιρείες βάσει δεδομένων βασίζονται στο Fluentd για να διαφοροποιήσουν τα προϊόντα και τις υπηρεσίες τους μέσω καλύτερης χρήσης και κατανόησης των δεδομένων καταγραφής τους.