MongoDB: deploy a 3-nodes Replica Set with internal encryption

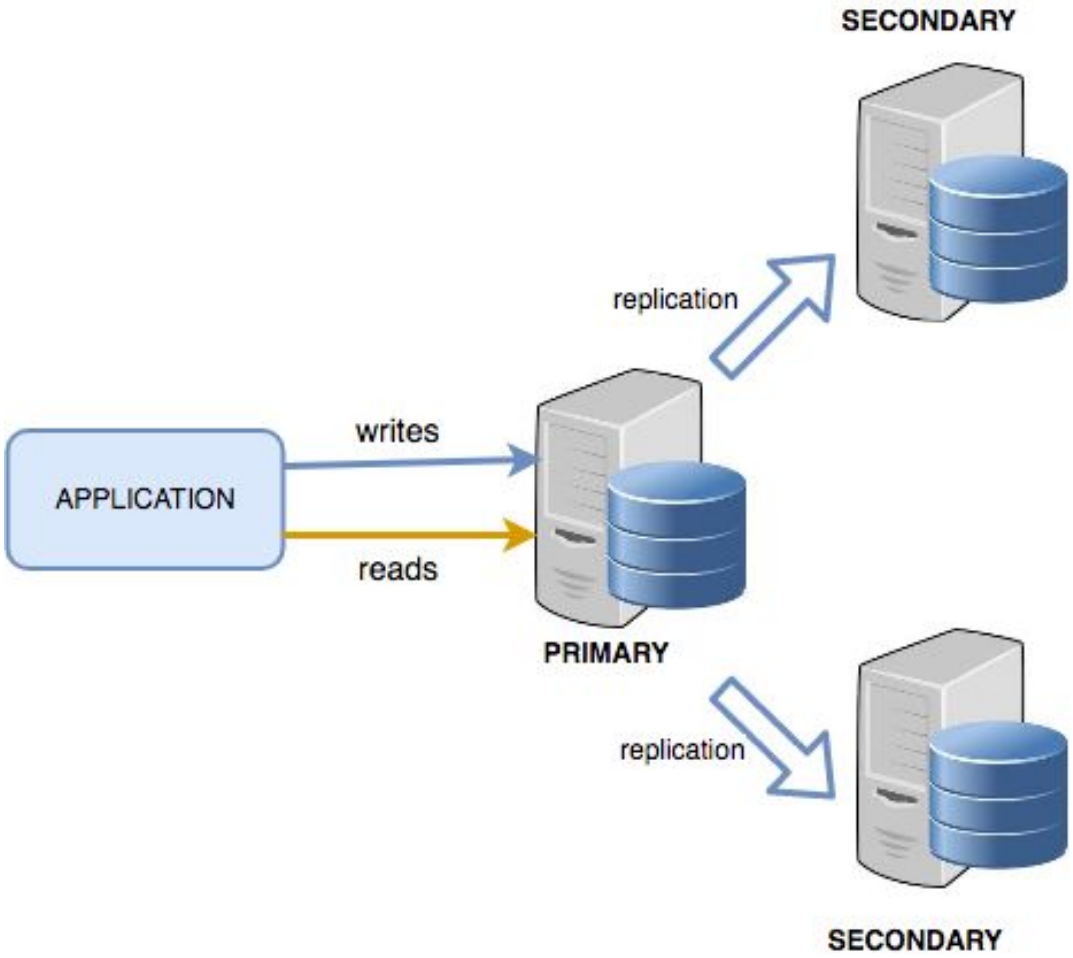
Corrado Pandiani

Barcelona, 22 June 2018

What is a Replica Set

- Group of mongod processes that maintain the same data set
- Provide redundancy and high availability
- Basis for all production deployments
- Can provide increased read capacity as clients can send read operations to different servers
- Automatic failover
- Internals are similar (more or less) to MySQL's
  - Asyncronous replication
  - Events are replicated reading a primary node's collection: **oplog (binlog)**
  - **Primary = Master**
  - **Secondary = Slave**
  - **Arbiter = Slave** (without data)
- MongoDB provides HA by-design
  - Odd number of nodes requested

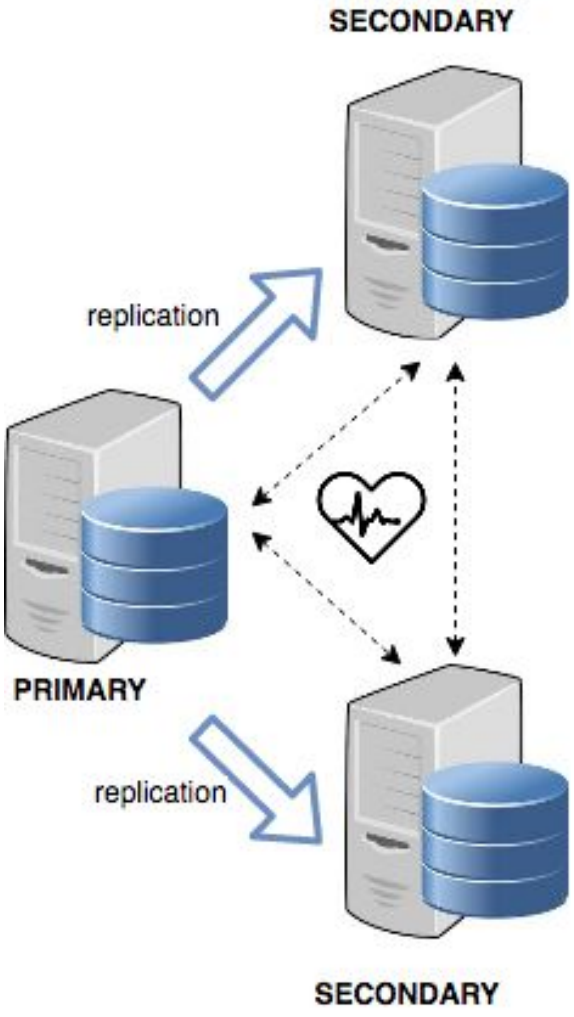
A 3-nodes RS


SECONDARY
replication
writes
APPLICATION
reads
PRIMARY
replication
SECONDARY

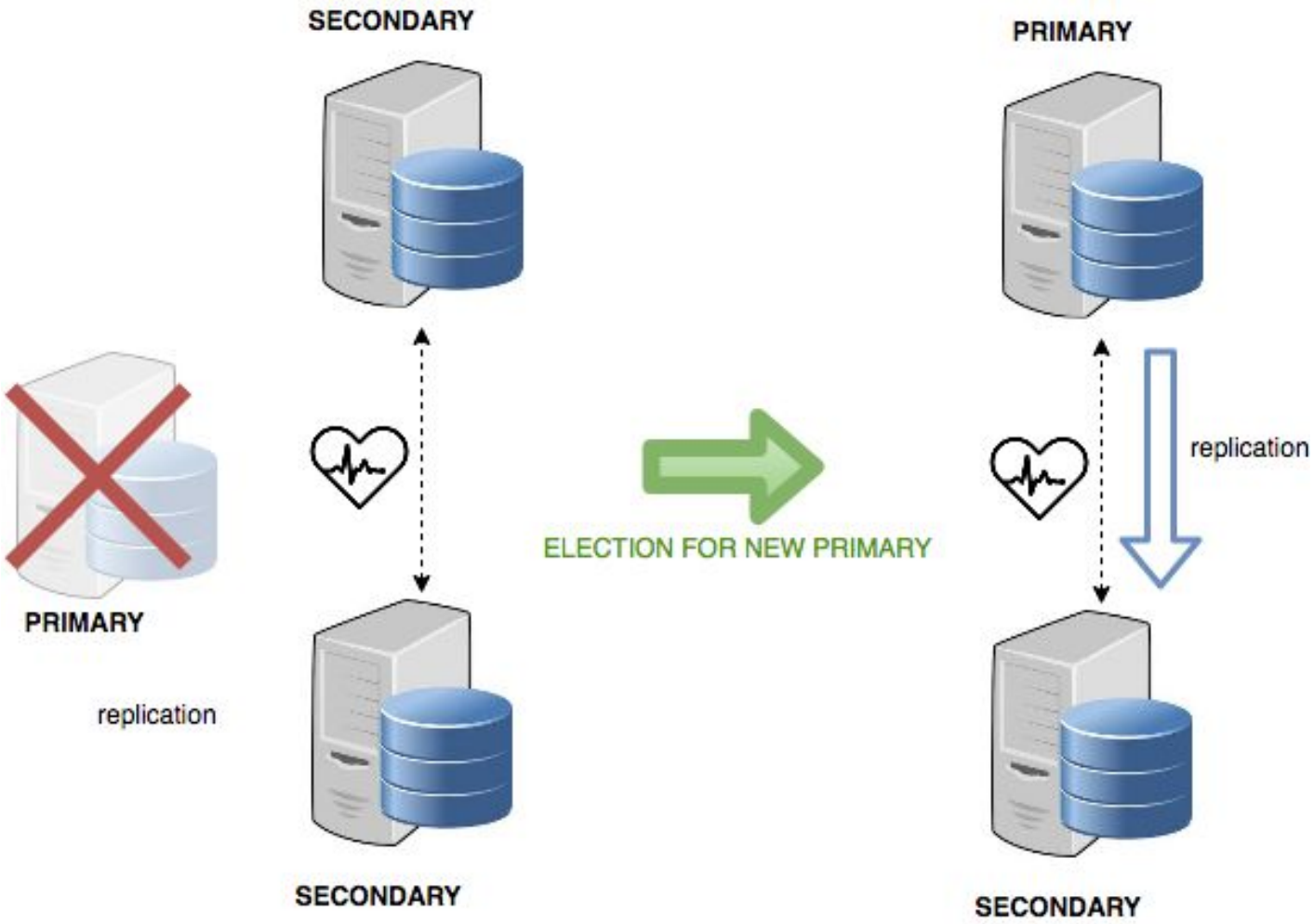
Reads

- By default, clients read from the primary
- Clients can specify a read preference to send read operations to secondaries
- Backups from secondaries
- Data returned by secondaries could not reflect the primary's state because of async replication

How RS works



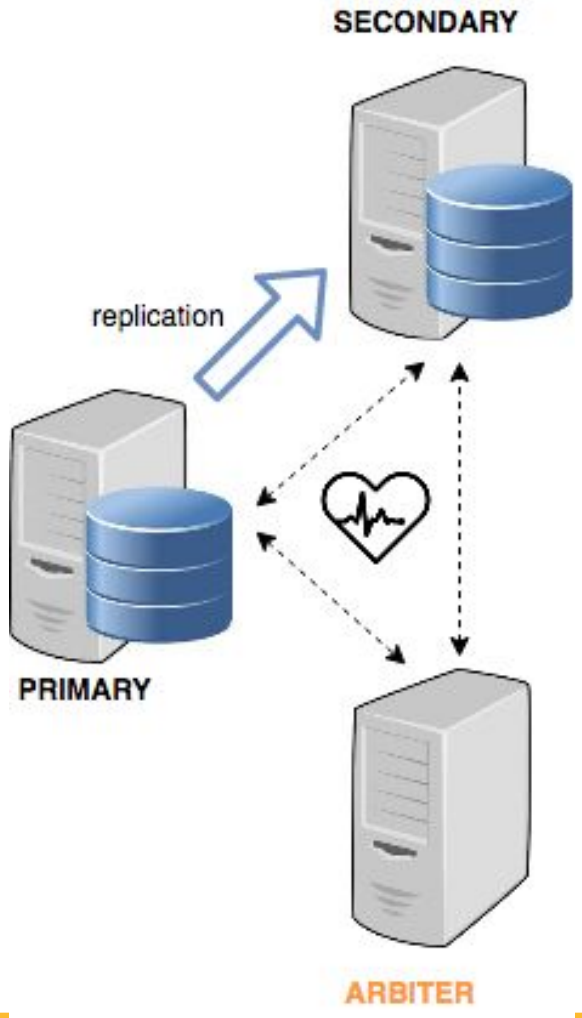
Automatic failover



Automatic failover

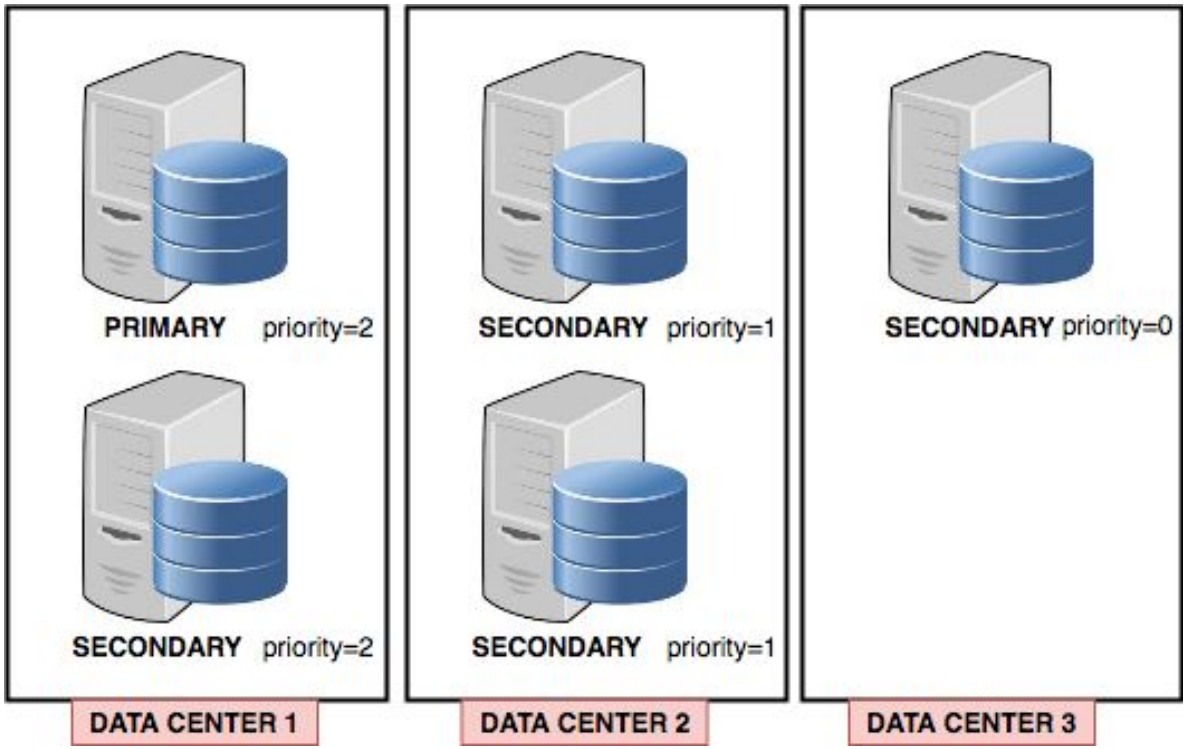
- When a primary does not communicate with the other members
- `electionTimeoutMillis` period (10 seconds by default)
- The cluster attempts to complete the election of a new primary and resume normal operations
- The replica set cannot process write operations until the election completes successfully
- The replica set can continue to serve read queries if such queries are configured to run on secondaries while the primary is offline
- An eligible secondary calls for an election to nominate itself as the new primary

Arbiter node



Other features: priority

- Priority = Weight of the node
- Define which nodes can be elected as Primary
- Replica Set with members in multiple data centers



Other features: hidden

- **HIDDEN SECONDARY**
  - Maintains a copy of the primary's data
  - Invisible to client applications
  - Run Backups, Statistics or special tasks
  - Must be priority = 0 : cannot be elected as Primary but votes during election
- **DELAYED SECONDARY**
  - Reflects earlier state of the dataset
  - Recover from unsuccessful application upgrades and operator errors, Backups
  - Must be priority = 0 : cannot be elected as Primary but votes during election

Deploy a 3-nodes Replica Set

Step-by-step guide

Environment

- We have 3 Linux machines
- Ubuntu 16.04
- Percona Server for MongoDB 3.4.13
- Members:
  - `psmdb1 : 192.168.56.101`
  - `psmdb2 : 192.168.56.102`
  - `psmdb3 : 192.168.56.103`

Connectivity

- We need to ensure that each node is accessible by the others on port 27017 (mongodb default)
- Since the members are (usually) on the same network we can simply test the connectivity between nodes to be sure
  - `psmdb1> mongo --host 192.168.56.102 --port 27017`
  - `psmdb1> mongo --host 192.168.56.103 --port 27017`
  - `psmdb2> mongo --host 192.168.56.103 --port 27017`
  - …

Hostnames

- Ensure that each member of a replica set is accessible by way of resolvable DNS or hostnames.
- This is also very important to configure properly the encryption using X.509 certificates.
- Set up your systems' `/etc/hosts`

```
root@psmdb2:~# cat /etc/hosts
127.0.0.1       localhost

192.168.56.101  psmdb1
192.168.56.102  psmdb2
192.168.56.103  psmdb3
```

Choose a name for the Replica Set

- Every member needs to be configured to access a certain Replica Set by setting simply its name
- We decide our Replica Set name could be **rs-test**
- Set the RS name into `/etc/mongod.conf`

```
replication:
   replSetName: “rs-test”
```

- Restart the server on each member

```
sudo service mongod restart
```

Initiate replication

- Connect to one of the member
- Initiates the RS using rs.initiate() to configure what are the members

```
mongo> rs.initiate( {
… _id: “rs-test”,
… members: [
… { _id: 0, host: “psmdb1:27017” },
… { _id: 1, host: “psmdb2:27017” },
… { _id: 2, host: “psmdb3:27017” }
… ] })
```

- MongoDB initiates a replica set, using the default replica set configuration

View RS configuration rs.conf()

```
rs-test:PRIMARY> rs.conf()
{
    "_id" : "rs-test",
    "version" : 68835,
    "protocolVersion" : NumberLong(1),
    "members" : [
        {
                "_id" : 0,
                "host" : "psmdb1:27017",
                "arbiterOnly" : false,
                "buildIndexes" : true,
                "hidden" : false,
                "priority" : 1,
                "tags" : {

                },
                "slaveDelay" : NumberLong(0),
                "votes" : 1
        },
        {
                "_id" : 1,
                "host" : "psmdb2:27017",
                "arbiterOnly" : false,
                "buildIndexes" : true,
                "hidden" : false,
                "priority" : 1,
                "tags" : {

                },
                "slaveDelay" : NumberLong(0),
                "votes" : 1
        },
        {
                "_id" : 2,
                "host" : "psmdb3:27017",
                "arbiterOnly" : false,
                "buildIndexes" : true,
                "hidden" : false,
                "priority" : 1,
                "tags" : {

                },
                "slaveDelay" : NumberLong(0),
                "votes" : 1
        }
    ],
    "settings" : {
        "chainingAllowed" : true,
        "heartbeatIntervalMillis" : 2000,
        "heartbeatTimeoutSecs" : 10,
        "electionTimeoutMillis" : 10000,
        "catchUpTimeoutMillis" : 60000,
        "getLastErrorModes" : {

        },
        "getLastErrorDefaults" : {
                "w" : 1,
                "wtimeout" : 0
        },
        "replicaSetId" : ObjectId("5aa2600d377adb63d28e7f0f")
    }
}
```

View RS status
rs.status()

```
rs-test:PRIMARY> rs.status()
{
    "set" : "rs-test",
    "date" : ISODate("2018-04-25T10:32:52.675Z"),
    "myState" : 1,
…
…
…
    },
    "members" : [
        {
                "_id" : 0,
                "name" : "psmdb1:27017",
                "health" : 1,
                "state" : 2,
                "stateStr" : "SECONDARY",
                "uptime" : 1288,
                "optime" : {
                        "ts" : Timestamp(1524652365, 1),
                        "t" : NumberLong(42)
                },
                "optimeDurable" : {
                        "ts" : Timestamp(1524652365, 1),
                        "t" : NumberLong(42)
                },
                "optimeDate" :
ISODate("2018-04-25T10:32:45Z"),
                "optimeDurableDate" :
ISODate("2018-04-25T10:32:45Z"),
                "lastHeartbeat" :
...
```

```
{
        "_id" : 1,
        "name" : "psmdb2:27017",
        "health" : 1,
        "state" : 1,
        "stateStr" : "PRIMARY",
        "uptime" : 1303,
        "optime" : {
                "ts" : Timestamp(1524652365, 1),
                "t" : NumberLong(42)
        },
        "optimeDate" : ISODate("2018-04-25T10:32:45Z"),
        "electionTime" : Timestamp(1524651084, 1),
        "electionDate" : ISODate("2018-04-25T10:11:24Z"),
        "configVersion" : 68835,
        "self" : true
 },
…
…
…
```

Test replication

- Connect to PRIMARY member and create some document

```
rs-test:PRIMARY> use test
switched to db test

rs-test:PRIMARY> db.barcelona.insert( {name:"Corrado", surname:"Pandiani"} )
WriteResult({ "nInserted" : 1 })

rs-test:PRIMARY> db.barcelona.find().pretty()
{
     "_id" : ObjectId("5ae05ac27e6680071caf94b7"),
     "name" : "Corrado",
     "surname" : "Pandiani"
}
```

Test replication

- Connect to a SECONDARY member and look for the replicated document

```
rs-test:SECONDARY> use test
switched to db test

rs-test:SECONDARY> show collections
2018-04-25T12:42:38.678+0200 E QUERY    [thread1] Error: listCollections failed: {
    "ok" : 0,
    "errmsg" : "not master and slaveOk=false",
    "code" : 13435,
    "codeName" : "NotMasterNoSlaveOk"
} :
_getErrorWithCode@src/mongo/shell/utils.js:25:13
DB.prototype._getCollectionInfosCommand@src/mongo/shell/db.js:807:1
DB.prototype.getCollectionInfos@src/mongo/shell/db.js:819:19
DB.prototype.getCollectionNames@src/mongo/shell/db.js:830:16
shellHelper.show@src/mongo/shell/utils.js:775:9
shellHelper@src/mongo/shell/utils.js:672:15
@(shellhelp2):1:1
```

Error ?!

Test replication

- Enable slaveOk()

```
rs-test:SECONDARY> rs.slaveOk()

rs-test:SECONDARY> show collections
barcelona
foo
restaurants
```

- Let's look for the document now

```
rs-test:SECONDARY> db.barcelona.find().pretty()
{
    "_id" : ObjectId("5ae05ac27e6680071caf94b7"),
    "name" : "Corrado",
    "surname" : "Pandiani"
}
```

Replica Set

internal encryption

Replica Set internal encryption

- MongoDB supports x.509 certificate authentication for use with a secure TLS/SSL connection
- Replica set members can use x.509 certificates to verify their membership to the replica set
- Enabling internal authentication also enables **Role-Base Access Control (RBAC)**
- Clients must authenticate as a user in order to connect and perform operations in the deployment.

Create an admin user

```
rs-test:PRIMARY> use admin
switched to db admin
rs-test:PRIMARY> db.createUser({user:
'admin', pwd: 'secret', roles:['root']})
Successfully added user: { "user" : "admin",
"roles" : [ "root" ] }
```

Enable RBAC

Put the following into `/etc/mongod.conf`

```
security:
      authorization: enabled
```

Restart the server and connect using the admin user

```
sudo service mongod restart
```

```
mongo -u admin -p secret
--authenticationDatabase "admin"
```

Certificate requirements

- Any certificate of any single member needs to be signed by the same *Certification Authority* (CA)
- **The *Common Name* (CN) must be the same as the hostname of the specific machine MongoDB is running on (IMPORTANT)**
- The *Distinguished Name* (DN), found in the member certificate's subject, must specify a non-empty value for *at least one* of the following attributes: Organization (O), the Organizational Unit (OU) or the Domain Component (DC).
- The *Organization* attributes (O's), the *Organizational Unit* attributes (OU's), and the *Domain Components* (DC's) must match those from the certificates for the other cluster members.
- Also we need to have **openssl** installed on our systems

Self-signed certificates

- Using a public CA to sign certificates can be quite costly
- Using a public CA is not necessary inside a private infrastructure
- We will create a *self-signed* CA certificate and sign the individual certificates with it.

Generate a new private key

- Connect to one of the node
- Generate a new 8192 bit private key with OpenSSL and save it as mongoCA.key. Use the following command:

```
openssl genrsa -out mongoCA.key -aes256
8192
```

- When requested choose a really strong passphrase !

Sign a new CA certificate

- During certificate creation, some fields have to be filled out. They can be chosen arbitrarily but may correspond to your organization's details.

```
root@psmdb1:~# openssl req -x509 -new -extensions v3_ca -key mongoCA.key -days 365 -out
mongoCA.crt
Enter pass phrase for mongoCA.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:IT
State or Province Name (full name) [Some-State]:Milan
Locality Name (eg, city) []:San Donato Milanese
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Percona
Organizational Unit Name (eg, section) []:DBA
Common Name (e.g. server FQDN or YOUR name) []:psmdb1
Email Address []: corrado.pandiani@percona.com
```

Issue certificates for MongoDB instances

- First generate certificate requests
- Then sign them using our CA certificate
- For the first server run the following commands

```
openssl req -new -nodes -newkey rsa:4096 -keyout psmdb1.key -out psmdb1.csr

openssl x509 -CA mongoCA.crt -CAkey mongoCA.key -CAcreateserial -req -days 365 -in
psmdb1.csr -out psmdb1.crt

cat psmdb1.key psmdb1.crt > psmdb1.pem
```

- For other servers use these commands changing `psmdb1` to `psmdb2` and to `psmdb3` (hostnames)
- Remember: when requested, fill out the same organization data but CN must be the same as the hostname of the machine

Place files

- Copy `mongoCA.crt` file to each member
- Copy generated `<hostname>.pem` files to the relative member
- Create on each member a directory which only the MongoDB user can read and copy both files there

```
sudo mkdir -p /etc/mongodb/ssl
sudo chmod 700 /etc/mongodb/ssl
sudo chown -R mongod:mongod /etc/mongodb
sudo cp psmdb1.pem /etc/mongodb/ssl/
sudo cp mongoCA.crt /etc/mongodb/ssl/
```

Configure mongod

- Change `/etc/mongod.conf` on each server

```
net:
    port: 27017
    ssl:
        mode: requireSSL
        PEMKeyFile: /etc/mongodb/ssl/psmdb1.pem
        CAFile: /etc/mongodb/ssl/mongoCA.crt
        clusterFile: /etc/mongodb/ssl/psmdb1.pem
security:
    authorization: enabled
    clusterAuthMode: x509
```

- Restart MongoDB

```
service mongod restart
```

Connect to the server

- Connect to MongoDB on one of the members

```
mongo admin --ssl --sslCAFile
/etc/mongodb/ssl/mongoCA.crt --sslPEMKeyFile
/etc/mongodb/ssl/psmdb1.pem -u admin -p 1234 --host
psmdb1
```

- Note: we had to create a user to access the database because using encryption requires to activate the Role-Based Access Control
- Note: not only we encrypted internal connections, also client connections are

Blog posts

https://www.percona.com/blog/2018/05/17/mongodb-replica-set-transport-encryption-part-1/

https://www.percona.com/blog/2018/05/23/deploy-mongodb-replica-set-with-transport-encryption-part-2/

https://www.percona.com/blog/2018/05/31/mongodb-deploy-replica-set-with-transport-encryption-part-3/

THANK YOU

corrado.pandiani@percona.com