MongoDB – Big Data mit Open Source

CommitterConf Essen 2014

29. Oktober 2014

Tilman Beitter
Linux Consultant & Trainer
B1 Systems GmbH
beitter@b1-systems.de

Vorstellung B1 Systems

- gegründet 2004
- primär Linux/Open Source-Themen
- national & international tätig
- über 60 Mitarbeiter
- unabhängig von Soft- und Hardware-Herstellern
- Leistungsangebot:
  - Beratung & Consulting
  - Support
  - Entwicklung
  - Training
  - Betrieb
  - Lösungen
- dezentrale Strukturen

Schwerpunkte

- Virtualisierung (XEN, KVM & RHEV)
- Systemmanagement (Spacewalk, Red Hat Satellite, SUSE Manager)
- Konfigurationsmanagement (Puppet & Chef)
- Monitoring (Nagios & Icinga)
- IaaS Cloud (OpenStack & SUSE Cloud & RDO)
- Hochverfügbarkeit (Pacemaker)
- Shared Storage (GPFS, OCFS2, DRBD & CEPH)
- Dateiaustausch (ownCloud)
- Paketierung (Open Build Service)
- Administratoren oder Entwickler zur Unterstützung des Teams vor Ort

Partner


SUSE
PartnerNet
Platinum
CERTIFIED
SUSE PARTNER


redhat
PREMIER
BUSINESS
PARTNER


ubuntu
Advantage reseller


puppet
labs


pfSense


Nagios
Authorized
Reseller


Hortonworks


ARISTA


rackspace
the open cloud company


ownCloud
YOUR CLOUD, YOUR DATA, YOUR WAY!
Gold Partner

Einführung in MongoDB

Was ist MongoDB?

- Open Source NoSQL Datenbank
- schemafrei
- skalierbar
- dokumentenorientiert
- entwickelt durch MongoDB Inc.

Vergleich RDBMS und NoSQL

Vergleich RDBMS und MongoDB-NoSQL

RDBMS	**NoSQL**
Database	Database
Table	Collection
Index	Index
Row	BSON Document
Column	BSON Field
Join	Embedding and Linking
Primary Key	_id Field

NoSQL

- „Not Only SQL“
- nicht relational
- keine Tabellenschemata
- Vermeidung von JOINs
- horizontale Skalierung

Das BSON Format

- „Binary JSON“
- bietet mehr Datentypen als JSON
- ermöglicht Speicherung von Binärdaten
- entwickelt für MongoDB
- erweitert zur offiziellen Spezifikation (bsonspec.org)

Das BSON Format

Beispieldokument im BSON-Format:

```
{
  "Benutzername": "linus.torvalds",
  "Name": {
    "Vorname": "linus",
    "Nachname": "torvalds",
  },
  "ÜbergeordneterChef": {},
  "Kollegen": [ "s.example0", "g.example1" ],
  "IstAbteilungsleiter": true,
  "Abteilungen": null,
}
```

Systemkonfiguration

Hardwarevoraussetzungen

Minimale Systemanforderungen:

- Architektur:
  - 64bit Betriebssystem
  - Multicore CPU
- Arbeitsspeicher und Speicherplatz:
  - MongoS: OS-Empfehlung
  - Configserver: OS-Empfehlung
  - Datenbankserver: abhängig von Datenbankgröße

Basiskonfiguration

Anpassung der Systemkonfiguration

- Ulimits
- Readahead
- Swap
- Filesystem
- Hugepages

Dateien und Ports

- Hauptkonfigurationsdatei: `/etc/mongodb.conf`
- Logdatei: `/var/log/mongodb/mongodb.log`
- Standardports (TCP):
  - 27017: `mongod` & `mongos`
  - 27018: monood mit –shardsvr
  - 27019: mongod mit –configsvr
  - 27017: mongod status page (= port + 1000)

Komponenten der MongoDB

Komponenten und Begrifflichkeiten 1/2

- `mongod`
- `mongos`
- ConfigServer
- ReplicaSets
- Sharding

Komponenten und Begrifflichkeiten 2/2

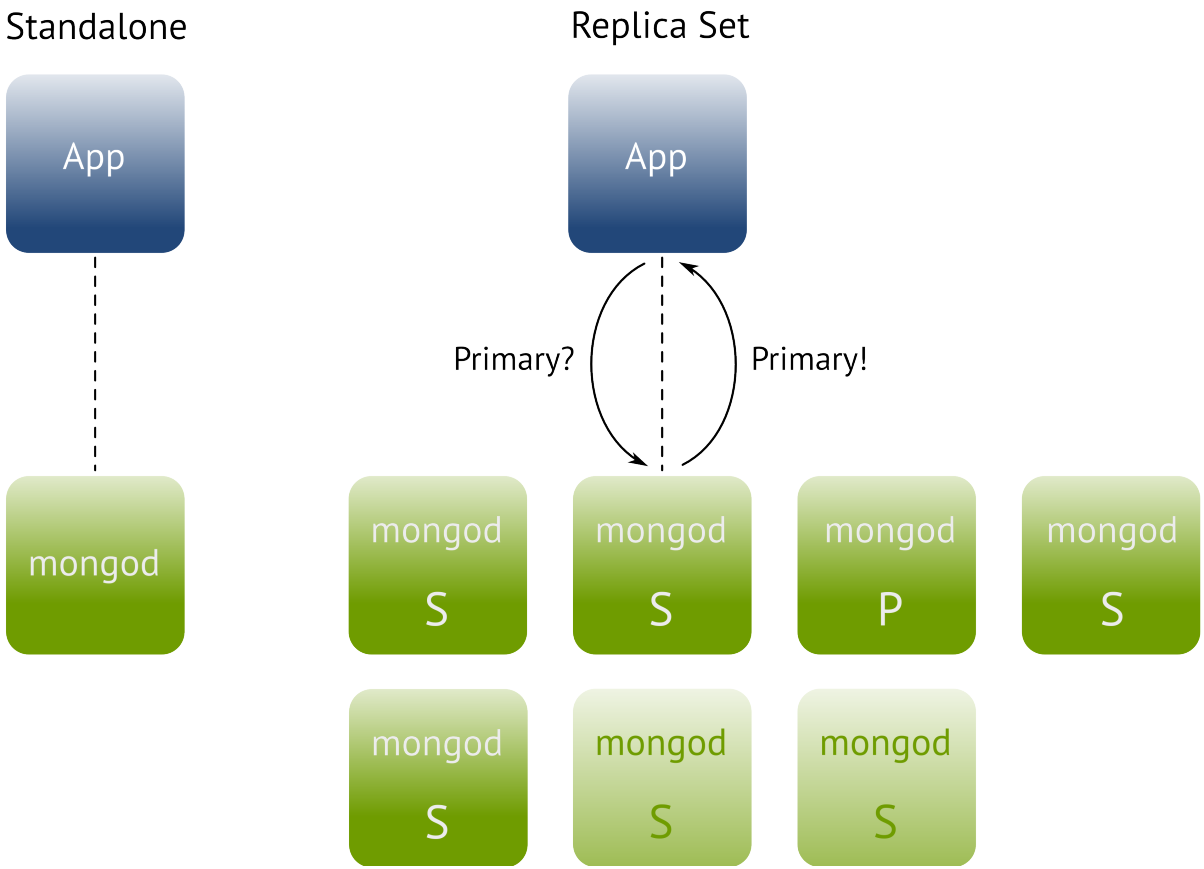


Abbildung : MongoDB Standalone/ReplicaSet

Aufbau – MongoDB Standalone

Standalone-Instanz

- Testzwecke
- Entwicklung
- kleine Applikationen (keine Redundanz!)

Konfiguration Standalone

Basiskonfiguration:

```
# cat /etc/mongodb.conf
fork = true
port = 27017
bind_ip = 127.0.0.1
dbpath = /data/mongodb
logpath = /var/log/mongodb/mongodb.log
logappend = true
nohttpinterface = true
```

Authentifizierung 1/3

- Authentifizierung in Basiskonfiguration inaktiv
- Benutzerauthentifizierung pro Datenbank
- Rechtevergabe anhand eines Rollensystems

Authentifizierung 2/3

Anlegen Admin-User zur Aktivierung des Auth-Systems:

```
$ mongo admin
connecting to: admin
> db.addUser("admin", "password");
{
    "user" : "admin",
    "readOnly" : false,
    "pwd" : "9e7085a6bd3bd9d413444d3adad882bd",
    "_id" : ObjectId("526c39bbaa18b118ec0fc039")
}
```

Authentifizierung 3/3

Anlage eines Applikationsbenutzers:

```
$ mongo admin -u admin -p
connecting to: admin
> use customers
> db.addUser({user:"demo",pwd:"password",roles:["read"]})
{
"user" : "demo",
"pwd" : "cd9297683105acbc2b7d9a4a8e2c043b",
"roles" : [
"read"
],
"_id" : ObjectId("544df94cb6809cdd0bba4107")
}
```

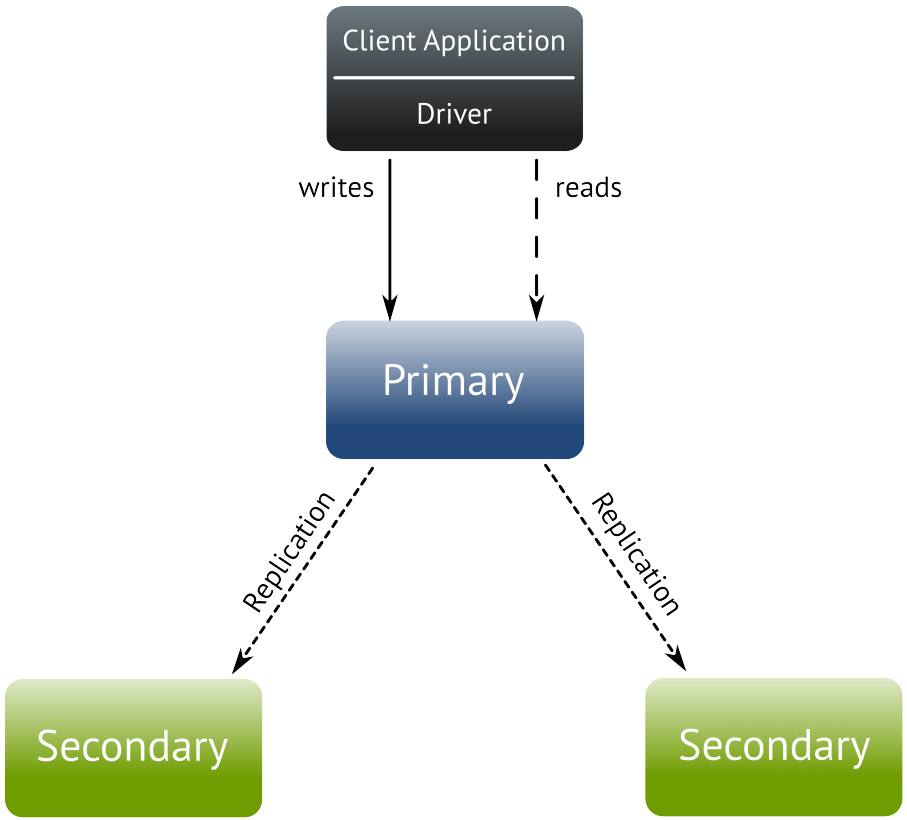
Liveaufbau Standalone

Aufbau – MongoDB ReplicaSet

MongoDB ReplicaSet 1/4

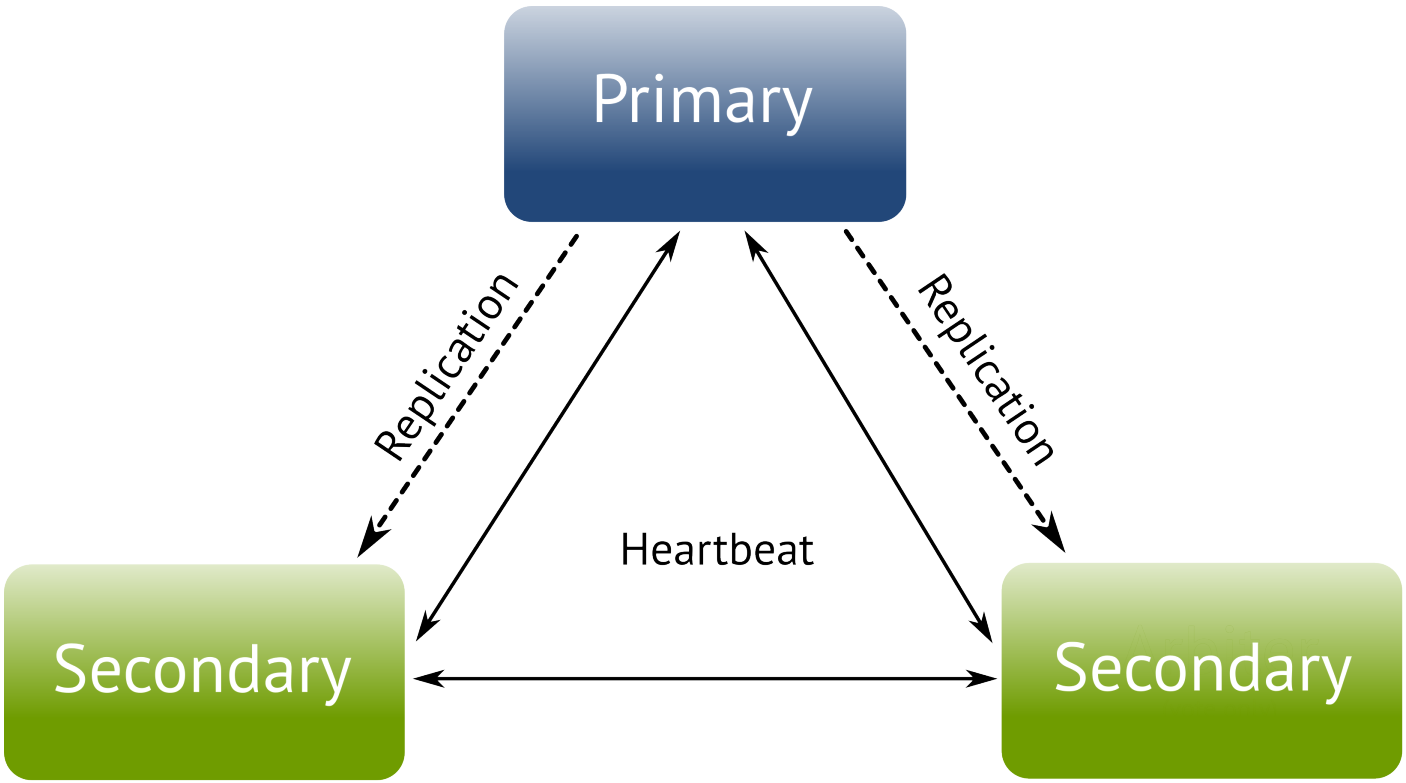
- bietet Redundanz
- kann Kapazität bei Reads erhöhen
- erleichtert Backupstrategien und Restores
- besteht aus 3-12 `mongod`
- existentes Standalone-System kann migriert werden

MongoDB ReplicaSet 2/4



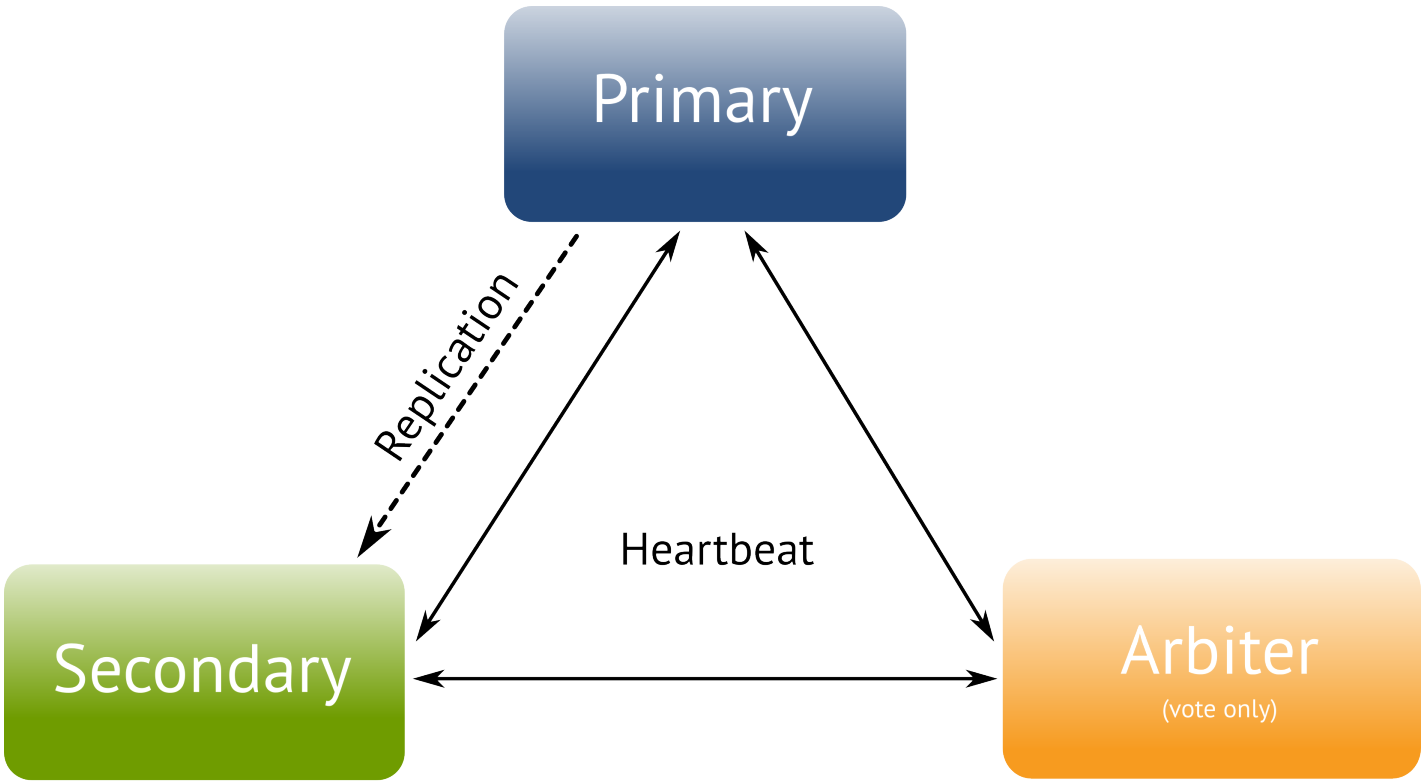
Quelle: http://docs.mongodb.org/manual/core/replication-introduction/

MongoDB ReplicaSet 3/4



Quelle: http://docs.mongodb.org/manual/core/replication-introduction/

MongoDB ReplicaSet 4/4



Quelle: http://docs.mongodb.org/manual/core/replication-introduction/

Konfiguration ReplicaSet

Basiskonfiguration:

```
# cat /etc/mongodb.conf
fork = true
port = 27017
bind_ip = 127.0.0.1
dbpath = /data/mongodb
logpath = /var/log/mongodb/mongodb.log
logappend = true
nohttpinterface = true
auth = true

keyFile = /etc/mongodb.key
replSet = rs00
```

Konfiguration ReplicaSet

Für die interne Authentifizierung der einzelnen Clusterkomponenten wird ein Authkey erstellt:

```
# openssl rand -base64 741 > /etc/mongodb.key
# chmod 0600 /etc/auth.key
# chown mongodb:mongodb /etc/mongodb.key
```

Initialisierung ReplicaSet

Basiskonfiguration:

```
$ mongo admin -u admin -p
> newSet = { _id: 'rs00', members: [
... {_id: 0, host: 'mongodb-rs00-01:27117'},
... {_id: 1, host: 'mongodb-rs00-02:27127'},
... {_id: 2, host: 'mongodb-rs00-03:27137'}]}
> rs.initiate(newSet)
  [...]
> rs.status()
  [...]
```

Liveaufbau ReplicaSet

Aufbau – MongoDB ShardedCluster

MongoDB ShardedCluster 1/5

- bietet vertikale Skalierbarkeit
- kann Kapazität von Writes erhöhen
- erweitert verfügbaren Speicherplatz

MongoDB ShardedCluster 2/5

Minimalsetup für einen redundanten MongoDB Cluster:

- 2 ReplicaSets mit jeweils 3 `mongod`-Instanzen
- 3 ConfigServer
- 2 MongoS Instanzen

MongoDB ShardedCluster 3/5

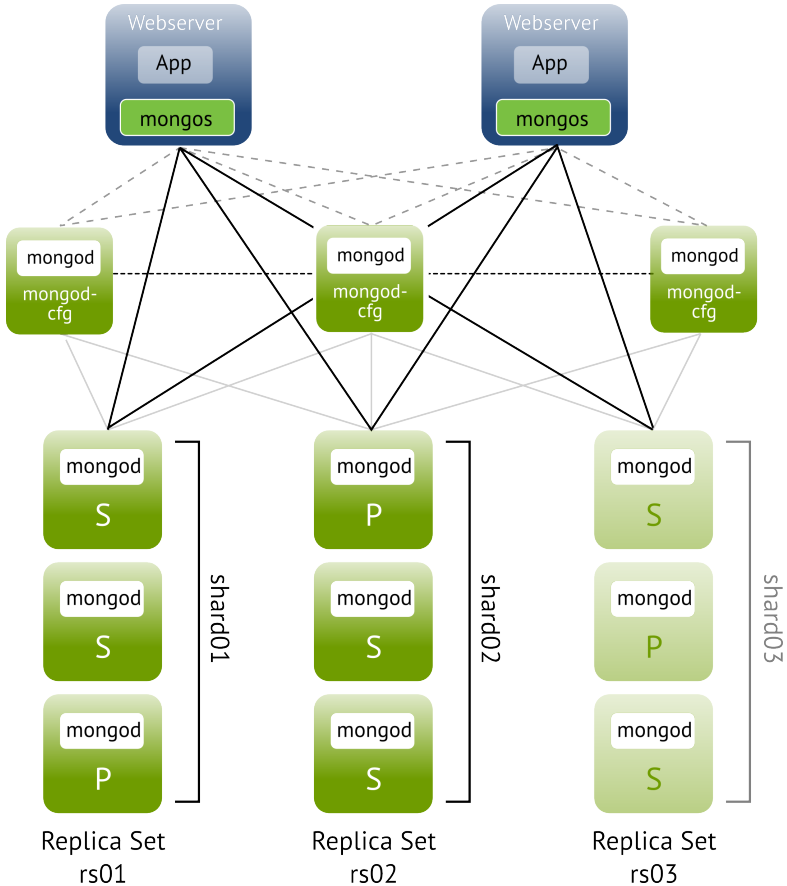


Abbildung : Sharded Cluster

MongoDB ShardedCluster 4/5

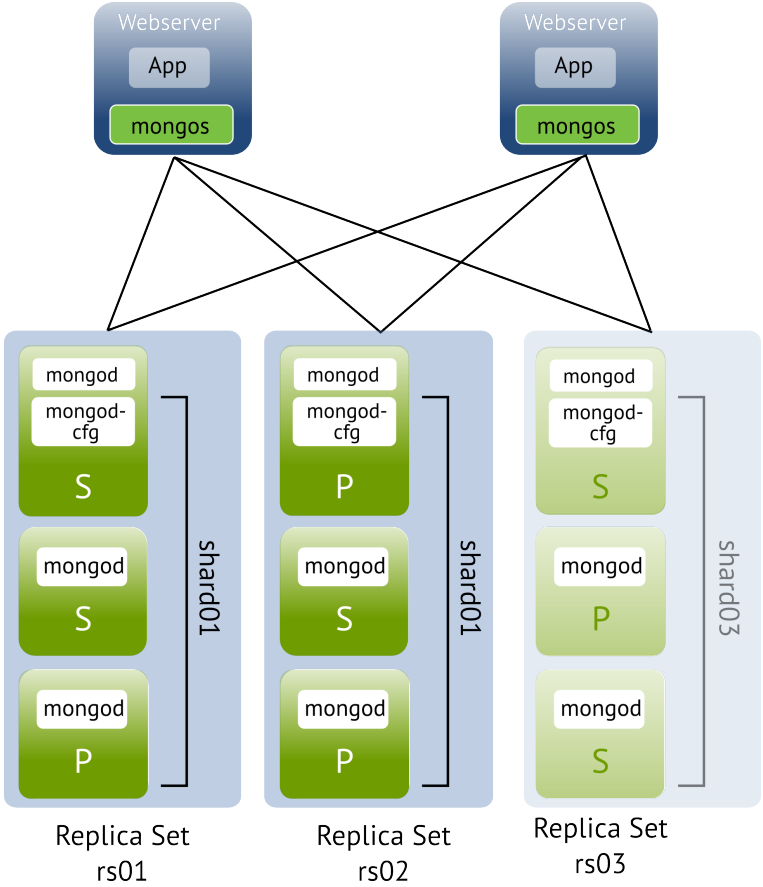


Abbildung : Sharded Cluster - Beispiel 2

MongoDB ShardedCluster 5/5

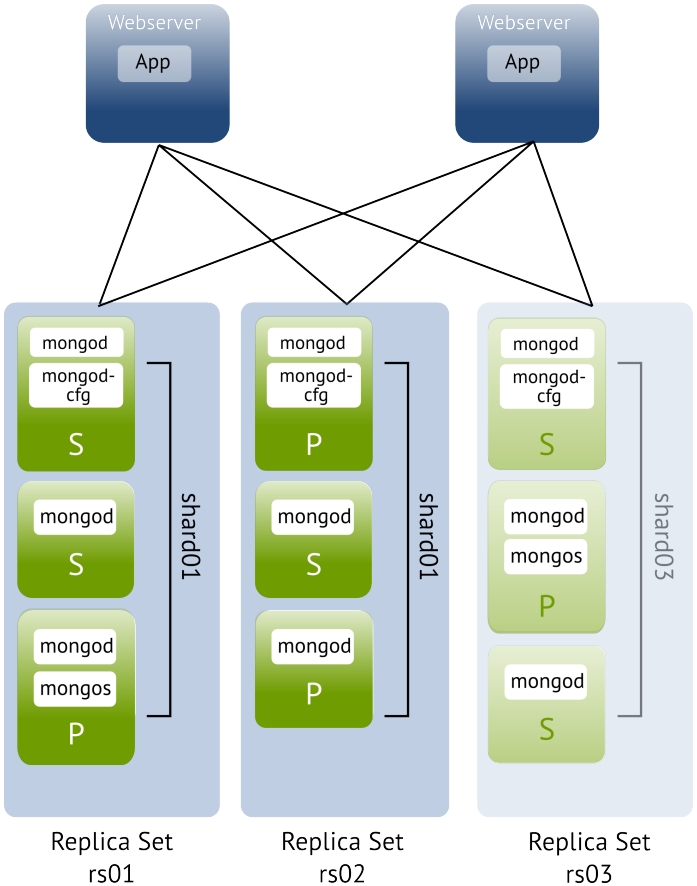


Abbildung : Sharded Cluster - Beispiel 3

Konfiguration ConfigServer

Basiskonfiguration:

```
# cat /etc/mongodb.conf
fork = true
port = 27019
bind_ip = 127.0.0.1
dbpath = /data/mongodb
auth = true
keyFile = /etc/mongodb.key
logpath = /var/log/mongodb/mongodb.log
logappend = true
nohttpinterface = true

configsvr = true
```

Konfiguration MongoS

Basiskonfiguration:

```
# cat /etc/mongodb.conf
fork = true
bind_ip = 127.0.0.1
port = 27017
keyFile = /etc/mongodb.key
logpath = /var/log/mongodb/mongodb.log
logappend = true
nohttpinterface = true

configdb = mongodb-cfg-01:27019,mongodb-cfg-02:27029,mongodb-cfg-03:27039
```

Initialisierung Cluster 1/2

Anlegen Adminuser Cluster:

```
$ mongo admin
connecting to: admin
mongos> db.addUser("admin", "password");
{
    "user" : "admin",
    "readOnly" : false,
    "pwd" : "9e7085a6bd3bd9d413444d3adad882bd",
    "_id" : ObjectId("526c39bbaa18b118ec0fc039")
}
```

Initialisierung Cluster 2/2

Anlegen Adminuser Cluster:

```
$ mongo admin -u admin -p
connecting to: admin
mongos> use config
switched to db config
mongos> sh.addShard("rs00/mongodb-rs00-01:27117")
{ "shardAdded" : "rs00", "ok" : 1 }
mongos> sh.status()
   [...]
```

Aktivierung Sharding

Anlegen Adminuser Cluster:

```
mongos> sh.enableSharding("demodata")
{ "ok" : 1 }
mongos> sh.shardCollection("demodata.products", {_id:1})
{ "collectionsharded" : "demodata.products", "ok" : 1 }
mongos> sh.status()
   [...]
```

Liveaufbau ReplicaSet

Optionale Komponenten

Optionale Komponenten

- Arbiter
- Hidden Member
- Delayed Member

Vielen Dank für Ihre Aufmerksamkeit!

Bei weiteren Fragen wenden Sie sich bitte an info@b1-systems.de
oder +49 (0)8457 - 931096