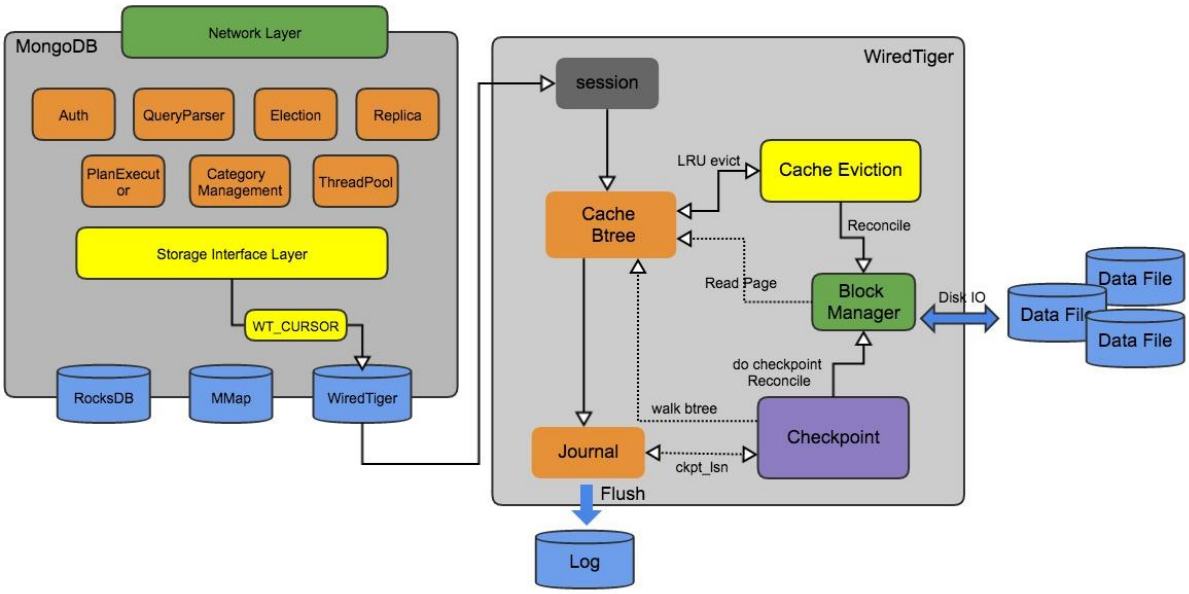
2. 설치 및 구성, 하드웨어와 프로비저닝, 복제 Replication set 구성

● WiredTiger

. WiredTiger는 NoSQL 데이터베이스 용 오픈 소스 key/value 저장소 엔진이며 MongoDB의 기본 저장소 엔진이다. 데이터 저장을 위해 B-트리 및 로그 구조 병합트리를 모두 지원한다.또한 행 저장소와 열 저장소를 모두 지원합니다. WiredTiger는 다중 코어 아키텍처에서 더 나은 확장성을 위해 다중 버전 동시성 제어를 지원합니다. WiredTiger는 런타임시 사용자가 스키마를 제공하는 기본 스키마 지원도 있습니다. WiredTiger는 C로 개발되었으며 Java / Python 언어 바인딩을 제공합니다.



- `session module`

  , wt 엔진의 상위 계층과의 상호 작용을 담당하는 핸들, 각 세션은 여러 커서와 연결되며 커서는 세션에 속합니다.

- `cache module`

  , 주로 메모리의 btree 페이지 (데이터 페이지, 인덱스 페이지, 오버플로 페이지)로 구성

- `evict module`

  , 캐시 메모리가 부족한 경우 캐시 제거를 트리거하고 btree를 촉진하며 LRU에 따라 제거를 정렬합니다.

- `Journal module`

  , InnoDB의 redolog와 유사한 WAL 로그로 데이터 지속성을 보장하고 타이밍 및 정량적 임계 값을 플러시합니다.

- `checkpoint module`

  , InnoDB 체크 포인트 메커니즘과 유사하게 btree 플래싱을 비동기 적으로 실행하고 체크 포인트 후 log_ckpt_lsn을 업데이트하도록 로그 모듈에 알립니다 (lsn 개념은 InnoDB와 일치 함).

- `block manager module`

  , 디스크 IO의 읽기 및 쓰기를 담당하며 캐시, 제거, 체크 포인트 모듈은 모두 이 모듈을 통해 디스크에 액세스합니다.

. 체크포인트

. WiredTiger의 체크 포인트는 기존 관계형 데이터베이스 시스템의 일반적인 체크 포인트상이

. 기본적으로 WiredTiger 테이블은 여러 개의 준비 전용 체크 포인트와 쓰기 가능한 라이브 트리로 구성

. 체크 포인트는 주기적으로 수행되거나 수동으로 트리거

. 체크 포인트 프로세스 중에 테이블의 더티 페이지가 디스크로 플러시되고 새 체크 포인트가 생성

. 체크 포인트는 타임 스탬프별로 정렬

. 체크 포인트는이를 참조하는 실행중인 트랜잭션이없는 경우에만 삭제

. Copy-on-write 기술은 새 체크 포인트를 만드는 오버 헤드를 줄이는 데 사용

. WiredTiger의 성공적인 체크 포인트에는 4 단계가 포함.

1. 모든 더티 리프 페이지가 디스크로 플러시
2. 모든 더티 내부 페이지가 디스크에 기록
3. 체크 포인트 파일에서 fsync가 호출되어 체크 포인트가 디스크에 Write
4. 마지막으로 B- 트리의 루트는 원자 적으로 새 루트로 변경.

. 이전 루트는 체크 포인트가되고 새 루트는 새로운 "라이브 트리"가 생성

. 체크 포인트가 삭제되면 WiredTiger는 인접한 두 체크 포인트 사이에 종속성이 있는지 확인하고 두 B- 트리에 공통 페이지가 있으면 병합

. 동시성 제어

. WiredTiger는 Reader가 Writer를 차단하지 않는 MVCC (Multi-Version Concurrency Control)를 사용합니다. 쓰기-쓰기 충돌의 경우 하나의 트랜잭션이 롤백되고 재 시도됩니다. 또한 WiredTiger의 메인 메모리에 맞도록 트랜잭션의 작업 세트가 필요합니다. WiredTiger에서 트랜잭션은 "명명 된 스냅 샷"이라고하는 특정 버전의 데이터베이스를 고정 할 수 있습니다. 이러한 트랜잭션 내의 쿼리는 스냅 샷에서 작동하는 것처럼 실행됩니다.

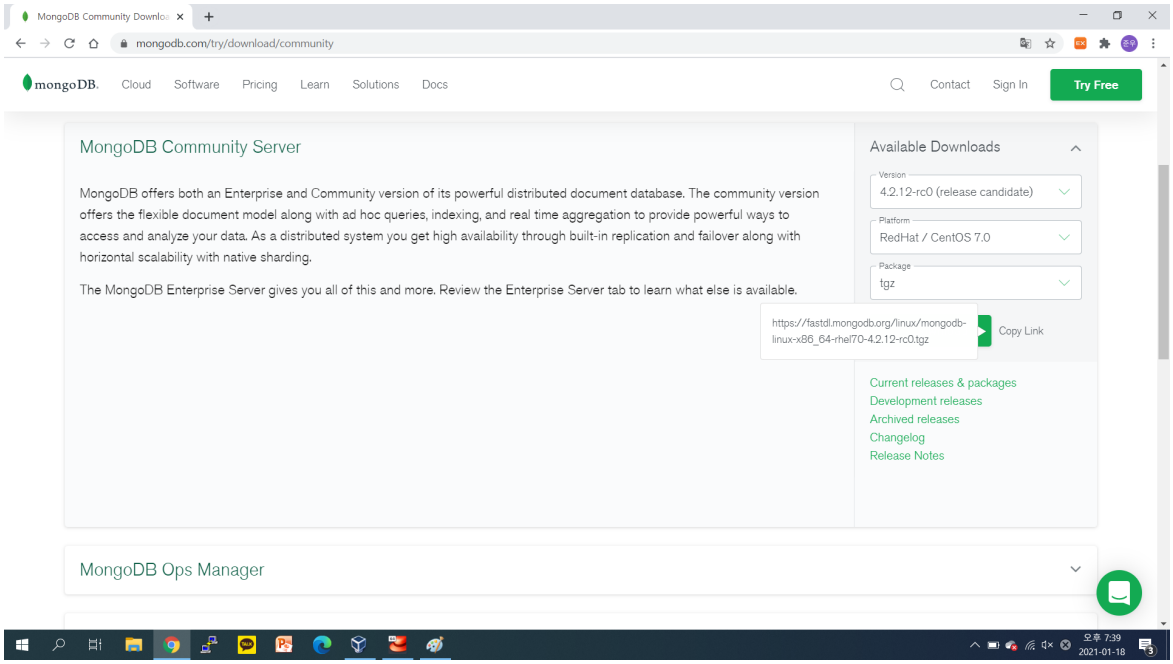
. 압축

. 기본적으로 컬렉션 과 인덱스 모두 압축을 지원한다.블록 단위 압축이며 알고리즘 타입에는 snappy(기본알고리즘, good Compression, Low overhead), Zlib(Better Compression, CPU high overhead)가 있다.

. 참조 : https://www.programmersought.com/article/21954954398/

● 설치 메뉴얼

1. binary 설치
   . 공식 사이트 download 카테고리 접속 후 release version, OS version 후 copy link 클릭 해서 경로 추출 또는 tgz 다운로드



. 해당 link 참조해서 wget하여 다운로드 or tgz파일 설치 경로에 업로드

```
[root@mongodb yum.repos.d]# wget https://fastdl.mongodb.org/linux/mongodb-linux-x86_64-rhel70-4.2.12
-rc0.tgz
--2021-01-10 14:20:21--  https://fastdl.mongodb.org/linux/mongodb-linux-x86_64-rhel70-4.2.12-rc0.tgz
Resolving fastdl.mongodb.org (fastdl.mongodb.org)... 54.230.221.38, 54.230.221.114, 54.230.221.51, .
..
Connecting to fastdl.mongodb.org (fastdl.mongodb.org)|54.230.221.38|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 133206601 (127M) [application/gzip]
Saving to: 'mongodb-linux-x86_64-rhel70-4.2.12-rc0.tgz'

100%[======================================================>] 133,206,601 5.63MB/s   in 22s

2021-01-10 14:20:43 (5.89 MB/s) - 'mongodb-linux-x86_64-rhel70-4.2.12-rc0.tgz' saved [133206601/1332
06601]

[root@mongodb yum.repos.d]# _
```

. tar -xvzf <파일명>

```
vi /etc/mongodb.conf(테스트 해본결과 해당 경로 아니어도 가능)

logpath=/MongoDB/log/mongod.log ← 로그 파일 경로(파일명 까지 입력해야 db start시 에러안남)
logappend=true ← 로그 파일이 존재할 경우 이어서 기록
fork=true ← mongoDB를 데몬으로 실행
dbpath=/MongoDB//data ← data 디렉터리 경로
pidfilepath=/MongoDB/log/mongod.pid ← pid를 파일로 지정 및 경로 설정
bind_ip=0.0.0.0 ← 접속 가능한 IP 설정 모든 접속 허가 시 0.0.0.0(replication set 구성시 ACL 문제 없도록 전체 허가 설정)
port=10000 ← 데몬 통신 포트
verbose=true ← 로그 파일의 내용을 상세하게 기록
```

※ root 설치/기동이 아니라 별도의 mongodb 계정으로 구성하였기 때문에 관련 디렉토리 및 폴더 owner, group 설정

. .bash_profile에 bin 경로 path 설정하여 운영 시 편의성 제공(기동, 접속 명령어 접속 시 절대경로 지정하지 않아도 됨)

. mongod -config /etc/mongodb.conf 로 기동

. 접속 시 mongo localhost:<port number>

2. yum 설치

. repsitory 생성

```
vi /etc/yum.repos.d/mongodb-org.repo

[MongoDB]
name=MongoDB Repository
baseurl=http://repo.mongodb.org/yum/redhat/$releasever/mongodb-org/4.2/$basearch/
gpgcheck=1
enabled=1
gpgkey=https://www.mongodb.org/static/pgp/server-4.2.asc
```

. yum install mongodb-org(root 실행)

. 설치 시 default 경로로 설치

```
vi /etc/mongod.conf
(테스트 결과 build설치 처럼 필요한 옵션만 체크해서 conf파일 생성 후 해당 conf 파일로 db open 가능함)
# mongod.conf
# for documentation of all options, see:
#   http://docs.mongodb.org/manual/reference/configuration-options/
# where to write logging data.
systemLog:
  destination: file
  logAppend: true
  path: /MongoDB/log/mongod.log
# Where and how to store data.
storage:
  dbPath: /MongoDB/data/
  journal:
    enabled: true
    commitIntervalMs: 200
#  engine:
  wiredTiger:
    engineConfig:
      cacheSizeGB: 2
      journalCompressor: snappy
      directoryForIndexes: false
    collectionConfig:
      blockCompressor: snappy
    indexConfig:
      prefixCompression: true
# how the process runs
processManagement:
  fork: true  # fork and run in background
  pidFilePath: /var/run/mongodb/mongod.pid  # location of pidfile
  timeZoneInfo: /usr/share/zoneinfo
# network interfaces
net:
  port: 10001
  bindIp: 0.0.0.0  # Enter 0.0.0.0,:: to bind to all IPv4 and IPv6 addresses or, alternatively, use the net.bindIpAll setting.
setParameter:
  enableLocalhostAuthBypass: false
#security:
#operationProfiling:
#replication:
#sharding:
## Enterprise-Only Options
#auditLog:
#snmp:
```

. 대부분 /usr/bin 밑에 명령어 존재(.bash_profile에 path지정하면 좀 더 편하게 활용 가능)

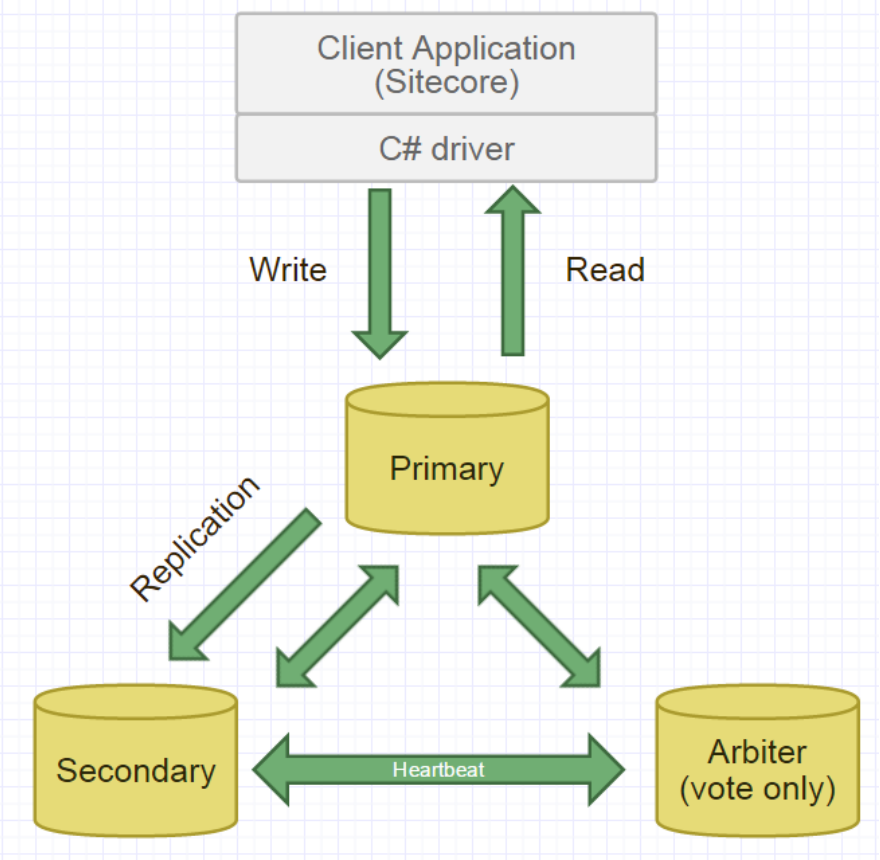
. root 로 기동/중지 시

```
#systemctl start mongod
#systemctl stop mongod

##linux 7버전 부터 systemctl 명령어 사용가능 6버전은 service
```

※ 보안 상 root로 프로세스 띄우는 것이 취약점이기 때문에 관리 OS 계정 생성하여 해당 계정으로 프로 세스 띄울 수 있도록 하자

● Replication set



. 서비스 무중단을 위한 MongoDB의 안정성을 보장하는 솔루션 중의 하나

. 경우에 따라서는 많은 수의 슬레이브 노드를 설정할 수 있지만 예상할 수 없는 장애가 발생 할 수 있으므로 최소 3대의 슬레이브 노드 설정을 적정 개수로 권장

. Hearbeat : 매 2초 마다 Secondary 상태를 체크합니다.

. Secondary 가 Down 되더라도 복제만 중지될 뿐 Primary에 대한 작업은 정상적입니다.

. Primary가 다운되면 Secondary가 Primary가 됩니다.

. OpLog는 복제가 실패하는 경우를 위해 로그정보를 저장해 줍니다 (기본 크리 1GB)

. primary - secondary 서버간 replication set 작동 절차

1. Application을 통해 Primary Server로 Write 작업이 발생합니다.
2. Memory Mapped Cache Area에 데이터를 먼저 저장합니다.
3. Primary Server가 정상 작동되지 않는 경우 Get Last Error가 발생합니다.
4. Primary Server의 Journal Area로그데이터를 Journal 파일에 백업합니다.
5. Primary Server는 Application에게 장애가 발생했음을 알립니다.
6. Primary Server는 Secondary Server로 복제 작업을 수행합니다.
7. Primary Sever는 Application에게 장애가 발생했음을 알립니다.

. 주요 특징

1. Primary 서버는 매2초 마다 Secondary 서버의 상태를 체크하며 동기화를 위한 HeartBeat 작업을 수행
2. Secondary 서버가 중지 되더라도 복제 작업만 중지 될 뿐 Primary Server에 대한 데이터 읽기 작업은 정상적으로 수행됨
3. Primary서버가 중지되면 Secondary 서버가 Primary 서버가 됨

4. OpLog는 primary 서버가 secondary 서버로 복제 작업을 수행하다 장애로 인해 작업을 수행할 수 없는 경우 동기화하지 못한 데이터를 추후 세컨더리 서버로 반영해 주기 위해 데이터를 백업

. Priority (우선 순위)

. 다중 구성일 경우 Primary 서버가 장애가 발생하면 MongoDB는 10초 이내에 다음 Primary 서버가 되어야 할 노드를 선택

. 아비터 서버가 활성화 되어 있다면 아비터가 적절한 서버를 선택해 주지만 사용자가 우선순위를 설정해 둔 경우에는 가장 높은 값을 부여 받은 서버가 Primary서버가 됨(0~1000 설정 가능)

. 멤버의 유형

. 데이터는 저장하지만 절대 Primary 서버는 되지 못하는 Only Secondary 서버

. Client Application을 위한 숨겨진 멤버인 Hidden Member(아비터 서버가 Primary 서버로 선출할 때 포함되지 않음)

. 데이터가 저장 되지 않으며 장애시에 Primary 선출할 때에 사용되는 Arbiter Member

. Primary 서버의 OpLog 정보를 정의된 시간동안 Secondary 서버에 적용하지 않고 Delay 한 후 적용 되는 Delayed Member

. 데이터는 저장하지만 투표시에 Primary로 선택되지 않는 Non Voting Memeber

. replication set 멤버의 동기화

. replication set 설정하게 되면 인스턴스를 활성화 했을때 자동으로 동기화 작업을 수행

. 사용자가 직접 안전한 멤버의 데이터 파일과 저널 파일을 복사하여 동기화하는 방법

. 구성

```
#master
#mongod --config /MongoDB/mongodb.conf --replSet repmongo/localhost:10000

#mongo localhost:10000/admin
>rs.status() ## replication 상태 확인

>rs.initate() ## replication 초기화
>exit -- primary 설정

>rs.add("10.0.2.16:10001")
>rs.addArb("10.0.2.17:10002")
```