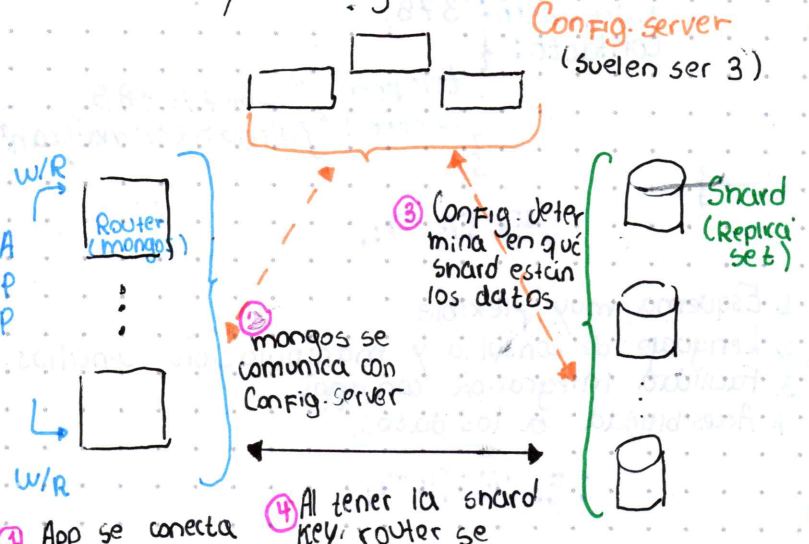
MongoDB

- Base de datos NoSQL orientada a documentos
- Eficiente en almacenamiento y recuperación de datos.
- Utilizado actualmente para:
  - Internet de las cosas
  - Visualización geoespacial
  - Gestión de contenidos
  - Aplicaciones móviles
  - Videojuegos
  - Log de eventos

ARQUITECTURA

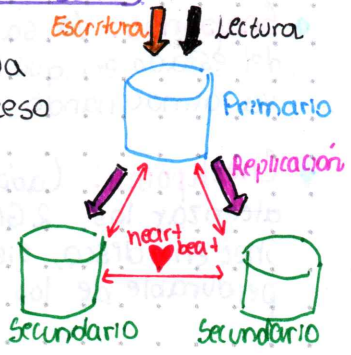
- Arquitectura master-slave
- mongod: Demonio (servicio, proceso primario) que funciona en un nodo y administra el acceso a los datos.
- mongos: Servidores de enrutamiento entre la aplicación y la base de datos.
- Config-server: Servidor que almacena los metadatos para localizar los datos de las operaciones. Es un indexador.
- Replica set (shard): Grupo de procesos mongod que almacenan las mismas copias de los datos
  - → Primary: Copias principales
  - → Secondary: " secundarias de Primary
  - → Arbitrer: Vota para decidir en caso de que Primary se caiga.



- Si se caen todos los Config-Server igual existe la posibilidad de que los enrutadores se conecten a algún shard utilizando caché.
- 3 Config-Server: Garantizan acceso aunque 1 ó 2 de ellos se caigan.
  - → Reparten carga de trabajo
  - → Se escribe en ellos si los metadatos cambian
- 2 o más mongos: Reparten carga de peticiones de apps.
- 2 o más shards: Para repartir los documentos en varios nodos del clúster.
- Chunks: "Trozos" en los que se reparten los documentos de una colección en varias shards.
- Shard key: Clave que determina cómo se divide una colección en varios shards.
- Range Based Sharding:
  - → Chunks se crean en base a rango de valores de la shard key.
    - ✓ Bueno para consultas → Documentos con un valor de shard key similar estarán probablemente en un mismo shard.
    - x Puede haber shards sobrecargados
- Hash Based Sharding:
  - → Cálculo de valor hash en base a la shard key.
    - ✓ Buen equilibrio y carga de los datos
    - x Consultas por rango sobre la shard key requerirán acceder a varios shards.

Replicación de los datos

- Cada réplica es controlada por una instancia del proceso mongod.
- Primario almacena en una estructura de log los cambios en los datos, luego son propagados a secund.



- Operaciones de escritura se realizan exclusivamente sobre el primario.
  → Las de lectura (por defecto también (se puede configurar tanto escritura y lectura)
- Heartbeat (♥): Comunicación entre todos del mismo replica set indicando que siguen vivos. Se envía cada 2 segundos. Si pasa más de 10 segundos donde no hubo respuesta, se da por caído.

- ¿Qué ocurre si el primario cae?
  Secundarios deben elegir un nuevo primario entre ellos, mediante votación.
  Solo el que tiene datos + recientes se puede volver.
- ¿Y si hay empate?
  Arbitrer: Instancia mongod que se encarga de desempatar las votaciones. No almacena réplicas.

- Casos especiales de secundarios
  Priority 0: No pueden convertirse en primarios.
  Hidden replica set: Invisibles a la app cliente (no pueden ser leídos por la app)
  Delayed replica set: Guardan las copias del principal con retraso. (Útil para backups y snapshots.

- Configuración de lectura
  → Por defecto, es siempre sobre el primario.
  Otras posibilidades:
  → Primary Preferred:
  Si el primario no está disponible, no se espera a que lo esté, se va a un secundario.
  → Secondary
  Siempre sobre secundarios, puede servir para disponibilidad pero falla en consistencia.
  → Secondary Preferred
  → Nearest
  Se escoge el con menor latencia.

Almacenamiento de los datos

- Snapsnot: Va sacando fotos instantaneas del estado en qué están los datos y los va almacenando (no se sobreescribe)

- Checkpoint: Cada 60 segundos o al alcanzar los 2GB, se escribe el Snap Shot en disco, siendo la nueva copia perdurable de los datos.

- Journal: MongoDB escribe todas las transacciones del Snapshot en un fichero de log para recuperar entre 2 checkpoints.

Configuración de escritura

- Niveles de Write Concern para confirmación de escritura.
  → Sin confirmación
  → Unicamente en primario.
  → En primario y uno o varios secundarios
  → Confirmación de escrito o no en Journal

MODELO DE DATOS

- Orientado a documentos en BSON

Colección → Tabla
Documento → Fila
Campos → Caumna

- Cada documento dentro de una colección tiene un identificador único SIEMPRE.

- Documento: Conjunto de pares campo-valor. Pueden tener tantos campos como se desee. Puede contener otros documentos. Un campo puede estar o no estar.

Ejemplo:

```
{
  _id: <ObjectId 1>
  nombre: "Julio",
  apellido1: "Gonzalez",
  apellido2: "Sañudo",
  numSocio: 378,
  Contacto: {
              telefono: "166777888",
              correo: "julioGS@kmail.com"
            }
}
```

VENTAJAS

1. Esquema muy flexible
2. Lenguaje de consulta y manipulación sencillos
3. Facilidad integración con apps.
4. Accesibilidad a los datos.

DESVENTAJAS

1. Aplicaciones clientes mas complejas
2. No garantiza ACID