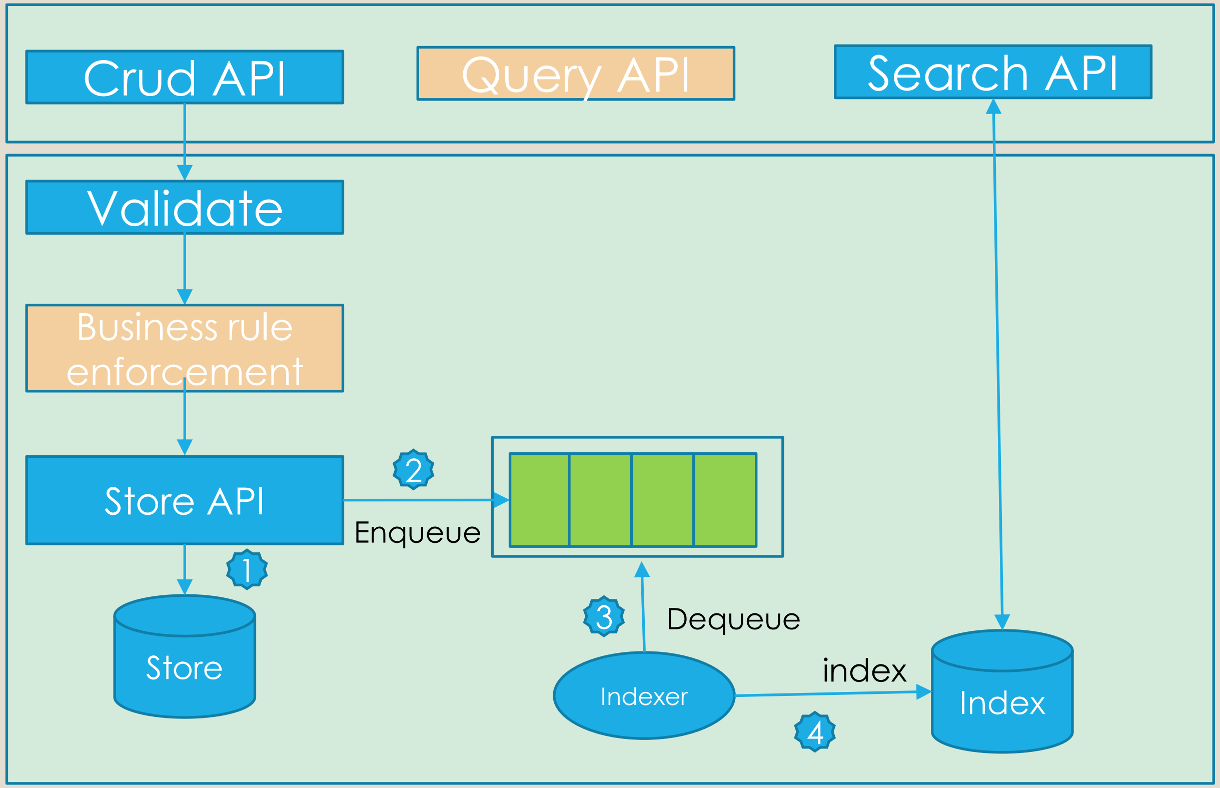
ADVANCED TOPIC IN BIG DATA

Quick Review

- By now, you should be familiar with strongly typed data protocols
- You should have reviewed gData, oData, Protocol Buffers
- You should have fair understanding of the overall architecture
- You should have some code working on your laptop

Architecture



Prototype Requirements:

Rest API that can handle <u>any structured data in Json</u>

- URIs, status codes, headers, data model, version

- Rest API with support for crd operations
  - Post, Get, Delete
- Rest API with support for validation
  - Json Schema describing the data model for the use case
  - Controller validates incoming payloads against json schema
- The semantics with ReST API operations such as update if not changed/read if changed
  - Update not required
  - Conditional read is required
- Storage of data in key/value store
- Must implement use case provided

Rest API Specifications

- Data Models
  - Payload structure and serialization
- URI conventions
  - /{type}/{id}
  - /plan/12xvxc345ssdsds
- Status Code
  - 200,201
  - 302,304
  - 401, 404, 403, 412, 429
  - 500
- Headers
  - Students should review the HTTP standard headers
  - Various uses of Etag, If-Match, If-None-Match, Authorization in Rest APIs
- Version
  - Accept
  - URL
- Security

- Example: https://www.hl7.org/fhir/http.html

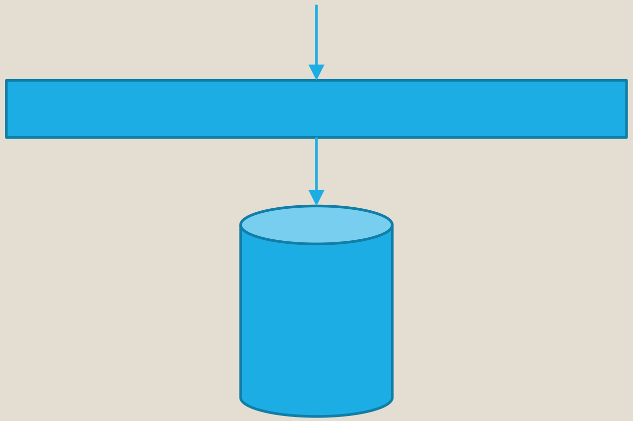
Tooling

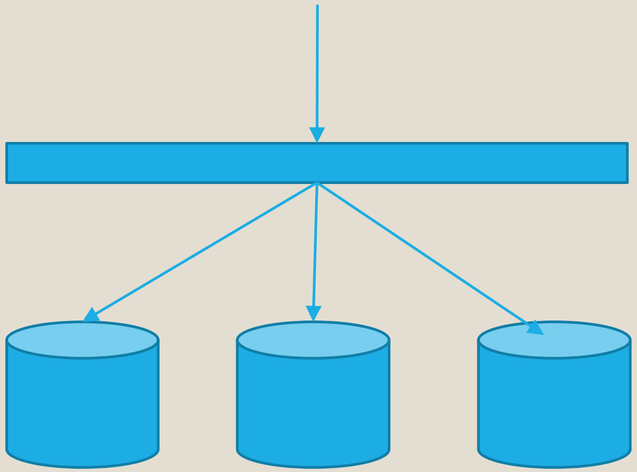
- Json simple for Json parsing
- Spring Boot for rest API development
- Elastic Search for search and retrieval capabilities
- Redis for Cache solutions
- Json Schema for schema validation
- Zuul for API Gateway pattern

But how do I distribute the data?

- single point of failure
- Limited space/storage
- Strongly consistent

- Highly available distributed system
- Seemingly unlimited storage
- What about consistency?





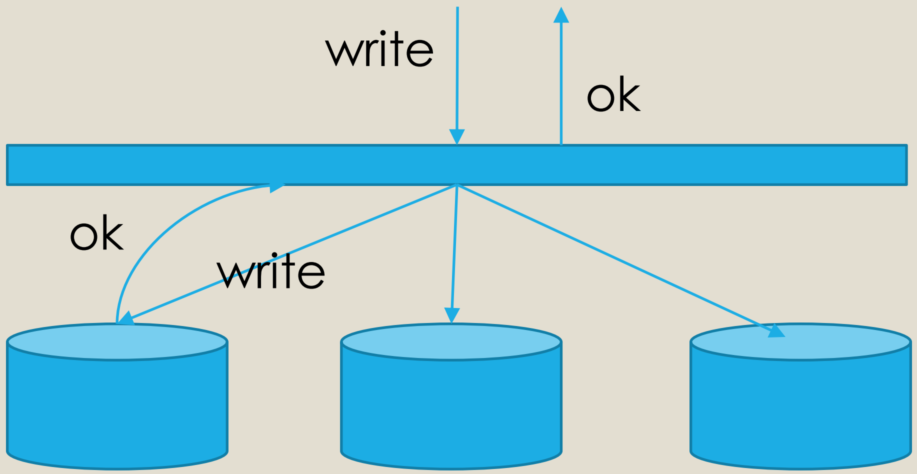
Key/value stores

- Key readings:
  - Dynamo: Amazon’s Highly Available Key-value Store :
    - http://www.allthingsdistributed.com/files/amazon-dynamo-sosp2007.pdf
  - Bigtable: A Distributed Storage System for Structured Data: http://static.googleusercontent.com/media/research.google.com/en//archive/bigtable-osdi06.pdf
  - CAP Theorem
    - *Consistency*
      - Eventual consistency, Read your own write, Strongly consistent
    - *Availability*
    - *Partition tolerance*
    - In the presence of network failure, you have to choose between consistency and high-availability

Problems

- In the presence of many servers, how do I determine the server that stores the object?
  - Consistent hashing to the rescue
- But what if one of the servers fails or the network connection to the server fails?
  - Replication techniques:
    - Primary/backup
    - Active replication
- If I have multiple servers and if an object is stored on more than one server
  - How do I keep the objects consistent?
    - Eventual consistency, strong consistency, weak consistency

Weak consistency



Quorum consistency

R=read replica count

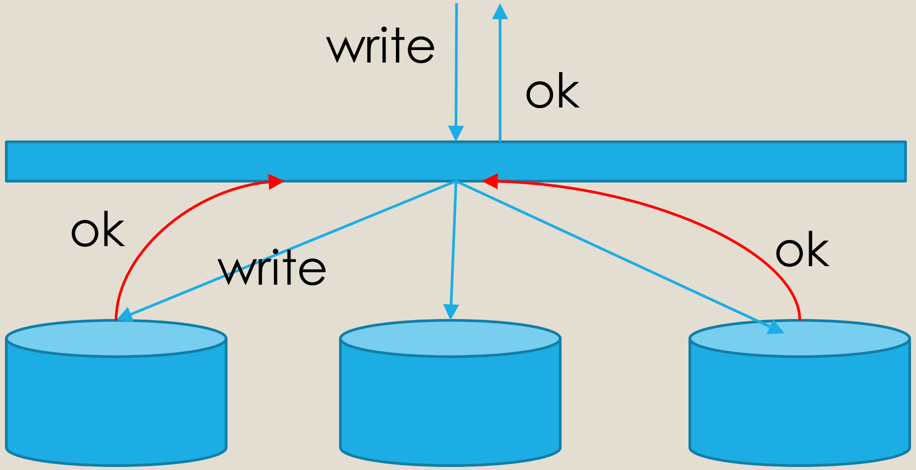
W=write replica count

N=replication factor

Q=**QUORUM** (Q = N / 2 + 1)

- If W + R > N, you will have consistency

- On read, two of the replica must respond
- On write, two of the replica must make the data durable before acknowledging the right



Data Modelling

- K1➔v1
- K2→v2
- K1?, K2?

Consistent hashing

- http://theory.stanford.edu/~tim/s17/l/l1.pdf

- How do you map a large number of objects into few servers?
  - h(x) mod n
- What if the number of servers changes, what would that do to the objects that have already been assigned?
- How do ensure a universal distribution of objects across servers?
- The key idea is:
  - hashing the names of all objects
  - hash the names of all the cache servers s
  - The object and cache names need to be hashed to the same range, such as 32-bit values.

Key design issues

- Partitioning algorithm
  - Uniform load distribution
- Schema less
- Replication strategy
- Recovering from partial failure
  - Joining a group
    - Load partitioning amongst replicas
- Load rebalancing
- Range query support
- Data versioning
- Support for structured data or simply Blobs
- Marshaling/Unmarshaling
  - How do you store int and floats in redis?

Mapping of meta-model into key/value store

- JSON payloads can be modeled as a graph.
- https://www.researchgate.net/publication/315679274_Query_Service_for_REST_APIs
- https://www.researchgate.net/publication/315679444_Business_Rules_for_REST_APIs

  How do we map a JSONObject into the key value store?
- What is the key signature?
- Do we store the data as a blob?

- Do we store the data as structured?

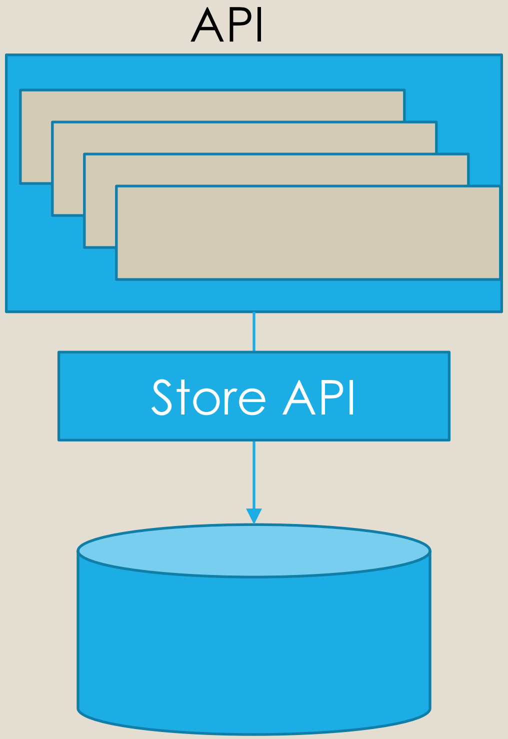
Trade-offs between storing the data as a blob versus structured storage

- Storing data as a blob is fast, atomic, reliable
  - But, how do you update the data?

- Storing the data as structured data requires more work on initial creation, but update are much quicker

A typical design pattern

A compound document
with nested objects



Should the compound document be decomposed into its constituent objects for storage, and/or indexing, etc…?