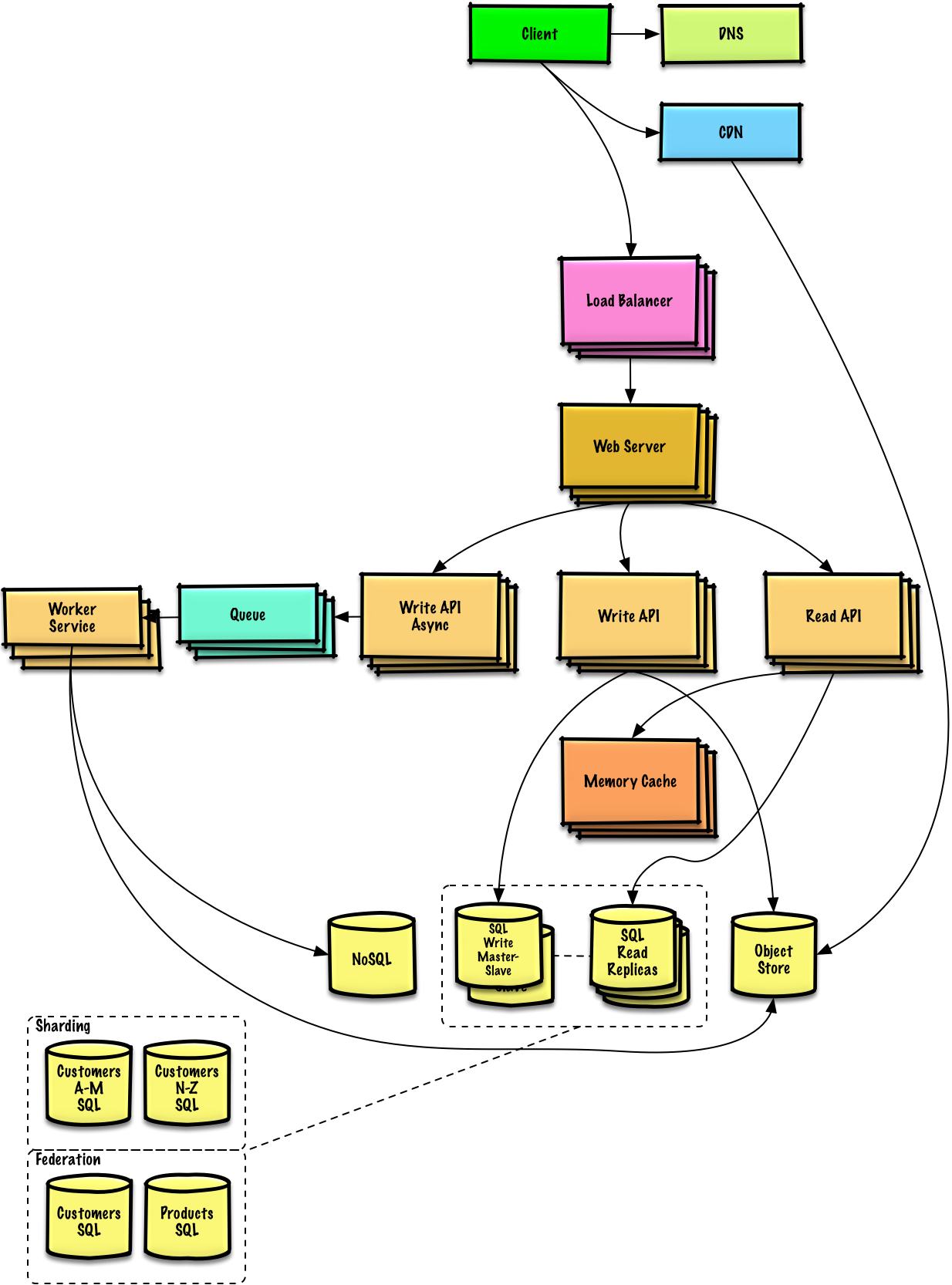
**Approaching a System Design Question:**

1. Requirements:
   1. Functionality – What problem does it solves
   2. Non-Functionality – efficiency
   3. Additional – nice to haves
   4. Constraints – Not to have / Can’t haves
2. Estimations:
   1. Traffic Estimates
   2. Storage Estimates
   3. Bandwidth Estimates
   4. Memory Estimates
   5. Read / Write Ratio – Read Heavy vs Write Heavy
   6. Estimate Summary
3. API’s:
   1. Rest Vs Soap
      1. Api dev key – user accounts, etc to limit abuse
4. Database Design:
   1. Relational vs NoSql
   2. NoSql
      1. Hbase
      2. Cassandra
      3. Graph DB
      4. Distributed Ledger
      5. Bigtable
   3. Block data – Files
   4. Distributed File Storage – S3, HDFS etc
5. Algorithm Design:
6. Application Component Design:
   1. Individual Read write Servers / applications – Async Write / Operations
   2. Message Queuing services - RabbitMQ, GoogleRPC
   3. Micro-services – Docker, Kubernetes
7. Data Partitioning, Fault tolerance & Replication
   1. Range based partitioning
   2. Hash based partitioning
      1. Consistent Hashing
8. Search Engines
   1. Full text search engines
   2. Hadoop
   3. Pig, Hive etc
9. Caching
   1. Cache eviction policy
   2. Mem-cache, Redis
   3. Content Delivery Network – caches the images / resources directly
10. Reverse proxy & Load Balancing
    1. Ngnix
    2. SSL encryption URL
11. Client-side
    1. Applications – Desktop, Web, Mobile
    2. Push, Poll, Long poll, Web-sockets, Server sent events
12. DB Purge & Cleanup
    1. Data de-duplication
13. Analytics & Telemetry
14. Security & Permissions
    1. Single Sign on, Oath2, Active Directory
    2. DMZ zones
    3. Firewall

<https://github.com/donnemartin/system-design-primer#system-design-topics-start-here>



**System design questions links:**

<https://hackernoon.com/top-10-system-design-interview-questions-for-software-engineers-8561290f0444>

<https://github.com/shashank88/system_design>

<https://github.com/donnemartin/system-design-primer>

<https://github.com/checkcheckzz/system-design-interview>

<http://blog.gainlo.co/index.php/category/system-design-interview-questions/>

**Approaching a coding question:**

1. Don’t jump to Solve the question since you have seen it before!!
2. Clarify and understand the question.
   1. Write the API signature
   2. Come up with example inputs, output.
3. Talk about brute force algorithm
4. Talk about improving it