Clarifying requirements

* Functional requirement
* Business requirement
* Non- Functional requirement
* A lot of users
* A lot of Data
* Server failure

Technologies like to use :

* Components like
  + Load Balancer
  + Zoo Keeper
  + Database
  + Cache
  + CDN
  + Message Queue
* Measurements
  + Five nine availability
  + QPS
  + Bandwidth estimation
  + Storage Size

Possible tradeoffs to discuss

* SQL VS No SQL
* Latency / Throughput
* Availability / Reliability
* Caching / Storing
* Scaling / Simplicity
* Developing duration / Complexity
* Functionality / Efficiency
* Specificity / Generality

Name of Technologies

Load Balancer :

* Azure Application Gateway
* Kemp LoadMaster
* Citrix ADC
* F5 BIG-IP Local Traffic Manager (LTM)
* Varnish Software
* Nginx
* Azure Traffic Manager
* HA Proxy

Relational Database

* IBM DB2
* Oracle DB
* Single Store
* SAP HANA Cloud
* Microsoft SQL Server
* PostgreSQL
* MySQL
* Dynamo DB

Non- Relational Databases

* Mongo DB ( document)
* Apache Cassandra (Wide Column)
* Apache HBase
* Apache Couch DB
* Neo4j (Graph)
* RavenDB
* Redis ( Key-value Pair)
* OrientDB
* SeyllaDB

Cache:

* Redis
* Memcached
* Apache Ignite
* Couchebase Server
* Hazelcast IMDG
* McRouter
* Varnish Cache
* Squid Caching Proxy

CDN:

* SatlkPath
* Sucuri
* CloudFare
* KeyCDN
* Rackspace
* Google Cloud CDN
* CacheFly
* Amazon CloudFront

Service Discovery Software :

* Zookeeper
* HashiCorp Consul
* Eureka
* Docker
* Traefik
* Etcd
* GRPC
* Hystreix
* SkyDNS

Messaging

* Apache Kafka
* AWS Kinesis

Messaging Queue

* AWS SQS
* Rabbit MQ

1. Domain Name Systems
2. Sequencers
3. Distributed Caching
4. Publish-Subscribe systems
5. Sharded Counters
6. Distributed Messaging Queues
7. Distributed Task Scheduling
8. Distributed Logging