Scalability Key takeaways

- Scalability core concepts you've learned in those lessons:
  - Scalability is a concern at any IT level: hardware (including network), Operating systems, and applications. Developing scalable application is a MUST.
  - In the database world, scalability has to be tuned along with performance. It's about achieving the best possible performance consuming as few resources as possible.
  - In a system, there is always a point from which scalability is compromised. When the workload overpasses this point, the response time of the system skyrockets.
  - An application can give good **performance**, but bad **scalability**: meaning that its performance will dramatically **drop under concurrency**.
  - It's about getting the best possible Response Time with the lowest DB Time
  - Tuning a database system for scalability is several orders of magnitude cheaper than scaling-up the database server (cost of hardware, software licenses by cores, etc ...).
  - It might be even useless to scale-up the hardware. If the scalability problem is caused by a piece of software (application, OS, database RDBMS software), you won't see any benefit scaling the hardware up.

- Horizontal vs vertical scalability:
  - Horizontal scalability is about adding more servers, working as a cluster: this is achieved by running a cluster database. Though not related to database, the "Seti at home" project was a great example of scaling up horizontally, with thousands of machines running in parallel.
  - Vertical scalability is about running your database on a bigger server

- Cluster database key concepts:
  - A cluster database is made of:
    - *A single database*: datafiles reside on a **shared storage**
    - *Several instances* running that database. Each instance runs on a **single server**, on which are running the background processes, and is allocated the memory for that instance.
  - A **cluster database software** ensures that instances are all synchronized with each other.
  - The workload against the database is balanced on all the instances. This load balancing is done by a "less busy" mechanism, meaning that a new session will always be connected to the less busy instance (in terms of CPU consumption).
  - Cluster database software must avoid **split brain**: split brain occurs when instances inside the cluster start to act as if they were alone in the cluster. This happens typically when the private connection is lost between instances, hence the cluster software is not able to synchronize them. To avoid split brain, the software cluster has to perform **node eviction**, removing one or more instances from the cluster configuration.
  - Be able to draw a typical high level architecture of a cluster database (slide 17 in Scalability lesson).
- Cluster database vs sharded database:
  - Be able to draw a high level architecture of a sharded database (slide 18 in Scalability lesson).
  - Sharded database key concepts:
    - Shard or replication group
    - Data is distributed evenly among all the shards usually by a hash algorithm. Other distribution methods are also available, like list or range.

- Inside a shard, a Master does the write, and some replicas are receiving the data in an asynchronous manner. Read operations can be performed against either the master or the replica.
- Production Sharded databases should be deployed on a number of servers equal to the number of shards, as per best practices.

- Be aware of the main differences between a Cluster relational database and a sharded database (slide 19, Scalability lesson). Nowadays, relational databases can usually be sharded or clustered (these two options are not mutually exclusive).