- Imagine you are a manager and you are having a hotel. 1 chef is there in the hotel.
  - Vertical Scaling:
    - To maximize the work, you can say the chef to work harder and he'll be paid more:
      - *Optimize processes and increase throughput with the same resources.*
    - Preparing before hand at non-peak hours. So that the chef doesn't have to work the things when the store is busy.
      - *Pre-processing and Cronjob.*
    - Hire 1 backup chef, so that if the actual chef is not available, the hotel will still be running without any problem
      - *Keep the backups available to prevent single points of failure.*
    - *Master-slave*
  - Horizontal Scaling:
    - Hire more chefs to scale-up the work
      - *Buying more machines of similar types to get more work done.*
  - Microservice:
    - Let you have a pizza shop, 3 teams are there. 1: pizza, 2: garlic bread, 3: pizza. How would u route them for a pizza and garlic bread order?
      - Its simpler if we route all the pizza orders to the pizza team and garlic bread orders to the garlic bread team.
      - This system will be scalable, as the particular team consists of the chefs who have expertise in that particular field.
      - *Microservice architecture. You have all of your responsibilities handled separately.*
  - Distributed Systems:
    - What if electricity goes down for a day?
      - Now your store will go down. You won't have business that day.
      - Buy 2 shops. It'll cost you more, and shops will be having lesser no. of chefs; but at least you'll have backup.
      - Complexity will be increased. When you'll get an order, you need to route this order to one shop that is preferable for the same.
      - *Distributed System*
      - *In facebook, it gets requests from multiple places all acroll the glove. So, it has to have local servers to route the request and give faster response, fault tolerance.*

- Load Balancer:
  - Let you are having 2 pizza shops PS1, PS2. (PS1 is little popular than PS2)
  - You have customers, delivery agents.
  - There should be one central entity that decides from which shop the orders should be delivered to the customer.
  - It is called ***Load Balancer***.
  - Let
    - from PS1: cooking-***1***hr, packing-5min, delivery-***10***min => 1 hr 15 min
    - from PS2: cooking-***15***min, packing-5min, delivery-***40***min => 1hr 5 min
  - So, the order should be processed from PS2 even if it is far from the customer.
  - Now the system is ***Fault Tolerant***.
- How would you make is flexible to change?
  -