## **CAP Theorem — Trade-offs in Distributed Systems**

### **What is CAP Theorem?**

* Proposed by Eric Brewer.
* States that **a distributed system can only guarantee two of these three properties at the same time:**
  + **Consistency (C)** – Every node sees the same data at the same time.
  + **Availability (A)** – Every request gets a response (even if it’s not the latest data).
  + **Partition Tolerance (P)** – System continues to operate even if network failures split it into isolated parts.

### **Why can't we have all three?**

* In a **distributed system**, network partitions (P) **can and will happen** — links fail, nodes crash, packets drop.
* So you must **choose between Consistency or Availability** when a partition occurs.

### **Types of Systems Based on CAP**

1. **CP (Consistency + Partition Tolerance)**
   1. System prioritizes *consistent data* over availability.
   2. If a partition occurs, some requests will fail instead of serving stale data.
   3. **Examples:** HBase, MongoDB (with strong consistency mode), Zookeeper.
2. **AP (Availability + Partition Tolerance)**
   1. System prioritizes *availability* over strict consistency.
   2. Data might be temporarily inconsistent but eventually consistent.
   3. **Examples:** Cassandra, DynamoDB, Riak.
3. **CA (Consistency + Availability)**
   1. Works **only in non-distributed (single node)** systems or when partitions never happen.
   2. **Examples:** Traditional RDBMS like PostgreSQL, MySQL in standalone mode.

### **Key Takeaways**

* **Partition tolerance (P)** is *non-negotiable* for distributed systems.
* In practice, you **trade between C and A** during network issues.
* **AP systems** are good for high availability but relax strict consistency.
* **CP systems** are good for applications needing strict correctness (e.g. banking).