# CAP theorem

Let's take a detailed look at the three distributed system characteristics to which the CAP theorem refers.

**Consistency**

Consistency means that all clients see the same data at the same time, no matter which node they connect to. For this to happen, whenever data is written to one node, it must be instantly forwarded or replicated to all the other nodes in the system before the write is deemed 'successful.'

**Availability**

Availability means that any client making a request for data gets a response, even if one or more nodes are down. Another way to state this—all working nodes in the distributed system return a valid response for any request, without exception.

**Partition tolerance**

A partition is a communications break within a distributed system—a lost or temporarily delayed connection between two nodes. Partition tolerance means that the cluster must continue to work despite any number of communication breakdowns between nodes in the system.
CAP theorem NoSQL database types

NoSQL databases are ideal for distributed network applications. Unlike their vertically scalable SQL (relational) counterparts, NoSQL databases are horizontally scalable and distributed by design—they can rapidly scale across a growing network consisting of multiple interconnected nodes. (See "SQL vs. NoSQL Databases: What's the Difference?" for more information.)

Today, NoSQL databases are classified based on the two CAP characteristics they support:

- **CP database:** A CP database delivers consistency and partition tolerance at the expense of availability. When a partition occurs between any two nodes, the system has to shut down the non-consistent node (i.e., make it unavailable) until the partition is resolved.

- **AP database:** An AP database delivers availability and partition tolerance at the expense of consistency. When a partition occurs, all nodes remain available but those at the wrong end of a partition might return an older version of data than others. (When the partition is resolved, the AP databases typically resync the nodes

to repair all inconsistencies in the system.)

- **CA database:** A CA database delivers consistency and availability across all nodes. It can't do this if there is a partition between any two nodes in the system, however, and therefore can't deliver fault tolerance.

We listed the CA database type last for a reason—in a distributed system, partitions can't be avoided. So, while we can discuss a CA distributed database in theory, for all practical purposes a CA distributed database can't exist. This doesn't mean you can't have a CA database for your distributed application if you need one. Many relational databases, such as PostgreSQL, deliver consistency and availability and can be deployed to multiple nodes using replication.

# Microservices and the CAP theorem

Microservices are loosely coupled, independently deployable application components that incorporate their own stack—including their own database and database model—and communicate with each other over a network. As you can run microservices on both cloud servers and on-premises data centers, they have become highly popular for hybrid and multicloud applications.

Understanding the CAP theorem can help you choose the best database when designing a microservices-based application running from multiple locations. For example, if the ability to quickly iterate the data model and scale horizontally is essential to your application, but you can tolerate eventual (as opposed to strict) consistency, an AP database like Cassandra or Apache CouchDB can meet your requirements and simplify your deployment. On the other hand, if your application depends heavily on data consistency—as in an eCommerce application or a payment service—you might opt for a relational database like PostgreSQL.