What is the use of leader and follower of each partition in topic in Apache Kafka broker?

In Apache Kafka, topics are divided into partitions to achieve **scalability and fault tolerance**. **Each partition has one leader and zero or more followers**. Here's a breakdown of the roles and uses of the leader and follower for each partition:

**Leader**

1. **Primary Responsibility:** The leader (One of the Kafka Brokers in Kafka Cluster) is responsible for all read and write operations for the partition. When producers send messages to the partition, they communicate with the leader. Similarly, when consumers read messages, they read from the leader.
2. **Coordination with Zookeeper/Kafka Controller**: The Kafka Controller (elected from among the brokers) manages the leadership and replication. If a leader fails, the controller promotes one of the followers to be the new leader.
3. **Data Consistency:** The leader ensures that all replicas (followers) have the same data. **It writes the messages to its local log and then replicates them to the followers.**

**Followers**

1. **Replication:** Followers replicate the leader's data to ensure fault tolerance. They continually pull data from the leader and write it to their local log.
2. **Readiness for Failover:** Followers are ready to take over as the leader in case the current leader fails. This provides high availability and resilience in the Kafka cluster.
3. **Data Consistency:** Followers help in maintaining data consistency across the cluster. **They acknowledge the data replication to the leader, which then ensures the data is committed.**

**Benefits of Leader and Follower Architecture**

1. **Scalability:** By distributing partitions and having a leader for each, Kafka can handle a large number of read and write operations simultaneously.
2. **Fault Tolerance:** If a leader fails, one of the followers can quickly be promoted to leader, minimizing downtime and data loss.
3. **Load Balancing:** Since each partition has a different leader, the load of read and write operations is distributed across multiple brokers, preventing any single broker from becoming a bottleneck.

**Key Points**

1. **Leader Election:** Managed by Kafka's internal controller (or Zookeeper in older versions), ensuring smooth transitions.
2. **Replication Factor:** Determines the number of replicas (including the leader). A higher replication factor increases fault tolerance.
3. **ISR (In-Sync Replicas):** The set of replicas (including the leader) that are currently up-to-date with the leader. Consumers can be configured to only read from leaders that have a majority of ISR to ensure data consistency.

In summary, the leader and follower architecture in Kafka ensures high availability, fault tolerance, and scalability by distributing the load and maintaining multiple copies of the data across the cluster.