* **Broker** (Kafka) - A single member server of the Kafka cluster
* **Cluster** (Kafka) - A group of one or more Kafka Brokers working together to satisfy Kafka production and consumption
* **Node** - A single computing instance. May be physical, as in a server in a datacenter, or virtual, as an instance might be in AWS, GCP, or Azure.
* **Zookeeper** - Used by Kafka Brokers to determine which broker is the leader of a given partition and topic, as well as track cluster membership and configuration for Kafka
* **Access Control List (ACL)** - Permissions associated with an object. In Kafka, this typically refers to a user’s permissions with respect to production and consumption, and/or the topics themselves.
* **JVM - The Java Virtual Machine** - Responsible for allowing host computers to execute the byte-code compiled against the JVM.
* **Data Partition** (Kafka) - Kafka topics consist of one or more partitions. A partition is a log which provides ordering guarantees for all of the data contained within it. Partitions are chosen by hashing key values.
* **Data Replication** (Kafka) - A mechanism by which data is written to more than one broker to ensure that if a single broker is lost, a replicated copy of the data is available.
* **In-Sync Replica (ISR)** - A broker which is up to date with the leader for a particular broker for all of the messages in the current topic. This number may be less than the replication factor for a topic.
* **Rebalance** - A process in which the current set of consumers changes (addition or removal of consumer). When this occurs, assignment of partitions to the various consumers in a consumer group must be changed.
* **Data Expiration** - A process in which data is removed from a Topic log, determined by data retention policies.
* **Data Retention** - Policies that determine how long data should be kept. Configured by time or size.
* **Batch Size** - The number of messages that are sent or received from Kafka
* **acks** - The number of broker acknowledgements that must be received from Kafka before a producer continues processing
* **Synchronous Production** - Producers which send a message and wait for a response before performing additional processing
* **Asynchronous Production** - Producers which send a message and do not wait for a response before performing additional processing
* **Avro** - A binary message serialization format
* **Message Serialization** - The process of transforming an applications internal data representation to a format suitable for interprocess communication over a protocol like TCP or HTTP.
* **Message Deserialization** - The process of transforming an incoming set of data from a form suitable for interprocess communication, into a data representation more suitable for the application receiving the data.
* **Retries** (Kafka Producer) - The number of times the underlying library will attempt to deliver data before moving on
* **Consumer Offset** - A value indicating the last seen and processed message of a given consumer, by ID.
* **Consumer Group** - A collection of one or more consumers, identified by group.idwhich collaborate to consume data from Kafka and share a consumer offset.
* **Consumer Group Coordinator** - The broker in charge of working with the Consumer Group Leader to initiate a rebalance
* **Consumer Group Leader** - The consumer in charge of working with the Group Coordinator to manage the consumer group
* **Topic Subscription** - Kafka consumers indicate to the Kafka Cluster that they would like to consume from one or more topics by specifying one or more topics that they wish to subscribe to.
* **Consumer Lag** - The difference between the offset of a consumer group and the latest message offset in Kafka itself
* **CCPA** - California Consumer Privacy Act
* **GDPR** - General Data Protection Regulation

Glossary of Terms: (same as provided at beginning of lesson)

* Broker (Kafka) - A single member server of the Kafka cluster
* Cluster (Kafka) - A group of one or more Kafka Brokers working together to satisfy Kafka production and consumption
* Node - A single computing instance. May be physical, as in a server in a datacenter, or virtual, as an instance might be in AWS, GCP, or Azure.
* Zookeeper - Used by Kafka Brokers to determine which broker is the leader of a given partition and topic, as well as track cluster membership and configuration for Kafka
* Access Control List (ACL) - Permissions associated with an object. In Kafka, this typically refers to a user’s permissions with respect to production and consumption, and/or the topics themselves.
* JVM - The Java Virtual Machine. Responsible for allowing host computers to execute the byte-code compiled against the JVM.
* Data Partition (Kafka) - Kafka topics consist of one or more partitions. A partition is a log which provides ordering guarantees for all of the data contained within it. Partitions are chosen by hashing key values.
* Data Replication (Kafka) - A mechanism by which data is written to more than one broker to ensure that if a single broker is lost, a replicated copy of the data is available.
* In-Sync Replica (ISR) - A broker which is up to date with the leader for a particular broker for all of the messages in the current topic. This number may be less than the replication factor for a topic.
* Rebalance - A process in which the current set of consumers changes (addition or removal of consumer). When this occurs, assignment of partitions to the various consumers in a consumer group must be changed.
* Data Expiration - A process in which data is removed from a Topic log, determined by data retention policies.
* Data Retention - Policies that determine how long data should be kept. Configured by time or size.
* Batch Size - The number of messages that are sent or received from Kafka
* acks - The number of broker acknowledgements that must be received from Kafka before a producer continues processing
* Synchronous Production - Producers which send a message and wait for a response before performing additional processing
* Asynchronous Production - Producers which send a message and do not wait for a response before performing additional processing
* Avro - A binary message serialization format
* Message Serialization - The process of transforming an applications internal data representation to a format suitable for interprocess communication over a protocol like TCP or HTTP.
* Message Deserialization - The process of transforming an incoming set of data from a form suitable for interprocess communication, into a data representation more suitable for the application receiving the data.
* Retries (Kafka Producer) - The number of times the underlying library will attempt to deliver data before moving on
* Consumer Offset - A value indicating the last seen processed of a given consumer, by ID.
* Consumer Group - A collection of one or more consumers, identified by group.idwhich collaborate to consume data from Kafka and share a consumer offset.
* Consumer Group Coordinator - The broker in charge of working with the Consumer Group Leader to initiate a rebalance
* Consumer Group Leader - The consumer in charge of working with the Group Coordinator to manage the consumer group
* Topic Subscription - Kafka consumers indicate to the Kafka Cluster that they would like to consume from one or more topics by specifying one or more topics that they wish to subscribe to.
* Consumer Lag - The difference between the offset of a consumer group and the latest message offset in Kafka itself
* CCPA - California Consumer Privacy Act
* GDPR - General Data Protection Regulation

Glossary of Key Terms You Will Learn in this Lesson

* **Data Schema** - Define the shape of a particular kind of data. Specifically, data schemas define the expected fields, their names, and value types for those fields. Data schemas may also indicate whether fields are required or optional.
* **Apache Avro** - A data serialization framework which includes facilities for defining and communicating data schemas. Avro is widely used in the Kafka ecosystem and data engineering generally.
* **Record** (Avro) - A single encoded record in the defined Avro format
* **Primitive Type** (Avro) - In Avro, a primitive type is a type which requires no additional specification - null, boolean, int, long, float, double, bytes, string.
* **Complex Type** (Avro) - In Avro, a complex type models data structures which may involve nesting or other advanced functionality: records, enums, maps, arrays, unions, fixed.
* **Schema Evolution** - The process of modifying an existing schema with new, deleted, or modified fields.
* **Schema Compatibility** - Determines whether or not two given versions of a schema are usable by a given client
* **Backward Compatibility** - means that consumer code developed against the most recent version of an Avro Schema can use data using the prior version of a schema without modification.
* **Forward Compatibility** - means that consumer code developed against the previous version of an Avro Schema can consume data using the newest version of a schema without modification.
* **Full Compatibility** - means that consumers developed against the latest schema can consume data using the previous schema, and that consumers developed against the previous schema can consume data from the latest schema as well. In other words, full compatibility means that a schema change is both forward and backward compatible.
* **None Compatibility** - disables compatibility checking by Schema Registry.

## Glossary of Key Terms in Lesson (same as presented at beginning of lesson)

* Data Schema - Define the shape of a particular kind of data. Specifically, data schemas define the expected fields, their names, and value types for those fields. Data schemas may also indicate whether fields are required or optional.
* Apache Avro - A data serialization framework which includes facilities for defining and communicating data schemas. Avro is widely used in the Kafka ecosystem and data engineering generally.
* Record (Avro) - A single encoded record in the defined Avro format
* Primitive Type (Avro) - In Avro, a primitive type is a type which requires no additional specification - null, boolean, int, long, float, double, bytes, string.
* Complex Type (Avro) - In Avro, a complex type models data structures which may involve nesting or other advanced functionality: records, enums, maps, arrays, unions, fixed.
* Schema Evolution - The process of modifying an existing schema with new, deleted, or modified fields.
* Schema Compatibility - Determines whether or not two given versions of a schema are usable by a given client
* Backward Compatibility - means that consumer code developed against the most recent version of an Avro Schema can use data using the prior version of a schema without modification.
* Forward Compatibility - means that consumer code developed against the previous version of an Avro Schema can consume data using the newest version of a schema without modification.
* Full Compatibility - means that consumers developed against the latest schema can consume data using the previous schema, and that consumers developed against the previous schema can consume data from the latest schema as well. In other words, full compatibility means that a schema change is both forward and backward compatible.
* None Compatibility - disables compatibility checking by Schema Registry.