

Confluent - Updated CCDAK New Dumps



BONUS!!! Download part of PassLeader CCDAK dumps for free: <https://drive.google.com/open?id=1C1mLurKW7O2FVL9DKTDENymIFCSHfWME>

If you are preparing for the Confluent Certified Developer for Apache Kafka Certification Examination (CCDAK) exam dumps our CCDAK Questions help you to get high scores in your Confluent Certified Developer for Apache Kafka Certification Examination (CCDAK) exam. Test your knowledge of the Confluent Certified Developer for Apache Kafka Certification Examination (CCDAK) exam dumps with PassLeader Confluent Certified Developer for Apache Kafka Certification Examination (CCDAK) practice questions. The software is designed to help with Confluent Certified Developer for Apache Kafka Certification Examination (CCDAK) exam dumps preparation. Confluent Certified Developer for Apache Kafka Certification Examination (CCDAK) practice test software can be used on devices that range from mobile devices to desktop computers.

Confluent Certified Developer for Apache Kafka Certification Examination (CCDAK) is a technical certification exam designed for Kafka developers. The Kafka platform has gained immense popularity in recent years, and organizations worldwide are seeking professionals with expertise in this technology. The CCDAK Exam is designed to test a developer's understanding of Kafka fundamentals, architecture, and its application in real-world scenarios.

The CCDAK certification exam is administered by Confluent, the creators of Apache Kafka. Confluent is a company that provides an enterprise-ready distribution of Apache Kafka, along with tools and services for building and managing Kafka-based applications. Confluent Certified Developer for Apache Kafka Certification Examination certification exam is intended for developers who work with Apache Kafka in their day-to-day roles and want to validate their skills and knowledge.

>> CCDAK New Dumps <<

100% Pass 2026 Professional Confluent CCDAK New Dumps

Our CCDAK real test was designed by many experts in different area, they have taken the different situation of customers into consideration and designed practical CCDAK study materials for helping customers save time. Whether you are a student or an office worker, we believe you will not spend all your time on preparing for CCDAK Exam, you are engaged in studying your specialized knowledge, doing housework, looking after children and so on. With our simplified information, you are able to study efficiently. And do you want to feel the true exam in advance? Just buy our CCDAK exam questions!

Confluent Certified Developer for Apache Kafka Certification Examination Sample Questions (Q206-Q211):

NEW QUESTION # 206

To prevent network-induced duplicates when producing to Kafka, I should use

- A. batch.size=1
- B. enable.idempotence=true
- C. retries=200000
- D. max.in.flight.requests.per.connection=1

Answer: B

Explanation:

Producer idempotence helps prevent the network introduced duplicates. More details here <https://cwiki.apache.org/confluence/display/KAFKA/Idempotent+Producer>

NEW QUESTION # 207

Match each configuration parameter with the correct deployment step in installing a Kafka connector.

Options	Questions
Place the connector's JAR file in the directory specified by the 'plugin.path' configuration.	
Configure the connector using a JSON or properties file with the necessary settings.	
Restart the Kafka Connect cluster.	
Verify that the connector is installed by listing all available connectors using the Kafka Connect REST API (/connector-plugins).	
Configure the connector using a JSON or properties file with the necessary settings.	

Answer:

Explanation:

Options	Questions
Place the connector's JAR file in the directory specified by the 'plugin.path' configuration.	Place the connector's JAR file in the directory specified by the 'plugin.path' configuration.
Configure the connector using a JSON or properties file with the necessary settings.	Restart the Kafka Connect cluster.
Restart the Kafka Connect cluster.	Verify that the connector is installed by listing all available connectors using the Kafka Connect REST API (/connector-plugins).
Verify that the connector is installed by listing all available connectors using the Kafka Connect REST API (/connector-plugins).	Configure the connector using a JSON or properties file with the necessary settings.
Configure the connector using a JSON or properties file with the necessary settings.	Configure the connector using a JSON or properties file with the necessary settings.

Explanation:

- * 1st: Place the connector's JAR file in the directory specified by plugin.path
- * 2nd: Restart the Kafka Connect cluster
- * 3rd: Verify using REST API (/connector-plugins)
- * 4th: Configure the connector
- * 5th: (Repeat of 4th, duplicate step)

1st # Place the connector's JAR file in the directory specified by the plugin.path configuration.

2nd # Restart the Kafka Connect cluster.

3rd # Verify that the connector is installed by listing all available connectors using the Kafka Connect REST API (/connector-plugins).

4th # Configure the connector using a JSON or properties file with the necessary settings.

5th # Configure the connector using a JSON or properties file with the necessary settings.

Kafka Connect requires that custom connectors be placed in the directory defined by plugin.path. After restarting the cluster, you can use the REST API to confirm availability and then deploy the connector configuration.

From Kafka Connect Documentation:

"After placing the JAR in the plugin.path, you must restart the Connect cluster to pick it up. Use the /connector-plugins REST endpoint to verify."

The duplication of configuration is an error in the question options and should occur only once.

Reference: Kafka Connect Plugin Installation Guide

NEW QUESTION # 208

Which statement is true about how exactly-once semantics (EOS) work in Kafka Streams?

- **A. EOS in Kafka Streams relies on transactional producers to atomically commit state updates to changelog topics and output records to Kafka.**
- B. Kafka Streams provides EOS by periodically checkpointing state stores and replaying changelogs to recover only unprocessed messages during failure.
- C. Kafka Streams disables log compaction on internal changelog topics to preserve all state changes for potential recovery.
- D. EOS in Kafka Streams is implemented by creating a separate Kafka topic for deduplication of all messages processed by the application.

Answer: A

Explanation:

Kafka Streams uses transactional producers to guarantee exactly-once semantics (EOS). This ensures that both the output records and state store updates are committed atomically, avoiding duplication or partial writes.

From Kafka Streams Documentation > Processing Guarantees:

"Kafka Streams leverages Kafka's transactional API to commit the output records and internal state updates as a single atomic unit, thereby providing exactly-once semantics."

* Option A is incorrect because log compaction is not disabled for EOS.

* Option C incorrectly describes a checkpointing system Kafka Streams does not use.

* Option D refers to deduplication, which is not how EOS is achieved in Streams.

Reference: Kafka Streams Processing Guarantees

NEW QUESTION # 209

You need to configure a sink connector to write records that fail into a dead letter queue topic.

Requirements:

* Topic name: DLQ-Topic

* Headers containing error context must be added to the messages Which three configuration parameters are necessary? (Select three.)

- A. `errors.log.enable=true`
- B. `errors.log.include.messages=true`
- **C. `errors.tolerance=all`**
- D. `errors.tolerance=none`
- **E. `errors.deadletterqueue.context.headers.enable=true`**
- **F. `errors.deadletterqueue.topic.name=DLQ-Topic`**

Answer: C,E,F

Explanation:

To send failed records to a dead letter queue (DLQ), you must configure:

* `errors.tolerance=all`: Tells the connector to not fail on errors but handle them (e.g., send to DLQ).

* `errors.deadletterqueue.topic.name=DLQ-Topic`: Specifies the DLQ topic.

* `errors.deadletterqueue.context.headers.enable=true`: Includes error context in message headers.

From Kafka Connect Error Handling Docs:

"Kafka Connect supports directing problematic records to a separate topic (DLQ) using `errors.*` configs.

Headers can include failure metadata."

Options D, E, F are related to logging, not DLQ behavior.

Reference: Kafka Connect Configurations > Error Handling

NEW QUESTION # 210

What is returned by a `producer.send()` call in the Java API?

- A. `Future<ProducerRecord>` object
- B. A Boolean indicating if the call succeeded
- **C. `Future<RecordMetadata>` object**
- D. Unit

Answer: C

See <https://kafka.apache.org/21/javadoc/org/apache/kafka/clients/producer/KafkaProducer.html>

• • • • •

CCDAK Key Concepts: <https://www.passleader.top/Confluent/CCDAK-exam-braindumps.html>

- BONUS!!! Download part of PassLeader CCDAK dumps for free: <https://drive.google.com/open?id=1C1mLurKW7O2FVL9DKTDENymlFCSHfWME>