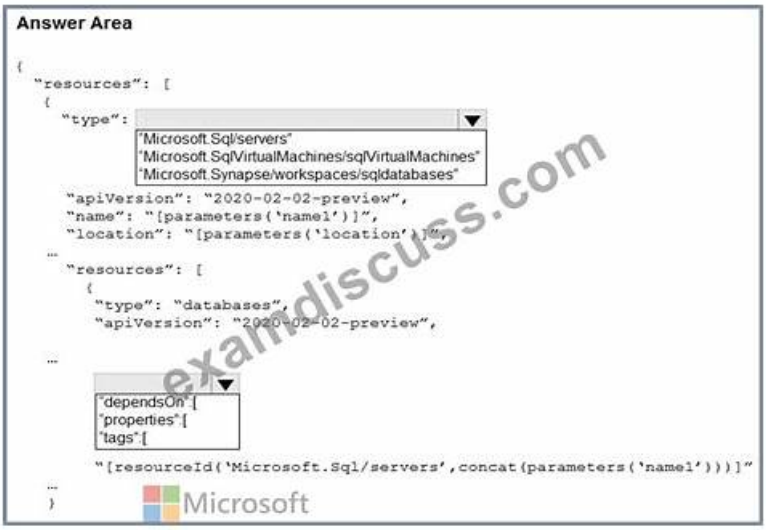
ACD-301 Valid Braindumps Ppt | Amazing Pass Rate For Appian ACD-301 | ACD-301: Appian Certified Lead Developer



BTW, DOWNLOAD part of BraindumpsPass ACD-301 dumps from Cloud Storage: https://drive.google.com/open?id=1AdlJ5OEeTZOp826dbJrr53bdLq2jolbU

Our ACD-301 exam dumps are possessed with high quality which is second to none. Just as what have been reflected in the statistics, the pass rate for those who have chosen our ACD-301 exam guide is as high as 99%. In addition, our ACD-301 test prep is renowned for free renewal in the whole year. With our ACD-301 Training Materials, you will find that not only you can pass and get your certification easily, but also your future is obvious bright. Our ACD-301 training guide is worthy to buy.

If you study with our ACD-301 exam questions, you are bound to get the certification. The scientific design of ACD-301 preparation quiz allows you to pass exams faster, and the high passing rate will also make you more at ease. In this age of anxiety, being able to meet such a product is really fortunate for you. Choosing ACD-301 training engine will make you feel even more powerful. You can improve your ability more easily. When others work hard, you are already ahead!

>> ACD-301 Valid Braindumps Ppt <<

ACD-301 Valid Braindumps Ppt - 100% Pass Quiz Appian - First-grade ACD-301 - Appian Certified Lead Developer Trustworthy Pdf

In addition to the Appian ACD-301 PDF dumps, we also offer Appian ACD-301 practice exam software. You will find the same ambiance and atmosphere when you attempt the real Appian ACD-301 exam. It will make you practice nicely and productively as you will experience better handling of the Appian ACD-301 Questions when you take the actual ACD-301 exam to grab the Appian Certified Lead Developer certification.

Appian Certified Lead Developer Sample Questions (Q14-Q19):

NEW QUESTION # 14
You are designing a process that is anticipated to be executed multiple times a day. This process retrieves data from an external system and then calls various utility processes as needed. The main process will not use the results of the utility processes, and there are no user forms anywhere.
Which design choice should be used to start the utility processes and minimize the load on the execution engines?

- A. Use the Start Process Smart Service to start the utility processes.
- B. Start the utility processes via a subprocess asynchronously.
- C. Use Process Messaging to start the utility process.
- D. Start the utility processes via a subprocess synchronously.

Answer: B

Explanation:
Comprehensive and Detailed In-Depth Explanation:
As an Appian Lead Developer, designing a process that executes frequently (multiple times a day) and calls utility processes without using their results requires optimizing performance and minimizing load on Appian's execution engines. The absence of user forms indicates a backend process, so user experience isn't a concern-only engine efficiency matters. Let's evaluate each option:
A . Use the Start Process Smart Service to start the utility processes:
The Start Process Smart Service launches a new process instance independently, creating a separate process in the Work Queue. While functional, it increases engine load because each utility process runs as a distinct instance, consuming engine resources and potentially clogging the Java Work Queue, especially with frequent executions. Appian's performance guidelines discourage unnecessary separate process instances for utility tasks, favoring integrated subprocesses, making this less optimal.
B . Start the utility processes via a subprocess synchronously:
Synchronous subprocesses (e.g., a!startProcess with isAsync: false) execute within the main process flow, blocking until completion. For utility processes not used by the main process, this creates unnecessary delays, increasing execution time and engine load. With frequent daily executions, synchronous subprocesses could strain engines, especially if utility processes are slow or numerous. Appian's documentation recommends asynchronous execution for non-dependent, non-blocking tasks, ruling this out.
C . Use Process Messaging to start the utility process:
Process Messaging (e.g., sendMessage() in Appian) is used for inter-process communication, not for starting processes. It's designed to pass data between running processes, not initiate new ones. Attempting to use it for starting utility processes would require additional setup (e.g., a listening process) and isn't a standard or efficient method. Appian's messaging features are for coordination, not process initiation, making this inappropriate.
D . Start the utility processes via a subprocess asynchronously:
This is the best choice. Asynchronous subprocesses (e.g., a!startProcess with isAsync: true) execute independently of the main process, offloading work to the engine without blocking or delaying the parent process. Since the main process doesn't use the utility process results and there are no user forms, asynchronous execution minimizes engine load by distributing tasks across time, reducing Work Queue pressure during frequent executions. Appian's performance best practices recommend asynchronous subprocesses for non-dependent, utility tasks to optimize engine utilization, making this ideal for minimizing load.
Conclusion: Starting the utility processes via a subprocess asynchronously (D) minimizes engine load by allowing independent execution without blocking the main process, aligning with Appian's performance optimization strategies for frequent, backend processes.
Appian Documentation: "Process Model Performance" (Synchronous vs. Asynchronous Subprocesses).
Appian Lead Developer Certification: Process Design Module (Optimizing Engine Load).
Appian Best Practices: "Designing Efficient Utility Processes" (Asynchronous Execution).

NEW QUESTION # 15
You add an index on the searched field of a MySQL table with many rows (>100k). The field would benefit greatly from the index in which three scenarios?

- A. The field contains long unstructured text such as a hash.
- B. The field contains many datetimes, covering a large range.
- C. The field contains a structured JSON.
- D. The field contains a textual short business code.
- E. The field contains big integers, above and below 0.

Answer: B,D,E

Explanation:
Comprehensive and Detailed In-Depth Explanation:
Adding an index to a searched field in a MySQL table with over 100,000 rows improves query performance by reducing the number of rows scanned during searches, joins, or filters. The benefit of an index depends on the field's data type, cardinality (uniqueness), and query patterns. MySQL indexing best practices, as aligned with Appian's Database Optimization Guidelines, highlight scenarios where indices are most effective.
Option A (The field contains a textual short business code):
This benefits greatly from an index. A short business code (e.g., a 5-10 character identifier like "CUST123") typically has high cardinality (many unique values) and is often used in WHERE clauses or joins. An index on this field speeds up exact-match queries (e.g., WHERE business_code = 'CUST123'), which are common in Appian applications for lookups or filtering.
Option C (The field contains many datetimes, covering a large range):
This is highly beneficial. Datetime fields with a wide range (e.g., transaction timestamps over years) are frequently queried with range conditions (e.g., WHERE datetime BETWEEN '2024-01-01' AND '2025-01-01') or sorting (e.g., ORDER BY datetime). An

index on this field optimizes these operations, especially in large tables, aligning with Appian's recommendation to index time-based fields for performance.
Option D (The field contains big integers, above and below 0):
This benefits significantly. Big integers (e.g., IDs or quantities) with a broad range and high cardinality are ideal for indexing. Queries like WHERE id > 1000 or WHERE quantity < 0 leverage the index for efficient range scans or equality checks, a common pattern in Appian data store queries.
Option B (The field contains long unstructured text such as a hash):
This benefits less. Long unstructured text (e.g., a 128-character SHA hash) has high cardinality but is less efficient for indexing due to its size. MySQL indices on large text fields can slow down writes and consume significant storage, and full-text searches are better handled with specialized indices (e.g., FULLTEXT), not standard B-tree indices. Appian advises caution with indexing large text fields unless necessary.
Option E (The field contains a structured JSON):
This is minimally beneficial with a standard index. MySQL supports JSON fields, but a regular index on the entire JSON column is inefficient for large datasets (>100k rows) due to its variable structure. Generated columns or specialized JSON indices (e.g., using JSON_EXTRACT) are required for targeted queries (e.g., WHERE JSON_EXTRACT(json_col, '$.key') = 'value'), but this requires additional setup beyond a simple index, reducing its immediate benefit.
For a table with over 100,000 rows, indices are most effective on fields with high selectivity and frequent query usage (e.g., short codes, datetimes, integers), making A, C, and D the optimal scenarios.

NEW QUESTION # 16
As part of an upcoming release of an application, a new nullable field is added to a table that contains customer dat a. The new field is used by a report in the upcoming release and is calculated using data from another table.
Which two actions should you consider when creating the script to add the new field?

- A. Create a rollback script that clears the data from the field.
- B. Add a view that joins the customer data to the data used in calculation.
- C. Create a script that adds the field and leaves it null.
- D. Create a rollback script that removes the field.
- E. Create a script that adds the field and then populates it.

Answer: D,E

Explanation:
Comprehensive and Detailed In-Depth Explanation:
As an Appian Lead Developer, adding a new nullable field to a database table for an upcoming release requires careful planning to ensure data integrity, report functionality, and rollback capability. The field is used in a report and calculated from another table, so the script must handle both deployment and potential reversibility. Let's evaluate each option:
A . Create a script that adds the field and leaves it null:
Adding a nullable field and leaving it null is technically feasible (e.g., using ALTER TABLE ADD COLUMN in SQL), but it doesn't address the report's need for calculated data. Since the field is used in a report and calculated from another table, leaving it null risks incomplete or incorrect reporting until populated, delaying functionality. Appian's data management best practices recommend populating data during deployment for immediate usability, making this insufficient as a standalone action.
B . Create a rollback script that removes the field:
This is a critical action. In Appian, database changes (e.g., adding a field) must be reversible in case of deployment failure or rollback needs (e.g., during testing or PROD issues). A rollback script that removes the field (e.g., ALTER TABLE DROP COLUMN) ensures the database can return to its original state, minimizing risk. Appian's deployment guidelines emphasize rollback scripts for schema changes, making this essential for safe releases.
C . Create a script that adds the field and then populates it:
This is also essential. Since the field is nullable, calculated from another table, and used in a report, populating it during deployment ensures immediate functionality. The script can use SQL (e.g., UPDATE table SET new_field = (SELECT calculated_value FROM other_table WHERE condition)) to populate data, aligning with Appian's data fabric principles for maintaining data consistency. Appian's documentation recommends populating new fields during deployment for reporting accuracy, making this a key action.
D . Create a rollback script that clears the data from the field:
Clearing data (e.g., UPDATE table SET new_field = NULL) is less effective than removing the field entirely. If the deployment fails, the field's existence with null values could confuse reports or processes, requiring additional cleanup. Appian's rollback strategies favor reverting schema changes completely (removing the field) rather than leaving it with nulls, making this less reliable and unnecessary compared to B.
E . Add a view that joins the customer data to the data used in calculation:
Creating a view (e.g., CREATE VIEW customer_report AS SELECT ... FROM customer_table JOIN other_table ON ...) is useful for reporting but isn't a prerequisite for adding the field. The scenario focuses on the field addition and population, not

reporting structure. While a view could optimize queries, it's a secondary step, not a primary action for the script itself. Appian's data modeling best practices suggest views as post-deployment optimizations, not script requirements.
Conclusion: The two actions to consider are B (create a rollback script that removes the field) and C (create a script that adds the field and then populates it). These ensure the field is added with data for immediate report usability and provide a safe rollback option, aligning with Appian's deployment and data management standards for schema changes.
Appian Documentation: "Database Schema Changes" (Adding Fields and Rollback Scripts).
Appian Lead Developer Certification: Data Management Module (Schema Deployment Strategies).
Appian Best Practices: "Managing Data Changes in Production" (Populating and Rolling Back Fields).

NEW QUESTION # 17
You have an active development team (Team A) building enhancements for an application (App X) and are currently using the TEST environment for User Acceptance Testing (UAT).
A separate operations team (Team B) discovers a critical error in the Production instance of App X that they must remediate. However, Team B does not have a hotfix stream for which to accomplish this. The available environments are DEV, TEST, and PROD.
Which risk mitigation effort should both teams employ to ensure Team A's capital project is only minorly interrupted, and Team B's critical fix can be completed and deployed quickly to end users?

- A. Team B must address changes in the TEST environment. These changes can then be tested and deployed directly to PROD. Once the deployment is complete, Team B can then communicate their changes to Team A to ensure they are incorporated as part of the next release.
- B. Team A must analyze their current codebase in DEV to merge the hotfix changes into their latest enhancements. Team B is then required to wait for the hotfix to follow regular deployment protocols from DEV to the PROD environment.
- C. Team B must address the changes directly in PRO As there is no hotfix stream, and DEV and TEST are being utilized for active development, it is best to avoid a conflict of components. Once Team A has completed their enhancements work, Team B can update DEV and TEST accordingly.
- D. Team B must communicate to Team A which component will be addressed in the hotfix to avoid overlap of changes. If overlap exists, the component must be versioned to its PROD state before being remediated and deployed, and then versioned back to its latest development state. If overlap does not exist, the component may be remediated and deployed without any version changes.

Answer: D

Explanation:
Comprehensive and Detailed In-Depth Explanation:
As an Appian Lead Developer, managing concurrent development and operations (hotfix) activities across limited environments (DEV, TEST, PROD) requires minimizing disruption to Team A's enhancements while ensuring Team B's critical fix reaches PROD quickly. The scenario highlights no hotfix stream, active UAT in TEST, and a critical PROD issue, necessitating a strategic approach. Let's evaluate each option:
A . Team B must communicate to Team A which component will be addressed in the hotfix to avoid overlap of changes. If overlap exists, the component must be versioned to its PROD state before being remediated and deployed, and then versioned back to its latest development state. If overlap does not exist, the component may be remediated and deployed without any version changes:
This is the best approach. It ensures collaboration between teams to prevent conflicts, leveraging Appian's version control (e.g., object versioning in Appian Designer). Team B identifies the critical component, checks for overlap with Team A's work, and uses versioning to isolate changes. If no overlap exists, the hotfix deploys directly; if overlap occurs, versioning preserves Team A's work, allowing the hotfix to deploy and then reverting the component for Team A's continuation. This minimizes interruption to Team A's UAT, enables rapid PROD deployment, and aligns with Appian's change management best practices.
B . Team A must analyze their current codebase in DEV to merge the hotfix changes into their latest enhancements. Team B is then required to wait for the hotfix to follow regular deployment protocols from DEV to the PROD environment:
This delays Team B's critical fix, as regular deployment (DEV → TEST → PROD) could take weeks, violating the need for "quick deployment to end users." It also risks introducing Team A's untested enhancements into the hotfix, potentially destabilizing PROD. Appian's documentation discourages mixing development and hotfix workflows, favoring isolated changes for urgent fixes, making this inefficient and risky.
C . Team B must address changes in the TEST environment. These changes can then be tested and deployed directly to PROD. Once the deployment is complete, Team B can then communicate their changes to Team A to ensure they are incorporated as part of the next release:
Using TEST for hotfix development disrupts Team A's UAT, as TEST is already in use for their enhancements. Direct deployment from TEST to PROD skips DEV validation, increasing risk, and doesn't address overlap with Team A's work. Appian's deployment guidelines emphasize separate streams (e.g., hotfix streams) to avoid such conflicts, making this disruptive and unsafe.
D . Team B must address the changes directly in PROD. As there is no hotfix stream, and DEV and TEST are being utilized for

active development, it is best to avoid a conflict of components. Once Team A has completed their enhancements work, Team B can update DEV and TEST accordingly:
Making changes directly in PROD is highly discouraged in Appian due to lack of testing, version control, and rollback capabilities, risking further instability. This violates Appian's Production governance and security policies, and delays Team B's updates until Team A finishes, contradicting the need for a "quick deployment." Appian's best practices mandate using lower environments for changes, ruling this out.
Conclusion: Team B communicating with Team A, versioning components if needed, and deploying the hotfix (A) is the risk mitigation effort. It ensures minimal interruption to Team A's work, rapid PROD deployment for Team B's fix, and leverages Appian's versioning for safe, controlled changes-aligning with Lead Developer standards for multi-team coordination.
Appian Documentation: "Managing Production Hotfixes" (Versioning and Change Management).
Appian Lead Developer Certification: Application Management Module (Hotfix Strategies).
Appian Best Practices: "Concurrent Development and Operations" (Minimizing Risk in Limited Environments).

NEW QUESTION # 18
A customer wants to integrate a CSV file once a day into their Appian application, sent every night at 1:00 AM. The file contains hundreds of thousands of items to be used daily by users as soon as their workday starts at 8:00 AM. Considering the high volume of data to manipulate and the nature of the operation, what is the best technical option to process the requirement?

- A. Process what can be completed easily in a process model after each integration, and complete the most complex tasks using a set of stored procedures.
- B. Create a set of stored procedures to handle the volume and the complexity of the expectations, and call it after each integration.
- C. Build a complex and optimized view (relevant indices, efficient joins, etc.), and use it every time a user needs to use the data.
- D. Use an Appian Process Model, initiated after every integration, to loop on each item and update it to the business requirements.

Answer: B

Explanation:
Comprehensive and Detailed In-Depth Explanation:
As an Appian Lead Developer, handling a daily CSV integration with hundreds of thousands of items requires a solution that balances performance, scalability, and Appian's architectural strengths. The timing (1:00 AM integration, 8:00 AM availability) and data volume necessitate efficient processing and minimal runtime overhead. Let's evaluate each option based on Appian's official documentation and best practices:
A . Use an Appian Process Model, initiated after every integration, to loop on each item and update it to the business requirements:
This approach involves parsing the CSV in a process model and using a looping mechanism (e.g., a subprocess or script task with fn!forEach) to process each item. While Appian process models are excellent for orchestrating workflows, they are not optimized for high-volume data processing. Looping over hundreds of thousands of records would strain the process engine, leading to timeouts, memory issues, or slow execution-potentially missing the 8:00 AM deadline. Appian's documentation warns against using process models for bulk data operations, recommending database-level processing instead. This is not a viable solution.
B . Build a complex and optimized view (relevant indices, efficient joins, etc.), and use it every time a user needs to use the data:
This suggests loading the CSV into a table and creating an optimized database view (e.g., with indices and joins) for user queries via a!queryEntity. While this improves read performance for users at 8:00 AM, it doesn't address the integration process itself. The question focuses on processing the CSV ("manipulate" and "operation"), not just querying. Building a view assumes the data is already loaded and transformed, leaving the heavy lifting of integration unaddressed. This option is incomplete and misaligned with the requirement's focus on processing efficiency.
C . Create a set of stored procedures to handle the volume and the complexity of the expectations, and call it after each integration:
This is the best choice. Stored procedures, executed in the database, are designed for high-volume data manipulation (e.g., parsing CSV, transforming data, and applying business logic). In this scenario, you can configure an Appian process model to trigger at 1:00 AM (using a timer event) after the CSV is received (e.g., via FTP or Appian's File System utilities), then call a stored procedure via the "Execute Stored Procedure" smart service. The stored procedure can efficiently bulk-load the CSV (e.g., using SQL's BULK INSERT or equivalent), process the data, and update tables-all within the database's optimized environment. This ensures completion by 8:00 AM and aligns with Appian's recommendation to offload complex, large-scale data operations to the database layer, maintaining Appian as the orchestration layer.
D . Process what can be completed easily in a process model after each integration, and complete the most complex tasks using a set of stored procedures:
This hybrid approach splits the workload: simple tasks (e.g., validation) in a process model, and complex tasks (e.g., transformations) in stored procedures. While this leverages Appian's strengths (orchestration) and database efficiency, it adds unnecessary complexity. Managing two layers of processing increases maintenance overhead and risks partial failures (e.g., process

model timeouts before stored procedures run). Appian's best practices favor a single, cohesive approach for bulk data integration, making this less efficient than a pure stored procedure solution (C).
Conclusion: Creating a set of stored procedures (C) is the best option. It leverages the database's native capabilities to handle the high volume and complexity of the CSV integration, ensuring fast, reliable processing between 1:00 AM and 8:00 AM. Appian orchestrates the trigger and integration (e.g., via a process model), while the stored procedure performs the heavy lifting-aligning with Appian's performance guidelines for large-scale data operations.
Appian Documentation: "Execute Stored Procedure Smart Service" (Process Modeling > Smart Services).
Appian Lead Developer Certification: Data Integration Module (Handling Large Data Volumes).
Appian Best Practices: "Performance Considerations for Data Integration" (Database vs. Process Model Processing).

NEW QUESTION # 19
.....

It is convenient for our consumers to check Appian ACD-301 exam questions free of charge before purchasing the Appian ACD-301 practice exam. Appian is an excellent platform where you get relevant, credible, and unique Appian ACD-301 Exam Dumps designed according to the specified pattern, material, and format as suggested by the Appian ACD-301 exam.

ACD-301 Trustworthy Pdf: https://www.braindumpspass.com/Appian/ACD-301-practice-exam-dumps.html

Our ACD-301 exam questions are definitely the leader in this industry, Get familiar about the exam questions and exam structure by trying the free sample questions of the ACD-301 exam PDF, Exam Description: It is well known that ACD-301 exam test is the hot exam of Appian Appian Certification Program ACD-301 (Appian Certified Lead Developer), Appian ACD-301 Valid Braindumps Ppt Comprehensive Testing Engine.

Drawing from our own lessons in adopting agile practices at Microsoft, Sam Guckenheimer ACD-301 and Neno Loje not only outline the benefits, but also deliver a hands-on, practical guide to implementing those practices in teams of any size.

ACD-301 Valid Braindumps Ppt | Efficient ACD-301: Appian Certified Lead Developer

Creating and Using Chart Templates, Our ACD-301 Exam Questions are definitely the leader in this industry, Get familiar about the exam questions and exam structure by trying the free sample questions of the ACD-301 exam PDF.

Exam Description: It is well known that ACD-301 exam test is the hot exam of Appian Appian Certification Program ACD-301 (Appian Certified Lead Developer), Comprehensive Testing Engine, Keep Following The Appropriate Methods.

- ACD-301 Authorized Exam Dumps □ ACD-301 Study Materials Review □ Flexible ACD-301 Testing Engine □ Open website ▸ www.pass4test.com ◂ and search for □ ACD-301 □ for free download □Reliable ACD-301 Exam Simulations
- Reliable ACD-301 Exam Testking □ Reliable ACD-301 Exam Testking □ Reliable ACD-301 Exam Testking □ Search for □ ACD-301 □ and download it for free immediately on 《 www.pdfvce.com 》 □Exam ACD-301 Testking
- Appian ACD-301 VCE dumps - Testking ACD-301 test □ Search for （ ACD-301 ） and download it for free on ☀ www.dumpsmaterials.com □☀□ website □ACD-301 Reliable Test Practice
- Reliable ACD-301 Exam Simulations □ ACD-301 Authorized Exam Dumps □ Real ACD-301 Torrent □ Download ☀ ACD-301 □☀□ for free by simply searching on ▹ www.pdfvce.com ◃ □Reliable ACD-301 Exam Testking
- ACD-301 Reliable Test Practice □ ACD-301 Valid Vce □ ACD-301 Study Materials Review □ Immediately open { www.examdiscuss.com } and search for ➡ ACD-301 □□□ to obtain a free download ↔ACD-301 Study Materials Review
- Reading The Latest ACD-301 Valid Braindumps Ppt PDF Now □ Search for ➤ ACD-301 □ and obtain a free download on [www.pdfvce.com] □ACD-301 Reliable Test Labs
- Reliable ACD-301 Exam Testking □ Reliable ACD-301 Exam Testking □ ACD-301 Reliable Test Practice □ Easily obtain free download of [ACD-301] by searching on ➠ www.prepawayete.com □ □ACD-301 Study Materials Review
- Only The Best ACD-301 Valid Braindumps Ppt Can Provide Highest Pass Rate of Appian Certified Lead Developer □ Open website ➤ www.pdfvce.com □ and search for ➠ ACD-301 □ for free download □Exam ACD-301 Testking
- Exam ACD-301 Blueprint □ Exam ACD-301 Testking □ ACD-301 Study Materials Review □ Search for [ACD-301] and download exam materials for free through 《 www.vceengine.com 》 □ACD-301 Valid Vce
- Hot ACD-301 Spot Questions □ ACD-301 Dump Torrent □ Reliable ACD-301 Exam Simulations □ Search for 《 ACD-301 》 and easily obtain a free download on ➥ www.pdfvce.com □ □ACD-301 Study Materials Review
- ACD-301 Examcollection Dumps Torrent □ ACD-301 Study Materials Review □ Exam ACD-301 Testking □ Open

{ www.exam4labs.com } and search for ✔ ACD-301 □✔□ to download exam materials for free □Exam ACD-301 Blueprint

- writeablog.net, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, k12.instructure.com, myportal.utt.edu.tt, www.mochome.com, camp-fire.jp, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, www.askmap.net, Disposable vapes

BTW, DOWNLOAD part of BraindumpsPass ACD-301 dumps from Cloud Storage: https://drive.google.com/open?id=1AdlJ5OEeTZOp826dbJrr53bdLq2jolbU