

Appian ACD-301 Realistic Actual Test Pdf Pass Guaranteed Quiz



Our ACD-301 exam guide is suitable for everyone whether you are a business man or a student, because you just need 20-30 hours to practice it that you can attend to your exam. There is no doubt that you can get a great grade. If you follow our learning pace, you will get unexpected surprises. Only when you choose our ACD-301 Guide Torrent will you find it easier to pass this significant ACD-301 examination and have a sense of brand new experience of preparing the ACD-301 exam.

We will offer you the privilege of 365 days free update for ACD-301 latest exam dumps. While, other vendors just give you 90 days free update. As a wise person, it is better to choose our ACD-301 study material without any doubts. Due to the high quality and ACD-301 accurate questions & answers, many people have passed their actual test with the help of our products. Now, quickly download ACD-301 free demo for try. You will get 100% pass with our verified ACD-301 training vce.

>> Actual ACD-301 Test Pdf <<

Appian ACD-301 Simulated Test, ACD-301 Exam Simulator

Nowadays passing the test ACD-301 certification is extremely significant for you and can bring a lot of benefits to you. Passing the test ACD-301 certification does not only prove that you are competent in some area but also can help you enter in the big company and double your wage. Buying our ACD-301 Study Materials can help you pass the test easily and successfully. We provide the study materials which are easy to be mastered, professional expert team and first-rate service to make you get an easy and efficient learning and preparation for the ACD-301 test.

Appian Certified Lead Developer Sample Questions (Q30-Q35):

NEW QUESTION # 30

For each scenario outlined, match the best tool to use to meet expectations. Each tool will be used once Note: To change your responses, you may deselected your response by clicking the blank space at the top of the selection list.

☐

Answer:

Explanation:

☐

NEW QUESTION # 31

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 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.
- 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 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.
- 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.

Answer: A

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 # 32

Your client's customer management application is finally released to Production. After a few weeks of small enhancements and patches, the client is ready to build their next application. The new application will leverage customer information from the first application to allow the client to launch targeted campaigns for select customers in order to increase sales. As part of the first application, your team had built a section to display key customer information such as their name, address, phone number, how long they have been a customer, etc. A similar section will be needed on the campaign record you are building. One of your developers shows you the new object they are working on for the new application and asks you to review it as they are running into a few issues. What feedback should you give?

- A. Provide guidance to the developer on how to address the issues so that they can proceed with their work.
- B. Point the developer to the relevant areas in the documentation or Appian Community where they can find more information on the issues they are running into.
- C. Ask the developer to convert the original customer section into a shared object so it can be used by the new application.
- D. Create a duplicate version of that section designed for the campaign record.

Answer: C

Explanation:

Comprehensive and Detailed In-Depth Explanation:

The scenario involves reusing a customer information section from an existing application in a new application for campaign management, with the developer encountering issues. Appian's best practices emphasize reusability, efficiency, and maintainability, especially when leveraging existing components across applications.

Option B (Ask the developer to convert the original customer section into a shared object so it can be used by the new application): This is the recommended approach. Converting the original section into a shared object (e.g., a reusable interface component) allows it to be accessed across applications without duplication. Appian's Design Guide highlights the use of shared components to promote consistency, reduce redundancy, and simplify maintenance. Since the new application requires similar customer data (name, address, etc.), reusing the existing section—after ensuring it is modular and adaptable—addresses the developer's issues while aligning with the client's goal of leveraging prior work. The developer can then adjust the shared object (e.g., via parameters) to fit the campaign context, resolving their issues collaboratively.

Option A (Provide guidance to the developer on how to address the issues so that they can proceed with their work):

While providing guidance is valuable, it doesn't address the root opportunity to reuse existing code. This option focuses on fixing the new object in isolation, potentially leading to duplicated effort if the original section could be reused instead.

Option C (Point the developer to the relevant areas in the documentation or Appian Community where they can find more information on the issues they are running into):

This is a passive approach and delays resolution. As a Lead Developer, offering direct support or a strategic solution (like reusing components) is more effective than redirecting the developer to external resources without context.

Option D (Create a duplicate version of that section designed for the campaign record):

Duplication violates Appian's principle of DRY (Don't Repeat Yourself) and increases maintenance overhead. Any future updates to customer data display logic would need to be applied to multiple objects, risking inconsistencies.

Given the need to leverage existing customer information and the developer's issues, converting the section to a shared object is the most efficient and scalable solution.

NEW QUESTION # 33

You are required to create an integration from your Appian Cloud instance to an application hosted within a customer's self-managed environment.

The customer's IT team has provided you with a REST API endpoint to test with: <https://internal.network/api/api/ping>.

Which recommendation should you make to progress this integration?

- A. Set up a VPN tunnel.
- B. Add Appian Cloud's IP address ranges to the customer network's allowed IP listing.
- C. Expose the API as a SOAP-based web service.
- D. Deploy the API/service into Appian Cloud.

Answer: A

Explanation:

Comprehensive and Detailed In-Depth Explanation:

As an Appian Lead Developer, integrating an Appian Cloud instance with a customer's self-managed (on-premises) environment requires addressing network connectivity, security, and Appian's cloud architecture constraints. The provided endpoint (<https://internal.network/api/api/ping>) is a REST API on an internal network, inaccessible directly from Appian Cloud due to firewall restrictions and lack of public exposure. Let's evaluate each option:

A . Expose the API as a SOAP-based web service:

Converting the REST API to SOAP isn't a practical recommendation. The customer has provided a REST endpoint, and Appian fully supports REST integrations via Connected Systems and Integration objects. Changing the API to SOAP adds unnecessary complexity, development effort, and risks for the customer, with no benefit to Appian's integration capabilities. Appian's documentation emphasizes using the API's native format (REST here), making this irrelevant.

B . Deploy the API/service into Appian Cloud:

Deploying the customer's API into Appian Cloud is infeasible. Appian Cloud is a managed PaaS environment, not designed to host customer applications or APIs. The API resides in the customer's self-managed environment, and moving it would require significant architectural changes, violating security and operational boundaries. Appian's integration strategy focuses on connecting to external systems, not hosting them, ruling this out.

C . Add Appian Cloud's IP address ranges to the customer network's allowed IP listing:

This approach involves whitelisting Appian Cloud's IP ranges (available in Appian documentation) in the customer's firewall to allow direct HTTP/HTTPS requests. However, Appian Cloud's IPs are dynamic and shared across tenants, making this unreliable for long-term integrations—changes in IP ranges could break connectivity. Appian's best practices discourage relying on IP whitelisting for cloud-to-on-premises integrations due to this limitation, favoring secure tunnels instead.

D . Set up a VPN tunnel:

This is the correct recommendation. A Virtual Private Network (VPN) tunnel establishes a secure, encrypted connection between Appian Cloud and the customer's self-managed network, allowing Appian to access the internal REST API (<https://internal.network/api/api/ping>). Appian supports VPNs for cloud-to-on-premises integrations, and this approach ensures reliability, security, and compliance with network policies. The customer's IT team can configure the VPN, and Appian's documentation recommends this for such scenarios, especially when dealing with internal endpoints.

Conclusion: Setting up a VPN tunnel (D) is the best recommendation. It enables secure, reliable connectivity from Appian Cloud to the customer's internal API, aligning with Appian's integration best practices for cloud-to-on-premises scenarios.

Appian Documentation: "Integrating Appian Cloud with On-Premises Systems" (VPN and Network Configuration).

Appian Lead Developer Certification: Integration Module (Cloud-to-On-Premises Connectivity).

Appian Best Practices: "Securing Integrations with Legacy Systems" (VPN Recommendations).

NEW QUESTION # 34

You are developing a case management application to manage support cases for a large set of sites. One of the tabs in this application's site is a record grid of cases, along with information about the site corresponding to that case. Users must be able to filter cases by priority level and status.

You decide to create a view as the source of your entity-backed record, which joins the separate case/site tables (as depicted in the following image).

Which three columns should be indexed?

- A. modified_date
- B. case_id
- C. name
- D. status
- E. priority
- F. site_id

Answer: D,E,F

Explanation:

Indexing columns can improve the performance of queries that use those columns in filters, joins, or order by clauses. In this case, the columns that should be indexed are site_id, status, and priority, because they are used for filtering or joining the tables. Site_id is used to join the case and site tables, so indexing it will speed up the join operation. Status and priority are used to filter the cases by the user's input, so indexing them will reduce the number of rows that need to be scanned. Name, modified_date, and case_id do not need to be indexed, because they are not used for filtering or joining. Name and modified_date are only used for displaying information in the record grid, and case_id is only used as a unique identifier for each record. Verified Appian Records Tutorial, Appian Best Practices As an Appian Lead Developer, optimizing a database view for an entity-backed record grid requires indexing columns frequently used in queries, particularly for filtering and joining. The scenario involves a record grid displaying cases with site information, filtered by "priority level" and "status," and joined via the site_id foreign key. The image shows two tables (site

and case) with a relationship via site_id. Let's evaluate each column based on Appian's performance best practices and query patterns:

A . site_id: This is a primary key in the site table and a foreign key in the case table, used for joining the tables in the view. Indexing site_id in the case table (and ensuring it's indexed in site as a PK) optimizes JOIN operations, reducing query execution time for the record grid. Appian's documentation recommends indexing foreign keys in large datasets to improve query performance, especially for entity-backed records. This is critical for the join and must be included.

B . status: Users filter cases by "status" (a varchar column in the case table). Indexing status speeds up filtering queries (e.g., WHERE status = 'Open') in the record grid, particularly with large datasets. Appian emphasizes indexing columns used in WHERE clauses or filters to enhance performance, making this a key column for optimization. Since status is a common filter, it's essential.

C . name: This is a varchar column in the site table, likely used for display (e.g., site name in the grid). However, the scenario doesn't mention filtering or sorting by name, and it's not part of the join or required filters. Indexing name could improve searches if used, but it's not a priority given the focus on priority and status filters. Appian advises indexing only frequently queried or filtered columns to avoid unnecessary overhead, so this isn't necessary here.

D . modified_date: This is a date column in the case table, tracking when cases were last updated. While useful for sorting or historical queries, the scenario doesn't specify filtering or sorting by modified_date in the record grid. Indexing it could help if used, but it's not critical for the current requirements. Appian's performance guidelines prioritize indexing columns in active filters, making this lower priority than site_id, status, and priority.

E . priority: Users filter cases by "priority level" (a varchar column in the case table). Indexing priority optimizes filtering queries (e.g., WHERE priority = 'High') in the record grid, similar to status. Appian's documentation highlights indexing columns used in WHERE clauses for entity-backed records, especially with large datasets. Since priority is a specified filter, it's essential to include.

F . case_id: This is the primary key in the case table, already indexed by default (as PKs are automatically indexed in most databases). Indexing it again is redundant and unnecessary, as Appian's Data Store configuration relies on PKs for unique identification but doesn't require additional indexing for performance in this context. The focus is on join and filter columns, not the PK itself.

Conclusion: The three columns to index are A (site_id), B (status), and E (priority). These optimize the JOIN (site_id) and filter performance (status, priority) for the record grid, aligning with Appian's recommendations for entity-backed records and large datasets. Indexing these columns ensures efficient querying for user filters, critical for the application's performance.

Appian Documentation: "Performance Best Practices for Data Stores" (Indexing Strategies).

Appian Lead Developer Certification: Data Management Module (Optimizing Entity-Backed Records).

Appian Best Practices: "Working with Large Data Volumes" (Indexing for Query Performance).

NEW QUESTION # 35

.....

Getcertkey has built customizable Appian ACD-301 practice exams (desktop software & web-based) for our customers. Users can customize the time and ACD-301 questions of Appian ACD-301 Practice Tests according to their needs. You can give more than one test and track the progress of your previous attempts to improve your marks on the next try.

ACD-301 Simulated Test: https://www.getcertkey.com/ACD-301_braindumps.html

Appian Actual ACD-301 Test Pdf It is definitely a meaningful investment for you and you cannot miss this opportunity to being outstanding. For that purpose, ACD-301 exam dumps contains specially created real exam like practice questions and answers, Appian Actual ACD-301 Test Pdf Big companies are always on the lookout for capable candidates, We are legal authorized company devoting to researching and selling professional ACD-301 exam dumps many years.

It's ideal for students in quality programs, Intrusion Detection with Snort Intrusion ACD-301 Detection with Snort, It is definitely a meaningful investment for you and you cannot miss this opportunity to being outstanding.

Valid Appian ACD-301 exam pdf & ACD-301 practice exam & ACD-301 braindumps2go dumps

For that purpose, ACD-301 exam dumps contains specially created real exam like practice questions and answers, Big companies are always on the lookout for capable candidates.

We are legal authorized company devoting to researching and selling professional ACD-301 exam dumps many years, So take action!

- Latest ACD-301 Test Format * Reliable ACD-301 Exam Camp ☐ Certification ACD-301 Dump ☐ Search for ☐ ACD-301 ☐ and download it for free immediately on > www.dumpsquestion.com < ☐ Valid ACD-301 Test Simulator
- Efficient Actual ACD-301 Test Pdf - Easy and Guaranteed ACD-301 Exam Success ☐ Download ➡ ACD-301 ☐☐☐ for

[illegible]