

Free PDF Appian - Fantastic Test ACD301 Book



BTW, DOWNLOAD part of ExamCost ACD301 dumps from Cloud Storage: <https://drive.google.com/open?id=1wtiM91HLrSXjKF6oN-9yIpEXBhZ559f>

The Appian Lead Developer certification has become very popular to survive in today's difficult job market in the technology industry. Every year, hundreds of Appian aspirants attempt the ACD301 exam since passing it results in well-paying jobs, salary hikes, skills validation, and promotions. Lack of Real ACD301 Exam Questions is their main obstacle during ACD301 certification test preparation.

Appian ACD301 Exam Syllabus Topics:

Topic	Details
Topic 1	<ul style="list-style-type: none">Project and Resource Management: This section of the exam measures skills of Agile Project Leads and covers interpreting business requirements, recommending design options, and leading Agile teams through technical delivery. It also involves governance, and process standardization.
Topic 2	<ul style="list-style-type: none">Extending Appian: This section of the exam measures skills of Integration Specialists and covers building and troubleshooting advanced integrations using connected systems and APIs. Candidates are expected to work with authentication, evaluate plug-ins, develop custom solutions when needed, and utilize document generation options to extend the platform's capabilities.
Topic 3	<ul style="list-style-type: none">Application Design and Development: This section of the exam measures skills of Lead Appian Developers and covers the design and development of applications that meet user needs using Appian functionality. It includes designing for consistency, reusability, and collaboration across teams. Emphasis is placed on applying best practices for building multiple, scalable applications in complex environments.

>> Test ACD301 Book <<

Free ACD301 pdf torrent & Appian ACD301 exam answers & ACD301 vce dumps

After taking a bird's eye view of applicants' issues, ExamCost has decided to provide them with the real ACD301 Questions. These

ACD301 dumps pdf is according to the new and updated syllabus so they can prepare for ACD301 certification anywhere, anytime, with ease. A team of professionals has made the product of ExamCost after much hard work with their complete potential so the candidates can prepare for Appian Lead Developer (ACD301) practice test in a short time.

Appian Lead Developer Sample Questions (Q21-Q26):

NEW QUESTION # 21

Your Appian project just went live with the following environment setup: DEV > TEST (SIT/UAT) > PROD. Your client is considering adding a support team to manage production defects and minor enhancements, while the original development team focuses on Phase 2. Your client is asking you for a new environment strategy that will have the least impact on Phase 2 development work. Which option involves the lowest additional server cost and the least code retrofit effort?

- **A. Phase 2 development work stream: DEV > TEST (SIT/UAT) > PROD
Production support work stream: DEV > TEST2 (SIT/UAT) > PROD**
- B. Phase 2 development work stream: DEV > TEST (SIT) > STAGE (UAT) > PROD Production support work stream: DEV2 > STAGE (SIT/UAT) > PROD
- C. Phase 2 development work stream: DEV > TEST (SIT/UAT) > PROD Production support work stream: DEV2 > TEST (SIT/UAT) > PROD
- D. Phase 2 development work stream: DEV > TEST (SIT) > STAGE (UAT) > PROD Production support work stream: DEV > TEST2 (SIT/UAT) > PROD

Answer: A

Explanation:

Comprehensive and Detailed In-Depth Explanation:

The goal is to design an environment strategy that minimizes additional server costs and code retrofit effort while allowing the support team to manage production defects and minor enhancements without disrupting the Phase 2 development team. The current setup (DEV > TEST (SIT/UAT) > PROD) uses a single development and testing pipeline, and the client wants to segregate support activities from Phase 2 development. Appian's Environment Management Best Practices emphasize scalability, cost efficiency, and minimal refactoring when adjusting environments.

Option C (Phase 2 development work stream: DEV > TEST (SIT/UAT) > PROD; Production support work stream: DEV > TEST2 (SIT/UAT) > PROD):

This option is the most cost-effective and requires the least code retrofit effort. It leverages the existing DEV environment for both teams but introduces a separate TEST2 environment for the support team's SIT/UAT activities. Since DEV is already shared, no new development server is needed, minimizing server costs. The existing code in DEV and TEST can be reused for TEST2 by exporting and importing packages, with minimal adjustments (e.g., updating environment-specific configurations). The Phase 2 team continues using the original TEST environment, avoiding disruption. Appian supports multiple test environments branching from a single DEV, and the PROD environment remains shared, aligning with the client's goal of low impact on Phase 2. The support team can handle defects and enhancements in TEST2 without interfering with development workflows.

Option A (Phase 2 development work stream: DEV > TEST (SIT) > STAGE (UAT) > PROD; Production support work stream: DEV > TEST2 (SIT/UAT) > PROD):

This introduces a STAGE environment for UAT in the Phase 2 stream, adding complexity and potentially requiring code updates to accommodate the new environment (e.g., adjusting deployment scripts). It also requires a new TEST2 server, increasing costs compared to Option C, where TEST2 reuses existing infrastructure.

Option B (Phase 2 development work stream: DEV > TEST (SIT) > STAGE (UAT) > PROD; Production support work stream: DEV2 > STAGE (SIT/UAT) > PROD):

This option adds both a DEV2 server for the support team and a STAGE environment, significantly increasing server costs. It also requires refactoring code to support two development environments (DEV and DEV2), including duplicating or synchronizing objects, which is more effort than reusing a single DEV.

Option D (Phase 2 development work stream: DEV > TEST (SIT/UAT) > PROD; Production support work stream: DEV2 > TEST (SIT/UAT) > PROD):

This introduces a DEV2 server for the support team, adding server costs. Sharing the TEST environment between teams could lead to conflicts (e.g., overwriting test data), potentially disrupting Phase 2 development. Code retrofit effort is higher due to managing two DEV environments and ensuring TEST compatibility.

Cost and Retrofit Analysis:

Server Cost: Option C avoids new DEV or STAGE servers, using only an additional TEST2, which can often be provisioned on existing hardware or cloud resources with minimal cost. Options A, B, and D require additional servers (TEST2, DEV2, or STAGE), increasing expenses.

Code Retrofit: Option C minimizes changes by reusing DEV and PROD, with TEST2 as a simple extension. Options A and B require updates for STAGE, and B and D involve managing multiple DEV environments, necessitating more significant refactoring. Appian's recommendation for environment strategies in such scenarios is to maximize reuse of existing infrastructure and avoid

unnecessary environment proliferation, making Option C the optimal choice.

NEW QUESTION # 22

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 minimally 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 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.
- C. 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.
- D. 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.

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.

References:

- * 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 # 23

As part of an upcoming release of an application, a new nullable field is added to a table that contains customer data. 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. Create a script that adds the field and leaves it null.
- C. Create a rollback script that removes the field.
- D. Add a view that joins the customer data to the data used in calculation.
- E. Create a script that adds the field and then populates it.

Answer: C,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.

References:

- * 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 # 24

You are reviewing the Engine Performance Logs in Production for a single application that has been live for six months. This application experiences concurrent user activity and has a fairly sustained load during business hours. The client has reported performance issues with the application during business hours.

During your investigation, you notice a high Work Queue - Java Work Queue Size value in the logs. You also notice unattended process activities, including timer events and sending notification emails, are taking far longer to execute than normal. The client increased the number of CPU cores prior to the application going live. What is the next recommendation?

- **A. Add more engine replicas.**
- B. Add execution and analytics shards
- C. Optimize slow-performing user interfaces.
- D. Add more application servers.

Answer: A

Explanation:

As an Appian Lead Developer, analyzing Engine Performance Logs to address performance issues in a Production application requires understanding Appian's architecture and the specific metrics described. The scenario indicates a high "Work Queue - Java Work Queue Size," which reflects a backlog of tasks in the Java Work Queue (managed by Appian engines), and delays in unattended process activities (e.g., timer events, email notifications). These symptoms suggest the Appian engines are overloaded, despite the client increasing CPU cores. Let's evaluate each option:

* A. Add more engine replicas: This is the correct recommendation. In Appian, engine replicas (part of the Appian Engine cluster) handle process execution, including unattended tasks like timers and notifications. A high Java Work Queue Size indicates the engines are overwhelmed by concurrent activity during business hours, causing delays. Adding more engine replicas distributes the workload, reducing queue size and improving performance for both user-driven and unattended tasks. Appian's documentation recommends scaling engine replicas to handle sustained loads, especially in Production with high concurrency. Since CPU cores were already increased (likely on application servers), the bottleneck is likely the engine capacity, not the servers.

* B. Optimize slow-performing user interfaces: While optimizing user interfaces (e.g., SAIL forms, reports) can improve user experience, the scenario highlights delays in unattended activities (timers, emails), not UI performance. The Java Work Queue Size issue points to engine-level processing, not UI rendering, so this doesn't address the root cause. Appian's performance tuning guidelines prioritize engine scaling for queue-related issues, making this a secondary concern.

* C. Add more application servers: Application servers handle web traffic (e.g., SAIL interfaces, API calls), not process execution or unattended tasks managed by engines. Increasing application servers would help with UI concurrency but wouldn't reduce the Java Work Queue Size or speed up timer /email processing, as these are engine responsibilities. Since the client already increased CPU cores (likely on application servers), this is redundant and unrelated to the issue.

* D. Add execution and analytics shards: Execution shards (for process data) and analytics shards (for reporting) are part of Appian's data fabric for scalability, but they don't directly address engine workload or Java Work Queue Size. Shards optimize data storage and query performance, not real-time process execution. The logs indicate an engine bottleneck, not a data storage issue, so this isn't relevant.

Appian's documentation confirms shards are for long-term scaling, not immediate performance fixes.

Conclusion: Adding more engine replicas (A) is the next recommendation. It directly resolves the high Java Work Queue Size and delays in unattended tasks, aligning with Appian's architecture for handling concurrent loads in Production. This requires collaboration with system administrators to configure additional replicas in the Appian cluster.

References:

- * Appian Documentation: "Engine Performance Monitoring" (Java Work Queue and Scaling Replicas).
- * Appian Lead Developer Certification: Performance Optimization Module (Engine Scaling Strategies).
- * Appian Best Practices: "Managing Production Performance" (Work Queue Analysis).

NEW QUESTION # 25

You are reviewing log files that can be accessed in Appian to monitor and troubleshoot platform-based issues.

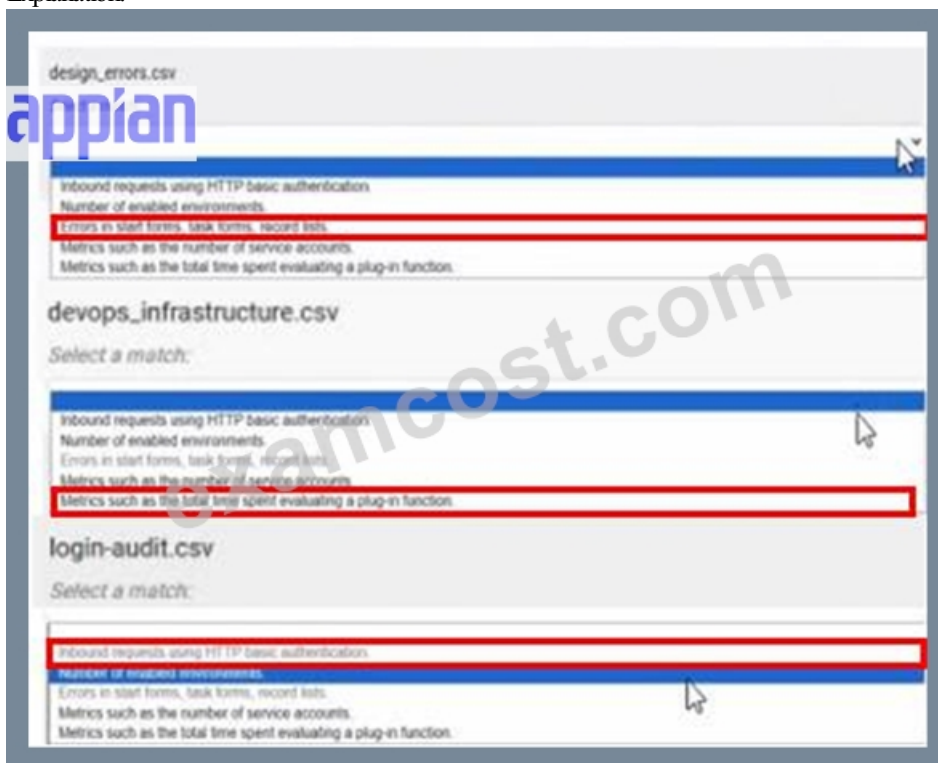
For each type of log file, match the corresponding Information that it provides. Each description will either be used once, or not at all.

Note: To change your responses, you may deselect your response by clicking the blank space at the top of the selection list.



Answer:

Explanation:



NEW QUESTION # 26

.....

Students often feel helpless when purchasing test materials, because most of the test materials cannot be read in advance, students often buy some products that sell well but are actually not suitable for them. But if you choose ACD301 practice test, you will certainly not encounter similar problems. All the materials in ACD301 Exam Torrent can be learned online or offline. You can use your mobile phone, computer or print it out for review. With ACD301 practice test, if you are an office worker, you can study on commute to work, while waiting for customers, and for short breaks after work.

Excellect ACD301 Pass Rate: <https://www.examcost.com/ACD301-practice-exam.html>

- ACD301 Valid Real Exam ☐ Real ACD301 Torrent ☐ Exam ACD301 Papers ☐ Immediately open ☐ www.examcollectionpass.com ☐ and search for ☀ ACD301 ☐ ☀ ☐ to obtain a free download ☐ ACD301 Dumps PDF
- New ACD301 Test Book ☐ Test ACD301 Preparation ☐ ACD301 Study Test ☐ Search for ➡ ACD301 ☐ and download exam materials for free through ☐ www.pdfvce.com ☐ ☐ ACD301 Reliable Test Cost
- TRY Appian ACD301 DUMPS - SUCCESSFUL PLAN TO PASS THE EXAM ☐ Open ☐ www.practicevce.com ☐ enter ☐ ACD301 ☐ and obtain a free download ☐ ACD301 Free Sample

- P.S. Free 2025 Appian ACD301 dumps are available on Google Drive shared by ExamCost: <https://drive.google.com/open?id=1wtTiM91HLrSXjKF6oN-9yIpEXBhZ559f>