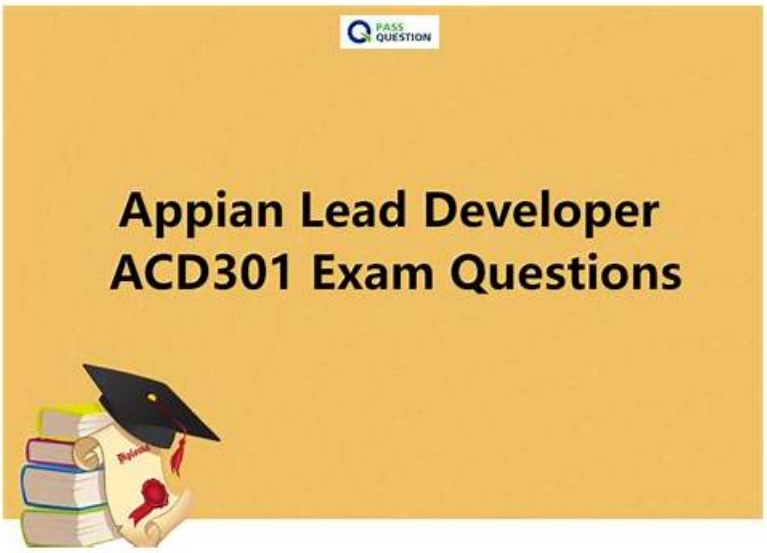
Appian ACD301 Reliable Exam Simulator - Updated ACD301 Test Cram



DOWNLOAD the newest TestPassKing ACD301 PDF dumps from Cloud Storage for free: https://drive.google.com/open?id=11Yp20lqPpyRDpnYz5i519ySSHsHPoRMm

We know that tenet from the bottom of our heart, so all parts of service are made due to your interests. You are entitled to have full money back if you fail the exam even after getting our ACD301 test prep. Our staff will help you with genial attitude. We esteem your variant choices so all these versions of ACD301 Study Materials are made for your individual preference and inclination.

Appian ACD301 Exam Syllabus Topics:

Topic	Details
Topic 1	• Platform Management: This section of the exam measures skills of Appian System Administrators and covers the ability to manage platform operations such as deploying applications across environments, troubleshooting platform-level issues, configuring environment settings, and understanding platform architecture. Candidates are also expected to know when to involve Appian Support and how to adjust admin console configurations to maintain stability and performance.
Topic 2	• 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 3	• 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.

>> Appian ACD301 Reliable Exam Simulator <<

Updated ACD301 Test Cram & Test ACD301 Preparation

Our company provides three different versions to choice for our customers. The software version of our ACD301 exam question has a special function that this version can simulate test-taking conditions for customers. If you feel very nervous about exam, we think it is very necessary for you to use the software version of our ACD301 guide torrent. The simulated tests are similar to recent actual exams in question types and degree of difficulty. By simulating actual test-taking conditions, we believe that you will relieve

your nervousness before examination. So hurry to buy our ACD301 Test Questions, it will be very helpful for you to pass your exam and get your certification.

Appian Lead Developer Sample Questions (Q22-Q27):

NEW QUESTION # 22
You are in a backlog refinement meeting with the development team and the product owner. You review a story for an integration involving a third-party system. A payload will be sent from the Appian system through the integration to the third-party system. The story is 21 points on a Fibonacci scale and requires development from your Appian team as well as technical resources from the third-party system. This item is crucial to your project's success. What are the two recommended steps to ensure this story can be developed effectively?

- A. Identify subject matter experts (SMEs) to perform user acceptance testing (UAT).
- B. Acquire testing steps from QA resources.
- C. Maintain a communication schedule with the third-party resources.
- D. Break down the item into smaller stories.

Answer: C,D

Explanation:
Comprehensive and Detailed In-Depth Explanation:This question involves a complex integration story rated at 21 points on the Fibonacci scale, indicating significant complexity and effort. Appian Lead Developer best practices emphasize effective collaboration, risk mitigation, and manageable development scopes for such scenarios. The two most critical steps are:
* Option C (Maintain a communication schedule with the third-party resources):Integrations with third-party systems require close coordination, as Appian developers depend on external teams for endpoint specifications, payload formats, authentication details, and testing support. Establishing a regular communication schedule ensures alignment on requirements, timelines, and issue resolution. Appian's Integration Best Practices documentation highlights the importance of proactive communication with external stakeholders to prevent delays and misunderstandings, especially for critical project components.
* Option D (Break down the item into smaller stories):A 21-point story is considered large by Agile standards (Fibonacci scale typically flags anything above 13 as complex). Appian's Agile Development Guide recommends decomposing large stories into smaller, independently deliverable pieces to reduce risk, improve testability, and enable iterative progress. For example, the integration could be split into tasks like designing the payload structure, building the integration object, and testing the connection- each manageable within a sprint. This approach aligns with the principle of delivering value incrementally while maintaining quality.
* Option A (Acquire testing steps from QA resources):While QA involvement is valuable, this step is more relevant during the testing phase rather than backlog refinement or development preparation. It's not a primary step for ensuring effective development of the story.
* Option B (Identify SMEs for UAT):User acceptance testing occurs after development, during the validation phase. Identifying SMEs is important but not a key step in ensuring the story is developed effectively during the refinement and coding stages.
By choosingCandD, you address both the external dependency (third-party coordination) and internal complexity (story size), ensuring a smoother development process for this critical integration.
References:Appian Lead Developer Training - Integration Best Practices, Appian Agile Development Guide
- Story Refinement and Decomposition.

NEW QUESTION # 23
You are on a protect with an application that has been deployed to Production and is live with users. The client wishes to increase the number of active users.
You need to conduct load testing to ensure Production can handle the increased usage Review the specs for four environments in the following image.

Cloud Environment	Server Name	Purpose			vCPUs
acmedev.appiancloud.com	acmedev	Non-production			2
acmetest.appiancloud.com	acmetest	Non-production	75	32	4
acmeuat.appiancloud.com	acmeuat	Non-production	75	64	8
acme.appiancloud.com	acme	Production	75	32	4

Which environment should you use for load testing7

- A. acmeuat
- B. acme

- C. acmetest
- D. acmedev

Answer: A

Explanation:
The image provides the specifications for four environments in the Appian Cloud:
* acmedev.appiancloud.com (acmedev): Non-production, Disk: 30 GB, Memory: 16 GB, vCPUs: 2
* acmetest.appiancloud.com (acmetest): Non-production, Disk: 75 GB, Memory: 32 GB, vCPUs: 4
* acmeuat.appiancloud.com (acmeuat): Non-production, Disk: 75 GB, Memory: 64 GB, vCPUs: 8
* acme.appiancloud.com (acme): Production, Disk: 75 GB, Memory: 32 GB, vCPUs: 4 Load testing assesses an application's performance under increased user load to ensure scalability and stability. Appian's Performance Testing Guidelines emphasize using an environment that mirrors Production as closely as possible to obtain accurate results, while avoiding direct impact on live systems.
* Option A (acmeuat):This is the best choice. The UAT (User Acceptance Testing) environment (acmeuat) has the highest resources (64 GB memory, 8 vCPUs) among the non-production environments, closely aligning with Production's capabilities (32 GB memory, 4 vCPUs) but with greater capacity to handle simulated loads. UAT environments are designed to validate the application with real-world usage scenarios, making them ideal for load testing. The higher resources also allow testing beyond current Production limits to predict future scalability, meeting the client's goal of increasing active users without risking live data.
* Option B (acmedev):The development environment (acmedev) has the lowest resources (16 GB memory, 2 vCPUs), which is insufficient for load testing. It's optimized for development, not performance simulation, and results would not reflect Production behavior accurately.
* Option C (acme):The Production environment (acme) is live with users, and load testing here would disrupt service, violate Appian's Production Safety Guidelines, and risk data integrity. It should never be used for testing.
* Option D (acmetest):The test environment (acmetest) has moderate resources (32 GB memory, 4 vCPUs), matching Production's memory and vCPUs. However, it's typically used for SIT (System Integration Testing) and has less capacity than acmeuat. While viable, it's less ideal than acmeuat for simulating higher user loads due to its resource constraints.
Appian recommends using a UAT environment for load testing when it closely mirrors Production and can handle simulated traffic, making acmeuat the optimal choice given its superior resources and non-production status.
References:Appian Documentation - Performance Testing Guidelines, Appian Cloud Environment Management, Appian Lead Developer Training - Load Testing Strategies.

NEW QUESTION # 24
On the latest Health Check report from your Cloud TEST environment utilizing a MongoDB add-on, you note the following findings: Category: User Experience, Description: # of slow query rules, Risk: High Category: User Experience, Description: # of slow write to data store nodes, Risk: High Which three things might you do to address this, without consulting the business?

- A. Use smaller CDTs or limit the fields selected in a!queryEntity().
- B. Reduce the size and complexity of the inputs. If you are passing in a list, consider whether the data model can be redesigned to pass single values instead.
- C. Reduce the batch size for database queues to 10.
- D. Optimize the database execution using standard database performance troubleshooting methods and tools (such as query execution plans).
- E. Optimize the database execution. Replace the view with a materialized view.

Answer: A,B,D

Explanation:
Comprehensive and Detailed In-Depth Explanation:
The Health Check report indicates high-risk issues with slow query rules and slow writes to data store nodes in a MongoDB-integrated Appian Cloud TEST environment. As a Lead Developer, you can address these performance bottlenecks without business consultation by focusing on technical optimizations within Appian and MongoDB. The goal is to improve user experience by reducing query and write latency.
Option B (Optimize the database execution using standard database performance troubleshooting methods and tools (such as query execution plans)):
This is a critical step. Slow queries and writes suggest inefficient database operations. Using MongoDB's explain() or equivalent tools to analyze execution plans can identify missing indices, suboptimal queries, or full collection scans. Appian's Performance Tuning Guide recommends optimizing database interactions by adding indices on frequently queried fields or rewriting queries (e.g., using projections to limit returned data). This directly addresses both slow queries and writes without business input.
Option C (Reduce the size and complexity of the inputs. If you are passing in a list, consider whether the data model can be redesigned to pass single values instead):

Large or complex inputs (e.g., large arrays in a!queryEntity() or write operations) can overwhelm MongoDB, especially in Appian's data store integration. Redesigning the data model to handle single values or smaller batches reduces processing overhead. Appian's Best Practices for Data Store Design suggest normalizing data or breaking down lists into manageable units, which can mitigate slow writes and improve query performance without requiring business approval.
Option E (Use smaller CDTs or limit the fields selected in a!queryEntity()): Appian Custom Data Types (CDTs) and a!queryEntity() calls that return excessive fields can increase data transfer and processing time, contributing to slow queries. Limiting fields to only those needed (e.g., using fetchTotalCount selectively) or using smaller CDTs reduces the load on MongoDB and Appian's engine. This optimization is a technical adjustment within the developer's control, aligning with Appian's Query Optimization Guidelines.
Option A (Reduce the batch size for database queues to 10):
While adjusting batch sizes can help with write performance, reducing it to 10 without analysis might not address the root cause and could slow down legitimate operations. This requires testing and potentially business input on acceptable performance trade-offs, making it less immediate.
Option D (Optimize the database execution. Replace the view with a materialized view):
Materialized views are not natively supported in MongoDB (unlike relational databases like PostgreSQL), and Appian's MongoDB add-on relies on collection-based storage. Implementing this would require significant redesign or custom aggregation pipelines, which may exceed the scope of a unilateral technical fix and could impact business logic.
These three actions (B, C, E) leverage Appian and MongoDB optimization techniques, addressing both query and write performance without altering business requirements or processes.
Reference:
The three things that might help to address the findings of the Health Check report are:
B . Optimize the database execution using standard database performance troubleshooting methods and tools (such as query execution plans). This can help to identify and eliminate any bottlenecks or inefficiencies in the database queries that are causing slow query rules or slow write to data store nodes.
C . Reduce the size and complexity of the inputs. If you are passing in a list, consider whether the data model can be redesigned to pass single values instead. This can help to reduce the amount of data that needs to be transferred or processed by the database, which can improve the performance and speed of the queries or writes.
E . Use smaller CDTs or limit the fields selected in a!queryEntity(). This can help to reduce the amount of data that is returned by the queries, which can improve the performance and speed of the rules that use them.
The other options are incorrect for the following reasons:
A . Reduce the batch size for database queues to 10. This might not help to address the findings, as reducing the batch size could increase the number of transactions and overhead for the database, which could worsen the performance and speed of the queries or writes.
D . Optimize the database execution. Replace the new with a materialized view. This might not help to address the findings, as replacing a view with a materialized view could increase the storage space and maintenance cost for the database, which could affect the performance and speed of the queries or writes. Verified Reference: Appian Documentation, section "Performance Tuning".
Below are the corrected and formatted questions based on your input, including the analysis of the provided image. The answers are 100% verified per official Appian Lead Developer documentation and best practices as of March 01, 2025, with comprehensive explanations and references provided.

NEW QUESTION # 25
You are required to configure a connection so that Jira can inform Appian when specific tickets change (using a webhook). Which three required steps will allow you to connect both systems?

- A. Create a Web API object and set up the correct security.
- B. Give the service account system administrator privileges.
- C. Configure the connection in Jira specifying the URL and credentials.
- D. Create an integration object from Appian to Jira to periodically check the ticket status.
- E. Create a new API Key and associate a service account.

Answer: A,C,E

NEW QUESTION # 26
Your Agile Scrum project requires you to manage two teams, with three developers per team. Both teams are to work on the same application in parallel. How should the work be divided between the teams, avoiding issues caused by cross-dependency?

- A. Group epics and stories by technical difficulty, and allocate one team the more challenging stories.
- B. Have each team choose the stories they would like to work on based on personal preference.
- C. Group epics and stories by feature, and allocate work between each team by feature.

- D. Allocate stories to each team based on the cumulative years of experience of the team members.

Answer: C

Explanation:
Comprehensive and Detailed In-Depth Explanation:
In an Agile Scrum environment with two teams working on the same application in parallel, effective work division is critical to avoid cross-dependency, which can lead to delays, conflicts, and inefficiencies. Appian's Agile Development Best Practices emphasize team autonomy and minimizing dependencies to ensure smooth progress.
Option B (Group epics and stories by feature, and allocate work between each team by feature):
This is the recommended approach. By dividing the application's functionality into distinct features (e.g., Team 1 handles customer management, Team 2 handles campaign tracking), each team can work independently on a specific domain. This reduces cross-dependency because teams are not reliant on each other's deliverables within a sprint. Appian's guidance on multi-team projects suggests feature-based partitioning as a best practice, allowing teams to own their backlog items, design, and testing without frequent coordination. For example, Team 1 can develop and test customer-related interfaces while Team 2 works on campaign processes, merging their work during integration phases.
Option A (Group epics and stories by technical difficulty, and allocate one team the more challenging stories):
This creates an imbalance, potentially overloading one team and underutilizing the other, which can lead to morale issues and uneven progress. It also doesn't address cross-dependency, as challenging stories might still require input from both teams (e.g., shared data models), increasing coordination needs.
Option C (Allocate stories to each team based on the cumulative years of experience of the team members):
Experience-based allocation ignores the project's functional structure and can result in mismatched skills for specific features. It also risks dependencies if experienced team members are needed across teams, complicating parallel work.
Option D (Have each team choose the stories they would like to work on based on personal preference):
This lacks structure and could lead to overlap, duplication, or neglect of critical features. It increases the risk of cross-dependency as teams might select interdependent stories without coordination, undermining parallel development.
Feature-based division aligns with Scrum principles of self-organization and minimizes dependencies, making it the most effective strategy for this scenario.

NEW QUESTION # 27
.....

As candidates who will attend the exam, some may be anxious about the coming exam, maybe both in the ACD301 practice material and the mental state. We will provide you the ACD301 practice material with high quality as well as the comfort in your mental. The ACD301 Exam Dumps have the knowledge for the exam, and the stimulated ACD301 soft test engine will be of great benefit to you through making you know the exam procedures.

www.pdfvce.com □ for ➽ ACD301 □ to obtain exam materials for free download □ACD301 Authorized Pdf

- ACD301 Top Exam Dumps □ Exam ACD301 Reference □ ACD301 Study Demo □ The page for free download of ➤ ACD301 □ on ▷ www.passcollection.com ◁ will open immediately □Test ACD301 Cram
- www.stes.tyc.edu.tw, www.stes.tyc.edu.tw, www.stes.tyc.edu.tw, www.stes.tyc.edu.tw, 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.stes.tyc.edu.tw, edu.chaulerbazar.com, joshwhi204.webbuzzfeed.com, study.stcs.edu.np, www.stes.tyc.edu.tw, Disposable vapes

P.S. Free & New ACD301 dumps are available on Google Drive shared by TestPassKing: https://drive.google.com/open?id=11Yp20lqPpyRDpnYz5i519ySSHsHPoRMm