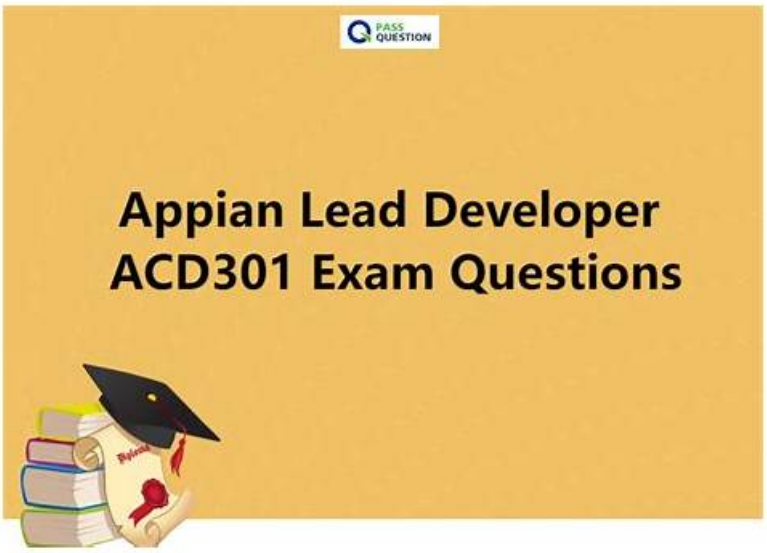
# ACD301 Clear Exam & Customizable ACD301 Exam Mode




What's more, part of that Test4Sure ACD301 dumps now are free: https://drive.google.com/open?id=1hLoerey6JEFb6oT99PQFBG8fDLNnHYFZ

Our ACD301 questions pdf is up to date, and we provide user-friendly ACD301 practice test software for the ACD301 exam. Moreover, we are also providing money back guarantee on all of ACD301 test products. If the ACD301 braindumps products fail to deliver as promised, then you can get your money back. The ACD301 Sample Questions include all the files you need to prepare for the ACD301 exam. With the help of the ACD301 practice exam questions and test software, you will be able to feel the real ACD301 exam scenario, and it will allow you to assess your skills.

## Appian ACD301 Exam Syllabus Topics:

| Topic | Details |
|---|---|
| Topic 1 | • 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 | • 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 | • 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. |
| Topic 4 | • Data Management: This section of the exam measures skills of Data Architects and covers analyzing, designing, and securing data models. Candidates must demonstrate an understanding of how to use Appian's data fabric and manage data migrations. The focus is on ensuring performance in high-volume data environments, solving data-related issues, and implementing advanced database features effectively. |
| Topic 5 | • Proactively Design for Scalability and Performance: This section of the exam measures skills of Application Performance Engineers and covers building scalable applications and optimizing Appian components for performance. It includes planning load testing, diagnosing performance issues at the application level, and designing systems that can grow efficiently without sacrificing reliability. |

**>> ACD301 Clear Exam <<**

# ACD301 Exam Torrent & ACD301 Test Collection & ACD301 Top Quiz

Don't waste your time with unhelpful study methods. There are plenty of options available, but not all of them are suitable to help you pass the Appian Lead Developer (ACD301) exam. Some resources out there may even do more harm than good by leading you astray. Our ACD301 Exam Dumps are available with a free demo and up to 1 year of free updates.

# Appian Lead Developer Sample Questions (Q24-Q29):

**NEW QUESTION # 24**
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 column should be indexed?

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

**Answer: C,D,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 References: 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.
References:
* 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 # 25**
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. Group epics and stories by feature, and allocate work between each team by feature.
- C. Have each team choose the stories they would like to work on based on personal preference.
- D. Allocate stories to each team based on the cumulative years of experience of the team members.

**Answer: B**

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 # 26**
You are the project lead for an Appian project with a supportive product owner and complex business requirements involving a customer management system. Each week, you notice the product owner becoming more irritated and not devoting as much time to the project, resulting in tickets becoming delayed due to a lack of involvement. Which two types of meetings should you schedule to address this issue?

- A. A meeting with the sponsor to discuss the product owner's performance and request a replacement.
- B. A risk management meeting with your program manager to escalate the delayed tickets.
- C. A sprint retrospective with the product owner and development team to discuss team performance.

- D. An additional daily stand-up meeting to ensure you have more of the product owner's time.

**Answer: B,C**

Explanation:
Comprehensive and Detailed In-Depth Explanation:
As an Appian Lead Developer, managing stakeholder engagement and ensuring smooth project progress are critical responsibilities. The scenario describes a product owner whose decreasing involvement is causing delays, which requires a proactive and collaborative approach rather than an immediate escalation to replacement. Let's analyze each option:
A . An additional daily stand-up meeting: While daily stand-ups are a core Agile practice to align the team, adding another one specifically to secure the product owner's time is inefficient. Appian's Agile methodology (aligned with Scrum) emphasizes that stand-ups are for the development team to coordinate, not to force stakeholder availability. The product owner's irritation might increase with additional meetings, making this less effective.
B . A risk management meeting with your program manager: This is a correct choice. Appian Lead Developer documentation highlights the importance of risk management in complex projects (e.g., customer management systems). Delays due to lack of product owner involvement constitute a project risk. Escalating this to the program manager ensures visibility and allows for strategic mitigation, such as resource reallocation or additional support, without directly confronting the product owner in a way that could damage the relationship. This aligns with Appian's project governance best practices.
C . A sprint retrospective with the product owner and development team: This is also a correct choice. The sprint retrospective, as per Appian's Agile guidelines, is a key ceremony to reflect on what's working and what isn't. Including the product owner fosters collaboration and provides a safe space to address their reduced involvement and its impact on ticket delays. It encourages team accountability and aligns with Appian's focus on continuous improvement in Agile development.
D . A meeting with the sponsor to discuss the product owner's performance and request a replacement: This is premature and not recommended as a first step. Appian's Lead Developer training emphasizes maintaining strong stakeholder relationships and resolving issues collaboratively before escalating to drastic measures like replacement. This option risks alienating the product owner and disrupting the project further, which contradicts Appian's stakeholder management principles.
Conclusion: The best approach combines B (risk management meeting) to address the immediate risk of delays with a higher-level escalation and C (sprint retrospective) to collaboratively resolve the product owner's engagement issues. These align with Appian's Agile and leadership strategies for Lead Developers.
Reference:
Appian Lead Developer Certification: Agile Project Management Module (Risk Management and Stakeholder Engagement).
Appian Documentation: "Best Practices for Agile Development in Appian" (Sprint Retrospectives and Team Collaboration).

**NEW QUESTION # 27**
What are two advantages of having High Availability (HA) for Appian Cloud applications?

- A. In the event of a system failure, your Appian instance will be restored and available to your users in less than 15 minutes, having lost no more than the last 1 minute worth of data.
- B. An Appian Cloud HA instance is composed of multiple active nodes running in different availability zones in different regions.
- C. A typical Appian Cloud HA instance is composed of two active nodes.
- D. Data and transactions are continuously replicated across the active nodes to achieve redundancy and avoid single points of failure.

**Answer: A,D**

Explanation:
Comprehensive and Detailed In-Depth Explanation:High Availability (HA) in Appian Cloud is designed to ensure that applications remain operational and data integrity is maintained even in the face of hardware failures, network issues, or other disruptions. Appian's Cloud Architecture and HA documentation outline the benefits, focusing on redundancy, minimal downtime, and data protection. The question asks for two advantages, and the options must align with these core principles.
* Option B (Data and transactions are continuously replicated across the active nodes to achieve redundancy and avoid single points of failure):This is a key advantage of HA. Appian Cloud HA instances use multiple active nodes to replicate data and transactions in real-time across the cluster. This redundancy ensures that if one node fails, others can take over without data loss, eliminating single points of failure. This is a fundamental feature of Appian's HA setup, leveraging distributed architecture to enhance reliability, as detailed in the Appian Cloud High Availability Guide.
* Option D (In the event of a system failure, your Appian instance will be restored and available to your users in less than 15 minutes, having lost no more than the last 1 minute worth of data):This is another significant advantage. Appian Cloud HA is engineered to provide rapid recovery and minimal data loss. The Service Level Agreement (SLA) and HA documentation specify that in the case of a failure, the system failover is designed to complete within a short timeframe (typically under 15 minutes), with

data loss limited to the last minute due to synchronous replication. This ensures business continuity and meets stringent uptime and data integrity requirements.
* Option A (An Appian Cloud HA instance is composed of multiple active nodes running in different availability zones in different regions):This is a description of the HA architecture rather than an advantage. While running nodes across different availability zones and regions enhances fault tolerance, the benefit is the resulting redundancy and availability, which are captured in Options B and D: This option is more about implementation than a direct user or operational advantage.
* Option C (A typical Appian Cloud HA instance is composed of two active nodes):This is a factual statement about the architecture but not an advantage. The number of nodes (typically two or more, depending on configuration) is a design detail, not a benefit. The advantage lies in what this setup enables (e.g., redundancy and quick recovery), as covered by B and D.
The two advantages-continuous replication for redundancy (B) and fast recovery with minimal data loss (D)
-reflect the primary value propositions of Appian Cloud HA, ensuring both operational resilience and data integrity for users.
References:Appian Documentation - Appian Cloud High Availability Guide, Appian Cloud Service Level Agreement (SLA), Appian Lead Developer Training - Cloud Architecture.
The two advantages of having High Availability (HA) for Appian Cloud applications are:
* B. Data and transactions are continuously replicated across the active nodes to achieve redundancy and avoid single points of failure. This is an advantage of having HA, as it ensures that there is always a backup copy of data and transactions in case one of the nodes fails or becomes unavailable. This also improves data integrity and consistency across the nodes, as any changes made to one node are automatically propagated to the other node.
* D. In the event of a system failure, your Appian instance will be restored and available to your users in less than 15 minutes, having lost no more than the last 1 minute worth of data. This is an advantage of having HA, as it guarantees a high level of service availability and reliability for your Appian instance.
If one of the nodes fails or becomes unavailable, the other node will take over and continue to serve requests without any noticeable downtime or data loss for your users.
The other options are incorrect for the following reasons:
* A. An Appian Cloud HA instance is composed of multiple active nodes running in different availability zones in different regions. This is not an advantage of having HA, but rather a description of how HA works in Appian Cloud. An Appian Cloud HA instance consists of two active nodes running in different availability zones within the same region, not different regions.
* C. A typical Appian Cloud HA instance is composed of two active nodes. This is not an advantage of having HA, but rather a description of how HA works in Appian Cloud. A typical Appian Cloud HA instance consists of two active nodes running in different availability zones within the same region, but this does not necessarily provide any benefit over having one active node.
Verified References: Appian Documentation, section "High Availability".

**NEW QUESTION # 28**
You are running an inspection as part of the first deployment process from TEST to PROD. You receive a notice that one of your objects will not deploy because it is dependent on an object from an application owned by a separate team.
What should be your next step?

- A. Halt the production deployment and contact the other team for guidance on promoting the object to PROD.
- B. Check the dependencies of the necessary object. Deploy to PROD if there are few dependencies and it is low risk.
- C. Create your own object with the same code base, replace the dependent object in the application, and deploy to PROD.
- D. Push a functionally viable package to PROD without the dependencies, and plan the rest of the deployment accordingly with the other team's constraints.

**Answer: A**

Explanation:
Comprehensive and Detailed In-Depth Explanation:As an Appian Lead Developer, managing a deployment from TEST to PROD requires careful handling of dependencies, especially when objects from another team's application are involved. The scenario describes a dependency issue during deployment, signaling a need for collaboration and governance. Let's evaluate each option:
* A. Create your own object with the same code base, replace the dependent object in the application, and deploy to PROD:This approach involves duplicating the object, which introduces redundancy, maintenance risks, and potential version control issues. It violates Appian's governance principles, as objects should be owned and managed by their respective teams to ensure consistency and avoid conflicts. Appian's deployment best practices discourage duplicating objects unless absolutely necessary, making this an unsustainable and risky solution.
* B. Halt the production deployment and contact the other team for guidance on promoting the object to PROD:This is the correct step. When an object from another application (owned by a separate team) is a dependency, Appian's deployment process requires coordination to ensure both applications' objects are deployed in sync. Halting the deployment prevents partial deployments that could break functionality, and contacting the other team aligns with Appian's collaboration and governance guidelines. The other team can provide the necessary object version, adjust their deployment timeline, or resolve the dependency, ensuring a stable PROD environment.

* C. Check the dependencies of the necessary object. Deploy to PROD if there are few dependencies and it is low risk:This approach risks deploying an incomplete or unstable application if the dependency isn' t fully resolved. Even with "few dependencies" and "low risk," deploying without the other team's object could lead to runtime errors or broken functionality in PROD. Appian's documentation emphasizes thorough dependency management during deployment, requiring all objects (including those from other applications) to be promoted together, making this risky and not recommended.
* D. Push a functionally viable package to PROD without the dependencies, and plan the rest of the deployment accordingly with the other team's constraints:Deploying without dependencies creates an incomplete solution, potentially leaving the application non-functional or unstable in PROD. Appian's deployment process ensures all dependencies are included to maintain application integrity, and partial deployments are discouraged unless explicitly planned (e.g., phased rollouts). This option delays resolution and increases risk, contradicting Appian's best practices for Production stability.
Conclusion: Halting the production deployment and contacting the other team for guidance (B) is the next step. It ensures proper collaboration, aligns with Appian's governance model, and prevents deployment errors, providing a safe and effective resolution.
References:
* Appian Documentation: "Deployment Best Practices" (Managing Dependencies Across Applications).
* Appian Lead Developer Certification: Application Management Module (Cross-Team Collaboration).
* Appian Best Practices: "Handling Production Deployments" (Dependency Resolution).

**NEW QUESTION # 29**
.....