Up-to-date Questions and Answers from authentic resources to improve knowledge and pass the exam at very first attempt. ----- Guaranteed.

701-100 MCQs
701-100 Exam Questions
701-100 Practice Test
701-100 TestPrep
701-100 Study Guide

killexams.com

LPI

701-100

LPIC-OT Exam 701: DevOps Tools Engineer

ORDER FULL VERSION

https://killexams.com/pass4sure/exam-detail/701-100

Question: 1173

.gitignore negation: Ignore all .txt except README.txt?

A. **/.txt
!README.txt
B. !.txt
!/README.txt
C. .txt
README.txt
D. .txt
README.txt

Answer: D
Explanation: *.txt ignores all, !README.txt exceptions it; must precede for precedence.

Question: 1174

In a serverless e-commerce backend using FaaS, developers deploy a function to resize images uploaded to object storage, triggered by S3 event notifications. Lab metrics show execution duration averaging 300ms, with a memory allocation of 512MB, and billing at $0.20 per 1 million invocations. During peak hours, the function handles 5000 triggers per minute without errors, but cold starts add 1.2 seconds initially. What FaaS feature is highlighted by the event-triggered execution and granular resource allocation in this image processing scenario?

A. Fine-grained billing per invocation and duration
B. Stateless execution with ephemeral environments
C. Automatic concurrency handling without provisioning
D. Event integration with external services like S3

Answer: D
Explanation: Event integration with external services like S3 allows triggers from uploads to invoke the function automatically, enabling seamless workflows like image resizing with metrics like 300ms duration and 512MB memory. This decouples components in serverless setups. Automatic concurrency handling without provisioning scales to 5000 triggers per minute, fine-grained billing per invocation and duration charges $0.20 per million, and stateless execution with ephemeral environments addresses cold starts of 1.2 seconds but not triggers.

Question: 1175

A team is using continuous integration (CI) practices to improve their software development process. They

notice that builds are frequently failing due to integration issues. What strategy should they implement to reduce these failures?

A. Increase the frequency of integration tests
B. Decrease the number of team members involved in development
C. Increase the number of features developed in each sprint
D. Implement feature toggles to control feature releases

Answer: A
Explanation: Increasing the frequency of integration tests can help identify integration issues early in the development process. By testing more often, the team can catch problems before they escalate, leading to more stable builds and a smoother CI process. This proactive approach enables the team to address integration issues promptly.

Question: 1176

Your team uses copyleft licenses like GPL for core DevOps tools. In a scenario, integrating a permissive library, distribution param: build_id=2026. What is permitted?

A. Reject permissive for purity
B. Must make permissive copyleft
C. Separate distributions
D. Include permissive in GPL distribution, as compatible one-way

Answer: D
Explanation: Include permissive in GPL distribution, as compatible one-way from permissive to copyleft. Must make permissive copyleft isn't required. Reject permissive for purity limits options. Separate distributions complicates.

Question: 1177

In a scenario where workloads are containerized and deployed via Kubernetes, a pod in the staging environment fails due to an invalid reference in the service catalog for a persistent volume. The catalog entry specifies a storage class that doesn't exist in staging but does in production. Lab parameters show the volume request at 10Gi with ReadWriteMany access, leading to provisioning errors. To adhere to standard platforms, what complex pipeline enhancement is required?

A. Disable persistent volumes in staging to avoid provisioning issues
B. Manually provision volumes before each deployment
C. Merge staging and production storage classes into a single universal class
D. Add environment-specific overrides in the catalog with validation scripts in the deployment pipeline

Answer: D
Explanation: Adding environment-specific overrides in the catalog with validation scripts in the deployment pipeline allows for tailored configurations while ensuring references are valid per environment, preventing failures. Merging staging and production storage classes into a single universal class may not account for differences in underlying infrastructure. Disabling persistent volumes in staging limits testing realism. Manually provisioning volumes before each deployment is inefficient and prone to human error.

Question: 1178

Merge ff-only fails "not fast-forward". Cause?

A. Remote ahead local
B. Unrelated histories
C. Diverged
D. Local ahead remote

Answer: C
Explanation: ff-only requires local superset; diverged prevents.

Question: 1179

You cloned a repository and made some local changes. However, you realize that you need to update your local repository with the latest changes from the remote origin/main branch before pushing your changes. What is the correct sequence of commands to safely update your local branch?

A. git pull origin main; git push origin feature
B. git fetch origin; git merge origin/main
C. git fetch origin; git rebase origin/main
D. git pull origin main

Answer: C
Explanation: The safest way to update your local branch with the latest changes from the remote origin/main is to first fetch the updates using git fetch origin, and then rebase your local changes onto origin/main with git rebase origin/main. This keeps the commit history cleaner and avoids unnecessary merge commits.

Question: 1180

A DevOps team is monitoring an API and notices that response times are increasing during peak usage

hours. Which of the following strategies could help improve the API's performance under load?

A. Implement rate limiting to restrict the number of requests from clients.
B. Optimize database queries and consider indexing frequently accessed fields.
C. Increase the timeout settings for API requests.
D. Disable caching to ensure that clients always receive the latest data.

Answer: B
Explanation: Optimizing database queries and considering indexing frequently accessed fields can significantly improve the API's performance under load. This approach reduces the time it takes to retrieve data from the database, leading to faster response times and a better user experience.

Question: 1181

You are using a dependency manager to manage your project's libraries. After running a command to update your dependencies, you notice a new library was added that is licensed under a license incompatible with your existing licenses. What should you do next?

A. Contact the library author for clarification on license compatibility.
B. Ignore the new library and continue with the project.
C. Continue using the new library since it was added automatically.
D. Remove the new library and replace it with a compatible alternative.

Answer: D
Explanation: If a new library is added that has an incompatible license, it is crucial to remove it and replace it with a library that has a compatible license. Using incompatible licenses can lead to legal issues and prevent you from distributing your software.

Question: 1182

For a real-time chat app using TDD, test for message delivery in 100ms fails. Code with WebSockets green. Scenario: Network latency 200ms causes 30% drops. Parameters: Retry count 3, backoff 50ms. What TDD extension addresses reliability?

A. Add end-to-end tests with simulated latency using tools like Toxiproxy, ensuring retries in red-green cycle
B. Use synchronous HTTP
C. Test only local loopback
D. Disable retries for simplicity

Answer: A

Explanation: Simulated conditions in tests verify resilience early. Local only misses real, no retries unreliable, HTTP not real-time.

Question: 1183

Your organization is using PostgreSQL for its applications. You notice that a specific query is taking a long time to execute. Which of the following steps would you take first to diagnose the performance issue?

A. Optimize the database schema by normalizing tables
B. Increase the instance size to improve performance
C. Add more indexes to the affected tables
D. Analyze the query execution plan using EXPLAIN

Answer: D
Explanation: Analyzing the query execution plan using EXPLAIN is the first step to diagnose performance issues in PostgreSQL. This allows you to understand how the query is being executed and identify potential bottlenecks or inefficiencies.

Question: 1184

Scenario: A team's Ruby on Rails app via Bundler uses 'rails' (MIT), 'devise' (MIT), transitive 'orm_adapter' (BSD). App GPL v2. Bundle audit clean, but SBOM conflict. Primary issue?

A. Bundler lock ignores BSD for GPL composition
B. BSD permissive compatible with GPL but requires warranty disclaimer in gemspec
C. GPL network clause inapplicable to Ruby gems
D. Rails MIT overrides transitive BSD

Answer: B
Explanation: GPL v2 compatible with BSD inbound, but BSD explicitly disclaims warranties and requires notice retention, which GPL's terms must honor in compositions; Bundler doesn't alter licenses, SBOM exposes for compliance.

Question: 1185

A DevOps tool generates the following JSON configuration snippet to define a server cluster: {"cluster_name": "prod-db", "nodes": 3, "active": true, "tags": [prod, database, secure]}. When an automated parser attempts to read this file, it throws a fatal syntax error. What is the exact reason this JSON is structurally invalid?

A. The array elements prod, database, secure are string literals but are missing the required double quotes.
B. Numeric values like 3 are not natively supported in JSON and must be represented as strings.
C. Boolean values like true must be enclosed in double quotes to be parsed correctly in strict JSON.
D. The use of underscores in key names like cluster_name violates the camelCase requirement of the JSON standard.

Answer: A
Explanation: The array elements prod, database, secure are string literals but are missing the required double quotes. In JSON, all string values, whether keys or values, must be strictly enclosed in double quotes (e.g., ["prod", "database", "secure"]). Single quotes or unquoted text are invalid syntax. Boolean values like true are native JSON types and must NOT be enclosed in quotes. Numeric values are natively supported and do not need quotes. The JSON standard does not enforce any specific naming convention like camelCase; underscores in key names are perfectly valid.

Question: 1186

In a hybrid cloud DevOps setup, your team deploys a MEAN stack application with MongoDB as the database. Lab tests indicate high latency in API responses due to synchronous database calls during peak loads. To optimize using standard platforms, you consider caching layers. The lab parameters show that implementing a cache with TTL of 60 seconds reduces queries by 70%. What is the most suitable standard component for in-memory caching to handle this, ensuring data consistency with invalidation on writes?

A. Apply Varnish for HTTP caching at the web layer
B. Integrate Redis with pub/sub for cache invalidation
C. Configure Memcached with consistent hashing for distribution
D. Use Apache Ignite for grid-based caching

Answer: B
Explanation: Integrating Redis with pub/sub for cache invalidation is the best choice as it supports fast in-memory storage, publish-subscribe messaging for real-time invalidation on database writes, and seamless integration with Node.js in MEAN stacks, ensuring consistency without over-fetching. Applying Varnish for HTTP caching at the web layer is effective for static content but not for dynamic API data. Configuring Memcached with consistent hashing for distribution provides speed but lacks built-in pub/sub for proactive invalidation. Using Apache Ignite for grid-based caching offers advanced features but introduces unnecessary complexity for simple caching needs.

Question: 1187

A company is experiencing issues with service discovery in their microservices architecture. What approach can they take to improve this aspect?

A. Using a single database for all services
B. Increasing the number of instances
C. Hardcoding service endpoints
D. Implementing a service registry

Answer: D
Explanation: Implementing a service registry improves service discovery by allowing services to dynamically register and discover each other, enhancing scalability and flexibility. Hardcoding endpoints and using a single database create tight coupling, while increasing instances does not address the discovery issue.

Question: 1188

In modern software development, a team deploys a containerized app to Kubernetes, but pod evictions occur due to resource overcommitment. Metrics show CPU requests at 500m and limits at 1000m, with bursts to 1500m. What design adjustment complies with best practices?

A. Set CPU requests to 800m and use horizontal pod autoscaling with custom metrics
B. Increase node sizes without request adjustments
C. Use vertical scaling only
D. Disable limits entirely

Answer: A
Explanation: Adjusting requests to 800m and enabling HPA with metrics prevents evictions by scaling based on usage, aligning with container design in modern dev. Disabling limits risks node instability, increasing nodes is costly, vertical scaling limits flexibility.

Question: 1189

You are managing a project that relies on multiple open-source libraries. One library is licensed under the MIT license, while another is under the GPL. If you want to include both libraries in your proprietary application, what must you consider?

A. The GPL library will require you to open-source your application.
B. You can use both libraries without any issues.
C. You must only use the GPL library to comply with licensing.
D. The MIT license is incompatible with the GPL license.

Answer: A
Explanation: The GPL license is more restrictive than the MIT license. If you include a GPL-licensed

library in your proprietary application, you will be required to open-source your application under the same GPL license, which would conflict with your proprietary intentions.

Question: 1190

An IaaS platform runs containerized apps on VMs. Orchestration fails due to resource fragmentation, with lab results showing 40% wasted CPU. Features like bin packing and reservations. What advanced concept to apply?

A. Overprovision CPUs by 20% on all VMs
B. Use smaller VMs without reservations
C. Implement cluster autoscaler with node affinity and resource bin packing algorithms
D. Disable orchestration and manage manually

Answer: C
Explanation: Implementing cluster autoscaler with node affinity and resource bin packing algorithms minimizes fragmentation by efficiently allocating resources. Overprovisioning CPUs by 20% on all VMs risks instability. Disabling orchestration and managing manually reduces automation. Using smaller VMs without reservations increases management overhead.

Question: 1191

You are managing a legacy project that still uses CVS for source code management. A new developer joins the team and attempts to check out a module using 'cvs checkout projectX', but receives an error indicating a conflict because another developer has locked a file. The project has strict requirements for sequential updates due to regulatory compliance, and the team lead explains that this locking is intentional to avoid merge issues. In this context, what is the primary characteristic of CVS as a centralized SCM that enforces this behavior, especially when compared to modern distributed systems?

A. CVS allows concurrent edits without locks, leading to automatic merges
B. CVS's branching model automatically resolves conflicts without user intervention
C. CVS enforces file locking to ensure only one user modifies a file at a time, typical of centralized SCMs
D. CVS uses distributed repositories for better scalability in large teams

Answer: C
Explanation: CVS, as a centralized SCM, typically uses a locking mechanism where users must lock files before editing to prevent concurrent modifications, which aligns with the error seen when trying to checkout a locked module. This is designed for environments needing strict control, like the regulatory compliance in the scenario, unlike distributed systems where concurrent work is encouraged and conflicts are resolved during merges. This locking ensures sequential updates but can slow down development in larger teams.

Question: 1192

Artifactory OSS vs Nexus: both cache pip 'requests==2.31.0.whl'. Pipeline 'pip download -d wheels/'. Nexus tasks clean old snapshots weekly. Vuln in 2.31.0 prompts purge. What distinguishes repo roles?

A. Nexus tasks for automated cleanup
B. Semantic purge policies
C. Preview wheel validation
D. GitOps cache reconciliation

Answer: A
Explanation: Nexus 'cleanup tasks' automate old/vuln artifact purge (e.g., weekly snapshots), unlike Artifactory manual; both cache but Nexus excels scheduled maintenance for compliance. Semantic labels, loops sync, preview test—but tasks key.

Question: 1193

A video streaming service uses AWS EKS for orchestration but faces cold-start latency spikes in Fargate pods during traffic surges. CloudWatch metrics show 5s startup for sporadic requests. To design for serverless cloud deployment, which strategy minimizes latency while maintaining auto-scaling?

A. ECS with spot instances for cost savings
B. EKS Fargate with minimum 10 pods always running
C. AWS Lambda with provisioned concurrency for warm starts
D. Persistent EC2 nodes with manual scaling policies

Answer: C
Explanation: Lambda's provisioned concurrency pre-warms execution environments, reducing cold starts to sub-100ms for bursty workloads like streaming, integrated with API Gateway for serverless scaling without cluster management overhead. EKS Fargate's 5s spikes indicate poor fit for sporadic loads; Lambda optimizes cloud-native serverless design with pay-per-use and global replication. This suits designing software for ephemeral cloud services.

Question: 1194

Relational PostgreSQL on Trove with pg_trgm extension indexes 1B JSON rows for fuzzy search, but pg_stat_statements shows high CPU from GIN scans during peaks. Work_mem:4G**B.** What indexing concept overloads, and what BRIN hybrid cuts costs?

A. Materialized view refreshes
B. GIN trigram overlap penalties
C. Parallel sequential scans vacuum
D. Exclusion constraints on tsranges

Answer: B
Explanation: Relational full-text uses GIN for trigrams, CPU-heavy on overlaps. DevOps adds BRIN for sorted prefixes, tunes effective_cache; CI benchmarks queries.

Question: 1195

During a DevOps pipeline optimization for a financial services firm using a FaaS platform, engineers deploy a function triggered by HTTP requests to process transaction validations. In lab tests, the function exhibits a cold start latency of 800ms on initial invocation, reducing to 50ms on subsequent calls within a 5-minute window, with execution costs calculated at $0.0000002 per millisecond of compute time. When scaling to handle 10,000 concurrent requests, the platform automatically provisions ephemeral containers. What key feature of FaaS is demonstrated by the automatic handling of cold starts and per-invocation billing in this scenario?

A. Event-driven execution with automatic scaling
B. Serverless architecture eliminating provisioning overhead
C. Stateless function design for independent invocations
D. Pay-per-use pricing model excluding idle time

Answer: D
Explanation: Pay-per-use pricing model excluding idle time charges only for actual execution time, such as $0.0000002 per millisecond, without costs for idle periods, which aligns with the observed cold start latency of 800ms dropping to 50ms on warm invocations. This encourages efficient code design for sporadic workloads like transaction validations. Event-driven execution with automatic scaling handles triggers and concurrency like 10,000 requests, serverless architecture eliminating provisioning overhead removes server management, and stateless function design for independent invocations ensures each call is isolated but doesn't directly address billing or latency.

Question: 1196

As a lead engineer, you investigate a repository where commits are not in chronological order due to cherry-picking, affecting bisect operations. The structure shows non-linear history. To linearize the history for better debugging while preserving changes, which rebase option do you apply?

A. git rebase --interactive --autosquash
B. git rebase --onto

C. git rebase --keep-base
D. git rebase --root --interactive

Answer: B
Explanation: git rebase --onto replants a range of commits onto a new base, linearizing history by excluding merged branches, improving bisect efficiency for debugging, without altering the root or squashing unnecessarily, maintaining individual commit integrity.

Question: 1197

NPM 'next.js' (MIT), transitive 'amphtml/validator' (Apache). Dual prop/MIT.

A. Both permissive; dual fine
B. NPM dual block
C. Next GPL
D. Validator BSD strict

Answer: A
Explanation: MIT/Apache enable proprietary dual-licensing seamlessly.

Question: 1198

A software team is preparing to perform a database migration that involves significant schema changes. What is the best practice for ensuring data consistency during the migration?

A. Migrate data in bulk at the end of the migration
B. Disable all applications during the migration
C. Use data synchronization tools throughout the process
D. Migrate data without validation checks

Answer: C
Explanation: Using data synchronization tools throughout the migration process is the best practice for ensuring data consistency. These tools can help keep the data in sync between the old and new schemas, minimizing the risk of data loss or inconsistency during the migration and ensuring a smooth transition.

Question: 1199

A development team is facing challenges with managing dependencies in their application deployed on a PaaS platform. Which feature can help streamline dependency management?

A. Manual installation of libraries
B. Limited support for third-party libraries
C. Built-in dependency resolution
D. Static resource allocation

Answer: C
Explanation: Built-in dependency resolution in PaaS platforms automates the management of application dependencies, simplifying the deployment process and reducing the likelihood of version conflicts and errors.

Question: 1200

Legacy monolith integrated via SOA wrapper fails during traffic spike: ESB queues 1M messages, services timeout. Database shared. Risk?

A. Monitoring gaps in wrapper layer
B. Scaling ESB ignores service independence
C. Coupled components cascade failures via central bus
D. Async queues hide latency until saturation

Answer: C
Explanation: SOA's ESB centralization couples services, queuing overloads cascade timeouts despite healthy backends; contrasts microservices' resilience via direct calls/circuit breakers.

KILLEXAMS.COM

Killexams.com is a leading online platform specializing in high-quality certification exam preparation. Offering a robust suite of tools, including MCQs, practice tests, and advanced test engines, Killexams.com empowers candidates to excel in their certification exams. Discover the key features that make Killexams.com the go-to choice for exam success.

Exam Questions Based on Current Exam Objectives

Killexams.com provides exam questions aligned with the latest official exam objectives and latest syllabus. Our content is reviewed and updated regularly to reflect recent changes announced by certification vendors. By studying these questions, candidates will become cover the structure, difficulty level, and topic coverage of the actual exam, helping them prepare more effectively and efficiently.

Comprehensive Exam MCQs (PDF Format)

Killexams.com offers multiple-choice questions (MCQs) in easy-to-read PDF format, covering all major domains of the exam. Each PDF contains a structured collection of questions and verified answers designed to support focused study. These MCQs help candidates reinforce key concepts, identify knowledge gaps, and improve exam readiness through consistent practice.

Realistic Practice Tests (Online & Desktop)

To support hands-on preparation, Killexams.com provides practice tests through both an Online Test Engine and a Desktop Exam Simulator. These tools are designed to simulate a real exam environment, allowing candidates to practice under exam-like conditions. Performance tracking, test history, and result analysis help users evaluate their progress and focus on areas that need improvement.

Risk-Free Purchase Policy

Killexams.com follows a transparent and customer-friendly purchase policy. If users are not satisfied with the study materials, they may request assistance or a refund in accordance with our published terms and conditions. This policy reflects our commitment to customer satisfaction, fairness, and confidence in our preparation resources.

Regularly Updated Content

Our question bank is reviewed and updated on an ongoing basis to stay aligned with the latest exam outlines and vendor updates. This ensures candidates are studying relevant material and preparing with content that reflects current exam expectations, helping them stay confident and well-prepared.