Oracle 1Z0-1084-25 Exam? No Problem. Crack it Instantly with This Simple Method

DOWNLOAD the newest ExamsReviews 1Z0-1084-25 PDF dumps from Cloud Storage for free: https://drive.google.com/open?id=1E4rquhgIIfiZH4w4CPM0CFw96ZM_53Sk

Users are buying something online (such as 1Z0-1084-25 learning materials), always want vendors to provide a fast and convenient sourcing channel to better ensure the user's use. Because without a quick purchase process, users of our 1Z0-1084-25 learning materials will not be able to quickly start their own review program. So, our company employs many experts to design a fast sourcing channel for our 1Z0-1084-25 Learning Materials. All users can implement fast purchase and use our learning materials.

All smart devices are suitable to use Oracle Cloud Infrastructure 2025 Developer Professional pdf dumps of ExamsReviews. Therefore, you can open this Oracle Cloud Infrastructure 2025 Developer Professional real dumps document and study for the Oracle 1Z0-1084-25 test at any time from your comfort zone. These Oracle 1Z0-1084-25 are updated, and ExamsReviews regularly amends the content as per new changes in the Oracle 1Z0-1084-25 real certification test.

>> Certification 1Z0-1084-25 Exam Dumps <<

100% Free 1Z0-1084-25 – 100% Free Certification Exam Dumps | Valid New Soft Oracle Cloud Infrastructure 2025 Developer Professional Simulations

Up to now, more than 98 percent of buyers of our 1Z0-1084-25 practice braindumps have passed it successfully. And our 1Z0-1084-25 training materials can be classified into three versions: the PDF, the software and the app version. Though the content is the same, but the displays are different due to the different study habbits of our customers. So we give emphasis on your goals, and higher quality of our 1Z0-1084-25 Actual Exam.

Oracle 1Z0-1084-25 Exam Syllabus Topics:

Topic	Details
Topic 1	• Cloud Native Applications and Containerization: This section of the exam covers containerization technologies for cloud-native applications. It explains Docker architecture, its components, and the process of pulling and pushing container images using Oracle Cloud Infrastructure Registry (OCIR). It also explores container orchestration, deploying applications on Oracle Kubernetes Engine (OKE), and using OCI Service Mesh for Kubernetes deployments.

Topic 2	• Monitoring & Troubleshooting Cloud-Native Applications: This section of the exam focuses on monitoring and troubleshooting cloud-native applications. It covers using OCI Monitoring to track metrics, OCI Logging for managing logs and performing tasks related to monitoring, logging, and tracing for better observability and issue resolution.
Topic 3	• Testing and Securing Cloud-Native Applications: This section focuses on testing strategies and security for cloud-native applications. It discusses different testing methodologies, securing sensitive information using OCI Vault, and implementing security measures to address cloud-native development challenges.
Topic 4	• Leveraging Serverless Technologies for Cloud Native Development: This section of the exam measures the skills of professionals in serverless development within OCI. It covers creating serverless applications using Oracle Functions, building API gateways for routing traffic, and integrating systems through OCI Streaming Service. Additionally, it explores event-driven architectures using OCI Event Service and how OCI Queue enables asynchronous messaging between microservices.
Topic 5	• Cloud Native Fundamentals: This section of the exam measures the skills of target audience and covers the essential principles of cloud-native development. It explains the core concepts, key pillars, and advantages of cloud-native applications. The section also focuses on microservices architecture, including its design methodology and how it supports scalable, distributed applications.

Oracle Cloud Infrastructure 2025 Developer Professional Sample Questions (Q30-Q35):

NEW QUESTION # 30
You are creating an API deployment in Oracle Cloud Infrastructure (OCI) API Gateway and you want to configure request policies to control access. Which is NOT available in OCI API Gateway?

- A. Enabling Cross-Origin Resource Sharing (CORS) support.
- B. Limiting the number of requests sent to the backend services.
- C. Providing authentication and authorization.
- D. Controlling access to the backend OCI resources.

Answer: D

Explanation:
The correct answer is: Controlling access to the backend OCI resources. OCI API Gateway does not provide direct control over access to backend OCI resources. It primarily focuses on managing and securing access to APIs exposed through the gateway. The gateway acts as a front-end for APIs and provides features such as authentication, authorization, rate limiting, and CORS support. While you can configure authentication and authorization policies, limit the number of requests, and enable CORS support in OCI API Gateway, it does not directly control access to backend OCI resources. Access to backend resources is typically managed through other means, such as IAM policies, network security rules, or resource-specific access controls.

NEW QUESTION # 31
Which is ONE of the differences between a microservice and a serverless function?

- A. Microservices are used for long running operations while serverless functions are used for short running operations.
- B. Microservices always use a data store while serverless functions never use a data store.
- C. Microservices are stateless while serverless functions are stateful.
- D. Microservices are triggered by events while serverless functions are not.

Answer: A

Explanation:
The correct answer is: Microservices are used for long running operations while serverless functions are used for short running operations. One of the key differences between microservices and serverless functions is the duration of their execution. Microservices are typically designed to handle long-running operations and may continuously run and process requests as part of a larger system. They are often deployed and managed as long-lived services. On the other hand, serverless functions are designed to

handle short-lived operations or tasks that execute in response to specific events or triggers. They are event-driven and execute only when invoked, providing a lightweight and ephemeral computing model. Serverless functions are often used for executing small, isolated pieces of code without the need for managing infrastructure or scaling concerns. While both microservices and serverless functions can be stateless or stateful depending on the specific implementation, the key distinction lies in the typical duration and execution pattern of these components within an application architecture.

NEW QUESTION # 32
Which statement accurately describes the Oracle Cloud Infrastructure (OCI) Load Balancer integration with OCI Container Engine for Kubernetes (OKE)?

- A. OCI Load Balancer instance must be manually provisioned for each Kubernetes service that requires traffic balancing.
- B. OKE service provisions a single OCI Load Balancer instance shared with all the Kubernetes services with LoadBalancer type in the YAML configuration.
- C. OCI Load Balancer instance provisioning is triggered by the OCI Events service for each Kubernetes service with LoadBalancer type in the YAML configuration.
- D. OKE service provisions an OCI Load Balancer instance for each Kubernetes service with LoadBalancer type in the YAML configuration.

Answer: D

Explanation:
The statement that accurately describes the Oracle Cloud Infrastructure (OCI) Load Balancer integration with OCI Container Engine for Kubernetes (OKE) is: "OKE service provisions an OCI Load Balancer instance for each Kubernetes service with LoadBalancer type in the YAML configuration." When you define a Kubernetes service in your YAML configuration with the LoadBalancer type, the OKE service automatically provisions an OCI Load Balancer instance specifically for that service. This Load Balancer instance is dedicated to the Kubernetes service and provides traffic balancing functionality. Each Kubernetes service that requires load balancing will have its own OCI Load Balancer instance provisioned by OKE.

NEW QUESTION # 33
A company is developing a new application that needs to process transactions in real time. The company wants to ensure that all transactions are processed in order and that no transaction is lost. Which of these is a correct strategy for leveraging OCI Queue in this scenario?

- A. Use a single queue to process all transactions.
- B. Use a priority queue to prioritize requests.
- C. Use a separate queue for each type of transaction.
- D. Use a separate queue for each application instance.

Answer: A

Explanation:
OCI Queue is a service for enabling asynchronous (decoupled) communication in a serverless manner3. Queue handles high-volume transactional data that requires independent processing without loss or duplication3. Queue supports ordering of messages within a queue by using the FIFO (first-in-first-out) delivery option3. Therefore, using a single queue to process all transactions ensures that all transactions are processed in order and that no transaction is lost. Verified Reference: Overview of Queue

NEW QUESTION # 34
You are developing a distributed application and you need a call to a path to always return a specific JSON content deploy an OCI API Gateway with the below API deployment specification. What is the correct value for type? { "routes" : [{ "path" : "/hello", "methods" : ["Get"), "backend" : { "type" : " ---------- ", "status" : 200, "headers" : [{ "name" : "Content-Type", "value" : "application/json" }] "body" : "{\"myjson\": \"consistent response\"}" } }]}

- A. HTTP_BACKEND
- B. STOCK_RESPONSE_BACKEND
- C. JSON_BACKEND
- D. CONSTANT_BACKEND

Answer: B

Explanation:
The correct value for the "type" field in the API deployment specification is "STOCK_RESPONSE_BACKEND". By setting the "type" to "STOCK_RESPONSE_BACKEND", you are indicating that the backend for the specified route should return a pre-defined response. This type of backend is commonly used when you want a specific response to be returned consistently, regardless of the actual backend service implementation. In this case, the API deployment specification is configured to have a single route with the path "/hello" and the method "GET". The backend section specifies the type as "STOCK_RESPONSE_BACKEND". Additionally, it defines the response status code as 200, sets the "Content-Type" header to "application/json", and provides the JSON content in the "body" field. Using this configuration, any request to the "/hello" path with the "GET" method will always receive a consistent JSON response with the content "{"myjson": "consistent response"}".

NEW QUESTION # 35
......