KCNA参考書内容 & KCNA PDF

BONUS！！！ It-Passports KCNAダンプの一部を無料でダウンロード：https://drive.google.com/open?id=1NLZy-UE9Y6cYxgWex24QwqBhALZ1jqUw

人はそれぞれの夢を持っています。あなたの夢は何でしょうか。昇進ですか。あるいは高給ですか。私の夢はLinux FoundationのKCNA認定試験に受かることです。この認証の証明書を持っていたら、全ての難問は解決できるようになりました。この試験に受かるのは難しいですが、大丈夫です。私はIt-PassportsのLinux FoundationのKCNA試験トレーニング資料を選びましたから。私が自分の夢を実現することを助けられますから。あなたもITに関する夢を持っていたら、速くIt-PassportsのLinux FoundationのKCNA試験トレーニング資料を選んでその夢を実現しましょう。It-Passportsは絶対信頼できるサイトです。

個人がLinux Foundation KCNA試験を正常に完了すると、彼らは認定されたKubernetesとクラウドネイティブアソシエイトと見なされ、キャリアに大きな利益をもたらすことができます。この認定は世界的に認識されており、個人がクラウドネイティブテクノロジーの分野で自分の専門知識と信頼性を実証するのに役立ちます。さらに、この認定を保持している個人は、Linux Foundationが提供する他のKubernetesやCloud-Native認定を追求するために適切に位置付けられており、この急速に進化するテクノロジー環境のスキルと専門知識をさらに高めることができます。

>> KCNA参考書内容 <<

KCNA PDF、KCNA過去問無料

まだKCNA試験に昼夜を問わず滞在していますか？ 答えが「はい」の場合は、It-PassportsのKCNA試験資料を試してください。 私たちLinux Foundationは、最も正確で有用な情報を含むコンテンツだけでなく、最も迅速で最も効率的なアシスタントを提供するアフターサービスについても専門的です。 当社のKCNA練習トレントを20〜30時間使用すると、KCNA試験に参加する準備が整い、期待されるKubernetes and Cloud Native Associateスコアを達成できると主張できます。

Linux Foundation KCNA認定試験は、個人がKubernetesとCloud-Native Technologiesの専門知識を実証する絶好の機会です。クラウドコンピューティングの専門家に対する需要の増加に伴い、この認定は、個人が自分のキャリアを前進させ、専門的な目標を達成するのに役立ちます。

Linux Foundation Kubernetes and Cloud Native Associate 認定 KCNA 試験問題 (Q170-Q175):

質問 # 170
Which Kubernetes-native deployment strategy supports zero-downtime updates of a workload?

- A. Canary
- B. BlueGreen
- C. RollingUpdate
- D. Recreate

正解：C

解説：
D (RollingUpdate) is correct. In Kubernetes, the Deployment resource's default update strategy is RollingUpdate, which replaces Pods gradually rather than all at once. This supports zero-downtime updates when the workload is properly configured (sufficient replicas, correct readiness probes, and appropriate maxUnavailable / maxSurge settings). As new Pods come up and become Ready, old Pods are terminated in a controlled way, keeping the service available throughout the rollout.
RollingUpdate's "zero downtime" is achieved by maintaining capacity while transitioning between versions. For example, with multiple replicas, Kubernetes can create new Pods, wait for readiness, then scale down old Pods, ensuring traffic continues to flow to healthy instances. Readiness probes are critical: they prevent traffic from being routed to a Pod until it's actually ready to serve.
Why other options are not the Kubernetes-native "strategy" answer here:
Recreate (B) explicitly stops old Pods before starting new ones, causing downtime for most services.
Canary (A) and BlueGreen (C) are real deployment patterns, but in "Kubernetes-native deployment strategy" terms, the built-in Deployment strategies are RollingUpdate and Recreate. Canary/BlueGreen typically require additional tooling/controllers (service mesh, ingress controller features, or progressive delivery operators) to manage traffic shifting between versions.
So, for a Kubernetes-native strategy that supports zero-downtime updates, the correct and verified choice is RollingUpdate (D).

質問 # 171
You are running a sensitive application in Kubernetes that requires access to the host network. Which of the following security measures is MOST important to mitigate the risk of potential security breaches?

- A. Using Network Policies to restrict access to the host network.
- B. Deploying the application in a private Kubernetes cluster.
- C. Using a container security scanner to identify and fix vulnerabilities.
- D. Employing a secrets management solution like HashiCorp Vault or AWS Secrets Manager.
- E. Configuring strong passwords for all Kubernetes users.

正解：A

解説：
While accessing the host network can be necessary for some applications, it introduces security risks. Using Network Policies to restrict access to the host network for your sensitive application is crucial. This allows you to control which pods can communicate with the host network, reducing the risk of unauthorized access and potential security breaches.

質問 # 172
To visualize data from Prometheus you can use expression browser or console templates. What is the other data visualization tool commonly used together with Prometheus?

- A. Graphite
- B. Grafana
- C. GraphQL
- D. Nirvana

正解：B

解説：
The most common visualization tool used with Prometheus is Grafana, so A is correct. Prometheus includes a built-in expression browser that can graph query results, but Grafana provides a much richer dashboarding experience: reusable dashboards, variables, templating, annotations, alerting integrations, and multi-data- source support.
In Kubernetes observability stacks, Prometheus scrapes and stores time-series metrics (cluster and application metrics). Grafana queries Prometheus using PromQL and renders the results into dashboards for SREs and developers. This pairing is widespread

because it cleanly separates concerns: Prometheus is the metrics store and query engine; Grafana is the UI and dashboard layer. Option B (Graphite) is a separate metrics system with its own storage/query model; while Grafana can visualize Graphite too, the question asks what is commonly used together with Prometheus, which is Grafana. Option D (GraphQL) is an API query language, not a metrics visualization tool. Option C ("Nirvana") is not a standard Prometheus visualization tool in common Kubernetes stacks.
In practice, this combo enables operational outcomes: dashboards for error rates and latency (often derived from histograms), capacity monitoring (node CPU/memory), workload behavior (Pod restarts, HPA scaling), and SLO reporting. Grafana dashboards often serve as the shared language during incidents: teams correlate alerts with time-series patterns and quickly identify when regressions began.
Therefore, the verified correct tool commonly used with Prometheus for visualization is Grafana (A).
==========

質問 # 173
Which persona is normally responsible for defining, testing, and running an incident management process?

- A. Quality Engineers
- B. Site Reliability Engineers
- C. Application Developers
- D. Project Managers

正解： B

解説：
The role most commonly responsible for defining, testing, and running an incident management process is Site Reliability Engineers (SREs), so A is correct. SRE is an operational engineering discipline focused on ensuring reliability, availability, and performance of services in production. Incident management is a core part of that mission: when outages or severe degradations occur, someone must coordinate response, restore service quickly, and then drive follow-up improvements to prevent recurrence.
In cloud native environments (including Kubernetes), incident response involves both technical and process elements. On the technical side, SREs ensure observability is in place-metrics, logs, traces, dashboards, and actionable alerts-so incidents can be detected and diagnosed quickly. They also validate operational readiness: runbooks, escalation paths, on-call rotations, and post-incident review practices. On the process side, SREs often establish severity classifications, response roles (incident commander, communications lead, subject matter experts), and "game day" exercises or simulated incidents to test preparedness.
Project managers may help coordinate schedules and communication for projects, but they are not typically the owners of operational incident response mechanics. Application developers are crucial participants during incidents, especially for debugging application-level failures, but they are not usually the primary maintainers of the incident management framework. Quality engineers focus on testing and quality assurance, and while they contribute to preventing defects, they are not usually the owners of real-time incident operations.
In Kubernetes specifically, incidents often span multiple layers: workload behavior, cluster resources, networking, storage, and platform dependencies. SREs are positioned to manage the cross-cutting operational view and to continuously improve reliability through error budgets, SLOs/SLIs, and iterative hardening. That' s why the correct persona is Site Reliability Engineers.
==========

質問 # 174
Which statement best describes the role of kubelet on a Kubernetes worker node?

- A. kubelet manages the container runtime and ensures that all Pods scheduled to the node are running as expected.
- B. kubelet configures networking rules on each node to handle traffic routing for Services in the cluster.
- C. kubelet acts as the primary API component that stores and manages cluster state information.
- D. kubelet monitors cluster-wide resource usage and assigns Pods to the most suitable nodes for execution.

正解： A

解説：
The kubelet is the primary node-level agent in Kubernetes and is responsible for ensuring that workloads assigned to a worker node are executed correctly. Its core function is to manage container execution on the node and ensure that all Pods scheduled to that node are running as expected, which makes option A the correct answer.
Once the Kubernetes scheduler assigns a Pod to a node, the kubelet on that node takes over responsibility for running the Pod. It continuously watches the API server for Pod specifications that target its node and then interacts with the container runtime (such as containerd or CRI-O) through the Container Runtime Interface (CRI). The kubelet starts, stops, and restarts containers to match the desired state defined in the Pod specification.

In addition to lifecycle management, the kubelet performs ongoing health monitoring. It executes liveness, readiness, and startup probes, reports Pod and node status back to the API server, and enforces resource limits defined in the Pod specification. If a container crashes or becomes unhealthy, the kubelet initiates recovery actions such as restarting the container.
Option B is incorrect because configuring Service traffic routing is the responsibility of kube-proxy and the cluster's networking layer, not the kubelet. Option C is incorrect because cluster-wide resource monitoring and Pod placement decisions are handled by the kube-scheduler. Option D is incorrect because cluster state is managed by the API server and stored in etcd, not by the kubelet.
In summary, the kubelet acts as the executor and supervisor of Pods on each worker node. It bridges the Kubernetes control plane and the actual runtime environment, ensuring that containers are running, healthy, and aligned with the declared configuration.
Therefore, Option A is the correct and verified answer.

質問 # 175

.....