KaozhengPro

IT認證考試題庫

專業平臺

考證寶提供最新考古題與模擬試題
協助您高效通過認證考試

www.kaozhengpro.com

Exam : **CGOA**

Title : Certified GitOps Associate Exam

Version : DEMO

1.In GitOps practices, when does CD take part?
A. CD takes part simultaneously with CI, both components of GitOps practices.
B. CD takes part after CI to automate the deployment of applications based on changes in the Git repository.
C. CD takes part before CI stage in order to ensure the successful deployment of applications.
D. CI plays a significant role in GitOps practices.
Answer: B
Explanation:
In GitOps, Continuous Deployment (CD) follows after Continuous Integration (CI). CI is responsible for building and testing application code, while CD automates the delivery and deployment of these changes into runtime environments. The Git repository serves as the single source of truth, and when CI merges new changes into the main branch, CD reconciles the state of the environment to match what is declared in Git.
“GitOps builds on the principles of DevOps by using Git as the source of truth for declarative infrastructure and applications. CI pipelines handle the integration and testing of code, and CD pipelines or agents automatically reconcile the desired state in Git with the actual state in the cluster.”
This shows that CD is triggered after CI to handle deployment automation, ensuring systems remain in sync with what is declared in version control.
Reference: GitOps Principles (CNCF GitOps Working Group), GitOps Working Group Terminology & Principles documents.

2.In the context of GitOps, what happens to a GitOps-managed Kubernetes cluster if there is drift divergence?
A. The GitOps-managed Kubernetes cluster ignores the drift divergence and continues to operate as it is.
B. The GitOps-managed Kubernetes cluster automatically reconciles the drift divergence to return the cluster to the Desired State.
C. The GitOps-managed Kubernetes cluster notifies the administrator about the drift divergence and waits for manual intervention.
D. The GitOps-managed Kubernetes cluster rolls back to the previous known state before the drift divergence occurred.
Answer: B
Explanation:
A GitOps-managed Kubernetes cluster uses reconciliation loops to continuously compare the actual state of the system with the desired state declared in Git. When drift (divergence between declared configuration and live cluster state) is detected, the GitOps operator automatically reconciles the difference to bring the system back into alignment.
“In GitOps, a reconciliation loop ensures that the desired state as declared in Git is continuously compared with the observed state of the system. If drift is detected, the system automatically takes corrective action to reconcile the difference and restore the declared configuration.”
This ensures consistency, reliability, and self-healing. Manual intervention is not required for drift correction, as the automated reconciliation is a core principle of GitOps.
Reference: GitOps Principles (CNCF GitOps Working Group), GitOps Principles Document — Principle 4: Software agents automatically pull the desired state declarations from the source and continuously

observe actual system state, reconciling differences.

3.In GitOps, what does the principle of Versioned and Immutable mean?
A. All changes to configuration and infrastructure should be made directly on production environments.
B. All software versions should be stored in a Git repository.
C. Configuration and infrastructure code should be version-controlled and treated as immutable artifacts.
D. Configuration and infrastructure code should be modified directly on production environments.
Answer: C
Explanation:
One of the four fundamental GitOps principles is Versioned and Immutable. This means that the entire system's desired state must be stored in a Git repository with version control. Each change must be represented as a commit, and Git's immutability guarantees a reliable, auditable history of how the system evolved.
"The desired state is stored in a version control system. The record of truth is stored in an immutable history, and changes can be audited and reverted if necessary. This guarantees that the system's configuration is versioned, immutable, and traceable."
Thus, configuration and infrastructure must be version-controlled and immutable, never changed directly in production.
Reference: GitOps Principles (CNCF GitOps Working Group), Principle 2: The desired system state is stored as versioned and immutable.

4.When are progressive delivery patterns useful in software development and deployment?
A. Progressive delivery patterns are only useful for one-time, single-deployment scenarios, not ongoing, continuous delivery.
B. Progressive delivery patterns are primarily beneficial for small development teams rather than for large organizations.
C. Progressive delivery patterns are useful in several software development and deployment scenarios, as they offer advantages such as risk reduction, improved quality, and better user experience.
D. Progressive delivery patterns are useful during initial project development instead of in subsequent phases.
Answer: C
Explanation:
Progressive delivery is a GitOps pattern used to release software gradually, reducing risks associated with deploying new versions. Techniques such as canary releases, feature flags, and blue-green deployments allow teams to incrementally roll out changes, validate functionality with subsets of users, and minimize potential disruptions.
"Progressive delivery builds on continuous delivery by enabling safer, incremental rollouts. This pattern reduces risk, improves reliability, enhances user experience, and allows for validation of features with a portion of users before wider release."
Therefore, progressive delivery is useful in multiple scenarios (not just one-time deployments or small teams), making option C correct.
Reference: GitOps Patterns (CNCF GitOps Working Group), Progressive Delivery Patterns documentation.

5.When deciding whether to use an in-cluster reconciler or an external reconciler, what factors should be considered?
A. The version of Kubernetes and the availability of network resources.
B. The size of the cluster and the complexity of the reconciler logic.
C. The programming language the applications are written in.
D. The location of the state store and the number of replicas.
Answer: B
Explanation:
In GitOps, reconcilers ensure the actual state matches the desired state. Reconcilers may run inside the cluster (in-cluster) or outside (external). The choice depends primarily on operational scale and the complexity of reconciliation logic.
“When determining reconciler placement, factors such as the size of the environment, the operational complexity of the reconciler, and the performance requirements should be evaluated. In-cluster reconcilers are common for straightforward deployments, while external reconcilers may be chosen for large-scale or complex systems.”
Thus, the most important considerations are cluster size and complexity of reconciler logic, making B correct.
Reference: GitOps Related Practices (CNCF GitOps Working Group), GitOps Reconciler Guidelines.