

Google

GCP-CDOE

Cloud DevOps Engineer

For More Information – Visit link below:

<https://www.examsempire.com/>

Product Version

1. Up to Date products, reliable and verified.
2. Questions and Answers in PDF Format.



<https://examsempire.com/>

Visit us at: <https://www.examsempire.com/gcp-cdoe>

Latest Version: 6.0

Question: 1

You deploy a new release of an internal application during a weekend maintenance window when there is minimal user traffic. After the window ends, you learn that one of the new features isn't working as expected in the production environment. After an extended outage, you roll back the new release and deploy a fix. You want to modify your release process to reduce the mean time to recovery so you can avoid extended outages in the future. What should you do? Choose 2 answers

- A. Before merging new code, require 2 different peers to review the code changes.
- B. Adopt the blue/green deployment strategy when releasing new code via a CD server.
- C. Integrate a code linting tool to validate coding standards before any code is accepted into the repository.
- D. Require developers to run automated integration tests on their local development environments before release.
- E. Configure a CI server
- F. Add a suite of unit tests to your code and have your CI server run them on commit and verify any changes.

Answer: B,E

Question: 2

You need to deploy a new service to production. The service needs to automatically scale using a Managed Instance Group (MIG) and should be deployed over multiple regions. The service needs a large number of resources for each instance and you need to plan for capacity. What should you do?

- A. Use the n1-highcpu-96 machine type in the configuration of the MIG.
- B. Monitor results of Stackdriver Trace to determine the required amount of resources.
- C. Validate that the resource requirements are within the available quota limits of each region.
- D. Deploy the service in one region and use a global load balancer to route traffic to this region.

Answer: C

Explanation:

https://cloud.google.com/compute/quotas#understanding_quotas

<https://cloud.google.com/compute/quotas>

Question: 3

You support a web application that runs on App Engine and uses CloudSQL and Cloud Storage for data storage. After a short spike in website traffic, you notice a big increase in latency for all user requests, increase in CPU use, and the number of processes running the application. Initial troubleshooting reveals:

After the initial spike in traffic, load levels returned to normal but users still experience high latency. Requests for content from the CloudSQL database and images from Cloud Storage show the same high latency.

No changes were made to the website around the time the latency increased. There is no increase in the number of errors to the users. You expect another spike in website traffic in the coming days and want to make sure users don't experience latency. What should you do?

- A. Upgrade the GCS buckets to Multi-Regional.
- B. Enable high availability on the CloudSQL instances.
- C. Move the application from App Engine to Compute Engine.
- D. Modify the App Engine configuration to have additional idle instances.

Answer: D

Explanation:

Scaling App Engine scales the number of instances automatically in response to processing volume. This scaling factors in the `automatic_scaling` settings that are provided on a per-version basis in the configuration file. A service with basic scaling is configured by setting the maximum number of instances in the `max_instances` parameter of the `basic_scaling` setting. The number of live instances scales with the processing volume. You configure the number of instances of each version in that service's configuration file. The number of instances usually corresponds to the size of a dataset being held in memory or the desired throughput for offline work. You can adjust the number of instances of a manually-scaled version very quickly, without stopping instances that are currently running, using the Modules API `set_num_instances` function.

<https://cloud.google.com/appengine/docs/standard/python/how-instances-are-managed>

https://cloud.google.com/appengine/docs/standard/python/config/appref#max_idle_instances

Optional. The maximum number of idle instances that App Engine should maintain for this version. Specify a value from 1 to 1000. If not specified, the default value is automatic, which means App Engine will manage the number of idle instances. Keep the following in mind: A high maximum reduces the number of idle instances more gradually when load levels return to normal after a spike. This helps your application maintain steady performance through fluctuations in request load, but also raises the number of idle instances (and consequent running costs) during such periods of heavy load.

Question: 4

You use Cloud Build to build your application. You want to reduce the build time while minimizing cost and development effort. What should you do?

- A. Use Cloud Storage to cache intermediate artifacts.
- B. Run multiple Jenkins agents to parallelize the build.
- C. Use multiple smaller build steps to minimize execution time.
- D. Use larger Cloud Build virtual machines (VMs) by using the machine-type option.

Answer: C

Explanation:

<https://cloud.google.com/storage/docs/best-practices>

[https://cloud.google.com/build/docs/speeding-up-](https://cloud.google.com/build/docs/speeding-up-builds#caching_directories_with_google_cloud_storage)

[builds#caching_directories_with_google_cloud_storage](https://cloud.google.com/build/docs/speeding-up-builds#caching_directories_with_google_cloud_storage) Caching directories with Google Cloud Storage

To increase the speed of a build, reuse the results from a previous build. You can copy the results of a previous build to a Google Cloud Storage bucket, use the results for faster calculation, and then copy the new results back to the bucket. Use this method when your build takes a long time and produces a small number of files that does not take time to copy to and from Google Cloud Storage.

Question: 5

You encounter a large number of outages in the production systems you support. You receive alerts for all the outages that wake you up at night. The alerts are due to unhealthy systems that are automatically restarted within a minute. You want to set up a process that would prevent staff burnout while following Site Reliability Engineering practices. What should you do?

- A. Eliminate unactionable alerts.
- B. Create an incident report for each of the alerts.
- C. Distribute the alerts to engineers in different time zones.
- D. Redefine the related Service Level Objective so that the error budget is not exhausted.

Answer: A

Explanation:

Eliminate badmonitoring : Unactionable alerts (i.e., spam)

<https://cloud.google.com/blog/products/management-tools/meeting-reliability-challenges-with-sre-principles>

agree with kyubiblaze about having to remove unactionable items aka spam: "good monitoring alerts on actionable problems" @ <https://cloud.google.com/blog/products/management-tools/meeting-reliability-challenges-with-sre-principles>

Question: 6

You support a high-traffic web application with a microservice architecture. The home page of the application displays multiple widgets containing content such as the current weather, stock prices, and news headlines. The main serving thread makes a call to a dedicated microservice for each widget and then lays out the homepage for the user. The microservices occasionally fail; when that happens, the serving thread serves the homepage with some missing content. Users of the application are unhappy if this degraded mode occurs too frequently, but they would rather have some content served instead of no content at all. You want to set a Service Level Objective (SLO) to ensure that the user experience does not degrade too much. What Service Level Indicator (SLI) should you use to measure this?

- A. A quality SLI: the ratio of non-degraded responses to total responses
- B. An availability SLI: the ratio of healthy microservices to the total number of microservices

- C. A freshness SLI: the proportion of widgets that have been updated within the last 10 minutes
- D. A latency SLI: the ratio of microservice calls that complete in under 100 ms to the total number of microservice calls

Answer: B

Explanation:

<https://cloud.google.com/blog/products/gcp/available-or-not-that-is-the-question-cre-life-lessons>

Question: 7

You are ready to deploy a new feature of a web-based application to production. You want to use Google Kubernetes Engine (GKE) to perform a phased rollout to half of the web server pods. What should you do?

- A. Use a partitioned rolling update.
- B. Use Node taints with NoExecute.
- C. Use a replica set in the deployment specification.
- D. Use a stateful set with parallel pod management policy.

Answer: A

Explanation:

<https://medium.com/velotio-perspectives/exploring-upgrade-strategies-for-stateful-sets-in-kubernetes-c02b8286f>

Question: 8

You support an application deployed on Compute Engine. The application connects to a Cloud SQL instance to store and retrieve data.

a. After an update to the application, users report errors showing database timeout messages. The number of concurrent active users remained stable. You need to find the most probable cause of the database timeout. What should you do?

- A. Check the serial port logs of the Compute Engine instance.
- B. Use Stackdriver Profiler to visualize the resources utilization throughout the application.
- C. Determine whether there is an increased number of connections to the Cloud SQL instance.
- D. Use Cloud Security Scanner to see whether your Cloud SQL is under a Distributed Denial of Service (DDoS) attack.

Answer: B

Thank You for Trying Our Product
Special 16 USD Discount Coupon: NSZUBG3X

Email: support@examsempire.com

**Check our Customer Testimonials and ratings
available on every product page.**

Visit our website.

<https://examsempire.com/>