ExamsIndex
Practice Materials & Tests

DEMO VERSION

Cloudera

CDP-3002 Exam

CDP Data Engineer- Certification Exam

Exam Latest Version: 7.4

Question 1. (Multi Select)

A Data Engineer is planning the migration of legacy spark-submit batch jobs to production workflows running via the Cloudera Data Engineering CLI (Spark over Kubernetes). Which TWO statements accurately reflect best practices or operational characteristics for deploying and managing Spark applications in this CDE environment?

A: The deployment mode for Spark jobs in CDE must be manually specified as —deploy-mode=client, ensuring the driver runs on the submitting machine.
B: The CDE CLI command cde job run is preferred over cde spark submit for production execution because it requires the job definition to be permanently defined beforehand, simplifying repeated execution and scheduling.
C: Custom containerized execution environments are exclusively supported via the custom-runtime-image resource type, allowing Docker images to run as both Driver and Executor pods.
D: When setting resource limits, CDE defaults the maximum number of executors to the cluster's CPU core limit divided by the configured —executor-cores setting.
E: Job dependencies, such as application JAR files or external reference configurations, must be managed using the files resource type, ensuring they are mounted to the standard lapp/mount directory on the workload pods.

Correct Answer: B, E

Explanation:

Statement B is correct: 'cde job run' requires a pre-created job definition and is generally suited for production environments where a job is run multiple times, contrasting with 'cde spark submit', which is better for development testing as it doesn't create permanent definitions. Statement E is correct: The 'files' resource type is used for application code (like JARs or Python files) and external reference configuration files, which are mounted to the '/app/mount' location on the Spark driver and executor pods. Statement A is incorrect: Spark jobs in CDE run using Kubernetes, often implicitly in cluster mode, and requiring '—deploy-mode=client' is usually inappropriate for production Spark jobs. The CDE CLI abstracts the cluster management details. Statement C is partially correct but too restrictive: While 'custom-runtime-image' supports custom containers, python dependencies can also be managed via 'python-env'. Statement D is incorrect: CDE autoscaling sets the executor range based on CPU cores configured for the Virtual Cluster to maximize efficiency, but the specific calculation involving hard limits isn't stated as a guaranteed default, rather the executor range is set to match the CPU core range of the

Virtual Cluster by default.

Question 2. (Single Select)

A Data Engineer identifies severe performance degradation due to high shuffle I/O caused by too many small shuffle partitions (default 200) in a Spark application running on a CDE Virtual Cluster. To optimize the job, they must set the number of shuffle partitions to 100.

Which option correctly demonstrates how this performance configuration should be applied to the job?

A: Update the Virtual Cluster configuration using cde-utils.sh add-spark-config-in-virtual-cluster -c 'spark.sql.shuffle.partitions=100', then rerun the existing job definition.
B: Set spark.app.id=100 in the application code, ensuring the driver manages partition count implicitly.
C: Execute the job using cde job run —name my_job —conf spark.driver.cores=100, overriding the setting specifically for this run.
D: Modify the Spark driver configuration to include —conf spark.default.parallelism=100 during job creation.
E: Dynamically adjust the cluster's memory fraction by setting spark.memory.storageFraction=0.9 during job submission to force fewer partitions.

Correct Answer: A

Explanation:

To resolve performance issues caused by an inefficient number of shuffle partitions, the property 'spark.sql.shuffe.partitions' must be tuned. In CDE, configuring this value at the Virtual Cluster level using 'cde-utils.sh add-spark-config-in-virtual-cluster' applies the configuration to all jobs running on that cluster. This is suitable if all jobs require this tuning. Alternatively, it can be applied per job using the '—conf flag with 'cde spark submit' or 'cde job run'. Option A provides a mechanism for tuning cluster operations via the administrative utilities. Option B is incorrect as 'spark.app.id' is not supported in CDE job configurations. Option C uses the wrong configuration key (cores instead of shuffle partitions). Option D uses the wrong tuning parameter for shuffle output partitions. Option E relates to memory storage capacity, not shuffle parallelism.

Question 3. (Single Select)

A Data Engineer is configuring a Spark job in Cloudera Data Engineering (CDE) intended to run on a Kubernetes Virtual Cluster. The job processes highly variable datasets, and the engineer aims to minimize costs by only consuming the necessary executor resources. Which pair of configuration flags must be explicitly defined when submitting this job using the cde spark submit command to guarantee resource elasticity within defined bounds?

A: —driver-memory and —driver-cores
B: —executor-memory and —executor-cores
C: —min-executors and —max-executors
D: —conf spark.dynamicAllocation.initialExecutors and —conf spark.dynamicAllocation.enabled=true
E: —conf spark.kubernetes.authenticate.driver.serviceAccountName and —conf spark.kubernetes.namespace

Correct Answer: C

Explanation:

When deploying a Spark job via the CDE CLI using cde spark submit, the minimum and maximum boundaries for resource allocation are controlled by —min-executors and --max-executors. Explicitly setting these flags directly dictates the range within which the Spark application can dynamically allocate executors on the underlying Kubernetes cluster managed by CDE, ensuring the job starts with sufficient resources (minimum) and respects cluster capacity limits (maximum) for cost efficiency.

Question 4. (Single Select)

A data engineering team observes that Spark jobs deployed on their CDE Virtual Cluster (which manages Kubernetes resources) efficiently scale up resources when needed, but hold onto idle executors for too long, delaying scale-down and increasing cloud costs. Which CDE utility command parameter, related to cluster autoscaling, should the engineer decrease to accelerate

the return of resources to the cluster pool?

A: —unremovable-node-recheck-timeout
B: —scale-down-delay-after-add
C: —scale-down-delay-after-failure
D: —scale-down-unneeded-time
E: —scale-down-delay-after-delete

Correct Answer: D

Explanation:

To speed up the process of scaling down, the user can decrease the value of the —scale-down-unneeded-time option, which specifies the amount of time before a node qualifies for scale-down. The default value for this is 10 minutes. By lowering this value, the cluster autoscaler detects idle nodes (which host released Spark executors) more quickly and proceeds with node removal, aligning with the goal of maximizing resource utilization and reducing cost. While other options manage different scale-down delays, —scale-down-unneeded-time directly controls when an idle resource is marked for deletion.

Question 5. (Single Select)

A performance analysis of a dynamic Spark application on Kubernetes shows that while the application started with minimal resources, scaling requests eventually hit a ceiling imposed by the Virtual Cluster (VC) configuration. The current VC configuration has a maximum capacity set to 50 cores. The user submits a job with the following configuration settings: —executor-cores 4 —min-executors 2 --max- executors 20. If dynamic allocation attempts to scale beyond 20 executors, what is the controlling factor that limits the total resources the Spark application can consume?

A: The spark.driver.memory setting, as the driver prevents resource overallocation.
B: The —max-executors 20 flag defined explicitly during job submission.
C: The Virtual Cluster's absolute Auto-Scale Max Capacity setting of 50 cores, regardless of job configuration.
D: The total number of shuffle partitions defined by spark.sql.shuffe.partitions.
E: The limit defined by the Resource Quota set at the CDP environment level.

Correct Answer: B

Explanation:

The maximum number of executors a specific job is allowed to request, even when utilizing dynamic allocation, is governed by the max-executors flag passed during the job submission or configured in the job definition. If this value is set to 20, the Spark application running within the CDE Virtual Cluster (K8s) will not request more than 20 executors, even if the Virtual Cluster itself has a higher overall resource ceiling (like the 50 cores mentioned, which could theoretically support 12 executors running 4 cores each, but the job limit is the immediate constraint, 20 executors 4 cores/executor = 80 cores total requested, which would still fail if VC max capacity is 50 cores). Assuming the VC capacity is sufficient for the job's theoretical max, the job-specific —max-executors is the tightest constraint imposed by the user on the application's runtime scale.

ExamsIndex

Demo PDF Complete

Your CDP-3002 Demo (5 Questions)

Get the Complete Version

Full Questions with Detailed Explanations

Interactive Web-Based Exams Available

To get 30% off, use Coupon Code: NEWYEAR30

https://examsindex.com/exam/cdp-3002