DATABRICKS DATA ENGINEER ASSOCIATE · CHEAT SHEET · CERTIFYDECK

Databricks Certified Data Engineer Associate

Dominios, pesos oficiales y habilidades clave del temario de Databricks Data Engineer Associate, para repasar antes del examen.

Nivel: Intermediate (Associate) Duración: 90 min Fuente: guía de estudio oficial de Microsoft Learn

1. Databricks Intelligence Platform

6–6%

Databricks Intelligence Platform

- Databricks separa un plano de control, gestionado en la cuenta de Databricks, de un plano de proceso: clásico en la cuenta de nube del cliente o serverless en la de Databricks. Delta Lake añade a los archivos Parquet un registro de transacciones con ACID, validación del esquema e historial de versiones. Unity Catalog gobierna tablas, volúmenes y linaje desde un metastore regional compartido por varias áreas de trabajo.
- El proceso serverless para notebooks, trabajos y pipelines lo gestiona Databricks y reduce el tiempo inactivo, aunque no admite R, el Spark UI ni disparadores de streaming por intervalo. El proceso clásico de uso general o de trabajos se configura con modo de acceso estándar o dedicado, escalado automático, terminación automática y pools. Los SQL warehouses serverless, pro o clásicos sirven las cargas SQL con Photon.

2. Data Ingestion and Loading

21–21%

Data Ingestion and Loading

- Lakeflow Connect ofrece conectores administrados para aplicaciones SaaS y bases de datos, que crean pipelines de ingesta incremental hacia tablas de streaming, y conectores estándar basados en API de SQL y Spark, como Auto Loader, COPY INTO o Kafka. Los archivos locales pequeños se suben con la interfaz de carga. En modo desencadenado un pipeline procesa lo disponible y se detiene; en modo continuo, los datos según llegan.
- COPY INTO carga archivos de una ubicación en una tabla Delta existente de forma reintentable e idempotente: omite los archivos ya cargados salvo que se use la opción force. Admite filtros FILES o PATTERN, transformaciones con SELECT, VALIDATE y mergeSchema para inferir y evolucionar el esquema. Con Unity Catalog requiere READ VOLUME en el volumen o READ FILES en la ubicación externa, además de USE CATALOG y USE SCHEMA.
- Auto Loader (fuente cloudFiles) ingiere archivos nuevos de forma incremental y con garantía exactly-once. Con cloudFiles.schemaLocation infiere y hace evolucionar el esquema: addNewColumns detiene el flujo para añadir columnas, rescue las guarda en _rescued_data y los schema hints fijan tipos. Por lotes se usa Trigger.AvailableNow en Lakeflow Jobs; los archivos se detectan por listado de directorios o por notificaciones con file events.
- Los conectores gestionados de Lakeflow Connect usan una conexión de Unity Catalog, un pipeline de ingesta serverless y tablas de streaming de destino; los de base de datos añaden una pasarela continua en proceso clásico y un volumen de staging. Quien crea el pipeline necesita USE CONNECTION y privilegios sobre el catálogo y el esquema, el historial se activa con SCD_TYPE_2 y se despliegan con Declarative Automation Bundles.
- Una conexión JDBC de Unity Catalog guarda el controlador, la URL y las credenciales; se lee con la opción databricks.connection o con remote_query, y externalOptionsAllowList limita las opciones del usuario. Las lecturas en paralelo usan partitionColumn, lowerBound, upperBound y numPartitions. Para API REST se usan conexiones HTTP o secretos con dbutils.secrets.get, se aterriza en volúmenes y se programa con Lakeflow Jobs.

3. Data Transformation and Modeling

22–22%

Data Transformation and Modeling

- La capa silver lee las tablas bronze, preferiblemente en streaming, y aplica limpieza: dropna con subset, how o thresh para nulos, fillna y replace con tipos compatibles, coalesce para valores por defecto y dropDuplicates por claves, con marca de agua en streaming. Para tipos se usa to_date con formato y try_cast, que devuelve NULL en lugar de fallar con valores mal formados.
- Los joins combinan filas por una condición: INNER solo devuelve coincidencias, LEFT conserva la tabla izquierda con NULL, SEMI y ANTI filtran por existencia y CROSS JOIN, o un join sin ON, genera el producto cartesiano. broadcast() o la sugerencia BROADCAST evitan el shuffle de la tabla grande. UNION elimina duplicados y UNION ALL los conserva; en PySpark union es posicional y unionByName empareja por nombre.
- En PySpark, withColumn añade columnas, withColumnRenamed las renombra (sin error si el nombre no existe), drop devuelve un DataFrame sin ellas y filter o where conserva las filas que cumplen la condición. split devuelve un array y split_part una parte; explode crea una fila por elemento y explode_outer conserva los arrays NULL o vacíos. En tablas Delta, renombrar o eliminar columnas solo con metadatos requiere column mapping.
- dropDuplicates elimina duplicados comparando todas las columnas o un subconjunto, y en streaming necesita un watermark para acotar el estado; MERGE solo con WHEN NOT MATCHED evita reinsertar registros, pero el lote debe venir deduplicado. count(*) incluye NULL y count(col) no; approx_count_distinct estima únicos con HyperLogLog++; avg o mean ignoran NULL; summary añade cuartiles a describe.
- spark.sql.shuffle.partitions fija las particiones de los shuffles de DataFrames y SQL, y en Databricks el valor auto deja que AQE las calcule; spark.default.parallelism solo afecta a los RDD. spark.sql.autoBroadcastJoinThreshold limita el broadcast (-1 lo desactiva) y la pista BROADCAST lo fuerza en una consulta. La memoria del driver depende del tipo de driver. Tras cada cambio, se vuelve a medir con la Spark UI.

4. Working with Lakeflow Jobs

16–16%

Working with Lakeflow Jobs

- Lakeflow Jobs controla el flujo con políticas de reintentos (max_retries, intervalo y retry_on_timeout), dependencias Run if (All succeeded, All done, At least one failed…), la tarea de condición If/else, que bifurca según valores de tarea o parámetros, y la tarea For each, que ejecuta una tarea anidada por cada entrada. Las reparaciones vuelven a ejecutar solo las tareas fallidas y sus dependientes, desde el principio.
- Un trabajo de Lakeflow Jobs es un DAG de tareas: notebook, SQL (consulta, alerta o archivo .sql en un SQL warehouse serverless o pro), dashboard (refresca un dashboard publicado y avisa a suscriptores) y pipeline (una actualización por ejecución, o continua si el trabajo es continuo). Las dependencias se fijan con depends_on y la condición Run if decide si la tarea se ejecuta según el resultado de las previas.
- Los desencadenadores de Lakeflow Jobs incluyen programado (simple o avanzado con cron de Quartz y zona horaria; UTC evita saltos por horario de verano), llegada de archivos (volumen o ubicación externa de Unity Catalog, sin comodines, recursivo) y actualización de tabla (una o todas las tablas, con SELECT para la identidad Run as). Las opciones de tiempo mínimo entre ejecuciones y de espera tras el último cambio controlan la frecuencia.
- Usa una programación cuando el requisito es una hora fija o el origen no notifica cambios; usa desencadenadores de llegada de archivos o de actualización de tabla cuando los datos llegan a horas imprevisibles, sin clúster permanente ni conocer los procesos de origen. Para baja latencia, un trabajo continuo; si las tareas dependen entre sí en el mismo equipo, dependencias dentro de un único trabajo en vez de horarios encadenados.

5. Implementing CI/CD

10–10%

Implementing CI/CD

- Las carpetas Git (antes Databricks Repos) son un cliente Git en el área de trabajo: se clonan con CAN MANAGE sobre la carpeta principal y desde el cuadro de diálogo de Git se crean ramas, se cambia de rama, se hace pull y Commit & Push. Si no puedes hacer commit en main, trabaja en otra rama y abre la pull request en el proveedor; cada persona usa su propia carpeta y rama, y Merge no reescribe el historial, a diferencia de Rebase y Reset.
- Las variables personalizadas se declaran en variables y se usan con ${var.nombre}; su valor sale, por este orden, de --var, de BUNDLE_VAR_, de variable-overrides.json, del mapeo variables del destino y del default. Son de despliegue (para cada ejecución, parámetros del job), admiten type: complex y lookup de ID, y los ajustes de targets se unen a los de nivel superior por task_key o job_cluster_key, con prioridad del destino.
- Un bundle tiene un único databricks.yml que incluye archivos de recursos (include) y define jobs, pipelines y código como fuente; cada destino puede fijar mode, presets, permissions y run_as. El modo development antepone [dev usuario], pausa programaciones y desactiva el bloqueo; el modo production valida la rama Git y exige pipelines development: false. El despliegue sigue los recursos por ID; cambiar nombre, destino o área rompe la asociación.
- La CLI de Databricks gestiona el ciclo del bundle: validate comprueba la configuración, plan muestra cambios sin aplicarlos, deploy -t despliega en un destino, run ejecuta jobs o pipelines (--only, --params, --validate-only) y destroy --auto-approve elimina lo desplegado. generate y deployment bind incorporan recursos existentes; en CI se usa una entidad de servicio con variables de entorno o federación de identidades.

6. Troubleshooting, Monitoring, and Optimization

10–10%

Troubleshooting, Monitoring, and Optimization

- La vista de matriz de la pestaña Runs muestra el historial de ejecuciones y tareas con barras cuya altura indica la duración, y avisa si se supera el tiempo esperado (umbral Warning de Run duration). El desglose por fases (Queued, Waiting for resources, Library installation, Running) indica dónde crece el tiempo; para referencias de más de 60 días se consultan las tablas system.lakeflow.
- El estado de una ejecución depende de las tareas hoja: Succeeded with failures si fallan tareas pero todas las hoja terminan bien, Failed si falla alguna hoja, Skipped si se supera el máximo de ejecuciones simultáneas. La vista Graph muestra el DAG para localizar la tarea que bloquea a las Upstream failed, Timeline revela tareas lentas y solapes, y la pestaña Runs ofrece el gráfico de ejecuciones y los cinco errores más frecuentes.
- En la Spark UI se empieza por el jobs timeline, se abre la etapa más larga y se leen sus columnas Input, Output, Shuffle Read y Shuffle Write y el número de tareas. Hay sesgo si en Summary Metrics la duración Max supera mucho al percentil 75; el spill aparece en los detalles de la etapa y se reduce con spark.sql.shuffle.partitions=auto, menos datos en memoria o más memoria por ejecutor.
- Liquid clustering (CLUSTER BY) sustituye al particionado y a ZORDER, permite cambiar las claves sin reescribir datos y es incremental; OPTIMIZE FULL reagrupa los datos existentes y CLUSTER BY AUTO deja que Databricks elija las claves. Predictive optimization ejecuta OPTIMIZE, VACUUM y ANALYZE solo en tablas administradas de Unity Catalog, con herencia de cuenta a catálogo, esquema y tabla.
- Un init script de clúster que devuelve un código distinto de cero impide el arranque: se diagnostica con la entrega de registros y los logs del init script. Para conflictos de bibliotecas, %pip instala en ámbito de notebook y dbutils.library.restartPython() carga la nueva versión. Con ejecutores perdidos se mira primero el event log; la memoria se verifica duplicando la memoria por núcleo, y un collect() grande pide un driver mayor.

7. Governance and Security

15–15%

Governance and Security

- Las tablas administradas de Unity Catalog (Delta Lake o Iceberg) guardan sus datos en la ubicación administrada del esquema, del catálogo o del metastore, se optimizan solas y borran sus archivos tras DROP TABLE al vencer el periodo de recuperación. Las externas usan LOCATION sobre una ubicación externa y DROP solo elimina metadatos. ALTER TABLE ... SET MANAGED convierte una externa conservando nombre, permisos e historial.
- En Unity Catalog se concede con GRANT y se retira con REVOKE, desde SQL o la pestaña de permisos de Catalog Explorer; leer una tabla exige USE CATALOG, USE SCHEMA y SELECT. Los privilegios en catálogos y esquemas se heredan a los objetos actuales y futuros; los del metastore no. MANAGE delega la gestión sin dar acceso a datos, y la instrucción DENY solo se aplica a hive_metastore.
- Un filtro de filas es una UDF de SQL booleana asociada con ALTER TABLE ... SET ROW FILTER que excluye las filas donde devuelve FALSE; una máscara de columna (SET MASK, con USING COLUMNS para más columnas) sustituye valores. Usa is_account_group_member para grupos de cuenta. Las políticas ABAC con etiquetas gobernadas escalan a muchas tablas y las vistas dinámicas sirven para datos combinados sin acceso a las tablas base.
- Las políticas ABAC de Unity Catalog aplican filtros de filas y máscaras de columna desde un catálogo o esquema a todas las tablas cuyas etiquetas gobernadas cumplan sus condiciones (WHEN y MATCH COLUMNS), con una UDF y cláusulas TO/EXCEPT; cubren tablas nuevas y los propietarios no pueden quitarlas. No conceden acceso: hace falta SELECT directo.