Prueba Teórica – Gerente Especializado en Cloudera y Spark

Fecha de generación: 2025-05-07

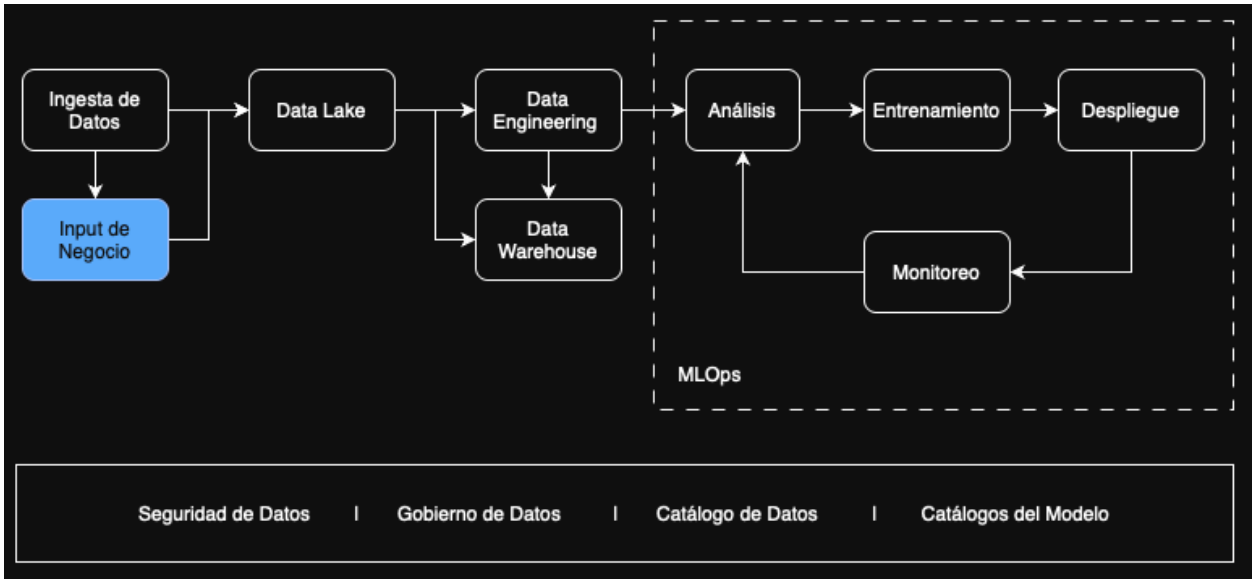
Duración estimada: 45 minutos

Instrucciones: Responda de manera clara y concisa. Se evaluará la profundidad técnica, claridad conceptual y alineación con buenas prácticas.

1. **Cloudera Machine Learning y CDP**
   Explique cómo integraría Cloudera Machine Learning (CML) dentro de una arquitectura analítica moderna para una empresa que usa Cloudera Data Platform (CDP). Incluya en su respuesta qué beneficios ofrece CML, qué rol juega dentro del ciclo de vida del dato, y cómo se relaciona con otros componentes como Data Engineering, Data Warehouse o SDX.

Para entender la importancia de entornos de desarrollo, entrenamiento, despliegue y monitoreo de modelo y soluciones analíticas en empresas con operaciones a gran escala es importante replantearse el ciclo de desarrollo de modelos productivos básico que se puede ver en el siguiente diagrama:

Figura 1. Ciclo de desarrollo.



CML permite tener las herramientas necesarias para llevar a cabo el despliegue de un modelo de datos en entorno productivo para todas las fases del proceso de operaciones de ML en un entorno que garantiza y facilita el gobierno de datos, el linaje, transparencia, seguridad y monitoreo de los datos y el modelo en cada parte, siendo así que en particular CML juego el papel de orquestador y ambiente durante el uso de despliegue productivo en el ciclo del dato. En cuanto a su relación con otras componentes tenemos que :

Exploración/Análisis: Por medio de SDX se tiene acceso directo al Data Lake, permitiendo análisis directo de los datos.

Data Engineering: Permite el uso de notebooks por medio de sesiones, para la transformación de datos con Spark o pandas.

Entrenamiento: Compatibilidad de frameworks como skit-learn, MLLib de Spark, Tensor Flow, etc. Para el entrenamiento de modelos, incluyendo configuración de infraestructura de cómputo como CPU/GPUs.

Despliegue: Facilita la creación de endpoint de un sólo clic para la integración del modelo productivo con aplicaciones para uso de los clientes y de explotación analítica.

Monitoreo: Registros, métricas de desempeño y auditoría desde SDX son algunas de las opciones por las cuales CML utiliza las herramientas de CDP para el monitoreo del modelo.

2. **Apache Spark**
   Describa los principios fundamentales de Apache Spark y por qué es preferido sobre Hadoop MapReduce en muchos escenarios de procesamiento de datos. Incluya en su respuesta conceptos como evaluación perezosa (lazy evaluation), tolerancia a fallos, y paralelismo en memoria.

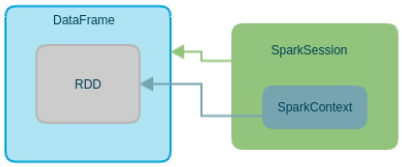
Respuesta:

A continuación los principios fundamentales de Spark:

Cómputo In-memory: Spark guarda resultados de procesos intermedios en la memoria temporal (RAM) y no en disco, a diferencia de MapReduce, esto ahorra tiempo al momento de ejecución al evitar la lectura y escritura en disco para pasos intermedios.

Lazy Evaluation: Los comandos de consulta y transformación del código no son inmediatamente ejecutadas, Spark crea una DAG (Directed Acyclic Graph, o gráfica dirigida sin ciclos), en donde se almacena el orden de ejecución de las operaciones, y no es sino hasta que se llama a una acción que esta cadena de acciones es ejecutada, evitando así el cómputo innecesario de bloques de código.

Tolerancia a fallos: La estructura en la cual están basados los Data Frame por medio de un RDD (Resilient Distributed Dataset) que permite el almacenamiento de objetos en los nodos de un cluster recordando las indicaciones particulares para llegar a ese objeto, de manera que si fallaba un nodo, spark recalcula la parte afectada. Esto se desarrolla en un contexto y sesiones de Spark como en el siguiente diagrama.



Programación unificada: Spark permite la manipulación de diferentes formatos de datos dentro del mismo motor, permitiendo un procesamiento ágil.

Paralelismo transparente: Spark ejecuta en paralelo de manera automática sin importar si se utiliza el API desde Python, R u otro.

3. **Seguridad de la Información y Gobierno del Dato**
   En un entorno empresarial con información confidencial de clientes, ¿cómo implementaría un esquema robusto de gobierno y seguridad del dato? Considere aspectos como clasificación de datos, control de accesos, linaje, cifrado, auditoría y cumplimiento normativo (p. ej., GDPR, LFPDPPP).

Respuesta:

Para garantizar la seguridad y el gobierno de los datos propondría trabajar con las siguientes líneas de acción de manera paralela y constante:

Clasificación de datos: Usar herramientas de perfilamiento de datos como Caludesa Data Catalog para tener entendimiento exhaustivo de los datos que se tienen recopilados, clasificarlos según el nivel y tipo de dato (uso interno, transaccional, confidencial, restringido, etc.) Usar herramientas como Atlas para definir el acceso a los datos basado en la sensibilidad de los mismos y los roles de los usuarios.

Control de Acceso Basado en Roles: Se deben de definir roles de acceso para los usuarios y clientes alineados con el negocio que cumplan con criterios de acceso mínimo, es decir, sólo dar acceso a los usuarios a los datos mínimos necesarios para el desempeño de labores y proyectos.

Data lineage: Con el uso de Apache Atlas proporcionar el linaje consistente de los datos desde las fuentes de extracción, la transformación e ingeniería de características. Identificando fuentes de datos y procesos. Crear flujos documentados y de ser posible visualizaciones y herramientas que permitan la trazabilidad de los datos. Alinear las definiciones de los datos con las disposiciones y usos establecidos en normativas para tener una auditoría más ágil.

Cifrado de datos: Cifrado de punta a punta de la información entre nodos, clientes, interfaces y APIs, así como el cifrado en reposo por medio de las configuraciones nativas en Spark.

Auditoría interna y cumplimiento: Tener logs, monitoreos y alertas con respecto al uso de los datos por cada uno de los clientes de datos. Registrar, diferenciar y respetar el consentimiento del uso de datos de los usuarios, permitir el ejercicio de los derechos ARCO de los usuarios, tener políticas de borrado de datos y logs así como de manejo de datos.

4. **Buenas Prácticas de Programación en Python y SQL**
   Describa tres buenas prácticas clave que aplicaría al desarrollar soluciones analíticas en Python y SQL en un entorno empresarial de misión crítica. Fundamente cada una con un ejemplo.

Respuesta:

Ejemplo 1: Se debe de garantizar la eficiencia y reusabilidad del código por medio de modularizar el código, es decir, generar funciones auxiliares para actividades o partes del código que serán utilizadas más de una vez.

```python
# Función para calcular el monto total y agregarlo en un dataframe que contiene
# montos y cantidades

def transform_sales(df):
    """
    Calcula el total de ventas por fila multiplicando el precio por la cantidad.

    Parámetros:
    df (pd.DataFrame): DataFrame que contiene al menos las columnas 'price' y
    'quantity'.

    Regresa:
    pd.DataFrame: El mismo DataFrame con una nueva columna llamada 'total' que
    representa el producto de 'price' y 'quantity'.

    """
    # Calcula el total de ventas por producto (precio * cantidad)
    return df.assign(total=df['price'] * df['quantity'])
```

Ejemplo 2: Documentar y comentar todas las funciones y partes de código de manera que sea documentado y pueda ser reutilizable y mantenido incluso por personas que no participaron en su creación. Aplica también para SQL, al definir cada uno de los filtros, condiciones y campos que se están utilizando en la query.

```sql
-- ============================================
-- Consulta: Top 5 clientes con mayor facturación
-- Fecha: 2025-05-10
-- Autor: Equipo de Datos
-- Objetivo: Identificar a los clientes más valiosos del último trimestre
-- ============================================

-- Paso 1: Seleccionar órdenes del último trimestre con sus montos
WITH ordenes_filtradas AS (
    SELECT
        o.client_id,
        o.order_id,
        o.order_date,
        o.total_amount
    FROM orders o
    WHERE o.order_date >= DATE_TRUNC('quarter', CURRENT_DATE)
),

-- Paso 2: Agregar métricas por cliente
facturacion_por_cliente AS (
    SELECT
        of.client_id,
        COUNT(of.order_id) AS total_ordenes,
        SUM(of.total_amount) AS total_gastado
    FROM ordenes_filtradas of
    GROUP BY of.client_id
)

-- Paso 3: Seleccionar el top 5 clientes
SELECT
    fpc.client_id,
    fpc.total_ordenes,
    fpc.total_gastado
FROM facturacion_por_cliente fpc
ORDER BY fpc.total_gastado DESC
LIMIT 5;
```

Ejemplo 3: Mantener tests unitarios y pruebas automatizadas para cada una de las funciones que se determinan, así como controles de calidad de datos en cada campo.

Por ejemplo, supongamos que tenemos una función calcular descuento(monto, porcentaje), que devuelve el monto aplicado un descuento, los siguientes son algunos de los tests a aplicar:

```python
import pytest
from src.utils import calcular_descuento

def test_descuento_valido():
    """Verifica que el descuento se aplique correctamente"""
    resultado = calcular_descuento(100.0, 20.0)
    assert resultado == 80.0

def test_descuento_cero():
    """Verifica que no aplicar descuento mantenga el precio original"""
    assert calcular_descuento(50.0, 0.0) == 50.0

def test_descuento_maximo():
    """Verifica que un 100% de descuento dé un precio de cero"""
    assert calcular_descuento(200.0, 100.0) == 0.0

def test_descuento_invalido_negativo():
    """Debe lanzar ValueError si el descuento es negativo"""
    with pytest.raises(ValueError):
        calcular_descuento(100.0, -10.0)

def test_descuento_invalido_excesivo():
    """Debe lanzar ValueError si el descuento es mayor al 100%"""
    with pytest.raises(ValueError):
        calcular_descuento(100.0, 150.0)
```

5. **Frameworks de trabajo: Scrum, DevOps y DataOps**
Explique cómo combinaría las prácticas de Scrum, DevOps y DataOps en un proyecto de analítica avanzada sobre Cloudera. Incluya ejemplos de cómo facilitaría la colaboración entre equipos, cómo aplicaría automatización, y cómo garantiza la calidad e integridad del dato.

Respuesta:

Scrum es un framework de desarrollo que se origina de Agile, de manera que el uso de los puntos en este framework son directamente aplicables a un caso de uso de analítica avanzada, automatización o productivización de un modelo por medio de DevOps o MLOps. A continuación se muestra un ejemplo de la implementación de un ciclo de scrum para un proyecto de datos basado en productivizar un modelo y generar un reporte en un dashboard.

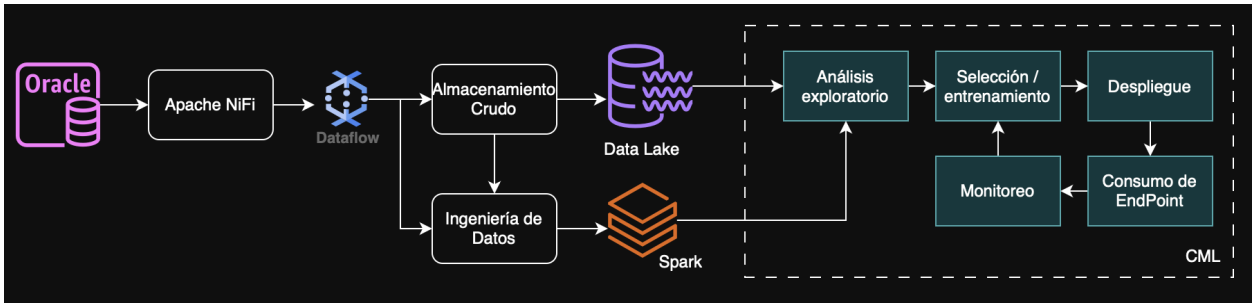
Fase del Sprint	Actividad Scrum	DevOps	DataOps
Refinamiento	Se define y/o priorizar las tareas del backlog que son necesarias para el desarrollo e implementación del modelo y su puesta en producción	Implementación de un pipeline de ingesta/ingeniería de características necesarias para el modelo.	Con ayuda de Atlas se captura el linaje de columnas y los campos infestados/utilizados para el proyecto.
Daily standup	Implementación y validación del modelo, selección del mismo y ajuste a hiper parámetros y mejora en el entrenamiento,	Test unitarios para cada una de las funciones y componentes utilizadas en el modelo, CI automático y registro de logs.	Se tienen conversaciones con el negocio para validar reglas expertas y de calidad de los datos.
Revisión	Revisamos las métricas de evaluación del mejor modelo desarrollado y se revisa el dashboard presentado.	Se utiliza CML para generar el despliegue del modelo con un REST API /Endpoint para el consumo del modelo por equipos analíticos o de negocio.	Usamos SDX para poder tener una auditoría y monitoreo de los datos.
Retro	Se recapitulan las lesiones encontradas en el desarrollo del modelo, así como la retroalimentación de usuarios finales.	Se trabaja en mejoras en el CI/CD o validaciones que puedan exponer áreas de oportunidad o mejoras.	Se realiza un ajuste a las reglas de calidad o incluso ajustes al modelo para mejorar su usabilidad por el cliente.

Para mejorar la integración con los equipos buscaría utilizar la interfaz de CML y CDP para crear entornos colaborativos, implementar controles de calidad y revisión automáticos, política de uso basadas en el rol, un catálogo compartido y estandarizado, además de una correcta definición de los roles y responsabilidades correspondientes a los ítems del backlog.

6. **Arquitectura de solución para analítica de datos**
Diseñe a grandes rasgos una arquitectura de procesamiento y análisis de datos usando Cloudera, para una empresa que requiere integrar datos desde Oracle, analizarlos en Spark y generar modelos predictivos. Mencione los componentes involucrados, flujo de datos, y cómo gestionaría seguridad, gobierno y escalabilidad.

Respuesta:

La arquitecturas propuesta se representa en el siguiente diagrama:



Ingesta: Utilizaremos en la componente de Apache NiFi para conectar directamente con nuestra fuente de datos en Oracle y procesar la información dentro del data lake. Se construirán catálogos de datos y configuración de permisos.

Procesamiento: Con Cloudera Data Engineering podemos generar el procesamiento distribuido en un entorno de Apache Spark. Podemos estructurar en un sistema HDFS y posteriormente transformar y limpiar los datos en un datawarehouse estructurado.

CML (MFLOps): Con la interfaz de CML podemos utilizar la información ingestada y estructurada en las fases anteriores para hacer análisis, exploración, desarrollo de modelos predictivos, entrenamiento , su despliegue y monitoreo.

En cuanto a la escalabilidad, las herramientas de ingesta, de almacenamiento y también de procesamiento que se tienen en la plataforma de Cloudera, y particularmente, el desarrollo en un ambiente Spark permite la escalabilidad en cada parte del proceso de manera dinámica. Nifi tiene la capacidad de escalar horizontalmente, por su parte Spark permite procesamiento en paralelo sobre ejecutores en YARN o Kubernetes, aunado con la infraestructura HDFS permite una arquitectura distribuida por medio del particionamiento.

Finalmente, para garantizar el cumplimiento normativo y altos estándares de seguridad se tomarían en cuenta las siguientes medidas y líneas de trabajo:

- **Control de acceso:** Se definen reglas de acceso por nivel de seguridad por medio de Apache Ranger, se definen políticas de acceso y etiquetas de visibilidad con ayuda de Atlas.
- **Data Lineage:** Con la ayuda de Atlas se mantiene el gobierno y linaje de datos a lo largo de todo el proceso.
- **Utilización de Logs:** Análisis y seguimiento por medio de SDX.