DeepLearning.AI

Cuestionario de la semana 4

Estado: Traducido automáticamente del Inglés

Traducido automáticamente del Inglés

¿Todo listo para revisar lo que aprendiste antes de comenzar la tarea? Estoy aquí para ayudarte.

Detalles de la tarea

Fecha límite

11 de agosto 23:59 -0311 de ago. 23:59 -03

Enviado

9 de agosto 2:16 -039 de ago. 2:16 -03

Intentos

Quedan 2 (3 intentos cada 24 horas)

Límite de tiempo

30 minutos por intento30 min por intento

Envíos

1 restantes (1 total dentro del límite de tiempo)

Tu calificación

Para aprobar necesitas al menos un 80 %. Guardamos tu puntaje más alto.

96,66 %

Cuestionario de la semana 4

Tarea calificada • 30 min

Fecha límite11 de ago. 23:59 -03

Tu calificación: 96,66 %

Tu calificación más reciente: 96,66 %•

Tu calificación más alta: 96,66 %•

Para aprobar necesitas al menos un 80 %. Guardamos tu puntaje más alto.

1.

Pregunta 1

¿Cuáles son las ventajas de la orquestación frente a la simple programación de Cron?

Con Orchestration, puede utilizar exclusivamente la interfaz de usuario (IU) para configurar, definir y activar su DAG.

La orquestación permite establecer dependencias entre tareas de canalización de datos, supervisar tareas, recibir alertas y crear planes de emergencia, en comparación con la programación de tareas con Cron.

La orquestación simplifica la tarea de programación eliminando la necesidad de programación o scripting que se necesita para programar tareas de canalización de datos con Cron.

La orquestación reduce la sobrecarga operativa en comparación con la programación de tareas de canalización de datos con Cron.

Correcto

¡Así es!

1 / 1 punto

2.

Pregunta 2

Algunos de los motores de orquestación requieren que defina su canalización de datos como un grafo acíclico dirigido (DAG). ¿Qué es un DAG?

Un DAG es un grafo utilizado para visualizar su canalización de datos, donde cada tarea es un nodo y las dependencias se ilustran como aristas que indican que los datos fluyen sólo en una dirección y no forman ningún ciclo.

Un DAG es un grafo utilizado para visualizar su canalización de datos, donde cada tarea es un nodo y las dependencias se ilustran como aristas, que fluyen en una dirección y pueden apuntar a una tarea anterior.

Un DAG es un grafo utilizado para visualizar su canalización de datos, donde cada nodo representa datos y las tareas que deben aplicarse a los datos se ilustran como aristas.

Correcto

¡Buen trabajo!

1 / 1 punto

3.

Pregunta 3

Uno de los componentes principales de Airflow es el programador. ¿Cuál es la función del programador?

El programador es donde puede supervisar, activar manualmente y solucionar los problemas de comportamiento de sus DAG generales y de las tareas dentro de cada DAG.

El programador almacena los scripts Python que definen sus DAGs y los ejecuta en un horario.

El programador supervisa todos sus DAG y sus tareas, y activa las tareas en función de la programación o cuando se completan sus dependencias.

El programador se encarga de ejecutar directamente las tareas y actualizar su estado en la IU.

Correcto

¡Gran trabajo!

1 / 1 punto

4.

Pregunta 4

¿Cuál de las siguientes tareas puede realizar en la IU de Airflow? Seleccione todas las que correspondan.

Crear una variable global.

Correcto

Puede crear manualmente variables globales en la IU de Airflow y utilizarlas en su código mediante el método `Variable.get()`.

Activar manualmente un DAG.

Correcto

Esta es una de las funciones de la IU de Airflow que le permite probar sus DAG.

Actualice el script Python que contiene la definición DAG.

Compruebe los registros de cualquier tarea para entender cualquier error de ejecución en su DAG.

Correcto

Esta es una de las funciones de la IU de Airflow que le ayuda a comprender las razones específicas de los fallos de las tareas ofreciéndole mensajes de error detallados que le guían en la solución de problemas y la resolución de problemas con sus DAG.

1 / 1 punto

5.

Pregunta 5

Al crear una instancia de DAG en Airflow, ¿qué parámetro debe especificar si no desea que el programador inicie las ejecuciones del DAG para cualquier intervalo omitido cuando desactive el DAG?

fecha_inicio

tags

descripción

ponerse al día

Correcto

Si este parámetro se establece en true, cuando se desactiva el DAG, el programador iniciará una ejecución DAG para cualquier intervalo perdido cuando el DAG estaba en pausa. Si no desea que esto suceda, debe establecer catchup en False.

1 / 1 punto

6.

Pregunta 6

XCom es una función de Airflow que permite compartir datos entre tareas. Según los vídeos de esta semana, ¿para cuál de los siguientes casos de uso debería utilizarse XCom? Seleccione todos los que corresponda.

Para pasar información sobre fechas entre tareas.

Correcto

¡Gran trabajo! XCom está diseñado para pasar pequeñas cantidades de información, como fechas, entre tareas.

Para pasar grandes DataFrames entre tareas.

Para pasar metadatos entre tareas.

Correcto

Gran trabajo XCom está diseñado para pasar pequeñas cantidades de información, como metadatos, entre tareas.

Para pasar una métrica de precisión de valor único calculada en una tarea ascendente a una tarea descendente.

Correcto

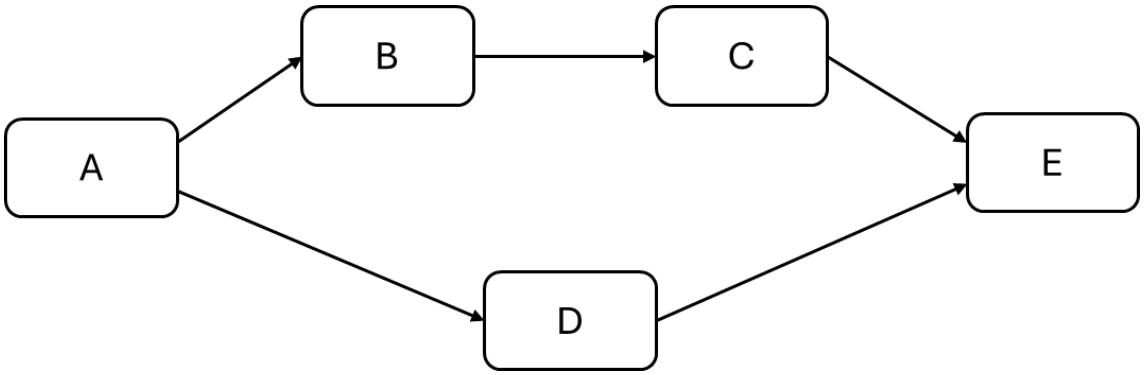
Gran trabajo XCom está diseñado para pasar pequeñas cantidades de información, como una única métrica de valor, entre tareas.

1 / 1 punto

7.

Pregunta 7

Considere el siguiente DAG.



¿Cuál de las siguientes afirmaciones representa correctamente la estructura del DAG?

A >> D >> [B, C] >> E

A >> [B, C] >> D >> E

A >> [B>> C, D] >> E

Correcto

¡Buen trabajo!

1 / 1 punto

8.

Pregunta 8

¿Cuál de las siguientes afirmaciones es cierta acerca de TaskFlow API?

TaskFlow API es un antiguo paradigma de programación para definir tareas y sus dependencias en una determinada canalización de datos.

TaskFlow API representa una API REST que permite a las aplicaciones externas interactuar con su DAG.

TaskFlow API es un paradigma de programación que te permite escribir tus DAGs de una forma más fácil y concisa utilizando decoradores.

TaskFlow API es un nuevo paradigma de programación que se basa en el uso explícito de operadores Airflow.

Correcto

¡Buen trabajo!

1 / 1 punto

9.

Pregunta 9

Según los materiales de clase de esta semana, ¿cuál o cuáles de las siguientes son las mejores prácticas para escribir DAG de flujo de aire? Seleccione todas las que correspondan.

Incluir código de nivel superior antes de la definición DAG en el archivo DAG.

Utilizar variables para almacenar valores que se utilizan más de una vez o que pueden tener que actualizarse

Correcto

Así es. Evite los valores codificados directamente en el código para hacerlo más legible y menos propenso a errores.

Agrupa operaciones relacionadas en una sola tarea.

Esto no debería estar seleccionado

A la hora de identificar las tareas de su canalización de datos, debe mantenerlas simples, de forma que cada tarea represente una operación. Esto ayuda a proporcionar más visibilidad a tu canalización de datos y mejora la legibilidad de tu código.

0.7 / 1 punto

10.

Pregunta 10

Considere el siguiente código:

```
task_1 = PythonOperator(task_id="extract", python_callable=extract_data)
task_2 = PythonOperator(task_id="transform", python_callable=transform_data)
task_3 = PythonOperator(task_id="load", python_callable=load_data)

#define dependencies
None >> None >> None
```

¿Cuál de las siguientes afirmaciones define correctamente las dependencias?

extraer_datos >> transformar_datos >> cargar_datos

"extraer" >> "transformar" >> "cargar"

tarea_1 >> tarea_2 >> tarea_3

Correcto

¡Buen trabajo! En este ejemplo, has utilizado las variables que representan las instancias de PythonOperator.

1 / 1 punto