

Logs de compilación

Last updated by | Walter Arboleda Castañeda | Nov 12, 2024 at 3:02 PM GMT-5

Contents

- [Objetivo](#)
- [Configuración](#)
- [Salida](#)
- [Ejemplos de configuraciones para generar log de compilac...](#)
- [Ejemplos para generar log de compilación con selección d...](#)
 - [Nota](#)

Objetivo

Crear archivos logs que evidencien el funcionamiento de una rutina **sin** la necesidad de realizar ejecuciones con el 100% de los datos, y que estos sean aceptados para el proceso de calendarización.

Configuración

A continuación se describe el paso a paso para la generación de un log de compilación:

1. Se recomienda consultar y entender la función [TAMBLESAMPLE](#) ejecutada por el impala-helper para hacer posible esta funcionalidad.
2. Realice las configuraciones necesarias en el archivo de parámetros tal cual como la rutina debe ejecutar en producción.
3. Configure un ambiente python de acuerdo a la versión instalada en los ambientes donde será desplegada la rutina.
4. En el archivo de ejecución del orquestador2 debe realizar la siguiente configuracion:
 - Los logs de compilación deben almacenarse en la carpeta `logs_calendarizacion` en la raiz del repositorio. Para esto puede utilizar las siguientes lineas de código que identifican el directorio actual y crean la carpeta

```
import pkg_resources
# Identificar el directorio actual
path = pkg_resources.resource_filename(__name__, "")

# Ruta de almacenamiento de logs
logs_path = os.path.join(path.split("src")[0], "logs_calendarizacion")

# Crear la carpeta logs_calendarización
if not os.path.exists(logs_path):
    print("No existía la carpeta de logs para calendarización, se está creando.")
    os.mkdir(logs_path)
```

- El orquestador2 identificará esta configuración mediante los parámetros globales, por tanto se pueden ingresar mediante el archivo `config.json`, o ingresarlos como parámetros adicionales en el archivo de `ejecucion.py`. **En ambas configuraciones se debe recordar configurar de nuevo el paquete al estado en el que será ejecutado en producción luego de finalizar este proceso.**
- Ingrese la ruta de almacenamiento de los logs mediante el parámetro `log_path`.

```
# k: Diccionario de parámetros globales
kw["log_path"] = logs_path
```

- Ingrese el tipo de log que va a generar, en este caso compilación (cmp)

```
# k: Diccionario de parámetros globales
kw["log_type"] = "cmp"
```

- Defina el porcentaje de datos sobre el cual desea hacer la ejecución. Cabe resaltar que el porcentaje a elegir debe garantizar que viajan datos por todos los componentes de la rutina, es decir, todas las tablas deben contener un mínimo de datos, excepto aquellas que su condición es estar vacías. Con esto se garantiza el funcionamiento correcto de la rutina (Puede crear logs de compilación con el 100% de los datos).

```
# Ejecutar la rutina con el 5 porciento de datos a consumir de las fuentes principales
kw["porcentaje_limit"] = 5
```

- Para evaluar que están viajando datos por toda la rutina establezca los controles en los puntos que considere necesario validar en tiempo real el flujo de la información

```
# Cada tabla debe digitarse tal cual como se encuentra en la rutina, incluyendo parámetros
controls = {
    "table_1_{param}": {'control': ['>', 0]},
    "zona.table_2": {'control': ['>', 0]},
    "zona.table_{param}_3": {'control': ['>', 0],
                            'filter': 'where llave_sistema > 0'}
}
kw["ctrls"] = controls
```

- El desarrollador de la rutina tiene la opción de seleccionar las tablas a las cuales se le va a extraer una muestra de la información para la generación de los logs de compilación, para activar dicha opción el usuario debe crear una estructura tipo lista, donde especifique zona y nombre de la tabla, como se muestra a continuación:

```
# Definición de tablas a muestrear
list_tbl_porc = [
    "resultados_vspc_clientes.master_customer_data",
    "resultados riesgos.cun_ci_decision_hist",
    "{resultados_rr}.az_administrator"
]
kw["list_tbl_porc"] = list_tbl_porc
```

- La lista permite parametrizar las zonas de las tablas a muestrear o también se puede incluir completamente la zona y la tabla.
A través de los kwargs asignamos dicha variable para que sea inicializada mediante el orquestador.

4. Para la ejecución de la rutina puede:

- Realizar una instalación local en su maquina, esto puede relizarse ubicando la ruta donde se encuentra su archivo setup.py o setup.cfg (sin incluir el nombre del archivo) y ejecute

```
pip install <path_setup> donde <path_setup> es la ruta en su maquina
```

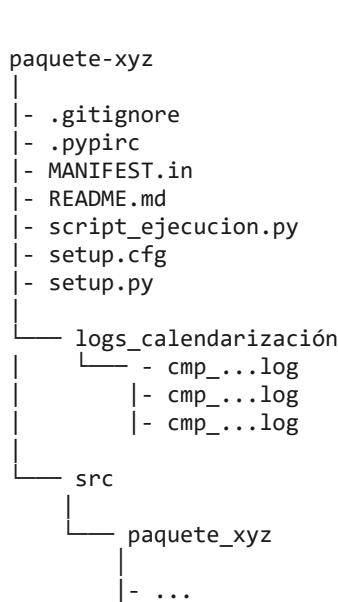
- Ejecute la rutina con el comando

```
python -m nombre_paquete.ejecucion Recuerde añadir los parámetros adicionales por consola si así lo ha configurado
```

- Puede realizar una ejecución del proceso mediante un proceso debug o simulando los parámetros mediante una ejecución tradicional del archivo ejecucion.py .

Salida

Cada ejecucion generará un archivo cmp_.log. Genere los logs que sean necesarios según las [políticas y contratos](#) de calendarización.



```
paquete-xyz
|- .gitignore
|- .pypirc
|- MANIFEST.in
|- README.md
|- script_ejecucion.py
|- setup.cfg
|- setup.py

logs_calendarización
└── - cmp_...log
    ├── - cmp_...log
    └── - cmp_...log

src
└── paquete_xyz
    |- ...
```

Ejemplos de configuraciones para generar log de compilación

- Configuraición en archivo ejecución.py

```

#####
# Generación Logs - Compilación
#####
path = pkg_resources.resource_filename(__name__, "")
logs_path = os.path.join(path.split("src")[0], "logs_calendarizacion")
kw["log_path"] = logs_path
kw["log_type"] = "cmp"
kw["porcentaje_limit"] = 5

controls = {
    "df__canales_cname_unicos_{year_trx}_{month_trx}_{t_exec}": {'control': ['>', 0]},
    "df__canales_aho_cte_{year_trx}_{month_trx}_{t_exec}": {'control': ['>', 0]},
    "df__canales_carteras_{year_trx}_{month_trx}_{t_exec}": {'control': ['>', 0],
                                                            'filter': 'where llave_sistema > 0'}
}
kw["ctrls"] = controls

if not os.path.exists(logs_path):
    print("No existía la carpeta de logs para calendarización, se está creando.")
    os.mkdir(logs_path)
#####

# Paso de los parámetros a los step y al orquestador
steps = [ Step_1(**kw) , Step_2(**kw) , ...]
orquestador = Orchestrator('Orch2-Ejemplo', steps , **kw)

```

- Configuracion Ingresando los parámetros por consola

```
#####
# Generación Logs - Compilación
#####

# Parámetro para determinar tipo de log
parser.add_argument('-lt',
                    '--log_type',
                    type = str,
                    help = 'Tipo de log: normal, compilación o estabilidad',
                    default = '')

# Parámetro para determinar porcentaje de datos
parser.add_argument('-pl',
                    '--porcentaje_limit',
                    type = int,
                    help = 'Porcentaje para el LIMIT en los logs de compilación',
                    default = 100)

args = parser.parse_args()
kw["log_type"] = args.log_type
kw["porcentaje_limit"] = args.porcentaje_limit
path = pkg_resources.resource_filename(__name__, "")

if kw["log_type"] in ["cmp", "est"]:
    logs_path = os.path.join(path.split("src")[0], "logs_calendarizacion")
else:
    logs_path = os.path.join(path.split("src")[0], "logs")
del kw["porcentaje_limit"]

kw["log_path"] = logs_path

if not os.path.exists(logs_path):
    print("No existía la carpeta de logs para calendarización, se está creando.")
    os.mkdir(logs_path)
#####

# Paso de los parámetros a los step y al orquestador
steps = [ Step_1(**kw) , Step_2(**kw) , ...]
orquestador = Orchestrator('Orch2-Ejemplo', steps , **kw)
```

Ejemplos para generar log de compilación con selección de tablas a muestrear

Nota

Utilice esta opción en los siguientes escenarios:

- En toda la rutina solo se necesita hacer muestreo sobre la(s) tabla que consume mas recursos
- Utiliza la librería [param-manager](#), para este caso es escencial asegurarse que no se ejecutará muestreo sobre la tabla paramétrica, pues la función TAMBLESAMPLE no ejecuta en tablas kudu.

- Configuración en archivo ejecución.py

```

#####
# Generación Logs - Compilación
#####
path = pkg_resources.resource_filename(__name__, "")
logs_path = os.path.join(path.split("src")[0], "logs_calendarizacion")
kw["log_path"] = logs_path
kw["log_type"] = "cmp"
kw["porcentaje_limit"] = 5

controls = {
    "df__canales_cname_unicos_{year_trx}_{month_trx}_{t_exec}": {'control': ['>', 0]},
    "df__canales_aho_cte_{year_trx}_{month_trx}_{t_exec}": {'control': ['>', 0]},
    "df__canales_carteras_{year_trx}_{month_trx}_{t_exec}": {'control': ['>', 0],
                                                            'filter': 'where llave_sistema > 0'}
}
kw["ctrls"] = controls

list_tbl_porc = [
    "resultados_vspc_clientes.master_customer_data",
    "resultados riesgos.cun_ci_decision_hist",
    "{resultados_rr}.az_administrator"
]
kw["list_tbl_porc"] = list_tbl_porc

if not os.path.exists(logs_path):
    print("No existía la carpeta de logs para calendarización, se está creando.")
    os.mkdir(logs_path)
#####

# Paso de los parámetros a los step y al orquestador
steps = [ Step_1(**kw) , Step_2(**kw) , ...]
orquestador = Orchestrator('Orch2-Ejemplo', steps , **kw)

```

- Configuración Ingresando los parámetros por consola

```

#####
# Generación Logs - Compilación
#####

# Parámetro para determinar tipo de log
parser.add_argument('-lt',
                    '--log_type',
                    type = str,
                    help = 'Tipo de log: normal, compilación o estabilidad',
                    default = '')

# Parámetro para determinar porcentaje de datos
parser.add_argument('-pl',
                    '--porcentaje_limit',
                    type = int,
                    help = 'Porcentaje para el LIMIT en los logs de compilación',
                    default = 100)

args = parser.parse_args()
kw["log_type"] = args.log_type
kw["porcentaje_limit"] = args.porcentaje_limit
path = pkg_resources.resource_filename(__name__, "")

if kw["log_type"] in ["cmp", "est"]:
    logs_path = os.path.join(path.split("src")[0], "logs_calendarizacion")
else:
    logs_path = os.path.join(path.split("src")[0], "logs")
del kw["porcentaje_limit"]

kw["log_path"] = logs_path

# Definición de controles para la rutina
controls = {
    "df_canales_cname_unicos_{year_trx}_{month_trx}_{t_exec}": {'control': ['>', 0]},
    "df_canales_aho_cte_{year_trx}_{month_trx}_{t_exec}": {'control': ['>', 0]},
    "df_canales_carteras_{year_trx}_{month_trx}_{t_exec}": {'control': ['>', 0],
                                                            'filter': 'where llave_sistema > 0'}
}
kw["ctrls"] = controls

# Definición de tablas a muestrear
list_tbl_porc = [
    "resultados_vspc_clientes.master_customer_data",
    "resultados riesgos.cun_ci_decision_hist",
    "{resultados_rr}.az_administrator"
]
kw["list_tbl_porc"] = list_tbl_porc

if not os.path.exists(logs_path):
    print("No existía la carpeta de logs para calendarización, se está creando.")
    os.mkdir(logs_path)
#####

# Paso de los parámetros a los step y al orquestador
steps = [ Step_1(**kw) , Step_2(**kw) , ...]
orquestador = Orchestrator('Orch2-Ejemplo', steps , **kw)

```

Importante resaltar que, si la opción para listar tablas a las a muestrear no se activa, la generación de los logs de compilación va a muestrear todas las tablas que estén en zonas de resultados y de datos crudos. Si el desarrollador de la rutina quiere este tipo de configuración puede hacer uso también del comodín "*" en los kwargs suministrados al constructor del orquestador de la siguiente manera.

```
# Definicion de tablas a muestrear
list_tbl_porc = ["*"]
kw["list_tbl_porc"] = list_tbl_porc

if not os.path.exists(logs_path):
    print("No existía la carpeta de logs para calendarización, se está creando.")
    os.mkdir(logs_path)
#####
# Paso de los parámetros a los step y al orquestador
steps = [ Step_1(**kw) , Step_2(**kw) , ...]
orquestador = Orchestrator('Orch2-Ejemplo', steps , **kw)
```

