

Atualização de Dashboards

DEFINIÇÕES	INFORMAÇÕES
Nome do Processo	Power BI via API
Dono do Processo	@ Rodrigo de Moraes da Silva
Data da Última Atualização	10/02/2023
Autor	@ Bruno Hideki Wakiyama

Índice

- 1 - Visão Geral
- 2 - Atualização via API
 - 2.1 - Configurações das plataformas
 - 2.1.1 - Adicionar o token do Databricks
 - 2.1.1.1 - Token Databricks
 - 2.1.1.2 - Token Power BI
 - 2.1.1.2.1 - Criar uma aplicação na Azure
 - 2.1.1.2.2 - Registrar os tokens no CLI do Databricks
 - 2.2.1 - Criar um notebook de atualização
 - 2.2.2.1 - Notebook para chamar outro repositório
 - 2.3 - Configuração Power BI
 - 2.3.1 - Acesso a API Power BI
 - 2.3.2 - Configuração da task do Job
- 3 - Atualização via XMLA
 - 3.1 Login da plataforma
 - 3.2 - Atualização das tabelas


1 - Visão Geral

Nesta documentação será abordada a atualização dos dashboards Power BI via API

2 - Atualização via API

2.1 - Configurações das plataformas

Nessa seção será listado o passo a passo de como criar as conexões com o Databricks Azure.

 Essas configurações **já foram realizadas**, caso queira adicionar a atualização do dashboard, utilizar a seção: 3 - Configuração Power BI

2.1.1 - Adicionar o token do Databricks

2.1.1.1 - Token Databricks

Será necessário gerar um token no Databricks e adicionar ele via CLI.

O token deve ser gerado na seção do usuário conforme a imagem 1 abaixo:

User Settings

[Access tokens](#) [Git integration](#) [Notebook settings](#) [Email preferences](#) [Language settings](#) [Preview](#)

Personal access tokens can be used for secure authentication to the Databricks API instead of passwords.

[Generate new token](#)

Comment	Creation ↑	Expiration	
Schedule	2022-10-19 14:56:28 -03	Never	✕
Power BI API	2022-10-17 11:41:26 -03	Never	✕

Após gerar o token, ele deve ser registrado via CLI do databricks. Utilizamos o prompt comand para realizar essas funções.

Foi criado um *scope* “databricks” e dentro dele um *access control rule (ac/s)* com o token gerado na etapa anterior.

```
C:\Users\bruno.hideki\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\Scripts>databricks secrets list-scopes
Scope      Backend      KeyVault URL
-----
adobeNIKE   DATABRICKS   N/A
awsS3       DATABRICKS   N/A
costAzureDB DATABRICKS   N/A
databricks  DATABRICKS   N/A
emailNotificador DATABRICKS   N/A
iHubAPI     DATABRICKS   N/A
jdbcHML     DATABRICKS   N/A
jdbcMySQL   DATABRICKS   N/A
jdbcPGSQL   DATABRICKS   N/A
jdbcPRD     DATABRICKS   N/A
jdbcPRDWrite DATABRICKS   N/A
kv-ad-sql   AZURE_KEYVAULT https://cofre-001.vault.azure.net/
powerbi     DATABRICKS   N/A
sharepointIFC DATABRICKS   N/A
snowflakeNIKE DATABRICKS   N/A
```

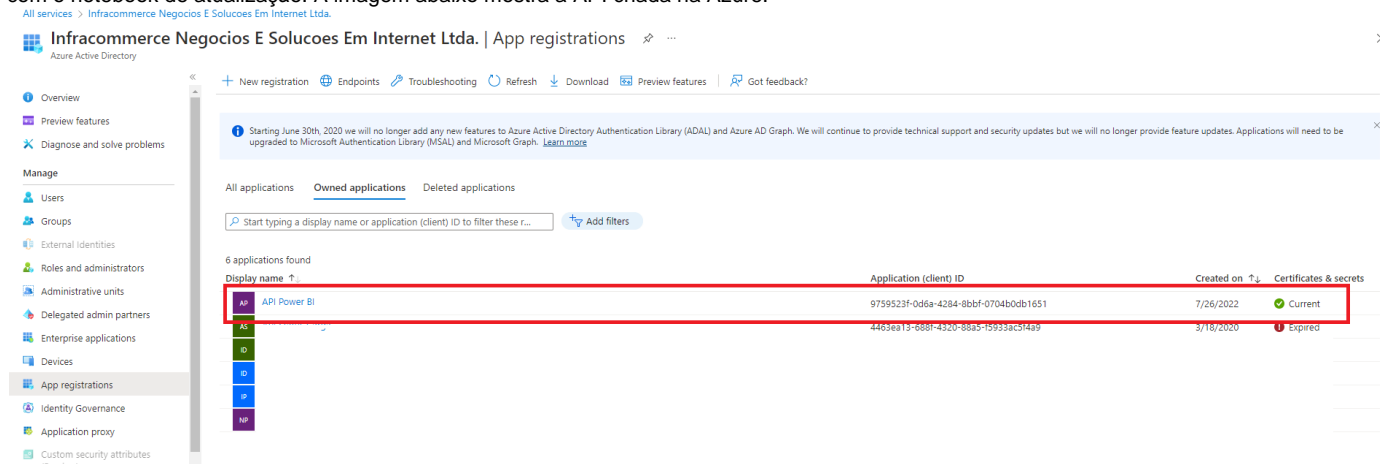
2.1.1.2 - Token Power BI

Será necessário criar uma conexão com a Microsoft Azure, abaixo os passos para conexão com o Token:

- 2.1.1.2.1 - Criar uma aplicação na Azure
- 2.1.1.2.2 - Registrar os tokens no CLI do Databricks

2.1.1.2.1 - Criar uma aplicação na Azure

Foi criado uma conexão “API Power BI” na Azure, ela serve como conector com o Databricks. Cada conexão gera identificadores para conectar com o notebook de atualização. A imagem abaixo mostra a API criada na Azure:



Para criar um novo conector, seguir a documentação abaixo:

Para mais informações, consultar a seguinte documentação
<https://learn.microsoft.com/pt-br/power-bi/developer/embedded/embed-service-principal?source=recommendations>


2.1.1.2.2 - Registrar os tokens no CLI do Databricks

Após gerar o token, ele deve ser registrado via CLI do databricks. Utilizamos o prompt comand para realizar essas funções.

Foi criado um *scope* “powerbi” e dentro dele um *access control rule (ac/s)* com o token gerado na etapa anterior. As *ac/s* adicionadas foram as seguintes: *client_id*, *client_secret* e *tenant_name*. Essas nomenclaturas correspondem aos parametros criados no notebook

```
Cmd 10
1 client_id = dbutils.secrets.get(scope='powerbi', key='client_id')
2 client_secret = dbutils.secrets.get(scope='powerbi', key='client_secret')
3 tenant_name = dbutils.secrets.get(scope='powerbi', key='tenant_name')
```

```
C:\Users\bruno.hideki\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10_qbz5n2kfra8p0\LocalCache\local-packages\Python310\Scripts>databricks secrets list-scopes
Scope      Backend      KeyVault URL
-----
adobeNIKE  DATABRICKS   N/A
awsS3      DATABRICKS   N/A
costAzureDB DATABRICKS   N/A
databricks DATABRICKS   N/A
emailNotificador DATABRICKS   N/A
iHubAPI    DATABRICKS   N/A
jdbcHML    DATABRICKS   N/A
jdbcMYSQL  DATABRICKS   N/A
jdbcPGSQL  DATABRICKS   N/A
jdbcPRD    DATABRICKS   N/A
jdbcPRDWrite DATABRICKS   N/A
kv-ad-sql  AZURE_KEYVAULT https://cofre-001.vault.azure.net/
powerbi    DATABRICKS   N/A
sharepointIFC DATABRICKS   N/A
snowflakeNike DATABRICKS   N/A
```

 Para mais informações, consultar a documentação técnica da Microsoft
<https://learn.microsoft.com/pt-br/azure/databricks/dev-tools/cli/>

2.2.1 - Criar um notebook de atualização

O notebook criado para atualização do Power Bi está localizado no seguinte path do Github:

```
TEVEC/infra-data-products/data_warehouse_layer
/power_bi_auto_refresh_script.py
```

 Para mais informações de atualizações via Rest API segue a documentação da Microsoft
<https://learn.microsoft.com/en-us/power-bi/connect-data/asynchronous-refresh>

2.2.2.1 - Notebook para chamar outro repositório


Esse tópico é um caso exclusivo pois os jobs estão rodando no repositório *Infra-data-visualization* enquanto o script anterior é no *infra-data-products*.

Para atuar no caso, o Juliano Silva criou um notebook que ele faz uma call em outro repositório. A ideia é que toda vez que atualizarmos qualquer dashboard, ele vai chamar esse job, independentemente de quantos estiverem rolando ao mesmo tempo.

Segue o notebook criado pelo Juliano, ele está localizado no repositório que vamos rodar os nossos Jobs, no caso, *infra-data-visualization*

```
TEVEC/infra-data-visualization/Power BI API Refresh/scheduler.py
```

Abaixo o job criado no Databricks Azure na imagem 6:

 **Importante:** não mexer nos parametros nesse estágio

update_powerbi

Runs **Tasks**

Task name *

update_powerbi

Type *

Notebook

Source *

Git provider (develop)

Edit

Path *

data_warehouse_layer/power_bi_auto_refresh_script

Cluster *

Shared_job_cluster

16 GB - 4 Cores - DBR 10.4 LTS - Spark 3.2.1 - Scala 2.12

Dependent libraries

msal

Add

Parameters

WORKSPACE

Nao Mexer

DATASET

Nao Mexer

Add

Advanced options

Git information

×

Git repository URL ?

https://github.com/TEVEC/infra-data-products.git

Git provider

GitHub Enterprise

Git reference (branch / tag / commit) ?

develop

branch

Cancel

Confirm

Após essas configurações é possível continuar somente para a parte de adicionar um novo dashboard para atualizar.

2.3 - Configuração Power BI

Essa seção é para adicionar uma nova atualização do Power BI.

2.3.1 - Acesso a API Power BI

Adicionar ao Workspace que está localizado o Dashboard o acesso "api power bi" como **administrador**, conforme a imagem 8.

Acessar

B2B Estratégia [PRD]

Adicione administradores, membros ou colaboradores. [Saiba mais](#)

API Power BI AppID: 9759523f-0d6a-4284-8bbf-0704b0db1651

Administrador

Adicionar

 Pesquisar

NOME	PERMISSÃO	
Anderson Robert da Silva ⓘ	Visualizador	...
API Power BI (Entidade de Serviço) ⓘ	Administrador	...
Igor Abdo ⓘ	Administrador	...
Infracommerce BI ⓘ	Administrador	...

2.3.2 - Configuração da task do Job

- Task name: [Nome da task]
- Type: Notebook
- Souce:

Git information



Git repository URL

https://github.com/TEVEC/infra-data-visualization.git

Git provider

GitHub Enterprise



Git reference (branch / tag / commit)

master

branch



Cancel

Confirm

Note que ele referencia o a URL infra-data-visualization mesmo por o notebook estar localizado no infra-data-products

- Path: Power BI API Refresh/scheduler
- Dependent libraries: msal (Instalar essa biblioteca no tipo "pypi")
- Parametros:

__WORKSPACE: [Nome do Workspace que foi liberado no 3.1]

__DATASET: [Nome do dashboard]

__TABLE_NAME: [Nome da tabela, se for todas colocar 'all']

__PARTITION_REFRESH: [Se for atualização incremental = true, se for carga full = false]

JOB_NAME: update_powerbi (Sempre será esse pois ele chama o job criado no tópico 2.2.1)

TIMEOUT_IN_SECONDS: 1000000 (Deixar sempre assim)

Task name * ⓘ

Power_BI_Refresh

Type *

Notebook

Source * ⓘ

Git provider (master) Edit

Path * ⓘ

Power BI API Refresh/scheduler

Cluster * ⓘ

PRDMASTERB2B_Faturamento_cluster 96 GB · 48 Cores · DBR 11.1 · Spark 3.3.0 · Scal... Edit

Dependent libraries ⓘ

msal

Add

Parameters ⓘ

UI | JSON

__WORKSPACE	B2B Estratégia [PRD]	X
__DATASET	Faturamento - Lojas B2B	X
JOB_NAME	update_powerbi	X
TIMEOUT_IN_SECONDS	1000000	X

Add

Depends on

FaturamentoB2B X

Advanced options

3 - Atualização via XMLA

A atualização via XMLA funciona somente na **versão v18**

O XMLA é um tipo de conexão que já está integrado com o Power BI

Para mais informações conferir a documentação:
<https://learn.microsoft.com/pt-br/power-bi/enterprise/service-premium-connect-tools>

3.1 Login da plataforma

Para logar selecionar as seguintes informações:

Tipo do servidor: Analysis Services

Nome do servidor: Verificar o link do workspace nas configurações

Configurações

B2B Clientes [PRD]

Sobre

Premium

Conexões do Azure

Modo de licença ⓘ

- ☐ Pro
- ☐ Premium por usuário
- ☒ Premium por capacidade
- ☐ Inserido ⓘ

Selecionar uma capacidade

Capacidade P3 – Brazil South

[Saiba mais sobre as capacidades Premium](#)

Formato de armazenamento padrão

Formato de armazenamento de conjunto de dados grande


[Saiba mais sobre os formatos de armazenamento de conjunto de dados](#)

Conexão do Workspace

powerbi://api.powerbi.com/v1.0/myorg/B2B%20Clientes%20%5BPRD%5D

Copiar

Autenticação: Azure Active Directory - universal com MFA

 Conectar ao Servidor

SQL Server

Tipo de servidor:

Analysis Services

Nome do servidor:

/myorg/Intelig%C3%A2ncia%20Artificial%20%5BPRD%5D

Autenticação:

Azure Active Directory – universal com MFA

Nome de usuário:

infracommerce.bi@accinfra.onmicrosoft.com

Conectar

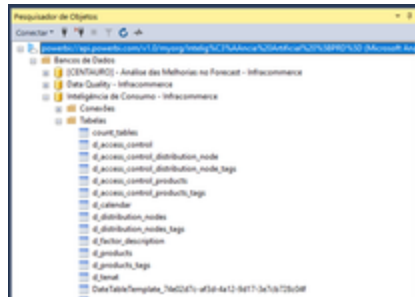
Cancelar

Ajuda

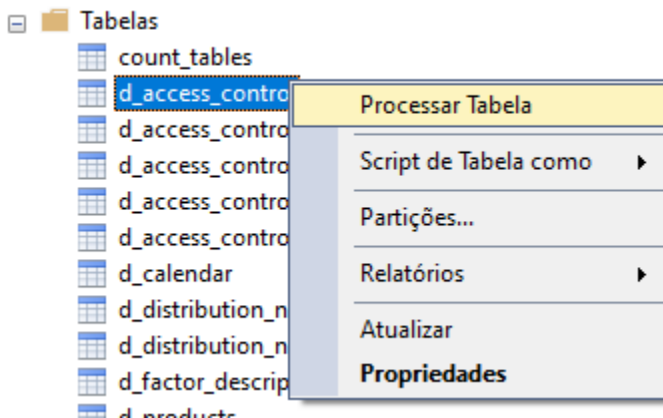
Opções >>

3.2 - Atualização das tabelas

A seguir abrir o caminho: Banco de Dados > [Dashboard] > Tables. Conforme a imagem 13 abaixo



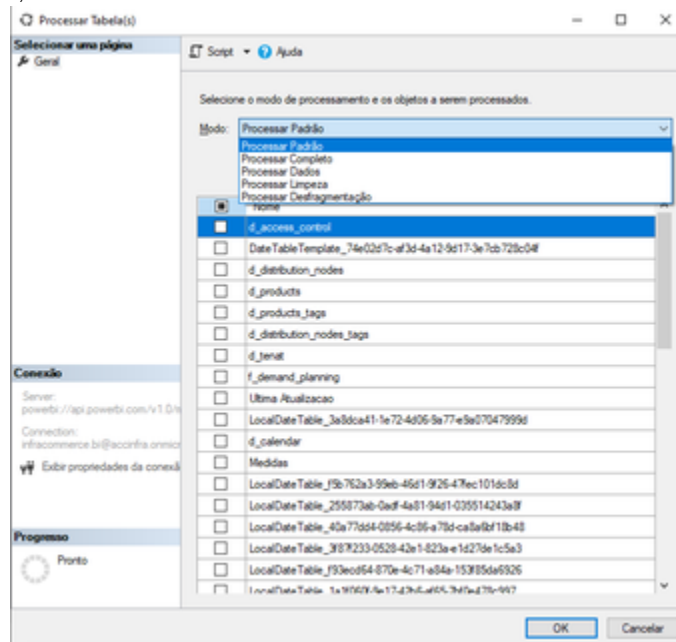
Clicar na tabela que deseja atualizar com o botão direito e selecionar "Processar Tabela"



Podemos selecionar qualquer dessas maneiras de atualizar. Nesse caso, caso seja incremental utilizar o **"Processar Padrão"** e caso seja a carga **full** utilizar **"Processar Completo"**.

Note que podemos selecionar quaisquer tabelas de uma vez para atualizar.

Após selecionar o modo e qual tabela, dar o "Ok" e vai atualizar



Para mais informações dos modos de atualização, utilizar as seguintes documentações:
<https://learn.microsoft.com/en-us/power-bi/connect-data/asynchronous-refresh#parameters>
<https://learn.microsoft.com/en-us/analysis-services/tmsl/refresh-command-tmsl?view=asallproducts-allversions#request>

Após acabar o Dashboard vai registrar a seguinte mensagem no seu histórico

Histórico de atualização



Agendado OneDrive

Detalhes	Tipo	Iniciar	Terminar	Status	Mensagem
	Ponto de extremidade via XMLA	15/02/2023, 11:59:44	15/02/2023, 11:59:47	Concluído	