Data Community Austria Day 2025



THANK YOU

Data Community Austria Day 2025

Awesome Partner

Microsoft

Platinum

redgate

ACP IT for innovators.

Gold

b.telligent
smart data. smart decisions.

Lucient

Bronze

Tabular Editor

paiqo
The Platform & AI Company

Power BI Camp
www.linearis.at

Who am I?

Marisol Steinau

- Autodidact Data Enthusiast
- > 8 years experience using Microsoft Data Platform
- Freelance Data Solution Architect
- Passionate about Architecture & DevOps
- LinkedIn: linkedin.com/in/marisol-steinau-bb1253253

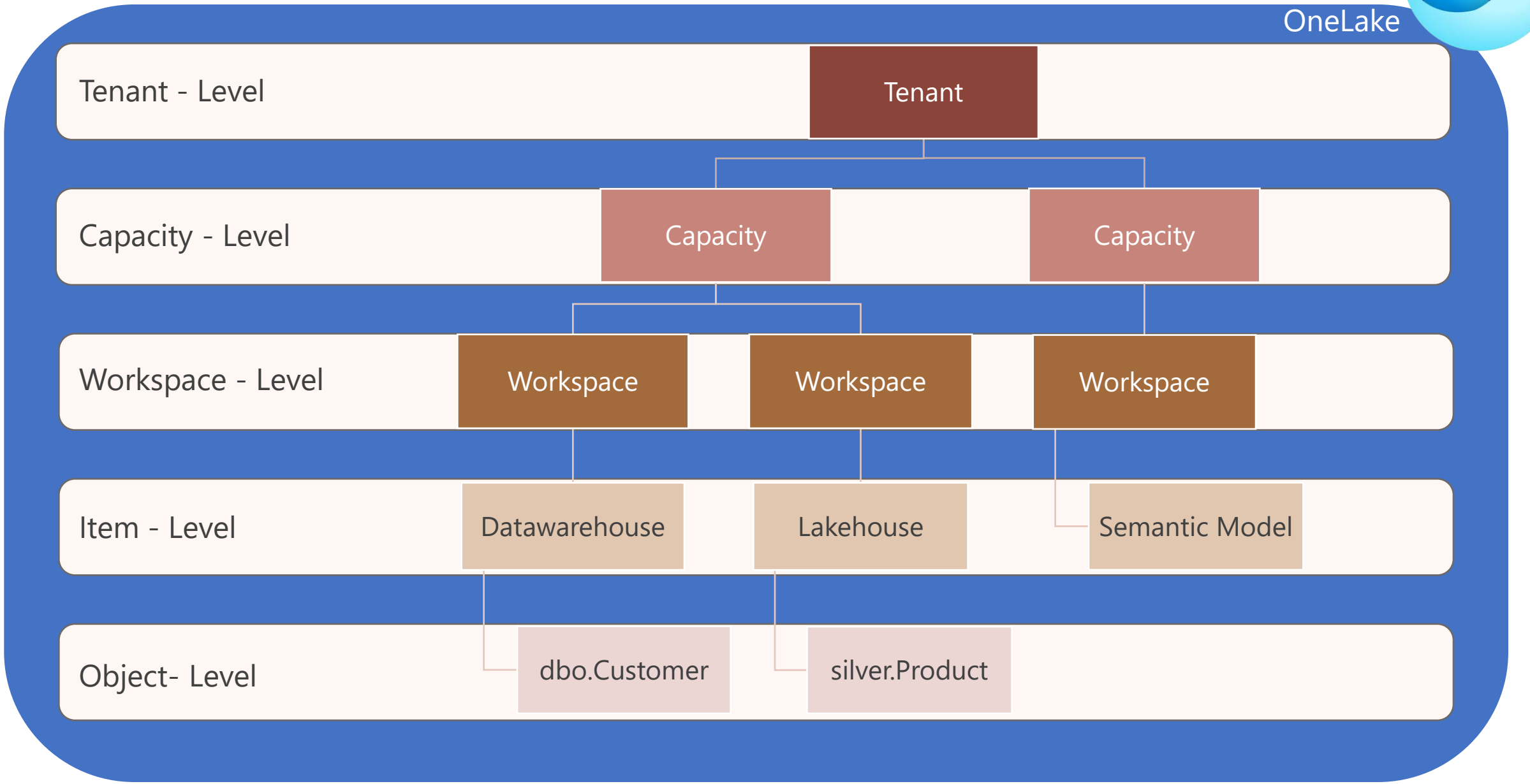


Agenda



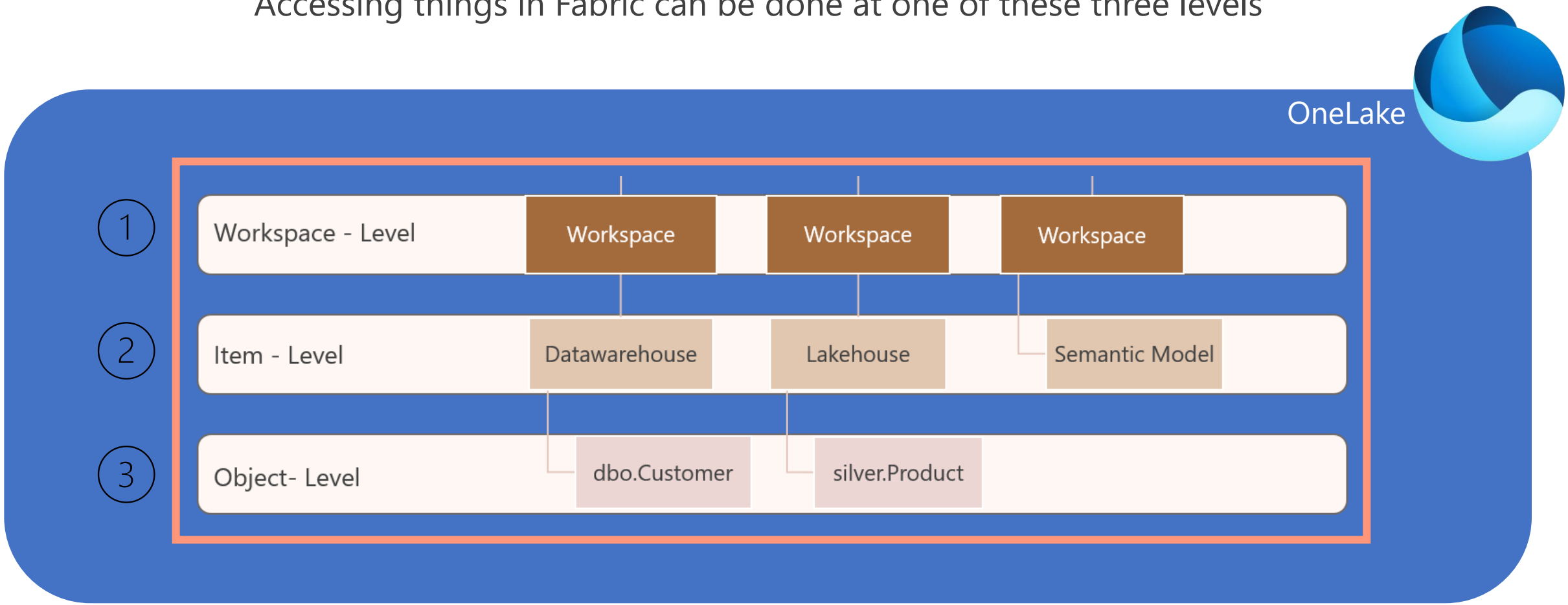
- Workspace – Level Access
- Item – Level Access
- Object – Level Access
- OneLake Role-Based-Access-control (RBAC)
- Takeaways

Fabric Hierarchy



Fabric Hierarchy

Accessing things in Fabric can be done at one of these three levels



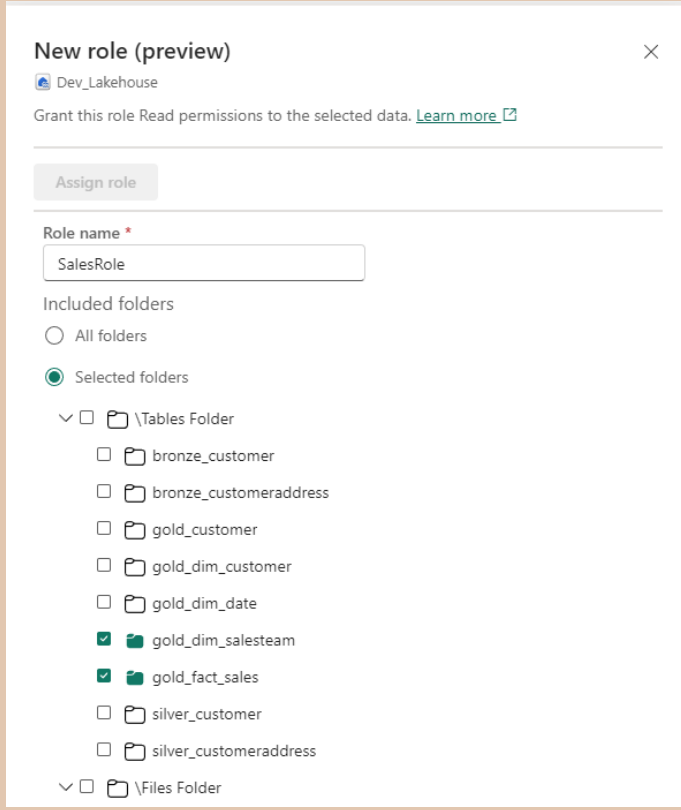
Fabric Hierarchy

Accessing the underlaying files of a Lakehouse can be done through RBAC

OneLake

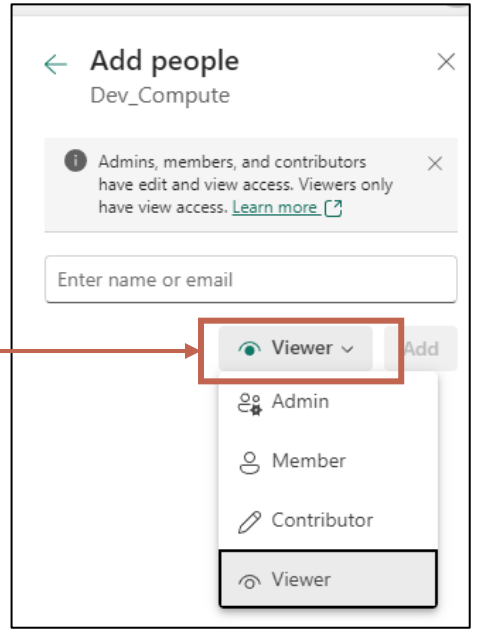
4

Lakehouse



Workspace – Level Access

- User or security groups can be given Workspace-level access
- When giving access, the person or group is assigned a workspace role:
  - **Admin**
  - **Member**
  - **Contributor**
  - **Viewer**
- This role applies to all items in the workspace. For example, a *Viewer* in the workspace will be able to view all items in the workspace, not only specific items.



Workspace – Level Roles

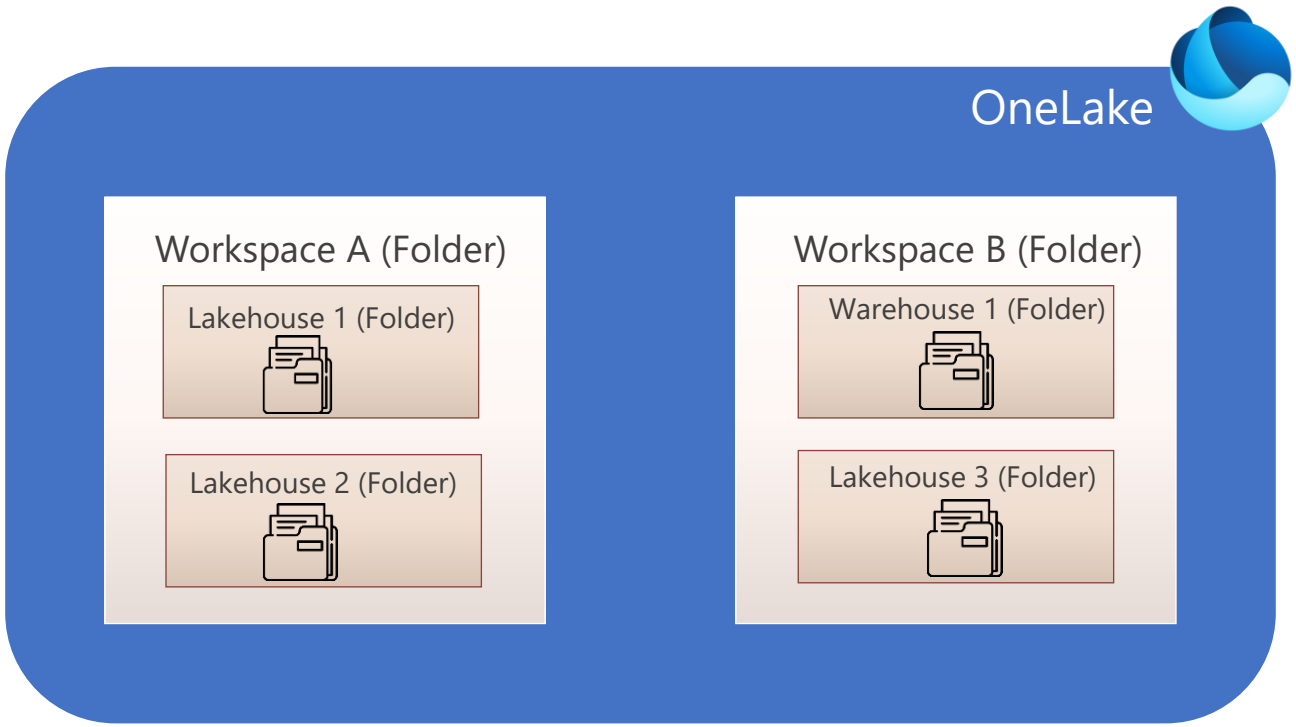
Capability	Admin	Member	Contributor	Viewer
Update and delete the workspace.	✓			
Add or remove people, including other admins.	✓			
Add members or others with lower permissions.	✓	✓		
Allow others to reshare items.[1]	✓	✓		
Create or modify database mirroring items.	✓	✓		
Create or modify warehouse items.	✓	✓		
Create or modify SQL database items.	✓	✓		
View and read content of data pipelines, notebooks, Spark job definitions, ML models and experiments, and eventstreams.	✓	✓	✓	✓
View and read content of KQL databases, KQL query-sets, and real-time dashboards.	✓	✓	✓	✓
Connect to SQL analytics endpoint of Lakehouse or the Warehouse	✓	✓	✓	✓
Read Lakehouse and Data warehouse data and shortcuts[2] with T-SQL through TDS endpoint.	✓	✓	✓	✓
Read Lakehouse and Data warehouse data and shortcuts[2] through OneLake APIs and Spark.	✓	✓	✓	
Read Lakehouse data through Lakehouse explorer.	✓	✓	✓	
Write or delete data pipelines, notebooks, Spark job definitions, ML models, and experiments, and eventstreams.	✓	✓	✓	
Write or delete Eventhouses[3], KQL Querysets, Real-Time Dashboards, and schema and data of KQL Databases, Lakehouses, data warehouses, and shortcuts.	✓	✓	✓	
Execute or cancel execution of notebooks, Spark job definitions, ML models, and experiments.	✓	✓	✓	
Execute or cancel execution of data pipelines.	✓	✓	✓	
View execution output of data pipelines, notebooks, ML models and experiments.	✓	✓	✓	✓
Schedule data refreshes via the on-premises gateway.[4]	✓	✓	✓	
Modify gateway connection settings.[4]	✓	✓	✓	

https://learn.microsoft.com/en-us/fabric/get-started/roles-workspaces

ALWAYS apply the principle of least privilege



Workspace – Level Access





Admin



Member



Contributor

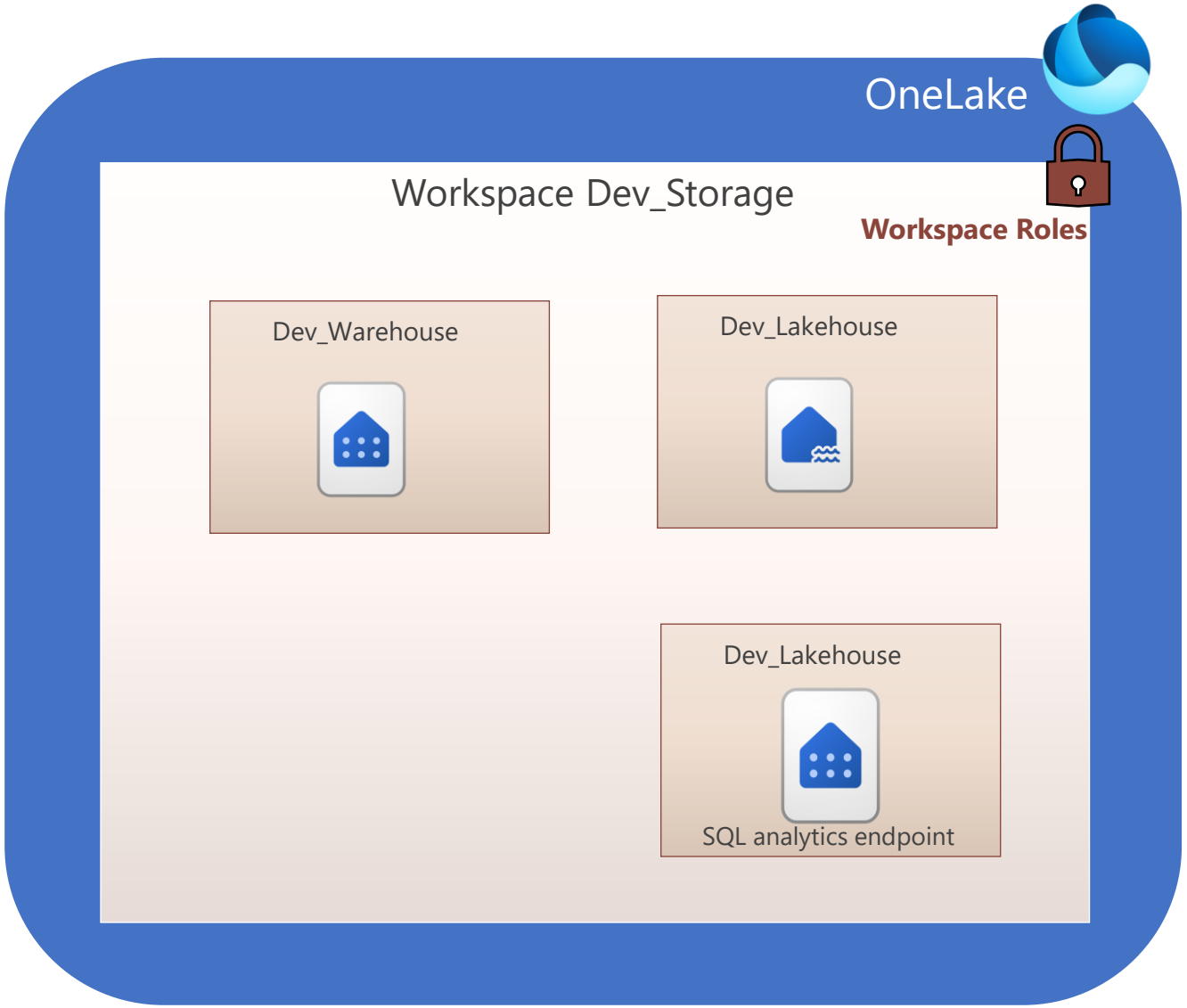
Full Access

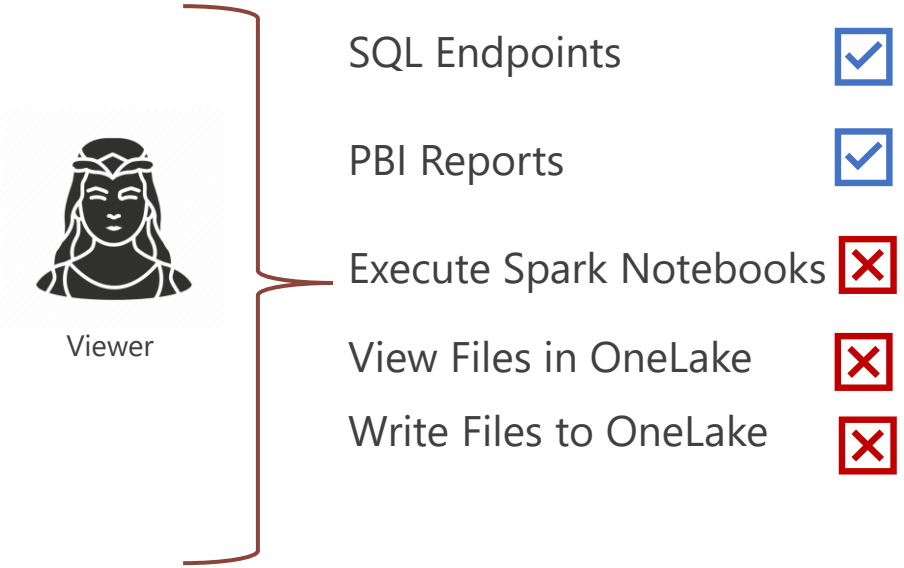


Viewer

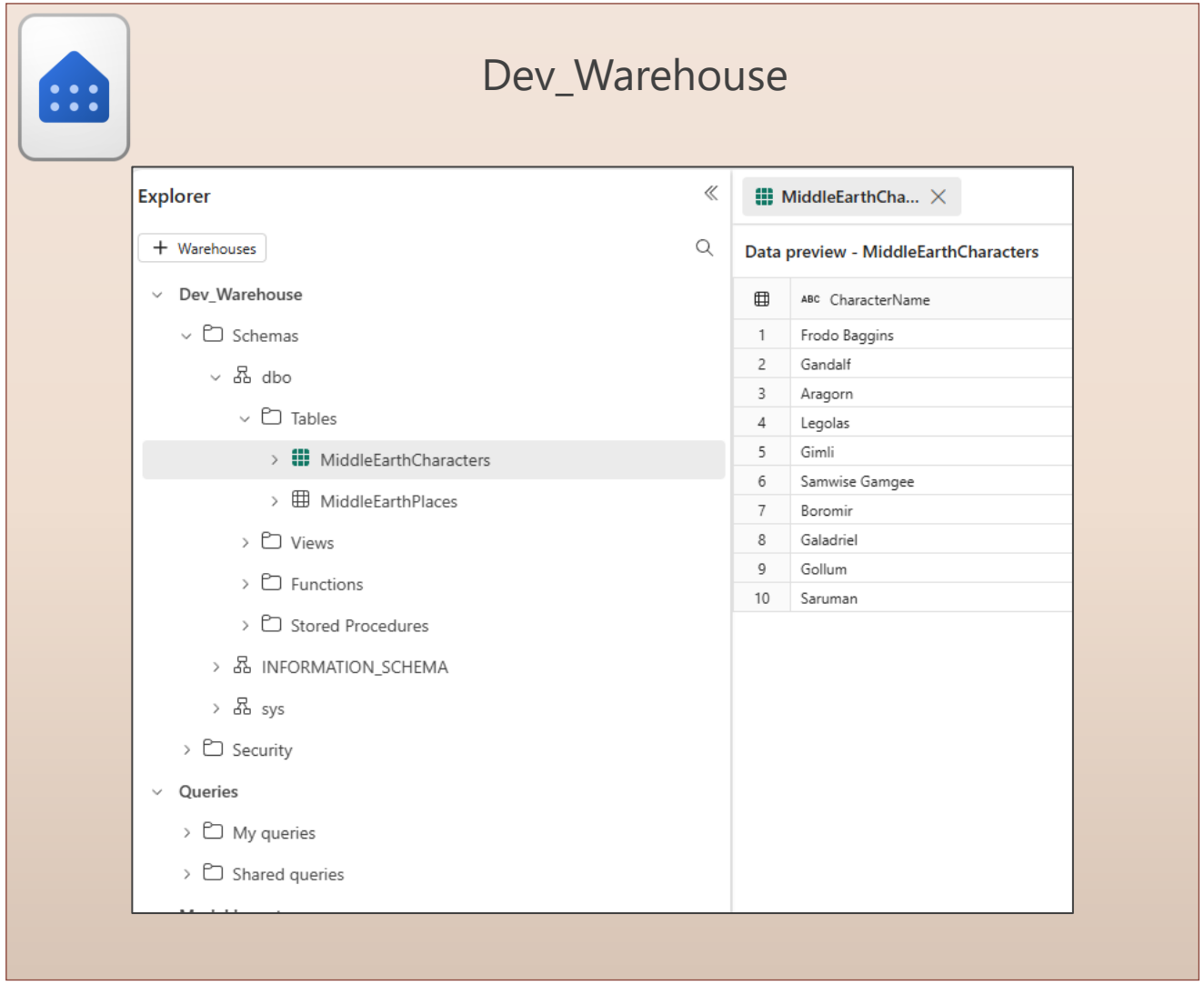
- SQL Endpoints ☑
- PBI Reports ☑
- Execute Spark Notebooks ☒
- View Files in OneLake ☒
- Write Files to OneLake ☒

Workspace – Level Access



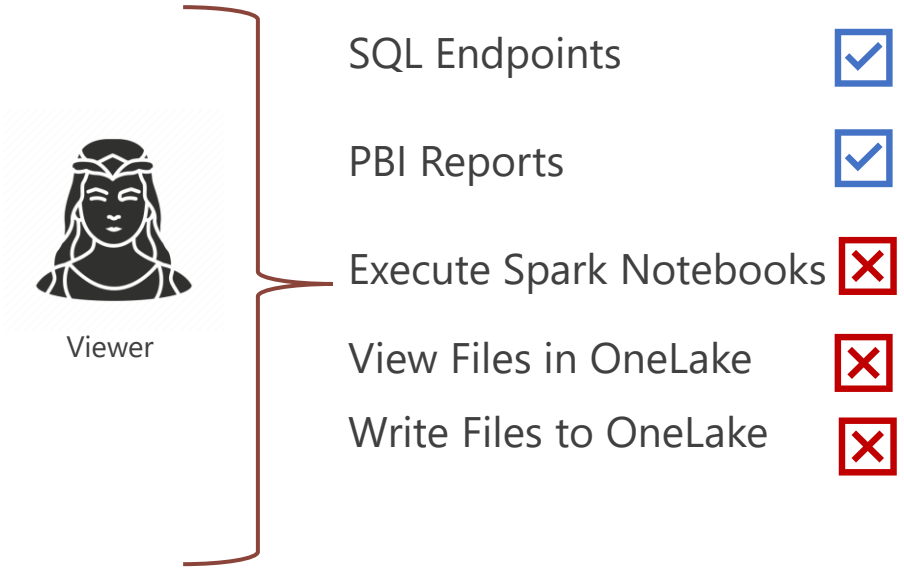


Workspace – Level Access

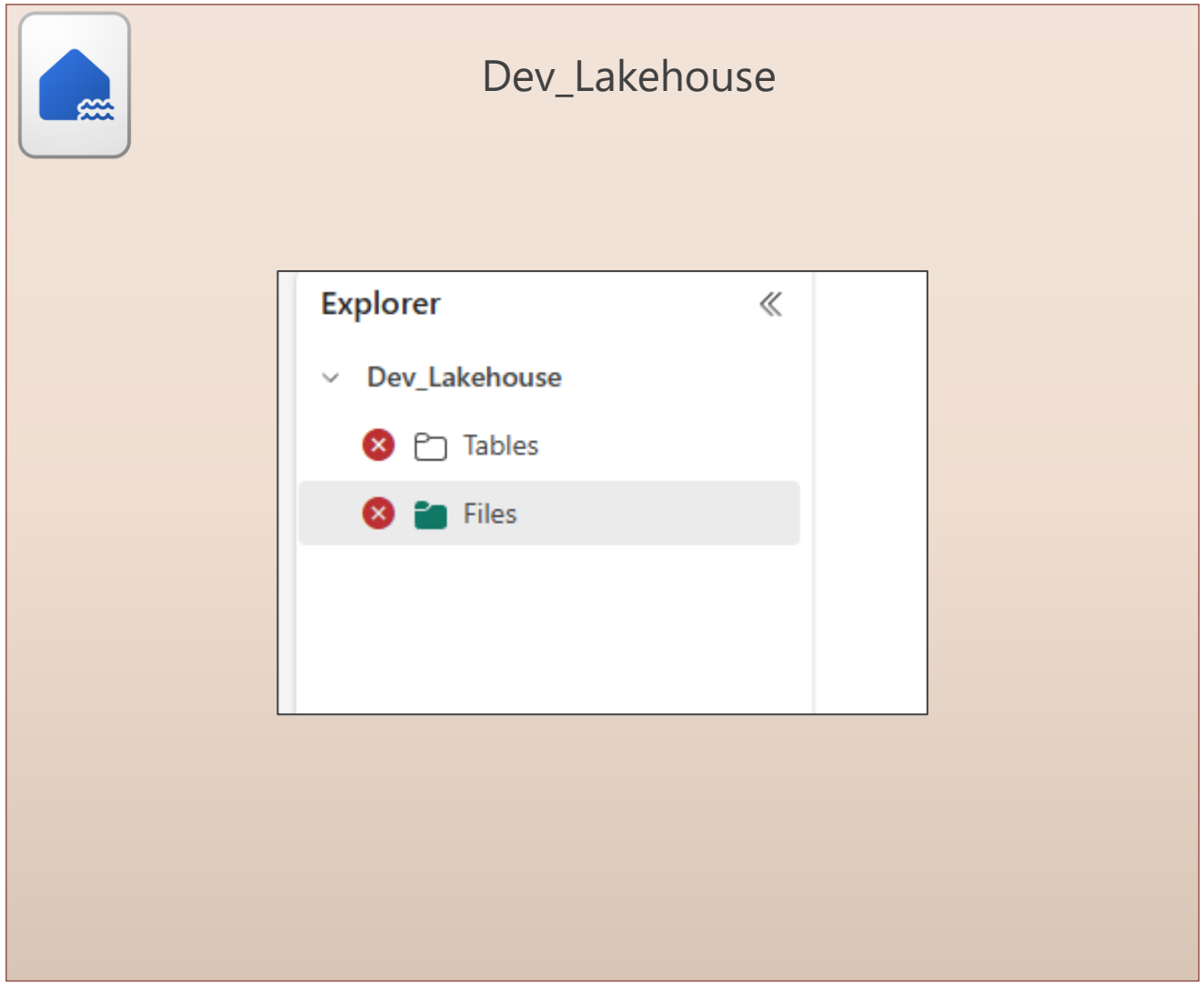


- Viewer can read all the data with *Viewer* role
- Only read, no DDL or DML statements

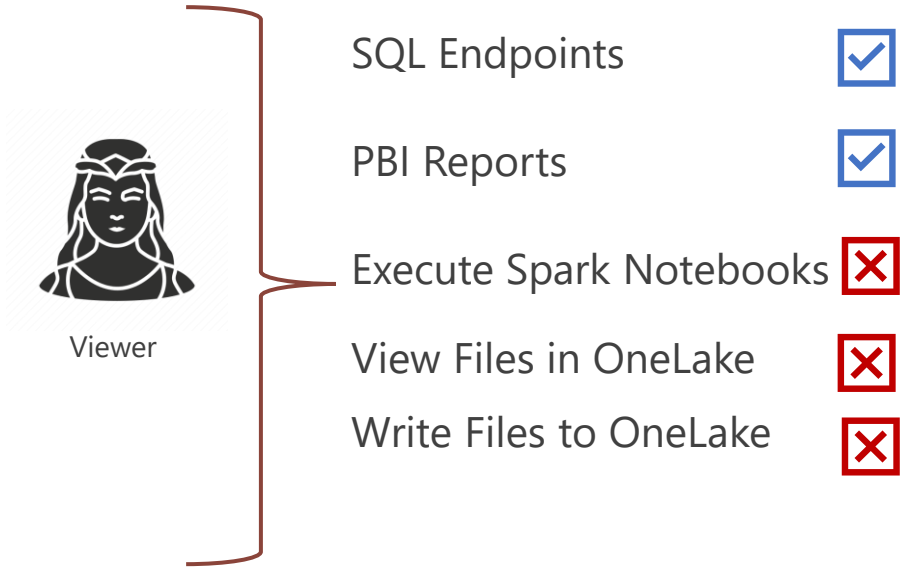
>> The same applies to the SQL analytics endpoint of the Lakehouse



Workspace – Level Access

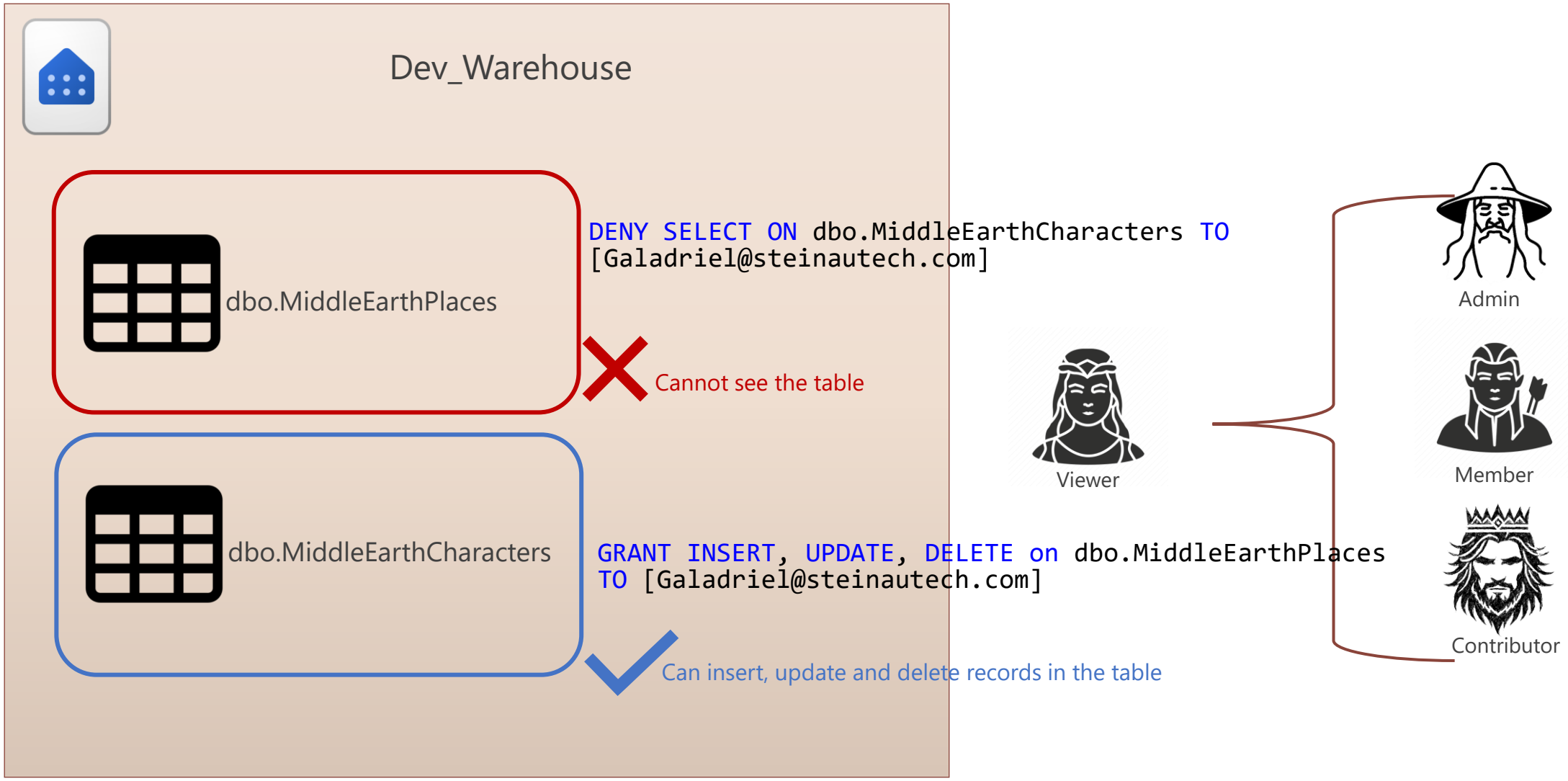


- *Viewer* cannot read any tables or view any underlying files in the Lakehouse



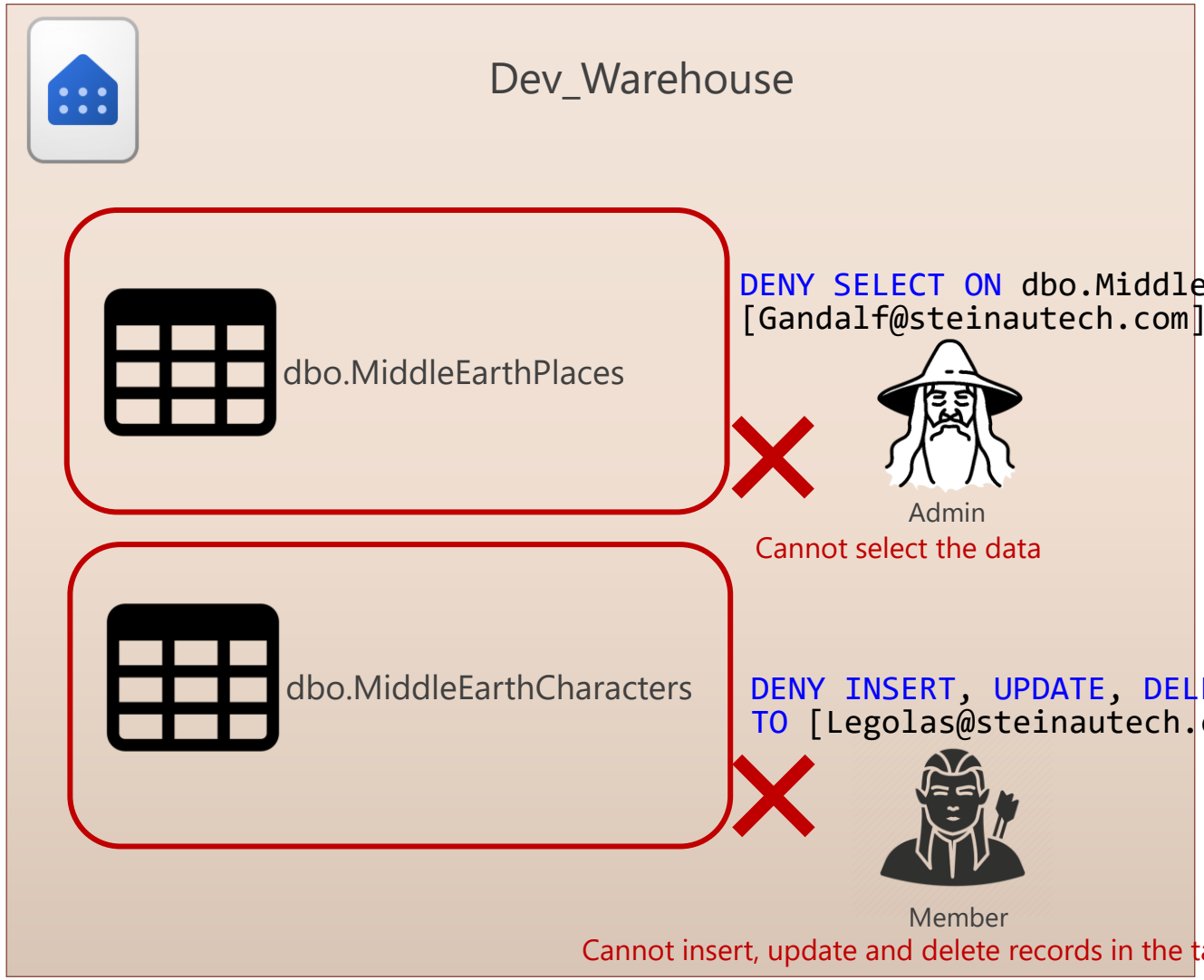
Workspace – Level Access

What if I want to control the access to items in a more fine-grained level **in the Warehouse**?

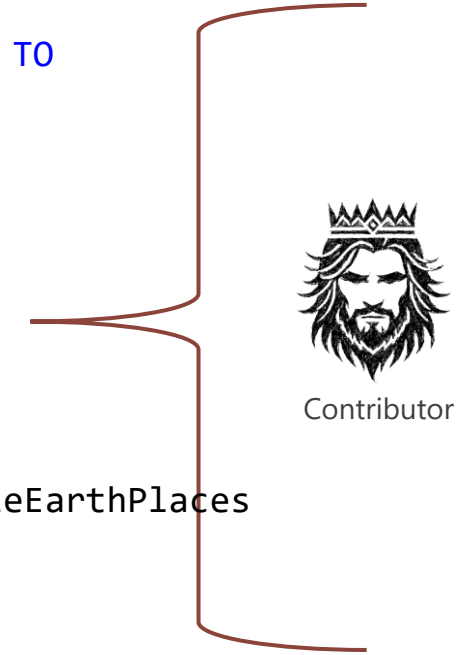


Workspace – Level Access

Very interesting…but beware!

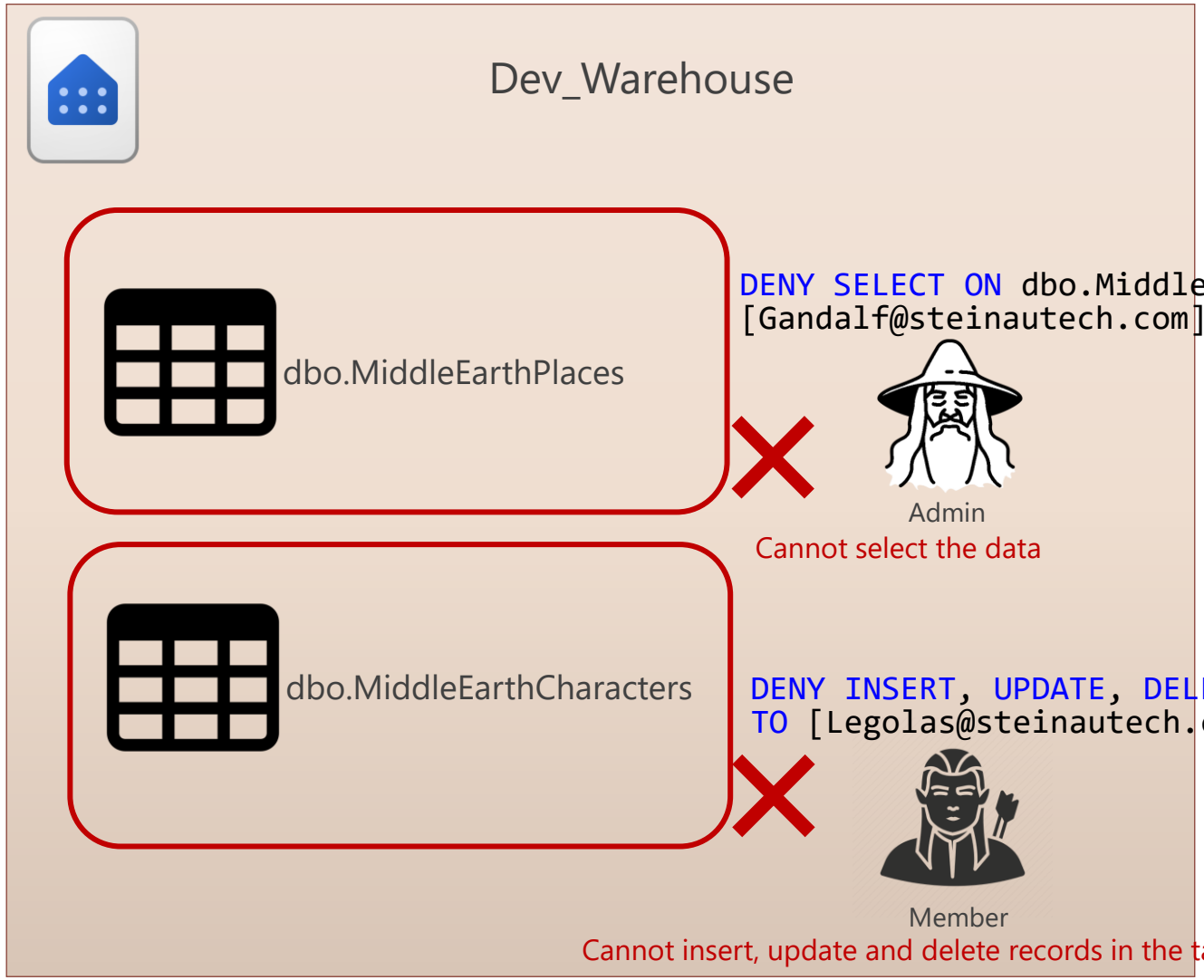


- The *Contributor* can restrict access and operations on objects to *Admins*, *Members* and other *Contributors*
- The user with restricted access cannot grant, deny, or revoke permissions to himself



Workspace – Level Access

Very interesting…but beware!



- The *Contributor* can restrict access and operations on objects to *Admins*, *Members* and other *Contributors*
- The user with restricted access cannot grant, deny, or revoke permissions to himself

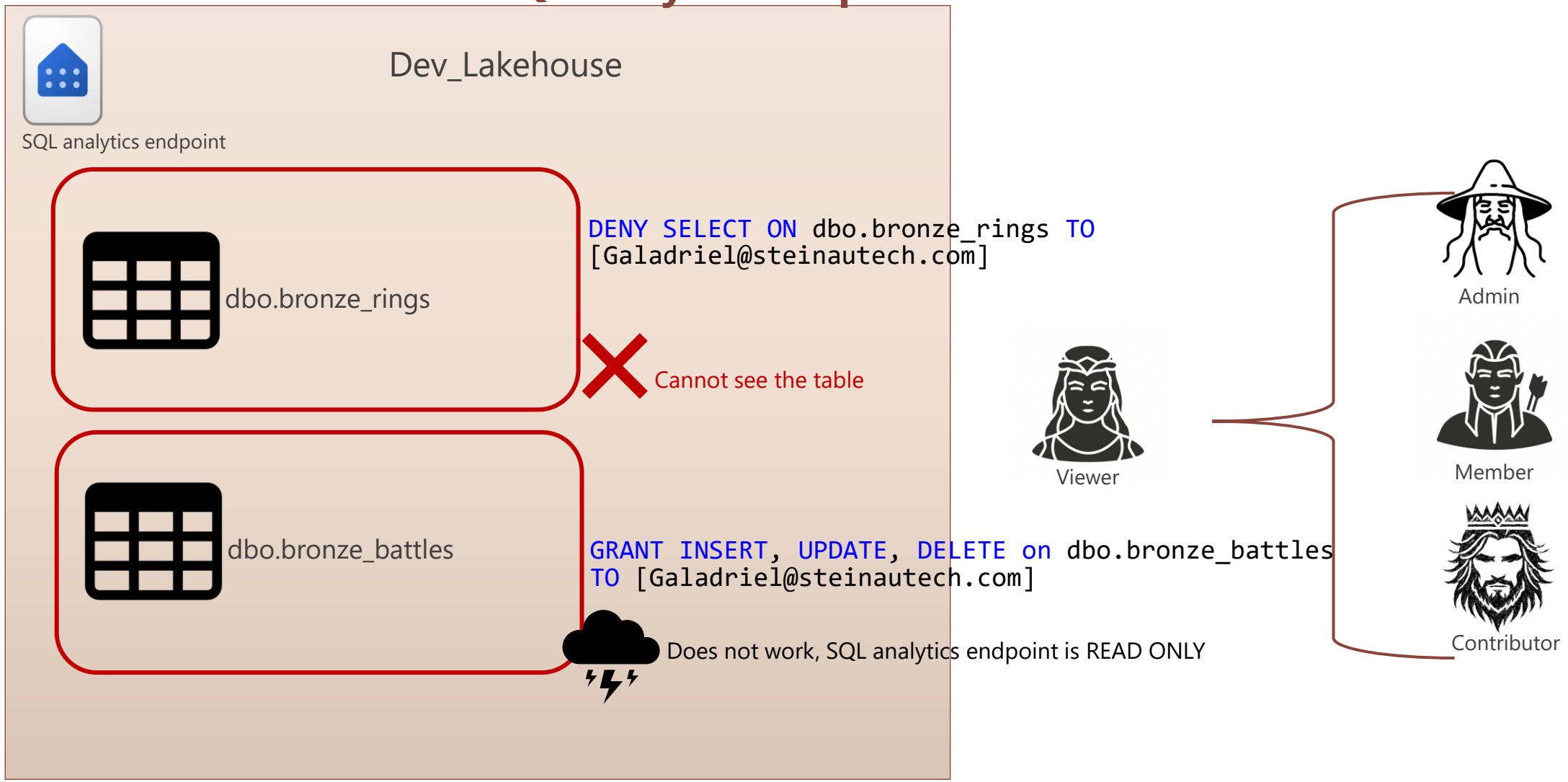


Contributor

- When using Workspaces roles combined with GRANT / DENY permissions, the later take precedence independently of the Workspace role hierarchy!
- Does not apply for the *Viewer* role

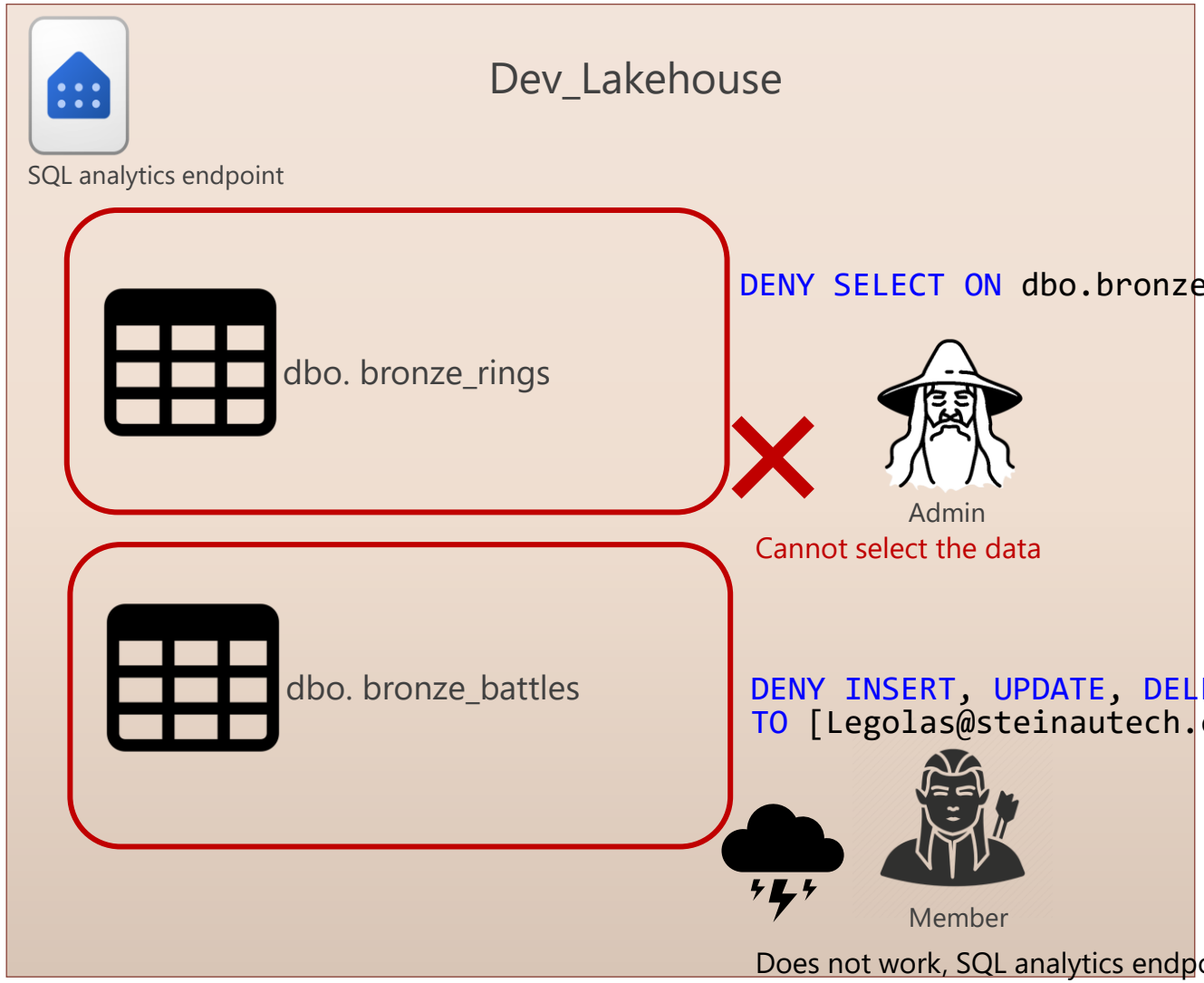
Workspace – Level Access

What if I want to control the access to items in a more fine granular level **in the SQL analytics endpoint of the Lakehouse**?

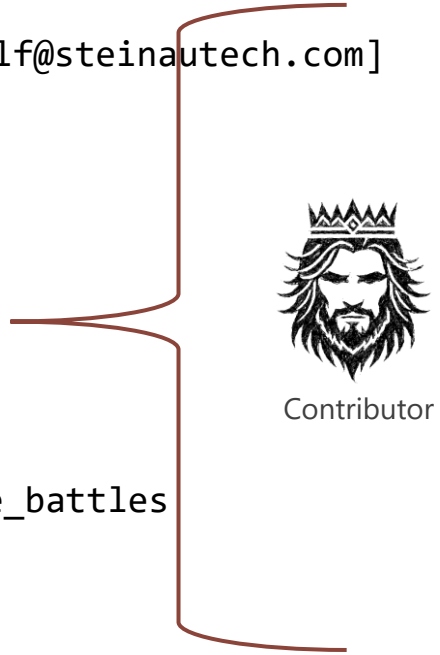


Workspace – Level Access

Very interesting…but beware!

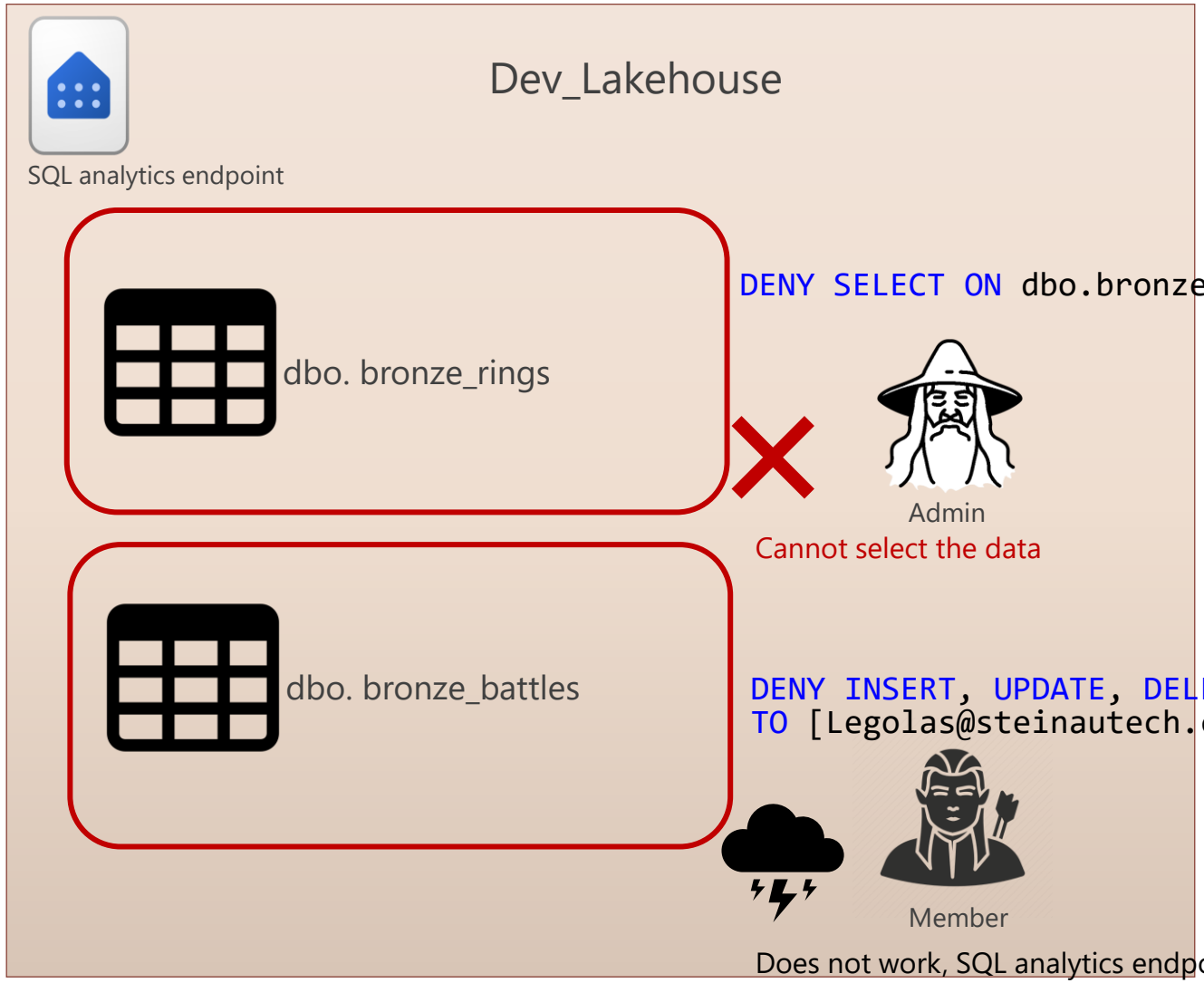


- The *Contributor* can restrict access on tables to *Admins Members* and other *Contributors*
- The user with restricted access cannot grant, deny, or revoke permissions to himself



Workspace – Level Access

Very interesting…but beware!



- The *Contributor* can restrict access on tables to *Admins Members* and other *Contributors*
- The user with restricted access cannot grant, deny, or revoke permissions to himself



Contributor

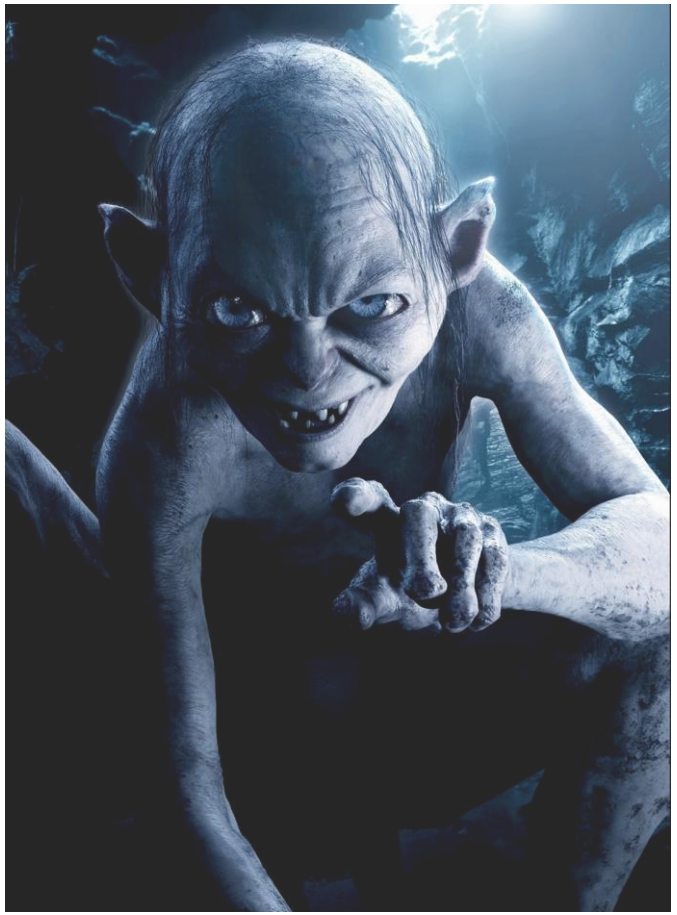
- SQL analytics endpoint is READ ONLY
- Only possible to `DENY SELECT ON` table
- No Workspace role can execute any DML statement

Workspace – Level Access

Remember the tiny little difference between the SQL analytics endpoint of the Lakehouse and the Lakehouse….?



SQL analytics endpoint





Lakehouse

Workspace – Level Access

`DENY SELECT ON` table only applies to SQL analytics endpoint!



Dev_Lakehouse

SQL analytics endpoint

dbo. bronze_rings

Cannot select the data



Admin



Member



Contributor

Dev_Lakehouse

dbo. bronze_rings

Can select and insert, update and delete records in the table through Spark

```
DENY SELECT ON dbo.bronze_rings TO
[Gandalf@steinautech.com], [Legolas@steinautech.com],
[Aragorn@steinautech.com]
```

How do I find out who has access to a workspace?

Fabric REST API Request

```
home/steini> az rest `
    --method GET `
    --url https://api.fabric.microsoft.com/v1/admin/workspaces/eef0ef80-d2cd-4f16-81de-393e15831352/users `
    --resource https://api.fabric.microsoft.com
```

Workspace Id


```
{
  "accessDetails": [
    {
      "principal": {
        "displayName": "Galadriel",
        "id": "bd821275-8fec-406f-b42f-843bb58c920b",
        "type": "User",
        "userDetails": {
          "userPrincipalName": "Galadriel@steinautech.com"
        }
      },
      "workspaceAccessDetails": {
        "type": "Workspace",
        "workspaceRole": "Viewer"
      }
    },
    {
      "principal": {
        "displayName": "Legolas",
        "id": "970568e2-8228-4d23-a928-f23cc2c55d4d",
        "type": "User",
        "userDetails": {
          "userPrincipalName": "Legolas@steinautech.com"
        }
      },
      "workspaceAccessDetails": {
        "type": "Workspace",
        "workspaceRole": "Member"
      }
    },
    {
      "principal": {
        "displayName": "Gandalf",
        "id": "c0e2f954-1065-47a6-a782-1a8574f448b3",
        "type": "User",
        "userDetails": {
          "userPrincipalName": "Gandalf@steinautech.com"
```

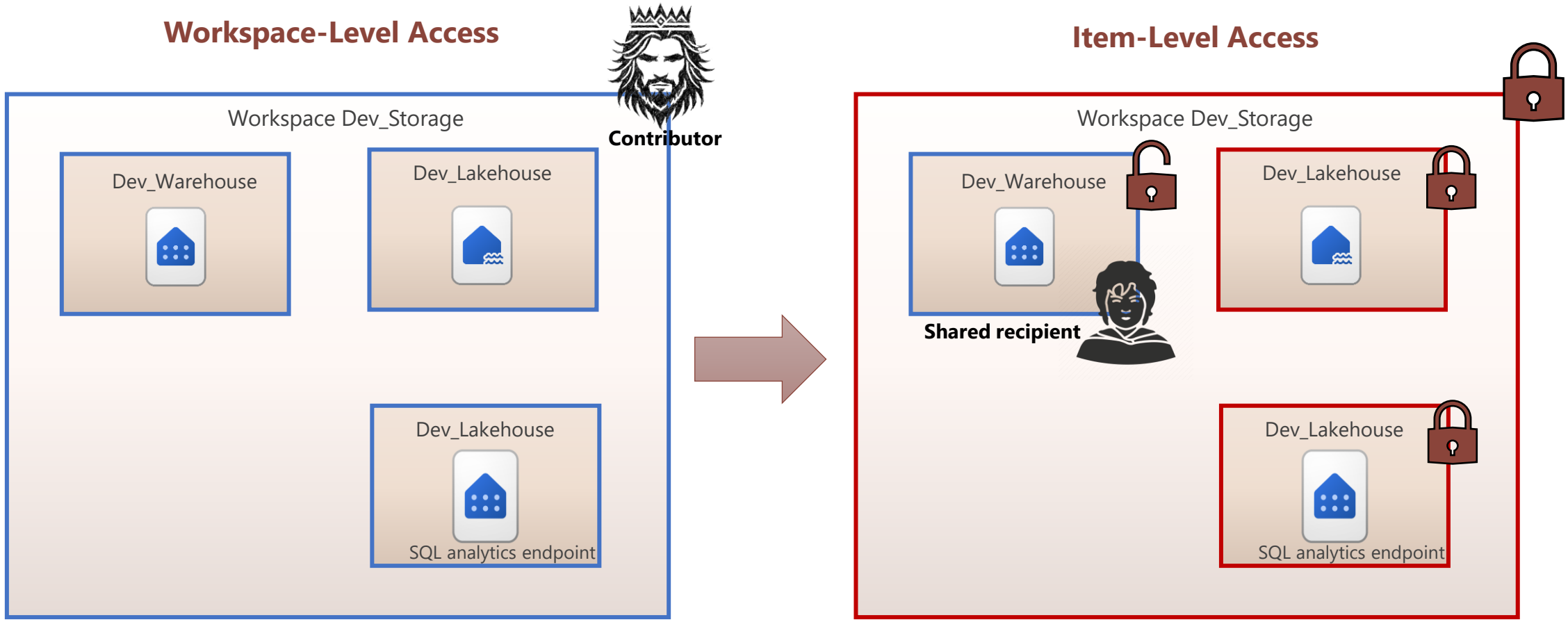
How do I verify the table access level in the Warehouse or SQL analytics endpoint of the Lakehouse?

```
SELECT
    princ.name AS UserName,
    princ.type_desc,
    perm.permission_name,
    perm.state_desc AS PermissionState,
    perm.class_desc,
    obj.name AS ObjectName
FROM
    sys.database_permissions perm
JOIN
    sys.database_principals princ ON
perm.grantee_principal_id = princ.principal_id
LEFT JOIN
    sys.objects obj ON perm.major_id = obj.object_id
WHERE
    princ.name = 'Galadriel@steinautech.com';
```

Item – Level Access

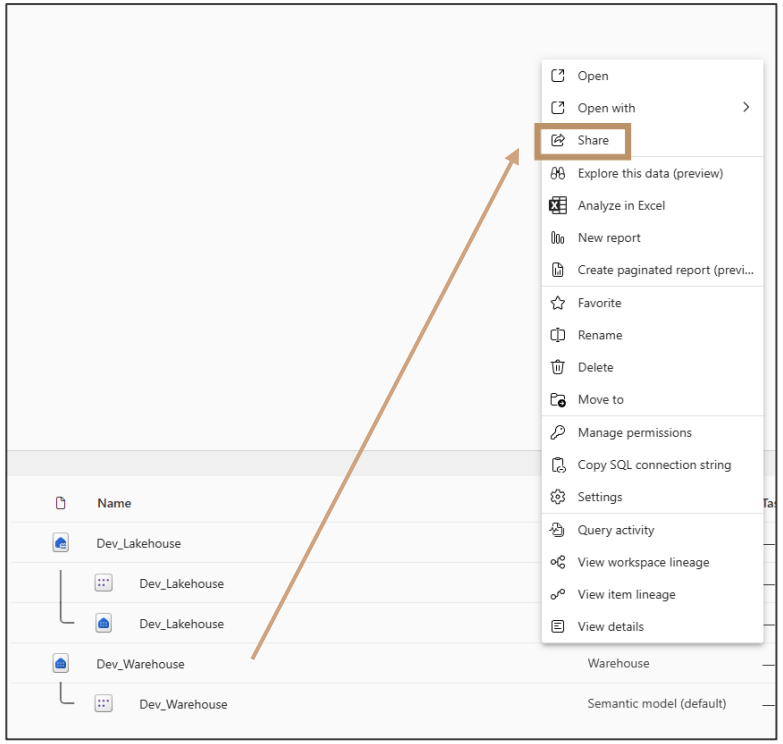
Has access

No access



Item – Level Access

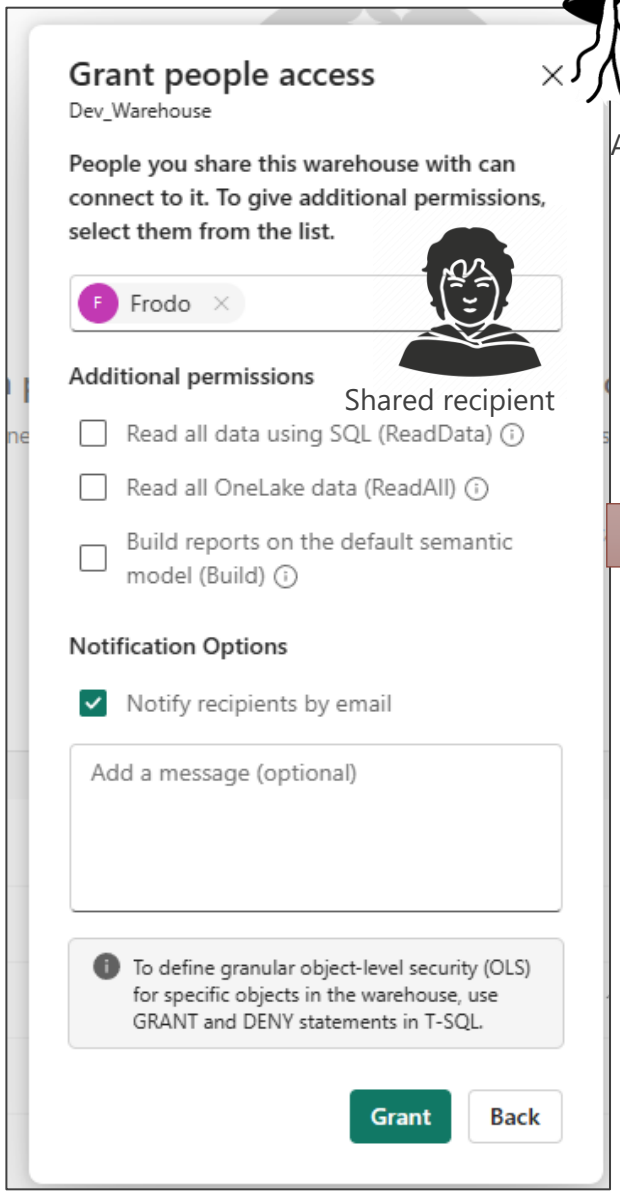
- User or security groups can be given Item-level access
- The user or security group can then access the item via the OneLake catalog.
- It is not possible to share all items, for example data pipelines and dataflows. The only option to share those items is to give access to the workspace.
- When sharing an item, there are different permissions that need to be taken into account. Those permissions vary a bit across items.
- You must be *Admin* or *Member* of the workspace to share an item.
- It is possible to reshare an item as a *Contributor* or *Viewer* if the user has the permission to reshare the item. Resharing is then possible up to the level of permission that the user resharing the item has.



Item – Level Access



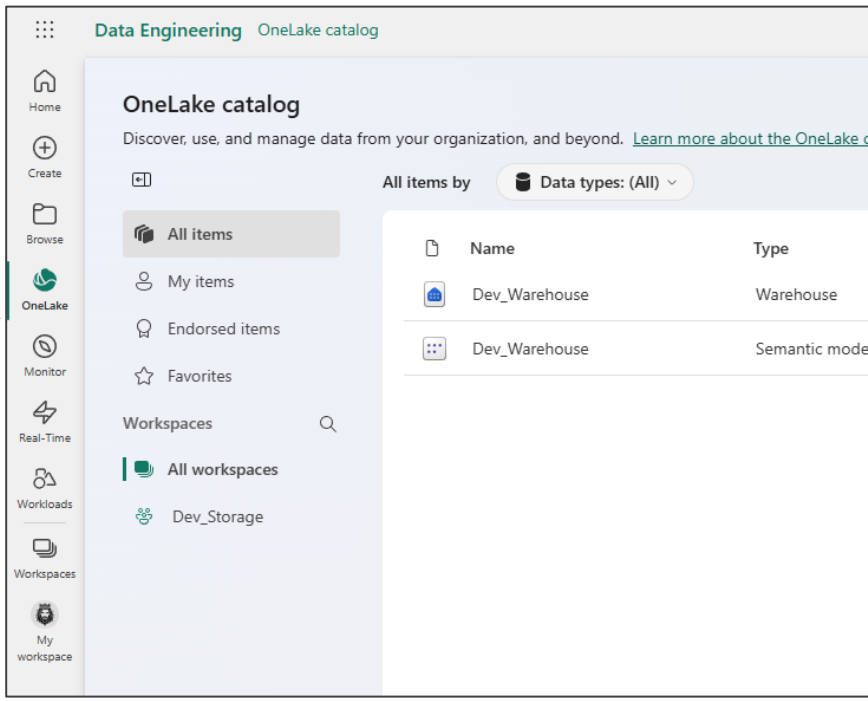
Admin



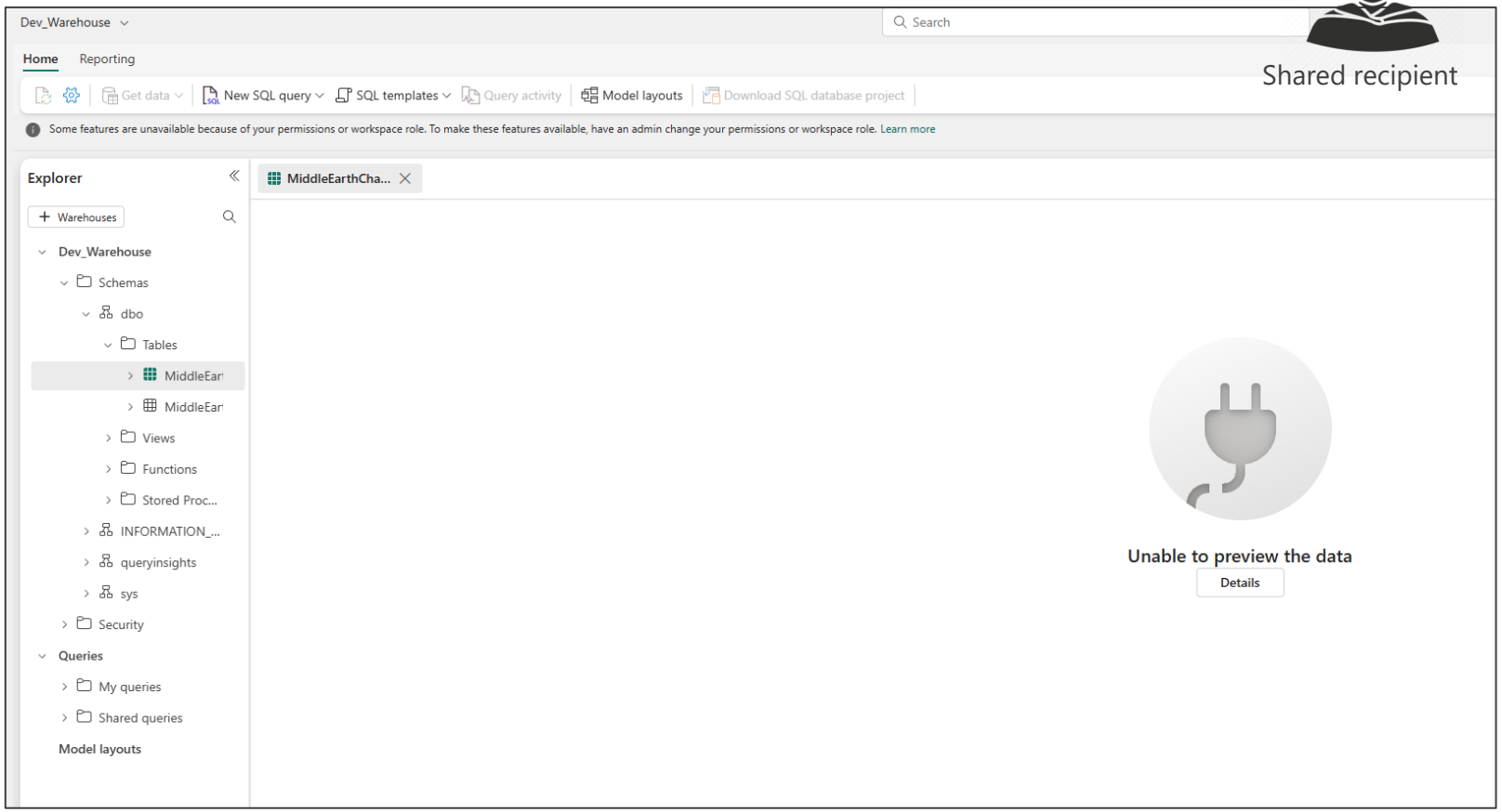
Shared recipient



Shared recipient



Item – Level Access





Shared recipient

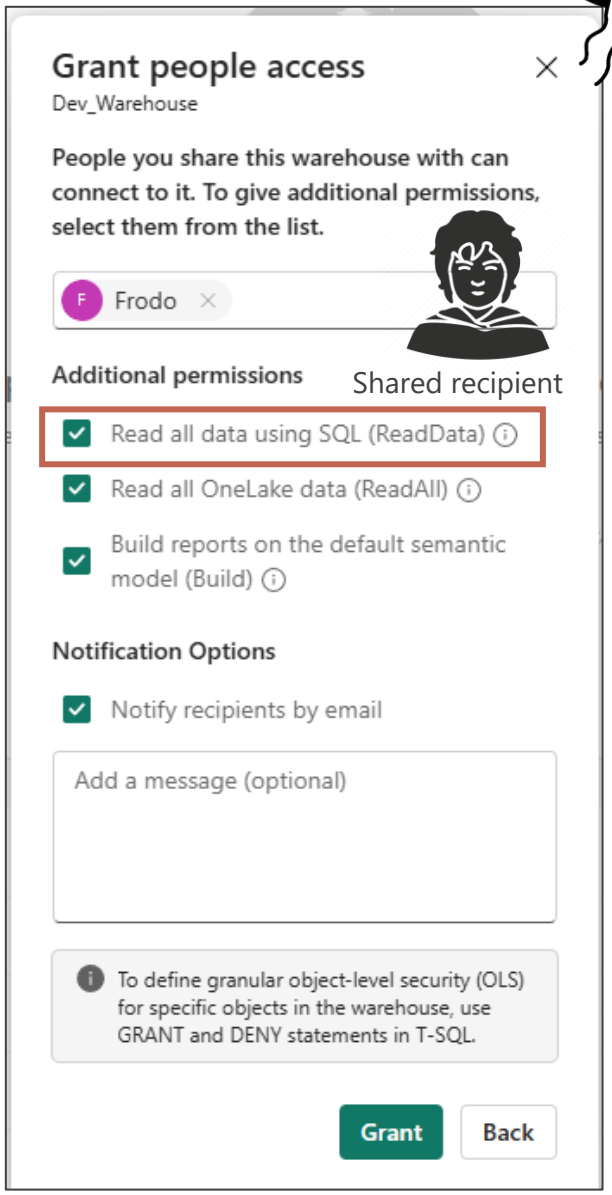
Without additional permissions: The shared recipient receives by default the Read permission. He can connect to the Warehouse but can't query any table or view or execute any function or stored procedure. This is the equivalent of CONNECT permission in SQL Server.

>> Access to objects within the Warehouse can be provided using T-SQL GRANT statement.

Item – Level Access



Admin

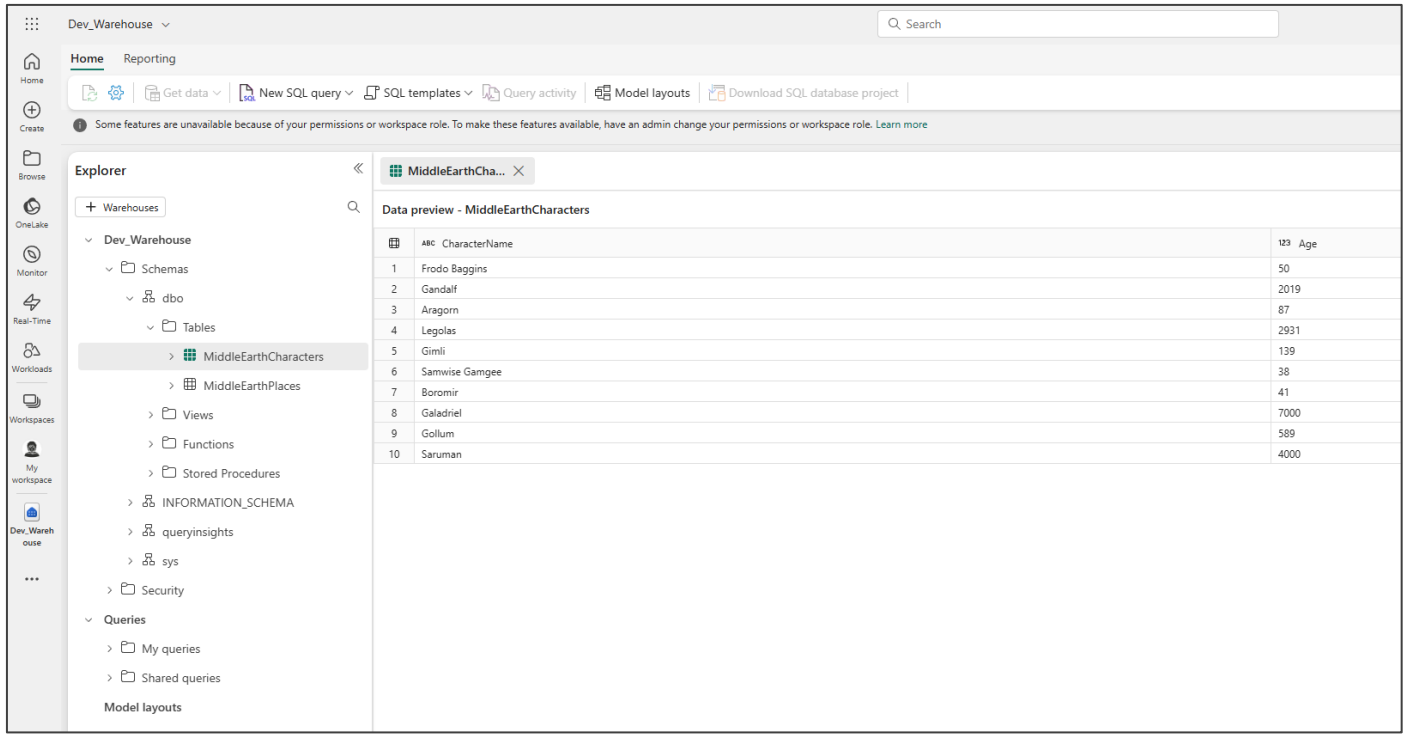


Shared recipient

- Read all data using SQL (ReadData): Shared recipient can read all objects within the Warehouse. Equivalent to *db_datareader* role in SQL Server.
- Further restriction and fine granular access to some objects within the Warehouse with T-SQL GRANT/REVOKE/DENY statements.



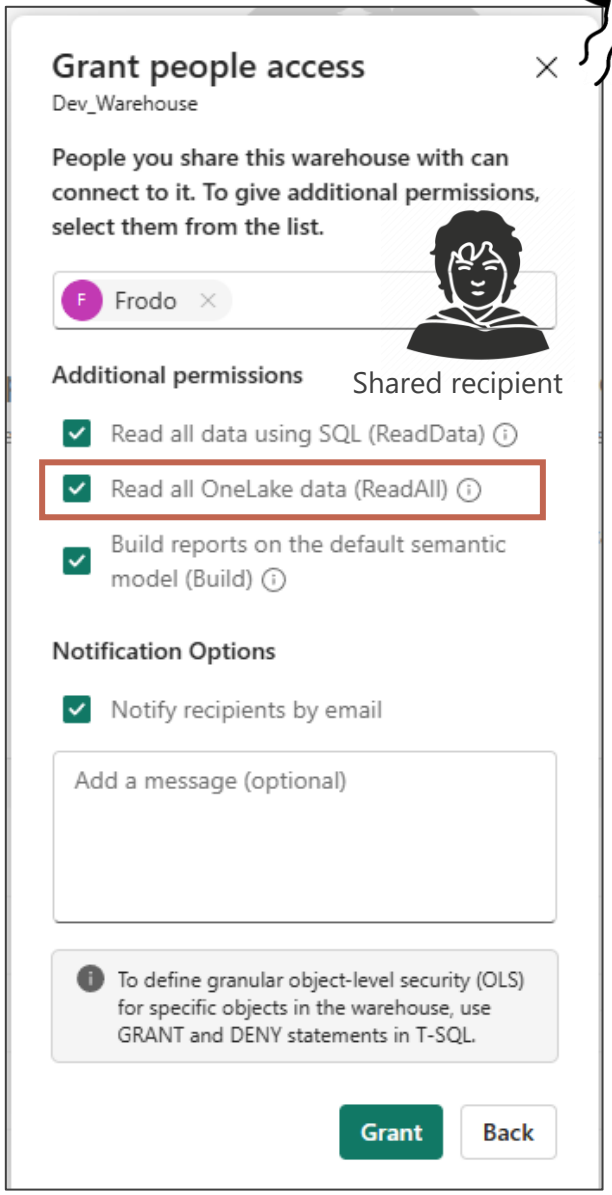
Shared recipient



Item – Level Access



Admin

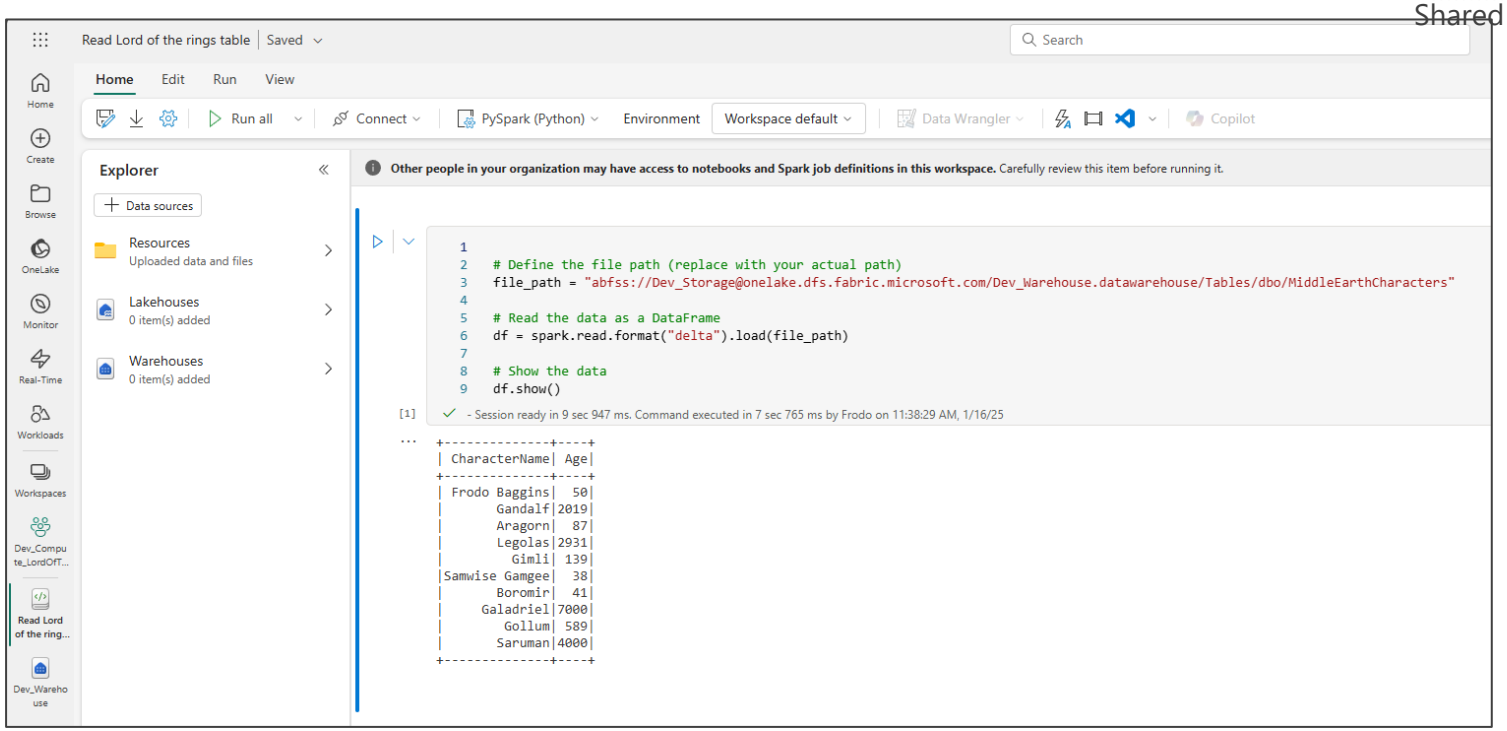


Shared recipient

- Read all data using Apache Spark (ReadAll): Shared recipient has read access to underlying files in OneLake and can consume them using Spark.
- ReadAll should be provided only if the shared recipient needs complete access to the warehouse's files using the Spark engine.



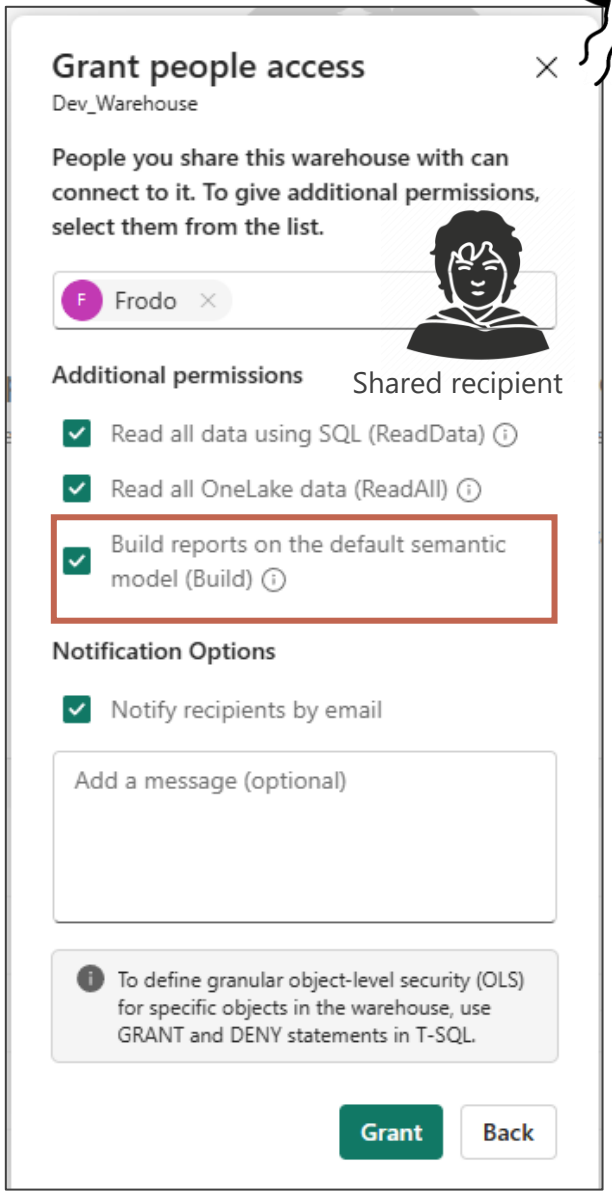
Shared recipient



Item – Level Access



Admin

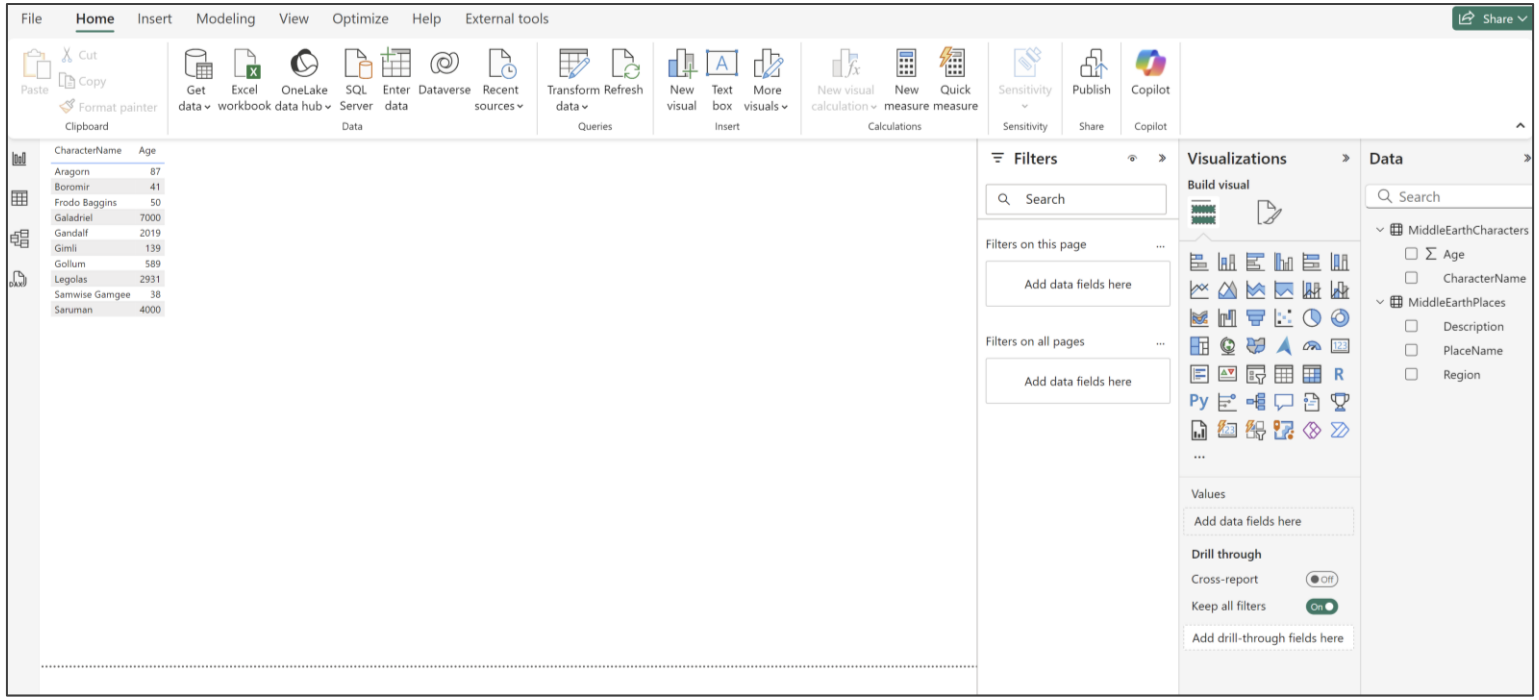


Shared recipient

- Build reports on the default semantic model (Build): Shared recipient can build reports on top of the default semantic model that is connected to the Warehouse.
- The Build checkbox is selected by default, but can be unchecked.



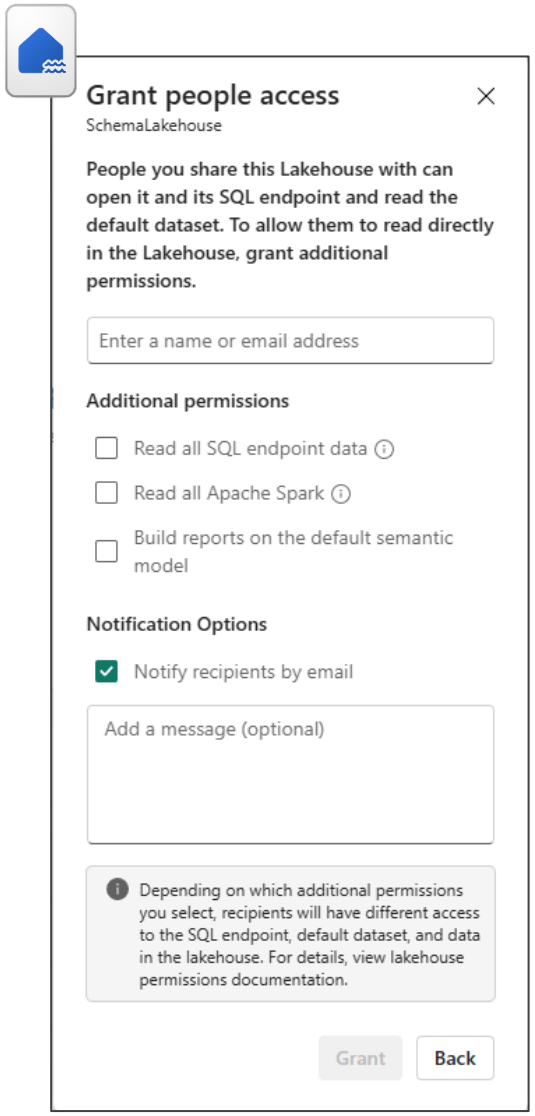
Shared recipient

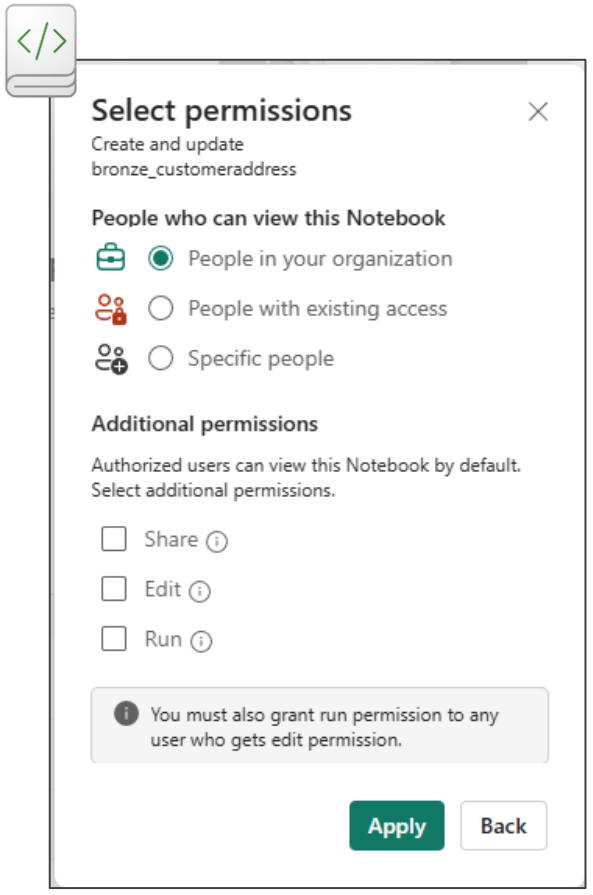


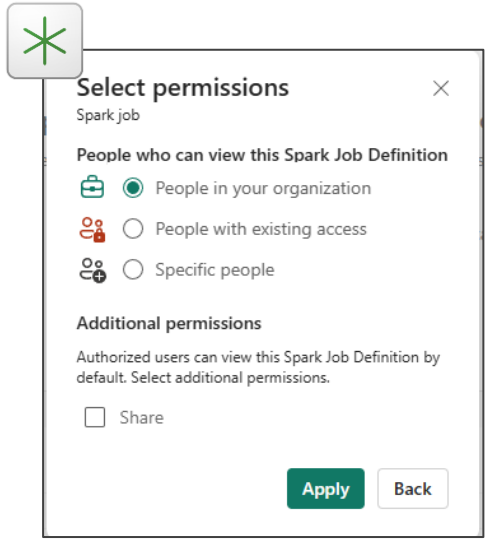
Item – Level Access

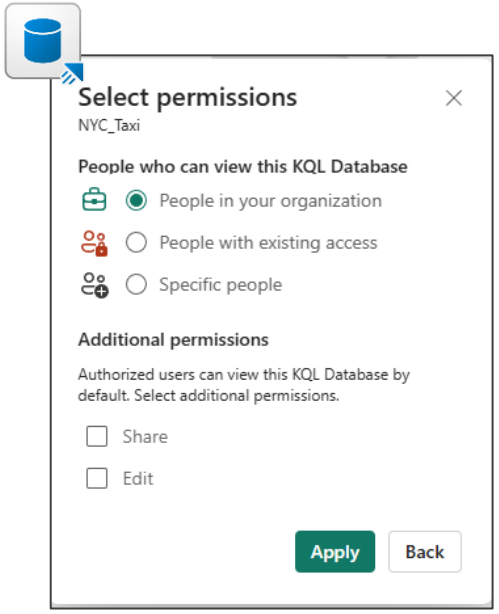
Items	Item-level access via share
Custom semantic model	Yes
Dataflow Gen2	No
Data pipeline	No
Default semantic model	No
Environment	Yes
Eventhouse	No
Eventstream	No
Experiments	Yes
KQL database	Yes
KQL queryset	Yes
Lakehouse	Yes
ML model	Yes
Notebook	Yes
Real time dashboard	Yes
Reflex	No
Power BI report	Yes
SQL analytics endpoint	No
Spark job	Yes
Warehouse	Yes

Item – Level Access





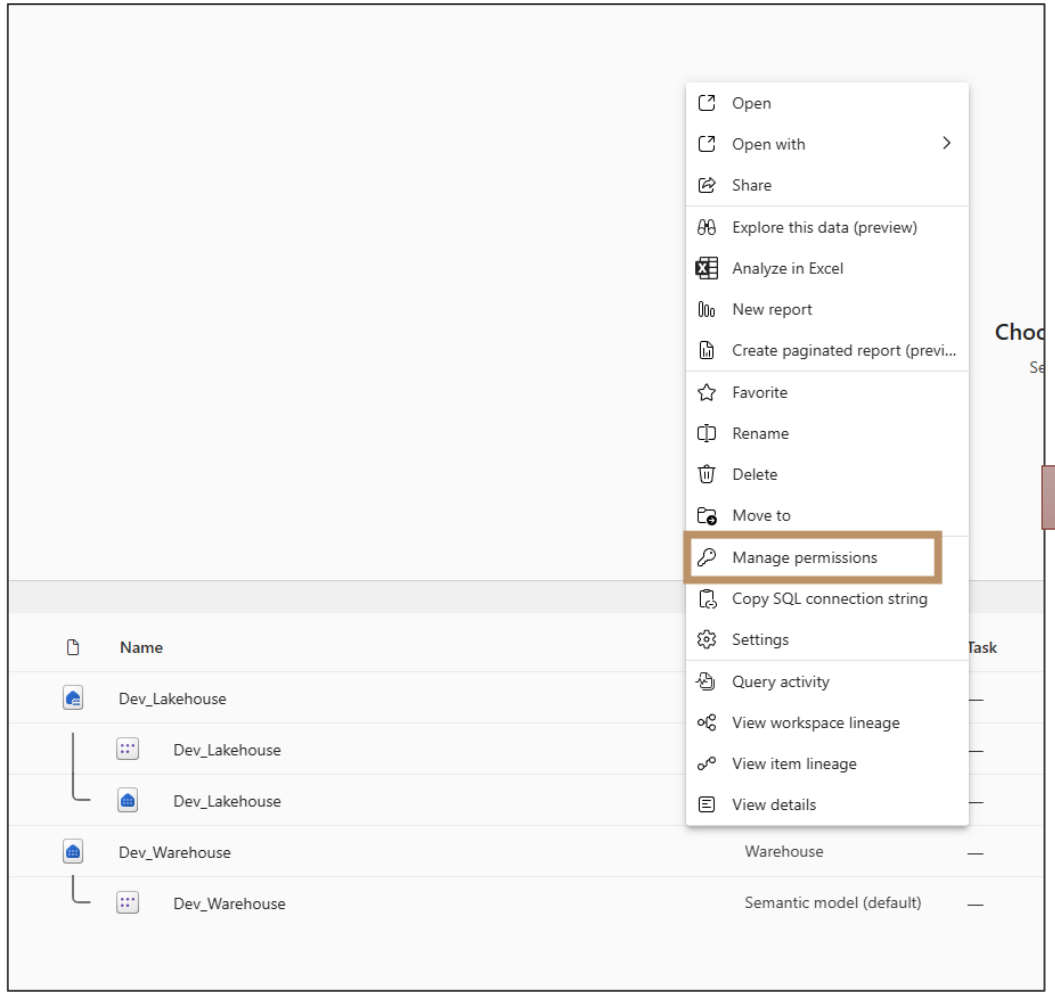




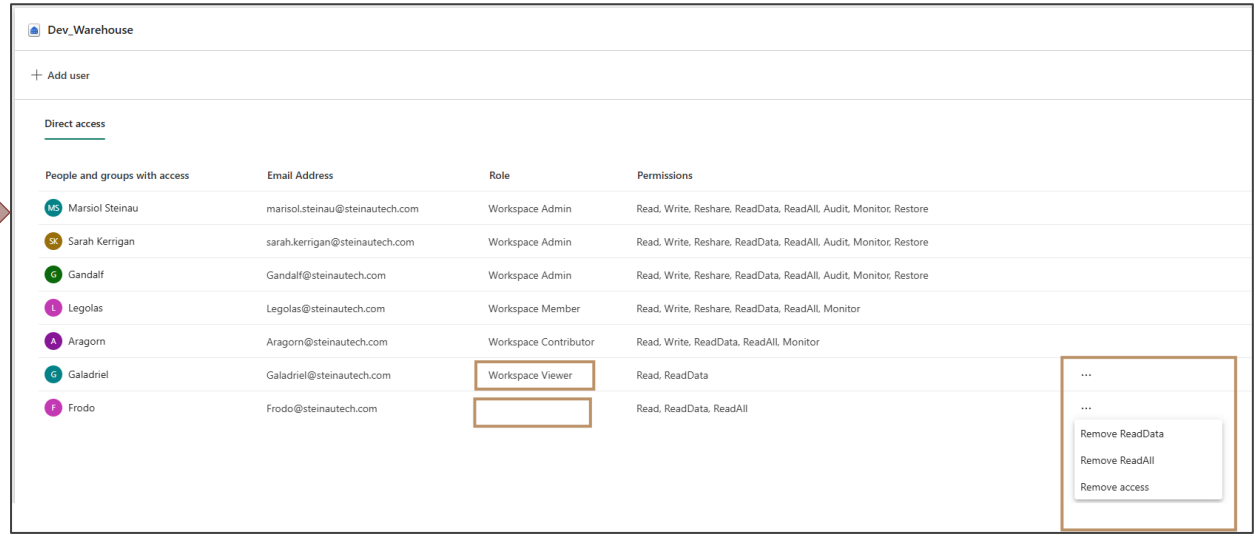
Item permission model

- Read all SQL endpoint data: Read data from the SQL analytics endpoint of the Lakehouse or Warehouse
- Read all Apache Spark: Read data from Lakehouse or Warehouse through OneLake APIs and Spark. Read data from Lakehouse through Lakehouse Explorer.
- Build: Build new content on the semantic model.
- Edit: Shared recipient can edit the item or its content.
- Share: Shared recipient can share the item and grant up to the permissions that they have.
- Run: Run or cancel execution of the item.

How do I find out who has access to an item?



- Admins or Members of a workspace can manage permissions
- For members of the workspace with the *Viewer* role or shared recipients, item permissions **Read**, **ReadData** and **ReadAll** can be provided



How do I find out who has access to an item?

```
az rest `
--method GET `
--url https://api.fabric.microsoft.com/v1/admin/workspaces/eef0ef80-d2cd-4f16-81de-393e15831352/items/75f9a53a-83c3-42f2-b653-cc2ed6d5d8a0/users `
--resource https://api.fabric.microsoft.com > accessDetails.json
```

Ho has

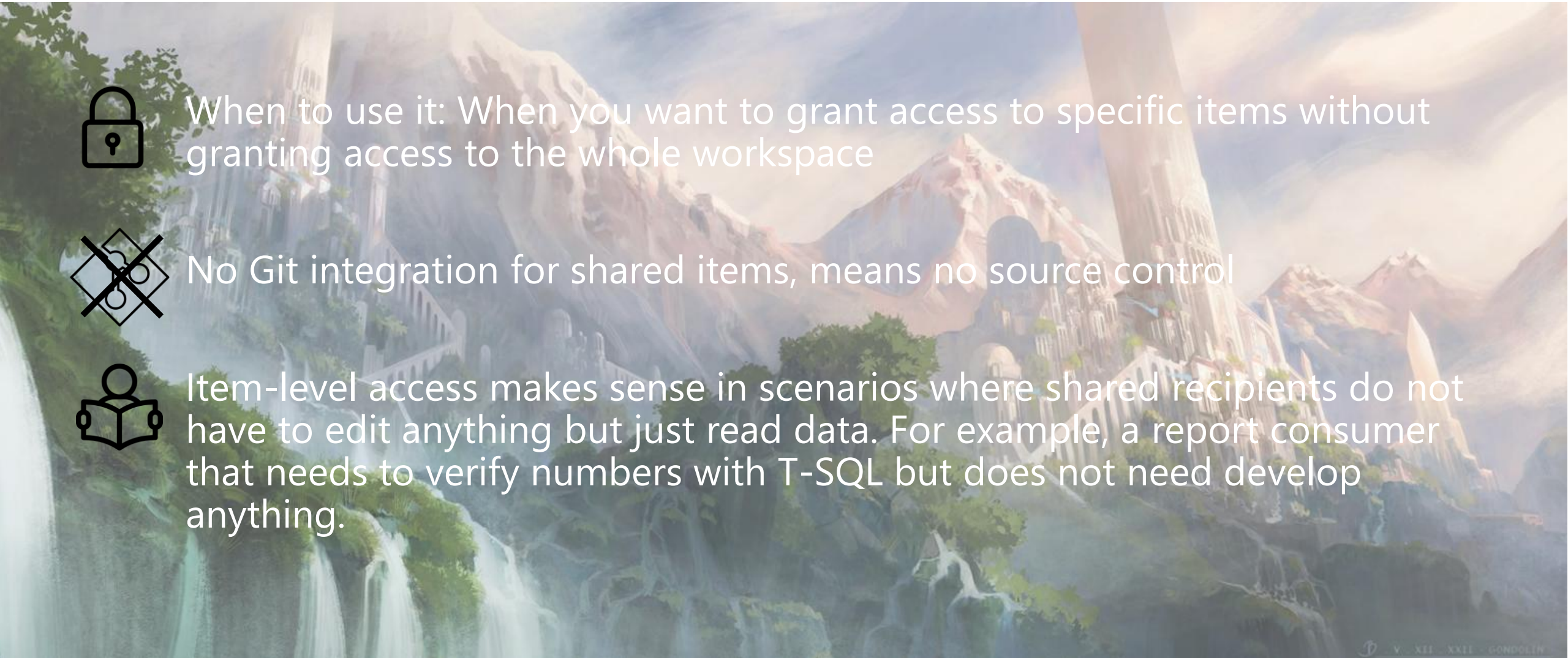

```
"accessDetails": [
  {
    "itemAccessDetails": {
      "additionalPermissions": null,
      "permissions": [
        "Read",
        "Write"
      ],
      "type": null
    },
    "principal": {
      "displayName": "Aragorn",
      "id": "227ed3ed-48ab-401f-9f3d-e660e670d996",
      "type": "User",
      "userDetails": {
        "userPrincipalName": "Aragorn@steinautech.com"
      }
    }
  },
  {
    "itemAccessDetails": {
      "additionalPermissions": null,
      "permissions": [
        "Read"
      ],
      "type": null
    },
    "principal": {
      "displayName": "Frodo",
      "id": "4c0ebfca-01e4-4e20-86bc-3e93669b1311",
      "type": "User",
      "userDetails": {
        "userPrincipalName": "Frodo@steinautech.com"
      }
    }
  },
```

```
"accessDetails": [
  {
    "itemAccessDetails": {
      "additionalPermissions": null,
      "permissions": [
        "Read"
      ],
      "type": null
    },
    "principal": {
      "displayName": "Galadriel",
      "id": "bd821275-8fec-406f-b42f-843bb58c920b",
      "type": "User",
      "userDetails": {
        "userPrincipalName": "Galadriel@steinautech.com"
      }
    }
  },
  {
    "itemAccessDetails": {
      "additionalPermissions": null,
      "permissions": [
        "Read",
        "Write",
        "ReShare"
      ],
      "type": null
    },
    "principal": {
      "displayName": "Legolas",
      "id": "970568e2-8228-4d23-a928-f23cc2c55d4d",
      "type": "User",
      "userDetails": {
        "userPrincipalName": "Legolas@steinautech.com"
      }
    }
  },
```

How do I verify the table access level in the Warehouse or SQL analytics endpoint of the Lakehouse?

```
SELECT
    princ.name AS UserName,
    princ.type_desc,
    perm.permission_name,
    perm.state_desc AS PermissionState,
    perm.class_desc,
    obj.name AS ObjectName
FROM
    sys.database_permissions perm
JOIN
    sys.database_principals princ ON
perm.grantee_principal_id = princ.principal_id
LEFT JOIN
    sys.objects obj ON perm.major_id = obj.object_id
WHERE
    princ.name = 'Galadriel@steinautech.com';
```

Item – Level Access
Things to consider



When to use it: When you want to grant access to specific items without granting access to the whole workspace

No Git integration for shared items, means no source control

Item-level access makes sense in scenarios where shared recipients do not have to edit anything but just read data. For example, a report consumer that needs to verify numbers with T-SQL but does not need develop anything.

Object – Level Access

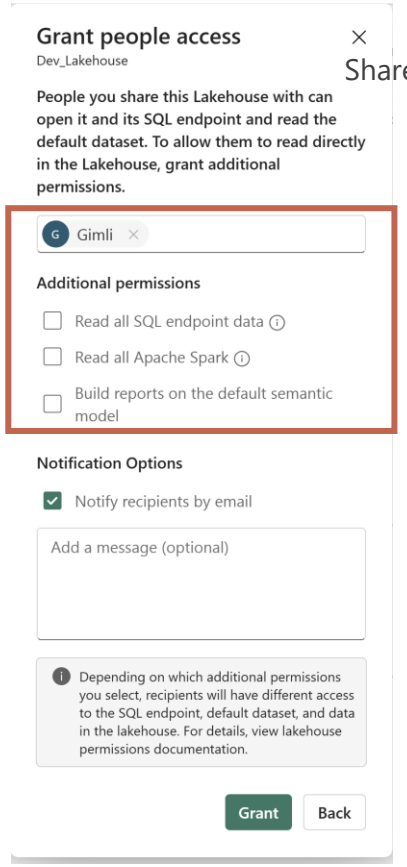
- Also known as engine level access
- Only possible in the Warehouse or SQL analytics endpoint of the Lakehouse



Admin



Shared recipient

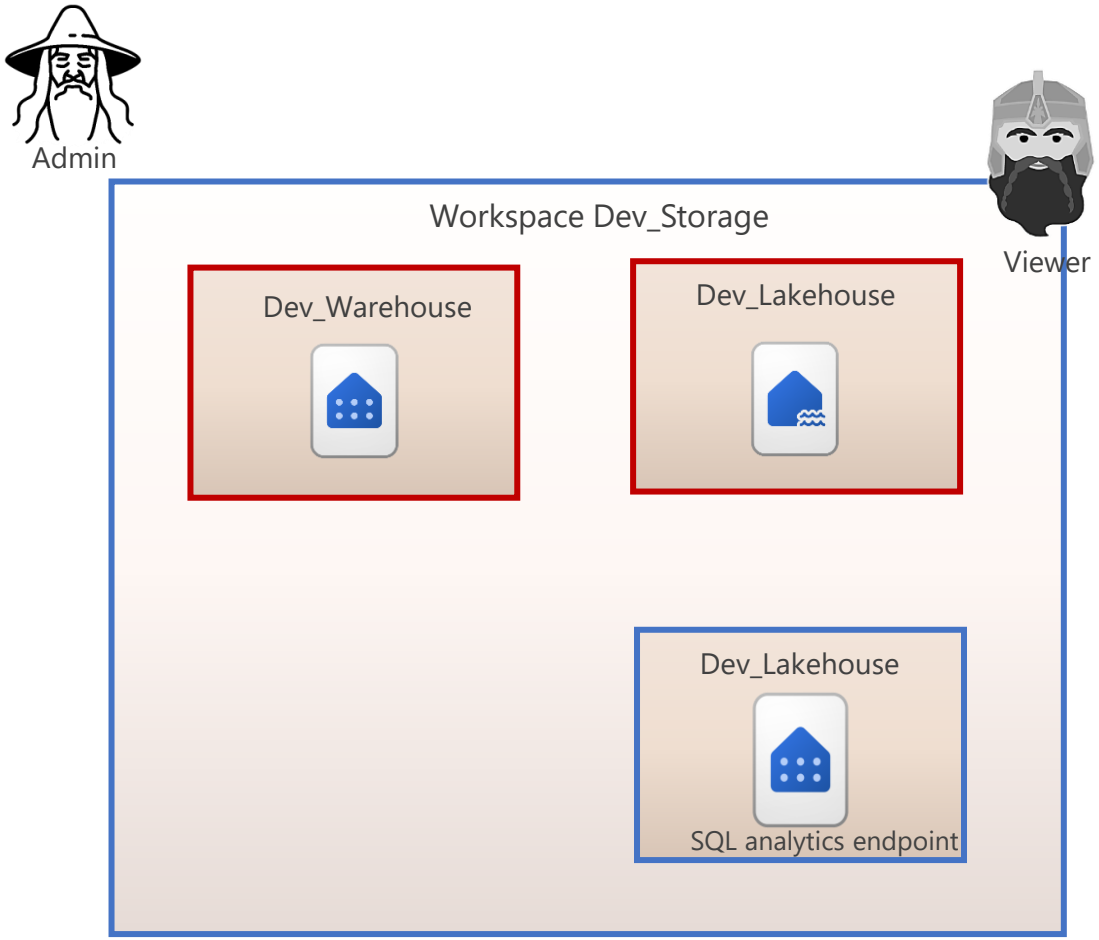


Option 1

- Share permission on the item level without selecting any additional permissions!
- The shared recipient can't see any data
- Access to objects within the Warehouse or SQL analytics endpoint of Lakehouse can be provided using T-SQL GRANT statement

Object – Level Access

- Also known as engine level access
- Only possible in the Warehouse or SQL analytics endpoint of the Lakehouse



Has access

No access

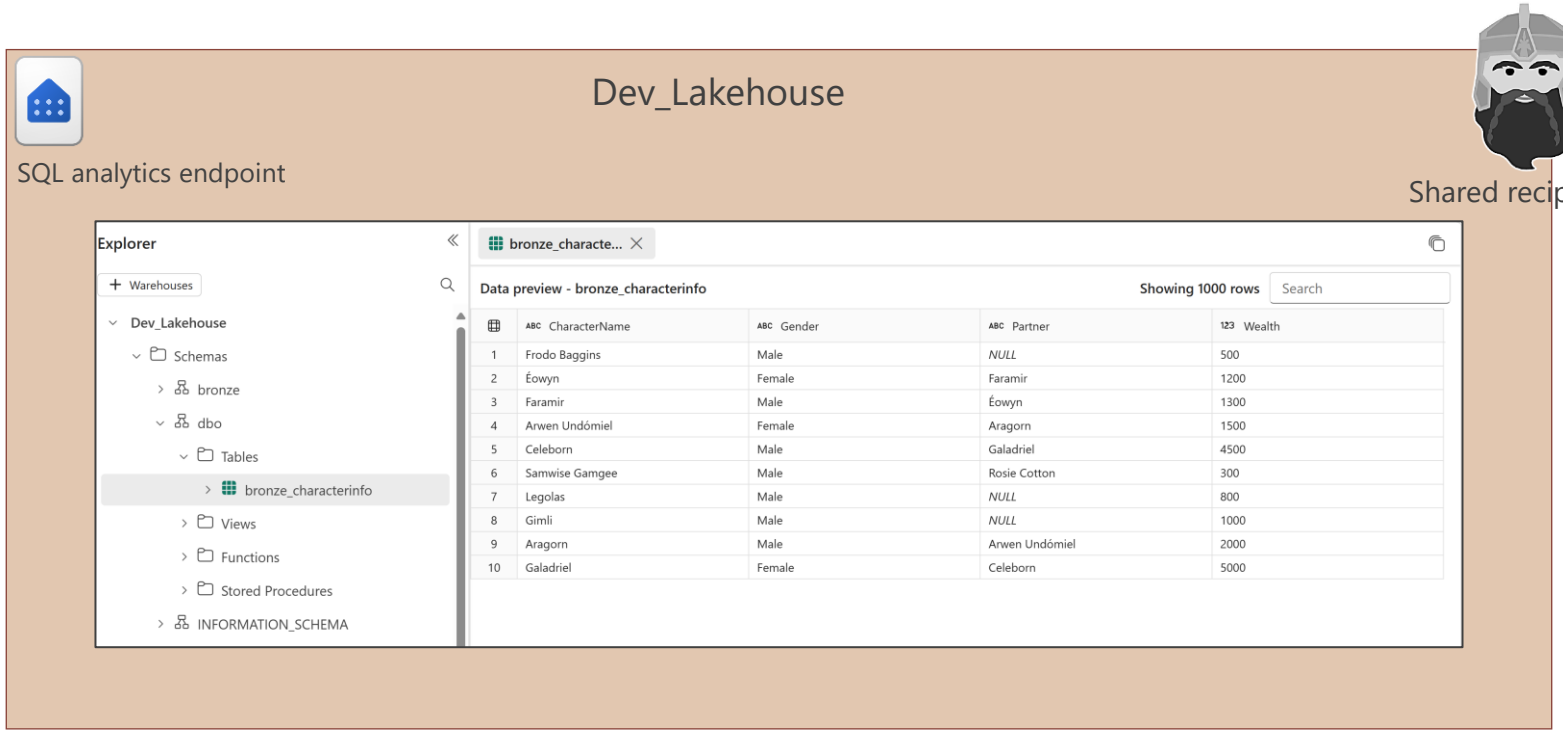
Option 2

- Access to the whole workspace with *Viewer* role
- The *Viewer* can see everything
- Denial of access to objects within the Warehouse or SQL analytics endpoint of Lakehouse can be provided using T-SQL DENY statement

Object – Level Access

Object-Level-Security (OLS): is a security mechanism that controls access to specific database objects, such as tables, views, or procedures, based on user privileges or roles. It ensures that users or roles can only interact with and manipulate the objects they have been granted permission for, protecting the integrity and confidentiality of the database schema and its associated resources.

```
GRANT SELECT ON dbo.bronze_characterinfo TO [Gimli@steinautech.com]
```



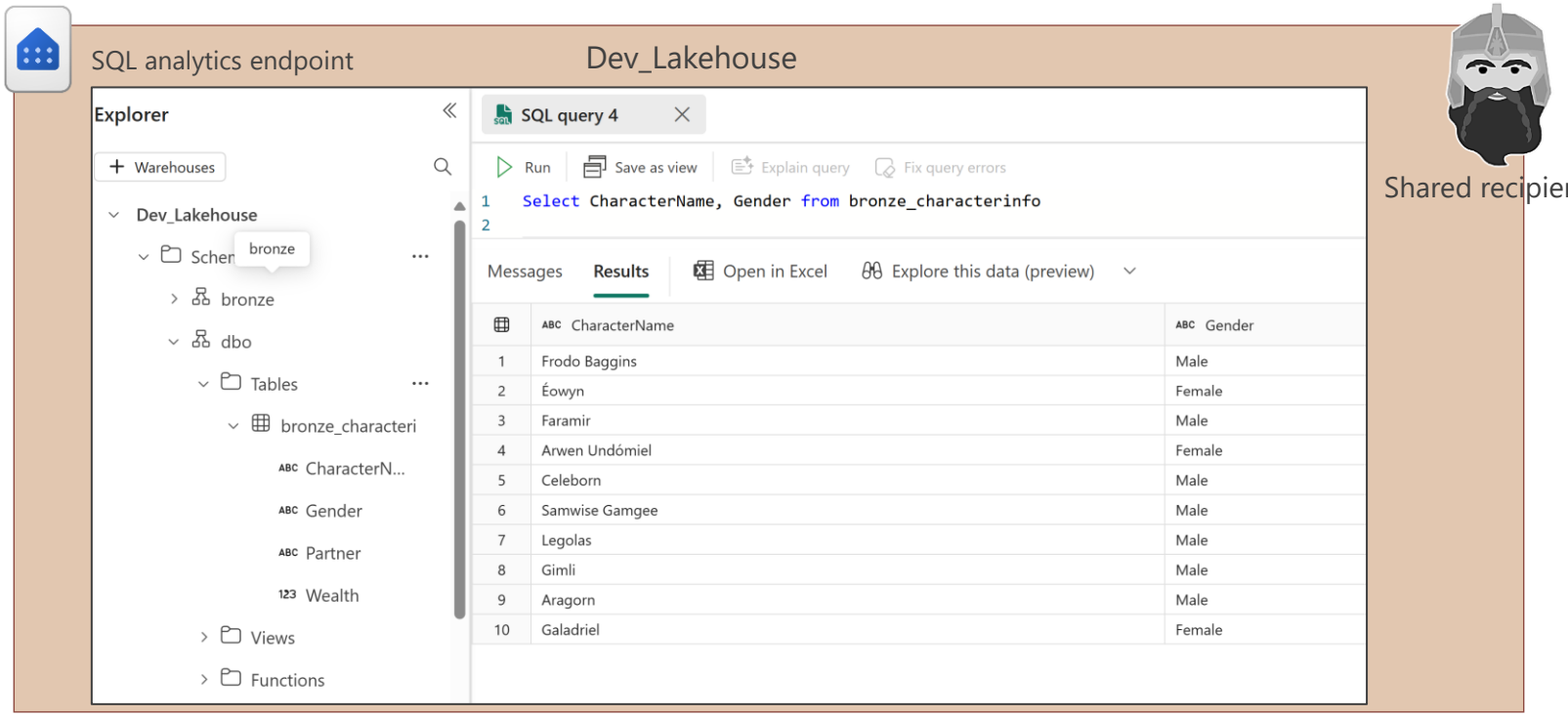
How do I verify the OLS in the Warehouse or SQL analytics endpoint of the Lakehouse?

```
SELECT
    princ.name AS UserName,
    princ.type_desc,
    perm.permission_name,
    perm.state_desc AS PermissionState,
    perm.class_desc,
    obj.name AS ObjectName
FROM
    sys.database_permissions perm
JOIN
    sys.database_principals princ ON
perm.grantee_principal_id = princ.principal_id
LEFT JOIN
    sys.objects obj ON perm.major_id = obj.object_id
WHERE
    princ.name = 'Gimli@steinautech.com';
```

Object – Level Access

Column-Level-Security (CLS): is a database security measure that limits access to specific columns or fields within a database table, allowing users to see and interact with only the authorized columns while concealing sensitive or restricted information. It offers fine-grained control over data access, safeguarding confidential data within a database.

```
GRANT SELECT ON dbo.bronze_characterinfo(CharacterName, Gender)
TO [Gimli@steinautech.com]
```



Shared recipient

- Columns for which shared recipient has GRANT permission have to be explicitly selected in the query by shared recipient
- Otherwise message explaining permission denials

How do I verify the CLS in the Warehouse or SQL analytics endpoint of the Lakehouse for my own user?

```
SELECT *
FROM sys.fn_my_permissions('dbo.bronze_characterinfo', 'Object')
```

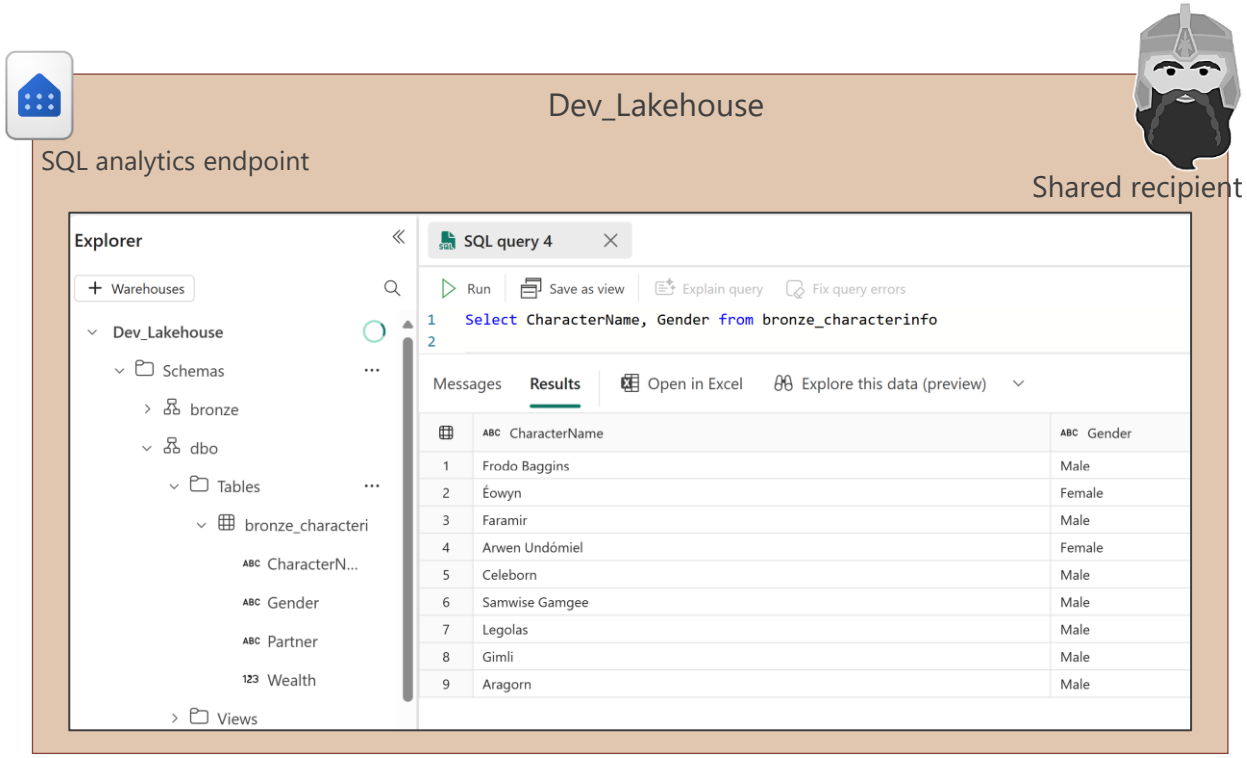
Object – Level Access

Row-Level-Security (RLS): is a database security feature that restricts access to individual rows or records within a database table based on specific criteria, such as user roles or attributes. It ensures that users can only view or manipulate data that is explicitly authorized for their access, enhancing data privacy and control.

```
CREATE SCHEMA Security;

CREATE FUNCTION
Security.tvf_charactersecurity(@CharacterName AS
nvarchar(100))
    RETURNS TABLE
WITH SCHEMABINDING
AS
    RETURN
    SELECT 1 AS tvf_securitypredicate_result
    WHERE USER_NAME() != 'Gimli@steinautech.com' OR
@CharacterName != 'Galadriel';

CREATE SECURITY POLICY CharacterSecurityPolicy
ADD FILTER PREDICATE
Security.tvf_charactersecurity(CharacterName)
ON dbo.bronze_characterinfo
WITH (STATE = ON);
```



How do I verify RLS in the Warehouse or SQL analytics endpoint of the Lakehouse?

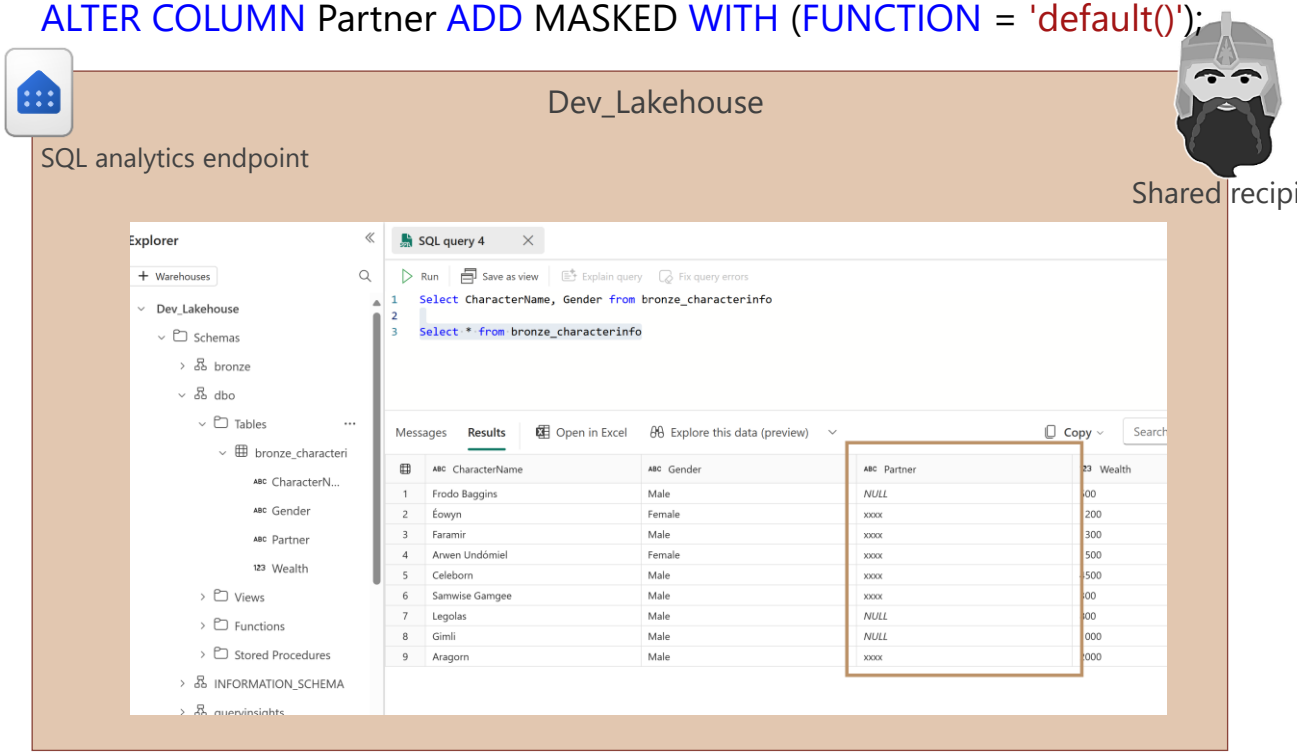
```
SELECT
name AS PolicyName,
object_id AS PolicyID,
is_enabled AS IsEnabled
FROM
sys.security_policies
WHERE
name = 'CharacterSecurityPolicy';
```

Object – Level Access

Dynamic data masking (DDM): helps prevent unauthorized viewing of sensitive data by enabling administrators to specify how much sensitive data to reveal, with minimal effect on the application layer. Dynamic data masking can be configured on designated database fields to hide sensitive data in the result sets of queries.

```
ALTER TABLE dbo.bronze_characterinfo

ALTER COLUMN Partner ADD MASKED WITH (FUNCTION = 'default()');
```



- Users without the *Administrator*, *Member*, or *Contributor* rights on the workspace, and without elevated permissions on the SQL analytics endpoint or Warehouse, will see masked data

How do I verify dynamic data masking in the Warehouse or SQL analytics endpoint of the Lakehouse?

```
SELECT
    t.name AS TableName,
    c.name AS ColumnName,
    m.masking_function

FROM sys.masked_columns m

JOIN sys.columns c ON m.column_id = c.column_id

AND m.object_id = c.object_id

JOIN sys.tables t ON m.object_id = t.object_id

WHERE t.name = 'bronze_characterinfo';
```

OneLake RBAC

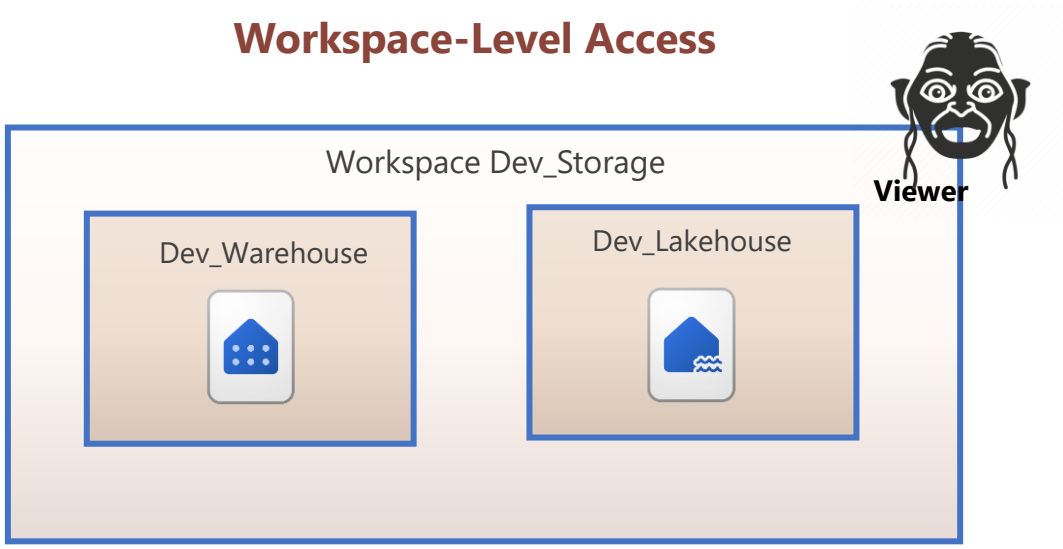
- Role-Based-Access-Control can be applied to individuals or security groups
- Applies to Lakehouse Items only (Lakehouse UX, notebooks, OneLake APIs)
- Does not apply to workspace *Admins*, *Members* or *Contributors*
- Applies only to *Viewers* or shared recipients without any further permissions

Has access

No access

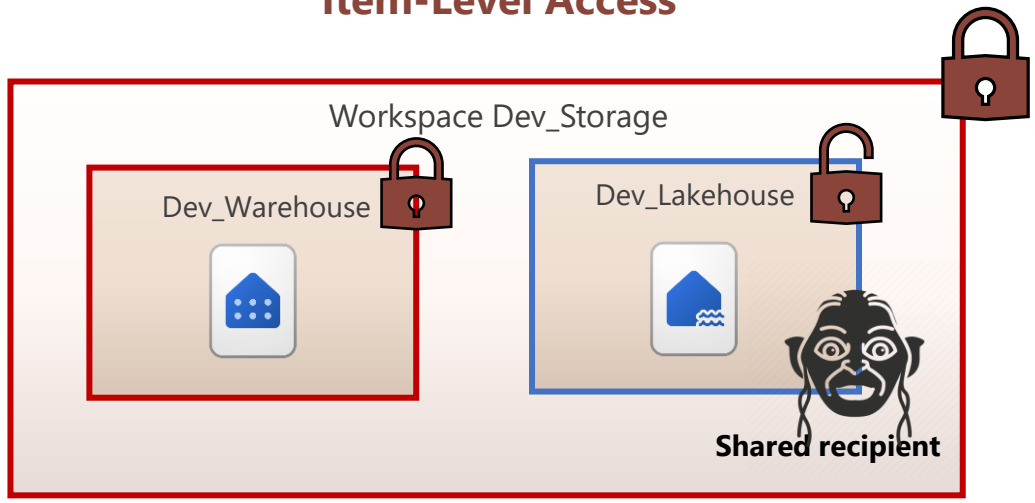
Option 1

Workspace-Level Access

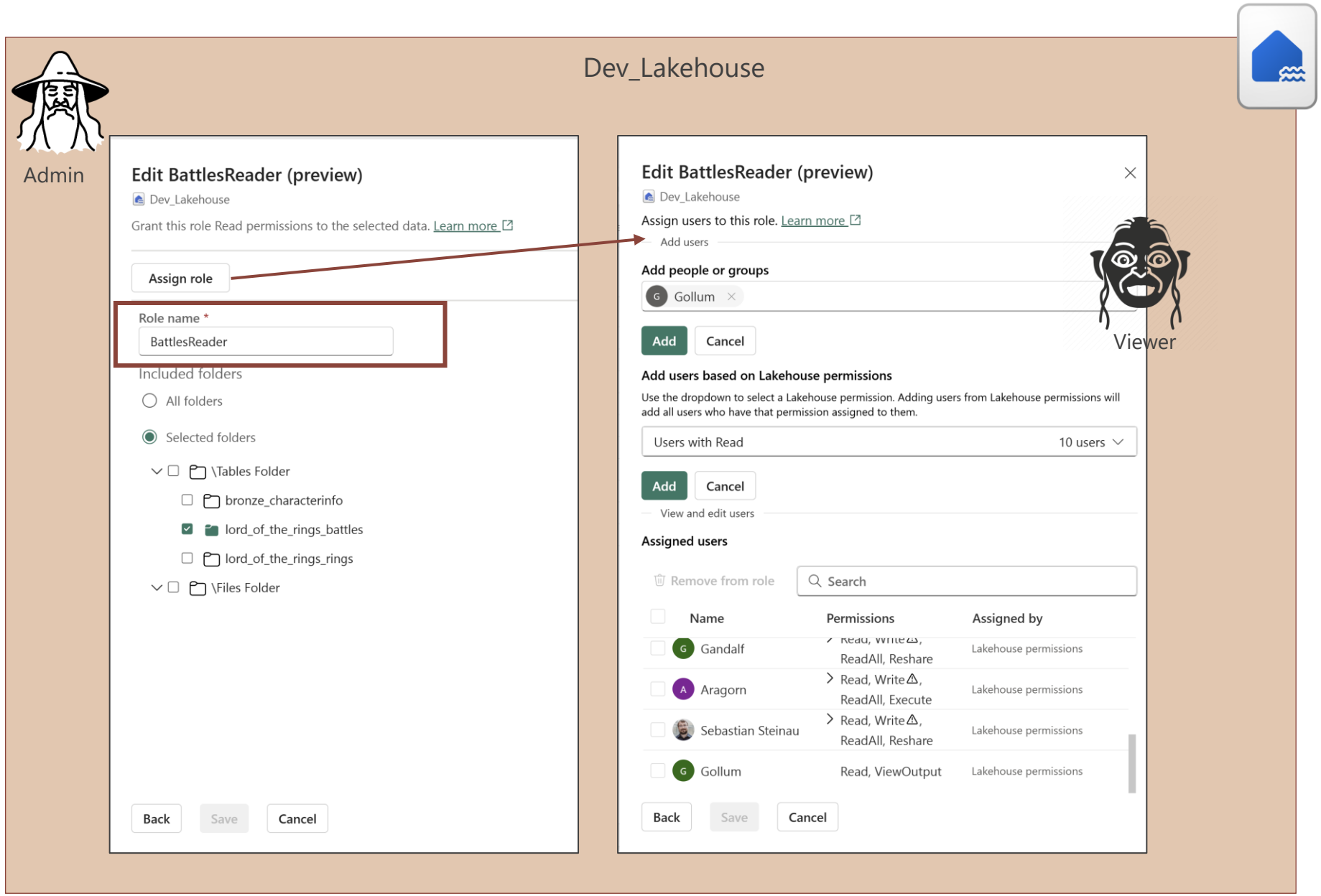


Option 2

Item-Level Access

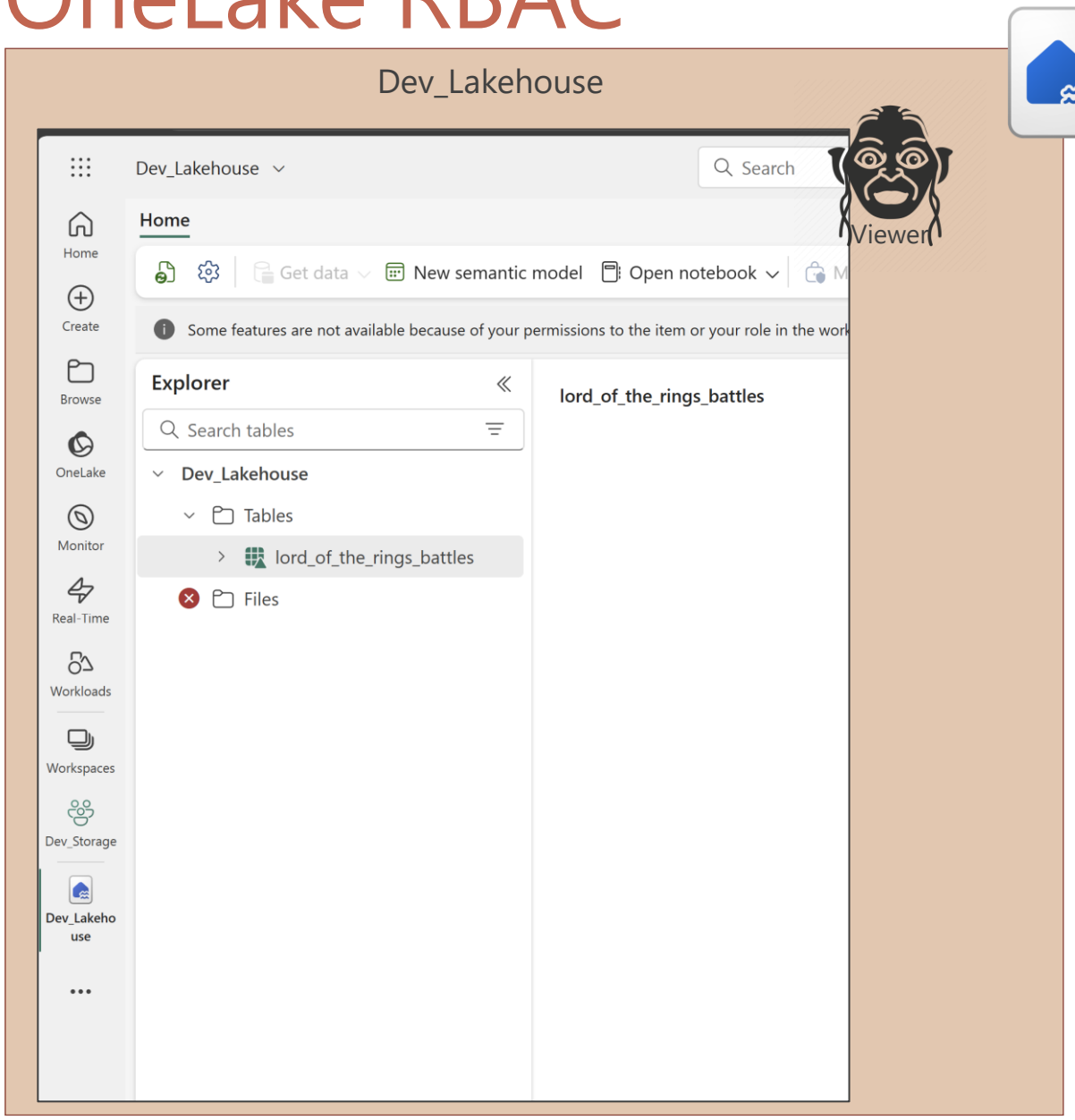


OneLake RBAC



- *Admins*, *Members* and *Contributors* can create OneLake data access roles to grant access to only specific folders in a Lakehouse
- Access can be granted to single folders or all folders
- The role can also be assigned to users based on their existing permissions

OneLake RBAC



- The *Viewer* can now see the tables folder in the Lakehouse for which the role assigned to him grants access
- The *Viewer* can also see all underlying files

Since Workspace *Admin, Member and Contributor* Roles automatically grant Write permissions to OneLake, it overrides any OneLake RBAC Read permissions!

OneLake RBAC

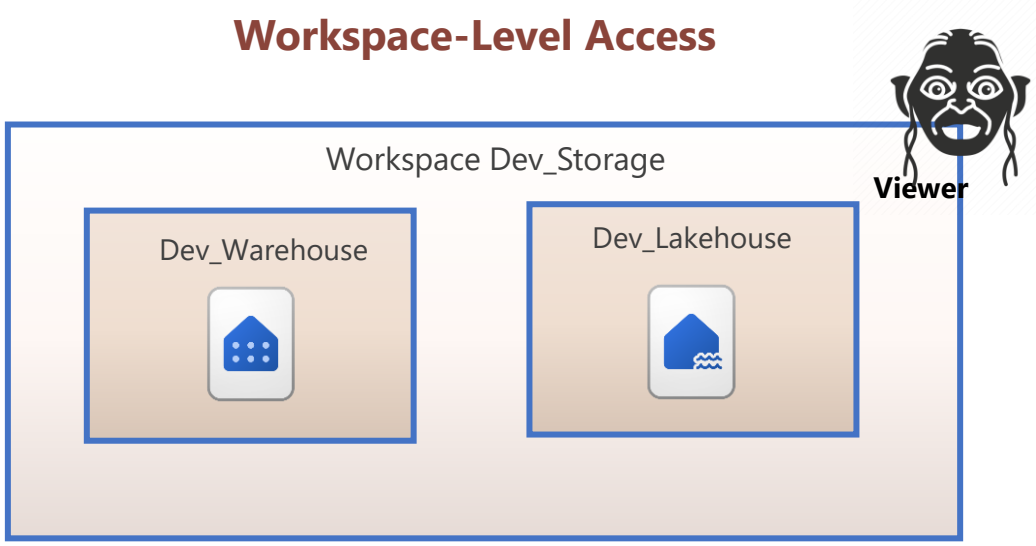
- **Option 1:** *Viewer* has access to all items in the Workspace, cannot read Lakehouse and is granted access to specific Lakehouse folders through RBAC. However, *Viewer* can access data through SQL analytics endpoint!
- **Option 2:** Shared recipient has no access to the Workspace where Lakehouse is located. Access to specific folder in Lakehouse is granted through RBAC when no further permissions have been assigned

Has access

No access

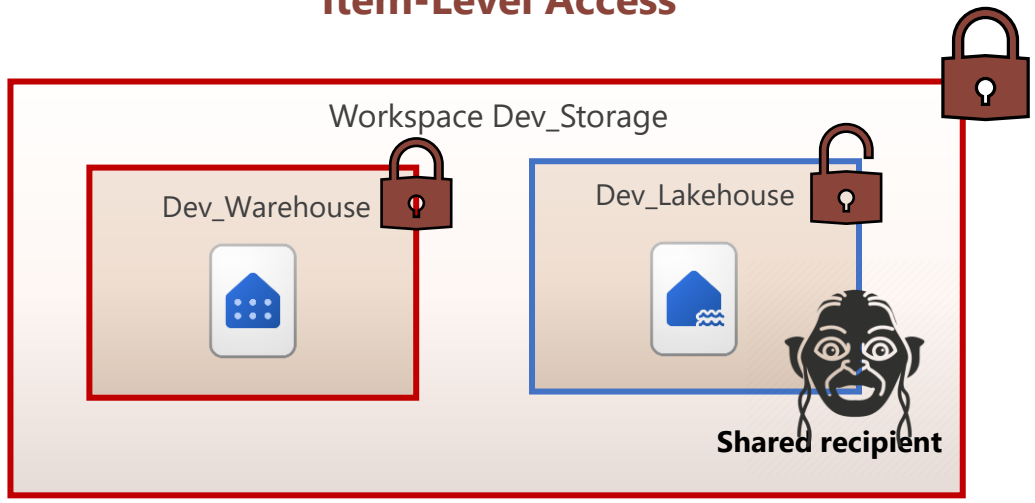
Option 1

Workspace-Level Access

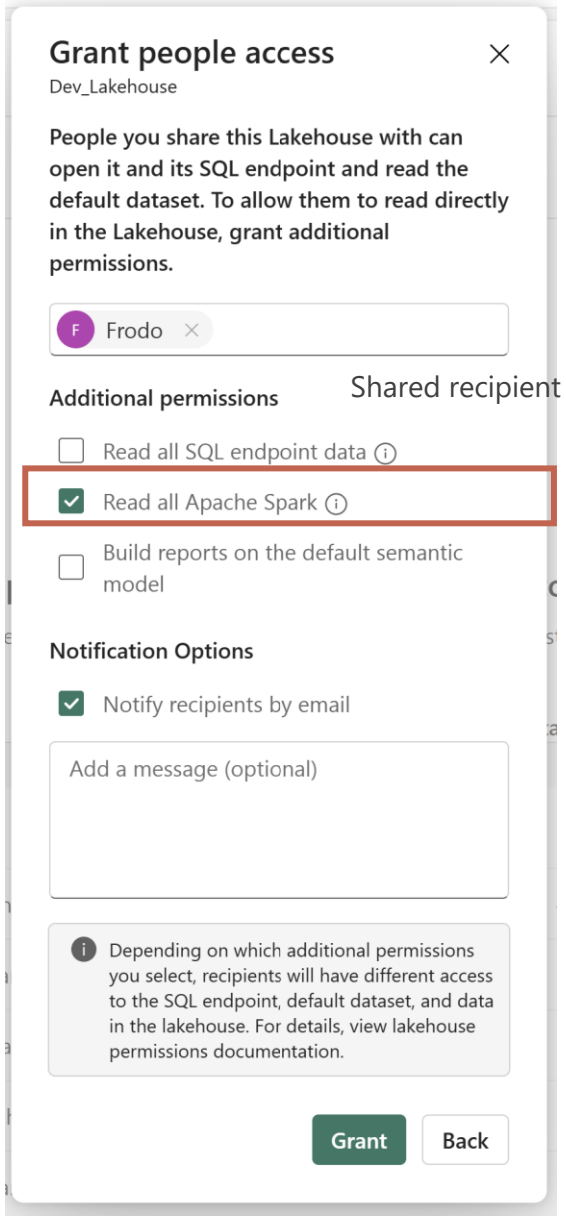


Option 2

Item-Level Access



OneLake RBAC





Admin

Shared recipient

Without RBAC

- Can read ALL underlying files through Spark



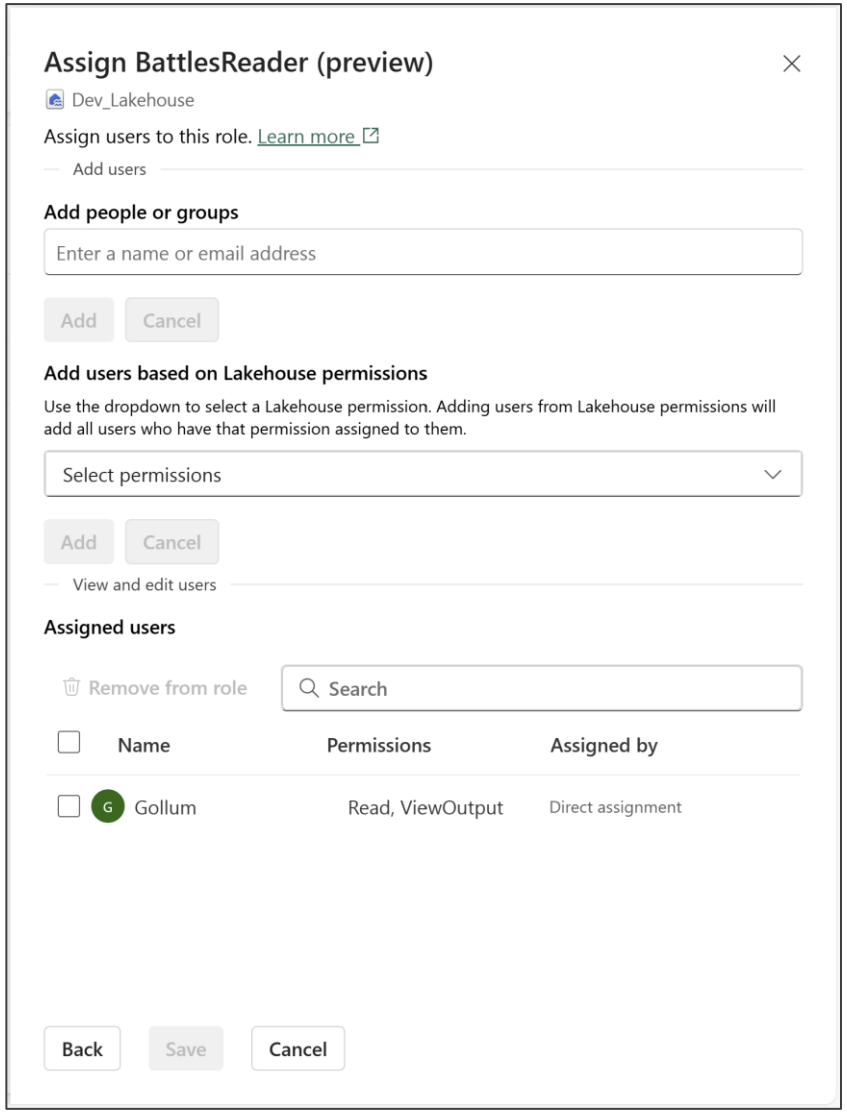
Shared recipient

With RBAC

- Can read only underlying files for which RBAC applies through Spark

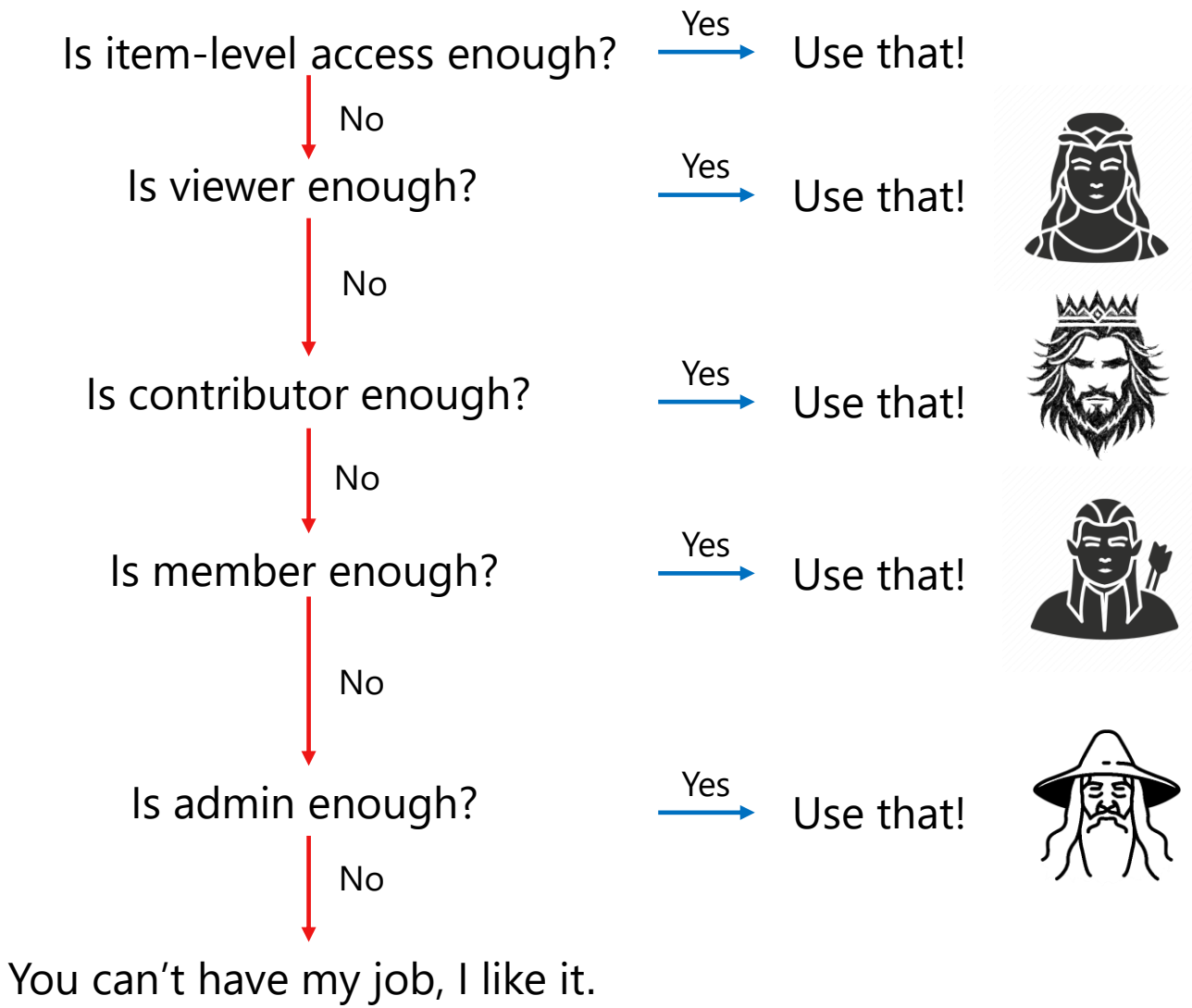
For item-level-access when shared recipient has ReadAll permission, RBAC can be used to restrict the access

How do I find out who has an RBAC role?



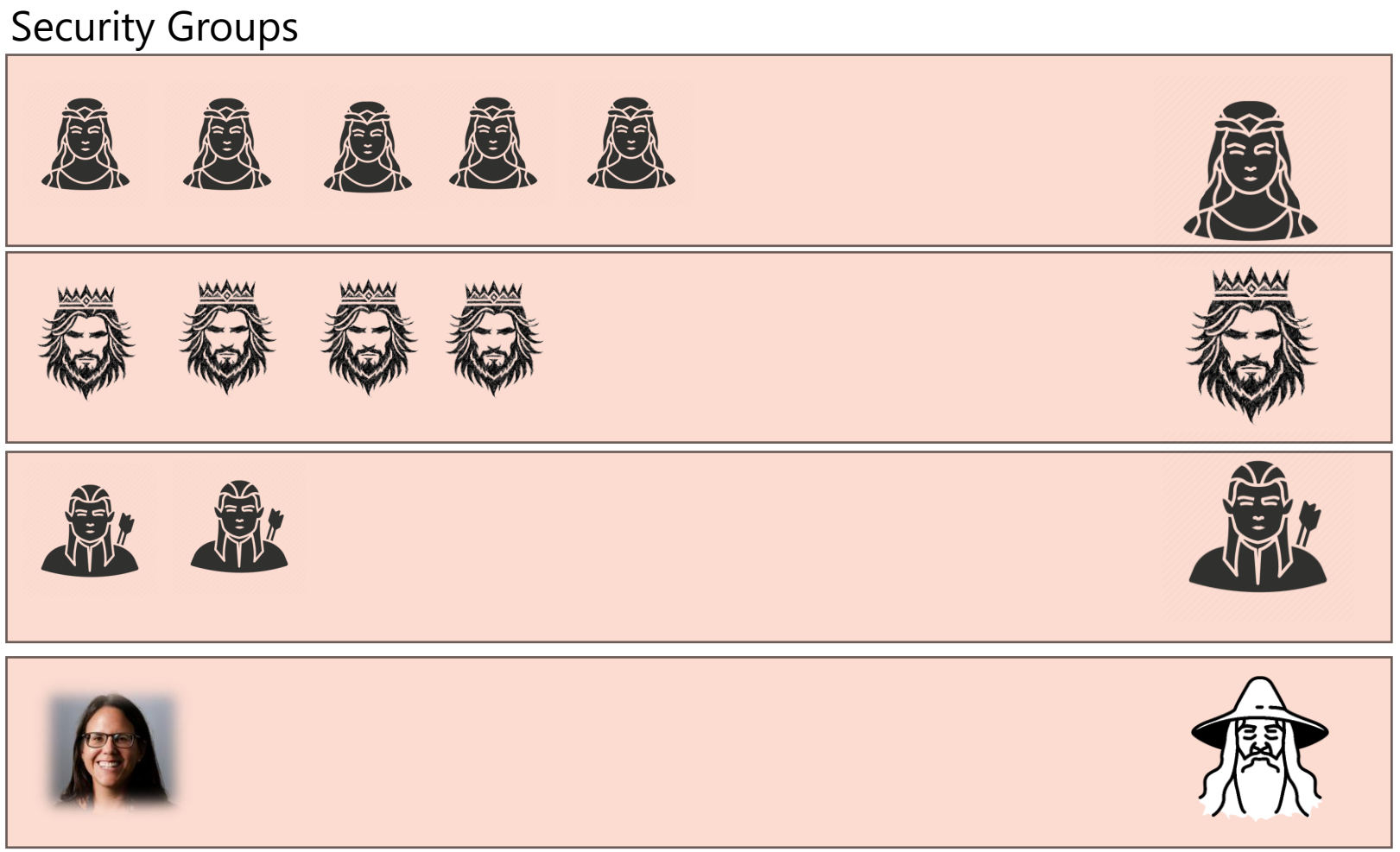
Take aways for designing proper access control

Principle of least privilege



Take aways for designing proper access control

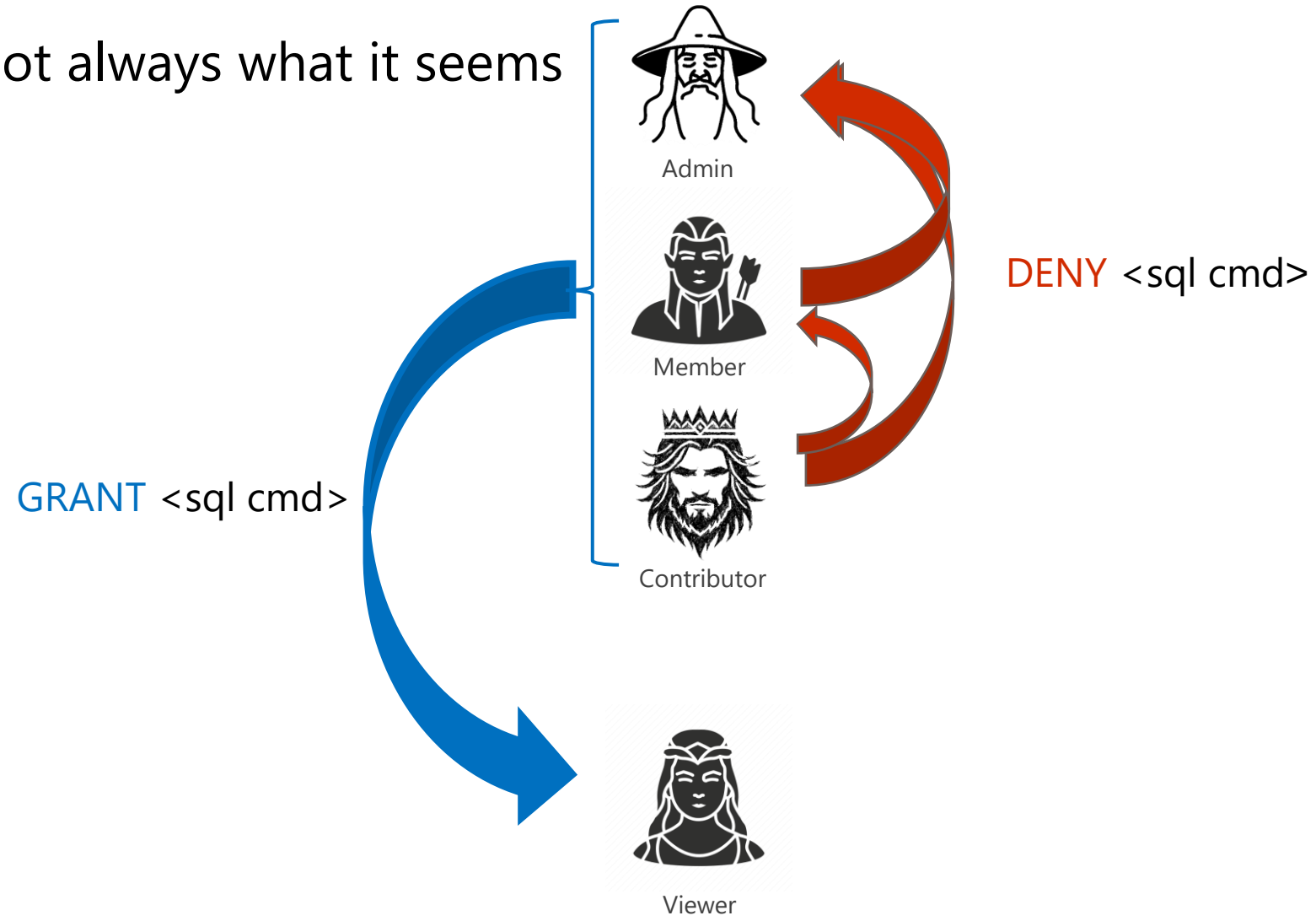
Minimize effort*



*This slide minimized effort and looks that way

Take aways for designing proper access control

A role is not always what it seems



Thank you for your attention.
Any questions?

Feedback is kindly appreciated <3



