

INSTALL AND SIGN IN

```
pip install ms-fabric-cli
```

Needs Python 3.10, 3.11, 3.12 or 3.13. Upgrade with `pip install --upgrade ms-fabric-cli`; check with `fab --version`.

```
fab auth login
```

Pick **Interactive with a web browser** to sign in as yourself.

```
fab auth login -u <client-id> -p <client-secret> --tenant <tenant-id>
```

A service principal with a secret, for headless runs.

```
fab auth login -u <client-id> --federated-token <token> --tenant <tenant-id>
```

A service principal with a federated credential, as in a CI/CD pipeline.

```
fab auth status
```

Who you're signed in as. `fab auth logout` signs out.

GET AROUND

Run **fab** alone for the `fab:/$` prompt, and type commands without **fab**. From your shell, put **fab** in front of each one. **quit** leaves the prompt.

```
ls
```

At `/`, every workspace you can access. Inside a workspace, its items.

```
cd 'Sales Performance Demo.Workspace'
```

Quote names with spaces. `cd ..` goes up a level; `pwd` shows where you are.

```
open 'Sales Performance.Report'
```

Opens the item in your browser.

```
fab find 'sales' -P type=Report -l
```

Searches the OneLake catalog across workspaces; `-l` adds IDs.

```
fab get 'Sales Performance Demo.Workspace' -q id
```

An ID for `fab api` calls. Works on items too.

```
help
```

Every command. `acl` or `job` alone lists its subcommands; `fab <command> --help` shows the flags.

COPY AND MOVE CONTENT

```
cp 'Sales Performance.Report' '../Sales Demo 2.Workspace'
```

Copies without the sensitivity label; `-f` skips the prompt, `-r` copies a folder with its subfolders. A whole workspace copies only at the prompt. A report copy keeps its original semantic model.

```
fab set '<ws>.Workspace/<report>.Report' -q 'definition.parts[0].payload.datasetReference.byConnection.connectionString' -i '<connection-string>'
```

Rebinds a report (PBIR definition version 2). `fab get` with the same `-q` and `-f` prints the current string; change its workspace, initial catalog and semantic model ID. Version 1: `-q semanticModelId -i <model-id>`.

```
fab export '<ws>.Workspace/<item>' -o <folder>
```

Writes the item's definition to disk. `fab import` brings it back; `fab deploy` runs a fabric-cicd deployment.

WHO HAS ACCESS

```
acl ls ./
```

Everyone with access to the current workspace: role, identity, type.

```
fab acl set '<ws>.Workspace' -I <object-id> -R viewer
```

Roles: admin, member, contributor, viewer. `-I` is the Microsoft Entra object ID.

```
fab acl rm '<ws>.Workspace' -I <object-id>
```

Removes that identity's access.

acl set and **acl rm** work on workspaces, gateways and connections. Listing an item's access needs the tenant-level Fabric Administrator role, and in 1.7.0 **acl ls** reads pipelines and lakehouses but not reports or models. For those:

```
fab api 'admin/workspaces/<workspace-id>/items/<item-id>/users?type=Report'
```

Who has access to a report; `type=SemanticModel` for a model.

JOBS

Notebooks, data pipelines, Spark job definitions and lakehouse table maintenance.

```
fab job run-list '<ws>.Workspace/<pipeline>.DataPipeline'
```

Recent runs. Then `run-status --id <job-id>` for one run.

```
fab job run-cancel '<ws>.Workspace/<pipeline>.DataPipeline' --id <job-id>
```

Cancels a hung run. `start` and `run` kick one off; `run-sch` schedules.

SEMANTIC MODEL REFRESH, THROUGH FAB API

```
fab api -A powerbi 'groups/<workspace-id>/datasets/<model-id>/refreshes?$top=5'
```

Refresh history: the last 60 entries without `$top`. Needs Write permission on the model.

```
-q "text.value[.]{status: status, start: startTime, error: serviceExceptionJson}"
```

Add to keep only the status, start time and error code. `fab api` puts the response body under `text`.

```
fab api -A powerbi -X post 'groups/<workspace-id>/datasets/<model-id>/refreshes'
```

Starts a refresh now.

-A picks the API: fabric (the default), powerbi, storage, azure.