**Connection profiles**

When you invoke dbt from the command line, dbt parses your dbt\_project.yml and obtains the profile name, which dbt needs to connect to your [data warehouse](https://docs.getdbt.com/terms/data-warehouse). dbt then checks your profiles.yml file for a profile with the same name. A profile contains all the details required to connect to your data warehouse.

Profiles.yml can only be configured in dbt core and not in dbt cloud as it can be configured from UI in the cloud. After profile.yml file is configured according to the data warehouse we wanted to connect,

This file generally lives outside of your dbt project to avoid sensitive credentials being checked in to version control, but profiles.yml can be safely checked in when [using environment variables](https://docs.getdbt.com/docs/core/connect-data-platform/connection-profiles#advanced-using-environment-variables) to load sensitive credentials.

dbt\_core:

  outputs:

    dev:

      account: yj51109.central-india.azure

      database: healthcare\_target

      password: Datadoers.123

      role: ACCOUNTADMIN

      schema: analytics

      threads: 4

      type: snowflake

      user: Shafiqahmed

      warehouse: COMPUTE\_WH

  target: dev

dbt\_core\_2: (any sensible name can be placed)  
Enter the second profile in here in the same file with different configurations

***In your profiles.yml file, you can store as many profiles as you need. Typically, you would have one profile for each warehouse you use. Most organizations only have one profile.***

**Setting up your profile**[**​**](https://docs.getdbt.com/docs/core/connect-data-platform/connection-profiles#setting-up-your-profile)

To set up your profile, copy the correct sample profile for your warehouse into your profiles.yml file and update the details as follows:

* Profile name: Replace the name of the profile with a sensible name – it’s often a good idea to use the name of your organization. Make sure that this is the same name as the profile indicated in your dbt\_project.yml file.
* target: This is the default target your dbt project will use. It must be one of the targets you define in your profile. Commonly it is set to dev.
* Populating your target:
  + type: The type of data warehouse you are connecting to
  + Warehouse credentials: Get these from your database administrator if you don’t already have them. Remember that user credentials are very sensitive information that should not be shared.
  + schema: The default schema that dbt will build objects in.
  + threads: The number of threads the dbt project will run on.

***Use the debug command to check whether you can successfully connect to your warehouse. Simply run dbt debug from within a dbt project to test your connection.***

*A typical profile for an analyst using dbt locally will have a target named dev, and have this set as the default.*

*If you do have multiple targets in your profile, and want to use a target other than the default, you can do this using the --target option when issuing a dbt command*

To ensure the user credentials you use in your target allow dbt to run, you will need to ensure the user has appropriate privileges. While the exact privileges needed varies between data warehouses, at a minimum your user must be able to:

* read source data
* create schemas¹
* read system [tables](https://docs.getdbt.com/terms/table)

***If your user is unable to be granted the privilege to create schemas, your dbt runs should instead target an existing schema that your user has permission to create relations within.***

**Understanding threads**[**​**](https://docs.getdbt.com/docs/core/connect-data-platform/connection-profiles#understanding-threads)

When dbt runs, it creates a directed acyclic graph (DAG) of links between models. The number of threads represents the maximum number of paths through the graph dbt may work on at once – increasing the number of threads can minimize the run time of your project. The default value for threads in user profiles is [4 threads](https://docs.getdbt.com/docs/dbt-versions/release-notes/Dec-2022/default-thread-value).

**dbt Core Environments**

***dbt makes it easy to maintain separate production and development environments through the use of***[***targets***](https://docs.getdbt.com/reference/dbt-jinja-functions/target)***within a***[***profile***](https://docs.getdbt.com/docs/core/connect-data-platform/profiles.yml)*.* A typical profile, when using dbt locally (for example, running from your command line), will have a target named dev and have this set as the default. This means that while making changes, your objects will be built in your *development* target without affecting production queries made by your end users. ***Once you are confident in your changes, you can deploy the code to production, by running your dbt project with a prod target.***

RUNNING DBT IN PRODUCTION

Targets offer the flexibility to decide how to implement your separate environments – whether you want to use separate schemas, databases, or entirely different clusters altogether***! We recommend using different schemas within one database to separate your environments. This is the easiest to set up and is the most cost-effective solution in a modern cloud-based data stack.***

In practice, this means that most of the details in a target will be consistent across all targets, except for the schema and user credentials. ***If you have multiple dbt users writing code, it often makes sense for each user to have their own development environment. A pattern we've found useful is to set your dev target schema to be dbt\_<username>. User credentials should also differ across targets so that each dbt user is using their own data warehouse user.***

**Deploy dbt jobs**

In addition to setting up a schedule, there are other considerations when setting up dbt to run in production:

* The complexity involved in creating a new dbt job or editing an existing one.
* Setting up notifications if a step within your job returns an error code (for example, a model can't be built or a test fails).
* Accessing logs to help debug any issues.
* Pulling the latest version of your git repo before running dbt (continuous deployment).
* Running and testing your dbt project before merging code into master (continuous integration).
* Allowing access for team members that need to collaborate on your dbt project.

Advanced: Customizing a profile directory[​](https://docs.getdbt.com/docs/core/connect-data-platform/connection-profiles#advanced-customizing-a-profile-directory)

**The parent directory for profiles.yml is determined using the following precedence:**

1. --profiles-dir option (Highest)
2. DBT\_PROFILES\_DIR environment variable
3. current working directory
4. ~/.dbt/ directory (lowest)

**To check the expected location of your profiles.yml file for your installation of dbt, you can run the following:**

$ dbt debug --config-dir  
To view your profiles.yml file, run:  
  
***open /Users/alice/.dbt***

***You may want to have your profiles.yml file stored in a different directory than ~/.dbt/*** – for example, if you are [using environment variables](https://docs.getdbt.com/docs/core/connect-data-platform/connection-profiles#advanced-using-environment-variables) to load your credentials, you might choose to include this file in the root directory of your dbt project.

**Note that the file always needs to be called profiles.yml, regardless of which directory it is in.**

There are multiple ways to direct dbt to a different location for your profiles.yml file:

1. Use the --profiles-dir option when executing a dbt command[​](https://docs.getdbt.com/docs/core/connect-data-platform/connection-profiles#1-use-the---profiles-dir-option-when-executing-a-dbt-command)

This option can be used as follows:

***$ dbt run --profiles-dir path/to/directory***

If using this method, the --profiles-dir option needs to be provided every time you run a dbt command.

***By using this command, we are navigating to the profiles.yml file stored in any particular location. This command is used in the terminal when we want to run the models but the profiles.yml file should be of our choice or stored in somewhere else location and we will give the location to the directory along with this command***

2. Use the DBT\_PROFILES\_DIR environment variable to change the default location[​](https://docs.getdbt.com/docs/core/connect-data-platform/connection-profiles#2-use-the-dbt_profiles_dir-environment-variable-to-change-the-default-location)

Specifying this environment variable overrides the directory that dbt looks for your profiles.yml file in. You can specify this by running:

***$ export DBT\_PROFILES\_DIR=path/to/directory***

By default, dbt looks for the **profiles.yml** file in specific locations, such as the current working directory or the **~/.dbt/** directory. However, if you want to specify a custom directory where your **profiles.yml** file is located, you can use the **DBT\_PROFILES\_DIR** environment variable.

To do this, you can set the **DBT\_PROFILES\_DIR** environment variable to the desired directory path. The **export** command is used in Unix-like systems (such as Linux or macOS) to set environment variables. The provided command:

***$ export DBT\_PROFILES\_DIR=path/to/directory***

sets the **DBT\_PROFILES\_DIR** environment variable to the path **path/to/directory**. You need to replace **path/to/directory** with the actual path where your **profiles.yml** file is located.

By specifying the **DBT\_PROFILES\_DIR** environment variable, dbt will prioritize that directory over the default locations when searching for the **profiles.yml** file. This allows you to specify a custom directory and have dbt use the **profiles.yml** file from that location.

Note that the specific command syntax may vary depending on the operating system or shell you are using. The example provided assumes a Unix-like environment. In different systems or shells, the syntax for setting environment variables might differ.

Advanced: Using environment variables[​](https://docs.getdbt.com/docs/core/connect-data-platform/connection-profiles#advanced-using-environment-variables)

Credentials can be placed directly into the profiles.yml file or loaded from environment variables. Using environment variables is especially useful for production deployments of dbt. You can find more information about environment variables [here](https://docs.getdbt.com/reference/dbt-jinja-functions/env_var).