**Packages**

**What is a package?**[**​**](https://docs.getdbt.com/docs/build/packages#what-is-a-package)

Software engineers frequently modularize code into libraries. These libraries help programmers operate with leverage: they can spend more time focusing on their unique business logic, and less time implementing code that someone else has already spent the time perfecting.

In dbt, libraries like these are called *packages*. dbt's packages are so powerful because so many of the analytic problems we encountered are shared across organizations, for example:

dbt *packages* are in fact standalone dbt projects, with models and macros that tackle a specific problem area. As a dbt user, by adding a package to your project, the package's models and macros will become part of your own project. This means:

* Models in the package will be materialized when you dbt run.
* You can use ref in your own models to refer to models from the package.
* You can use macros in the package in your own project.

**USING PYTHON PACKAGES**

*Defining and installing dbt packages is different from*[*defining and installing Python packages*](https://docs.getdbt.com/docs/build/python-models#using-pypi-packages)*.*

**How do I add a package to my project?**[**​**](https://docs.getdbt.com/docs/build/packages#how-do-i-add-a-package-to-my-project)

1. Add a packages.yml file to your dbt project. This should be at the same level as your dbt\_project.yml file.
2. Specify the package(s) you wish to add using one of the supported syntaxes, for example:
3. Run dbt deps to install the package(s). Packages get installed in the dbt\_packages directory – by default this directory is ignored by git, to avoid duplicating the source code for the package.

**packages.yml**

packages:  
 - package: dbt-labs/snowplow  
 version: 0.7.0  
  
 - git: "https://github.com/dbt-labs/dbt-utils.git"  
 revision: 0.9.2  
  
 - local: /opt/dbt/redshift