## **Specifying resources:**

**Shorthand**[**​**](https://docs.getdbt.com/reference/node-selection/syntax#shorthand)

Select resources to build (run, test, seed, snapshot) or check freshness: --select, -s

**Examples**[**​**](https://docs.getdbt.com/reference/node-selection/syntax#examples)

By default, dbt run will execute *all* of the models in the dependency graph. During development (and deployment), it is useful to specify only a subset of models to run. Use the --select flag with dbt run to select a subset of models to run. Note that the following arguments (--select, --exclude, and --selector) also apply to other dbt tasks, such as test and build.

The --select flag accepts one or more arguments. Each argument can be one of:

1. a package name
2. a model name
3. a fully-qualified path to a directory of models
4. a selection method (path:, tag:, config:, test\_type:, test\_name:)

**Examples**:

$ dbt run --select my\_dbt\_project\_name *# runs all models in your project*  
$ dbt run --select my\_dbt\_model *# runs a specific model*  
$ dbt run --select path.to.my.models *# runs all models in a specific directory*  
$ dbt run --select my\_package.some\_model *# run a specific model in a specific package*  
$ dbt run --select tag:nightly *# run models with the "nightly" tag*  
$ dbt run --select path/to/models *# run models contained in path/to/models*  
$ dbt run --select path/to/my\_model.sql *# run a specific model by its path*

**dbt supports a shorthand language for defining subsets of nodes. This language uses the characters +, @, \*, and ,.**

*# multiple arguments can be provided to --select*  
$ dbt run --select my\_first\_model my\_second\_model  
  
*# these arguments can be projects, models, directory paths, tags, or sources*  
$ dbt run --select tag:nightly my\_model finance.base.\*  
  
*# use methods and intersections for more complex selectors*  
$ dbt run --select path:marts/finance,tag:nightly,config.materialized:table

As your selection logic gets more complex, and becomes unwieldly to type out as command-line arguments, consider using a [yaml selector](https://docs.getdbt.com/reference/node-selection/yaml-selectors). You can use a predefined definition with the --selector flag. Note that when you're using --selector, most other flags (namely --select and --exclude) will be ignored.

## **About dbt test command**

dbt test runs tests defined on models, sources, snapshots, and seeds. It expects that you have already created those resources through the appropriate commands.

The tests to run can be selected using the --select flag discussed [here](https://docs.getdbt.com/reference/node-selection/syntax).

* *# run tests for one\_specific\_model*  
  dbt test --select one\_specific\_model  
    
  *# run tests for all models in package*  
  dbt test --select some\_package.\*  
    
  *# run only tests defined singularly*  
  dbt test --select test\_type:singular  
    
  *# run only tests defined generically*  
  dbt test --select test\_type:generic  
    
  *# run singular tests limited to one\_specific\_model*  
  dbt test --select one\_specific\_model,test\_type:singular  
    
  *# run generic tests limited to one\_specific\_model*  
  dbt test --select one\_specific\_model,test\_type:generic

To run tests on all sources, use the following command:

$ dbt test --select source:\*

(You can also use the -s shorthand here instead of --select)

To run tests on one source (and all of its tables):

$ dbt test --select source:jaffle\_shop

And, to run tests on one source [table](https://docs.getdbt.com/terms/table) only:

$ dbt test --select source:jaffle\_shop.orders

How do I run models downstream of one source?

To run models downstream of a source, use the source: selector:

$ dbt run --select source:jaffle\_shop+

(You can also use the -s shorthand here instead of --select)

To run models downstream of one source [table](https://docs.getdbt.com/terms/table):

$ dbt run --select source:jaffle\_shop.orders+

Check out the [model selection syntax](https://docs.getdbt.com/reference/node-selection/syntax) for more examples!