**Environments in dbt**

In software engineering, environments are used to enable engineers to develop and test code without impacting the users of their software.

“Production” (or *prod*) refers to the environment that end users interact with, while “development” (or *dev*) is the environment that engineers work in. This means that engineers can work iteratively when writing and testing new code in *development*, and once they are confident in these changes, deploy their code to *production*.

In traditional software engineering, different environments often use completely separate architecture. For example, the dev and prod versions of a website may use different servers and databases.

[Data warehouses](https://docs.getdbt.com/terms/data-warehouse) can also be designed to have separate environments – the \_production\_ environment refers to the relations (for example, schemas, tables, and [views](https://docs.getdbt.com/terms/view)) that your end users query (often through a BI tool).

**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

You can learn more about different ways to run dbt in production in [this article](https://docs.getdbt.com/docs/deploy/deployments).

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.