Review

Traditional Data Teams

- Data engineers are responsible for maintaining data infrastructure and the ETL process for creating tables and views.
- Data analysts focus on querying tables and views to drive business insights for stakeholders.

ETL and ELT

- ETL (extract transform load) is the process of creating new database objects by extracting data from multiple data sources, transforming it on a local or third party machine, and loading the transformed data into a data warehouse.
- ELT (extract load transform) is a more recent process of creating new database objects by first extracting and loading raw data into a data warehouse and then transforming that data directly in the warehouse.
- The new ELT process is made possible by the introduction of cloud-based data warehouse technologies.

Analytics Engineering

- Analytics engineers focus on the transformation of raw data into transformed data that is ready for analysis. This new role on the data team changes the responsibilities of data engineers and data analysts.
- Data engineers can focus on larger data architecture and the EL in ELT.
- Data analysts can focus on insight and dashboard work using the transformed data.
- Note: At a small company, a data team of one may own all three of these roles and responsibilities. As your team grows, the lines between these roles will remain blurry.

dbt

- dbt empowers data teams to leverage software engineering principles for transforming data.
- The focus of this course is to build your analytics engineering mindset and dbt skills to give you more leverage in your work.

dbt, data platforms, and version control

If you are new to dbt, the underlying architecture may be new to you. First, read this quick overview and then follow the steps for creating the necessary accounts. There are effectively two ways in which to use dbt: dbt CLI and dbt Cloud.

- **dbt Cloud** is a hosted version that streamlines development with an online Integrated Development Environment (IDE) and an interface to run dbt on a schedule.
- **dbt Core** is a command line tool that can be run locally.

This course will assume you are using dbt Cloud, but the concepts and practices can easily be extended to dbt CLI.

Data platform

dbt is designed to handle the transformation layer of the 'extract-load-transform' framework for data platforms. dbt creates a connection to a data platform and runs SQL code against the warehouse to transform data.

In our demos, we will be using Snowflake as our data warehouse. You will need access to your own data platform to complete the practice exercises. Read more in our documentation on dbt's supported databases.

Version control

dbt also enables developers to leverage a version control system to manage their code base. A popular version control system is git. If you are unfamiliar with git, don't worry, dbt Cloud provides a UI that makes it simple to use a git workflow.

In this course, we will be demoing with **GitHub** to host the code base that we build. If you would prefer to use another service for version control with dbt Cloud, check out our documentation for other version control options.

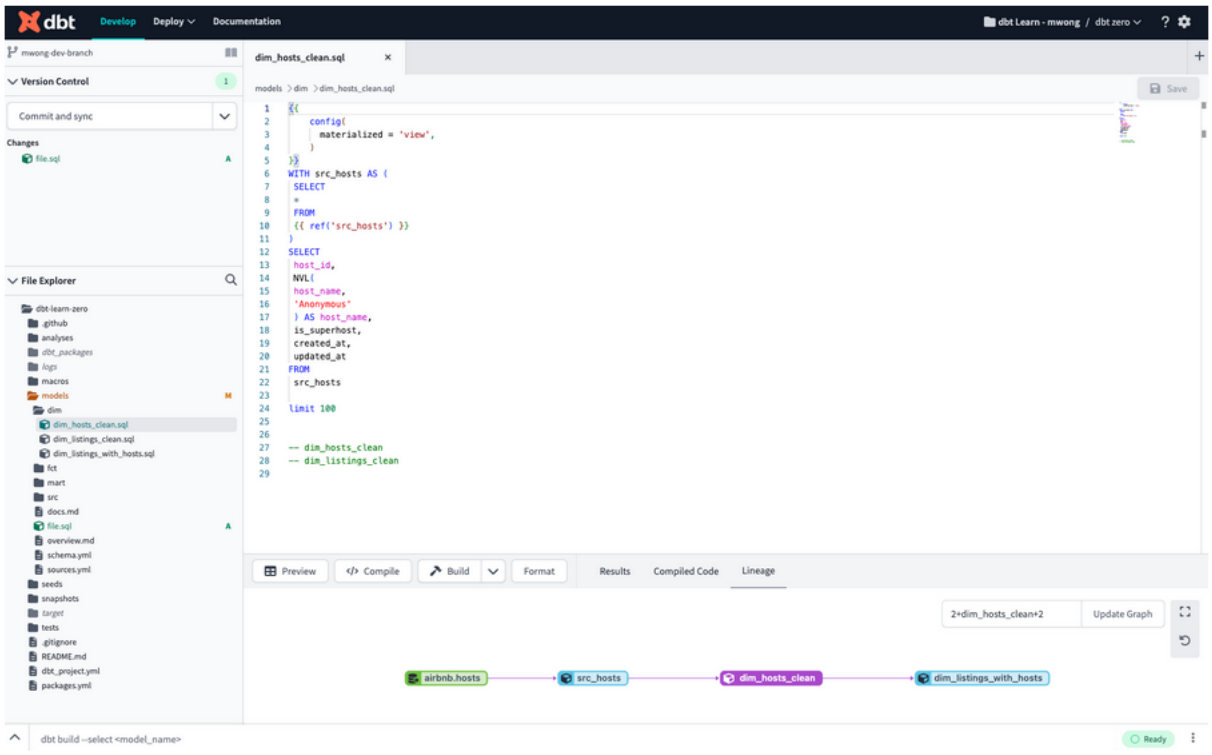
All in all, dbt is going to be the transformation interface between the code we write (stored and managed in a git repository) and the sample data we have to work with (stored and transformed in your data platform).

Now let's start setting up your individual accounts. In the following lessons, we will work to ensure all of these are connected appropriately.

Review

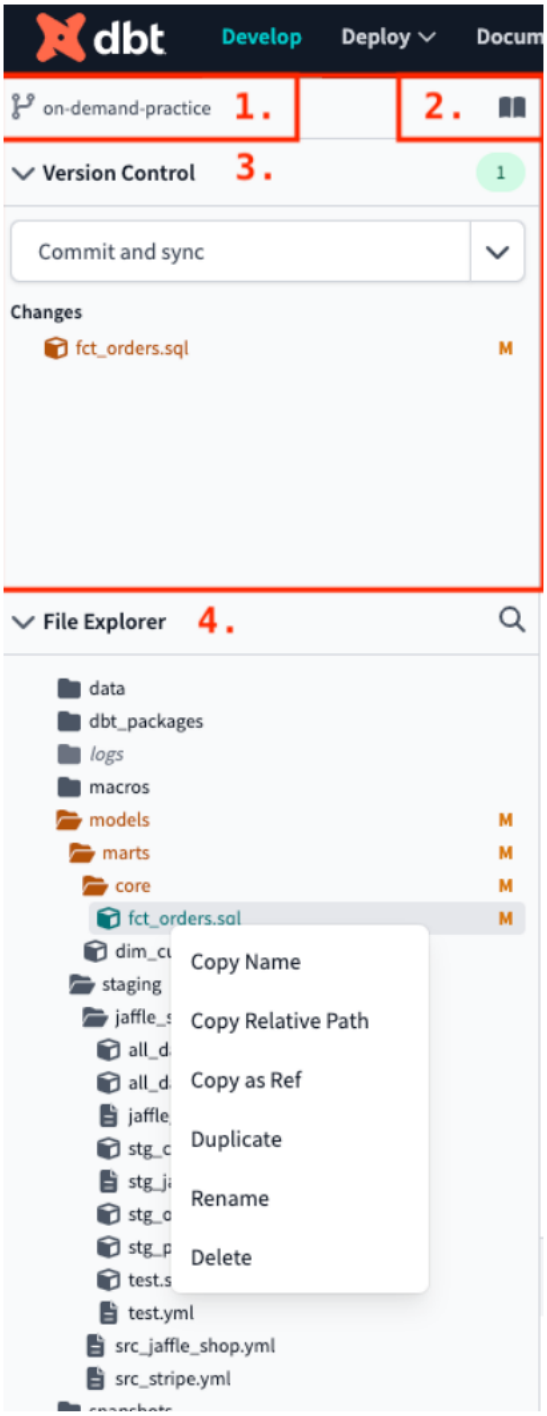
dbt Cloud IDE

The dbt Cloud integrated development environment (IDE) is a single interface for building, testing, running, and version-controlling dbt projects from your browser. With the Cloud IDE, you can compile dbt code into SQL and run it against your database directly



Basic layout

The IDE streamlines your workflow, and features a popular user interface layout with files and folders on the left, editor on the right, and command and console information at the bottom.



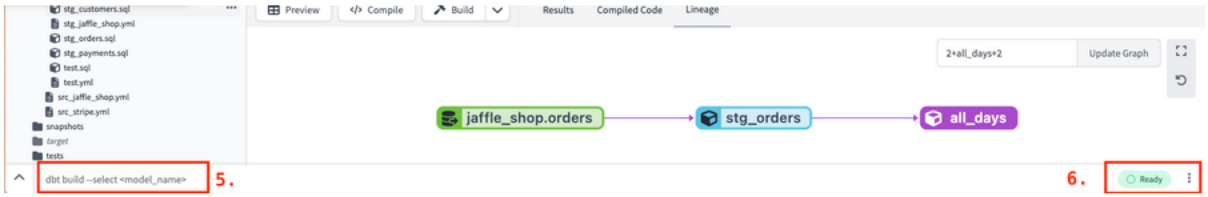
1. Git repository link — Clicking the Git repository link, located on the upper left of the IDE, takes you to your repository on the same active branch.

2. Documentation site button — Clicking the Documentation site book icon, located next to the Git repository link, leads to the dbt Documentation site. The site is powered by the latest dbt artifacts generated in the IDE using the dbt docs generate command from the Command bar.

3. Version Control — The IDE's powerful Version Control section contains all git-related elements, including the Git actions button and the Changes section.

4. File Explorer — The File Explorer shows the filetree of your repository. You can:

- Click on any file in the filetree to open the file in the File Editor.
- Click and drag files between directories to move files.
- Right click a file to access the sub-menu options like duplicate file, copy file name, copy as ref, rename, delete.
    - **Note:** To perform these actions, the user must not be in read-only mode, which generally happens when the user is viewing the default branch.
- Use file indicators, located to the right of your files or folder name, to see when changes or actions were made:
    - Unsaved (•) — The IDE detects unsaved changes to your file/folder
    - Modification (M) — The IDE detects a modification of existing files/folders
    - Added (A) — The IDE detects added files
    - Deleted (D) — The IDE detects deleted files



5.Command bar — The Command bar, located in the lower left of the IDE, is used to invoke dbt commands. When a command is invoked, the associated logs are shown in the Invocation History Drawer.

6. IDE Status button — The IDE Status button, located on the lower right of the IDE, displays the current IDE status. If there is an error in the status or in the dbt code that stops the project from parsing, the button will turn red and display "Error". If there aren't any errors, the button will display a green "Ready" status. To access the IDE Status modal, simply click on this button.

Explore Editing Features in the IDE

Review the Steps

Configure BigQuery for dbt Cloud

- Setting up BigQuery for dbt Cloud involves creating roles, adding principles, and working with projects
- Before you begin, make sure you have the BigQuery Admin or Owner role so you can complete these tasks
- To develop in dbt Cloud, it is recommended developers have two BigQuery projects, one for raw data and one for transformed data
- Developers will use a BigQuery service account to connect to dbt Cloud
- You will create and the service account in the BigQuery project where the transformed data will be stored
- The service account will need BigQuery Editor and BigQuery Job User roles
- The service account should also be added as a principle to the BigQuery project with the raw data
- The role to assign in the raw data project is BigQuery Data Viewer

Configure Github for dbt Cloud

- Once you have a Github account make sure you're invited to your team's Github organization
- Create a new repo using a name like `internal-analytics` or `dbt-project'
- Provide a quick description for the repo like "A dbt project for managing data transformations"
- Set the repo to private (you can make it public later)
- It is critical that you do not add a README file at this stage. This will be covered when you initialize your project later
- Leave other settings blank for now

Set Up a New dbt Cloud Project

- Login to BigQuery and Github in separate browser tabs before you begin
- In a third tab login to dbt Cloud
- Most people will login to dbt Cloud at cloud.getdbt.com. If you are on a virtual private cloud or based outside of the U.S., this will be a different link provided by your dbt account team
- In dbt Cloud, follow the **Complete project setup** flow
- Name the project 'Analytics' and select BigQuery as the warehouse
- Obtain a key from the BigQuery service account and upload the service account JSON file in dbt Cloud
- dbt Cloud will autopopulate settings for the environment
- It's important to note that each user will enter a schema for the dataset value. The standard recommended is dbt_firstinitiallastname
- Test your data platform connection, and follow the flow to connect Github and select the specific repository
- Click initialize dbt project to create a simple project structure to start building on
- Click commit and push and execute dbt run to see sample models

Review

Set Up a Production Environment

- Deployment environments are used for running code on a schedule and development environments are used for developing code
- A production environment is a deployment environment where developers can run jobs
- A production environment will let the team create a production deployment pipeline for testing and deploying to production
- dbt recommends creating a service account with a role that has access to read from raw data but not write
- The connection detail values for a production environment should always differ from the values in your development environment

Schedule a Job

- dbt Cloud Administrators can configure when and how jobs run in a production environment
- Administrators can set up daily jobs and select the days, and intervals for those jobs
- Running a simple daily job is one way to quickly test your data platform connection
- The audit log will maintain a record of who scheduled a job, when it was scheduled, and the status of the job

Set Up Folders by Data Maturity

- dbt Cloud Administrators can create folder structures to optimize dbt Cloud projects
- The maturity model is one structure used by some team
- You must create a new branch each time you create a new folder structur
- Staging folders are used to organize source conformed logic and store data before it's transformed
- Marts folders are used to organize business confirmed logic and combine staging models to roll up into key business concept
- Use a file named .gitkeep for each folder to help Git recognize an empty director
- Update the dbt_project.yml file and README file to align with the maturity model
- Commit changes to the folder structure with a descriptive message and a pull reques
- Once the pull request has been reviewed, merge the pull request and make sure the changes are reflected in the code and in the dbt Cloud IDE

Set Up Folders by Domain

- dbt Cloud Administrators can set folders up by domain
- This structure lays the groundwork for different teams to build future models
- Create a new branch each time you create a new folder structure
- Create team folders using folder names that apply to your business
- Add a .gitkeep file to each folder
- Update the dbt_project.yml file and README file to align with the model
- Commit changes to the folder structure with a descriptive message and a pull request
- Once the pull request has been reviewed, merge the pull request and make sure the changes are reflected in the code and in the dbt Cloud IDE