Apache
Airflow

Apache Airflow: Overview

- *Introduction to Apache Airflow*
- *Orchestrating Your Data Pipelines*
- *A Modern Workflow Management Platform*
- *Understanding Airflow's Core Concepts*

Core Concepts

1. *Directed Acyclic Graph (DAG)*
2. *Operators*
3. *Tasks*
4. *Task Instances*
5. *Scheduler*
6. *Worker*

The Recipe Book (DAG)

- DAGs are workflow blueprints written in Python.
- They define tasks with clear, ordered steps.
- DAGs prevent loops, ensuring forward progress.
- They visually map all connected tasks.

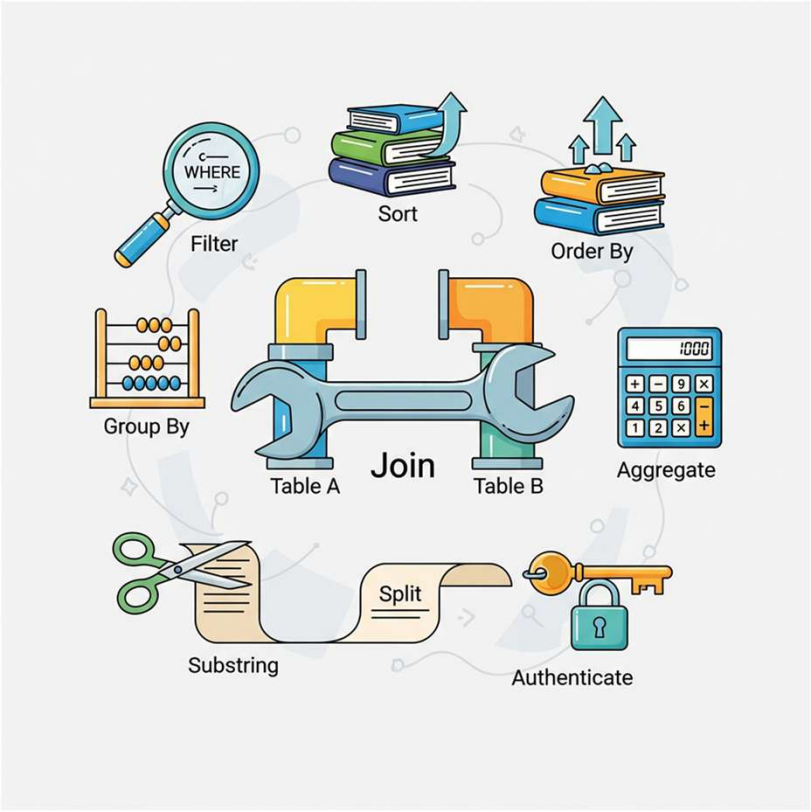
Analogy: "Every product we make has a detailed **Recipe Book**. It doesn't contain the food itself, but the exact step-by-step instructions: 'First, wash the carrots, THEN chop them, THEN sauté the onions...' Crucially, it shows the order, and you can't go backwards!"

Airflow Term: "This 'recipe' is what we call a **DAG**, or **Directed Acyclic Graph**.

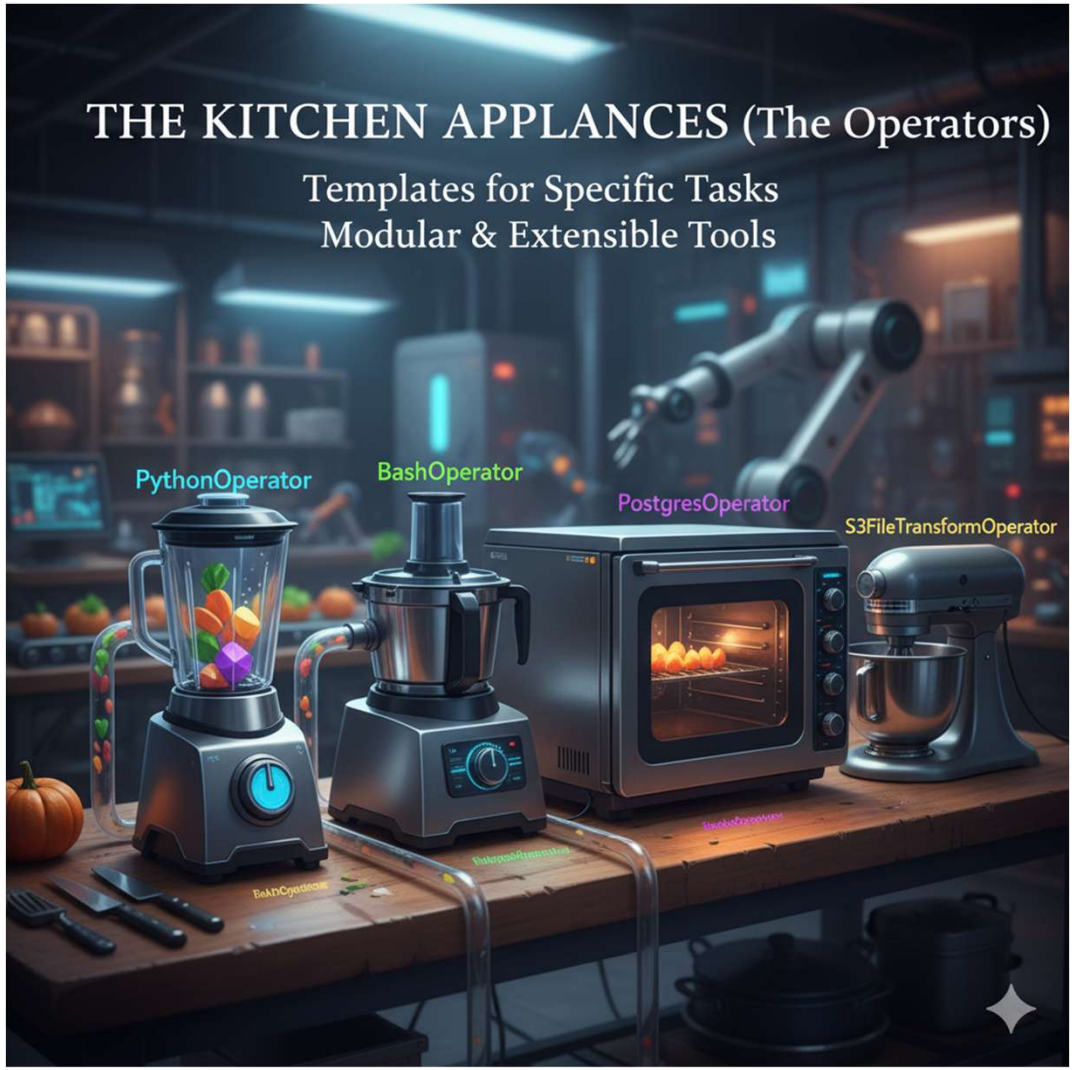
D.A.G

- DAGs ensure clear, ordered steps from start to finish.
- **Directed**: Means the steps have a clear flow, from start to finish.
- **Acyclic**: Means no loops in task execution
- **Graph**: It's a visual map of all your tasks and how they connect.
- **Python Code**!: And the coolest part? You write these recipes in Python!"
- Key takeaway: **DAGs are your foolproof, step-by-step workflow blueprints!**

The Operators



- Operators are specialized tools for tasks.
- They are pre-defined templates for jobs.
- Examples include Python and Bash Operators.
- Hundreds more exist for various services.



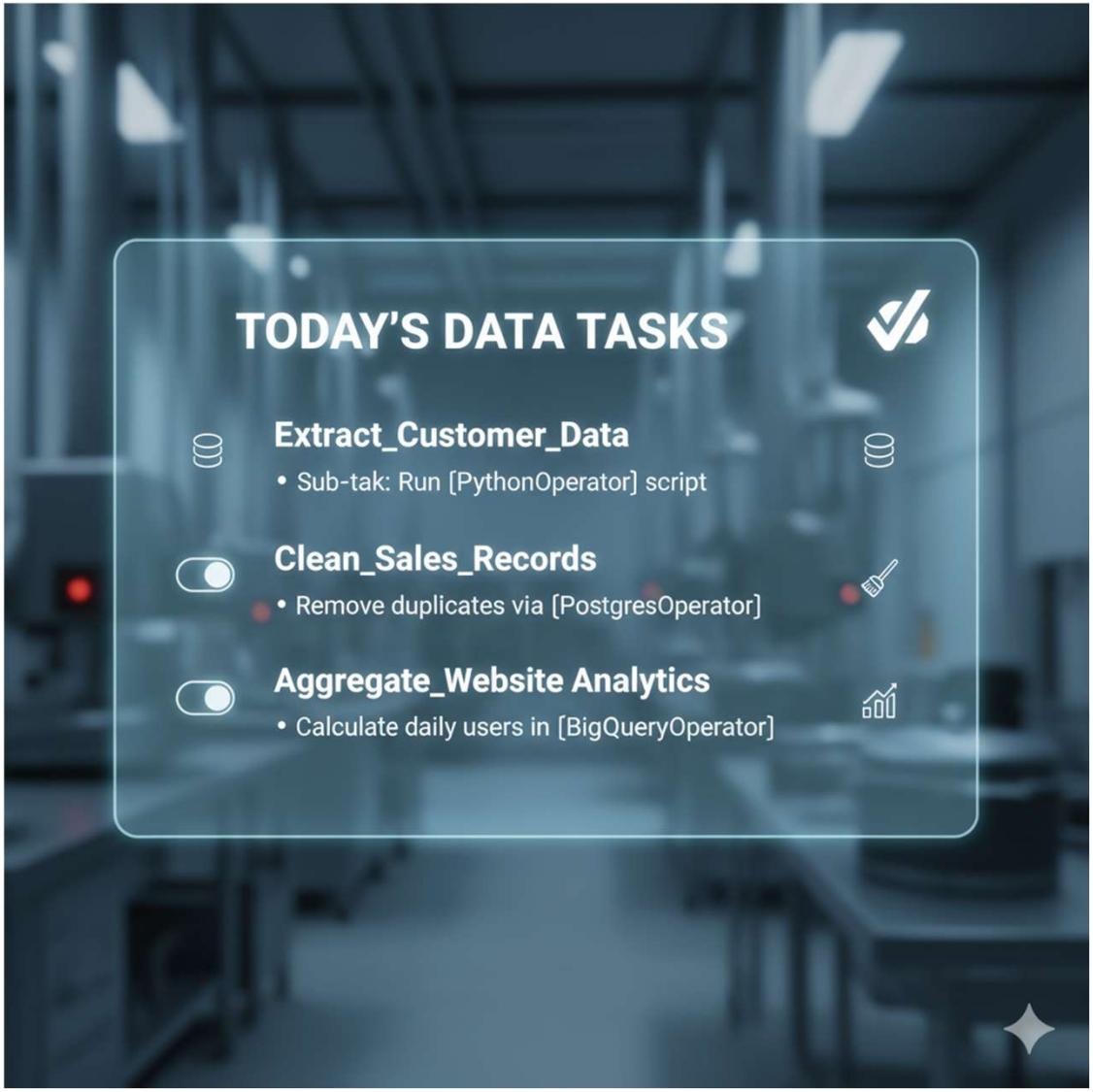
The Kitchen Appliances (The Operators) 🛠️

- **Analogy:** "Our factory has a full suite of **Kitchen Appliances**: blenders for smoothies, ovens for baking, food processors for chopping. Each appliance has a single, specialized function."

- **Airflow Term:** "In Airflow, these are **Operators**. They are pre-defined templates for a specific type of task.

The Specific Jobs (The Tasks) 🏃‍♀️

- **Analogy:** "When the recipe says 'Chop the carrots,' that's a **Specific Job**. We use the food processor (our appliance/Operator) to do that particular action."
- **Airflow Term:** "A **Task** is an *instance* of an Operator within your DAG. It's an Operator configured with specific parameters for a concrete piece of work. For example, your DAG might have a task named `prepare_customer_data` which uses a `PythonOperator` to call a Python function that cleans customer records."



DefiningTasks

- Tasks are instances of an Operator.
- They represent a specific unit of work.
- Tasks execute within your defined DAGs.
- Each task performs a concrete operation.
- They are configured with specific parameters.

The Scheduler

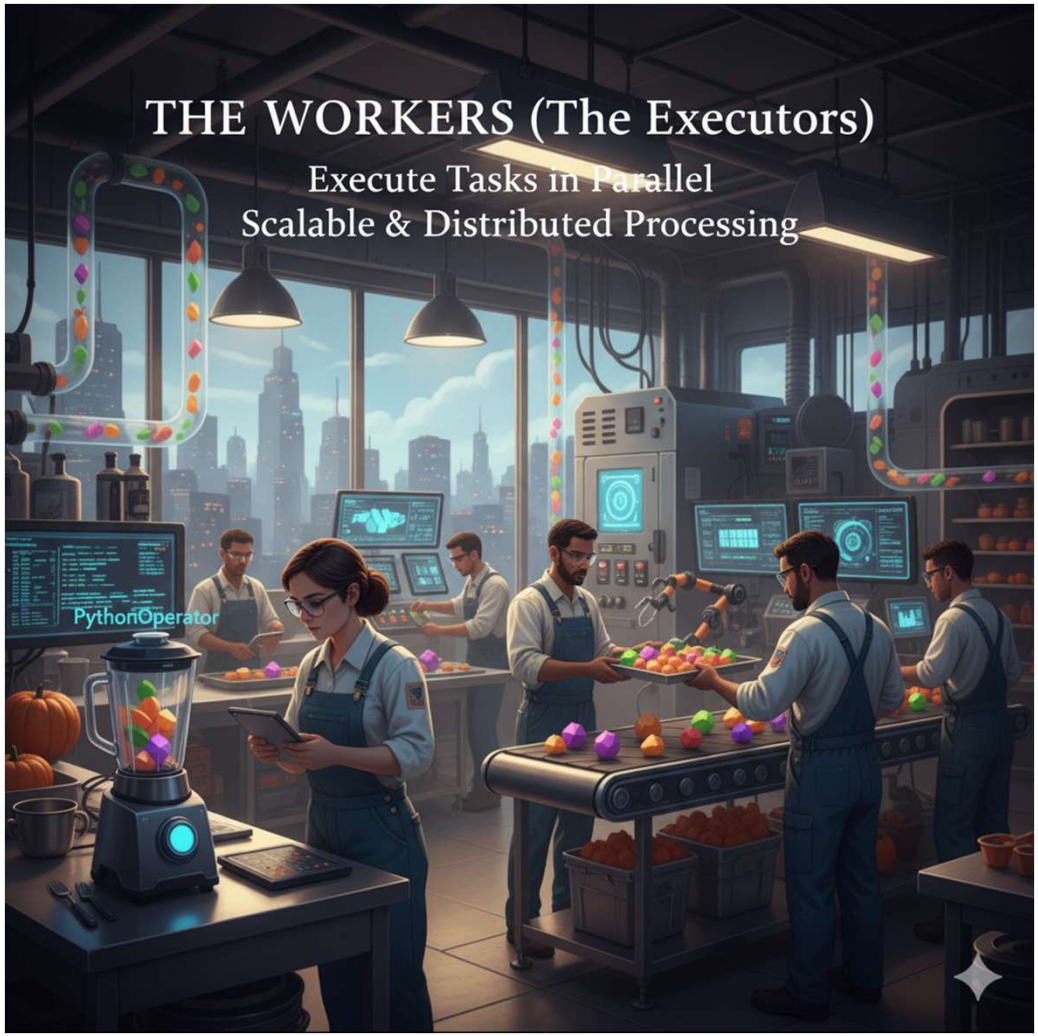
- Scheduler is the heart of Airflow.
- It continuously monitors all defined workflows.
- It triggers tasks based on dependencies and schedules.
- The scheduler acts as an alarm clock and project manager.

The Head Chef (The Airflow Scheduler) 👨‍🍳

- **Analogy:** The **Head Chef** constantly checks the order book, looks at all the recipes, and decides 'Okay, the Breakfast Smoothie line starts at 7 AM, then the Lunch Salad Prep at 10 AM.' They are the ultimate taskmaster, making sure everything kicks off when it's supposed to."
- **Airflow Term:** "In Airflow, this is the **Scheduler**. It's the persistent heart of Airflow, continuously monitoring all your defined workflows and triggering tasks based on their dependencies and schedules."
- Note: *The scheduler is the alarm clock and project manager rolled into one!*

Workers/Executors

- Workers are managed by Executors in Airflow.
- Executors assign specific tasks to the Workers.
- Airflow can scale Workers for simultaneous task execution.
- Workers perform the heavy lifting of data jobs.



The Busy Bees (The Workers / Executors) 👷

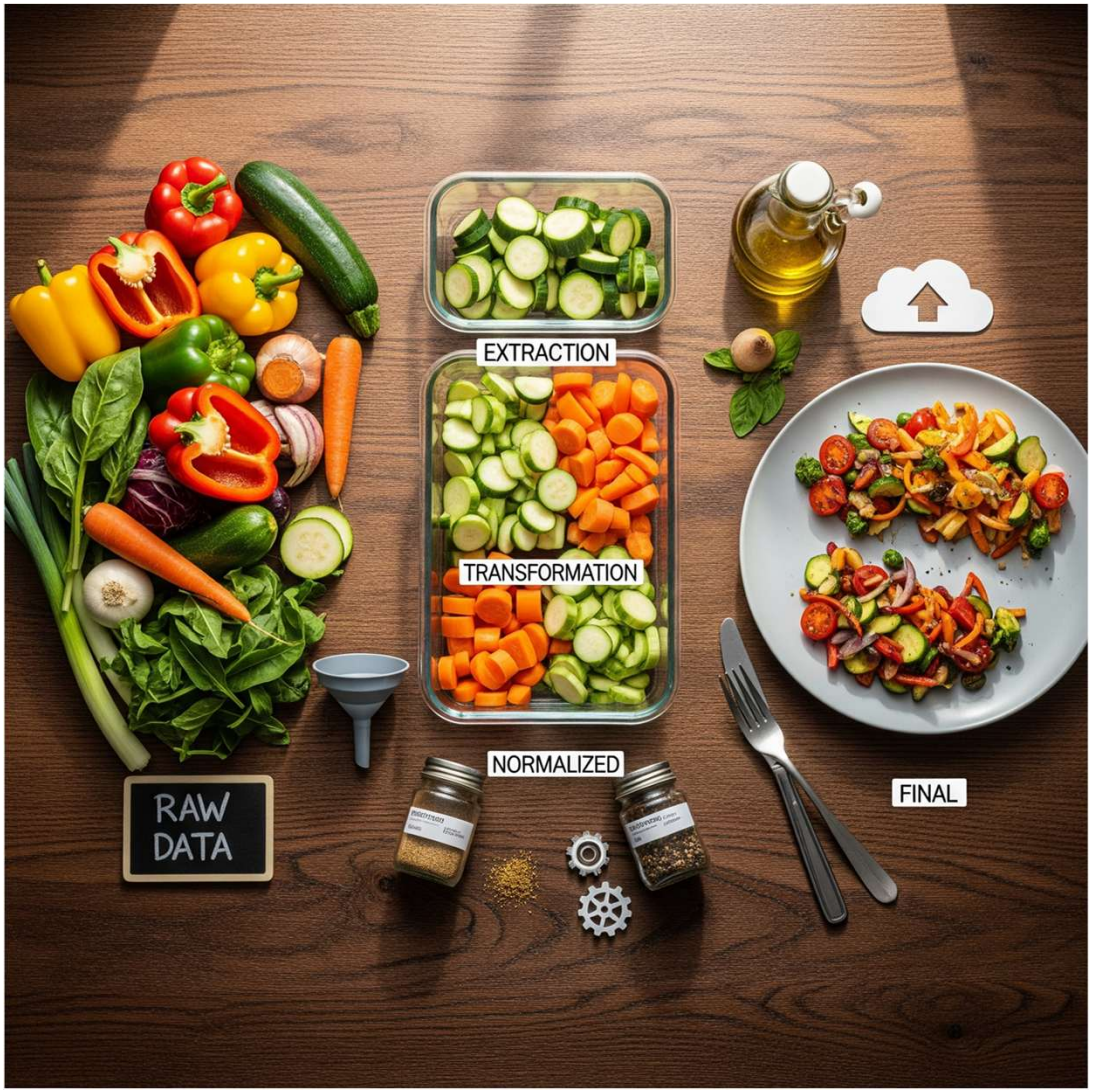
- **Analogy:** "Our factory has a team of **Busy Bees** – the actual hands-on workers. The Head Chef assigns them jobs: 'Worker A, go use the blender for the smoothie. Worker B, start the oven for the bread.' They do the heavy lifting."
- **Airflow Term:** "These are your **Workers**, managed by **Executors**. The Executor tells the Workers which task to run. Airflow can scale horizontally, meaning you can have many workers doing tasks simultaneously, just like a big, busy factory floor!"

- **Key takeaway:** *Workers are the tireless hands that get the data jobs done!*

ETL, ELT, and Apache Airflow

ETL: Prepared Meal Kit

- ETL creates a 'prepared meal kit' data.
- Airflow extracts raw data from various sources.
- Data transforms in a dedicated staging area.
- Cleaned data loads into the data warehouse.
- Airflow sequentially schedules and triggers each step.



ETL Explained

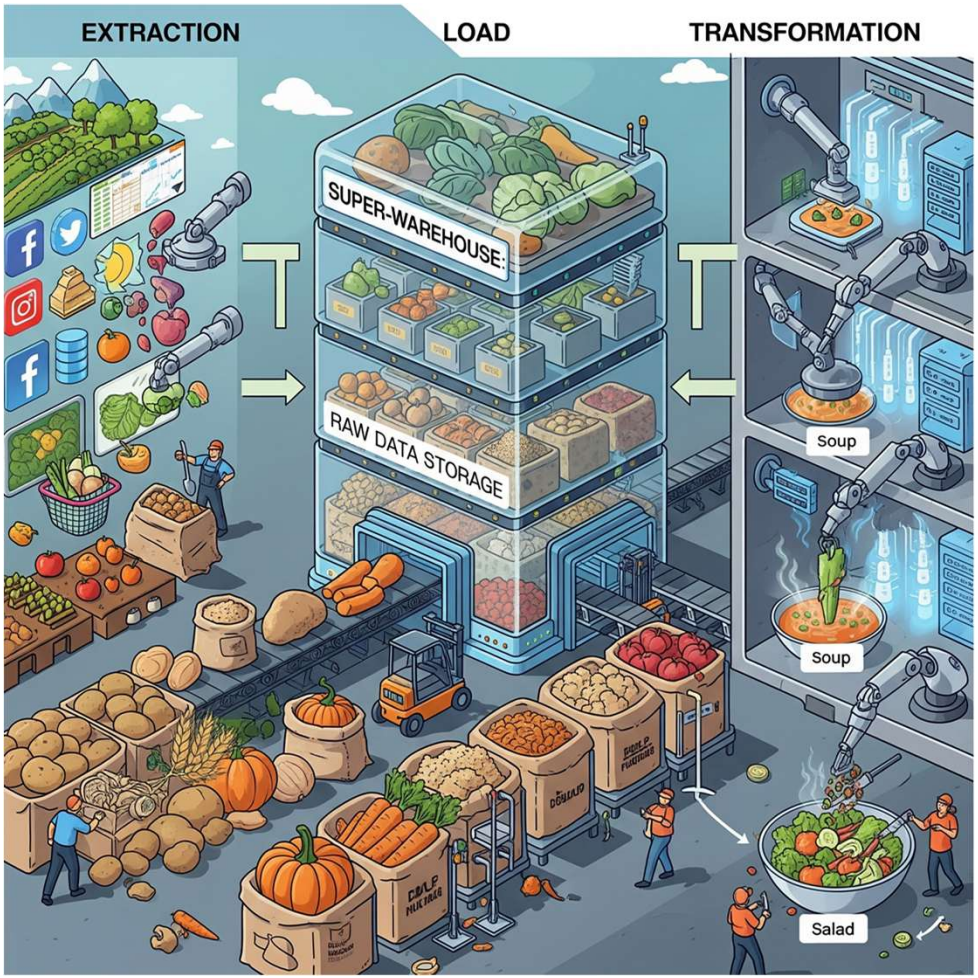
How Airflow Helps:

Airflow's job is to orchestrate this entire process. You write a **DAG (Directed Acyclic Graph)**, which is a blueprint for the workflow. This DAG contains a series of **tasks** that run in a specific order:

- **Extract Task:** Airflow's **Scheduler** triggers a task that connects to the source system (like a database or API) to pull the data.
- **Transform Task:** Once the extraction is complete, Airflow triggers a second task. This task runs code (for example, a Python script or SQL commands) on a separate server to clean, aggregate, and format the data.
- **Load Task:** After the transformation is done, Airflow triggers the final task. This task moves the now-clean, structured data into the target data warehouse.

Airflow ensures each step completes successfully before the next one begins. If a task fails, it can automatically retry it or send an alert.

ELT: Super-Warehouse Approach



- ELT handles vast amounts of diverse raw data.
- Airflow extracts and loads all raw ingredients quickly.
- Raw data is loaded directly into a Super-Warehouse.
- Transformations happen on-demand within the warehouse.
- Airflow orchestrates ingestion and triggers transformations.

ELT Explained

How Airflow Helps:

Airflow orchestrates the ELT pipeline as a series of tasks, but the order and location of the work are different:

- **Extract Task:** Airflow triggers a task to pull data from sources, similar to ETL, but the goal is to load it as quickly as possible.
- **Load Task:** The next task, triggered by Airflow, pushes the raw, unstructured data directly into a powerful data warehouse (like Snowflake or BigQuery).
- **Transform Task:** Once the data is loaded, Airflow triggers one or more transformation tasks. These tasks run SQL commands or scripts **directly inside the data warehouse** to perform the cleaning and structuring. Because the data warehouse is so powerful, this transformation is often much faster and can be done on-demand for different analysis needs.

Airflow's core role remains the same: it ensures the tasks run in the correct sequence, handles failures, and provides a clear view of the entire pipeline's status. It's the system that turns your data pipeline into an automated, reliable process.

Key Advantages:
Why Choose Apache Airflow?

- **Automation on Steroids:** "No more manual clicks or forgotten scripts! Airflow automates your factory from end to end."
- **Code-First Brilliance:** "Your recipes (DAGs) are just Python code. This means they are:
  - **Versionable:** Like saving different versions of your recipe.
  - **Testable:** You can test your recipes before the big cook-off.
  - **Maintainable:** Easy to update and fix."
- **The Crystal Ball (Powerful UI):** "Airflow comes with a beautiful web interface. It's your factory's control panel! See which lines are running, which jobs failed, view logs, and troubleshoot like a pro."
- **Reliability & Resilience:** "What if a blender breaks? Airflow can automatically retry a task, or alert you immediately. Your factory keeps running smoothly, even with hiccups."
- **Scale Up, Not Out (Scalability):** "Need to make more meals? Just add more workers (executors/workers)! Airflow is built to handle massive, growing workloads."
- **Community & Extensibility:** "Airflow has a huge, supportive community, and tons of pre-built 'appliances' (operators) for almost every data tool out there. It's like an ever-expanding kitchen gadget store!"