

# Building the business case for dbt Cloud

## About this guide

*This guide is based on extensive interviews with dozens of dbt Cloud customers, trends that surfaced in our recent State of Analytics Engineering Report, as well as on-the-ground learnings from the field teams at dbt Labs. We used these inputs to provide a roadmap that can be used by anyone trying to introduce dbt Cloud into their organization to help promote successful adoption at scale. Of course, your situation is going to be unique. We recommend pairing the resources and templates in this guide with a visit to the dbt Community Slack. Here you'll find 65,000+ data professionals who face similar problems. The community is passionate about improving analytics workflows, sharing best practices, and growing their careers. We look forward to engaging with you there, as well.*

## Why dbt Cloud?

Despite recent advancements in analytics—AI, cloud, open table formats, self-service—we've still yet to solve core data problems plaguing our organizations: data quality, data velocity, and cost optimization. This is in part because, despite its best efforts, the modern data stack has created data silos. These silos need to be centralized so organizations can build the holistic context needed to confidently embrace data at scale. The data control plane is where this centralization should happen.

**dbt Cloud is a data control plane that centralizes the metadata from your analytics workflow and**

**makes it actionable, so your teams can ship and use trusted data, faster.**

dbt Cloud is interoperable across data platforms and provides a universal, real-time view of what's happening across your data estate. Its built-in user experiences are designed to accelerate data delivery and help users understand and improve data quality and compute costs along the way. dbt Cloud offers varied user interfaces and integrations so that stakeholders of all technical stripes can participate in the data workflow and have the trusted insights needed to translate data into strategic decisions.



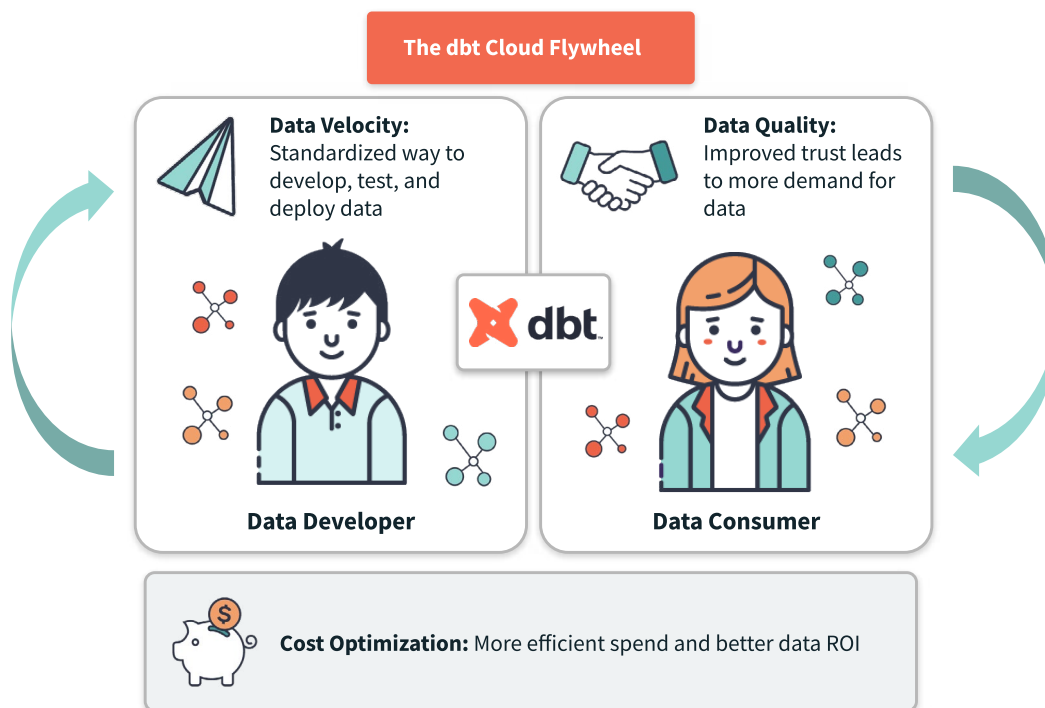
# Scale analytics successfully with dbt Cloud

Everyone wants to “do more with data” and become a “data-driven company”, but there are common roadblocks that derail these initiatives at scale. Data teams are still grappling with issues related to data quality, data velocity, and ambiguous data ownership. This is shown in [our most recent State of Analytics Engineering report](#).

The adoption of dbt Cloud by data teams has led to significant improvements in data speed, quality, and trust. These teams have reported impressive results, including [50x multipliers](#) in data velocity and [dramatic increases in data quality and data trust](#) as evidenced by the 30-80%+ reduction in data inconsistencies and [improved eNPS scores](#). By implementing dbt Cloud as their standard, these teams are experiencing tangible benefits.

It’s time to help more teams at more organizations realize similar, outsized results.

This paper includes detailed, step-by-step guidance designed to help you successfully adopt and ramp dbt Cloud within your organization. We include templates for mapping technology solutions to business goals, parameters to consider as you evaluate proposed solutions, considerations for scoping a proof of value, and more—so you can have a measured, structured approach to securing the support you need to adopt dbt Cloud at your organization. The tips, templates, and tales in these pages have been gathered via dozens of interviews with dbt Cloud customers—several of whom you’ll meet below.



# Align with company objectives

When seeking budget or headcount approval, the best place to start is where the business wants to end: your company objectives. Mapping technical or people problems to the organizational goals

they jeopardize will not only help you speak the same language as non-data stakeholders, but also provide leverage in your request for resources.

## Template: Map analytics workflow problems to business goals

Contributing KRIs	Owner	Current	Goal	Contributing factors
Increase upsell by 30%	Mktg	10%	30%	Requested changes to account views take three weeks to fulfill
				Dashboards take on average 30 seconds to load, frustrating analysts and reps
				Disparities across on-prem & cloud data reduces trust in account information

In the above example, company upsell rates haven't progressed as expected. Polling key stakeholders on why they believe they're behind on achieving this goal will help connect business

context to data roots. Use trends identified in these interviews to fill in the "contributing factors" column. This will highlight the need for additional data resources to hit these company-wide goals.

# Benchmark, and propose ideal outcomes

Consider how improvements to data team work-flows can improve the issues outlined in Step 1. Add quantified benchmarks and define desired outcomes.

## Template: Benchmark current state and set measurable desired outcomes

Problem		Desired outcome
Requested changes to account views take three weeks to fulfill	→	Enable analysts to own transformation workflows and deliver data in one week or less
Dashboards take an average of thirty seconds to load, frustrating analysts and reps	→	Eliminate redundant systems and automate orchestration from source to dashboard to achieve sub-10-second load time
Disparities across on-prem and cloud data reduces trust in account information	→	Standardize on a cloud solution that includes semantic capabilities and conduct routine testing for 100% parity

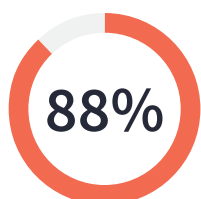
### Step 3

## Identify solutions that support desired outcomes

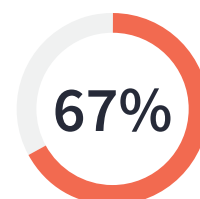
If you're evaluating dbt Cloud against other solutions, use your list of desired outcomes to inform a feature comparison matrix. This will help you narrow the field of potential contenders you'll explore in a subsequent proof of value ("POV").

During customer interviews, we discovered that data leaders faced challenges in gaining support for migrating to an analytics engineering

workflow. The primary reason for this was a perceived "switching cost" associated with moving away from legacy technology or existing approaches. For this reason, we've chosen to populate our comparison matrix with features of a typical "homegrown" transformation solution. We've also shaped subsequent cost/value calculators to consider company uplift along with total cost of ownership.



*Percentage of data practitioners that cited "legacy tech debt" as top challenge in implementing analytics engineering workflows*



*Percentage of dbt Cloud users that said enabling non-engineers to absorb more development work has been a top benefit of implementing analytics engineering workflows*

## Legacy in-house tooling vs. dbt Cloud

	Legacy in-house	dbt Cloud
Modularity and portability	Stored procedures with unknown ownership and dependencies	<ul style="list-style-type: none"> <li>• SQL and Jinja for macros</li> <li>• Pre-built packages</li> <li>• Data contracts and cross-team collaboration with dbt Mesh</li> <li>• Clear lineage of objects and columns (DAG)</li> </ul>
Accessible authoring	Combination of Python and other scripting languages	<ul style="list-style-type: none"> <li>• SQL and Python supported languages</li> <li>• User-friendly in-browser IDE that uses SQL SELECT statements for development</li> <li>• Cloud CLI with VSCode plug-in for more technical developers</li> <li>• Low-code visual editing experience powered with governed SQL under the hood</li> </ul>
Job scheduling, monitoring, and alerting	<ul style="list-style-type: none"> <li>• Multiple tasks to invoke jobs</li> <li>• Would require additional orchestration tools and manual processes</li> </ul>	<ul style="list-style-type: none"> <li>• Out-of-the-box scheduler and job monitoring</li> <li>• Alerting for job status, errors, warnings</li> <li>• Data freshness checks</li> <li>• Robust logging</li> <li>• API for service integration</li> </ul>
Version control	No standardized version control	<ul style="list-style-type: none"> <li>• Out-of-the-box integration with git providers (GitHub, GitLab, Azure DevOps, etc.)</li> <li>• Enables a code promotion path from development to QA to production</li> <li>• Changed code can automatically trigger CI test runs</li> <li>• Compare changes to promote high data quality without driving up costs</li> </ul>
Documentation	Ad hoc	<ul style="list-style-type: none"> <li>• dbt Explorer generates interactive, automated documentation with visual catalog to navigate dependencies</li> <li>• Automated and always up-to-date to eliminate human error</li> <li>• Understand lineage down to the column-level with multiple lenses to get context into environment status</li> </ul>
Dependency management	Indiscernible lineage	<ul style="list-style-type: none"> <li>• DAG generates as you code for up-to-date lineage</li> <li>• DAG in the IDE for seamless authoring</li> <li>• Column-level lineage in dbt Explorer</li> <li>• Auto-exposures for downstream BI dashboards</li> </ul>
Data quality	Ad hoc testing	<ul style="list-style-type: none"> <li>• Automated testing of code logic and expected code performance</li> <li>• Snapshots to record changes to mutable tables</li> <li>• Build and deliver consistent metrics to any endpoint with dbt Semantic Layer</li> <li>• Troubleshoot issues quickly with column-level lineage</li> <li>• Recommendations to improve reliability and test coverage</li> <li>• Out-of-the-box tests, testing packages, unit testing, and custom testing capabilities</li> </ul>
Cost optimization	Difficult to identify inefficiencies and optimize	<ul style="list-style-type: none"> <li>• Defer to production to optimize compute</li> <li>• Understand relative popularity/usage of models to fine-tune or decommission</li> <li>• Identify slow or long-running models with signals on how to improve</li> <li>• Reduce compute time with parallel execution</li> <li>• Run tests on changes (Slim CI) and validate model logic before it builds (unit tests)</li> </ul>
Implementation & support	Engineering team	<ul style="list-style-type: none"> <li>• dbt Learn, rapid onboarding</li> <li>• Professional services for support, audit, and refactoring</li> <li>• Vibrant community of 100K+ members</li> </ul>
Licensing cost	Requires additional engineering support for maintenance	<ul style="list-style-type: none"> <li>• No additional infrastructure cost</li> <li>• Annual license</li> </ul>
Security	<ul style="list-style-type: none"> <li>• On-premises, limited external networking requirements</li> <li>• Custom-developed integrations</li> </ul>	<ul style="list-style-type: none"> <li>• ELT architecture reduces risk/surface area</li> <li>• Enterprise-grade security features like SSO, RBAC, and SOC-2 compliance</li> </ul>
Governance	<ul style="list-style-type: none"> <li>• Copy-paste into version control, if the developer remembers</li> <li>• Manually copy code into production for deployments</li> <li>• No CI</li> </ul>	<ul style="list-style-type: none"> <li>• Version control is central to dbt Cloud (native integrations to GitLab, GitHub, ADO, etc.)</li> <li>• Data contracts</li> <li>• Terraform provider and robust APIs</li> <li>• Apply linting and code formatting rules</li> </ul>

# Define a time-bound proof of value

Once you've narrowed your prospective solutions, it's time to put each one to the test. Your POV should be structured to show a demonstrable impact on the problem(s) prioritized in Step 1. It should consider various use cases that might arise for your teams or in your workflow in your efforts to resolve the problems you've identified for improvement. The below list contains some recommended use-case examples.

For each, consider things like total time to perform the task, cost of context-switching, resources required, stakeholders involved, cycles to completion, process governance, and quality of outcomes.

## **Use case examples:**

1. *A data engineer updates a customer account model to reflect changes in product type definitions*
2. *A job/model refresh fails midway through the pipeline*
3. *A financial analyst identifies the inaccuracy of the account's ARR, proposes a fix, and tests*
4. *An analytics engineer needs to identify and resolve the root cause of a pipeline failure*
5. *Business user requires an additional "group-by" dimension for specific analysis*
6. *Analyst notices that the core dataset is missing critical attributes*

# Tabulate results

Record key results against your baseline for two weeks. We've populated the table with example inputs based on conversations with dbt Cloud customers, but your own may differ.

## Template: Key results after two week proof of value test

	KPI	Legacy in-house	dbt Cloud
<b>Baseline</b>	# of practitioners developing	4 Engineers (E)	2 Engineer (E), 2 Analysts (A)
	Total # hours spent in development	60 (E) hours	40 (E) hours; 20 (A) hours Analysts can develop in the Cloud IDE or Visual Editor, enabling engineers to serve as mentors
	Average compute hours for transformation workflows	40 hours	20 hours dbt Cloud parallelizes transformations to reduce run time
<b>Accessibility</b> Analysts self-serve validation and change requests to enable engineers to work on higher leverage tasks	# hours spent mapping lineage, answering questions, understanding dependencies	30 (E) hours	5 (E) hours, 10 (A) hours
<b>Accessibility key result</b>	<b>Time servicing ad hoc requests</b>	<b>3 weeks</b>	<b>1 week</b>
<b>Velocity</b> User-friendly IDE enables more people to develop; DDL/DML abstraction speeds development pipelines	<b># data marts developed*</b> *may extend beyond the POC period	1	4 4x productivity due to more developers enabled via Cloud IDE and visual editor
	<b># of developers contributing</b>	10	30 more developers enabled via Cloud IDE and visual editor
	<b># hours servicing ad-hoc requests</b>	15 (E) hours	2 (A) hr Analysts enabled to self-serve requests
	<b># hours spent managing infrastructure</b>	5 (E) hours	0 dbt Cloud is a managed service
<b>Velocity KR</b>	<b>Data product velocity improvement</b>	N/A	30-50% Increased speed to delivery and productivity improvements
<b>Quality</b> Automated testing and alerting. Snapshots to record changes and automated lineage documentation	<b># models tested in 2 weeks</b>	0/5 Break-fix work only done reactively in production	50/50 Construct and automate custom tests with the ability to adjust alert sensitivity
	<b># hours spent fixing broken pipelines</b>	20 (E) hours	2 (A) hours Alerting for test failure and unified experience for troubleshooting reduces time to fix prior to production
	<b>Data downtime</b>	90% availability	99.9% availability



Quality key result	Support tickets for data validation or data issues	15/week	2/week
<b>Governance</b> Documentation generated automatically, and discoverable in interactive data catalog dbt Explorer	# hours spent documenting	5 (E) hours	1 (A) hour Data assets are documented as a part of the development process. Column descriptions can be auto-generated.
	# hours spent investigating data dependencies	20 (E) hours	0 DAG generated as you code
	# hours spent validating metric accuracy	5 (E) hours	1 (A) hours Analysts enabled to self-serve requests
Governance key result	Support tickets for documentation	2/week	0/week
<b>Total hours expended</b>		<b>160 (E) hours</b>	<b>45 (E) hours; 36 (A) hours</b>

## Strategic potential of data democratization

By expanding the pool of individuals who can safely contribute to and self-serve information about transformation workflows, while simultaneously automating data quality controls, data teams are freed up to focus on higher-leverage work such as platform expansion, pipeline optimization, or real-time processing.

These are the value-generating activities that could significantly accelerate critical business decisions. The following table includes additional examples of company uplift translated from the calculations above. It's organized by risk reduction, growth/revenue impact, and operational efficiency.

## Template: dbt Cloud impact mapped to organizational goals

Organizational uplift	dbt Cloud impact
<b>Risk reduction</b>	<ul style="list-style-type: none"> <li>100% of data models now tested prior to production (0/5 → 50/50)</li> <li>10% reduction in data downtime (dbt Cloud has a 99.9% uptime guarantee)</li> </ul>
<b>Revenue growth</b>	<ul style="list-style-type: none"> <li>50% increase in speed to market (ship data products faster to build competitive strategies that drive revenue)</li> </ul>
<b>Operational efficiency</b>	<ul style="list-style-type: none"> <li>50% productivity increase for data team (160 hour → 81 hours: reduction in hours engineering spends on data development, pipeline resolution, and responding to ad-hoc requests)</li> <li>4x increase in data marts developed (1 → 4: more developers contributing to development workflow)</li> </ul>

# Project cost comparison

Your POV results can be used to project annual costs. Use the average salary for analysts and engineers (or anyone contributing to data development) to calculate cost per number of hours expended. The below is based on an

organization with a data team of about five. Also, check out the interactive [Total Cost of Ownership Calculator](#) to build a model specific to your business situation.

## Proof of value inputs

### Monthly costs

dbt Cloud requires fewer labor and compute hours—even while supporting the creation of more data marts—than a legacy approach.

	Legacy in-house	dbt Cloud
# data marts developed	1	4
Labor expended	160 (Engineer) hours	45 (Engineer) hours; 36 (Analyst) hours
Labor cost*	E: $160 * \$78/\text{hr} = \$12,480$	E: $45 * \$78/\text{hr} = \$3,510$ A: $36 * \$47/\text{hr} = \$1,692$ = $\$5,202$
Average compute hours per week	100 hours	50 hours
Average compute cost**	$100 * \$16/\text{hr} * 4 \text{ weeks} = \$6,400$	$50 * \$16/\text{hr} * 4 \text{ weeks} = \$3,200$

\* Salary assumptions: Average data engineer salary = \$150k USD (\$78/hr). Average analyst salary = \$90k USD (\$47/hr).

\*\* Assuming average compute cost of \$16/hr

## Annual costs

The table below converts the monthly costs into annual costs for labor, warehouse compute costs, and includes infrastructure and licensing costs

for each approach. Even with a higher annual licensing cost, the total cost of ownership is much lower with dbt Cloud.

	Legacy in-house	dbt Cloud
# data marts developed annually	12	48
Annual labor costs	$\$12,480 * 12 = \$150,000$	$\$5,202 * 12 = \$62,424$
Annual transformation infra costs	$\$15,000$	0
Annual license costs	0	~\$24,000
Annual warehouse infrastructure costs	$(\$6,400 * 12) = \$76,800$	$(\$3,200 * 12) = \$38,400$
<b>Total annual costs</b> (infrastructure/licensing + transformation labor hours + warehouse infrastructure)	<b>\$241,800</b>	<b>\$124,824</b>

## Total annual cost per data mart / use case

The table below takes the annual costs calculated in the table above and derives the total cost per data mart or use case. As mentioned above, standardizing on dbt Cloud makes it more feasible






for teams to scalably create more data marts. The cost per data mart is lower with dbt Cloud than with legacy approaches.

# data marts developed annually	12	48
<b>Total annual costs</b> (transformation labor hours + infrastructure + licensing)	<b>\$241,800</b>	<b>\$124,824</b>
Cost (including salaried hours) per data mart	$\$241,800 / 12 = \$20,150$	$\$124,824 / 48 = \$2,600$
Cost (excluding salaried hours) per data mart	$((\$15,000 + 76,800) / 12) = \$7,650$	$((\$24k + 38,400) / 48) = \$1,300$

# Present findings

Once you've finished your analysis, you should be prepared to present summary findings. The tables above can be used to derive the following key metrics Cloud customers have found to be the most impactful:

## Key metrics dbt Cloud customers used to build the business case

				
<p><b>% increase in data uptime</b></p> <p><i>JetBlue reduced maintenance windows from eight to zero hours</i></p>	<p><b>% increase in data production velocity</b></p> <p><i>Siemens increased data delivery time by 93%</i></p>	<p><b>% increase in company data trust scores</b></p> <p><i>Safety Culture increased data team eNPS by +89</i></p>	<p><b>% increase in engineering productivity</b></p> <p><i>Forrester TEI report found productivity increased by 30%</i></p>	<p><b>% reduction in change request tickets</b></p> <p><i>Conde Nast increased self-service 30%</i></p>

# Create the implementation plan

If your project plan includes migrating off an existing solution, adding an implementation timeline can eliminate last minute questions or roadblocks related to resource availability. Consider a phased approach to migration that

includes a gradually broadening scope of dbt functionality. Follow the [Analytics Development Lifecycle](#) (ADLC) as you develop a longer term implementation strategy.



# Prepare to ramp

While enabling more users to get involved in the data development workflow might sound daunting, dbt Cloud accelerates ramp time across many vectors:



## dbt Cloud IDE

*A unified, web-based interface for building, testing, running, and version-controlling dbt projects. It compiles dbt code into SQL and executes it directly on your data platform. With built-in features like dbt Assist, syntax highlighting, and code formatting, it promotes fast and well-governed data development and empowers more users to safely participate in data development workflows.*



## dbt Community

*dbt fosters a passionate, vibrant, and growing community of over 100,000 members who are eager to share best practices, use cases, and open-source packages designed to make your analytics engineering workflows successful. Any questions you may have about your deployment or use case have likely already been asked...and answered. It's easy to get involved and contribute to the conversation.*



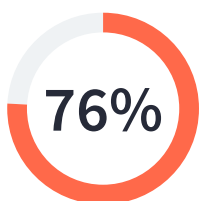
## Automated documentation with dbt Explorer

*Accelerate knowledge sharing with dbt Cloud's built-in documentation catalog, dbt Explorer. Detailed context about your data estate is automatically updated after a fully successful job run, ensuring accuracy, relevance, and discoverability. Using dbt Explorer, you can navigate your end-to-end DAG, understand lineage and dependencies, and have the context necessary to troubleshoot and optimize your pipelines.*

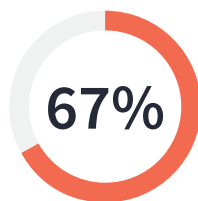
Combined, these features significantly reduce ramp time for anyone less familiar with the command line or git best practices. 100% of dbt users surveyed estimated that non-engineers needed less than six weeks to ramp, with 50% claiming one to three weeks was ample time. Additionally, the presence of a “power user” or

mentor was cited as the #1 most effective aid in further accelerating ramp.

dbt also offers a variety of [on-demand learning courses](#)—from fundamentals to advanced use cases—designed to help your teams get ramped and successful using dbt Cloud.



*Percentage of data practitioners that estimate non-engineers can ramp on dbt in less than 3 weeks*



*Percentage of data practitioners that say a “power user” mentor is the #1 indicator of success for new users*

## Summary

As illustrated above, dbt Cloud can dramatically improve data team efficiency through workflow automation and the democratization of development. By infusing software engineering best practices like testing, version control, and documentation into the analytics workflow,

and enabling more members of the data team to participate in all areas of development, each member of the data team can refocus on pursuing higher-leverage work that elevates the business as a whole.

## Top outcomes reported after implementing dbt Cloud:



**Improved pipeline velocity**



**Increased participation in data development**



**Reduced time spent troubleshooting and improved trust in data and data teams**



**Cory Dambert**

Director, BI & Analytics

**PROSPER**

### **Top feature**

Analysts from different groups were siloed from one another—relying on stored procedures that lacked any indication of ownership or dependencies. dbt Cloud provided a centralized development environment where even analysts less familiar with SQL could quickly reference transformation logic and lineage and skill up at their own pace.

### **Biggest benefit**

The biggest benefit I receive from dbt is analyst self-service. It's normal for data engineers to handle ad hoc requests for data parity, but on the whole that work isn't moving the organization forward. If analysts can be empowered to write code the right way, using tools to maintain a proper process flow, my team gets time back to work on optimization and investigate new solutions. That's huge.

## Why dbt Cloud?

The solution we had built in-house was great, but as more layers of code were added, we started seeing discrepancies between data sets. When that happens, you lose all trust. Unwinding unnecessary overhead from maintaining a system that just couldn't scale, and rebuilding our pipeline with dbt Cloud at the center, changed the game for us.



---

## Do data differently.

To see how dbt Cloud can support your business case, connect with our team for a quick, personalized demo: [getdbt.com/contact](https://getdbt.com/contact)

