

Table of Contents

[Guide to the dbt Cloud Architect Certification 5](#_heading=h.omj3b7buxu9m)

[Context 5](#_heading=h.f5frex4fp7er)

[1. Configure data warehouse connections 5](#_heading=h.spup1p3g1x6b)

[2. Configure Git connections 8](#_heading=h.p20y9fqntlcp)

[3. Manage dbt Cloud environments 9](#_heading=h.sm2h90au8pcd)

[4. Define and orchestrate jobs 11](#_heading=h.76i11s70lap3)

[5. Security and licensing 14](#_heading=h.raywhovgsojy)

[6. Monitoring and Alerting 15](#_heading=h.lsqu59oqyfhl)

[7. Implement dbt Mesh 16](#_heading=h.bj233ybbrnkm)

[8. Using dbt Explorer 18](#_heading=h.r7vmyjxn5ine)

[9. Key good practices 19](#_heading=h.qxauxkneyid9)

[10. Exam Questions 20](#_heading=h.gr900bh0vvhd)

# Guide to the dbt Cloud Architect Certification

## Context

This guide covers the essential configurations to master dbt Cloud according to the certification themes. The best practices are based on official documentation and advanced use cases.

## Configure data warehouse connections

Understand how to connect the warehouse*Example:* To connect dbt Cloud to BigQuery, you must provide the login information (project ID, default dataset, authentication method). This allows dbt Cloud to run SQL queries on your data.*Explanation:* The connection is essential for dbt to create, modify, or read the tables and views in your warehouse. Each warehouse (BigQuery, Snowflake, Databricks, etc.) has its own specific parameters to fill in.

Understanding Connections:  
dbt Cloud now supports account-level logins (Account Settings > Connections). A single connection can be reused by multiple projects to simplify management.

*Example:*  
 To connect dbt Cloud to Azure Databricks:

* Fill in the server hostname, port, and HTTP path in the "Settings" section of the dbt Cloud connection.
* Add the personal access token in the development credentials

Configure IP Whitelisting

*Example:*  
 In the dbt Cloud account settings, whitelist your company's IP range (192.168.1.0/24).  
*Explanation:*  
 This ensures that only authorized machines (e.g., those on your company's network) can access dbt Cloud, increasing the security of your data.

1. Go to *Account Settings → IP Restrictions*.
2. Create an "Allow" rule with your CIDR ranges (e.g. 192.0.2.0/24).
3. Enable restrictions after verification to prevent accidental lockout.

*Example:*  
 In *Account Settings > IP Restrictions*, click "Add Rule", select "Allow", and then add the CIDR range of your corporate VPN (e.g. 10.10.10.0/24). Click on "Save" and then "Enable IP restrictions"

Create and test a connection for the project

*Example:* After entering the connection parameters for Snowflake, click on "Test Connection". If the test is successful, the connection is recorded.*Explanation:* Testing the connection verifies that dbt Cloud can actually connect to the warehouse with the rights and settings provided before proceeding further.

OAuth authentication

*Example:D in the Google Cloud console, create an OAuth ID and copy the Client ID and Secret into the dbt Cloud configuration.Explanation:The Client ID and Secret are used by dbt Cloud to request OAuth access tokens from your identity provider.*

*Example:*

In the GCP console, create an OAuth ID, retrieve the client\_id and client\_secret, and then populate them in the OAuth configuration of dbt Cloud for BigQueryAfter entering the connection parameters (host, port, token), click "Test Connection". If the test is successful, click on "Save"

1. Enable *External OAuth* (Okta/Entra ID only).
2. Retrieve the Client ID and Client Secret from your IdP.
3. Bind the dbt connection to the OAuth configuration in *Connections → OAuth method*.

OAuth authentication to access data in the IDE

*Example:*  
 For BigQuery, set up OAuth in the Google Cloud Console, and then sign in via the "Sign in with Google" button in the dbt Cloud IDE.  
*Explanation:*  
 OAuth allows users to authenticate without sharing a password, using temporary tokens to access data in a secure manner.  
*Example:*   
For BigQuery, create an OAuth 2.0 client in the Google Cloud Console, configure the redirect URI provided by dbt Cloud, and then populate the client ID and secret in the dbt Cloud connection

## Configure Git connections

Embed Git repositories

* **GitHub** : Install the dbt app in your GitHub org and link developer accounts individually.
* **Other providers** : Use *Managed Repository* or *Git Clone* with SSH keys.

Required permissions

* Read/write access to *pull requests*, *content*, and *webhooks*.

Connect Git repository to dbt  
*Example :D the project settings, click "Connect to GitHub," allow dbt Cloud to access your account, and then select the repository. Explanation: This allows dbt Cloud to version your models, track changes, and collaborate via pull requests.*

*Example:*  
 In the dbt Cloud project settings, select "Git Integration", choose GitHub, authenticate, and then select the repository to connect

Configure integrations with Git providers  
*Example:P for GitLab, select "GitLab" as the provider, follow OAuth authentication, and then choose the repository to connect. Explanation: The direct integration allows dbt Cloud to automatically sync the code with your Git repository, making CI/CD easy.*

*Example:*  
 For GitLab, select "GitLab" in the integrations, allow dbt Cloud to access your GitLab account, and then choose the repository to associate with the project

## Manage dbt Cloud environments

Access Control  
Configure permissions per environment via *Groups & Licenses* :

1. Select a group (e.g. "Developers").
2. In *Access & permissions*, restrict access to specific environments.

Environment variables  
Set them in the environment settings to:

* Overcrowd target.name or threads.
* Manage multi-environment configurations (e.g. approx: prod).

Report (Deferral)  
Enable *Compare changes against* in jobs to reference models from other environments.

Understanding Environment Access Control  
*Example: Assign the "Developers" group to the development environment only, and the "Ops" group to production. Explanation: This limits who can run jobs or change settings depending on the environment, protecting production from accidental errors.*

*Example:*  
 Assign user groups to specific environments via "Groups & Licenses" to limit production access to certain members.

Determine when to use a service account  
*Example: Use a service account for scheduled jobs or CI/CD integrations, not for human users. Explanation: A service account ensures that automations continue to work even if an employee leaves the company.*

*Example:*  
 Use a service account for automated jobs or CI/CD integrations to avoid using personal credentials.

Rotate key-pair authentication via API  
*Example: Automate token creation/deletion via dbt Cloud API every 90 days. Explanation: Rotating keys regularly reduces the risk of unauthorized access in the event of a compromise.*

*Example:*  
 Automate the creation and deletion of service tokens via the dbt Cloud API to ensure that access keys are rotated evenly.

Understanding Environment Variables  
*Example:D set the variable DBT\_ENV\_CUSTOM\_ENV to prod or dev depending on the environment. Explanation: Variables allow you to dynamically adapt the behavior of dbt (e.g. choose the target schema or enable/disable features).*

*Example:*  
 Set the environment variable DBT\_ENV\_CUSTOM\_ENV to prod or dev to adapt the behavior of the models according to the environment.

Create a new dbt Cloud deployment environment  
*Example :D "Environments", click "Create Environment", name it "Staging", configure the branch and connection, and save. Explanation: Each environment (dev, staging, prod) allows you to separate the test and production phases, limiting the risks.*

*Example:*  
 In *Deploy > Environments*, click "Create Environment", choose "Deployment", configure the settings, and then "Save".

Set the default schema/dataset for the Example environment  
*:D in the production environment, set target\_schema to analytics\_prod. Explanation: This ensures that tables created by dbt are well isolated by environment, avoiding data collisions.*

*Example:*  
 In the environment configuration, fill in target\_schema to analytics\_prod for the production environment.

Understanding custom branches and their configuration  
*Example: Assign the feature/optimization branch to a test environment to validate this functionality before merging. Explanation: Working on dedicated branches allows you to test changes without impacting the main branch.*

*Example:*  
 Assign the feature/my-new-feature branch to a test environment to commit specific changes before merge.

Configure dbt to allow deferral to other environments  
*Example: Enable "Defer to production" on the CI job to reference production artifacts when testing new branches. Explanation: The deferral speeds up testing by reusing models already calculated in production, avoiding the need to recalculate everything.*

*Example:*  
 Enable the "Defer to environment" option in the CI job settings to reference the production job artifacts during testing.

## Define and orchestrate jobs

IC with deferral

*Example: Create a CI job on the main branch with "Defer to prod" enabled. Explanation: This allows you to validate only the changes made, relying on the production artifacts for the rest.*

*Example:*   
Create a CI job that uses the main branch, enable "Defer to production" so that the unmodified templates reference those of the production job.

1. Create a CI job in a dedicated environment.
2. Enable *Defer to production* to compare the state with the baseline environment.

Understanding the steps of a dbt job  
*Example: A job can include: dbt deps (install dependencies), dbt run (run models), dbt test (test results). Why: Each step has a specific role in the data transformation pipeline.*

*Example:*  
 A job can include the following steps:

1. DBT Seed
2. DBT Run
3. DBT Test

Schedule a job for a scheduled run  
*Example:D set a CRON schedule to start the job every day at 2 a.m. Explanation:Automation ensures that data is always up to date without manual intervention.*

*Example:*  
 In the job configuration, set a CRON schedule to run the job every day at 2 a.m.

Implement commands in the correct order  
*Example:Order: dbt deps → dbt seed → dbt run → dbt test. Explanation: This ensures that dependencies are installed, test data loaded, and then models run and tested.*

*Example:*   
Recommended order: dbt deps → dbt seed → dbt run → dbt test.

Create a new job dbt Cloud  
*Example :D "Jobs", click on "Create Job", name it, choose the environment, add the steps, and then save. Explanation: A job orchestrates the automatic execution of your dbt transformations.*

*Example:*  
 In *Deploy > Jobs*, click "Create Job", enter a name, select the environment, add the commands, and then "Save".

Configure advanced options (variables, threads, deferral, target, version)  
*Example:D the job settings, set threads to 4, override DBT\_TARGET to staging, force the dbt version to 1.7.0.Explanation:These options allow you to optimize performance and precisely control the behavior of the job.*

*Example:*  
 In the advanced section of the job, set threads to 8, override the environment variable DBT\_TARGET to staging, and force the dbt version to 1.7.0.

Generate documentation for a job  
*Example: Add dbt docs generate at the end of the job and enable "Generate docs on run". Explanation: This automatically updates the template documentation after each run.*

*Example:*  
 Add the dbt docs generate command to the end of the job and check "Generate docs on run" to update the documentation site.

Setting up job chainingExample  
*:D in the job settings, select "Run after job" and choose the previous job. Explanation: Chaining allows you to run jobs in a specific order, for example, load the data and then start the transformations.*

*Example:*  
 In the job settings, select "Run after job" and choose the production job to trigger that job after the previous one is successful.

Configure an Advanced CI  
*Example: Add conditional steps to run certain tests only on the main branch. Why: This helps optimize the CI pipeline and limit costly testing to critical branches.*

*Example:*  
 Add conditional steps in the CI job to run certain tests only on specific branches.

Configure self deferralExample  
*:Enable "Defer to last successful run" to have the CI job use its own previous artifacts. Explanation: This speeds up testing by avoiding recalculating all the models for each run.*

*Example:*  
 Enable the "Defer to last successful run" option so that the CI job relies on its own last successful run if needed.

Understand when to use the different types of deferral  
*Example:Use the deferral to the production for CI jobs, the self-deferral for maintenance jobs. Explanation: The choice depends on the context: reuse the production for consistency, or your own history for speed.*

*Example:*  
 Use the deferral to the prod for CI jobs, and the self-deferral for maintenance or quick fix jobs.

Job chaining  
Trigger one job after another via:

* *Run when another job finishes* in the triggers.
* Webhooks for integration with external tools.

Advanced settings

| **Parameter** | **Usage** |
| --- | --- |
| dbt\_version | Overload dbt Core version |
| target.name | Specify a target schema |
| Threads | Check Parallelism |
| generate\_docs | Enable documentation generation |

## Security and licensing

Create Service Tokens for the Example API*:D "Service Tokens", click on "New Token", name it "Terraform", assign the rights, copy the token. Explanation: Service tokens allow external tools to interact with the dbt Cloud API in a secure way.*

*Example:*  
 In Account Settings > Service Tokens, click on "+ New Token", name the token, assign the permissions, and then copy the generated token.

Assign permission sets*Example:When creating a token, choose "Job Admin" to allow job management. Explanation: Limiting permissions reduces the attack surface in the event of a compromise.*

*Example:* When creating a service token, select "Job Admin" to allow jobs to be executed and managed.

Create License Mappings*Example: Associate the "Data Analysts" group with the "Read Only" license. Explanation: This allows you to control who can edit, run, or only view dbt projects.*

*Example:* Associate users in the "Data Engineers" group with a "Developer" license to give them access to the IDE.

Add and remove users*Example:D "Users", click on "Add User", fill in the email, assign a role, then "Invite". Why: Managing users ensures that only authorized members access dbt Cloud.*

*Example:* In Account Settings > Users, click on "Add User", fill in the email, choose the role, then "Invite". To delete, click the trash next to the user.

Add an SSO application for dbt Cloud Enterprise*Example:D "SSO", configure a SAML application with Okta, by filling in the metadata from dbt Cloud.Explanation:SSO centralizes authentication and simplifies access management.*

*Example:* In Account Settings > SSO, select "Add SSO Application", configure the SAML or OIDC with your IdP (e.g. Okta).

Create and assign RBACs*Example: Create a "Viewer" role, assign it to the "Interns" group. Why: RBAC helps define precisely who has access to which features*

*Example:* Create a "Read Only" role, assign it to an "Interns" group to limit access to viewing jobs and artifacts.

Service tokens  
Create them for:

* Terraform/production integrations.
* API access without user context.

RBAC and SSO

1. Configure SSO (Okta/Entra ID) in *Account Settings → SSO*.
2. Map IdP groups to dbt permissions:
   * The group names must match exactly.
   * Use SCIM to automate provisioning.

## Monitoring and Alerting

Configure email notifications*Example: Add the team's email in the "Notifications" section of the job to receive an alert in case of failure. Explanation: Alerts allow you to react quickly in the event of a problem on critical jobs.*

*Example:* In the job configuration, add an email notification for "Failed" statuses to the data team's address.

Use webhooks for event-driven integration*Example: Configure a webhook that sends a POST to a Slack endpoint every time a job fails. Why: Webhooks allow you to integrate dbt Cloud with other monitoring or automation tools.*

*Example:* In Account Settings > Webhooks, create a webhook that sends a JSON POST to a Slack URL every time a job fails.

Notifications  
Turn on alerts:

* Email/Slack for job failures.
* Webhooks for events like job.run.failed.

Example of a webhook payload:

JSON

{

"eventType": "job.run.failed",

"data": {

"jobId": "123",

"runStatusCode": 30,

"runStatusMessage": "Connection timeout"

}

}

## Implement dbt Mesh

Create additional dbt projects*Example:Create a "marketing" project and a "finance" project in dbt Cloud, each with its own Git repository.Explanation:This separates business domains while making it easier to collaborate via dbt Mesh.*

*Example:* In dbt Cloud, click on "New Project", configure the Git repository and the target warehouse for each business domain.

Understand the relationship between environment types and cross-project references*Example: Configure the production environment for each project to be used as a source for cross-references. Why: Cross-references rely on artifacts in the target environment to ensure data consistency.*

*Example:* Set up a production environment in each project to serve as a source of truth during cross-references.

Use template governance*Example:D set access: public in the customer template of the marketing project to be accessible to the finance project. Why: Model governance controls the visibility and reuse of models across projects.*

*Example:* Set access: public in the dbt\_project.yml file to make certain templates accessible to other mesh projects.

Multiple projects

1. Create projects linked via *dependencies* in dbt\_project.yml.
2. Define a project-based production environment for cataloging.

Model governance

* Use access: public|private to control visibility.
* Define contracts (contract: enforced) to guarantee the schema.

## Using dbt Explorer

Use dbt Explorer for lineage, troubleshooting, optimization*Example :D Explorer, select a model, visualize its dependency graph, and look at execution logs to identify bottlenecks. Explanation: Exploring makes it easier to understand dependencies and optimize performance.*

*Example:* In dbt Explorer, select a model, visualize its dependency graph, identify slow patterns, and view execution logs.

Find public models and cross-project references*Example:D In Explorer, filter on "Public models" to see models exposed by other projects and add them as dependencies. Why: This accelerates the discovery and reuse of data assets across the organization.*

*Example:* In Explorer, filter on "Public models" to see models exposed by other projects and add them as dependencies in your own project.

Use cases:

* Visualize dependencies via the lineage graph.
* Identify slow or expensive patterns[16](https://www.getdbt.com/blog/navigate-and-understand-your-dbt-cloud-projects-with-dbt-explorer).
* Find cross-references with global search.

Optimization:  
 Filter by status: error or materialization: incremental to target improvements.

## Key good practices

* **Connections** : Reuse connections between projects to reduce maintenance.
* **Security** : Combine SSO, RBAC, and IP whitelisting for defense in depth.
* **Orchestration** : Chain jobs with triggers rather than schedules for efficiency.
* **Mesh** : Use namespaces to avoid name conflicts between projects.

## Exam Questions

Here is a series of questions and answers to help you prepare for the dbt Cloud Architect certification exam:

**Questions about configuring data warehouse connections**

**Question 1:** What are the essential steps to set up a data warehouse connection in dbt Cloud?

**Answer:** The steps include understanding the connection to the warehouse, setting up the IP whitelist, creating and testing a connection for the project, authenticating via OAuth to access the data in the IDE, and adding the Client ID and Secret for OAuth.

**Question 2:** When should you use a service account for authentication?

**Answer:** A service account should be used when you need automated, programmatic access, such as deployment jobs, API integrations, or when you want to avoid relying on individual user credentials.

**Questions about dbt Cloud environments**

**Question 3:** How do you configure deferral between environments in dbt Cloud?

**Answer:** The deferral allows dbt to reference models from another environment rather than reconstructing them. You must configure the source environment in the job settings, specify the reference environment, and use the appropriate commands such as --defer and --state.

**Question 4:** What is the difference between standard deferral and deferral choke?

**Answer:** The self deferral allows a job to reference its own previous executions, while the standard deferral references another environment or job. The self deferral is useful for incremental jobs that need to build on their own previous results.

**Security and licensing questions**

**Question 5:** How do you create and manage service tokens for API access?

**Answer:** Service tokens are created in the dbt Cloud account settings, with specific permissions assigned on a least-privilege basis. They enable programmatic access to the dbt Cloud API for automation and integrations.

**Question 6:** Explain the process of configuring Role-Based Access Control (RBAC).

**Answer:** RBAC involves creating custom roles with specific permissions, assigning those roles to users or groups, and managing license mappings. This allows granular control over access to projects, environments, and features1.

**Questions about jobs and orchestration**

**Question 7:** How do you set up a CI job with deferral?

**Answer:** A CI job with deferral requires setting up the production environment as a baseline, enabling the deferral in the CI job settings, and using commands like dbt run --defer --state to build only the modified models.

**Question 8:** What are the steps to set up job chaining?

**Answer:** Job chaining allows you to trigger a job after the completion of another. This is configured by specifying the triggering jobs in the target job settings, defining the trigger conditions (success, failure, or completion), and managing dependencies between jobs.

**Questions about monitoring and alerting**

**Question 9:** How do you set up webhooks for event integrations?

**Answer:** Webhooks are configured in the job notification settings, specifying the destination URL, the triggering events (success, failure, start), and the format of the data sent. They allow integration with external systems such as Slack, PagerDuty, or custom monitoring systems.

**Questions about dbt Mesh and cross-project references**

**Question 10:** How do you set up cross-project baselines in a dbt mesh environment?

**Answer:** Cross-project references require setting up multiple projects in dbt Cloud, defining relationships between environment types, using template governance to expose public templates, and properly configuring cross-project access permissions[1](https://www.getdbt.com/dbt-assets/dbt-certificate-study-guide-for-cloud-architect).

**Advanced technical questions**

**Question 11:** What are the different types of questions you may encounter in the exam?

**Answer:** The exam includes multiple-choice questions, fill-in-the-blank questions, matching questions, hotspot questions, building lists, and discrete options multiple-choice questions (DOMC).

**Question 12:** What are the technical specifications of the dbt Cloud Architect exam?

**Answer:** The exam is 2 hours long, includes 65 questions, requires a score of 65% to pass, costs $200, is available only in English, and the certification expires 2 years after graduation. It is monitored online and uses Caveon web browsers.

These questions cover the 8 main areas assessed in the dbt Cloud Architect certification exam and will help you prepare effectively for this advanced certification.