

Table des matières

[Guide pour la certification dbt Cloud Architect 5](#_Toc201296157)

[Contexte 5](#_Toc201296158)

[1. Configurer les connexions d'entrepôt de données 5](#_Toc201296159)

[Comprendre comment connecter l’entrepôt 5](#_Toc201296160)

[Comprendre les connexions 5](#_Toc201296161)

[Configurer la liste blanche IP 6](#_Toc201296162)

[Créer et tester une connexion pour le projet 6](#_Toc201296163)

[Authentification OAuth 6](#_Toc201296164)

[Authentification OAuth pour accéder aux données dans l’IDE 7](#_Toc201296165)

[2. Configurer les connexions Git 8](#_Toc201296166)

[Intégrer des dépôts Git 8](#_Toc201296167)

[Permissions requises 8](#_Toc201296168)

[Connecter le dépôt Git à dbt 8](#_Toc201296169)

[Configurer les intégrations avec les fournisseurs Git 8](#_Toc201296170)

[3. Gérer les environnements dbt Cloud 9](#_Toc201296171)

[Contrôle d'accès 9](#_Toc201296172)

[Variables d'environnement 9](#_Toc201296173)

[Report (Deferral) 9](#_Toc201296174)

[Comprendre le contrôle d’accès aux environnements 9](#_Toc201296175)

[Déterminer quand utiliser un compte de service 9](#_Toc201296176)

[Rotation de l’authentification par paire de clés via l’API 10](#_Toc201296177)

[Comprendre les variables d’environnement 10](#_Toc201296178)

[Créer un nouvel environnement de déploiement dbt Cloud 10](#_Toc201296179)

[Définir le schéma/dataset par défaut pour l’environnement 10](#_Toc201296180)

[Comprendre les branches personnalisées et leur configuration 11](#_Toc201296181)

[Configurer dbt pour permettre le deferral vers d’autres environnements 11](#_Toc201296182)

[4. Définir et orchestrer les jobs 12](#_Toc201296183)

[CI avec deferral 12](#_Toc201296184)

[Comprendre les étapes d’un job dbt 12](#_Toc201296185)

[Planifier un job pour une exécution programmée 12](#_Toc201296186)

[Implémenter les commandes dans le bon ordre 13](#_Toc201296187)

[Créer un nouveau job dbt Cloud 13](#_Toc201296188)

[Configurer les options avancées (variables, threads, deferral, target, version) 13](#_Toc201296189)

[Générer la documentation sur un job 13](#_Toc201296190)

[Configurer le chaînage de jobs (job chaining) 14](#_Toc201296191)

[Configurer un CI avancé 14](#_Toc201296192)

[Configurer l’auto-deferral (self deferral) 14](#_Toc201296193)

[Comprendre quand utiliser les différents types de deferral 14](#_Toc201296194)

[Chaînage de jobs 15](#_Toc201296195)

[Paramètres avancés 15](#_Toc201296196)

[5. Sécurité et licences 16](#_Toc201296197)

[Créer des tokens de service pour l’API 16](#_Toc201296198)

[Assigner des jeux de permissions 16](#_Toc201296199)

[Créer des mappings de licences 16](#_Toc201296200)

[Ajouter et supprimer des utilisateurs 16](#_Toc201296201)

[Ajouter une application SSO pour dbt Cloud Enterprise 17](#_Toc201296202)

[Créer et assigner des RBAC 17](#_Toc201296203)

[Jetons de service 17](#_Toc201296204)

[RBAC et SSO 17](#_Toc201296205)

[6. Surveillance et alertes 18](#_Toc201296206)

[Configurer les notifications email 18](#_Toc201296207)

[Utiliser les webhooks pour l’intégration événementielle 18](#_Toc201296208)

[Notifications 18](#_Toc201296209)

[Exemple de payload webhook : 18](#_Toc201296210)

[7. Implémenter dbt Mesh 19](#_Toc201296211)

[Créer des projets dbt supplémentaires 19](#_Toc201296212)

[Comprendre la relation entre types d’environnement et références cross-project 19](#_Toc201296213)

[Utiliser la gouvernance des modèles 19](#_Toc201296214)

[Projets multiples 20](#_Toc201296215)

[Gouvernance des modèles 20](#_Toc201296216)

[8. Utiliser dbt Explorer 21](#_Toc201296217)

[Utiliser dbt Explorer pour la lignée, le troubleshooting, l’optimisation 21](#_Toc201296218)

[Trouver les modèles publics et références cross-project 21](#_Toc201296219)

[Cas d'usage : 21](#_Toc201296220)

[Optimisation : 21](#_Toc201296221)

[9. Bonnes pratiques clés 22](#_Toc201296222)

[10. Questions d’examens 23](#_Toc201296223)

# Guide pour la certification dbt Cloud Architect

## Contexte

Ce guide couvre les configurations essentielles pour maîtriser dbt Cloud selon les thèmes de certification. Les pratiques recommandées s'appuient sur la documentation officielle et les cas d'usage avancés.

## Configurer les connexions d'entrepôt de données

Comprendre comment connecter l’entrepôt*Exemple :*Pour connecter dbt Cloud à BigQuery, vous devez fournir les informations de connexion (ID de projet, dataset par défaut, méthode d’authentification). Cela permet à dbt Cloud d’exécuter des requêtes SQL sur vos données.*Explication :*La connexion est essentielle pour que dbt puisse créer, modifier ou lire les tables et vues dans votre entrepôt. Chaque entrepôt (BigQuery, Snowflake, Databricks…) a ses propres paramètres spécifiques à renseigner.

Comprendre les connexions  
dbt Cloud gère désormais les connexions au niveau du compte (Account Settings > Connections). Une même connexion peut être réutilisée par plusieurs projets pour simplifier la gestion.

*Exemple :*  
Pour connecter dbt Cloud à Azure Databricks :

* Renseignez le nom d’hôte du serveur, le port et le chemin HTTP dans la section « Paramètres » de la connexion dbt Cloud.
* Ajoutez le jeton d’accès personnel dans les informations d’identification de développement

Configurer la liste blanche IP

*Exemple :*  
Dans les paramètres du compte dbt Cloud, ajoutez la plage IP de votre entreprise (192.168.1.0/24) à la liste blanche.  
*Explication :*  
Cela garantit que seules les machines autorisées (par exemple, celles du réseau de votre entreprise) peuvent accéder à dbt Cloud, renforçant ainsi la sécurité de vos données.

1. Accédez à *Account Settings → IP Restrictions*.
2. Créez une règle "Allow" avec vos plages CIDR (ex: 192.0.2.0/24).
3. Activez les restrictions après vérification pour éviter le verrouillage accidentel.

*Exemple :*  
Dans *Account Settings > IP Restrictions*, cliquez sur « Add Rule », sélectionnez « Allow », puis ajoutez la plage CIDR de votre VPN d’entreprise (ex : 10.10.10.0/24). Cliquez sur « Save » puis « Enable IP restrictions »

Créer et tester une connexion pour le projet

*Exemple :*Après avoir saisi les paramètres de connexion pour Snowflake, cliquez sur « Test Connection ». Si le test est concluant, la connexion est enregistrée.*Explication :*Tester la connexion permet de vérifier que dbt Cloud peut effectivement se connecter à l’entrepôt avec les droits et paramètres fournis avant d’aller plus loin.

Authentification OAuth

*Exemple :  
Dans la console Google Cloud, créez un identifiant OAuth et copiez le Client ID et Secret dans la configuration dbt Cloud.  
Explication :  
Le Client ID et Secret sont utilisés par dbt Cloud pour demander des jetons d’accès OAuth auprès de votre fournisseur d’identité.*

*Exemple :*

Dans la console GCP, créez un identifiant OAuth, récupérez le client\_id et le client\_secret, puis renseignez-les dans la configuration OAuth de dbt Cloud pour BigQuery  
Après avoir saisi les paramètres de connexion (hôte, port, jeton), cliquez sur « Tester la connexion ». Si le test est réussi, cliquez sur « Enregistrer »

1. Activez *External OAuth* (Okta/Entra ID uniquement).
2. Récupérez le Client ID et Client Secret depuis votre IdP.
3. Liez la connexion dbt à la configuration OAuth dans *Connections → OAuth method*.

Authentification OAuth pour accéder aux données dans l’IDE

*Exemple :*  
Pour BigQuery, configurez OAuth dans Google Cloud Console, puis connectez-vous via le bouton « Sign in with Google » dans l’IDE dbt Cloud.  
*Explication :*  
OAuth permet aux utilisateurs de s’authentifier sans partager de mot de passe, utilisant des jetons temporaires pour accéder aux données de manière sécurisée.  
*Exemple :*  
Pour BigQuery, créez un client OAuth 2.0 dans Google Cloud Console, configurez l’URI de redirection fourni par dbt Cloud, puis renseignez le client ID et le secret dans la connexion dbt Cloud

## Configurer les connexions Git

Intégrer des dépôts Git

* **GitHub** : Installez l'application dbt dans votre organisation GitHub et liez les comptes développeurs individuellement.
* **Autres fournisseurs** : Utilisez *Managed Repository* ou *Git Clone* avec des clés SSH.

Permissions requises

* Accès en lecture/écriture aux *pull requests*, *contenu*, et *webhooks*.

Connecter le dépôt Git à dbt  
*Exemple :  
Dans les paramètres du projet, cliquez sur « Connect to GitHub », autorisez dbt Cloud à accéder à votre compte, puis sélectionnez le dépôt.  
Explication :  
Cela permet à dbt Cloud de versionner vos modèles, de suivre les modifications et de collaborer via des pull requests.*

*Exemple :*  
Dans les paramètres du projet dbt Cloud, sélectionnez « Git Integration », choisissez GitHub, authentifiez-vous, puis sélectionnez le dépôt à connecter

Configurer les intégrations avec les fournisseurs Git  
*Exemple :  
Pour GitLab, sélectionnez « GitLab » comme fournisseur, suivez l’authentification OAuth, puis choisissez le dépôt à connecter.  
Explication :  
L’intégration directe permet à dbt Cloud de synchroniser automatiquement le code avec votre dépôt Git, facilitant le CI/CD.*

*Exemple :*  
Pour GitLab, sélectionnez « GitLab » dans les intégrations, autorisez dbt Cloud à accéder à votre compte GitLab, puis choisissez le dépôt à associer au projet

## Gérer les environnements dbt Cloud

Contrôle d'accès  
Configurez des permissions par environnement via *Groups & Licenses* :

1. Sélectionnez un groupe (ex: "Developers").
2. Dans *Access & permissions*, restreignez l'accès à des environnements spécifiques.

Variables d'environnement  
Définissez-les dans les paramètres d'environnement pour :

* Surcharger target.name ou threads.
* Gérer des configurations multi-environnements (ex: env: prod).

Report (Deferral)  
Activez *Compare changes against* dans les jobs pour référencer les modèles d'autres environnements.

Comprendre le contrôle d’accès aux environnements  
*Exemple :  
Attribuez le groupe « Développeurs » uniquement à l’environnement de développement, et le groupe « Ops » à la production.  
Explication :  
Cela limite qui peut exécuter des jobs ou modifier des paramètres selon l’environnement, protégeant la production des erreurs accidentelles.*

*Exemple :*  
Attribuez des groupes d’utilisateurs à des environnements spécifiques via « Groups & Licenses » pour limiter l’accès à la production à certains membres.

Déterminer quand utiliser un compte de service  
*Exemple :  
Utilisez un compte de service pour les jobs planifiés ou les intégrations CI/CD, pas pour les utilisateurs humains.  
Explication :  
Un compte de service garantit que les automatisations continuent de fonctionner même si un employé quitte l’entreprise.*

*Exemple :*  
Utilisez un compte de service pour les jobs automatisés ou les intégrations CI/CD afin d’éviter d’utiliser des identifiants personnels.

Rotation de l’authentification par paire de clés via l’API  
*Exemple :  
Automatisez la création/suppression de tokens via l’API dbt Cloud tous les 90 jours.  
Explication :  
La rotation régulière des clés réduit le risque d’accès non autorisé en cas de compromission.*

*Exemple :*  
Automatisez la création et la suppression de tokens de service via l’API dbt Cloud pour garantir la rotation régulière des clés d’accès.

Comprendre les variables d’environnement  
*Exemple :  
Définissez la variable DBT\_ENV\_CUSTOM\_ENV à prod ou dev selon l’environnement.  
Explication :  
Les variables permettent d’adapter dynamiquement le comportement de dbt (ex : choisir le schéma cible ou activer/désactiver des fonctionnalités).*

*Exemple :*  
Définissez la variable d’environnement DBT\_ENV\_CUSTOM\_ENV à prod ou dev pour adapter le comportement des modèles selon l’environnement.

Créer un nouvel environnement de déploiement dbt Cloud  
*Exemple :  
Dans « Environments », cliquez sur « Create Environment », nommez-le « Staging », configurez la branche et la connexion, puis enregistrez.  
Explication :  
Chaque environnement (dev, staging, prod) permet de séparer les phases de test et de production, limitant les risques.*

*Exemple :*  
Dans *Deploy > Environments*, cliquez sur « Create Environment », choisissez « Deployment », configurez les paramètres, puis « Save ».

Définir le schéma/dataset par défaut pour l’environnement  
*Exemple :  
Dans l’environnement de production, mettez target\_schema à analytics\_prod.  
Explication :  
Cela garantit que les tables créées par dbt sont bien isolées par environnement, évitant les collisions de données.*

*Exemple :*  
Dans la configuration de l’environnement, renseignez target\_schema à analytics\_prod pour l’environnement de production.

Comprendre les branches personnalisées et leur configuration  
*Exemple :  
Assignez la branche feature/optimisation à un environnement de test pour valider cette fonctionnalité avant de merger.  
Explication :  
Travailler sur des branches dédiées permet de tester des changements sans impacter la branche principale.*

*Exemple :*  
Assignez la branche feature/my-new-feature à un environnement de test pour valider des modifications spécifiques avant merge.

Configurer dbt pour permettre le deferral vers d’autres environnements  
*Exemple :  
Activez « Defer to production » sur le job CI pour référencer les artefacts de production lors du test de nouvelles branches.  
Explication :  
Le deferral accélère les tests en réutilisant les modèles déjà calculés en production, évitant de tout recalculer.*

*Exemple :*  
Activez l’option « Defer to environment » dans les paramètres du job CI pour référencer les artefacts du job de production lors des tests.

## Définir et orchestrer les jobs

CI avec deferral

*Exemple :  
Créez un job CI sur la branche main avec « Defer to prod » activé.  
Explication :  
Cela permet de valider uniquement les changements apportés, en s’appuyant sur les artefacts de production pour le reste.*

*Exemple :*  
Créez un job CI qui utilise la branche main, activez « Defer to production » pour que les modèles non modifiés référencent ceux du job de prod.

1. Créez un job CI dans un environnement dédié.
2. Activez *Defer to production* pour comparer l'état avec l'environnement de référence.

Comprendre les étapes d’un job dbt  
*Exemple :  
Un job peut inclure : dbt deps (installer les dépendances), dbt run (exécuter les modèles), dbt test (tester les résultats).  
Explication :  
Chaque étape a un rôle précis dans le pipeline de transformation de données.*

*Exemple :*  
Un job peut inclure les étapes suivantes :

1. dbt seed
2. dbt run
3. dbt test

Planifier un job pour une exécution programmée  
*Exemple :  
Définissez une planification CRON pour lancer le job tous les jours à 2h du matin.  
Explication :  
L’automatisation garantit que les données sont toujours à jour sans intervention manuelle.*

*Exemple :*  
Dans la configuration du job, définissez une planification CRON pour exécuter le job tous les jours à 2h du matin.

Implémenter les commandes dans le bon ordre  
*Exemple :  
Ordre : dbt deps → dbt seed → dbt run → dbt test.  
Explication :  
Cela garantit que les dépendances sont installées, les données de test chargées, puis les modèles exécutés et testés.*

*Exemple :*  
Ordre recommandé : dbt deps → dbt seed → dbt run → dbt test.

Créer un nouveau job dbt Cloud  
*Exemple :  
Dans « Jobs », cliquez sur « Create Job », nommez-le, choisissez l’environnement, ajoutez les étapes, puis sauvegardez.  
Explication :  
Un job orchestre l’exécution automatique de vos transformations dbt.*

*Exemple :*  
Dans *Deploy > Jobs*, cliquez sur « Create Job », donnez un nom, sélectionnez l’environnement, ajoutez les commandes, puis « Save ».

Configurer les options avancées (variables, threads, deferral, target, version)  
*Exemple :  
Dans les paramètres du job, définissez threads à 4, surchargez DBT\_TARGET à staging, forcez la version dbt à 1.7.0.  
Explication :  
Ces options permettent d’optimiser les performances et de contrôler précisément le comportement du job.*

*Exemple :*  
Dans la section avancée du job, définissez threads à 8, surchargez la variable d’environnement DBT\_TARGET à staging, et forcez la version dbt à 1.7.0.

Générer la documentation sur un job  
*Exemple :  
Ajoutez dbt docs generate à la fin du job et activez « Generate docs on run ».  
Explication :  
Cela met à jour automatiquement la documentation des modèles après chaque exécution.*

*Exemple :*  
Ajoutez la commande dbt docs generate à la fin du job et cochez « Generate docs on run » pour mettre à jour le site de documentation.

Configurer le chaînage de jobs (job chaining)  
*Exemple :  
Dans les paramètres du job, sélectionnez « Run after job » et choisissez le job précédent.  
Explication :  
Le chaînage permet d’exécuter des jobs dans un ordre précis, par exemple, charger les données puis lancer les transformations.*

*Exemple :*  
Dans les paramètres du job, sélectionnez « Run after job » et choisissez le job de production pour déclencher ce job après la réussite du précédent.

Configurer un CI avancé  
*Exemple :  
Ajoutez des étapes conditionnelles pour n’exécuter certains tests que sur la branche main.  
Explication :  
Cela permet d’optimiser le pipeline CI et de limiter les tests coûteux aux branches critiques.*

*Exemple :*  
Ajoutez des étapes conditionnelles dans le job CI pour n’exécuter certains tests que sur des branches spécifiques.

Configurer l’auto-deferral (self deferral)  
*Exemple :  
Activez « Defer to last successful run » pour que le job CI utilise ses propres artefacts précédents.  
Explication :  
Cela accélère les tests en évitant de recalculer tous les modèles à chaque run.*

*Exemple :*  
Activez l’option « Defer to last successful run » pour que le job CI s’appuie sur son propre dernier run réussi si besoin.

Comprendre quand utiliser les différents types de deferral  
*Exemple :  
Utilisez le deferral vers la prod pour les jobs CI, le self-deferral pour les jobs de maintenance.  
Explication :  
Le choix dépend du contexte : réutiliser la prod pour la cohérence, ou son propre historique pour la rapidité.*

*Exemple :*  
Utilisez le deferral vers la prod pour les jobs CI, et le self-deferral pour les jobs de maintenance ou de correction rapide.

Chaînage de jobs  
Déclenchez un job après un autre via :

* *Run when another job finishes* dans les triggers.
* Webhooks pour l'intégration avec des outils externes.

Paramètres avancés

| **Paramètre** | **Utilisation** |
| --- | --- |
| dbt\_version | Surcharger la version de dbt Core |
| target.name | Spécifier un schéma cible |
| threads | Contrôler le parallélisme |
| generate\_docs | Activer la génération de documentation |

## Sécurité et licences

Créer des tokens de service pour l’API*Exemple :  
Dans « Service Tokens », cliquez sur « New Token », nommez-le « Terraform », assignez les droits, copiez le token.  
Explication :  
Les tokens de service permettent à des outils externes d’interagir avec l’API dbt Cloud de façon sécurisée.*

*Exemple :*  
Dans Account Settings > Service Tokens, cliquez sur « + New Token », nommez le token, assignez les permissions, puis copiez le token généré.

Assigner des jeux de permissions*Exemple :  
Lors de la création d’un token, choisissez « Job Admin » pour permettre la gestion des jobs.  
Explication :  
Limiter les permissions réduit la surface d’attaque en cas de compromission.*

*Exemple :*Lors de la création d’un token de service, sélectionnez « Job Admin » pour permettre l’exécution et la gestion des jobs.

Créer des mappings de licences*Exemple :  
Associez le groupe « Data Analysts » à la licence « Read Only ».  
Explication :  
Cela permet de contrôler qui peut modifier, exécuter ou seulement consulter les projets dbt.*

*Exemple :*Associez les utilisateurs du groupe « Data Engineers » à une licence « Developer » pour leur donner accès à l’IDE.

Ajouter et supprimer des utilisateurs*Exemple :  
Dans « Users », cliquez sur « Add User », renseignez l’email, assignez un rôle, puis « Invite ».  
Explication :  
Gérer les utilisateurs garantit que seuls les membres autorisés accèdent à dbt Cloud.*

*Exemple :*Dans Account Settings > Users, cliquez sur « Add User », renseignez l’email, choisissez le rôle, puis « Invite ». Pour supprimer, cliquez sur la corbeille à côté de l’utilisateur.

Ajouter une application SSO pour dbt Cloud Enterprise*Exemple :  
Dans « SSO », configurez une application SAML avec Okta, en renseignant les métadonnées de dbt Cloud.  
Explication :  
Le SSO centralise l’authentification et simplifie la gestion des accès.*

*Exemple :*Dans Account Settings > SSO, sélectionnez « Add SSO Application », configurez le SAML ou OIDC avec votre IdP (ex : Okta).

Créer et assigner des RBAC*Exemple :  
Créez un rôle « Viewer », assignez-le au groupe « Stagiaires ».  
Explication :  
Le RBAC permet de définir précisément qui a accès à quelles fonctionnalités*

*Exemple :*Créez un rôle « Read Only », assignez-le à un groupe « Interns » pour limiter l’accès à la consultation des jobs et des artefacts.

Jetons de service  
Créez-les pour :

* Intégrations Terraform/production.
* Accès API sans contexte utilisateur.

RBAC et SSO

1. Configurez le SSO (Okta/Entra ID) dans *Account Settings → SSO*.
2. Mappez les groupes IdP aux permissions dbt :
   * Les noms de groupes doivent correspondre exactement.
   * Utilisez SCIM pour automatiser le provisionnement.

## Surveillance et alertes

Configurer les notifications email*Exemple :  
Ajoutez l’email de l’équipe dans la section « Notifications » du job pour recevoir une alerte en cas d’échec.  
Explication :  
Les alertes permettent de réagir rapidement en cas de problème sur les jobs critiques.*

*Exemple :*Dans la configuration du job, ajoutez une notification email pour les statuts « Failed » à l’adresse de l’équipe data.

Utiliser les webhooks pour l’intégration événementielle*Exemple :  
Configurez un webhook qui envoie un POST à un endpoint Slack à chaque échec de job.  
Explication :  
Les webhooks permettent d’intégrer dbt Cloud à d’autres outils de monitoring ou d’automatisation.*

*Exemple :*Dans Account Settings > Webhooks, créez un webhook qui envoie un POST JSON à une URL Slack à chaque échec de job.

Notifications  
Activez les alertes :

* E-mail/Slack pour les échecs de jobs.
* Webhooks pour des événements comme job.run.failed.

Exemple de payload webhook :

json

{

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

"data": {

"jobId": "123",

"runStatusCode": 30,

"runStatusMessage": "Connection timeout"

}

}

## Implémenter dbt Mesh

Créer des projets dbt supplémentaires*Exemple :  
Créez un projet « marketing » et un projet « finance » dans dbt Cloud, chacun avec son propre dépôt Git.  
Explication :  
Cela permet de séparer les domaines métier tout en facilitant la collaboration via dbt Mesh.*

*Exemple :*Dans dbt Cloud, cliquez sur « New Project », configurez le dépôt Git et l’entrepôt cible pour chaque domaine métier.

Comprendre la relation entre types d’environnement et références cross-project*Exemple :  
Configurez l’environnement de production de chaque projet pour servir de source lors des références croisées.  
Explication :  
Les références croisées s’appuient sur les artefacts de l’environnement cible pour garantir la cohérence des données.*

*Exemple :*Configurez un environnement de production dans chaque projet pour servir de source de vérité lors des références croisées.

Utiliser la gouvernance des modèles*Exemple :  
Définissez access: public dans le modèle customer du projet marketing pour qu’il soit accessible au projet finance.  
Explication :  
La gouvernance des modèles contrôle la visibilité et la réutilisation des modèles entre projets.*

*Exemple :*Définissez access: public dans le fichier dbt\_project.yml pour rendre certains modèles accessibles à d’autres projets mesh.

Projets multiples

1. Créez des projets liés via *dependencies* dans dbt\_project.yml.
2. Définissez un environnement de production par projet pour le catalogage.

Gouvernance des modèles

* Utilisez access: public|private pour contrôler la visibilité.
* Définissez des contrats (contract: enforced) pour garantir le schéma.

## Utiliser dbt Explorer

Utiliser dbt Explorer pour la lignée, le troubleshooting, l’optimisation*Exemple :  
Dans Explorer, sélectionnez un modèle, visualisez son graphe de dépendance et consultez les logs d’exécution pour identifier les goulets d’étranglement.  
Explication :  
Explorer facilite la compréhension des dépendances et l’optimisation des performances.*

*Exemple :*Dans dbt Explorer, sélectionnez un modèle, visualisez son graphe de dépendance, identifiez les modèles lents et consultez les logs d’exécution.

Trouver les modèles publics et références cross-project*Exemple :  
Dans Explorer, filtrez sur « Public models » pour voir les modèles exposés par d’autres projets et les ajouter comme dépendances.  
Explication :  
Cela accélère la découverte et la réutilisation des assets de données à l’échelle de l’organisation.*

*Exemple :*Dans Explorer, filtrez sur « Public models » pour voir les modèles exposés par d’autres projets et ajoutez-les comme dépendances dans votre propre projet.

Cas d'usage :

* Visualiser les dépendances via le graphe de lignage.
* Identifier les modèles lents ou coûteux[16](https://www.getdbt.com/blog/navigate-and-understand-your-dbt-cloud-projects-with-dbt-explorer).
* Trouver des références croisées avec la recherche globale.

Optimisation :  
Filtrez par status: error ou materialization: incremental pour cibler les améliorations.

## Bonnes pratiques clés

* **Connexions** : Réutilisez les connexions entre projets pour réduire la maintenance.
* **Sécurité** : Combinez SSO, RBAC et IP whitelisting pour une défense en profondeur.
* **Orchestration** : Chaînez les jobs avec des triggers plutôt que des schedules pour l'efficacité.
* **Mesh** : Utilisez des namespaces pour éviter les conflits de noms entre projets.

## Questions d’examens

Voici une série de questions et réponses pour vous préparer à l'examen de certification dbt Cloud Architect :

**Questions sur la configuration des connexions d'entrepôt de données**

**Question 1 :** Quelles sont les étapes essentielles pour configurer une connexion d'entrepôt de données dans dbt Cloud ?

**Réponse :** Les étapes incluent la compréhension de la connexion à l'entrepôt, la configuration de la liste blanche IP, la création et le test d'une connexion pour le projet, l'authentification via OAuth pour accéder aux données dans l'IDE, et l'ajout du Client ID et du Secret pour OAuth.

**Question 2 :** Dans quel contexte devez-vous utiliser un compte de service pour l'authentification ?

**Réponse :** Un compte de service doit être utilisé lorsque vous avez besoin d'un accès automatisé et programmatique, notamment pour les jobs de déploiement, les intégrations API, ou lorsque vous voulez éviter de dépendre des credentials d'utilisateurs individuels.

**Questions sur les environnements dbt Cloud**

**Question 3 :** Comment configurez-vous le deferral entre environnements dans dbt Cloud ?

**Réponse :** Le deferral permet à dbt de référencer des modèles d'un autre environnement plutôt que de les reconstruire. Vous devez configurer l'environnement source dans les paramètres de job, spécifier l'environnement de référence, et utiliser les commandes appropriées comme --defer et --state.

**Question 4 :** Quelle est la différence entre self deferral et deferral standard ?

**Réponse :** Le self deferral permet à un job de référencer ses propres exécutions précédentes, tandis que le deferral standard référence un autre environnement ou job. Le self deferral est utile pour les jobs incrémentaux qui doivent construire sur leurs propres résultats précédents.

**Questions sur la sécurité et les licences**

**Question 5 :** Comment créez-vous et gérez-vous les tokens de service pour l'accès API ?

**Réponse :** Les tokens de service sont créés dans les paramètres de compte dbt Cloud, avec des permissions spécifiques assignées selon le principe du moindre privilège. Ils permettent l'accès programmatique à l'API dbt Cloud pour l'automatisation et les intégrations.

**Question 6 :** Expliquez le processus de configuration du contrôle d'accès basé sur les rôles (RBAC).

**Réponse :** Le RBAC implique la création de rôles personnalisés avec des permissions spécifiques, l'assignation de ces rôles aux utilisateurs ou groupes, et la gestion des mappings de licences. Cela permet un contrôle granulaire de l'accès aux projets, environnements et fonctionnalités[1](https://www.getdbt.com/dbt-assets/dbt-certificate-study-guide-for-cloud-architect).

**Questions sur les jobs et l'orchestration**

**Question 7 :** Comment configurez-vous un job CI avec deferral ?

**Réponse :** Un job CI avec deferral nécessite la configuration de l'environnement de production comme référence, l'activation du deferral dans les paramètres du job CI, et l'utilisation de commandes comme dbt run --defer --state pour construire uniquement les modèles modifiés.

**Question 8 :** Quelles sont les étapes pour configurer le chaînage de jobs ?

**Réponse :** Le chaînage de jobs permet de déclencher un job après la completion d'un autre. Cela se configure en spécifiant les jobs déclencheurs dans les paramètres du job cible, définissant les conditions de déclenchement (succès, échec, ou completion), et en gérant les dépendances entre jobs.

**Questions sur le monitoring et les alertes**

**Question 9 :** Comment configurez-vous les webhooks pour les intégrations événementielles ?

**Réponse :** Les webhooks se configurent dans les paramètres de notification des jobs, en spécifiant l'URL de destination, les événements déclencheurs (succès, échec, début), et le format des données envoyées. Ils permettent l'intégration avec des systèmes externes comme Slack, PagerDuty, ou des systèmes de monitoring personnalisés.

**Questions sur dbt Mesh et les références cross-projet**

**Question 10 :** Comment configurez-vous des références cross-projet dans un environnement dbt Mesh ?

**Réponse :** Les références cross-projet nécessitent la configuration de projets multiples dans dbt Cloud, la définition des relations entre types d'environnements, l'utilisation de la gouvernance de modèles pour exposer les modèles publics, et la configuration appropriée des permissions d'accès entre projets[1](https://www.getdbt.com/dbt-assets/dbt-certificate-study-guide-for-cloud-architect).

**Questions techniques avancées**

**Question 11 :** Quels sont les différents types de questions que vous pouvez rencontrer dans l'examen ?

**Réponse :** L'examen comprend des questions à choix multiples, des questions à compléter, des questions d'appariement, des questions hotspot, des listes à construire, et des questions à choix multiples d'options discrètes (DOMC).

**Question 12 :** Quelles sont les spécifications techniques de l'examen dbt Cloud Architect ?

**Réponse :** L'examen dure 2 heures, comprend 65 questions, nécessite un score de 65% pour réussir, coûte 200$, est disponible uniquement en anglais, et la certification expire 2 ans après l'obtention. Il est surveillé en ligne et utilise les navigateurs Web Caveon.

Ces questions couvrent les 8 domaines principaux évalués dans l'examen de certification dbt Cloud Architect et vous aideront à vous préparer efficacement pour cette certification avancée.