Examen blanc

Terraform Associate 004 — 50 questions

QCM · 8 domaines officiels · teste Terraform **1.12**

Une seule bonne réponse, sauf mention **(choisissez DEUX)**

Notez vos lettres (1 → 50), puis ouvrez **QUIZ-TERRAFORM-004-CORRECTION.md**.

Domaine 1

Infrastructure as Code — Q1 à Q4

© 2026 — Formation Terraform / Azure · Corrigé : QUIZ-TERRAFORM-004-CORRECTION.md

Question 1 / 50

Quel est l'intérêt principal de l'Infrastructure as Code ?

A. Remplacer les administrateurs systèmes

B. Versionner, reproduire et réviser l'infrastructure

C. Supprimer le besoin d'un fournisseur cloud

D. Chiffrer automatiquement les serveurs

© 2026 — Formation Terraform / Azure · Corrigé : QUIZ-TERRAFORM-004-CORRECTION.md

Question 2 / 50

Terraform est avant tout…

A. un langage impératif de scripting

B. un outil de provisioning **déclaratif**

C. un système d'exploitation

D. un outil de gestion de configuration (configuration management)

Question 3 / 50

Quel est l'avantage du modèle déclaratif ?

A. On écrit la suite exacte des commandes

B. On décrit l'**état désiré** ; Terraform calcule comment l'atteindre

C. Il faut tout coder en bash

D. Cela supprime le besoin de state

© 2026 — Formation Terraform / Azure · Corrigé : QUIZ-TERRAFORM-004-CORRECTION.md

Question 4 / 50

Pourquoi Terraform est-il multi-cloud / agnostique ?

A. Il embarque toutes les API en dur

B. Il ne fonctionne qu'avec un seul cloud

C. Il s'appuie sur des **providers** (plugins)

D. Il génère du code propre à chaque cloud

© 2026 — Formation Terraform / Azure · Corrigé : QUIZ-TERRAFORM-004-CORRECTION.md

Domaine 2

Terraform Fundamentals — Q5 à Q10

Question 5 / 50

Quel langage sert à écrire la configuration Terraform ?

> **A.** YAML

> **B.** XML

> **C. HCL** (HashiCorp Configuration Language)

> **D.** Python

© 2026 — Formation Terraform / Azure · Corrigé : QUIZ-TERRAFORM-004-CORRECTION.md

Question 6 / 50

Que fait `terraform init` ?

> **A.** Applique les changements

> **B.** Initialise le répertoire et **télécharge les providers / le backend**

> **C.** Détruit l'infrastructure

> **D.** Formate le code

© 2026 — Formation Terraform / Azure · Corrigé : QUIZ-TERRAFORM-004-CORRECTION.md

Question 7 / 50

Quel bloc déclare la source et la version des providers requis ?

> **A.** `required_providers` (dans le bloc `terraform {}`)
>
> **B.** `provider {}`
>
> **C.** `backend {}`
>
> **D.** `variable {}`

Question 8 / 50

Que signifie la contrainte `version = "~> 1.2.0"` ?

> **A.** Exactement `1.2.0`

> **B.** `>= 1.2.0` et `< 1.3.0`

> **C.** `>= 1.2.0` et `< 2.0.0`

> **D.** N'importe quelle version `1.x`

© 2026 — Formation Terraform / Azure · Corrigé : QUIZ-TERRAFORM-004-CORRECTION.md

Question 9 / 50

À quoi sert le bloc `provider "azurerm" { ... }` ?

> **A.** Déclarer des variables d'entrée

> **B. Configurer** le provider (authentification, région, options)

> **C.** Définir le backend distant

> **D.** Créer une ressource

Question 10 / 50

Comment épingler la version de Terraform lui-même (le binaire) ?

> **A.** `required_version` dans le bloc `terraform {}`
>
> **B.** `version` dans le bloc `provider {}`
>
> **C.** Une variable `TF_VERSION`
>
> **D.** Ce n'est pas possible

© 2026 — Formation Terraform / Azure · Corrigé : QUIZ-TERRAFORM-004-CORRECTION.md

Domaine 3

Core Terraform Workflow — Q11 à Q18

© 2026 — Formation Terraform / Azure · Corrigé : QUIZ-TERRAFORM-004-CORRECTION.md

Question 11 / 50

Quel est l'ordre correct du workflow de base ?

> **A.** `apply` → `init` → `plan`

> **B.** `init` → `plan` → `apply`

> **C.** `plan` → `init` → `apply`

> **D.** `init` → `apply` → `plan`

© 2026 — Formation Terraform / Azure · Corrigé : QUIZ-TERRAFORM-004-CORRECTION.md

Question 12 / 50

Quelle commande prévisualise les changements sans les appliquer ?

> **A.** `terraform show`
>
> **B.** `terraform plan`
>
> **C.** `terraform preview`
>
> **D.** `terraform validate`

© 2026 — Formation Terraform / Azure · Corrigé : QUIZ-TERRAFORM-004-CORRECTION.md

Question 13 / 50

Que fait `terraform fmt` ?

> **A.** Vérifie la cohérence de la config

> **B. Reformate** le code selon les conventions standard

> **C.** Détruit les ressources

> **D.** Affiche le state

Question 14 / 50

Que fait `terraform validate` ?

> **A.** Applique la configuration

> **B.** Vérifie la **validité de la config** sans appeler les API cloud

> **C.** Compare les coûts

> **D.** Verrouille le state

Question 15 / 50

Quelle commande détruit l'infrastructure gérée ?

> **A.** `terraform delete`

> **B.** `terraform remove`

> **C.** `terraform destroy`

> **D.** `terraform apply -clean`

© 2026 — Formation Terraform / Azure · Corrigé : QUIZ-TERRAFORM-004-CORRECTION.md

Question 16 / 50

Quelle option applique sans confirmation interactive ?

> **A.** `-force`

> **B.** `-yes`

> **C.** `-auto-approve`

> **D.** `-no-prompt`

Question 17 / 50

Pourquoi `terraform plan -out=tf.plan` puis `terraform apply tf.plan` ?

> **A.** Pour appliquer **exactement** le plan revu (pas de recalcul)

> **B.** Pour télécharger les providers plus vite

> **C.** Parce que `apply` exige toujours un fichier

> **D.** Pour ignorer le verrou du state

© 2026 — Formation Terraform / Azure · Corrigé : QUIZ-TERRAFORM-004-CORRECTION.md

Question 18 / 50

Dans un plan, que signifie le symbole `-/+` devant une ressource ?

> **A.** Mise à jour sur place (in-place)

> **B.** Création simple

> **C. Remplacement** : destruction puis recréation

> **D.** Aucune action

Domaine 4

Terraform Configuration — Q19 à Q32

© 2026 — Formation Terraform / Azure · Corrigé : QUIZ-TERRAFORM-004-CORRECTION.md

Question 19 / 50

Comment déclare-t-on une variable d'entrée ?

> **A.** `input "x" {}`

> **B.** `variable "x" {}`

> **C.** `var "x" {}`

> **D.** `set x = ...`

Question 20 / 50

Comment référence-t-on la variable `region` ?

> **A.** `${region}`

> **B.** `var.region`

> **C.** `variable.region`

> **D.** `self.region`

Question 21 / 50

Quelle source de valeur de variable a la priorité la plus élevée ?

> **A.** Variable d'environnement `TF_VAR_*`
>
> **B.** Fichier `terraform.tfvars`
>
> **C.** Option `-var` / `-var-file` en ligne de commande
>
> **D.** Valeur `default` de la variable

© 2026 — Formation Terraform / Azure · Corrigé : QUIZ-TERRAFORM-004-CORRECTION.md

Question 22 / 50

À quoi sert un bloc `output` ?

> **A.** Définir une variable d'entrée

> **B. Exposer** une valeur (réutilisable, affichée après l'apply)

> **C.** Créer une ressource

> **D.** Configurer un provider

Question 23 / 50

Que fait `sensitive = true` sur une variable ou un output ?

> **A.** Chiffre la valeur dans le state
>
> **B. Masque** la valeur dans la sortie CLI / les logs de plan
>
> **C.** Empêche son utilisation
>
> **D.** La rend obligatoire

Question 24 / 50

Quelle est la différence entre une ressource et une data source ?

A. Aucune, ce sont des synonymes

B. La data source **lit** l'existant ; la ressource **crée / gère**

C. La data source crée, la ressource lit

D. La data source ne fonctionne qu'en local

© 2026 — Formation Terraform / Azure · Corrigé : QUIZ-TERRAFORM-004-CORRECTION.md

Question 25 / 50

Comment référence-t-on l'attribut d'une data source `azurerm_dns_zone` nommée `labs` ?

> **A.** `azurerm_dns_zone.labs.name`

> **B.** `data.azurerm_dns_zone.labs.name`

> **C.** `var.labs.name`

> **D.** `module.labs.name`

Question 26 / 50

À quoi servent les `locals` ?

> **A.** Stocker le state localement

> **B.** Définir des **valeurs locales** calculées et réutilisables

> **C.** Déclarer des providers

> **D.** Importer des ressources

© 2026 — Formation Terraform / Azure · Corrigé : QUIZ-TERRAFORM-004-CORRECTION.md

Question 27 / 50

À quoi sert le meta-argument `count` ?

> **A.** Limiter le parallélisme

> **B.** Créer **N instances** d'une même ressource

> **C.** Compter les providers

> **D.** Numéroter les outputs

© 2026 — Formation Terraform / Azure · Corrigé : QUIZ-TERRAFORM-004-CORRECTION.md

Question 28 / 50

Avantage de `for_each` par rapport à `count` ?

> **A.** Plus rapide à taper

> **B. Clés stables** (map/set) : supprimer un élément ne décale pas les autres

> **C.** Ne nécessite pas de provider

> **D.** Ignore les dépendances

© 2026 — Formation Terraform / Azure · Corrigé : QUIZ-TERRAFORM-004-CORRECTION.md

Question 29 / 50

À quoi sert `depends_on` ?

> **A.** Déclarer une dépendance **explicite** non déduite automatiquement

> **B.** Supprimer une dépendance

> **C.** Forcer le parallélisme

> **D.** Importer un module

Question 30 / 50

Que fait `lifecycle { prevent_destroy = true }` ?

> **A.** Ignore les changements de tags

> **B. Empêche** la destruction de la ressource (l'apply échoue)

> **C.** Crée un backup

> **D.** Accélère la destruction

Question 31 / 50

Que fait `create_before_destroy = true` ?

> **A.** Détruit puis recrée la ressource

> **B.** Crée le **remplaçant avant** de détruire l'ancien (évite la coupure)

> **C.** Empêche tout remplacement

> **D.** Duplique la ressource en permanence

© 2026 — Formation Terraform / Azure · Corrigé : QUIZ-TERRAFORM-004-CORRECTION.md

Question 32 / 50

Quelle affirmation est correcte au sujet des `provisioners` ?

> **A.** C'est la méthode recommandée par défaut

> **B.** Ils doivent être utilisés en **dernier recours**

> **C.** Ils remplacent les providers

> **D.** Ils gèrent le verrou du state

© 2026 — Formation Terraform / Azure · Corrigé : QUIZ-TERRAFORM-004-CORRECTION.md

Domaine 5

Terraform Modules — Q33 à Q37

© 2026 — Formation Terraform / Azure · Corrigé : QUIZ-TERRAFORM-004-CORRECTION.md

Question 33 / 50

Qu'est-ce qu'un module ?

A. Un fichier de state

B. Un **conteneur réutilisable** de ressources (entrées = variables, sorties = outputs)

C. Un provider tiers

D. Une commande CLI

© 2026 — Formation Terraform / Azure · Corrigé : QUIZ-TERRAFORM-004-CORRECTION.md

Question 34 / 50

Qu'appelle-t-on le module racine (root module) ?

> **A.** Le registry public

> **B.** Le **répertoire de travail** où l'on lance `terraform`

> **C.** Le premier module téléchargé

> **D.** Le provider par défaut

© 2026 — Formation Terraform / Azure · Corrigé : QUIZ-TERRAFORM-004-CORRECTION.md

Question 35 / 50

Comment passe-t-on une valeur à un module ?

> **A.** Via ses **variables d'entrée** (arguments du bloc `module`)

> **B.** Via le state

> **C.** Via une variable d'environnement obligatoire

> **D.** Via `terraform import`

Question 36 / 50

Comment lit-on la sortie `ip` d'un module nommé `vm` ?

> **A.** `output.vm.ip`

> **B.** `module.vm.ip`

> **C.** `var.vm.ip`

> **D.** `vm.output.ip`

Question 37 / 50

Dans `source = "Azure/naming/azurerm"`, à quoi correspond cette valeur ?

> **A.** Un chemin de fichier local

> **B.** Une référence de module du **Terraform Registry** (`namespace/nom/provider`)

> **C.** Une URL de state

> **D.** Un nom de provider

Domaine 6

State Management — Q38 à Q44

© 2026 — Formation Terraform / Azure · Corrigé : QUIZ-TERRAFORM-004-CORRECTION.md

Question 38 / 50

À quoi sert le fichier d'état (`terraform.tfstate`) ?

> **A.** Stocker les identifiants cloud

> **B.** Faire la correspondance entre la config et les **ressources réelles**

> **C.** Contenir le code des providers

> **D.** Journaliser les commandes

Question 39 / 50

Par défaut, où le state est-il stocké ?

> **A.** Dans une base distante

> **B. Localement**, dans `terraform.tfstate`

> **C.** Dans le registry

> **D.** En mémoire vive uniquement

© 2026 — Formation Terraform / Azure · Corrigé : QUIZ-TERRAFORM-004-CORRECTION.md

Question 40 / 50

Citez deux avantages d'un backend distant. choisissez DEUX

> **A.** Partage du state au sein de l'équipe

> **B.** Verrouillage (lock) du state

> **C.** Suppression du besoin de providers

> **D.** Accélération automatique des `apply`

Question 41 / 50

Sur Azure Storage, comment le verrou du state est-il assuré ?

> **A.** Par une table DynamoDB

> **B.** Par un **lease (bail)** sur le blob du state

> **C.** Par un fichier `.lock` manuel

> **D.** Il n'y a pas de verrou

© 2026 — Formation Terraform / Azure · Corrigé : QUIZ-TERRAFORM-004-CORRECTION.md

Question 42 / 50

Quelle commande liste les ressources gérées dans le state ?

> **A.** `terraform show all`
>
> **B.** `terraform state list`
>
> **C.** `terraform list`
>
> **D.** `terraform output`

© 2026 — Formation Terraform / Azure · Corrigé : QUIZ-TERRAFORM-004-CORRECTION.md

Question 43 / 50

Vous renommez une ressource dans le code. Quelle commande évite un détruire/recréer ?

> **A.** `terraform state rm`

> **B.** `terraform taint`

> **C.** `terraform state mv`

> **D.** `terraform refresh`

Question 44 / 50

Comment Terraform détecte-t-il une dérive (drift, modif faite hors Terraform) ?

> **A.** Impossible à détecter

> **B.** En rafraîchissant le state lors d'un `plan` (le diff montre l'écart)

> **C.** Uniquement via `terraform fmt`

> **D.** En relançant `init`

© 2026 — Formation Terraform / Azure · Corrigé : QUIZ-TERRAFORM-004-CORRECTION.md

Domaine 7

Maintain Infrastructure — Q45 à Q48

© 2026 — Formation Terraform / Azure · Corrigé : QUIZ-TERRAFORM-004-CORRECTION.md

Question 45 / 50

Quelle commande intègre dans le state une ressource créée à la main ?

A. `terraform import`

B. `terraform add`

C. `terraform state push`

D. `terraform apply -refresh`

© 2026 — Formation Terraform / Azure · Corrigé : QUIZ-TERRAFORM-004-CORRECTION.md

Question 46 / 50

Quelle commande moderne force la recréation d'une ressource (ex-`taint`) ?

> **A.** `terraform taint <addr>`

> **B.** `terraform apply -replace=<addr>`

> **C.** `terraform recreate <addr>`

> **D.** `terraform refresh <addr>`

© 2026 — Formation Terraform / Azure · Corrigé : QUIZ-TERRAFORM-004-CORRECTION.md

Question 47 / 50

Que fait `terraform state rm <adresse>` ?

> **A.** Détruit la ressource dans le cloud

> **B. Retire** la ressource du state, sans la détruire réellement

> **C.** Renomme la ressource

> **D.** Verrouille le state

Question 48 / 50

À quoi servent les workspaces Terraform (CLI) ?

A. Gérer **plusieurs états** avec la même configuration

B. Éditer le code à plusieurs en temps réel

C. Remplacer les modules

D. Stocker les secrets

Domaine 8

HCP Terraform — Q49 à Q50

© 2026 — Formation Terraform / Azure · Corrigé : QUIZ-TERRAFORM-004-CORRECTION.md

Question 49 / 50

Que propose HCP Terraform (ex-Terraform Cloud) ?

A. Uniquement un éditeur de texte

B. Exécution distante, **state managé**, verrouillage, RBAC, policies

C. Le remplacement du langage HCL

D. Un hyperviseur de VMs

© 2026 — Formation Terraform / Azure · Corrigé : QUIZ-TERRAFORM-004-CORRECTION.md

Question 50 / 50

À quoi servent les variable sets dans HCP Terraform ?

A. Chiffrer le state

B. Partager un jeu de variables entre plusieurs workspaces / projets

C. Définir le backend local

D. Remplacer les modules

© 2026 — Formation Terraform / Azure · Corrigé : QUIZ-TERRAFORM-004-CORRECTION.md

Fin de l'examen blanc

Vérifiez vos réponses

➡️ Ouvrez **QUIZ-TERRAFORM-004-CORRECTION.md** pour le corrigé + explications.

Barème indicatif : ≥ 70 % = niveau visé pour le passage de l'examen 004.