# Terraform-Associate-004인증덤프공부자료 & Terraform-Associate-004학습자료


KoreaDumps Terraform-Associate-004 최신 PDF 버전 시험 문제집을 무료로 Google Drive에서 다운로드하세요: https://drive.google.com/open?id=1dUmQkczclKG_FbPTwA7A5xkVZ0ta5eAz

KoreaDumps는 몇년간 최고급 덤프품질로 IT인증덤프제공사이트중에서 손꼽히는 자리에 오게 되었습니다. HashiCorp Terraform-Associate-004 덤프는 많은 덤프들중에서 구매하는 분이 많은 인기덤프입니다. HashiCorp Terraform-Associate-004시험준비중이신 분이시라면HashiCorp Terraform-Associate-004한번 믿고 시험에 도전해보세요. 좋은 성적으로 시험패스하여 자격증 취득할것입니다.

KoreaDumps는HashiCorp Terraform-Associate-004인증시험을 아주 쉽게 패스할 수 있도록 도와드리는 사이트입니다. KoreaDumps의 엘리트한 전문가가 끈임 없는 노력으로 최고의HashiCorp Terraform-Associate-004자료를 만들었습니다. 이 자료로 여러분은 100%HashiCorp의Terraform-Associate-004인증시험을 패스할 수 있으며, KoreaDumps을 선택함으로 성공을 선택한 것입니다. KoreaDumps가 제공하는 시험가이드로 효과적인 학습으로 많은 분들이 모두 인증시험을 패스하였습니다. 이건 모두 KoreaDumps 인증시험덤프로 공부하였기 때문입니다. 그 중HashiCorp Terraform-Associate-004인증시험을 패스한 분들도 모두 KoreaDumps인증시험덤프를 사용하였기 때문입니다.

**>> Terraform-Associate-004인증덤프공부자료 <<**

## Terraform-Associate-004 덤프 HashiCorp 자격증

KoreaDumps에서 출시한 HashiCorp 인증 Terraform-Associate-004시험덤프는KoreaDumps의 엘리트한 IT전문가들이 IT인증실제시험문제를 연구하여 제작한 최신버전 덤프입니다. 덤프는 실제시험의 모든 범위를 커버하고 있어 시험통과율이 거의 100%에 달합니다. 제일 빠른 시간내에 덤프에 있는 문제만 잘 이해하고 기억하신다면 시험패스는 문제없습니다.

## HashiCorp Terraform-Associate-004 시험요강:

| 주제 | 소개 |
|---|---|
| 주제 1 | • Core Terraform workflow: This domain focuses on the essential workflow steps: initializing directories, validating configurations, generating execution plans, applying changes, destroying infrastructure, and formatting code. |
| 주제 2 | • Terraform state management: This domain focuses on managing Terraform's state file, understanding local and remote backends, implementing state locking, and handling resource drift. |
| 주제 3 | • Terraform fundamentals: This domain addresses installing and managing provider plugins, understanding Terraform's provider architecture, and how Terraform tracks infrastructure state. |

| | |
|---|---|
| 주제 4 | • HCP Terraform: This domain covers using HashiCorp Cloud Platform Terraform for infrastructure provisioning, collaboration and governance features, organizing workspaces and projects, and configuring integrations. |
| 주제 5 | • Infrastructure as Code (IaC) with Terraform: This domain covers the foundational concept of Infrastructure as Code and how Terraform enables managing resources across multiple cloud providers and services through a unified workflow. |
| 주제 6 | • Maintain infrastructure with Terraform: This domain addresses importing existing infrastructure into Terraform, inspecting state using CLI commands, and using verbose logging for troubleshooting. |

## 최신 Terraform Associate Terraform-Associate-004 무료샘플문제 (Q151-Q156):

**질문 # 151**
You modified your Terraform configuration to fix a typo in the resource ID by renaming it from photoes to photos. What configuration will you add to update the resource ID in state without destroying the existing resource?
Original configuration:
resource "aws_s3_bucket" "photoes" {
bucket_prefix = "images"
}
Updated configuration:
resource "aws_s3_bucket" "photos" {
bucket_prefix = "images"
}

- A. moved {
  from = aws_s3_bucket.photoes
  to = aws_s3_bucket.photos
  }
- B. None. Terraform will automatically update the resource ID.
- C. moved {
  bucket.photoes = aws_s3_bucket.photos
  }
- D. moved {
  aws_s3_bucket.photoes = aws_s3_bucket.photos
  }

**정답：A**

**설명:**
Rationale for Correct Answer (A):
Terraform does not automatically update state references when resource identifiers are renamed. Instead, you must use a moved block in your configuration to inform Terraform how to map the old resource to the new one. This prevents Terraform from destroying and recreating the resource.
Analysis of Incorrect Options:
B & C: Incorrect syntax - moved requires from and to.
D: Incorrect, Terraform won't auto-detect renames and will plan to destroy and recreate the resource unless a moved block is provided.
Key Concept:
The moved block is essential for refactoring resource names without losing resources in state.
Reference:
Terraform Exam Objective - Implement and Maintain State.

**질문 # 152**
You can define multiple backend blocks in your Terraform configuration to store your state in multiple locations.

- A. False
- B. True

**정답：A**

**설명：**
Rationale for Correct Answer: A Terraform configuration supports only one backend at a time. The backend determines where state is stored and how locking works. Terraform does not support writing the same state to multiple backends simultaneously via multiple backend blocks.
Analysis of Incorrect Options (Distractors):
A (True): Incorrect-Terraform allows only a single backend configuration for a given working directory/root module.
Key Concept: Single-backend design: one state location per configuration/workspace.
Reference: Terraform Objectives - Navigate Terraform State and Backends (backend configuration and limitations).

**질문 # 153**
You created infrastructure outside the Terraform workflow that you now want to manage using Terraform.
Which command brings the infrastructure into Terraform state?

- A. terraform get
- B. terraform init
- C. terraform refresh
- D. terraform import

**정답：D**

**설명：**
The terraform import command allows Terraform to takeexistinginfrastructure andbring it under Terraform's management.
A (terraform get)is incorrect because it is used to fetch modules.
B (terraform refresh)is incorrect because it only updates Terraform's state to match the infrastructure but does not import resources.
D (terraform init)is incorrect because it only initializes the Terraform working directory.
Official Terraform Documentation Reference:
terraform import - HashiCorp Documentation

**질문 # 154**
Your team is collaborating on infrastructure using Terraform and wants to format code to follow Terraform language style conventions. How can you update your code to meet these requirements?

- A. Run terraform validate prior to executing terraform plan or terraform apply.
- B. Run terraform fmt to update your Terraform configurations.
- C. Replace all tabs with spaces within your Terraform configuration files.
- D. Terraform automatically formats configuration on terraform apply.

**정답：B**

**설명：**
Rationale for Correct answer: terraform fmt automatically rewrites .tf files to match Terraform's canonical formatting (indentation, spacing, alignment conventions). This is the standard tool for enforcing consistent style across a team.
Analysis of Incorrect Options (Distractors):
B: Insufficient-formatting is more than tabs vs spaces; terraform fmt handles the full HCL style.
C: terraform validate checks syntax and internal consistency, not formatting.
D: Terraform does not auto-format configuration during apply.
Key Concept: Standardizing HCL formatting with terraform fmt.
Reference: Terraform Objectives - Read, Generate, and Modify Configurations (formatting and configuration authoring), Understand Terraform Basics and CLI (core CLI commands).

**질문 # 155**
What does Terraform not reference when running a terraform apply -refresh-only ?

- A. Cloud provider
- B. Credentials
- C. Terraform resource definitions in configuration files
- D. State file

**정답：C**

**설명：**
When running a terraform apply -refresh-only, Terraform does not reference the configuration files, but only the state file, credentials, and cloud provider. The purpose of this command is to update the state file with the current status of the real resources, without making any changes to theml.

**질문 # 156**
.....

참고: KoreaDumps에서 Google Drive로 공유하는 무료, 최신 Terraform-Associate-004 시험 문제집이 있습니다: https://drive.google.com/open?id=1dUmQkczclKG_FbPTwA7A5xkVZ0ta5eAz