HashiCorp Terraform Infrastructure as Code (IaC) Practice Test (Sample)

Study Guide

BY EXAMZIFY

Everything you need from our exam experts!

Table of Contents

Copyright 1
Table of Contents 2
Introduction 3
How to Use This Guide 4
Questions 5
Answers 8
Explanations 10
Next Steps 16

Introduction

Preparing for a certification exam can feel overwhelming, but with the right tools, it becomes an opportunity to build confidence, sharpen your skills, and move one step closer to your goals. At Examzify, we believe that effective exam preparation isn't just about memorization, it's about understanding the material, identifying knowledge gaps, and building the test-taking strategies that lead to success.

This guide was designed to help you do exactly that.

Whether you're preparing for a licensing exam, professional certification, or entry-level qualification, this book offers structured practice to reinforce key concepts. You'll find a wide range of multiple-choice questions, each followed by clear explanations to help you understand not just the right answer, but why it's correct.

The content in this guide is based on real-world exam objectives and aligned with the types of questions and topics commonly found on official tests. It's ideal for learners who want to:

- Practice answering questions under realistic conditions,
- Improve accuracy and speed,
- Review explanations to strengthen weak areas, and
- Approach the exam with greater confidence.

We recommend using this book not as a stand-alone study tool, but alongside other resources like flashcards, textbooks, or hands-on training. For best results, we recommend working through each question, reflecting on the explanation provided, and revisiting the topics that challenge you most.

Remember: successful test preparation isn't about getting every question right the first time, it's about learning from your mistakes and improving over time. Stay focused, trust the process, and know that every page you turn brings you closer to success.

Let's begin.

How to Use This Guide

This guide is designed to help you study more effectively and approach your exam with confidence. Whether you're reviewing for the first time or doing a final refresh, here's how to get the most out of your Examzify study guide:

1. Start with a Diagnostic Review

Skim through the questions to get a sense of what you know and what you need to focus on. Your goal is to identify knowledge gaps early.

2. Study in Short, Focused Sessions

Break your study time into manageable blocks (e.g. 30 – 45 minutes). Review a handful of questions, reflect on the explanations.

3. Learn from the Explanations

After answering a question, always read the explanation, even if you got it right. It reinforces key points, corrects misunderstandings, and teaches subtle distinctions between similar answers.

4. Track Your Progress

Use bookmarks or notes (if reading digitally) to mark difficult questions. Revisit these regularly and track improvements over time.

5. Simulate the Real Exam

Once you're comfortable, try taking a full set of questions without pausing. Set a timer and simulate test-day conditions to build confidence and time management skills.

6. Repeat and Review

Don't just study once, repetition builds retention. Re-attempt questions after a few days and revisit explanations to reinforce learning. Pair this guide with other Examzify tools like flashcards, and digital practice tests to strengthen your preparation across formats.

There's no single right way to study, but consistent, thoughtful effort always wins. Use this guide flexibly, adapt the tips above to fit your pace and learning style. You've got this!

Questions

1. **How do you target a specific resource when applying changes?**
   - **A. By specifying the resource ID**
   - **B. By using the -target option with the resource address**
   - **C. By editing the configuration file directly**
   - **D. By labeling the resource in the code**

2. **What command is used to enable a new workspace in Terraform?**
   - **A. terraform new workspace [workspace_name]**
   - **B. terraform workspace create [workspace_name]**
   - **C. terraform workspace switch [workspace_name]**
   - **D. terraform workspace new [workspace_name]**

3. **What command do you use to initialize a Terraform configuration?**
   - **A. terraform apply**
   - **B. terraform validate**
   - **C. terraform init**
   - **D. terraform refresh**

4. **What command would you run to switch to a different workspace in Terraform?**
   - **A. terraform change workspace <workspace_name>**
   - **B. terraform switch workspace <workspace_name>**
   - **C. terraform workspace select <workspace_name>**
   - **D. terraform set workspace <workspace_name>**

5. **What is the purpose of the 'terraform apply' command?**
   - **A. To validate the configuration files**
   - **B. To apply changes to the infrastructure**
   - **C. To change the execution plan**
   - **D. To clean up resources**

6. What is the main purpose of the 'terraform graph' command?

   A. To validate resource configurations
   B. To generate a visual representation of the dependency graph for resources
   C. To display the state of the current infrastructure
   D. To execute scripts on a remote machine

7. What is the primary language used to write Terraform configurations?

   A. Python
   B. JavaScript
   C. HashiCorp Configuration Language (HCL)
   D. JSON

8. What is the purpose of the 'terraform fmt' command?

   A. To validate configuration syntax
   B. To format Terraform configuration files to have a canonical format
   C. To generate documentation for Terraform files
   D. To optimize resource deployment times

9. What is the purpose of the 'terraform validate' command?

   A. To check the validity of Terraform configuration files without executing them
   B. To run all configurations and verify their results
   C. To deploy resources and confirm their setup
   D. To compare the current state with the desired state

10. Which file extension is commonly used for Terraform configuration files?

   A. .terraf
   B. .tf
   C. .tfjson
   D. .tfstate

Answers

1. **B**
2. **D**
3. **C**
4. **C**
5. **B**
6. **B**
7. **C**
8. **B**
9. **A**
10. **B**

Explanations

1. **How do you target a specific resource when applying changes?**

   A. By specifying the resource ID

   B. By using the -target option with the resource address

   C. By editing the configuration file directly

   D. By labeling the resource in the code

To target a specific resource when applying changes in Terraform, using the -target option with the resource address is the correct approach. This functionality allows operators to specify a particular resource or module for Terraform to apply changes, rather than applying changes to all resources defined in the configuration.   When you use the -target option in the command line, you can define the exact resource in your Terraform state that you want to affect. The resource address typically comes in the format of `resource_type.resource_name`, which uniquely identifies the resource within the Terraform configuration. This focused approach helps manage and apply changes more granularly, especially useful in complex infrastructures where you might not want to impact other resources during updates. It's important to note that while other approaches might seem logical, they do not align with how Terraform is designed to manage state and resources. Specifying the resource ID directly does not work since Terraform requires the full resource address. Editing the configuration file directly alters the desired state configuration but does not apply targeted changes. Similarly, labeling resources in the code may help with organization but does not facilitate the targeting mechanism when executing commands.

2. **What command is used to enable a new workspace in Terraform?**

   A. terraform new workspace [workspace_name]

   B. terraform workspace create [workspace_name]

   C. terraform workspace switch [workspace_name]

   D. terraform workspace new [workspace_name]

The command used to enable a new workspace in Terraform is "terraform workspace create [workspace_name]." This command is specifically designed to create a new workspace with the name you specify.   In Terraform, workspaces are a way to manage different states of your infrastructure configurations. Each workspace can have its own state file, allowing you to deploy multiple environments (like development, staging, and production) without conflict. Utilizing the command "terraform workspace create [workspace_name]" ensures that a new isolated state is initiated for that workspace. When utilizing Terraform, understanding the workspace commands is crucial for managing environment-specific configurations effectively. This command specifically serves to create a workspace; thus, the functionality aligns directly with the purpose of initializing a new workspace in your projects.

3. What command do you use to initialize a Terraform configuration?

A. terraform apply

B. terraform validate

C. terraform init

D. terraform refresh

When working with Terraform, the command used to initialize a Terraform configuration is "terraform init." This command sets up the working environment by creating the necessary backend configuration files, downloading the required provider plugins, and preparing the directory for the deployment of infrastructure. It is essentially the first command you run after writing your configuration files because it lays the groundwork for any further operations, such as applying or validating the configuration. During the initialization process, Terraform also checks for any changes in the configuration and ensures that the working directory is in the correct state to proceed. This makes it an essential step in the Terraform workflow, ensuring that all dependencies are managed before any infrastructure changes are made. While other commands have important roles in the Terraform lifecycle—like "apply" which applies the changes defined in the configuration, "validate" which checks whether the configuration is syntactically valid, and "refresh" which updates the state file with the real infrastructure state—none of these commands perform the initialization task. Therefore, "terraform init" is the correct and necessary choice to start with before engaging in further Terraform operations.

4. What command would you run to switch to a different workspace in Terraform?

A. terraform change workspace <workspace_name>

B. terraform switch workspace <workspace_name>

C. terraform workspace select <workspace_name>

D. terraform set workspace <workspace_name>

To switch to a different workspace in Terraform, the correct command is "terraform workspace select <workspace_name>". This command is designed specifically for managing workspaces within a Terraform configuration. Using this command, users can easily specify the name of the workspace they wish to activate, facilitating a seamless transition between different environments or configurations. Terraform workspaces allow users to manage multiple state files within a single Terraform configuration, providing a way to isolate different environments like development, testing, and production. The command "terraform workspace select" directly addresses this need by enabling the selection of the desired workspace, ensuring that any subsequent Terraform operations are executed in the context of that workspace. Other options, while they may sound plausible, do not correspond to valid Terraform commands for changing workspaces. Therefore, the use of "terraform workspace select" is essential for effectively navigating between workspaces within Terraform projects.

5. What is the purpose of the 'terraform apply' command?

A. To validate the configuration files

B. To apply changes to the infrastructure

C. To change the execution plan

D. To clean up resources

The 'terraform apply' command is essential for managing infrastructure because its primary purpose is to apply changes to the infrastructure as defined in the Terraform configuration files. When this command is executed, Terraform reads the current state of the infrastructure and compares it against the desired state described in the configuration files. It generates an execution plan that specifies the changes that will be made. Once this plan is reviewed and confirmed by the user, 'terraform apply' proceeds to create, update, or delete resources according to the specifications in the configuration. This process ensures that the actual state of the infrastructure is modified to align with the desired configuration, thus enabling effective Infrastructure as Code practices. Choosing this option reflects a clear understanding of the core functionality of Terraform in provisioning and managing infrastructure resources through code-based configurations.

6. What is the main purpose of the 'terraform graph' command?

A. To validate resource configurations

B. To generate a visual representation of the dependency graph for resources

C. To display the state of the current infrastructure

D. To execute scripts on a remote machine

The primary purpose of the 'terraform graph' command is to generate a visual representation of the dependency graph for the resources defined in your Terraform configuration. This command analyzes the resources and their relationships, allowing users to visualize how different components depend on one another, which can be particularly useful for understanding complex infrastructure setups. Using 'terraform graph', users can output the dependency graph in a format that can be interpreted by graph visualization tools. This can help in troubleshooting, optimizing resource arrangements, and ensuring that the infrastructure behaves as expected during provisioning. By gaining insight into these relationships, developers and infrastructure managers can make informed decisions about changes, upgrades, and potential impacts to their infrastructure. Other choices, while related to Terraform's capabilities, do not accurately describe the primary function of the 'terraform graph' command. For example, validating resource configurations is primarily addressed by the 'terraform validate' command, and displaying the state of the current infrastructure is done through the 'terraform show' command. Executing scripts on a remote machine does not align with Terraform's functionality, which focuses on provisioning and managing infrastructure rather than executing remote commands directly. Thus, the emphasis on visualizing resource dependencies makes the choice about generating a dependency graph the correct answer.

7. What is the primary language used to write Terraform configurations?

A. Python

B. JavaScript

C. HashiCorp Configuration Language (HCL)

D. JSON

The primary language used to write Terraform configurations is HashiCorp Configuration Language (HCL). HCL is designed specifically for configuring infrastructure as code and is optimized for readability and usability, making it easier for users to define resources and their properties. HCL provides a concise syntax that allows users to express complex infrastructure requirements in a clear and understandable manner. For example, it includes features like simple block structures and variable interpolation, which are tailored to the needs of defining infrastructure. Additionally, HCL is inherently compatible with Terraform's workflow, providing seamless integration into the tooling and enabling users to leverage its full capabilities. While JSON can also be used for Terraform configurations, it is less common due to its verbosity and complexity compared to HCL. JSON is often more difficult for users to read and maintain, especially as configurations become larger and more complex. Thus, while it is technically possible to use JSON, HCL is preferred for its clarity and ease of use.

8. What is the purpose of the 'terraform fmt' command?

A. To validate configuration syntax

B. To format Terraform configuration files to have a canonical format

C. To generate documentation for Terraform files

D. To optimize resource deployment times

The 'terraform fmt' command is designed specifically to format Terraform configuration files to ensure they adhere to a canonical style. This command normalizes and beautifies the code, making it more readable and consistent across a codebase. Using 'terraform fmt' is important for maintaining best practices in coding style, which can enhance collaboration among teams and improve overall maintainability of the Infrastructure as Code (IaC) files. While other commands serve different functions, such as validating syntax or generating documentation, 'terraform fmt' focuses solely on formatting, ensuring that the code follows the conventions set by Terraform. This helps eliminate discrepancies that could arise when different developers have varying styles, allowing for a cohesive code standard across projects.

9. What is the purpose of the 'terraform validate' command?

A. To check the validity of Terraform configuration files without executing them

B. To run all configurations and verify their results

C. To deploy resources and confirm their setup

D. To compare the current state with the desired state

The purpose of the 'terraform validate' command is to check the validity of Terraform configuration files without executing them. This command does a syntax check and evaluates whether the configuration files are well-formed and logical in structure. It helps to identify any errors in the HCL (HashiCorp Configuration Language) before attempting to apply the configurations to the infrastructure, allowing users to catch mistakes early in the development process. This command ensures that the configuration can be parsed correctly by Terraform, helping to prevent runtime errors during resource creation or modification. It does not interact with the actual infrastructure, nor does it attempt to deploy or validate any state; rather, it focuses purely on the correctness of the syntax and structure of the files written. By running this command, users can gain confidence that their Terraform scripts are free of basic errors before proceeding to apply configurations that change the state of their infrastructure.

10. Which file extension is commonly used for Terraform configuration files?

A. .terraf

B. .tf

C. .tfjson

D. .tfstate

The commonly used file extension for Terraform configuration files is .tf. This extension indicates that the file contains the declarative configuration code which defines the desired infrastructure state and resources that Terraform will manage.   Terraform uses this format for its primary configuration files, which are written in HashiCorp Configuration Language (HCL), enabling the user to describe the infrastructure components in a human-readable format. The .tf extension is essential for Terraform to recognize and interpret the configurations when commands are executed. Other file extensions serve different purposes within the Terraform ecosystem. For instance, .tfjson is used for JSON-formatted versions of the configuration files, which are less commonly manipulated directly by users. The .tfstate extension is related to the Terraform state files, which track the current state of the infrastructure and are critical for resource management by Terraform. The .terraf extension is not a recognized or standard file type within Terraform practices.

Next Steps

Congratulations on reaching the final section of this guide. You've taken a meaningful step toward passing your certification exam and advancing your career.

As you continue preparing, remember that consistent practice, review, and self-reflection are key to success. Make time to revisit difficult topics, simulate exam conditions, and track your progress along the way.

If you need help, have suggestions, or want to share feedback, we’d love to hear from you. Reach out to our team at hello@examzify.com.

Or visit your dedicated course page for more study tools and resources:

https://hashicorpterraformiac.examzify.com

We wish you the very best on your exam journey. You've got this!