HASHICORP TERRAFORM INFRASTRUCTURE AS CODE (IAC) PRACTICE TEST (SAMPLE)

STUDY GUIDE

SAMPLE STUDY GUIDE WITH LIMITED QUESTIONS

VISIT US ONLINE FOR THE FULL VERSION STUDY GUIDE

https://hashicorpterraformiac.examzify.com

Table of Contents

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. **Which command applies changes to the infrastructure as specified in the Terraform configuration files?**

   A. terraform apply

   B. terraform push

   C. terraform create

   D. terraform deploy

2. **Which file extensions are associated with Terraform configuration files?**

   A. .json and .yaml

   B. .tf and .tfvars

   C. .xml and .config

   D. .txt and .log

3. **Which command allows you to see anytime your infrastructure will change?**

   A. terraform status

   B. terraform change

   C. terraform plan

   D. terraform up

4. **Can a Terraform local value reference other local values?**

   A. True

   B. False

5. **What does the Terraform command 'terraform apply -refresh-only' do?**

   A. Applies the necessary changes without refreshing state

   B. Shows planned changes without applying them

   C. Applies necessary changes after refreshing the state

   D. Only refreshes the state without applying any changes

6. **Which command outputs the current state of the managed infrastructure?**

   A. terraform show

   B. terraform display

   C. terraform status

   D. terraform info

7. Which of the following is NOT a valid string function in Terraform?

A. split

B. chomp

C. slice

D. join

8. What file is typically used to specify multiple environments in Terraform?

A. terraform.netvars

B. terraform.tfvars

C. terraform.env

D. terraform.config

9. What is the primary purpose of 'terraform import' command?

A. To apply a state configuration

B. To remove a resource from the state

C. To import existing resources into Terraform management

D. To create new resources

10. Which of the following is a valid terraform import command?

A. terraform import aws_instance.foo module=foo

B. terraform import aws_instance.foo address _id=i-abcd1234

C. terraform import 'aws_instance.baz[0]' i-abcd1234

D. terraform import resource.aws_instance.foo i-abcd1234

Answers

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

Explanations

1. Which command applies changes to the infrastructure as specified in the Terraform configuration files?

A. terraform apply

B. terraform push

C. terraform create

D. terraform deploy

The command that applies changes to the infrastructure as specified in the Terraform configuration files is "terraform apply." This command reads the configuration files to determine the desired state of the infrastructure and then interacts with the provider APIs to make the necessary changes to achieve that state. During this process, Terraform will carry out tasks such as creating, updating, or deleting resources to align the real-world infrastructure with the specifications outlined in the Terraform files. "Terraform apply" also provides a crucial safety check, as it typically shows a preview of the changes that will be made before actually applying them, allowing users to review and confirm these changes. This step is fundamental in Infrastructure as Code practices, where understanding the implications of changes is essential for maintaining stable and reliable infrastructure. The other options do not represent valid Terraform commands. For example, "terraform push" is not a valid command in Terraform's command set, as Terraform synchronizes the state through its state file rather than a push mechanism. "Terraform create" does not exist, as Terraform manages resources differently by defining their configuration and applying changes rather than using a command that implies a direct creation action. Similarly, "terraform deploy" is not recognized as an appropriate Terraform command, and while it may seem like a logical choice, it

2. Which file extensions are associated with Terraform configuration files?

A. .json and .yaml

B. .tf and .tfvars

C. .xml and .config

D. .txt and .log

Terraform configuration files primarily use the extensions .tf and .tfvars. The .tf extension is used for the main configuration files where the infrastructure resources are defined, including providers, resources, variables, and outputs that Terraform will use to provision and manage the infrastructure. The .tfvars extension is specifically designated for variable definition files, where you can set values for the variables declared within your .tf configuration files. This allows for better organization and separation of configuration settings, particularly in different environments or for different deployments. Using these specific extensions helps Terraform recognize the files appropriately, ensuring the configuration is processed correctly during execution. Other formats such as .json or .yaml might be used for different purposes in other tools or environments, but they are not part of the standard file extensions recognized by Terraform for configuration.

3. Which command allows you to see anytime your infrastructure will change?

A. terraform status

B. terraform change

C. terraform plan

D. terraform up

The command that allows you to see anytime your infrastructure will change is 'terraform plan'. This command is essential in the Terraform workflow as it generates an execution plan, showing what actions Terraform will take to change the current infrastructure to match the desired configuration specified in the Terraform files. When you run 'terraform plan', Terraform compares the current state of your infrastructure against the configurations defined in your .tf files. It will output the changes it intends to make, including additions, modifications, and deletions, without actually applying them. This allows you to review the changes beforehand, ensuring that you understand the impact of the adjustments you are about to make. This step is crucial for preventing unintended disruptions or resource modifications when managing infrastructure as code. Other commands serve different purposes: 'terraform status' is not a valid command in Terraform, while 'terraform change' does not exist either. 'terraform up' is not used in Terraform; typically, 'terraform apply' is the command that applies the changes described in your configuration. Thus, the ability to preview changes before applying them makes 'terraform plan' the clear choice for understanding how your infrastructure will change.

4. Can a Terraform local value reference other local values?

A. True

B. False

Local values in Terraform allow you to define named values that can be reused throughout your configuration. A key feature of local values is their ability to reference other local values, enabling you to create more modular and manageable Terraform configurations. When you define a local value, you can use it later in the same configuration file, and it's not limited to simple values; it can indeed reference another local value that you've defined earlier in the same block. This promotes cleaner code and reduces redundancy, as you can construct complex expressions or structures using these references. For example, if you define one local value that calculates a base URL, another local value can reference the first to construct an endpoint by appending a path. This interconnectedness of local values allows for a more organized setup and enhances the readability of your Terraform code.

5. What does the Terraform command 'terraform apply -refresh-only' do?

A. Applies the necessary changes without refreshing state

B. Shows planned changes without applying them

C. Applies necessary changes after refreshing the state

D. Only refreshes the state without applying any changes

The command 'terraform apply -refresh-only' specifically performs the action of refreshing the Terraform state file with the current infrastructure state without applying any changes to the resources defined in the configuration. This command is useful for ensuring that the Terraform state accurately reflects the actual state of the resources in the cloud or wherever the infrastructure is deployed. When you run this command, Terraform reads the existing resources from the provider (like AWS, Azure, etc.) and updates the state file accordingly. However, because the flag indicates a refresh-only operation, it deliberately avoids making any modifications to the actual infrastructure. This means that even if the current state of the resources differs from what is defined in the Terraform configuration files, no changes will be applied. This function is critical in scenarios where you want to synchronize the Terraform state with the real-world state of resources without triggering any updates or modifications, which can be particularly beneficial for auditing purposes or troubleshooting discrepancies between the planned and actual states.

6. Which command outputs the current state of the managed infrastructure?

A. terraform show

B. terraform display

C. terraform status

D. terraform info

The command that outputs the current state of the managed infrastructure is "terraform show." This command provides a formatted display of the state file, allowing users to see the existing resources and their configurations, as they are currently being managed by Terraform. It is particularly useful for checking the current state of the infrastructure against what is defined in the configuration files. "Terraform show" parses the state file or a specified plan file, giving a comprehensive overview of all the resources, including their attributes and relationships. This functionality is crucial for users seeking to verify what resources exist and how they are configured within their environment. The other options do not pertain to outputting the current state of managed infrastructure. For instance, "terraform display" and "terraform status" are not valid Terraform commands, and "terraform info" could lead to confusion, as it is also not a recognized command in Terraform's command set. Understanding the utility of "terraform show" is fundamental for users managing infrastructure with Terraform, as it aids in validation and debugging processes.

7. Which of the following is NOT a valid string function in Terraform?

A. split

B. chomp

C. slice

D. join

In Terraform, string functions are specifically designed to manipulate and transform string values in various ways. The chosen answer indicates that "slice" is not a valid string function, which aligns with the key understanding of Terraform's function set. The "split" function allows you to divide a string using a specified delimiter and returns a list of substrings. The "chomp" function is used to remove trailing newline characters from a string. The "join" function concatenates a list of strings into a single string, using a specified separator. On the other hand, "slice" is primarily a collection function used to extract a portion of a list or a map in Terraform, rather than dealing specifically with string manipulation. Therefore, while "split," "chomp," and "join" are valid string functions, "slice" does not fall within this category and therefore is correctly identified as not being a valid string function in Terraform.

8. What file is typically used to specify multiple environments in Terraform?

A. terraform.netvars

B. terraform.tfvars

C. terraform.env

D. terraform.config

The file that is commonly used to specify multiple environments in Terraform is `terraform.tfvars`. This file allows you to define variable values that can be passed to your Terraform configuration. When managing multiple environments, it is effective to use different `tfvars` files for each environment, such as `dev.tfvars`, `prod.tfvars`, etc. This approach helps streamline the process of deploying infrastructure by ensuring that each environment can have its own specific configurations, settings, and variable values. Using `terraform.tfvars` makes it easy to manage these configurations since Terraform automatically loads the variables defined in this file when running commands. This eliminates the need for providing values explicitly on the command line or modifying configuration files directly, allowing for a cleaner separation of environment-specific settings. While other file options exist, such as `.netvars`, `.env`, and `.config`, they do not serve the same primary function of directly storing environment-specific variable values in a standard and recognized manner as `terraform.tfvars` does in Terraform usage. This is why `terraform.tfvars` is the preferred choice for managing multiple environments effectively.

9. What is the primary purpose of 'terraform import' command?

A. To apply a state configuration

B. To remove a resource from the state

C. To import existing resources into Terraform management

D. To create new resources

The primary purpose of the 'terraform import' command is to bring existing resources that were created outside of Terraform under its management. This command allows users to connect the resources that are already provisioned in a cloud provider or on-premises infrastructure to Terraform's state file. Once imported, Terraform can track these resources, manage their configurations, and apply subsequent changes as specified in the Terraform configuration files. This command is particularly useful in scenarios where teams want to begin using Terraform for resources that were manually provisioned earlier or by other means, ensuring that these resources can be managed consistently moving forward. The imported resource will then be represented in Terraform's state file and can be modified or destroyed based on future Terraform plans and applies. In contrast, applying a state configuration involves applying changes derived from Terraform configurations to resources, removing a resource from the state is about disassociating Terraform management from it, and creating new resources relates to deploying new infrastructure rather than managing pre-existing resources.

10. Which of the following is a valid terraform import command?

A. terraform import aws_instance.foo module=foo

B. terraform import aws_instance.foo address _id=i-abcd1234

C. terraform import 'aws_instance.baz[0]' i-abcd1234

D. terraform import resource.aws_instance.foo i-abcd1234

In Terraform, the `import` command allows you to take existing infrastructure and bring it under Terraform management, which is essential for integrating resources that were created outside of Terraform. The correct syntax for the import command is crucial for it to function properly. For option C, the command `terraform import 'aws_instance.baz[0]' i-abcd1234` is correctly structured. This command designates the specific resource type—here, an AWS EC2 instance with an index in a list—followed by the unique identifier of the existing resource. The use of quotes around `aws_instance.baz[0]` helps to appropriately escape special characters for the Terraform command line, ensuring the correct parsing of the resource address. The use of the index `[0]` indicates that this is the first instance of a resource within a list, which corresponds with how Terraform handles indexed resources. The second part of the command, `i-abcd1234`, is the identifier of the existing AWS instance, formatted correctly to satisfy Terraform's requirements for association with a resource. Thus, this option conforms to the necessary syntax and structure for a valid Terraform import command, effectively facilitating the import process of the specified AWS instance.

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!