TERRAFORM ASSOCIATE PRACTICE EXAM (SAMPLE)

STUDY GUIDE

SAMPLE STUDY GUIDE WITH LIMITED QUESTIONS

VISIT US ONLINE FOR THE FULL VERSION STUDY GUIDE

https://terraformassociate.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. **What feature stops multiple admins from changing the Terraform state at the same time?**

   A. State versioning

   B. State locking

   C. State mirroring

   D. State auditing

2. **What happens when you run the terraform taint command on a managed resource?**

   A. Terraform will only mark the resource without modifying it

   B. Terraform will immediately destroy and recreate the resource

   C. Terraform will ignore the resource and move on

   D. Terraform will backup the resource before destroying it

3. **How can you ensure that a resource is recreated during the next apply?**

   A. By editing the configuration

   B. By tainting the resource

   C. By applying manually

   D. By deleting the resource

4. **Which of the following best describes the term 'resource' in Terraform?**

   A. Any physical or virtual machine managed by Terraform

   B. A specific configuration and attributes used to define an infrastructure component

   C. Only the cloud provider's underlying services

   D. A group of related Terraform modules

5. **Will terraform apply fail if terraform plan was not executed first?**

   A. True

   B. False

   C. Only in specific conditions

   D. It depends on the configuration

6. Which option cannot be used to keep secrets out of Terraform configuration files?

A. Environment variables

B. Secure string

C. Terraform variables file

D. Remote state file

7. Which provisioner runs a process on the created resource?

A. local-exec

B. remote-exec

C. null-exec

D. command-exec

8. Are all modules on the official Terraform Module Registry verified by HashiCorp?

A. True

B. False

C. Some modules are verified

D. Verification is optional

9. You need to migrate a workspace to use a remote backend. After updating your configuration, what command do you run to perform the migration?

A. terraform apply

B. terraform migrate

C. terraform init

D. terraform push

10. Which command would you use to manage different state files for development and production environments?

A. terraform select

B. terraform workspace

C. terraform environment

D. terraform switch

Answers

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

Explanations

1. What feature stops multiple admins from changing the Terraform state at the same time?

A. State versioning

B. State locking

C. State mirroring

D. State auditing

State locking is a critical feature in Terraform that prevents multiple administrators from making simultaneous changes to the Terraform state file. When a user starts a plan or apply operation, Terraform creates a lock on the state file, ensuring that no other operations can modify it until the current operation is completed. This mechanism helps avoid race conditions and potential conflicts that could arise if multiple changes were attempted at the same time. For example, if two users were to apply changes to the infrastructure concurrently without state locking, they could inadvertently overwrite each other's changes, leading to an inconsistent state. By using state locking, Terraform ensures that the integrity of the state is maintained and that operations are executed in a controlled manner, which is essential for the reliability and stability of the infrastructure management process. State versioning provides a history of changes, allowing users to roll back to previous states, but it does not prevent concurrent access to the state. State mirroring refers to duplicating the state across different backends for redundancy and availability but does not address the issue of simultaneous modifications. State auditing tracks changes made to the state for accountability but still allows for multiple changes without coordination. Thus, state locking is the key feature that safeguards against concurrent modifications, maintaining a single source of truth for the Terraform-managed infrastructure.

2. What happens when you run the terraform taint command on a managed resource?

A. Terraform will only mark the resource without modifying it

B. Terraform will immediately destroy and recreate the resource

C. Terraform will ignore the resource and move on

D. Terraform will backup the resource before destroying it

When the terraform taint command is executed on a managed resource, it effectively marks that resource for recreation. This means that during the next apply operation, Terraform will recognize that the resource is tainted and will destroy the existing instance of that resource before creating a new one in its place. The goal of this command is to ensure that a new version of the resource will be provisioned, which is particularly useful if the resource is in a problematic state or if you want to enforce a re-evaluation of its configuration. The marking process does not directly modify the resource at the moment the taint command is run; rather, it flags it for actions to take effect later during the apply phase. Consequently, the infrastructure is updated accordingly during the subsequent apply, guaranteeing consistency and allowing for any configuration changes to be reapplied. This behavior is pivotal in maintaining the desired state that Terraform manages.

3. How can you ensure that a resource is recreated during the next apply?

A. By editing the configuration

B. By tainting the resource

C. By applying manually

D. By deleting the resource

To ensure that a resource is recreated during the next apply, tainting the resource is an effective method. When a resource is tainted, it signals Terraform that the resource is in a bad state or needs to be replaced during the next apply operation. This can happen for various reasons, such as when a manual change is made directly to the resource outside of Terraform, and you want to ensure that Terraform can manage the resource again correctly. By using the `terraform taint` command, you mark the designated resource as needing recreation. During the next `terraform apply`, Terraform recognizes that the resource has been tainted and therefore plans to destroy the existing physical resource and create a new one. This approach is essential in managing the lifecycle of resources efficiently and effectively, ensuring that you can react to any issues or changes that may have occurred. The other options, while they can influence the state of the infrastructure in different ways, do not directly indicate that Terraform should recreate a resource. Editing the configuration may change how a resource is defined but does not guarantee a recreation unless specific attributes that trigger a recreation are modified. Applying manually could just reinforce existing configurations without enforcing a recreation. Deleting the resource could lead to its removal, but it doesn't communicate an intention to

4. Which of the following best describes the term 'resource' in Terraform?

A. Any physical or virtual machine managed by Terraform

B. A specific configuration and attributes used to define an infrastructure component

C. Only the cloud provider's underlying services

D. A group of related Terraform modules

The term 'resource' in Terraform best refers to a specific configuration and attributes used to define an infrastructure component. In Terraform, a resource represents a single piece of infrastructure that you want to create or manage. This could include components like virtual machines, storage accounts, or networking configurations, among others. When you define a resource in a Terraform configuration file, you specify its type (such as AWS EC2 instance, Azure Virtual Machine, etc.) and its properties or attributes (like size, region, and any other configurations pertinent to that resource). This enables Terraform to understand what infrastructure components you need and allows it to manage these components effectively through the declarative configuration model. Aside from resources, other terms like modules or providers relate to other aspects of Terraform but do not capture the essence of what a resource is meant to represent in the context of infrastructure management and configuration.

5. Will terraform apply fail if terraform plan was not executed first?

A. True

B. False

C. Only in specific conditions

D. It depends on the configuration

When you use Terraform, the command `terraform apply` is designed to execute the changes specified in a configuration file based on the current state of your infrastructure. It does not require a `terraform plan` to be run first for the apply to be successful. The primary role of `terraform plan` is to generate an execution plan, which shows you what actions Terraform will take to reach the desired state defined in your configuration. However, it's not a prerequisite for running `terraform apply`. You can apply changes directly without reviewing the plan first by simply executing `terraform apply`. In practice, while it's recommended to run `terraform plan` before applying changes—especially in a production environment—to ensure that you understand the upcoming changes and can catch any unintended actions, Terraform is fully capable of applying configurations based on the current state even if the plan command has not been executed beforehand. Thus, the application of the changes will not inherently fail just because a plan was not previously generated, leading to the conclusion that the correct answer is that it will not fail simply due to the absence of a prior plan step.

6. Which option cannot be used to keep secrets out of Terraform configuration files?

A. Environment variables

B. Secure string

C. Terraform variables file

D. Remote state file

Using secure strings to keep secrets out of Terraform configuration files refers to an approach that may not be universally supported in all contexts. It may imply an idea but is not a specific feature of Terraform like the other options. In contrast, environment variables are commonly used to inject sensitive data such as API keys without hardcoding them into configuration files. Terraform variables files allow users to define sensitive variables, which can be loaded with a command, but they still require careful management to ensure the files themselves do not inadvertently expose sensitive information. Remote state files, when configured properly, can store state securely and can help manage sensitive data from being directly embedded in configurations. However, secure strings as a standalone method may lack the robust support or consistency found in Terraform's established methods for managing secrets effectively. Therefore, it stands out as the least actionable option specifically related to keeping secrets away from configuration files.

7. Which provisioner runs a process on the created resource?

A. local-exec

B. remote-exec

C. null-exec

D. command-exec

The provisioner that runs a process on the created resource is the remote-exec provisioner. This type of provisioner is designed to execute scripts or commands directly on the remote resource that has just been created. For example, if you are provisioning a virtual machine in the cloud, the remote-exec provisioner allows you to run commands within the context of that machine after it has been provisioned. Using the remote-exec provisioner is beneficial when you need to configure the resource or install software as part of the deployment process. It connects to the resource via SSH (for Linux-based systems) or WinRM (for Windows), executing the specified commands in the proper environment of the created resource. In contrast, the local-exec provisioner runs commands on the machine where Terraform is executed, not on the target resource itself. Null-exec is a placeholder provisioner that does nothing, and "command-exec" is not a recognized Terraform provisioner type. Thus, remote-exec is the correct choice for running processes on the resource that has been created.

8. Are all modules on the official Terraform Module Registry verified by HashiCorp?

A. True

B. False

C. Some modules are verified

D. Verification is optional

The assertion that all modules on the official Terraform Module Registry are verified by HashiCorp is incorrect. In reality, not every module listed on the registry has undergone a verification process. The verification process is intended to provide a level of quality assurance, signaling that certain modules meet specific standards of security, reliability, and usability. However, it is up to module authors to seek verification, and as such, many modules remain unverified. The presence of unverified modules means that users should exercise caution when choosing which modules to include in their infrastructure-as-code projects. This situation emphasizes the importance of reviewing the code, documenting best practices, and considering the source and maintenance of the modules being utilized. While it's true that some modules may receive a verification badge, indicating that they've been reviewed by HashiCorp, it is not comprehensive coverage across all modules in the registry. Hence, the claim that all modules are verified is inaccurate, leading to the conclusion that the correct answer is that it is false.

9. You need to migrate a workspace to use a remote backend. After updating your configuration, what command do you run to perform the migration?

A. terraform apply

B. terraform migrate

C. terraform init

D. terraform push

The correct choice is to use the terraform init command to perform the migration of a workspace to a remote backend. This command initializes the Terraform configuration. When you change the backend configuration in your Terraform setup to switch to a remote backend, running terraform init is necessary because it prepares your working directory for other commands by setting up the specific backend, downloading necessary plugins, and ensuring everything is configured correctly. During this initialization, Terraform recognizes the backend change and handles the migration of the existing state file to the new remote backend. This process includes any state file modifications needed to align the current state with the remote backend management, ensuring that your resources are correctly tracked and managed in the new environment. The other options do not serve the purpose of migrating a workspace to a remote backend. For instance, using terraform apply would attempt to apply any changes in your configuration but would not address backend migration. Similarly, terraform migrate is not a valid command in Terraform, and terraform push is not used for backend migration; it is associated with specific workflow processes in some tools, but not standard Terraform functionality. Thus, running terraform init is the proper step for handling the backend migration process effectively.

10. Which command would you use to manage different state files for development and production environments?

A. terraform select

B. terraform workspace

C. terraform environment

D. terraform switch

Using the command associated with workspaces is the appropriate way to manage different state files for development and production environments in Terraform. This command enables you to create, switch, and manage multiple workspaces, where each workspace maintains its own separate state file. In Terraform, the default workspace is called "default." When you create additional workspaces, Terraform stores the state files for these workspaces separately. This allows you to manage different configurations and environments effectively without conflicts between them. For example, you might have a workspace for production and another for development. By using the workspace command, you can switch between these environments seamlessly, ensuring that actions taken in one do not inadvertently affect the other. The other options do not accurately reflect the method for managing state files in this context. Using the options associated with selecting or switching environments does not exist in Terraform's command repertoire, as does a non-existent environment command. Only workspaces provide the necessary functionality for isolating state files per environment within Terraform.

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://terraformassociate.examzify.com

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