[Jun 16, 2022 Free HashiCorp Infrastructure Automation TA-002-P Exam Question [Q50-Q72

[Jun 16, 2022] Free HashiCorp Infrastructure Automation TA-002-P Exam Question
TA-002-P dumps & HashiCorp Infrastructure Automation sure practice dumps

Difficulty in Attempting HashiCorp Certified: Terraform Associate TA-002-P Professional Exam

A comprehensive range of HashiCorp Certified: Terraform Associate TA-002-P PROFESSIONAL exam dumps for Certification have been recognized. The truth that applicants need to prepare mindfully doesn't make endorsements easy. It needs some investment to earn from HashiCorp professional course. Each exam includes answers and questions that help candidates complete their final assessment. You will complete the evaluation after you have taken the exam and taken it in our modules. Yet, it doesn't stop there; on account of our full aides, you will, in any situation, be admissible in your profession. You will deliver your results later on. To design any material for you, we have a high-level plan. In the progression of an object, we have utilized the most recent subtleties.

Hands-on experience is the most reliable form of preparation there is. Analyzing the exam guide for information about the competencies evaluated in the certification exam is a good practice to prepare for the certification.

- Learn the Terraform Core workflow- Understand basic things about Terraform Cloud & Terraform Enterprise- Learn Terraform Cloud interactively with Katacoda- Commence training terraform with the console- Study Terraform Basics- Go through HashiCorp's resource library- Study to Terraform with HashiCorp's official learning platform

The examination is scored based on a set standard built by HashiCorp experts who are motivated by certification industry's most

reliable practices and guidelines.
In order to earn Terraform Associate certification, one must pass a minimum of any four HashiCorp Terraform Associate certification exams. Terraform Associate certification is legitimate for 2 years from the date of achievement. When it come to recertification, the candidate can renew the certification by passing any four Terraform Associate exams at a Pearson VUE test center. Achieving certification automatically renews your Terraform Associate certification if the Terraform Associate certification is not expired.
Getting the certification renews your Terraform Associate certification, even if the applicant's previous certification has expired.
This examination can not be instantly finished because the HashiCorp Certified: Terraform Associate TA-002-P PROFESSIONAL exam dumps need to pass the examinations, these exam dumps require time and correct and up to date content to pass the exam with effectiveness. Several applicants are doubtful about the nature of questions posed in the exam and the complexity of exam questions and the time needed to finish the questions before writing a credential Professional certification. The most suitable way to pass the Professional Test is to question and prepare with HashiCorp Certified: Terraform Associate TA-002-P PROFESSIONAL exam dumps.
This examination can not be instantly finished because the **HASHICORP TA-002 practice exam** and **HASHICORP TA-002 practice test** need to pass the examinations, these exam dumps require time and correct and up to date content to pass the exam with effectiveness. Several applicants are doubtful about the nature of questions posed in the exam and the complexity of exam questions and the time needed to finish the questions before writing a credential Professional certification. The most suitable way to pass the Professional Test is to question and prepare with **HASHICORP TA-002 practice exam** and **HASHICORP TA-002 practice test**.

How to schedule HashiCorp Certified: Terraform Associate TA-002-P Professional Exam

To apply for the HashiCorp Certified: Terraform Associate TA-002-P Professional Exam, You have to follow these steps:
? Step 1: Go to the HashiCorp Certified: Terraform Associate TA-002-P Professional Official Site. You must first create an account, use your email address to register. You must purchase your training through your local distributor. If you are a partner, you must first create an account on the Partner Portal. You must use your company email address to register.
? Step 2: Read the instruction Carefully
? Step 3: Follow the given steps
? Step 4: Apply for the HashiCorp Certified: Terraform Associate TA-002-P-Professional Exam

NO.50 State is a requirement for Terraform to function
* True
* False
State is a necessary requirement for Terraform to function. It is often asked if it is possible for Terraform to work without state, or for Terraform to not use state and just inspect cloud resources on every run.

Purpose of Terraform State

State is a necessary requirement for Terraform to function. It is often asked if it is possible for Terraform to work without state, or for Terraform to not use state and just inspect cloud resources on every run. This page will help explain why Terraform state is required.

As you’ll see from the reasons below, state is required. And in the scenarios where Terraform may be able to get away without state, doing so would require shifting massive amounts of complexity from one place (state) to another place (the replacement concept).

1. Mapping to the Real World

Terraform requires some sort of database to map Terraform config to the real world. When you have a resource resource “aws_instance” “foo” in your configuration, Terraform uses this map to know that instance i-abcd1234 is represented by that resource.

For some providers like AWS, Terraform could theoretically use something like AWS tags. Early prototypes of Terraform actually had no state files and used this method. However, we quickly ran into problems. The first major issue was a simple one: not all resources support tags, and not all cloud providers support tags.

Therefore, for mapping configuration to resources in the real world, Terraform uses its own state structure.

2. Metadata

Alongside the mappings between resources and remote objects, Terraform must also track metadata such as resource dependencies.

Terraform typically uses the configuration to determine dependency order. However, when you delete a resource from a Terraform configuration, Terraform must know how to delete that resource. Terraform can see that a mapping exists for a resource not in your configuration and plan to destroy. However, since the configuration no longer exists, the order cannot be determined from the configuration alone.

To ensure correct operation, Terraform retains a copy of the most recent set of dependencies within the state. Now Terraform can still determine the correct order for destruction from the state when you delete one or more items from the configuration.

One way to avoid this would be for Terraform to know a required ordering between resource types. For example, Terraform could know that servers must be deleted before the subnets they are a part of. The complexity for this approach quickly explodes, however: in addition to Terraform having to understand the ordering semantics of every resource for every cloud, Terraform must also understand the ordering across providers.

Terraform also stores other metadata for similar reasons, such as a pointer to the provider configuration that was most recently used with the resource in situations where multiple aliased providers are present.

3. Performance

In addition to basic mapping, Terraform stores a cache of the attribute values for all resources in the state. This is the most optional feature of Terraform state and is done only as a performance improvement.

When running a terraform plan, Terraform must know the current state of resources in order to effectively determine the changes that it needs to make to reach your desired configuration.

For small infrastructures, Terraform can query your providers and sync the latest attributes from all your resources. This is the default behavior of Terraform: for every plan and apply, Terraform will sync all resources in your state.

For larger infrastructures, querying every resource is too slow. Many cloud providers do not provide APIs to query multiple resources at once, and the round trip time for each resource is hundreds of milliseconds. On top of this, cloud providers almost always have API rate limiting so Terraform can only request a certain number of resources in a period of time. Larger users of Terraform make heavy use of the -refresh=false flag as well as the -target flag in order to work around this. In these scenarios, the cached state is treated as the record of truth.

4. Syncing

In the default configuration, Terraform stores the state in a file in the current working directory where Terraform was run. This is okay for getting started, but when using Terraform in a team it is important for everyone to be working with the same state so that operations will be applied to the same remote objects.

Remote state is the recommended solution to this problem. With a fully-featured state backend, Terraform can use remote locking as a measure to avoid two or more different users accidentally running Terraform at the same time, and thus ensure that each Terraform run begins with the most recent updated state.

NO.51 Named workspaces are not a suitable isolation mechanism for strong separation between staging and production?
* True
* False
Explanation

Organizations commonly want to create a strong separation between multiple deployments of the same infrastructure serving different development stages (e.g. staging vs. production) or different internal teams. In this case, the backend used for each deployment often belongs to that deployment, with different credentials and access controls. Named workspaces are not a suitable isolation mechanism for this scenario.

https://www.terraform.io/docs/state/workspaces.html#when-to-use-multiple-workspaces

NO.52 You have been given requirements to create a security group for a new application. Since your organization standardizes on Terraform, you want to add this new security group with the fewest number of lines of code.

What feature could you use to iterate over a list of required tcp ports to add to the new security group?
* dynamic backend
* splat expression
* terraform import
* dynamic block
Explanation

A dynamic block acts much like a for expression, but produces nested blocks instead of a complex typed value. It iterates over a given complex value and generates a nested block for each element of that complex value.

https://www.terraform.io/docs/configuration/expressions.html#dynamic-blocks

NO.53 Which of the following actions are performed during a terraform init?
* Initializes downloaded and/or installed providers
* Initializes the backend configuration
* Provisions the declared resources in your configuration
* Download the declared providers which are supported by HashiCorp
The terraform init command is used to initialize a working directory containing Terraform configuration files. This is the first command that should be run after writing a new Terraform configuration or cloning an existing one from version control. It is safe to run this command multiple times.

This command is always safe to run multiple times, to bring the working directory up to date with changes in the configuration. Though subsequent runs may give errors, this command will never delete your existing configuration or state.

terraform init command does –

* Copy a Source Module

* Backend Initialization

* Child Module Installation

* Plugin Installation

https://www.terraform.io/docs/commands/init.html

NO.54 You run a local-exec provisioner in a null resource called null_resource.run_script and realize that you need to rerun the script.

Which of the following commands would you use first?
* terraform taint null_resource.run_script
* terraform apply -target=null_resource.run_script
* terraform validate null_resource.run_script
* terraform plan -target=null_resource.run_script

NO.55 What resource dependency information is stored in Terraform’s state?
* Only implicit dependencies are stored in state.
* Both implicit and explicit dependencies are stored in state.
* Only explicit dependencies are stored in state.
* No dependency information is stored in state.

Terraform state captures all dependency information, both implicit and explicit. One purpose for state is to determine the proper order to destroy resources. When resources are created all of their dependency information is stored in the state. If you destroy a resource with dependencies, Terraform can still determine the correct destroy order for all other resources because the dependencies are stored in the state.

https://www.terraform.io/docs/state/purpose.html#metadata

NO.56 Which of the following Terraform commands will automatically refresh the state unless supplied with additional flags or arguments? Choose TWO correct answers.
* terraform state
* terraform apply
* terraform plan
* terraform validate
* terraform output

NO.57 What command should you run to display all workspaces for the current configuration?
* terraform workspace
* terraform workspace show
* terraform workspace list
* terraform show workspace

terraform workspace list

The command will list all existing workspaces.

NO.58 Terraform Enterprise (also referred to as pTFE) requires what type of backend database for a clustered deployment?
* PostgreSQL
* Cassandra
* MySQL
* MSSQL

Explanation

External Services mode stores the majority of the stateful data used by the instance in an external PostgreSQL database and an external S3-compatible endpoint or Azure blob storage. There is still critical data stored on the instance that must be managed with snapshots. Be sure to check the PostgreSQL Requirements for information that needs to be present for Terraform Enterprise to work. This option is best for users with expertise managing PostgreSQL or users that have access to managed PostgreSQL offerings like AWS RDS.

NO.59 You have used Terraform to create an ephemeral development environment in the cloud and are now ready to destroy all the infrastructure described by your Terraform configuration. To be safe, you would like to first see all the infrastructure that will be deleted by Terraform.

Which command should you use to show all of the resources that will be deleted? (Choose two.)
* Run terraform plan -destroy.
* This is not possible. You can only show resources that will be created.
* Run terraform state rm *.
* Run terraform destroy and it will first output all the resources that will be deleted before prompting for approval.
Reference: https://www.terraform.io/docs/cli/commands/state/rm.html

NO.60 Which one of the following will run echo 0 and echo 1 on a newly created host?
* provisioner “local-exec” { command = “echo 0”

command = “echo 1”

}
* provisioner “remote-exec” {

inline = [

echo 0,

echo 1

]

}
* provisioner “remote-exec” {

command = “${echo 0}”

command = “${echo 1}”

}
* provisioner “remote-exec” {

inline = [

“echo 0”,

“echo 1”

]

}
remote-exec Provisioner

Example usage

resource “aws_instance” “web” {

…

provisioner “remote-exec” {

inline = [

“puppet apply”,

“consul join ${aws_instance.web.private_ip}”,

]

}

}

NO.61 What command should you run to display all workspaces for the current configuration?
* terraform workspace
* terraform workspace show
* terraform workspace list
* terraform show workspace
terraform workspace list

The command will list all existing workspaces.

Reference: https://www.terraform.io/docs/cli/commands/workspace/list.html

NO.62 Which of the following state management command allow you to retrieve a list of resources that are part of the state file?
* terraform state list
* terraform state view
* terraform view
* terraform list
The terraform state list command is used to list resources within a Terraform state.

Usage: terraform state list [options] [address…]

The command will list all resources in the state file matching the given addresses (if any). If no addresses are given, all resources are listed.

https://www.terraform.io/docs/commands/state/list.html

NO.63 Refer below code where pessimistic constraint operator has been used to specify a version of a provider.

terraform { required_providers { aws = “~> 1.1.0” }}

Which of the following options are valid provider versions that satisfy the above constraint. (select two)
* 1.1.1
* 1.2.9
* 1.1.8
* 1.2.0

Pessimistic constraint operator, constraining both the oldest and newest version allowed. For example, ~> 0.9 is equivalent to >= 0.9, < 1.0, and ~> 0.8.4, is equivalent to >= 0.8.4, < 0.9

NO.64 You have provisioned some virtual machines (VMs) on Google Cloud Platform (GCP) using the gcloud command line tool. However, you are standardizing with Terraform and want to manage these VMs using Terraform instead.

What are the two things you must do to achieve this? (Choose two.)
* Provision new VMs using Terraform with the same VM names
* Use the terraform import command for the existing VMs
* Write Terraform configuration for the existing VMs
* Run the terraform import-gcp command

Explanation

The terraform import command is used to import existing infrastructure. Import existing Google Cloud resources into Terraform with Terraformer.

Reference: https://www.terraform.io/docs/cli/import/usage.html https://cloud.google.com/docs/terraform

NO.65 A fellow developer on your team is asking for some help in refactoring their Terraform code. As part of their application’s architecture, they are going to tear down an existing deployment managed by Terraform and deploy new. However, there is a server resource named aws_instance.ubuntu[1] they would like to keep to perform some additional analysis.

What command should be used to tell Terraform to no longer manage the resource?
* terraform apply rm aws_instance.ubuntu[1]
* terraform state rm aws_instance.ubuntu[1]
* terraform plan rm aws_instance.ubuntu[1]
* terraform delete aws_instance.ubuntu[1]

NO.66 Which argument(s) is (are) required when declaring a Terraform variable?
* type
* default
* description
* All of the above
* None of the above

Explanation

The variable declaration can also include a default argument.

Reference: https://www.terraform.io/docs/language/values/variables.html

NO.67 Which of the below command will upgrade the provider version to the latest acceptable one?

* terraform plan upgrade
* terraform provider -upgrade
* terraform init -upgrade
* terraform init -update

To upgrade to the latest acceptable version of each provider, run terraform init -upgrade. This command also upgrades to the latest versions of all Terraform modules.

https://www.terraform.io/docs/configuration/providers.html

NO.68 Module variable assignments are inherited from the parent module and do not need to be explicitly set.
* True
* False

NO.69 Which one of the following will run echo 0 and echo 1 on a newly created host?
* provisioner “local-exec” { command = “echo 0”

command = “echo 1”

}
* provisioner “remote-exec” {

inline = [

echo 0,

echo 1

]

}
* provisioner “remote-exec” {

command = “${echo 0}”

command = “${echo 1}”

}
* provisioner “remote-exec” {

inline = [

“echo 0”,

“echo 1”

]

}

Explanation

remote-exec Provisioner

Example usage

resource “aws_instance” “web” {

…

provisioner “remote-exec” {

inline = [

“puppet apply”,

“consul join ${aws_instance.web.private_ip}”,

]

}

}

NO.70 Refer to the following terraform variable definition

variable “track_tag” { type = list default = [“data_ec2″,”integration_ec2″,”digital_ec2”]} track_tag = { Name = element(var.track_tag,count.index)} If count.index is set to 2, which of the following values will be assigned to the name attribute of track_tag variable?
* integration_ec2
* digital_ec2
* track_tag
* data_ec2

NO.71 Terraform must track metadata such as resource dependencies. Where is this data stored?
* workspace
* backend
* state file
* metadata store

Terraform typically uses the configuration to determine dependency order. However, when you delete a resource from a Terraform configuration, Terraform must know how to delete that resource. Terraform can see that a mapping exists for a resource not in your configuration and plan to destroy. However, since the configuration no longer exists, the order cannot be determined from the configuration alone.

To ensure correct operation, Terraform retains a copy of the most recent set of dependencies within the state. Now Terraform can still determine the correct order for destruction from the state when you delete one or more items from the configuration.

https://www.terraform.io/docs/state/purpose.html#metadata

NO.72 True or False. The terraform refresh command is used to reconcile the state Terraform knows about (via its state file) with the real-world infrastructure. If drift is detected between the real-world infrastructure and the last known-state, it will modify the

infrastructure to correct the drift.
* False
* True

Explanation

https://www.terraform.io/docs/commands/refresh.html

Sample Questions

During a terraform plan, a resource is successfully created but eventually fails during provisioning. What happens to the resource?

- It is automatically deleted- The terraform plan is rolled back and all provisioned resources are removed- The resource is marked as tainted- Terraform attempts to provision the resource up to three times before exiting with an error

Explanation

If a resource successfully creates but fails during provisioning, Terraform will error and mark the resource as ?tainted'. A resource that is tainted has been physically created, but can't be considered safe to use since provisioning failed. Terraform also does not automatically roll back and destroy the resource during the apply when the failure happens, because that would go against the execution plan: the execution plan would've said a resource will be created, but does not say it will ever be deleted.

True or False? When using the Terraform provider for Vault, the tight integration between these HashiCorp tools provides the ability to mask secrets in the terraform plan and state files.

- False- True

Explanation

Currently, Terraform has no mechanism to redact or protect secrets that are returned via data sources, so secrets read via this provider will be persisted into the Terraform state, into any plan files, and in some cases in the console output produced while planning and applying. These artifacts must, therefore, all be protected accordingly.

HashiCorp TA-002-P Actual Questions and Braindumps:
https://www.examcollectionpass.com/HashiCorp/TA-002-P-practice-exam-dumps.html]