

## Terraform-Associate-003 Dumps

### HashiCorp Certified: Terraform Associate (003)

<https://www.certleader.com/Terraform-Associate-003-dumps.html>



**NEW QUESTION 1**

Which of the following is not a key principle of infrastructure as code?

- A. Self-describing infrastructure
- B. Idempotence
- C. Versioned infrastructure
- D. Golden images

**Answer:** D

**Explanation:**

The key principle of infrastructure as code that is not listed among the options is golden images. Golden images are pre-configured, ready-to-use virtual machine images that contain a specific set of software and configuration. They are often used to create multiple identical instances of the same environment, such as for testing or production. However, golden images are not a principle of infrastructure as code, but rather a technique that can be used with or without infrastructure as code. The other options are all key principles of infrastructure as code, as explained below:

? Self-describing infrastructure: This means that the infrastructure is defined in code that describes its desired state, rather than in scripts that describe the steps to create it. This makes the infrastructure easier to understand, maintain, and reproduce.

? Idempotence: This means that applying the same infrastructure code multiple times will always result in the same state, regardless of the initial state. This makes the infrastructure consistent and predictable, and avoids errors or conflicts caused by repeated actions.

? Versioned infrastructure: This means that the infrastructure code is stored in a version control system, such as Git, that tracks the changes and history of the code. This makes the infrastructure code reusable, auditable, and collaborative, and enables practices such as branching, merging, and rollback. References = [Introduction to Infrastructure as Code with Terraform], [Infrastructure as Code in a Private or Public Cloud]

**NEW QUESTION 2**

A terraform apply can not infrastructure.

- A. change
- B. destroy
- C. provision
- D. import

**Answer:** D

**Explanation:**

The terraform import command is used to import existing infrastructure into Terraform's state. This allows Terraform to manage and destroy the imported infrastructure as part of the configuration. The terraform import command does not modify the configuration, so the imported resources must be manually added to the configuration after the import. References = [Importing Infrastructure]

**NEW QUESTION 3**

HashiCorp Configuration Language (HCL) supports user-defined functions.

- A. True
- B. False

**Answer:** B

**Explanation:**

HashiCorp Configuration Language (HCL) does not support user-defined functions. You can only use the built-in functions that are provided by the language. The built-in functions allow you to perform various operations and transformations on values within expressions. The general syntax for function calls is a function name followed by comma-separated arguments in parentheses, such as max(5, 12, 9). You can find the documentation for all of the available built-in functions in the Terraform Registry or the Packer Documentation, depending on which tool you are using. References = : Functions - Configuration Language | Terraform : Functions - Configuration Language | Packer

**NEW QUESTION 4**

You must initialize your working directory before running terraform validate.

- A. True
- B. False

**Answer:** A

**Explanation:**

You must initialize your working directory before running terraform validate, as it will ensure that all the required plugins and modules are installed and configured properly. If you skip this step, you may encounter errors or inconsistencies when validating your configuration files.

**NEW QUESTION 5**

Why would you use the -replace flag for terraform apply?

- A. You want Terraform to ignore a resource on the next apply
- B. You want Terraform to destroy all the infrastructure in your workspace
- C. You want to force Terraform to destroy a resource on the next apply
- D. You want to force Terraform to destroy and recreate a resource on the next apply

**Answer:** D

**Explanation:**

The -replace flag is used with the terraform apply command when there is a need to explicitly force Terraform to destroy and then recreate a specific resource during the next apply. This can be necessary in situations where a simple update is insufficient or when a resource must be re-provisioned to pick up certain changes.

**NEW QUESTION 6**

What does this code do?

```
terraform {
  required_providers {
    aws = "~> 3.0"
  }
}
```

- A. Requires any version of the AWS provider > = 3.0 and <4.0
- B. Requires any version of the AWS provider >= 3.0
- C. Requires any version of the AWS provider > = 3.0 major releases
- D. like 4.1
- E. Requires any version of the AWS provider > 3.0

**Answer:** A

**Explanation:**

This is what this code does, by using the pessimistic constraint operator (~>), which specifies an acceptable range of versions for a provider or module.

**NEW QUESTION 7**

When should you use the force-unlock command?

- A. You have a high priority change
- B. Automatic unlocking failed
- C. apply failed due to a state lock
- D. You see a status message that you cannot acquire the lock

**Answer:** B

**Explanation:**

You should use the force-unlock command when automatic unlocking failed. Terraform will lock your state for all operations that could write state, such as plan, apply, or destroy. This prevents others from acquiring the lock and potentially corrupting your state. State locking happens automatically on all operations that could write state and you won't see any message that it is happening. If state locking fails, Terraform will not continue. You can disable state locking for most commands with the -lock flag but it is not recommended. If acquiring the lock is taking longer than expected, Terraform will output a status message. If Terraform doesn't output a message, state locking is still occurring if your backend supports it. Terraform has a force-unlock command to manually unlock the state if unlocking failed. Be very careful with this command. If you unlock the state when someone else is holding the lock it could cause multiple writers. Force unlock should only be used to unlock your own lock in the situation where automatic unlocking failed. To protect you, the force-unlock command requires a unique lock ID. Terraform will output this lock ID if unlocking fails. This lock ID acts as a nonce, ensuring that locks and unlocks target the correct lock. The other situations are not valid reasons to use the force-unlock command. You should not use the force-unlock command if you have a high priority change, if apply failed due to a state lock, or if you see a status message that you cannot acquire the lock. These situations indicate that someone else is holding the lock and you should wait for them to finish their operation or contact them to resolve the issue. Using the force-unlock command in these cases could result in data loss or inconsistency. References = [State Locking], [Command: force-unlock]

**NEW QUESTION 8**

While attempting to deploy resources into your cloud provider using Terraform, you begin to see some odd behavior and experience slow responses. In order to troubleshoot you decide to turn on Terraform debugging. Which environment variables must be configured to make Terraform's logging more verbose?

- A. TF\_LOG\_PAIR
- B. TF\_LOG
- C. TF\_VAR\_log\_path
- D. TF\_VAR\_log\_level

**Answer:** B

**Explanation:**

To make Terraform's logging more verbose for troubleshooting purposes, you must configure the TF\_LOG environment variable. This variable controls the level of logging and can be set to TRACE, DEBUG, INFO, WARN, or ERROR, with TRACE providing the most verbose output. References = Detailed debugging instructions and the use of environment variables like TF\_LOG for increasing verbosity are part of Terraform's standard debugging practices

**NEW QUESTION 9**

Which is the best way to specify a tag of v1.0.0 when referencing a module stored in Git (for example. Git::https://example.com/vpc.git)?

- A. Append pref=v1.0.0 argument to the source path
- B. Add version = v1.0.0 parameter to module block

C. Nothing modules stored on GitHub always default to version 1.0.0

**Answer:** A

**Explanation:**

The best way to specify a tag of v1.0.0 when referencing a module stored in Git is to append ?ref=v1.0.0 argument to the source path. This tells Terraform to use a specific Git reference, such as a branch, tag, or commit, when fetching the module source code. For example, source = "git::https://example.com/vpc.git?ref=v1.0.0". This ensures that the module version is consistent and reproducible across different environments. References = [Module Sources], [Module Versions]

**NEW QUESTION 10**

Which of the following statements about Terraform modules is not true?

- A. Modules can call other modules
- B. A module is a container for one or more resources
- C. Modules must be publicly accessible
- D. You can call the same module multiple times

**Answer:** C

**Explanation:**

This is not true, as modules can be either public or private, depending on your needs and preferences. You can use the Terraform Registry to publish and consume public modules, or use Terraform Cloud or Terraform Enterprise to host and manage private modules.

**NEW QUESTION 10**

Which of these are features of Terraform Cloud? Choose two correct answers.

- A. Automated infrastructure deployment visualization
- B. Automatic backups
- C. A web-based user interface (UI)
- D. Remote state storage

**Answer:** CD

**Explanation:**

These are features of Terraform Cloud, which is a hosted service that provides a web-based UI, remote state storage, remote operations, collaboration features, and more for managing your Terraform infrastructure.

**NEW QUESTION 13**

Which command should you run to check if all code in a Terraform configuration that references multiple modules is properly formatted without making changes?

- A. terraform fmt -write=false
- B. terraform fmt -list -recursive
- C. terraform fmt -check -recursive
- D. terraform fmt -check

**Answer:** C

**Explanation:**

This command will check if all code in a Terraform configuration that references multiple modules is properly formatted without making changes, and will return a non-zero exit code if any files need formatting. The other commands will either make changes, list the files that need formatting, or not check the modules.

**NEW QUESTION 15**

You cannot install third party plugins using terraform init.

- A. True
- B. False

**Answer:** B

**Explanation:**

You can install third party plugins using terraform init, as long as you specify the plugin directory in your configuration or as a command-line argument. You can also use the terraform providers mirror command to create a local mirror of providers from any source.

**NEW QUESTION 17**

When you use a remote backend that needs authentication, HashiCorp recommends that you:

- A. Write the authentication credentials in the Terraform configuration files
- B. Keep the Terraform configuration files in a secret store
- C. Push your Terraform configuration to an encrypted git repository
- D. Use partial configuration to load the authentication credentials outside of the Terraform code

**Answer:** D

**Explanation:**

This is the recommended way to use a remote backend that needs authentication, as it allows you to provide the credentials via environment variables, command-line arguments, or interactive prompts, without storing them in the Terraform configuration files.

**NEW QUESTION 19**

Before you can use a remote backend, you must first execute terra-form init.

- A. True
- B. False

**Answer:** A

**Explanation:**

Before using a remote backend in Terraform, it is mandatory to run terraform init. This command initializes a Terraform working directory, which includes configuring the backend. If a remote backend is specified, terraform init will set up the working directory to use it, including copying any existing state to the remote backend if necessary. References = This principle is a fundamental part of working with Terraform and its backends, as outlined in general Terraform documentation and best practices. The specific HashiCorp Terraform Associate (003) study materials in the provided files did not include direct references to this information.

**NEW QUESTION 22**

What are some benefits of using Sentinel with Terraform Cloud/Terra form Cloud? Choose three correct answers.

- A. You can enforce a list of approved AWS AMIs
- B. Policy-as-code can enforce security best practices
- C. You can check out and check in cloud access keys
- D. You can restrict specific resource configurations, such as disallowing the use of CIDR=0.0.0.0/0.
- E. Sentinel Policies can be written in HashiCorp Configuration Language (HCL)

**Answer:** ABD

**Explanation:**

These are some of the benefits of using Sentinel with Terraform Cloud/Terraform Enterprise, as they allow you to implement logic-based policies that can access and evaluate the Terraform plan, state, and configuration. The other options are not true, as Sentinel does not manage cloud access keys, and Sentinel policies are written in Sentinel language, not HCL.

**NEW QUESTION 25**

Setting the TF\_LOG environment variable to DEBUG causes debug messages to be logged into stdout.

- A. True
- B. False

**Answer:** A

**Explanation:**

Setting the TF\_LOG environment variable to DEBUG causes debug messages to be logged into stdout, along with other log levels such as TRACE, INFO, WARN, and ERROR. This can be useful for troubleshooting or debugging purposes.

**NEW QUESTION 26**

What value does the Terraform Cloud private registry provide over the public Terraform Module Registry?

- A. The ability to share modules publicly with any user of Terraform
- B. The ability to restrict modules to members of Terraform Cloud or Enterprise organizations
- C. The ability to tag modules by version or release
- D. The ability to share modules with public Terraform users and members of Terraform Cloud Organizations

**Answer:** B

**Explanation:**

The Terraform Cloud private registry provides the ability to restrict modules to members of Terraform Cloud or Enterprise organizations. This allows you to share modules within your organization without exposing them to the public. The private registry also supports importing modules from your private VCS repositories. The public Terraform Module Registry, on the other hand, publishes modules from public Git repositories and makes them available to any user of Terraform. References = : Private Registry - Terraform Cloud : Terraform Registry - Provider Documentation

**NEW QUESTION 30**

You want to define a single input variable to capture configuration values for a server. The values must represent memory as a number, and the server name as a string.

Which variable type could you use for this input?

- A. List
- B. Object
- C. Map
- D. Terraform does not support complex input variables of different types

**Answer:** B

**Explanation:**

This is the variable type that you could use for this input, as it can store multiple attributes of different types within a single value. The other options are either invalid or incorrect for this use case.

**NEW QUESTION 31**



Which of the following is not a valid string function in Terraform?

- A. choaf
- B. join
- C. Split
- D. slice

**Answer:** A

**Explanation:**

This is not a valid string function in Terraform. The other options are valid string functions that can manipulate strings in various ways2.

**NEW QUESTION 35**

In a Terraform Cloud workspace linked to a version control repository speculative plan run start automatically commit changes to version control.

- A. True
- B. False

**Answer:** A

**Explanation:**

When you use a remote backend that needs authentication, HashiCorp recommends that you:

**NEW QUESTION 40**

You are writing a child Terraform module that provisions an AWS instance. You want to reference the IP address returned by the child module in the root configuration. You name the instance resource "main".

Which of these is the correct way to define the output value?

A)

```
output "instance_ip_addr" {  
    return aws_instance.main.private_ip  
}
```

B)

```
output "aws_instance.instance_ip_addr" {  
    return aws_instance.main.private_ip  
}
```

C)

```
output "aws_instance.instance_ip_addr" {  
    value = ${main.private_ip}  
}
```

D)

```
output "instance_ip_addr" {  
    value = aws_instance.main.private_ip  
}
```

- A. Option A
- B. Option B
- C. Option C

D. Option D

**Answer:** D

#### NEW QUESTION 44

You have deployed a new webapp with a public IP address on a cloud provider. However, you did not create any outputs for your code. What is the best method to quickly find the IP address of the resource you deployed?

- A. In a new folder, use the terraform\_remote\_state data source to load in the state file, then write an output for each resource that you find the state file
- B. Run terraform state list to find the name of the resource, then terraform state show to find the attributes including public IP address
- C. Run terraform output ip\_address to view the result
- D. Run terraform destroy then terraform apply and look for the IP address in stdout

**Answer:** B

#### Explanation:

This is a quick way to inspect the state file and find the information you need without modifying anything<sup>5</sup>. The other options are either incorrect or inefficient.

#### NEW QUESTION 46

When should you write Terraform configuration files for existing infrastructure that you want to start managing with Terraform?

- A. You can import infrastructure without corresponding Terraform code
- B. Terraform will generate the corresponding configuration files for you
- C. Before you run terraform Import
- D. After you run terraform import

**Answer:** C

#### Explanation:

You need to write Terraform configuration files for the existing infrastructure that you want to import into Terraform, otherwise Terraform will not know how to manage it. The configuration files should match the type and name of the resources that you want to import.

#### NEW QUESTION 47

Which of the following does terraform apply change after you approve the execution plan? (Choose two.)

- A. Cloud infrastructure Most Voted
- B. The .terraform directory
- C. The execution plan
- D. State file
- E. Terraform code

**Answer:** AD

#### Explanation:

The terraform apply command changes both the cloud infrastructure and the state file after you approve the execution plan. The command creates, updates, or destroys the infrastructure resources to match the configuration. It also updates the state file to reflect the new state of the infrastructure. The .terraform directory, the execution plan, and the Terraform code are not changed by the terraform apply command. References = Command: apply and Purpose of Terraform State

#### NEW QUESTION 50

The public Terraform Module Registry is free to use.

- A. True
- B. False

**Answer:** A

#### Explanation:

The public Terraform Module Registry is free to use, as it is a public service that hosts thousands of self-contained packages called modules that are used to provision infrastructure. You can browse, use, and publish modules to the registry without any cost.

#### NEW QUESTION 52

When do changes invoked by terraform apply take effect?

- A. After Terraform has updated the state file
- B. Once the resource provider has fulfilled the request
- C. Immediately
- D. None of the above are correct

**Answer:** B

#### Explanation:

Changes invoked by terraform apply take effect once the resource provider has fulfilled the request, not after Terraform has updated the state file or immediately. The state file is only a reflection of the real resources, not a source of truth.

#### NEW QUESTION 54

You have a Terraform configuration that defines a single virtual machine with no references to it, You have run terraform apply to create the resource, and then removed the resource definition from your Terraform configuration file. What will happen you run terraform apply in the working directory again?

- A. Terraform will remove the virtual machine from the state file, but the resource will still exist
- B. Nothing
- C. Terraform will error
- D. Terraform will destroy the virtual machine

**Answer:** D

**Explanation:**

This is what will happen if you run terraform apply in the working directory again, after removing the resource definition from your Terraform configuration file. Terraform will detect that there is a resource in the state file that is not present in the configuration file, and will assume that you want to delete it.

**NEW QUESTION 55**

You have multiple team members collaborating on infrastructure as code (IaC) using Terraform, and want to apply formatting standards for readability. How can you format Terraform HCL (HashiCorp Configuration Language) code according to standard Terraform style convention?

- A. Run the terraform fmt command during the code linting phase of your CI/CD process Most Voted
- B. Designate one person in each team to review and format everyone's code
- C. Manually apply two spaces indentation and align equal sign "=" characters in every Terraform file (\*.tf)
- D. Write a shell script to transform Terraform files using tools such as AWK, Python, and sed

**Answer:** A

**Explanation:**

The terraform fmt command is used to rewrite Terraform configuration files to a canonical format and style. This command applies a subset of the Terraform language style conventions, along with other minor adjustments for readability. Running this command on your configuration files before committing them to source control can help ensure consistency of style between different Terraform codebases, and can also make diffs easier to read. You can also use the -check and -diff options to check if the files are formatted and display the formatting changes respectively<sup>2</sup>. Running the terraform fmt command during the code linting phase of your CI/CD process can help automate this process and enforce the formatting standards for your team. References = [Command: fmt]<sup>2</sup>

**NEW QUESTION 60**

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 Terraform will delete. Which command should you use to show all of the resources that will be deleted? Choose two correct answers.

- A. Run terraform state rm ??
- B. Run terraform show :destroy
- C. Run terraform destroy and it will first output all the resource that will be deleted before prompting for approval
- D. Run terraform plan .destory

**Answer:** CD

**Explanation:**

To see all the resources that Terraform will delete, you can use either of these two commands:  
? terraform destroy will show the plan of destruction and ask for your confirmation before proceeding. You can cancel the command if you do not want to destroy the resources.  
? terraform plan -destroy will show the plan of destruction without asking for confirmation. You can use this command to review the changes before running terraform destroy. References = : Destroy Infrastructure : Plan Command: Options

**NEW QUESTION 65**

It is best practice to store secret data in the same version control repository as your Terraform configuration.

- A. True
- B. False

**Answer:** B

**Explanation:**

It is not a best practice to store secret data in the same version control repository as your Terraform configuration, as it could expose your sensitive information to unauthorized parties or compromise your security. You should use environment variables, vaults, or other mechanisms to store and provide secret data to Terraform.

**NEW QUESTION 68**

Which Terraform collection type should you use to store key/value pairs?

- A. Set
- B. Map
- C. Tuple
- D. list

**Answer:** B

**Explanation:**



The Terraform collection type that should be used to store key/value pairs is map. A map is a collection of values that are accessed by arbitrary labels, called keys.

The keys and values can be of any type, but the keys must be unique within a map. For example, `var = { key1 = "value1", key2 = "value2" }` is a map with two key/value pairs. Maps are useful for grouping related values together, such as configuration options or metadata. References = [Collection Types], [Map Type Constraints]

**NEW QUESTION 71**

Which task does terraform init not perform?

- A. Validates all required variables are present
- B. Sources any modules and copies the configuration locally
- C. Connects to the backend
- D. Sources all providers used in the configuration and downloads them

**Answer:** A

**Explanation:**

The terraform init command is used to initialize a working directory containing Terraform configuration files. This command performs several different initialization steps to prepare the current working directory for use with Terraform, which includes initializing the backend, installing provider plugins, and copying any modules referenced in the configuration. However, it does not validate whether all required variables are present; that is a task performed by terraform plan or terraform apply1.

References = This information can be verified from the official Terraform documentation on the terraform init command provided by HashiCorp Developer1.

**NEW QUESTION 72**

terraform validate reports syntax check errors for which of the following?

- A. Code contains tabs for indentation instead of spaces
- B. There is a missing value for a variable
- C. The state file does not match the current infrastructure
- D. None of the above

**Answer:** D

**Explanation:**

The terraform validate command is used to check for syntax errors and internal consistency within Terraform configurations, such as whether all required arguments are specified. It does not check for indentation styles, missing variable values (as variables might not be defined at validation time), or state file consistency with the current infrastructure. Therefore, none of the provided options are correct in the context of what terraform validate reports. References = Terraform's official documentation details the purpose and function of the terraform validate command, specifying that it focuses on syntax and consistency checks within Terraform configurations themselves, not on external factors like the state file or infrastructure state. Direct references from the HashiCorp Terraform Associate (003) study materials to this specific detail were not found in the provided files.

**NEW QUESTION 75**

Which backend does the Terraform CU use by default?

- A. Depends on the cloud provider configured
- B. HTTP
- C. Remote
- D. Terraform Cloud
- E. Local

**Answer:** E

**Explanation:**

This is the backend that the Terraform CLI uses by default, unless you specify a different backend in your configuration. The local backend stores the state file in a local file named terraform.tfstate, which can be used to track and manage the state of your infrastructure.

**NEW QUESTION 79**

Which type of block fetches or computes information for use elsewhere in a Terraform configuration?

- A. data
- B. local
- C. resource
- D. provider

**Answer:** A

**Explanation:**

In Terraform, a data block is used to fetch or compute information from external sources for use elsewhere in the Terraform configuration. Unlike resource blocks that manage infrastructure, data blocks gather information without directly managing any resources. This can include querying for data from cloud providers, external APIs, or other Terraform states. References = This definition and usage of data blocks are covered in Terraform's official documentation, highlighting their role in fetching external information to inform Terraform configurations.

**NEW QUESTION 84**

When should you run terraform init?

- A. Every time you run terraform apply
- B. Before you start coding a new Terraform project
- C. After you run terraform plan for the time in a new terraform project and before you run terraform apply

D. After you start coding a new terraform project and before you run terraform plan for the first time.

**Answer:** D

**Explanation:**

You should run terraform init after you start coding a new Terraform project and before you run terraform plan for the first time. This command will initialize the working directory by downloading the required providers and modules, creating the initial state file, and performing other necessary tasks. References = : Initialize a Terraform Project

**NEW QUESTION 86**

How can terraform plan aid in the development process?

- A. Initializes your working directory containing your Terraform configuration files
- B. Validates your expectations against the execution plan without permanently modifying state
- C. Formats your Terraform configuration files
- D. Reconciles Terraform's state against deployed resources and permanently modifies state using the current status of deployed resources

**Answer:** B

**Explanation:**

The terraform plan command is used to create an execution plan. It allows you to see what actions Terraform will take to reach the desired state defined in your configuration files. It evaluates the current state and configuration, showing a detailed outline of the resources that will be created, updated, or destroyed. This is a critical step in the development process as it helps you verify that the changes you are about to apply will perform as expected, without actually modifying any state or infrastructure.

References:

? Terraform documentation on terraform plan: Terraform Plan

**NEW QUESTION 90**

You have a list of numbers that represents the number of free CPU cores on each virtual cluster:



```
numcpus = [ 18, 3, 7, 11, 2 ]
```

What Terraform function could you use to select the largest number from the list?

- A. top(numcpus)
- B. max(numcpus)
- C. ceil (numcpus)
- D. high[numcpus]

**Answer:** B

**Explanation:**

In Terraform, the max function can be used to select the largest number from a list of numbers. The max function takes multiple arguments and returns the highest one. For the list numcpus = [18, 3, 7, 11, 2], using max(numcpus...) will return 18, which is the largest number in the list.

References:

? Terraform documentation on max function: Terraform Functions - max

**NEW QUESTION 95**

What does Terraform use the .terraform.lock.hcl file for?

- A. There is no such file
- B. Tracking specific provider dependencies
- C. Preventing Terraform runs from occurring
- D. Storing references to workspaces which are locked

**Answer:** B

**Explanation:**

The .terraform.lock.hcl file is a new feature in Terraform 0.14 that records the exact versions of each provider used in your configuration. This helps ensure consistent and reproducible behavior across different machines and runs.

**NEW QUESTION 98**

All modules published on the official Terraform Module Registry have been verified by HashiCorp.

- A. True
- B. False

**Answer:** B

**Explanation:**

Not all modules published on the official Terraform Module Registry have been verified by HashiCorp. While HashiCorp verifies some modules, there are many community-contributed modules that are not verified. Verified modules have a "Verified" badge indicating that HashiCorp has reviewed them for security and best practices, but the registry also includes unverified modules.

References:

? Terraform Module Registry documentation: Terraform Registry

**NEW QUESTION 101**

You have declared a variable called `var.list` which is a list of objects that all have an attribute `id`. Which options will produce a list of the IDs? Choose two correct answers.

- A. `[ var.list [ * ], id ]`
- B. `[ for o in var.list : o.id ]`
- C. `var.list[*].id`
- D. `{ for o in var.list : o => o.id }`

**Answer:** BC

**Explanation:**

These are two ways to produce a list of the IDs from a list of objects that have an attribute `id`, using either a `for` expression or a `splat` expression syntax.

**NEW QUESTION 105**

Which parameters does `terraform import` require? Choose two correct answers.

- A. Provider
- B. Resource ID
- C. Resource address
- D. Path

**Answer:** BC

**Explanation:**

These are the parameters that `terraform import` requires, as they allow Terraform to identify the existing resource that you want to import into your state file, and match it with the corresponding configuration block in your files.

**NEW QUESTION 110**

Which of the following is not a benefit of adopting infrastructure as code?

- A. Versioning
- B. A Graphical User Interface
- C. Reusability of code
- D. Automation

**Answer:** B

**Explanation:**

Infrastructure as Code (IaC) provides several benefits, including the ability to version control infrastructure, reuse code, and automate infrastructure management. However, IaC is typically associated with declarative configuration files and does not inherently provide a graphical user interface (GUI). A GUI is a feature that may be provided by specific tools or platforms built on top of IaC principles but is not a direct benefit of IaC itself<sup>1</sup>.  
References = The benefits of IaC can be verified from the official HashiCorp documentation on [What is Infrastructure as Code with Terraform](#)<sup>1</sup>.

**NEW QUESTION 113**

Which command add existing resources into Terraform state?

- A. `terraform init`
- B. `terraform plan`
- C. `terraform refresh`
- D. `terraform import`
- E. All of these

**Answer:** D

**Explanation:**

This is the command that can add existing resources into Terraform state, by matching them with the corresponding configuration blocks in your files.

**NEW QUESTION 117**

Any user can publish modules to the public Terraform Module Registry.

- A. True
- B. False

**Answer:** A

**Explanation:**

The Terraform Registry allows any user to publish and share modules. Published modules support versioning, automatically generate documentation, allow browsing version histories, show examples and READMEs, and more. Public modules are managed via Git and GitHub, and publishing a module takes only a few minutes. Once a module is published, releasing a new version of a module is as simple as pushing a properly formed Git tag<sup>1</sup>.  
References = The information can be verified from the Terraform Registry documentation on [Publishing Modules](#) provided by HashiCorp Developer<sup>1</sup>.

**NEW QUESTION 119**

Which are examples of infrastructure as code? Choose two correct answers.

- A. Cloned virtual machine images
- B. Versioned configuration files
- C. Change management database records
- D. Doctor files

**Answer:** B

**Explanation:**

These are examples of infrastructure as code (IaC), which is a practice of managing and provisioning infrastructure through machine-readable definition files, rather than physical hardware configuration or interactive configuration tools.

**NEW QUESTION 124**

Which of the following commands would you use to access all of the attributes and details of a resource managed by Terraform?

- A. terraform state list `??provider_type.name??`
- B. terraform state show `??provider_type.name??`
- C. terraform get `??provider_type.name??`
- D. terraform state list

**Answer:** B

**Explanation:**

The terraform state show command allows you to access all of the attributes and details of a resource managed by Terraform. You can use the resource address (e.g. provider\_type.name) as an argument to show the information about a specific resource. The terraform state list command only shows the list of resources in the state, not their attributes. The terraform get command downloads and installs modules needed for the configuration. It does not show any information about resources. References = [Command: state show] and [Command: state list]

**NEW QUESTION 127**

How do you specify a module's version when publishing it to the public terraform Module Registry?

- A. Configuration it in the module's Terraform code
- B. Mention it on the module's configuration page on the Terraform Module Registry
- C. The Terraform Module Registry does not support versioning modules
- D. Tag a release in the associated repo

**Answer:** D

**Explanation:**

This is how you specify a module's version when publishing it to the public Terraform Module Registry, as it uses the tags from your version control system (such as GitHub or GitLab) to identify module versions. You need to use semantic versioning for your tags, such as v1.0.0.

**NEW QUESTION 132**

Which command must you first run before performing further Terraform operations in a working directory?

- A. terraform import
- B. terraform workspace
- C. terraform plan
- D. terraform init

**Answer:** D

**Explanation:**

terraform init is the first command that should be run after writing a new Terraform configuration or cloning an existing one from version control. It initializes a working directory containing Terraform configuration files and downloads any required providers and modules. The other commands are used for different purposes, such as importing existing resources, switching between workspaces, generating execution plans, etc.

**NEW QUESTION 135**

.....

## Thank You for Trying Our Product

\* 100% Pass or Money Back

All our products come with a 90-day Money Back Guarantee.

\* One year free update

You can enjoy free update one year. 24x7 online support.

\* Trusted by Millions

We currently serve more than 30,000,000 customers.

\* Shop Securely

All transactions are protected by VeriSign!

**100% Pass Your Terraform-Associate-003 Exam with Our Prep Materials Via below:**

<https://www.certleader.com/Terraform-Associate-003-dumps.html>