

## Exam Questions TA-002-P

HashiCorp Certified: Terraform Associate

<https://www.2passeasy.com/dumps/TA-002-P/>



### NEW QUESTION 1

- (Exam Topic 1)

What features stops multiple admins from changing the Terraform state at the same time?

- A. Version control
- B. Backend types
- C. Provider constraints
- D. State locking

**Answer:** D

#### Explanation:

Somewhat ambiguous question however the key phrase is "feature". You need a remote backend first with a State Locking feature available to avoid this scenario.  
<https://blog.gruntwork.io/how-to-manage-terraform-state-28f5697e68fa>

### NEW QUESTION 2

- (Exam Topic 1)

What command should you run to display all workspaces for the current configuration?

- A. terraform workspace
- B. terraform workspace show
- C. terraform workspace list
- D. terraform show workspace

**Answer:** C

#### Explanation:

terraform workspace list

The command will list all existing workspaces.

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

### NEW QUESTION 3

- (Exam Topic 1)

What is the provider for this fictitious resource?

```
resource "aws_vpc" "main" {  
    name = "test"  
}
```

- A. vpc
- B. main
- C. aws
- D. test

**Answer:** C

#### Explanation:

Reference: <https://docs.aws.amazon.com/cloudformation-cli/latest/userguide/resource-types.html>

### NEW QUESTION 4

- (Exam Topic 1)

When you initialize Terraform, where does it cache modules from the public Terraform Module Registry?

- A. On disk in the /tmp directory
- B. In memory
- C. On disk in the .terraform sub-directory
- D. They are not cached

**Answer:** C

#### Explanation:

"A hidden .terraform directory, which Terraform uses to manage cached provider plugins and modules, record which workspace is currently active, and record the last known backend configuration in case it needs to migrate state on the next run. This directory is automatically managed by Terraform, and is created during initialization." <https://www.terraform.io/cli/init>

### NEW QUESTION 5

- (Exam Topic 1)

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?

- A. terraform apply rm aws\_instance.ubuntu[1]
- B. terraform state rm aws\_instance.ubuntu[1]
- C. terraform plan rm aws\_instance.ubuntu[1]
- D. terraform delete aws\_instance.ubuntu[1]

**Answer:** B

**Explanation:**

"You can use terraform state rm in the less common situation where you wish to remove a binding to an existing remote object without first destroying it, which will effectively make Terraform "forget" the object while it continues to exist in the remote system." <https://www.terraform.io/cli/commands/state/rm>

**NEW QUESTION 6**

- (Exam Topic 1)

You have a simple Terraform configuration containing one virtual machine (VM) in a cloud provider. You run terraform apply and the VM is created successfully. What will happen if you delete the VM using the cloud provider console, and run terraform apply again without changing any Terraform code?

- A. Terraform will remove the VM from state file
- B. Terraform will report an error
- C. Terraform will not make any changes
- D. Terraform will recreate the VM

**Answer:** D

**NEW QUESTION 7**

- (Exam Topic 1)

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

- A. split
- B. join
- C. slice
- D. chomp

**Answer:** C

**Explanation:**

<https://www.terraform.io/language/functions>

**NEW QUESTION 8**

- (Exam Topic 1)

When should you use the force-unlock command?

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

**Answer:** C

**Explanation:**

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. Source: <https://www.terraform.io/language/state/locking>  
<https://www.terraform.io/cli/commands/force-unlock>

**NEW QUESTION 9**

- (Exam Topic 1)

terraform validate validates the syntax of Terraform files.

- A. True
- B. False

**Answer:** A

**Explanation:**

<https://www.terraform.io/cli/commands/validate>

The terraform validate command validates the syntax and arguments of the Terraform configuration files. Reference: <https://www.terraform.io/docs/cli/code/index.html>

**NEW QUESTION 10**

- (Exam Topic 1)

What is the name assigned by Terraform to reference this resource?

```
resource "azurerm_resource_group" "dev" {  
  name = "test"  
  location = "westus"  
}
```

- A. dev
- B. azure\_rm\_resource\_group
- C. azure\_rm
- D. test

**Answer:** A

#### NEW QUESTION 10

- (Exam Topic 1)

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.)

- A. Run `terraform plan -destroy`.
- B. This is not possible.
- C. You can only show resources that will be created.
- D. Run `terraform state rm *`.
- E. Run `terraform destroy` and it will first output all the resources that will be deleted before prompting for approval.

**Answer:** AD

#### Explanation:

Reference: <https://www.terraform.io/docs/cli/commands/state/rm.html>

#### NEW QUESTION 15

- (Exam Topic 1)

Terraform can only manage resource dependencies if you set them explicitly with the `depends_on` argument.

- A. True
- B. False

**Answer:** A

#### Explanation:

"Use the `depends_on` meta-argument to handle hidden resource or module dependencies that Terraform cannot automatically infer. You only need to explicitly specify a dependency when a resource or module relies on another resource's behavior but does not access any of that resource's data in its arguments."  
[https://www.terraform.io/language/meta-arguments/depends\\_on](https://www.terraform.io/language/meta-arguments/depends_on)

#### NEW QUESTION 20

- (Exam Topic 1)

Which argument(s) is (are) required when declaring a Terraform variable?

- A. `type`
- B. `default`
- C. `description`
- D. All of the above
- E. None of the above

**Answer:** B

#### Explanation:

The variable declaration can also include a default argument.

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

#### NEW QUESTION 24

- (Exam Topic 1)

Which task does `terraform init` not perform?

- A. Sources all providers present in the configuration and ensures they are downloaded and available locally
- B. Connects to the backend
- C. Sources any modules and copies the configuration locally
- D. Validates all required variables are present

**Answer:** D

#### Explanation:

Reference: <https://www.terraform.io/docs/cli/commands/init.html>

#### NEW QUESTION 28

- (Exam Topic 1)

The `terraform.tfstate` file always matches your currently built infrastructure.

- A. True
- B. False

**Answer:** B

**Explanation:**

Reference: <https://www.terraform.io/docs/language/state/index.html>

**NEW QUESTION 33**

- (Exam Topic 1)

Which two steps are required to provision new infrastructure in the Terraform workflow? (Choose two.)

- A. Destroy
- B. Apply
- C. Import
- D. Init
- E. Validate

**Answer:** BD

**Explanation:**

Reference: <https://www.terraform.io/guides/core-workflow.html>

**NEW QUESTION 36**

- (Exam Topic 1)

A provider configuration block is required in every Terraform configuration. Example:

```
provider "provider_name" {  
    . . .  
}
```

- A. True
- B. False

**Answer:** B

**Explanation:**

Unlike many other objects in the Terraform language, a provider block may be omitted if its contents would otherwise be empty. Terraform assumes an empty default configuration for any provider that is not explicitly configured. <https://www.terraform.io/language/providers/configuration>

**NEW QUESTION 39**

- (Exam Topic 1)

terraform init initializes a sample main.tf file in the current directory.

- A. True
- B. False

**Answer:** B

**Explanation:**

Reference: <https://www.terraform.io/docs/cli/commands/init.html>

**NEW QUESTION 41**

- (Exam Topic 1)

You're building a CI/CD (continuous integration/ continuous delivery) pipeline and need to inject sensitive variables into your Terraform run. How can you do this safely?

- A. Pass variables to Terraform with a `--var` flag
- B. Copy the sensitive variables into your Terraform code
- C. Store the sensitive variables in a `secure_vars.tf` file
- D. Store the sensitive variables as plain text in a source code repository

**Answer:** A

**Explanation:**

<https://blog.gruntwork.io/a-comprehensive-guide-to-managing-secrets-in-your-terraform-code-1d586955ace1>

**NEW QUESTION 46**

- (Exam Topic 1)

Terraform provisioners that require authentication can use the \_\_\_\_\_ block.

- A. connection
- B. credentials
- C. secrets
- D. ssh

**Answer:** A

**Explanation:**

<https://www.terraform.io/language/resources/provisioners/connection>

"Most provisioners require access to the remote resource via SSH or WinRM and expect a nested connection block with details about how to connect."

"Connection blocks don't take a block label and can be nested within either a resource or a provisioner."

#### NEW QUESTION 50

- (Exam Topic 1)

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.)

- A. Provision new VMs using Terraform with the same VM names
- B. Use the terraform import command for the existing VMs
- C. Write Terraform configuration for the existing VMs
- D. Run the terraform import-gcp command

**Answer:** BC

#### Explanation:

You should create the equivalent configuration first, and then run import to load it on the state file.

#### NEW QUESTION 53

- (Exam Topic 1)

Where does the Terraform local backend store its state?

- A. In the /tmp directory
- B. In the terraform.tfvars file
- C. In the terraform.tfstate file
- D. In the user's .terraformrc file

**Answer:** C

#### Explanation:

<https://www.terraform.io/language/state>

The local backend stores state on the local filesystem, locks that state using system APIs, and performs operations locally.

Reference: <https://www.terraform.io/docs/language/settings/backends/local.html>

#### NEW QUESTION 54

- (Exam Topic 1)

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

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

**Answer:** B

#### Explanation:

Reference: <https://docs.microsoft.com/en-us/azure/devops/learn/what-is-infrastructure-as-code#:~:text=Idempotence%20is%20a%20principle%20of,of%20the%20environment's%20starting%20state.>

#### NEW QUESTION 55

- (Exam Topic 1)

Your security team scanned some Terraform workspaces and found secrets stored in a plaintext in state files. How can you protect sensitive data stored in Terraform state files?

- A. Delete the state file every time you run Terraform
- B. Store the state in an encrypted backend
- C. Edit your state file to scrub out the sensitive data
- D. Always store your secrets in a secrets.tfvars file.

**Answer:** B

#### NEW QUESTION 57

- (Exam Topic 1)

Terraform can import modules from a number of sources – which of the following is not a valid source?

- A. FTP server
- B. GitHub repository
- C. Local path
- D. Terraform Module Registry

**Answer:** A

#### Explanation:

<https://www.terraform.io/language/modules/sources>

#### NEW QUESTION 61

- (Exam Topic 1)

Terraform validate reports syntax check errors from which of the following scenarios?

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

**Answer: B**

**Explanation:**

The terraform validate command is used to validate the syntax of the terraform files. Terraform performs a syntax check on all the terraform files in the directory, and will display an error if any of the files doesn't validate. This command does not check formatting (e.g. tabs vs spaces, newlines, comments etc.). The following can be reported: invalid HCL syntax (e.g. missing trailing quote or equal sign) invalid HCL references (e.g. variable name or attribute which doesn't exist) same provider declared multiple times same module declared multiple times same resource declared multiple times invalid module name interpolation used in places where it's unsupported (e.g. variable, depends\_on, module.source, provider) missing value for a variable (none of -var foo=... flag, -var-file=foo.vars flag, TF\_VAR\_foo environment variable, terraform.tfvars, or default value in the configuration) <https://www.typeerror.org/docs/terraform/commands/validate> <https://learning-ocean.com/tutorials/terraform/terraform-validate>

**NEW QUESTION 65**

- (Exam Topic 1)

Module variable assignments are inherited from the parent module and do not need to be explicitly set.

- A. True
- B. False

**Answer: B**

**NEW QUESTION 66**

- (Exam Topic 1)

How would you reference the "name" value of the second instance of this fictitious resource?

```
resource "aws_instance" "web" {  
    count = 2  
    name = "terraform-${count.index}"  
}
```

- A. element(aws\_instance.web, 2)
- B. aws\_instance.web[1].name
- C. aws\_instance.web[1]
- D. aws\_instance.web[2].name
- E. aws\_instance.web.\*.name

**Answer: B**

**Explanation:**

<https://www.terraform.io/language/meta-arguments/count#referring-to-instances> Reference: <https://www.terraform.io/docs/configuration-0-11/interpolation.html>

**NEW QUESTION 67**

- (Exam Topic 1)

You need to constrain the GitHub provider to version 2.1 or greater.

Which of the following should you put into the Terraform 0.12 configuration's provider block?

- A. version >= 2.1
- B. version ~> 2.1
- C. version = "<= 2.1"
- D. version = ">= 2.1"

**Answer: D**

**Explanation:**

version = ">= 1.2.0, < 2.0.0"

A version constraint is a string literal containing one or more conditions, which are separated by commas. Each condition consists of an operator and a version number.

Version numbers should be a series of numbers separated by periods (like 1.2.0), optionally with a suffix to indicate a beta release.

The following operators are valid:

= (or no operator): Allows only one exact version number. Cannot be combined with other conditions.

!=: Excludes an exact version number.

>, >=, <, <=: Comparisons against a specified version, allowing versions for which the comparison is true. "Greater-than" requests newer versions, and "less-than" requests older versions.

~>: Allows only the rightmost version component to increment. For example, to allow new patch releases within a specific minor release, use the full version number: ~> 1.0.4 will allow installation of 1.0.5 and 1.0.10 but not 1.1.0. This is usually called the pessimistic constraint operator.

<https://www.terraform.io/language/expressions/version-constraints>

**NEW QUESTION 70**

- (Exam Topic 1)

You would like to reuse the same Terraform configuration for your development and production environments with a different state file for each. Which command would you use?

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

**Answer:** B

**Explanation:**

<https://www.terraform.io/language/state/workspaces#when-to-use-multiple-workspaces>

**NEW QUESTION 71**

- (Exam Topic 1)

You have declared an input variable called environment in your parent module. What must you do to pass the value to a child module in the configuration?

- A. Add node\_count = var.node\_count
- B. Declare the variable in a terraform.tfvars file
- C. Declare a node\_count input variable for child module
- D. Nothing, child modules inherit variables of parent module

**Answer:** C

**Explanation:**

"That module may call other modules and connect them together by passing output values from one to input values of another."

<https://www.terraform.io/language/modules/develop>

**NEW QUESTION 75**

- (Exam Topic 1)

If a module declares a variable with a default, that variable must also be defined within the module.

- A. True
- B. False

**Answer:** B

**NEW QUESTION 80**

- (Exam Topic 1)

When using a module block to reference a module stored on the public Terraform Module Registry such as:

```
module "consul" {  
    source = "hashicorp/consul/aws"  
}
```

How do you specify version 1.0.0?

- A. Modules stored on the public Terraform Module Registry do not support versioning
- B. Append ?ref=v1.0.0 argument to the source path
- C. Add version = "1.0.0" attribute to module block
- D. Nothing – modules stored on the public Terraform Module Registry always default to version 1.0.0

**Answer:** C

**Explanation:**

Version

When using modules installed from a module registry, we recommend explicitly constraining the acceptable version numbers to avoid unexpected or unwanted changes.

Use the version argument in the module block to specify versions: module "consul" {

source = "hashicorp/consul/aws" version = "0.0.5"

servers = 3

}

Reference: <https://www.terraform.io/docs/language/modules/sources.html>

**NEW QUESTION 82**

- (Exam Topic 1)

In Terraform 0.13 and above, outside of the required\_providers block, Terraform configurations always refer to providers by their local names.

- A. True
- B. False

**Answer:** A

**Explanation:**

Outside of the required\_providers block, Terraform configurations always refer to providers by their local names.

Reference: <https://www.terraform.io/docs/language/providers/requirements.html> <https://www.terraform.io/language/providers/requirements#local-names>

#### NEW QUESTION 86

- (Exam Topic 1)

Terraform and Terraform providers must use the same major version number in a single configuration.

- A. True
- B. False

**Answer:** B

#### Explanation:

<https://www.terraform.io/language/expressions/version-constraints#terraform-core-and-provider-versions>

#### NEW QUESTION 88

- (Exam Topic 1)

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. Run terraform output ip\_address to view the result
- B. 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
- C. Run terraform state list to find the name of the resource, then terraform state show to find the attributes including public IP address
- D. Run terraform destroy then terraform apply and look for the IP address in stdout

**Answer:** C

#### Explanation:

<https://www.terraform.io/cli/commands/state/show>

#### NEW QUESTION 92

- (Exam Topic 1)

What is terraform refresh intended to detect?

- A. Terraform configuration code changes
- B. Empty state files
- C. State file drift
- D. Corrupt state files

**Answer:** C

#### Explanation:

"The terraform refresh command reads the current settings from all managed remote objects and updates the Terraform state to match. Warning: This command is deprecated, because its default behavior is unsafe if you have misconfigured credentials for any of your providers. See below for more information and recommended alternatives." <https://www.terraform.io/cli/commands/refresh>

#### NEW QUESTION 95

- (Exam Topic 1)

Which of the following is not true of Terraform providers?

- A. Providers can be written by individuals
- B. Providers can be maintained by a community of users
- C. Some providers are maintained by HashiCorp
- D. Major cloud vendors and non-cloud vendors can write, maintain, or collaborate on Terraform providers
- E. None of the above

**Answer:** E

#### Explanation:

<https://registry.terraform.io/providers/hashicorp/google/latest> - This provider is collaboratively maintained by the Google Terraform Team at Google and the Terraform team at HashiCorp

<https://www.terraform.io/language/providers>

#### NEW QUESTION 99

- (Exam Topic 1)

Which of the following is not an action performed by terraform init?

- A. Create a sample main.tf file
- B. Initialize a configured backend
- C. Retrieve the source code for all referenced modules
- D. Load required provider plugins

**Answer:** A

#### NEW QUESTION 104

- (Exam Topic 2)

Which one of the following command will rewrite Terraform configuration files to a canonical format and style.

- A. terraform graph -h
- B. terraform init

- C. terraform graph
- D. terraform fmt

**Answer:** D

**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.

**NEW QUESTION 105**

- (Exam Topic 2)

When TF\_LOG\_PATH is set, TF\_LOG must be set in order for any logging to be enabled.

- A. False
- B. True

**Answer:** B

**Explanation:**

TF\_LOG\_PATH specifies where the log should persist its output to. Note that even when TF\_LOG\_PATH is set, TF\_LOG must be set in order for any logging to be enabled.

For example, to always write the log to the directory you're currently running terraform from: export TF\_LOG\_PATH=./terraform.log  
export TF\_LOG=TRACE

**NEW QUESTION 109**

- (Exam Topic 2)

What is the standard workflow that a developer follows while working with terraform open source version?

- A. Run terraform refresh to update the terraform state , then write the terraform code , and finally run terraform apply.
- B. Run terraform destroy first since you need to start from fresh every time , before running terraform apply.
- C. Write terraform code , and run terraform push , to update the terraform state to the remote repo , which in turn will take care of the next steps.
- D. Write the terraform code on the developer machine , run terraform plan to check the changes , and run terraform apply to provision the infra.

**Answer:** D

**Explanation:**

You do not need to run terraform refresh as terraform plan implicitly will run terraform refresh. <https://www.terraform.io/guides/core-workflow.html>

**NEW QUESTION 112**

- (Exam Topic 2)

In regards to deploying resources in multi-cloud environments, what are some of the benefits of using Terraform rather than a provider's native tooling? (select three)

- A. Terraform can help businesses deploy applications on multiple clouds and on-premises infrastructure.
- B. Terraform is not cloud-agnostic and can be used to deploy resources across a single public cloud.
- C. Terraform simplifies management and orchestration, helping operators build large-scale, multi-cloud infrastructure.
- D. Terraform can manage cross-cloud dependencies.

**Answer:** ACD

**Explanation:**

Terraform is cloud-agnostic and allows a single configuration to be used to manage multiple providers, and to even handle cross-cloud dependencies. This simplifies management and orchestration, helping operators build large-scale multi-cloud infrastructures.

<https://www.terraform.io/intro/use-cases.html>

**NEW QUESTION 116**

- (Exam Topic 2)

Provisioners should only be used as a last resort.

- A. False
- B. True

**Answer:** B

**Explanation:**

Provisioners are a Last Resort

Terraform includes the concept of provisioners as a measure of pragmatism, knowing that there will always be certain behaviors that can't be directly represented in Terraform's declarative model.

However, they also add a considerable amount of complexity and uncertainty to Terraform usage. Firstly, Terraform cannot model the actions of provisioners as part of a plan because they can in principle take any action. Secondly, successful use of provisioners requires coordinating many more details than Terraform usage usually requires: direct network access to your servers, issuing Terraform credentials to log in, making sure that all of the necessary external software is installed, etc.

The following sections describe some situations which can be solved with provisioners in principle, but where better solutions are also available. We do not recommend using provisioners for any of the use-cases described in the following sections.

Even if your specific use-case is not described in the following sections, we still recommend attempting to solve it using other techniques first, and use provisioners only if there is no other option.

<https://www.terraform.io/docs/provisioners/index.html>

#### NEW QUESTION 120

- (Exam Topic 2)

You want terraform plan and apply to be executed in Terraform Cloud's run environment but the output is to be streamed locally. Which one of the below you will choose?

- A. Local Backends
- B. This can be done using any of the local or remote backends
- C. Remote Backends
- D. Terraform Backends

**Answer:** C

#### Explanation:

The remote backend stores Terraform state and may be used to run operations in Terraform Cloud. When using full remote operations, operations like terraform plan or terraform apply can be executed in

Terraform Cloud's run environment, with log output streaming to the local terminal.

Remote plans and applies use variable values from the associated Terraform Cloud workspace. <https://www.terraform.io/docs/backends/types/remote.html>

#### NEW QUESTION 125

- (Exam Topic 2)

Which one of the following will run echo 0 and echo 1 on a newly created host?

- A. provisioner "local-exec" { command = "echo 0" command = "echo 1" }
- B. provisioner "remote-exec" { inline = [echo 0,echo 1]}
- C. provisioner "remote-exec" {command = "\${echo 0}" command = "\${echo 1}"}
- D. provisioner "remote-exec" { inline = ["echo 0","echo 1"]}

**Answer:** D

#### Explanation:

remote-exec Provisioner Example usage

```
resource "aws_instance" "web" {
```

```
# ...
```

```
provisioner "remote-exec" { inline = [
```

```
"puppet apply",
```

```
"consul join ${aws_instance.web.private_ip}",
```

```
]
```

```
}
```

```
}
```

#### NEW QUESTION 126

- (Exam Topic 2)

Which of the following clouds does not have a provider maintained HashiCorp?

- A. IBM Cloud
- B. DigitalOcean
- C. OpenStack
- D. AWS

**Answer:** A

#### Explanation:

IBM Cloud does not have a provider maintained by HashiCorp, although IBM Cloud does maintain their own Terraform provider.

<https://www.terraform.io/docs/providers/index.html>

#### NEW QUESTION 128

- (Exam Topic 2)

The terraform init 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.

- A. False
- B. True

**Answer:** B

#### Explanation:

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

#### NEW QUESTION 129

- (Exam Topic 2)

Which Terraform command will force a marked resource to be destroyed and recreated on the next apply?

- A. terraform fmt
- B. terraform destroy
- C. terraform taint
- D. terraform refresh

**Answer:** C

**Explanation:**

The terraform taint command manually marks a Terraform-managed resource as tainted, forcing it to be destroyed and recreated on the next apply.

This command will not modify infrastructure, but does modify the state file in order to mark a resource as tainted. Once a resource is marked as tainted, the next plan will show that the resource will be destroyed and recreated and the next apply will implement this change.

Forcing the recreation of a resource is useful when you want a certain side effect of recreation that is not visible in the attributes of a resource. For example: re-running provisioners will cause the node to be different or rebooting the machine from a base image will cause new startup scripts to run.

Note that tainting a resource for recreation may affect resources that depend on the newly tainted resource. For example, a DNS resource that uses the IP address of a server may need to be modified to reflect the potentially new IP address of a tainted server. The plan command will show this if this is the case.

<https://www.terraform.io/docs/commands/taint.html>

**NEW QUESTION 131**

- (Exam Topic 2)

Which of the following Terraform files should be ignored by Git when committing code to a repo? (select Three)

- A. Files named exactly terraform.tfvars or terraform.tfvars.json.
- B. Any files with names ending in .auto.tfvars or .auto.tfvars.json.
- C. input.tf
- D. terraform.tfstate
- E. output.tf

**Answer:** ABD

**Explanation:**

The .gitignore file should be configured to ignore Terraform files that either contain sensitive data or are not required to save.

Terraform state (terraform.tfstate) can contain sensitive data, depending on the resources in use and your definition of "sensitive." The state contains resource IDs and all resource attributes. For resources such as databases, this may contain initial passwords.

When using local state, state is stored in plain-text JSON files.

The terraform.tfvars file may contain sensitive data, such as passwords or IP addresses of an environment that you may not want to share with others.

**NEW QUESTION 135**

- (Exam Topic 2)

Workspaces in Terraform provides similar functionality in the open-source, Terraform Cloud, and Enterprise versions of Terraform.

- A. True
- B. False

**Answer:** B

**Explanation:**

<https://www.terraform.io/docs/cloud/migrate/workspaces.html>

Workspaces, managed with the terraform workspace command, aren't the same thing as Terraform Cloud's workspaces. Terraform Cloud workspaces act more like completely separate working directories; CLI workspaces are just alternate state files.

**NEW QUESTION 136**

- (Exam Topic 2)

terraform refresh command will not modify infrastructure, but does modify the state file.

- A. True
- B. False

**Answer:** A

**Explanation:**

The terraform refresh command is used to reconcile the state Terraform knows about (via its state file) with the real-world infrastructure. This can be used to detect any drift from the last-known state, and to update the state file. This does not modify infrastructure, but does modify the state file.

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

**NEW QUESTION 140**

- (Exam Topic 2)

terraform refresh will update the state file?

- A. True
- B. False

**Answer:** A

**Explanation:**

The terraform refresh command is used to reconcile the state Terraform knows about (via its state file) with the real-world infrastructure. This can be used to detect any drift from the last-known state, and to update the state file.

This does not modify infrastructure, but does modify the state file. If the state is changed, this may cause changes to occur during the next plan or apply.

**NEW QUESTION 141**

- (Exam Topic 2)

You have declared a variable name my\_var in terraform configuration without a value associated with it. variable my\_var {}

After running terraform plan it will show an error as variable is not defined.

- A. True

B. False

**Answer:** B

**Explanation:**

Input variables are usually defined by stating a name, type and a default value. However, the type and default values are not strictly necessary. Terraform can deduct the type of the variable from the default or input value.

Variables can be predetermined in a file or included in the command-line options. As such, the simplest variable is just a name while the type and value are selected based on the input.

```
variable "variable_name" {}
```

```
terraform apply -var variable_name="value"
```

The input variables, like the one above, use a couple of different types: strings, lists, maps, and boolean. Here are some examples of how each type are defined and used.

String

Strings mark a single value per structure and are commonly used to simplify and make complicated values more user-friendly. Below is an example of a string variable definition.

```
variable "template" { type = string
```

```
default = "01000000-0000-4000-8000-000030080200"
```

```
}
```

A string variable can then be used in resource plans. Surrounded by double quotes, string variables are a simple substitution such as the example underneath.

```
storage = var.template List
```

Another type of Terraform variables lists. They work much like a numbered catalogue of values. Each value can be called by their corresponding index in the list.

Here is an example of a list variable definition.

```
variable "users" { type = list
```

```
default = ["root", "user1", "user2"]
```

```
}
```

Lists can be used in the resource plans similarly to strings, but you'll also need to denote the index of the value you are looking for.

```
username = var.users[0] Map
```

Maps are a collection of string keys and string values. These can be useful for selecting values based on predefined parameters such as the server configuration by the monthly price.

```
variable "plans" { type = map default = {
```

```
"5USD" = "1xCPU-1GB" "10USD" = "1xCPU-2GB" "20USD" = "2xCPU-4GB"
```

```
}
```

```
}
```

You can access the right value by using the matching key. For example, the variable below would set the plan to "1xCPU-1GB".

```
plan = var.plans["5USD"]
```

The values matching to their keys can also be used to look up information in other maps. For example, underneath is a shortlist of plans and their corresponding storage sizes.

```
variable "storage_sizes" { type = map
```

```
default = {
```

```
"1xCPU-1GB" = "25"
```

```
"1xCPU-2GB" = "50"
```

```
"2xCPU-4GB" = "80"
```

```
}
```

```
}
```

These can then be used to find the right storage size based on the monthly price as defined in the previous example.

```
size = lookup(var.storage_sizes, var.plans["5USD"])
```

Boolean

The last of the available variable type is boolean. They give the option to employ simple true or false values. For example, you might wish to have a variable that decides when to generate the root user password on a new deployment.

```
variable "set_password" { default = false
```

```
}
```

The above example boolean can be used similarly to a string variable by simply marking down the correct variable.

```
create_password = var.set_password
```

By default, the value is set to false in this example. However, you can overwrite the variable at deployment by assigning a different value in a command-line variable.

```
terraform apply -var set_password="true"
```

**NEW QUESTION 143**

- (Exam Topic 2)

You want to get involved in the development of Terraform. As this is an open source project, you would like to contribute a fix for an open issue of Terraform. What programming language will need to use to write the fix?

A. It depends on which command issue related to.

B. Python

C. Go

D. Java

**Answer:** C

**Explanation:**

Basic programming knowledge. Terraform and Terraform Plugins are written in the Go programming language, but even if you've never written a line of Go before, you're still welcome to take a dive into the code and submit patches. The community is happy to assist with code reviews and offer guidance specific to Go.

**NEW QUESTION 148**

- (Exam Topic 2)

What allows you to conveniently switch between multiple instances of a single configuration within its single backend?

A. Local backends

B. Providers

C. Remote backends

D. Workspaces

**Answer:** D

**Explanation:**

Named workspaces allow conveniently switching between multiple instances of a single configuration within its single backend. ... A common use for multiple workspaces is to create a parallel, distinct copy of a set of infrastructure in order to test a set of changes before modifying the main production infrastructure. Workspaces, allowing multiple states to be associated with a single configuration. The configuration still has only one backend, but multiple distinct instances of that configuration to be deployed without configuring a new backend or changing authentication credentials.  
<https://www.terraform.io/docs/state/workspaces.html>

**NEW QUESTION 151**

- (Exam Topic 2)

You are using a terraform operation that writes state. Unfortunately automatic state unlocking has failed for that operation. Which of the below commands can be used to remove the already acquired lock on the state?

- A. terraform unlock
- B. terraform force-unlock
- C. terraform state unlock
- D. None of the above

**Answer:** B

**Explanation:**

Command: force-unlock

Manually unlock the state for the defined configuration.

This will not modify your infrastructure. This command removes the lock on the state for the current configuration. The behavior of this lock is dependent on the backend being used. Local state files cannot be unlocked by another process.

<https://www.terraform.io/docs/commands/force-unlock.html> <https://www.terraform.io/docs/state/locking.html>

Terraform has a force-unlock command to manually unlock the state if unlocking failed.

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.

**NEW QUESTION 154**

- (Exam Topic 2)

How does Terraform handle working with so many providers?

- A. Terraform ships with all of the plugins embedded in the Terraform binary.
- B. Terraform uses a plugin architecture for providers and only installs the provider plugins required by your configuration in the configuration's working directory.
- C. Terraform uses a plugin architecture for providers and only installs the provider plugins required by your configuration in a shared, system-wide plugins directory.
- D. Terraform allows you to select the providers you want to support during the Terraform installation process.

**Answer:** B

**Explanation:**

Terraform is built on a plugin-based architecture. All providers and provisioners that are used in Terraform configurations are plugins, even the core types such as AWS and Heroku. Users of Terraform are able to write new plugins in order to support new functionality in Terraform.

**NEW QUESTION 159**

- (Exam Topic 2)

Which of the following best describes a Terraform provider?

- A. A plugin that Terraform uses to translate the API interactions with the service or provider.
- B. Serves as a parameter for a Terraform module that allows a module to be customized.
- C. Describes an infrastructure object, such as a virtual network, compute instance, or other components.
- D. A container for multiple resources that are used together.

**Answer:** A

**Explanation:**

A provider is responsible for understanding API interactions and exposing resources. Providers generally are an IaaS (e.g. Alibaba Cloud, AWS, GCP, Microsoft Azure, OpenStack), PaaS (e.g. Heroku), or SaaS services (e.g. Terraform Cloud, DNSimple, Cloudflare).

<https://www.terraform.io/docs/providers/index.html>

**NEW QUESTION 162**

- (Exam Topic 2)

What is the command you can use to set an environment variable named "var1" of type String?

- A. export TF\_VAR\_VAR1
- B. set TF\_VAR\_var1
- C. variable "var1" { type = "string" }
- D. export TF\_VAR\_var1

**Answer:** D

**Explanation:**

The environment variable must be in the format TF\_VAR\_name, so for the QUESTION NO: TF\_VAR\_var1 is the correct choice.

[https://www.terraform.io/docs/commands/environment-variables.html#tf\\_var\\_name](https://www.terraform.io/docs/commands/environment-variables.html#tf_var_name)

#### NEW QUESTION 166

- (Exam Topic 2)

Which of the following best describes the default local backend?

- A. The local backend is where Terraform Enterprise stores logs to be processed by an log collector.
- B. The local backend stores state on the local filesystem, locks the state using system APIs, and performs operations locally.
- C. The local backend is the directory where resources deployed by Terraform have direct access to in order to update their current state.
- D. The local backend is how Terraform connects to public cloud services, such as AWS, Azure, or GCP.

**Answer: B**

#### Explanation:

The local backend stores state on the local filesystem, locks that state using system APIs, and performs operations locally.

```
terraform {  
  backend "local" {  
    path = "relative/path/to/terraform.tfstate"  
  }  
}
```

<https://www.terraform.io/docs/backends/types/local.html>

#### NEW QUESTION 170

- (Exam Topic 2)

What is the purpose of using the local-exec provisioner? (Select Two)

- A. To invoke a local executable.
- B. Executes a command on the resource to invoke an update to the Terraform state.
- C. To execute one or more commands on the machine running Terraform.
- D. Ensures that the resource is only executed in the local infrastructure where Terraform is deployed.

**Answer: AC**

#### Explanation:

The local-exec provisioner invokes a local executable after a resource is created. This invokes a process on the machine running Terraform, not on the resource. Note that even though the resource will be fully created when the provisioner is run, there is no guarantee that it will be in an operable state - for example system services such as sshd may not be started yet on compute resources.

Example usage

```
resource "aws_instance" "web" {  
  # ...  
  provisioner "local-exec" {  
    command = "echo ${aws_instance.web.private_ip} >> private_ips.txt"  
  }  
}
```

Note: Provisioners should only be used as a last resort. For most common situations there are better alternatives.

<https://www.terraform.io/docs/provisioners/local-exec.html>

#### NEW QUESTION 174

- (Exam Topic 2)

Terraform must track metadata such as resource dependencies. Where is this data stored?

- A. workspace
- B. backend
- C. state file
- D. metadata store

**Answer: C**

#### Explanation:

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>

#### NEW QUESTION 178

- (Exam Topic 3)

Which of the below options is the equivalent Terraform 0.12 version of the snippet which is written in Terraform 0.11?

"\${var.instance\_id}"

- A. variable.instance\_id
- B. var.instance\_ids
- C. var.instance\_id
- D. None of the above

**Answer: C**

#### NEW QUESTION 179

- (Exam Topic 3)

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?

- A. dynamic backend
- B. splat expression
- C. terraform import
- D. dynamic block

**Answer:** D

**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>

**NEW QUESTION 181**

- (Exam Topic 3)

A colleague has informed you that a new version of a Terraform module that your team hosts on an Amazon S3 bucket is broken. The Amazon S3 bucket has versioning enabled. Your colleague tells you to make sure you are not using the latest version in your configuration. You have the following configuration block in your code that refers to the module:

module "infranet" { source = "s3::https://s3-us-west-2.amazonaws.com/infrabucket/infra\_module.zip"} What is the best way to ensure that you are not using the latest version of the module?

- A. Add a module version constraint in your configuration's backend block and specify a previous version.
- B. Add a version key to the module configuration and specify a previous version.
- C. Delete the latest version of the module in S3 to rollback to the previous version.
- D. Add a version property to the module in Terraform's state file and specify a previous version.

**Answer:** C

**Explanation:**

Version constraints are supported only for modules installed from a module registry, such as the Terraform Registry or Terraform Cloud's private module registry. Other module sources can provide their own versioning mechanisms within the source string itself, or might not support versions at all. In particular, modules sourced from local file paths do not support version; since they're loaded from the same source repository.

Only Terraform Registries support module versioning by using the version key, one cannot configure a previous version of the module in the configuration. Deleting the latest version of the module in S3 is the only option of the available options that ensures you won't use the latest version. You could also modify the source URL to specify a versionId URL parameter for a previous version.

<https://www.terraform.io/docs/configuration/modules.html#source>

**NEW QUESTION 185**

- (Exam Topic 3)

Eric needs to make use of module within his terraform code. Should the module always be public and open-source to be able to be used?

- A. False
- B. True

**Answer:** A

**Explanation:**

Terraform module need not be public and open-source. Module can be placed in

- \* Local paths
- \* Terraform Registry
- \* GitHub
- \* Bitbucket
- \* Generic Git, Mercurial repositories
- \* HTTP URLs
- \* S3 buckets
- \* GCS buckets <https://www.terraform.io/docs/modules/sources.html>

**NEW QUESTION 186**

- (Exam Topic 3)

Which of the following challenges would Terraform be a candidate for solving? (Select THREE)

- A. Enable self-service infrastructure to allocate resources on your proprietary private cloud.
- B. Reduce the number of workflows needed for managing infrastructure across each of the companies public and private clouds.
- C. Utilize a single tool for all of the infrastructure and configuration management needs.
- D. Have a single interoperable tool to manage the variety of services including GitHub repositories, MySQL database, and Kubernetes clusters.

**Answer:** ABD

**NEW QUESTION 189**

- (Exam Topic 3)

You have created two workspaces PROD and DEV. You have switched to DEV and provisioned DEV infrastructure from this workspace. Where is your state file stored?

- A. terraform.d
- B. terraform.tfstate
- C. terraform.tfstate.DEV

D. terraform.tfstate.d

**Answer:** D

**Explanation:**

Terraform stores the workspace states in a directory called terraform.tfstate.d. This directory should be treated similarly to default workspace state file terraform.tfstate main.tf provider.tf terraform.tfstate.d DEV terraform.tfstate # DEV workspace state file PROD terraform.tfstate # PROD workspace state file terraform.tfvars # Default workspace state file variables.tf

**NEW QUESTION 191**

- (Exam Topic 3)

Forcing the recreation of a resource is useful when you want a certain side effect of recreation that is not visible in the attributes of a resource. What command will do this?

- A. terraform taint
- B. terraform apply
- C. terraform graph
- D. terraform refresh

**Answer:** A

**Explanation:**

The terraform taint command manually marks a Terraform-managed resource as tainted, forcing it to be destroyed and recreated on the next apply. This command will not modify infrastructure, but does modify the state file in order to mark a resource as tainted. Once a resource is marked as tainted, the next plan will show that the resource will be destroyed and recreated and the next apply will implement this change. Forcing the recreation of a resource is useful when you want a certain side effect of recreation that is not visible in the attributes of a resource. For example: re-running provisioners will cause the node to be different or rebooting the machine from a base image will cause new startup scripts to run. Note that tainting a resource for recreation may affect resources that depend on the newly tainted resource. For example, a DNS resource that uses the IP address of a server may need to be modified to reflect the potentially new IP address of a tainted server. The plan command will show this if this is the case. This example will taint a single resource:  
\$ terraform taint aws\_security\_group.allow\_all  
The resource aws\_security\_group.allow\_all in the module root has been marked as tainted. <https://www.terraform.io/docs/commands/taint.html>

**NEW QUESTION 196**

- (Exam Topic 3)

Which of the following value will be accepted for var1? variable "var1" {  
type = string  
}

- A. None of the above
- B. Both A and B
- C. "5"
- D. 5

**Answer:** B

**Explanation:**

Terraform automatically converts number and bool values to strings when needed.

**NEW QUESTION 198**

- (Exam Topic 3)

Your company has been using Terraform Cloud for a some time now . But every team is creating their own modules , and there is no standardization of the modules , with each team creating the resources in their own unique way . You want to enforce a standardization of the modules across the enterprise . What should be your approach.

- A. Create individual workspaces for each team , and ask them to share modules across workspaces.
- B. Implement a Private module registry in Terraform cloud , and ask teams to reference them.
- C. Upgrade to Terraform enterprise , since this is not possible in terraform cloud.
- D. Upload the modules in the terraform public module registry , and ask teams to reference them

**Answer:** B

**Explanation:**

Terraform Cloud's private module registry helps you share Terraform modules across your organization. It includes support for module versioning, a searchable and filterable list of available modules, and a configuration designer to help you build new workspaces faster. By design, the private module registry works much like the public Terraform Registry. If you're already used the public registry, Terraform Cloud's registry will feel familiar. Understand the different offerings in Terraform OS, Terraform Cloud and Terraform Enterprise. Terraform Cloud's private module registry helps you share Terraform modules across your organization.  
<https://www.terraform.io/docs/cloud/registry/index.html> <https://www.terraform.io/docs/cloud/registry/publish.html>

**NEW QUESTION 203**

- (Exam Topic 3)

Which of the below features of Terraform can be used for managing small differences between different environments which can act more like completely separate working directories.

- A. Repositories

- B. Workspaces
- C. Environment Variables
- D. Backends

**Answer:** B

**Explanation:**

workspaces allow conveniently switching between multiple instances of a single configuration within its single backend. They are convenient in a number of situations, but cannot solve all problems.

A common use for multiple workspaces is to create a parallel, distinct copy of a set of infrastructure in order to test a set of changes before modifying the main production infrastructure. For example, a developer working on a complex set of infrastructure changes might create a new temporary workspace in order to freely experiment with changes without affecting the default workspace.

Non-default workspaces are often related to feature branches in version control. The default workspace might correspond to the "master" or "trunk" branch, which describes the intended state of production infrastructure. When a feature branch is created to develop a change, the developer of that feature might create a corresponding workspace and deploy into it a temporary "copy" of the main infrastructure so that changes can be tested without affecting the production infrastructure. Once the change is merged and deployed to the default workspace, the test infrastructure can be destroyed and the temporary workspace deleted.

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

**NEW QUESTION 207**

- (Exam Topic 3)

Which of the following allows Terraform users to apply policy as code to enforce standardized configurations for resources being deployed via infrastructure as code?

- A. Sentinel
- B. Module registry
- C. Functions
- D. Workspaces

**Answer:** A

**Explanation:**

Sentinel is a language and framework for policy built to be embedded in existing software to enable fine-grained, logic-based policy decisions. A policy describes under what circumstances certain behaviors are allowed. Sentinel is an enterprise-only feature.

[https://www.youtube.com/watch?v=Vy8s7AAvU6g&feature=emb\\_title](https://www.youtube.com/watch?v=Vy8s7AAvU6g&feature=emb_title)

**NEW QUESTION 208**

- (Exam Topic 3)

You have multiple developers working on a terraform project (using terraform OSS), and have saved the terraform state in a remote S3 bucket . However ,team is intermittently experiencing inconsistencies in the provisioned infrastructure / failure in the code . You have traced this problem to simultaneous/concurrent runs of terraform apply command for 2/more developers . What can you do to fix this problem?

- A. Use terraform workspaces feature, this will fix this problem by default , as every developer will have their own state file , and terraform will merge them on server side on its own.
- B. Structure your team in such a way that only one individual will run terraform apply , everyone will just make changes and share with hi
- C. Then there will be no chance of any inconsistencies.
- D. Stop using remote state , and store the developer tfstate in their own machine . Once a day , all developers should sit together and merge the state files manually , to avoid any inconsistencies.
- E. Enable terraform state locking for the S3 backend using DynamoDB tabl
- F. This prevents others from acquiring the lock and potentially corrupting your state.

**Answer:** D

**Explanation:**

S3 backend support state locking using DynamoDB. <https://www.terraform.io/docs/state/locking.html>

**NEW QUESTION 210**

- (Exam Topic 3)

Ric wants to enable detail logging and he wants highest verbosity of logs. Which of the following environment variable settings is correct option for him to select.

- A. Set TF\_LOG = DEBUG
- B. Set VAR\_TF = TRACE
- C. Set TF\_LOG = TRACE
- D. Set VAR\_TF\_LOG = TRACE

**Answer:** C

**Explanation:**

<https://www.terraform.io/docs/internals/debugging.html>

**NEW QUESTION 213**

- (Exam Topic 3)

Terraform Cloud always encrypts state at rest and protects it with TLS in transit. Terraform Cloud also knows the identity of the user requesting state and maintains a history of state changes.

- A. False
- B. True

**Answer:** B

**Explanation:**

Terraform Cloud always encrypts state at rest and protects it with TLS in transit. Terraform Cloud also knows the identity of the user requesting state and maintains a history of state changes. This can be used to control access and track activity. Terraform Enterprise also supports detailed audit logging.  
<https://www.terraform.io/docs/state/sensitive-data.html#recommendations>

**NEW QUESTION 216**

- (Exam Topic 3)

What happens when a terraform apply command is executed?

- A. Creates the execution plan for the deployment of resources.
- B. Applies the changes required in the target infrastructure in order to reach the desired configuration.
- C. The backend is initialized and the working directory is prepped.
- D. Reconciles the state Terraform knows about with the real-world infrastructure.

**Answer:** B

**Explanation:**

The terraform apply command is used to apply the changes required to reach the desired state of the configuration, or the pre-determined set of actions generated by a terraform plan execution plan.  
<https://www.terraform.io/docs/commands/apply.html>

**NEW QUESTION 219**

- (Exam Topic 3)

When multiple engineers start deploying infrastructure using the same state file, what is a feature of remote state storage that is critical to ensure the state doesn't become corrupt?

- A. Object Storage
- B. State Locking
- C. WorkSpaces
- D. Encryption

**Answer:** B

**Explanation:**

If supported by your backend, Terraform will lock your state for all operations that could write state. This prevents others from acquiring the lock and potentially corrupting your state.  
State locking happens automatically on all operations that could write state. 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.  
Not all backends support locking. Please view the list of backend types for details on whether a backend supports locking or not.  
<https://www.terraform.io/docs/state/locking.html>

**NEW QUESTION 222**

- (Exam Topic 3)

A data block requests that Terraform read from a given data source and export the result under the given local name.

- A. False
- B. True

**Answer:** B

**NEW QUESTION 224**

- (Exam Topic 3)

Taint the resource "aws\_instance" "baz" resource that lives in module bar which lives in module foo.

- A. terraform taint module.foo.module.bar.baz
- B. terraform taint module.foo.bar.aws\_instance.baz
- C. terraform taint module.foo.module.bar.aws\_instance.baz
- D. terraform taint foo.bar.aws\_instance.baz

**Answer:** C

**Explanation:**

Check resource addressing <https://www.terraform.io/docs/internals/resource-addressing.html>

**NEW QUESTION 226**

- (Exam Topic 3)

Hanah is writing a terraform configuration with nested modules, there are multiple places where she has to use the same conditional expression but she wants to avoid repeating the same values or expressions multiple times in the configuration,. What is a better approach to dealing with this?

- A. Expressions
- B. Local Values
- C. Variables
- D. Functions

**Answer:** B

**Explanation:**

<https://www.terraform.io/docs/configuration/locals.html>

**NEW QUESTION 229**

- (Exam Topic 3)

You have created a terraform script that uses a lot of new constructs that have been introduced in terraform v0.12. However, many developers who are cloning the script from your git repo, are using v0.11, and getting errors. What can be done from your end to solve this problem?

- A. Force developer to use v0.12 by using terraform setting 'required\_version' and set it to >=0.12.
- B. Refactor the code to support both v0.11, and v0.12. It might be a difficult process, but there is no other way.
- C. Add a condition in front of each such specific construct, to check whether the running terraform version id v0.11 or v0.12, and ,work accordingly.
- D. Add comments in your code to tell developers to use v0.12 . If they use v0.11 , that should be their problem , which they need to figure out.

**Answer:** A

**Explanation:**

<https://www.terraform.io/docs/configuration/terraform.html>

**NEW QUESTION 231**

- (Exam Topic 3)

Why is it a good idea to declare the required version of a provider in a Terraform configuration file?

- \* 1. terraform
- \* 2. {
- \* 3. required\_providers
- \* 4. {
- \* 5. aws = "~> 1.0"
- \* 6. }
- \* 7. }

- A. To remove older versions of the provider.
- B. To ensure that the provider version matches the version of Terraform you are using.
- C. Providers are released on a separate schedule from Terraform itself; therefore a newer version could introduce breaking changes.
- D. To match the version number of your application being deployed via Terraform.

**Answer:** C

**NEW QUESTION 232**

- (Exam Topic 4)

Which of the following is an invalid variable name?

- A. count
- B. web
- C. var1
- D. instance\_name

**Answer:** A

**Explanation:**

<https://www.terraform.io/intro/examples/count.html>

**NEW QUESTION 236**

- (Exam Topic 4)

A Terraform output that sets the "sensitive" argument to true will not store that value in the state file.

- A. True
- B. False

**Answer:** B

**Explanation:**

Reference: <https://www.terraform.io/language/values/outputs>

**NEW QUESTION 237**

- (Exam Topic 4)

How can a ticket-based system slow down infrastructure provisioning and limit the ability to scale? (Choose two.)

- A. A full audit trail of the request and fulfillment process is generated
- B. A request must be submitted for infrastructure changes
- C. As additional resources are required, more tickets are submitted
- D. A catalog of approved resources can be accessed from drop down lists in a request form

**Answer:** BC

**NEW QUESTION 242**

- (Exam Topic 4)

Select the answer below that completes the following statement: Terraform Cloud can be managed from the CLI but requires \_\_\_\_\_?

- A. an API token
- B. a TOTP token
- C. a username and password
- D. authentication using MFA

**Answer:** A

**Explanation:**

API and CLI access are managed with API tokens, which can be generated in the Terraform Cloud UI. Each user can generate any number of personal API tokens, which allow access with their own identity and permissions. Organizations and teams can also generate tokens for automating tasks that aren't tied to an individual user.

**NEW QUESTION 246**

- (Exam Topic 4)

HashiCorp offers multiple versions of Terraform, including Terraform open-source, Terraform Cloud, and Terraform Enterprise. Which of the following Terraform features are only available in the Enterprise edition? (select four)

- A. SAML/SSO
- B. Sentinel
- C. Audit Logs
- D. Clustering
- E. Private Module Registry
- F. Private Network Connectivity

**Answer:** ACF

**Explanation:**

While there are a ton of features that are available to open source users, many features that are part of the Enterprise offering are geared towards larger teams and enterprise functionality. To see what specific features are part of Terraform Cloud and Terraform Enterprise, check out this link.

<https://www.hashicorp.com/products/terraform/pricing/>

**NEW QUESTION 247**

- (Exam Topic 4)

A user has created a module called "my\_test\_module" and committed it to GitHub. Over time, several commits have been made with updates to the module, each tagged in GitHub with an incremental version number. Which of the following lines would be required in a module configuration block in terraform to select tagged version v1.0.4?

- A. source = "git::https://example.com/my\_test\_module.git@tag=v1.0.4"
- B. source = "git::https://example.com/my\_test\_module.git&ref=v1.0.4"
- C. source = "git::https://example.com/my\_test\_module.git#tag=v1.0.4"
- D. source = "git::https://example.com/my\_test\_module.git?ref=v1.0.4"

**Answer:** D

**Explanation:**

<https://www.terraform.io/docs/modules/sources.html#selecting-a-revision>

**NEW QUESTION 250**

- (Exam Topic 4)

As a developer, you want to ensure your plugins are up to date with the latest versions. Which Terraform command should you use?

- A. terreform providers- upgrade
- B. terreform apply -upgrade
- C. terreform refresh -upgrade
- D. terreformn Init -upgrade

**Answer:** D

**NEW QUESTION 252**

- (Exam Topic 4)

In the example below, the depends\_on argument creates what type of dependency?

- A. implicit dependency
- B. internal dependency
- C. explicit dependency
- D. non-dependency resource

**Answer:** C

**NEW QUESTION 254**

- (Exam Topic 4)

Your team has started using terraform OSS in a big way , and now wants to deploy multi region deployments (DR) in aws using the same terraform files . You want to deploy the same infra (VPC,EC2 ...) in both us-east-1 ,and us-west-2 using the same script , and then peer the VPCs across both the regions to enable DR traffic. But , when you run your script , all resources are getting created in only the default provider region. What should you do? Your provider setting is as below  
# The default provider configuration provider "aws" { region = "us-east-1" }

- A. No way to enable this via a single script . Write 2 different scripts with different default providers in the 2 scripts , one for us-east , another for us-west.

- B. Create a list of regions , and then use a for-each to iterate over the regions , and create the same resources ,one after the one , over the loop.
- C. Use provider alias functionality , and add another provider for us-west region . While creating the resources using the tf script , reference the appropriate provider (using the alias).
- D. Manually create the DR region , once the Primary has been created , since you are using terraform OSS , and multi region deployment is only available in Terraform Enterprise.

**Answer:** C

**Explanation:**

You can optionally define multiple configurations for the same provider, and select which one to use on a per-resource or per-module basis. The primary reason for this is to support multiple regions for a cloud platform; other examples include targeting multiple Docker hosts, multiple Consul hosts, etc.

To include multiple configurations for a given provider, include multiple provider blocks with the same provider name, but set the alias meta-argument to an alias name to use for each additional configuration. For example:

```
# The default provider configuration provider "aws" {  
  region = "us-east-1"  
}  
# Additional provider configuration for west coast region provider "aws" {  
  alias = "west" region = "us-west-2"  
}
```

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

**NEW QUESTION 259**

- (Exam Topic 4)

Where can Terraform not load a provider from?

- A. Plugins directory
- B. Provider plugin cache
- C. Official HashrCorp distribution on releases, hashicorp.com
- D. Source code

**Answer:** D

**NEW QUESTION 260**

- (Exam Topic 4)

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

- A. True
- B. False

**Answer:** B

**NEW QUESTION 264**

- (Exam Topic 4)

Which of the following can you do with terraform plan? Choose two correct answers.

- A. View the execution plan and check if the changes match your expectations
- B. Schedule Terraform to run at a planned time in the future
- C. Execute a plan in a different workspace
- D. Save a generated execution plan to apply later

**Answer:** AD

**Explanation:**

<https://learn.hashicorp.com/tutorials/terraform/plan>

**NEW QUESTION 269**

- (Exam Topic 4)

John is writing a module and within the module, there are multiple places where he has to use the same conditional expression but he wants to avoid repeating the same values or expressions multiple times in a configuration,. What is a better approach to dealing with this?

- A. Local Values
- B. Expressions
- C. Functions
- D. Variables

**Answer:** A

**Explanation:**

A local value assigns a name to an expression, allowing it to be used multiple times within a module without repeating it.

<https://www.terraform.io/docs/configuration/locals.html>

**NEW QUESTION 272**

- (Exam Topic 4)

Provider dependencies are created in several different ways. Select the valid provider dependencies from the following list: (select three)

- A. Explicit use of a provider block in configuration, optionally including a version constraint.
- B. Use of any resource belonging to a particular provider in a resource or data block in configuration.

- C. Existence of any resource instance belonging to a particular provider in the current state.
- D. Existence of any provider plugins found locally in the working directory.

**Answer:** ABC

**Explanation:**

The existence of a provider plugin found locally in the working directory does not itself create a provider dependency. The plugin can exist without any reference to it in the terraform configuration. <https://www.terraform.io/docs/commands/providers.html>

**NEW QUESTION 273**

- (Exam Topic 4)

Which of the following arguments are required when declaring a Terraform output?

- A. sensitive
- B. description
- C. default
- D. value

**Answer:** D

**NEW QUESTION 277**

- (Exam Topic 4)

Running terraform fmt without any flags in a directory with Terraform configuration files will check the formatting of those files without changing their contents.

- A. True
- B. False

**Answer:** B

**Explanation:**

The terraform fmt command is used to rewrite Terraform configuration files to a canonical format and style.

**NEW QUESTION 279**

- (Exam Topic 4)

When using providers that require the retrieval of data, such as the HashiCorp Vault provider, in what phase does Terraform actually retrieve the data required?

- A. terraform delete
- B. terraform plan
- C. terraform init
- D. terraform apply

**Answer:** C

**NEW QUESTION 282**

- (Exam Topic 4)

Which task does terraform init not perform?

- A. Sources any modules and copies the configuration locally
- B. Validates all required variables are present
- C. Connects to the backend
- D. Sources all providers present in the configuration and ensures they are downloaded and available locally

**Answer:** B

**NEW QUESTION 283**

- (Exam Topic 4)

In the example below, where is the value of the DNS record's IP address originating from?

```
* 1. resource "aws_route53_record" "www"
* 2. {
* 3.   zone_id = aws_route53_zone.primary.zone_id
* 4.   name = "www.example.com"
* 5.   type = "A"
* 6.   ttl = "300"
* 7.   records = [module.web_server.instance_ip_address] 8. }
```

- A. The regular expression named module.web\_server
- B. The output of a module named web\_server
- C. By querying the AWS EC2 API to retrieve the IP address
- D. Value of the web\_server parameter from the variables.tf file

**Answer:** B

**Explanation:**

In a parent module, outputs of child modules are available in expressions as module.<MODULE NAME>.<OUTPUT NAME>.

For example, if a child module named web\_server declared an output named instance\_ip\_address, you could access that value as module.web\_server.instance\_ip\_address.

#### NEW QUESTION 284

- (Exam Topic 4)

Terraform variable names are saved in the state file.

- A. True
- B. False

**Answer:** B

#### Explanation:

Terraform stores information about your infrastructure in a state file. This state file keeps track of resources created by your configuration and maps them to real-world resources. <https://learn.hashicorp.com/tutorials/terraform/state-cli>

#### NEW QUESTION 287

- (Exam Topic 4)

Which of the following is true about Terraform's implementation of infrastructure as code? (Choose two.)

- A. It is only compatible with AWS infrastructure management
- B. You cannot reuse infrastructure configuration
- C. You can version your infrastructure configuration
- D. It requires manual configuration of infrastructure resources
- E. It allows you to automate infrastructure provisioning

**Answer:** CE

#### NEW QUESTION 289

- (Exam Topic 4)

You're preparing to install Terraform on client workstations and want to see which operating systems are supported. Which of the following operating systems is supported?

- A. Windows
- B. Amazon Linux
- C. FreeBSD
- D. Solaris
- E. MacOS
- F. All of the above

**Answer:** F

#### NEW QUESTION 291

- (Exam Topic 4)

Which statements best describes what the local variable assignment is doing in the following code snippet:

- A. Create a distinct list of route table name objects
- B. Create a map of route table names to subnet names
- C. Create a map of route table names from a list of subnet names
- D. Create a list of route table names eliminating duplicates

**Answer:** D

#### NEW QUESTION 295

- (Exam Topic 4)

Talal is a DevOps engineer and he has deployed the production infrastructure using Terraform. He is using a very large configuration file to maintain and update the actual infrastructure. As the infrastructure have grown to a very complex and large, he has started experiencing slowness when he run runs terraform plan. What are the options for him to resolve this slowness?

- A. Use -refresh=true flag as well as the -target flag with terraform plan in order to work around this.
- B. Run terraform refresh every time before running terraform plan.
- C. Break large configurations into several smaller configurations that can each be independently applied.
- D. Use -refresh=false flag as well as the -target flag with terraform plan in order to work around this.

**Answer:** CD

#### Explanation:

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.

Although 'Use -refresh=false flag as well as the -target flag with terraform plan in order to work around this.' is a solution, but its not always recommended. Instead of using -target as a means to operate on isolated portions of very large configurations, prefer instead to break large configurations into several smaller configurations that can each be independently applied. Data sources can be used to access information about resources created in other configurations, allowing a complex system architecture to be broken down into more manageable parts that can be updated independently.

Option 'Run terraform refresh every time before running terraform plan.' and 'Use -refresh=true flag as well as the -target flag with terraform plan in order to work around this.' is not correct because in both the cases terraform will query every resources of the infrastructure.

#### NEW QUESTION 299

- (Exam Topic 4)

You have created a main.tf Terraform configuration consisting of an application server, a database, and a load balancer. You ran terraform apply and all resources

were created successfully. Now you realize that you do not actually need the load balancer so you run terraform destroy without any flags What will happen?

- A. Terraform will destroy the application server because it is listed first in the code
- B. Terraform will prompt you to confirm that you want to destroy all the infrastructure
- C. Terraform will destroy the main.tf file
- D. Terraform will prompt you to pick which resource you want to destroy
- E. Terraform will immediately destroy all the infrastructure

**Answer: B**

### NEW QUESTION 303

- (Exam Topic 4)

You are creating a Terraform configuration which needs to make use of multiple providers, one for AWS and one for Datadog. Which of the following provider blocks would allow you to do this?

A)

```
provider {  
  "aws" {  
    profile = var.aws_profile  
    region  = var.aws_region  
  }  
  
  "datadog" {  
    api_key = var.datadog_api_key  
    app_key = var.datadog_app_key  
  }  
}
```

B)

```
provider "aws" {  
  profile = var.aws_profile  
  region  = var.aws_region  
}  
  
provider "datadog" {  
  api_key = var.datadog_api_key  
  app_key = var.datadog_app_key  
}
```

C)

```
terraform {  
  provider "aws" {  
    profile = var.aws_profile  
    region  = var.aws_region  
  }  
  
  provider "datadog" {  
    api_key = var.datadog_api_key  
    app_key = var.datadog_app_key  
  }  
}
```

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

**Answer:** B

**Explanation:**

<https://www.terraform.io/language/providers/configuration>

#### NEW QUESTION 305

- (Exam Topic 4)

What resource dependency information is stored in Terraform's state?

- A. Only implicit dependencies are stored in state.
- B. Both implicit and explicit dependencies are stored in state.
- C. Only explicit dependencies are stored in state.
- D. No dependency information is stored in state.

**Answer:** B

**Explanation:**

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>

#### NEW QUESTION 307

- (Exam Topic 4)

State is a requirement for Terraform to function

- A. True
- B. False

**Answer:** A

**Explanation:**

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.

#### NEW QUESTION 308

- (Exam Topic 4)

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

- A. True
- B. False

**Answer:** B

**Explanation:**

<https://registry.terraform.io/>

Only modules considered "Verified Modules" are reviewed by Hashicorp, otherwise anyone can publish modules on the Terraform Registry.

Reference: <https://www.terraform.io/registry/modules/verified> <https://www.terraform.io/registry/modules/publish>

**NEW QUESTION 311**

- (Exam Topic 4)

True or False: Workspaces provide identical functionality in the open-source, Terraform Cloud, and Enterprise versions of Terraform.

A. True

B. False

**Answer:** B

**Explanation:**

<https://www.terraform.io/docs/cloud/workspaces/index.html> <https://www.terraform.io/docs/state/workspaces.html>

**NEW QUESTION 313**

- (Exam Topic 4)

You decide to move a Terraform state file to Amazon S3 from another location. You write the code below into a file called\

```
terraform {  
  backend "s3" {  
    bucket = "my-tf-bucket"  
    region = "us-east-1"  
  }  
}
```

You immediately run terraform apply but don't see any changes. Your state file didn't move. Which command will migrate your current state file to the new S3 remote backend?

A. terraform push

B. terraform init

C. terraform refresh

D. terraform state

**Answer:** B

**NEW QUESTION 317**

- (Exam Topic 4)

How would you reference the Volume IDs associated with the ebs\_block\_device blocks in this configuration?

```
resource "aws_instance" "example" {  
  ami = "ami-abc123"  
  instance_type = "t2.micro"  
  
  ebs_block_device {  
    device_name = "sda2"  
    volume_size = 16  
  }  
  
  ebs_block_device {  
    device_name = "sda3"  
    volume_size = 20  
  }  
}
```

A. aws\_instance.example.ebs\_block\_device.[\*].volume\_id

B. aws\_instance.example.ebs\_block\_device.volume\_id

C. aws\_instance.example.ebs\_block\_device[sda2,sda3].volume\_id

D. aws\_instance.example.ebs\_block\_device.\*.volume\_id

**Answer:** A

**Explanation:**

[https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/device\\_naming.html](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/device_naming.html)

#### NEW QUESTION 322

- (Exam Topic 4)

Your developers are facing a lot of problem while writing complex expressions involving difficult interpolations . They have to run the terraform plan every time and check whether there are errors , and also check terraform apply to print the value as a temporary output for debugging purposes. What should be done to avoid this?

- A. Use terraform console command to have an interactive UI with full access to the underlying terraform state to run your interpolations , and debug at real-time.
- B. Add a breakpoint in your code, using the watch keyword , and output the value to console for temporary debugging.
- C. Use terraform zipmap function , it will be able to easily do the interpolations without complex code.
- D. Use terraform console command to have an interactive UI , but you can only use it with local state , and it does not work with remote state.

**Answer:** A

#### Explanation:

The terraform console command provides an interactive console for evaluating expressions. This is useful for testing interpolations before using them in configurations, and for interacting with any values currently saved in state.

<https://www.terraform.io/docs/commands/console.html>

#### NEW QUESTION 325

- (Exam Topic 4)

Module version is required to reference a module on the Terraform Module Registry.

- A. True
- B. False

**Answer:** B

#### NEW QUESTION 327

- (Exam Topic 4)

You wanted to destroy some of the dependent resources from real infrastructure. You choose to delete those resources from your configuration file and run terraform plan and then apply. Which of the following way your resources would be destroyed?

- A. Terraform can still determine the correct order for destruction from the state even when you delete one or more items from the configuration.
- B. Those would be destroyed in the order in which they were written in the configuration file previously before you have deleted them from configuration file.
- C. The resource will be destructed in random order as you have already deleted them from configuration.
- D. You can not destroy resources by deleting them from configuration file and running plan and apply.

**Answer:** A

#### Explanation:

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.

#### NEW QUESTION 331

- (Exam Topic 4)

Terraform will sync all resources in state by default for every plan and apply, hence for larger infrastructures this can slow down terraform plan and terraform apply commands?

- A. False
- B. True

**Answer:** B

#### Explanation:

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.

<https://www.terraform.io/docs/state/purpose.html>

#### NEW QUESTION 335

- (Exam Topic 4)

How does Terraform determine dependencies between resources?

- A. Terraform automatically builds a resource graph based on resources, provisioners, special meta-parameters, and the state file, if present.
- B. Terraform requires all dependencies between resources to be specified using the depends\_on parameter
- C. Terraform requires resources in a configuration to be listed in the order they will be created to determine dependencies
- D. Terraform requires resource dependencies to be defined as modules and sourced in order

**Answer:** A

#### Explanation:

<https://learn.hashicorp.com/tutorials/terraform/dependencies>

#### NEW QUESTION 340

- (Exam Topic 4)

Which of the following statements about local modules is incorrect:

- A. Local modules are not cached by terraform init command
- B. Local modules are sourced from a directory on disk
- C. Local modules support versions
- D. All of the above (all statements above are incorrect)
- E. None of the above (all statements above are correct)

**Answer:** C

#### Explanation:

Version constraints are supported only for modules installed from a module registry, such as the public Terraform Registry or Terraform Cloud's private module registry. Other module sources can provide their own versioning mechanisms within the source string itself, or might not support versions at all. In particular, modules sourced from local file paths do not support version; since they're loaded from the same source repository, they always share the same version as their caller.

<https://www.terraform.io/language/modules/syntax>

#### NEW QUESTION 345

- (Exam Topic 4)

Select two answers to complete the following sentence: Before a new provider can be used, it must be \_\_\_\_\_ and \_\_\_\_\_.

- A. approved by HashiCorp
- B. uploaded to source control
- C. declared in the configuration
- D. initialized

**Answer:** CD

#### Explanation:

Each time a new provider is added to configuration -- either explicitly via a provider block or by adding a resource from that provider -- Terraform must initialize the provider before it can be used. Initialization downloads and installs the provider's plugin so that it can later be executed.

#### NEW QUESTION 350

- (Exam Topic 4)

Your organization has moved to AWS and has manually deployed infrastructure using the console. Recently, a decision has been made to standardize on Terraform for all deployments moving forward.

What can you do to ensure that all existing is managed by Terraform moving forward without interruption to existing services?

- A. Submit a ticket to AWS and ask them to export the state of all existing resources and use terraform import to import them into the state file.
- B. Delete the existing resources and recreate them using new a Terraform configuration so Terraform can manage them moving forward.
- C. Resources that are manually deployed in the AWS console cannot be imported by Terraform.
- D. Using terraform import, import the existing infrastructure into your Terraform state.

**Answer:** D

#### Explanation:

Terraform is able to import existing infrastructure. This allows us take resources we've created by some other means (i.e. via console) and bring it under Terraform management.

This is a great way to slowly transition infrastructure to Terraform.

The terraform import command is used to import existing infrastructure.

To import a resource, first write a resource block for it in our configuration, establishing the name by which it will be known to Terraform.

Example:

```
resource "aws_instance" "import_example" {  
  # ...instance configuration...  
}
```

Now terraform import can be run to attach an existing instance to this resource configuration.

```
$ terraform import aws_instance.import_example i-03efafa258104165f  
aws_instance.import_example: Importing from ID "i-03efafa258104165f"...
```

```
aws_instance.import_example: Import complete!
```

```
Imported aws_instance (ID: i-03efafa258104165f) aws_instance.import_example: Refreshing state... (ID: i-03efafa258104165f) Import successful!
```

The resources that were imported are shown above. These resources are now in your Terraform state and will henceforth be managed by Terraform.

This command locates the AWS instance with ID i-03efafa258104165f (which has been created outside

Terraform) and attaches its existing settings, as described by the EC2 API, to the name aws\_instance.import\_example in the Terraform state.

#### NEW QUESTION 352

- (Exam Topic 4)

terraform apply will fail if you have not run terraform plan first to update the plan output.

- A. True
- B. False

**Answer:** B

#### NEW QUESTION 355

- (Exam Topic 4)

You have written a terraform IaC script which was working till yesterday , but is giving some vague error from today , which you are unable to understand . You want more detailed logs that could potentially help you troubleshoot the issue , and understand the root cause. What can you do to enable this setting? Please note , you are using terraform OSS.

- A. Terraform OSS can push all its logs to a syslog endpoint
- B. As such, you have to set up the syslog sink, and enable TF\_LOG\_PATH env variable to the syslog endpoint and all logs will automatically start streaming.
- C. Detailed logs are not available in terraform OSS, except the crash messag
- D. You need to upgrade to terraform enterprise for this point.
- E. Enable the TF\_LOG\_PATH to the log sink file location, and logging output will automatically be stored there.
- F. Enable TF\_LOG to the log level DEBUG, and then set TF\_LOG\_PATH to the log sink file location. Terraform debug logs will be dumped to the sink path, even in terraform OSS.

**Answer:** D

**Explanation:**

Terraform has detailed logs which can be enabled by setting the TF\_LOG environment variable to any value. This will cause detailed logs to appear on stderr. You can set TF\_LOG to one of the log levels TRACE, DEBUG, INFO, WARN or ERROR to change the verbosity of the logs. TRACE is the most verbose and it is the default if TF\_LOG is set to something other than a log level name. To persist logged output you can set TF\_LOG\_PATH in order to force the log to always be appended to a specific file when logging is enabled. Note that even when TF\_LOG\_PATH is set, TF\_LOG must be set in order for any logging to be enabled.

**NEW QUESTION 358**

- (Exam Topic 4)

terraform validate validate validates that your infrastructure matches the Terraform state file.

- A. True
- B. False

**Answer:** B

**Explanation:**

The terraform validate command validates the configuration files in a directory, referring only to the configuration and not accessing any remote services such as remote state, provider APIs, etc. Validate runs checks that verify whether a configuration is syntactically valid and internally consistent, regardless of any provided variables or existing state. It is thus primarily useful for general verification of reusable modules, including correctness of attribute names and value types. Source: <https://www.terraform.io/cli/commands/validate>

**NEW QUESTION 361**

- (Exam Topic 4)

Terraform Cloud is more powerful when you integrate it with your version control system (VCS) provider. Select all the supported VCS providers from the answers below. (select four)

- A. GitHub
- B. CVS Version Control
- C. Azure DevOps Server
- D. Bitbucket Cloud
- E. GitHub Enterprise

**Answer:** ACDE

**Explanation:**

Terraform Cloud supports the following VCS providers:

- <https://www.terraform.io/docs/cloud/vcs/github.html>
- <https://www.terraform.io/docs/cloud/vcs/github.html>
- <https://www.terraform.io/docs/cloud/vcs/github-enterprise.html>
- <https://www.terraform.io/docs/cloud/vcs/gitlab-com.html>
- <https://www.terraform.io/docs/cloud/vcs/gitlab-eece.html>
- <https://www.terraform.io/docs/cloud/vcs/bitbucket-cloud.html>
- <https://www.terraform.io/docs/cloud/vcs/bitbucket-server.html>
- <https://www.terraform.io/docs/cloud/vcs/azure-devops-server.html>
- <https://www.terraform.io/docs/cloud/vcs/azure-devops-services.html> <https://www.terraform.io/docs/cloud/vcs/index.html#supported-vcs-providers>

**NEW QUESTION 365**

- (Exam Topic 4)

When does Sentinel enforce policy logic during a Terraform Enterprise run?

- A. Before the plan phase
- B. During the plan phase
- C. Before the a apply phase
- D. After the apply phase

**Answer:** C

**Explanation:**

"Enforcing policy checks on runs - Policies are checked when a run is performed, after the terraform plan but before it can be confirmed or the terraform apply is executed."

**NEW QUESTION 370**

- (Exam Topic 4)

Why does this backend configuration not follow best practices?

```
terraform {  
  backend "s3" {  
    bucket     = "terraform-state-prod"  
    key        = "network/terraform.tfstate"  
    region     = "us-east-1"  
    access_key = "AKIAIOSFODNN7EXAMPLE"  
    secret_key = "wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY"  
  }  
  
  required_providers {  
    aws = {  
      source = "hashicorp/aws"  
      version = "~> 3.38"  
    }  
  }  
  
  required_version = ">= 0.15"  
}
```

- A. You should not store credentials in Terraform Configuration
- B. You should use the local enhanced storage backend whenever possible
- C. An alias meta-argument should be included in backend blocks whenever possible
- D. The backend configuration should contain multiple credentials so that more than one user can execute terraform plan and terraform apply

**Answer:** A

#### NEW QUESTION 371

- (Exam Topic 4)

Which feature of Terraform allows multiple state files for a single configuration file depending upon the environment?

- A. Terraform Modules
- B. Terraform Enterprise
- C. Terraform Workspaces
- D. Terraform Remote Backends

**Answer:** C

#### NEW QUESTION 373

- (Exam Topic 4)

Named workspaces are not a suitable isolation mechanism for strong separation between staging and production?

- A. True
- B. False

**Answer:** A

#### 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>

#### NEW QUESTION 378

- (Exam Topic 4)

You're writing a Terraform configuration that needs to read input from a local file called id\_rsa.pub. Which built-in Terraform function can you use to import the file's contents as a string?

- A. fileset("id\_rsa.pub")
- B. filebase64("id\_rsa.pub")
- C. templatefile("id\_rsa.pub")
- D. file("id\_rsa.pub")

**Answer:** D

#### Explanation:

<https://www.terraform.io/language/functions/file>

#### NEW QUESTION 382

- (Exam Topic 4)

Changing the Terraform backend from the default "local" backend to a different one after doing your first terraform apply is:

- A. Mandatory
- B. Optional

- C. Impossible
- D. Discouraged

**Answer:** B

#### NEW QUESTION 384

- (Exam Topic 4)

What kind of configuration block will create an infrastructure object with settings specified in the block?

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

**Answer:** C

#### NEW QUESTION 388

- (Exam Topic 4)

You cannot install third party plugins using terraform init.

- A. True
- B. False

**Answer:** B

#### Explanation:

<https://www.terraform.io/cli/commands/init>

For providers that are published in either the public Terraform Registry or in a third-party provider registry, terraform init will automatically find, download, and install the necessary provider plugins.

#### NEW QUESTION 390

- (Exam Topic 4)

When using parent/child modules to deploy infrastructure, how would you export a value from one module to import into another module.

For example, a module dynamically deploys an application instance or virtual machine, and you need the IP address in another module to configure a related DNS record in order to reach the newly deployed application.

- A. Export the value using terraform export and input the value using terraform input.
- B. Configure the pertinent provider's configuration with a list of possible IP addresses to use.
- C. Configure an output value in the application module in order to use that value for the DNS module.
- D. Preconfigure the IP address as a parameter in the DNS module.

**Answer:** C

#### Explanation:

Output values are like the return values of a Terraform module, and have several uses:

- \* A child module can use outputs to expose a subset of its resource attributes to a parent module.
- \* A root module can use outputs to print certain values in the CLI output after running terraform apply.
- \* When using remote state, root module outputs can be accessed by other configurations via a terraform\_remote\_state data source.

<https://www.terraform.io/docs/configuration/outputs.html>

#### NEW QUESTION 392

- (Exam Topic 4)

Suppose terraformcode is taking up some values which are not defined inside the code files. In which of the following options issue might have occurred?

- A. Issue in main.tf file
- B. Issue in vars.tf file
- C. Issue in terraform.tfvars
- D. Issue in Environment Variables

**Answer:** D

#### NEW QUESTION 395

- (Exam Topic 4)

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

- A. The module's configuration page on the Terraform Module Registry
- B. Terraform Module Registry does not support versioning modules
- C. The release tags in the associated repo Most Voted
- D. The module's Terraform code

**Answer:** C

#### Explanation:

<https://www.terraform.io/registry/modules/publish>

#### NEW QUESTION 399

- (Exam Topic 4)

Which are forbidden actions when the Terraform state file is locked? (Choose three.)

- A. terraform destroy
- B. terraform fmt
- C. terraform state list
- D. terraform apply
- E. terraform plan
- F. terraform validate

**Answer:** ADE

#### NEW QUESTION 403

- (Exam Topic 4)

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

- A. Terraform attempts to provision the resource up to three times before exiting with an error
- B. the terraform plan is rolled back and all provisioned resources are removed
- C. it is automatically deleted
- D. the resource is marked as tainted

**Answer:** D

#### 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.

#### NEW QUESTION 406

- (Exam Topic 4)

```
resource "aws_s3_bucket" "example" { bucket = "my-test-s3-terraform-bucket" ...} resource "aws_iam_role" "test_role" { name = "test_role" ...}
```

Due to the way that the application code is written , the s3 bucket must be created before the test role is created , otherwise there will be a problem. How can you ensure that?

- A. This will already be taken care of by terraform native implicit dependenc
- B. Nothing else needs to be done from your end.
- C. Add explicit dependency using depends\_on . This will ensure the correct order of resource creation.
- D. Create 2 separate terraform config scripts , and run them one by one , 1 for s3 bucket , and another for IAM role , run the S3 bucket script first.
- E. This is not possible to control in terraform . Terraform will take care of it in a native way , and create a dependency graph that is best suited for the parallel resource creation.

**Answer:** B

#### Explanation:

Use the depends\_on meta-argument to handle hidden resource dependencies that Terraform can't automatically infer.

Explicitly specifying a dependency is only necessary when a resource relies on some other resource's behavior but doesn't access any of that resource's data in its arguments.

#### NEW QUESTION 407

- (Exam Topic 4)

What are some of the problems of how infrastructure was traditionally managed before Infrastructure as Code? (select three)

- A. Requests for infrastructure or hardware required a ticket, increasing the time required to deploy applications
- B. Traditional deployment methods are not able to meet the demands of the modern business where resources tend to live days to weeks, rather than months to years
- C. Traditionally managed infrastructure can't keep up with cyclic or elastic applications
- D. Pointing and clicking in a management console is a scalable approach and reduces human error as businesses are moving to a multi-cloud deployment model

**Answer:** ABC

#### Explanation:

Businesses are making a transition where traditionally-managed infrastructure can no longer meet the demands of today's businesses. IT organizations are quickly adopting the public cloud, which is predominantly API-driven. To meet customer demands and save costs, application teams are architecting their applications to support a much higher level of elasticity, supporting technology like containers and public cloud resources. These resources may only live for a matter of hours; therefore the traditional method of raising a ticket to request resources is no longer a viable option Pointing and clicking in a management console is NOT scale and increases the change of human error.

#### NEW QUESTION 410

- (Exam Topic 4)

When do you need to explicitly execute terraform refresh?

- A. Before every terraform plan
- B. Before every terraform apply
- C. Before every terraform import
- D. None of the above

**Answer:** D

**Explanation:**

Wherever possible, avoid using terraform refresh explicitly and instead rely on Terraform's behavior of automatically refreshing existing objects as part of creating a normal plan. Source: <https://www.terraform.io/cli/commands/refresh>

**NEW QUESTION 411**

- (Exam Topic 4)

In the below configuration, how would you reference the module output vpc\_id ?

```
module "vpc" {  
  source = "terraform-aws-modules/vpc/aws"  
  cidr   = "10.0.0.0/16"  
  name   = "test-vpc"  
}
```

Type your answer in the field provided. The text field is not case-sensitive and all variations of the correct answer are accepted.

- A. Mastered
- B. Not Mastered

**Answer:** A

**Explanation:**

module.vpc.id

**NEW QUESTION 414**

- (Exam Topic 4)

Once a new Terraform backend is configured with a Terraform code block, which command(s) is (are) used to migrate the state file?

- A. terraform apply
- B. terraform push
- C. terraform destroy, then terraform apply
- D. terraform init

**Answer:** B

**Explanation:**

<https://www.terraform.io/cli/commands/state/push>

**NEW QUESTION 415**

- (Exam Topic 4)

You want to define multiple data disks as nested blocks inside the resource block for a virtual machine. What Terraform feature would help you define the blocks using the values in a variable?

- A. Local values
- B. Dynamic blocks
- C. Count arguments
- D. Collection functions

**Answer:** B

**NEW QUESTION 418**

- (Exam Topic 4)

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

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

**Answer:** CD

**NEW QUESTION 421**

- (Exam Topic 4)

A variable az has the following default value. What will be the datatype of the variable? az=["us-west-1a","us-east-1a"]

- A. Object
- B. List
- C. Map
- D. String

**Answer:** B

**NEW QUESTION 423**

- (Exam Topic 4)

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 `ref=v1.0.0` argument to the source path Most Voted
- B. Add `version = "1.0.0"` parameter to module block
- C. Nothing "modules stored on GitHub always default to version 1.0.0"
- D. Modules stored on GitHub do not support versioning

**Answer:** A

**Explanation:**

<https://www.terraform.io/language/modules/sources#selecting-a-revision>

**NEW QUESTION 428**

- (Exam Topic 4)

Given the Terraform configuration below, in which order will the resources be created?

```
* 1. resource "aws_instance" "web_server"
* 2. {
* 3.   ami = "ami-b374d5a5"
* 4.   instance_type = "t2.micro"
* 5. }
* 6. resource "aws_eip" "web_server_ip"
* 7. {
* 8.   vpc = true instance = aws_instance.web_server.id
* 9. }
```

- A. `aws_eip` will be created first `aws_instance` will be created second
- B. `aws_eip` will be created first `aws_instance` will be created second
- C. Resources will be created simultaneously
- D. `aws_instance` will be created first `aws_eip` will be created second

**Answer:** D

**Explanation:**

Implicit and Explicit Dependencies

By studying the resource attributes used in interpolation expressions, Terraform can automatically infer when one resource depends on another. In the example above, the reference to `aws_instance.web_server.id` creates an implicit dependency on the `aws_instance` named `web_server`.

Terraform uses this dependency information to determine the correct order in which to create the different resources.

```
# Example of Implicit Dependency resource "aws_instance" "web_server" { ami = "ami-b374d5a5"
instance_type = "t2.micro"
}
resource "aws_eip" "web_server_ip" { vpc = true
instance = aws_instance.web_server.id
}
```

In the example above, Terraform knows that the `aws_instance` must be created before the `aws_eip`. Implicit dependencies via interpolation expressions are the primary way to inform Terraform about these relationships, and should be used whenever possible.

Sometimes there are dependencies between resources that are not visible to Terraform. The `depends_on` argument is accepted by any resource and accepts a list of resources to create explicit dependencies for.

For example, perhaps an application we will run on our EC2 instance expects to use a specific Amazon S3 bucket, but that dependency is configured inside the application code and thus not visible to Terraform. In that case, we can use `depends_on` to explicitly declare the dependency:

```
# Example of Explicit Dependency
# New resource for the S3 bucket our application will use. resource "aws_s3_bucket" "example" {
bucket = "terraform-getting-started-guide" acl = "private"
}
# Change the aws_instance we declared earlier to now include "depends_on" resource "aws_instance" "example" {
ami = "ami-2757f631" instance_type = "t2.micro"
# Tells Terraform that this EC2 instance must be created only after the
# S3 bucket has been created. depends_on = [aws_s3_bucket.example]
}
```

<https://learn.hashicorp.com/terraform/getting-started/dependencies.html>

**NEW QUESTION 431**

- (Exam Topic 4)

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 when you run `terraform apply` in the working directory again?

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

**Answer:** B

**Explanation:**

If you remove the resource from your config file and the resource is in your state file, terraform will apply the configuration in the config file - which is to delete the resource

**NEW QUESTION 432**

- (Exam Topic 4)

Select the operating systems which are supported for a clustered Terraform Enterprise: (select four)

- A. Unix
- B. Red Hat
- C. CentOS
- D. Amazon Linux
- E. Ubuntu

**Answer:** BCDE

**Explanation:**

<https://www.terraform.io/docs/enterprise/before-installing/index.html#operating-systemrequirements>

#### NEW QUESTION 436

- (Exam Topic 4)

Which of the following is not valid source path for specifying a module?

- A. source = "./module?version=v1.0.0"
- B. source = "github.com/hashicorp/example?ref=v1.0.0"
- C. source = "./module"
- D. source = "hashicorp/consul/aws"

**Answer:** A

#### NEW QUESTION 438

- (Exam Topic 4)

Select all features which are exclusive to Terraform Enterprise. (Select Three)

- A. Sentinel
- B. Cost Estimation
- C. Audit Logs
- D. Clustering
- E. SAML/SSO

**Answer:** CDE

**Explanation:**

Sentinel and Cost Estimation are also available in Terraform Cloud <https://www.hashicorp.com/products/terraform/pricing/>

#### NEW QUESTION 439

- (Exam Topic 4)

As a member of the operations team, you need to run a script on a virtual machine created by Terraform. Which provisioner is best to use in your Terraform code?

- A. local-exec
- B. file
- C. null-exec
- D. remote-exec

**Answer:** D

**Explanation:**

<https://www.terraform.io/language/resources/provisioners/remote-exec>

#### NEW QUESTION 441

- (Exam Topic 4)

Which argument(s) are required when declaring a Terraform variable?

- A. type
- B. default
- C. description
- D. All of the above
- E. None of the above

**Answer:** E

**Explanation:**

Terraform CLI defines the following OPTIONAL arguments for variable declarations: default - A default value which then makes the variable optional. type - This argument specifies what value types are accepted for the variable. description - This specifies the input variable's documentation. validation - A block to define validation rules, usually in addition to type constraints. sensitive - Limits Terraform UI output when the variable is used in configuration. nullable - Specify if the variable can be null within the module. <https://www.terraform.io/language/values/variables#arguments>

#### NEW QUESTION 442

- (Exam Topic 4)

terraform apply is failing with the following error. What next step should you take to determine the root cause of the problem?

Error loading state: AccessDenied: Access Denied status code: 403, request id: 288766CE5CCA24A0, host id: FOOBAR

- A. Set TF\_LOG=DEBUG
- B. Review syslog for Terraform error messages
- C. Run terraform login to reauthenticate with the provider
- D. Review /var/log/terraform.log for error messages

**Answer:** A

**Explanation:**

Terraform has detailed logs which can be enabled by setting the TF\_LOG environment variable to any value. This will cause detailed logs to appear on stderr. You can set TF\_LOG to one of the log levels (in order of decreasing verbosity) TRACE, DEBUG, INFO, WARN or ERROR to change the verbosity of the logs.

**NEW QUESTION 444**

- (Exam Topic 4)

In the following code snippet, the block type is identified by which string?

- A. "aws\_instance"
- B. resource
- C. "db"
- D. instance\_type

**Answer:** B

**NEW QUESTION 447**

- (Exam Topic 4)

When writing Terraform code, HashiCorp recommends that you use how many spaces between each nesting level?

- A. 1
- B. 2
- C. 4

**Answer:** C

**Explanation:**

The Terraform parser allows you some flexibility in how you lay out the elements in your configuration files, but the Terraform language also has some idiomatic style conventions which we recommend users always follow for consistency between files and modules written by different teams. Automatic source code formatting tools may apply these conventions automatically.

Indent two spaces for each nesting level.

When multiple arguments with single-line values appear on consecutive lines at the same nesting level, align their equals signs:

```
ami = "abc123" instance_type = "t2.micro"
```

When both arguments and blocks appear together inside a block body, place all of the arguments together at the top and then place nested blocks below them.

Use one blank line to separate the arguments from the blocks.

Use empty lines to separate logical groups of arguments within a block.

For blocks that contain both arguments and "meta-arguments" (as defined by the Terraform language semantics), list meta-arguments first and separate them from other arguments with one blank line. Place meta-argument blocks last and separate them from other blocks with one blank line.

```
resource "aws_instance" "example" { count = 2 # meta-argument first
```

```
ami = "abc123" instance_type = "t2.micro" network_interface {
```

```
# ...
```

```
}
```

```
lifecycle { # meta-argument block last create_before_destroy = true
```

```
}
```

```
}
```

Top-level blocks should always be separated from one another by one blank line. Nested blocks should also be separated by blank lines, except when grouping together related blocks of the same type (like multiple provisioner blocks in a resource).

Avoid separating multiple blocks of the same type with other blocks of a different type, unless the block types are defined by semantics to form a family. (For example: root\_block\_device, ebs\_block\_device and

ephemeral\_block\_device on aws\_instance form a family of block types describing AWS block devices, and can therefore be grouped together and mixed.)

**NEW QUESTION 451**

- (Exam Topic 4)

What does terraform refresh modify?

- A. Your cloud infrastructure
- B. Your state file
- C. Your Terraform plan
- D. Your Terraform configuration

**Answer:** B

**Explanation:**

The terraform refresh command reads the current settings from all managed remote objects and updates the Terraform state to match. Source:

<https://www.terraform.io/cli/commands/refresh>

**NEW QUESTION 454**

- (Exam Topic 4)

Given the Terraform configuration below, in which order will the resources be created?

- A. Larger image
- B. resources will be created simultaneously
- C. aws\_eip will be created first aws\_instance will be created second

D. aws\_instance will be created first aws\_eip will be created second

**Answer:** D

**Explanation:**

The aws\_instance will be created first, and then aws\_eip will be created second due to the aws\_eip's resource dependency of the aws\_instance id

**NEW QUESTION 459**

- (Exam Topic 4)

What does terraform destroy do?

- A. Destroy all infrastructure in the Terraform state file
- B. Destroy all Terraform code files in the current directory while leaving the state file intact
- C. Destroy all infrastructure in the configured Terraform provider
- D. Destroy the Terraform state file while leaving infrastructure intact

**Answer:** A

**Explanation:**

The terraform destroy command terminates resources managed by your Terraform project. This command is the inverse of terraform apply in that it terminates all the resources specified in your Terraform state. It does not destroy resources running elsewhere that are not managed by the current Terraform project.

<https://learn.hashicorp.com/tutorials/terraform/aws-destroy>

**NEW QUESTION 464**

.....

## THANKS FOR TRYING THE DEMO OF OUR PRODUCT

Visit Our Site to Purchase the Full Set of Actual TA-002-P Exam Questions With Answers.

We Also Provide Practice Exam Software That Simulates Real Exam Environment And Has Many Self-Assessment Features. Order the TA-002-P Product From:

<https://www.2passeasy.com/dumps/TA-002-P/>

## Money Back Guarantee

### TA-002-P Practice Exam Features:

- \* TA-002-P Questions and Answers Updated Frequently
- \* TA-002-P Practice Questions Verified by Expert Senior Certified Staff
- \* TA-002-P Most Realistic Questions that Guarantee you a Pass on Your FirstTry
- \* TA-002-P Practice Test Questions in Multiple Choice Formats and Updatesfor 1 Year