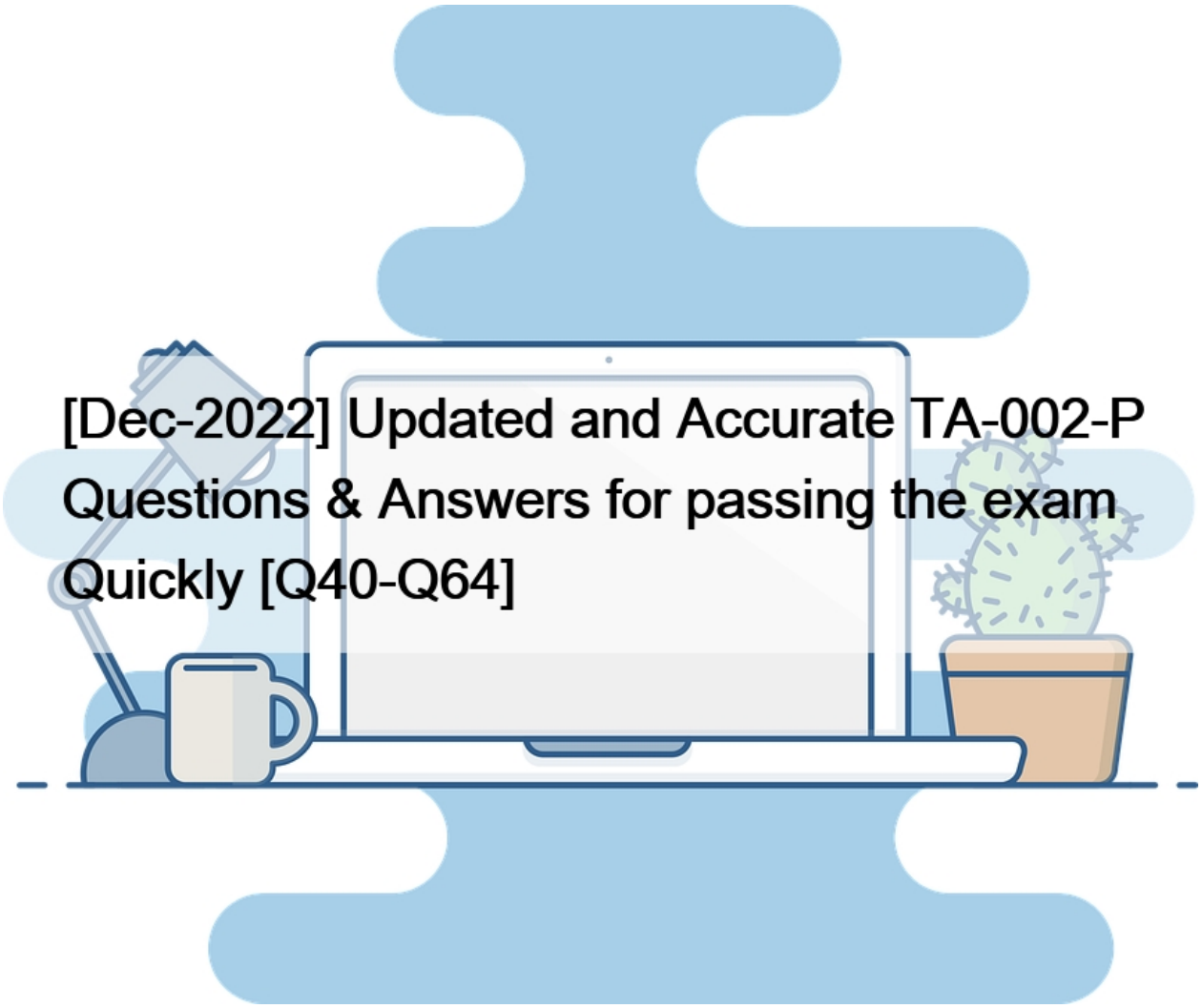
[Dec-2022 Updated and Accurate TA-002-P Questions & Answers for passing the exam Quickly [Q40-Q64



[Dec-2022] Updated and Accurate TA-002-P Questions & Answers for passing the exam Quickly
Download Real TA-002-P Exam Dumps for candidates. 100% Free Dump Files

NO.40 Provisioners should only be used as a last resort.
* False
* True
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

NO.41 Which flag would be used within a Terraform configuration block to identify the specific version of a provider required?
* required-provider
* required-version
* required_providers
* required_versions

For production use, you should constrain the acceptable provider versions via configuration file to ensure that new versions with breaking changes will not be automatically installed by terraform init in the future.

Example

terraform {

required_providers {

aws = “>= 2.7.0”

}

}

NO.42 Which of the following is allowed as a Terraform variable name?
* count
* name
* source
* version

NO.43 FILL BLANK

What is the name of the default file where Terraform stores the state?

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

answer are accepted.
terraform.tfstate

“This state is stored by default in a local file named “terraform.tfstate”, but it can also be stored remotely,

which works better in a team environment.” https://www.terraform.io/language/state

NO.44 After running into issues with Terraform, you need to enable verbose logging to assist with troubleshooting the error. Which

of the following values provides the MOST verbose logging?
* ERROR
* INFO
* WARN
* TRACE
* DEBUG
Explanation

Terraform has detailed logs that 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.

Examples:

export TF_LOG=DEBUG

export TF_LOG=TRACE

NO.45 Most Terraform providers interact with _____.
* API
* VCS Systems
* Shell scripts
* None of the above
Explanation

Terraform relies on plugins called “providers” to interact with cloud providers, SaaS providers, and other

APIs, as per: https://www.terraform.io/language/providers

NO.46 You have declared a variable called var.list which is a list of objects that all have an attribute id.

Which options will produce a list of the IDs? (Choose two.)
* { for o in var.list : o => o.id }
* var.list[*].id
* [var.list[*].id]
* [for o in var.list : o.id]
Explanation

https://www.terraform.io/language/expressions/splat

A splat expression provides a more concise way to express a common operation that could otherwise be

performed with a for expression.

NO.47 Which parameters does terraform import require? Choose two correct answers.
* Provider
* Path

* Resource address
* Resource ID

NO.48 Examine the following Terraform configuration, which uses the data source for an AWS AMI.

What value should you enter for the ami argument in the AWS instance resource?

```
data "aws_ami" "ubuntu" {
  ...
}

resource "aws_instance" "web" {
  ami = _____
  instance_type = "t2.micro"

  tags = {
    Name = "HelloWorld"
  }
}
```


* aws_ami.ubuntu
* data.aws_ami.ubuntu
* data.aws_ami.ubuntu.id
* aws_ami.ubuntu.id

resource “aws_instance” “web” {

ami = data.aws_ami.ubuntu.id

NO.49 What does terrafom plan do ?
* Create an execution plan by evaluating the difference between configuration file and state file.
* Performs a refresh, unless explicitly disabled, and then apply the changes that are necessary to achieve

the desired state specified in the configuration files.
* Create an execution plan by evaluating the difference between configuration file and actual

infrastructure.
* Checks whether the execution plan for a set of changes matches your expectations by making changes to

real resources or to the state.

NO.50 Which of the following is not an advantage of using infrastructure as code operations?
* Self-service infrastructure deployment
* Troubleshoot via a Linux diff command
* Public cloud console configuration workflows
* Modify a count parameter to scale resources
* API driven workflows

Explanation

terraform is used to deploy the infrastructure, not to troubleshoot it

NO.51 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.
* Create individual workspaces for each team , and ask them to share modules across workspaces.
* Implement a Private module registry in Terraform cloud , and ask teams to reference them.
* Upgrade to Terraform enterprise , since this is not possible in terraform cloud.
* Upload the modules in the terraform public module registry , and ask teams to reference them

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

NO.52 Provider dependencies are created in several different ways. Select the valid provider dependencies from the following list: (select three)
* Explicit use of a provider block in configuration, optionally including a version constraint.
* Use of any resource belonging to a particular provider in a resource or data block in configuration.
* Existence of any resource instance belonging to a particular provider in the current state.
* Existence of any provider plugins found locally in the working directory.

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

NO.53 does not require GO language to be installed as a prerequisite and it does not require a Windows Server as well.
* False
* True

NO.54 Given the Terraform configuration below, in which order will the resources be created?
* Larger image
* resources will be created simultaneously
* aws_eip will be created first aws_instance will be created second
* aws_instance will be created first aws_eip will be created second

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

NO.55 As a developer, you want to ensure your plugins are up to date with the latest versions. Which Terraform

command should you use?
* terreform providers- upgrade
* terreform apply -upgrade

* terreform refresh -upgrade
* terreformn lnit -upgrade

NO.56 A user creates three workspaces from the command line – prod, dev, and test. Which of the following

commands will the user run to switch to the dev workspace?
* terraform workspace dev
* terraform workspace select dev
* terraform workspace -switch dev
* terraform workspace switch dev

Explanation

The terraform workspace select command is used to choose a different workspace to use for further operations.

https://www.terraform.io/docs/commands/workspace/select.html

NO.57 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?
* Delete the state file every time you run Terraform
* Store the state in an encrypted backend
* Edit your state file to scrub out the sensitive data
* Always store your secrets in a secrets.tfvars file.

NO.58 Named workspaces are not a suitable isolation mechanism for strong separation between staging and production?
* True
* False

Explanation

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

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

NO.59 Which of the following state management command allow you to retrieve a list of resources that are part of the

state file?
* terraform state list
* terraform state view
* terraform view
* terraform list

Explanation

The terraform state list command is used to list resources within a Terraform state.

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

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

given, all resources are listed.

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

NO.60 When writing Terraform code, HashiCorp recommends that you use how many spaces between each nesting

level?
* 0
* 1
* 2
* 4
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.)

NO.61 Consider the following Terraform 0.12 configuration snippet:

1. variable “vpc_cidrs” {

2. type = map

3. default = {

4. us-east-1 = “10.0.0.0/16”

5. us-east-2 = “10.1.0.0/16”

6. us-west-1 = “10.2.0.0/16”

7. us-west-2 = “10.3.0.0/16”

8. }

9. }

10.

11. resource “aws_vpc” “shared” {

12. cidr_block = _____

13. }

How would you define the cidr_block for us-east-1 in the aws_vpc resource using a variable?
* var.vpc_cidrs.0
* vpc_cidrs[“us-east-1”]
* var.vpc_cidrs[“us-east-1”]
* var.vpc_cidrs[0]

NO.62 You want to use terraform import to start managing infrastructure that was not originally provisioned through infrastructure as code. Before you can import the resource’s current state, what must you do in order to prepare to manage these resources using Terraform?
* Run terraform refresh to ensure that the state file has the latest information for existing resources.
* Update the configuration file to include the new resources.
* Shut down or stop using the resources being imported so no changes are inadvertently missed.
* Modify the Terraform state file to add the new resources.
The current implementation of Terraform import can only import resources into the state. It does not generate configuration. A future version of Terraform will also generate configuration.

Because of this, prior to running terraform import it is necessary to write manually a resource configuration block for the resource, to which the imported object will be mapped.

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.

NO.63 A terraform apply can not_____infrastructure.
* import
* provision
* destroy
* change

NO.64 How would you be able to reference an attribute from the vsphere_datacenter data source for use with the

argument within the vsprere_folder resource in the following configuration?

* vsphere_datacenter.dc.id
* data.vsphere_datacenter.dc
* data.dc,id
* data.vsphere_datacenter.dc.id

Prepare Important Exam with TA-002-P Exam Dumps: https://www.vceprep.com/TA-002-P-latest-vce-prep.html]