TA-002-P$^{Q\&As}$

HashiCorp Certified: Terraform Associate

Pass HashiCorp TA-002-P Exam with 100% Guarantee

Free Download Real Questions & Answers **PDF** and **VCE** file from:

https://www.passapply.com/ta-002-p.html

100% Passing Guarantee
100% Money Back Assurance

Following Questions and Answers are all new published by HashiCorp Official Exam Center

- **Instant Download** After Purchase
- **100% Money Back** Guarantee
- **365 Days** Free Update
- **800,000+** Satisfied Customers

QUESTION 1

You need to specify a dependency manually.

What resource meta-parameter can you use to make sure Terraform respects the dependency?

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

A. depends_on

B. PlaceHolder

C. PlaceHolder

D. PlaceHolder

Correct Answer: A

depends_on

QUESTION 2

Which of the following actions are performed during a terraform init?

A. Initializes downloaded and/or installed providers

B. Initializes the backend configuration

C. Provisions the declared resources in your configuration

D. Download the declared providers which are supported by HashiCorp

Correct Answer: ABD

The terraform init command is used to initialize a working directory containing Terraform configuration files. This is the first command that should be run after writing a new Terraform configuration or cloning an existing one from version

control. It is safe to run this command multiple times.

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

terraform init command does

*

Copy a Source Module

*

Backend Initialization

*

Child Module Installation

*

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

QUESTION 3

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

Correct Answer: ABD

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.

QUESTION 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

Correct Answer: D

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

QUESTION 5

Which are examples of infrastructure as code? (Choose two.)

A. Cloned virtual machine images

B. Change management database records

C. Versioned configuration files

D. Docker files

Correct Answer: CD