Q1. You are adding a new variable to your configuration. Which of the following is NOT a valid variable type in Terraform?

1. number
2. bool
3. string
4. float
5. map

Q2. 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)

1. GitHub.com
2. CVS Version Control
3. GitHub Enterprise
4. Azure DevOps Server
5. Bitbucket Cloud

Q3. A user has created three workspaces using the command line - prod, dev, and test. The user wants to create a fourth workspace named stage. Which command will the user execute to accomplish this?

1. terraform workspace -create stage
2. terraform workspace -new stage
3. terraform workspace create stage
4. terraform workspace new stage

Q4. 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?

1. terraform workspace -switch dev
2. terraform workspace dev
3. terraform workspace select dev
4. terraform workspace switch dev

Q5. Oscar is modifying his Terraform configuration file but isn't 100% sure it's correct. He is afraid that changes made could negatively affect production workloads. How can Oscar validate the changes that will be made without impacting existing workloads?

1. run terraform apply using a local-exec provisioner so the configuration won't impact existing workloads
2. run a terraform plan and validate the changes that will be made
3. run terraform refresh to compare his existing configuration file against the current one
4. run a terraform validate to ensure the changes won't impact the production workloads

Q6. In Terraform, most resource dependencies are handled automatically. Which of the following statements describes best how Terraform resource dependencies are handled?

1. Terraform analyzes any expressions within a resource block to find references to other objects and treats those references as implicit ordering requirements when creating, updating, or destroying resources.
2. Resource dependencies are handled automatically by the depends\_on meta\_argument, which is set to true by default.
3. Resource dependencies are identified and maintained in a file called resource.dependencies. Each terraform provider is required to maintain a list of all resource dependencies for the provider and it's included with the plugin during initialization when terraform init is executed. The file is located in the terraform.d folder.
4. The Terraform binary contains a built-in reference map of all defined Terraform resource dependencies. Updates to this dependency map are reflected in Terraform versions. To ensure you are working with the latest resource dependency map you much be running the latest version of Terraform.

Q7. Larissa is interested in using a module to create an AWS VPC. She finds this code but isn't sure what all the declarations are beyond the source and version (such as name, cidr, azs, etc). What are these declarations used for?

1. module "vpc" {

2. source = "terraform-aws-modules/vpc/aws"

3. version = "2.21.0"

4.

5. name = var.vpc\_name

6. cidr = var.vpc\_cidr

7.

8. azs = var.vpc\_azs

9. private\_subnets = var.vpc\_private\_subnets

10. public\_subnets = var.vpc\_public\_subnets

11.

12. enable\_nat\_gateway = var.vpc\_enable\_nat\_gateway

13.

14. tags = var.vpc\_tags}

1. the value of these variables will be obtained from values created within the child module
2. this is where the variable declarations are so Terraform is aware of these variables within the calling module
3. these are variables that are passed into the child module likely used for resource creation
4. these are the outputs that the child module will return

Q8. When using constraint expressions to signify a version of a provider, which of the following are valid provider versions that satisfy the expression found in the following code snippet: (select two)

1. terraform {

2. required\_providers {

3. aws = "~> 1.2.0"

4. }

5. }

1. 1.3.1
2. 1.2.3
3. 1.2.9
4. 1.3.0

Q9. Which are some of the benefits of using Infrastructure as Code in general? (select three)

1. it can be versioned
2. it can be reused
3. it is always platform agnostic
4. it can be shared

Q10. Which of the following Terraform subcommands could be used to remove the lock on the state for the current configuration?

1. state-unlock
2. unlock
3. force-unlock
4. Removing the lock on a state file is not possible

Q11. During a terraform apply, a resource is successfully created but eventually fails during provisioning. What happens to the resource?

1. the resource is marked as tainted
2. it is automatically deleted
3. the terraform plan is rolled back and all provisioned resources are removed
4. Terraform attempts to provision the resource up to three times before exiting with an error

Q12. True or False? Rather than use state, Terraform can inspect cloud resources on every run.

1. False
2. True

Q13. user runs terraform init on their RHEL based server and per the output, two provider plugins are downloaded:

1. $ terraform init

2.

3. Initializing the backend...

4.

5. Initializing provider plugins...

6. - Checking for available provider plugins...

7. - Downloading plugin for provider "aws" (hashicorp/aws) 2.44.0...

8. - Downloading plugin for provider "random" (hashicorp/random) 2.2.1...

9.

10. Terraform has been successfully initialized!

Where are these plugins downloaded and stored on the server?

1. The .terraform.d directory in the current working directory
2. The .terraform/providers directory in the current working directory
3. /etc/terraform/plugins
4. The .terraform.plugins directory in the current working directory

Q14. While Terraform is generally written using the HashiCorp Configuration Language (HCL). What other syntax can Terraform be expressed in?

1. YAML
2. JSON
3. XML
4. TypeScript

Q15. From the code below, identify the implicit dependency:

1. resource "aws\_eip" "public\_ip" {

2. vpc = true

3. instance = aws\_instance.web\_server.id

4. }

5.

6. resource "aws\_instance" "web\_server" {

7. ami = "ami-2757f631"

8. instance\_type = "t2.micro"

9. depends\_on = [aws\_s3\_bucket.company\_data]

10. }

1. The EC2 instance labeled web\_server
2. The EIP with an id of ami-2757f631
3. The S3 bucket labeled company\_data
4. The AMI used for the EC2 instance

Q16. True or False? Each Terraform workspace uses its own state file to manage the infrastructure associated with that particular workspace.

1. True
2. False

Q17. Which of the following is an invalid variable name?

1. var1
2. count
3. web
4. instance\_name

Q18. Which Terraform command will check and report errors within modules, attribute names, and value types to make sure they are syntactically valid and internally consistent?

1. terraform format
2. terraform fmt
3. terraform validate
4. terraform show

Q19. Emma is a Terraform expert, and she has automated all the things with Terraform. During a recent deployment, a virtual machine was deployed but a local script did not work correctly, and therefore needs to be destroyed and recreated. How can Emma easily have Terraform recreate this one resource without having to destroy everything that was created?

1. use terraform state rm aws\_instance.web to remove the resource from the state file, which will cause Terraform to recreate the instance again
2. use terraform refresh to refresh the state and make Terraform aware of the error
3. use terraform apply -replace=aws\_instance.web to mark the virtual machine for replacement
4. use terraform import to import the error so Terraform is aware of the problem

Q20. A "backend" in Terraform determines how state is loaded and how an operation such as apply is executed. Which of the following is not a supported backend type?

1. s3
2. consul
3. terraform enterprise
4. github
5. artifactory