2023

IPSpecialist

SECOND EDITION

TERRAFORM ASSOCIATE CERTIFICATION

Practice Questions

Up to Date Exam Blueprint
Latest exam questions and regularly refreshed content for you to master your exam certification.

Ace Exams with Confidence:
Our precise and comprehensive study material ensures passing grades.

Terraform Associate Certification

Practice Questions

Second Edition

www.ipspecialist.net

Document Control

Proposal Name	:	Terraform Associate Certification – Practice Questions
Document Edition	:	Second Edition
Document Release Date	:	21st September 2023

Registration Office at: Office 32, 19-21 Crawford Street, London W1H 1PJ, United Kingdom
www.ipspecialist.net

Feedback:

If you have any comments regarding the quality of this book or otherwise alter it to suit your needs better, you can contact us through email at info@ipspecialist.net

Please include the book's title and ISBN in your message.

About IPSpecialist

IPSPECIALIST LTD. IS COMMITTED TO EXCELLENCE AND DEDICATED TO YOUR SUCCESS.

Our philosophy is to treat our customers like family. We want you to succeed and are willing to do everything possible to help you make it happen. We have the proof to back up our claims. We strive to accelerate billions of careers with great courses, accessibility, and affordability. We believe continuous learning and knowledge evolution are the most important things to keep re-skilling and up-skilling the world.

Planning and creating a specific goal is where IPSpecialist helps. We can create a career track that suits your visions and develop the competencies you need to become a professional Network Engineer. We can also assist you with the execution and evaluation of your proficiency level based on the career track you choose, as they are customized to fit your specific goals.

We help you STAND OUT through our detailed IP training content packages.

Course Features:

- Self-Paced Learning
    - Learn at your own pace and in your own time
- Covers Complete Exam Blueprint
    - Prep for the exam with confidence
- Case Study-Based Learning
    - Relate the content to real-life scenarios

- Subscriptions that Suit You
    - Get more and pay less with IPS subscriptions
- Career Advisory Services
    - Let the industry experts plan your career journey
- Virtual Labs to test your skills
    - With IPS vRacks, you can evaluate your exam preparations
- Practice Questions
    - Practice questions to measure your preparation standards
- On Request Digital Certification
    - On request: digital certification from IPSpecialist LTD.

About the Authors:

This book has been compiled with the help of multiple professional engineers. These engineers specialize in different fields, e.g., Networking, Security, Cloud, Big Data, IoT, etc. Each engineer develops content in their specialized field that is compiled to form a comprehensive certification guide.

About the Technical Reviewers:

Nouman Ahmed Khan

AWS/Azure/GCP-Architect, CCDE, CCIEx5 (R&S, SP, Security, DC, Wireless), CISSP, CISA, CISM, CRISC, ISO27K-LA is a Solution Architect working with a global telecommunication provider. He works with enterprises, mega-projects, and service providers to help them select the best-fit technology solutions. He also works as a consultant to understand customer business processes and helps select an appropriate technology strategy to support business goals. He has more than eighteen years of experience working with global clients. One of his notable experiences was his tenure with a large managed security services provider, where he was responsible for managing the complete MSSP product portfolio. With his extensive knowledge and expertise in various areas of technology, including cloud computing, network infrastructure, security, and risk management, Nouman has become a trusted advisor for his clients.

Abubakar Saeed

Having started from the grassroots level as an engineer and contributed to the Introduction of Internet in Pakistan and elsewhere, a professional journey of over twenty-nine years in various organizations, national and international. Experienced in leading businesses with a focus on Innovation and Transformation.

He is also experienced in Managing, Consulting, Designing, and implementing projects. Heading Operations, Solutions Design, and Integration. Emphasizing on adhering to Project timelines and delivering as per customer expectations, advocate for adopting technology to simplify operations and enhance efficiency.

Dr. Fahad Abdali

Dr. Fahad Abdali is a seasoned leader with extensive experience in managing diverse businesses. With an impressive twenty years track record, Dr. Abdali brings a wealth of expertise to the table. Holding a bachelor's degree from the NED University of Engineers & Technology and Ph.D. from the University of Karachi, he has consistently demonstrated a deep commitment to academic excellence and professional growth. Driven by a passion for innovation and a keen understanding of industry dynamics, he has successfully navigated complex challenges, driving growth and fostering organizational success.

Mehwish Jawed

Mehwish Jawed is working as a Senior Research Analyst. She holds a Master's and Bachelors of Engineering degree in Telecommunication Engineering from NED University of

Engineering and Technology. She also worked under the supervision of HEC Approved supervisor. She has more than three published papers, including both conference and journal papers. She has a great knowledge of TWDM Passive Optical Network (PON). She also worked as a Project Engineer, Robotic Trainer in a private institute and has research skills in the field of communication networks. She has both technical knowledge and industry-sounding information, which she utilizes effectively when needed. She also has expertise in cloud platforms, like AWS, GCP, Oracle, and Microsoft Azure.

Hareem Khan

Hareem Khan is currently working as a Technical Content Developer, having command over networking and security. She has completed training in CCNA and Cybersecurity, holding a BE in Telecommunications Engineering from the NED University of Engineering and Technology. She has strong knowledge of all the basics of IP and Security Networks and Routing and Switching Protocols.

Free Resources:

For Free Resources: Please visit our website and register to access your desired Resources Or contact us at: info@ipspecialist.net

Career Report: This report is a step-by-step guide for a novice who wants to develop his/her career in computer networks. It answers the following queries:

- What are the current scenarios and prospects?
- Is this industry moving toward saturation, or are new opportunities knocking at the door?
- What will the monetary benefits be?
- Why get certified?
- How to plan, and when will I complete the certifications if I start today?
- Is there any career track I can follow to accomplish the specialization level?

Furthermore, this guide provides a comprehensive career path toward being a specialist in networking and highlights the tracks needed to obtain certification.

IPS Personalized Technical Support for Customers: Good customer service means helping customers efficiently and in a friendly manner. It is essential to be able to handle issues for customers and do your best to ensure they are satisfied. Good service

is one of the most important things that can set our business apart from others.
Excellent customer service will attract more customers and attain maximum customer retention.

IPS offers personalized TECH support to its customers to provide better value for money. If you have any queries related to technology and labs, you can ask our technical team for assistance via Live Chat or Email.

Our Products

Study Guides

IPSpecialist Study Guides are the ideal guides to developing the hands-on skills necessary to pass the exam. Our workbooks cover the official exam blueprint and explain the technology with real-life case study-based labs. The content covered in each workbook consists of individually focused technology topics presented in an easy-to-follow, goal-oriented, step-by-step approach. Every scenario features

detailed breakdowns and thorough verifications to help you completely understand the task and associated technology.

We extensively used mind maps in our workbooks to visually explain the technology. Our workbooks have become a widely used tool to learn and remember information effectively.

Practice Questions

IP Specialists' Practice Questions are dedicatedly designed from a certification exam perspective. The collection of these questions from our Study Guides is prepared to keep the exam blueprint in mind, covering not only important but necessary topics. It is an ideal document to practice and revise your certification.

Exam Cram

Our Exam Cram notes are a concise bundling of condensed notes of the complete exam blueprint. It is an ideal and handy document to help you remember the most important technology concepts related to the certification exam.

Hands-on Labs

IPSpecialist Hands-on Labs are the fastest and easiest way to learn real-world use cases. These labs are carefully designed to prepare you for the certification exams and your next job role. Whether you are starting to learn technology or solving a real-world scenario, our labs will help you learn the core concepts in no time.

IPSpecialist self-paced labs are designed by subject matter experts and provide an opportunity to use products in a variety of pre-

designed scenarios and common use cases, giving you hands-on practice in a simulated environment to help you gain confidence. You have the flexibility to choose from topics and products about which you want to learn more.

Companion Guide

Companion Guides are portable desk guides for the IPSpecialist course materials that users (students, professionals, and experts) can access at any time and from any location. Companion Guides are intended to supplement online course material by assisting users in concentrating on key ideas and planning their study time for quizzes and examinations.

About this Certifications

HashiCorp has launched a new certification program called Terraform Associate. It is for folks with some Terraform experience but not quite ready to take the full-fledged "HashiCorp Certified Engineer" exam. The certification's purpose is to assist applicants in improving their grasp of Terraform and its key ideas so that they may become more successful Terraform users. HashiCorp's suite of tools, which includes Packer, Serf, Consul, and Vault, is a leader in the DevOps sector. As a technical certification, the Terraform Associate exam is an excellent opportunity for applicants to demonstrate their knowledge of HashiCorp's most popular product.

Pre-requisites

- Terminal skills are fundamental.
- Basic knowledge of on-premises and cloud architecture is required.

Objectives

The HashiCorp Terraform Associate Exam is a certification exam that assesses and validates an individual's expertise. HCPAE is another name for it. HashiCorp is in charge of organizing and managing the exam. The exam is designed to evaluate candidates' understanding of a certain set of skills or technology. The candidate's ability to start, provide, manage, govern, and maintain infrastructure on public cloud platforms is assessed in this test. Terraform Associate Dumps includes all the practice questions and answers from the HashiCorp Terraform Associate exam. This certification is intended to assess a candidate's professional abilities and knowledge of the HashiCorp

Terraform software, which allows them to build up, deploy, and manage cloud-native applications in various contexts.

Terraform Associate Certification

Exam Questions	Case study, short answer, repeated answer, MCQs
Number of Questions	50-60
Time to Complete	60 minutes
Exam Fee	70.50 USD

How to Become HashiCorp Certified Terraform Associate

You must pass the HashiCorp Terraform Associate Certification to become a HashiCorp Terraform Associate. The exam is a one-shot test with a two-year certification period. It is made up of only one question. The Associate test is the first step on your path to professional certification. It is intended to be a simplified version of the professional exam, with just enough content to give you a sense of how a true professional-level Terraform project may look. The Assistant test is also intended to be the first step in a more extensive process. We know that most users would not take the complete professional certification immediately; instead, they will start using Terraform on their projects and work their way up as they get more experience.

Recommended Knowledge

- Understand infrastructure as code (IaC) concepts

- Understand Terraform's purpose (vs. other IaC)
- Understand Terraform basics
- Use the Terraform CLI (outside of core workflow)
- Interact with Terraform modules
- Navigate Terraform workflow
- Implement and maintain state
- Read, generate, and modify the configuration
- Understand Terraform Cloud and Enterprise capabilities

The following general domains and their weights on the exam are included in this exam curriculum:

	Domain
Domain 1	Infrastructure as Code (IaC)
Domain 2	Terraform Basics
Domain 3	CLI, modules, and workflow
Domain 4	Configurations
Domain 5	Terraform Cloud and Enterprise capabilities

PRACTICE QUESTIONS

1. IaC is also sometimes known as _____.

A. Infrastructure automation

B. Infrastructure as a service

C. Built-in infrastructure

D. Programmable infrastructure

2. _____ arrangement is faster than manual arrangement.

A. Automated

B. Detailed

C. Hybrid

D. None of the mentioned

3. IaC makes the job of _____ easier.

A. Developers

B. Cloud Engineers

C. IT staff

D. None of the above

4. The low-level syntax of the Terraform language is defined in terms of syntax called?

A. Go

B. Ruby

C. Python

D. HCL

5. What is the main purpose of Terraform?

A. Software development

B. Project management

C. Provide infrastructure

D. None

6. Terraform is a product of _____.

A. Amazon

B. Google

C. HashiCorp

D. None

7. Terraform supports which of the following cloud provider?

A. AWS

B. Oracle

C. GCP

D. All of the above

8. Terraform is a decalartive language.

A. True

B. False

9. IaC automates which of the following processes?

A. Testing process
B. Recovery process
C. Deployment process
D. Both B and C

10. Which of the following is not an advantage of Terraform?

A. Managing multiple cloud providers
B. Human readable configurations
C. Track resources
D. None of the above

11. Which of the following is the disadvantage of IaC?

A. Requires additional tools
B. Fast spread of errors
C. Both A and B
D. None of the above

12. What is meant by Cloud agnostic?

A. A way to develop and support your app on a single cloud provider.
B. A group of cloud evangelists who are unconcerned about specific cloud technology.
C. When a technology is not tied to a single cloud and can function similarly across a variety of cloud environments.
D. A technology embedded in a single public cloud can only be used with that Cloud.

13. IaC enables organizations to _____.

A. Recording and deploying infrastructure code in a repeatable, predictable manner enables better DevOps methods.

B. DevOps managers will be able to hire better engineers for their teams as a result of this tool.

C. Allows engineers to deploy resources using the AWS Management Console as a graphical interface.

D. Provides a solution for enterprises to reduce the number of people working in DevOps departments.

14. What is IaC?

A. On Linux-based systems, a DevOps methodology for optimizing performance.

B. You write this type of code to recover from production outages.

C. A method for aligning all load balancers in a datacenter in the United States.

D. A way for deploying resources in the Cloud and elsewhere using human-readable code.

15. Kubernetes is IaC.

A. True

B. False

16. IaC enables _____.

A. DevOps

B. Unit testing

C. Debugging

D. None of the above

17. The codification of deployment means that it can be tracked in _____.

A. Between the process of testing

B. A website

C. Version control

D. None of the above

18. Which of the following is an example of IaC tools?

A. AWS CloudFormation

B. SaltStack

C. Terraform

D. All of the above

19. Terraform can be used to modify the configuration for _____.

A. Physical networks

B. Servers

C. SDNs

D. All of the above

20. In terraform, you cannot use two cloud providers for high availability.

A. True

B. False

21. Which of the following database providers is supported by Terraform?

A. MySQL

B. Influx DB

C. Mango DB

D. All of the above

22. Which of the following can be categorized as Terraform resources?

A. Compute Instance

B. VM

C. Virtual network

D. All of the above

23. Which of the following statements is NOT true about Terraform?

A. Terraform is a set of tools.

B. Terraform is free.

C. Terraform is cloud agnostic.

D. Terraform enables DevOps.

24. Terraform is _____ tool.

A. Software development

B. Orchestration

C. PaaS

D. None of the above

25. Which of the following are the features of Terraform?

A. Remote Terraform execution

B. Workspaces for organizing infrastructure

C. Version control integration

D. All of the above

26. Which of the following is the correct sequence of Terraform workflow?

A. Write>Plan>Apply

B. Plan>Write>Apply

C. Plan>Apply>Write

D. Apply>Plan>Write

27. Which of the following is considered best practice in the writing phase?

A. Flat file

B. Store file locally on the system

C. Version control system

D. None of the mentioned

28. According to Terraform workflow code, it cannot be modified once written.

A. True

B. False

29. In which of the following phases of Terraform workflow do we deploy changes defined by code to the actual environment?

A. Write

B. Apply

C. Plan

D. Destroy

30. What of the following command initializes terraform directory?

A. terraform apply

B. terraform plan

C. terraform destroy

D. terraform init

31. Which of the following is the first step performed by terraform init?

A. Downloads supporting components.

B. Sets up backend

C. Connects to user

D. None of the above

32. Terraform init can only download modules from the Terraform public registry.

A. True

B. False

33. With Terraform plan command, we can do _____ iteration/s between the write and plan phase of our project

A. Single

B. Any number of

C. Three

D. All of the above

34. Which of the following command reads the code and then creates and shows a plan of deployment?

A. terraform plan

B. terraform apply

C. terraform destroy

D. terraform init

35. Which of the following command is a read-only command?

A. terraform plan

B. terraform apply

C. terraform destroy

D. terraform init

36. Which of the following command is used to deploy code into real infrastructure?

A. terraform plan

B. terraform apply

C. terraform destroy

D. terraform init

37. Which of the following commands is used to remove deployed infrastructure?

A. terraform plan

B. terraform apply

C. terraform destroy

D. terraform init

38. With data block, Terraform creates resources from the scratch

A. True

B. False

39. With _____ Terraform creates and starts tracking resources from scratch.

A. Provider

B. Keyword

C. Arguments

D. Resource block

40. Terraform executes code in files with the _____ extension.

A. .tf

B. .txt

C. .py

D. None

41. Terraform works with which cloud providers?

A. Only Azure and AWS are available.

B. Alibaba, AWS, Azure, and Oracle

C. Only AWS, Azure, and GCP

D. Majority of significant cloud providers and a growing list of common and unusual cloud providers.

42. What are some of the advantages of utilizing Terraform as a tool for IaC?

A. Software-defined networks should be automated.

B. With ease, interacts and manages communication with control-layer APIs.

C. Tracks the status of each deployed resource.

D. All of the above

43. It is never a bad idea to run Terraform multiple times.

A. True

B. False

44. Which of the following statement correctly defines providers?

A. A provider is a plugin to translate API interactions with the service.

B. A provider is a built-in function in Terraform.

C. A provider is a reserved keyword in Terraform.

D. None of the Above

45. _____ define arguments for the provider.

A. Keyword

B. Configuration parameters

C. Functions

D. None

46. The syntax for using the sensitive parameter would be a Boolean value; by default, it is _____.

A. True
A. False
B. Null
C. None of the above

47. In the Destroy provisioner, you are having _____ the value one to the same status .txt file upon being deleted or getting destroyed.

A. Input
B. Output
C. Both of the above
D. None of the above

48. Terraform pulls down the providers when you initialize your different projects using the terraform _____ command.

A. wget
B. init
C. Both of the above
D. None of the above

49. Validation block checks to ensure that the variable will only store values of the typed string if the number of correctors in the string is greater than _____.

A. One

B. Two
C. Three
D. Four

50. You can enable a _____ parameter known as sensitive to prevent Terraform from showing its value during Terraform execution runs.

A. config
B. enable
C. Both of the above
D. None of the above

51. Review the actions performed when you deploy the Terraform code by using the command terraform _____.

A. Plan
B. Apply
C. Run
D. None of the above

52. Terraform state file is stored only remotely.

A. True
B. False

53. Terraform state is stored in the file called _____.

A. terraform.tfstate
B. terraform.pystate
C. terraform.jsonstate
D. None of the mentioned

54. Terraform also tracks the _____ resources deployed.

A. Security policies of

B. Health of

C. Dependencies between

D. Both C and D

55. In which of the following scenarios can we tweak the state outside the workflow to remove or change resources tracked by Terraform?

A. For advanced state management.

B. To see the names of resources managed by Terraform.

C. To see details of resources managed by Terraform.

D. All of the above

56. Which of the following subcommand is used to delete a resource from the Terraform state file?

A. terraform state list

B. terraform state rm

C. terraform state show

D. None

57. By default, Terraform saves state files _____.

A. Locally

B. Remotely

C. Locally and a backup copy on remote services.

D. None

58. Local state storage offers greater availability than remote state storage.

A. True

B. False

59. Remote state storage is more secure because _____.

A. Only the owner of the resource can access it.

B. It cannot be shared in a different region.

C. Due to policies created by cloud vendors

D. All of the above

60. State locking is not supported by _____ backends.

A. AWS S3 storage

B. GCP Storage

C. Hashicorp's console

D. All the remote state storage

61. _____ values can be used by other Terraform projects or code.

A. Variable

B. Output

C. Input

D. None of the above

62. How does Terraform handle dependencies in your infrastructure when deploying or removing resources?

A. The Terraform state file is used to handle them.

B. Each dependence must be manually coded in by the operator (user).

C. For detecting dependencies, Terraform Cloud capabilities like Sentinel are used.

D. It does not deal with dependencies

63. What are the advantages of storing Terraform state remotely?

A. It provides no benefit; it is the same as storing a state locally.

B. Terraform deployments can now be completed more quickly.

C. Granular access, integrity, security, availability, and cooperation are all provided.

D. It provides agility, memory optimization, and automated Terraform code patching.

64. Which command would you use to see all the resources that the Terraform state file has created and is tracking?

A. terraform state list

B. terraform state to show all

C. terraform state rm <name-of-resource>

D. terraform validate

65. What is the purpose of the terraform state mechanism?

A. It enables the synchronization of configuration files between two Terraform projects.

B. It maps real-world resources into Terraform configuration and code.

C. It determines whether your Terraform bug updates are up to date.

D. It allows Terraform to monitor the status of common cloud vendors' services.

66. Where can you set the location of the state file in Terraform code?

A. In the aws_instance resource

B. The terraform remote state resource is used.

C. You cannot specify or change the location of the state file.

D. Using the backend attribute in the terraform block.

67. Terraform supports remote state storage in which of the following cloud storage services?

A. Alibaba Cloud OSS

B. Amazon S3

C. Azure Blob Storage

D. All of the above

68. Which of the following resources can be exposed to teams across regions through remote state storage?

A. VPC IDs

B. NAT instance IDs

C. Subnets

D. All of the above

69. Terraform requires each remote object to be associated with a _____ resource instance/s.

A. Single

B. More than one

C. Up to 5

D. None of the above

70. Terraform keeps a copy of the most recent set of dependents within the state.

A. True

B. False

71. Which of the following is not the function of the backend in Terraform state?

A. Decides where the state is stored

B. Delivering API for state locking

C. Creating firewall rules

D. None

72. Manually retrieving the state from the remote state is possible.

A. True

B. False

73. Any operation that has the potential to _____ state is automatically locked.

A. Read

B. Write

C. Both A and B

D. None

74. Terraform Cloud encrypts the state at rest and encrypts it in transit with _____.

A. AES

B. DES

C. SSL

D. TLS

75. Terraform state cannot store sensitive information.

A. True
B. False

76. _____ command is used to deploy the actions proposed in the plan.

A. terraform plan
B. terraform apply
C. terraform destroy
D. None of the Above

77. The directory which holds the main code in Terraform is called the _____.

A. Root module
B. Child module
C. Public module
D. None of the mentioned

78. Modules can be referenced from which of the following resources?

A. Terraform Public Registry
B. Private Registry
C. Local folder
D. All of the above

79. Word module can be used as a variable in a code.

A. True

B. False

80. What of the following is the correct purpose of **for_each** parameter?

A. Module blocks include count

B. Iterating over a complex variable

C. Set dependencies

D. None

81. Modules can take an _____ inputs.

A. 5

B. Hundreds

C. Arbitrary number of

D. None

82. The terraform module inputs can be used as a _____ inside the module code.

A. Function

B. Variable

C. Dataset

D. All of the above

83. Which of the following is the standard variable reference notation interval; var.server-name.

A. . /module/<variable-name>

B. var.<variable-name>

C. /module/<variable-name>

D. <variable-name>.var

84. The outputs declared inside Terraform module code can be _____ in the root module or your main code.

A. Use as a variable

B. Use as a function

C. Fed back

D. None

85. Which of the following are the uses of output values?

A. Child modules

B. Root modules

C. Both A and B

D. None of the above

86. Which of the following are the key uses of Terraform modules?

A. Reliable retrieval and resolution of dependencies.

B. To enable enterprises to swiftly upgrade their Terraform version.

C. To make Terraform's multi-threaded deployment option available.

D. To avoid reinventing the wheel by making code reusable elsewhere.

87. Which convention should you use in your Terraform code to access an output variable called "returned-variable" declared in a Terraform module called "prod-module"?

A. outputs.returned-variable

B. cannot share variable back.

C. module.prod-module.returned-variable

D. var.returned-variable

88. How can the Terraform module code return outputs that the main Terraform code activating it can use?

A. Run the module code, save the outputs to GitHub, and then reference them in the main code.

B. There is no method to integrate module outputs into the main Terraform code.

C. By including them as environment variables in the main code.

D. In the Terraform module code, output block resources are used.

89. Given the following code:

```
module "my-test-module"
{
source = "./testm"
version = "0.0.5"
region = var.datacenter
}
```

Which attributes in the preceding sample are being passed to the module as an input?

A. region

B. source

C. version

D. This snippet of Terraform code contains no inputs.

90. Terraform code is saved as plain text files with the _____ file extension.

A. .py
B. .docx
C. .hcl
D. .tf

91. Which of the following is true about Terraform Registry?

A. Can be used only in private projects.
B. Only contains child modules
C. Provides a large collection of Terraform modules that are open to the public.
D. None of the above

92. Which of the following is the function of the module block?

A. Lists what type of URLs can be used.
B. Lists meta-arguments
C. Lists syntax to call child module
D. None of the above

93. Which of the following are the examples of meta-arguments?

A. providers
B. depends_on
C. count
D. All of the above

94. Which of the following module block is used to call other modules?

A. output

B. module

C. source

D. None of the above

95. The _____ parameter is preferred for modules from a registry.

A. source

B. count

C. version

D. None

96. Terraform evaluates all module's configuration files.

A. True

B. False

97. The root module for a workspace in _____ is set to the top level of the configuration directory by default.

A. Terraform Cloud

B. Terraform Enterprise

C. Both A and B

D. Terraform CLI

98. When compared with traditional languages, output values can be considered as _____.

A. Function arguments

B. Return values

C. Local variables

D. None

99. Output values make available information about your infrastructure to other Terraform setups.

A. True

B. False

100. _____ locals block can be used to declare a group of linked local values.

A. Couple of

B. Multiple

C. Up to 10

D. Single

101. By default, terraform comes bundled with?

A. All the built-in functions

B. No built-in functions

C. User-defined functions

D. All of the Above

102. In terraform, you can create your own user-defined functions.

A. True

B. False

103. The general syntax for function calls is a function name followed by?

A. parenthesized underscore-separated arguments

B. parenthesized comma-separated arguments

C. parenthesized hyphen-separated arguments

D. None of the above

104. Which of the function is used for inserting files into your resources where applicable

A. file function

B. max function

C. flatten function

D. None of the above

105. Which of the function is used to create a singular list out of a provided set of lists

A. file function

B. max function

C. flatten function

D. None of the above

106. Which of the function is used for determining the max integer value from a provided list.

A. file function

B. max function

C. flatten function

D. None of the above

107. The Terraform console command provides an interactive console for evaluating.

A. Expressions.

B. Equations

C. Boolean

D. None of the above

108. How many type constraints are there?

A. One

B. Two

C. Three

D. None of the above

109. Which of the following type constraints allows for a single type of value to be assigned to a variable, such as a number type, string type, or boolean (or bool) type?

A. Primitive

B. Complex types

C. Any constraint

D. None of the above

110. Which of the following type constraints includes multiple value types in a single variable? Such value types can be constructed using list, tuple, map, or object data structures.

A. Primitive

B. Complex types

C. Any constraint

D. None of the above

111. The complex types can be broken into how many further types?

A. One

B. Two

C. Three

D. None of the above

112. Structural type is a type of primitive. True or False?

A. True

B. False

113. Which of the following allows multiple values of a single or one primitive type to be grouped against a variable?

A. Collection type

B. Structural Type

C. Complex Type

D. None of the above

114. Which of the following allows multiple values of different primitive types to be grouped.

A. Collection type

B. Structural Type

C. Complex Type

D. None of the above

115. Which of the following helps to construct repeatable nested configuration blocks inside Terraform resources.

A. Terraform functions

B. Type Constraints

C. Dynamic Block

D. None of the above

116. The content block defines the body of each generated block and is part of the syntax of?

A. Ingress Block

B. dynamic block.

C. for_each loop

D. None of the above

117. Using dynamic blocks, can we maintain our code easily?

A. True

B. False

118. Given the following snippet of Terraform code:

```
variable "training" {
  type = object({
    name = string
    age = number
  })
  default = {
    name = "Ryan"
    age = 36
  }
}
```

Which of the following type constraints can the variable configured in the code be classified as?

A. Tuple

B. Collection

C. Structural

D. Primitive

119. Collection variable types allow multiple values of one primitive type variable to be grouped together. True or False?

A. True

B. False

120. The dynamic blocks feature in Terraform CANNOT be used with which of the following types of resources?

A. provisioner blocks

B. lifecycle blocks

C. provider blocks

D. resource blocks

121. Which of the following data types represent a primitive type value in Terraform?

A. Map, String, Number

B. Number, String, Boolean

C. Array, String, Float

D. Number, Object, Set

122. What would the output of the following function be, assuming that the value of the variable name is "specialist"?

join("--",["IPS", var.name])

A. IPS_specialist

B. IPS--specialist

C. IPSpecialist

D. Specialist-IP

123. To deploy the code, we use the following command

A. terraform plan

B. terraform apply --auto-approve

C. terraform validate

D. None of the above

124. In structural type, you can have an object that has values of type strings, bool, or even numbers. And you can group similar variables using the tuple or set variable types.

A. True

B. False

125. Which of the following function generates a string by concatenating all elements of a given list of strings with the specified delimiter?

A. max

B. timestamp

C. join

D. None of the above

126. What does the Terraform Registry consist of?

A. Publicly available Terraform providers

B. Publicly available modules

C. Images similar to Docker

D. Both A and B

127. What is the purpose of Terraform Cloud?

A. It manages your Terraform configuration in a fluid, ever-changing environment.

B. Easy access to shared state and secret data.

C. It helps teams use Terraform together.

D. Both B and C

128. What are the benefits of using HashiCorp Sentinel with your Terraform deployments?

A. It can execute multiple terraform apply commands in the backend to speed up deployments.

B. It can optimize your code and format it to make it look better.

C. It can store your deployment information in the cloud to keep it secure.

D. It can help make your deployment more secure and act as protection against accidental deployments.

129. What is the Terraform public registry?

A. A repository of publicly available Terraform providers and modules.

B. A Linux package repository that contains Terraform binaries.

C. A paid access repository that has proprietary modules and providers.

D. A public repository of all the Terraform projects that HashiCorp is working on.

130. Which of the following statements is NOT accurate about the difference between open-source Terraform workspaces and Terraform Cloud workspaces?

A. Variable values are stored in Terraform local workspaces in .tfvars files or in a shell environment.

B. Open-source Terraform workspaces can automatically back up your configuration to Terraform Cloud.

C. State can be stored on disk or in a remote backend for open-source Terraform workspaces.

D. Terraform Cloud workspaces store the Terraform configuration in a linked version control repository.

131. How can HashiCorp Vault help secure your Terraform deployments?

A. It can execute multiple terraform apply commands in the backend to speed up deployments.

B. It can store your long-lived credentials in a secure way and dynamically inject short-lived, temporary keys to Terraform at deployment.

C. It can optimize your code and format it to make it look better.

D. It can store your deployment information in the cloud to keep it secure.

132. Which one of the following answers is NOT a key benefit of Terraform Cloud?

A. Remote Terraform execution

B. Cloud cost estimation

C. Remote state management

D. A built-in version control similar to GitHub

133. What does the Terraform Vault provider offer Terraform users?

A. Allows you to store sensitive data securely that can be used for your Terraform configurations.

B. A secure place to manage access to the secrets for your Terraform configurations and integrate with other popular cloud vendors.

C. Provides short-lived, temporary credentials for users with only the permissions needed for infrastructure creation.

D. All of the above

134. Which of the following is a framework that enforces adherence to policies within your Terraform code?

A. Vault

B. HashiCorp Sentinel

C. Terraform OSS

135. Terraform Code Sentinel has its own language, in which you write policies in.

A. True

B. False

136. Which of the policy ensures that no security groups in AWS openly allow traffic from all IP addresses on port 22

A. CIS

B. Vault

C. Sentinel

137. Which of the following is used to securely store sensitive data and provide short-lived, temporary credentials to users instead of actual long-lived credentials, like AWS CLI access keys?

A. HashiCorp Vault
B. HashiCorp Sentinel
C. TerrraformOSS
D. None of the above

138. With HashiCorp Sentinel, you can get all the secrets, centrally manage access to them, and integrate with Terraform and other popular cloud vendors.

A. True
B. False

139. Vault is pretty flexible.

A. True
B. False

140. Developers and end-users do not need long-lived credentials on their machines and do not open up larger attack surface areas.

A. True
B. False

141. Access to encrypted data inside the Vault can be controlled via fine-grained ACLs within the Vault.

A. True
B. False

142. HashiCorp Sentinel is a framework that enforces adherence to policies within your Terraform code

A. True
B. False

143. What is meant by Terraform Cloud Workspaces?

A. The workspace feature in Terraform Cloud does the same thing as the open-source
B. Terraform workspace feature is slightly different in the sense that it is all hosted in the cloud instead of your local system.
C. You can interact with the Cloud workspace using APIs as well
D. All of the above

144. What is meant by Terraform Registry?

A. The registry is a repository of publicly available Terraform providers and modules
B. It is an integral part of Terraform's adoption by the masses because it's publicly accessible to anyone using Terraform
C. Both A and B
D. None of the above

145. What is meant by HashiCorp Vault?

A. HashiCorp Vault is secret management software.
B. It is used to store sensitive data securely and provide short-lived, temporary credentials to users in place of actual long-lived credentials, like AWS CLI access keys
C. Both A and B
D. None of the above

146. Which of the following statement is true about codification?

A. It codifies the process of security enforcement in Terraform code

B. It can help standardize security testing and automation right into your Terraform deployment pipeline, as it automatically runs before your Terraform deployments.

C. Both A and B

D. None of the above

147. Which of the following statement is true about Testing and Automation?

A. It automatically runs before your Terraform deployments.

B. It can help standardize security testing and automation right into your Terraform deployment pipeline

C. Both A and B

D. None of the above

148. It is possible to use sentinel policies with sandbox Terraform deployments.

A. True

B. False

149. Which of the following are the benefits of Sentinel?

A. Sandboxing-Guardrails for automation

B. Codification

C. Testing and Automation

D. All of the above

150. You can apply Sentinel policies against your Terraform code to sandbox your deployments.

A. True
B. False

151. What does the terraform fmt command do?

A. It formats your Terraform code for readability and consistency.
B. It deletes cached files from Terraform deployments.
C. It pulls down the dependencies required by your code.
D. It reviews the deployment plan and afterward formats the code.

152. Which environment variable can you set to show the most verbose debug logs possible when running Terraform commands?

A. TF_LOG=LOW
B. TF_LOG_EXTREME=DEBUG
C. TF_LOG=TRACE
D. TF_LOG_PATH=./home/acg

153. Which of the following commands will allow you to change your current workspace to an already existing workspace named "production"?

A. terraform workspace init production
B. terraform workspace select production
C. terraform state show production
D. terraform workspace new production

154. What is the purpose of the terraform import command?

A. It imports and installs the latest version of Terraform.

B. It brings external, unmanaged resources into your Terraform configuration to be tracked and managed by it.

C. It exports a resource created by Terraform to a cloud provider of your choice.

D. It imports variables from GitHub to be used within your Terraform code.

155. When working locally, Terraform always starts off with a single workspace called _____ that cannot be deleted.

A. Default

B. New

C. Test

D. None of the above

156. One of your coworkers is new to Terraform and would like to create a workspace called new-hire. Which of the following commands should he use?

A. terraform workspace init new-hire

B. terraform workspace new-hire

C. terraform workspace -new new-hire

D. terraform workspace new new-hire

157. John is new to Terraform and wants to enable extensive logging so that he can access all the information he needs.
What is the environment variable he needs to change?

A. TF_var_log
B. TF_Debug
C. TF_LOG
D. TF_help

158. Which option will you choose to run provisioners that are not linked to any resources?
A. null_resource
B. local-exec
C. remote-exec
D. salt-masterless

159. What version of Google Cloud is Terraform using as a provider?
Google = "~> **1.9.0**"
A. 1.0.0
B. 1.9.1
C. 1.9.2
D. 1.8.0

160. Terraform examines the code and appends any missing arguments before applying the terraform plan.
A. False
B. True

161. Is it possible to add/allow numerous state files for a single configuration using Terraform workspaces?
A. False

B. True

162. Is the standard backend type of Terraform compatible with a remote management system?
A. False
B. True

163. Do the state files get updated when you use the terraform refresh command?
A. False
B. True

164. To start the Terraform console, what command do you use?
A. terraform consul
B. terraform apply -config
C. terraform console
D. terraform plan

165. Which of the following aids users in the deployment of policy as code?
A. Workspaces
B. Resources
C. Functions
D. Sentinel

166. You have been told to abandon static values in favor of more dynamic programming. How are you going to do it? Choose the

appropriate choice from the list below.

A. Input variables
B. Depends_on
C. Functions
D. Local values

167. What is the maximum number of concurrent operations can the terraform apply command support?

A. 1
B. 5
C. 10
D. 100

168. You are attempting to access Terraform Enterprise. To save the API token, which of the following commands is used?

A. terraform API-get
B. terraform cloud – get api
C. terraform get
D. terraform login

169. What are the two backend types that Terraform supports?

A. Enhanced
B. Remote-backend
C. Standard
D. Local-backend

170. Is the terraform state-unlock command used to unlock the state file that has been locked?

A. True
B. False

171. The remote-exec provisioner supports SMB (Server Message Block) and RDP (Remote Desktop) connection types. Is this statement true or false?
A. False
B. True

172. The terraform init command downloads community providers automatically. Is this statement true or false?
A. False
B. True

173. You can scale the resources by incrementing the number using the count meta-argument.
A. False
B. True

174. Which of the following does not qualify as a module source option?
A. HTTP URLs
B. Bit buckets
C. Terraform registry
D. Local Path
E. BLOB storage

175. Which of the following commands can be used to syntactically check the terraform settings before applying or planning?

A. terraform validate
B. terraform check
C. terraform show
D. terraform fmt

176. The state file becomes locked when many team members are working on the same state file. How do you get rid of the lock?

A. terraform force-unlock LOCK_ID
B. terraform force-unlock STATE_FILE
C. terraform unlock LOCK_ID
D. terraform force-unlock=true

177. To gain maximum verbosity in terraform logs, which of the following log commands should be set?

A. set the TF_LOG=TRACE in environment variable
B. set the TF_LOG=INFO in environment variable
C. set the TF_LOG=DEBUG in environment variable
D. set the TF_LOG=WARN in environment variable

178. A user wishes to see a list of all Terraform-deployed resources. How can this be accomplished?

A. terraform state show
B. terraform show list
C. terraform show
D. terraform state list

179. With the terraform apply command, which of the following flags can be used?

A. init

B. get

C. auto-approve

D. console

180. What is the purpose of using the terraform taint command?

A. When you want Terraform to destroy a resource on the next application, use this command.

B. When Terraform should ignore a resource on the next **applywrong**

C. When you want Terraform to destroy and regenerate a resource on the next application, use the **applycorrect** command.

D. When you want Terraform to destroy all of your workspace's infrastructure

181. Which provisioner starts a process on the Terraform-created resource?

A. null-exec

B. local-exec

C. file

D. remote-exec

182. What is the advantage of the private Terraform Cloud/Terraform Enterprise module registry over the public

Terraform Module Registry?

A. The ability to exchange modules with other Terraform users and Terraform Enterprise Organizations members.

B. The option to tag modules by release or version is incorrect.

C. Modules can only be used by members of Terraform Cloud or Enterprise organizations.

D. The ability to share modules with any Terraformcorrect user publicly.

183. Terraform configurations in Terraform 0.13 and later always refer to providers by their local names outside of the required providers block. True or False?

A. False

B. True

184. What is the location of the Terraform local backend's state?

A. In the terraform.tfvars file

B. In the user's .terraformrc file

C. In the terraform.tfstate file

D. In the /tmp directory

185. Which of the following is best for safeguarding sensitive data in state files?

A. Secure Socket Layer (SSL)

B. Enhanced remote backends

C. Signed Terraform providers

D. Blockchain

186. You wish to apply formatting standards for readability to many team members working on infrastructure as code (IaC) using Terraform.
What is the best way to prepare Terraform HCL (HashiCorp Configuration Language) code in accordance with Terraform style conventions?

A. During the linting code phase of your CI/CD process, run the terraform fmt command.

B. Each team should choose one individual to examine and prepare everyone's code.

C. Create a shell script that uses AWK, Python, and sed to modify Terraform files.

D. In every Terraform file (*.tf), manually indent two spaces and align equal sign "=" letters.

187. Providers for Terraform are always downloaded from the internet. True or False?

A. True
B. False

188. What sets the Terraform remote backend apart from existing state backends like S3, Consul, and others?

A. Only paying customers to get access to it.

B. It does not display the results of a terraform apply locally command.

C. It can run Terraform runs on dedicated on-premises infrastructure or in Terraform Cloud.

D. All of the above

189. When you run Terraform for the first time in a configuration directory, what command does it require?

A. terraform init

B. terraform workspace

C. terraform import

D. terraform plan

190. What data is stored in the default "local" Terraform backend?

A. Provider plugins

B. tfplan files

C. Terraform binary

D. State file

191. Where do you specify a state backend in your Terraform configuration?

A. The datasource block

B. The provider block

C. The terraform block

D. The resource block

192. What features does Terraform already come with? (Select three responses)

A. max()

B. regex()

C. alltrue()

D. delete()

193. Which of the following is not a legitimate Terraform string function?

A. split()

B. join()

C. slice()

D. chomp()

194. Variables with a key/value type structure must be entered. In this use scenario, what kind of variable would be employed?

A. list

B. tuple

C. map

D. set

195. To hold data about servers, primarily server name (of type string) and memory size (of type number), you wish to define a single input

variable. Which type of variable should you pick?

A. list

B. map

C. object

D. set

196. What are some acceptable Terraform Structural types? (Select two responses)

A. optional

B. object

C. pair

D. tuple

197. Which of the following does not qualify as a Terraform Collection?

A. list

B. map

C. tree

D. set

198. You are expanding your configuration by including a new variable. Which of the following does Terraform not accept as a primitive variable type?

A. string

B. number

C. float

D. bool

199. Which version restriction should be used to determine the versions that each provider's lower and upper bounds can have? Another name for this constraint operator is pessimistic.

A. >=

B. ~>

C. !=

D. <>

200. Which Terraform feature is best suited for handling minute variations between several contexts, such as development and production?

A. Workspaces

B. States

C. Repositories

D. Versions

201. To manage the infrastructure connected to each workspace, the Terraform CLI employs a state file for each workspace.

A. True

B. False

202. To ensure that the organization-wide standard for naming convention in Terraform Configuration is followed, you should implement a sentinel policy. To access Terraform Configuration, which sentinel import should be used?

A. tfplan

B. tfconfig

C. tfstate

D. tflan

203. Sentinel Policy Rules are defined by imports available through Terraform Cloud. Of the following, which is not an authorized import?

A. tfplan

B. tfconfig

C. tfstate

D. tfapply

204. Which of the following features enables Terraform users to enforce standard configurations for resources deployed using infrastructure as code using policy as code?

A. Resources

B. Functions

C. Sentinel

D. Workspaces

205. What three meta-arguments can a module use besides source and version? (Select three responses)

A. for_each

B. count

C. max

D. depends_on

206. Which of the following must be provided as the calling argument for a child module?

A. version

B. source

C. providers

D. depends_on

207. Which of the following names for a Terraform variable is acceptable?

A. 1234

B. 1_aws_vpc

C. invalid

D. count

208. You must activate verbose logging after experiencing Terraform problems to help with error troubleshooting. The MOST verbose logging is provided by which of the following values?

A. TRACE

B. DEBUG

C. WARN

D. INFO

209. Can a local value in Terraform refer to other local values in Terraform?

A. True

B. False

210. Which of the following is unacceptable when defining an input variable in Terraform?

A. default

B. type

C. description

D. validation

E. sensitive

F. nullable

G. depends_on

211. Only when the depends_on option is explicitly set for the dependent resources can Terraform manage resource dependencies.

A. True

B. False

212. Which provider configuration can be used to define numerous Amazon Web Services providers in various regions?

A. provider

B. source

C. region

D. alias

213. Official Terraform providers are owned and maintained by HashiCorp.

A. True

B. False

214. What are the two acceptable values for "on_failure"-specified provisioners? (Select two responses)

A. continue

B. fail

C. abort

D. retry

215. Which provisioner invokes a process on the resource created by Terraform?

A. null_resource

B. file

C. local-exec

D. remote-exec

216. Which provisioner starts a procedure on the Terraform-running machine rather than the resource?

A. null_resource

B. file

C. local-exec

D. remote-exec

217. What kinds of connections does the file provisioner support? Pick all the available options.

A. ssh

B. sftp

C. winrm

D. rdc

218. Which provisioner moves files or directories from the Terraform-running computer to the resource that was just created?

A. null_resource

B. file

C. local-exec

D. remote-exec

219. Which choice will you make to execute provisioners that aren't connected to any resources?

A. null_resource

B. file

C. local-exec

D. remote-exec

220. Provisioners should only be used as a last resort.

A. True

B. False

221. The Terraform backend can be moved only if no resources are being handled.

A. False

B. True

222. What kind of backend does not allow state locking among the following?

A. local

B. s3

C. remote

D. artifactory

223. All common backend types support state locking, encryption at rest, and remote state storage.

A. True

B. False

224. Which of the upcoming backend types is NOT supported?

A. Console

B. GitHub

C. Local

D. S3

225. Which of the following types of backends may Terraform configure?

A. local

B. standard

C. enhanced

D. advanced

226. What two configuration variables are available to a default local backend? (Choose two answers)

A. path

B. workspace_dir

C. working_dir

D. path_dest

227. You have recently observed that your CI/CD pipeline's Terraform jobs are failing. The returned error makes some reference to exceeding a rate restriction. What might you give into the terraform apply command to sluggish your operations without changing the duration of the builds?

A. -parallelism={NUMBER_OF_OPERATIONS}

B. -concurrent={NUMBER_OF_OPERATIONS}

C. -rate-limit={NUMBER_OF_OPERATIONS}

D. -refresh=false

228. Terraform uses parallelism to speed up the resource provisioning process. How many resources will Terraform automatically provide at once during a terraform apply?

A. 100

B. 10

C. 5

D. 1

229. You recently introduced a new set of resources to your configuration, and you only want to use the terraform plan command to access them. When viewing their plans using the terraform plan command, what flag should you specify?

A. -target={resources}

B. -refresh=true

C. -state={new_state_file}

D. -lock=true

230. Which terraform state subcommands will provide you with all of your state's resources?

A. list

B. show

C. refresh

D. apply

231. In the Terraform Workflow, the terraform plan command must be executed before terraform apply.

A. True

B. False

232. The member of your team is concerned that running the terraform fmt command on their current directory may significantly alter their configuration files. Which flag should they enter into the command to recognize the differences?

A. -diff

B. -check

C. -refresh

D. -list=true

233. State locking is supported by which of the following Terraform backend types?

A. consul

B. kubernetes

C. s3

D. All of the above

234. Terraform may evaluate cloud resources on each run to confirm that the actual resources match the expected state without needing a state file.

A. True

B. False

235. If used as variables in the Terraform code, usernames and passwords will be stored in the state file as plain text.

A. True

B. False

236. The Terraform state always matches the remote cloud resources defined in the configuration?

A. True

B. False

237. A binary created statically using the _____ programming language called Terraform Core.

A. Java

B. C#

C. Python

D. Go

238. Terraform keeps track of the compatible versions of dependencies like providers and modules through a dependency lock file. What does the file's name mean?

A. .terraform.lock.hcl

B. .terraform.lock.tf

C. .dependency.lock.hcl

D. .dependency.lock.tf

239. Which Terraform files from the list below should Git ignore when adding code to a repository? (Choose two)

A. output.tf

B. terraform.tfstate

C. terraform.tfvars

D. variables.tf

240. There are several different platforms where Terraform is offered as a single binary. Which platform from the list below is not supported?

A. Solaris

B. AIX

C. FreeBSD

D. OpenBSD

241. How many spaces between each nesting level should you use while developing Terraform code, according to HashiCorp?

A. 4

B. 2

C. 1

D. 5

242. Which of the following languages does Terraform configuration support?

A. XML

B. JSON

C. Hashicorp Configuration Language (HCL)

D. YAML

243. Which of the following is NOT an authorized block type for Terraform?

A. provider

B. resource

C. output

D. module

E. data

F. bucket

244. Does Terraform facilitate the deployment of numerous clouds?

A. True

B. False

245. What benefit do Terraform and other Infrastructure as Code tools offer?

A. Manage and track infrastructure

B. Automate infrastructure changes

C. Reusable configuration

D. Collaboration using VCS (version control system)

E. All of the above

246. Which of the subsequent best sums up Terraform?

A. A programming language

B. An infrastructure as code (IaC) tool

C. A cloud provider

D. A containerization tool

247. What benefit does using Terraform as an IaC tool have?

A. Multiple cloud platforms' infrastructure can be managed with Terraform

B. You can monitor resource changes during your deployments with Terraform's state

C. To cooperate on infrastructure securely, you can commit your configurations to version control

D. All of the above

248. What are a few advantages of implementing infrastructure as code in a company? (Choose three)

A. IaC is written using an imperative method, in which particular commands must be carried out in the proper order

B. IaC makes use of a human-readable configuration language to facilitate the speedy writing of infrastructure code

C. IaC enables you to securely collaborate on infrastructure by committing your configurations to version control

D. It is possible to manage infrastructure across several cloud platforms using IaC code

249. Which advantage of utilizing Infrastructure as Code with Terraform is NOT one?

A. To cooperate on infrastructure securely, you can commit your configurations to version control

B. Control the infrastructure across several cloud platforms

C. Reducing vulnerabilities in your applications that are visible to the public

D. You can develop infrastructure code more quickly with the help of the human-readable configuration language

250. What is the primary workflow for Terraform?

A. Plan, write, apply

B. Write, plan, apply

C. Apply, write, plan

D. Apply, plan, write

251. What are the three attributes of Terraform Cloud? (Select three responses)

A. Remote state management

B. Remote Terraform Execution

C. Private Module Registry

D. Terraform Linting

252. Which options are unavailable in Terraform OSS/CLI and Terraform Cloud Free Tier?

A. Audit Logging

B. Policy as code (Sentinel)

C. Workspaces

D. Public Module Registry

E. Single Sign-On (SSO)

253. Which Terraform functionality is available only in the Enterprise edition and not in the Terraform Cloud for Business edition?

A. Application-level logging

B. SSO (Single Sign On)

C. Drift detection

D. Audit logging

254. Your boss has requested that you develop a new cloud automation provider that incorporates support for a Private Module registry into the service. What plan and cloud provider do you select?

A. Terraform Cloud with a Terraform Enterprise

B. Amazon Web Services and the Enterprise Terraform Plus Plan

C. The Azure Supercharged Automation Professional Direct plan from Microsoft

D. The Google GCP Terraform Deluxe Plan

255. Which file does a Terraform configuration commonly use to define resources?

A. main.tf

B. terraform.tfvars

C. variables.tf

D. outputs.tf

256. What is the procedure for using Terraform to install new infrastructure?

A. Generate a terraform plan to incorporate the existing infrastructure into the state file, modify the code as necessary, and execute a terraform apply to enact the infrastructure modifications

B. Craft a Terraform configuration, utilize the command "terraform show" to inspect the suggested alterations, and employ "terraform apply" to establish fresh infrastructure

C. Utilize "terraform import" to bring the present infrastructure into the state file, implement code adjustments, and then execute "terraform apply" to enhance the infrastructure accordingly

D. Run terraform init to write a Terraform configuration, terraform plan to view planned infrastructure changes, and terraform apply to create new infrastructure

257. Which of the following descriptions of the Terraform language is the most accurate?

A. The Infrastructure as Code provisioning language Terraform is changeable, imperative, and based on either YAML or the HashCorp Configuration Language

B. The Infrastructure as Code provisioning language Terraform is immutable, declarative, and based on the HashCorp Configuration Language, with the option of JSON

C. Based on the Hashicorp Configuration Language or JSON, Terraform is a declarative, changeable Infrastructure as Code configuration management language

D. Based on the Hashicorp Configuration Language or JSON, Terraform is an immutable, imperative Infrastructure as a Code configuration management language

258. You have been provided with specifications to create a security group for a new application. Your company uses Terraform as a standard. Thus, you want to add this new security group with the fewest possible lines of code. What function could you employ to repeatedly add a list of necessary TCP ports to the new security group?

A. terraform import

B. dynamic blocks

C. splat expression

D. dynamic backend

259. Which of the following sums up a dynamic block the best?

A. Requests that Terraform export the output under the specified local name after reading from the specified data source

B. Defines an object of the specified type with the specified local name

C. Instead of a complex typed value, create nested configuration blocks

D. Exports a value that a configuration or module has exported

260. Which of the following qualifies as an official Terraform plugin?

A. Terraform provisioner

B. Terraform module

C. Terraform provider

D. Terraform registry

261. Terraform builds a dependency graph from the Terraform configurations. Which stage in the graph-building process is NOT correct?

A. Resources are mapped to provisioners if they have any defined

B. Resources are mapped to providers

C. Resources nodes are added to the graph from the configuration

D. Resources are not added to the graph that are no longer present in the configuration but are present in the state file

262. Of the following, which one best sums up a terraform state file?

A. A file with a list of the Terraform providers that are readily available

B. The current status of the terraform-managed infrastructure is kept in a file

C. A list of the terraform modules used in a configuration in a file

D. A file that houses a terraform plan's output

263. You altered your Terraform configuration by injecting certain secrets from variables. What happens once the terraform apply command has been executed and they have been loaded into state?

A. They are displayed in clear text

B. They are displayed as the variables they are referenced by

C. They appear as values that have been encrypted

D. They are left out of the state

264. What is the place where Terraform stores its state by default?

A. The location where Terraform is now running

B. Located in the user's root directory

C. At the exact spot where Terraform is set up. For example, /usr/bin/terraform

D. Located at /.terraform.d/plugins

265. What method should larger teams use to implement Terraform's state?

A. By setting up a remote backend so that several teams can collaborate and be aware of the resources being created and destroyed

B. By storing the state in a cloud instance and allowing team members to SSH into the instance to work on their configuration files

C. Synchronizing your state with a GitHub repository so that other users can compare it

D. You become a member of the group so that you can communicate state file updates by using the daily standup

266. A sizable DevOps team uses Terraform, and numerous changes may be made to your Terraform files throughout the organization. What would you do to ensure that terraform apply locks to the state file?

A. To the command, add the -lock=true parameter

B. Absolutely nothing; Terraform will handle the locking on its own

C. To lock in your suggested modifications, run the terraform plan first. To commit them, run terraform apply

D. Make the command's -state-lock=true argument

267. Terraform will lock your state for any operations that potentially write state if supported by your backend. What is the function of this?

A. Prevents other Terraform developers from making code commits that might replace your improvements

B. Prevents coworkers from manually altering the managed infrastructure

C. This stops someone else from getting the lock and maybe compromising your state

D. Assures that following the initial Terraform apply, the state file cannot be moved

268. You use Terraform to manage the AWS cloud resources, and you want to eliminate all development resources to reduce costs. Your team member has asked you to continue running the Amazon Aurora development instance. How can all cloud resources be destroyed without affecting the database instance?

A. Before executing the terraform destroy command, perform the terraform state rm command to remove the database instance from the terraform state

B. Create a database backup, use terraform destroy, and then restore the backup to reconstruct the database

C. Run a terraform destroy, modify the configuration file to include only database instance, and then run terraform apply

D. Delete the other resource from AWS manually

269. Which typical action doesn't force Terraform to update its state?

A. terraform state list

B. terraform plan

C. terraform apply

D. terraform destroy

270. You recently introduced additional resource blocks from a different supplier to your configuration. What command must be executed before running a terraform plan/apply?

A. terraform plan

B. terraform apply

C. terraform init

D. terraform validate

271. How do you initialize a directory and check out the settings from version control?

A. terraform init -from-module={MODULE-SOURCE}

B. terraform init -source={PATH}

C. terraform init {PATH}

D. terraform init -plugin-dir={PATH}

272. When a new module is added to a configuration, Terraform must download it before using it. What are the two commands for downloading and updating modules? (Choose two)

A. terraform plan

B. terraform refresh

C. terraform init

D. terraform get

273. What Terraform command should I use to generate an execution plan?

A. terraform plan

B. terraform apply

C. terraform init

D. terraform validate

274. What files are checked by terraform plan by default?

A. All *.tf files in the current directory

B. Only files in the .terraform directory

C. Only files you specify with the -file-path flag

D. All files on your hard drive

275. Which Terraform command is used to update infrastructure?

A. terraform destroy

B. terraform apply

C. terraform plan

D. terraform validate

276. Which Terraform command is used to destroy infrastructure?

A. terraform destroy

B. terraform apply

C. terraform plan

D. terraform validate

277. What are the two ways you can remove all of your managed infrastructure? (Choose two)

A. terraform init -destroy

B. terraform apply -destroy

C. terraform destroy

D. terraform plan -destroy

278. Other teams are working on the same configuration. How can you format documents consistently?

A. terraform fmt

B. terraform apply

C. terraform plan

D. terraform validate

279. What Terraform command alters an HCL (HashiCorp Configuration Language) file to follow the suggested HCL file spacing guidelines?

A. terraform fmt

B. terraform apply

C. terraform plan

D. terraform validate

280. You must ensure your Terraform is readable and adheres to the HCL canonical style. You have a main.tf file in the current directory that calls modules located in a modules directory. What command might you issue to quickly rewrite your Terraform in the current directory and all subdirectories to adhere to the HCL style?

A. terraform fmt -check -recursive

B. terraform fmt -diff

C. terraform fmt -check

D. terraform fmt -list=true

281. Which command can check a configuration's syntactic and internal consistency?

A. terraform validate

B. terraform apply

C. terraform plan

D. terraform fmt

282. How could you obtain the terraform validate command's JSON output?

A. terraform validate -json

B. terraform validate json

C. terraform validate -output=json

D. terraform json validate

283. Does the terraform validate command establish connections to external APIs and state when used?

A. It does not, in fact

B. Only if the backend is set up to do so

C. Yes, it does if the -remote=true flag is specified

D. It will try to determine if there are providers set

284. Which command offers an interactive command-line console where expressions can be tested and experimented with?

A. terraform show

B. terraform eval

C. terraform console

D. terraform exec

285. What command reads an output variable's value from the state file?

A. terraform exec

B. terraform show

C. terraform output

D. terraform state

286. You have stored the file terraform.tfvars in the same directory as your Terraform configuration and defined the values for your

variables. Which of the subsequent instructions will incorporate those values into an execution strategy?

A. terraform plan

B. terraform plan -var-file=terraform.tfvars

C. All of the above

D. None of the above

287. Which two phases in the Terraform workflow are necessary to deploy new infrastructure? Select TWO correct responses.

A. terraform init

B. terraform import

C. terraform apply

D. terraform validate

E. terraform destroy

288. Using the Azure console, a few resources were manually generated. It is business policy for you to use Terraform to handle all infrastructure. How can you use Terraform to manage manually deployed resources without affecting others?

A. Run a terraform get to get the manually deployed resources that are not under Terraform management

B. Run terraform apply after deleting the resources that were manually established using the Azure console and adding the new resources in the configuration

C. To import already-existing resources managed by Terraform, use terraform import

D. Resources made outside of Terraform are ineligible for management by Terraform

289. How is terraform import implemented?

A. As a part of terraform init

B. As a part of the terraform plan

C. As a part of terraform refresh

D. By an explicit call

E. All of the above

290. For Terraform to correctly import resources, what must be supplied with the terraform import command?

A. Resource ID, resource type, and the resource name

B. The resource name

C. The complete resource ARN

D. Only resource Id

291. To begin managing the infrastructure not initially provisioned using infrastructure as code, you should use terraform import. What must you do to get ready to manage these resources using Terraform before you can import the resource's present state?

A. Add the new resources that correspond to the resources you want to import to the Terraform configuration file

B. Alter the Terraform state file to include the newly added resources so that Terraform has a record of the resources to be handled

C. Stop utilizing or shutting down the resources being imported so that no alterations are unintentionally overlooked

D. Run terraform apply -refresh-only to make sure the state file contains the most recent data for resources that already exist

292. A user requests a list of every resource deployed with Terraform. How is this accomplished?

A. terraform state show

B. terraform state list

C. terraform show

D. terraform show list

293. What about the Terraform backend is FALSE?

A. Terraform maintains its state data files on a backend

B. Terraform, by default, uses the local backend, which keeps the state in a local file on disk

C. There can only be one backend block provided by a terraform configuration

D. Named values (such as input variables, local variables, or data source properties) may be referred to by a backend block

294. A user requests to view the resource block for the state file resource aws_instance with the name foo. How is this accomplished?

A. terraform show aws_instance.foo

B. terraform show aws_instance foo

C. terraform state show aws_instance.foo

D. terraform state show aws_instance foo

295. Which of the following commands offers the state's JSON representation?

A. terraform state -json

B. terraform state show -json

C. terraform show -json

D. terraform show state –json

296. What is the point of using the terraform taint command?

A. When you want Terraform to compel the destruction of a resource on the subsequent application

B. When you want to make Terraform recreate and destroy a resource on the subsequent application

C. When you want Terraform to disregard a resource during the subsequent application

D. When you want Terraform to obliterate every piece of equipment in your workspace

297. Which command should you use in its place now that terraform.taint has been deprecated in version 0.15.2?

A. terraform apply -replace

B. terraform plan -replace

C. terraform apply -taint

D. terraform plan –taint

298. A single database server deployed with many other resources must be destroyed and recreated using Terraform. You shouldn't alter the Terraform code. What command would be appropriate to carry out this action?

A. terraform apply -replace=aws_instance.database

B. terraform apply -destroy=aws_instance.database

C. terraform state recreate aws_instance.database

D. terraform state destroy aws_instance.database

299. You have a problematic EC2 instance in the cloud. It manages a comparatively modest transitory workload; restarting or destroying it has no adverse effects. What complete command would you use to target this instance for entertainment specifically?

A. terraform apply -replace=aws_instance.{INSTANCE_NAME}

B. terraform apply -replace aws_instance

C. terraform apply -replace {INSTANCE_NAME}

D. terraform destroy --target=aws.instance{INSTANCE_NAME} and terraform apply

300. What doesn't get processed when terraform refresh is run?

A. State file

B. Configuration file

C. Credentials

D. Cloud provider

301. Which Terraform commands will automatically refresh the state unless given additional flags or arguments? Select TWO correct responses.

A. terraform plan

B. terraform state

C. terraform apply

D. terraform validate

E. terraform output

302. Which command is advised to use in its place now that terraform refresh has been deprecated in version 0.15.4? Select TWO correct responses.

A. terraform apply -refresh-only

B. terraform plan -refresh-only

C. terraform apply -refresh

D. terraform plan –refresh

303. Which of the following commands will allow you to check out the adjustments Terraform has picked up during refresh? Select TWO correct responses.

A. terraform apply -refresh-only -auto-approve

B. terraform apply -refresh-only

C. terraform refresh

D. terraform plan -refresh-only

304. What occurs when Terraform configuration is applied? Select TWO correct responses.

A. Any infrastructure changes specified in your setup are implemented using Terraform

B. The plugins needed by the configuration are obtained through Terraform

C. Terraform adds any configuration modifications it makes to the state file

D. Terraform fixes configuration-related formatting issues

E. Terraform completely demolishes your infrastructure and rebuilds it from scratch

305. What distinguishes the Terraform remote backend from state backends like S3, http, consul, etc.?

A. It can carry out Terraform runs using either specialized on-site infrastructure or Terraform Cloud

B. The results of a local terraform apply are not displayed

C. Enterprise clients are the only ones who can use it

D. All of the above

306. In a cloud service, you have a straightforward Terraform configuration with one virtual machine (VM). After executing the terraform apply, the VM is successfully constructed. What would happen if you use the cloud provider console to destroy the VM and then run terraform apply again without making any changes to the Terraform code?

A. Terraform will remove the VM from the state file

B. Terraform will report an error

C. Terraform will not make any changes

D. Terraform will recreate the VM

307. You want to use formatting guidelines for readability when working on infrastructure as code (IaC) with several team members using Terraform. How can Terraform HCL (HashiCorp Configuration Language) code formatted per accepted Terraform style guidelines?

A. Run the terraform fmt command while your CI/CD process is in the code linting stage

B. Each team should assign a member to format and review everyone's code

C. In each Terraform file (*.tf), manually indent by two spaces and align the equal sign ("=")

D. Create a shell script that uses AWK, Python, and sed to modify Terraform files

308. You have launched a new web application with a public IP address on a cloud service provider. You did not provide any outputs for your code, though. How can you quickly determine the IP address of the resource you deployed?

A. To see the result, run terraform output ip_address

B. Use the terraform_remote_state data source to load the state file into a new folder. After that, produce an output for each resource you discover in the state file

C. To determine the resource's name and its properties, including the public IP address, use the terraform state list and terraform state show

D. Run terraform destroy and terraform apply, then check output for the IP address

309. Which command would you run, and with what specific flag, to increase the number of operations Terraform uses concurrently to construct your resources? (Select two answers)

A. terraform apply

B. -parallelism={NUMBER-OF-OPERATIONS}

C. terraform init

D. -concurrent={NUMBER-OF-OPERATIONS}

310. What Terraform command can be used to unlock the current configuration's state?

A. terraform unlock

B. terraform force-unlock

C. Removing the lock on the state file is not possible

D. terraform state unlock

311. What data is stored in the default local Terraform backend?

A. *.tfplan files

B. Terraform binary

C. Provider plugins

D. terraform.tfstate file

ANSWERS

1. Answer: D

Explanation: IaC is an IT infrastructure provisioning method in which systems are designed, managed, and provisioned automatically using code rather than less flexible scripting or a human procedure. That is why IaC is sometimes referred to as programmable infrastructure.

Reference: https://en.wikipedia.org/wiki/Infrastructure_as_code

2. Answer: A

Explanation: By using automated configurations compared to weeks or months of tedious configuration, deployment of each system can often be shortened to a few seconds by reducing the time required to manually specify communication interfaces, alert limits, and how data is to be saved and presented.

Reference: https://stackify.com/what-is-infrastructure-as-code-how-it-works-best-practices-tutorials/

3. Answer: C

Explanation: IaC makes the information technology processes easier, not just a single process.

Reference: https://stackify.com/what-is-infrastructure-as-code-how-it-works-best-practices-tutorials/

4. Answers: D

Explanation: The HashiCorp Configuration Language is the syntax for Terraform configurations (HCL). It aims to achieve a balance between being both human-readable and editable as well as machine-friendly.

Reference:
https://www.terraform.io/language/syntax/configuration

5. Answer: C

Explanation: Terraform is an infrastructure as code (IaC) tool that lets you build, edit, and version infrastructure in a secure and efficient manner. This covers low-level and high-level components, such as compute instances, storage, networking, DNS records, SaaS services, etc.

Reference: https://www.terraform.io/intro/use-cases

6. Answer: C

Explanation: HashiCorp invented Terraform, an open-source infrastructure as code software application. The HashiCorp Configuration Language is a declarative configuration language that allows users to define and deliver data center architecture.

Reference: https://www.terraform.io/

7. Answer: D

Explanation: Terraform supports all the major Cloud providers, including Alibaba, Oracle, AWS, GCP, Azure, and smaller ones.

Reference: https://www.terraform.io/

8. Answer: A

Explanation: Terraform is a declarative language that describes an intended objective rather than the processes to get there. Terraform solely evaluates implicit and explicit associations between resources when establishing an order of operations; the ordering of blocks and the files they are arranged into are often unimportant.

Reference: https://www.terraform.io/intro

9. Answer: D

Explanation: Both your recovery and deployment operations are automated with IaC. It enhances monitoring and testability, lowers the cost of innovation and experimentation, and streamlines deployments. When you do run into challenges, IaC can help you overcome them faster.

Reference:
https://learn.hashicorp.com/tutorials/terraform/infrastructure-as-code

10. Answer: D

Explanation: Terraform is capable of managing infrastructure across a variety of cloud platforms. The human-readable

configuration language aids in the rapid development of infrastructure code. Terraform's state feature allows you to keep track of resource changes as they happen throughout your deployments.

Reference:
https://learn.hashicorp.com/tutorials/terraform/infrastructure-as-code

11. Answer: C

Explanation: Despite its benefits, IaC has potential drawbacks. Additional tools, such as a configuration management system, are required. Errors can swiftly spread across systems. As a result, it is critical to keep an eye on version control and do thorough pre-testing.

Reference:https://docs.microsoft.com/en-us/devops/deliver/what-is-infrastructure-as-code

12. Answer: C

Explanation: Cloud agnostic refers to tools, platforms, or applications that are interoperable with any cloud infrastructure and may be moved between them without causing any problems.

Reference:https://www.cloudzero.com/blog/cloud-agnostic#:~:text=Cloud%20agnostic%20refers%20to%20tools,environments%20without%20any%20operational%20issues.&text=Organizations%20may%20also%20have%20other,vendors%20or%20on%2Dpremise%20infrastructure.

13. Answer: A

Explanation: Infrastructure as Code is a fantastic technique to enable rapid deployments and cross-team cooperation.

Reference:https://docs.microsoft.com/en-us/devops/deliver/what-is-infrastructure-as-code

14. Answer: D

Explanation: Instead of managing your infrastructure resources manually through a user interface, IaC is code that delivers them across numerous platforms.

Reference:https://docs.microsoft.com/en-us/devops/deliver/what-is-infrastructure-as-code

15. Answer: A

Explanation: Kubernetes has been around for a few years and allows you to write your infrastructure as code. This is advantageous in two ways: your infrastructure can now be versioned and committed to a Git repository and easily "deployed" elsewhere.

Reference: https://kubernetes.io/

16. Answer: A

Explanation: Faster speed and consistency: IaC aims to make things faster by removing manual processes and slack in the system, which is the backbone of DevOps.

Reference:https://docs.microsoft.com/en-us/devops/deliver/what-is-infrastructure-as-code

17. Answer: C

Explanation: The codification of deployment means that it can be tracked in version control such as Git. Enabling better visibility and collaboration across teams. Now distributed can work on the same chunk of code that can deploy infrastructure and make sure that they agree on immutable and stored in version control before it is deployed.

Reference: https://www.redhat.com/en/topics/automation/what-is-infrastructure-as-code-iac

18. Answer: D

Explanation: AWS CloudFormation, Red Hat Ansible, Chef, Puppet, SaltStack, and HashiCorp Terraform are examples of infrastructure-as-code tools. Some tools employ a domain-specific language (DSL), while others use YAML or JSON as a standard template format.

Reference: https://en.wikipedia.org/wiki/Infrastructure_as_code

19. Answer: C

Explanation: Control and infrastructure layers are common in SDN systems. Terraform may be used to codify software-defined network setup. By interacting with the control layer, terraform may use this configuration to automatically set up and adjust settings.

Reference: https://www.terraform.io/docs/terraform-tools

20. Answer: B

Explanation: Terraform is your multi-cloud all-in-one solution! Terraform enables you to dynamically create infrastructure across numerous cloud providers, allowing you to maintain your workflows and processes even while the world around you changes. You can use terraform domains in highly available solutions across two public clouds and achieve high availability beyond what a single vendor can offer.

Reference: https://registry.terraform.io/browse/providers

21. Answer: D

Explanation: If you go inside database providers, you will notice that terraform has providers for interfacing with MySQL, Mango DB, and influx DB.

Reference: https://registry.terraform.io/browse/providers

22. Answer: D

Explanation: The most crucial element of the Terraform language is resources. Each resource block describes one or more infrastructure items, such as virtual networks and compute instances, as well as higher-level components like DNS records.

Reference: https://www.terraform.io/intro

23. Answer: B

Explanation: HashiCorp's Terraform Cloud is a commercial SaaS offering. Many of its capabilities, such as remote state storage, remote runs, and VCS connections, are free for small teams. They offer premium solutions for larger teams with additional collaboration and governance capabilities.

Reference: https://www.terraform.io/intro

24. Answer: B

Explanation: Terraform, developed by HashiCorp, is an open-source Infrastructure as Code (IaC) infrastructure orchestrator. The necessity to automate the lifespan of environments is addressed by orchestration.

Reference: https://www.terraform.io/intro

25. Answer: D

Explanation: Terraform Cloud provides a team-oriented remote Terraform process that is easy to understand for new Terraform users and comfortable for established Terraform users. Remote Terraform execution, a workspace-based organizational architecture, version control integration, command-line integration, remote state management with cross-workspace data sharing, and a private Terraform module registry serve as the foundations of this workflow.

Reference: https://www.terraform.io/cloud-docs/overview

26. Answer: A

Explanation: In Terraform workflow, first, we write the code, then we review changes the code will make, and then we deploy the code to real infrastructure. In the writing phase, we generally start with either a version control system as a best practice or a flat-file if you are working individually. In the planning phase, we see what changes the code will make within our actual environment. Finally, you deploy changes to the actual environment and create real resources in the Cloud.

Reference:https://www.terraform.io/intro/core-workflow#:~:text=The%20core%20Terraform%20workflow%20has,Apply%20%2D%20Provision%20reproducible%20infrastructure.

27. Answer: C

Explanation: Version control is considered best practice, so you and your team can collaborate over the issues within your code. In the writing phase, we generally start with either a version control system as a best practice or a flat-file if you are working individually.

Reference:https://www.terraform.io/intro/core-workflow#:~:text=The%20core%20Terraform%20workflow%20has,Apply%20%2D%20Provision%20reproducible%20infrastructure.

28. Answer: B

Explanation: After reviewing the code, you can go back and modify your code. Planning and reviewing changes that our code will make is an important step because, at this point, we are not deploying any infrastructure, but you can see in detail what changes the code will make within our actual environment.

Reference:https://www.terraform.io/intro/core-workflow#:~:text=The%20core%20Terraform%20workflow%20has,Apply%20%2D%20Provision%20reproducible%20infrastructure**.**

29. Answers: B

Explanation: There are four primary stages to the Terraform workflow:

- **Write:** Create the Terraform configuration file specifying the ideal infrastructure state.
- **Init**: Terraform and download the necessary providers and modules.
- **Plan**: Produce an early version of the modifications Terraform will apply to your infrastructure.
- **Apply:** Apply the changes to your infrastructure by doing so.

The actual modifications to your infrastructure are made by Terraform during the Apply phase. It creates, updates, or deletes resources using the execution plan that was developed during the Plan phase.

Reference:https://www.terraform.io/intro/core-workflow#:~:text=The%20core%20Terraform%20workflow%20has,Apply%20%2D%20Provision%20reproducible%20infrastructure**.**

30. Answer: D

Explanation: The terraform init basically initializes the working directory that contains your Terraform. Plan command basically reads the code and then creates and shows a “plan” of deployment. Terraform apply is the final command to realize your code into real

infrastructure being deployed. Terraform destroy command looks at the recorded, stored state file created during deployment and destroys all the resources created by your code.

Reference: https://www.terraform.io/cli/init

31. Answer: A

Explanation: The first step that Terraform init does is to download the supporting or ancillary components required for your code to work. Such as providers that provide the libraries and code for your resources to make API calls to the infrastructure you are deploying into.

Reference: https://www.terraform.io/cli/commands/init

32. Answer: B

Explanation: The terraform init can either download modules or plugins from Terraform public registry over the internet or from your custom URLs where you have uploaded your custom modules written for Terraform.

Reference: https://www.terraform.io/cli/commands/init

33. Answer: B

Explanation: With Terraform plan command, we can do any number of iterations between the write and plan phase of our project. After reviewing the code, you can go back and modify your code. Planning and reviewing changes that our code will make is

an important step because, at this point, we are not deploying any infrastructure, but you can see in detail what changes the code will be made within our actual environment.

Reference:
https://www.terraform.io/cli/commands/plan#:~:text=The%20terraform%20plan%20command%20creates,is%20up%2Dto%2Ddate.

34. Answer: A

Explanation: The plan command basically reads the code and then creates and shows a "plan" of deployment. It allows the user to review the action plan before executing anything. You can look at this plan as a team or individual and decide on final execution. At this point, Terraform also authenticates with the credentials of the platform that you are trying to deploy into.

Reference:
https://www.terraform.io/cli/commands/plan#:~:text=The%20terraform%20plan%20command%20creates,is%20up%2Dto%2Ddate.

35. Answer: A

Explanation: The plan command is a sort of read-only command. It makes API calls in the backend with your preferred platform, but it does not make any changes to your environment.

Reference:
https://www.terraform.io/cli/commands/plan#:~:text=The%20terraform%20plan%20command%20creates,is%20up%2Dto%2Ddate.

36. Answer: B

Explanation: The terraform apply is the command to ensure that your code is realized into real infrastructure being deployed. It also updates the deployment state tracking mechanism known as "state file". Terraform apply creates this state file, which is essential to Terraform working because subsequent commands will come back and look at the state file before making changes to your Terraform infrastructure.

Reference: https://www.terraform.io/cli/commands/apply

37. Answer: C

Explanation: The terraform destroy command looks at the recorded, stored state file created during deployment and destroys all the resources created by your code. Use this command with caution as it is non-reversible command. It is best to take backups and ensure you want to delete infrastructure before using this command.

Reference: https://www.terraform.io/cli/commands/destroy

38. Answer: B

Explanation: With data source Terraform fetches data of an already existing resource environment. The main difference between a data source and a resource block is that a data source block fetches and tracks details of already existing resources.

Reference: https://www.terraform.io/language/data-sources

39. Answer: D

Explanation: With resource block, Terraform creates and starts tracking resources from scratch. The resource configuration arguments change according to which resources you are creating.

Reference: https://www.terraform.io/language/resources/syntax

40. Answer: A

Explanation: The Terraform language uses plain text files with the .tf extension to store code. There is also a JSON-based version of the language with the suffix. tf.

Reference: https://www.terraform.io/language

41. Answer: D

Explanation: The list of providers who are supported is extensive and growing. All of the cloud providers mentioned in these replies, as well as a slew of others, are supported.

Reference: https://registry.terraform.io/browse/providers

42. Answer: D

Explanation: Terraform automates software-defined network deployments and ensures that they are always efficient and consistent. Terraform's intelligent features automate most of the human effort involved in deploying infrastructure via vendor APIs. Terraform is important since it keeps track of the deployment and ensures it is done consistently.

Reference:https://learn.hashicorp.com/tutorials/terraform/infrastructure-as-code

43. Answer: A

Explanation: It is acceptable to repeat this command numerous times to keep the working directory up to date with configuration changes. Even though successive runs may result in errors, this command will never overwrite your current settings or state.

Reference:
https://www.terraform.io/cli/commands#:~:text=The%20usual%20way%20to%20run,automatically%20without%20any%20extra%20arguments**.**

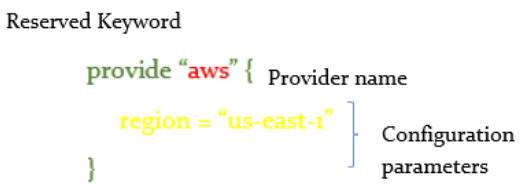
44. Answer: A

Explanation: Terraform employs a provider as a plugin to translate API interactions with the service. Understanding API interactions and exposing resources are the responsibilities of a provider. Terraform can interface with any API; therefore, practically any infrastructure type can be represented as a resource.

Reference: https://www.terraform.io/language/providers

45. Answer: D

Explanation: The configuration parameters help define the arguments for the provider. They vary according to the provider we use.



Reference: h4tps://www.terraform.io/language/providers

46. Answer: B

Explanation

The syntax for using the sensitive parameter would be a Boolean value set to either be true or false; by default, it is false.

Reference:
https://www.terraform.io/language/syntax/configuration

47. Answer: B

Explanation

In the Destroy provisioner, you have to output the value one to the same status .txt file upon being deleted or getting destroyed.

Reference: https://www.terraform.io/cli/commands/destroy

48. Answer: B

Explanation

Terraform pulls down the providers when you initialize your different projects, using The terraform init command.

Reference: https://www.terraform.io/cli/commands/init

49. Answer: D

Explanation

Validation block checks to ensure that this variable will only store values of the typed string if the number of correctors in the string is greater than four.
Reference: https://www.terraform.io/language/values/variables

50. Answer: A

Explanation

You can also enable a config parameter known as sensitive to prevent Terraform from showing its value during Terraform execution runs, the default behavior within Terraform.

Reference: https://www.terraform.io/language/values/variables

51. Answer: A

Explanation

Review the actions performed when deploying the Terraform code using the command terraform plan.
Reference:
https://www.terraform.io/cli/commands/plan#:~:text=The%20terraform%20plan%20command%20creates,is%20up%2Dto%2Ddate.

52. Answer: B

Explanation: By default, Terraform state is stored locally in a file called terraform.tfstate but can be stored remotely in services such as AWS S3. Prior to any modification operation, Terraform refreshes the state file.

Reference:
https://www.terraform.io/language/state#:~:text=This%20state%20is%20stored%20by,make%20changes%20to%20your%20infrastructure.

53. Answer: A

Explanation: Terraform uses this state to map real-world resources to your configuration, keep track of information, and optimize large-scale infrastructure performance. This state is kept in the "terraform. tfstate" file, but it can also be saved remotely, which is more useful in a collaborative environment.
Reference:
https://www.terraform.io/language/state#:~:text=This%20state%20is%20stored%20by,make%20changes%20to%20your%20infrastructure.

54. Answer: C

Explanation: Terraform tracks the dependency between the resources deployed. For example, Terraform must know that it must configure a subnet before deploying a virtual machine in AWS. State file also helps boost deployment performance by acting as a cache for resource attributes.
Reference:
https://learn.hashicorp.com/tutorials/terraform/dependencies?in=terraform/0-13

55. Answers: D

Explanation: Normally, we do not need to mess with the Terraform state file outside the core workflow of Terraform. There are some scenarios in which you might want to tweak the state outside the workflow to remove or change resources tracked by Terraform. It can be used for advanced state management. We can remove resource so that Terraform does not track it. You can use Terraform state commands list subcommand to drag the details of resources and the names of resources Terraform manages.

Reference:
https://learn.hashicorp.com/tutorials/terraform/dependencies?in=terraform/0-13

56. Answer: B

Explanation: A resource can be deleted from the Terraform state file using the rm subcommand for the Terraform state. If you need to remove a resource from Terraform's control or if you mistakenly deleted a resource from the cloud provider, this can be helpful.

When using the terraform state rm subcommand, you must give the resource address of the resource you want to remove. For example, you would use the following command to remove an EC2 instance from the state file:

terraform state rm aws_instance.my_instance

Following the execution of the state rm command, Terraform will delete the resource from the state file.

Reference: https://www.terraform.io/cli/commands/state/rm

57. Answer: A

Explanation: By default, Terraform saves state files locally on the same system that generates these commands. This method is usually used for individual projects or testing purposes.
Reference:
https://www.terraform.io/language/state#:~:text=This%20state%20is%20stored%20by,make%20changes%20to%20your%20infrastructure.

58. Answer: B

Explanation: The advantages of storing files remotely are being able to read files across the distributed teams and better security and availability in the Cloud, which ensure solid backups.
Reference:
https://www.terraform.io/language/settings/backends/local
https://www.terraform.io/language/state/remote

59. Answer: C

Explanation: Different cloud vendors offer different security policies for access to files; you can get granular in who can read and write to the files. This is how a remote state enables collaboration between distributed teams securely.
Reference: https://www.terraform.io/language/state/remote

60. Answer: D

Explanation: State locking locks state file, so another operator who has access to your system or code repository does not end up executing a parallel run of Terraform deployment by mistake. All

remote state storage backends do not support state locking. Some backends supporting state locking include AWS S3, GCP storage, and Hashicorp's console.
Reference: https://www.terraform.io/language/state/locking

61. Answer: B

Explanation: State files also contain the output values in your Terraform code, which means other Terraform projects or codes can use these output values. If a state file is stored remotely is extremely useful for distributed teams working on data by plans, which requires successful execution and are outputs from previous Terraform deployments. Suppose you have a Terraform code that deploys a database. You execute it, deploy infrastructure to the Cloud, and configure Terraform to save the state remotely. Another team that perhaps deploys applications can remotely reference your state file and access output values.
Reference:
https://www.terraform.io/language/state#:~:text=This%20state%20is%20stored%20by,make%20changes%20to%20your%20infrastructure

62. Answer: A

Explanation: The Terraform state file may show a map of all resources and their dependencies. Terraform's correct operation requires it. Prior to any modification operation, Terraform refreshes the state file. Terraform also tracks the dependency between the

resources deployed. For example, Terraform must know that it must configure a subnet before deploying a virtual machine in AWS.
Reference:
https://learn.hashicorp.com/tutorials/terraform/dependencies?in=terraform/0-13#:~:text=Terraform%20uses%20this%20dependency%20information,created%20before%20the%20Elastic%20IP.

63. Answer: C

Explanation: As opposed to local state storage, which does not offer flexibility in collaboration and availability, Terraform offers remote state storage. There are a number of platforms on which you can set Terraform to store state remotely, including AWS S3 storage and GCP Storage. The advantages of storing files remotely are being able to read files across the distributed teams and better security and availability in the Cloud, which ensure solid backups.
Reference: https://www.terraform.io/language/state/remote

64. Answer: A

Explanation: When you use the list with the terraform state command, all of the resources in the Terraform state will be listed.
Reference:
https://www.terraform.io/language/state#:~:text=This%20state%20is%20stored%20by,make%20changes%20to%20your%20infrastructure.

65. Answer: B

Explanation: It is essentially a mapping between real-world resource IDs and configurations to Terraform logical resources.

Reference:
https://www.terraform.io/language/state#:~:text=This%20state%20is%20stored%20by,make%20changes%20to%20your%20infrastructure.

66. Answer: D

Explanation: A unique block called the terraform block is used to configure Terraform. It has options that govern Terraform's operation, such as the state file's backend.

The backend attribute in the terraform block can be used to provide the state file's location in the Terraform code. For example, you would use the following settings to store the state file in an S3 bucket:

```
terraform {
  backend "s3" {
    bucket = "my-bucket"
    key = "terraform.tfstate"
  }
}
```

Additionally, you may set up Terraform to use a remote state backend like Terraform Cloud or Terraform Enterprise by using the backend attribute.

Reference:
https://www.terraform.io/language/state#:~:text=This%20state%20is%20stored%20by,make%20changes%20to%20your%20infrastructure.

67. Answer: D

Explanation: Terraform writes state data to a remote data store with a remote state, which all team members may subsequently share. Terraform Cloud, HashiCorp Consul, Amazon S3, Azure Blob Storage, Google Cloud Storage, Alibaba Cloud OSS, and others all enable storing state. A backend or Terraform Cloud implements a remote state, both of which can be configured in the root module of your setup.
Reference: https://registry.terraform.io/browse/providers

68. Answer: D

Explanation: Using remote state storage, a core infrastructure team can take care of the core machines, networking, etc., while also exposing some information to other teams so that they can manage their own infrastructure. For example, you can expose VPC IDs, subnets, NAT instance IDs, and other Terraform states via remote state and consume other Terraform states.
Reference: https://www.terraform.io/language/state/remote

69. Answer: A

Explanation: Terraform employs its own state structure to translate configuration to real-world resources. Terraform requires each distant object to be bound to exactly one resource instance, which is typically ensured because Terraform is in charge of constructing the objects and maintaining their identities in the state. If you instead choose to import objects produced outside of Terraform, double-

check that each unique object is imported to only one resource instance.
Reference: https://www.terraform.io/language/state/remote

70. Answer: A

Explanation: Terraform keeps a copy of the state's most recent set of dependents to ensure proper execution. Terraform can still figure out the proper order for removing one or more items from the configuration when you delete one or more items from the state.
Reference:
https://www.terraform.io/language/state#:~:text=This%20state%20is%20stored%20by,make%20changes%20to%20your%20infrastructure.

72. Answer: C

Explanation: The backend in Terraform state is in charge of keeping track of the configuration's current state and providing an API for state locking. This enables Terraform to keep track of the resources it oversees and plan modifications across many users and environments.

Firewall rules are not created by the backend. The cloud service provider or another infrastructure management tool can do this role.
Reference: https://www.terraform.io/language/settings/backends

72. Answer: A

Explanation: The terraform state pull command can still be used to manually obtain the state from the remote state. This will load and output your remote state. You can save it to a file or do whatever you want.
Reference: https://www.terraform.io/cli/commands/state/pull

73. Answer: B

Explanation: Terraform will lock your state for any operations that potentially write state if your backend supports it. Outsiders will be unable to gain access to the lock and contaminate your state because of this. All operations with the potential to write state are locked by default. There will be no evidence that this is taking place.
Reference:
https://www.terraform.io/language/state/locking#:~:text=If%20supported%20by%20your%20backend,message%20that%20it%20is%20happening.

74. Answer: D

Explanation: A cryptographic protocol called TLS (Transport Layer Security) offers safe communication over a computer network. It serves as a defense against message forging, manipulation, and eavesdropping while data is in transit.

SSL (Secure Sockets Layer), a more dated encryption system, was replaced by TLS. The recommended protocol for usage with Terraform Cloud is TLS because it is more secure than SSL.

Reference:
https://registry.terraform.io/providers/hashicorp/tls/latest/docs

75. Answer: B

Explanation: Depending on the resources used and your definition of "sensitive," Terraform state can contain sensitive data. All resource IDs and properties are stored in the state. This may contain initial passwords for resources such as databases.
Reference: https://www.terraform.io/plugin/sdkv2/best-practices/sensitive-state

76. **Answer: B**
Explanation: Actions suggested in the plan are implemented using the terraform apply command. In the Terraform workflow, it is the last step.

To create a plan using the terraform plan command, go ahead and use that before using the terraform apply command. When everything is as you had hoped, you may deploy the modifications to your infrastructure by using the Terraform apply command.

In order to get your infrastructure to the correct condition, the terraform apply command will create, update, or delete the appropriate resources. It will also update the Terraform state file and make the necessary modifications.
Reference: https://www.terraform.io/cli/commands/apply

77. **Answer: A**

Explanation: Each Terraform configuration contains at least one root module, which is made up of the resources defined in the .tf files in the primary working directory. A module can call other modules, allowing you to quickly integrate the resources of a child module in the setup.

Reference:
https://www.terraform.io/language/modules/syntax#:~:text=Every%20Terraform%20configuration%20has%20at,configuration%20in%20a%20concise%20way.

78. **Answer: D**

Explanation: Modules can download or reference from the Terraform Public Registry, which contains a collection of all publicly available modules. On referencing modules from Terraform Public Registry, Terraform downloads them and places them in a directory on a system. You can also host your modules in a private registry hosted by yourself or an organization and reference them like a Public Registry. You can use it when your concern is security and closed source code. You can also save the module code in a local folder on your system and reference it using its path.

Reference:
https://www.terraform.io/language/modules/syntax#:~:text=Every%20Terraform%20configuration%20has%20at,configuration%20in%20a%20concise%20way.

79. Answers: B

Explanation: The Module is a reserved keyword. A keyword cannot be used to identify a variable, function, or other object.

Reference:
https://www.terraform.io/language/modules/syntax#:~:text=Every%20Terraform%20configuration%20has%20at,configuration%20in%20a%20concise%20way.

80. Answer: B

Explanation: Few parameters are allowed inside the module block including count, which allows spawning multiple separate instances of the module's resources; for_each parameter, which allows iterating over a complex variable. providers parameter allows you to tie down specific providers to your module. depends_on module allows you to set dependencies for your module.
Reference: https://www.terraform.io/language/meta-arguments/for_each

81. Answer: C

Explanation: Modules can take an arbitrary number of inputs and written outputs back into your main code. Once you have invoked a module using the module block in your code. You can use these outputs that it returns and plug them back into your code.
Reference:
https://www.terraform.io/language/modules/syntax#:~:text=Every%20Terraform%20configuration%20has%20at,configuration%20in%20a%20concise%20way.

82. Answer: B

Explanation: The terraform module inputs are arbitrarily named parameters that you pass inside the module block. These inputs can be used as a variable inside the module code.
Reference:
https://www.terraform.io/language/modules/syntax#:~:text=Every%20Terraform%20configuration%20has%20at,configuration%20in%20a%20concise%20way.

83. Answer: B

Explanation: Terraform module code is just like any other module code you may encounter. You pass the input variable. Once the module consumes this input variable, you will be using this variable using the standard variable reference notation interval; var.<server-name>.
Reference:
https://www.terraform.io/language/modules/syntax#:~:text=Every%20Terraform%20configuration%20has%20at,configuration%20in%20a%20concise%20way

84. Answer: C

Explanation: The outputs declared inside the Terraform module code can be fed back into the root module or your main code. For example, the output invocation convention when using an output returned by a module back inside your main code is the module.
<name-of-module>.<name-of-output>
Reference: https://www.terraform.io/language/values/outputs

85. Answer: C

Explanation: There are various applications for output values:

A parent module can access a portion of a child module's resource properties via outputs.

After running terraform apply, a root module can use outputs to print specific values in the CLI output. Root module outputs can be accessed by other configurations using a terraform_remote_state data source when using a remote state.

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

86. Answer: D

Explanation: Terraform's major method for packaging and reusing resource configurations is through modules. The main purpose of these modules is to reuse the code instead of reinventing the wheel.
Reference: https://www.terraform.io/language/modules/syntax

87. Answer: C

Explanation: The outputs declared inside the Terraform module code can be fed back into the root module or your main code. For example, when using an output returned by a module back inside your main code, the output invocation convention is the module. <name-of-module>.<name-of-output>.
Reference: https://www.terraform.io/language/values

88. Answer: D

Explanation: The outputs indicated by the output block in the Terraform module code can be sent back into the root module or your main code.
Reference: https://www.terraform.io/language/values

89. Answer: C

Explanation: In this example, the datacenter region is passed to the module using the region input. The module can then use this data to produce resources in the designated region.

Using the var keyword, you can pass an input to a module. For example, the following code would be used to pass the value us-east-1 to the region input:

```
module "my-test-module" {
  source = "./testm"
  version = "0.0.5"
  region = var.datacenter
}
variable "datacenter" {
  type = string
}
```

Reference: https://www.terraform.io/language/values/variables

90. Answer: D

Explanation: The Terraform language uses plain text files with the .tf extension to store code. A JSON-based form of the language with the .tf.json file extension is also available. Terraform configuration files are files that contain Terraform code.
Reference: https://www.terraform.io/language

91. Answer: C

Explanation: The Terraform Registry contains a large number of freely available Terraform modules for configuring various types of infrastructure. These modules are free to use, and if you indicate the right source and version in a module call block, Terraform will automatically download them.
Reference: https://registry.terraform.io/browse/providers

92. Answer: C

Explanation: The syntax for invoking a child module from a parent module is documented in Module Blocks. Module Sources describes the types of paths, addresses, and URIs that can be used in a module block's source parameter. Special arguments that can be used with any module are documented in the Meta-Arguments section.
Reference: https://www.terraform.io/language/modules/syntax

93. Answer: D

Explanation: Special arguments for every module, such as providers, depend_on, count, and for each, are documented in the Meta-Arguments section.

Reference: https://www.terraform.io/language/meta-arguments/depends_on

94. Answer: B

Explanation: The module block in Terraform is used to call and use other modules as part of your setup. You can instantiate and set up reusable modules in your Terraform code using this block.
Reference: https://www.terraform.io/language/settings

95. Answer: C

Explanation: We recommend explicitly restricting the permissible version when utilizing modules installed from a module registry to avoid unexpected or undesirable modifications. A version constraint string can be passed to the version argument. Terraform will utilize the most recent installed version of the module that satisfies the constraint; if no suitable versions are available, Terraform will download the most recent version that satisfies the constraint.
Reference: https://www.terraform.io/registry/modules/use

96. Answer: A

Explanation: Terraform assesses all module's configuration files, thereby considering the whole thing as a single document. Separating separate blocks into multiple files is solely for the benefit of readers and maintainers and has no bearing on the module's behavior.
Reference: https://www.terraform.io/language/modules/syntax

97. Answer: C

Explanation: The root module for a workspace defaults to the top level of the configuration directory (provided through the version control repository or direct upload) in Terraform Cloud and Terraform Enterprise, but workspace settings can specify a subdirectory to use instead.

Reference: https://www.terraform.io/enterprise
https://www.terraform.io/cloud-docs

98. Answer: B

Explanation: Comparing Terraform modules to function definitions can be handy if you are familiar with traditional programming languages:

- Function arguments are analogous to input variables.
- Function return values are analogous to output values.

Local values are similar to the temporary local variables of a function.

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

99. Answer: A

Explanation: Output values provide information about your infrastructure to other Terraform setups and make it available on the command line. In computer languages, output values are equivalent to return values.

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

100. Answer: D

Explanation: A single local block can be used to declare a group of linked local values. For example:

```
locals {
 s_name = "forum"
 o_name      = "Community Team"
}
```

Reference: https://www.terraform.io/language/values/locals

101. Answer: A

Explanation:

By default, terraform comes bundled with all the built-in functions. You do not need to use any additional providers or modules to use these functions.

Reference: https://www.terraform.io/intro

102. Answer: B

Explanation:

In terraform, you cannot create your own user-defined functions like in a programming language because the Terraform language does not support user-defined functions; only the built-in functions are available for use.

Reference: https://www.terraform.io/intro

103. Answer: B

Explanation:

The general syntax of the function is just like how you would invoke functions anywhere else in a programming language. You pass the function some arguments and get values in return. The general syntax for function calls is a function name followed by parenthesized comma-separated arguments.

Reference:
https://www.terraform.io/language/syntax/configuration

104. Answer: A

Explanation:

The **file** function returns the contents of a file at the specified path as a string. Because strings in Terraform are Unicode character sequences, the file function will interpret the file contents as UTF-8 encoded text and return the resulting Unicode characters. If the file contains invalid UTF-8 sequences, this function will fail.

Reference: https://www.terraform.io/language/functions/file

105. Answer: C

Explanation:

The **flatten** takes a list and replaces any list elements with a flattened sequence of the list's contents.

Reference: https://www.terraform.io/language/functions/flatten

106. Answer: B

Explanation:

The **max** function will return us the greatest value from the list of values entered.

Reference: https://www.terraform.io/language/functions/max

107. Answer: A

Explanation:

The Terraform console command displays an interactive console where expressions can be evaluated. If the current state of your deployment is empty or has not yet been created, the console can be used to experiment with expression syntax and built-in functions.

Reference: https://www.terraform.io/cli/commands/console

108. Answer: B

Explanation:

Type constraints control the type of variable values that you can pass to your Terraform code. There are two types of constraints:

- Primitive
- Complex

Reference: https://www.terraform.io/language/expressions/type-constraints

109. Answer: A

Explanation:

Primitive constraints allow for a single type of value to be assigned to a variable, such as a number type, string type, or boolean (or

bool) type.

Reference: https://www.terraform.io/language/expressions/type-constraints

110. Answer: B

Explanation:

Complex types constraints include multiple value types in a single variable, which can be constructed using list, tuple, map, or object data structures.

Reference: https://www.terraform.io/language/expressions/type-constraints

111. Answer: B

Explanation:

Complex types include multiple value types in a single variable. Such value types can be constructed using list, tuple, map, or object data structures. The complex types can be broken into two further types.

- Collection
- Structural

Reference: https://www.terraform.io/language/expressions/type-constraints

112. Answer: B

Explanation:

Complex type is divided into two further types: collection and structural type. Primitive does not divide into any further type.

Reference: https://www.terraform.io/language/expressions/type-constraints

113. Answer: A

Explanation:

Collection types allow multiple values of a single or one primitive type to be grouped against a variable. For example, you can have a list of type strings, or you can have a map of type numbers, or you can have a set of type strings, but you cannot mix more than one type against a single variable.

Reference: https://www.terraform.io/language/expressions/type-constraints

114. Answer: B

Explanation:

Structural types allow multiple values of different primitive types to be grouped. So, as opposed to the type of collection, which only allows a single type of value within a variable, the structural type allows more than one type of value assigned within a variable.

Reference: https://www.terraform.io/language/expressions/type-constraints

115. Answer: C

Explanation:

Dynamic blocks help construct repeatable nested configuration blocks inside Terraform resources. They can be used inside

resources, such as resources, data, provider, and provisional blocks inside a Terraform resource.

Reference:

https://www.terraform.io/language/expressions/dynamic-blocks

116. Answer: B

Explanation:

Dynamic blocks help construct repeatable nested configuration blocks inside Terraform resources. The content block defines the body of each generated block and is part of the syntax of a dynamic block.

Reference:

https://www.terraform.io/language/expressions/dynamic-blocks

117. Answer: A

Explanation:

You have the dynamic block keyword, and then you pass it the name of whatever block inside the resource you are trying to replicate. In this case, the nested ingress block is inside the AWS security group resource. This helps us to maintain our code easily.

Reference:

https://www.terraform.io/language/expressions/dynamic-blocks

118. Answer: C

Explanation:

A structural variable type allows multiple values of various primitive types to be grouped together as a single value. In this case, the variable training has 2 sets of values: a string and a number.

Reference: https://k21academy.com/terraform-iac/variables-in-terraform/

119. Answer: A

Explanation:

Collection types allow multiple values of a single or one primitive type to be grouped against a variable. For example, you can have a list of type strings, or you can have a map of type numbers, or you can have a set of type strings, but you cannot mix more than one type against a single variable.

Reference: https://www.terraform.io/language/expressions/type-constraints

120. Answer: B

Explanation:

Dynamic blocks CANNOT be used with lifecycle blocks, as Terraform must process this type of block before it is safe to evaluate expressions.

Reference:
https://www.terraform.io/language/expressions/dynamic-blocks

121. Answer: B

Explanation:

A primitive type includes basic data types that are not comprised of more than one value or of nested data values, such as a numeric value, a sequence of characters, or binary (true/false) logic.
Reference: https://www.terraform.io/language/expressions/types

122. Answer: B

Explanation:

The join function would concatenate both strings ("IPS" and "specialist") using the provided delimiter (--).
Reference: https://www.terraform.io/language/functions/join

123. Answer: B

Explanation:

The terraform apply --auto-approve command is used to deploy the code in terraform. The --auto-approve flag prevents Terraform from explicitly prompting you to enter yes before deploying the code.
Reference:
https://www.terraform.io/cli/commands/apply#:~:text=%2Dauto%2Dapprove%20%2D%20Skips%20interactive,never%20prompt%20in%20that%20case.

124. Answer: A

Explanation:

In structural type, you can have an object that has values of type strings, bool, or even numbers. And you can group similar variables using the tuple or set variable types.

Reference: https://www.terraform.io/language/expressions/type-constraints

125. Answer: C

Explanation:

A join generates a string by concatenating all elements of a given list of strings with the specified delimiter. The join function expects a separator as the first argument, so we will give it the separator comma and then a list of strings we want to concatenate using that separator.

Reference: https://www.terraform.io/language/functions/join

126. Answer: D

Explanation:

Publicly available Terraform providers and Publicly available modules both the registry consists of both publicly available Terraform providers and modules. Images similar to the Docker registry consist of available providers like AWS, Azure, GCP, and modules that allow easy integration in your Terraform configuration.

Reference: https://registry.terraform.io/browse/providers

127. Answer: D

Explanation: The purpose of Terraform Cloud includes:

Option B. Making shared state and confidential data simple to access promotes collaboration and centralises the management of

Terraform state and confidential data.

Option C. Facilitating team collaboration and increasing team productivity when using Terraform by providing tools for teamwork, version control integration, and remote execution of Terraform plans and applications.

Reference: https://www.terraform.io/cloud-docs

128. Answer: D

Explanation:

When provisioning infrastructure, HashiCorp Sentinel, a policy as a code framework, is linked with Terraform to enforce compliance, security, and governance standards. Although it offers a number of advantages, its main goal is to improve security and guarantee that deployments follow preset standards, guarding against unintentional or unauthorized changes to your infrastructure.

Reference: https://www.terraform.io/cloud-docs/sentinel

129. Answer: A

Explanation:

The Terraform public registry links end-users with the providers that power all of Terraform's resource types and/or helps them find modules for quickly deploying common infrastructure configurations.

Reference: https://registry.terraform.io/browse/providers

130. Answer: B

Explanation:

You cannot automatically backup your setup to Terraform Cloud using open-source Terraform workspaces. While Terraform Cloud offers functionality for state management and automated backups as part of its service, Terraform Open-Source requires you to configure and manage your state and backups manually.
Reference: https://www.terraform.io/language/state/workspaces

131. Answer: B

Explanation:

HashiCorp Vault is basically a centralized, secret management software. It can store your long-lived credentials in a secure way and dynamically inject short-lived, temporary keys to Terraform at deployment.
Reference: https://www.terraform.io/intro

132. Answer: D

Explanation:

Though Terraform Cloud integrates and can use repositories like GitHub and BitBucket, it does not have version control built-in.
Reference: https://www.terraform.io/cloud-docs

133. Answer: D

Explanation:

Vault allows you to securely store sensitive data and can be used with all your configurations. This really helps, especially once your infrastructure configurations get large and complicated. Vault allows you to securely store sensitive data and can be used with all your configurations. This also integrates with cloud vendors like AWS, Azure, and GCP, allowing easy configuration across all your cloud vendors. Vault allows you to provide short-lived, temp credentials that allow users to have only the permissions needed to deploy the infrastructure. The credentials will rotate according to the rotation schedule you define. This allows you to not have to worry about keeping up with long-lived credentials.

A built-in version control similar to GitHub Vault allows you to store any sensitive data securely and can be used with all your configurations. This really helps, especially once your infrastructure configurations get large and complicated.

Reference: https://www.terraform.io/language/providers

134. Answer: B

Explanation:

HashiCorp Sentinel is a framework that enforces adherence to policies within your Terraform code. In other words, its code that enforces restrictions on your Terraform Code Sentinel has its own language in which you write policies.

Reference: https://www.terraform.io/cloud-docs/sentinel

135. Answer: A

Explanation:

Sentinel language is the name of the language used by Terraform Code Sentinel (formerly known as Sentinel). Written regulations that can be applied during Terraform deployments are its intended use. With the help of this language, you may create unique IaC policies and rules and enforce them using Terraform workflows.
Reference: https://www.terraform.io/cloud-docs/sentinel

136. Answer: C

Explanation:

Sentinel policy ensures that no security groups in AWS openly allow traffic from all IP addresses on port 22.
Reference: https://www.terraform.io/cloud-docs/sentinel

137. **Answer: A**

Explanation:

HashiCorp Vault stores sensitive data securely and provides short-lived, temporary credentials to users in place of actual long-lived credentials, like AWS CLI access keys. Vault handles rotating these temporarily provided credentials in the backend as per an expiration schedule configurable in Vault.
Reference:
https://registry.terraform.io/providers/hashicorp/vault/latest/docs

138. Answer: B

Explanation:

HashiCorp Sentinel is not intended to handle secrets or control access to secrets from a central location. Its main function, especially when using Terraform, is to enforce policies and compliance requirements within your infrastructure as code (IaC) workflows. Other tools and services, like HashiCorp Vault, which is made expressly for managing secrets and controlling access to secrets, are frequently used in the administration of secrets and access control to secrets.

Reference:
https://registry.terraform.io/providers/hashicorp/vault/latest/docs

139. Answer: A

Explanation:

Vault is pretty flexible. It offers a user interface, CLI and API-based access, and an Enterprise offering for large organizations, allowing for distributed Vault deployments and high availability.

Reference:
https://registry.terraform.io/providers/hashicorp/vault/latest/docs

140. Answer: A

Explanation:

Developers and end-users do not need long-lived credentials on their machines and do not open up larger attack surface areas. Developers do not have access to actual secrets, only short-lived credentials, with only the permissions that they require.

Reference: https://www.balbix.com/insights/attack-vectors-and-breach-methods/

141. Answer: A

Explanation:

HashiCorp Vault is secret management software. It stores sensitive data securely and provides short-lived, temporary credentials to users in place of actual long-lived credentials, like AWS CLI access keys. Vault handles rotating these temporarily provided credentials in the backend as per an expiration schedule configurable in Vault. It can also generate cryptographic keys to encrypt data at rest and while in transit. Access to encrypted data inside the Vault can be controlled via fine-grained ACLs within the Vault.

Reference:
https://registry.terraform.io/providers/hashicorp/vault/latest/docs

142. Answer: A

Explanation: HashiCorp Sentinel is a framework that enforces adherence to policies within your Terraform code. In other words, its code that enforces restrictions on your Terraform Code Sentinel has its own language in which you write policies.

Reference: https://www.terraform.io/cloud-docs/sentinel

143. **Answer: D**

Explanation: The workspace feature in Terraform Cloud does the same thing as the open-source Terraform workspace feature; it is slightly different in the sense that it is all hosted in the cloud instead of your local system. And you can interact with the Cloud workspace using APIs as well. Think of Cloud workspaces as

directories for distinct deployments, hosted in the cloud where you do not need to worry about your workspaces' segregation, storage, and even security.
Reference: https://www.terraform.io/cloud-docs/workspaces

144. Answer: C

Explanation: The registry is a publicly available Terraform providers and modules repository. It is an integral part of Terraform's adoption by the masses because it's publicly accessible to anyone using Terraform.
Reference: https://registry.terraform.io/browse/providers

145. Answer: C

Explanation: HashiCorp Vault is secret management software. It stores sensitive data securely and provides short-lived, temporary credentials to users in place of actual long-lived credentials, like AWS CLI access keys.
Reference:
https://registry.terraform.io/providers/hashicorp/vault/latest

146. Answer: A

Explanation: In the context of Terraform and security, the term "codification" refers to the procedure of directly encoding security guidelines and regulations into Terraform code. You can standardize security enforcement and testing as part of your deployment pipeline by automating security checks before Terraform

deployments. As a result, both claims (A and B) about codification are true.

Reference: https://www.terraform.io/cloud-docs/sentinel/manage-policies

147. Answer: C

Explanation: It can help standardize security testing and automation right into your Terraform deployment pipeline, as it automatically runs before your Terraform deployments.

Reference: https://www.testim.io/blog/what-is-test-automation/

148. Answer: A

Explanation: By enforcing restrictions on the configuration, plan, run, and status of the deployment, sentinel policies can be used to sandbox Terraform deployments. For example, a Sentinel policy could be used to enforce particular tagging criteria or to prevent the deployment of resources to particular areas.

Reference: https://www.terraform.io/cloud-docs/sentinel

149. Answer: D

Explanation: Sandboxing, Codification, Testing, and Automation are the benefits of Sentinel

Reference: https://www.terraform.io/cloud-docs/sentinel

150. Answer: A

Explanation: You can apply Sentinel policies against your Terraform code to sandbox your deployments. For example, stop a dev user from deploying into a prod workspace
and kind of act as a guardrail against accidental deployments.
Reference: https://www.terraform.io/cloud-docs/sentinel

151. Answer: A

Explanation

The terraform fmt command can be safely run at any time but is especially useful before pushing code to a version control system, as it makes your code look cleaner and more consistent.
Reference: https://www.terraform.io/cli/commands/fmt

152. Answer: C

Explanation

You can set TF_LOG to the TRACE, DEBUG, INFO, WARN, or ERROR log level to change the verbosity of the log. TRACE is the default log level, and it is the most verbose.
Reference: https://www.terraform.io/internals/debugging

153. Answer: B

Explanation

The command terraform workspace select <workspace-name> selects or switches to an existing workspace of your choice using the name you have provided.

Reference: https://www.terraform.io/language/state/workspaces

154. **Answer: B**

Explanation

To link pre-existing resources that were not initially developed by Terraform with state that is controlled by Terraform, use the terraform import command. By doing this, you may control these external resources so that you can manage and monitor them using the configuration and state files for Terraform.

Reference:
https://www.terraform.io/cli/commands/workspace#:~:text=The%20terraform%20workspace%20command%20is,listed%20in%20the%20navigation%20bar.

155. Answer: A

Explanation

Terraform's default workspace is automatically generated when working locally and cannot be removed. It acts as your Terraform configurations' and state management's default workspace.

Reference:
https://developer.hashicorp.com/terraform/language/state/workspaces

156. Answer: D

Explanation:

The right Terraform command to establish a new workspace called "new-hire" is *terraform workspace new new-hire*. With the given name, this command will initialize and establish a new workspace.

Reference:
https://developer.hashicorp.com/terraform/cli/commands/workspace/new

157. Answer: C

Explanation

Terraform does not give detailed logging by default. The environment variable TF_LOG must be enabled to allow thorough logging.
You can set the TRACE, INFO, WARN, ERROR, and DEBUG levels by enabling TF_LOG.
Reference:
https://www.terraform.io/internals/debugging#:~:text=You%20can%20set%20TF_LOG%20to,JSON%20encoding%20as%20the%20formatting.

158. Answer: A

Explanation

You can associate provisioners that are not directly associated with a resource with a null_resource if you need to run them.
Reference:
https://www.terraform.io/language/resources/provisioners/null_resource

159. Answer: B and C

Explanation

The operator ~> (Pessimistic Constraint Operator) signifies that only modest (rightmost version increase) modifications are accepted, according to the Terraform doc. As a result, ~> 1.9.0 denotes that the relevant module/provider need allows 1.9.1 to 1.9.x, but not 1.10.0 or 1.0.0 or 1.8.0.

Terraform is looking for any upgrade that is higher than 1.9.0, which can be either 1.9.1 or 1.9.2. As a result, both options are correct.

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

160. Answer: A

Explanation

Before executing a terraform plan, Terraform does not automatically append missing arguments. Your Terraform code will normally raise an error or warning during the planning phase, indicating that any missing needed parameters or configuration settings are present. As the user, it is your obligation to confirm that your Terraform configuration is accurate and complete before using it.
Reference: https://www.terraform.io/cli/commands/plan

161. Answer: B

Explanation:

Terraform workspaces enable you to associate several state files with a single configuration file.
Reference: https://www.terraform.io/language/state/workspaces

162. Answer: A

Explanation:

Although each workspace normally corresponds to a different state file, Terraform workspaces are used to manage many state environments for the same configuration. For a single setup, many workspaces can be created and used, but each workspace maintains a separate state file.
It is uncommon with Terraform to use numerous state files for the same setup. Instead, workspaces are employed to control many deployments of the same configuration or settings, such as development, staging, and production, each with a unique state file. Each state file shows the condition of the infrastructure in that particular environment.
Reference:
https://www.terraform.io/language/settings/backends/remote

163. Answer: B

Explanation:

The state files for resources that Terraform manages are updated using the terraform refresh command. It makes inquiries into the actual infrastructure to obtain the resources' current status and

characteristics, and it subsequently updates the state file with this updated data.
When external changes cause the state file to fall out of sync with the actual infrastructure, or if you want to make sure Terraform has the most recent knowledge of the resources it maintains, running terraform refresh can be helpful.
Reference: https://www.terraform.io/cli/commands/refresh

164. Answer: C

Explanation:

terraform console [options] [dir]

This command allows you to evaluate and experiment with expressions using an interactive command-line interface.

Reference: https://www.terraform.io/cli/commands/console

165. Answer: D

Explanation

The concept of creating code in a high-level language to manage and automate policies is known as policy as code. Proven software development best practices such as version control, automated testing, and automated deployment can be implemented by encoding policy as code in text files.

Sentinel is based on this concept and delivers all of the advantages of policy as code.

Reference: https://docs.hashicorp.com/sentinel/concepts/policy-as-code

166. Answer: A

Explanation

- **Input variables** – Input variables are equivalent to function arguments in Terraform when compared to any traditional programming language.
- **Local values** - Local values are similar to the temporary local variables of a function.
- **Terraform Functions** - To change and combine values, the Terraform language contains a set of built-in functions that can be called from within expressions. Because the Terraform language does not support user-defined functions, only the built-in functions are accessible to use.
- **Depends_on** - It is a meta argument for terraform resource development that indicates explicit dependencies.

Reference: https://www.terraform.io/language/values/variables

167. Answer: C

Explanation

The default number of concurrent operations enabled by the Terraform apply command is 10.

-parallelism=n – Limits the number of concurrent operations Terraform performs as it walks the graph; the default is 10.

Reference: https://www.terraform.io/language/values/variables

168. Answer: D

Explanation

Terraform Cloud, Terraform Enterprise, or any other host that delivers Terraform services can utilize the terraform login command to receive and preserve an API token automatically.

Terraform will retrieve an API token by default and save it in plain text in the credentials.tfrc.json local CLI configuration file. When you run terraform login, it will explain where the API token will be saved and give you the option to cancel if the current configuration isn't what you want.

Terraform login is as follows: terraform login [hostname]

Terraform will presume you wish to log in to Terraform Cloud at app.terraform.io if you do not give an explicit hostname.

Reference: https://www.terraform.io/cli/commands/login

169. Answer: A and C

Explanation

The backends of Terraform are categorized into two categories based on how they manage state and operations:

- Enhanced backends are capable of both storing and performing operations. Only two improved backends are available: local and remote.
- Backends that merely hold state and rely on the local backend to complete actions are known as standard backends.

Reference: https://www.terraform.io/language/settings/backends

170. Answer: A

Explanation

A previously locked state file can be unlocked with the terraform state-unlock command. In order to prevent conflicts and corruption, state locking is a method that restricts concurrent access to the Terraform state. You can use terraform state-unlock to unlock a state file that has gotten locked as a result of an earlier interruption or problem and allow other operations to continue.

The proper command is as follows:

Terraform force-unlock [options] LOCK_ID [DIR]

The state file is unlocked with the command above.

Reference: https://www.terraform.io/cli/commands/force-unlock

171. Answer: A

Explanation

The ssh and winrm are the only connection types supported by remote-exec provisioner; SMB and RDP are not.

Reference:
https://www.terraform.io/language/resources/provisioners/connection

172. Answer: B

Explanation

The terraform init command can automatically download any community provider from a Terraform registry.

Community providers are set up similarly to other providers.

Reference: https://www.terraform.io/cli/commands/init

173. Answer: B

Explanation

One of the reserved terms is count. Instead of duplicating the resources, the count can be used to scale.

The count meta-argument takes a single number and generates many copies of the resource or module.

Reference: https://www.terraform.io/cli/commands/init

174. Answer: E

Explanation

Since these are permissible source choices for a module, options A, B, C, and D are incorrect.

Option E is correct since BLOB storage cannot be used as a module source.

In a module block, the source parameter tells Terraform where to look for the source code for the chosen child module. Terraform uses this to download the source code to a directory on a local disk so that other Terraform commands may utilize it during the terraform init module installation step.

The permissible source options for a module are currently as follows:

- Local Paths

- Terraform Registry
- GitHub
- Bitbucket
- Generic Git, Mercurial repositories
- HTTP URLs
- S3 buckets
- GCS buckets

Reference: https://www.terraform.io/cli/commands/init

175. Answer: A

Explanation

The fmt is used to rewrite Terraform configuration files to a canonical format and style, and Option D is incorrect.

Option A is CORRECT since it validates the terraform configuration.

The show is used to provide human-readable output from a state or plan file; Option C is incorrect.

Option B is incorrect because there is no command called to check in Terraform.

The terraform validate command verifies that a configuration is syntactically correct and internally consistent, regardless of any variables or current state.

Validation necessitates an initialized working directory as well as the installation of any referenced plugins and modules.

Reference: https://www.terraform.io/cli/commands/validate

176. Answer: A

Explanation

Option A is the RIGHT CHOICE. To remove the lock on the state file, use force-unlock with LOCK_ID.

Option B is incorrect since the parameter should be LOCK_ID, not STATE_FILE.

Option C is incorrect since the state lock is removed using force-unlock rather than unlock.

Option D is incorrect since force-unlock requires the input LOCK_ID.

Manually unlock the state for the provided configuration with force-unlock. There will be no changes to your infrastructure as a result of this. This command releases the lock on the current configuration's state. This lock's behavior is determined by the backend being utilized.

Another process cannot unlock local state files. Usage: terraform force-unlock LOCK_ID [DIR]

Reference: https://www.terraform.io/cli/commands/force-unlock

177. Answer: A

Explanation

Option C is incorrect since, unlike trace, this command does not provide extensive verbose information.

Option B is incorrect since this command will provide you with information and give you a lot of it.

Option A is correct because it provides more detailed information than the other options.

Option D is INACCURATE since it just prints warning messages.

The TF_LOG environment variable can be set to any value to allow detailed logging in Terraform. Detailed logs will emerge on the stderr as a result of this.

When logging is enabled, you may use the TF_LOG_PATH variable to require the log to always be appended to a certain file. Even if TF_LOG_PATH is defined, TF_LOG must also be set before any logging may be enabled.

Reference: https://support.hashicorp.com/hc/en-us/articles/360001113727-Enabling-debug-and-trace-run-logs-in-Terraform-CLI-Cloud-or-Enterprise

178. Answer: D

Explanation

Option A is incorrect because this tool displays the characteristics of a single Terraform state file resource.

Since the terraform state list command is used to list resources within a Terraform state, Option D is correct.

Option C is incorrect because this command produces a human-readable output of all resources and attributes.

Option B is incorrect since it attempts to display all of the list file's resources and attributes. The list is used as an input file for the display command in this command.

The command will display a list of all resources in the state file that match the specified addresses (if any). All resources are listed if no addresses are specified.

The resources are arranged alphabetically after being sorted by module depth. This means that resources in your immediate configuration are listed first, followed by resources that are layered deeper within modules.

The state can contain thousands of resources that can be filtered using the id option for sophisticated infrastructures.

Reference: https://www.terraform.io/cli/commands/state/list

179. Answer: C

Explanation

Whether you pass, terraform apply the filename of a previously-saved plan file or not has a big impact on how it behaves.

The terraform apply command is used to carry out the tasks in a Terraform plan.

-auto-approve: Before applying, the proposal is bypassed for interactive approval. This option is disregarded when you pass a previously-saved plan file because Terraform views passing the plan file as approval and will never prompt in that instance.

-input=false: Turns off all interactive prompts in Terraform. This also disables Terraform from prompting for interactive plan acceptance. Thus, Terraform will conservatively presume you do not want to apply the plan, resulting in the operation failing.

-lock=false: Terraform's default behavior of attempting to take a read/write lock on the state for the duration of the operation is disabled with -lock=false.

-lock-timeout=DURATION: Instructs Terraform to retry acquiring a lock for a period of time before issuing an error unless locking is disabled with -lock=false. A duration syntax consists of a number followed by a letter denoting the time unit, such as "3s" for three seconds.

-compact-warnings: This displays any warning messages in a compressed format that just shows the summary messages unless the warnings are accompanied by at least one error, in which case the warning text may provide important context for the faults.

-no-color: Disables output terminal formatting sequences. If you are running Terraform in a situation where the output will be displayed by a system that doesn't understand terminal formatting, use this option.

-parallelism=n: As Terraform travels the graph, limit the number of concurrent operations. The default value is 10.

Terraform apply also accepts the traditional parameters -state, -state-out, and -backup for setups that solely use the local backend.

Reference: https://www.terraform.io/cli/commands/state/list

180. Answer: B

Explanation

The terraform taint command taints a Terraform-managed resource, causing it to be deleted and regenerated on the next application.
Reference: https://www.terraform.io/cli/commands/taint

181. Answer: D

Explanation

After a remote resource is created, the remote-exec provisioner runs a script on it. This can be used to start a configuration management tool, boot up a cluster, and so on. Instead, use the local-exec provisioner to start a local process.
Reference:
https://www.terraform.io/language/resources/provisioners/remote-exec#:~:text=The%20remote%2Dexec%20provisioner%20invokes,the%20local%2Dexec%20provisioner%20instead.

182. Answer: A

Explanation

The Terraform Registry is a public index of modules that have been shared via this protocol. This public registry is the quickest and most convenient method to get started with Terraform and discover modules others in the community produce.
References:

https://www.terraform.io/registry/modules/use

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

183. Answer: B

Explanation

To interface with distant systems, Terraform uses "providers," which are plugins. Terraform configurations must specify which providers they need in order for Terraform to install and use them.

Some providers also require setup (such as endpoint URLs or cloud regions) before they may be used.

Reference:
https://www.terraform.io/language/providers/requirements

184. Answer: C

Explanation

The local backend saves the state to the local filesystem, locks it with system APIs, and executes operations locally.
This state is saved locally by default in a file called "terraform. tfstate," but it can also be saved remotely, which is more useful in a collaborative setting. Terraform makes plans and adjustments to your infrastructure using this local state.
Reference:
https://www.terraform.io/language/state#:~:text=This%20state%20is%20stored%20by,make%20changes%20to%20your%20infrastructure.

185. Answer: B

Explanation

There are now a range of backends that will encrypt the state at rest and will not keep the state in cleartext on computers operating due

to the use of remote backends and, in particular, the availability of Terraform Cloud.

Reference: https://www.terraform.io/plugin/sdkv2/best-practices/sensitive-state

186. Answer: D

Explanation

For each nesting level, indent two spaces.

Align the equal signs of numerous arguments with single-line values that appear on successive lines at the same nesting level.

Reference: https://www.terraform.io/language/syntax/style

187. Answer: A

Explanation

Terraform configurations must specify which providers they need in order for Terraform to install and use them.

Reference:
https://www.terraform.io/language/providers/configuration

188. Answer: C

Explanation

If you and your team are using Terraform to manage important infrastructure, we recommend using Terraform Cloud or Terraform Enterprise's remote backend.

Reference: https://www.terraform.io/cloud-docs/recommended-practices/part1

189. Answer: A

Explanation

The terraform init command creates a working directory in which Terraform configuration files can be found.
Reference: https://www.terraform.io/cli/commands/init

190. Answer: D

Explanation

The local backend saves the state to the local filesystem, locks it with system APIs, and executes operations locally.
Reference:
https://www.terraform.io/language/settings/backends/local

191. Answer: C

Explanation

Within the top-level terraform block, a nested backend block is used to configure backends.
Reference: https://www.terraform.io/language/settings

192. **Answer:** B, C, and D

Explanation: Terraform is an infrastructure as code (IaC) tool that enables declarative infrastructure definition and management. It

offers various built-in functions to handle and transform the data in your setups. Among the choices you gave:

- regex() is a function that compares a given string to a regular expression pattern.
- alltrue() is a function that determines whether each item in a list is true.

With the help of the delete() function, an element can be taken out of a list or map.

Reference:
https://www.terraform.io/docs/language/functions/index.html

193. **Answer:** D

Explanation: The "chomp()" function is not an actual Terraform string function among the alternatives offered. The last three operations, "join()," "split()," and "slice()," are all recognized string operations in Terraform.

Reference:
https://www.terraform.io/docs/language/functions/string.html

194. **Answer:** C

Explanation: Terraform's "map" variable type expresses variables with a key/value type structure. A map works well when storing and accessing data using a specific key, enabling you to link keys with associated values.

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

195. **Answer:** C

Explanation: Use an "object" variable type to store information about servers that have numerous attributes, such as server name (string) and memory size (number). You can define a structured set of attributes with various data types using an object variable.

Reference:
https://www.terraform.io/docs/language/values/variables.html#object-variable-type

196. **Answer:** B and D

Explanation: The structural kinds that are permitted in Terraform include:

An object is a sophisticated data structure that enables the definition of a group of properties with diverse data types.

A tuple is a collection of ordered values, each of which may have a distinct data type.

Reference:
https://www.terraform.io/docs/language/types/index.html

197. **Answer:** C

Explanation: Terraform's word "collection" refers to data structures that may store numerous values. As they may hold many elements, lists, maps, and sets are examples of collections in Terraform. But in Terraform, "tree" is not considered a built-in collection type.

Reference:
https://www.terraform.io/docs/language/types/index.html

198. **Answer:** C

Explanation: The following primitive variable types are supported by Terraform:

- string: A string is made up of various characters.
- number: Represents whole numbers.
- bool: Denotes boolean (true or false) values.

However, Terraform does not recognize the primitive variable type "float". The "number" type, which can handle both integers and floating-point values, is frequently used for working with floating-point numbers.

Reference:

https://www.terraform.io/docs/language/values/variables.html#type-constraints

199. **Answer:** B

Explanation: The "pessimistic" constraint operator is another name for the version restriction operator "~>". It is used in Terraform configuration to establish the variety of versions that a provider's lower and upper boundaries may have. With this operator, you can define a minimum version and an upper-bound version that is compatible with it but does not go higher than the supplied minimum version.

Reference:

https://www.terraform.io/docs/language/providers/requirements.html

200. **Answer:** A

Explanation: Terraform workspaces are made to manage several contexts or environments inside a single setup. They enable you to share the same configuration code while maintaining distinct state files for various environments, such as development and production. This helps manage minor differences between environments without duplicating the complete configuration.

Reference:
https://www.terraform.io/docs/language/state/workspaces.html

201. **Answer:** A

Explanation: Separate instances of state data within the same Terraform working directory are called Terraform CLI Workspaces.

Reference:
https://developer.hashicorp.com/terraform/cli/workspaces

202. **Answer:** B

Explanation: Use the tfconfig import to retrieve Terraform Configuration from a Sentinel policy. As a result, you can examine and apply policies based on the information and organization of the Terraform configuration files.

Reference:
https://learn.hashicorp.com/tutorials/sentinel/terraform-imports

203. **Answer:** D

Explanation: tfapply is not an authorized import, whereas tfplan, tfconfig, and tfstate are in Sentinel policies for Terraform Cloud.

Reference: https://docs.hashicorp.com/sentinel/import/index

204. **Answer:** C

Explanation: Sentinel is a framework for policy as code provided by HashiCorp that enables Terraform users to impose best practices, compliance standards, and standard configurations on resources deployed using infrastructure as code. You can use it to create and implement rules that limit or permit specific behaviors based on predetermined standards.

Reference: https://www.hashicorp.com/solutions/policy-and-governance

205. **Answer:** A, B, and D

Explanation: The following meta-arguments can be used in Terraform in place of "source" and "version" when defining a module:

- for_each: Using the components of a map or set as the basis, this meta-argument enables you to generate many instances of a module.
- count: Based on a numerical number, this meta-argument controls how many module instances should be created.
- depends_on: By defining explicit dependencies between resources or modules, this meta-argument ensures that one is produced before the other.

Reference:
https://www.terraform.io/docs/language/modules/syntax.html#meta-arguments

206. **Answer:** B

Explanation: In Terraform, the "source" argument is required when calling a child module. The "source" option instructs Terraform where to locate the module's resources and logic and the location of the child module's configuration.

Reference:
https://www.terraform.io/docs/language/modules/syntax.html#module-calls

207. **Answer:** D

Explanation: Variable names in Terraform must follow particular naming conventions:

- count: In Terraform, "count" is a recognized variable name frequently used to define resource instances based on numerical values.

Reference:

https://www.terraform.io/docs/language/values/variables.html#name-validation

208. **Answer:** A

Explanation: The level of detail presented in the logs in Terraform is determined by the logging levels. One of the choices offered is:

- TRACE: The most verbose output, including thorough debugging information, is provided by this logging level.

Reference: https://www.terraform.io/docs/internals/debugging.html

209. **Answer:** A

Explanation: In Terraform, local values can refer to other local values. Utilizing other locals' values as building blocks enables you to construct more complicated local values.

Reference:
https://www.terraform.io/docs/language/values/locals.html#referring-to-other-locals

210. **Answer:** G

Explanation: Terraform does not allow using the "depends_on" option when defining an input variable. Although it doesn't apply to input variables, it creates explicit relationships between resources.

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

211. **Answer:** B

Explanation: Terraform's "depends_on" option builds an explicit dependency between resources. It's crucial to remember that if "depends_on" is set, Terraform may not automatically handle resource dependencies. Although it can affect the sequence in which resources are produced, the "depends_on" option does not entirely regulate resource dependencies.

Based on implicit and explicit dependencies, Terraform automatically analyses the resource tree to find the correct order of activities. Although "depends_on" can change the sequence of events, it is not the only aspect to consider when identifying resource dependencies.

Reference:https://www.terraform.io/docs/language/meta-arguments/depends_on.html

212. **Answer:** D

Explanation: To define many instances of the same provider with various configurations in Terraform, use the "alias" option in a provider configuration block. This is frequently employed when working with many resources across various regions of the same cloud provider, such as Amazon Web Services (AWS).

You can manage resources in several locations using the same provider type by specifying aliases by creating separate provider instances with distinct settings, including the region.

Reference:
https://www.terraform.io/docs/language/providers/configuration.html#alias

213. **Answer:** A

Explanation: HashiCorp owns and operates the official Terraform suppliers. To offer seamless integration with diverse infrastructure services, these providers were created by HashiCorp or in partnership with cloud providers and organizations.

Reference: https://www.terraform.io/docs/providers/index.html

214. **Answer:** A and B

Explanation: Terraform's "on_failure" option controls what happens when a provisioner fails to create resources when employing provisioners. One of the choices offered is:

- continue: If the provisioner fails, Terraform will still proceed to create the resource if "on_failure" is set to "continue."
- fail: Terraform will consider the entire resource creation to have failed if "on_failure" is set to "fail" if the provisioner fails.

Reference:
https://www.terraform.io/docs/language/resources/provisioners/syntax.html#attributes

215. **Answer:** D

Explanation: To start a process on the resource that Terraform has built, use the "remote-exec" provisioner in Terraform. Typically, this provisioner uses SSH to execute instructions on the newly formed resource.

Reference:
https://www.terraform.io/docs/language/resources/provisioners/index.html

216. **Answer:** C

Explanation: Instead of launching a process on the generated resource, Terraform's "local-exec" provisioner launches one on the

Terraform computer. It gives you the option to execute a command locally during resource generation.

Reference:
https://www.terraform.io/docs/language/resources/provisioners/index.html

217. **Answer:** A and C

Explanation: The following connection types are supported by Terraform's "file" provisioner:

- ssh: To copy files, the file provisioner can establish an SSH connection to the target resource.
- winrm: The file provisioner copies files by establishing connections to Windows-based resources.

Reference:
https://www.terraform.io/docs/language/resources/provisioners/file.html

218. **Answer:** B

Explanation: To transfer files or directories from the computer executing Terraform to the newly generated resource, utilize the "file" provisioner in Terraform. Transferring configuration files, scripts, or other assets to the newly created resource is frequently used.

Reference:
https://www.terraform.io/docs/language/resources/provisioners/file.html

219. **Answer:** A

Explanation: Terraform uses the "null_resource" resource type to specify a resource that does not directly manage or produce anything. It's frequently used as a stand-in to start provisioners that aren't linked to particular resources. Due to this, you can run any scripts or commands as part of your Terraform workflow.

Reference:
https://www.terraform.io/docs/language/resources/null_resource.html

220. **Answer:** A

Explanation: Using provisioners in Terraform setups should only be done as a last resort. It is often advised to manage infrastructure primarily through resource definitions, even though they are a powerful tool for taking action on resources after they have been established. Provisioners are often used when no other options exist to complete a task.

Reference:
https://www.terraform.io/docs/language/resources/provisioners/index.html#when-to-use-provisioners

221. **Answer:** A

Explanation: The Terraform backend can be moved even when resources are managed. It is crucial to highlight that you should exercise caution when moving the backend to prevent data loss or discrepancies between the state file and the actual resources. To keep

your infrastructure's integrity, it is advised that backend migrations be carefully planned and carried out.

Reference: https://www.terraform.io/docs/cli/state/move.html

222. **Answer:** A

Explanation: Terraform's "local" backend type does not support state locking. This is because there is no need for locking techniques. After all, the state file is kept on the local filesystem and is not shared by users.

Reference: https://www.terraform.io/docs/language/settings/backends/index.html

223. **Answer:** A

Explanation: All of Terraform's standard backend types support state locking, encryption at rest, and remote state storage. The security, cooperation, and integrity of the Terraform state depend on these properties.

Reference: https://www.terraform.io/docs/language/settings/backends/index.html

224. **Answer:** B

Explanation: "GitHub" is not a supported backend type in Terraform among the options offered. Terraform supports several other backend types for storing remote states; however, "github" is not one of them.

Reference:
https://www.terraform.io/docs/language/settings/backends/index.html

225. **Answer:** B

Explanation: Terraform supports establishing "standard" backends for remote state storage. With these standard backend types, you can save the Terraform state remotely for collaboration and security using Amazon S3, HashiCorp Consul, and HashiCorp Vault.

Reference:
https://www.terraform.io/docs/language/settings/backends/index.html

226. **Answer:** A and B

Explanation: You can set up the following parameters in Terraform for a default local backend:

- path: Defines the location where the Terraform state file will be kept on your local computer.
- working_dir: The working directory in which Terraform commands are executed is specified by working_dir. It impacts where the state file is kept and how module paths are determined.

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

227. **Answer:** B

Explanation: Use the -concurrent flag to restrict the number of concurrent operations carried out simultaneously if rate-limiting problems occur during Terraform operations in your CI/CD pipeline. Without altering the total time of the builds, this can assist in slowing down the rate of operations and prevent rate limit issues.

Reference:
https://www.terraform.io/docs/cli/commands/index.html

228. **Answer:** B

Explanation: By default, Terraform will automatically create or update up to 10 resources concurrently during a terraform apply. This parallelism hastens the provisioning of resources. If necessary, the value can be changed using the -parallelism flag.

Reference:

https://www.terraform.io/docs/cli/commands/apply.html#parallelism-n

229. **Answer:** A

Explanation: Use the -target flag followed by the resource addresses you wish to target when you only want to plan specific resources added to your configuration and disregard the others. This will restrict the application of the terraform plan command to the listed resources alone.

Reference:
https://www.terraform.io/docs/cli/commands/plan.html#resource-targeting

230. **Answer:** A

Explanation: You may get a list of all the resources available in your Terraform state by using the terraform state list command.

Reference: https://www.terraform.io/docs/cli/state/index.html

231. **Answer:** A

Explanation: It is recommended and frequently necessary to perform the terraform plan command before terraform apply in the Terraform workflow. The plan command from Terraform creates an execution plan that describes the modifications it will make to your infrastructure. Before using the terraform apply command to make the adjustments, you can review this plan to understand and confirm the alterations that will be done.

Reference: https://www.terraform.io/docs/cli/run/workflow.html

232. **Answer:** B

Explanation: The -check flag can be used by your teammate if they are worried about substantial modifications to their configuration files when they run the terraform fmt command. This flag checks to see if the files require formatting adjustments without actually making them. It will highlight the variations between the suggested and existing formatting modifications.

Reference: https://www.terraform.io/docs/cli/commands/fmt.html

233. **Answer:** D

Explanation: Different Terraform backend types allow state locking, such as Consul, Kubernetes, and S3. State locking is crucial to maintaining the state's integrity throughout concurrent Terraform activities. It prevents concurrent updates to the state file.

Reference: https://www.terraform.io/docs/state/locking.html

234. **Answer:** B

Explanation: Terraform tracks the actual resources and their expected state using the state file as the source of truth. Terraform determines what adjustments are necessary to put the real resources in line with the planned state throughout each run by comparing the state file with the expected configuration. Terraform relies on the state file to comprehend the current infrastructure condition and accurately determine the necessary adjustments.

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

235. **Answer:** A

Explanation: By default, the state file will keep usernames and passwords in plain text if you utilize them as variables in Terraform code. If the state file is compromised, this might potentially reveal private information. Passwords and access keys are sensitive information that should be handled carefully.

If you want to avoid keeping sensitive data directly in your Terraform configuration, use the sensitive data handling tools that Terraform offers, such as sensitive input variables. It would help if you also thought about using external secret management services.

Reference:
https://www.terraform.io/docs/language/values/variables.html#sensitive-variables

236. **Answer:** B

Explanation: While the main objective of Terraform is to make sure that the Terraform state corresponds to the distant cloud resources specified in the configuration, there are times when this may not be the case. This may occur for several reasons, including manually adjusted by tools other than Terraform or external changes made to the resources.

The state may not always match the resources provided by the configuration. However, Terraform offers mechanisms to help identify and correct such inconsistencies.

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

237. **Answer:** D

Explanation: Terraform Core is a binary produced statically with the Go programming language. The Terraform tool's core component, Terraform Core, is mainly created using the Go programming language.

Reference:
https://www.terraform.io/docs/internals/architecture.html

238. **Answer:** A

Explanation: Terraform tracks compatible versions of dependencies like providers and modules using a dependency lock file called .terraform.lock.hcl. The use of the same versions of dependencies across many environments and runs is helped by the lock file.

Reference: https://www.terraform.io/docs/cli/state/lock.html

239. **Answer:** B and C

Explanation: To avoid the exposure of sensitive or environment-specific data, it is common practice to exclude specific Terraform files from version control repositories. The following files should not be opened:

- terraform.tfstate: This file includes local state data and sensitive information. Version control shouldn't be used because it can reveal private information about your infrastructure.
- terraform.tfvars: This file frequently contains variable values that could include private information like API keys or passwords. It is advised to keep this file out of version control to prevent disclosing such information.

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

240. **Answer:** B

Explanation: On various operating systems, Terraform is available as a single binary, although AIX (IBM AIX) is not one of the supported operating systems. While Linux, macOS, and Windows are the main operating systems supported by Terraform, AIX is not listed as an officially supported platform.

Reference: https://learn.hashicorp.com/tutorials/terraform/install-cli

241. **Answer:** A

Explanation: HashiCorp advises using four spaces between each nesting level while writing Terraform code. Code readability and maintainability are both enhanced by consistent indentation.

Reference:
https://www.terraform.io/docs/language/style/syntax.html#spaces-indentation

242. **Answer:** C

Explanation: HashiCorp Configuration Language (HCL) is the primary language Terraform configuration uses to define infrastructure as code. HCL was created primarily for writing comprehensible, human-readable configuration files. While Terraform can use JSON and YAML for specific tasks (such as variable files and state outputs), HCL is the main language for creating Terraform configuration.

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

243. **Answer:** F

Explanation: The following block types are permitted in Terraform:

- resource
- output
- module
- provider
- data

The block type "bucket" is not permitted in Terraform, though. Terraform configurations use blocks like provider, resource, output, module, and data to specify different components of your infrastructure.

Reference:
https://www.terraform.io/docs/language/structure/index.html

244. **Answer:** A

Explanation: Terraform is indeed made to make it easier to install and manage infrastructure across many cloud providers, including well-known ones like AWS, Azure, Google Cloud, and others. Using a single configuration language, Terraform offers a consistent and unified way to define, deploy, and manage infrastructure resources across various cloud environments.

Reference: https://www.terraform.io/intro/use-cases.html#multi-cloud-and-hybrid-cloud-management

245. **Answer:** E

Explanation: Numerous advantages are provided by Terraform and other Infrastructure as Code (IaC) tools, such as:

- **Manage and track infrastructure:** Infrastructure as code (IaC) solutions like Terraform assist you in declaratively defining, managing, and tracking your infrastructure resources, which makes it simpler to comprehend and maintain your infrastructure.
- **Automate infrastructure changes:** With the help of IaC technologies, you can automate the provisioning, updating, and scaling of infrastructure resources, which lowers the amount of manual work needed to make changes.
- **Reusable configuration:** Using IaC, you can create modules for reusable configuration that can be used in various applications while maintaining consistency and effectiveness.
- **Collaboration with VCS:** Git and other version control systems (VCS) can track changes, cooperate on infrastructure code, and enable teams to work well together.

Terraform and other solutions offer these advantages to simplify infrastructure administration and boost the effectiveness of managing cloud resources.

Reference: https://www.terraform.io/intro/vs/index.html

246. **Answer:** B

Explanation: The main function of Terraform is as an Infrastructure as Code (IaC) tool. It enables you to consider your infrastructure as software that can be versioned, automated, and maintained in a consistent and repeatable manner by allowing you to describe and manage your infrastructure resources using code.

Reference: https://www.terraform.io/intro/index.html

247. **Answer:** D

Explanation: There are numerous advantages to using Terraform as an Infrastructure as Code (IaC) technology, including:

Terraform offers a consistent way to manage infrastructure across numerous cloud platforms like AWS, Azure, Google Cloud, and others, using the same configuration language and workflows. Terraform may be used to manage the infrastructure of several cloud platforms.

Terraform's state allows you to track resource changes during deployments: Your infrastructure's current status is tracked by Terraform in a state file. This enables you to recognize and comprehend adjustments made throughout deployments.

You can enter your configurations into version control to securely collaborate on infrastructure: Version Control Systems (VCS) like Git can be used to version Terraform configurations, facilitating collaboration, history tracking, and safe code sharing.

By offering these advantages, Terraform helps to improve cooperation, expedite infrastructure management, and increase the dependability of your deployments.

Reference: https://www.terraform.io/intro/vs/index.html

248. **Answer:** B, C, and D

Explanation: IaC uses a configuration language accessible by humans to make building infrastructure code more efficient. The configuration languages used by Infrastructure as Code (IaC) technologies like Terraform are made understandable by humans and simple to comprehend. This facilitates the writing and upkeep of infrastructure code more quickly.

By committing your configurations to version control, IaC makes it possible for you to collaborate on infrastructure safely: IaC enables secure collaboration, change tracking, and the capacity to roll back to earlier versions whenever necessary by allowing you to store your infrastructure code in version control systems like Git.

IaC code can be used to control infrastructure across many cloud platforms: A uniform approach to managing infrastructure resources across many cloud platforms is offered by IaC solutions like Terraform. This makes resource management in multi-cloud or hybrid-cloud setups simpler.

Reference: https://www.terraform.io/intro/vs/index.html

249. **Answer:** C

Explanation: Terraform and other Infrastructure as Code (IaC) techniques can improve the security and consistency of your infrastructure, but their main focus is on defining and controlling the configuration and provisioning of infrastructure resources. It is more connected to secure application development practices and secure coding methodologies than it is to IaC's fundamental advantage of directly decreasing vulnerabilities in public-facing applications.

Reference: https://www.terraform.io/intro/vs/index.html

250. **Answer:** B

Explanation: The following steps make up Terraform's main workflow:

- Write: You create the Terraform configuration code specifying the ideal condition for your infrastructure resources.
- Plan: To create an execution plan, use the Terraform plan command. This strategy outlines the adjustments Terraform will undertake to reach the intended state. Before the changes are implemented, you can examine and confirm them.
- Apply: Use the terraform apply command to implement your planned modifications and provision or update the real infrastructure resources according to your configuration.

Before making changes to your infrastructure, this approach ensures you have a clear plan.

Reference: https://www.terraform.io/intro/getting-started/build.html

251. **Answer:** A, B, and C

Explanation: The following three qualities are offered by Terraform Cloud:

- **Remote state management**: Terraform Cloud provides a central location for storing and managing your Terraform state files, fostering teamwork and guaranteeing uniform state management across teams.

- **Remote Terraform Execution:** Terraform Cloud enables remote Terraform command execution, which is useful when using a centralized platform to conduct Terraform activities.
- **Private Module Registry:** Terraform Cloud has a module registry where you may exchange and publish your private Terraform modules with other company members, facilitating collaboration and code reuse for infrastructure.

Reference: https://www.terraform.io/docs/cloud/index.html

252. **Answer:** B and E

Explanation: The following settings are not accessible in Terraform OSS/CLI (Open Source Software/Command-Line Interface) or Terraform Cloud Free Tier:

- **Policy as Code (Sentinel):** The free tiers do not include the Policy as Code capability utilizing Sentinel. The paid editions of Terraform Cloud provide this feature.
- **Single Sign-On (SSO):** The free levels of Terraform Cloud do not offer Single Sign-On (SSO). The commercial editions of the software also have SSO as a feature.

Reference: https://www.terraform.io/docs/cloud/free/index.html

253. **Answer:** B

Explanation: SSO (Single Sign-On) is particularly noted as a feature exclusive to the Terraform Enterprise edition. Terraform Cloud for Business incorporates numerous capabilities accessible in the Terraform Enterprise edition. SSO enables users to log into numerous services with just one set of credentials.

Reference: https://www.hashicorp.com/products/terraform/pricing

254. **Answer:** A

Explanation: When used with a Terraform Enterprise subscription, Terraform Cloud offers the opportunity to create a Private Module Registry. This improves reuse and collaboration by enabling you to create and distribute Terraform modules inside your company.

Reference:
https://www.terraform.io/docs/cloud/registry/index.html

255. **Answer:** A

Explanation: The main.tf file is frequently used in a typical Terraform project to specify and declare the resources you want to manage with Terraform. The configuration code for the infrastructure resources, including virtual machines, networks, storage, etc., is contained in this file.

Reference: https://www.terraform.io/docs/configuration/index.html

256. **Answer:** D

Explanation: The following is the proper way to use Terraform to install new infrastructure:

Run terraform init to download the necessary provider plugins and initialize the working directory.

- Create a configuration for Terraform: To specify the needed infrastructure resources, create or edit the Terraform

configuration files (such as main.tf, variables.tf, etc.).

- Run terraform plan: This command creates an execution plan outlining the infrastructure changes Terraform will make. Before implementing the modifications, you can review them.
- Run terraform apply: Using the configuration, this command builds new infrastructure resources and implements the planned changes.

Reference: https://www.terraform.io/intro/getting-started/build.html

257. **Answer:** B

Explanation: The terms "immutable, declarative, and based on the HashiCorp Configuration Language (HCL)" best describe the Terraform language. HCL is the main language for defining Terraform configurations, while JSON is also supported as a substitute for HCL for configuration.

Reference: https://www.terraform.io/docs/configuration/index.html

258. **Answer:** B

Explanation: Use dynamic blocks to quickly and efficiently add a list of required TCP ports to a new security group in Terraform. You can build repeating configuration blocks using dynamic blocks based on input data.

You can write a single block of code that defines the security group rules for the specified TCP ports using dynamic blocks. As a result,

the code is shorter and easier to maintain because there is no longer a need to manually duplicate the block for each port.

Reference:
https://www.terraform.io/docs/language/expressions/dynamic-blocks.html

259. **Answer:** C

Explanation: Using the information you supply, a dynamic block in Terraform enables you to generate nested configuration blocks dynamically. You utilize it when you want to create numerous similar configuration blocks but do not want to duplicate the code.

You can manage complex configurations more effectively by using dynamic blocks, which let you create blocks based on a list of objects or other dynamic data.

Reference:
https://www.terraform.io/docs/language/expressions/dynamic-blocks.html

260. **Answer:** C

Explanation: An official Terraform provider is a plugin that enables communication between Terraform and a particular cloud, service, or infrastructure provider. It outlines the assets, information sources, and features that Terraform can control for that specific provider. Official Terraform providers are accessible for various cloud platforms and services and are managed by HashiCorp.

Reference: https://www.terraform.io/docs/providers/index.html

261. **Answer:** D

Explanation: Resources should be added to the dependency graph in Terraform based on the configuration. This is the correct procedure for doing so. Resources still in the state file but no longer included in the configuration are not automatically added to the graph during this procedure. The state file and the settings should be updated to ensure proper graph creation.

Reference:
https://www.terraform.io/docs/language/state/purpose.html

262. **Answer:** B

Explanation: The current status of the infrastructure that Terraform controls is recorded in a Terraform state file. It includes details about the produced resources, their setups, and their present conditions. This aids Terraform in comprehending the adjustments required to achieve the target state specified in the configuration.

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

263. **Answer:** A

Explanation: Depending on the resources being used and how you define "sensitive," the Terraform state may contain sensitive information. The state includes all resource properties and resource IDs. This could include initial passwords for resources like databases.

Reference:
https://developer.hashicorp.com/terraform/language/state/sensitive-data

264. **Answer:** A

Explanation: Terraform keeps its state files by default in the same directory where the terraform apply command is executed. This can be altered by defining a backend configuration to save the state remotely.

Reference:
https://www.terraform.io/docs/language/state/index.html#default-local-backend-file

265. **Answer:** A

Explanation: A remote backend to store the Terraform state for larger teams is advised. This eliminates problems from having numerous local state files and enables multiple team members to work together on the same infrastructure while monitoring changes.

Reference:
https://www.terraform.io/docs/language/state/remote.html

266. **Answer:** B

Explanation: On all operations that have the potential to write state, state locking occurs automatically. There will be no indication of it occurring on your end. If state locking is unsuccessful, Terraform

will stop. With the -lock switch, you can turn off state locking for most commands, but doing so is not advised.

Reference:
https://developer.hashicorp.com/terraform/language/state/locking

267. **Answer:** C

Explanation: Terraform's state locking prevents several actions from accessing the state file simultaneously. This ensures that only one action at a time can affect the state to avoid conflicts and potential data corruption. The intention is to prevent scenarios when several users attempt to make modifications simultaneously, as this may result in a compromised or inconsistent state.

Reference:
https://www.terraform.io/docs/language/state/index.html#locking-state-to-prevent-conflicts

268. **Answer:** A

Explanation: In the less frequent circumstance, if you want to remove a binding to an already-existing remote object without first deleting it, you can use terraform state rm to essentially "forget" the object while it still exists in the distant system.

Reference:
https://developer.hashicorp.com/terraform/cli/commands/state/rm

269. **Answer:** A

Explanation: Terraform does not update its state when a terraform state list is run. This command will only read the state file and make no changes.

Reference:
https://developer.hashicorp.com/terraform/cli/commands/state/list

270. **Answer:** C

Explanation: The first command that should be run before terraform plan/apply is terraform init after creating a new Terraform configuration, cloning an existing one from version control, adding a new provider, or adding a new module.

Reference:
https://developer.hashicorp.com/terraform/cli/commands/init

271. **Answer:** A

Explanation: terraform init -from-module={MODULE-SOURCE} can be used as a shortcut to check out a configuration from version control and initialize the working directory for it when given a version control source.

Reference:
https://developer.hashicorp.com/terraform/cli/commands/init#copy-a-source-module

272. **Answer:** C and D

Explanation: Every time a new module is added to a configuration, Terraform must install it before it can be used. Installing and

updating modules can be done with the terraform get and terraform init commands.

Reference:
https://developer.hashicorp.com/terraform/tutorials/modules/module-create#install-the-local-module

273. **Answer:** A

Explanation: To create an execution plan in Terraform, use the terraform plan command. Taking into account the setup and condition right now, it previews the modifications that will be made to your infrastructure. It displays the resources generated, changed, or eliminated when you use the terraform apply command to implement your modifications.

Reference: https://www.terraform.io/docs/cli/commands/plan.html

274. **Answer:** A

Explanation: The terraform plan, by default, examines each .tf file in the directory where the Terraform command is currently being run. To create an execution plan, it assesses the configuration in the files.

Reference: https://www.terraform.io/docs/cli/commands/plan.html

275. **Answer:** B

Explanation: The infrastructure can be modified using the terraform apply command based on the current configuration. According to the

modifications you have made to your Terraform files and the execution plan produced by terraform plan, it updates the resources.

Reference: https://www.terraform.io/docs/cli/commands/apply.html

276. **Answer:** A

Explanation: Terraform infrastructure that has already been built can be destroyed with the terraform destroy command. It discharges the cloud provider of the resources listed in the configuration files.

Reference:
https://www.terraform.io/docs/cli/commands/destroy.html

277. **Answer:** B and C

Explanation: The terraform apply -destroy command is simply an alias for the terraform destroy command for convenience.

Reference:
https://developer.hashicorp.com/terraform/cli/commands/destroy#usage

278. **Answer:** A

Explanation: You may format your Terraform configuration files consistently and uniformly using the terraform fmt command. It ensures the code is appropriately structured and adheres to accepted style recommendations.

Reference: https://www.terraform.io/docs/cli/commands/fmt.html

279. **Answer:** A

Explanation: HashiCorp Configuration Language (HCL) files are automatically formatted using the terraform fmt command by the recommended spacing and style recommendations. It aids in maintaining legible and consistent code formatting.

Reference: https://www.terraform.io/docs/cli/commands/fmt.html

280. **Answer:** A

Explanation: Subdirectory-level files are processed using the terraform fmt -recursive command. Only the specified directory (or the current directory) is processed by default.

Reference:

https://developer.hashicorp.com/terraform/cli/commands/fmt#recursive

281. **Answer:** A

Explanation: The terraform validate command performs checks to ensure a configuration is syntactically correct and internally consistent.

Reference:
https://developer.hashicorp.com/terraform/cli/commands/validate

282. **Answer:** A

Explanation: When you use the -json option, Terraform will generate validation results in JSON format, which can be used for

tool integrations, such as highlighting issues in a text editor.

Reference:
https://developer.hashicorp.com/terraform/cli/commands/validate#json-output-format

283. **Answer:** A

Explanation: The terraform validate command validates the configuration files in a directory without contacting remote services such as remote state, provider APIs, etc.

Reference:
https://developer.hashicorp.com/terraform/cli/commands/validate

284. **Answer:** C

Explanation: The terraform console command displays an interactive console where you can evaluate and experiment with expressions.

Reference:
https://developer.hashicorp.com/terraform/cli/commands/console

285. **Answer:** C

Explanation: The value of an output variable can be obtained from the state file using the terraform output command.

Reference:
https://developer.hashicorp.com/terraform/cli/commands/output

286. **Answer:** C

Explanation: You can specify the variable definition file using the -var-file option. If such things exist, Terraform will also automatically load the terraform.tfvars or terraform.tfvars.json files, which define various variables.

Reference: https://developer.hashicorp.com/terraform/language/values/variables#variable-definitions-tfvars-files

287. **Answer:** A and C

Explanation: The "terraform init" phase initializes the working directory and downloads the necessary provider plugins, while the "terraform apply" phase creates or updates the infrastructure according to the Terraform configuration.

Reference: https://developer.hashicorp.com/terraform/intro/core-workflow

288. **Answer:** C

Explanation: You can use the terraform import command to handle manually deployed resources in Terraform without affecting others. By connecting existing resources to the appropriate resource settings in your Terraform code, you may manage existing resources using Terraform.

Reference: https://www.terraform.io/docs/cli/import/index.html

289. **Answer:** D

Explanation: When importing current resources into your Terraform state, you must explicitly use the terraform import command. You must execute a different command to link existing resources with relevant resource configurations in your Terraform code.

Reference: https://www.terraform.io/docs/cli/import/index.html

290. **Answer:** A

Explanation: The terraform import aws_instance.foo i-abcd1234 command will find the existing resource from AWS instance ID i-abcd1234 and import it into your Terraform state at the given ADDRESS aws_instance.foo.

The resource type being imported determines ID. For AWS instances, the instance ID is i-abcd1234, whereas, for AWS Route53 zones, the zone ID is Z12ABC4UGMOZ2N.

ADDRESS must be a valid resource address comprising two parts: resource_type.resource_name.

Reference:
https://developer.hashicorp.com/terraform/cli/commands/import

291. **Answer:** A

Explanation: You should first add the new resource block in terraform configuration matching the remote resource or just an empty resource block; then, you can import the remote resource

using terraform import by resource ID, e.g., terraform import aws_instance.foo i-abcd1234 where i-abcd1234 is EC2 instance Id.

Reference:
https://developer.hashicorp.com/terraform/cli/commands/import

292. **Answer:** B

Explanation: A Terraform state's resources can be listed using the terraform state list command.

Reference: https://www.terraform.io/docs/cli/import/index.html

293. **Answer:** D

Explanation: The backend configuration in Terraform defines how and where the Terraform state is saved. Backend settings cannot, however, refer to named values like input variables, local variables, or data source characteristics because they are distinct from variable references.

Reference:

https://www.terraform.io/docs/language/settings/backends/index.html

294. **Answer:** C

Explanation: The attributes of a single resource in the Terraform state are displayed using the terraform state show command.

Reference:
https://developer.hashicorp.com/terraform/cli/commands/state/sho

w

295. **Answer:** C

Explanation: A state or plan file can output JSON in a human-readable format using the terraform show -json command.

Reference:
https://developer.hashicorp.com/terraform/cli/commands/show

296. **Answer:** B

Explanation: The terraform taint command alerts Terraform to the degradation or damage of a specific object. The next plan you write will include a proposal from Terraform to replace it.

Reference:
https://developer.hashicorp.com/terraform/cli/commands/taint

297. **Answer:** A

Explanation: To tell Terraform that an item has degraded or been damaged, it is advised to use terraform apply -replace rather than the deprecated terraform taint command. The next plan you write will include a proposal from Terraform to replace it.

Reference:
https://developer.hashicorp.com/terraform/cli/commands/taint

298. **Answer:** A

Explanation: Terraform is told to plan to replace (delete and recreate) the resource instance with the supplied address when the planning option -replace=ADDRESS is used.

Reference:
https://developer.hashicorp.com/terraform/cli/commands/plan#planning-options

299. **Answer:** A

Explanation: Terraform is told to plan to replace (delete and recreate) the resource instance with the supplied address when the planning option -replace=ADDRESS is used.

Reference:
https://developer.hashicorp.com/terraform/cli/commands/plan#replace-address

300. **Answer:** B

Explanation: Terraform changes the state file to reflect the most recent status of the real-world resources when you run terraform refresh. However, the configuration files (.tf files) are not reprocessed. Only the state is updated to reflect the real resource state in the cloud provider; the configuration stays the same.

Reference:
https://www.terraform.io/docs/cli/commands/refresh.html

301. **Answer:** A and C

Explanation: By default, before running, the terraform plan and terraform apply commands both automatically refresh the state. This ensures that the execution plan is based on the most recent data and that the current state of the resources is synchronized with the state file.

References:
https://www.terraform.io/docs/cli/commands/plan.html
https://www.terraform.io/docs/cli/commands/apply.html

302. **Answer:** A and B

Explanation: The terraform refresh command has been removed in versions of Terraform 0.15.4 and later, and the -refresh-only flag has been added to the terraform apply and terraform plan commands. With this flag, you can run a refresh without modifying or creating an execution plan.

Reference: https://github.com/hashicorp/terraform/pull/29038

303. **Answer:** B and D

Explanation: You can examine Terraform's modifications gathered throughout the refresh process using the terraform apply -refresh-only and terraform plan -refresh-only commands, respectively. To conduct a refresh without making any modifications or creating an execution plan, use the -refresh-only flag.

For further details, you can visit the given URL.

https://github.com/hashicorp/terraform/pull/29038

304. **Answer:** A and B

Explanation: Any infrastructure changes specified in your setup are implemented when Terraform configuration is applied. It may be necessary to create, update, or destroy assets by the configuration.

Terraform automatically gets the plugins required by the configuration to communicate with the listed providers and provisioners throughout the apply process.

Reference:
https://www.terraform.io/docs/cli/run/index.html#applying-configuration

305. **Answer:** A

Explanation: The Terraform remote backend, also called the "remote" backend, enables the execution of Terraform runs utilizing either Terraform Cloud or specialized on-site infrastructure. This enables remote management of infrastructure state by organizations, facilitating collaboration and better Terraform Cloud features.

Reference:
https://www.terraform.io/docs/language/settings/backends/remote.html

306. **Answer:** D

Explanation: Without making any changes to the Terraform code, Terraform will notice that the actual state of the resource (in this case, the VM) differs from the expected state specified in the

Terraform configuration when you destroy a resource using the cloud provider console and then run terraform apply. Generating a new VM will attempt to get the real state back in line with the desired state.

Reference:
https://www.terraform.io/docs/cli/run/index.html#detailed-explanation-of-tf-actions

307. **Answer:** A

Explanation: Using terraform fmt at the code linting stage of your CI/CD process is advised. Automatically formatting your Terraform code by the recognized Terraform style requirements ensures that everyone on your team works with readable and consistent code.

Reference: https://www.terraform.io/docs/cli/commands/fmt.html

308. **Answer:** C

Explanation: The terraform state show command can display extensive information about a given resource, including its features, like the public IP address. The terraform state list command lists the resources handled by Terraform.

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

309. **Answer:** A and B

Explanation: You issue the terraform apply command to apply your Terraform setup and create resources.

The -parallelism flag and the appropriate number of operations regulate how many concurrent operations Terraform employs while creating resources.

Reference:
https://www.terraform.io/docs/cli/commands/apply.html#parallelism-n

310. **Answer:** B

Explanation: If the state has been locked due to an earlier stopped or unsuccessful operation, it can be manually unlocked using the terraform force-unlock command. When the state is locked, you will obtain a specific lock ID that must be used for this command.

Reference:
https://www.terraform.io/docs/cli/commands/state/force-unlock.html

311. **Answer:** D

Explanation: The terraform.tfstate file, which holds the current state of your infrastructure as controlled by Terraform, is kept in the local Terraform backend by default. It keeps track of the resources Terraform has produced and their present state.

Reference:
https://www.terraform.io/docs/language/state/backends.html#local-backend

ABOUT OUR PRODUCTS

Other products from IPSpecialist LTD regarding CSP technology are:

AWS Certified Cloud Practitioner Study guide

AWS Certified SysOps Admin - Associate Study guide

AWS Certified Solution Architect - Associate Study guide

AWS Certified Developer Associate Study guide

AWS Certified Advanced Networking – Specialty Study guide

AWS Certified Security – Specialty Study guide

AWS Certified Big Data – Specialty Study guide

AWS Certified Machine Learning – Specialty Study guide

Microsoft Certified: Azure Fundamentals

Microsoft Certified: Azure Administrator

 Microsoft Certified: Azure Solution Architect

 Microsoft Certified: Azure DevOps Engineer

 Microsoft Certified: Azure Developer Associate

 Microsoft Certified: Azure Security Engineer

 Microsoft Certified: Azure Data Fundamentals

 Microsoft Certified: Azure AI Fundamentals

 Microsoft Certified: Azure Database Administrator Associate

 Google Certified: Associate Cloud Engineer

 Google Certified: Professional Cloud Developer

Microsoft Certified: Azure Data Engineer Associate

Microsoft Certified: Azure Data Scientist

Ansible Certified: Advanced Automation

Oracle Certified: OCI Foundations Associate

Oracle Certified: OCI Developer Associate

Oracle Certified: OCI Architect Associate

Kubernetes Certified: Application Developer

Other Network & Security related products from IPSpecialist LTD are:

- CCNA Routing & Switching Study Guide

- CCNA Security Second Edition Study Guide
- CCNA Service Provider Study Guide
- CCDA Study Guide
- CCDP Study Guide
- CCNP Route Study Guide
- CCNP Switch Study Guide
- CCNP Troubleshoot Study Guide
- CCNP Security SENSS Study Guide
- CCNP Security SIMOS Study Guide
- CCNP Security SITCS Study Guide
- CCNP Security SISAS Study Guide
- CompTIA Network+ Study Guide
- Certified Blockchain Expert (CBEv2) Study Guide
- EC-Council CEH v10 Second Edition Study Guide
- EC-Council CEH v12 First Edition Study Guide