Exam Session - Cert Prep: Certified Terraform Associate

cloudacademy.com/quiz/exam/3761207/results

#1

Where is the default location that Terraform stores it's state in?

✓

The current working directory in which Terraform is ran.

✗

At the users root directory.

✗

In the same location that Terraform is installed.

E.g. /usr/bin/terraform

✗

In ~/.terraform.d/plugins

Explanation

In the default configuration, Terraform stores the state in a file in the current working directory where Terraform was run.

Concept: Understand Terraform's purpose (vs other IaC)

https://www.terraform.io/docs/language/state/purpose.html

#2

What are complex types in terraform?

✓

A type that groups multiple values into a single value.

✗

A variation of a string type.

✗

A variance of a data source.

✗

A type that derives its value from RegEx logic.

Explanation

A complex type is a type that groups multiple values into a single value. Complex types are represented by type constructors, but several of them also have shorthand keyword versions.

Concept: Read, generate, and modify configuration

https://www.terraform.io/docs/language/expressions/type-constraints.html#complex-types

#3

How do you access module attributes?

✓

Through the child module, by declaring an output value to selectively export certain values to be accessed by the calling module.

✗

Through the parent module, by declaring an output value to selectively export certain values to be accessed by the calling module.

✗

By specifying the *outputs* block.

✗

When apply is ran, you must pass in *-resource-output={ATTRIBUTE.NAME}*.

Explanation

The resources defined in a module are encapsulated, so the calling module cannot access their attributes directly. However, the child module can declare output values to selectively export certain values to be accessed by the calling module.

Concept: Interact with modules

https://www.terraform.io/docs/language/modules/syntax.html#accessing-module-output-values

#4

In Terraform, what type of versioning is module versioning following?

✓
Semantic

✗
A-Semantic

✗
Pre-Semantic

✗
Post-Semantic

Explanation

Each module in the registry is versioned. These versions syntactically must follow semantic versioning. In addition to pure syntax, we encourage all modules to follow the full guidelines of semantic versioning.

Concept: Interact with Modules

https://www.terraform.io/docs/registry/modules/use.html#module-versions

#5

What Terraform command modifies a HashiCorp Configuration Language (HCL) file to adhere to the recommended spacing rules for HCL files?

✓
fmt

✗
trim

✗
align

✗
rewrite

Explanation

The **fmt** command rewrites Terraform configuration files to a canonical format and style.

https://www.terraform.io/docs/commands/fmt.html
Covered in this lecture
Terraform Configuration
Course:Managing Infrastructure With Terraform

15m

#6

How do you lock a provider's file?

✓
terraform providers lock

✗
terraform lock

✗
terraform apply lock

✗
terraform lock provider -provider={PROVIDER_NAME}

Explanation

Use the terraform providers schema command to get machine-readable information about the resources and configuration options offered by each provider.

Concept: Understand Terraform basics

https://www.terraform.io/docs/cli/plugins/index.html
#7

Your teammate is worried that if they run the terraform fmt command on their current directory, it will change their configuration files too much. What flag do you tell them to pass into the command such that they can see the differences?

✓
-diff

✗
-check

✗
-refresh

✗
-list=true

Explanation

Using *-diff* will display the changes between the current, and the to-be-formatted file.

Concept: Use the Terraform CLI (outside of core workflow)

https://www.terraform.io/docs/cli/commands/fmt.html#diff
#8

What is especially important to remember when configuring dynamic blocks?

✓
Limit the amount of dynamic blocks that are within your resource block as they can become hard to maintain and read.

✗
There's nothing important to remember about dynamic blocks.

✗
Dynamic blocks are only used within data sources.

✗
Dynamic blocks cannot be nested, and if they are they will likely cause terraform state issues.

Explanation

Overuse of dynamic blocks can make configuration hard to read and maintain, so we recommend using them only when you need to hide details in order to build a clean user interface for a re-usable module. Always write nested blocks out literally where possible.

Concept: Read, generate, and modify configuration

https://www.terraform.io/docs/language/expressions/dynamic-blocks.html#best-practices-for-dynamic-blocks
#9

Which terraform state subcommand will give you all of the resources in your state?

✓

list

✗

show

✗

refresh

✗

apply

Explanation

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

Concept: Use the Terraform CLI (outside of core workflow)

https://www.terraform.io/docs/cli/commands/state/index.html

#10

Where are Terraform Workspace state files stored?

✓

`terraform.tfstate.d`

✗

`terraform.tfstate`

✗

`.tfstate`

✗

`~/.terraform.d/plugins`

Explanation

Terraform stores the states for its workspaces under terraform.tfstate.d

https://www.terraform.io/docs/language/state/workspaces.html#workspace-internals

#11

How do you correctly reference a private registry module source?

✓

<HOSTNAME>/<NAMESPACE>/<NAME>/<PROVIDER>

✗

<NAMESPACE>/<NAME>/<PROVIDER>

✗

<HOSTNAME>/<NAMESPACE>/<PROVIDER>

✗

<NAMESPACE>/<NAME>/<PROVIDER>/<HOSTNAME>

Explanation

When specifying a source for a private registry, the correct Syntax is <HOSTNAME>/<NAMESPACE>/<NAME>/<PROVIDER>. It is different than the public registry because it includes the <HOSTNAME> field.

Concept: Interact with Terraform Modules

https://www.terraform.io/docs/registry/modules/use.html#private-registry-module-sources

#12

What is the recommended way to implement Terraform's state for larger teams?

✓

By configuring a remote backend such that multiple teams can work in tandem and know which resources are being created and destroyed.

✗

By sticking the state in a cloud instance, and having team members SSH into the instance to work on their configuration files.

✗

Having your state synced to a github repo for members to compare to.

✗

By using the daily standup you are a part of sot that you can share changes to the state file.

Explanation

Remote state is the recommended solution to this problem. With a fully-featured state backend, Terraform can use remote locking as a measure to avoid two or more different users accidentally running Terraform at the same time, and thus ensure that each Terraform run begins with the most recent updated state.

Concept: Understand Terraform's purpose (vs other IaC)

https://www.terraform.io/docs/language/state/purpose.html

#13

What are the two accepted values for provisioners that have the "on_failure" key specified? (Choose 2 answers)

✓

Continue

✓

Fail

✗

Abort

✗

Retry

Explanation

By default, provisioners that fail will also cause the Terraform apply itself to fail. The on_failure setting can be used to change this.

Concept: Understand Terraform Basics

https://www.terraform.io/docs/language/resources/provisioners/syntax.html#failure-behavior

#14

Which of the following statements about HashiCorp Sentinel is false?

✗
It was written for integration with Go.

✗
It can be executed locally.

✗
It can be executed in the cloud.

✓
It is open source.

Explanation

Sentinel is an enterprise offering, so it's not open source, but if we read some more there are Terraform cost estimators built into the Sentinel policy runtimes, which means that the enterprise offering that we're going to be paying for can quickly be recovered through building policies that estimate costs for our Terraform resources. And lastly, it can be executed in the cloud and locally. It was written for integration with Go and it's embedded in HashiCorp products, which means it's incredibly fast conceptually and practically; it takes less than an hour to learn, less than two to write.

/course/introduction-sentinel-policy-as-code-1257/sentinel-design/

#15

What are two ways terraform accepts plug-in installation? (Choose 2 answers)

✓
Local

✓
HTTPS

✗
Git

✗
s3

Explanation

Terraform's configuration file includes options for caching downloaded plugins, or explicitly specifying a local or HTTPS mirror to install plugins from. For more information, see CLI Config File.

Concept: Understand Terraform basics

https://www.terraform.io/docs/cli/plugins/index.html
#16

How would you reference a variable file with terraform apply?

✓
-var-file={PATH}

✗
-var-file-path={PATH}

✗
By declaring the variable file path in the terraform configuration block.

✗
By setting each one in your local environment with the variable having TF_VARS prepended to it.

Explanation

To set lots of variables, it is more convenient to specify their values in a variable definitions file (with a filename ending in either .tfvars or .tfvars.json) and then specify that file on the command line with -var-file

Concept: Interact with Terraform modules

https://www.terraform.io/docs/language/values/variables.html#variable-definitions-tfvars-files
#17

You are part of a large DevOps team using the current version of Terraform, and there can be multiple changes going on to your terraform files across the company. What would you do to ensure that the state file is locked when you run terraform apply?

✗
Add the *-lock=true* flag to the command.

✓
Nothing, terraform will manage the locking by itself.

✗
First run *terraform plan* to lock in your proposed changes. Then run *terraform apply* to commit them.

✗
Add the *-state-lock=true* to the command.

Explanation

If supported by your backend, Terraform will lock your state for all operations that could write state. This prevents others from acquiring the lock and potentially corrupting your state.

State locking happens automatically on all operations that could write state. You won't see any message that it is happening. If state locking fails, Terraform will not continue. You can disable state locking for most commands with the -lock flag but it is not recommended.

https://www.terraform.io/language/state/locking

#18

You have a Terraform variable that is declared as follows:variable "num" { default = 3 }You also have a defined the following environment variables in your BASH shell: num=1, TF_num=5, TF_VAR_num=10. You also have a terraform.tfvars file with the following contentsnum = 7When you run the following apply command, what is the value assigned to the num variable?terraform apply -var num=4

✓
4

✗
3

✗
1

✗
10

Explanation

The -var and -var-file methods of assigning variables have the highest precedence.

https://www.terraform.io/docs/configuration/variables.html

#19

How would you get the JSON output of the terraform validate command?

✓ *terraform validate -json*

✗ *terraform validate json*

✗ *terraform validate -output=json*

✗ *terraform json validate*

Explanation

-json produces output in a machine-readable JSON format, suitable for use in text editor integrations and other automated systems. Always disables color.

Concept: Navigate Terraform Workflow

https://www.terraform.io/docs/cli/commands/validate.html#usage

#20

What two configuration variables are available to a default local backend? (Choose 2 answers)

✓ path

✓ workspace_dir

✗ working_dir

✗

path_dest

Explanation

The local backend stores state on the local filesystem, locks that state using system APIs, and performs operations locally. It has two configuration variable accessible to it, and they are:

Path: (Optional) The path to the tfstate file. This defaults to "terraform.tfstate" relative to the root module by default.

Workspace_dir: (Optional) The path to non-default workspaces.

Both are optional.

Concept: Implement and Maintain State

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

#21

Say you wanted to increase the number of operations that terraform is concurrently using to create your resources. Which command would you run, with what specific flag, to accomplish this? (Choose 2 answers)

✓ *terraform apply*

✓ *-parallelism={NUMBER-OF-OPERATIONS}*

✗ *terraform init*

✗ *-concurrent={NUMBER-OF-OPERATIONS}*

Explanation

By default, Terraform runs with *parallelism* set to 10. If you wanted to increase or decrease this, you could alter the number of parallel operations by passing in the *parallelism* flag.

Concept: Navigate Terraform Workflow

https://www.terraform.io/docs/cli/commands/apply.html#parallelism-n

#22

By default, when running 'terraform plan', what files are scanned?

✓
All *.tf files in the current directory.

✗
Only files in the *.terraform* directory

✗
Only files you specify with the *-file-path* flag.

✗
All files on your hard drive.

Explanation

The `plan` subcommand looks in the current working directory for the root module configuration.

Important to note that it is possible to have a data directory other than the current working directory, but that will be deprecated in Terraform Version 15.

https://www.terraform.io/docs/cli/commands/plan.html
#23

What are three meta-arguments, along with source and version, that a module can use? (Choose 3 answers)

✓
for_each

✓
count

✗
max

✓
depends_on

Explanation

Along with source and version, Terraform defines a few more optional meta-arguments that have special meaning across all modules, described in more detail in the following pages:

count - Creates multiple instances of a module from a single module block. See the count page for details.

for_each - Creates multiple instances of a module from a single module block. See the for_each page for details.

providers - Passes provider configurations to a child module. See the providers page for details. If not specified, the child module inherits all of the default (un-aliased) provider configurations from the calling module.

depends_on - Creates explicit dependencies between the entire module and the listed targets. See the depends_on page for details.

In addition to the above, the lifecycle argument is not currently used by Terraform but is reserved for planned future features.

Concept: Interact with Modules

https://www.terraform.io/docs/language/modules/syntax.html#meta-arguments

#24

If an input variable has no type value set, what type does it accept?

✓

Any type.

✗

None, it has to have a type value set.

✗

Terraform infers the type when it is referenced.

✗

Type string. As strings can be interpreted in a number of ways by Terraform.

Explanation

The type argument in a variable block allows you to restrict the type of value that will be accepted as the value for a variable. If no type constraint is set then a value of any type is accepted.

Concept: Read, generate, and modify configuration

https://www.terraform.io/docs/language/values/variables.html#type-constraints

#25

Does the validate command connect to remote APIs and state when being ran?

✓

No it does not.

✗

Only if configured to do so on the backend.

✗

If the *-remote=true* is set, yes it does.

✗

If there are providers set, it will attempt to.

Explanation

Validate runs checks that verify whether a configuration is syntactically valid and internally consistent, regardless of any provided variables or existing state. It is thus primarily useful for general verification of reusable modules, including the correctness of attribute names and value types.

Concept: Navigate Terraform workflow

https://www.terraform.io/docs/cli/commands/validate.html

#26

What Terraform feature is most applicable for managing small differences between different environments, for example development and production?

✓

Workspaces

✗

Plans

✗

States

✕

Versions

Explanation

https://www.terraform.io/docs/state/workspaces.html

Covered in this lecture

State

Course:Managing Infrastructure With Terraform

5m

#27

What is required when configuring a partial backend configuration?

✓

That an empty backend is configured in the root of the Terraform configuration files.

✕

That the backend has the following: A configuration file, and a initialized state.

✕

The address, path, and scheme are all set in the configuration block for the backend.

✕

Terraform does not let you specify a partial configuration for backends.

Explanation

When using partial configuration, Terraform requires at a minimum that an empty backend configuration is specified in one of the root Terraform configuration files, to specify the backend type.

Concept: Implement and Maintain State

https://www.terraform.io/docs/language/settings/backends/configuration.html#partial-configuration

#28

What are Data Sources in terraform?

✓

Data to be fetched or computed for use elsewhere in terraform configuration.

✗

Similar to resources, they specify data to be created in the corresponding provider.

✗

A binary set of operators that tell resources how to behave with certain meta-arguments.

✗

Data sources are a way for terraform to keep track of all resources created in the provider's infrastructure.

Explanation

Data sources allow data to be fetched or computed for use elsewhere in Terraform configuration. Use of data sources allows a Terraform configuration to make use of information defined outside of Terraform, or defined by another separate Terraform configuration.

Concept: Read, generate, and modify configuration

https://www.terraform.io/docs/language/data-sources/index.html

#29

You just added a new set of resources to your configuration and would only like to see them when you run your 'terraform plan' command. What flag do you specify when running the plan command to only see their plans?

✓

-target={resources}

✗

-refresh=true

✗

-state={new_state_file}

✗

-lock=true

Explanation

The `-target` option can be used to focus Terraform's attention on only a subset of resources. Resource Address syntax is used to specify the constraint.

This means we can specify specific resources granularly when we want to see what will be planned.

https://www.terraform.io/docs/cli/commands/plan.html#resource-targeting

#30

What are some built-in functions that terraform provides? (Choose 3 answers.)

✓

max()

✓

regex()

✓

alltrue()

✗

delete()

Explanation

Max: Returns the highest integer of the provided arguments.

Regex: Applies a regular expression to a string and returns the matching substrings.

Alltrue: Returns true if all elements in a given collection are true/"true" or empty.

Concept: Read, generate, and modify configuration

https://www.terraform.io/docs/language/functions/index.html

#31

How can you init a directory with a specific source?

✓

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

✗
terraform init -source={PATH}

✗
terraform init {PATH}

✗
terraform init -plugin-dir={PATH}

Explanation

By default, terraform init assumes that the working directory already contains a configuration and will attempt to initialize that configuration.

Optionally, init can be run against an empty directory with the -from-module=MODULE-SOURCE option, in which case the given module will be copied into the target directory before any other initialization steps are run.

Concept: Navigate Terraform workflow

https://www.terraform.io/docs/cli/commands/init.html#copy-a-source-module

#32

What must be provided with the import command for Terraform to successfully import resources?

✓
The module path, and the resource spec.

✗
The resource name.

✗
The full resource ARN.

✗
Just the ID of the resource

Explanation

ADDRESS must be a valid resource address. Because any resource address is valid, the import command can import resources into modules as well as directly into the root of your state.

ID is dependent on the resource type being imported. For example, for AWS instances it is the instance ID (i-abcd1234) but for AWS Route53 zones it is the zone ID (Z12ABC4UGMOZ2N). Please reference the provider documentation for details on the ID format. If you're unsure, feel free to just try an ID. If the ID is invalid, you'll just receive an error message.

Resource Address Specification(s): [module path][resource spec]

Concept: Use the Terraform CLI (outside of core workflow)

https://www.terraform.io/docs/cli/commands/import.html

#33

How do you configure a backend?

✓

Through the backend block, under the terraform block.

✗

In the provider block.

✗

In the backend block.

✗

When you are running terraform apply, pass the *-backend={BACKEND_NAME}* flag.

Explanation

Backends are configured with a nested `backend` block within the top-level `terraform` block:

```
terraform {
  backend "remote" {
    organization = "example_corp"

    workspaces {
      name = "my-app-prod"
    }
  }
}
```

Concept: Implement and maintain state

https://www.terraform.io/docs/language/settings/backends/configuration.html

#34

You have an EC2 instance that is acting up in the cloud. It handles a relatively light ephemeral workload, so it can be restarted/destroyed with no repercussions. What full command would you use to target only this instance for recreation?

✓

terraform apply -replace=aws_intstance.{INSTANCE_NAME}

✗

terraform apply -replace aws_intstance

✗

terraform apply -replace {INSTANCE_NAME}

✗

terraform destroy --target=aws.instance{INSTANCE_NAME}

terraform apply

Explanation

-replace=ADDRESS - Instructs Terraform to plan to replace the resource instance with the given address. This is helpful when one or more remote objects have become degraded, and you can use replacement objects with the same configuratation to align with immutable infrastructure patterns. Terraform will use a "replace" action if the specified resource would normally cause an "update" action or no action at all. Include this option multiple times to replace several objects at once. You cannot use -replace with the -destroy option, and it is only available from Terraform v0.15.2 onwards. For earlier versions, use terraform taint to achieve a similar result.

https://developer.hashicorp.com/terraform/cli/commands/plan#replace-address

#35

Where does the 'local-exec' provisioner execute its code provided in its block?

✗

On the remote resource specified.

✓
On the local machine running terraform.

✗
On a spot-instance on your cloud provider.

✗
In a container on your machine provided by the Terraform binary.

Explanation

The `local-exec` provisioner invokes a local executable after a resource is created. This invokes a process on the machine running Terraform, not on the resource.

https://www.terraform.io/docs/language/resources/provisioners/local-exec.html

#36

What is a provider block without an alias meta argument?

✓
The default provider configuration.

✗
A broken provider configuration.

✗
A partial provider configuration.

✗
There must be an alias meta argument.

Explanation

A provider block without an alias argument is the default configuration for that provider. Resources that don't set the provider meta-argument will use the default provider configuration that matches the first word of the resource type name. (For example, an aws_instance resource uses the default aws provider configuration unless otherwise stated.)

Concept: Understand Terraform basics

https://www.terraform.io/docs/language/providers/configuration.html#default-provider-configurations

#37

Which configuration file formats are supported by Terraform? (Select all that apply)

✓
HCL

✓
JSON

✗
Ruby

✗
Go

Explanation

Terraform supports both HashiCorp Configuration Language (HCL) and JSON formats for configurations.

https://www.terraform.io/docs/configuration/
Covered in this lecture
Terraform Configuration
Course:Managing Infrastructure With Terraform

15m

#38

Given the following provider configurations, how would you reference the western provider in a resource configuration block? provider "aws" { region = "us-east-1" } provider "aws" { alias = "west" region = "us-west-2" }

✓
By providing the meta-argument:

provider: aws.west

Under the resource block.

✗

By prepending***aws.west***.*{RESOURCE_NAME}* to your resource name.

By adding the suffix *.aws.west* to your resource.

By adding the *alias* meta argument to your resource

Explanation

To create multiple configurations for a given provider, include multiple provider blocks with the same provider name. For each additional non-default configuration, use the alias meta-argument to provide an extra name segment.

Concept: Understand Terraform basics

https://www.terraform.io/docs/language/providers/configuration.html#alias-multiple-provider-configurations

#39

What are two complex types in terraform? (Choose 2 answers)

A Collection Type

A Structural Type

A String Type

A float64 type

Explanation

Collection and Structural types are the two types that are considered complex types in terraform. They are made up of values that are called element types.

Concept: Read, generate, and modify configuration

https://www.terraform.io/docs/language/expressions/type-constraints.html#complex-types

#40

When are output variables ran and sent to stdout?

✓
Only with terraform apply.

✗
Only on terraform plan or apply.

✗
With any terraform command.

✗
Only if you specify the *-outputs* flag on apply.

Explanation

Outputs are only rendered when Terraform applies your plan. Running terraform plan will not render outputs.

Concept: Interact with modules

https://www.terraform.io/docs/language/values/variables.html#declaring-an-output-value
#41

When specifying a module, what is the best practice for the implementation of the meta-argument version?

✓
The best practice is to explicitly set the version argument as a version constraint string from the Terraform registry.

✗
The best practice is to use no version and accept the latest version.

✗
The best practice is to download the module, place it in your working directory, then source that module, and specify the version that was downloaded.

✗

The best practice is to always ensure you append *beta* to the end of the version. This allows you and your team to always be working on the latest and greatest features for that module.

Explanation

Anywhere that Terraform lets you specify a range of acceptable versions for something, it expects a specially formatted string known as a version constraint.

A version constraint is a string literal containing one or more conditions, which are separated by commas.

E.g. version = ">= 1.3.1, < 2.0.0"

Concept: Interact with Terraform Modules

https://www.terraform.io/docs/language/modules/syntax.html#best-practices

#42

What does the following provisioner block specify? provisioner "local-exec" { when = destroy command = "echo 'Destroy-time provisioner'" } }

✓ Before the resource is destroyed, the provisioner will invoke `"echo 'Destroy-time provisioner'"`

✗ If the resource receives a *'destroy'* command locally, it will echo `'Destroy-time provisioner'`

✗ After the resource is destroyed, it will invoke `"echo 'Destroy-time provisioner'"`

✗ On the next *'terraform apply'* the resource will be destroyed

Explanation

If `when = destroy` is specified, the provisioner will run when the resource it is defined within is *destroyed*. Destroy provisioners are run before the resource is destroyed. If they fail, Terraform will error and rerun the provisioners again on the next `terraform apply`. Due to this behavior, care should be taken for destroy provisioners to be safe to run multiple times.

https://www.terraform.io/docs/language/resources/provisioners/syntax.html#destroy-time-provisioners
#43

What is the core terraform workflow?

✗
Plan, write, apply.

✓
Write, plan, apply.

✗
Apply, write, plan.

✗
Apply, plan, write.

Explanation

The core Terraform workflow has three steps:

Write - Author infrastructure as code.
Plan - Preview changes before applying.
Apply - Provision reproducible infrastructure.

Concept: Navigate Terraform workflow

https://www.terraform.io/guides/core-workflow.html
#44

What are the two types of backends? (Choose two.)

✓
Enhanced.

✓
Standard.

✗
Simple.

✕
Advanced.

Explanation

Terraform's backends are divided into two main types, according to how they handle state and operations:

- Enhanced backends can both store state and perform operations. There are only two types of enhanced backends: local and remote.
- Standard backends only store state, and rely on the local backend for performing operations.

Concept: Implement and Maintain State

https://www.terraform.io/docs/language/settings/backends/index.html#backend-types

#45

As a prestigious Sr. Cloud Engineer, your colleague comes up to you and asks for a new Development workspace. What's the fastest way to accomplish this?

✓
terraform workspace new dev

✕
Head to the Terraform Enterprise console and create a new workspace there.

✕
Specify in the configuration block the new workspace to be created.

✕
Have them submit a Jira ticket and tell them you'll get around to it in the next Sprint.

Explanation

Workspaces are managed with the `terraform workspace` set of commands. To create a new workspace and switch to it, you can use `terraform workspace new`; to switch workspaces you can use `terraform workspace select`; etc.

If you have Terraform Enterprise, you could head to the console and create a new workspace there. But since the question doesn't mention it, it's best not to assume that that's what needs to be done. The same goes for the configuration block answer.

https://www.terraform.io/docs/language/settings/backends/remote.html#workspaces
#46

You are a part of a growing Cloud Infrastructure team. Your boss comes up to you and asks you to transition the team off of local backends, and onto remote backends. She specifically mentions the fact that security is of the utmost concern, and that authentication needs to be prioritized. You are using Amazon S3 as a remote backend. Within terraform what do you do to ensure that the access to the S3 buckets are only accessed by members of your team? (Choose 2 answers)


Specify the file path to the API Key.


Make sure that the S3 bucket's bucket policy allows Put, Get, and List actions.


Export your AWS API key to *TF_BACKEND_KEY*


Encrypt your AWS buckets with SSE.

Explanation

Firstly, you must specify the correct file path to the required key. Secondly, ensuring that the correct permissions are set on the bucket such that terraform can correctly access it is required. After that, your organization can use S3 securely as a remote backend.

Concept: Implement and maintain state

https://www.terraform.io/docs/language/settings/backends/s3.html#example-configuration
#47

You are required to setup Terraform logs. Your boss asks you to make sure they always end up in one location such that they can be collected, and that they be set to the informational level. How would you accomplish this? (Choose 2 answers)


Export the environment variable of *TF_LOG* to be *INFO*

Export the *TF_LOG_PATH* environment variable to the requested path location.

Only invoke the *terraform apply* command in the location your boss wants the logs, because terraform automatically saves a *.log* file in the working directory.

Export the *TF_PATH_LOG* environment variable to the requested path location.

Explanation

First, you must specify the *TF_LOG* variable to *INFO*. After that, you'll need to make sure that the *TF_LOG_PATH* is set to the location your boss requires it be set to.

To persist logged output you can set TF_LOG_PATH in order to force the log to always be appended to a specific file when logging is enabled. Note that even when TF_LOG_PATH is set, TF_LOG must be set in order for any logging to be enabled.

Concept: Use the Terraform CLI (outside of core workflow)

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

#48

What are three Terraform Cloud features? (Choose 3 answers)

Remote state management.

Remote Terraform Execution.

Private Module Registry.

Terraform Linting.

Explanation

Remotes management of state, remote execution of Terraform, and the functionalities of a Private Module Registry are all Terraform Enterprise features.

Concept: Understand Terraform Cloud and Enterprise capabilities

https://www.terraform.io/docs/cloud/overview.html

#49

Your boss has asked you to come up with a new cloud automation provider that supports a Private Module registry as part of the offering. Which Cloud Provider and plan do you choose?

✓ Terraform Cloud with a Terraform Enterprise

✗ Amazon Web Services and the Enterprise Terraform Plus Plan

✗ The Azure Supercharged Automation Professional Direct plan from Microsoft

✗ The Google GCP Terraform Deluxe Plan

Explanation

Terraform Cloud's private module registry helps you share Terraform modules across your organization. It includes support for module versioning, a searchable and filterable list of available modules, and a configuration designer to help you build new workspaces faster.

https://www.terraform.io/docs/cloud/registry/index.html

#50

What is required to setup detailed Terraform logs?

✓ Setting the *TF_LOG* envrionment variable.

✗ Adding the meta-argument *log* in the Terraform main configuration block.

✗ Configuring each provider with the *log* meta-argument.

✗ Running *terraform log debug*

Explanation

Terraform has detailed logs which can be enabled by setting the TF_LOG environment variable to any value. This will cause detailed logs to appear on stderr.

Concept: Use the Terraform CLI (outside of core workflow)

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