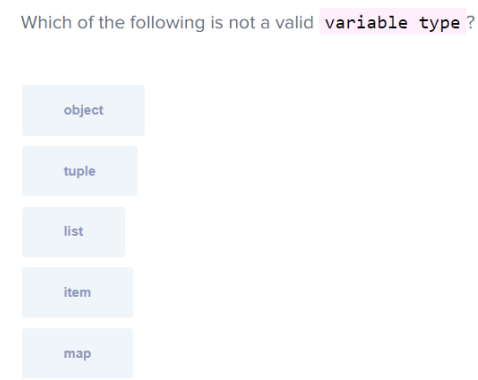
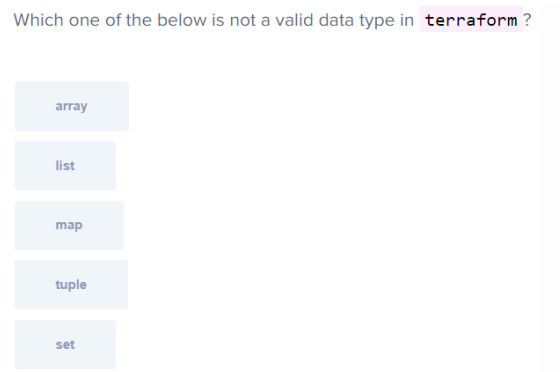


item .



array

.

Variable.tf:

```
variable "name" {
    type = string
    default = "Mark"
}
variable "number" {
    type = bool
    default = true
}
variable "distance" {
    type = number
    default = 5
}
variable "jedi" {
    type = map
    default = {
    filename = "/root/first-jedi"
    content = "phanius"
    }
}
variable "gender" {
    type = list(string)
    default = ["Male", "Female"]
}
variable "hard_drive" {
    type = map
    default = {
        slow = "HHD"
        fast = "SSD"
    }
}
variable "users" {
    type = set(string)
    default = ["tom", "jerry", "pluto", "daffy", "donald", "jerry", "chip", "dale"]
}
```

How would you fetch the value of the key called slow from the variable called hard_drive in a terraform configuration?

var.hard_drive["slow"]

What is the index of the element called Female in the variable called gender? 1
What is the type of variable called users? set(string)
However, this variable has been defined incorrectly! Identify the mistake.
The element called jerry is repeated twice. Remember, a set cannot have duplicate elements.

We have now updated the main.tf file in the same directory (/root/terraform-projects/variables) and added some resource blocks.

Main.tf

```
resource "local_file" "jedi" {
    filename = "/root/first-jedi"
    content = "phanius"
}
```

What is the value for the argument called content used in the resource block for the resource jedi?

Now, let's update this resource and add variables instead. Use the default value declared in the variable called jedi.
This variable is a map. For the argument called content use the value of the key by the same name.
And, similarly, for the argument called filename use the value by the same name.
When ready, run terraform init, plan and apply to create this resource.

Variable.tf

```
variable "jedi" {
    type = map
    default = {
    filename = "/root/first-jedi"
    content = "phanius"
    }
}
```

Main.tf

```
resource "local_file" "jedi" {
    filename = var.jedi["filename"]
    content = var.jedi["content"]
}
```

Terraform init , plan , apply .

```
iac-server $ terraform apply

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # local_file.jedi will be created
  + resource "local_file" "jedi" {
      + content              = "phanius"
      + content_base64sha256 = (known after apply)
      + content_base64sha512 = (known after apply)
      + content_md5          = (known after apply)
      + content_sha1         = (known after apply)
      + content_sha256       = (known after apply)
      + content_sha512       = (known after apply)
      + directory_permission = "0777"
      + file_permission      = "0777"
      + filename             = "/root/first-jedi"
      + id                   = (known after apply)
    }

Plan: 1 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

local_file.jedi: Creating...
local_file.jedi: Creation complete after 0s [id=23760ef64e0124aa08fecf101c302d3b64220d1a]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
```

How can we use environment variables to pass input variables in terraform scripts?
Export variables using the prefix TF_VAR_ followed by the variable name and a value.

Which method has the highest priority in Variable Definition Precedence?
Command line file –var .

Which one of the following commands is a valid way to make use of a custom variable file with the terraform apply command? terraform apply -var-file variables.tfvars

main.tf
```
resource local_file games {
  filename = var.filename
  content = "football"
}
```

terraform.tfvars
```
filename = "/root/football.txt"
```

throw.auto.tfvars
```
filename = "/root/baseball.txt"
```

basket.auto.tfvars
```
filename = "/root/basketball.txt"
```

What will happen if we run terraform plan command right now?
The configuration file uses a variable called filename which is not declared. As a result, the plan command will not work.

```
iac-server $ terraform plan

Warning: Value for undeclared variable

The root module does not declare a variable named "filename" but a value was
found in file "throw.auto.tfvars". To use this value, add a "variable" block
to the configuration.

Using a variables file to set an undeclared variable is deprecated and will
become an error in a future release. If you wish to provide certain "global"
settings to all configurations in your organization, use TF_VAR_...
environment variables to set these instead.


Error: Reference to undeclared input variable

  on main.tf line 2, in resource "local_file" "games":
   2:   filename = var.filename

An input variable with the name "filename" has not been declared. This
variable can be declared with a variable "filename" {} block.
```

The terraform plan command did not run as there was no reference for the input variable called filename in the configuration files.

Declare the variable called filename with type string in the file variables.tf.
Don't have to specify a default value.
variables.tf
```
variable filename {
  type = string
}
```

If we run terraform apply with a -var command line flag as shown below, which value would be considered by terraform?
terraform apply -var filename=/root/tennis.txt
/root/tennis.txt

Terraform follows a variable definition precedence order to determine the value and
the command line flag of –var or –var-file takes the highest priority.

Resource attributes

main.tf
resource "time_static" "time_update" {
}

What is the resource_type of the resource that's currently defined in the main.tf file?
As you can see, the resource block is empty. This is because time_static does not need any arguments to be supplied to work.
When applied as it is, terraform creates a logical resource locally (similar to random_pet) with the current time.

Which of the following attributes are exported by the time_static resource?
The resource exports an attribute called id. We can see the attributes for the resource once its created by using terraform commands

How do we refer to the attribute called id using a reference expression?
The syntax of the reference is resource_type.resource_name.attribute.
time_static.time_update.id

Now, update the main.tf file and add a new local_file resource called time with the following requirements:
filename: /root/time.txt
content: Time stamp of this file is <id from time_update resource>
Use a reference expression and interpolation.
When ready, run terraform init, plan and apply.

main.tf
```
resource "time_static" "time_update" {
}
resource "local_file" "time" {
 filename  = "/root/time.txt"
 content   = "Time stamp of this file is ${time_static.time_update.id}"
}
```

```
Plan: 2 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

time_static.time_update: Creating...
time_static.time_update: Creation complete after 0s [id=2023-04-28T08:15:27Z]
local_file.time: Creating...
local_file.time: Creation complete after 0s [id=8260585ee1909cecd89e531e37b1cceb8d402f2b]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
```

What is the attribute called id that is created for the local file resource called time?
Make use of the terraform show command and identify the attribute values.

What is the attribute called rfc3339 that is created for the time_static resource called time_update? Make use of the terraform show command and identify the attribute values.

Resource dependencies :

Which argument should be used to explicitly set dependencies for a resource? depends_on
Resource A relies on another Resource B but doesn't access any of its attributes in its own arguments. What is this type of dependency called? Explicit dependency

How do we make use of implicit dependency? When we use reference expressions to link resources, the dependency created is called implicit dependency .

In the configuration directory /root/terraform-projects/key-generator, create a file called key.tf with the following specifications:
Resource Type: tls_private_key
Resource Name: pvtkey
algorithm: RSA
rsa_bits: 4096
When ready, run terraform init, plan and apply.

main.tf :

```
resource "tls_private_key" "pvtkey" {
  algorithm = "RSA"
  rsa_bits  = 4096
}
```

```
iac-server $ terraform apply

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # tls_private_key.pvtkey will be created
  + resource "tls_private_key" "pvtkey" {
      + algorithm                   = "RSA"
      + ecdsa_curve                 = "P224"
      + id                          = (known after apply)
      + private_key_openssh         = (sensitive value)
      + private_key_pem             = (sensitive value)
      + private_key_pem_pkcs8       = (sensitive value)
      + public_key_fingerprint_md5  = (known after apply)
      + public_key_fingerprint_sha256 = (known after apply)
      + public_key_openssh          = (known after apply)
      + public_key_pem              = (known after apply)
      + rsa_bits                    = 4096
    }

Plan: 1 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

tls_private_key.pvtkey: Creating...
tls_private_key.pvtkey: Creation complete after 3s [id=5e10ac123fbb65d6bef9bb09ac828c4d
```

Resource tls_private_key generates a secure private key and encodes it as PEM. It is a logical resource that lives only in the terraform state.
You can see the details of the resource, including the private key by running the terraform show command.

```
iac-server $ terraform show
# tls_private_key.pvtkey:
resource "tls_private_key" "pvtkey" {
    algorithm                   = "RSA"
    ecdsa_curve                 = "P224"
    id                          = "5e10ac123fbb65d6bef9bb09ac828c4d27a95252"
    private_key_openssh         = (sensitive value)
    private_key_pem             = (sensitive value)
    private_key_pem_pkcs8       = (sensitive value)
    public_key_fingerprint_md5  = "b6:ea:0b:13:2e:5a:c9:10:eb:b0:b9:fb:87:4a:69:d5"
    public_key_fingerprint_sha256 = "SHA256:MG+U+bM3WKopwJ5WrmVkVOEKFYGoRBikWlJU1nj+0
    public_key_openssh          = <<~EOT
        ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAACAQDUBIDhPVhsaqKRM6rKaUp3LMBqp2oyx5Ad3owKJ
tLzdt0znkKxF50uRsVq0QAB81Or3VE2oN7a4/JhnWEY91v7/4qOBMo28RyUVGMd0Y5uRQOhOTIwCybc0LUOIy
XV4h0rKQsM52OyX//+i1poFvxp3I4mqEu2ZfKIj1boK8tgQEyXOzWj6ymfgca82mXHmUtQjBZlUg6QBu3/dap
qOnliZ65rJUMpW1a81b5VcDEiP8qNEgwwCpdhTHfHuMSuzdWCq/UjgPbB0x6goFdQkKvl8aqE+eUfV2AUjoKv
3qgtnMsfC4hbqg0lSxcj1+zBVriz+LJM0FHLx1zOTnNLaZH4kINk4drcLj1XMPmBlicnOAvFxQeUJHLoSRL3c
PKPAf0THGLUXEF+npiStfqRxDiDjAy2TdNSzVjiq4kLEkY05SS2nH6CnecOQ==
    EOT
    public_key_pem              = <<~EOT
        -----BEGIN PUBLIC KEY-----
        MIICIjANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEA1ASA4T1YbGqikTOqymlK
        dyzAaqdqMseQHd6MCiQVIH7rCDY/ZDoq2+rbexR0Rq+Qviu+EYFKjfAYZlW2lAMc
        5Db7S83bdM55CsRedLkbFatEAAfNTq91RNqDe2uPyYZ1hGPdb+/+KjgTKNvEclFR
        jHdGObkUDoTkyMAsm3NC1DiMnQKPlbXopIwnd27prW2tOEj4jPC+3AFTj1bU0TYa
        GVotsSp8Vl1eIdKykLDOdjsl///otaaBb8adyOJqhLtmXyiI9W6CvLYEBMlzs1o+
        spn4HGvNplx5lLUIwWZVIOkAbt/3WqQSXKoYVlzzDvaCpJmQlO2i/2E2DPIUNFOR
        qCCEMxjBtpge6gHajp5YmeuayVDKVtWvNW+VXAxIj/KjRIMMAqXYUx3x7jErs3Vg
        qv1I4D2wdMeoKBXUJCr5fGqhPnlH1dgFI6CrwEGIQk/R86tPl++eCf1cs0e9fD0z
        toAxIDBMpCwEjmkUJT4Gkd6oLZzLHwuIW6oNJUsXI9fswVa4s/iyTNBRy8dczk5z
```

Now, let's use the private key created by this resource in another resource of type local file. Update the key.tf file with the requirements:

Resource Name: key_details
File Name: /root/key.txt
Content: use a reference expression to use the attribute called private_key_pem of the pvtkey resource.
When ready, run terraform init, plan and apply.

key.tf

```
resource "tls_private_key" "pvtkey" {
  algorithm = "RSA"
  rsa_bits  = 4096
}

resource "local_file" "key_details" {
  filename  = "/root/key.txt"
  content   = tls_private_key.pvtkey.private_key_pem
}
```

```
iac-server $ terraform apply
tls_private_key.pvtkey: Refreshing state... [id=5e10ac123fbb65d6bef9bb09ac828c4d27a95252]

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # local_file.key_details will be created
  + resource "local_file" "key_details" {
      + content              = "{tls_private_key.pvtkey.private_key_pem}"
      + content_base64sha256 = (known after apply)
      + content_base64sha512 = (known after apply)
      + content_md5          = (known after apply)
      + content_sha1         = (known after apply)
      + content_sha256       = (known after apply)
      + content_sha512       = (known after apply)
      + directory_permission = "0777"
      + file_permission      = "0777"
      + filename             = "/root/key.txt"
      + id                   = (known after apply)
    }

Plan: 1 to add, 0 to change, 0 to destroy.

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

local_file.key_details: Creating...
local_file.key_details: Creation complete after 0s [id=abcc70e38206b566b638797d88128ce3fe8e8b93]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
iac-server $
```

Now destroy these two resources.
Use terraform destroy.

```
iac-server $ terraform destroy
tls_private_key.pvtkey: Refreshing state... [id=5e10ac123fbb65d6bef9bb09ac828c4d27a95252]
local_file.key_details: Refreshing state... [id=d1dd9bff3a67053cc7b700b6d16234cdda6bd394]

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  - destroy

Terraform will perform the following actions:

  # local_file.key_details will be destroyed
  - resource "local_file" "key_details" {
      - content              = <<~EOT
            -----BEGIN RSA PRIVATE KEY-----
            MIIJJwIBAAKCAgEA1ASA4T1YbGqikTOqymlKdyzAaqdqMseQHd6MCiQVIH7rCDY/
            ZDoq2+rbexR0Rq+Qviu+EYFKjfAYZlW2lAMc5Db7S83bdM55CsRedLkbFatEAAfN
            Tq91RNqDe2uPyYZ1hGPdb+/+KjgTKNvEclFRjHdGObkUDoTkyMAsm3NC1DiMnQKP
            lbXopIwnd27prW2tOEj4jPC+3AFTj1bU0TYaGVotsSp8Vl1eIdKykLDOdjsl///o
            taaBb8adyOJqhLtmXyiI9W6CvLYEBMlzs1o+spn4HGvNplx5lLUIwWZVIOkAbt/3
            WqQSXKoYVlzzDvaCpJmQlO2i/2E2DPIUNFORqCCEMxjBtpge6gHajp5YmeuayVDK
            VtWvNW+VXAxIj/KjRIMMAqXYUx3x7jErs3Vgqv1I4D2wdMeoKBXUJCr5fGqhPnlH
            1dgFI6CrwEGIQk/R86tPl++eCf1cs0e9fD0ztoAxIDBMpCwEjmkUJT4Gkd6oLZzL
            HwuIW6oNJUsXI9fswVa4s/iyTNBRy8dczk5zS2mR+JCDZOHa3C49VzD5gZYnJzgL
            xcUHlCRy6EkS96ub6DGMUcnfNLUAOJgdoUsXk5+ZZK8DI2cjFcQ0g6AGXSjMXkXT
            yjwH9Exxi1FxBfp6YkrX6kcQ4g4wMtk3TUs1Y4quJCxJGNOUktpx+gp3nDkCAwEA
            AQKCAgBY2tJQk28dcMtU4zxuvBXTQFQb3rHpOi2x9vTlwx1/kvFZbbK/hIk1xYXa
```

Within this directory, create two local_file type resources in main.tf file.
Resource 1:
Resource Name: whale
File Name: /root/whale
content: whale

Resource 2:
Resource Name: krill
File Name: /root/krill
content: krill

Resource called whale should depend on krill but do not use reference expressions.
When ready, run terraform init, plan and apply.

Use explicit dependency using depends_on.

```
resource "local_file" "whale" {
 filename   = "/root/whale"
 content    = "whale"
 depends_on  =  [local_file.krill]
}

resource "local_file" "krill" {
 filename  = "/root/krill"
 content   = "krill"
}
```

```
iac-server $ terraform apply

An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # local_file.krill will be created
  + resource "local_file" "krill" {
      + content              = "krill"
      + content_base64sha256 = (known after apply)
      + content_base64sha512 = (known after apply)
      + content_md5          = (known after apply)
      + content_sha1         = (known after apply)
      + content_sha256       = (known after apply)
      + content_sha512       = (known after apply)
      + directory_permission = "0777"
      + file_permission      = "0777"
      + filename             = "/root/krill"
      + id                   = (known after apply)
    }

  # local_file.whale will be created
  + resource "local_file" "whale" {
      + content              = "whale"
      + content_base64sha256 = (known after apply)
      + content_base64sha512 = (known after apply)
      + content_md5          = (known after apply)
      + content_sha1         = (known after apply)
      + content_sha256       = (known after apply)
      + content_sha512       = (known after apply)
      + directory_permission = "0777"
      + file_permission      = "0777"
      + filename             = "/root/whale"
      + id                   = (known after apply)
    }
```

Since we specified explicit dependency , krill is created first but in configuration its mentioned after whale .

Output variables :

main.tf

```
resource "random_uuid" "id1" {

}
resource "random_uuid" "id2" {

}
resource "random_uuid" "id3" {

}
resource "random_uuid" "id4" {

}
resource "random_uuid" "id5" {

}
```

```
resource "random_uuid" "id6" {

}
resource "random_uuid" "id7" {

}
resource "random_integer" "order1" {
  min     = 1
  max     = 99999

}
resource "random_integer" "order2" {
  min     = 1
  max     = 222222

}
```

Output.tf

```
output "id1" {
   value = random_uuid.id1.result
}
output "id2" {
    value = random_uuid.id2.result

}
output "id3" {
    value = random_uuid.id3.result

}

output "id4" {
    value = random_uuid.id4.result

}
output "id5" {
    value = random_uuid.id5.result
}

output "id6" {
    value = random_uuid.id6.result

}
output "id7" {
    value = random_uuid.id7.result

}
output "order1" {
 value = random_integer.order1.result

}
output "order2" {
 value = random_integer.order1.result

}
```

Which provider is used by the configuration files in this directory? Random
Which two resource types are configured in the configuration files? Uuid and integer

What is the value of the output variable called id2 ?

```
iac-server $ terraform output
id1 = 691c5b48-6fa1-1ad6-2140-44a479628321
id2 = 8f16e8d8-9117-ea78-7506-24e02bef3600
id3 = 263d9839-135b-97c0-306f-13c83a2c990b
id4 = a3793f4b-bf89-63a5-0a69-2f34b3fbec0e
id5 = e1b8ec99-ecaa-6b6a-db93-301a14c5d824
id6 = 1f2f4883-207a-42e3-70ee-90c6cd2b3f4a
id7 = 15df5edf-9c60-2e44-a918-55af1e8f7ba4
order1 = 27288
order2 = 27288
```

What is the value of the output variable called order1 ?

We have a new configuration directory located at the path /root/terraform-projects/output. Inspect the configuration files that are created in this directory.
What is the value of the output variable pet-name ?

```
iac-server $ pwd
/root/terraform-projects/data
iac-server $ cd ..
iac-server $ ls
data  output
iac-server $ cd output/
iac-server $ ls
main.tf  terraform.tfstate  variable.tf
iac-server $ terraform output
pet-name = cow
iac-server $
```

We have just updated the main.tf file in this directory with a new resource block.
Add a new output variable with the following specifications:
Output Variable Name: welcome_message
Value: content of the resource called welcome
When ready, run terraform init, plan and apply

Main.tf

```
resource "random_pet" "my-pet" {

  length    = var.length
}

output "pet-name" {

            value = random_pet.my-pet.id
            description = "Record the value of pet ID generated by the random_pet resource"
}

resource "local_file" "welcome" {
    filename = "/root/message.txt"
    content = "Welcome to Kodekloud."
}

output "welcome_message" {
  value = local_file.welcome.content
}
```

Variables.tf

```
variable "prefix" {
  default = "Mrs"
}

variable "separator" {
          default = "."
}

variable "length" {
          default = "1"
}
```

```
Terraform will perform the following actions:

  # local_file.welcome will be created
  + resource "local_file" "welcome" {
      + content              = "Welcome to Kodekloud."
      + content_base64sha256 = (known after apply)
      + content_base64sha512 = (known after apply)
      + content_md5          = (known after apply)
      + content_sha1         = (known after apply)
      + content_sha256       = (known after apply)
      + content_sha512       = (known after apply)
      + directory_permission = "0777"
      + file_permission      = "0777"
      + filename             = "/root/message.txt"
      + id                   = (known after apply)
    }

Plan: 1 to add, 0 to change, 0 to destroy.

Changes to Outputs:
  + welcome_message = "Welcome to Kodekloud."

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

local_file.welcome: Creating...
local_file.welcome: Creation complete after 0s [id=d2d3e44fe87af01e8f96

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.

Outputs:

pet-name = cow
```