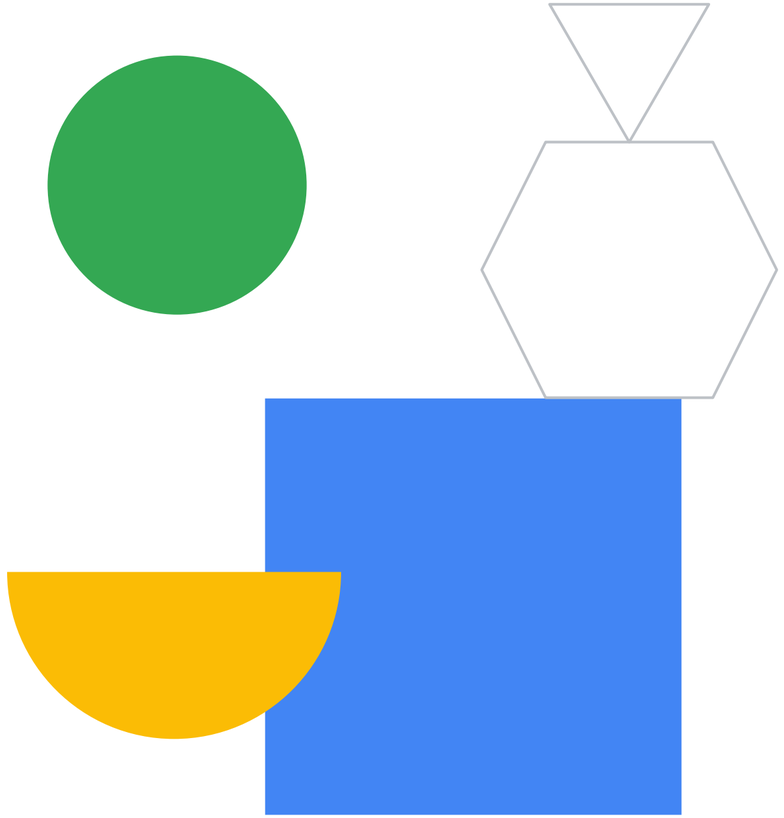
Google Cloud

Writing Infrastructure Code for Google Cloud



Objectives

Upon completion of this module, you will be able to:

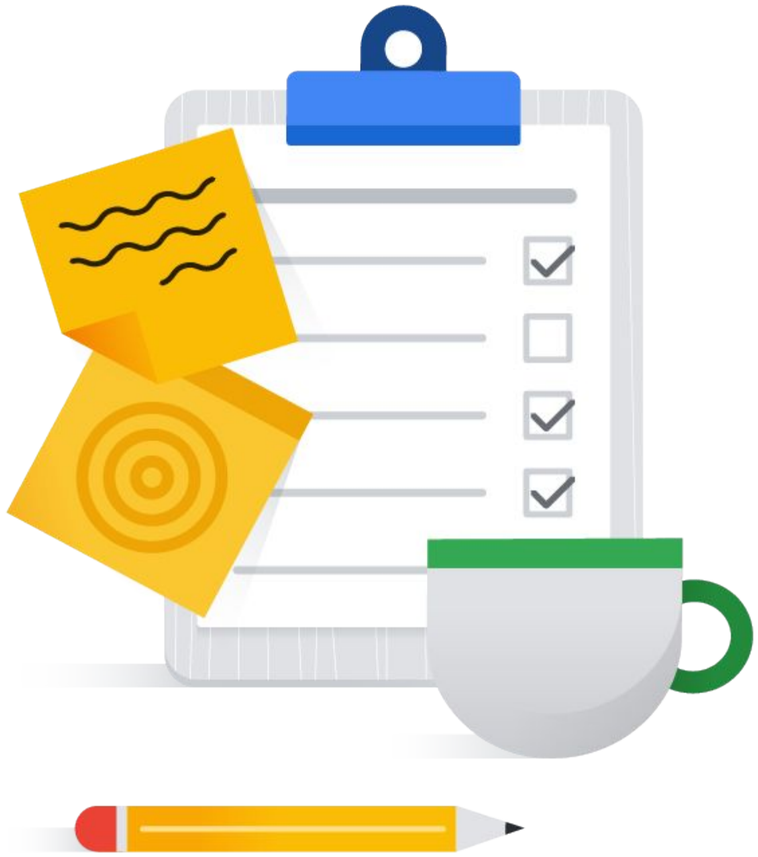
01 Declare the resources within Terraform.

02 Explain implicit and explicit resource dependencies.

03 Use variables and output values within the root configuration.

04 Explain Terraform Registry and Cloud Foundation Toolkit.

Topics

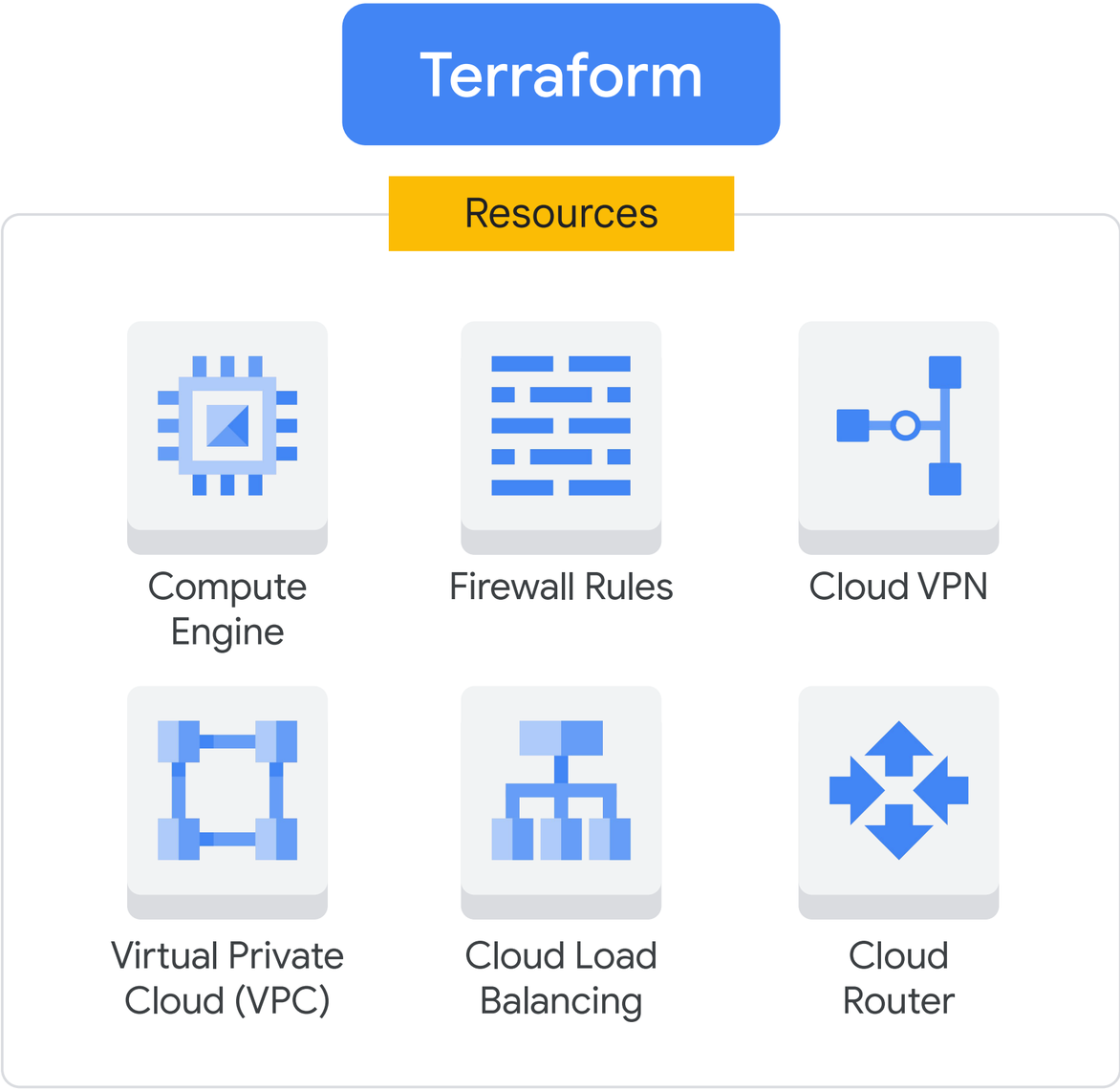


What are resources?

- Resources are infrastructure elements you can configure using Terraform.

  Examples:
  Compute Engine instance, VPC, Cloud Storage bucket, Firewall rules

- Terraform uses the underlying APIs of each Google Cloud service to deploy your resources.



Syntax to declare a resource

```
-- network/
   -- main.tf
   -- outputs.tf
   -- variables.tf
```

```
resource "resource_type" "resource_name" {
  #Resource arguments
}
```

- Resources are defined within a .tf file.
- The resource block represents a single infrastructure object.
- The `resource type` identifies the type of resource being created.
- The `resource type` depends on the provider being declared within a terraform module.
- Not all resource arguments must be defined.

Examples of a resource block

```
-- network/
   -- main.tf
   -- outputs.tf
   -- variables.tf
```

```
resource "google_compute_network" "vpc_network" {
   name                    = "vpc-network"  #Required argument
   project                 = "<Project_ID>"
   auto_create_subnetworks = false
   mtu                     = 1460
}
```

```
resource "google_compute_subnetwork" "subnetwork-ipv6" {
   name                    = "ipv6-test-subnetwork"
   ip_cidr_range           = "10.0.0.0/22"
   region                  = "us-west2"
   network                 = google_compute_network.vpc_network.id
}
```

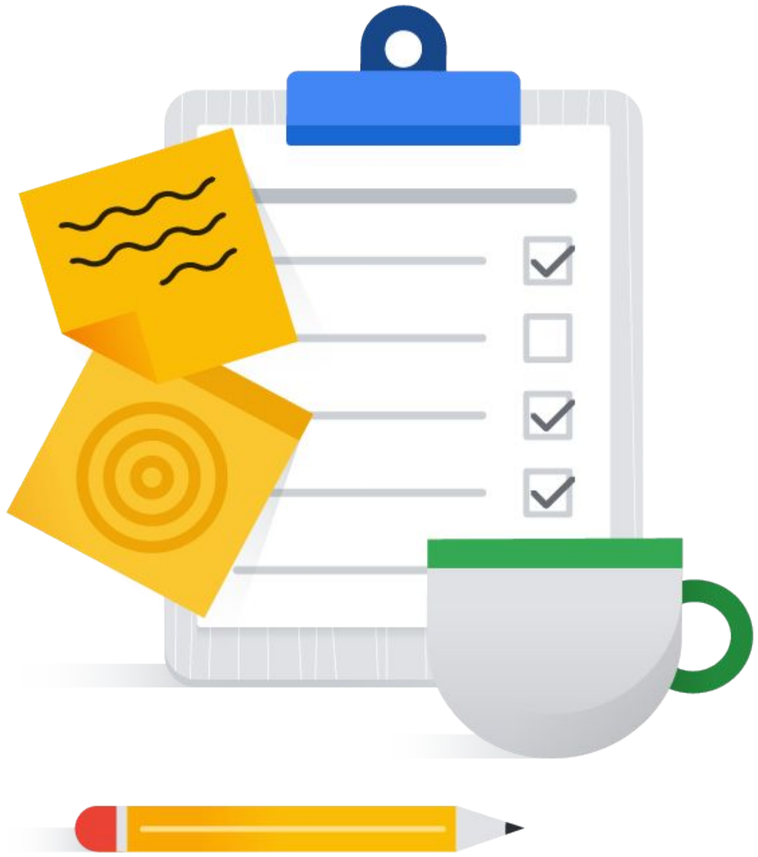
Syntax for referring to a resource attribute

-- main.tf

```
resource "google_compute_network" "vpc_network" {
   project                 = "my-project-name"
   name                    = "vpc-network"
   auto_create_subnetworks = false
   mtu                     = 1460
}

resource "google_compute_subnetwork" "subnetwork-ipv6" {
  name                     = "ipv6-test-subnetwork" #Required argument
  ip_cidr_range            = "10.0.0.0/22" #Required argument
  region                   = "us-west2"
  network                  = google_compute_network.vpc_network.id
}
```

- Use **`<resource_type>.<resource_name>.<attribute>`** to access any resource attributes.
- The Network ID is a computed resource attribute of a **`google_compute_network`** block.

Topics



Considerations for defining a resource block

- The resource name of a given resource type must be unique within the module.

A resource with the same name cannot exist within the same configuration.

```
-- network/
   -- main.tf
   -- variables.tf
   -- outputs.tf
```

```
resource "google_storage_bucket" "dev_bucket" {
  name     = "<unique_bucket_name>"
  location = "US"
}

resource "google_storage_bucket" "dev_bucket" {
  name     = "<unique_bucket_name>"
  location = "US"
}
```

Duplicated

Considerations for defining a resource block

- The resource name of a given resource type must be unique within the module.
- The resource type is not user-defined and is based on the provider.

```
resource "cloud_storage_bucket" "dev_bucket" {
  name          = "<unique_bucket_name>"
  location      = "US"
}
```

User-defined names are not allowed.

```
resource "google_storage_bucket" "dev_bucket" {
  name          = "<unique_bucket_name>"
  location      = "US"
}
```

The resource type is a keyword and must match the term mentioned in Terraform Registry.

Considerations for defining a resource block

- The resource name of a given resource type must be unique within the module.
- The resource type is not user-defined and is based on the provider.
- All configuration arguments must be enclosed within the resource block.

```
resource "google_storage_bucket" "dev_bucket" {
   name           = "<unique_bucket_name>"
}
   location       = "US"
```

```
Error: Unsupported argument

   On main.tf line 4:
   4: location       = “US”

An argument named “location” is not expected here.
```

Considerations for defining a resource block

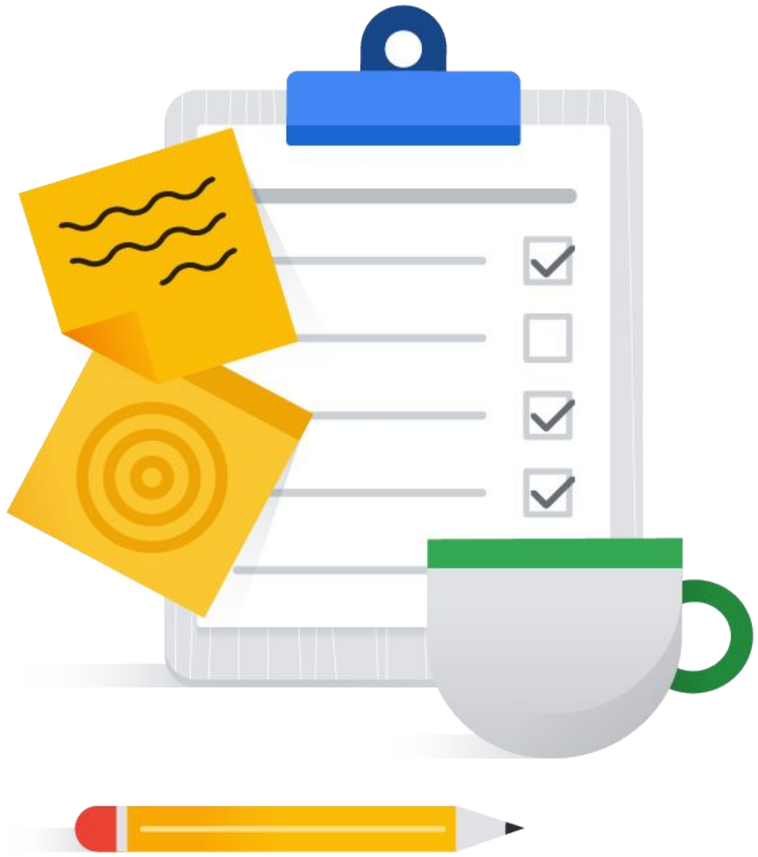
- The resource name of a given resource type must be unique within the module.
- The resource type is not user-defined and is based on the provider.
- All configuration arguments must be enclosed within the resource block.
- All required resource arguments must be defined.

```
resource "google_storage_bucket" "dev_bucket" {
location    = "US" #The name argument is missing
}
```

```
resource "google_storage_bucket" "dev_bucket" {
name        = "<unique_bucket_name>"
location    = "US"
}
```

```
Error: Missing required argument

   On main.tf line 9, in resource “google_storage_bucket”
   “mybucket1”:
   9: resource “google_storage_bucket” “dev_bucket” {

The argument "name" is required, but no definition was
found.
```

Topics



Variables overview

- Variables parameterize your configuration without altering the source code.
- Variables allow you to assign a value to the resource attribute at run time.
- Variables separate source code from value assignments.

```
resource "google_storage_bucket" "mybucket1" {
   name             = "my-project-name" #Required argument
   location         = "US"
   storage_class    = “standard”
}
```

Without variables, resource arguments are hardcoded within the configuration.

Syntax to declare an input variable

```
-- server/
   -- main.tf
   -- outputs.tf
   -- variables.tf
```

Syntax

```
variable "variable_name" {
   type          = <variable_type>
   description   = "<variable description>"
   default       = "<default value for variable>"
   sensitive     = true
}
```

Example: Variable for a bucket region. The variable name must be unique within a module.

```
variable "bucket_region" {
   type        = string
   description = "Region for the bucket"
   default     = "US"
   sensitive   = true
}
```

Variable arguments: type

```
-- server/
   -- main.tf
   -- outputs.tf
   -- variables.tf
```

```
variable "variable_name" {
   #Arguments
   type        = <variable_type>
   default     = "<default value for variable>"
   description = "<variable description>"
   sensitive   = true
}
```

```
variable "bucket_region" {
   type        = string
}
```

Variable arguments: `default`

```
-- server/
   -- main.tf          m
   -- outputs.tf
   -- variables.tf     v
```

When no value is set for the variable, the value specified in the `default` argument is used.

```
$terraform plan
Terraform will perform the following actions:

 # google_storage_bucket.mybucket1 will be created
  + resource "google_storage_bucket" "mybucket1" {
      + location                    = "US"
      + name                        = "<unique_bucket_name>"
      + project                     = (known after apply)
      + storage_class               = "REGIONAL"
..

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

m

```
resource "google_storage_bucket" "mybucket1" {
name           = "<unique_bucket_name>"
location       =  "US"
storage_class =  var.bucket_storage_class
//No value assigned for the storage class
}
```

v

```
variable "bucket_storage_class" {
type     = string
default  = "REGIONAL"
}
```

Variable arguments: description

```
-- server/
   -- main.tf
   -- outputs.tf
   -- variables.tf
```

```
variable "variable_name" {
    type          = <variable_type>
    default       = “<default value>"
    description   = “<variable description>”
    sensitive     = true
}
```

```
$terraform plan

var.bucket_region
  Specify the bucket region.

  Enter a value:
```

```
variable "bucket_region" {
    type          = string
    description   = “Specify the bucket region.”
}
```

The description will be displayed at run time, when no value is assigned for the variable.

Variable arguments: `sensitive`

```
-- server/
   -- main.tf
   -- outputs.tf
   -- variables.tf
```

- The acceptable value for `sensitive` is `true`.
- When set to `true`, the value is marked `sensitive` at run time.

```
variable "variable_name" {
    type          = <variable_type>
    default       = “<default value>”
    description   = “<variable description>”
    sensitive     = true
}
```

```
$terraform plan
Terraform will perform the following actions:

  # some_resource.a will be created
  + resource "some_resource" "example_resource" {
      + name    = (sensitive)
      + address = (sensitive)
    }

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

```
variable "user_information" {
    type = object({
        name    = string
        address = string
      })
  sensitive = true
}

resource "some_resource" "foo" {
    name    = var.user_information.name
    address = var.user_information.address
}
```

Syntax to reference and assign a value to a variable

Various ways to assign values to variables

- .tfvars files: Useful for quickly switching between sets of variables and versioning them.
- CLI options: Useful when running quick examples on simple files.
- Environment variables: Useful in scripts and pipelines.
- CLI prompt: If a required variable has not been set via one of the above.

```
# .tfvars file (Recommended method)
$terraform apply -var-file my-vars.tfvars

# CLI options
$terraform apply -var project_id="my-project"

# environment variables
$TF_VAR_project_id="my-project" \
  terraform apply

# If using terraform.tfvars
$terraform apply
```

Assign a value to the variable

Using the terraform.tfvars file

```
-- server/
   -- main.tf
   -- outputs.tf
   -- terraform.tfvars      t
   -- variables.tf          v
```

v

```
 variable "mybucket_storage_class" {
    type         = string
}

variable “bucket_region” {
    type = string
}
```

t

```
mybucket_storage_class = "REGIONAL"
bucket_region = "US"
```

Recommended method

Assign a value to the variable

Using the `-var` option

```
-- server/
   -- main.tf
   -- outputs.tf
   -- my-vars.tf          M
   -- terraform.tfvars    T
   -- variables.tf
```

T

```
mybucket_storage_class = "COLDLINE"
bucket_region = "US"
```

M

```
mybucket_storage_class = "NEARLINE"
bucket_region = "EU"
```

```
$cd /server
$terraform apply -var="mybucket_storage_class=REGIONAL”

$terraform apply -var-file my-vars
```

Assign a value to the variable

At run time

```
-- server/
   -- main.tf          M
   -- outputs.tf
   -- variables.tf     V
```

```
$terraform plan

var.mybucket_storageclass
  Set the storage class to the bucket.

  Enter a value:
```

Set the value of the storage class when running **`terraform plan`**.

M

```
resource "google_storage_bucket" "mybucket" {
  name           = "student102345688493"
  location       = var.bucket_region
  storage_class  = var.mybucket_storageclass
}
```

V

```
variable "mybucket_storageclass" {
    type         = string
    description  = "Set the storage class to the bucket."
}
```

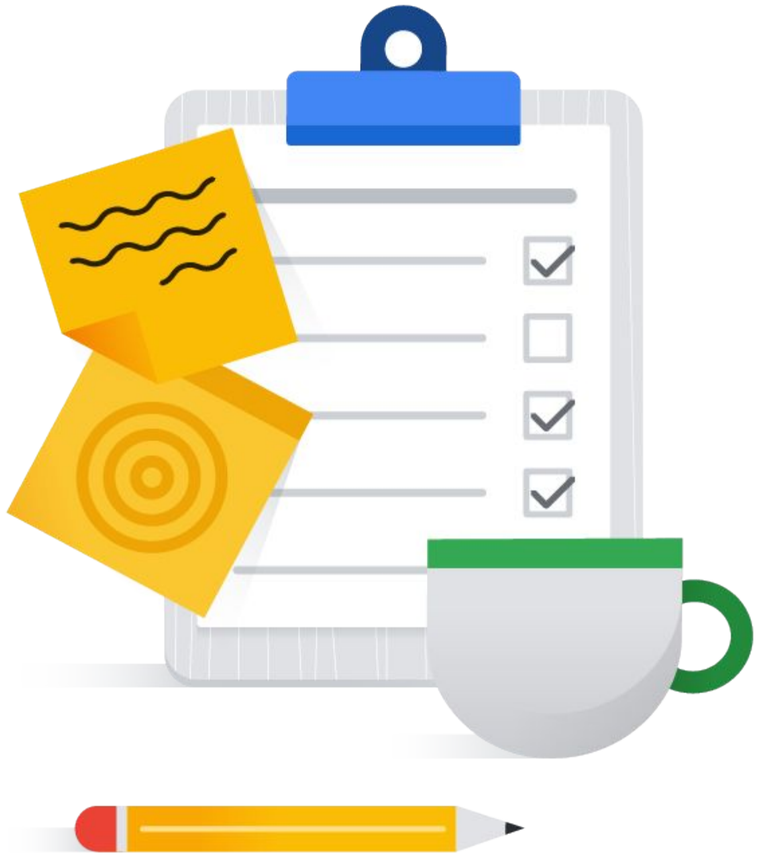
Validate variable values by using rules

```
variable "mybucket_storageclass" {
  type        = string
  description = "Set the storage class to the bucket."
  validation {
    condition = contains(["STANDARD", "MULTI_REGIONAL", "REGIONAL"], var.storageclass)
    error_message = "Allowed storage classes are STANDARD, MULTI_REGIONAL and REGIONAL."
  }
}
```

```
var.mybucket_storageclass
  Set the storage class to the bucket.
  Enter a value: ZONAL
google_compute_network.vpc_network: Refreshing state...

| Error: Invalid value for variable
|
|   on main.tf line 1:
|    1: variable "mybucket_storageclass" {
|
| Allowed storage classes are STANDARD, MULTI_REGIONAL and REGIONAL.
|
| This was checked by the validation rule at main.tf:4,3-13.
```

Validates the value assigned to the variable.

Topics



Parameterize only when necessary

```
-- server/
   -- main.tf
   -- outputs.tf
   -- terraform.tfvars
   -- variables.tf
```

- Only parameterize values that must vary for each instance or environment.
- Changing a variable with a default value is backward-compatible.
- Removing a variable is *not* backward-compatible.

Provide values in a .tfvars file

```
-- server/
   -- main.tf
   -- outputs.tf
   -- terraform.tfvars
   -- variables.tf
```

```
mybucket_storage_class = "REGIONAL"
bucket_region = "US"
```

Command-line options are ephemeral in nature and cannot be checked into source control.

```
$cd /server
$terraform apply -var="mybucket_storage_class=REGIONAL"
$terraform apply -var="bucket_region=US"
$terraform apply -var-file my-vars.txt
```

Give descriptive names to variables

```
-- server/
   -- main.tf
   -- outputs.tf
   -- terraform.tfvars
   -- variables.tf
```

```
variable "ram_size_gb" {
   type        = number
   description = "RAM size in GB."
}
```

```
variable "ram_size" {
   type        = number
   description = "RAM size in GB."
}
```

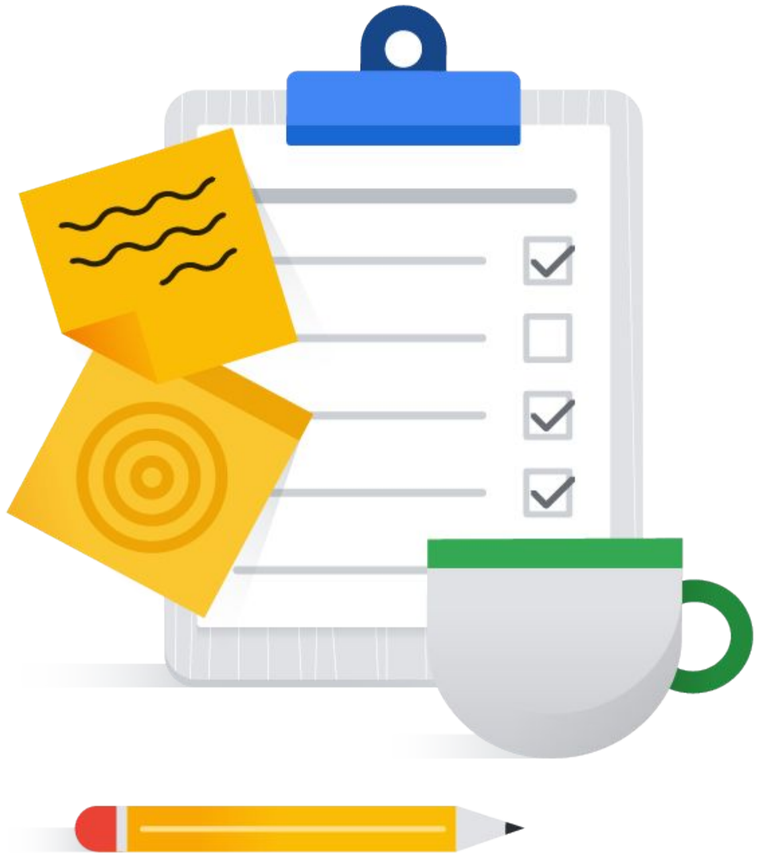
Provide meaningful descriptions

```
-- server/
   -- main.tf
   -- outputs.tf
   -- terraform.tfvars
   -- variables.tf
```

```
variable "bucket_region" {
   type          = string
   default       = “US”
   description   = “Specify the bucket region.”
}
```

```
variable "myregion" {
   type          = string
   default       = “US”
   description   = “Specify the region.”
}
```

Topics



Meta-arguments customize the behavior of resources

`count`	Create multiple instances according to the value assigned to the count.
`for_each`	Create multiple resource instances as per a set of strings.
`depends_on`	Specify explicit dependency.
`lifecycle`	Define life cycle of a resource.
`provider`	Select a non-default provider configuration.

count: Multiple resources of the same type

Redundant code

```
resource "google_compute_instance" "dev_VM1" {
  name          = "dev_VM1”
…}

resource "google_compute_instance" "dev_VM2" {
  name          = "dev_VM2”
…}

resource "google_compute_instance" "dev_VM3" {
  name          = "dev_VM3”
…}
```

Creates three instances of the same type with names:

- dev_VM1
- dev_VM2
- dev_VM3

```
resource "google_compute_instance" "Dev_VM" {
   count        = 3
   name           = "dev_VM${count.index + 1}"
 #other required arguments
…
}
```

for_each: Multiple resources of the same type with distinct values

Redundant code

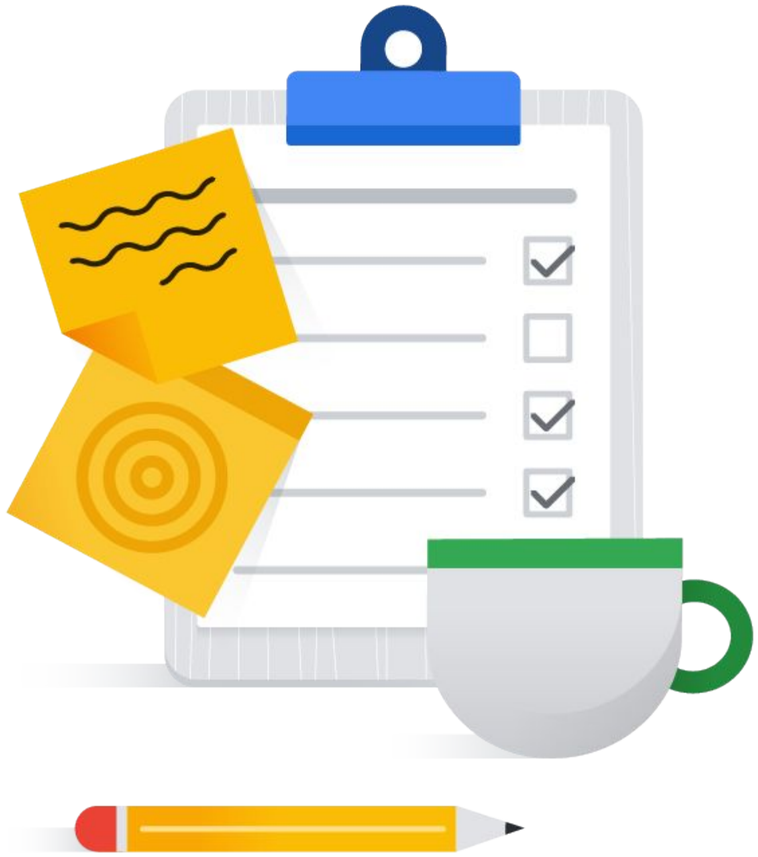
```
resource "google_compute_instance" "VM1" {
 name     = "dev-us-central1-a”
 location = “us-central1-a”
..
resource "google_compute_instance" "VM2" {
 name     = "dev-asia-east1-b”
 location = “asia-east1-b”
..
resource "google_compute_instance" "VM3" {
 name     = "dev-europe-west4-a”
 location = “europe-west4-a”
..
```

Creates three instances with the names:

- dev-us-central1-a
- dev-asia-east1-b
- europe-west4-a

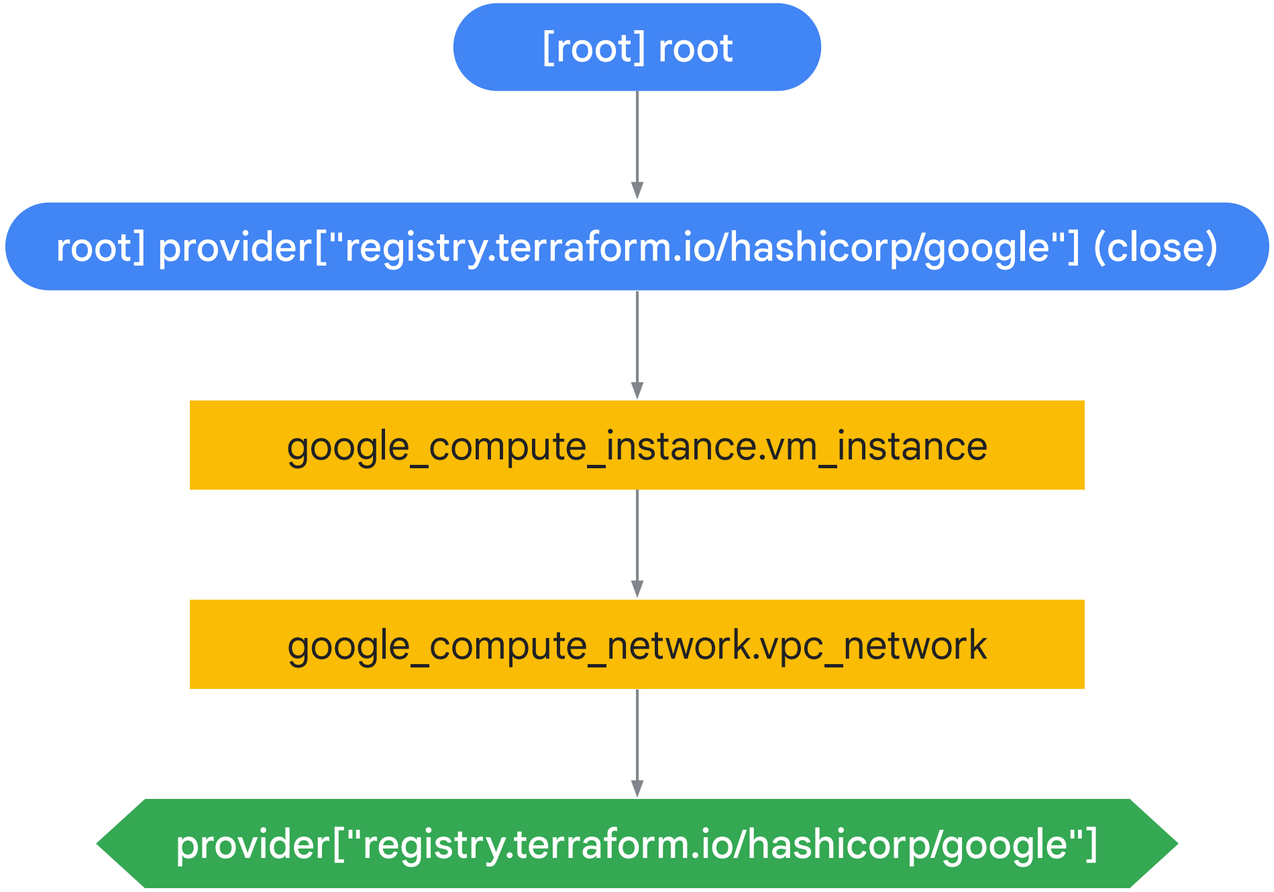
```
resource "google_compute_instance" "dev_VM" {
for_each = toset( ["us-central1-a", "asia-east1-b", "europe-west4-a"] )
name     = "dev-${each.value}"

zone     = each.value
#other required arguments
}
```

Topics



Dependency graph

- Built from the terraform configurations.
- Interpolates attributes during run time.
- Determines the correct order of operations.



Resource dependencies

Terraform can handle two kinds of dependencies.

Implicit dependency	Explicit dependency
Dependencies **known** to Terraform are detected automatically.	Dependencies **unknown** to Terraform must be configured explicitly.

Implicit resource dependencies are handled automatically

```
resource "google_compute_instance" "my_instance" {
  //All mandatory arguments

  network_interface {
   //implicit dependency
    network = google_compute_network.my_network.name
    access_config {
    }
  }
}

resource "google_compute_network" "my_network" {
    name = "my_network"
}
```

The reference to **mynetwork** in the network argument creates an implicit (**known**) dependency.

View implicit (known) dependencies via `terraform apply`

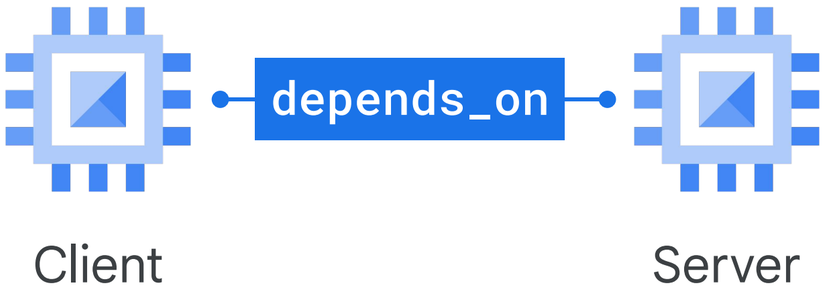
Terraform creates the network first.

After the implicit dependency is fulfilled (network is created), the compute instance is created.

```
$terraform apply

google_compute_network.mynetwork: Creating...
google_compute_network.mynetwork: Still creating... [10s elapsed]
google_compute_network.mynetwork: Still creating... [20s elapsed]
google_compute_network.mynetwork: Still creating... [30s elapsed]
google_compute_network.mynetwork: Creation complete after 32s
[id=projects/qwiklabs-gcp-01-e973d950dd4a/global/networks/mynetwork]


google_compute_instance.myinstance: Creating...
google_compute_instance.myinstance: Still creating... [10s elapsed]
google_compute_instance.myinstance: Creation complete after 13s
[id=projects/qwiklabs-gcp-01-e973d950dd4a/zones/us-central1-a/instances/myinstance]
```

Snippet of the `terraform apply` output

Explicit (unknown) dependencies are defined by using the depends_on argument



The client VM can only be created when the server VM is created.

```
resource "resource_type" "resource_name" {

..
depends_on = [<resource_type>.<resource_name>]
}
```

```
resource "google_compute_instance" "client" {
 ...
 depends_on = [google_compute_instance.server]
}

resource "google_compute_instance" "server" {

 #All required configuration options
}
```

View explicit (unknown) dependencies via `terraform apply`

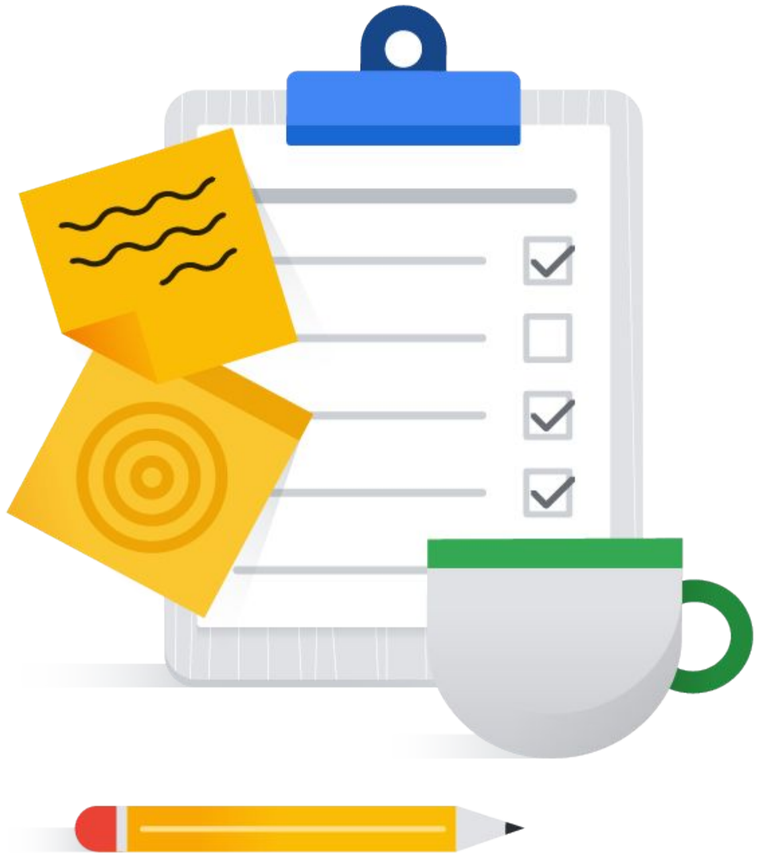
Server is created *before* client.

Due to explicit dependency, the client is created only *after* the server is created.

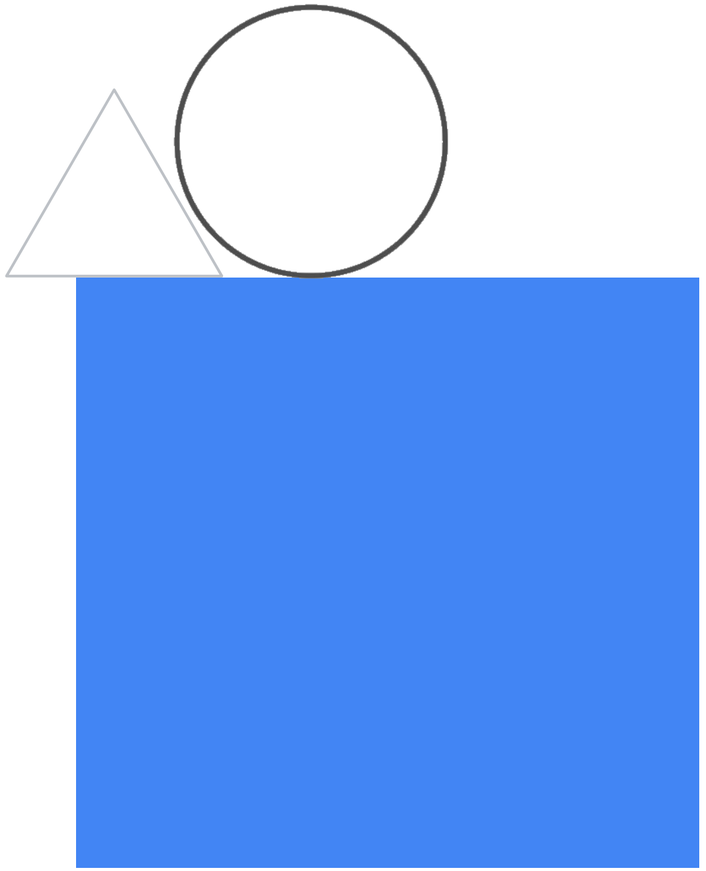
```
$terraform apply

google_compute_instance.server: Creating...
google_compute_instance.server: Still creating... [10s elapsed]
google_compute_instance.server: Creation complete after 12s
[id=projects/qwiklabs-gcp-01-e973d950dd4a/zones/us-central1-a/instances/server]

google_compute_instance.client: Creating...
google_compute_instance.client: Still creating... [10s elapsed]
google_compute_instance.client: Creation complete after 13s
[id=projects/qwiklabs-gcp-01-e973d950dd4a/zones/us-central1-a/instances/client]
```

Snippet of the **`terraform apply`** output

Topics

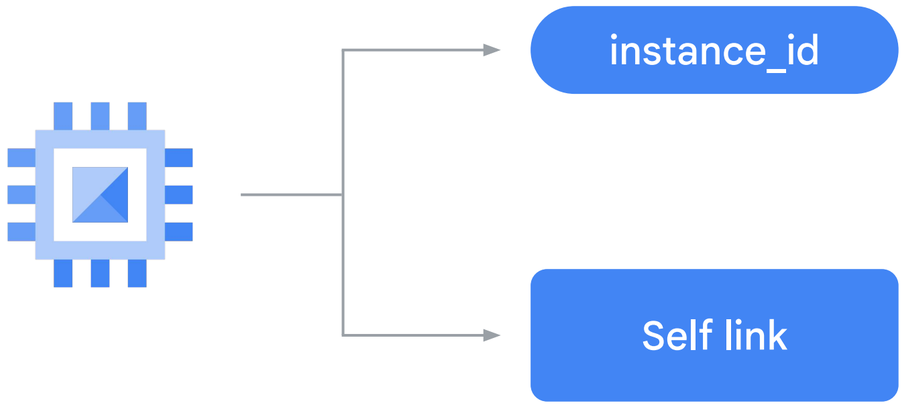


What are output values?

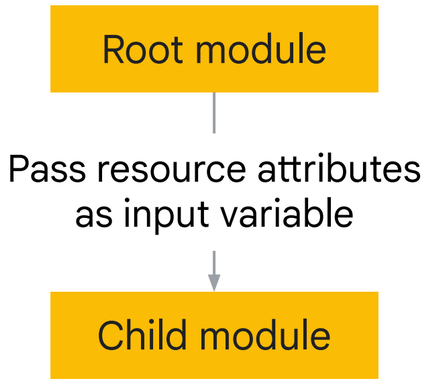


What are output values?

Output values expose information about the resource to the user of the Terraform configuration.



Print resource attributes.



Pass resource attributes.

Print resource attributes by using output values

```
-- server/
   -- main.tf
   -- outputs.tf
   -- variables.tf
```

M

O

M

```
resource "google_storage_bucket_object" "picture" {
…
}
```

O

```
output "picture_URL" {
  description = “URL of the picture uploaded
//value       = <resource_type>.<resource_name>.<attribute>
  value       = google_storage_bucket_object.picture.self_link
}
```

Note: We recommend that you use output values, instead of user-supplied inputs, for computed attributes of a resource.

```
Google_storage_bucket_object.picture: Creating...
Google_storage_bucket_object.picture: Creating complete after 1s
[]

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

Outputs:

picture_URL = “https://storage.googleapis.com/my-gallery/..
```

terraform output

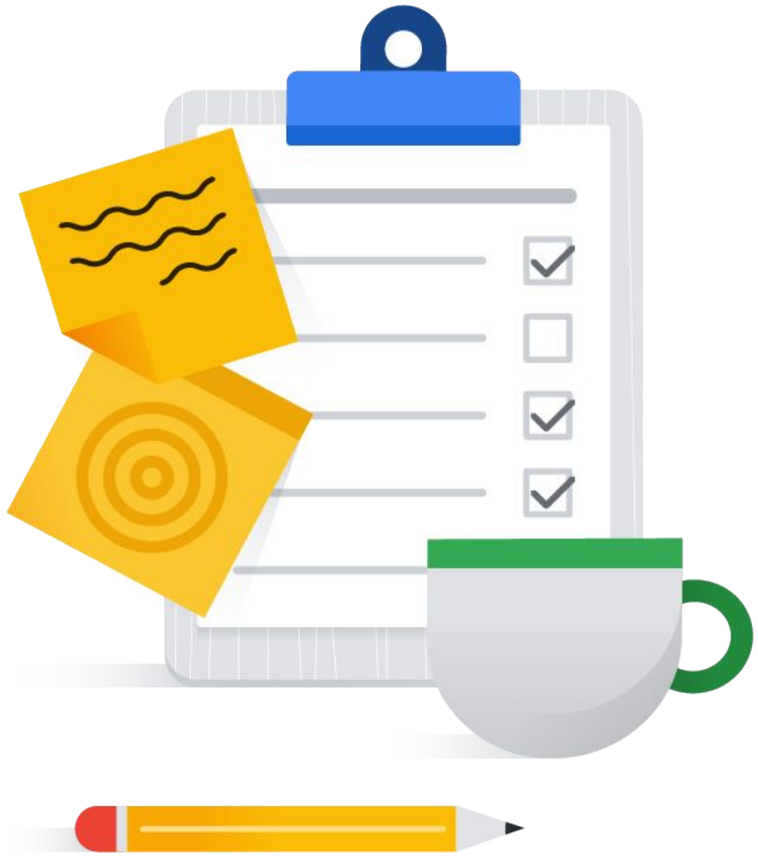
```
-- server/
   -- main.tf
   -- outputs.tf
   -- variables.tf
```

```
resource "google_compute_network" "vpc_network" {
project= "<PROJECT_ID>"
name = "vpc-network"
}
output network_id {
     value = google_compute_network.vpc_network.id
}
output network_link {
     value = google_compute_network.vpc_network.self_link
}
```

Query all output values.

```
$terraform output
network_id = "projects/<project-id>/global/networks/vpc-network"
network_link = "https://www.googleapis.com/../projects/<project-id>/../vpc-network"
```

Topics



Outputs Best Practices

Output useful information

```
-- server/
   -- main.tf
   -- outputs.tf
   -- variables.tf
```

```
resource "google_compute_network" "vpc_network" {
    project= "<PROJECT_ID>"
    name = "vpc-network"
}
output network_id {
     value = google_compute_network.vpc_network.id
}
output network_name {
     value = google_compute_network.vpc_network.name
}
```

```
Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
Outputs:
network_id = "projects/<project-id>/global/networks/vpc-network"
network_name = "vpc-network"
```

Provide meaningful descriptions

```
-- server/
   -- main.tf
   -- outputs.tf
   -- variables.tf
```

Provide meaningful descriptions for all outputs.

```
output "dev_server_URI" {
  description = "URI of the Dev instance"
  value       = google_compute_instance.dev_main.name
}
```

```
output "link" {
  description = "Mylink"
  value       = google_compute_instance.dev_main.name
}
```

Organize all outputs in an `outputs.tf` file

```
-- server/
   -- main.tf
   -- outputs.tf
   -- variables.tf
```

```
output "bucketname" {
  value = google_storage_bucket.mybucket.name
}
output "bucketlocation" {
  value = google_storage_bucket.mybucket.location
}
```

```
resource "google_storage_bucket" "mybucket" {
  name          = "student102345688493"
  location       = “US”
}
output "bucketname" {
  value = google_storage_bucket.mybucket.name
}
output "bucketlocation" {
  value = google_storage_bucket.mybucket.location
}
```

Mark sensitive outputs

```
-- server/
   -- main.tf
   -- outputs.tf
   -- variables.tf
```

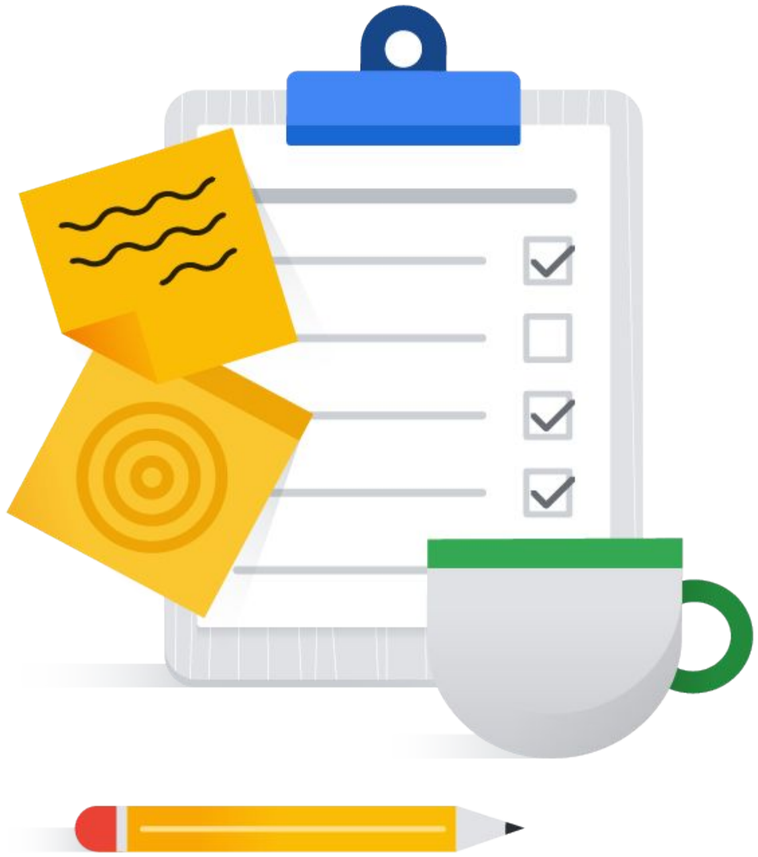
```
output "sql_user_password" {
 value       = google_sql_user.users.password
 description = "The password for sql user."
 sensitive   = true
}
```

Sensitive values are masked with the keyword “sensitive” in the output of `terraform plan` or `terraform apply`

```
$terraform plan
Terraform will perform the following actions:

  # some_resource.a will be created
  + resource "google_sql_user" "users" {
      + password    = (sensitive)
    }

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

Topics

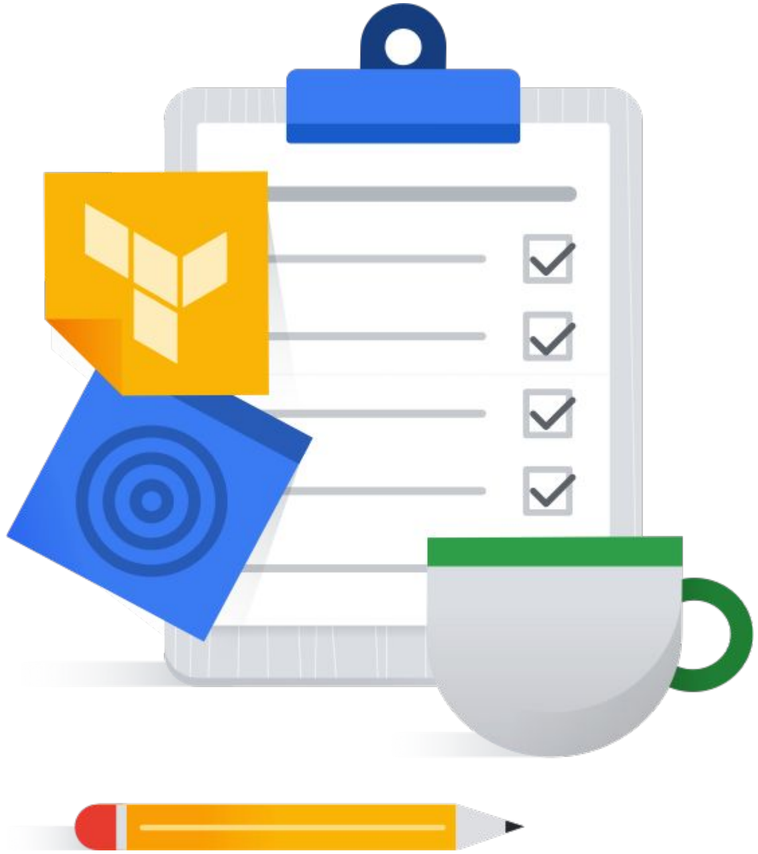
09 Terraform Registry and Cloud Foundation Toolkit



The Terraform Registry

- Interactive resource for discovering providers and modules.
- Solutions developed by HashiCorp, third-party vendors, and the Terraform community.

registry.terraform.io/browse/modules?provider=google



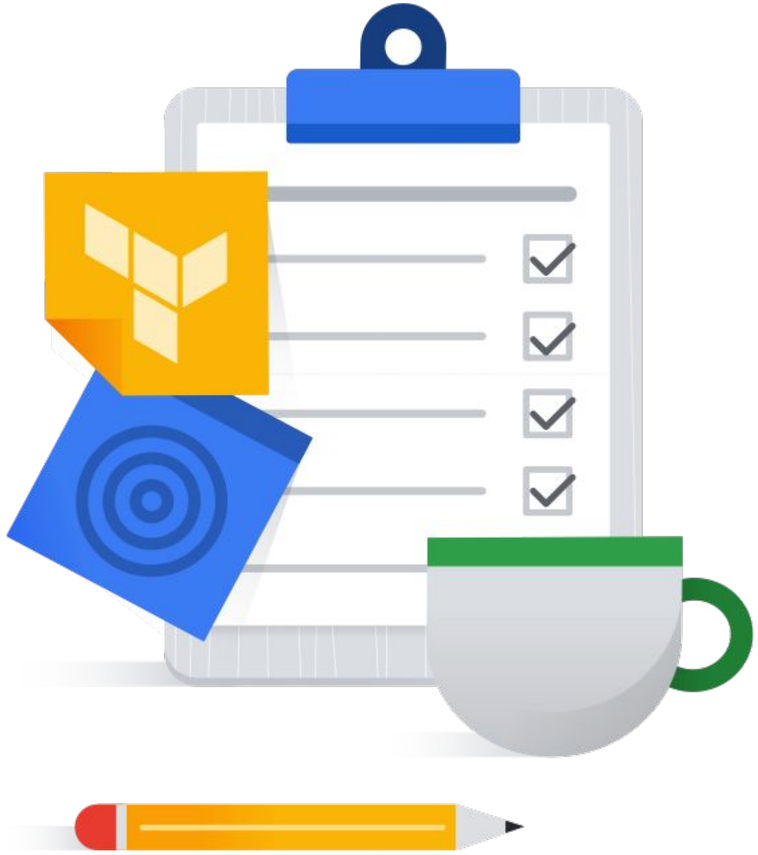
Cloud Foundation Toolkit (CFT)

- CFT provides a series of reference modules for Terraform that reflect Google Cloud best practices.
- CFT modules can be used without modification to quickly build a repeatable enterprise-ready foundation in Google Cloud.
- CFT modules are also referred to as Terraform blueprints.

cloud.google.com/foundation-toolkit

cloud.google.com/docs/terraform/blueprints/terraform-blueprints

You can also use **Cloud Foundation Fabric (CFF),** a collection of modules and examples for fast prototyping or to be modified and used in organizations.



CFT module versus standard Terraform

CFT projects_iam

```
module "project-iam-bindings" {
    source   =
    "terraform-google-modules/iam/google//modules/projects_iam"
    projects = ["my-project-one", "my-project-two"]
    mode     = "additive"

  bindings = {
    "roles/compute.networkAdmin" = [
      "group:my-group@my-org.com",
      "user:my-user@my-org.com",
    ]
    "roles/appengine.appAdmin" = [
      "group:my-group@my-org.com",
      "user:my-user@my-org.com",
    ]
  }
}
```

CFT module allows you to maintain the IAM roles for multiple projects within the same module.

Terraform google_project_iam_binding

```
resource "google_project_iam_member" "project1-net-grp" {
  project = "my-project-one"
  role    = "roles/compute.networkAdmin"
  member  = "group:my-group@my-org.com"
}

resource "google_project_iam_member" "project1-net-user" {...}

resource "google_project_iam_member" "project1-net-grp" {...}

resource "google_project_iam_member" "project2-net-user" {...}

resource "google_project_iam_member" "project2-net-grp" {...}

resource "google_project_iam_member" "project1-app-user" {...}

resource "google_project_iam_member" "project1-app-grp"{...}

resource "google_project_iam_member" "project2-app-user" {...}

resource "google_project_iam_member" "project2-app-grp"{...}
```

Infrastructure Manager

- Managed service
  - Terraform configuration defines the infrastructure
  - Infra Manager deploys onto Google Cloud
- Doesn’t manage the deployment of applications onto resources
  - Use products like Cloud Build, Cloud Deploy, or third-party apps

Lab

Creating Resource Dependencies

Quiz

Quiz | Question 1

Question

What is the most common use case for output values in Terraform?

A. Declare a resource within a Terraform configuration.

B. Parameterize a resource configuration.

C. Print resource attributes of a root module CLI after its deployment.

Quiz | Question 1

Answer

What is the most common use case for output values in Terraform?

A. Declare a resource within a Terraform configuration.

B. Parameterize a resource configuration.

C. Print resource attributes of a root module CLI after its deployment.

Quiz | Question 2

Question

Can a variable be assigned values in multiple ways?

A. Yes

B. No

Quiz | Question 2

Answer

A variable can be assigned values in multiple ways.

A. True

B. False

Quiz | Question 3

Question

Which dependency can be automatically detected by Terraform?

A. Implicit dependency

B. Explicit dependency

Quiz | Question 3

Answer

Which dependency can be automatically detected by Terraform?

A. Implicit dependency

B. Explicit dependency

Quiz | Question 4

Question

How many resource types can be represented in a single resource block?

A. Four

B. Three

C. Two

D. One

Quiz | Question 4

Answer

How many resource types can be represented in a single resource block?

A. Four

B. Three

C. Two

D. One

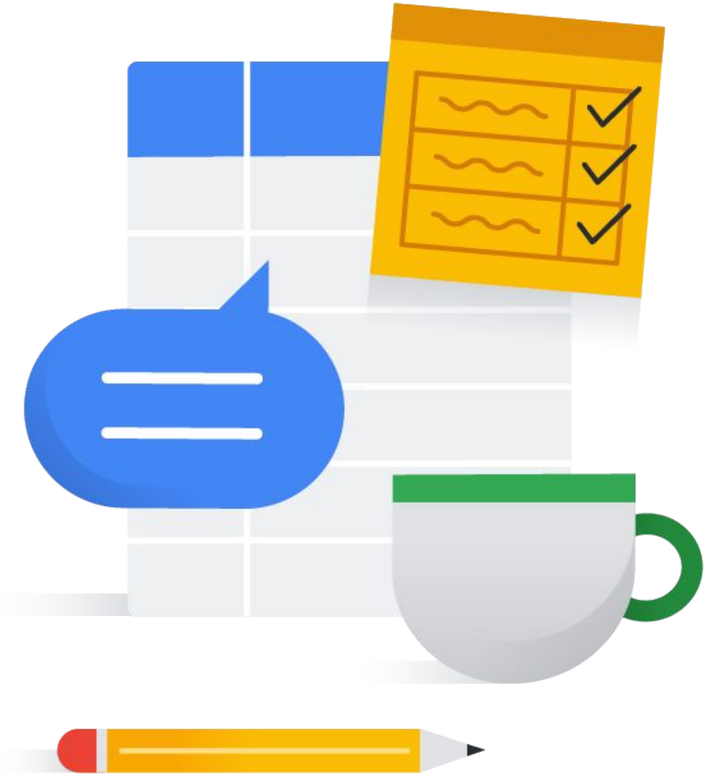
Module review

01 Declare the resources within Terraform.

02 Explain implicit and explicit resource dependencies.

03 Use variables and output values within the root configuration.

04 Explain the Terraform Registry and the Cloud Foundation Toolkit.



Google Cloud