Lab4
Variables and Workspaces

Contents

Lab Objectives 1
Teaching Points 1
Before you begin 2
Solution 2
Task 1 - Review provisioned terraform files 2
Task 2 - Run terraform plan and apply with hardcoded parameter values 3
Task 3 - Substitute hardcoded parameter values with variables 4
Task 4 - Overriding variable values at the command prompt 6
Task 5 - Override variable values using terraform.tfvars 6
Task 6 - Implement Terraform Workspaces 8
Task 7 - Lab Clean-up 11

Lab Objectives

In this lab you will:

- Deploy resources using a hard-coded terraform file
- Utilize variables to make your code reuseable
- Implement overrides using the command line and tfvars files
- Utilize terraform workspaces to allow multiple simultaneous deployments

Teaching Points

When writing terraform code, it is instinctive to supply explicit values when they are required, for example **name = "michael"**. This can make the code very static though, requiring manual changes of these values each time a modification is needed. A better practice is to use 'placeholders', variables, to which values can be passed at runtime, for example **name = var.username**. Passing values to these

variables can be achieved using variable file default values, tfvars files, command line supplied or by being prompted. By default, all changes made to your configuration will be tracked by a single state file. Workspaces allow multiple state files to exist simultaneously so that the same base code, provided with unique variable values, can be deployed. This makes your code flexible and reuseable.

Before you begin

1. Ensure you have completed Lab0 before attempting this lab.
2. In the IDE terminal pane, enter the following commands…

   cd ~/googlelabs/lab04

3. This shifts your current working directory to googlelabs/lab04. Ensure all commands are executed in this directory
4. Close any open files and use the Explorer pane to navigate to and open the pre-configured main.tf file in labs04.

Solution

There is no solution code for this lab as it involves multiple deployment phases. Reach out to your instructor if you encounter issues.

Task 1- Review provisioned terraform files

1. Review the provisioned files in lab04..

   main.tf Hard coded deployment of an **e2-small** GCE instance running **debian-11** in **us-central1-a**. This represents the type of resource we will create in this lab, in production there would be many other resources defined here. There is also an output block (lines 32-41) which displays information about the deployment, including the workspace.

   variables.tf All content currently commented out

   terraform.tfvars All content currently commented out

   dev.tfvars Variable values that will be used in a new workspace

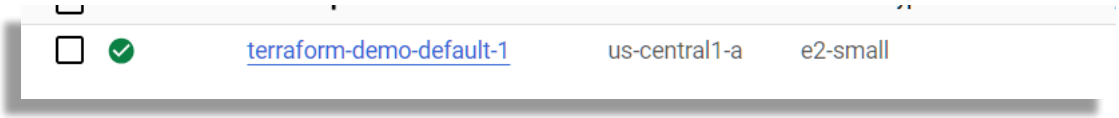
   prod.tfvars Variable values that will be used in a new workspace

Task 2- Run terraform plan and apply with hardcoded parameter values

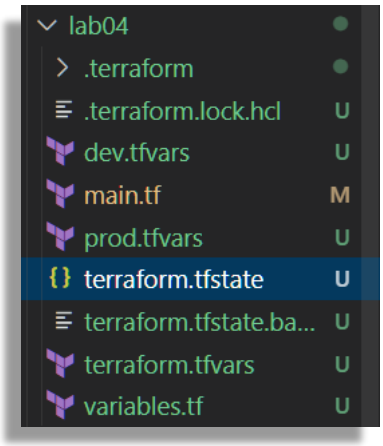
1. Ensure you have navigated to the **googlelabs\lab04** folder
2. Update line 11 of main.tf with your project id
3. Run **terraform init**
4. Run **terraform plan**
5. Review the plan output..

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

Changes to Outputs:
  + instance_details = {
      + name_of_vm     = "terraform-demo-default-1"
      + size_of_vm     = "e2-small"
      + workspace_used = "default"
      + zone_of_vm     = "us-central1-a"
    }
```

6. Run **terraform apply** followed by **yes**
7. Switch to the Console and verify the creation of the GCE instance in us-central1-a..



8. Note the creation of a state file in the root folder..



9. **Takeaway**: Hardcoded parameter values will always be used if they exist. This can make the code inflexible and static

Task 3- Substitute hardcoded parameter values with variables

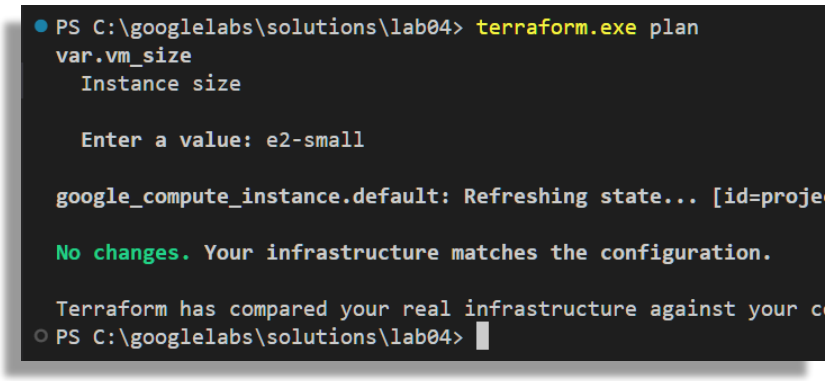
1. In **main.tf**; Replace hard-coded zone "us-central1-a" with **var.vm_location** (no quotes)
2. Replace hard image "debian-cloud/debian-11" with **var.vm_image**
3. Replace hard-coded instance machine_type "e2_small" with **var.vm_size**
4. Save the changes.
5. Run **terraform plan** and review the 3 errors…

```
Error: Reference to undeclared input variable

  on main.tf line 22, in resource "google_compute_instance" "default":
  22:       image = var.vm_image

An input variable with the name "vm_image" has not been declared. This variable can be
```

6. **Takeaway**: If variables are referenced in your code, then they **must** be declared.
7. In **variables.tf**; uncomment all lines **except** line 10 which provides default values for the **vm_size** variable. …

```
1   variable "vm_location" {
2     description = "Google Cloud zone"
3     type        = string
4     default     = "us-central1-a"
5   }
6
7   variable "vm_size" {
8     description = "Instance size"
9     type        = string
10  #  default     = "e2-small"
11  }
12
13  variable "vm_image" {
14    description = "Instance image"
15    type        = string
16    default     = "debian-cloud/debian-11"
17  }
```

8. Switch to main.tf, uncomment line 39
9. Run **terraform plan**
10. When prompted, enter **e2-small** as the instance size..



11. The planning completes using the variable default values if they exist and prompted for when there is no default value. Given that these values are the same as the old static values, the planning phase shows that there are no changes needed
12. In **variables.tf**; uncomment line 10 and save the file
13. Run **terraform plan**
14. The planning completes using all variable values drawn from the variables file. Given that these values are the same as the old static values, the planning phase shows that there are no changes needed.
15. Destroy your deployment using **terraform destroy** followed by **yes**
16. Switch to the console and confirm the deletion of the instance in us-central1-a
17. **Takeaway**: If a variable is declared but no value has been assigned, then you are prompted for a value. If a variable is declared with a default value then this value will be used unless overridden.

Task 4- Overriding variable values at the command prompt

1. Enter the following command..
   terraform plan -var="vm_size=e2.micro"

```
PS C:\googlelabs\solutions\lab04> terraform plan -var="vm_size=e2.micro"
```

```
Changes to Outputs:
  + instance_details = {
      + image_of_vm    = "debian-cloud/debian-11"
      + name_of_vm     = "terraform-demo-default-1"
      + size_of_vm     = "e2.micro"
      + workspace_used = "default"
      + zone_of_vm     = "us-central1-a"
    }
```

2. **Takeaway**: Supplying variable values at the command prompt using **-var=" "** has the highest priority and overrides values supplied ***anywhere*** else. In this case **e2.micro** is used as the machine_type value. The zone and image values are drawn from the variables file default values. Do not apply this deployment.

Task 5- Override variable values using terraform.tfvars

1. Uncomment line 1 and 2 in **terraform.tfvars**. This file now supplies values for 2 variables, values that conflict with those in variables.tf...

```
1   vm_location = "us-central1-b"
2   vm_size     = "e2-micro"
3   #vm_image = "ubuntu-os-cloud/ubuntu-2004-lts"
```

2. Run **terraform plan**

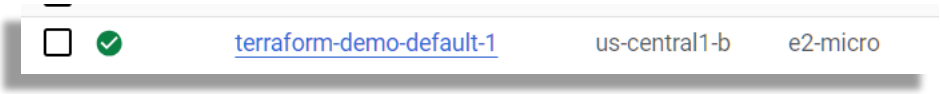
```
Changes to Outputs:
  + instance_details = {
      + image_of_vm    = "debian-cloud/debian-11"
      + name_of_vm     = "terraform-demo-default-1"
      + size_of_vm     = "e2-micro"
      + workspace_used = "default"
      + zone_of_vm     = "us-central1-b"
    }
```

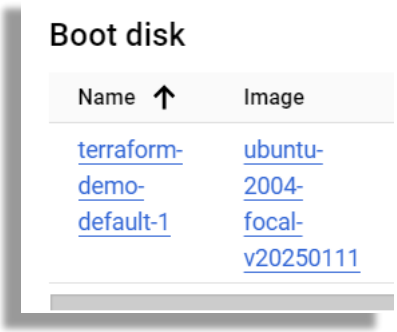
Notice that the size value **e2.micro** and the location value **us-central1-b** are drawn from **terraform.tfvars**, overriding the default size in the variables file. The image value is still drawn from variables.tf

3. Uncomment line 3 in **terraform.tfvars**. This file now supplies values for all variables, values that conflict with, and therefore override, those in variables.tf.
4. Run **terraform plan**

```
Changes to Outputs:
  + instance_details = {
      + image_of_vm    = "ubuntu-os-cloud/ubuntu-2004-lts"
      + name_of_vm     = "terraform-demo-default-1"
      + size_of_vm     = "e2-micro"
      + workspace_used = "default"
      + zone_of_vm     = "us-central1-b"
    }
```

5. Run **terraform apply** followed by **yes**
6. Switch to the console and verify the creation of the t3.micro instance in us-east-1 (N. Virginia)…





7. **Takeaway**: If terraform.tfvars exists and supplies **all** variable values, there is no rationale for specifying default values in the variables file as these will always be overridden. It is not uncommon therefore to define the variables using a variables file without declaring any default values.

Task 6- Implement Terraform Workspaces

1. Changing deployment parameters will affect the current deployment. Workspaces allow multiple deployments, using the same base code but with different parameters, to exist simultaneously, each with its own state file. If no new workspaces are created then your deployment is in the **default** workspace which always exists and cannot be deleted.

2. Create two workspaces; "**development**" and "**production**"…
   terraform workspace new development
   terraform workspace new production
   terraform workspace list

```
PS C:\googlelabs\solutions\lab04> terraform workspace list
  default
  development
* production
```

3. The * indicates your current workspace is now **production**. The **default** workspace always exists, and this is where your current deployment of an **e2-micro** GCE instance, running **ubuntu** in **us-central1-b** is tracked by the state file **terraform.tfstate** in the root folder.

4. Note the creation of a new folder "**terraform.tfstate.d**" with an empty subfolder for each of the new workspaces

5. Uncomment all lines in **prod.tfvars** and **dev.tfvars**

6. Run **terraform plan**

```
Changes to Outputs:
  + instance_details = {
      + name_of_vm      = "terraform-demo-production-1"
      + size_of_vm      = "e2-micro"
      + workspace_used = "production"
      + zone_of_vm      = "us-central1-b"
    }
```

7. The name given to the vm has been crafted to reflect the namespace, but all other variable values are still drawn from **terraform.tfvars**

8. Run **terraform plan --var-file=prod.tfvars**

```
googlelabs > solutions > lab04 > prod.tfvars > ...
1   vm_location = "europe-west2-b"
2   vm_size     = "n2-standard-4"
3

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  AZURE

Changes to Outputs:
  + instance_details = {
      + image_of_vm     = "ubuntu-os-cloud/ubuntu-2004-lts"
      + name_of_vm      = "terraform-demo-production-1"
      + size_of_vm      = "n2-standard-4"
      + workspace_used = "production"
      + zone_of_vm      = "europe-west2-b"
    }
```

9. t**fvar** files other than terraform.tfvars and <name>.auto.tfvars are not referenced unless specified at the command line. Here we have specified inclusion of **prod.tfvars** which contains location value **“europe-west2-b”** and size value **“n2-standard-4”** The image values is not defined in **prod.tfvars** and is therefore drawn from **terraform.tfvars**.

10. Notice that the plan will not destroy any resources. Recall that we currently have an GCE instance deployed in us-central1-b. This is in the default workspace and will not be affected as we are now in the production workspace.

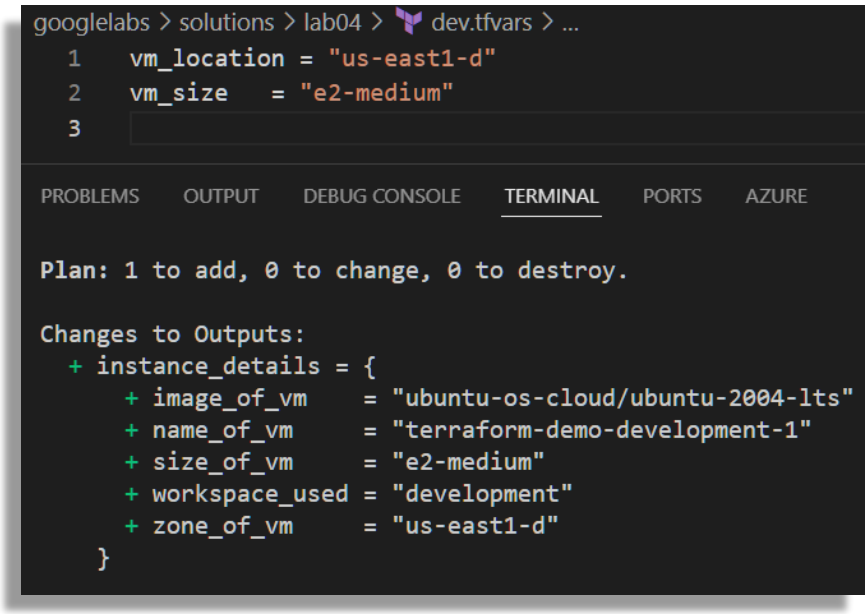
11. Run **terraform apply --var-file=prod.tfvars** followed by **yes**

12. Switch to the console and verify the creation of the **n2-standard-4** instance in **europe-west2-b**. Note that the **e2-micro** instance created in the default workspace still exists.

13. In the IDE, expand the terraform.tfstate.d folder and verify the creation of a new state file for the production workspace..

14. Move to the development work space; **terraform workspace select development**

15. Run **terraform plan --var-file=dev.tfvars**



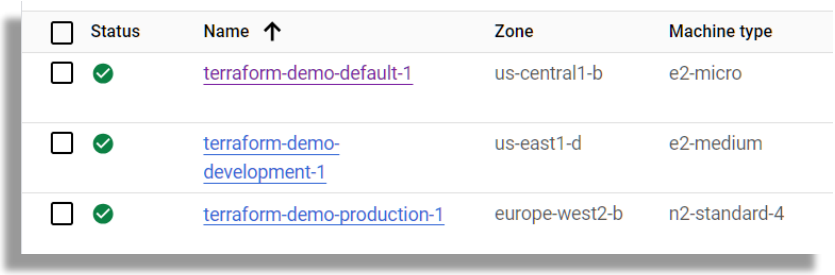
16. Here we have specified inclusion of **dev.tfvars** which contains location value **us-east1-d** and size value **e2-medium**. If there are conflicts, values from **dev.tfvars** override default variable values and values in the **terraform.tfvars** file. The plan shows that the location and size values are therefore drawn from

the **prod.tfvars** file. The image values is not defined in **prod.tfvars** and is therefore drawn from **terraform.tfvars**

17. Again, notice that the plan will not destroy any resources. Recall that we now have GCE instances deployed in us-central1-b and Europe-west2-b. These are in the default and production workspaces respectively and will not be affected as we are now in the **development** workspace.

18. Run **terraform apply --var-file=dev.tfvars** followed by **yes**

19. Switch to the console and verify the creation of the e2-medium development instances in us-east1-d alongside the default and production instances.



Status	Name	Zone	Machine type
✓	terraform-demo-default-1	us-central1-b	e2-micro
✓	terraform-demo-development-1	us-east1-d	e2-medium
✓	terraform-demo-production-1	europe-west2-b	n2-standard-4

20. In the IDE, expand the terraform.tfstate.d folder and verify the creation of a new state file for the development workspace..

Task 7- Lab Clean-up

1. When deleting resources in workspaces, always pay close attention to the workspace you are currently working in. Use **terraform workspace show** to determine your current workspace..

```
PS C:\googlelabs\solutions\lab04> terraform workspace show
development
```

2. Destroy the resources in the current workspace (dev) using..
   terraform destroy --var-file=dev.tfvars
   Note that the tfvars file **must** be specified when performing a destroy action id specified during an apply action.

3. Enter **yes** when prompted

4. Switch to the Console to confirm the deletion of the development instance in us-east1-d

5. The current workspace cannot be deleted. Move to the **production** workspace and delete the **development** workspace…

   terraform workspace select production
   terraform workspace delete development

6. Destroy the resources in the current workspace (production) using..
   terraform destroy --var-file=prod.tfvars
   Note that the tfvars file **must** be specified when performing a destroy action id specified during an apply action.

7. Enter **yes** when prompted

8. Switch to the console to confirm the deletion of the production instance in Europe-west2-b

9. The current workspace cannot be deleted. Move to the **default** workspace and delete the **production** workspace…

   terraform workspace select default
   terraform workspace delete production

10. Verify the deletion of the workspaces, leaving just the default workspace. This cannot be deleted..

terraform workspace list

```
PS C:\googlelabs\solutions\lab04> terraform workspace list
* default
```

11. Note that the workspace directories are deleted along with the workspaces themselves…

12. Destroy the resources in the current workspace (default) using **terraform destroy** followed by **yes**

13. Enter **yes** when prompted

14. Switch to the console to confirm the deletion of the default instance in us-central1-b

Congratulations, you have completed this lab