

## AGENDA – Ternary operator

- 1) SEARCH – Terraform expressions
- 2) Create “azurerm\_resourcegroup\_generic” folder and create main.tf file into it.

```
main.tf ...\azurerm_bastion
main.tf ...\azurerm_resourcegroup_generic X

Modules > azurerm_resourcegroup_generic > main.tf > ...
1  variable "dhondhu" {
2      default = "peelu"
3  }
4
5  resource "azurerm_resource_group" "rg" {
6      name = var.dhondhu
7      location = "westus"
8  }
9
10
11
12
13
14
15
16
```

- 3) Now suppose, we have requirement to bring name in output after terraform plan as
  - i) rg-peelu
  - ii) rg-ramu

4) So for doing above we have to do string interpolation

```
main.tf ...\azure_terraform_bastion | main.tf ...\azure_terraform_resource_group_generic X
Modules > azure_terraform_resource_group_generic > main.tf > ...
1  variable "dhondhu" {
2      default = "tony"
3  }
4
5  resource "azurerm_resource_group" "rg" {
6      name = "rg-${var.dhondhu}"
7      location = "westus"
8  }
9
```

5) Now for numbers

```
main.tf ...\azure_terraform_resource_group_generic X
> resource "azurerm_resource_group" "rg" > [?] name
variable "dhondhu" {
2      default = "tony"
3  }
4  resource "azurerm_resource_group" "rg" {
5      name      = 1 + 1
6      location = "westus"
7  }
8
PORTS  AZURE  DEBUG CONSOLE  PROBLEMS  OUTPUT  TERMINAL
+ id      = (known after apply)
+ location = "westus"
+ name    = "2"
}

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

Note: You didn't use the -out option to save this plan, so Terraform can't
```

```
main.tf X
> resource "azurerm_resource_group" "rg" > [?] name
variable "dhondhu" {
  default = "only"
}
resource "azurerm_resource_group" "rg" {
  name      = 1 * 10
  location  = "westus"
}

PORTS  AZURE  DEBUG CONSOLE  PROBLEMS  OUTPUT  TERMINAL

+ id      = (known after apply)
+ location = "westus"
+ name    = "10" I
}

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

Note: You didn't use the -out option to save this plan, so Terraform

```
4 resource "azurerm_resource_group" "rg" {
5   name      = 79 % 10 I
6   location  = "westus"
7 }
8

PORTS  AZURE  DEBUG CONSOLE  PROBLEMS  OUTPUT  TERMINAL

+ id      = (known after apply)
+ location = "westus"
+ name    = "9"
}

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

6) Equality operators

# Equality Operators

The equality operators both take two values of any type and produce boolean values as results.

- `a == b` returns `true` if `a` and `b` both have the same type and the same value, or `false` otherwise.
- `a != b` is the opposite of `a == b`.

Because the equality operators require both arguments to be of exactly the same type to decide equality, we recommend using these operators only with values of primitive types. For non-primitive types, use `Equals` or `EqualsIgnoreCase` methods, or use explicit type conversion functions to indicate which type you are intending to compare.

```
main.tf
> resource "azurerm_resource_group" "rg" > [?] name
variable "dhondhu" {
  default = "ony"
}
4 resource "azurerm_resource_group" "rg" {
5   name      = 79 == 79
6   location  = "westus"
7 }
8

PORTS  AZURE  DEBUG CONSOLE  PROBLEMS  OUTPUT  TERMINAL

+ id      = (known after apply)
+ location = "westus"
+ name    = true
}

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

```
main.tf X
> resource "azurerm_resource_group" "rg" > [?] name
variable "dhondhu" {
  default = "only"
}
4 resource "azurerm_resource_group" "rg" {
5   name      = 79 == 80
6   location = "westus"
7 }
8

PORTS  AZURE  DEBUG CONSOLE  PROBLEMS  OUTPUT  TERMINAL

+ id      = (known after apply)
+ location = "westus"
+ name    = "false"
}

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

7) Now terraform has ability to compare strings also as well

```
> resource "azurerm_resource_group" "rg" > [?] name
variable "dhondhu" {
  default = "only"
}
4 resource "azurerm_resource_group" "rg" {
5   name      = "dhondhu" == "dhondhu"
6   location = "westus"
7 }
8

PORTS  AZURE  DEBUG CONSOLE  PROBLEMS  OUTPUT  TERMINAL

+ id      = (known after apply)
+ location = "westus"
+ name    = "true"
}

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

```
main.tf X
> resource "azurerm_resource_group" "rg" > location
variable "dhondhu" {
  default = "tony"
}
resource "azurerm_resource_group" "rg" {
  name      = "dhondhu" == "tondu"
  location  = "westus"
}
```

PORTS AZURE DEBUG CONSOLE PROBLEMS OUTPUT TERMINAL

```
+ id      = (known after apply)
+ location = "westus"
+ name    = "false"
}
```

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

8) "rg-\${79==79}-\${var.dhondhu}"

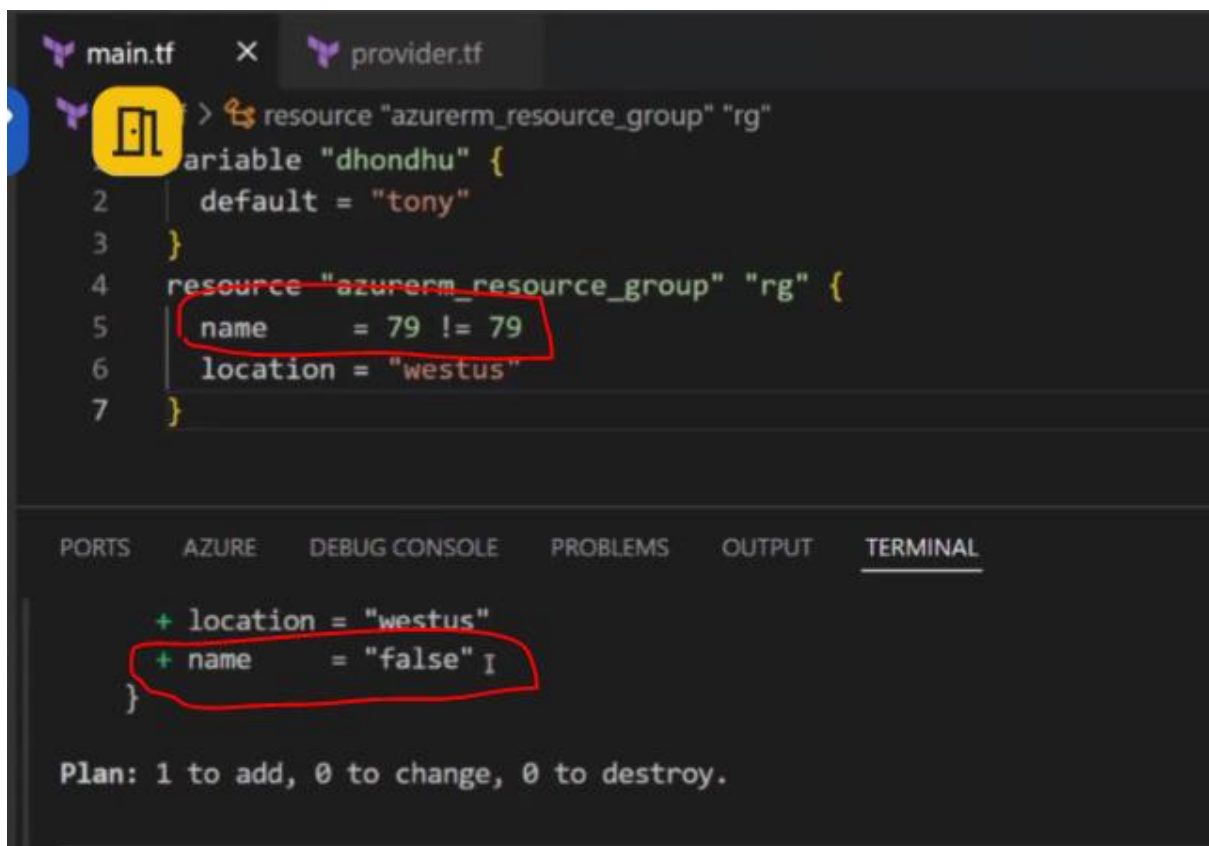
```
main.tf X provider New DevOps Engineer Jobs Jobs for D
> resource "azurerm_resource_group" "rg" > name
variable "dhondhu" {
  default = "tony"
}
resource "azurerm_resource_group" "rg" {
  name      = "rg-${79==79}-${var.dhondhu}"
  location  = "westus"
}
```

PORTS AZURE DEBUG CONSOLE PROBLEMS OUTPUT TERMINAL

```
# azurerm_resource_group.rg will be created
+ resource "azurerm_resource_group" "rg" {
  + id      = (known after apply)
  + location = "westus"
  + name    = "rg-true-tony"
}
```



9) 79 != 79



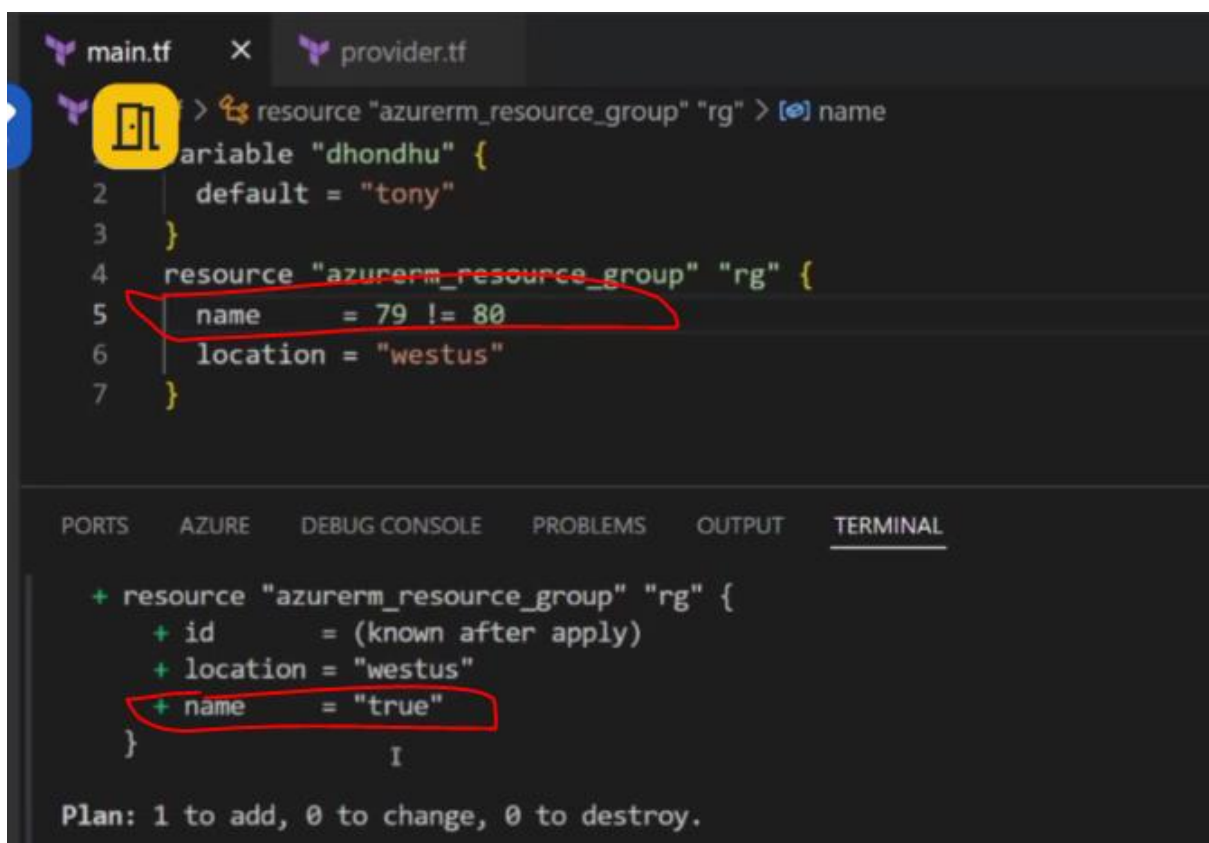
The screenshot shows a VS Code editor with two tabs: `main.tf` and `provider.tf`. The `main.tf` file contains the following Terraform code:

```
1 > resource "azurerm_resource_group" "rg"
2   variable "dhondhu" {
3     default = "tony"
4   }
5   resource "azurerm_resource_group" "rg" {
6     name      = 79 != 79
7     location  = "westus"
8   }
```

The code is partially obscured by a yellow icon. The `name` attribute in the resource block is circled in red. Below the code, the `TERMINAL` tab shows the output of a Terraform plan:

```
+ location = "westus"
+ name     = "false" I
```

The plan output is also circled in red. Below the plan, the text "Plan: 1 to add, 0 to change, 0 to destroy." is displayed.



The screenshot shows the same VS Code editor with the `main.tf` file. The `name` attribute in the resource block has been changed to `79 != 80`, which is circled in red:

```
1 > resource "azurerm_resource_group" "rg" > [?] name
2   variable "dhondhu" {
3     default = "tony"
4   }
5   resource "azurerm_resource_group" "rg" {
6     name      = 79 != 80
7     location  = "westus"
8   }
```

The `TERMINAL` tab shows the updated plan output:

```
+ resource "azurerm_resource_group" "rg" {
+   id      = (known after apply)
+   location = "westus"
+   name    = "true"
}
```

The new `name` attribute is circled in red. Below the plan, the text "Plan: 1 to add, 0 to change, 0 to destroy." is displayed.

10)

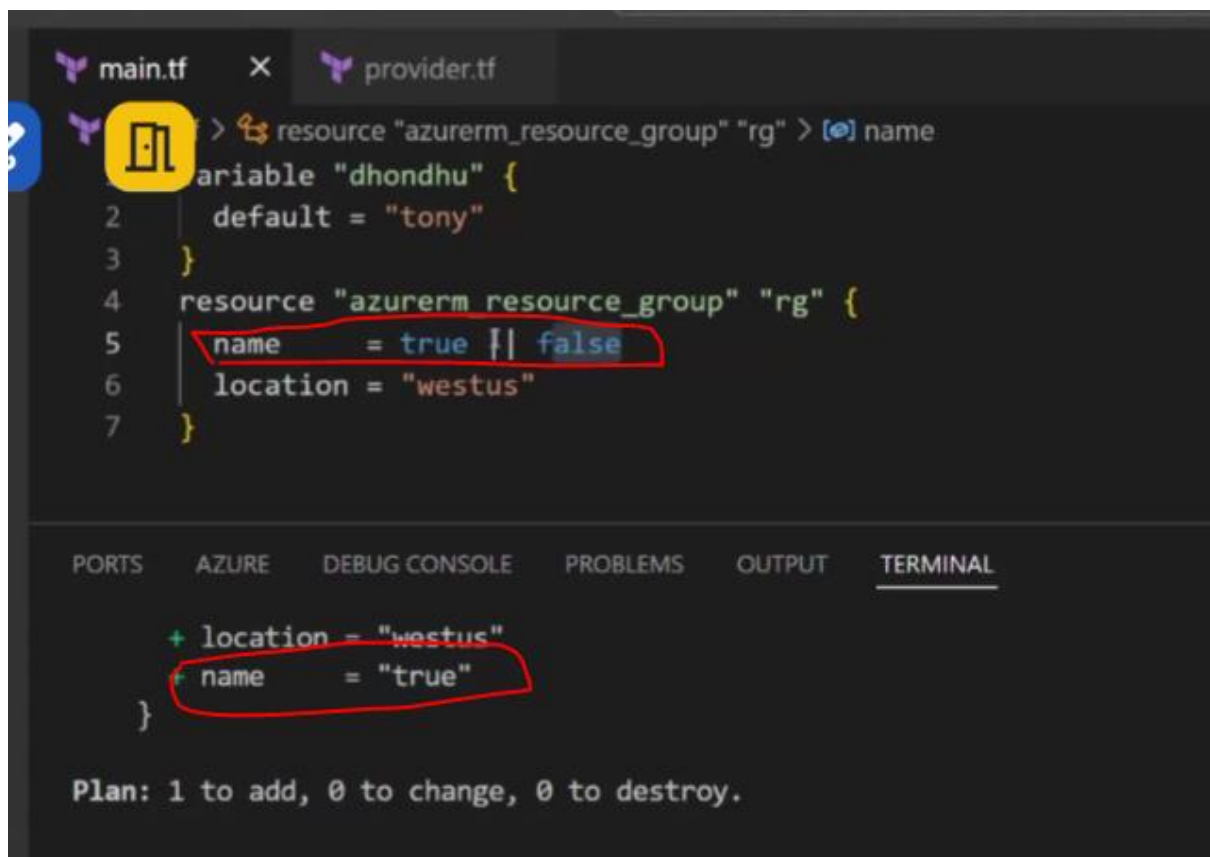
# Logical Operators

The logical operators all expect bool values and produce bool values as results.

- `a || b` returns `true` if either `a` or `b` is `true`, or `false` if both are `false`.
- `a && b` returns `true` if both `a` and `b` are `true`, or `false` if either one is `false`.
- `!a` returns `true` if `a` is `false`, and `false` if `a` is `true`.

Terraform does not have an operator for the "exclusive OR" operation. If you know

i) `true || false, false || false`



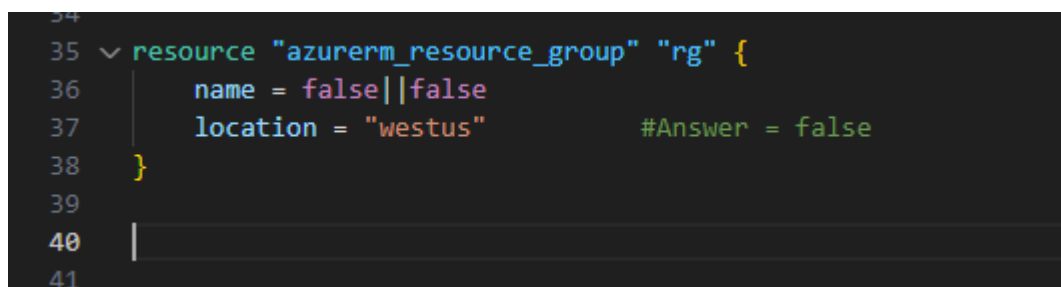
The screenshot shows a VS Code editor with two files: `main.tf` and `provider.tf`. The `main.tf` file contains the following Terraform code:

```
1 variable "dhondhu" {
2   default = "tony"
3 }
4 resource "azurerm_resource_group" "rg" {
5   name = true || false
6   location = "westus"
7 }
```

The line `name = true || false` is highlighted with a red box. Below the code, the terminal output shows the plan for the `azurerm_resource_group` resource:

```
+ location = "westus"
+ name     = "true"
}
```

The line `name = "true"` is highlighted with a red box. The terminal output also shows the plan summary: `Plan: 1 to add, 0 to change, 0 to destroy.`



The screenshot shows a VS Code editor with the following Terraform code:

```
35 resource "azurerm_resource_group" "rg" {
36   name = false || false
37   location = "westus"           #Answer = false
38 }
39
40
41
```

ii) `true && false, true && true`



```

39
40 resource "azurerm_resource_group" "rg" {
41     name = true&&false
42     location = "westus" #Answer = false
43 }
44
45 resource "azurerm_resource_group" "rg" {
46     name = true&&true
47     location = "westus" #Answer = true
48 }
49
50

```

iii) 79==79 && 78!=79

```

3
4 resource "azurerm_resource_group" "rg" {
5     name      = 79==79 && 78!=79
6     location = "westus"
7 }
8 true && true
9 true

```

## 12) Conditional Expressions

### Syntax

The syntax of a conditional expression is as follows:

```
condition ? true_val : false_val
```

If `condition` is `true` then the result is `true_val`. If `condition` is `false` then the result is `false_val`.

A common use of conditional expressions is to define defaults to replace invalid values:

```
var.a != "" ? var.a : "default-a"
```

If `var.a` is an empty string then the result is `"default-a"`, but otherwise it is the actual value of `var.a`.

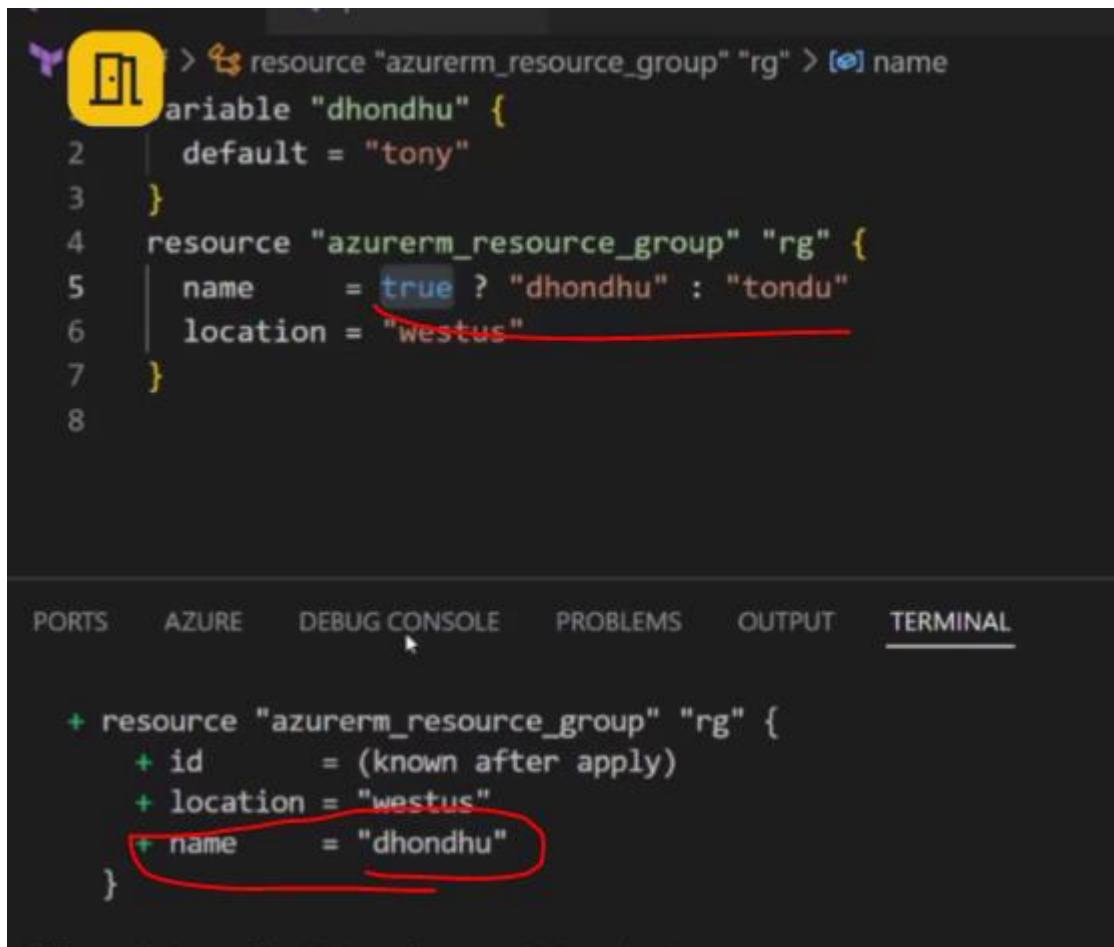
```

resource "azurerm_resource_group" "rg" {
    name = condition ? true_val : false_val
    location = "westus" #Answer = true
}

```

```
}
```

i) true ? "dhondhu" : "tondu"

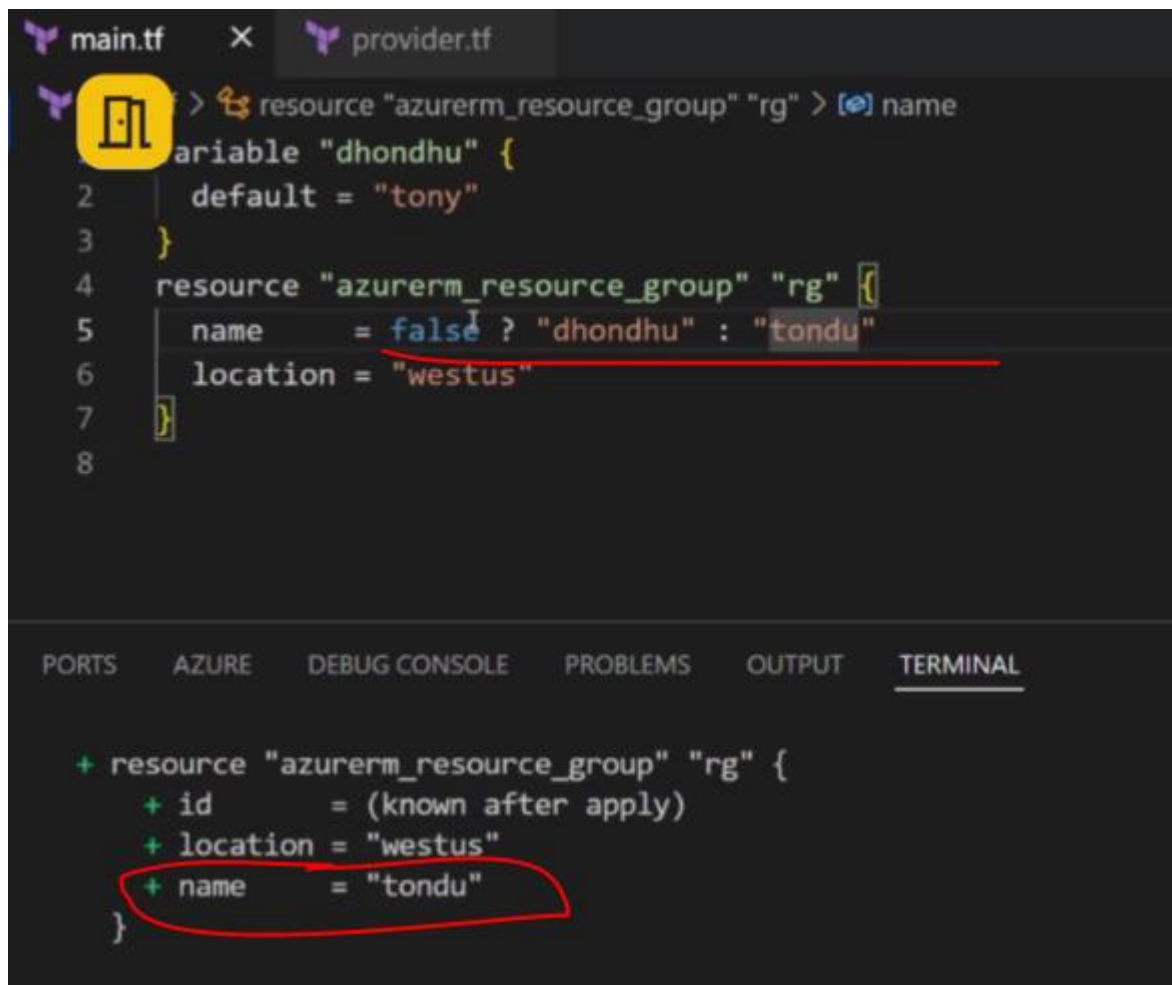


The screenshot shows a Visual Studio Code editor with an ARM template file. The template defines a variable 'dhondhu' with a default value of 'tony'. It then defines a resource 'azurerm\_resource\_group' 'rg' with a conditional 'name' property: 'true ? "dhondhu" : "tondu"'. The 'location' is set to 'westus'. The deployment output in the terminal shows the resource being created with 'name' set to 'dhondhu', which is circled in red. The 'location' is also 'westus'.

```
> resource "azurerm_resource_group" "rg" > [?] name
variable "dhondhu" {
2   | default = "tony"
3   | }
4   resource "azurerm_resource_group" "rg" {
5   |   name      = true ? "dhondhu" : "tondu"
6   |   location = "westus"
7   | }
8
```

```
+ resource "azurerm_resource_group" "rg" {
+   id          = (known after apply)
+   location    = "westus"
+   name        = "dhondhu"
+ }
```

ii) false ? "dhondhu" : "tondu"



The image shows a VS Code editor with two tabs: `main.tf` and `provider.tf`. The `main.tf` tab is active, displaying the following Terraform code:

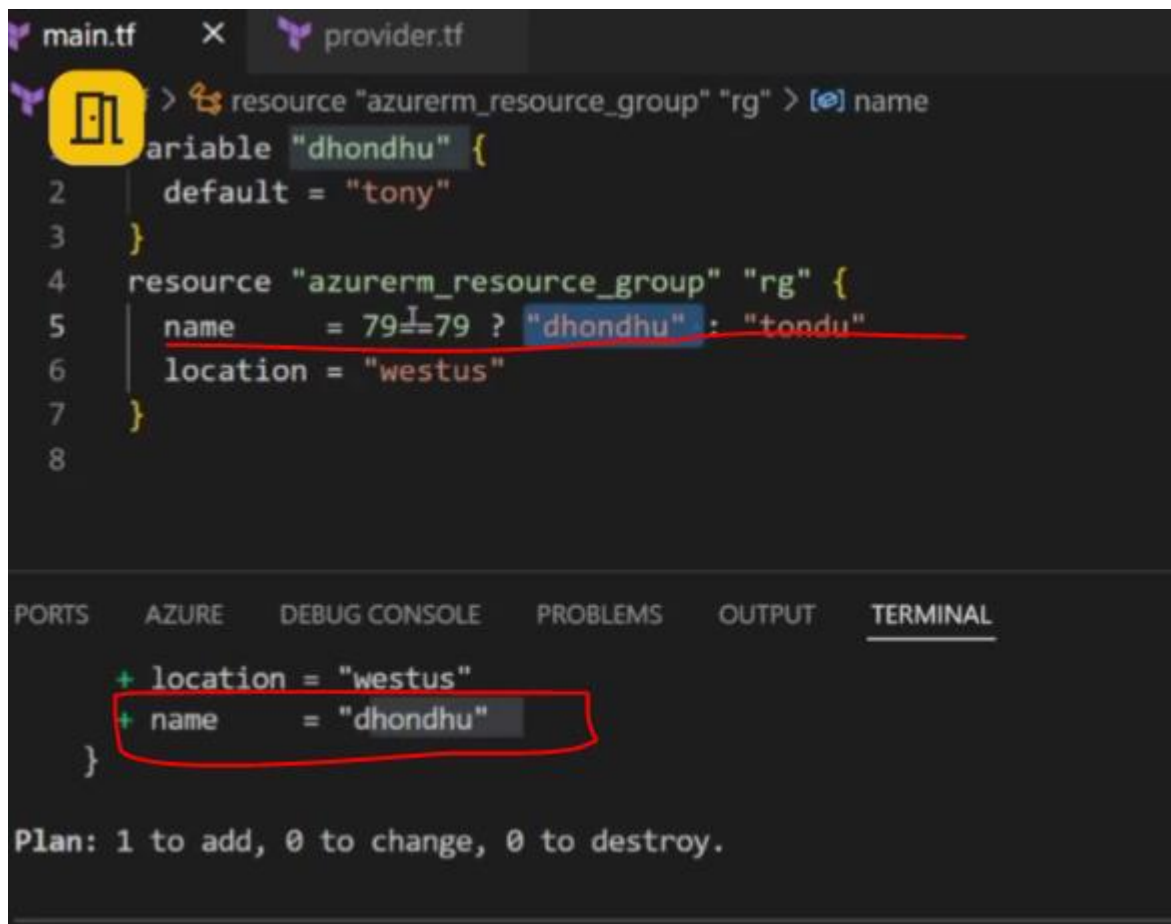
```
1 > resource "azurerm_resource_group" "rg" > [?] name
2 variable "dhondhu" {
3     default = "tony"
4 }
5 resource "azurerm_resource_group" "rg" {
6     name      = false ? "dhondhu" : "tondu"
7     location  = "westus"
8 }
```

A yellow circle highlights the first line of code. A red line underlines the conditional expression `name = false ? "dhondhu" : "tondu"`. The bottom panel shows the `TERMINAL` output:

```
+ resource "azurerm_resource_group" "rg" {
+   + id      = (known after apply)
+   + location = "westus"
+   + name     = "tondu"
+ }
```

A red circle highlights the `name = "tondu"` line in the terminal output.

i) `79==79 ? "dhondhu" : "tondu"`



The screenshot shows a VS Code editor with two tabs: 'main.tf' and 'provider.tf'. The 'main.tf' tab is active, displaying Terraform code. A yellow icon of a building is in the top left corner. The code defines a variable 'dhondhu' with a default value of 'tony' and a resource 'azurerm\_resource\_group' 'rg' with a name that is a conditional expression. A red line is drawn under the conditional expression. The terminal at the bottom shows the plan output for the resource, with the name value highlighted by a red box. The plan indicates that the resource will be added.

```
main.tf  X  provider.tf
> resource "azurerm_resource_group" "rg" > [?] name
variable "dhondhu" {
2   default = "tony"
3 }
4 resource "azurerm_resource_group" "rg" {
5   name      = 79==79 ? "dhondhu" : "tondu"
6   location = "westus"
7 }
8
```

PORTS AZURE DEBUG CONSOLE PROBLEMS OUTPUT TERMINAL

```
+ location = "westus"
+ name      = "dhondhu"
}
```

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

ii) 78==79 ? "dhondhu" : "tondu"

```
File Edit Selection ... < > azure_resource_group_g

main.tf x provider.tf
> resource "azurerm_resource_group" "rg" > [?] name
variable "dhondhu" {
  2   default = "tony"
  3 }
  4 resource "azurerm_resource_group" "rg" {
  5   name      = 78==79 ? "dhondhu" : "tondu"
  6   location = "westus"
  7 }
  8

PORTS AZURE DEBUG CONSOLE PROBLEMS OUTPUT TERMINAL

+ resource "azurerm_resource_group" "rg" {
  + id      = (known after apply)
  I + location = "westus"
  + name     = "tondu"
}
```

i) `var.dhondhu == "tony" && 64329!=64328 ? "dhondhu" : "tondu"`

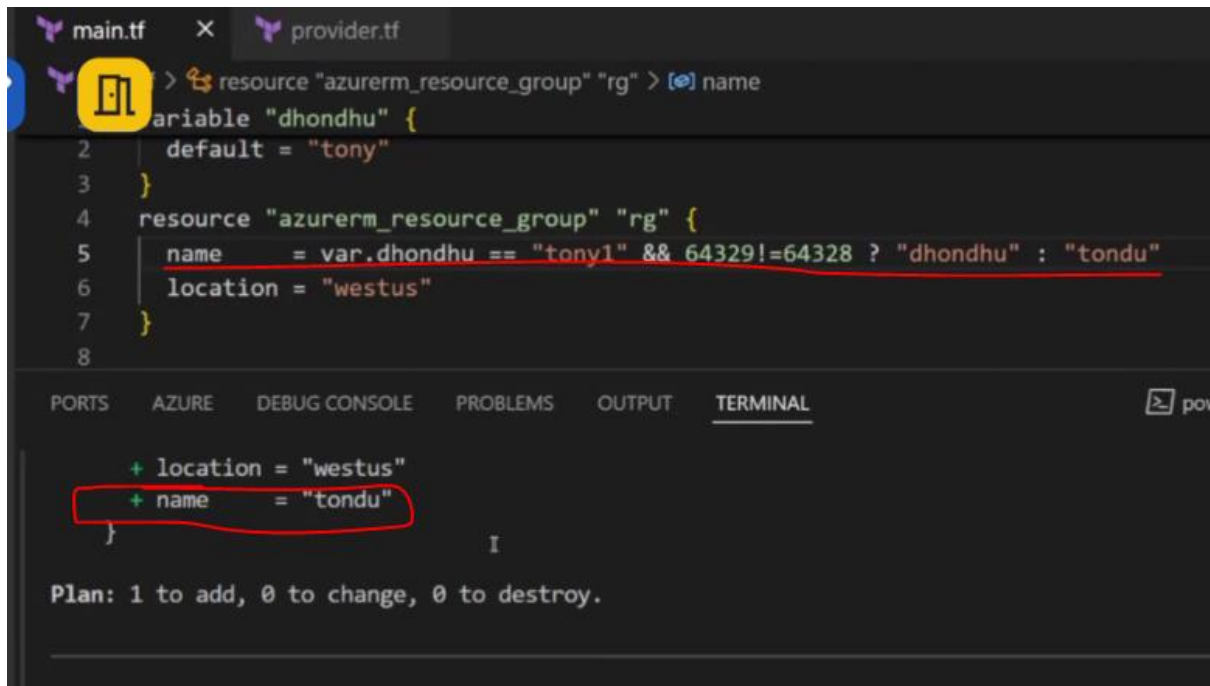
```
main.tf x provider.tf
> resource "azurerm_resource_group" "rg"
variable "dhondhu" {
  2   default = "tony"
  3 }
  4 resource "azurerm_resource_group" "rg" {
  5   name      = var.dhondhu == "tony" && 64329!=64328 ? "dhondhu" : "tondu"
  6   location = "westus"
  7 }
  8

PORTS AZURE DEBUG CONSOLE PROBLEMS OUTPUT TERMINAL

# azurerm_resource_group.rg will be created
+ resource "azurerm_resource_group" "rg" {
  + id      = (known after apply)
  + location = "westus"
  + name     = "dhondhu" I
}

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

ii) `var.dhondhu == "tony1" && 64329!=64328 ? "dhondhu" : "tondu"`



The screenshot shows a VS Code editor with two files: `main.tf` and `provider.tf`. The `main.tf` file contains the following Terraform configuration:

```
1 > resource "azurerm_resource_group" "rg" {
2   |   name = var.dhondhu
3   | }
4   | resource "azurerm_resource_group" "rg" {
5   |   name = var.dhondhu == "tony1" && 64329!=64328 ? "dhondhu" : "tondu"
6   |   location = "westus"
7   | }
8   |
```

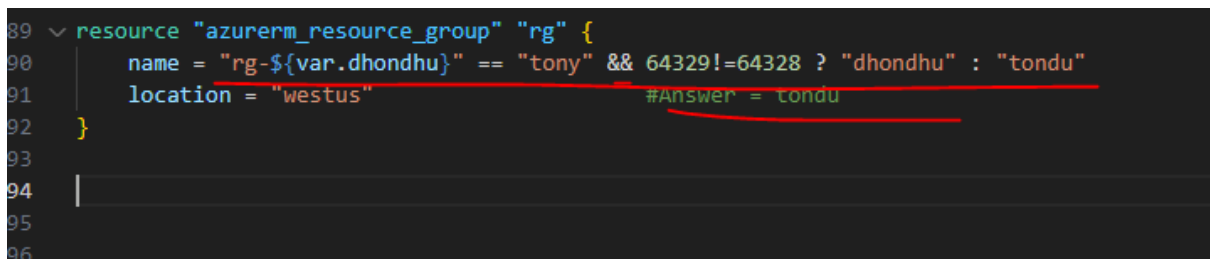
The bottom panel shows the `TERMINAL` output of the `terraform plan` command:

```
+ location = "westus"
+ name      = "tondu"
}
```

Below the plan output, it says: `Plan: 1 to add, 0 to change, 0 to destroy.`

i) `$` means value

`"rg-${var.dhondhu}" == "tony" && 64329!=64328 ? "dhondhu" : "tondu"`



The screenshot shows a VS Code editor with a Terraform configuration file. The configuration is as follows:

```
89 resource "azurerm_resource_group" "rg" {
90   name = "rg-${var.dhondhu}" == "tony" && 64329!=64328 ? "dhondhu" : "tondu"
91   location = "westus"
92 }
93
94
95
96
```

The line `name = "rg-${var.dhondhu}" == "tony" && 64329!=64328 ? "dhondhu" : "tondu"` is underlined in red. A comment `#Answer = tondu` is also present on the same line.

ii) `"rg-${var.dhondhu}" == "tony" || 64329!=64328 ? "dhondhu" : "tondu"`



```
File Edit Selection ... azure_resource_group_generic
main.tf x provider.tf
> resource "azurerm_resource_group" "rg" > [name]
variable "dhondhu" {
2 | default = "tony"
3 | }
4 | resource "azurerm_resource_group" "rg" {
5 |   name = "rg-${var.dhondhu}" == "tony" || 64329!=64328 ? "dhondhu" : "tondu"
6 |   location = "westus"
7 | }
false || true
true
Plan: 1 to add, 0 to change, 0 to destroy.
Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exact]
```

+++++

## AGENDA – Making VM

+++++

## AGENDA – Count loop

- 1)
  - 2) Now suppose we have to delete 3<sup>rd</sup> rg but it will delete 4<sup>th</sup> one as well. We can see below in terraform plan
- i) count =3

```
main.tf x provider.tf
> resource "azurerm_resource_group" "rg" {
  count = 3
  name  = "rg-tony-${count.index+1}"
  location = "westus"
}

# resource "azurerm_resource_group" "rg1" {
#   for_each = ["rg-tony-1", "rg-tony-2", "rg-tony-4", "rg-tony-5"]
#   name     = each.value
#   location = "westus"
```

PORTS AZURE DEBUG CONSOLE PROBLEMS OUTPUT TERMINAL

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

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee if you run "terraform apply" now.

ii) count = 0

```
main.tf x provider.tf
> resource "azurerm_resource_group" "rg" {
  count = 0
  name  = "rg-tony-${count.index+1}"
  location = "westus"
}
```

PORTS AZURE DEBUG CONSOLE PROBLEMS OUTPUT TERMINAL

PS C:\DevOpsInsiders\Batch16\azure-devsecops-batch-16\CodeSamples\ZELI  
p> terraform apply

No changes. Your infrastructure matches the configuration.

Terraform has compared your real infrastructure against your configuration and no changes are needed.

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

PS C:\DevOpsInsiders\Batch16\azure-devsecops-batch-16\CodeSamples\ZELI

i) In terraform.tfvars when passed create-rg value as true.

The screenshot shows the VS Code editor with two files: `main.tf` and `terraform.tfvars`. The `main.tf` file contains a variable `create_rg` of type `bool` and a resource `azurerm_resource_group` named `rg` with a `count` of `var.create_rg ? 1 : 0`. The `terraform.tfvars` file has `create_rg = true`. The terminal shows the output of `terraform plan`, indicating that one resource will be added. A red box highlights the plan output for the resource `rg`, showing its `name` as `rg-tony-1` and `location` as `westus`. A red arrow points from the `create_rg = true` in `terraform.tfvars` to the plan output.

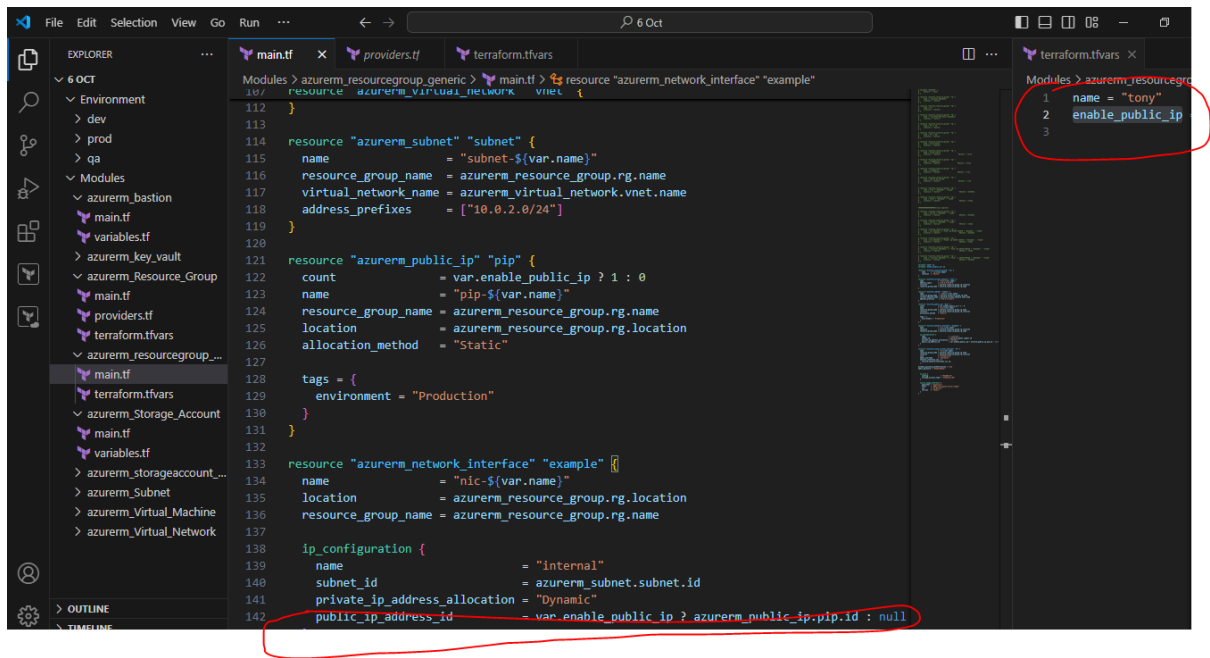
```
main.tf
1 resource "azurerm_resource_group" "rg" {
2   location = "westus"
3 }
4
5 variable "create_rg" {
6   type = bool
7 }
8
9 resource "azurerm_resource_group" "rg" {
10  count = var.create_rg ? 1 : 0
11  name = "rg-tony-${count.index+1}"
12  location = "westus"
13 }
14
15 terraform.tfvars
16 create_rg = true
17
18 terminal
19 + resource "azurerm_resource_group" "rg" {
20   + id = (known after apply)
21   + location = "westus"
22   + name = "rg-tony-1"
23 }
24
25 Plan: 1 to add, 0 to change, 0 to destroy.
```

ii) In terraform.tfvars when passed create-rg value as false.

The screenshot shows the VS Code editor with the same `main.tf` file as the previous image. The `terraform.tfvars` file now has `create_rg = false`. The terminal shows the output of `terraform plan`, indicating that no changes are required. A red box highlights the plan output, showing `No changes. Your infrastructure matches the configuration.` A red arrow points from the `create_rg = false` in `terraform.tfvars` to the plan output.

```
main.tf
1 resource "azurerm_resource_group" "rg" {
2   location = "westus"
3 }
4
5 variable "create_rg" {
6   type = bool
7 }
8
9 resource "azurerm_resource_group" "rg" {
10  count = var.create_rg ? 1 : 0
11  name = "rg-tony-${count.index+1}"
12  location = "westus"
13 }
14
15 terraform.tfvars
16 create_rg = false
17
18 terminal
19 PS C:\DevOpsInsiders\Batch16\azure-devsecops-batch-16\CodeSamples\ZEELECTRIC\modules\azurerm_resource
20 p> terraform plan
21
22 No changes. Your infrastructure matches the configuration.
23
24 Terraform has compared your real infrastructure against your configuration and found no differences
```

i)



+++++

