How to use Terraform output values?

PYTHON:

>>>Print (“Hello world”)

Hello world

JAVA:

Public class main {

Public static void main(String[] args) {

System.out.println(“Welcome to mr cloud book”);

}

}

In this article we are going to look on [terraform](https://www.terraform.io/) output values. Terraform output values will be really useful when you want to debug your terraform code. Terraform output values can help you to print the attributes reference(arn, instance\_state, outpost\_arn, public\_ip, public\_dns etc) on your console.

The typical syntax of Terraform output values is –

How to use terraform output locals?

Once you run your terraform apply command, it is going to print Hello this is output onto your console.

But there are more than just simply outputting static text values with [terraform output variables](https://developer.hashicorp.com/terraform/language/values/outputs). You can do a lot more.

output "my\_console\_output" {

value = “Hello,welcome to mr-cloud-book”

}

1. How to print the public\_ip of aws\_instance?

This is one of the most classic examples for terraform output values. As we know terraform is used as Infrastructure as code, so with the help of terraform you can provision many resources such - [aws\_instances](https://aws.amazon.com/" \t "_blank), aws\_vpc etc.

But is there a way by which you can know the public\_ip address of the instance which you are planning to provision using terraform?

Yes, there is a really convenient and easy way to achieve that.

Here is my aws\_instance which I have defined as inside my main.tf -

provider "aws" {

region = "ap-south-1"

profile = "mr-cloud-book"

}

resource "aws\_instance" "ec2\_example" {

ami = "ami-0cca134ec43cf708f"

instance\_type = "t2.micro"

tags = {

Name = "cloud - Terraform EC2"

}

}

You can simply append following terraform output value code inside your main.tf file so that it can print the public ip of your aws\_instance

output "my\_console\_output" {

value = aws\_instance.ec2\_example.public\_ip

}

Break down of the above script

1. aws\_instance - It is the keyword which you need write as is.
2. ec2\_example - The name which we have given at the time of creating aws\_instance
3. public\_ip - You can use [**attributes reference**](https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/instance#public_ip) page to get all the attribute which you want to print on console.

The same approach can be followed to print - arn,instance\_state,outpost\_arn, password\_data, primary\_network\_interface\_id, private\_dns, public\_dns etc.

2. How to create output.tf for terraform output values?

In the previous point we have seen how to create your first **terraform output values**. But if you have noticed we have created the terraform output values inside the same main.tffile.

It is completely okay to create **terraform output values** inside the main.tf but this practice is certainly not recommended in the industry.

The recommended way is to create a separate output.tf specially of the **terraform output values**, so that all the output values can be stored there.

The only change which you need to do over here is to create a new output.tf and store the following terraform code init -

output.tf

output "my\_console\_output" {

value = aws\_instance.ec2\_example.public\_ip

}

Your directory structure should look like this -

How to use terraform output locals?

3. How to prevent printing sensitive info on the console?

As terraform output values help us to print the attributes reference values but sometimes you can not print all the values on console.

So to prevent from printing sensitive values on the console you need to set sensitive = true.

Here is an example of the terraform code where we want to prevent showing the **public ip** to console -

output "my\_console\_output" {

value = aws\_instance.ec2\_example.public\_ip

sensitive = true

}

In the above code if you noticed we are using sensitive = true which tells terraform not to show public ip on console.

## How to use Terraform output for resource with count?

There are certain situation where you need to use terraform output for multiple resources using the count. But to achieve this you do not need to loop inside the terraform output block.

Here is an example in which I have created the **3 EC2 Instances** on AWS with terraform and I am using only single output block to print all the three public DNS info of the EC2 instances -

*#Following resource block will create 3 EC2 instances*

resource "aws\_instance" "ec2\_example" {

count = 3

ami = "ami-0cca134ec43cf708f"

instance\_type = "t2.micro"

tags = {

Name = "test-t2-micro"

}

}

**Output Block to fetch all the three public DNS**

output "fetched\_info\_from\_aws" {

value = ["${aws\_instance.ec2\_example.\*.public\_dns}"]

}

## Various use cases of output variables

Here are some common use cases for Terraform output variables include:

1. **Print IP address of a load balancer** - Exposing the IP address of a load balancer to be used by other parts of the infrastructure for routing traffic.

*#main.tf*

resource "aws\_elb" "example" {

name = "example"

internal = true

security\_groups = [aws\_security\_group.example.id]

listener {

instance\_port = 80

instance\_protocol = "http"

lb\_port = 80

lb\_protocol = "http"

}

}

*#The following output variable will print the IP address of loadbalancer*

output "elb\_dns\_name" {

value = aws\_elb.example.dns\_name

}

*TERRAFORM*

1. **Print DNS name of Database** - Exposing the DNS name of a database to be used by other parts of the infrastructure for connecting to the database.

*#main.tf*

resource "aws\_rds\_cluster" "example" {

cluster\_identifier = "example"

master\_username = "example"

master\_password = "example"

db\_subnet\_group\_name = aws\_db\_subnet\_group.example.name

}

*#The following output variable will print the DNS name of AWS RDS cluster*

output "rds\_cluster\_endpoint" {

value = aws\_rds\_cluster.example.endpoint

}

*TERRAFORM*

1. **Get the URL of static website deployed on AWS-** If you have ever deployed an static website on S3 bucket then you can print the website\_endpoint using the output variables in terraform -

*#main.tf*

resource "aws\_s3\_bucket" "example" {

bucket = "example-bucket"

acl = "public-read"

website {

index\_document = "index.html"

}

}

# The following output block will print the URL of static website deployed on S3 Bucket

output "website\_url" {

value = aws\_s3\_bucket.example.website\_endpoint

}

*TERRAFORM*

1. **Exporting the ARN of an AWS IAM role** - Exporting the ARN of an AWS IAM role to be used by other parts of the infrastructure for granting permissions.

*#main.tf*

resource "aws\_iam\_role" "example" {

name = "example"

assume\_role\_policy = <<EOF

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Principal": {

"Service": "ec2.amazonaws.com"

},

"Action": "sts:AssumeRole"

}

]

}

EOF

}

*#print the arn using the output variable*

output "role\_arn" {

value = aws\_iam\_role.example.arn

}

*TERRAFORM*

## How do you output multiple values in Terraform?

In Terraform, you can output multiple values by defining multiple output blocks in your configuration. Each output block should have a unique name and a corresponding value.

For example, you could output both the public and private IP addresses of an EC2 instance like this:

resource "aws\_instance" "example" {

ami = "ami-0ff8a91507f77f867"

instance\_type = "t2.micro"

}

*# Output-1*

output "public\_ip" {

value = aws\_instance.example.public\_ip

}

*#Output-2*

output "private\_ip" {

value = aws\_instance.example.private\_ip

}

*TERRAFORM*

**Using Map Function-** You can also output multiple values in a single output block by using the **map function**.

output "instance\_ips" {

value = {

public\_ip = aws\_instance.example.public\_ip

private\_ip = aws\_instance.example.private\_ip

}

}

*TERRAFORM*

**Using join function-** You can also output multiple attributes from a resource using join function

output "security\_groups" {

value = join(",",aws\_instance.example.\*.security\_groups)

}

*TERRAFORM*

## How to access the output variable in Terraform module?

In Terraform, you can access output variables defined in a [module](https://jhooq.com/terraform-module) by using the module or module.name syntax, where name is the name of the module. For example, if you have a module named "example" with an output variable named "output\_var", you can access it in the root module like this:

module "example" {

source = "path/to/module"

*# ... other module inputs*

}

*#output from module*

output "example\_output" {

value = module.example.output\_var

}

*TERRAFORM*

You can also use the output variables in other resources or modules in the same way. For example, if you have another module that needs to use the output of the example module, you can reference it like this:

module "example2" {

source = "path/to/module2"

example\_output = module.example.output\_var

*# ... other module inputs*

}

*TERRAFORM*

Alternatively, you can use the output block in the child module and then reference it in the root module by using the module and module.name syntax.

module "example" {

source = "path/to/module"

*# ... other module inputs*

}

output "example\_output" {

value = module.example.output\_var

}

*TERRAFORM*

## Explain terraform output command with example and how it is different from output variable?

The [output command](https://developer.hashicorp.com/terraform/cli/commands/output) is used to display the values of output variables defined in the configuration. It can be used to show the values of output variables after a successful terraform apply command has been run.

The syntax of the output command is:

terraform output [options] [NAME]

*TERRAFORM*

Where NAME is the name of the output variable that you want to display. If you don't specify a NAME, it will display all the output variables defined in your configuration.

For example, consider the following configuration:

resource "aws\_instance" "example" {

ami = "ami-0ff8a91507f77f867"

instance\_type = "t2.micro"

}

output "public\_ip" {

value = aws\_instance.example.public\_ip

}

output "private\_ip" {

value = aws\_instance.example.private\_ip

}

*TERRAFORM*

After running terraform apply, you can display the value of the public\_ip output variable by running the command terraform output public\_ip which will output something like this:

public\_ip = xx.xxx.xxx.xxx

*BASH*

Similarly, you can display the value of the private\_ip output variable by running the command terraform output private\_ip which will output something like this:

private\_ip = xxx.xxx.xxx.xxx

*BASH*

On the other hand, the output variable is a block defined in the Terraform configuration file, it allows you to extract information from the state file and make it available for use elsewhere in the same configuration file.

The output command is used to display the values of output variables that have been defined in the configuration, it does not change anything in the infrastructure, but it shows the current state of the output variables defined in the configuration file.

## How to display sensitive inside output block using Terraform nonsensitive function?

In Terraform, the [nonsensitive function](https://developer.hashicorp.com/terraform/language/functions/sensitive" \t "_blank) can be used to mark an output variable as sensitive, which means that its value will be obscured in the output of the terraform output command. This can be useful for sensitive information like passwords or private keys.

The nonsensitive function can be used to mark an output variable as non-sensitive, which means that its value will be displayed in plain text in the output of the terraform output command.

Here is an example of how you might use the nonsensitive function to mark an output variable as non-sensitive:

resource "aws\_iam\_access\_key" "example" {

user = aws\_iam\_user.example.name

}

output "access\_key" {

value = nonsensitive(aws\_iam\_access\_key.example.id)

}

*TERRAFORM*

In the above example, the aws\_iam\_access\_key resource is being created and the output variable access\_key is being set to the ID of the access key, but the nonsensitive function is used to mark the output variable as non-sensitive.