**Data Sources**

*Data sources* allow Terraform to use information defined outside of Terraform, defined by another separate Terraform configuration, or modified by functions.

# How to use Terraform Data sources?

[Terraform](https://www.terraform.io/) data sources can be beneficial if you want to retrieve or fetch the data from the cloud service providers such as [AWS](https://aws.amazon.com/), [AZURE](https://azure.microsoft.com/en-us/), and [GCP](https://cloud.google.com/). Most of the time when we use Terraform along with AWS/AZURE/GCP then we always send data in terms of instructions or configuration.

***But what if you want to get the information(arn, tags, owner\_id, etc.) back from the cloud service provider AWS/AZURE/GCP?***

**Answer** - We need to use the data sources to get the resource information back.

So Terraform Data Sources are a kind of an API that fetches the data/information from the resources running under the cloud infra and sending it back to terraform configuration for further use.

In this blog, we will look at the example in which we are going to create an aws\_instance resource and then create a data source to fetch some of the information associated with the aws\_instance.

## **1. Create an aws\_instance**

The motive of this exercise is to create an aws\_instance and then create a data source to fetch all the possible [Data Source: aws\_instance attributes](https://registry.terraform.io/providers/hashicorp/aws/latest/docs/data-sources/instance).

Let's first write the terraform configuration for starting a t2.mirco aws\_instance.

(\*Note- Replace the access\_key and secret\_key with your AWS account. [*Click here to know how to generate the access\_key and secret\_key*](https://jhooq.com/terraform-ec2-instance-setup/#2generate-access-keys-access-key-id-and-secret-access-key) )

provider "aws" {

region = "eu-central-1"

access\_key = "AKIATQ37NXB2JMXVGYPG"

secret\_key = "ockvEN1DzYynDuKIh56BVQv/tMqmzvKnYB8FttSp"

}

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

ami = "ami-0767046d1677be5a0"

instance\_type = "t2.micro"

tags = {

Name = "Terraform EC2"

}

}

*BASH*

## **2. Define a data source**

Now we have created our aws\_instance in [Step 1](https://jhooq.com/terraform-data-sources/#1-create-an-_aws_instance_), let's add the data source to the existing terraform configuration.

Here is the data source configuration for fetching all the information of aws\_instance -

data "aws\_instance" "myawsinstance" {

filter {

name = "tag:Name"

values = ["Terraform EC2"]

}

depends\_on = [

"aws\_instance.ec2\_example"

]

}

*BASH*

**Key points to pay attention for -**

1. **filter**: Although we have created only one instance but still we have used filter because in a production-like environment you might have multiple aws\_instance running, so you need to filter the instance anyhow. And since we have tagged our aws\_instance with the name Terraform EC2 so we are going to use the same name inside the filter also.
2. **depends\_on**: The second important parameter is depends\_on because data source does not know by its own which resource it belongs to, so we are going to add the depends\_on parameter.

## **3. Create Output variable for data source**

So far in [Step 1](https://jhooq.com/terraform-data-sources/#1-create-an-_aws_instance_) and [Step 2](https://jhooq.com/terraform-data-sources/#2-define-a-data-source) we have created the aws\_instance and data source, now let's create an output value so that we can see all the information fetched or retrieved by the data source.

Here is the terraform configuration for the output value -

output "fetched\_info\_from\_aws" {

value = data.aws\_instance.myawsinstance

}

*BASH*

**Key points to pay attention for -**

1. We have linked the output value to the data source which we have created in [Step 2](https://jhooq.com/terraform-data-sources/#2-define-a-data-source).
2. To link the output value we are going to use the data source name .i.e. - data.aws\_instance.myawsinstance

## **4. Apply the final terraform configuration along with data source and output values**

Alright now I am assuming you have gone through all the 3 steps([Step 1](https://jhooq.com/terraform-data-sources/#1-create-an-_aws_instance_),[Step 2](https://jhooq.com/terraform-data-sources/#2-define-a-data-source), and [Step 3](https://jhooq.com/terraform-data-sources/#3-create-output-variable-for-data-source)), so here is our final terraform configuration including aws\_instance, data source, and output values

provider "aws" {

region = "eu-central-1"

access\_key = "AKIATQ37NXB2JMXVGYPG"

secret\_key = "ockvEN1DzYynDuKIh56BVQv/tMqmzvKnYB8FttSp"

}

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

ami = "ami-0767046d1677be5a0"

instance\_type = "t2.micro"

tags = {

Name = "Terraform EC2"

}

}

data "aws\_instance" "myawsinstance" {

filter {

name = "tag:Name"

values = ["Terraform EC2"]

}

depends\_on = [

"aws\_instance.ec2\_example"

]

}

output "fetched\_info\_from\_aws" {

value = data.aws\_instance.myawsinstance.public\_ip

}

*BASH*

You can simply run the following terraform command to create your aws\_instance -

terraform init

*BASH*

terraform plan

*BASH*

terraform apply

*BASH*

**Here is the output after applying to terraform configuration -**

Outputs:

fetched\_info\_from\_aws = {

"ami" = "ami-0767046d1677be5a0"

"arn" = "arn:aws:ec2:eu-central-1:242396018804:instance/i-0eda1c6a59790eb7d"

"associate\_public\_ip\_address" = true

"availability\_zone" = "eu-central-1c"

"credit\_specification" = tolist([

{

"cpu\_credits" = "standard"

},

])

"disable\_api\_termination" = false

"ebs\_block\_device" = toset([])

"ebs\_optimized" = false

"enclave\_options" = tolist([

{

"enabled" = false

},

])

"ephemeral\_block\_device" = tolist([])

"filter" = toset([

{

"name" = "tag:Name"

"values" = tolist([

"Terraform EC2",

])

},

])

"get\_password\_data" = false

"get\_user\_data" = false

"host\_id" = tostring(null)

"iam\_instance\_profile" = ""

"id" = "i-0eda1c6a59790eb7d"

"instance\_id" = tostring(null)

"instance\_state" = "running"

"instance\_tags" = tomap(null) /\* of string \*/

"instance\_type" = "t2.micro"

"key\_name" = ""

"metadata\_options" = tolist([

{

"http\_endpoint" = "enabled"

"http\_put\_response\_hop\_limit" = 1

"http\_tokens" = "optional"

},

])

"monitoring" = false

"network\_interface\_id" = "eni-0ffc9d62eafcafcbc"

"outpost\_arn" = ""

"password\_data" = tostring(null)

"placement\_group" = ""

"private\_dns" = "ip-172-31-9-122.eu-central-1.compute.internal"

"private\_ip" = "172.31.9.122"

"public\_dns" = "ec2-3-122-249-219.eu-central-1.compute.amazonaws.com"

"public\_ip" = "3.122.249.219"

"root\_block\_device" = toset([

{

"delete\_on\_termination" = true

"device\_name" = "/dev/sda1"

"encrypted" = false

"iops" = 100

"kms\_key\_id" = ""

"tags" = tomap({})

"throughput" = 0

"volume\_id" = "vol-0fce01580b0175da8"

"volume\_size" = 8

"volume\_type" = "gp2"

},

])

"secondary\_private\_ips" = toset([])

"security\_groups" = toset([

"default",

])

"source\_dest\_check" = true

"subnet\_id" = "subnet-2183316d"

"tags" = tomap({

"Name" = "Terraform EC2"

})

"tenancy" = "default"

"user\_data" = tostring(null)

"user\_data\_base64" = tostring(null)

"vpc\_security\_group\_ids" = toset([

"sg-272bd157",

])

}

...

*BASH*

**Here is the screenshot from aws -**

1. aws\_instance is up and running -

Terraform aws\_instance up and running with data source

1. aws\_instance details (you can verify the output from [step 4](https://jhooq.com/terraform-data-sources/#4-apply-the-final-terraform-configuration-along-with-_data-source_-and-_output-values_)) -

Terraform data source with aws\_instance

## **5. Fetching only specific attribute using data source**

Now let's one step more further and instead of fetching all the attributes of the aws\_instance let's only fetch the public\_ip.

You only need to update the output value configuration -

output "fetched\_info\_from\_aws" {

value = data.aws\_instance.myawsinstance.public\_ip

}

*BASH*

Here is the public\_ip information fetched by the data source