Workaround the 16k Character Limit for EC2 User Data: Terraform Solution

Kevin De Notariis Follow
May 21 · 4 min read

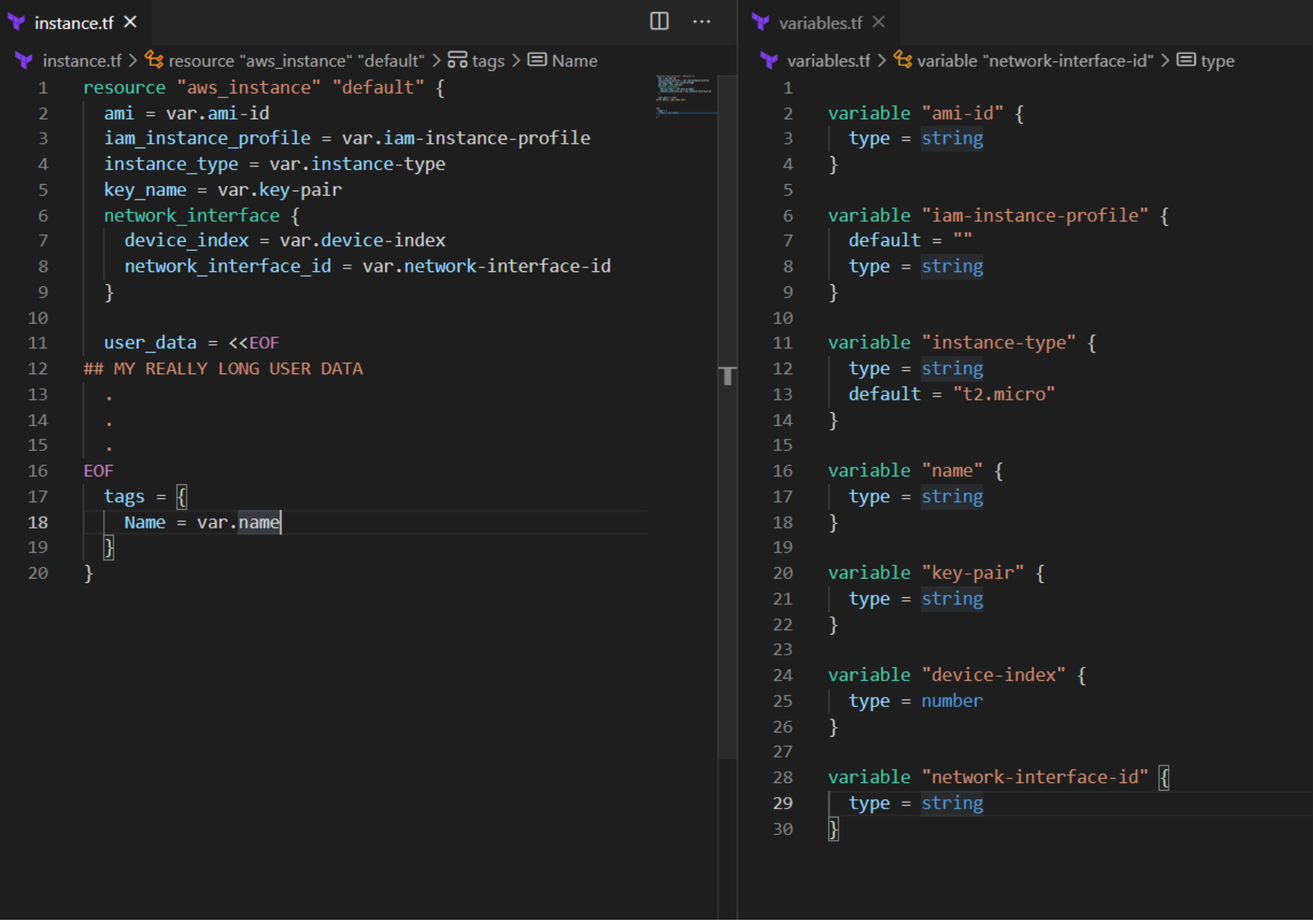
In this brief article, I would like to explain a workaround on the 16k characters limit for the user data in an EC2 instance.

It is pretty straightforward and the bottom line is: upload your configuration files to an S3 bucket and grab them in the `user data` of the instance and run them.

Setup

I suppose you already have an AWS infrastructure governed by your Terraform code with an EC2 instance where you want to stick a long `user data` configuration.

The situation might be something like the following:


instance.tf
instance.tf > resource "aws_instance" "default" > tags > Name
variables.tf
variables.tf > variable "network-interface-id" > type

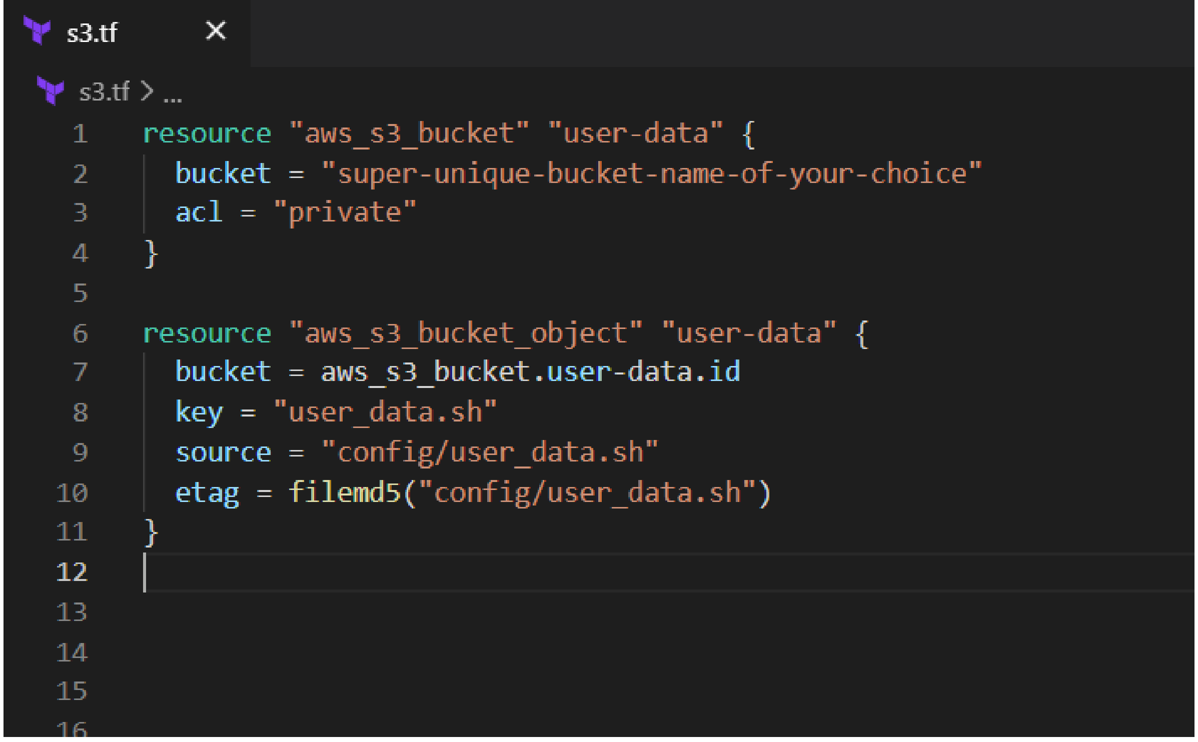
```
resource "aws_instance" "default" {
  ami = var.ami-id
  iam_instance_profile = var.iam-instance-profile
  instance_type = var.instance-type
  key_name = var.key-pair
  network_interface {
    device_index = var.device-index
    network_interface_id = var.network-interface-id
  }

  user_data = <<EOF
## MY REALLY LONG USER DATA
  .
  .
  .
EOF
  tags = {
    Name = var.name
  }
}
```

```

variable "ami-id" {
  type = string
}

variable "iam-instance-profile" {
  default = ""
  type = string
}

variable "instance-type" {
  type = string
  default = "t2.micro"
}

variable "name" {
  type = string
}

variable "key-pair" {
  type = string
}

variable "device-index" {
  type = number
}

variable "network-interface-id" {
  type = string
}
```

Example of a setup

Namely we have a module defining an EC2 instance in a file called `instance.tf`, where its parameters are variables defined in a `variables.tf` file (kind of usual setup for terraform).

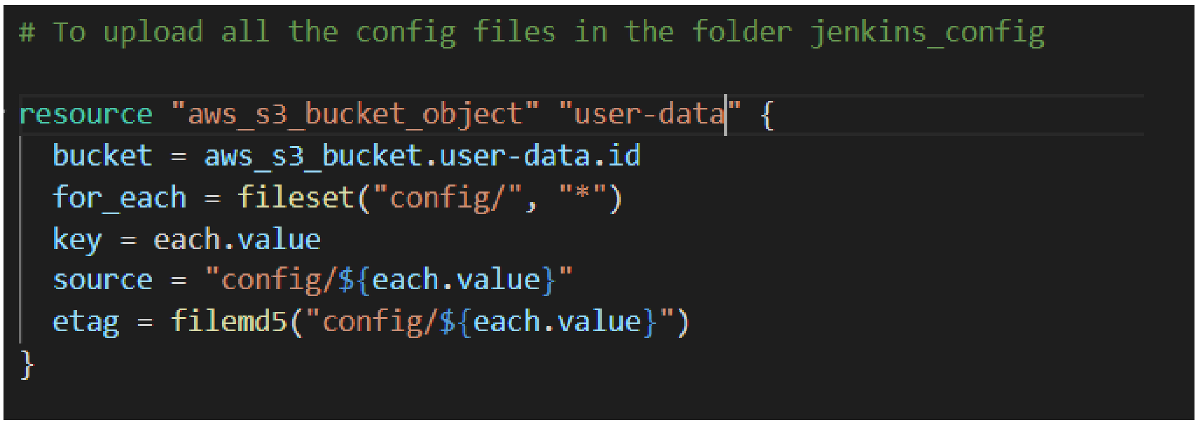
Supposing that the EC2 instance already has access to S3 buckets (otherwise you need to modify the `aws_iam_role` by adding the suitable policy) we can create an `s3.tf` file where we define our bucket that will store the config file(s) and we place the actual file:



```
resource "aws_s3_bucket" "user-data" {
  bucket = "super-unique-bucket-name-of-your-choice"
  acl = "private"
}

resource "aws_s3_bucket_object" "user-data" {
  bucket = aws_s3_bucket.user-data.id
  key = "user_data.sh"
  source = "config/user_data.sh"
  etag = filemd5("config/user_data.sh")
}
```

With these instructions, we create a bucket called `super-unique-bucket-name-of-your-choice` (which you should change to make it somewhat meaningful, but keeping in mind that it should be unique across **all** AWS) and we place there a `user_data.sh` file stored in a folder `config` .

If we had more than one bash script to run at creation time in the user data, we could upload all of them using the following:

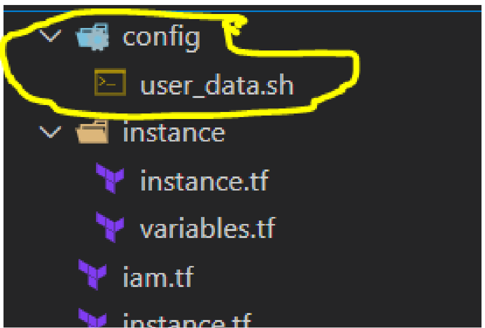


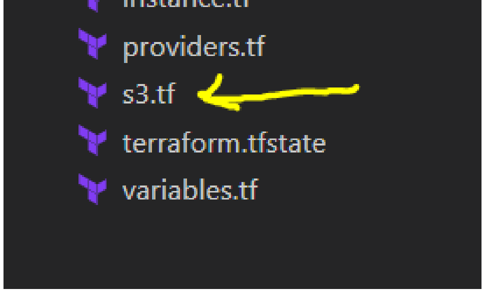
```
# To upload all the config files in the folder jenkins_config

resource "aws_s3_bucket_object" "user-data" {
  bucket = aws_s3_bucket.user-data.id
  for_each = fileset("config/", "*")
  key = each.value
  source = "config/${each.value}"
  etag = filemd5("config/${each.value}")
}
```

To upload multiple script to run in the user data

Single User Data Script

For the simplest case of one `user_data.sh` script, the folder structure should then be something like:

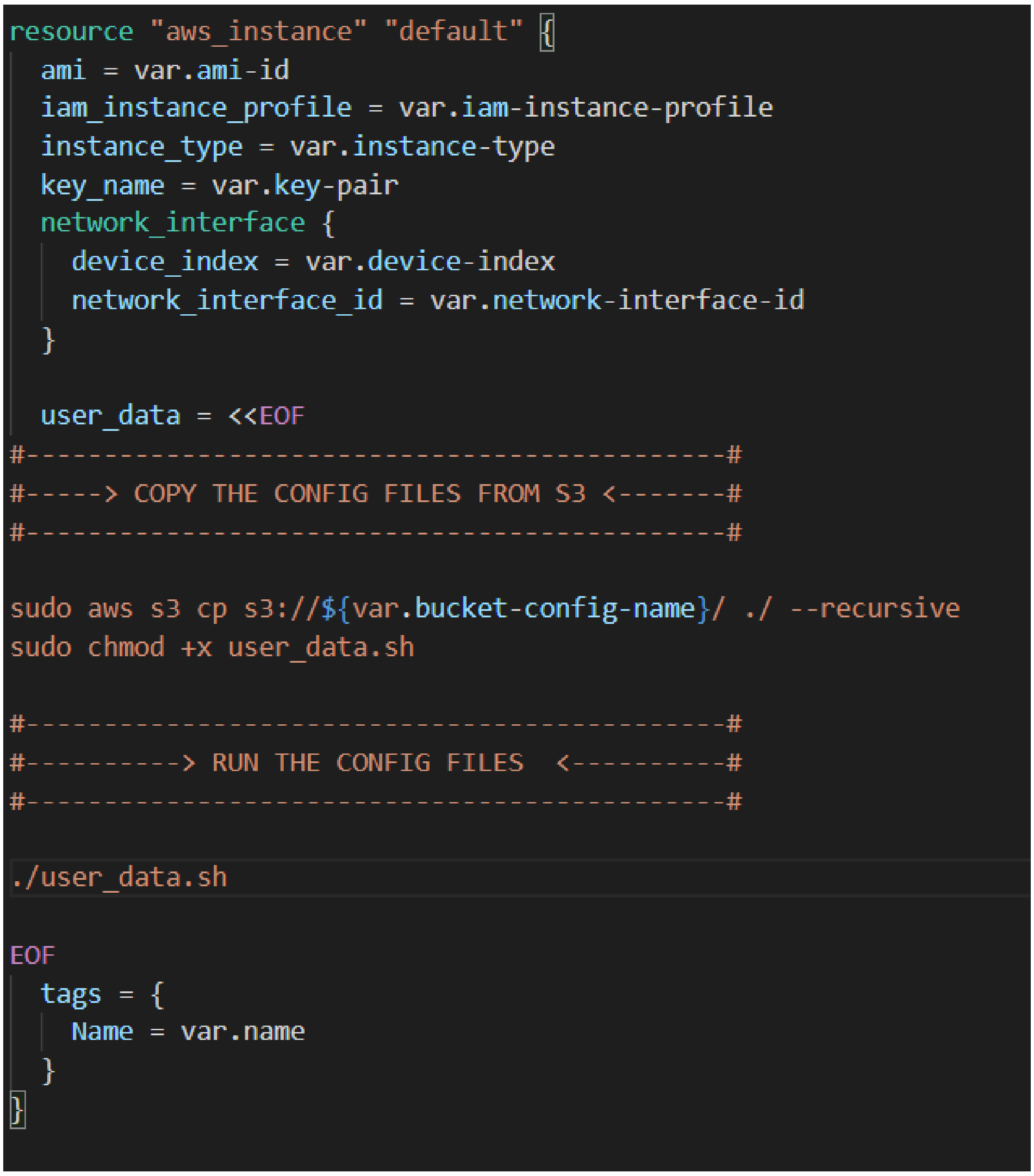




```
providers.tf
s3.tf
terraform.tfstate
variables.tf
```

folder structure

We need then to modify the `instance/instance.tf` user_data as follows:



```
resource "aws_instance" "default" {
  ami = var.ami-id
  iam_instance_profile = var.iam-instance-profile
  instance_type = var.instance-type
  key_name = var.key-pair
  network_interface {
    device_index = var.device-index
    network_interface_id = var.network-interface-id
  }

  user_data = <<EOF
#--------------------------------------------#
#-----> COPY THE CONFIG FILES FROM S3 <-------#
#--------------------------------------------#

sudo aws s3 cp s3://${var.bucket-config-name}/ ./ --recursive
sudo chmod +x user_data.sh

#--------------------------------------------#
#---------> RUN THE CONFIG FILES  <----------#
#--------------------------------------------#

./user_data.sh

EOF
  tags = {
    Name = var.name
  }
}
```

New user data of the instance

Where we added the following directives:

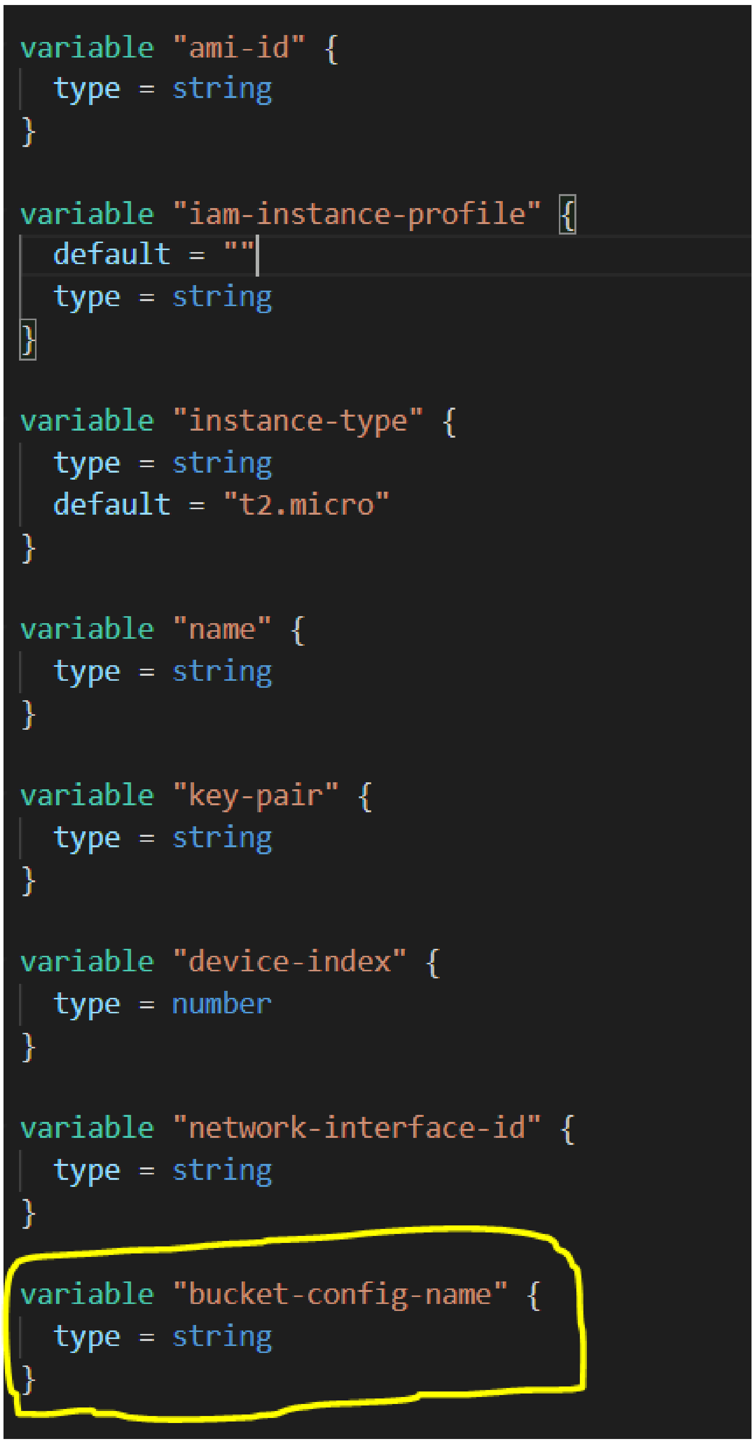
```
sudo aws s3 cp s3://${var.bucket-config-name}/ ./ --recursive

sudo chmod +x user_data.sh

./user_data.sh
```

The first one will copy (`cp`) from the bucket specified in the variable `var.bucket-config-name` all its elements in the 'current directory' (`./`).

Then, we mark the file that will be downloaded (`user_data.sh`) as an executable.

Finally, we execute that script.

Clearly, we need to define a new variable in `instance/variables.tf` :



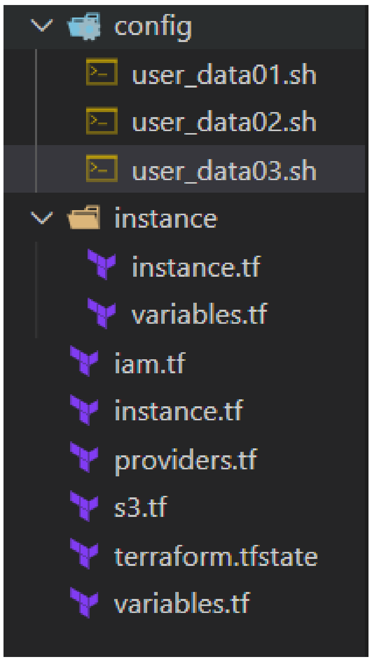
```
variable "ami-id" {
  type = string
}

variable "iam-instance-profile" {
  default = ""
  type = string
}

variable "instance-type" {
  type = string
  default = "t2.micro"
}

variable "name" {
  type = string
}

variable "key-pair" {
  type = string
}

variable "device-index" {
  type = number
}

variable "network-interface-id" {
  type = string
}

variable "bucket-config-name" {
  type = string
}
```

Which, in the `instance.tf` (in the root directory), should be populated with the name of the bucket, namely with the following directive:

```
bucket-config-name = aws_s3_bucket.user_data.id
```

Multiple User Data Scripts

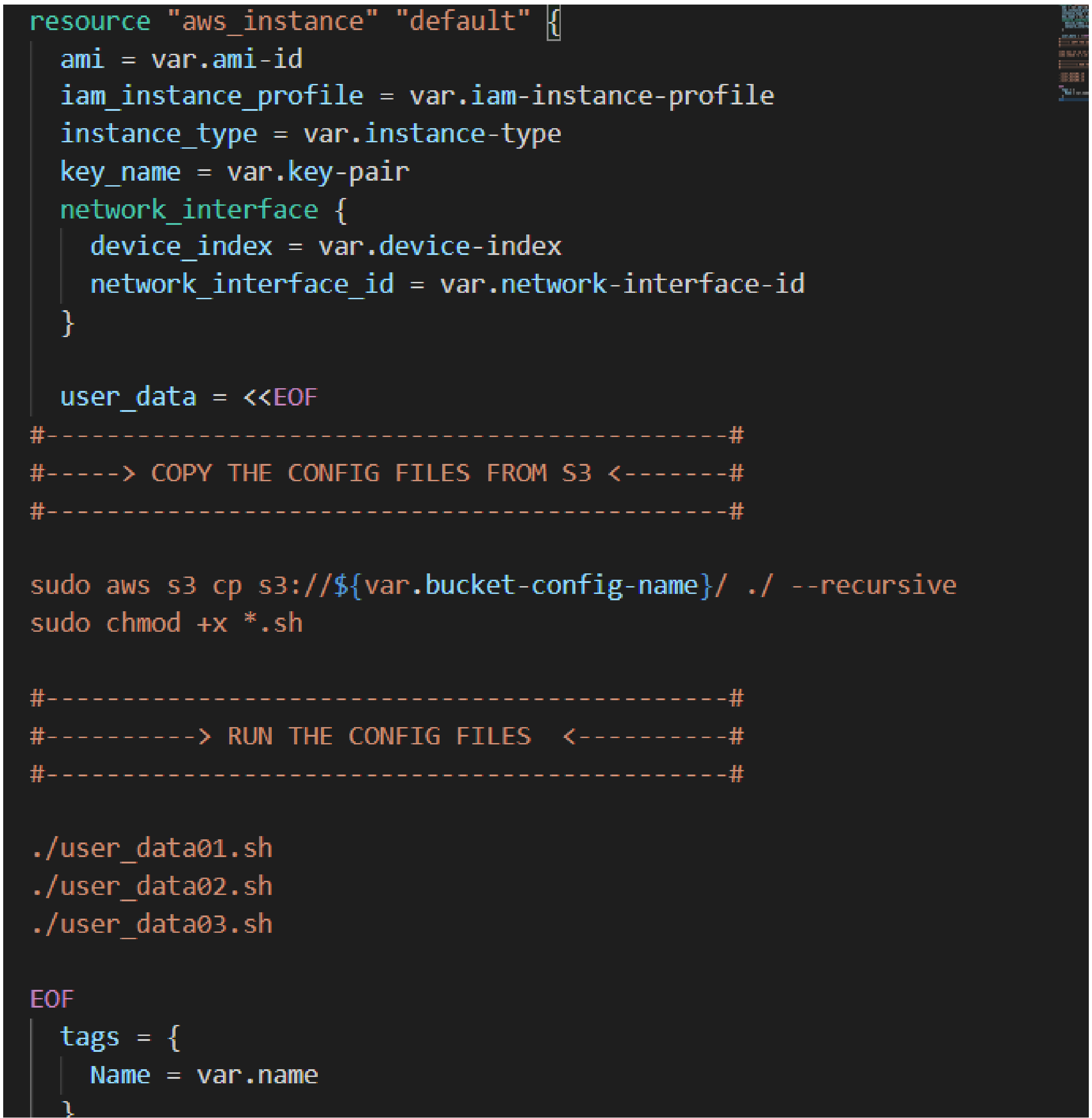
In the case of multiple scripts, the folder structure should instead looks like:



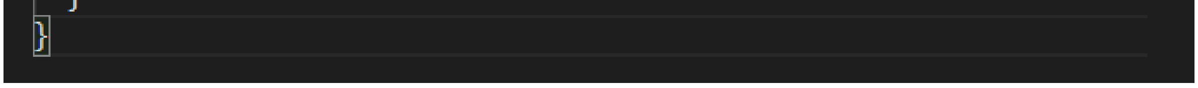
```
config
    user_data01.sh
    user_data02.sh
    user_data03.sh
instance
    instance.tf
    variables.tf
iam.tf
instance.tf
providers.tf
s3.tf
terraform.tfstate
variables.tf
```

folder structure for more user data scripts

In this case, the new `user_data` of the instance, should then be modified as follows:



```
resource "aws_instance" "default" {
  ami = var.ami-id
  iam_instance_profile = var.iam-instance-profile
  instance_type = var.instance-type
  key_name = var.key-pair
  network_interface {
    device_index = var.device-index
    network_interface_id = var.network-interface-id
  }

  user_data = <<EOF
#-----------------------------------------------#
#-----> COPY THE CONFIG FILES FROM S3 <-------#
#-----------------------------------------------#

sudo aws s3 cp s3://${var.bucket-config-name}/ ./ --recursive
sudo chmod +x *.sh

#-----------------------------------------------#
#----------> RUN THE CONFIG FILES  <----------#
#-----------------------------------------------#

./user_data01.sh
./user_data02.sh
./user_data03.sh

EOF
  tags = {
    Name = var.name
  }
```



```
}
```

new user data of the instance

Where we have added:

```
sudo aws s3 cp s3://${var.bucket-config-name}/ ./ --recursive

sudo chmod +x *.sh

./user_data01.sh

./user_data02.sh

./user_data03.sh
```

As before, we download the files in the current directory and then we change all of them (`*.sh`) to be executable.

After that, we simply run them.

Obviously, even in this case, we need to add the new variable `bucket-config-name` in the `variables.tf` in order to tell the instance where to download the script files.

Simple enough? Drop some feedback in the comments to let me know you thoughts and to help me improve the quality of these articles. Thanks!

More content at ***plainenglish.io***

Terraform AWS Ec2 S3 Programming