# Bootstrapping EC2 in Terraform

Pablo Perez Follow
Oct 25, 2018 · 2 min read

In CloudFormation you inject the bootstrapping logic for your AWS instance/autoscaling group by using the function *!Sub*.

However on Terraform, you have several options to inject the logic needed to bootstrap.

**Common options** to bootstrap EC2 in Terraform are:

**1.**- If the userdata logic is **small** you can just use **local variables.** We'll invoke the function base64encode to provide the property user_data_base64 with a base64-encoded representation.

```
provider "aws" {}
locals {
  instance-userdata = <<EOF
#!/bin/bash
export PATH=$PATH:/usr/local/bin
which pip >/dev/null
if [ $? -ne 0 ];
then
  echo 'PIP NOT PRESENT'
  if [ -n "$(which yum)" ];
  then
    yum install -y python-pip
  else
    apt-get -y update && apt-get -y install python-pip
  fi
else
  echo 'PIP ALREADY PRESENT'
fi
EOF
}

variable "amis" {
  type = "map"
  default = {
    "eu-west-1" = "ami-0c21ae4a3bd190229"
    "us-east-1" = "ami-0922553b7b0369273"
  }
}
variable "region" {
  type = "string"
  default = "us-east-1"
}

resource "aws_instance" "myinstance1" {
  ami           = "${lookup(var.amis, var.region)}"
  instance_type = "t2.micro"
  user_data_base64 = "${base64encode(local.instance-userdata)}"
}
```

If the userdata logic is large enough it might be worthy to use one of the following options:

2.- Using a **data source like template_file** to fetch the userdata content from a file.

```
provider "aws" {}
data "template_file" "myuserdata" {
  template = "${file("${path.cwd}/myuserdata.tpl")}"

}
variable "amis" {
  type = "map"
  default = {
    "eu-west-1" = "ami-0c21ae4a3bd190229"
    "us-east-1" = "ami-0922553b7b0369273"
  }
}
variable "region" {
  type = "string"
  default = "us-east-1"
}

resource "aws_instance" "myinstance1" {
  ami           = "${lookup(var.amis, var.region)}"
  instance_type = "t2.micro"
  user_data = "${data.template_file.myuserdata.template}"
}
```

**3.**- Using **datasource template_cloudinit_config**

Allows to use Mime Multi Part Archive so you can **concatenate** different sources to be used in the same userdata, as well as different types of sets of instructions like *cloud boothook* (content executed before the rest of the userdata and before other processes start), e.g. It's useful to configure the Docker daemon before it starts.

```
provider "aws" {}
data "template_file" "myuserdata" {
  template = "${file("${path.cwd}/myuserdata.tpl")}"

}
```

```
variable "amis" {
  type = "map"
  default = {
    "eu-west-1" = "ami-0c21ae4a3bd190229"
    "us-east-1" = "ami-0922553b7b0369273"
  }
}
variable "region" {
  type = "string"
  default = "us-east-1"
}

resource "aws_instance" "myinstance1" {
  ami           = "${lookup(var.amis, var.region)}"
  key_name = "ireland"
  instance_type = "t2.micro"
  user_data = "${data.template_cloudinit_config.config.rendered}"
}

data "template_cloudinit_config" "config" {
  base64_encode = true

part {
    content_type = "text/x-shellscript"
    content      = "${data.template_file.myuserdata.template}"
     }
  part {
    content_type = "text/x-shellscript"
    content  = "${file("${path.cwd}/installsysstat.sh")}"
  }
}
```

AWS Terraform Bootstrapping Ec2 Mime