

- Preconditions and postconditions

Validations

As of Terraform 0.13, you can add *validation blocks* to any input variable to perform checks that go beyond basic type constraints. For example, you can add a validation block to the `instance_type` variable to ensure not only that the value the user passes in is a string (which is enforced by the type constraint) but that the string has one of two allowed values from the AWS Free Tier:

```
variable "instance_type" {
  description = "The type of EC2 Instances to run (e.g.
t2.micro)"
  type        = string

  validation {
    condition      = contains(["t2.micro", "t3.micro"], var.instance_type)
    error_message = "Only free tier is allowed: t2.micro |
t3.micro."
  }
}
```

The way a validation block works is that the `condition` parameter should evaluate to `true` if the value is valid and `false` otherwise. The `error_message` parameter allows you to specify the message to show the user if they pass in an invalid value. For example, here's what happens if you try to set `instance_type` to `m4.large`, which is not in the AWS Free Tier:

```
$ terraform apply -var instance_type="m4.large"
Error: Invalid value for variable

  on main.tf line 17:
  1:   variable "instance_type" {
  |
  |     var.instance_type is "m4.large"

Only free tier is allowed: t2.micro | t3.micro.
```