```
  stores/mysql/terraform.tfstate"

    instance_type = "m4.large"
    min_size      = 2
    max_size      = 10
}
```

Module Locals

Using input variables to define your module's inputs is great, but what if you need a way to define a variable in your module to do some intermediary calculation, or just to keep your code DRY, but you don't want to expose that variable as a configurable input? For example, the load balancer in the `webserver-cluster` module in *modules/services/webserver-cluster/main.tf* listens on port 80, the default port for HTTP. This port number is currently copied and pasted in multiple places, including the load balancer listener:

```
resource "aws_lb_listener" "http" {
  load_balancer_arn = aws_lb.example.arn
  port              = 80
  protocol          = "HTTP"

  # By default, return a simple 404 page
  default_action {
    type = "fixed-response"

    fixed_response {
      content_type = "text/plain"
      message_body = "404: page not found"
      status_code  = 404
    }
  }
}
```

And the load balancer security group:

```
resource "aws_security_group" "alb" {
  name = "${var.cluster_name}-alb"

  ingress {
```

```
    from_port   = 80
    to_port     = 80
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  egress {
    from_port   = 0
    to_port     = 0
    protocol    = "-1"
    cidr_blocks = ["0.0.0.0/0"]
  }
}
```

The values in the security group, including the "all IPs" CIDR block `0.0.0.0/0`, the "any port" value of `0`, and the "any protocol" value of `"-1"` are also copied and pasted in several places throughout the module. Having these magical values hardcoded in multiple places makes the code more difficult to read and maintain. You could extract values into input variables, but then users of your module will be able to (accidentally) override these values, which you might not want. Instead of using input variables, you can define these as *local values* in a `locals` block:

```
locals {
  http_port    = 80
  any_port     = 0
  any_protocol = "-1"
  tcp_protocol = "tcp"
  all_ips      = ["0.0.0.0/0"]
}
```

Local values allow you to assign a name to any Terraform expression and to use that name throughout the module. These names are visible only within the module, so they will have no impact on other modules, and you can't override these values from outside of the module. To read the value of a local, you need to use a *local reference*, which uses the following syntax:

```
local.<NAME>
```

Use this syntax to update the `port` parameter of your load-balancer listener:

```
resource "aws_lb_listener" "http" {
  load_balancer_arn = aws_lb.example.arn
  port              = local.http_port
  protocol          = "HTTP"

  # By default, return a simple 404 page
  default_action {
    type = "fixed-response"

    fixed_response {
      content_type = "text/plain"
      message_body = "404: page not found"
      status_code  = 404
    }
  }
}
```

Similarly, update virtually all the parameters in the security groups in the module, including the load-balancer security group:

```
resource "aws_security_group" "alb" {
  name = "${var.cluster_name}-alb"

  ingress {
    from_port   = local.http_port
    to_port     = local.http_port
    protocol    = local.tcp_protocol
    cidr_blocks = local.all_ips
  }

  egress {
    from_port   = local.any_port
    to_port     = local.any_port
    protocol    = local.any_protocol
    cidr_blocks = local.all_ips
  }
}
```

Locals make your code easier to read and maintain, so use them often.