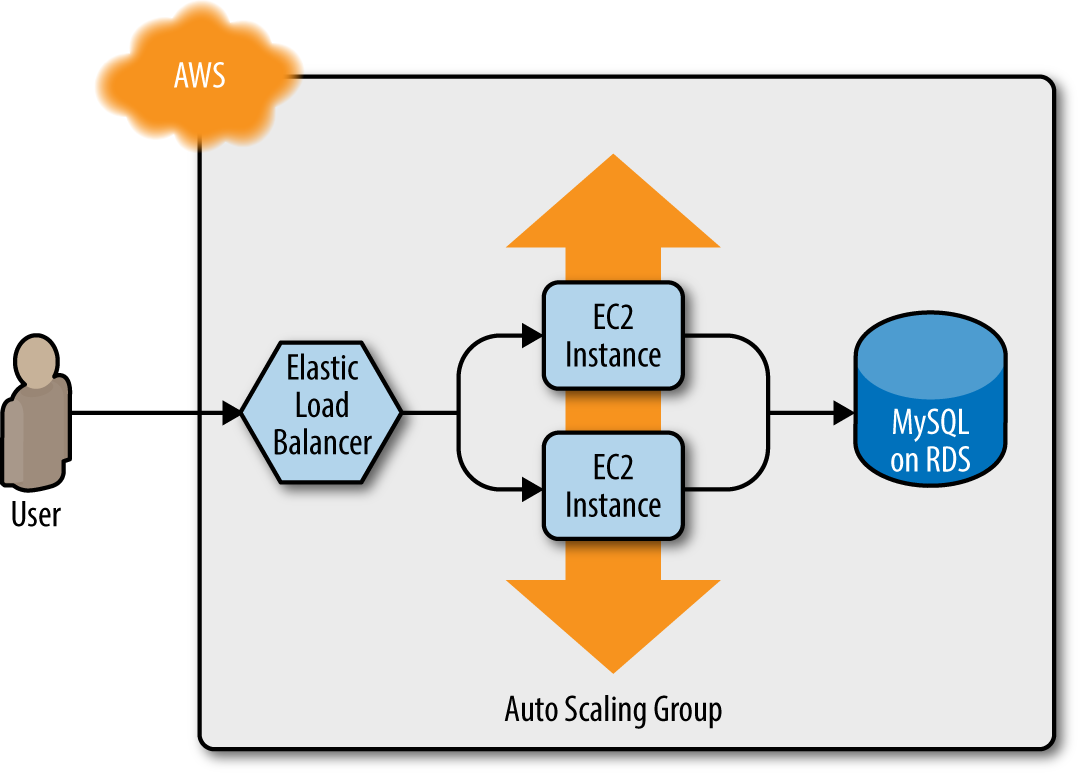
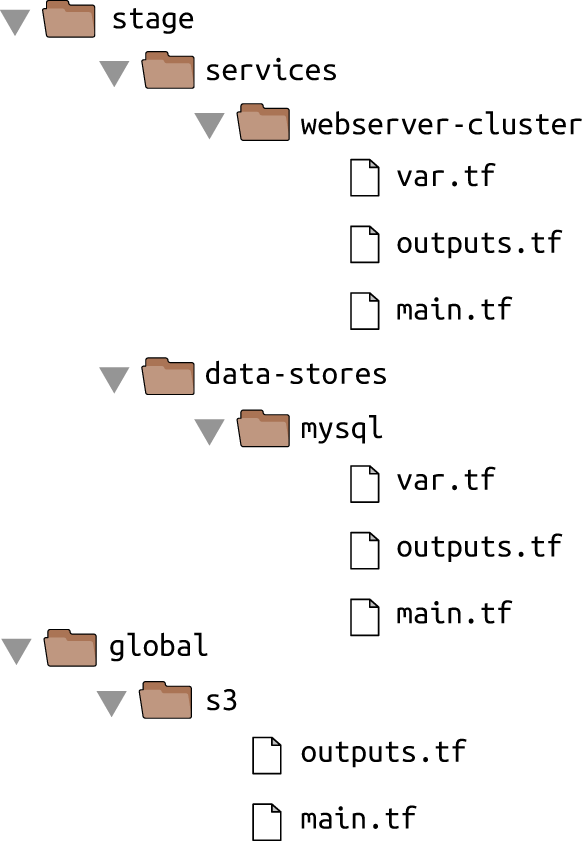
use data sources to fetch read-only information from AWS,such as the aws\_subnet\_ids data source, which returns a list of subnets in your VPC. There is another data sourcethat is particularly useful when working with state: terraform\_remote\_state.You can use this data source to fetch the Terraform state file stored by another set of Terraform configurations in acompletely read-only manner.

below is scenario





Passwords for DB can passed through AWS secrets manager or other options by cloud provider

**password = data.aws\_secretsmanager\_secret\_version.db\_password.secret\_string }**

**data "aws\_secretsmanager\_secret\_version" "db\_password" {**

**secret\_id = "mysql-master-password-stage"**

**}**

Passwords can be give via environment variable

**variable using the environment variable TF\_VAR\_<envvar>**

leaving no default value in variable secion.

variable "db\_password" {

description = "The password for the database"

type = string

}

like:

**$ export TF\_VAR\_db\_password="(YOUR\_DB\_PASSWORD)"**

**$ terraform apply**

## Secrets Are Always Stored in Terraform State

Reading secrets from a secrets store or environment variables is a good practice to ensure secrets aren’t stored inplain text in your code, but just a reminder: no matter how you read in the secret, if you pass it as an argument to aTerraform resource, such as aws\_db\_instance, that secret will be stored in the Terraform state file, in plain text.

This is a known weakness of Terraform, with no effective solutions available, so be extra paranoid with how you storeyour state files (e.g., always enable encryption) and who can access those state files (e.g., use IAM permissions tolock down access to your S3 bucket)!

=====================

As WEBSERS AS TO CONNECT TO DB ., it need ports and endpoint details

so we need to output the data form Db

**output "address" {**

**value = aws\_db\_instance.example.address**

**description = "Connect to the database at this endpoint"**

**}**

**output "port" {**

**value = aws\_db\_instance.example.port**

**description = "The port the database is listening on"**

**}**

These outputs are now also stored in the Terraform state for the database, which is in your S3/GCP bucket at the path /data-stores/mysql/terraform.tfstate. You can get the web server cluster code to read the data from this statefile by adding the terraform\_remote\_state datasource in stage/services/webserver-cluster/main.tf:

**data "terraform\_remote\_state" "db" {**

**backend = "s3"**

**config = {**

**bucket = "(YOUR\_BUCKET\_NAME)"**

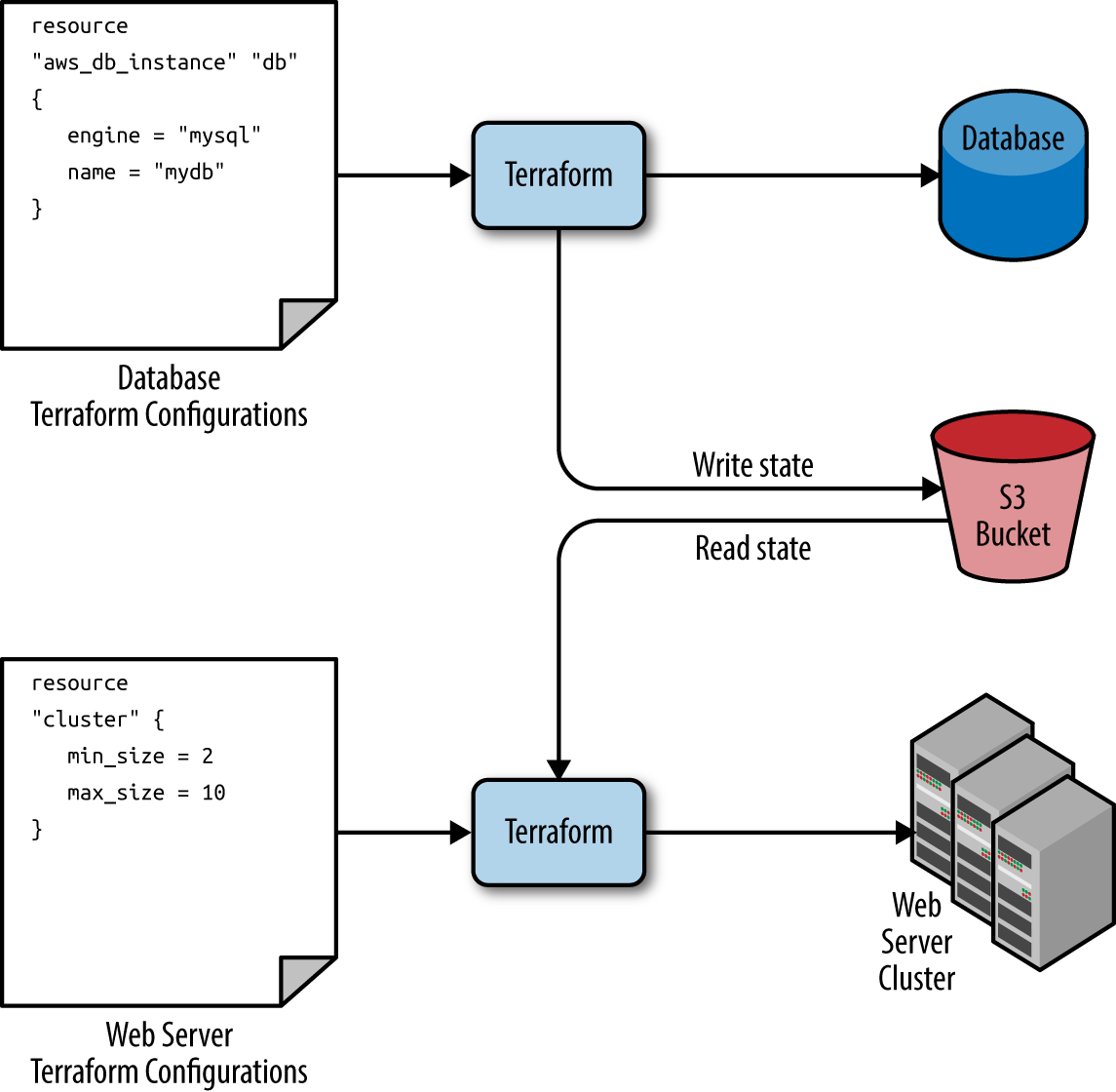
**key = "stage/data-stores/mysql/terraform.tfstate"**

**region = "us-east-2"**

**}**

**}**

This **terraform\_remote\_state** data source configures the web server cluster code to read the state file from the same S3 bucket and folder where the database stores its state, as shown below. **the data returned by terraform\_remote\_state isread-only.**



All of the database’s output variables arestored in the state file and you can read them from the **terraform\_remote\_state** data source using an attributereference of the form:

**data.terraform\_remote\_state.<NAME>.outputs.<ATTRIBUTE>**

For example, here is how you can update the User Data of the web server cluster Instances to pull the database addressand port out of the **terraform\_remote\_state** data source and expose that information in the HTTP response:

user\_data = <<EOF#!/bin/bashecho "Hello, World" >> index.htmlecho "${data.terraform\_remote\_state.db.outputs.address}" >> index.htmlecho "${data.terraform\_remote\_state.db.outputs.port}" >> index.htmlnohup busybox httpd -f -p ${var.server\_port} &EOF

As the User Data script is growing longer, defining it inline is becoming messier and messier.In general, embedding one programming language (Bash) inside another (Terraform) makes it more difficult to maintain each one,so let’s pause here for a moment to externalize the Bash script. To do that, youcan use the file built-in function and the **template\_file** data source. Let’s talk about these one at a time.

The **template\_file** data source has two arguments: **template**, which is a string to render, and **vars**, which is a mapof variables to make available while rendering. It has one output attribute called rendered, which is the result ofrendering template. To see this in action, add the following template\_file data source tostage/services/webserver-cluster/main.tf:

**file("user-data.sh") 🡪 it is a function 🡪 gets string of file path**

**data "template\_file" "user\_data" {**

**template = file("user-data.sh")**

**vars = {**

**server\_port = var.server\_port**

**db\_address = data.terraform\_remote\_state.db.outputs.address**

**db\_port = data.terraform\_remote\_state.db.outputs.port**

**}**

**}**

then update script with variables accordingly

**address: ${db\_address}**

**DB port: ${db\_port}**