Lab: Outputs

Terraform generates a significant amount of metadata that is too tedious to sort through with `terraform show`. Even with just one instance deployed, we wouldn't want to scan 38 lines of metadata every time. Outputs allow us to query for specific values rather than parse metadata in `terraform show`.

- Task 1: Create output values in the configuration file
- Task 2: Use the output command to find specific values
- Task 3: Suppress outputs of sensitive values in the CLI

Task 1: Create output values in the configuration file

Outputs allow customization of Terraform's output during a Terraform apply. Outputs define useful values that will be highlighted to the user when Terraform is executed. Examples of values commonly retrieved using outputs include IP addresses, usernames, and generated keys.

Create a new output value named "public_ip" to output the instance's public_ip attributes. In the `outputs.tf` file, add the following:

```
output "public_ip" {
  description = "This is the public IP of my web server"
  value = aws_instance.web_server.public_ip
}
```

Task 1.1: Run a Terraform Apply to view the outputs

After adding the new output blocks above, go ahead and run a `terraform apply -auto-approve` to see the new output values. Since we didn't change any resources, there is no risk of changes to our environment. You can see the new output value. Notice that you only see the name of the output and the value but you don't see the description inthe output.

```
You can apply this plan to save these new output values to the Terraform state, without

Apply complete! Resources: 0 added, 0 changed, 0 destroyed.

Outputs:

hello-world = "Hello World"
public_ip = "44.200.207.151"
```

```
public_url = "https://10.0.101.169:8080/index.html"
vpc_id = "vpc-0dcd2b053088ea107"
vpc_information = "Your demo_environment VPC has an ID of vpc-0dcd2b053088ea107"
```

Task 2: Use the `terraform output` command to find specific values

Step 2.1 Try the terraform output command with no specifications

```
terraform output
```

Step 2.2 Query specifically for the public_dns attributes

```
terraform output public_ip
```

Step 2.3 Wrap an output query to ping the DNS record

```
ping $(terraform output -raw public_dns)
```

Note that you will probably not recieve a response since we don't have ICMP open on our security group

Task 3: Suppress outputs of sensitive values in the CLI

As you'll find with many aspects of Terraform, you will sometimes be working with sensitive data. Whether it's a username and password, account numbers, or certicates, it's very common that you'll want to obfuscate these from the CLI output. Fortunately, Terraform provides us with the `sensitive` argument to use in the output block. This allows you to mark the value as sensitive (hence the name) and prevent the value from showing in the CLI output. It does not, however, prevent the value from being listed in the state file or anything like that.

In the `outputs.tf` file, add the new output block as shown below. Since a resource arn often includes the AWS account number, it might be a value we don't want to show in the CLI console, so let's obfuscate it to protect our account.

```
output "ec2_instance_arn" {
  value = aws_instance.web_server.arn
  sensitive = true
}
```

Task 3.1: Run a Terraform Apply to view the suppressed value

After adding the output blocks, run a `terraform apply -auto-approve` to see the new output value (or NOT see the new output value). Since we didn't change any resources, there is no risk of changes to our environment.

```
Changes to Outputs:
    # (1 unchanged attribute hidden)

You can apply this plan to save these new output values to the Terraform state, without

Apply complete! Resources: 0 added, 0 changed, 0 destroyed.

Outputs:

ec2_instance_arn = <sensitive>
hello-world = "Hello World"
public_ip = "44.200.207.151"
public_url = "https://10.0.101.169:8080/index.html"
vpc_id = "vpc-0dcd2b053088ea107"
vpc_information = "Your demo_environment VPC has an ID of vpc-0dcd2b053088ea107"
```