Output Data from Terraform

Terraform output values allow you to export structured data about your resources. You can use this data to configure other parts of your infrastructure with automation tools, or as a data source for another Terraform workspace. Outputs are also necessary to share data from a child module to your root module.

In this tutorial, you will use Terraform to deploy a web application on AWS. The supporting infrastructure includes a VPC, load balancer, EC2 instances, and a database. Then you will use outputs to get information about the resources you have deployed. Next, you will use the `sensitive` flag to reduce the risk of inadvertently disclosing the database administrator username and password. Finally, you will use the JSON format to output data in a machine-readable format.

Prerequisites

In order to follow this tutorial, you will need the following:

- The Terraform CLI, version 0.14 or later.
- AWS Credentials configured for use with Terraform.
- The git CLI.

Note: Some of the infrastructure in this tutorial may not qualify for the AWS free tier. Destroy the infrastructure at the end of the tutorial to avoid unnecessary charges. We are not responsible for any charges that you incur.

Create infrastructure

Clone the Learn Terraform Outputs GitHub repository for this tutorial.

```
$ git clone https://github.com/hashicorp/learn-terraform-outputs.git
```

Change to the repository directory.

```
$ cd learn-terraform-outputs
```

The configuration in `main.tf` defines a web application, including a VPC, load balancer, EC2 instances, and a database.

Initialize this configuration.

```
$ terraform init
```

Now apply the configuration.

```
$ terraform apply
```

Respond to the confirmation prompt with a `yes` to create the example infrastructure.

Output VPC and load balancer information

Output declarations can appear anywhere in your Terraform configuration files. However, we recommend putting them into a separate file called `outputs.tf` to make it easier for users to understand your configuration and what outputs to expect from it.

Add a block to `outputs.tf` to show the ID of the VPC.

```
output "vpc_id" {
  description = "ID of project VPC"
  value       = module.vpc.vpc_id
}
```

While the `description` argument is optional, you should include it in all output declarations to document the intent and content of the output.

You can use the result of any Terraform expression as the value of an output. Use expressions to declare outputs for the load balancer URL and number of web servers provisioned by this configuration by adding the following to `outputs.tf`.

```
output "lb_url" {
  description = "URL of load balancer"
  value       = "http://${module.elb_http.this_elb_dns_name}/"
}

output "web_server_count" {
  description = "Number of web servers provisioned"
  value       = length(module.ec2_instances.instance_ids)
}
```

The `lb_url` output uses string interpolation to create a URL from the load balancer's domain name. The `web_server_count` output uses the `length()` function to calculate the number of instances attached to the load balancer.

Terraform stores output values in its state file. In order to see these outputs, you need to update the state by applying this new configuration, even though the infrastructure will not change. Respond to the confirmation prompt with a `yes`.

```
$ terraform apply
random_string.lb_id: Refreshing state... [id=5YI]
module.vpc.aws_vpc.this[0]: Refreshing state... [id=vpc-004c2d1ba7394b3d6]

## ...

Plan: 0 to add, 0 to change, 0 to destroy.

Changes to Outputs:
  + lb_url           = "http://lb-5YI-project-alpha-dev-2144336064.us-east-
1.elb.amazonaws.com/"
  + vpc_id           = "vpc-004c2d1ba7394b3d6"
  + web_server_count = 4

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes


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

Outputs:

lb_url = "http://lb-5YI-project-alpha-dev-2144336064.us-east-1.elb.amazonaws.com/"
vpc_id = "vpc-004c2d1ba7394b3d6"
web_server_count = 4
```

Query outputs

Now that Terraform has loaded the outputs into your project's state, use the `terraform output` command to query all of them.

```
$ terraform output
lb_url = "http://lb-5YI-project-alpha-dev-2144336064.us-east-1.elb.amazonaws.com/"
vpc_id = "vpc-004c2d1ba7394b3d6"
web_server_count = 4
```

Next, query an individual output by name.

```
$ terraform output lb_url
"http://lb-5YI-project-alpha-dev-2144336064.us-east-1.elb.amazonaws.com/"
```

Starting with version 0.14, Terraform wraps string outputs in quotes by default. You can use the `-raw` flag when querying a specified output for machine-readable format.

```
$ terraform output -raw lb_url
http://lb-5YI-project-alpha-dev-2144336064.us-east-1.elb.amazonaws.com/
```

Use the `lb_url` output value with the `-raw` flag to cURL the load balancer and verify the response.

```
$ curl $(terraform output -raw lb_url)
<html><body><div>Hello, world!</div></body></html>
```

Redact sensitive outputs

You can designate Terraform outputs as *sensitive*. Terraform will redact the values of sensitive outputs to avoid accidentally printing them out to the console. Use sensitive outputs to share sensitive data from your configuration with other Terraform modules, automation tools, or Terraform Cloud workspaces.

Terraform will redact sensitive outputs when planning, applying, or destroying your configuration, or when you query all of your outputs. Terraform will **not** redact sensitive outputs in other cases, such as when you query a specific output by name, query all of your outputs in JSON format, or when you use outputs from a child module in your root module.

Add the following sensitive output blocks to your `outputs.tf` file.

```
output "db_username" {
  description = "Database administrator username"
  value       = aws_db_instance.database.username
  sensitive   = true
}

output "db_password" {
  description = "Database administrator password"
  value       = aws_db_instance.database.password
  sensitive   = true
}
```

Apply this change to add these outputs to your state file, and respond to the confirmation prompt with `yes`.

```
$ terraform apply
random_string.lb_id: Refreshing state... [id=5YI]
module.vpc.aws_vpc.this[0]: Refreshing state... [id=vpc-004c2d1ba7394b3d6]

## ...

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

Outputs:

db_password = <sensitive>
db_username = <sensitive>
lb_url = "http://lb-5YI-project-alpha-dev-2144336064.us-east-1.elb.amazonaws.com/"
vpc_id = "vpc-004c2d1ba7394b3d6"
web_server_count = 4
```

Notice that Terraform redacts the values of the outputs marked as sensitive.

Use `terraform output` to query the database password by name, and notice that Terraform will not redact the value when you specify the output by name.

```
$ terraform output db_password
"notasecurepassword"
```

Output values are stored as plain text in your state file — including those flagged as sensitive. Use the `grep` command to see the values of the sensitive outputs in your state file.

```
$ grep --after-context=10 outputs terraform.tfstate
  "outputs": {
    "db_password": {
      "value": "notasecurepassword",
      "type": "string",
      "sensitive": true
    },
    "db_username": {
      "value": "admin",
      "type": "string",
      "sensitive": true
    },
```

Tip: If you are using an operating system without the `grep` command, open the `terraform.tfstate` file in your text editor and search for `outputs` to see the relevant lines.

The `sensitive` argument for outputs can help avoid inadvertent exposure of those values. However, you must still keep your Terraform state secure to avoid exposing these values.

Generate machine-readable output

The Terraform CLI output is designed to be parsed by humans. To get machine-readable format for automation, use the `-json` flag for JSON-formatted output.

```
$ terraform output -json
{
  "db_password": {
    "sensitive": true,
    "type": "string",
    "value": "notasecurepassword"
  },
  "db_username": {
    "sensitive": true,
    "type": "string",
    "value": "admin"
  },
  "lb_url": {
    "sensitive": false,
    "type": "string",
    "value": "http://lb-5YI-project-alpha-dev-2144336064.us-east-
1.elb.amazonaws.com/"
  },
  "vpc_id": {
    "sensitive": false,
    "type": "string",
    "value": "vpc-004c2d1ba7394b3d6"
  },
  "web_server_count": {
    "sensitive": false,
    "type": "number",
    "value": 4
  }
}
```

Terraform does not redact sensitive output values with the `-json` option, because it assumes that an automation tool will use the output.

Clean up your infrastructure

In this tutorial you used Terraform outputs to query data about your infrastructure. Terraform outputs allow you to share data between Terraform workspaces, and with other tools and automation. Outputs are also the only way to share data from a child module to your configuration's root module.

Before moving on, destroy the infrastructure you created in this tutorial.

```
$ terraform destroy
```

Be sure to respond to the confirmation prompt with `yes`.

Next steps

Now that you know how to use Terraform outputs, check out the following resources for more information.

- Read the Terraform Outputs documentation.
- Manage sensitive data in state.

- Use and create Terraform modules.
- Connect Terraform Cloud Workspaces with run triggers, and use outputs from one workspace to configure another workspace.