Lab: Terraform Configuration Block

Terraform relies on plugins called “providers” to interact with remote systems and expand functionality. Terraform configurations must declare which providers they require so that Terraform can install and use them. This is performed within a Terraform configuration block.

- Task 1: Check Terraform version
- Task 2: Require specific versions of Terraform

Template:

```
terraform {
  # Block body
  <ARGUMENT> = <VALUE>
}
```

Example:

```
 terraform {
   required_version = ">= 1.0.0"
 }
```

Task 1: Check Terraform version

Run the following command to check the Terraform version:

```
terraform -version
```

You should see:

```
Terraform v1.0.8
```

Task 2: Require specific versions of Terraform

Create a file titled `terraform.tf` to define a minimum version of Terraform to be used.

`terraform.tf`

```
 terraform {
   required_version = ">= 1.0.0"
 }
```


```
terraform int
```

```
Terraform v1.0.8
```

This informs Terraform that it must be at least of version `1.0.0` to run the code. If Terraform is an earlier version it will throw an error.

Update the `required_version` line to `"=1.0.0"` and run `terraform init`.

What happened when you changed the version requirement?

```
Error: Unsupported Terraform Core version

  on terraform.tf line 2, in terraform:
   2:     required_version = "= 1.0.0"

This configuration does not support Terraform version 1.0.8. To proceed,
   either choose another supported Terraform
version or update this version constraint. Version constraints are
   normally set for good reason, so updating the
constraint may lead to other errors or unexpected behavior.
```

At this point you have now stated that Terraform can only run this code if its own version is equal to `1.0.0`.

> Note: Make sure the required version is set back to `">=1.0.0"` and you > have run `terraform init` again before proceeding.

Reference

Terraform Settings