Lab: Terraform Provider Installation

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: Install Terraform AWS Provider
- Task 2: Verify Terraform and AWS Provider version

Task 1: Install Terraform AWS Provider

Terraform Providers are plugins that implement resource types for particular clouds, platforms and generally speaking any remote system with an API. Terraform configurations must declare which providers they require, so that Terraform can install and use them. Popular Terraform Providers include: AWS, Azure, Google Cloud, VMware, Kubernetes and Oracle.

In the next step we will install the Terraform AWS provider, and set the provider version in a way that is very similar to how you did for Terraform. To begin you need to let Terraform know to use the provider through a `required_providers` block in the `terraform.tf` file as seen below.

`terraform.tf`

```
terraform {
  required_version = ">= 1.0.0"
  required_providers {
    aws = {
      source = "hashicorp/aws"
      version = "~> 3.0"
    }
  }
}
```

> Note: You can always find the latest version of a provider on its > registry page at https://registry.terraform.io.

Now that you have told Terraform to use this new provider you will have to run the `init` command. This will cause Terraform to notice the configuration change and download the provider.

```
terraform init
```

```
Initializing the backend...
```


```
Initializing provider plugins...
- Reusing previous version of hashicorp/random from the dependency lock
    file
- Reusing previous version of hashicorp/aws from the dependency lock file
- Using previously-installed hashicorp/aws v3.62.0
- Using previously-installed hashicorp/random v3.1.0

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to
    see
any changes that are required for your infrastructure. All Terraform
    commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget,
    other
commands will detect it and remind you to do so if necessary.
```

By default Terraform will always pull the latest provider if no version is set. However setting a version provides a way to ensure your Terraform code remains working in the event a newer version introduces a change that would not work with your existing code. To have more strict controls over the version you may want to require a specific version (e.g. `required_version = "= 1.0.0"`) or use the ~>operator to only allow the right-most version number to increment.

Task 2: Verify Terraform and AWS Provider version

To check the terraform version and provider version installed via `terraform init` run the `terraform version` command.

```
terraform version
```

```
Terraform v1.0.8
on linux_amd64
+ provider registry.terraform.io/hashicorp/aws v3.62.0
+ provider registry.terraform.io/hashicorp/random v3.1.0
```