Terraform Provider Versioning

Terraform Provider Versioning

- The provide add as a middle or interface between terraform and the service provider
- The provider has multiple versioning. For instance, Windows have a lot of versions such as Windows 10, 7, XP, server, and so on.
- Provider plugins are released separately from Terraform itself.
- They have a different set of version numbers.

Explicitly Setting Provider Version

- During terraform init, if the version argument is not specified, the most recent provider will be downloaded during initialization.
- For production use, you should constrain the acceptable provider versions via configuration, to ensure that new versions with breaking changes will not be automatically installed.
- AWS latter version is `hashicorp/aws: version = "~> 3.10.0"`. Type terraform init to see the version.
- This is because the new version must be tested in the test environment before deploying to production.

Arguments for Specifying the provider

- There are multiple ways of specifying the version of a provider.
  - `>=1.0`: Download plugins that are greater than or equal to one

    `<=1.0`: Download plugins that are less than or equal to one

    `~>2.0`: Download plugins in the 2.X range

    `>=2.10,<=2.30`: Download any plugins version between 2.10 and 2.30

```
version    = "2.7"
version    = ">= 2.8"
version    = "<= 2.8"
version    = ">=2.10,<=2.30"
```

This is for Terraform >=0.12 definition

```
provider "aws" {
    region = "us-east-1"
    version = ">=2.8,<=2.30"
}
resource "aws_s3_bucket" "terraform_s3" {
    bucket = "terraform-bucket-232"
    acl = "private"
}
```

This is for Terraform >0.12 definition

```
terraform {
  required_version = ">= 1.0.0"
  required_providers {
    azurerm = {
      source = "hashicorp/azurerm"
      version = ">= 2.0"
    }
  }
}
```

Create a simple terraform block and play with required_version

- `required_version` focuses on underlying Terraform CLI installed on your desktop
- If the running version of Terraform on your local desktop doesn't match the constraints specified in your terraform block, Terraform will produce an error and exit without taking any further actions.
- By changing the versions try to `terraform init` and observe whats happening

```
# Play with Terraform CLI Version (We installed 1.0.0 version)
  required_version = "~> 0.14.3" - Will fail (we can only have [0.14.0-0.14.9])
  required_version = "~> 0.14"   - Will fail
  required_version = "= 0.14.4"  - Will fail
  required_version = ">= 0.13"   - will pass
  required_version = "= 1.0.0"   - will pass
  required_version = "1.0.0"     - will pass
  required_version = ">= 1.0.0"   - will pass
  required_version = "> 1.0.0"   - will fail

# Terraform Block
terraform {
  required_version = ">= 1.0.0"
}

# To view my Terraform CLI Version installed on my desktop
terraform version

# Initialize Terraform
terraform init
```

Add Provider and play with Provider version

- `required_providers` block specifies all of the providers required by the current module, mapping each local provider name to a source address and a version constraint.

```
# Play with Provider Version
      version = "~> 2.0"
      version = ">= 2.0.0, < 2.60.0"
      version = ">= 2.0.0, <= 2.64.0"
```

```
terraform {
  required_version = ">= 1.0.0"
  required_providers {
    azurerm = {
      source = "hashicorp/azurerm"
      version = ">= 2.0"
    }
  }
}
```