HashiCorp Certified: Terraform Associate

Hands-On Labs

Lab: Fetch, Version and Upgrade Terraform Providers

Terraform relies on plugins called “providers” to interact with remote systems and expand functionality. Terraform providers can be versioned inside a Terraform configuration block. To prevent external changes from causing unintentional changes, it’s highly recommended that providers specify versions which they are tied to. Depending on the level of acceptable risk and management effort to be tracking version updates, that can either be hard locked to a particular version, or use looser mechanisms such as less than next major rev, or using tilde to track through bug fix versions.

- Task 1: Check Terraform and Provider version
- Task 2: Require specific versions of Terraform providers
- Task 3: Upgrade provider versions

Task 1: Check Terraform version

Run the following command to check the Terraform version:

```
terraform -version
```

You should see:

```
Terraform v1.0.10
```

Task 2: Require specific versions of Terraform

Update the `terraform.tf` to define a minimum version of the Terraform AWS provider and Terraform Random provider to be used.

> Note: Terraform has multi-provider support from a single set of configuration files. This is extremely beneficial and we can control the version parameters for each individual provider.

`terraform.tf`

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

```
    random = {
    source  = "hashicorp/random"
    version = "3.1.0"
  }
  }
}
```

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

You need to let Terraform know to use the provider through a `required_providers` block. 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 change and download the correct version if it was not already downloaded.

```
terraform init
```

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. `version = "= 3.0.0"`) or use the ~>operator to only allow the right-most version number to increment.

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

You can modify the version line of the AWS provider to be as specific or general as desired.

Update the `version` line within your `terraform.tf` file to try these different versioning techniques.

```
      version = "~> 3.0"
      version = ">= 3.0.0, < 3.1.0"
      version = ">= 3.0.0, <= 3.1.0"
      version = "~> 2.0"
      version = "~> 3.0"
```

Notice that for certain combinations of the version constraint you may encounter an error stating the version of the provider installed does not match the configured version.

```
terraform init
```

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

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

Error: Failed to query available provider packages
Could not retrieve the list of available versions for provider hashicorp/
    aws: locked provider registry.terraform.io/hashicorp/aws 3.62.0 does
    not match configured version constraint >= 3.0.0, < 3.1.0; must use
    terraform init -upgrade to allow selection of new versions
```

In these situations you have to intentionally install a version that matches the constraint by issuing a `terraform init -upgrade` command.

Task 3: How to upgrade provider versions

Terraform configurations must declare which providers they require so that Terraform can install and use them. Once installed, Terraform will record the provider versions in a dependency lock file to ensure that others using this configuration will utilize the same Terraform and provider versions. This file by default is saved as a `.terraform.lock.hcl` file, and its contents look as follows:

```
# This file is maintained automatically by "terraform init".
# Manual edits may be lost in future updates.

provider "registry.terraform.io/hashicorp/aws" {
  version     = "3.61.0"
  constraints = "~> 3.0"
  hashes = [
    "h1:fpZ14qQnn+uEOO2ZOlBFHgty48Ol8IOwd+ewxZ4z3zc=",
    "zh:0483ca802ddb0ae4f73144b4357ba72242c6e2641aeb460b1aa9a6f6965464b0",
    "zh:274712214ebeb0c1269cbc468e5705bb5741dc45b05c05e9793ca97f22a1baa1",
    "zh:3c6bd97a2ca809469ae38f6893348386c476cb3065b120b785353c1507401adf",
    "zh:53dd41a9aed9860adbbeeb71a23e4f8195c656fd15a02c90fa2d302a5f577d8c",
    "zh:65c639c547b97bc880fd83e65511c0f4bbfc91b63cada3b8c0d5776444221700",
    "zh:a2769e19137ff480c1dd3e4f248e832df90fb6930a22c66264d9793895161714",
    "zh:a5897a99332cc0071e46a71359b86a8e53ab09c1453e94cd7cf45a0b577ff590",
    "zh:bdc2353642d16d8e2437a9015cd4216a1772be9736645cc17d1a197480e2b5b7",
    "zh:cbeace1deae938f6c0aca3734e6088f3633ca09611aff701c15cb6d42f2b918a",
    "zh:d33ca19012aabd98cc03fdeccd0bd5ce56e28f61a1dfbb2eea88e89487de7fb3",
    "zh:d548b29a864b0687e85e8a993f208e25e3ecc40fcc5b671e1985754b32fdd658",
  ]
}

provider "registry.terraform.io/hashicorp/random" {
  version     = "3.0.0"
```

```
  constraints = "3.0.0"
  hashes = [
    "h1:yhHJpb4IfQQfuio7qjUXuUFTU/s+ensuEpm23A+VWz0=",
    "zh:0fcb00ff8b87dcac1b0ee10831e47e0203a6c46aafd76cb140ba2bab81f02c6b",
    "zh:123c984c0e04bad910c421028d18aa2ca4af25a153264aef747521f4e7c36a17",
    "zh:287443bc6fd7fa9a4341dec235589293cbcc6e467a042ae225fd5d161e4e68dc",
    "zh:2c1be5596dd3cca4859466885eaedf0345c8e7628503872610629e275d71b0d2",
    "zh:684a2ef6f415287944a3d966c4c8cee82c20e393e096e2f7cdcb4b2528407f6b",
    "zh:7625ccbc6ff17c2d5360ff2af7f9261c3f213765642dcd84e84ae02a3768fd51",
    "zh:9a60811ab9e6a5bfa6352fbb943bb530acb6198282a49373283a8fa3aa2b43fc",
    "zh:c73e0eaeea6c65b1cf5098b101d51a2789b054201ce7986a6d206a9e2dacaefd",
    "zh:e8f9ed41ac83dbe407de9f0206ef1148204a0d51ba240318af801ffb3ee5f578",
    "zh:fbdd0684e62563d3ac33425b0ac9439d543a3942465f4b26582bcfabcb149515",
  ]
}
```

By default, if a `.terraform.lock.hcl` file exists within a Terraform working directory, the provider versions specified in this file will be used. This lock file helps to ensure that runs accross teams will be consistent. As new versions of Terraform providers are released it is often beneficial to upgrade provider versions to take advantage of these updates. This can be accomplisehd by updating our configuration definiton to the desired provider version and running an upgrade.

In this case we are going to update the Terraform random provider from `3.0.0` to `3.1.0`.

> Note: If the provider is already at version `3.1.0` then youo can downgrade it to `3.0.0` following similar steps.

```
terraform {
  required_version = ">= 1.0.0"
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 3.0"
    }
    random = {
    source  = "hashicorp/random"
    version = "3.1.0"
  }
  }
}
```

```
terraform init -upgrade
```

> Note: You should never directly modify the lock file.

Once complete you can open the `.terraform.lock.hcl` to see the provider updates, as well as run

a `terraform version` to see that the new provider version has been installed and ready for use.

```
# This file is maintained automatically by "terraform init".
# Manual edits may be lost in future updates.

provider "registry.terraform.io/hashicorp/aws" {
  version     = "3.61.0"
  constraints = "~> 3.0"
  hashes = [
    "h1:fpZ14qQnn+uEOO2ZOlBFHgty48Ol8IOwd+ewxZ4z3zc=",
    "zh:0483ca802ddb0ae4f73144b4357ba72242c6e2641aeb460b1aa9a6f6965464b0",
    "zh:274712214ebeb0c1269cbc468e5705bb5741dc45b05c05e9793ca97f22a1baa1",
    "zh:3c6bd97a2ca809469ae38f6893348386c476cb3065b120b785353c1507401adf",
    "zh:53dd41a9aed9860adbbeeb71a23e4f8195c656fd15a02c90fa2d302a5f577d8c",
    "zh:65c639c547b97bc880fd83e65511c0f4bbfc91b63cada3b8c0d5776444221700",
    "zh:a2769e19137ff480c1dd3e4f248e832df90fb6930a22c66264d9793895161714",
    "zh:a5897a99332cc0071e46a71359b86a8e53ab09c1453e94cd7cf45a0b577ff590",
    "zh:bdc2353642d16d8e2437a9015cd4216a1772be9736645cc17d1a197480e2b5b7",
    "zh:cbeace1deae938f6c0aca3734e6088f3633ca09611aff701c15cb6d42f2b918a",
    "zh:d33ca19012aabd98cc03fdeccd0bd5ce56e28f61a1dfbb2eea88e89487de7fb3",
    "zh:d548b29a864b0687e85e8a993f208e25e3ecc40fcc5b671e1985754b32fdd658",
  ]
}

provider "registry.terraform.io/hashicorp/random" {
  version     = "3.1.0"
  constraints = "3.1.0"
  hashes = [
    "h1:rKYu5ZUbXwrLG1w81k7H3nce/Ys6yAxXhWcbtk36HjY=",
    "zh:2bbb3339f0643b5daa07480ef4397bd23a79963cc364cdfbb4e86354cb7725bc",
    "zh:3cd456047805bf639fbf2c761b1848880ea703a054f76db51852008b11008626",
    "zh:4f251b0eda5bb5e3dc26ea4400dba200018213654b69b4a5f96abee815b4f5ff",
    "zh:7011332745ea061e517fe1319bd6c75054a314155cb2c1199a5b01fe1889a7e2",
    "zh:738ed82858317ccc246691c8b85995bc125ac3b4143043219bd0437adc56c992",
    "zh:7dbe52fac7bb21227acd7529b487511c91f4107db9cc4414f50d04ffc3cab427",
    "zh:a3a9251fb15f93e4cfc1789800fc2d7414bbc18944ad4c5c98f466e6477c42bc",
    "zh:a543ec1a3a8c20635cf374110bd2f87c07374cf2c50617eee2c669b3ceeeaa9f",
    "zh:d9ab41d556a48bd7059f0810cf020500635bfc696c9fc3adab5ea8915c1d886b",
    "zh:d9e13427a7d011dbd654e591b0337e6074eef8c3b9bb11b2e39eaaf257044fd7",
    "zh:f7605bd1437752114baf601bdf6931debe6dc6bfe3006eb7e9bb9080931dca8a",
  ]
```

```
terraform version
```

```
Terraform v1.0.10

+ provider registry.terraform.io/hashicorp/aws v3.61.0
+ provider registry.terraform.io/hashicorp/random v3.1.0
```

Reference

Terraform Versioning Terraform Dependency Lock File