Lab: Terraform Backend Configuration

As we have seen the Terraform backend is configured for a given working directory within a terraform configuration file. A configuration can only provide one backend block per working directory. You do not need to specify every required argument in the backend configuration. Omitting certain arguments may be desirable if some arguments are provided automatically by an automation script running Terraform. When some or all of the arguments are omitted, we call this a partial configuration.

- Task 1: Terraform backend block
- Task 2: Partial backend configuration via a file
- Task 3: Partial backend configuration via command line
- Task 4: Declare backend configuration via interactive prompt
- Task 5: Specifying multiple partial backend configurations
- Task 6: Backend configuration from multiple locations
- Task 7: Change state backend configuration back to default

Task 1: Terraform backend block

By default there is no `backend` configuration block within Terraform configuration. Because no `backend` is included in the configuration Terraform will use it's default backend - `local` This is why we see the `terraform.tfstate` file in our working directory. If we want to be explicit about which backend Terraform should use it can be specified within the `backend` of the teraform configuration block as shown in previous labs.

Update the `terraform.tf` file to specify a local backend configuration with a path to the `terraform.tfstate` file.

```
terraform {
  backend "local" {
    path = "terraform.tfstate"
  }
}
```

```
terraform init
terraform plan
terraform apply
```


Task 2: Partial backend configuration via a file

Terraform backends do not support interpolations within its configuration block. This means it is not allowed to have variables as values when defining the backend configuration. Terraform does support the ability to specify partial backend configuration that can be merged into the backend block. This partial configuration can be provided via a file and then specified during the `terraform init`.

To specify a file, use the `-backend-config=PATH` option when running terraform init. If the file contains secrets it may be kept in a secure data store and downloaded to the local disk before running Terraform.

Create a two new directories. One called `state_configuration` containing two new files called `dev_local.hcl` and `test_local.hcl` and the other called `state_data` which will remain empty for the moment.

`dev_local.hcl`

```
path = "state_data/terraform.dev.tfstate"
```

`test_local.hcl`

```
path = "state_data/terraform.test.tfstate"
```

Below is a sample of the directory structure once these steps are complete.

```
|--- main.tf
|--- modules
|--- state_configuration
|    |--- dev_local.hcl
|    |--- test_local.hcl
|--- state_data
|--- terraform.tf
|--- variables.tf
```

Specify a particular state file for development by using the `backend-config` path:

```
terraform init -backend-config=state_configuration/dev_local.hcl -migrate-
    state
terraform plan
terraform apply
```

Notice that a new state file that will be created called: `terraform.dev.tfstate`. Cancel out of the apply, and show how you can do the same for your test environment configuration.

> Note: If you make a state file configuaration change, you most likely will need to provide the `-migrate-state` option to migrate state from one state file to the other. If that is not the intended action, then you will need to use `-reconfigure`. When modifying any teraform backend configuration it is important to understand the intent of your actions and the appropriate commands for that intent.

```
terraform init -backend-config=state_configuration/test_local.hcl -migrate
    -state
terraform plan
terraform apply
```

Task 3: Partial backend configuration via command line

Key/value pairs for a terraform backend configuration can be specified via the init command line. To specify a single key/value pair, use the `-backend-config="KEY=VALUE"` option when running terraform init.

```
terraform init -backend-config="path=state_data/terraform.prod.tfstate" -
    migrate-state
terraform plan
terraform apply
```

Note that many shells retain command-line flags in a history file, so this isn't recommended for secrets.

Task 4: Specifying multiple partial backend configurations

Terraform backend configuaration can also be split up into multiple files. If we update our `terraform.tf` to use the Terraform S3 backend we can break out each of the configuration items into a seperate configuration files if desired.

4.1 Full configuration within the `terraform.tf`

Update our Terraform configuration to utilize the s3 backend for storing state.

`terraform.tf`

```
terraform {
  backend "s3" {
```


```
    bucket = "my-terraform-state-ghm"
    key    = "dev/aws_infra"
    region = "us-east-1"
  }
}
```

```
terraform init -migrate-state
terraform plan
terraform apply
```

4.2 Partial configuration with a `terraform.tf` and `dev-s3-state.hcl`

The "s3" backend supports partial configuration that allows Terraform to be initialized with a dynamically set backend configuration. Let's update our Terraform configuration block to specify the bare minimum for our s3 backend configuration and utilize partial configuration files to provide the configuration values.

`terraform.tf`

```
terraform {
  backend "s3" {
  }
}
```

Create a `dev-s3-state.hcl` to specify where the state should be saved.

```
bucket = "my-terraform-state-ghm"
key    = "dev/aws_infra"
region = "us-east-1"
```

```
terraform init -backend-config="state_configuration/dev-s3-state.hcl" -
   migrate-state
terraform plan
terraform apply
```

4.3 Partial configuration with a `terraform.tf`, `s3-state-bucket.hcl` and `dev-s3-state-key.hcl`

We can break out our backend into multiple configuration files all containing partial configuration

`terraform.tf`

```
terraform {
```


```
  backend "s3" {
  }
}
```

s3-state-bucket.hcl

```
bucket = "my-terraform-state-ghm"
region = "us-east-1"
```

dev-s3-state-key.hcl

```
key    = "dev/aws_infra"
```

```
terraform init -backend-config="state_configuration/s3-state-bucket.hcl" \
  -backend-config="state_configuration/dev-s3-state-key.hcl" \
  -migrate-state
terraform plan
terraform apply
```

Task 5: Partial backend configuration via CLI prompt

If a required value for a backend configuration item is not specifed, Terraform will interactively ask you for the required values, unless interactive input is disabled. Terraform will not prompt for optional values.

Initialize terraform but intentionally leave out the `key` value which is required as part of the s3 back-end.

```
terraform init -backend-config="state_configuration/s3-state-bucket.hcl" \
  -migrate-state

Initializing modules...

Initializing the backend...
Backend configuration changed!

Terraform has detected that the configuration specified for the backend
has changed. Terraform will now check for existing state in the backends.

key
  The path to the state file inside the bucket

  Enter a value: dev/aws_infra
```


Task 6: Backend configuration from multiple locations

If backend settings are provided in multiple locations, the top-level settings are merged such that any command-line options override the settings in the main configuration and then the command-line options are processed in order, with later options overriding values set by earlier options.

Update the `terraform.tf` configuration to supply all information for the s3 backend as follows:

`terraform.tfstate`

```
terraform {
  backend "s3" {
    bucket = "my-terraform-state-ghm"
    key    = "dev/aws_infra"
    region = "us-east-1"
  }
}
```

Create a `prod-s3-state-key.hcl` file with the `state_configuration` directory with the following partial configuration:

`prod-s3-state-key.hcl`

```
key    = "prod/aws_infra"
```

```
terraform init -backend-config=state_configuration/s3-state-bucket.hcl \
-backend-config=state_configuration/prod-s3-state-key.hcl \
-migrate-state
```

```
Initializing modules...

Initializing the backend...
Backend configuration changed!

Terraform has detected that the configuration specified for the backend
has changed. Terraform will now check for existing state in the backends.

Do you want to copy existing state to the new backend?
  Pre-existing state was found while migrating the previous "s3" backend
     to the
  newly configured "s3" backend. An existing non-empty state already
     exists in
  the new backend. The two states have been saved to temporary files that
     will be
  removed after responding to this query.
```


```
  Previous (type "s3"): /var/folders/1c/qvs1hwp964z_dwd5qg07lv_00000gn/T/
     terraform2651199725/1-s3.tfstate
  New      (type "s3"): /var/folders/1c/qvs1hwp964z_dwd5qg07lv_00000gn/T/
     terraform2651199725/2-s3.tfstate

  Do you want to overwrite the state in the new backend with the previous
     state?
  Enter "yes" to copy and "no" to start with the existing state in the
     newly
  configured "s3" backend.

  Enter a value: yes
```

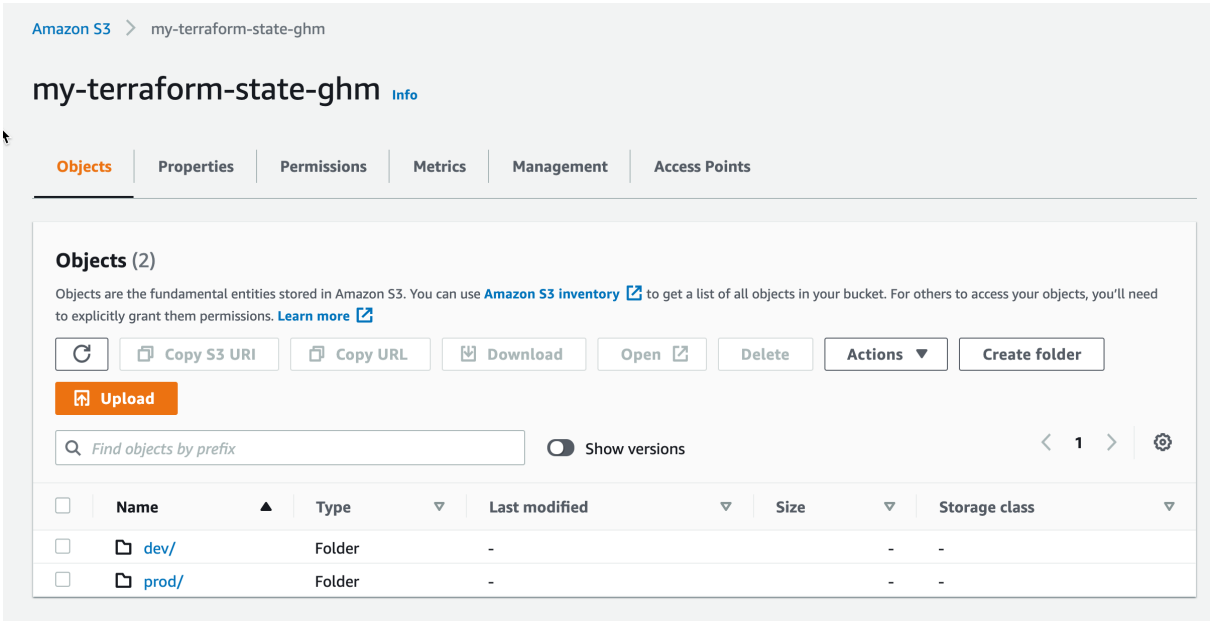

Figure 1: AWS S3 Backend - Keys

Task 7: Change state backend configuration back to default

Update the terraform backend configuration block to it's default value by removing the `backend` block from the `terraform.tf` file, and migrate state back to a local state file.

`terraform.tf`

```
terraform {
  required_version = ">= 1.0.0"
  required_providers {
    aws = {
```

```
      source  = "hashicorp/aws"
      version = "~> 3.0"
    }
    http = {
      source  = "hashicorp/http"
      version = "2.1.0"
    }
    random = {
      source  = "hashicorp/random"
      version = "3.1.0"
    }
    local = {
      source  = "hashicorp/local"
      version = "2.1.0"
    }
    tls = {
      source  = "hashicorp/tls"
      version = "3.1.0"
    }
  }
}
```

```
terraform init -migrate-state


Initializing modules...

Initializing the backend...
Terraform has detected you're unconfiguring your previously set "s3"
    backend.
Do you want to copy existing state to the new backend?
  Pre-existing state was found while migrating the previous "s3" backend
      to the
  newly configured "local" backend. An existing non-empty state already
      exists in
  the new backend. The two states have been saved to temporary files that
      will be
  removed after responding to this query.

  Previous (type "s3"): /var/folders/1c/qvs1hwp964z_dwd5qg07lv_00000gn/T/
      terraform2246932369/1-s3.tfstate
  New      (type "local"): /var/folders/1c/qvs1hwp964z_dwd5qg07lv_00000gn/
      T/terraform2246932369/2-local.tfstate

  Do you want to overwrite the state in the new backend with the previous
      state?
  Enter "yes" to copy and "no" to start with the existing state in the
      newly
```


```
  configured "local" backend.

  Enter a value: yes


Successfully unset the backend "s3". Terraform will now operate locally.
```

```
terraform plan
terraform apply
```

Once succesfully migrated you can delete the partial state backend configurations, the `state_configuration` and `state_data` directories.