CH04 Terraform Practices

- Believe Everyone Have Learned How to Leverage Terraform to Manage AWS Resource

- Now We Will Introduce How Develop Module and Make Your Terraform More Professional

Objectives

- What is Terragrunt?

- Modularize Everything

- Create AWS Resource in Multiple Region

What You Need In A Terraform Folder At Least Before?

```
Frontend/
├── Makefile              (Terraform Related Tasks)
├── env                   (Variable From Environment Variabl
├── asg.tf                (Define Cloud Provider Resources)
├── lb.tf                 (Define Cloud Provider Resources)
├── operations            (Some Helper Shell Script For Make
├── terraform.tfvars      (Some Predefined Variable Value)
├── ...
└── variables.tf          (Variable Definition)
```

After Using Terraform a Long Time...

- Have Multiple AWS Accounts

- Deploy Service Within Multiple Regions

- Trust Me, The Terraform Repository Will Become Mess

- And Need to Takes Time to Maintain Makefile and Helper Scripts

What is Terragrunt?

- Terragrunt is a Thin `Wrapper` for Terraform

- Provides Extra Tools for Keeping Your Terraform Configurations DRY (`Don't Repeat Yourself`)

- Working with Multiple Terraform `Modules`, and Managing `Remote State`

What is Terragrunt?

It's A Tool to Save Your Time, Force You to Produce Clean Code

What It Looks Like After Using Terragrunt

```
examples/
└── account_a
    ├── ap-northeast-1
    │   └── dev
    │       ├── env.tfvars
    │       ├── frontend
    │       │   └── terraform.tfvars
    │       └── terraform.tfvars
    └── us-west-2
        └── dev
            ├── env.tfvars
            ├── frontend
            │   └── terraform.tfvars
            └── terraform.tfvars
```

Exercise I

Try to Create A Fountend Server Group in Tokyo...

```
~$ cd ch04/examples/account_a/ap-northeast-1/dev/frontend

~$ terragrunt init

~$ terragrunt apply
```

Exercise II

If I Want to Achieve the Same Thing in Oregon...

```
~$ cd ch04/examples/account_a/us-west-2/dev/frontend

~$ terragrunt init

~$ terragrunt apply
```

What You Have Done Just Now?

- Create two Frontend Server Groups in Two Different Regions

- And Without Write Any Extra Terraform Code

- Let Us Go Through What Terragrunt Do!

```
~$ cd ch04/examples/account_a/us-west-2/dev
```

`account_a/ap-northeast-1/dev/terraform.tfvars`

- In folder `dev` Define Remote State Backend, Enviornment Variable

```
terragrunt = {
  remote_state {
    backend = "s3"

    config {
      encrypt         = true
      bucket          = "taipei-hug-workshop"
      key             = "account_a/ap-northeast-1/dev/${pa
      region          = "us-west-2"
    }
  }
...
}
```

`account_a/ap-northeast-1/dev/terraform.tfvars`

- In folder `dev` Define Remote State Backend, Enviornment Variable

```
terragrunt = {
...
  # Configure root level variables that all resources can
  terraform {
    extra_arguments "bucket" {
      commands = ["${get_terraform_commands_that_need_var

      required_var_files = [
        "${get_parent_tfvars_dir()}/env.tfvars",
      ]
    }
  }
}
```

account_a/ap-northeast-1/dev/frontend/terraform.tfvars

- In folder `frontend` Define Module Source, and the Variable Pass to Module terraform-aws-frontend

```
terragrunt = {
  # Terragrunt will copy the Terraform configurations spe
  # working directory, into a temporary folder, and execu
  terraform {
    source = "github.com/Taipei-HUG/terraform-aws-fronten
  }

  # Include all settings from the root terraform.tfvars f
  include = {
    path = "${find_in_parent_folders()}"
  }
}

...
```

account_a/ap-northeast-1/dev/frontend/terraform.tfvars

- In folder `frontend` Define Module Source, and the Variable Pass to Module terraform-aws-frontend

```
...

asg_config = {
  instance_count    = "1"
  instance_type     = "t3.small"
  root_volume_iops = "0"
  root_volume_size = "40"
  root_volume_type = "gp2"
}
```

Not Finish Yet...

We Have Not Understood Module Frontend Yet...

```
modules/
└── frontend
    ├── provision
    │   └── user_data
    ├── ami.tf
    ├── asg.tf
    ├── lb.tf
    ├── main.tf
    ├── outputs.tf
    ├── variables.tf
    └── vpc.tf
```

What the File main.tf Include?

```
provider "aws" {
  region = "${var.aws_region}"
  version = "1.35"
}

terraform {
  # The configuration for this backend will be filled in
  backend "s3" {}
  required_version = ">= 0.11.8"
}

provider "template" {
  version = "1.0.0"
}
```

If You Want to Create Something Afterward, Just ...

1. Develop/Find Module

2. Create Folder and *.tfvars Files

3. Execute terragrunt !

How to test Terraform?

- Kitchen (Reference)

- Terratest (Reference)

Key Takeaways

- Learned How to Use `Terragrunt`

- `Include/Retrive` Module from `GitHub`

- Create AWS Resource in Multiple Region, `But Not Writing Any Terraform Code`

Destroy Resource Created by Exercise

```
~$ cd ch04/examples/account_a/ap-northeast-1/dev/frontend

~$ terragrunt destroy

~$ cd ch04/examples/account_a/us-west-2/dev/frontend

~$ terragrunt destroy
```