NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

Terraform Part 2

Cloud Infrastructure Engineering

Nanyang Technological University & Skills Union - 2022/2023

Course Content

- Quick Check-In and Recap
- Cover some of Terraform Best Practices(Modules and Conditions)
- Instructor Demo and Activity

Self Study Check-In

What is the recommended way to manage Terraform state in a team environment?
a) Storing the state locally on each team member's machine
b) Sharing the state file via email
c) Using a remote backend like AWS S3 or Azure Storage
d) Committing the state file to a version control repository

Which of the following statements is true about Terraform's "plan" command?
a) It applies the changes to the infrastructure.
b) It validates the Terraform configuration files.
c) It creates an execution plan for applying changes to the infrastructure.
d) It destroys the infrastructure created by Terraform.

What is the purpose of a Terraform variable?
a) To define and store sensitive data
b) To specify the version of Terraform being used
c) To define input values for a Terraform module
d) To enforce naming conventions in Terraform resources

Overview of Best Practices today

- Using / Writing Modules
- Using Conditional Expressions

Modules

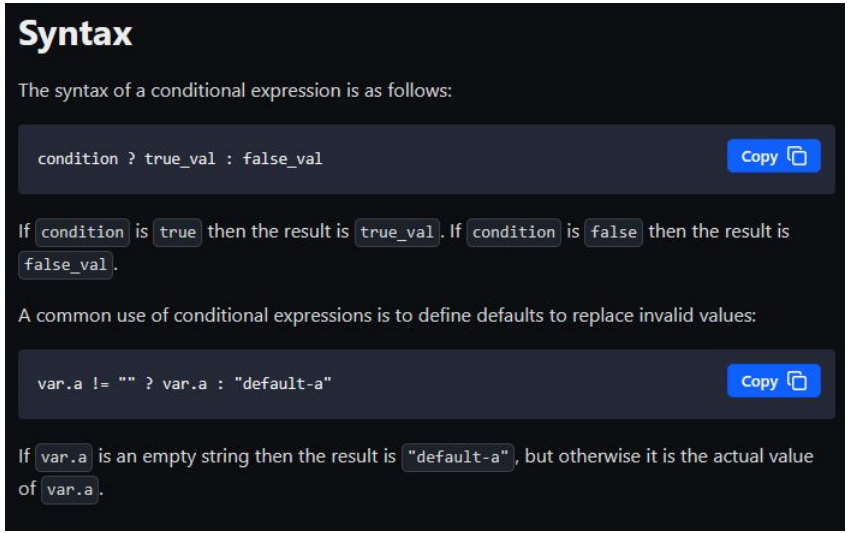
A module is a container for multiple resources that are used together. You can use modules to create lightweight abstractions, so that you can describe your infrastructure in terms of its architecture, rather than directly in terms of physical objects.

The .tf files in your working directory when you run terraform plan or terraform apply together form the root module. That module may call other modules and connect them together by passing output values from one to input values of another.

Conditional Expressions

A conditional expression uses the value of a boolean expression to select one of two values.

boolean = true or false value



Syntax

The syntax of a conditional expression is as follows:

```
condition ? true_val : false_val
```

Copy

If `condition` is `true` then the result is `true_val`. If `condition` is `false` then the result is `false_val`.

A common use of conditional expressions is to define defaults to replace invalid values:

```
var.a != "" ? var.a : "default-a"
```

Copy

If `var.a` is an empty string then the result is `"default-a"`, but otherwise it is the actual value of `var.a`.

Instructor Demo: Community Module

Activity: Community Module

Activity

For this activity we would be using the following community module: [terraform-aws-modules/terraform-aws-vpc: Terraform module which creates VPC resources on AWS 🇺🇦 (github.com)]

Step 1: Create a new folder for this activity and change directory into the folder and open it in VSCode

```
mkdir terraform-2

cd terraform-2

code .
```

Step 2: Within that directory, create the following files:

```
provider.tf

main.tf
```

Activity

provider.tf

```
provider "aws" {
    region = "ap-southeast-1"
}
```

Activity

main.tf

```
module "vpc" {
  source = "terraform-aws-modules/vpc/aws"

  name = "my-vpc"
  cidr = "10.0.0.0/16"

  azs             = ["ap-southeast-1a", "ap-southeast-1b", "ap-southeast-1c"]
  private_subnets = ["10.0.1.0/24", "10.0.2.0/24", "10.0.3.0/24"]
  public_subnets  = ["10.0.101.0/24", "10.0.102.0/24", "10.0.103.0/24"]

  enable_nat_gateway = true
}
```

Activity

Note: We would not be running “terraform apply” for this activity due to AWS service quota limitations.

Run the following commands and look at the resources getting created in the terminal:

terraform init

terraform plan

```
      + id                                     = (known after apply)
      + ipv6_cidr_block_association_id         = (known after apply)
      + ipv6_native                            = false
      + map_public_ip_on_launch                = false
      + owner_id                               = (known after apply)
      + private_dns_hostname_type_on_launch    = (known after apply)
      + tags                                   = {
          + "Name" = "my-vpc-public-ap-southeast-1c"
        }
      + tags_all                               = {
          + "Name" = "my-vpc-public-ap-southeast-1c"
        }
      + vpc_id                                 = (known after apply)
    }

  # module.vpc.aws_vpc.this[0] will be created
  + resource "aws_vpc" "this" {
      + arn                                  = (known after apply)
      + cidr_block                           = "10.0.0.0/16"
      + default_network_acl_id               = (known after apply)
      + default_route_table_id               = (known after apply)
      + default_security_group_id            = (known after apply)
      + dhcp_options_id                      = (known after apply)
      + enable_dns_hostnames                 = true
      + enable_dns_support                   = true
      + enable_network_address_usage_metrics = (known after apply)
      + id                                   = (known after apply)
      + instance_tenancy                     = "default"
      + ipv6_association_id                  = (known after apply)
      + ipv6_cidr_block                      = (known after apply)
      + ipv6_cidr_block_network_border_group = (known after apply)
      + main_route_table_id                  = (known after apply)
      + owner_id                             = (known after apply)
      + tags                                 = {
          + "Name" = "my-vpc"
        }
      + tags_all                             = {
          + "Name" = "my-vpc"
        }
    }

Plan: 31 to add, 0 to change, 0 to destroy.

Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if
you run "terraform apply" now.
PS C:\Users\jazee\OneDrive\Desktop\learn-tf\tf-2>
```

Instructor Demo: Custom Module

Activity: Custom Module

Activity

Step 1: Create a new folder for this activity and change directory into the folder and open it in VSCode

```
mkdir terraform-custom-module

cd terraform-custom-module

code .
```

Step 2: Within that directory, create the following files:

```
provider.tf

main.tf
```

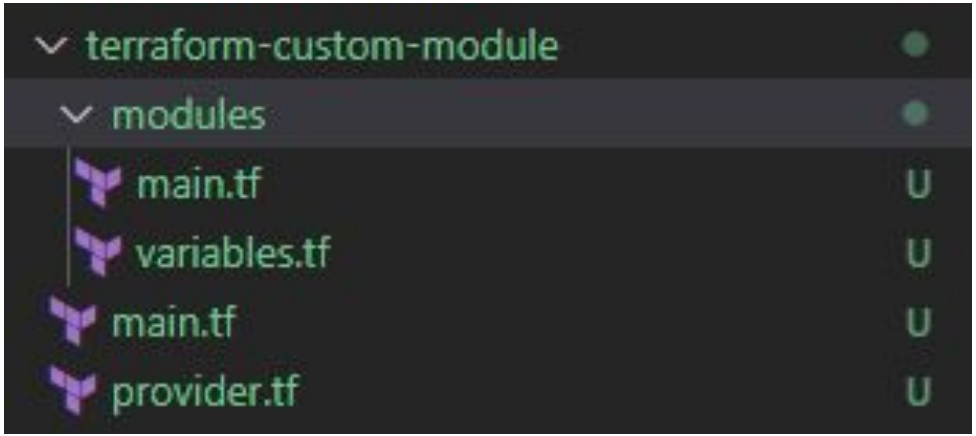
Step 3: Create a folder called modules and within that folder, create the following files:

```
main.tf

variables.tf
```

Activity

Folder Structure:



Activity

modules/main.tf

```
resource "random_integer" "suffix" {
  min = 1
  max = 50000
}

resource "aws_s3_bucket" "example" {
  bucket = "${var.resource_prefix}-sctps3bucket-${random_integer.suffix.result}"
}

resource "aws_s3_bucket_versioning" "versioning_example" {
  bucket = aws_s3_bucket.example.id
  versioning_configuration {
    status = var.versioning_status
  }
}

resource "aws_dynamodb_table" "personal_table" {
  count = var.create_dynamodb ? 1 : 0

  name         = "${var.resource_prefix}-sctpddb-${random_integer.suffix.result}"
  billing_mode = var.billing_mode
  hash_key     = "id"
  range_key    = "name"
  attribute {
    name = "id"
    type = "S"
  }
  attribute {
    name = "name"
    type = "S"
  }
}
```

Activity

modules/variables.tf

```
variable "versioning_status" {
  type    = string
  default = "Enabled"
}

variable "resource_prefix" {
  type    = string
  default = ""
}

variable "billing_mode" {
  type        = string
  description = "Billing Mode. Either PAY_PER_REQUEST or PROVISIONED"
  default     = "PAY_PER_REQUEST"
}

variable "create_dynamodb" {
  type    = bool
  default = false
}
```

Activity

main.tf

```
module "s3_dynamodb" {
  source = "./modules"

  resource_prefix = "jazeelsandbox" # Change this to a custom name you want to give
}
```

Activity

provider.tf

```
provider "aws" {
    region = "ap-southeast-1"
}
```

Activity

backend.tf

```
terraform {
  backend "s3" {
    bucket             = "sctp-ce2-tfstate-bkt"
    key                = "<YOUR NAME>-tf2-module.tfstate"
    region             = "ap-southeast-1"
  }
}
```

Activity

Learner:

- Clean up AWS.
- Remove/delete/terminate all service/ resources that created.

Instructor

- Clean up AWS.
- Remove/delete/terminate all service/ resources that created.
- Check the AWS account after learner clean up.

What’s Next?