Terraform Modules - Public Module Registry

Hashicorp maintains a public registry that helps you to consume Terraform modules from others. The Terraform Public Registry is an index of modules shared publicly and is the easiest way to get started with Terraform and find modules created by others in the community. It includes support for module versioning and searchable and filterable list of available modules for quickly deploying common infrastructure configurations.

Modules on the public Terraform Registry can be sourced using a registry source address of the form `<NAMESPACE>/<NAME>/<PROVIDER>`, with each module's information page on the registry site including the exact address to use.

- Task 1: Consuming Modules from the Terraform Module Registry
- Task 2: Exploring other modules from the Terraform Module Registry
- Task 3: Publishing to the Terraform Public Module Registry

Task 1: Consuming Modules from the Terraform Module Registry

In previous labs we began using modules from the Terraform Public Module Registry, including the auto scaling module. Let's add an S3 bucket to our configuration using the S3 public module.

`main.tf`

```
module "s3-bucket" {
  source  = "terraform-aws-modules/s3-bucket/aws"
  version = "2.11.1"
}

output "s3_bucket_name" {
  value = module.s3-bucket.s3_bucket_bucket_domain_name
}
```

```
terraform init
terraform plan


  # module.s3-bucket.aws_s3_bucket.this[0] will be created
  + resource "aws_s3_bucket" "this" {
      + acceleration_status         = (known after apply)
      + acl                         = "private"
      + arn                         = (known after apply)
      + bucket                      = (known after apply)
      + bucket_domain_name          = (known after apply)
```


```
      + bucket_regional_domain_name = (known after apply)
      + force_destroy               = false
      + hosted_zone_id              = (known after apply)
      + id                          = (known after apply)
      + region                      = (known after apply)
      + request_payer               = (known after apply)
      + tags_all                    = (known after apply)
      + website_domain              = (known after apply)
      + website_endpoint            = (known after apply)

      + versioning {
          + enabled    = (known after apply)
          + mfa_delete = (known after apply)
        }
    }

  # module.s3-bucket.aws_s3_bucket_public_access_block.this[0] will be
     created
  + resource "aws_s3_bucket_public_access_block" "this" {
      + block_public_acls       = false
      + block_public_policy     = false
      + bucket                  = (known after apply)
      + id                      = (known after apply)
      + ignore_public_acls      = false
      + restrict_public_buckets = false
    }
```

```
terraform apply

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

s3_bucket_name = "terraform-20211220113638832700000001.s3.amazonaws.com"
```

Task 2: Exploring other modules from the Terraform Module Registry

Another useful module that we may wish to utilize is the VPC Module. This is a simple Terraform module for creating VPC resource in AWS

Let's this to our configuration inside the `main.tf` of our root module:

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

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

  azs             = ["us-east-1a", "us-east-1b", "us-east-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
  enable_vpn_gateway = true

  tags = {
    Name = "VPC from Module"
    Terraform = "true"
    Environment = "dev"
  }
}
```

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

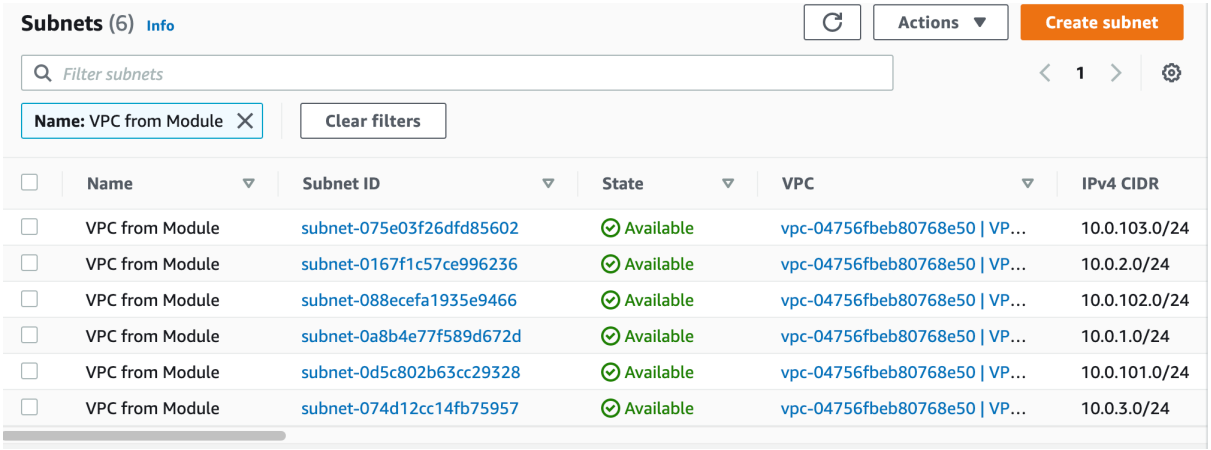

Figure 1: VPC Module

You can see that we can simplify our code base through the use of modules. The last two examples showcased building out an entire AWS VPC with private and public subnets, route tables, NAT gateways, VPN and Internet Gateways along with an S3 Bucket all via Terraform modules.

Task 3: Publishing to the Terraform Public Module Registry

Anyone can publish and share modules on the Terraform Registry, but there are some requirements that you should be aware of:

- The module must be on GitHub and must be a public repo. This is only a requirement for the public registry. If you're using a private registry, you may ignore this requirement.
- Module repositories must use a naming format: `terraform-<PROVIDER>-<NAME>` where reflects the type of infrastructure the module manages and is the main provider where it creates that infrastructure. The segment can contain additional hyphens. Examples: terraform-google-vault or terraform-aws-ec2-instance.
- The module repository must have a description which is used to populate the short description of the module. This should be a simple one sentence description of the module.
- The module must adhere to the standard module structure, `main.tf`, `variables.tf`, `outputs.tf` This allows the registry to inspect your module and generate documentation, track resource usage, parse submodules and examples, and more.
- x.y.z tags for releases. The registry uses tags to identify module versions. Release tag names must be a semantic version, which can optionally be prefixed with a v. For example, v1.0.4 and 0.9.2. To publish a module initially, at least one release tag must be present. Tags that don't look like version numbers are ignored.

Reference

Publishing to the Terraform Public Module Registry