使用资源配置工具terraform在aws上构建基础架 构

虽然可以直接在aws后台手动创建服务器等资源,但是这个创建过程重复过程成本比较高,所以可以把这个服务器资源使用terraform这个工具来管理,就可以把服务器资源的管理写到文件中了,服务器资源的管理可以通过修改文件实现

terraform支持aws, aliyun, ucloud等服务商, 完整的服务商列表参考: www.terraform.io/docs/provid...

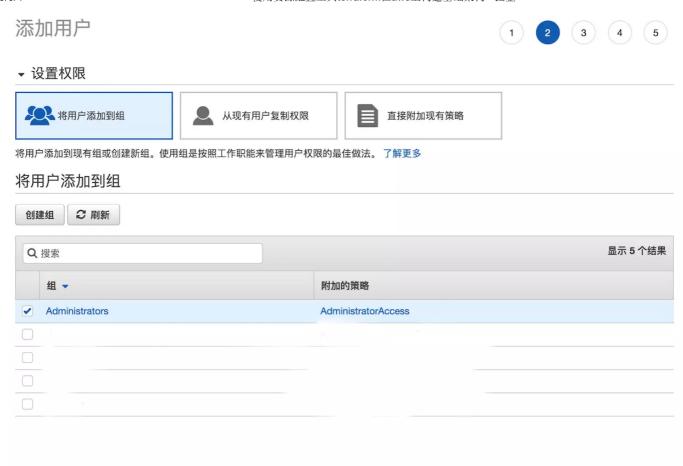
terraform是一个命令行工具,下载地址: www.terraform.io/downloads.h...

生成api访问密钥

到这个页面https://console.amazonaws.cn/iam/home?#/users 添加新用户,访问类型设置成编程访问



将用户添加到Administrators组,这样这个用户就有访问aws资源的权限了



点击下一步,操作效果如下



这里记录下这里生成的访问密钥ID和私有访问密钥,terraform通过这个密钥认证

配置访问密钥

把上面获取到的aws写入terraform配置文件,比如config.tf

```
provider "aws" {
  access_key = "aws的accesskey"
  secret_key = "aws的secretkey"
  region = "aws地区代码"
}
```

aws代码列表参考: docs.aws.amazon.com/zh_cn/gener...

然后使用这个 terraform init 命令下载aws插件, 操作效果如下

suxiaolin@suxiaolins-iMac → myterraform-aws terraform init

Initializing the backend...

Initializing provider plugins...

- Checking for available provider plugins...
- Downloading plugin for provider "aws" (terraform-providers/aws) 2.12.0...

The following providers do not have any version constraints in configuration, so the latest version was installed.

To prevent automatic upgrades to new major versions that may contain breaking changes, it is recommended to add version = "..." constraints to the corresponding provider blocks in configuration, with the constraint strings suggested below.

```
* provider.aws: version = "~> 2.12"
```

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see any changes that are required for your infrastructure. All Terraform commands should now work.

If you ever set or change modules or backend configuration for Terraform, rerun this command to reinitialize your working directory. If you forget, other commands will detect it and remind you to do so if necessary.

定义资源

定义vpc

vim vpc.tf

```
resource "aws vpc" "myvpc" {
    cidr block = "172.17.0.0/16"
    enable dns hostnames = true
    enable dns support
                            = true
    instance tenancy
                           = "default"
    tags = {
      Name = "myvpc"
    }
  }
使用terraform apply在aws上创建这个资源
suxiaolin@suxiaolins-iMac → myterraform-aws terraform apply
An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  + create
Terraform will perform the following actions:
  # aws_vpc.myvpc will be created
  + resource "aws_vpc" "myvpc" {
                                       = (known after apply)
     + arn
     + assign_generated_ipv6_cidr_block = false
                                      = "172.17.0.0/16"
     + cidr_block
     + default_network_acl_id
                                      = (known after apply)
     = (known after apply)
     + enable classiclink
     + enable_classiclink_dns_support = (known after apply)
     + enable_dns_hostnames
                                      = true
     + enable_dns_support
                                       = true
                                      = (known after apply)
                                      = "default"
     + instance_tenancy
                                   = (known after apply)
= (known after apply)
     + ipv6 association id
     + ipv6_cidr_block
     + main_route_table_id
     + owner_id
                                      = (known after apply)
                                       = {
     + tags
         + "Name" = "myvpc"
    }
Plan: 1 to add, 0 to change, 0 to destroy.
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
aws_vpc.myvpc: Creating...
aws_vpc.myvpc: Creation complete after 2s [id=vpc-054a32cba12d43efe]
Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
   and and an arrangement of the second
```

可以看到这个这个vpc已经创建成功了,可以在aws界面上确认



记下这个vpc的id: vpc-054a32cba12d43efe

定义子网

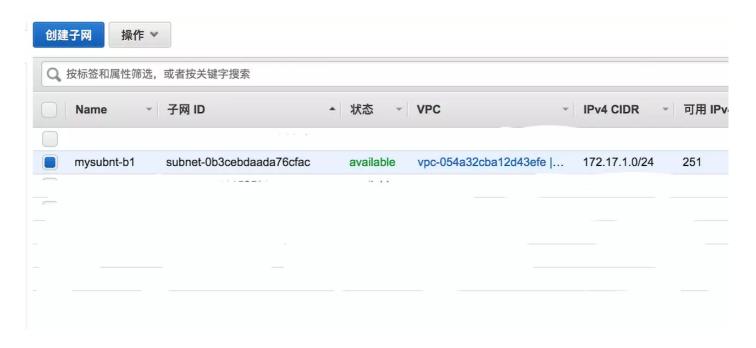
vim sn.tf

查看操作效果

```
suxiaolin@suxiaolins-iMac → myterraform-aws terraform apply
aws_vpc.myvpc: Refreshing state... [id=vpc-054a32cba12d43efe]
An execution plan has been generated and is shown below.
Resource actions are indicated with the following symbols:
  + create
Terraform will perform the following actions:
  # aws_subnet.mysubnt-b1 will be created
  + resource "aws_subnet" "mysubnt-b1" {
      + arn
                                        = (known after apply)
      + assign_ipv6_address_on_creation = false
      + availability_zone
                                        = "cn-northwest-1b"
                                        = (known after apply)
      + availability_zone_id
                                        = "172.17.1.0/24"
      + cidr_block
      + id
                                        = (known after apply)
                                        = (known after apply)
      + ipv6_cidr_block
      + ipv6_cidr_block_association_id = (known after apply)
      + map_public_ip_on_launch
                                        = false
      + owner_id
                                        = (known after apply)
      + tags
          + "Name" = "mysubnt-b1"
                                        = "vpc-054a32cba12d43efe"
      + vpc_id
Plan: 1 to add, 0 to change, 0 to destroy.
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
aws_subnet.mysubnt-b1: Creating...
aws_subnet.mysubnt-b1: Creation complete after 1s [id=subnet-0b3cebdaada76cfac]
```

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.

在这个aws界面上确认



记下这个subnet id: subnet-0b3cebdaada76cfac

定义实例

vim ec2.tf

同样可以在aws网页上确认

更新资源

比如把上面这个ec2规格修改成: t2.small vim ec2.tf修改后的ec2.tf如下

```
key_name = "dev"
subnet_id = "subnet-0b3cebdaada76cfac"
private_ip = "172.17.1.101"

tags = {
    Name = "myinstance"
}
```

然后使用这个 terraform apply 命令执行修改

```
# aws_instance.myinstance will be updated in-place
~ resource "aws_instance" "myinstance" {
                                   = "ami-0135cb179d33fbe3e"
      ami
                                   = "arn:aws-cn:ec2:cn-northwest-1:702752744573:instance/i-0b110e01
      arn
      associate_public_ip_address = false
      availability_zone
                                   = "cn-northwest-1b"
                                   = 2
      cpu_core_count
                                   = 1
      cpu_threads_per_core
      disable_api_termination
                                   = false
                                   = false
      ebs_optimized
                                   = false
      get_password_data
                                   = "i-0b110e01fcaaf4d98"
      id
      instance_state
                                   = "running"
                                   = "t2.medium" -> "t2.small"
    instance_type
      ipv6_address_count
                                   = []
      ipv6_addresses
                                   = "dev"
      key_name
      monitoring
                                   = false
      primary_network_interface_id = "eni-0af78fdbe42089189"
                                   = "ip-172-17-1-101.cn-northwest-1.compute.internal"
      private_dns
                                   = "172.17.1.101"
      private_ip
      security_groups
                                   = []
      source_dest_check
                                   = true
      cuhnat id
                                   - "cubnet_Ab3cebdaada76cfac"
```

可以看到terraform已经成功识别到了这个修改

可以在aws web界面上进行确认



删除资源

可以使用命令 terraform destroy 删除上面创建的所有资源

操作效果如下

Do you really want to destroy all resources?

Terraform will destroy all your managed infrastructure, as shown above. There is no undo. Only 'yes' will be accepted to confirm.

Enter a value: yes

aws_subnet.mysubnt-b1: Destroying... [id=subnet-0b3cebdaada76cfac]
aws_vpc.myvpc: Destroying... [id=vpc-054a32cba12d43efe]
aws_instance.myinstance: Destroying... [id=i-0b110e01fcaaf4d98]
aws_instance.myinstance: Still destroying... [id=i-0b110e01fcaaf4d98, 10s elapsed]
aws_subnet.mysubnt-b1: Still destroying... [id=subnet-0b3cebdaada76cfac, 10s elapsed]
aws_vpc.myvpc: Still destroying... [id=vpc-054a32cba12d43efe, 10s elapsed]
aws_instance.myinstance: Destruction complete after 20s
aws_subnet.mysubnt-b1: Still destroying... [id=subnet-0b3cebdaada76cfac, 20s elapsed]
aws_vpc.myvpc: Still destroying... [id=vpc-054a32cba12d43efe, 20s elapsed]
aws_subnet.mysubnt-b1: Destruction complete after 26s
aws_vpc.myvpc: Destruction complete after 26s

可以看到资源已经被删除了

导入资源

这个terraform可以导入已经存在的资源,参考: www.terraform.io/docs/import...

一些注意的点

1. 这个terraform是有状态的, *.state就是保存状态的文件

还可以关注的资源如下:

- 1. internet网关
- 2. nat网关

上面操作后的文件结构如下

suxiaolin@suxiaolins-iMac → myterraform-aws tree

```
config.tf
ec2.tf
sn.tf
terraform.tfstate
terraform.tfstate.backup
vpc.tf
```

0 directories, 6 files

这个terraform的配置文件可以定义变量,参考: www.terraform.io/docs/config...

参考资料

定义负载均衡器参考: www.terraform.io/docs/provid...

- 1. www.infoq.cn/article/9-r...
- 2. www.terraform.io/docs/provid...
- 3. www.terraform.io/docs/provid...
- 4. www.terraform.io/docs/provid...
- 5. www.terraform.io/downloads.h...
- 6. aws.amazon.com/cn/ec2/inst...
- 7. www.terraform.io/docs/config...