

# 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...](http://www.terraform.io/docs/provid...)

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

### 添加用户



#### 设置用户详细信息

您可以一次添加多个具有相同访问类型和权限的用户。 [了解更多](#)

用户名\*

[+](#) 添加其他用户

#### 选择 AWS 访问类型

选择这些用户将如何访问 AWS。在最后一步中提供访问密钥和自动生成的密码。 [了解更多](#)

- 访问类型\* ☒ 编程访问  
为 AWS API、CLI、SDK 和其他开发工具启用 访问密钥 ID 和 私有访问密钥。
- ☐ AWS 管理控制台访问  
启用 密码、使得用户可以登录到 AWS 管理控制台。

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

添加用户

- 1
- 2
- 3
- 4
- 5

设置权限

 将用户添加到组

 从现有用户复制权限

 直接附加现有策略

将用户添加到现有组或创建新组。使用组是按照工作职能来管理用户权限的最佳做法。 [了解更多](#)

将用户添加到组

创建组 刷新

Q 搜索

显示 5 个结果

组	附加的策略
<input checked="" type="checkbox"/> Administrators	AdministratorAccess
<input type="checkbox"/>	
<input type="checkbox"/>	
<input type="checkbox"/>	
<input type="checkbox"/>	

点击下一步，操作效果如下

添加用户

- 1
- 2
- 3
- 4
- 5

成功

您已成功创建了以下所示的用户。您可以查看和下载用户安全凭证。您还可以通过电子邮件向用户发送说明来登录到 AWS 管理控制台。这是最后一次这些凭证可供下载。不过，您可以随时创建新的凭证。

具有 AWS 管理控制台访问权限的用户可在以下位置登录: [signin.amazonaws.cn/console](https://signin.amazonaws.cn/console)

下载 .csv

用户	访问密钥 ID	私有访问密钥
<div>▼ <input checked="" type="checkbox"/> suxiaolin</div>	<div></div>	<div></div> 收藏

☒ 已创建用户 suxiaolin

☒ 将用户 suxiaolin 添加到 Administrators 组

☒ 已为用户 suxiaolin 创建访问密钥

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...](https://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" {
  + arn                                = (known after apply)
  + assign_generated_ipv6_cidr_block = false
  + cidr_block                        = "172.17.0.0/16"
  + 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_classiclink                = (known after apply)
  + enable_classiclink_dns_support   = (known after apply)
  + enable_dns_hostnames              = true
  + enable_dns_support                = true
  + id                                = (known after apply)
  + instance_tenancy                  = "default"
  + ipv6_association_id               = (known after apply)
  + ipv6_cidr_block                   = (known after apply)
  + main_route_table_id               = (known after apply)
  + 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.**

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

Create VPC 操作 ▾

按标签和属性筛选，或者按关键字搜索

<input type="checkbox"/>	Name ▾	VPC ID ▴	状态 ▾	Cidr 块	DHCP options set	Main Route table
<input type="checkbox"/>	myvpc	vpc-054a32cba12d43efe	available	172.17.0.0/16		
<input type="checkbox"/>						
<input type="checkbox"/>						

记下这个vpc的id: vpc-054a32cba12d43efe

## 定义子网

vim sn.tf

```
resource "aws_subnet" "mysubnt-b1" {  
    vpc_id          = "vpc-054a32cba12d43efe"  
    cidr_block      = "172.17.1.0/24"  
    availability_zone = "cn-northwest-1b"  
    map_public_ip_on_launch = false  
  
    tags {  
        Name = "mysubnt-b1"  
    }  
}
```

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"
  + availability_zone_id               = (known after apply)
  + cidr_block                         = "172.17.1.0/24"
  + id                                = (known after apply)
  + ipv6_cidr_block                    = (known after apply)
  + ipv6_cidr_block_association_id    = (known after apply)
  + map_public_ip_on_launch           = false
  + owner_id                           = (known after apply)
  + tags                              = {
    + "Name" = "mysubnt-b1"
  }
  + vpc_id                             = "vpc-054a32cba12d43efe"
}
```

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-0b3cebdada76cfac]

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

```
resource "aws_instance" "myinstance" {  
    ami          = "ami-0135cb179d33fbe3e"  
    instance_type = "t2.medium"  
  
    key_name = "dev"  
    subnet_id = "subnet-0b3cebdaada76cfac"  
    private_ip = "172.17.1.101"  
  
    tags = {  
        Name = "myinstance"  
    }  
}
```

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

```
resource "aws_instance" "myinstance" {  
    ami          = "ami-0135cb179d33fbe3e"  
    instance_type = "t2.small"
```

```

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                    = "ami-0135cb179d33fbe3e"
    arn                   = "arn:aws-cn:ec2:cn-northwest-1:702752744573:instance/i-0b110e01"
    associate_public_ip_address = false
    availability_zone      = "cn-northwest-1b"
    cpu_core_count         = 2
    cpu_threads_per_core   = 1
    disable_api_termination = false
    ebs_optimized          = false
    get_password_data      = false
    id                     = "i-0b110e01fcaaf4d98"
    instance_state         = "running"
    ~ instance_type        = "t2.medium" -> "t2.small"
    ipv6_address_count     = 0
    ipv6_addresses        = []
    key_name               = "dev"
    monitoring             = false
    primary_network_interface_id = "eni-0af78fdb42089189"
    private_dns            = "ip-172-17-1-101.cn-northwest-1.compute.internal"
    private_ip             = "172.17.1.101"
    security_groups        = []
    source_dest_check      = true
    subnet_id              = "subnet-0b3cebdaada76cfac"
}

```

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

search : myinstance 添加筛选条件					
<input type="checkbox"/>	Name	实例 ID	实例类型	可用区	实例状态
<input type="checkbox"/>	myinstance	i-0b110e01fcaaf4d98	t2.small	cn-northwest-1b	running

## 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...](http://www.terraform.io/docs/import...)

## 一些注意点

---

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...](http://www.terraform.io/docs/config...)

## 参考资料

---

定义负载均衡器参考：[www.terraform.io/docs/provid...](http://www.terraform.io/docs/provid...)

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