Terraform初体验（二） 第一个demo执行

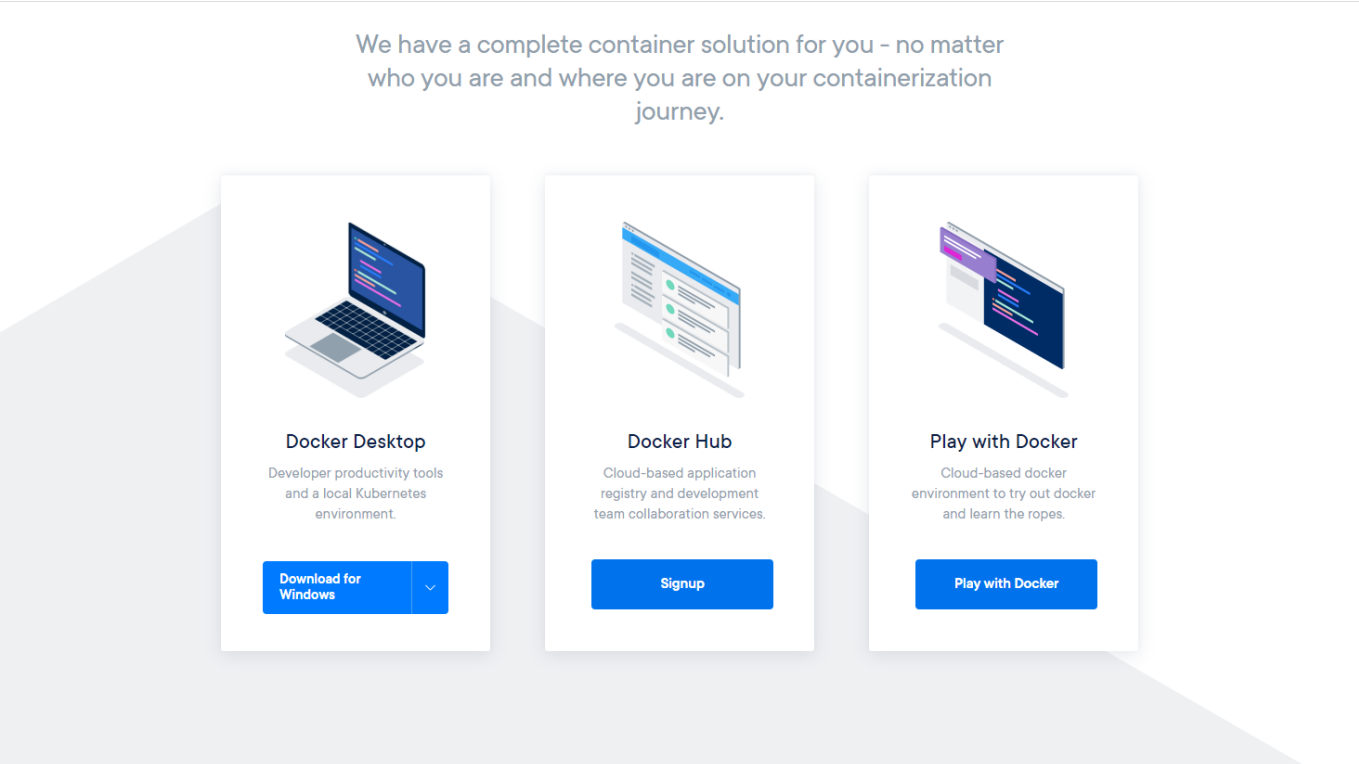
通过Terraform在本地运行docker nginx

前置条件：

1. 安装好windows docker
2. 安装好terraform

安装docker

安装windows docker可以直接登录docker.com下载安装即可，docker可以有图形化管理页面安装最新的19.03。为了简化第一次的操作，这里我们先不通过terraform来安装docker，docker下载安装地址 https://www.docker.com/get-started



编写main.tf

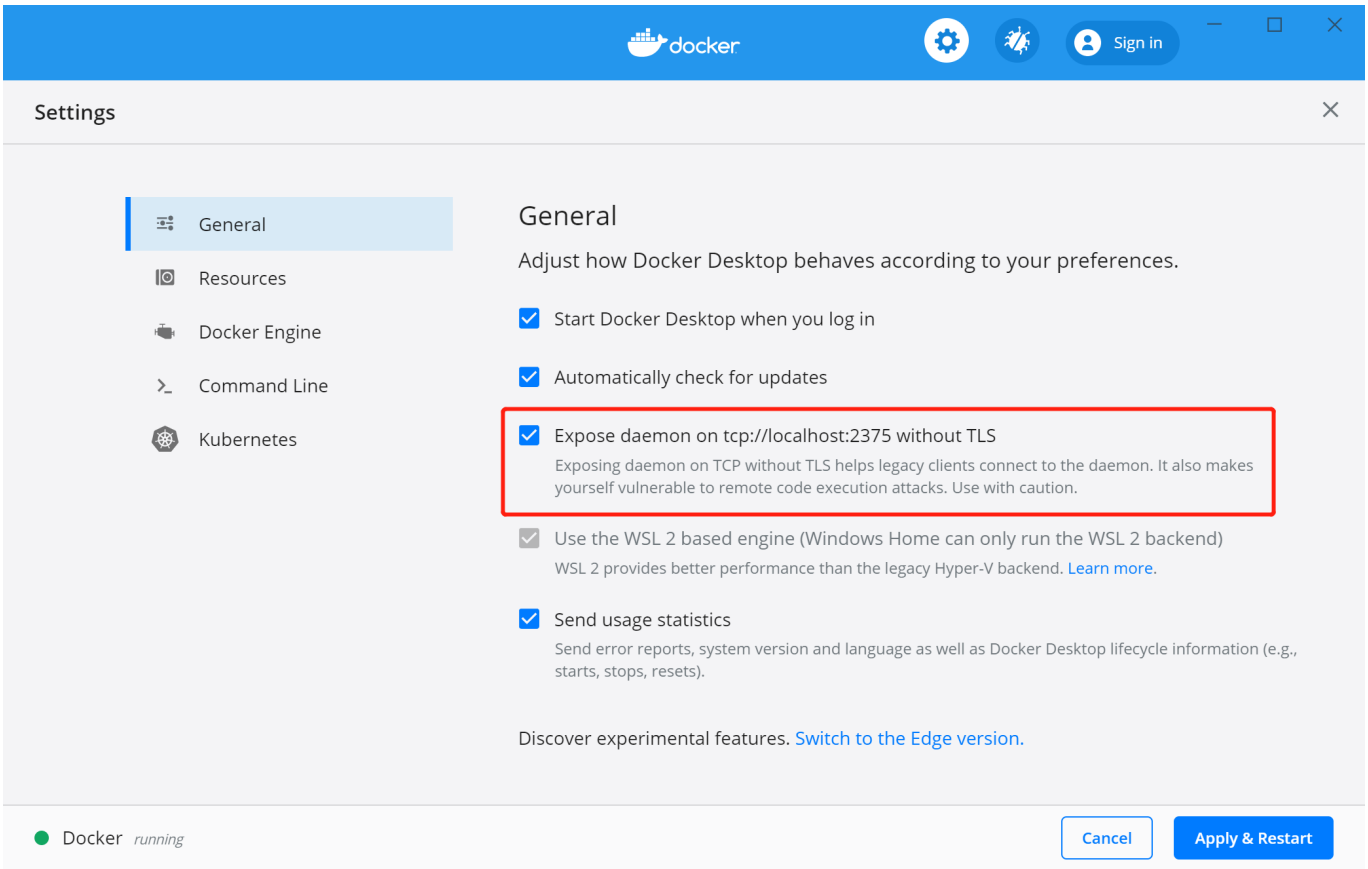
```
terraform {
  required_providers {
    docker = {
      source = "terraform-providers/docker"
    }
  }
}

provider "docker" {
  host    = "tcp://localhost:2375"
}
```

```
resource "docker_image" "nginx" {
  name         = "nginx:latest"
  keep_locally = false
}

resource "docker_container" "nginx" {
  image = docker_image.nginx.latest
  name  = "tutorial"
  ports {
    internal = 80
    external = 8000
  }
}
```

其中值得注意的是，官方的例子，在provider "docker"中指定的host是有些错误，无法运行，这里需要在docker desktop中设置开启"tcp://localhost:2375"，并替换tf文件中的host ="tcp://localhost:2375"。



执行main.tf

笔者使用的vs code，可以直接右键在终端中打开，然后依次进行以下步骤。

1. 初始化

在终端中执行`terraform init`。首次执行初始化操作，会有较长的时间去获取terraform中定义的source信息，在第一次初始化后没有source信息的变化，可以跳过初始化直接开始部署。

```
Microsoft Windows [版本 10.0.19041.508]
(c) 2019 Microsoft Corporation。保留所有权利。

E:\Ucloud\U_Doc\github\[redacted]\doc\terraform\test>terraform init

Initializing the backend...

Initializing provider plugins...
- Using previously-installed terraform-providers/docker v2.7.2

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, we recommend adding version constraints in a required_providers block
in your configuration, with the constraint strings suggested below.

* terraform-providers/docker: version = "~> 2.7.2"

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.

E:\Ucloud\U_Doc\github\[redacted]\doc\terraform\test>
```

2. 部署

在终端中执行`terraform plan`查看terraform执行计划，在终端中执行`terraform apply`完成部署。执行部署命令时，会将terraform的plan列出来展示给用户，并由用户确定执行。也可以输入`-auto-approve`跳过plan。

```
You, 4 days ago | 1 author (You)
terraform {
  required_providers {
    docker = {
      source = "terraform-providers/docker"
    }
  }
}

You, 4 days ago | 1 author (You)
provider "docker" {
  host    = "tcp://localhost:2375"
}

You, 4 days ago | 1 author (You)
resource "docker_image" "nginx" {
  name         = "nginx:latest"
```

输出 终端 调试控制台 问题

```
      + working_dir      = (known after apply)

      + ports {
          + external = 8000
          + internal = 80
          + ip       = "0.0.0.0"
          + protocol = "tcp"
        }
    }

  # docker_image.nginx will be created
  + resource "docker_image" "nginx" {
      + id           = (known after apply)
      + keep_locally = false
      + latest       = (known after apply)
      + name         = "nginx:latest"
    }

Plan: 2 to add, 0 to change, 0 to destroy.

------------------------------------------------------------------------

Note: You didn't specify an "-out" parameter to save this plan, so Terraform
can't guarantee that exactly these actions will be performed if
"terraform apply" is subsequently run.
```

输入"yes"

```
You, 4 days ago | 1 author (You)
terraform {
  required_providers {
    docker = {
      source = "terraform-providers/docker"
    }
  }
}

You, 4 days ago | 1 author (You)
provider "docker" {
  host    = "tcp://localhost:2375"
}

```

输出 终端 调试控制台 问题

```
      + id           = (known after apply)
      + keep_locally = false
      + latest       = (known after apply)
      + name         = "nginx:latest"
    }

Plan: 2 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
```

安装完成!

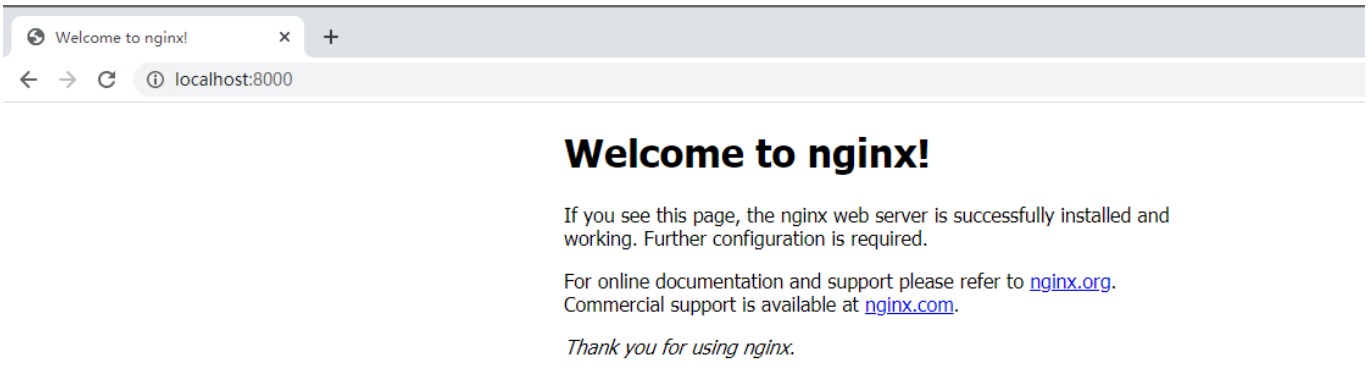
```
Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

docker_image.nginx: Creating...
docker_image.nginx: Still creating... [10s elapsed]
docker_image.nginx: Still creating... [20s elapsed]
docker_image.nginx: Still creating... [31s elapsed]
docker_image.nginx: Still creating... [41s elapsed]
docker_image.nginx: Still creating... [51s elapsed]
docker_image.nginx: Still creating... [1m1s elapsed]
docker_image.nginx: Still creating... [1m11s elapsed]
docker_image.nginx: Creation complete after 1m17s [id=sha256:7e4d58f0e5f3b60077e9a5d96b4be1b974b5a484f54f9393000a99f3b6816e3dnginx:latest]
docker_container.nginx: Creating...
docker_container.nginx: Creation complete after 1s [id=282af434b1b9e3d8e1d426001e55b659c8172af723ee2ee27b1a21c7a552e567]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
```

查看结果。



大家可以简单的字面理解main.tf中的语义，会在后面的内容中详细介绍，在此次执行中我们会在本地创建一个nginx的容器，并暴露800端口，我们访问localhost:800可以看到由terraform创建的容器可以正常运行。

3. 删除

在终端中执行`terraform destroy`。则删除由tf创建的docker容器。

```
输出  终端  调试控制台  问题                                                    2: cmd

  # docker_image.nginx will be destroyed
  - resource "docker_image" "nginx" {
      - id           = "sha256:7e4d58f0e5f3b60077e9a5d96b4be1b974b5a484f54f9393000a99f3b6816e3dnginx:latest" -> null
      - keep_locally = false -> null
      - latest       = "sha256:7e4d58f0e5f3b60077e9a5d96b4be1b974b5a484f54f9393000a99f3b6816e3d" -> null
      - name         = "nginx:latest" -> null
    }

Plan: 0 to add, 0 to change, 2 to destroy.

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

docker_container.nginx: Destroying... [id=282af434b1b9e3d8e1d426001e55b659c8172af723ee2ee27b1a21c7a552e567]
docker_container.nginx: Destruction complete after 1s
docker_image.nginx: Destroying... [id=sha256:7e4d58f0e5f3b60077e9a5d96b4be1b974b5a484f54f9393000a99f3b6816e3dnginx:la
docker_image.nginx: Destruction complete after 0s

Destroy complete! Resources: 2 destroyed.
```

附录

在我们执行`terraform -h`后看到terraform的相关操作命令和使用方法，整理给大家，如果你刚刚开始使用terraform，可以从这些基础命令开始，对于其他命令，请使用前阅读terraform的官方文档。

Usage: terraform [-version] [-help] [args]

Common commands: apply 构建或更改基础设施 console terraform传参的交互式控制台 destroy 删除由terraform控制的基础设施 env 工作空间管理 fmt 将配置文件重写为规范格式 get 下载并安装配置模块 graph 创建terraform资源的可视化图形 import 将现有基础设施导入terraform init 初始化terraform的工作目录 login 获取并保存远程主机的凭据 logout 删除远程主机的本地存储凭据 output 从状态文件读取输出 plan 生成并显示执行计划 providers 打印配置中使用的提供程序的树型结构 refresh 根据实际资源更新本地状态文件 show 检查

terraform的状态或计划 taint 手动标记污点以便资源重新创建 untaint 手动取消污点 validate 验证terraform文件 version terraform版本 workspace 工作空间管理

All other commands: 0.12upgrade 重写v0.12之前的模块源代码 0.13upgrade 重写v0.13之前的模块源代码 debug debug输出管理 force-unlock 手动解除terraform锁定状态 push 推送完成代码到企业仓库 state 关键状态管理