Lab 16: Using Docker with Terraform

In this lab, we're finally going to deploy docker image using TerraForm.

We're going to deploy an application called Node-red.

Now if you head to `https://nodered.org/`, you can get a lot more information about it.

Node-red is used as a flow based editor. As you can see here, it's used for several lot projects and things like that.

All right, so let's take a look at Node-red Docker.

Essentially, we'll be deploying a Node-red container using Docker.

Now, of course, Hashicorp TerraForm will be what we use to deploy all of this, so we'll be doing it as code versus running it as a script like you see here.

So within the Node-red Docker hub repo, you can see all the tags that are possible.

`https://hub.docker.com/r/nodered/node-red/tags`

We've got latest and latest minimal. These are the ones we'll be focusing on. But we're going to start with just latest.

Create a directory named `terraform-docker-container`.

`mkdir terraform-docker-container`

Navigate into the working directory.

`cd terraform-docker-container`

In the working directory, create a file called `main.tf` and paste the following Terraform configuration into it.

So the first thing we need to do is we need to download a Docker image that we will eventually deploy as a container. And the first thing we need to do is we need to provide a resource.

First thing we're going to do is create resource. The resources include Docker Container, Docker, Image, Docker Network and Docker volume.

These are the building blocks for our application that we will be building as code.

That's how this resource will be referenced throughout our scripts. So we're going to name this Node-red underscore image. And what we need to do is we need to specify the name of the image itself.

Again, this is the actual name of the image that you find in Docker Hub, not an arbitrary name that we're providing it.

```
terraform {
  required_providers {
    docker = {
      source  = "terraform-providers/docker"
      version = "~> 2.7.2"
    }
  }
}

provider "docker" {}

resource "docker_image" "nodered_image" {
```

```
  name  = "nodered/node-red:latest"
}

resource "docker_container" "nodered_container" {
  image = docker_image.nodered_image.latest
  name  = "nodered"

  ports {
    internal = 1880
    external = 1880
  }
}
```

Initialize the project, which downloads a plugin called a provider that lets Terraform interact with Docker.

`terraform init`

Fmt, which stands for Format, and then I'm going to do a dash diff which will show us the differences that are made.

`terraform fmt -diff`

As you can see, it gives us a nice little output of what it changed. And all it did was adjust some spacing and things like that, which really just cleans up the script and helps the script remain consistent throughout our deployment.

And now what we want to do is do a TerraForm plan.

`terraform plan`

So as you can see, we're going to be creating a resource. The resource is going to be the node-red image which will receive an ID And you can see here you've got the name.

So now that we've got our TerraForm plan, that's exactly what we want to do. We want to create a new image. And now we're going to terraform apply.

`terraform apply`

As you can see, it provides our plan information again so that we know what's going on before we apply. You always get this second chance to cancel the apply unless you use auto approve. We're going to try not to use that much as that can be pretty dangerous whenever you are working in production.So we're going to type. Yes, to approve.

And it is now pulling the image. So we'll give it a little bit of time here. And it's creating.

As you can see, we've got one added zero changed and zero destroyed.

Let's perform a `docker image ls` to take a look at that image.

You can go ahead and ignore any of other images. What we're focused on here of course, is our node-red image.

As you can see, it has downloaded and it is good to go. So our terraform apply worked and everything is good to go.

Let's go ahead and destroy our infrastructure. Enter yes to confirm this destroy

`terraform destroy`

As you can see, it is destroying the resource we created with ID latest name, everything else. And it's completed. Run `docker image ls` and it's gone.