Day 62 - Terraform and Docker

Terraform needs to be told which provider to be used in the automation, hence we need to give the provider name with source and version. For Docker, we can use this block of code in your

main.tf

Blocks and Resources in Terraform

Terraform block

Task-01

- Create a Terraform script with Blocks and Resources

```
terraform {
  required_providers {
    docker = {
      source  = "kreuzwerker/docker"
      version = "~> 2.21.0"
}
}
}
```

Note: **kreuzwerker/docker**, is shorthand for registry.terraform.io/kreuzwerker/docker.

Provider Block

A provider is a plugin that Terraform uses to create and manage your resources. In Order to make a provider available on Terraform, we need to make a **terraform init**.

This command downloads any plugins whatever need for our providers.

The provider block configures the specified provider, in this case, we use **docker** as a Provider.

```
provider "docker" {}
```

Resource: -

Use resource blocks to define components of your infrastructure. A resource might be a physical or virtual component such as a Docker container, or it can be a logical resource such as a Heroku application.

Resource blocks have two strings before the block: the resource type and the resource name.

The **Resource Type** are already predefined in a registry and User can set **Resource Name** as per their choice.

Registry URL: -
https://registry.terraform.io/providers/kreuzwerker/docker/latest/docs/resources/image

In this example, the first resource type is **docker_image** and the name are Nginx (user choice).

Task-02

- Create a resource Block for a nginx docker image

Hint:

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

- Create a resource Block for running a docker container for nginx

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

Note: In case Docker is not installed

```
sudo apt-get install docker.io
sudo docker ps
sudo chown $USER /var/run/docker.sock
```

Create a main.tf file and add Script

```
ubuntu@ip-172-31-85-217:~/Terraform$ vi main.tf
ubuntu@ip-172-31-85-217:~/Terraform$ ls -la
total 16
drwxrwxr-x 2 ubuntu ubuntu 4096 Apr 24 11:58 .
drwxr-x--- 6 ubuntu ubuntu 4096 Apr 24 11:58 ..
-rw-rw-r-- 1 ubuntu ubuntu  375 Apr 24 11:58 1
-rw-rw-r-- 1 ubuntu ubuntu  375 Apr 24 11:58 main.tf
```

```
terraform {
  required_providers {
    docker = {
      source  = "kreuzwerker/docker"
      version = "~> 2.21.0"
}
}
}

provider "docker" {}

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 = 80
 }
}
```

To install necessary plugins for providers, hit below command

Terraform init

```
ubuntu@ip-172-31-85-217:~/Terraform$ terraform init

Initializing the backend...

Initializing provider plugins...
- Finding kreuzwerker/docker versions matching "~> 2.21.0"...
- Installing kreuzwerker/docker v2.21.0...
- Installed kreuzwerker/docker v2.21.0 (self-signed, key ID BD080C4571C6104C)

Partner and community providers are signed by their developers.
If you'd like to know more about provider signing, you can read about it here:
https://www.terraform.io/docs/cli/plugins/signing.html

Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

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.
ubuntu@ip-172-31-85-217:~/Terraform$ terraform validate

  Warning: Deprecated attribute

    on main2.tf line 18, in resource "docker_container" "nginx":
    18:   image = docker_image.nginx.latest

  The attribute "latest" is deprecated. Refer to the provider documentation for details.

Success! The configuration is valid, but there were some validation warnings as shown above.
```

Install Docker and See the terraform plan execution,

terraform plan

```
ubuntu@ip-172-31-85-217:~/Terraform$ sudo docker --version
Docker version 20.10.21, build 20.10.21-0ubuntu1~22.04.2
ubuntu@ip-172-31-85-217:~/Terraform$ terraform plan

Planning failed. Terraform encountered an error while generating this plan.

  Warning: Deprecated attribute

    on main2.tf line 18, in resource "docker_container" "nginx":
    18:   image = docker_image.nginx.latest

  The attribute "latest" is deprecated. Refer to the provider documentation for details.


  Error: Error pinging Docker server: Got permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Get "http://%2Fvar%2Frun%2Fd
ocker.sock/_ping": dial unix /var/run/docker.sock: connect: permission denied

    with provider["registry.terraform.io/kreuzwerker/docker"],
    on main2.tf line 10, in provider "docker":
    10: provider "docker" {}
```

It is showing some permission related error

/var/run/docker.sock: connect permission denied

Solution: -

Add a permission

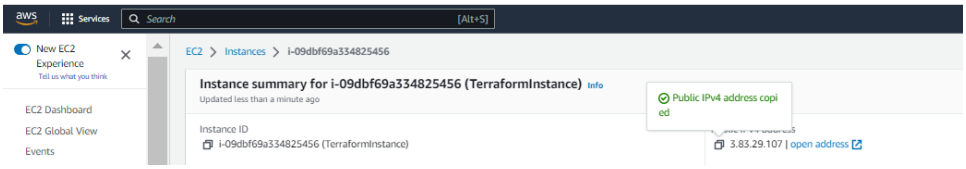
sudo chown $USER /var/run/docker.sock

```
ubuntu@ip-172-31-85-217:~/Terraform$ terraform plan

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following symbols:
  + create

Terraform will perform the following actions:

  # docker_container.nginx will be created
  + resource "docker_container" "nginx" {
      + attach           = false
      + bridge           = (known after apply)
      + command          = (known after apply)
      + container_logs   = (known after apply)
      + entrypoint       = (known after apply)
      + env              = (known after apply)
      + exit_code        = (known after apply)
      + gateway          = (known after apply)
      + hostname         = (known after apply)
      + id               = (known after apply)
      + image            = (known after apply)



      + ipc_mode         = (known after apply)
      + log_driver       = (known after apply)
      + logs             = false
      + must_run         = true
      + name             = "tutorial"
      + network_data     = (known after apply)
```

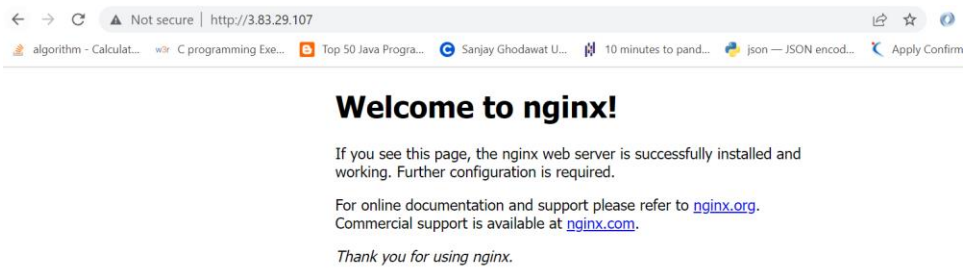
Apply the terraform plan, using **terraform apply** command.

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

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

  Warning: Deprecated attribute

    on main2.tf line 18, in resource "docker_container" "nginx":
    18:  image = docker_image.nginx.latest

  The attribute "latest" is deprecated. Refer to the provider documentation for details.

  (and one more similar warning elsewhere)


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: Creation complete after 6s [id=sha256:6efc10a0510f143a90b69dc564a914574973223e88418d65c1f8809e08dc0a1fnginx:latest]
docker_container.nginx: Creating...
docker_container.nginx: Creation complete after 1s [id=2b90cdf7d6803d2bff1b8b00eaba1870c6cdc9a3bd681f5cfd8508da602f0de9]

  Warning: Deprecated attribute

    on main2.tf line 18, in resource "docker_container" "nginx":
    18:  image = docker_image.nginx.latest

  The attribute "latest" is deprecated. Refer to the provider documentation for details.

  (and one more similar warning elsewhere)


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

Copy the public address and hit on browser



Your App is running...



Destroy the terraform execution, using **terraform execution** command.

```
        - ip       = "0.0.0.0" -> null
        - protocol = "tcp" -> null
      }
  }

  # docker_image.nginx will be destroyed
  - resource "docker_image" "nginx" {
      - id           = "sha256:6efc10a0510f143a90b69dc564a914574973223e88418d65c1f8809e08dc0a1fnginx:latest" -> null
      - image_id     = "sha256:6efc10a0510f143a90b69dc564a914574973223e88418d65c1f8809e08dc0a1f" -> null
      - keep_locally = false -> null
      - latest       = "sha256:6efc10a0510f143a90b69dc564a914574973223e88418d65c1f8809e08dc0a1f" -> null
      - name         = "nginx:latest" -> null
      - repo_digest  = "nginx@sha256:63b44e8ddb83d5dd8020327c1f40436e37a6fffd3ef2498a6204df23be6e7e94" -> null
    }

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

  Warning: Deprecated attribute

    on main2.tf line 18, in resource "docker_container" "nginx":
    18:  image = docker_image.nginx.latest

  The attribute "latest" is deprecated. Refer to the provider documentation for details.


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=3515f3f4f431c59614429dc50631ac3779c46642c78861631bb714318bdc264e]
docker_container.nginx: Destruction complete after 1s
docker_image.nginx: Destroying... [id=sha256:6efc10a0510f143a90b69dc564a914574973223e88418d65c1f8809e08dc0a1fnginx:latest]
docker_image.nginx: Destruction complete after 0s

Destroy complete! Resources: 2 destroyed.
```

Happy Learning...