Name : Ganesh Gupta   Div : D15C   Roll no : 13

Creating docker image using terraform

Step 1: Download and install Docker Desktop by visiting https://www.docker.com. Run the installer and follow the prompts to complete the installation, then verify by launching Docker Desktop or using the `docker --version` command.

```
C:\Users\Admin>docker

Usage:  docker [OPTIONS] COMMAND

A self-sufficient runtime for containers

Common Commands:
  run          Create and run a new container from an image
  exec         Execute a command in a running container
  ps           List containers
  build        Build an image from a Dockerfile
  pull         Download an image from a registry
  push         Upload an image to a registry
  images       List images
  login        Log in to a registry
  logout       Log out from a registry
  search       Search Docker Hub for images
  version      Show the Docker version information
  info         Display system-wide information

Management Commands:
  builder      Manage builds
  buildx*      Docker Buildx
  compose*     Docker Compose
  container    Manage containers
  context      Manage contexts
  debug*       Get a shell into any image or container
  desktop*     Docker Desktop commands (Alpha)
  dev*         Docker Dev Environments
  extension*   Manages Docker extensions
  feedback*    Provide feedback, right in your terminal!
  image        Manage images
```

```
C:\Users\Admin>docker --version
Docker version 27.1.1, build 6312585
```

Step 2: Now, create a folder named '**Terraform Scripts**' in which we save our different types of scripts which will be further used in this experiment.

Step 3: First, create a new folder named `Docker` inside the `TerraformScripts` folder. Then, open Notepad and create a new file named `docker.tf` within the `Docker` folder. Write the following contents into the `docker.tf` file to create an Ubuntu Linux container. Save the file when done.
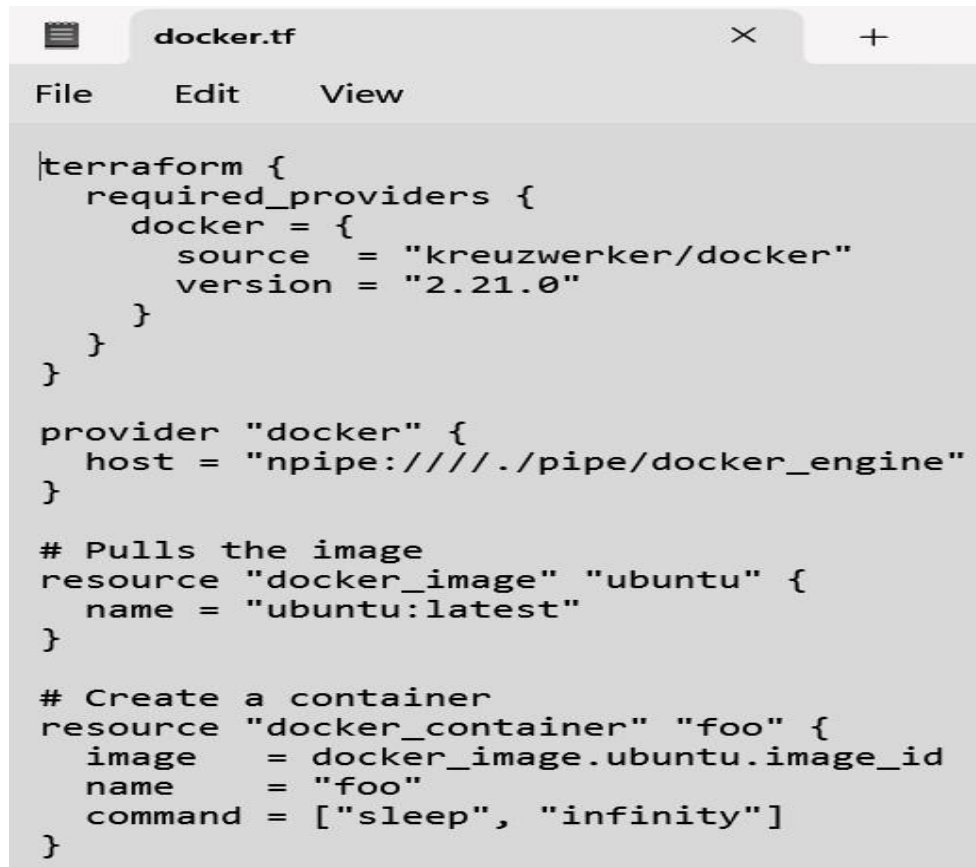
Script:

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

provider "docker" {
  host = "npipe:////./pipe/docker_engine"
}

# Pulls the image resource
"docker_image" "ubuntu" {
  name = "ubuntu:latest"
}

# Create a container resource
"docker_container" "foo" {
  image = docker_image.ubuntu.image_id
  name        = "foo"
  command = ["sleep", "infinity"]
}
```

This Terraform script configures the Docker provider to communicate with the Docker Engine using a Windows named pipe.

It pulls the latest Ubuntu image from Docker Hub and creates a container named "foo."

The container runs the `sleep infinity` command, which keeps it active indefinitely.

This setup is useful for scenarios where the container needs to remain running continuously.

```
docker.tf                                    ×        +

File    Edit     View

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

provider "docker" {
   host = "npipe:////./pipe/docker_engine"
}

# Pulls the image
resource "docker_image" "ubuntu" {
   name = "ubuntu:latest"
}

# Create a container
resource "docker_container" "foo" {
   image     = docker_image.ubuntu.image_id
   name      = "foo"
   command   = ["sleep", "infinity"]
}
```

Step 4: Execute the `terraform init` command to initialize the working directory, download the necessary provider plugins, and set up the backend for managing Terraform state.

```
PS C:\Users\Admin\Desktop\Terraformscripts\docker> 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.
PS C:\Users\Admin\Desktop\Terraformscripts\docker> |
```

Name : Ganesh Gupta   Div : D15C   Roll no : 13

Step 5: Run `terraform plan` to preview the actions Terraform will take to reach the desired state defined in your configuration, including creating, modifying, or deleting resources.

```
PS C:\Users\Admin\Desktop\Terraformscripts\docker> 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.foo will be created
  + resource "docker_container" "foo" {
      + attach              = false
      + bridge              = (known after apply)
      + command             = [
          + "sleep",
          + "infinity",
        ]
      + 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)
      + init                = (known after apply)
      + ip_address          = (known after apply)
      + ip_prefix_length    = (known after apply)
      + ipc_mode            = (known after apply)
      + log_driver          = (known after apply)
      + logs                = false
      + must_run            = true
      + name                = "foo"
      + network_data        = (known after apply)
      + read_only           = false
```

```
      + read_only           = false
      + remove_volumes      = true
      + restart             = "no"
      + rm                  = false
      + runtime             = (known after apply)
      + security_opts       = (known after apply)
      + shm_size            = (known after apply)
      + start               = true
      + stdin_open          = false
      + stop_signal         = (known after apply)
      + stop_timeout        = (known after apply)
      + tty                 = false

      + healthcheck (known after apply)

      + labels (known after apply)
    }

  # docker_image.ubuntu will be created
  + resource "docker_image" "ubuntu" {
      + id          = (known after apply)
      + image_id    = (known after apply)
      + latest      = (known after apply)
      + name        = "ubuntu:latest"
      + output      = (known after apply)
      + repo_digest = (known after apply)
    }

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


Note: You didn't use the -out option to save this plan, so Terraform can't guarantee to take exactly these actions if you run "terraform apply"
now.
PS C:\Users\Admin\Desktop\Terraformscripts\docker> |
```

Step 6: Execute "**terraform apply**" to apply the configuration, which will automatically create and run the Ubuntu container based on our configuration.

```
PS C:\Users\Admin\Desktop\Terraformscripts\docker> terraform apply

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.foo will be created
  + resource "docker_container" "foo" {
      + attach           = false
      + bridge           = (known after apply)
      + command          = [
          + "sleep",
          + "infinity",
        ]
      + 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)
      + init             = (known after apply)
      + ip_address       = (known after apply)
      + ip_prefix_length = (known after apply)
      + ipc_mode         = (known after apply)
      + log_driver       = (known after apply)
      + logs             = false
      + must_run         = true
      + name             = "foo"
      + network_data     = (known after apply)
      + read_only        = false
      + remove_volumes   = true
```

```
      + read_only        = false
      + remove_volumes   = true
      + restart          = "no"
      + rm               = false
      + runtime          = (known after apply)
      + security_opts    = (known after apply)
      + shm_size         = (known after apply)
      + start            = true
      + stdin_open       = false
      + stop_signal      = (known after apply)
      + stop_timeout     = (known after apply)
      + tty              = false

      + healthcheck (known after apply)

      + labels (known after apply)
    }

  # docker_image.ubuntu will be created
  + resource "docker_image" "ubuntu" {
      + id          = (known after apply)
      + image_id    = (known after apply)
      + latest      = (known after apply)
      + name        = "ubuntu:latest"
      + output      = (known after apply)
      + repo_digest = (known after apply)
    }

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

docker_image.ubuntu: Creating...
docker_image.ubuntu: Still creating... [10s elapsed]
docker_image.ubuntu: Still creating... [20s elapsed]
docker_image.ubuntu: Still creating... [30s elapsed]
docker_image.ubuntu: Creation complete after 37s [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]
docker_container.foo: Creating...
docker_container.foo: Creation complete after 2s [id=76c6390ec277dc11f709997c4eb636ee8b45f90723d0f6ec07d85caeda18ead9]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
PS C:\Users\Admin\Desktop\Terraformscripts\docker> |
```

Step 7: The command `**docker images**` lists all Docker images stored locally on your system, showing details like repository names, tags, image IDs, and creation dates.

Docker images, Before Executing Apply step:

```
PS C:\Users\Admin\Desktop\Terraformscripts\docker> docker images
REPOSITORY    TAG        IMAGE ID        CREATED        SIZE
```

Docker images, After Executing Apply step:

```
PS C:\Users\Admin\Desktop\Terraformscripts\docker> docker images
REPOSITORY    TAG        IMAGE ID        CREATED        SIZE
ubuntu        latest     edbfe74c41f8    3 weeks ago    78.1MB
```

Step 8: Execute Terraform destroy to delete the configuration, which will automatically delete the Ubuntu Container.

```
PS C:\Users\Admin\Desktop\Terraformscripts\docker> terraform destroy
docker_image.ubuntu: Refreshing state... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]
docker_container.foo: Refreshing state... [id=76c6390ec277dc11f709997c4eb636ee8b45f90723d0f6ec07d85caeda18ead9]

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

Terraform will perform the following actions:

  # docker_container.foo will be destroyed
  - resource "docker_container" "foo" {
      - attach            = false -> null
      - command           = [
          - "sleep",
          - "infinity",
        ] -> null
      - cpu_shares        = 0 -> null
      - dns               = [] -> null
      - dns_opts          = [] -> null
      - dns_search        = [] -> null
      - entrypoint        = [] -> null
      - env               = [] -> null
      - gateway           = "172.17.0.1" -> null
      - group_add         = [] -> null
      - hostname          = "76c6390ec277" -> null
      - id                = "76c6390ec277dc11f709997c4eb636ee8b45f90723d0f6ec07d85caeda18ead9" -> null
      - image             = "sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598a" -> null
      - init              = false -> null
      - ip_address        = "172.17.0.2" -> null
      - ip_prefix_length  = 16 -> null
      - ipc_mode          = "private" -> null
      - links             = [] -> null
      - log_driver        = "json-file" -> null
      - log_opts          = {} -> null
      - logs              = false -> null
      - max_retry_count   = 0 -> null
      - memory            = 0 -> null
      - memory_swap       = 0 -> null
      - must_run          = true -> null
      - name              = "foo" -> null
      - network_data      = [
          - {
              - gateway                   = "172.17.0.1"
              - global_ipv6_prefix_length = 0
              - ip_address                = "172.17.0.2"
              - ip_prefix_length          = 16
              - network_name              = "bridge"
                # (2 unchanged attributes hidden)
```

```
                - network_name            = "bridge"
                  # (2 unchanged attributes hidden)
              },
          ] -> null
        - network_mode      = "bridge" -> null
        - privileged        = false -> null
        - publish_all_ports = false -> null
        - read_only         = false -> null
        - remove_volumes    = true -> null
        - restart           = "no" -> null
        - rm                = false -> null
        - runtime           = "runc" -> null
        - security_opts     = [] -> null
        - shm_size          = 64 -> null
        - start             = true -> null
        - stdin_open        = false -> null
        - stop_timeout      = 0 -> null
        - storage_opts      = {} -> null
        - sysctls           = {} -> null
        - tmpfs             = {} -> null
        - tty               = false -> null
          # (8 unchanged attributes hidden)
      }

  # docker_image.ubuntu will be destroyed
  - resource "docker_image" "ubuntu" {
      - id           = "sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest" -> null
      - image_id     = "sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598a" -> null
      - latest       = "sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598a" -> null
      - name         = "ubuntu:latest" -> null
      - repo_digest  = "ubuntu@sha256:8a37d68f4f73ebf3d4efafbcf66379bf3728902a8038616808f04e34a9ab63ee" -> 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.foo: Destroying... [id=76c6390ec277dc11f709997c4eb636ee8b45f90723d0f6ec07d85caeda18ead9]
docker_container.foo: Destruction complete after 1s
docker_image.ubuntu: Destroying... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]
docker_image.ubuntu: Destruction complete after 1s

Destroy complete! Resources: 2 destroyed.
PS C:\Users\Admin\Desktop\Terraformscripts\docker>
```

Step 9: Docker images After Executing Destroy step

```
PS C:\Users\Admin\Desktop\Terraformscripts\docker> docker images
REPOSITORY    TAG        IMAGE ID    CREATED    SIZE
```