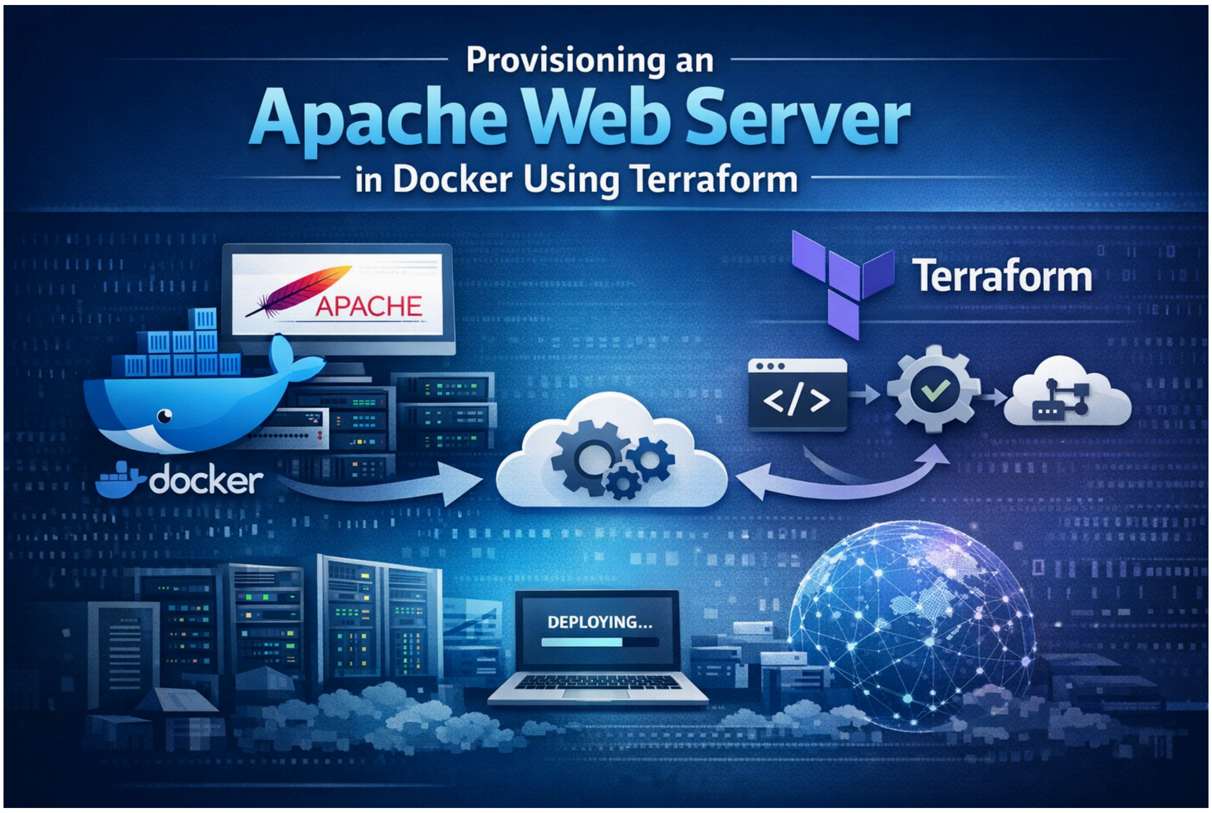
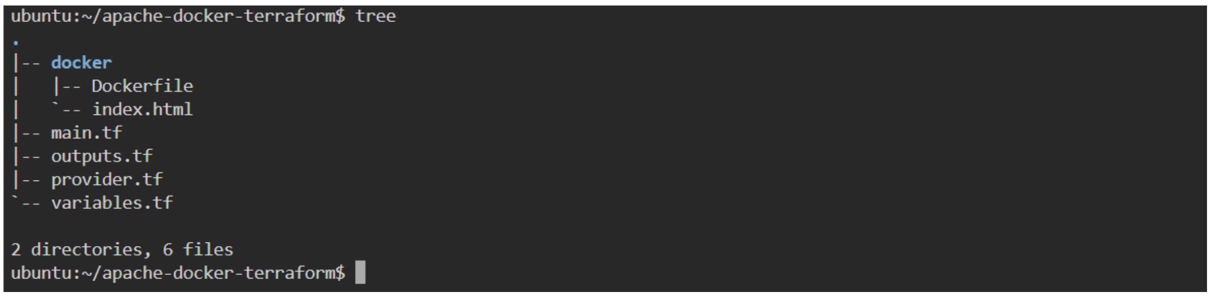
Provisioning an Apache Web Server in Docker Using Terraform



- **Project Structure**

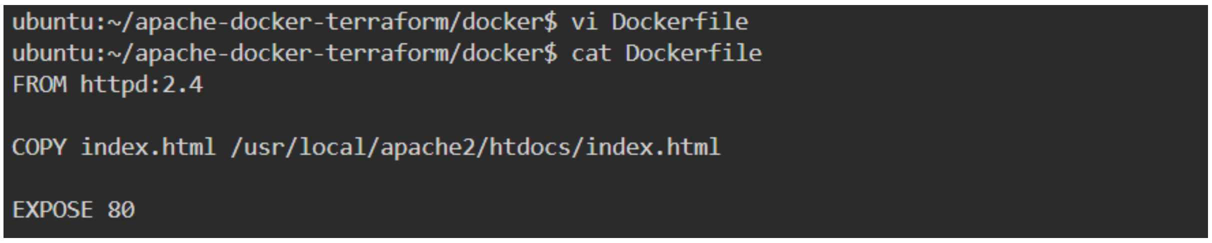
```
apache-docker-terraform/
├── main.tf
├── provider.tf
├── variables.tf
├── outputs.tf
├── terraform.tfvars
└── docker/
    ├── Dockerfile
    └── index.html
```



```
ubuntu:~/apache-docker-terraform$ tree
.
|-- docker
|   |-- Dockerfile
|   `-- index.html
|-- main.tf
|-- outputs.tf
|-- provider.tf
`-- variables.tf

2 directories, 6 files
ubuntu:~/apache-docker-terraform$
```

Step 1: Dockerfile (Apache Image)

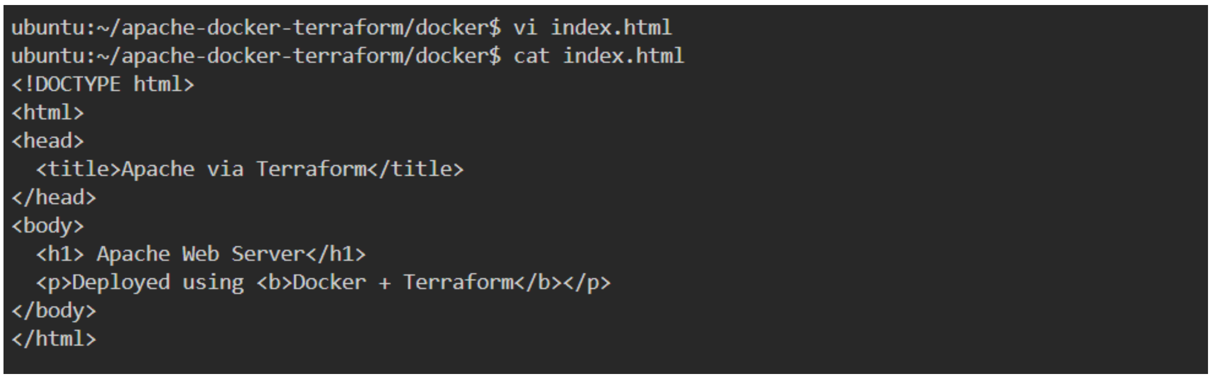
📄 `docker/Dockerfile`



```
ubuntu:~/apache-docker-terraform/docker$ vi Dockerfile
ubuntu:~/apache-docker-terraform/docker$ cat Dockerfile
FROM httpd:2.4

COPY index.html /usr/local/apache2/htdocs/index.html

EXPOSE 80
```

Step 2: Apache Web Page

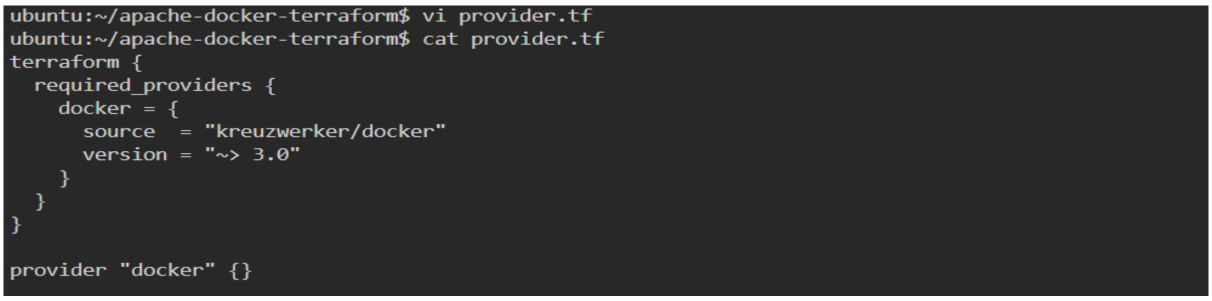
📄 `docker/index.html`



```
ubuntu:~/apache-docker-terraform/docker$ vi index.html
ubuntu:~/apache-docker-terraform/docker$ cat index.html
<!DOCTYPE html>
<html>
<head>
  <title>Apache via Terraform</title>
</head>
<body>
  <h1> Apache Web Server</h1>
  <p>Deployed using <b>Docker + Terraform</b></p>
</body>
</html>
```

Step 3: Terraform Provider

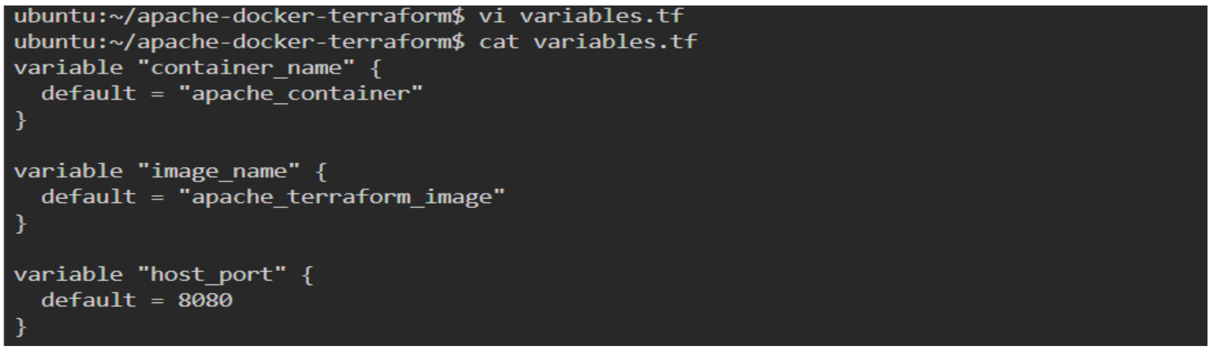
provider.tf



```
ubuntu:~/apache-docker-terraform$ vi provider.tf
ubuntu:~/apache-docker-terraform$ cat provider.tf
terraform {
  required_providers {
    docker = {
      source  = "kreuzwerker/docker"
      version = "~> 3.0"
    }
  }
}

provider "docker" {}
```

Step 4: Variables

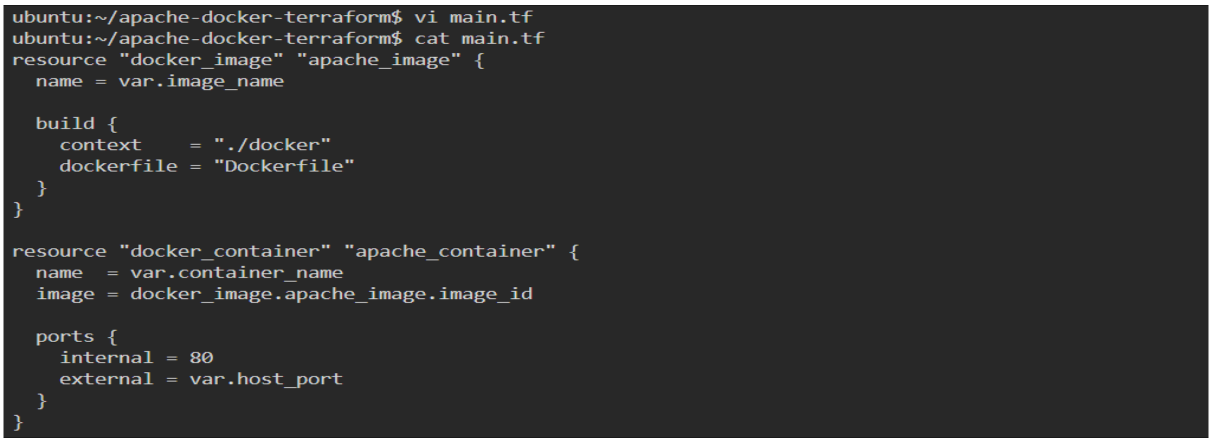
variables.tf



```
ubuntu:~/apache-docker-terraform$ vi variables.tf
ubuntu:~/apache-docker-terraform$ cat variables.tf
variable "container_name" {
  default = "apache_container"
}

variable "image_name" {
  default = "apache_terraform_image"
}

variable "host_port" {
  default = 8080
}
```

Step 5: Main Terraform Configuration

main.tf



```
ubuntu:~/apache-docker-terraform$ vi main.tf
ubuntu:~/apache-docker-terraform$ cat main.tf
resource "docker_image" "apache_image" {
  name = var.image_name

  build {
    context    = "./docker"
    dockerfile = "Dockerfile"
  }
}

resource "docker_container" "apache_container" {
  name  = var.container_name
  image = docker_image.apache_image.image_id

  ports {
    internal = 80
    external = var.host_port
  }
}
```

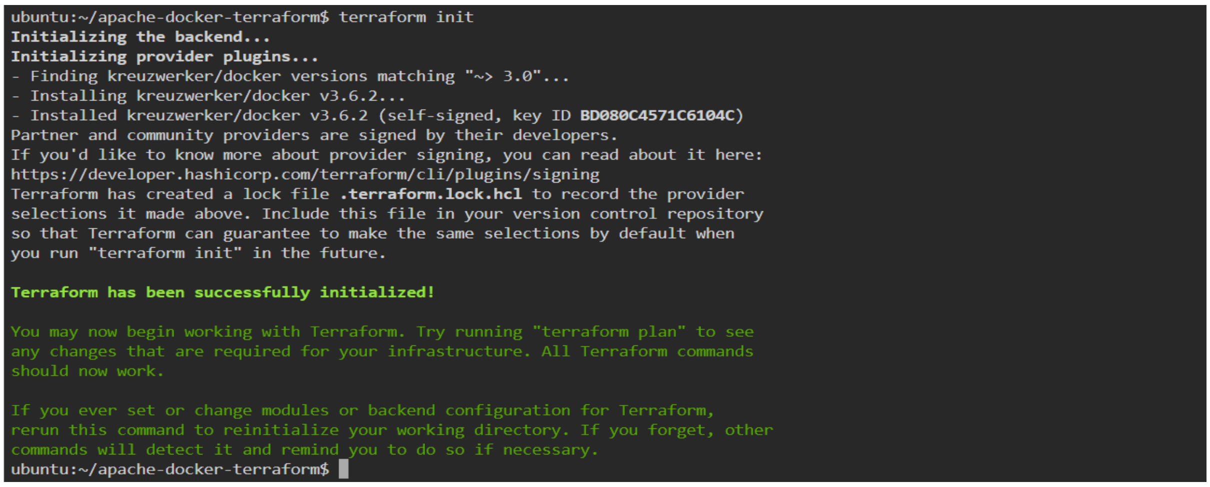
Step 6: Outputs

📄 outputs.tf

```
ubuntu:~/apache-docker-terraform$ vi outputs.tf
ubuntu:~/apache-docker-terraform$ cat outputs.tf
output "apache_url" {
  value = "http://localhost:${var.host_port}"
}
```

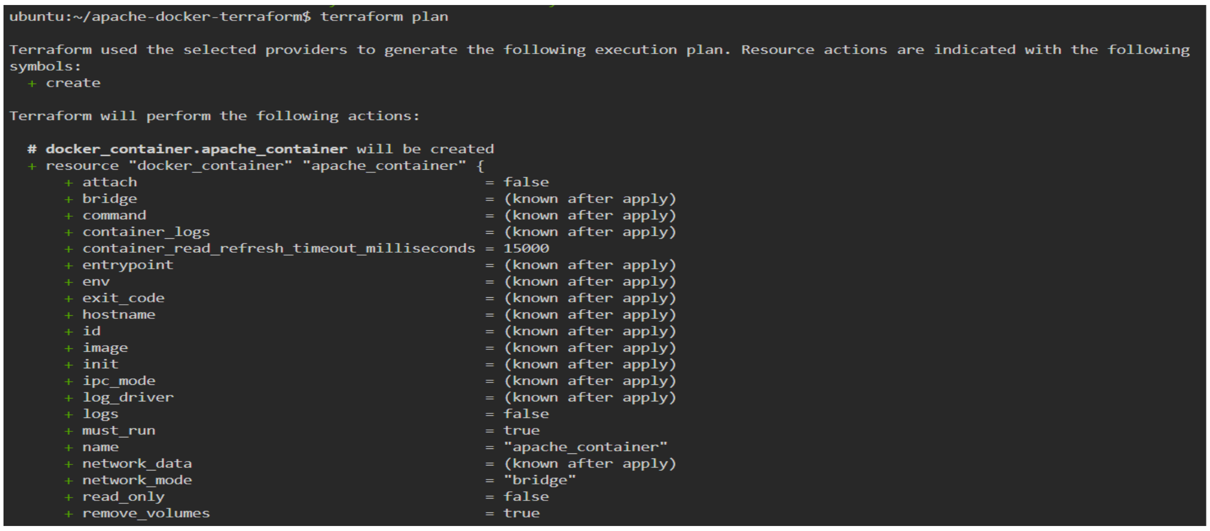
Step 7: Run the Project

- ### Initialize Terraform



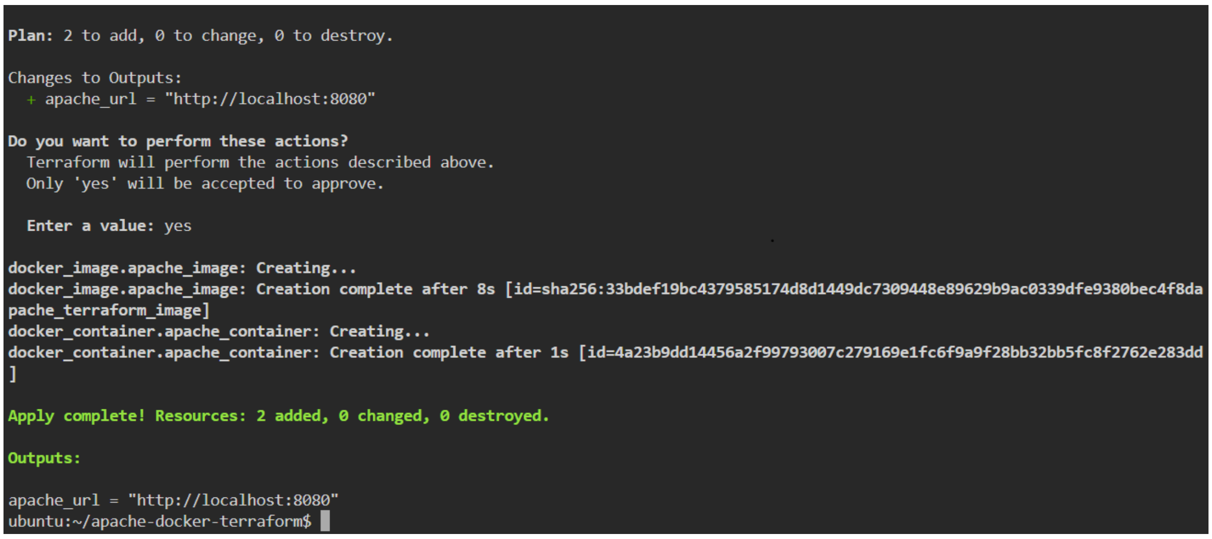
```
ubuntu:~/apache-docker-terraform$ terraform init
Initializing the backend...
Initializing provider plugins...
- Finding kreuzwerker/docker versions matching "~> 3.0"...
- Installing kreuzwerker/docker v3.6.2...
- Installed kreuzwerker/docker v3.6.2 (self-signed, key ID BD080C4571C6104C)
Partner and community providers are signed by their developers.
If you'd like to know more about provider signing, you can read about it here:
https://developer.hashicorp.com/terraform/cli/plugins/signing
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:~/apache-docker-terraform$
```

- ### Plan



```
ubuntu:~/apache-docker-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.apache_container will be created
  + resource "docker_container" "apache_container" {
      + attach                                      = false
      + bridge                                      = (known after apply)
      + command                                     = (known after apply)
      + container_logs                              = (known after apply)
      + container_read_refresh_timeout_milliseconds = 15000
      + entrypoint                                  = (known after apply)
      + env                                         = (known after apply)
      + exit_code                                   = (known after apply)
      + hostname                                    = (known after apply)
      + id                                          = (known after apply)
      + image                                       = (known after apply)
      + init                                        = (known after apply)
      + ipc_mode                                    = (known after apply)
      + log_driver                                  = (known after apply)
      + logs                                        = false
      + must_run                                    = true
      + name                                        = "apache_container"
      + network_data                                = (known after apply)
      + network_mode                                = "bridge"
      + read_only                                   = false
      + remove_volumes                              = true
```

- **Apply**

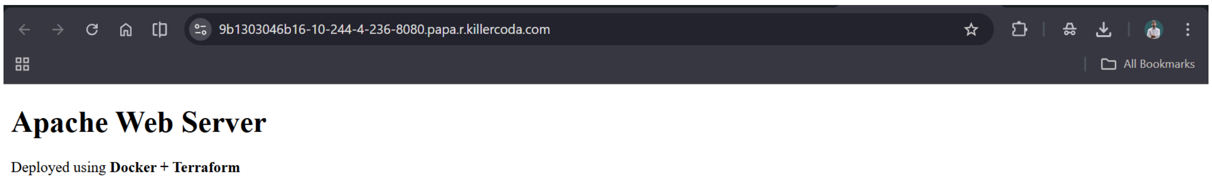


```
Plan: 2 to add, 0 to change, 0 to destroy.

Changes to Outputs:
  + apache_url = "http://localhost:8080"

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.apache_image: Creating...
docker_image.apache_image: Creation complete after 8s [id=sha256:33bdef19bc4379585174d8d1449dc7309448e89629b9ac0339dfe9380bec4f8da
pache_terraform_image]
docker_container.apache_container: Creating...
docker_container.apache_container: Creation complete after 1s [id=4a23b9dd14456a2f99793007c279169e1fc6f9a9f28bb32bb5fc8f2762e283dd
]

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

Outputs:

apache_url = "http://localhost:8080"
ubuntu:~/apache-docker-terraform$
```

Step 8: Access Apache

Open browser:

http://localhost:8080



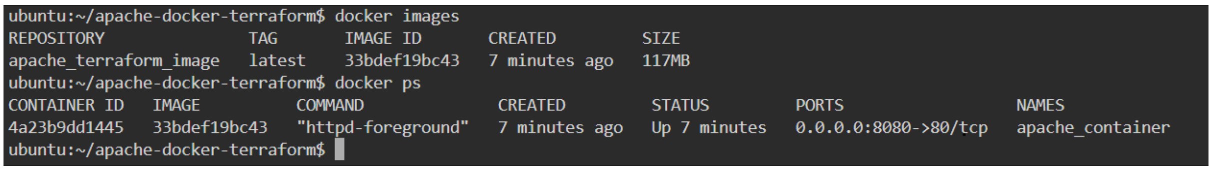
9b1303046b16-10-244-4-236-8080.papa.r.killercoda.com

All Bookmarks

Apache Web Server

Deployed using **Docker + Terraform**

🐳 Check Docker Images & Containers



```
ubuntu:~/apache-docker-terraform$ docker images
REPOSITORY             TAG       IMAGE ID       CREATED         SIZE
apache_terraform_image   latest    33bdef19bc43   7 minutes ago   117MB
ubuntu:~/apache-docker-terraform$ docker ps
CONTAINER ID   IMAGE          COMMAND              CREATED         STATUS         PORTS                  NAMES
4a23b9dd1445   33bdef19bc43   "httpd-foreground"   7 minutes ago   Up 7 minutes   0.0.0.0:8080->80/tcp   apache_container
ubuntu:~/apache-docker-terraform$
```

Successfully deployed an Apache Web Server using Docker containers managed through Terraform, implementing Infrastructure as Code (IaC) principles.

THANK YOU!

Kartik Akade