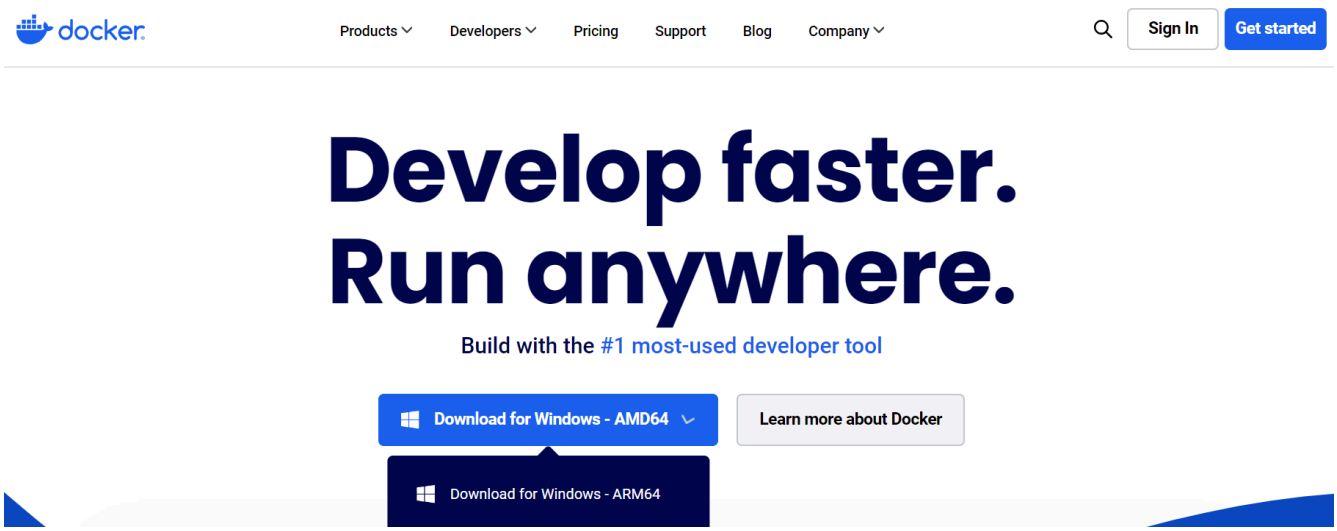
Name:Atharv Nikam **Div:D15C** **Roll No :36**

Exp 6

Aim : To Build, change, and destroy AWS / GCP /Microsoft Azure/ DigitalOcean infrastructure Using Terraform.(S3 bucket or Docker)

Step1: check if Docker is installed in your system to check open your powershell and type docker

If not then install docker from https://www.docker.com/ and download it



Check if docker has installed by opening command prompt and typing docker —version

```
Microsoft Windows [Version 10.0.22621.4037]
(c) Microsoft Corporation. All rights reserved.

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

C:\Users\athar>
```

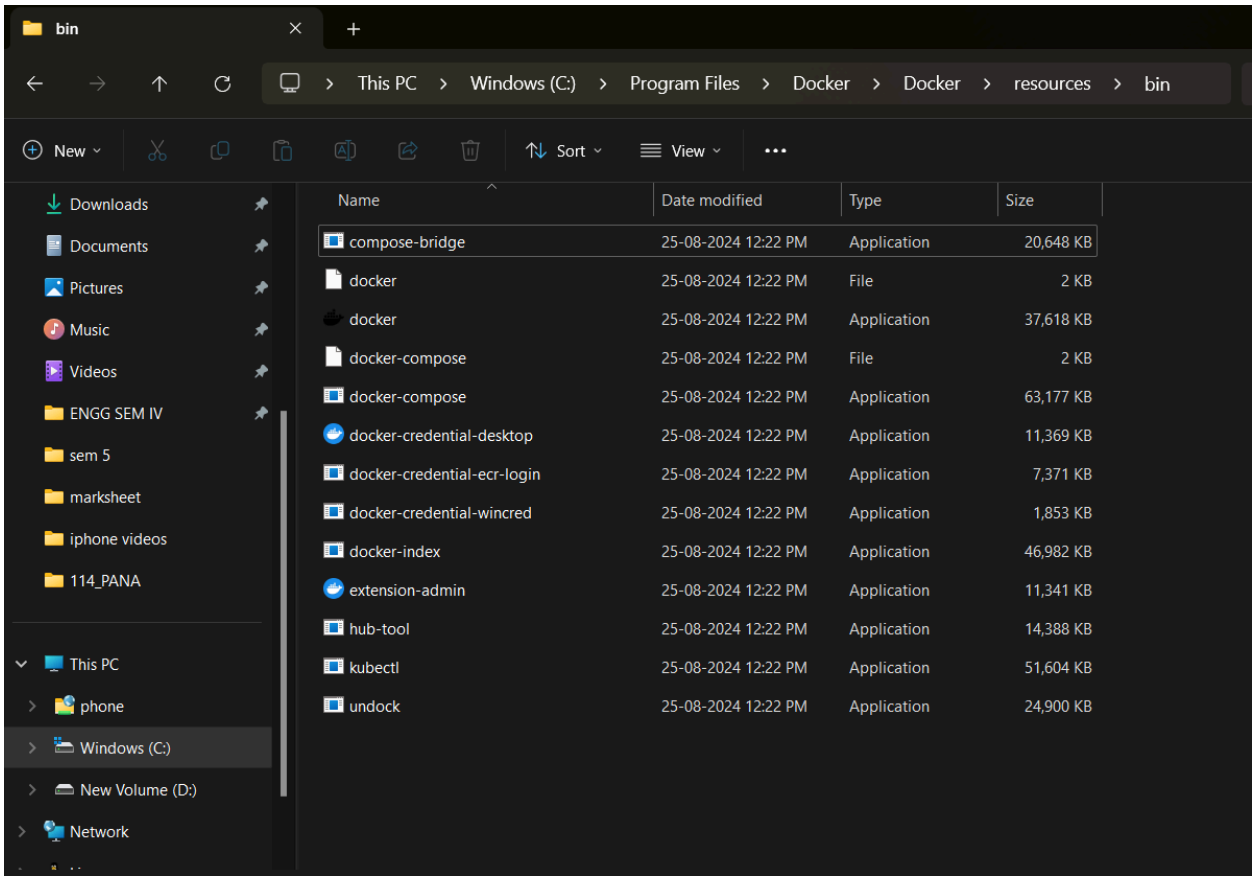
Name:Atharv Nikam **Div:D15C** **Roll No :36**

Type docker to see all the commands

```
C:\Users\athar>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
  checkpoint  Manage checkpoints
  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
  init*       Creates Docker-related starter files for your project
  manifest    Manage Docker image manifests and manifest lists
  network     Manage networks
  plugin      Manage plugins
  sbom*       View the packaged-based Software Bill Of Materials (SBOM) for an image
  scout*      Docker Scout
  system      Manage Docker
  trust       Manage trust on Docker images
  volume      Manage volumes

Swarm Commands:
  config      Manage Swarm configs
  node        Manage Swarm nodes
  secret      Manage Swarm secrets
  service     Manage Swarm services
  stack       Manage Swarm stacks
  swarm       Manage Swarm

Commands:
  attach      Attach local standard input, output, and error streams to a running container
  commit      Create a new image from a container's changes
  cp          Copy files/folders between a container and the local filesystem
  create      Create a new container
  diff        Inspect changes to files or directories on a container's filesystem
  events      Get real time events from the server
  export      Export a container's filesystem as a tar archive
  history     Show the history of an image
  import      Import the contents from a tarball to create a filesystem image
  inspect     Return low-level information on Docker objects
  kill        Kill one or more running containers
  load        Load an image from a tar archive or STDIN
  logs        Fetch the logs of a container
```

Name:Atharv Nikam **Div:D15C** **Roll No :36**

If you are not getting the version try these steps
Go to your file explorer and go to bin folder of your docker file or just paste this
C:\Program Files\Docker\Docker\resources\bin



Access the 'Edit the System Environment Variables' option on your computer. Then, select Environment Variables.

Look for a 'Path' variable under System variables. If it’s there, select it and click on Edit.

If it’s not there, click on New and create a 'Path' variable.
If the variable already exists, click on Edit and then on New to open a text box. Paste the path you copied into the box and click OK to close all the windows.

Now do step 1 again to check the docker version

Name:Atharv Nikam **Div:D15C** **Roll No :36**

Step 2: Create a folder Terraform scripts and then inside that create docker file and at last create a docker.ts file
After that open that folder in visual studio and paste this code

```
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_idname =
  "foo"
}
```

Name:Atharv Nikam **Div:D15C** **Roll No :36**

```
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"
}
```

Step 3:Open the terminal and go to folder where docker.tf is present

```
PS D:\all code\Terraform Scripts\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 D:\all code\Terraform Scripts\Docker>
```

Name:Atharv Nikam **Div:D15C** **Roll No :36**

Step 4:Run the Command ‘terraform plan’ This would create a execution plan and you could see an overview of your plan

```
PS D:\all code\Terraform Scripts\Docker> terraform plan

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


  Error: Error pinging Docker server: error during connect: Get "http://%2F%2F.%2F%2Fpipe%2F%2Fdocker_engine/_ping": open //.//pipe//docker_engine: The system cannot find the fil
e specified.

    with provider["registry.terraform.io/kreuzwerker/docker"],
    on docker.tf line 9, in provider "docker":
     9: provider "docker" {
```

Sometimes the docker engine os not working so on the docker Docker Desktops and try again

Step 5 :Run terraform plan again

```
PS D:\all code\Terraform Scripts\Docker> terraform plan

Terraform used the selected providers to generate the following execution p
  + create

Terraform will perform the following actions:

  # docker_container.foo will be created
  + resource "docker_container" "foo" {
      + 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)
      + 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
      + restart          = "no"
      + rm               = false
      + runtime          = (known after apply)
```

Name:Atharv Nikam **Div:D15C** **Roll No :36**

Step 6:run command terraform apply

```
docker_image.ubuntu: Creating...
docker_image.ubuntu: Still creating... [10s elapsed]
docker_image.ubuntu: Creation complete after 12s [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]
docker_container.foo: Creating...

Error: container exited immediately

  with docker_container.foo,
  on docker.tf line 17, in resource "docker_container" "foo":
  17: resource "docker_container" "foo" {
```

The script will give an error because this script took very less time to execute to resolve this issue we have to add this command

‘command=[“sleep”,”infinity”]’
Now we will get this

```
PS D:\all code\Terraform Scripts\Docker> terraform apply
docker_image.ubuntu: Refreshing state... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6

Terraform used the selected providers to generate the following execution plan. Resource actions are indi
  + 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            = "sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598a"
      + 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
      + restart          = "no"
      + rm               = false
      + runtime          = (known after apply)
      + security_opts    = (known after apply)
      + shm_size         = (known after apply)
      + start            = true
```

Name:Atharv Nikam **Div:D15C** **Roll No :36**

```
Plan: 1 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_container.foo: Creating...
docker_container.foo: Creation complete after 0s [id=305bdd50733837ad82692315d99bfd8178934f2e5ce0c8407d4744644d11d8ad]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS D:\all code\Terraform Scripts\Docker>
```

Docker images before terraform apply

```
PS D:\all code\Terraform Scripts\Docker> docker images
REPOSITORY    TAG       IMAGE ID    CREATED    SIZE
PS D:\all code\Terraform Scripts\Docker>
```

Docker images after terraform apply

```
PS D:\all code\Terraform Scripts\Docker> docker images
REPOSITORY    TAG       IMAGE ID       CREATED       SIZE
ubuntu        latest    edbfe74c41f8   3 weeks ago   78.1MB
PS D:\all code\Terraform Scripts\Docker>
```

Step 7: Now the image is created to destroy that we have to use the command Terraform destroy

```
PS D:\all code\Terraform Scripts\Docker> terraform destroy
docker_image.ubuntu: Refreshing state... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]
docker_container.foo: Refreshing state... [id=305bdd50733837ad82692315d99bfd8178934f2e5ce0c8407d4744644d11d8ad]

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          = "305bdd507338" -> null
      - id                = "305bdd50733837ad82692315d99bfd8178934f2e5ce0c8407d4744644d11d8ad" -> 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
```

Name:Atharv Nikam **Div:D15C** **Roll No :36**

Enter yes

```
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=305bdd50733837ad82692315d99bfd8178934f2e5ce0c8407d4744644d11d8ad]
docker_container.foo: Destruction complete after 0s
docker_image.ubuntu: Destroying... [id=sha256:edbfe74c41f8a3501ce542e137cf28ea04dd03e6df8c9d66519b6ad761c2598aubuntu:latest]
docker_image.ubuntu: Destruction complete after 1s

Destroy complete! Resources: 2 destroyed.
PS D:\all code\Terraform Scripts\Docker>
```

To check if the images are destroyed check run docker images

```
PS D:\all code\Terraform Scripts\Docker> docker images
REPOSITORY    TAG       IMAGE ID    CREATED    SIZE
PS D:\all code\Terraform Scripts\Docker>
```