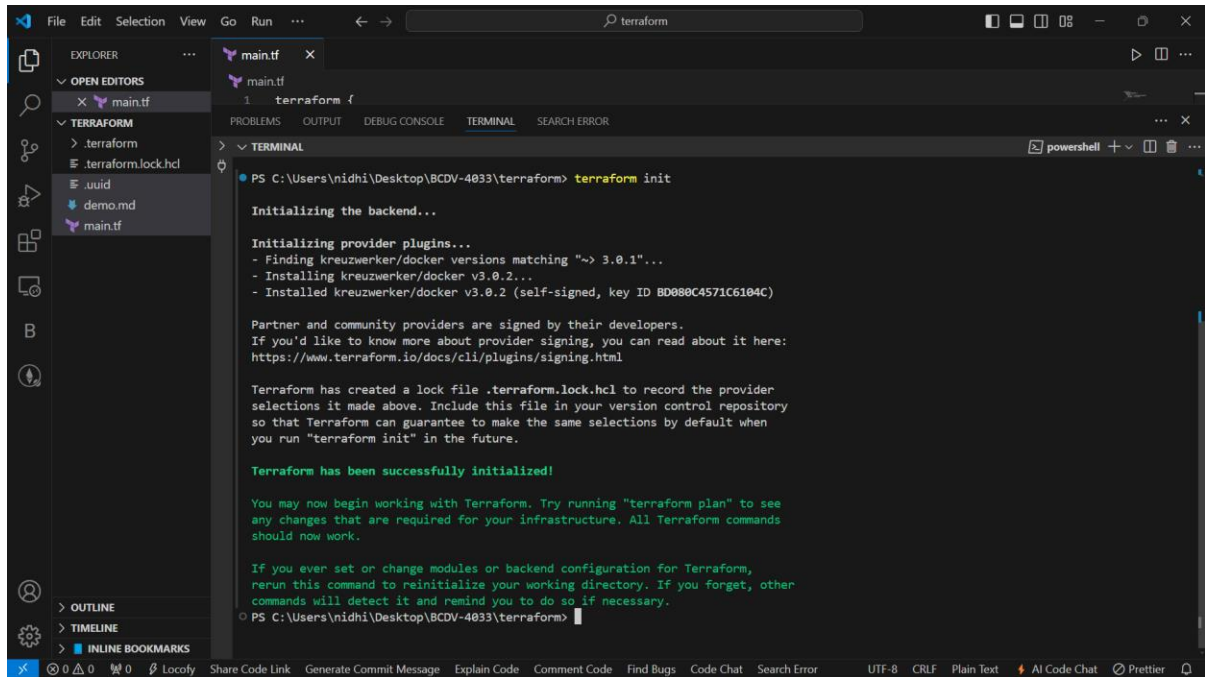


101527329 Nidhi Nakrani

lab 4

## Install Terraform in your local system.

Create a tf configuration for running docker resource. You can take help from this



```
PS C:\Users\nidhi\Desktop\BCDV-4033\terraform> terraform init

Initializing the backend...

Initializing provider plugins...
- Finding kreuzwerker/docker versions matching "~> 3.0.1"...
- Installing kreuzwerker/docker v3.0.2...
- Installed kreuzwerker/docker v3.0.2 (self-signed, key ID BD080C4571C6104C)

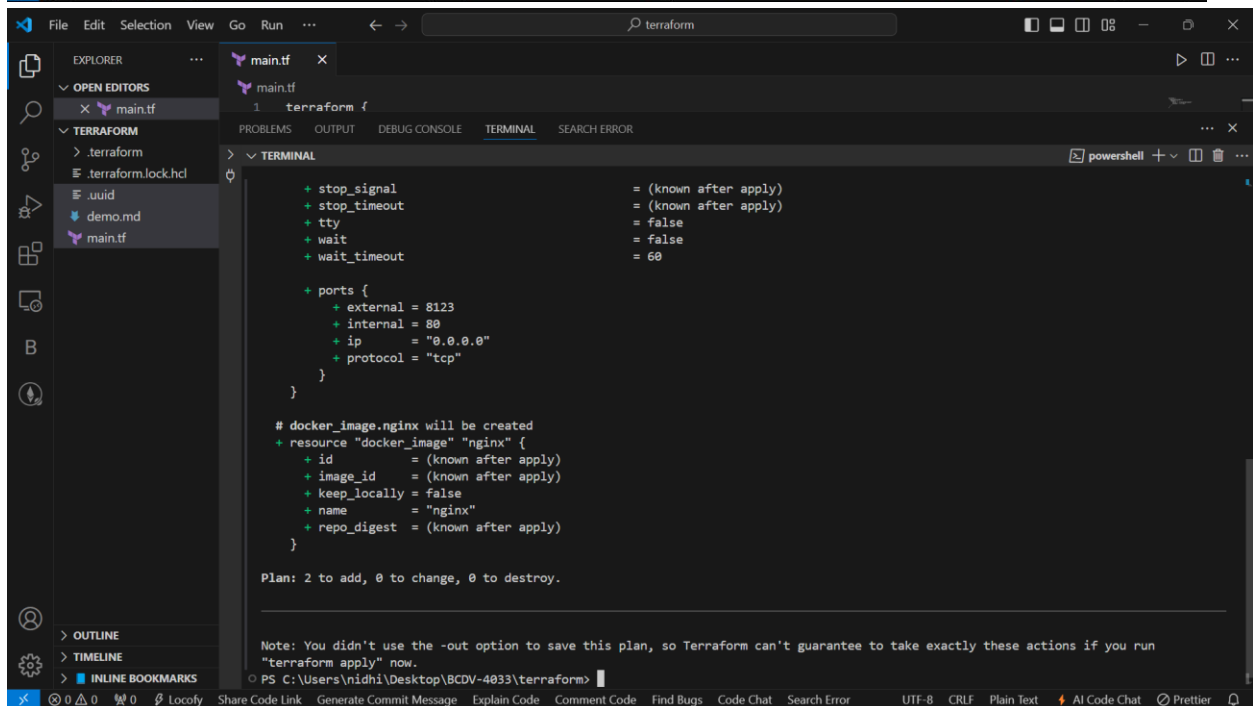
Partner and community providers are signed by their developers.
If you'd like to know more about provider signing, you can read about it here:
https://www.terraform.io/docs/cli/plugins/signing.html

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\nidhi\Desktop\BCDV-4033\terraform>
```



```
+ stop_signal           = (known after apply)
+ stop_timeout         = (known after apply)
+ tty                  = false
+ wait                 = false
+ wait_timeout         = 60

+ ports {
+   external = 8123
+   internal = 80
+   ip       = "0.0.0.0"
+   protocol = "tcp"
+ }

# docker_image.nginx will be created
+ resource "docker_image" "nginx" {
+   id           = (known after apply)
+   image_id     = (known after apply)
+   keep_locally = false
+   name         = "nginx"
+   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\nidhi\Desktop\BCDV-4033\terraform>
```

101527329 Nidhi Nakrani

lab 4

The image displays two screenshots of a Visual Studio Code editor interface, showing the process of applying a Terraform configuration to create a Docker container.

**Top Screenshot:** The editor shows a file named `main.tf` with the following Terraform configuration:

```
terraform {  
  + shm_size           = (known after apply)  
  + start              = true  
  + stdin_open         = false  
  + stop_signal        = (known after apply)  
  + stop_timeout       = (known after apply)  
  + tty                = false  
  + wait               = false  
  + wait_timeout       = 60  
  
  + ports {  
    + external = 8123  
    + internal = 80  
    + ip       = "0.0.0.0"  
    + protocol = "tcp"  
  }  
}
```

The terminal output shows the Terraform plan and the confirmation to apply the configuration:

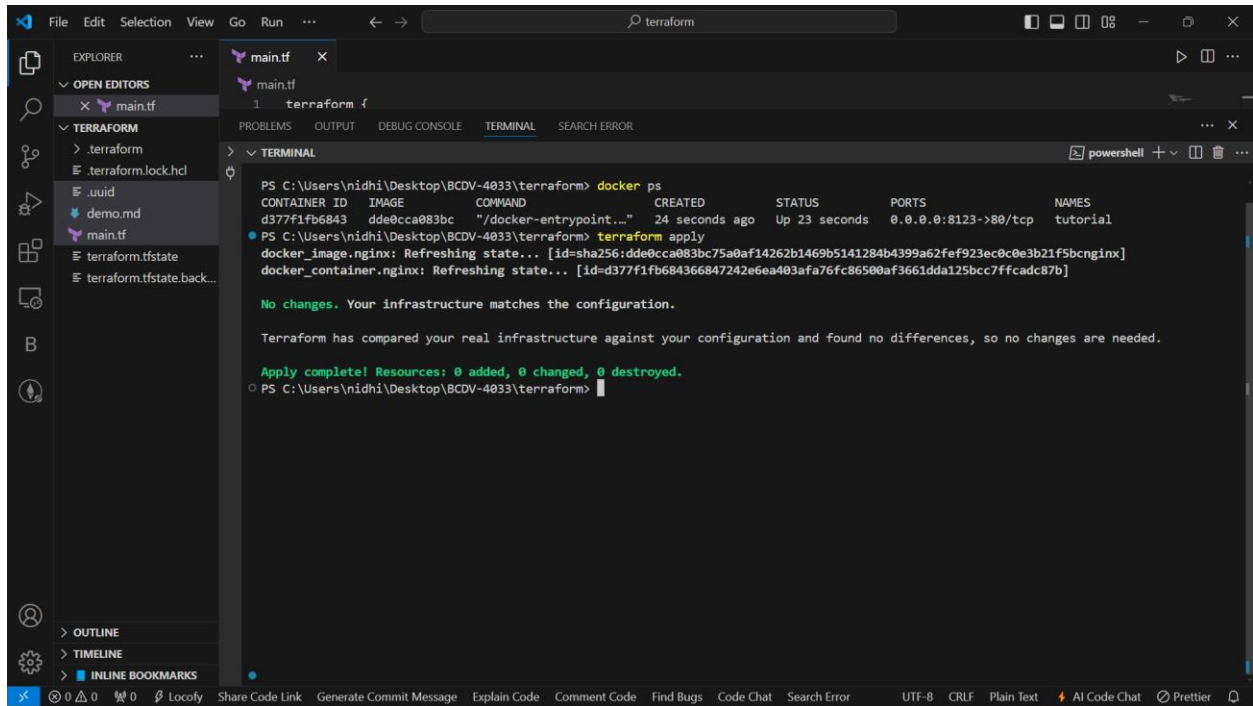
```
Plan: 1 to add, 0 to change, 0 to destroy.  
  
Do you want to perform these actions?  
Terraform will perform the actions described above.  
Only 'yes' will be accepted to approve.  
  
Enter a value: yes  
  
docker_container.nginx: Creating...  
docker_container.nginx: Creation complete after 4s [id=d377f1fb684366847242e6ea403afa76fc86500af3661dda125bcc7ffcad87b]  
  
Apply complete! Resources: 1 added, 0 changed, 0 destroyed.  
PS C:\Users\nidhi\Desktop\BCDV-4033\terraform>
```

**Bottom Screenshot:** The terminal output shows the Docker container running successfully:

```
PS C:\Users\nidhi\Desktop\BCDV-4033\terraform> docker ps  
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS                               NAMES  
d377f1fb6843   dde0cca083bc "/docker-entrypoint..." 24 seconds ago Up 23 seconds 0.0.0.0:8123->80/tcp          tutorial
```

101527329 Nidhi Nakrani  
lab 4

## Apply the initial resources.



The screenshot shows the Visual Studio Code interface with a file explorer on the left and a terminal window at the bottom. The file explorer shows a project structure with files like `main.tf`, `terraform.lock.hcl`, `terraform.tfstate`, and `terraform.tfstate.backup`. The terminal window displays the output of the `terraform apply` command, showing that the infrastructure matches the configuration and no changes are needed.

```
PS C:\Users\nidhi\Desktop\BCDV-4033\terraform> docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED         STATUS         PORTS           NAMES
d377f1fb6843   dde0cca083bc  "/docker-entrypoint..."  24 seconds ago  Up 23 seconds  0.0.0.0:8123->80/tcp  tutorial

PS C:\Users\nidhi\Desktop\BCDV-4033\terraform> terraform apply
docker_image.nginx: Refreshing state... [id=sha256:dde0cca083bc75a0af14262b1469b5141284b4399a62fef923ec0c0e3b21f5bcnginx]
docker_container.nginx: Refreshing state... [id=d377f1fb684366847242e6ea403afa76fc86500af3661dda125bcc7ffcad87b]

No changes. Your infrastructure matches the configuration.

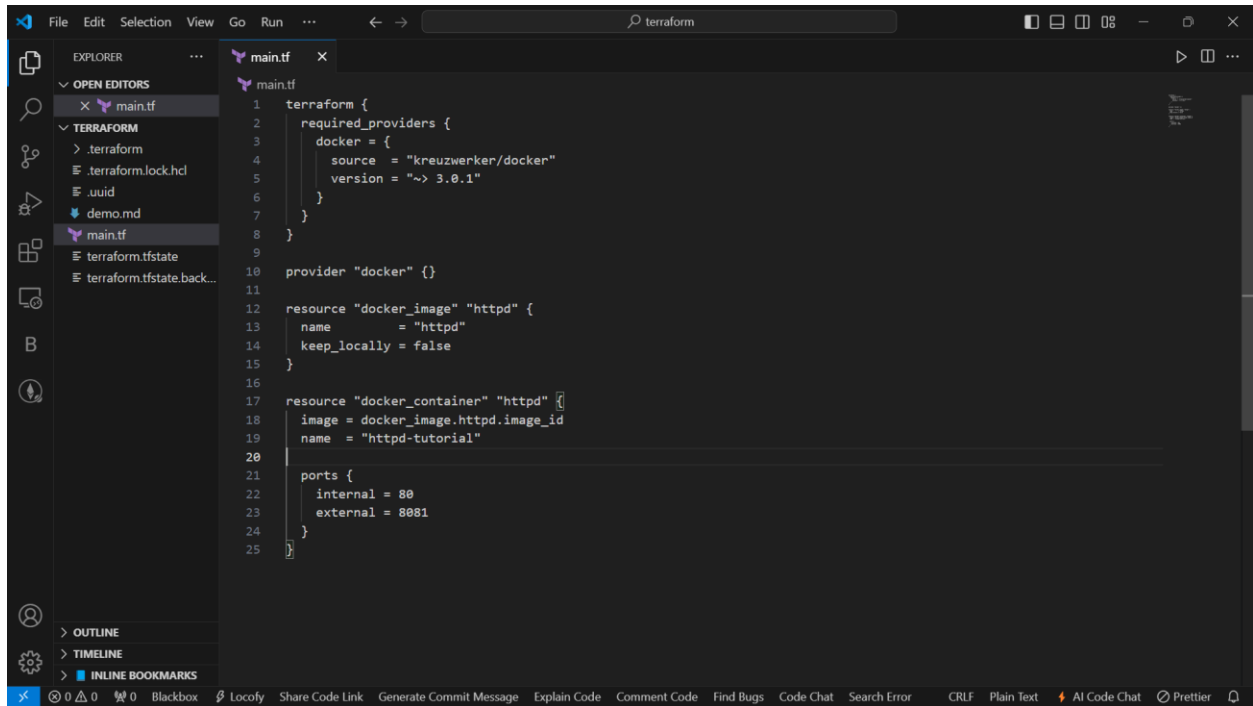
Terraform has compared your real infrastructure against your configuration and found no differences, so no changes are needed.

Apply complete! Resources: 0 added, 0 changed, 0 destroyed.
PS C:\Users\nidhi\Desktop\BCDV-4033\terraform>
```

101527329 Nidhi Nakrani

lab 4

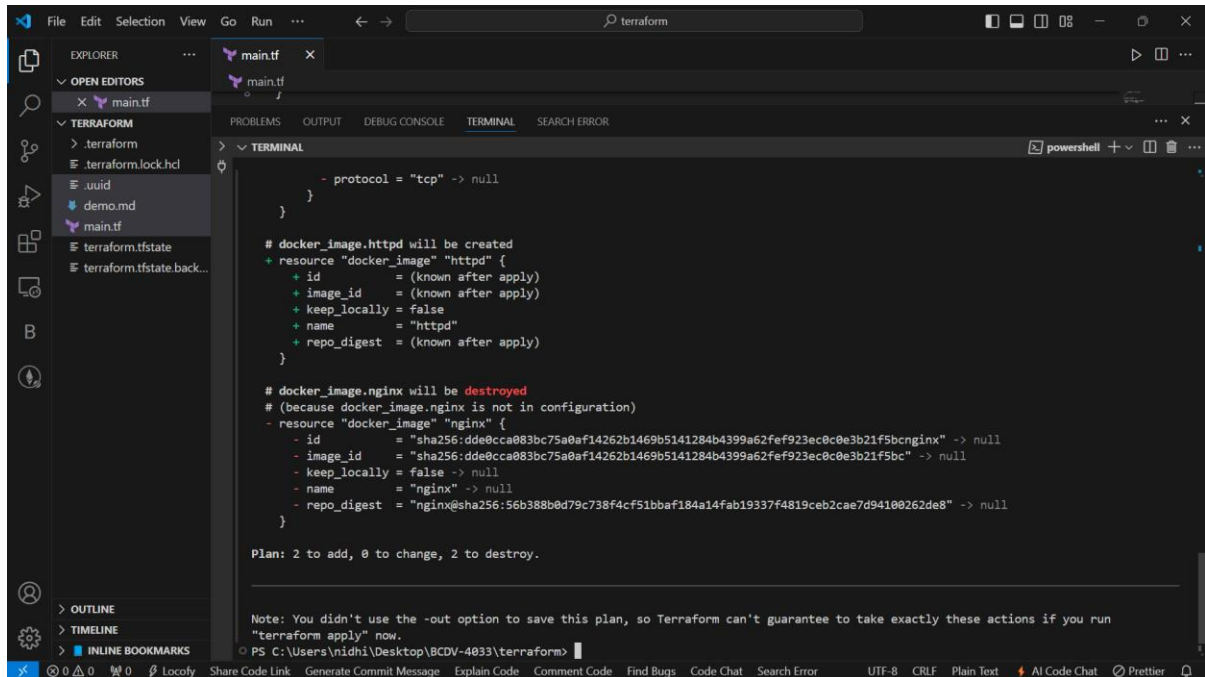
**Change infrastructure (your choice of what change it is).**



```
1 terraform {
2   required_providers {
3     docker = {
4       source = "kreuzwerker/docker"
5       version = "~> 3.0.1"
6     }
7   }
8 }
9
10 provider "docker" {}
11
12 resource "docker_image" "httpd" {
13   name = "httpd"
14   keep_locally = false
15 }
16
17 resource "docker_container" "httpd" {
18   image = docker_image.httpd.image_id
19   name = "httpd-tutorial"
20
21   ports {
22     internal = 80
23     external = 8081
24   }
25 }
```

101527329 Nidhi Nakrani  
lab 4

## Create a plan and use the plan to make changes to the resource.



The screenshot shows the VS Code interface with a file explorer on the left containing files like .terraform, .terraform.lock.hcl, .uid, demo.md, main.tf, terraform.tfstate, and terraform.tfstate.back... The main editor displays the main.tf file with Terraform configuration for a docker\_image resource. The configuration includes a resource 'docker\_image.httpd' and a resource 'docker\_image.nginx' that is marked for destruction. The terminal window shows the output of the terraform plan command, indicating that 2 resources will be added, 0 changed, and 2 destroyed. A note at the bottom of the terminal states: '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.'

```
main.tf
main.tf

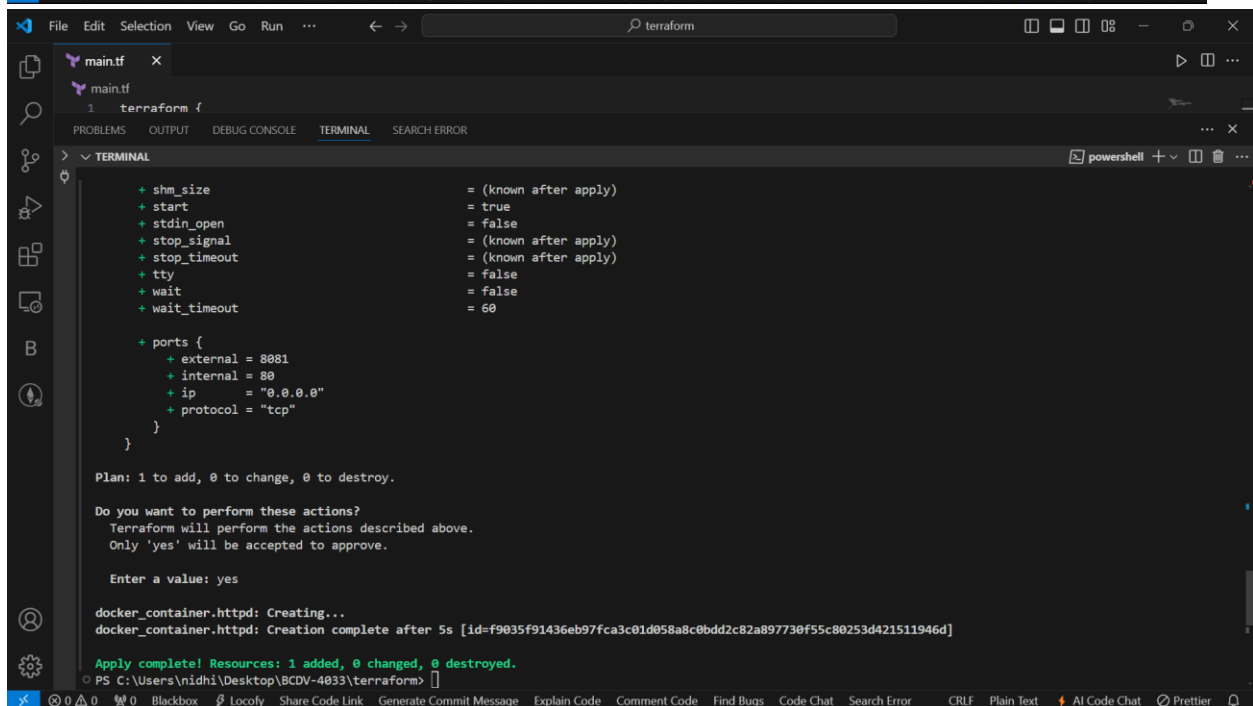
- protocol = "tcp" -> null
}

# docker_image.httpd will be created
+ resource "docker_image" "httpd" {
+   id           = (known after apply)
+   image_id     = (known after apply)
+   keep_locally = false
+   name        = "httpd"
+   repo_digest = (known after apply)
}

# docker_image.nginx will be destroyed
# (because docker_image.nginx is not in configuration)
- resource "docker_image" "nginx" {
-   id           = "sha256:dde0cca883bc75a0af14262b1469b5141284b4399a62fef923ec0c0e3b21f5bcnginx" -> null
-   image_id     = "sha256:dde0cca883bc75a0af14262b1469b5141284b4399a62fef923ec0c0e3b21f5bc" -> null
-   keep_locally = false -> null
-   name        = "nginx" -> null
-   repo_digest = "nginx@sha256:56b388b8d79c738f4cf51bbaf184a14fab19337f4819ceb2cae7d94100262de8" -> null
}

Plan: 2 to add, 0 to change, 2 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\nidhi\Desktop\BCDV-4033\terraform>
```



The screenshot shows the VS Code interface with the main.tf file open. The terminal window displays the output of the terraform apply command. It shows the creation of the docker\_container.httpd resource, including details like shm\_size, start, stdin\_open, stop\_signal, stop\_timeout, tty, wait, and wait\_timeout. The ports section is also shown with external, internal, ip, and protocol values. The terminal output indicates that the apply was successful, with 1 resource added, 0 changed, and 0 destroyed. The final output shows the creation of the docker\_container.httpd resource and the completion of the apply process.

```
main.tf
main.tf

1 terraform {
+   shm_size           = (known after apply)
+   start              = true
+   stdin_open         = false
+   stop_signal        = (known after apply)
+   stop_timeout       = (known after apply)
+   tty                = false
+   wait               = false
+   wait_timeout       = 60
+   ports {
+     external = 8081
+     internal = 80
+     ip       = "0.0.0.0"
+     protocol = "tcp"
+   }
}

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.httpd: Creating...
docker_container.httpd: Creation complete after 5s [id=f9035f91436eb97fca3c01d058a8c0bdd2c82a897730f55c80253d421511946d]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.
PS C:\Users\nidhi\Desktop\BCDV-4033\terraform>
```

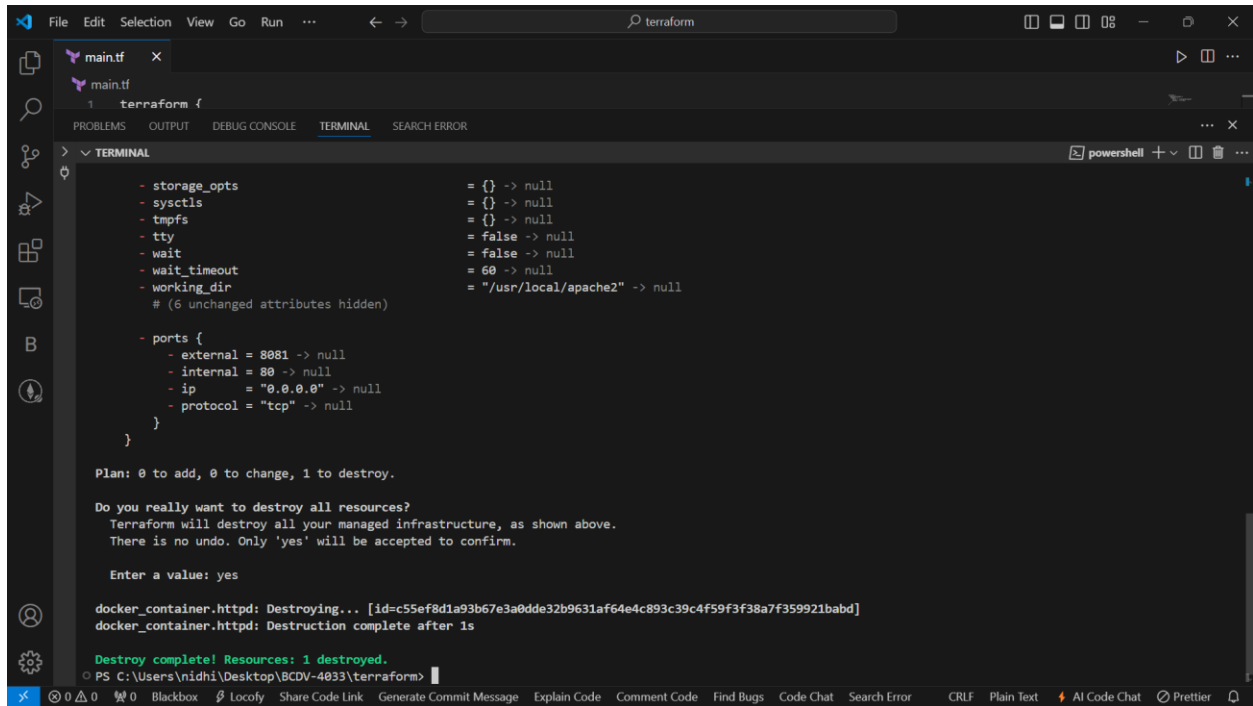
lab 4

The image shows a VS Code editor window with a Terraform configuration file named `main.tf`. The configuration defines a `docker_image` resource named `httpd` and a `docker_container` resource named `httpd`. The `docker_image` resource is configured with the `source` `"kreuzwerker/docker"` and `version` `"~> 3.0.1"`. The `docker_container` resource is configured with `image` `"httpd"`, `name` `"httpd-tutorial"`, and `ports` `{ internal = 80, external = 8081 }`.

The terminal output shows the execution of the `terraform apply` command. It displays the plan for adding the `docker_image.httpd` resource and the `docker_container.httpd` resource. The output indicates that the `docker_image.httpd` resource is destroyed, and the `docker_container.httpd` resource is created. The final output shows the resources created and destroyed.

101527329 Nidhi Nakrani  
lab 4

## Destroy the complete resource.



The screenshot shows the Visual Studio Code editor with a file named `main.tf` open. The file contains a Terraform configuration for a Docker container. The terminal window shows the execution of the `terraform destroy` command, which prompts for confirmation to destroy all resources. The user enters 'yes', and the terminal shows the destruction of the `docker_container.httpd` resource, completing after 1s.

```
main.tf
1 terraform {
  storage_opts = {} -> null
  sysctlfs = {} -> null
  tmpfs = {} -> null
  tty = false -> null
  wait = false -> null
  wait_timeout = 60 -> null
  working_dir = "/usr/local/apache2" -> null
  # (6 unchanged attributes hidden)

  ports {
    external = 8081 -> null
    internal = 80 -> null
    ip = "0.0.0.0" -> null
    protocol = "tcp" -> null
  }
}

Plan: 0 to add, 0 to change, 1 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

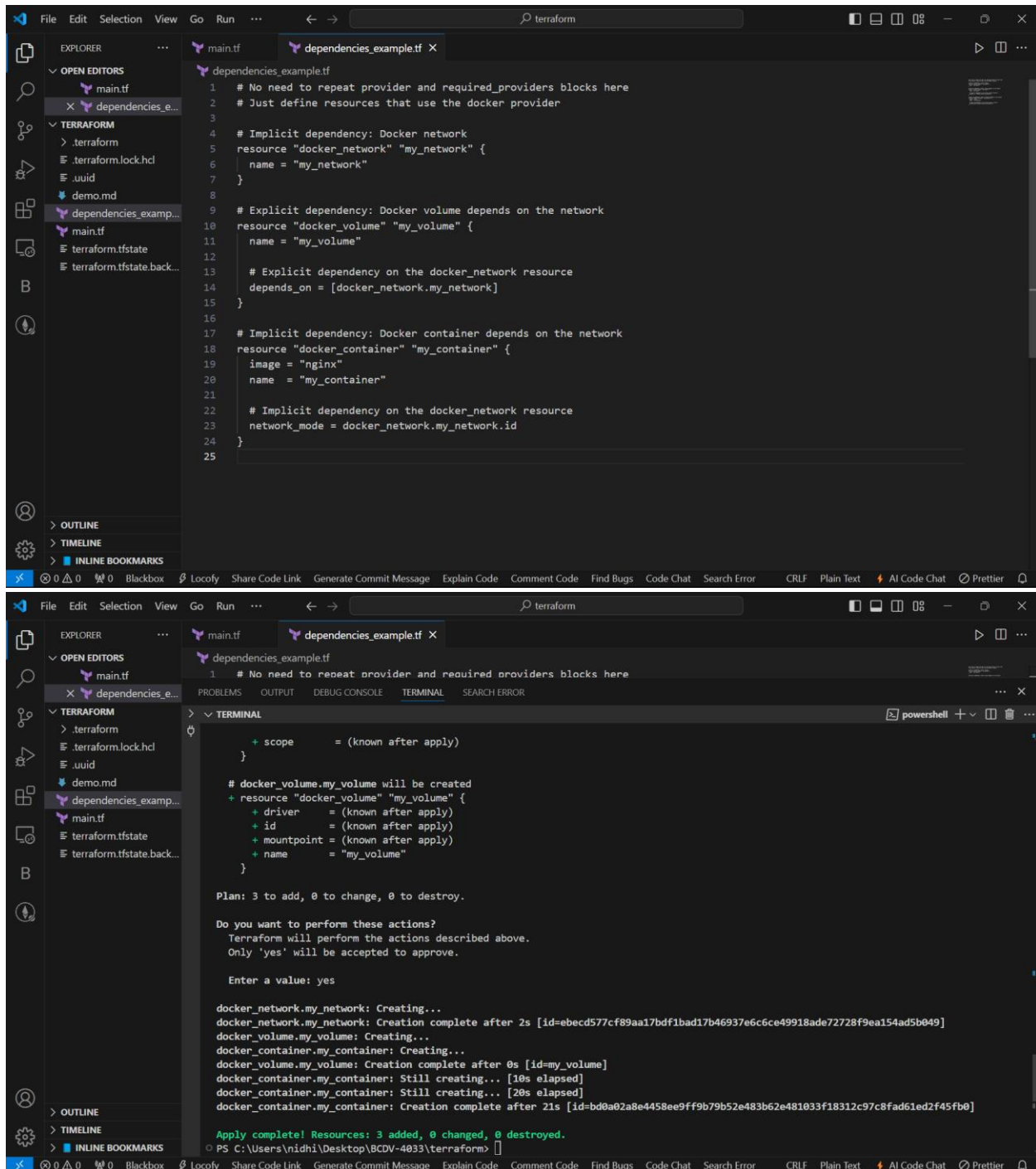
docker_container.httpd: Destroying... [id=c55ef8d1a93b67e3a0dde32b9631af64e4c893c39c4f59f3f38a7f359921babd]
docker_container.httpd: Destruction complete after 1s

Destroy complete! Resources: 1 destroyed.
PS C:\Users\nidhi\Desktop\BCDV-4033\terraform>
```

101527329 Nidhi Nakrani

lab 4

## Create resource with dependencies (implicit and explicit).



The image displays two screenshots of a Visual Studio Code editor window, illustrating the creation of Terraform resources with dependencies.

**Top Screenshot: Terraform Code**

The code is in a file named `dependencies_example.tf`. It defines three resources: `docker_network`, `docker_volume`, and `docker_container`.

```
1 # No need to repeat provider and required_providers blocks here
2 # Just define resources that use the docker provider
3
4 # Implicit dependency: Docker network
5 resource "docker_network" "my_network" {
6   name = "my_network"
7 }
8
9 # Explicit dependency: Docker volume depends on the network
10 resource "docker_volume" "my_volume" {
11   name = "my_volume"
12
13   # Explicit dependency on the docker_network resource
14   depends_on = [docker_network.my_network]
15 }
16
17 # Implicit dependency: Docker container depends on the network
18 resource "docker_container" "my_container" {
19   image = "nginx"
20   name = "my_container"
21
22   # Implicit dependency on the docker_network resource
23   network_mode = docker_network.my_network.id
24 }
25
```

**Bottom Screenshot: Terraform Execution Output**

The bottom screenshot shows the output of the Terraform command in the terminal. It displays the plan and the execution of the resources, confirming the dependencies.

```

+ scope      = (known after apply)
}

# docker_volume.my_volume will be created
+ resource "docker_volume" "my_volume" {
+   driver      = (known after apply)
+   id          = (known after apply)
+   mountpoint  = (known after apply)
+   name        = "my_volume"
}

Plan: 3 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_network.my_network: Creating...
docker_network.my_network: Creation complete after 2s [id=ebecd577cf89aa17bdf1bad17b46937e6c6ce49918ade72728f9ea154ad5b049]
docker_volume.my_volume: Creating...
docker_container.my_container: Creating...
docker_volume.my_volume: Creation complete after 0s [id=my_volume]
docker_container.my_container: Still creating... [10s elapsed]
docker_container.my_container: Still creating... [20s elapsed]
docker_container.my_container: Creation complete after 21s [id=bd0a02a8e4458ee9ff9b79b52e483b62e481033f18312c97c8fad61ed2f45fb0]

Apply complete! Resources: 3 added, 0 changed, 0 destroyed.
PS C:\Users\nidhi\Desktop\BCDV-4033\terraform>

```