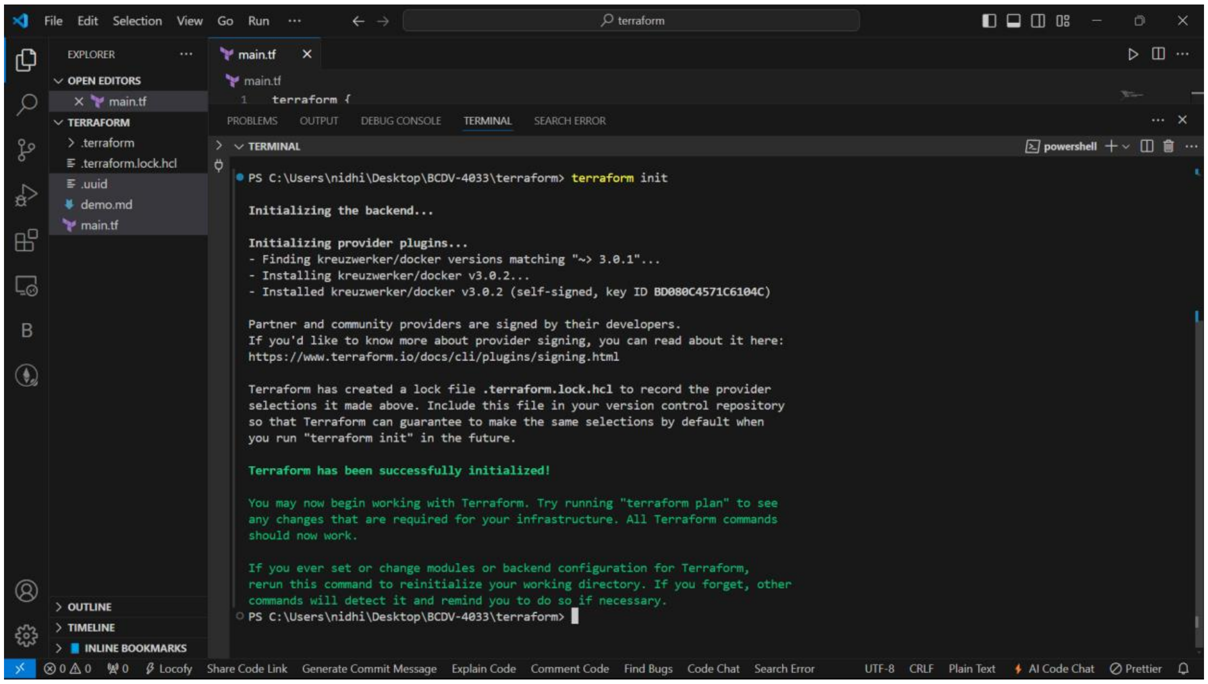
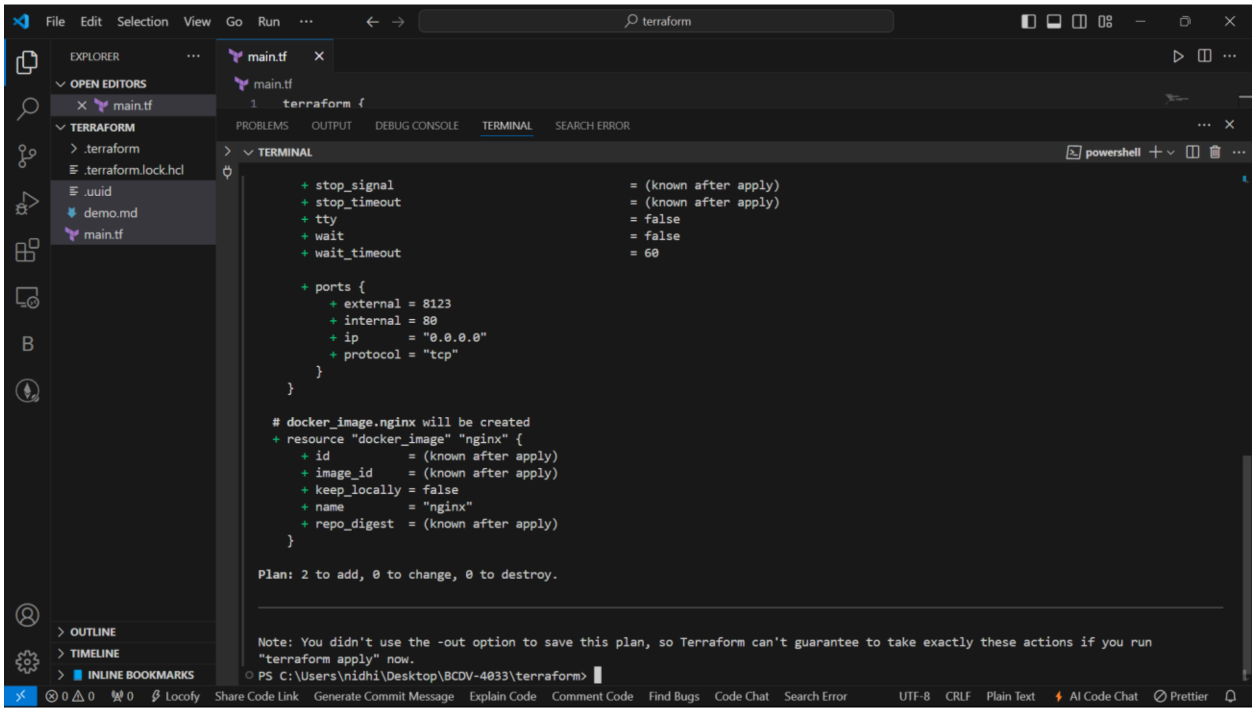
101527329   Nidhi Nakrani
lab 4

Install Terraform in your local system.
Create a tf configuration for running docker resource. You can take help from this



```
terraform {
```

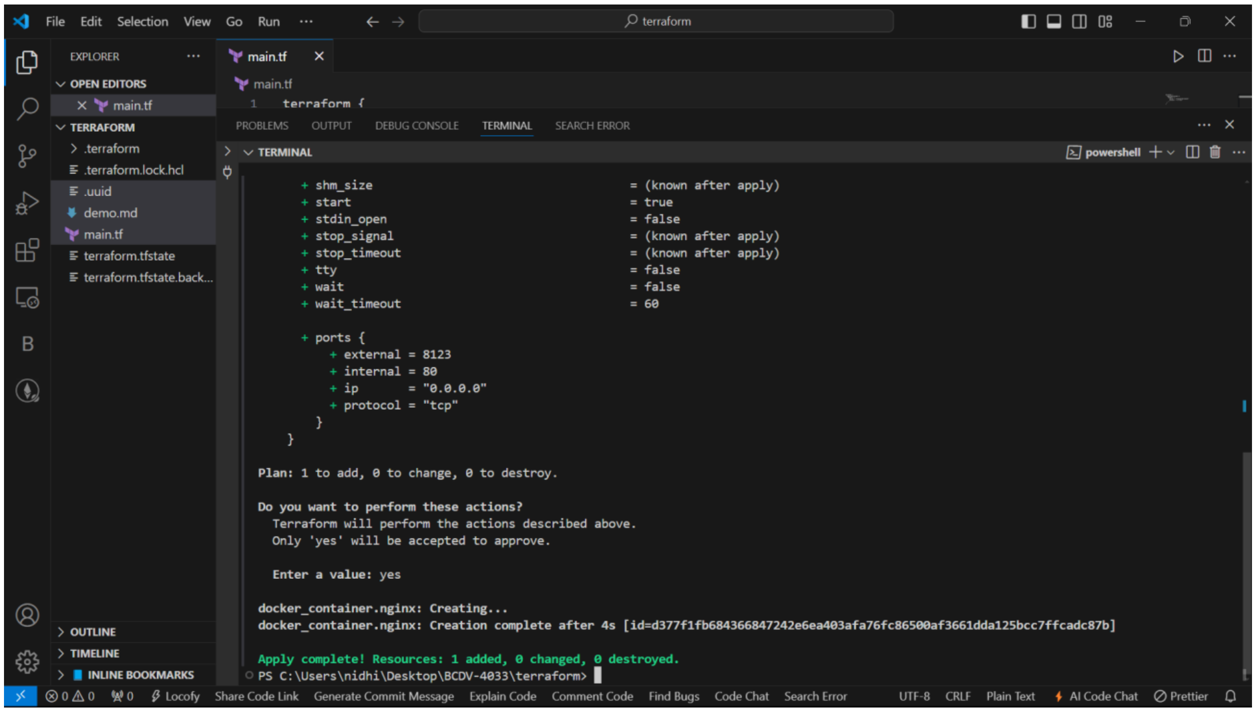
```
PS C:\Users\nidhi\Desktop\BCDV-4033\terraform> terraform init

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>
```



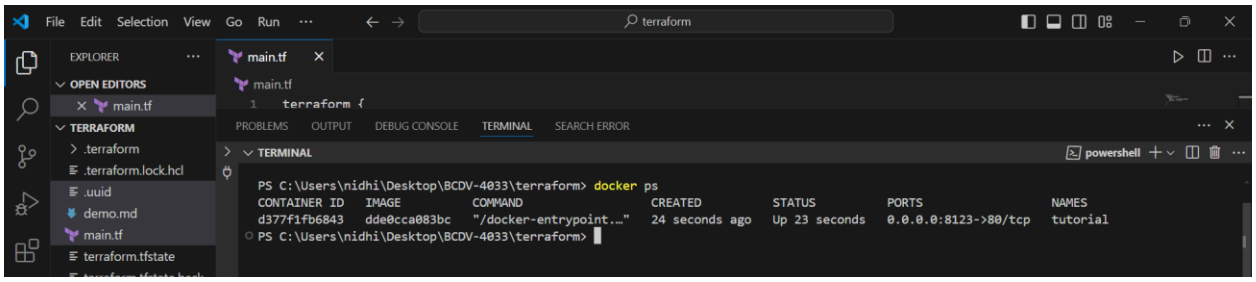
```
terraform {
```

```
      + 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



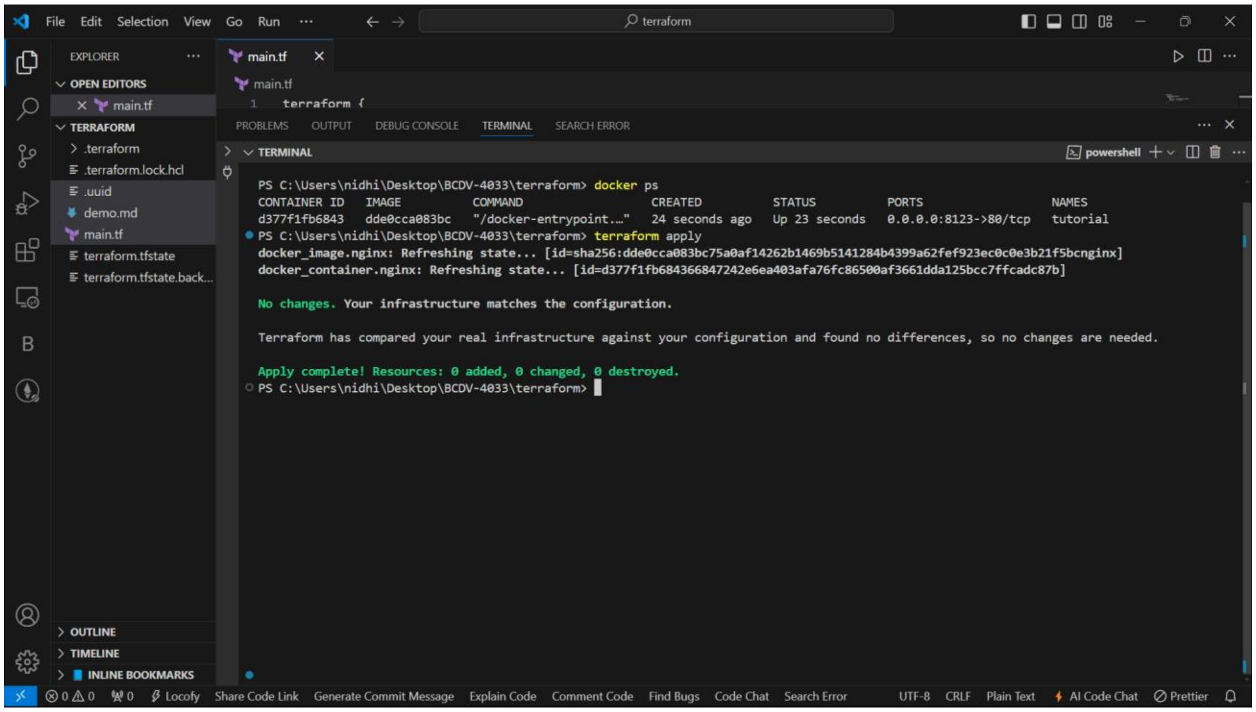
```
        + shm_size                                   = (known after apply)
        + start                                      = true
        + stdin_open                                 = false
        + stop_signal                                = (known after apply)
        + stop_timeout                               = (known after apply)
        + tty                                        = false
        + wait                                       = false
        + wait_timeout                               = 60

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

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=d377f1fb684366847242e6ea403afa76fc86500af3661dda125bcc7ffcadc87b]

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



```
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>
```

101527329 Nidhi Nakrani
lab 4

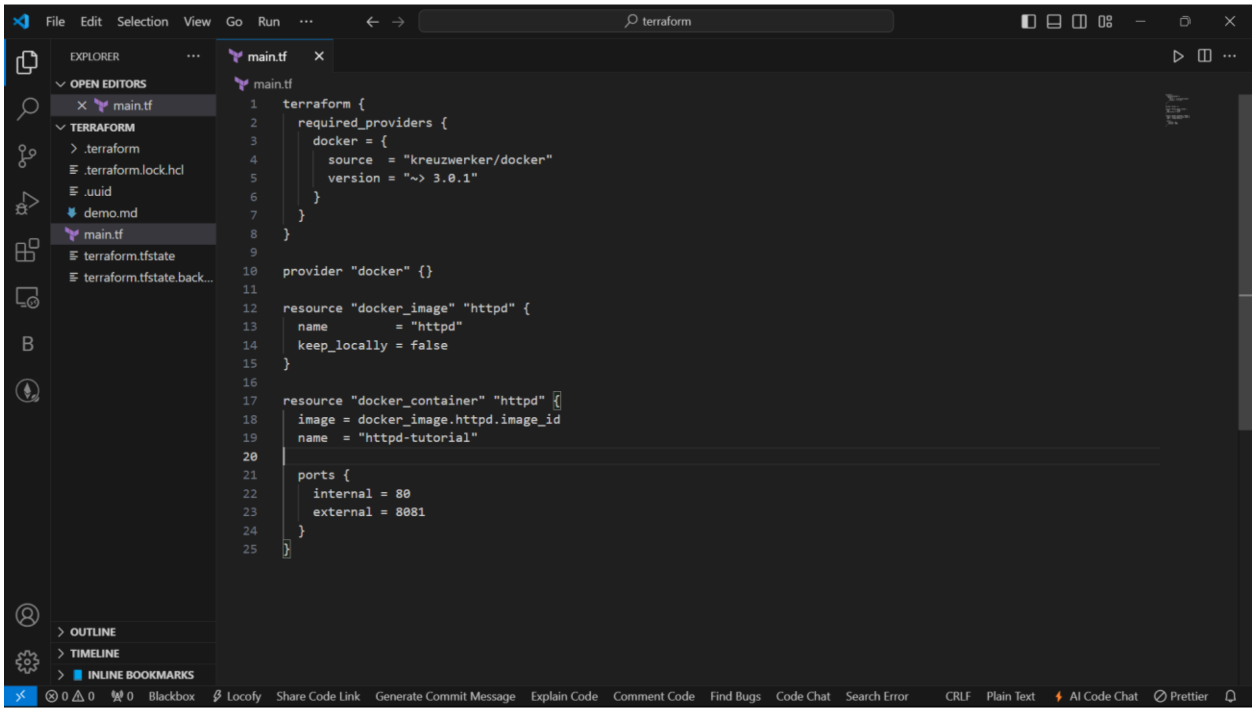
Apply the initial resources.



```
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=d377f1fb684366847242e6ea403afa76fc86500af3661dda125bcc7ffcadc87b]

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

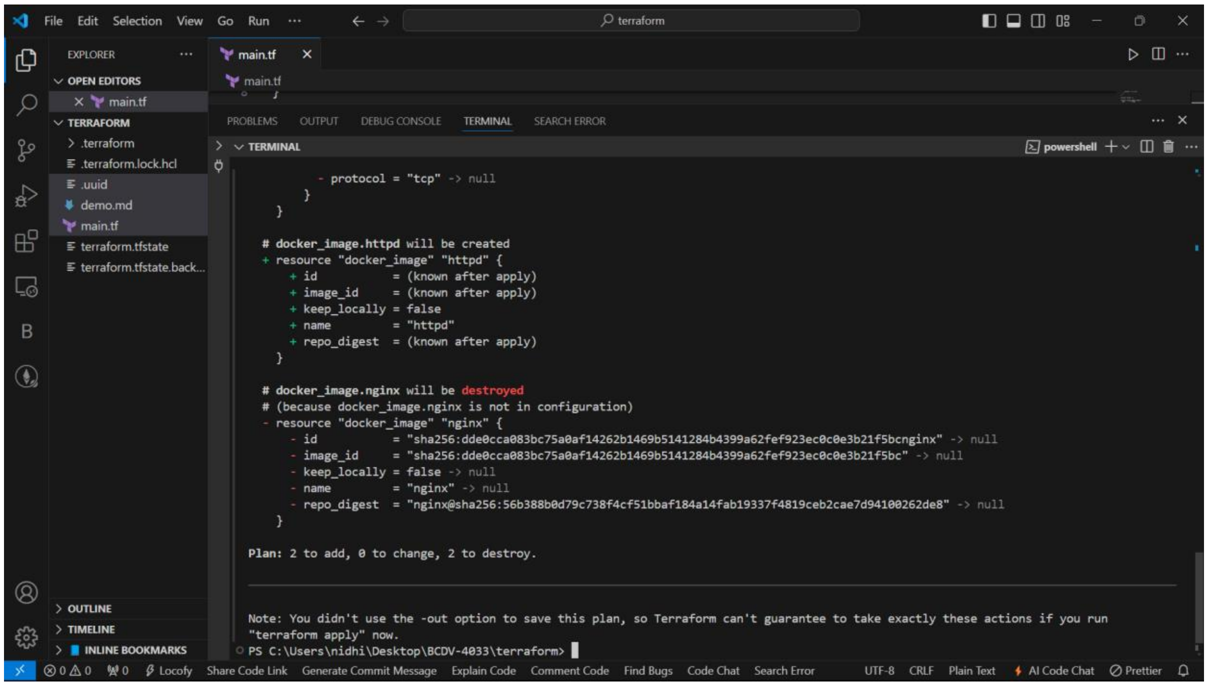
Change infrastructure (your choice of what change it is).


File Edit Selection View Go Run ···
terraform
EXPLORER
OPEN EDITORS
main.tf
TERRAFORM
.terraform
.terraform.lock.hcl
.uuid
demo.md
main.tf
terraform.tfstate
terraform.tfstate.back...
OUTLINE
TIMELINE
INLINE BOOKMARKS
0 0 0 Blackbox Locofy Share Code Link Generate Commit Message Explain Code Comment Code Find Bugs Code Chat Search Error CRLF Plain Text AI Code Chat Prettier

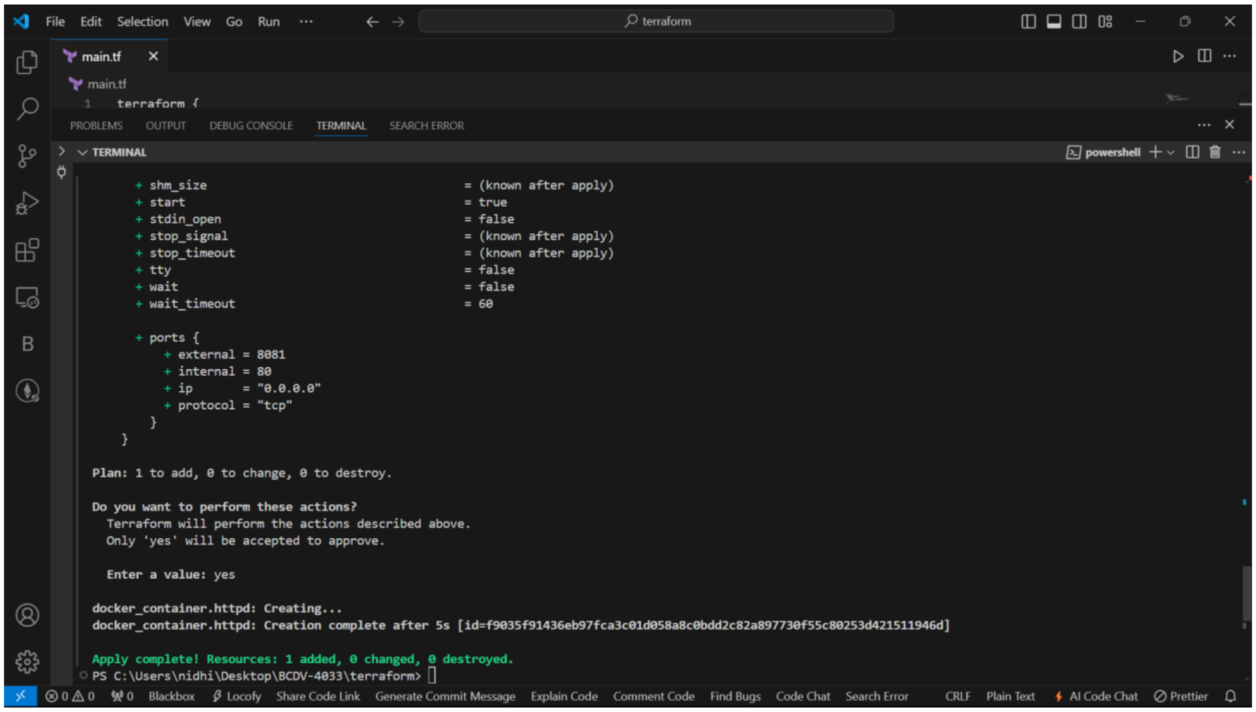
```
terraform {
  required_providers {
    docker = {
      source  = "kreuzwerker/docker"
      version = "~> 3.0.1"
    }
  }
}

provider "docker" {}

resource "docker_image" "httpd" {
  name         = "httpd"
  keep_locally = false
}

resource "docker_container" "httpd" {
  image = docker_image.httpd.image_id
  name  = "httpd-tutorial"

  ports {
    internal = 80
    external = 8081
  }
}
```

101527329 Nidhi Nakrani
lab 4

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



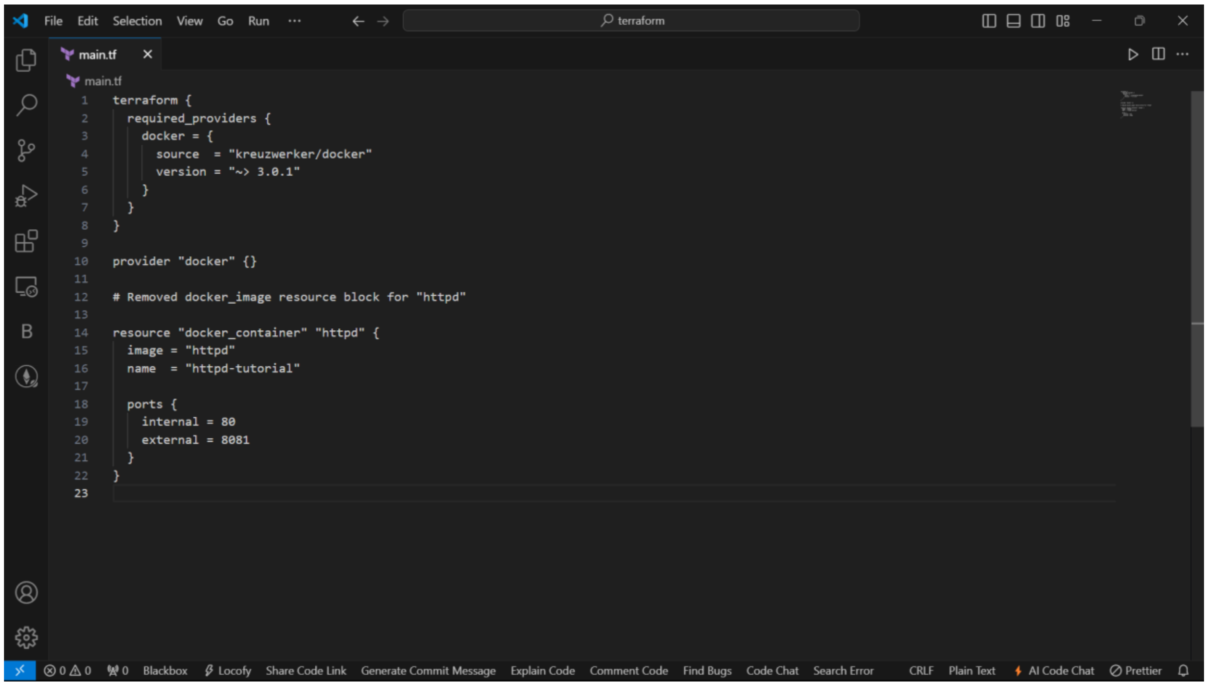
```
          - protocol = "tcp" -> null
        }
    }

  # 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:dde0cca083bc75a0af14262b1469b5141284b4399a62fef923ec0c0e3b21f5bcnginx" -> null
      - image_id     = "sha256:dde0cca083bc75a0af14262b1469b5141284b4399a62fef923ec0c0e3b21f5bc" -> null
      - keep_locally = false -> null
      - name         = "nginx" -> null
      - repo_digest  = "nginx@sha256:56b388b0d79c738f4cf51bbaf184a14fab19337f4819ceb2cae7d94100262de8" -> 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>
```



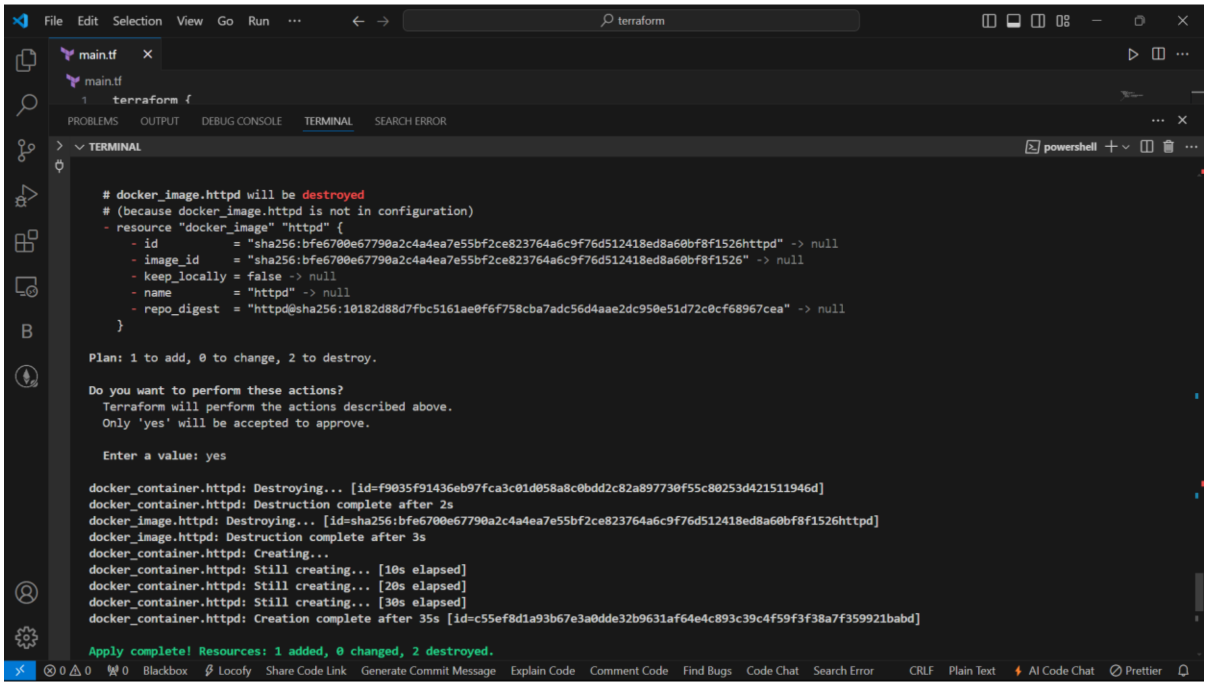
```
      + shm_size                                    = (known after apply)
      + start                                       = true
      + stdin_open                                  = false
      + stop_signal                                 = (known after apply)
      + stop_timeout                                = (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>
```

101527329   Nidhi Nakrani
lab 4

Make destructive changes (like removing one of the docker images).



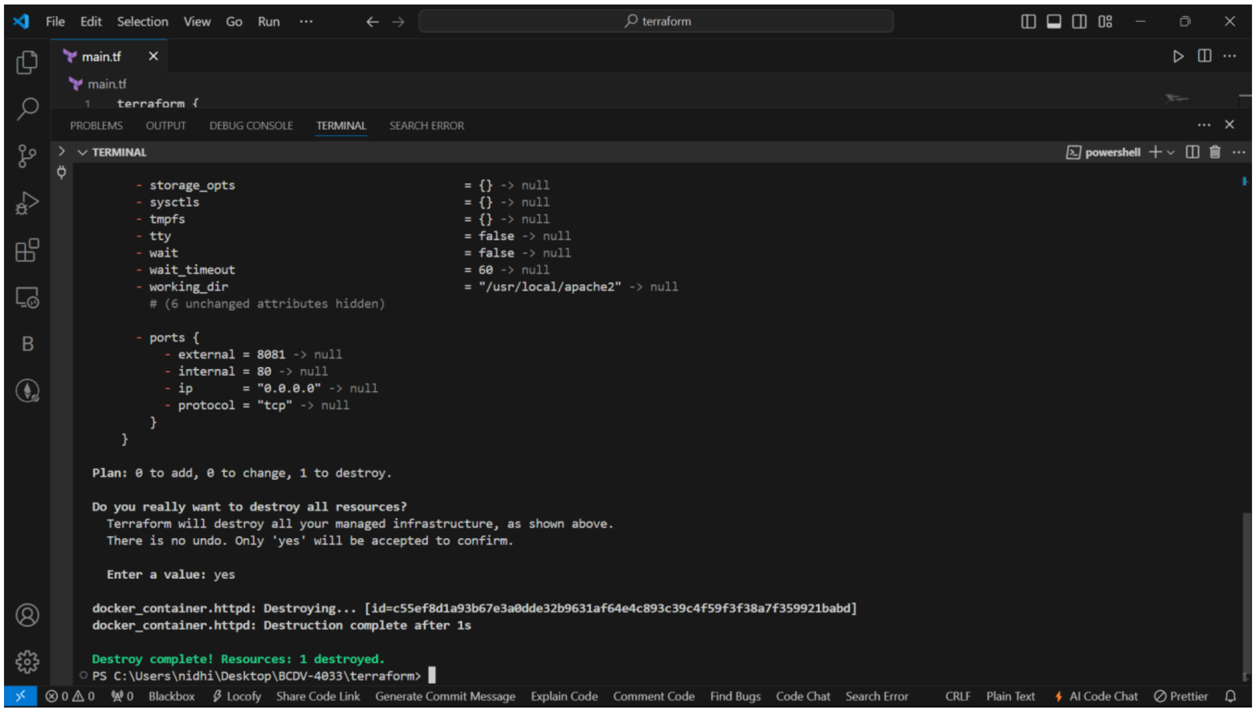
```
terraform {
  required_providers {
    docker = {
      source  = "kreuzwerker/docker"
      version = "~> 3.0.1"
    }
  }
}

provider "docker" {}

# Removed docker_image resource block for "httpd"

resource "docker_container" "httpd" {
  image = "httpd"
  name  = "httpd-tutorial"

  ports {
    internal = 80
    external = 8081
  }
}
```



```
  # docker_image.httpd will be destroyed
  # (because docker_image.httpd is not in configuration)
  - resource "docker_image" "httpd" {
      - id           = "sha256:bfe6700e67790a2c4a4ea7e55bf2ce823764a6c9f76d512418ed8a60bf8f1526httpd" -> null
      - image_id     = "sha256:bfe6700e67790a2c4a4ea7e55bf2ce823764a6c9f76d512418ed8a60bf8f1526" -> null
      - keep_locally = false -> null
      - name         = "httpd" -> null
      - repo_digest  = "httpd@sha256:10182d88d7fbc5161ae0f6f758cba7adc56d4aae2dc950e51d72c0cf68967cea" -> null
    }

Plan: 1 to add, 0 to change, 2 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: Destroying... [id=f9035f91436eb97fca3c01d058a8c0bdd2c82a897730f55c80253d421511946d]
docker_container.httpd: Destruction complete after 2s
docker_image.httpd: Destroying... [id=sha256:bfe6700e67790a2c4a4ea7e55bf2ce823764a6c9f76d512418ed8a60bf8f1526httpd]
docker_image.httpd: Destruction complete after 3s
docker_container.httpd: Creating...
docker_container.httpd: Still creating... [10s elapsed]
docker_container.httpd: Still creating... [20s elapsed]
docker_container.httpd: Still creating... [30s elapsed]
docker_container.httpd: Creation complete after 35s [id=c55ef8d1a93b67e3a0dde32b9631af64e4c893c39c4f59f3f38a7f359921babd]

Apply complete! Resources: 1 added, 0 changed, 2 destroyed.
```

101527329   Nidhi Nakrani
lab 4

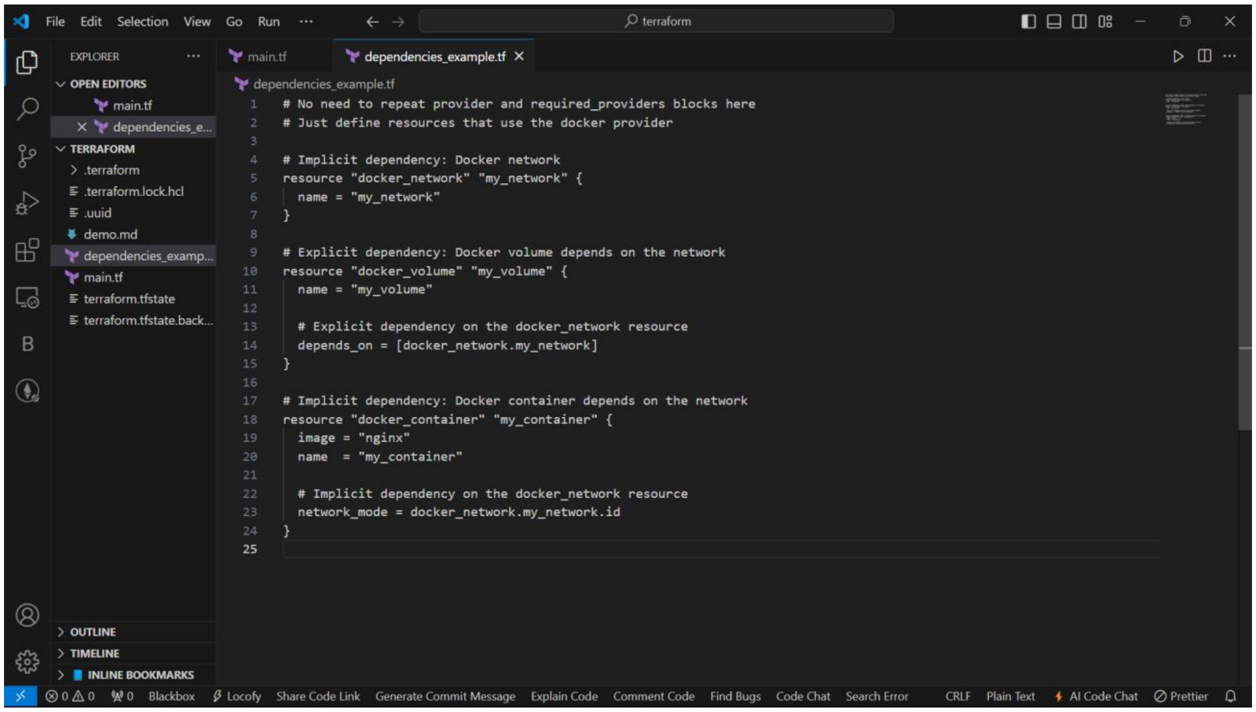
Destroy the complete resource.



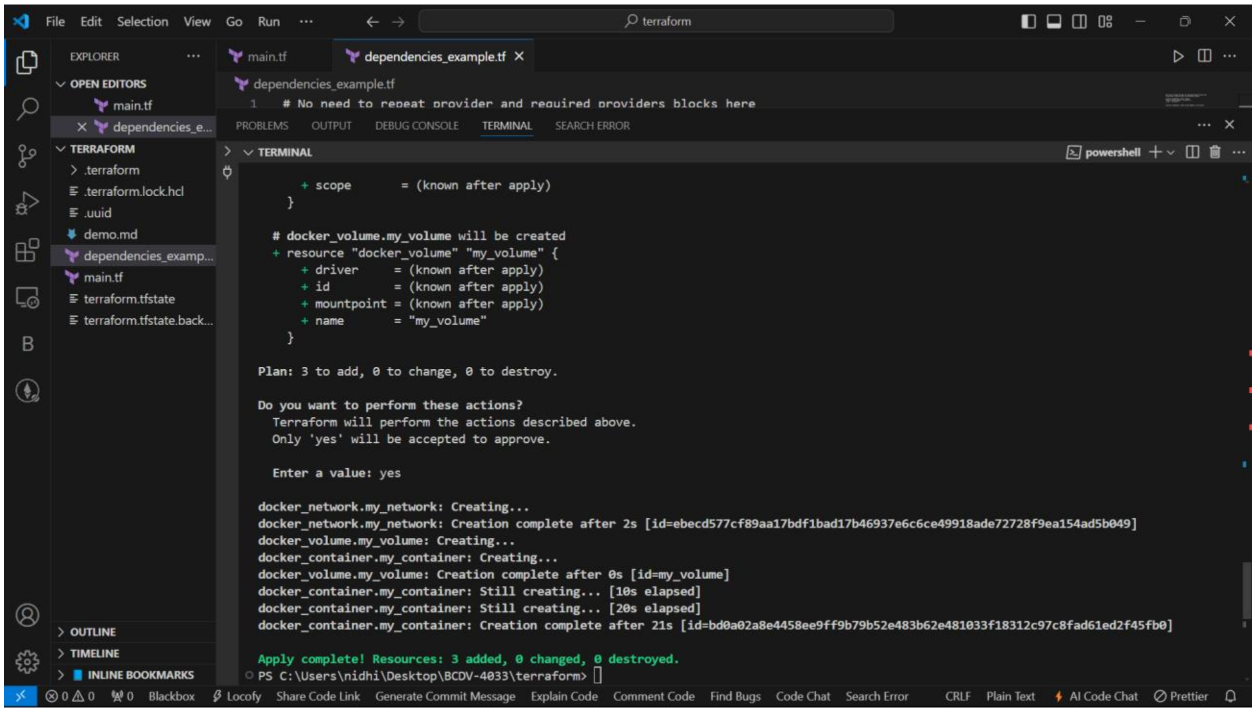
```
      - storage_opts                                   = {} -> null
      - sysctls                                        = {} -> null
      - tmpfs                                          = {} -> null
      - tty                                            = false -> null
      - wait                                           = 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).



```
# No need to repeat provider and required_providers blocks here
# Just define resources that use the docker provider

# Implicit dependency: Docker network
resource "docker_network" "my_network" {
  name = "my_network"
}

# Explicit dependency: Docker volume depends on the network
resource "docker_volume" "my_volume" {
  name = "my_volume"

  # Explicit dependency on the docker_network resource
  depends_on = [docker_network.my_network]
}

# Implicit dependency: Docker container depends on the network
resource "docker_container" "my_container" {
  image = "nginx"
  name  = "my_container"

  # Implicit dependency on the docker_network resource
  network_mode = docker_network.my_network.id
}
```



```
      + 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>
```