

Lab-4

1. Install Terraform in your local system.

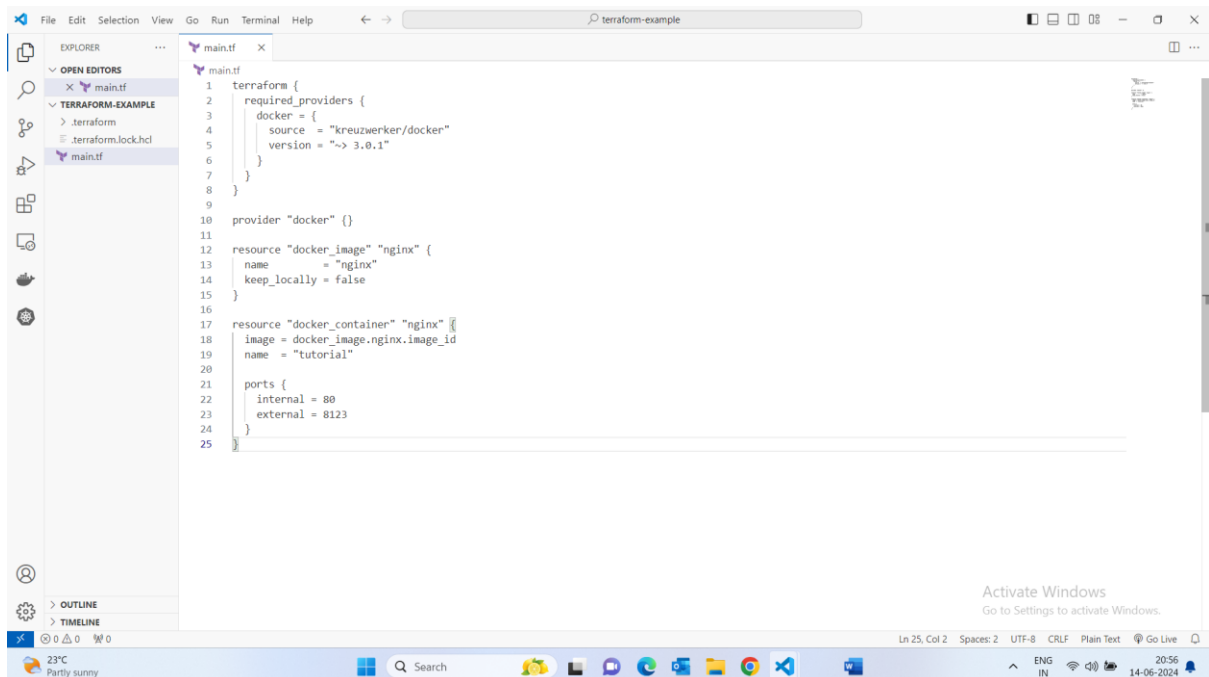
```
karan_acharya@DESKTOP-TJR4MPO:~$ sudo apt-get update && sudo apt-get install -y gnupg software-properties-common
[sudo] password for karan_acharya:
Hit:1 https://download.docker.com/linux/ubuntu jammy InRelease
Get:2 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Hit:3 http://archive.ubuntu.com/ubuntu jammy InRelease
Get:4 http://archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Hit:5 http://archive.ubuntu.com/ubuntu jammy-backports InRelease
Get:6 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [1517 kB]
Get:7 http://security.ubuntu.com/ubuntu jammy-security/main Translation-en [259 kB]
Get:8 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 Packages [1933 kB]
Get:9 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [1731 kB]
Get:10 http://security.ubuntu.com/ubuntu jammy-security/restricted Translation-en [329 kB]
Get:11 http://archive.ubuntu.com/ubuntu jammy-updates/main Translation-en [318 kB]
Get:12 http://archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 Packages [1990 kB]
Get:13 http://archive.ubuntu.com/ubuntu jammy-updates/restricted Translation-en [338 kB]
Get:14 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [1087 kB]
Fetched 9759 kB in 4s (2435 kB/s)
Reading package lists... Done
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
gnupg is already the newest version (2.2.27-3ubuntu2.1).
gnupg set to manually installed.
The following packages will be upgraded:
  python3-software-properties software-properties-common
2 upgraded, 0 newly installed, 0 to remove and 45 not upgraded.
Need to get 42.9 kB of archives.
After this operation, 0 B of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 software-properties-common all 0.99.22.9 [14.1 kB]
Get:2 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 python3-software-properties all 0.99.22.9 [28.8 kB]
Fetched 42.9 kB in 1s (66.7 kB/s)
(Reading database ... 43044 files and directories currently installed.)
Preparing to unpack .../software-properties-common_0.99.22.9_all.deb ...
Unpacking software-properties-common (0.99.22.9) over (0.99.22.8) ...
Preparing to unpack .../python3-software-properties_0.99.22.9_all.deb ...
Unpacking python3-software-properties (0.99.22.9) over (0.99.22.8) ...
Setting up python3-software-properties (0.99.22.9) ...
Setting up software-properties-common (0.99.22.9) ...
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for dbus (1.12.20-2ubuntu4.1) ...
karan_acharya@DESKTOP-TJR4MPO:~$ gpg --keyring /usr/share/keyrings/hashicorp-archive-keyring.gpg --no-default-keyring --keyring /usr/share/keyrings/hashicorp-archive-keyring.gpg --fingerprint
gpg: directory '/home/karan_acharya/.gnupg/' created
gpg: /home/karan_acharya/.gnupg/trustdb.gpg: trustdb created
gpg: /usr/share/keyrings/hashicorp-archive-keyring.gpg
-----BEGIN PGP PUBLIC KEY BLOCK-----
pub 2024-06-14 17:56:59 [sc] expires: 2026-01-09
uid [unknown] HashiCorp Security (HashiCorp Package Signing) <security+packaging@hashicorp.com>
sub rsa4096 2023-01-19 [s] [expires: 2026-01-09]
-----
karan_acharya@DESKTOP-TJR4MPO:~$ echo "deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] https://apt.releases.hashicorp.com $(lsb_release -cs) main" | \
sudo tee /etc/apt/sources.list.d/hashicorp.list
deb [signed-by=/usr/share/keyrings/hashicorp-archive-keyring.gpg] https://apt.releases.hashicorp.com jammy main
karan_acharya@DESKTOP-TJR4MPO:~$ sudo apt update
Hit:1 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:2 https://download.docker.com/linux/ubuntu jammy InRelease
Get:3 https://apt.releases.hashicorp.com jammy InRelease [12.9 kB]
Hit:4 http://archive.ubuntu.com/ubuntu jammy InRelease
Hit:5 http://archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:6 http://archive.ubuntu.com/ubuntu jammy-backports InRelease
Get:7 https://apt.releases.hashicorp.com jammy/main amd64 Packages [136 kB]
Fetched 149 kB in 1s (150 kB/s)
Reading package lists... Done
karan_acharya@DESKTOP-TJR4MPO:~$ sudo apt-get install terraform
Reading state information... Done
45 packages can be upgraded. Run 'apt list --upgradable' to see them.
karan_acharya@DESKTOP-TJR4MPO:~$ sudo apt-get install terraform
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  terraform
0 upgraded, 1 newly installed, 0 to remove and 45 not upgraded.
Need to get 27.7 MB of archives.
After this operation, 88.2 MB of additional disk space will be used.
Get:1 https://apt.releases.hashicorp.com jammy/main amd64 terraform amd64 1.8.5-1 [27.7 MB]
Fetched 27.7 MB in 5s (5048 kB/s)
Selecting previously unselected package terraform.
(Reading database ... 43044 files and directories currently installed.)
Unpacking terraform (1.8.5-1) ...
Setting up terraform (1.8.5-1) ...
karan_acharya@DESKTOP-TJR4MPO:~$ terraform -help
Usage: terraform [global options] <subcommand> [args]

The available commands for execution are listed below.
The primary workflow commands are given first, followed by
less common or more advanced commands.

Main commands:
  init      Prepare your working directory for other commands
  validate  Check whether the configuration is valid
  plan      Show changes required by the current configuration
  apply     Create or update infrastructure
  destroy   Destroy previously-created infrastructure

All other commands:
  console   Try Terraform expressions at an interactive command prompt
  fmt       Reformat your configuration in the standard style
  force-unlock Release a stuck lock on the current workspace
  get       Install or upgrade remote Terraform modules
  graph     Generate a Graphviz graph of the steps in an operation
  import    Associate existing infrastructure with a Terraform resource
  login     Obtain and save credentials for a remote host
  logout    Remove locally-stored credentials for a remote host
```

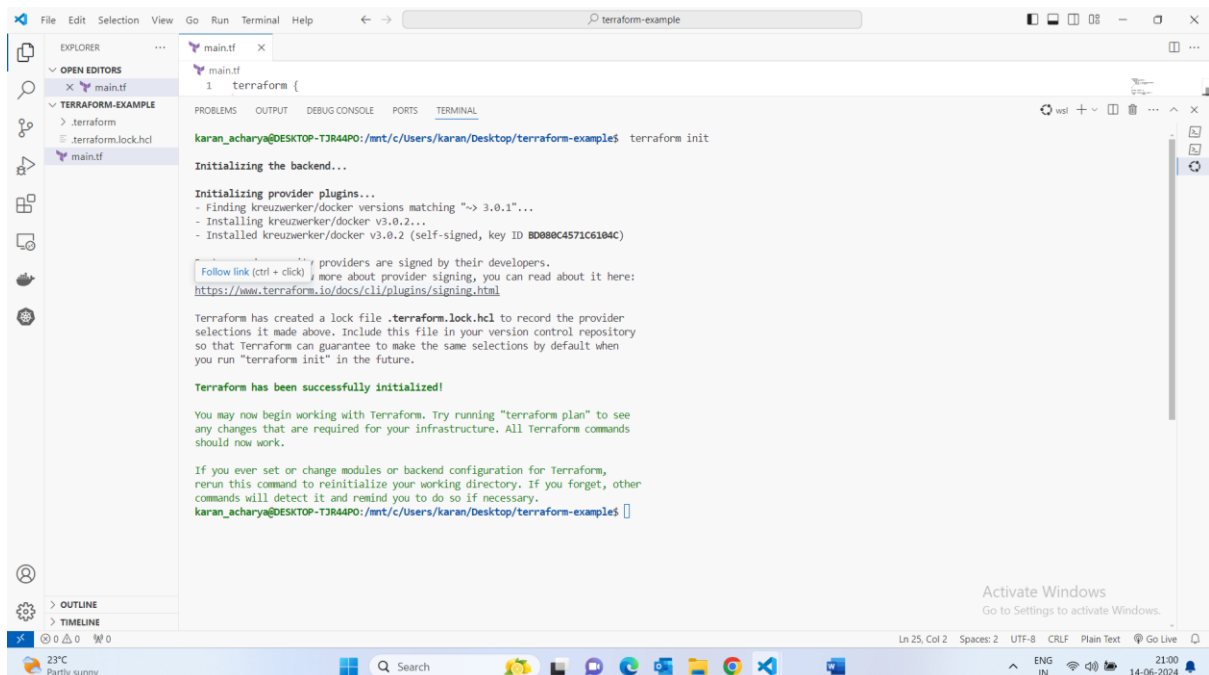
Create a tf configuration for running docker resource



```
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" "nginx" {
13   name = "nginx"
14   keep_locally = false
15 }
16
17 resource "docker_container" "nginx" {
18   image = docker_image.nginx.image_id
19   name = "tutorial"
20
21   ports {
22     internal = 80
23     external = 8123
24   }
25 }
```

Initialize Terraform

- To initialize the resources we will use command: **terraform init**.
- These command will initialize terraform working directory, moreover it will download all the necessary plugin and create a terraform.lock file which will manage its dependencies.



```
karan_acharya@DESKTOP-TJR44P0: /mnt/c/Users/karan/Desktop/terraform-example$ 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 80808C4571C6104C)

providers are signed by their developers.
Follow link (ctrl + click) for 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 selection by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

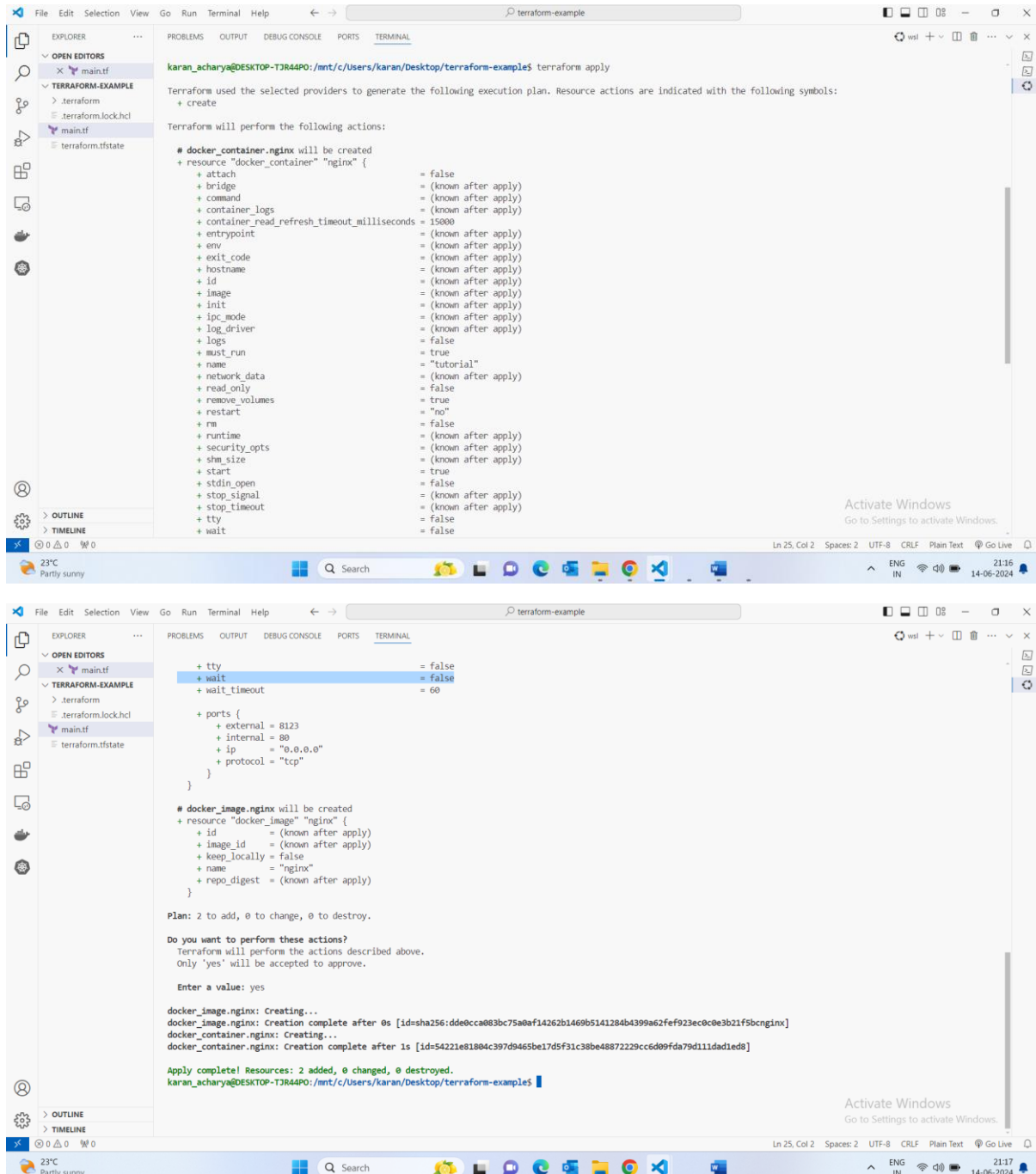
You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

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.
karan_acharya@DESKTOP-TJR44P0: /mnt/c/Users/karan/Desktop/terraform-example$
```

Apply the initial resources

- To apply the initial resources run the following command: `terraform apply`.
- It will create a docker container in the file and it will give output of resource id for the created container.

Output:



The first screenshot shows the terminal output of the `terraform apply` command. It displays the execution plan for creating a `docker_container.nginx` resource. The plan lists various attributes such as `attach`, `bridge`, `command`, `container_logs`, `container_read_refresh_timeout_milliseconds`, `entrypoint`, `env`, `exit_code`, `hostname`, `id`, `image`, `init`, `ipc_mode`, `log_driver`, `logs`, `must_run`, `name`, `network_data`, `read_only`, `remove_volumes`, `restart`, `rm`, `runtime`, `security_opts`, `shm_size`, `start`, `stdin_open`, `stop_signal`, `stop_timeout`, `tty`, and `wait`. The second screenshot shows the terminal output after the resources have been created. It displays the creation of `docker_image.nginx` and `docker_container.nginx` resources, along with their respective IDs. The output also shows the plan for adding, changing, or destroying resources, and the final status of the resources.

```
File Edit Selection View Go Run Terminal Help
terraform-example

EXPLORER
  OPEN EDITORS
    X main.tf
  TERRAFORM-EXAMPLE
    .terraform
    .terraform.lock.hcl
    main.tf
    terraform.tfstate

PROBLEMS OUTPUT DEBUG CONSOLE PORTS TERMINAL

karan_acharya@DESKTOP-T3R44PO: /mnt/c/Users/karan/Desktop/terraform-example$ 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.nginx will be created
+ resource "docker_container" "nginx" {
  + 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         = "tutorial"
  + 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
  + stdin_open   = false
  + stop_signal   = (known after apply)
  + stop_timeout = false
  + tty          = false
  + wait         = false
}

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
karan_acharya@DESKTOP-T3R44PO: /mnt/c/Users/karan/Desktop/terraform-example$
```

```
File Edit Selection View Go Run Terminal Help
terraform-example

EXPLORER
  OPEN EDITORS
    X main.tf
  TERRAFORM-EXAMPLE
    .terraform
    .terraform.lock.hcl
    main.tf
    terraform.tfstate

PROBLEMS OUTPUT DEBUG CONSOLE PORTS TERMINAL

+ 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.

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

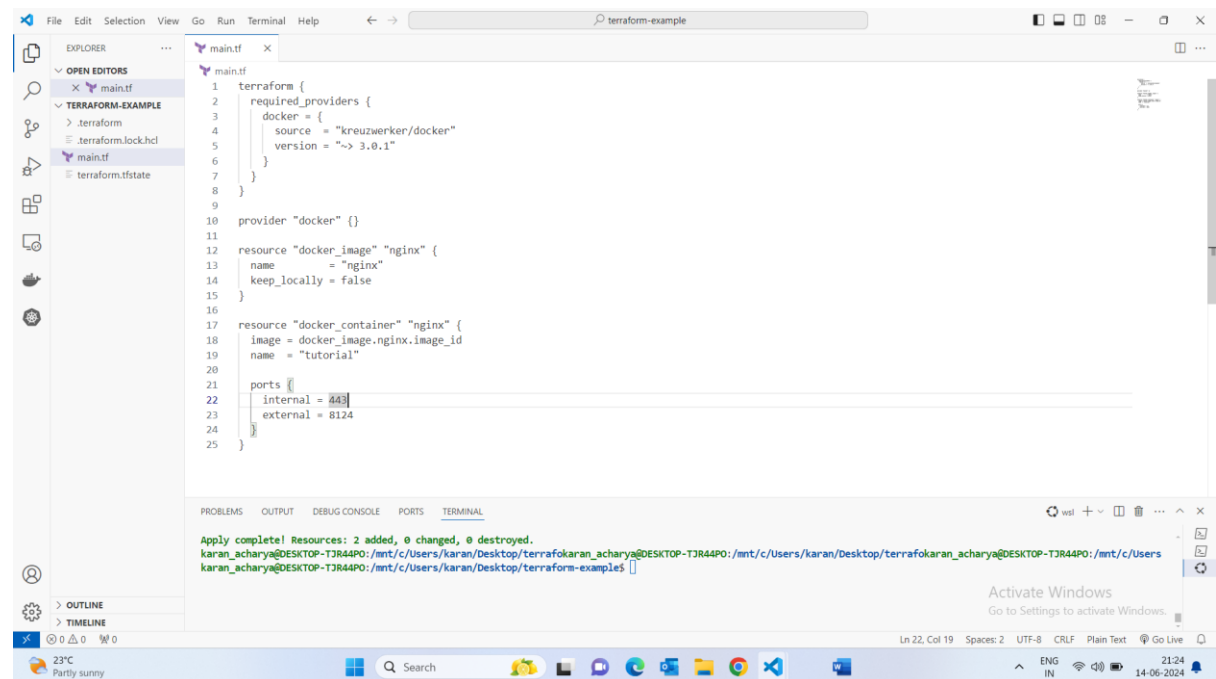
docker_image.nginx: Creating...
docker_image.nginx: Creation complete after 0s [id=sha256:dde0cca083bc75a0af14262b1469b5141284b4399a62fef923ec0ce3b21f5bcnginx]
docker_container.nginx: Creating...
docker_container.nginx: Creation complete after 1s [id=54221e81804c397d9465be17d5f31c38be48872229cc6d09fda79d111dad1ed8]

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.
karan_acharya@DESKTOP-T3R44PO: /mnt/c/Users/karan/Desktop/terraform-example$
```

Step 5: Make changes to the infrastructure

- Here in the main.tf file I had changed the ports { internal =80 external=8123 to internal=443 and external=8124}

Output:



```
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" "nginx" {
13   name = "nginx"
14   keep_locally = false
15 }
16
17 resource "docker_container" "nginx" {
18   image = docker_image.nginx.image_id
19   name = "tutorial"
20
21   ports {
22     internal = 443
23     external = 8124
24   }
25 }
```

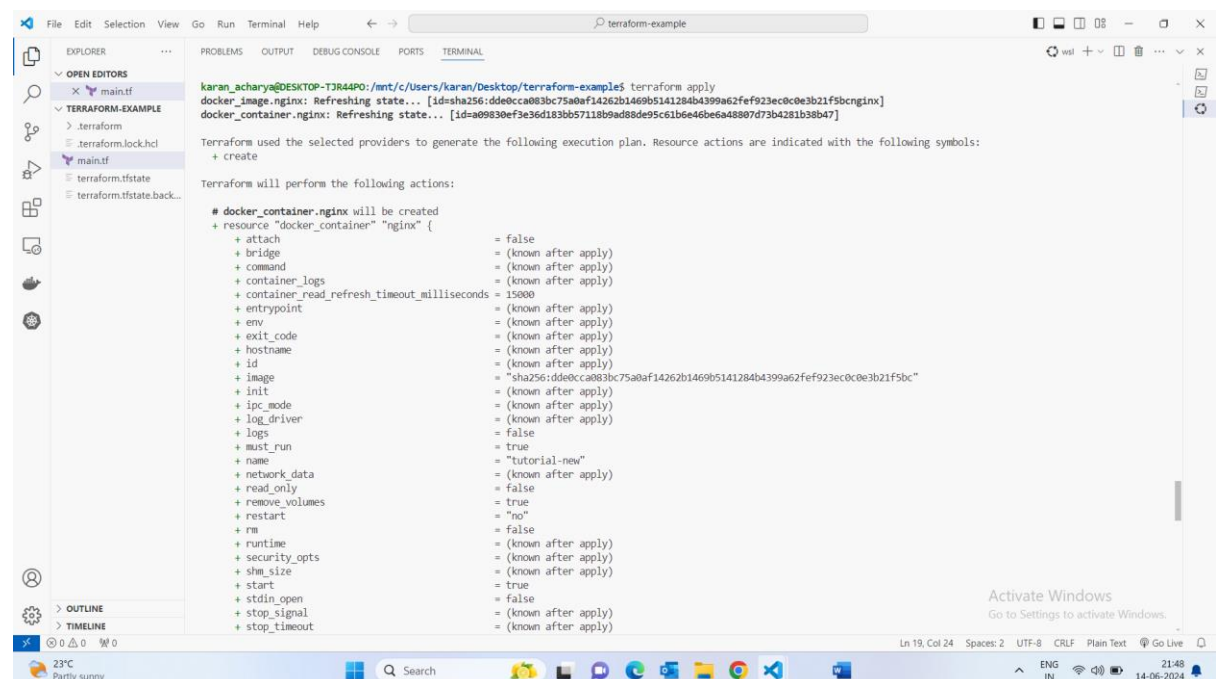
Apply complete! Resources: 2 added, 0 changed, 0 destroyed.

karan_acharya@DESKTOP-TJR44P0:/mnt/c/Users/karan/Desktop/terraformexample\$ terraform apply

Create a plan :

- It will generate a plan detailing the changes to be made and it will give the plan in output console.
- Command: **terraform plan**

Output:

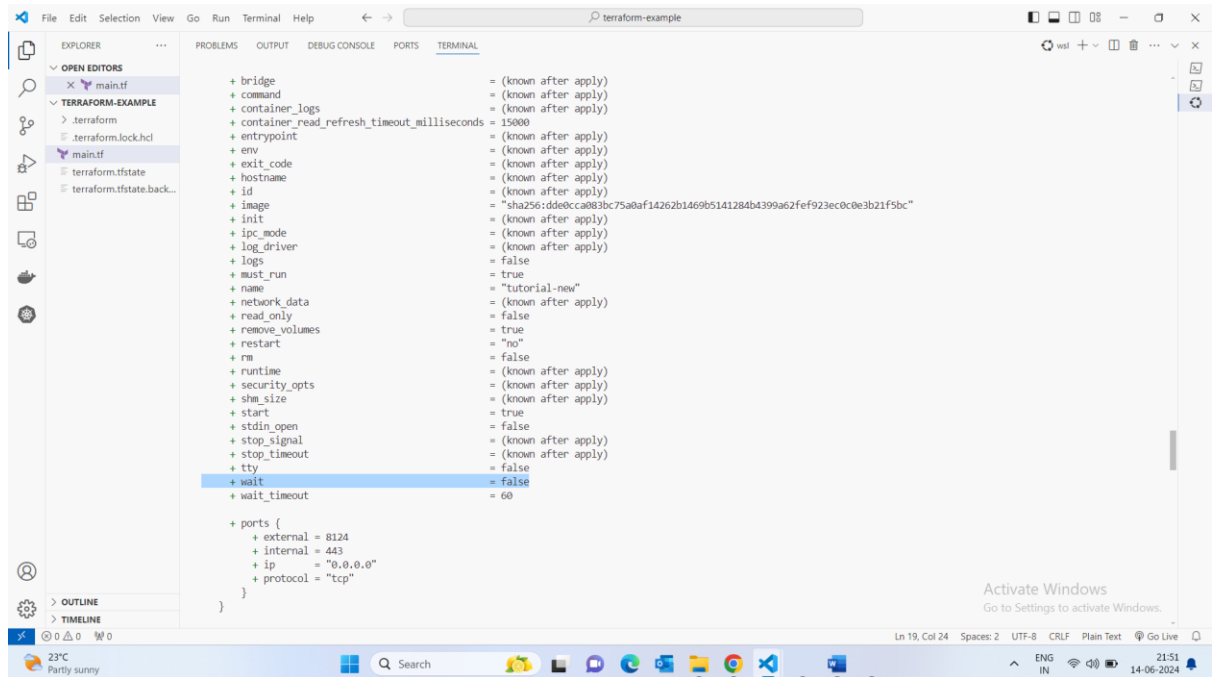


```
karan_acharya@DESKTOP-TJR44P0:/mnt/c/Users/karan/Desktop/terraformexample$ terraform plan
docker_image.nginx: Refreshing state... [id=sha256:dde0cca083bc75a0af14262b1469b5141284b4399a62fef923ec0c0e3b21f5bcnginx]
docker_container.nginx: Refreshing state... [id=a09830ef3e36d183bb57118b9ad88de95c61b6e46be648897d73b4261b38b47]

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:

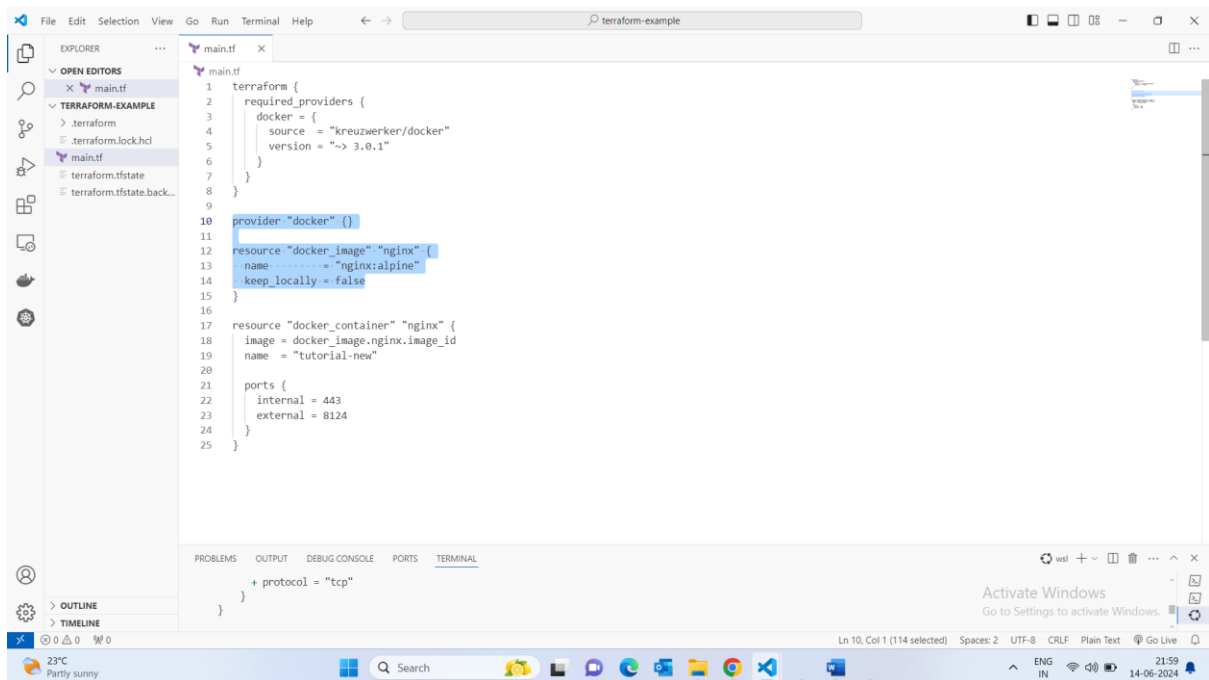
# docker_container.nginx will be created
+ resource "docker_container" "nginx" {
  + attach                        = false
  + bridge                       = (known after apply)
  + command                     = (known after apply)
  + container_logs              = (known after apply)
  + container_read_refresh_timeout_milliseconds = 15000
  + entrypoint                  = (known after apply)
  + env                         = (known after apply)
  + exit_code                   = (known after apply)
  + hostname                    = (known after apply)
  + id                         = (known after apply)
  + image                       = "sha256:dde0cca083bc75a0af14262b1469b5141284b4399a62fef923ec0c0e3b21f5bc"
  + init                        = (known after apply)
  + ipc_mode                    = (known after apply)
  + log_driver                  = (known after apply)
  + logs                        = false
  + must_run                   = true
  + name                        = "tutorial-new"
  + 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
  + stdin_open                  = false
  + stop_signal                 = (known after apply)
  + stop_timeout                = (known after apply)
}
```



Make destructive changes:

- Now will make destructive changes that is I will update the main.tf file to remove one of the docker images.

Output: The selected blue part is the destructive change I did



Destroy the complete resource:

- Run the command: terraform destroy to destroy the complete resource.
- It will destroy the container and give confirmation message.

Output:

The screenshot displays the Visual Studio Code interface with the Terraform CLI output in the terminal. The Explorer sidebar on the left shows the project structure: `main.tf`, `terraform.lock.hcl`, `terraform.tfstate`, and `terraform.tfstate.backup`. The terminal output shows the command `terraform destroy` being executed, followed by a refresh of the state and the generation of an execution plan. The plan indicates that the `docker_container.nginx` resource will be destroyed. The output lists various attributes of the resource, such as `attach`, `command`, `container_read_refresh_timeout_milliseconds`, `cpu_shares`, `dns`, `dns_opts`, `dns_search`, `entrypoint`, `env`, `group_add`, `hostname`, `id`, `image`, `init`, `ipc_mode`, `log_driver`, `log_opts`, `logs`, `max_retry_count`, `memory`, `memory_swap`, `must_run`, and `name`. The output also shows the `network_data` and `ports` attributes.

```
karan_acharya@DESKTOP-T3R44PD: /mnt/c/Users/karan/Desktop/terraform-example$ terraform destroy
docker_image.nginx: Refreshing state... [id=sha256:dde0cca083bc75a0af14262b1469b5141284b4399a62fef923ec0c0e3b21f5bcnginx]
docker_container.nginx: Refreshing state... [id=f3b7d5b7f6f8780ea9b2369fead19ef64ab2bf3e3cec4c8fb15b0e915fd88f8]

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.nginx will be destroyed
- resource "docker_container" "nginx" {
  - attach                                = false -> null
  - command                              = [
    - "nginx",
    - "-g",
    - "daemon off;",
  ] -> null
  - container_read_refresh_timeout_milliseconds = 15000 -> null
  - cpu_shares                            = 0 -> null
  - dns                                   = [] -> null
  - dns_opts                              = [] -> null
  - dns_search                            = [] -> null
  - entrypoint                            = [
    - "/docker-entrypoint.sh",
  ] -> null
  - env                                   = [] -> null
  - group_add                             = [] -> null
  - hostname                              = "f3b7d5b7f6f8" -> null
  - id                                    = "f3b7d5b7f6f8780ea9b2369fead19ef64ab2bf3e3cec4c8fb15b0e915fd88f8" -> null
  - image                                 = "sha256:dde0cca083bc75a0af14262b1469b5141284b4399a62fef923ec0c0e3b21f5bc" -> null
  - init                                  = false -> null
  - ipc_mode                              = "private" -> 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                                  = "tutorial-new" -> null
}
```

Ln 10, Col 1 (114 selected) Spaces: 2 UTF-8 CRLF Plain Text Go Live

23°C Partly sunny

ENG IN 22:06 14-06-2024

Activate Windows Go to Settings to activate Windows.

The screenshot shows the VS Code interface with the Explorer sidebar on the left. The 'TERRAFORM-EXAMPLE' folder is expanded, showing files like .terraform, main.tf, terraform.lock.hcl, terraform.tfstate, and terraform.tfstate.backup. The main editor displays the output of a Terraform destroy command. The output indicates that the 'docker_image.nginx' resource will be destroyed. It shows the resource details for 'nginx' and the plan to destroy it. The user is prompted to confirm the destruction of all resources, and they enter 'yes'. The output then shows the destruction of the 'docker_image.nginx' resource and the 'docker_container.nginx' resource. The final message is 'Destroy complete! Resources: 2 destroyed.'

```

}

# docker_image.nginx will be destroyed
- 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: 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.nginx: Destroying... [id=f3b7d5b7fef8780ea9b2369fead19ef64ab2bf3ec4c8fb15b0e915fd88f8]
docker_container.nginx: Destruction complete after 1s
docker_image.nginx: Destroying... [id=sha256:dde0cca083bc75a0af14262b1469b5141284b4399a62fef923ec0c0e3b21f5bcnginx]
docker_image.nginx: Destruction complete after 1s

Destroy complete! Resources: 2 destroyed.
karan_acharya@DESKTOP-T3R44PO: /mnt/c/Users/karan/Desktop/terraform-examples$
```

- Here in the above image the destroyed command destroyed the docker container and gave the confirmation message.

Create resources with dependencies

- Here I will update the configuration file to create resources and dependencies. I had created another file named as dependencies.tf

The screenshot shows the VS Code interface with the Explorer sidebar on the left. The 'TERRAFORM-EXAMPLE' folder is expanded, showing files like .terraform, main.tf, terraform.lock.hcl, terraform.tfstate, and terraform.tfstate.backup. The main editor displays the output of a Terraform plan command. The output shows the plan to create a new 'docker_image.nginx' resource. It shows the resource details for 'nginx' and the plan to create it. The user is prompted to confirm the actions, and they enter 'yes'. The output then shows the creation of the 'docker_image.nginx' resource and the 'docker_container.nginx' resource. The final message is 'Creation complete after 1s'.

```

dependencies.tf
1 resource "docker_image" "example" {
2   name = "example"
3   image = "krishna"
4 }
5

+ resource "docker_image" "nginx" {
+   id           = (known after apply)
+   image_id     = (known after apply)
+   keep_locally = false
+   name        = "nginx:alpine"
+   repo_digest = (known after apply)
+ }

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_image.nginx: Creating...
docker_container.nginx: Creating...
docker_image.nginx: Creation complete after 8s [id=sha256:70ea0d8cc5300acde42073a2fbc0d28964dd6e3c31263d92589c2320c3ccba4nginx:alpine]
docker_container.nginx: Creating...
docker_image.nginx: Creation complete after 1s [id=2f148651edcdf7eb6099cb8adf39f90791167ed8aa9c7b43228a5f38b5a4b082]
docker_container.nginx: Creation complete after 1s [id=2f148651edcdf7eb6099cb8adf39f90791167ed8aa9c7b43228a5f38b5a4b082]
```