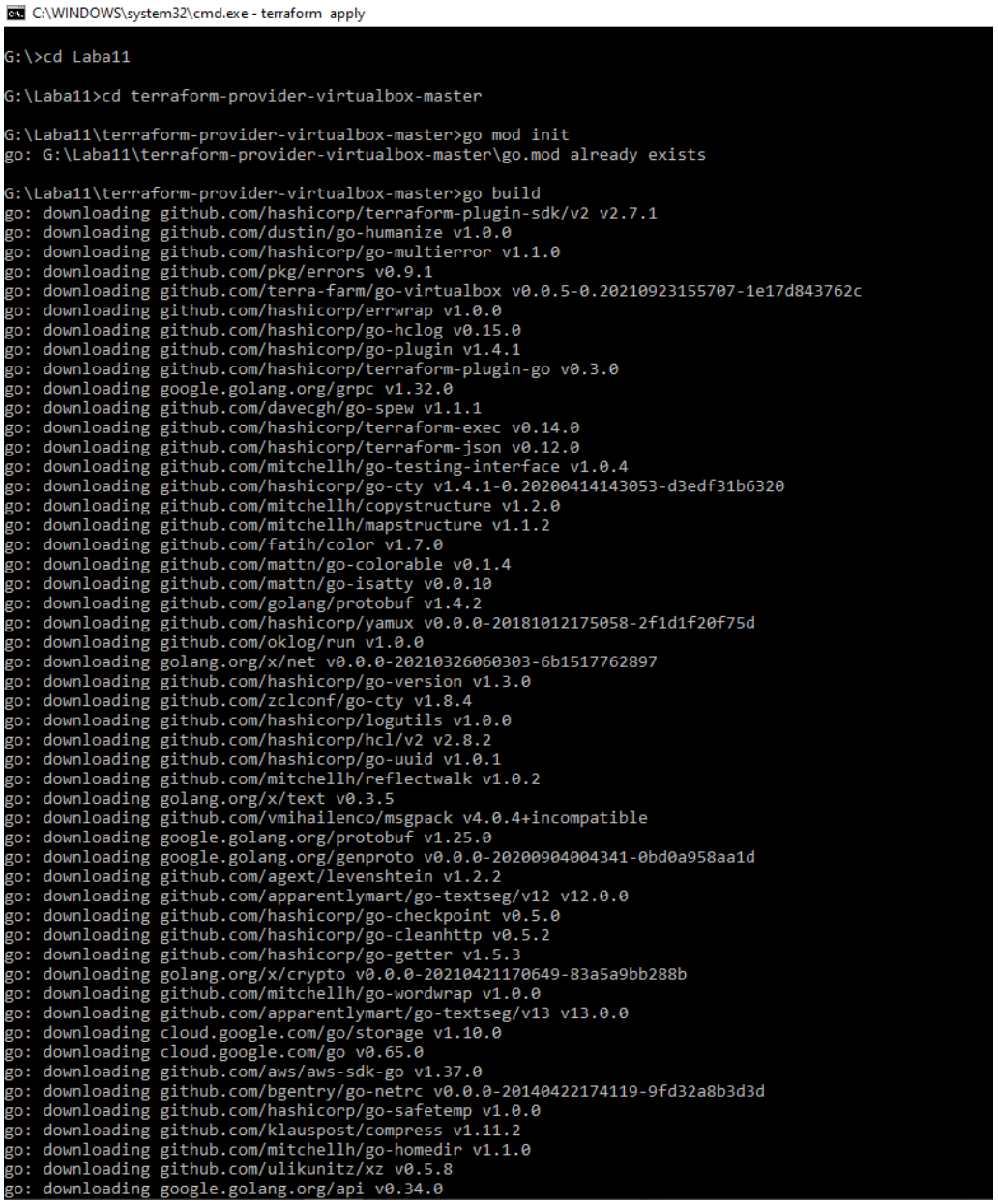
Лабораторна робота №11

Тема: Розгортання інфраструктури

Мета: Оволодіти навичками по автоматичному розгортанню інфраструктури звикористанням terraform

Хід роботи:

1. Запустив ПК.
2. Встановив terraform.
3. Встановив плагін для роботи в локальному варіанті з віртуальною



машиною VirtualBox.

4. Створив файл main.tf командою:

```
G:\Laba11\terraform>echo > main.tf
```

Записав дані та зберіг його.

5. Тепер ініціалізую плагіни та підключаю їх, за допомогою команди:

```
G:\Laba11\terraform>terraform init

Initializing the backend...

Initializing provider plugins...
- Finding terra-farm/virtualbox versions matching "0.2.2-alpha.1"...
- Installing terra-farm/virtualbox v0.2.2-alpha.1...
- Installed terra-farm/virtualbox v0.2.2-alpha.1 (self-signed, key ID 51EC33490F8CDBE5)

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.
```

6. Переглядаю створену конфігурацію:

```
G:\Laba11\terraform>terraform plan

Terraform used the selected providers to generate the following execution plan. Resource actions are indicated with the following s
  + create

Terraform will perform the following actions:

  # virtualbox_vm.node[0] will be created
  + resource "virtualbox_vm" "node" {
      + cpus   = 2
      + id     = (known after apply)
      + image  = "https://app.vagrantup.com/ubuntu/boxes/xenial64/versions/20190507.0.0/providers/virtualbox.box"
      + memory = "512 mib"
      + name   = "node-01"
      + status = "running"

      + network_adapter {
          + device                = "IntelPro1000MTServer"
          + host_interface        = "Realtek PCIe GbE Family Controller"
          + ipv4_address          = (known after apply)
          + ipv4_address_available = (known after apply)
          + mac_address           = (known after apply)
          + status                = (known after apply)
          + type                  = "bridged"
        }
    }

  # virtualbox_vm.node[1] will be created
  + resource "virtualbox_vm" "node" {
      + cpus   = 2
      + id     = (known after apply)
      + image  = "https://app.vagrantup.com/ubuntu/boxes/xenial64/versions/20190507.0.0/providers/virtualbox.box"
      + memory = "512 mib"
      + name   = "node-02"
      + status = "running"

      + network_adapter {
          + device                = "IntelPro1000MTServer"
          + host_interface        = "Realtek PCIe GbE Family Controller"
          + ipv4_address          = (known after apply)
          + ipv4_address_available = (known after apply)
          + mac_address           = (known after apply)
          + status                = (known after apply)
```

7. Вводжу команду, призначену для застосування змін, де підтверджую їївиконання:

```
G:\Laba11\terraform>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:

  # virtualbox_vm.node[0] will be created
  + resource "virtualbox_vm" "node" {
      + cpus   = 2
      + id     = (known after apply)
      + image  = "https://app.vagrantup.com/ubuntu/boxes/xenial64/versions/20190507.0.0/providers/virtualbox.box"
      + memory = "512 mib"
      + name   = "node-01"
      + status = "running"

      + network_adapter {
          + device                = "IntelPro1000MTServer"
          + host_interface        = "Realtek PCIe GbE Family Controller"
          + ipv4_address          = (known after apply)
          + ipv4_address_available = (known after apply)
          + mac_address           = (known after apply)
          + status                = (known after apply)
          + type                  = "bridged"
        }
    }

  # virtualbox_vm.node[1] will be created
  + resource "virtualbox_vm" "node" {
      + cpus   = 2
      + id     = (known after apply)
      + image  = "https://app.vagrantup.com/ubuntu/boxes/xenial64/versions/20190507.0.0/providers/virtualbox.box"
      + memory = "512 mib"
      + name   = "node-02"
      + status = "running"

      + network_adapter {
          + device                = "IntelPro1000MTServer"
          + host_interface        = "Realtek PCIe GbE Family Controller"
          + ipv4_address          = (known after apply)
          + ipv4_address_available = (known after apply)
          + mac_address           = (known after apply)
          + status                = (known after apply)
          + type                  = "bridged"
        }
    }

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

Changes to Outputs:
  + IPAddr   = (known after apply)
  + IPAddr_2 = (known after apply)
```

8. Знищую створену інфраструктуру за допомогою команди:

```
D:\Навчання\Мережі\Лр13\lr13>VBoxManage --version
6.1.30r148432

D:\Навчання\Мережі\Лр13\lr13>terraform destroy

Changes to Outputs:

You can apply this plan to save these new output values to the Terraform state, without changing any real
infrastructure.

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


Destroy complete! Resources: 0 destroyed.
```