

# 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

C:\WINDOWS\system32\cmd.exe - terraform apply

```
G:\>cd Laba11
G:\Laba11>cd terraform-provider-virtualbox-master
G:\Laba11\terraform-provider-virtualbox-master>go mod init
go: G:\Laba11\terraform-provider-virtualbox-master\go.mod already exists
G:\Laba11\terraform-provider-virtualbox-master>go build
go: downloading github.com/hashicorp/terraform-plugin-sdk/v2 v2.7.1
go: downloading github.com/dustin/go-humanize v1.0.0
go: downloading github.com/hashicorp/go-multierror v1.1.0
go: downloading github.com/pkg/errors v0.9.1
go: downloading github.com/terraform-farm/go-virtualbox v0.0.5-0.20210923155707-1e17d843762c
go: downloading github.com/hashicorp/errwrap v1.0.0
go: downloading github.com/hashicorp/go-hclog v0.15.0
go: downloading github.com/hashicorp/go-plugin v1.4.1
go: downloading github.com/hashicorp/terraform-plugin-go v0.3.0
go: downloading google.golang.org/grpc v1.32.0
go: downloading github.com/davecgh/go-spew v1.1.1
go: downloading github.com/hashicorp/terraform-exec v0.14.0
go: downloading github.com/hashicorp/terraform-json v0.12.0
go: downloading github.com/mitchellh/go-testing-interface v1.0.4
go: downloading github.com/hashicorp/go-cty v1.4.1-0.20200414143053-d3edf31b6320
go: downloading github.com/mitchellh/copystructure v1.2.0
go: downloading github.com/mitchellh/mapstructure v1.1.2
go: downloading github.com/fatih/color v1.7.0
go: downloading github.com/mattn/go-colorable v0.1.4
go: downloading github.com/mattn/go-isatty v0.0.10
go: downloading github.com/golang/protobuf v1.4.2
go: downloading github.com/hashicorp/yamux v0.0.0-20181012175058-2f1d1f20f75d
go: downloading github.com/oklog/run v1.0.0
go: downloading golang.org/x/net v0.0.0-20210326060303-6b1517762897
go: downloading github.com/hashicorp/go-version v1.3.0
go: downloading github.com/zclconf/go-cty v1.8.4
go: downloading github.com/hashicorp/logutils v1.0.0
go: downloading github.com/hashicorp/hcl/v2 v2.8.2
go: downloading github.com/hashicorp/go-uuid v1.0.1
go: downloading github.com/mitchellh/reflectwalk v1.0.2
go: downloading golang.org/x/text v0.3.5
go: downloading github.com/vmihailenco/msgpack v4.0.4+incompatible
go: downloading google.golang.org/protobuf v1.25.0
go: downloading google.golang.org/genproto v0.0.0-20200904004341-0bd0a958aa1d
go: downloading github.com/agext/levenshtein v1.2.2
go: downloading github.com/apparentlymart/go-textseg/v12 v12.0.0
go: downloading github.com/hashicorp/go-checkpoint v0.5.0
go: downloading github.com/hashicorp/go-cleanhttp v0.5.2
go: downloading github.com/hashicorp/go-getter v1.5.3
go: downloading golang.org/x/crypto v0.0.0-20210421170649-83a5a9bb288b
go: downloading github.com/mitchellh/go-wordwrap v1.0.0
go: downloading github.com/apparentlymart/go-textseg/v13 v13.0.0
go: downloading cloud.google.com/go/storage v1.10.0
go: downloading cloud.google.com/go v0.65.0
go: downloading github.com/aws/aws-sdk-go v1.37.0
go: downloading github.com/bgentry/go-netrc v0.0.0-20140422174119-9fd32a8b3d3d
go: downloading github.com/hashicorp/go-safetemp v1.0.0
go: downloading github.com/klauspost/compress v1.11.2
go: downloading github.com/mitchellh/go-homedir v1.1.0
go: downloading github.com/ulikunitz/xz v0.5.8
go: downloading google.golang.org/api v0.34.0
```

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 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)
  }
}
```

## 7. Вводжу команду, призначену для застосування змін, де підтверджую її виконання:

```
G:\Lab11\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.
```