Spis treści

[Add terraform to win11 3](#_Toc209036387)

[Login to azure 3](#_Toc209036388)

[Create project and resource group 3](#_Toc209036389)

[provider.tf 3](#_Toc209036390)

[variables.tf 3](#_Toc209036391)

[rg.tf 4](#_Toc209036392)

[outputs.tf 4](#_Toc209036393)

[Commands 5](#_Toc209036394)

[Add Storage 5](#_Toc209036395)

[storage.tf 5](#_Toc209036396)

[Add to output.tf 6](#_Toc209036397)

[Commands 6](#_Toc209036398)

[Add Networking (VNet, Subnet) 7](#_Toc209036399)

[Create network.tf 7](#_Toc209036400)

[Add to output.tf (existing one) 7](#_Toc209036401)

[Commands 7](#_Toc209036402)

[Add NSG (allow SSH) 8](#_Toc209036403)

[nsg.tf 8](#_Toc209036404)

[subnet-nsg.tf 8](#_Toc209036405)

[output.tf 9](#_Toc209036406)

[variables.tf 9](#_Toc209036407)

[commands 10](#_Toc209036408)

[Configure Public IP + NIC + bind NSG 10](#_Toc209036409)

[publicip.tf 10](#_Toc209036410)

[nic.tf 10](#_Toc209036411)

[outputs.tf 11](#_Toc209036412)

[commands 11](#_Toc209036413)

[Add Linux VM 11](#_Toc209036414)

[Vm.tf 11](#_Toc209036415)

[Output.tf 13](#_Toc209036416)

[Variables.tf 13](#_Toc209036417)

[Generate SHH for VM 13](#_Toc209036418)

[Commands 14](#_Toc209036419)

[Add nginx, html file and access for http, 14](#_Toc209036420)

[nsg.tf 14](#_Toc209036421)

[vm-extension.tf 14](#_Toc209036422)

[Commands 15](#_Toc209036423)

[Open website 15](#_Toc209036424)

[Add certificate to https 16](#_Toc209036425)

[Nsg.tf 16](#_Toc209036426)

[Commands 16](#_Toc209036427)

[Establish connection via secured shell 16](#_Toc209036428)

[publicId.tf 18](#_Toc209036429)

[Commands 19](#_Toc209036430)

## Add terraform to win11

[How to Install and Set Up Terraform on Windows (Step-by-Step Guide) - DEV Community](https://dev.to/kristarking/how-to-install-and-set-up-terraform-on-windows-step-by-step-guide-9md)

## Login to azure

az login

az account show --output table

# (optional) pick a subscription if you have more than one:

az account set --subscription "<YOUR\_SUBSCRIPTION\_ID>"

## Create project and resource group

mkdir advance-terraform

cd advance-terraform

### provider.tf

terraform {

required\_providers {

azurerm = { source = "hashicorp/azurerm", version = "~> 3.0" }

}

}

provider "azurerm" {

features {}

# If you want to pin explicitly:

# subscription\_id = "<YOUR\_SUBSCRIPTION\_ID>"

}

### variables.tf

variable "rg\_name" {

type = string

default = "advance-terraform"

description = "Name of the Resource Group"

}

variable "location" {

type = string

default = "westeurope" # closest to Poland

description = "Azure region"

}

variable "tags" {

type = map(string)

default = {

env = "dev"

app = "advance-terraform"

iac = "terraform"

}

description = "Common tags"

}

### rg.tf

resource "azurerm\_resource\_group" "rg" {

name = var.rg\_name

location = var.location

tags = var.tags

}

### outputs.tf

output "rg\_id" {

value = azurerm\_resource\_group.rg.id

description = "Resource Group ID"

}

output "rg\_name" {

value = azurerm\_resource\_group.rg.name

description = "Resource Group name"

}

### Commands

terraform init # downloads providers

terraform fmt # formats files

terraform fmt -check – check if are changes

terraform fmt -diff – show differances

terraform validate # syntax check

terraform plan # preview changes

terraform apply # actually create; confirm with 'yes'

az group show -n advance-terraform -o table – verify – in output you see resource name

terraform destroy – remove

## Add Storage

### storage.tf

# Unique suffix so the name is globally unique (required by Azure)

resource "random\_string" "sa\_suffix" {

length = 6

upper = false

special = false

}

# Storage Account (must be 3–24 chars, lowercase only, globally unique)

resource "azurerm\_storage\_account" "sa" {

name = "st${random\_string.sa\_suffix.result}"

resource\_group\_name = azurerm\_resource\_group.rg.name # same RG: advance-terraform

location = azurerm\_resource\_group.rg.location

account\_tier = "Standard"

account\_replication\_type = "LRS"

tags = var.tags

}

### Add to output.tf

output "storage\_account\_name" {

value = azurerm\_storage\_account.sa.name

description = "Storage account name"

}

Now it looks like

output "rg\_id" {

value = azurerm\_resource\_group.rg.id

description = "Resource Group ID"

}

output "rg\_name" {

value = azurerm\_resource\_group.rg.name

description = "Resource Group name"

}

output "storage\_account\_name" {

value = azurerm\_storage\_account.sa.name

description = "Storage account name"

}

### Commands

terraform init -upgrade # new provider "random" is used; this ensures it's installed

terraform fmt

terraform validate

terraform plan

terraform apply

## Add Networking (VNet, Subnet)

### Create network.tf

resource "azurerm\_virtual\_network" "vnet" {

name = "vnet-advance"

location = azurerm\_resource\_group.rg.location

resource\_group\_name = azurerm\_resource\_group.rg.name

address\_space = ["10.0.0.0/16"]

tags = var.tags

}

resource "azurerm\_subnet" "subnet" {

name = "subnet-advance"

resource\_group\_name = azurerm\_resource\_group.rg.name

virtual\_network\_name = azurerm\_virtual\_network.vnet.name

address\_prefixes = ["10.0.1.0/24"]

}

### Add to output.tf (existing one)

output "vnet\_name" {

value = azurerm\_virtual\_network.vnet.name

description = "Virtual Network name"

}

output "subnet\_name" {

value = azurerm\_subnet.subnet.name

description = "Subnet name"

}

### Commands

terraform fmt

terraform validate

terraform plan

terraform apply

## Add NSG (allow SSH)

### nsg.tf

resource "azurerm\_network\_security\_group" "nsg" {

name = "nsg-advance"

location = azurerm\_resource\_group.rg.location

resource\_group\_name = azurerm\_resource\_group.rg.name

security\_rule {

name = "Allow-SSH"

priority = 1001

direction = "Inbound"

access = "Allow"

protocol = "Tcp"

source\_port\_range = "\*"

destination\_port\_range = "22"

source\_address\_prefix = var.allowed\_ssh\_cidr # default is "\*" in your variables.tf

destination\_address\_prefix = "\*"

}

tags = var.tags

}

### subnet-nsg.tf

resource "azurerm\_subnet\_network\_security\_group\_association" "subnet\_nsg" {

subnet\_id = azurerm\_subnet.subnet.id

network\_security\_group\_id = azurerm\_network\_security\_group.nsg.id

}

### output.tf

# NSG basics

output "nsg\_name" {

value = azurerm\_network\_security\_group.nsg.name

description = "Network Security Group name"

}

output "nsg\_id" {

value = azurerm\_network\_security\_group.nsg.id

description = "Network Security Group resource ID"

}

# Show the SSH rule priority (useful sanity check)

output "nsg\_allow\_ssh\_priority" {

value = one([for r in azurerm\_network\_security\_group.nsg.security\_rule : r.priority if r.name == "Allow-SSH"])

description = "Priority of the Allow-SSH rule"

}

# Subnet ↔ NSG association (proves it’s attached)

output "subnet\_nsg\_association\_id" {

value = azurerm\_subnet\_network\_security\_group\_association.subnet\_nsg.id

description = "ID of the Subnet-NSG association"

}

### variables.tf

Add to existing file

variable "allowed\_ssh\_cidr" {

type = string

default = "\*" # quick start; later: "YOUR.PUBLIC.IP.ADDR/32"

description = "CIDR allowed to SSH to the VM"

}

### commands

terraform fmt

terraform validate

terraform plan

terraform apply

## Configure Public IP + NIC + bind NSG

### publicip.tf

resource "azurerm\_public\_ip" "pip" {

name = "pip-advance"

location = azurerm\_resource\_group.rg.location

resource\_group\_name = azurerm\_resource\_group.rg.name

allocation\_method = "Static" # keeps the IP stable

sku = "Basic" # fine for learning; "Standard" is also OK

tags = var.tags

}

### nic.tf

resource "azurerm\_network\_interface" "nic" {

name = "nic-advance"

location = azurerm\_resource\_group.rg.location

resource\_group\_name = azurerm\_resource\_group.rg.name

ip\_configuration {

name = "ipconfig1"

subnet\_id = azurerm\_subnet.subnet.id

private\_ip\_address\_allocation = "Dynamic"

public\_ip\_address\_id = azurerm\_public\_ip.pip.id

}

tags = var.tags

}

### outputs.tf

output "public\_ip\_address" {

value = azurerm\_public\_ip.pip.ip\_address

description = "Public IP that your future VM/NIC will use"

}

output "nic\_id" {

value = azurerm\_network\_interface.nic.id

description = "NIC resource ID"

}

### commands

terraform fmt

terraform validate

terraform plan

terraform apply

## Add Linux VM

### Vm.tf

resource "azurerm\_linux\_virtual\_machine" "vm" {

name = "vm-advance"

location = azurerm\_resource\_group.rg.location

resource\_group\_name = azurerm\_resource\_group.rg.name

size = "Standard\_B1s" # cheap learning size

admin\_username = var.admin\_username # from variables.tf

network\_interface\_ids = [

azurerm\_network\_interface.nic.id # your existing NIC

]

os\_disk {

caching = "ReadWrite"

storage\_account\_type = "Standard\_LRS"

}

# Ubuntu 22.04 LTS (Jammy)

source\_image\_reference {

publisher = "Canonical"

offer = "0001-com-ubuntu-server-jammy"

sku = "22\_04-lts"

version = "latest"

}

# SSH only (no passwords)

disable\_password\_authentication = true

admin\_ssh\_key {

username = var.admin\_username

public\_key = file(var.ssh\_public\_key\_path)

}

tags = var.tags

}

### Output.tf

output "vm\_public\_ip" {

value = azurerm\_public\_ip.pip.ip\_address

description = "Public IPv4 of the VM"

}

output "vm\_private\_ip" {

value = azurerm\_network\_interface.nic.ip\_configuration[0].private\_ip\_address

description = "Private IP from the subnet"

}

### Variables.tf

variable "ssh\_public\_key\_path" {

type = string

default = after generation

description = "Path to your SSH public key file"

}

variable "admin\_username" {

type = string

default = "azureuser" # change if you prefer

description = "Linux admin username used for SSH"

}

### Generate SHH for VM

Run script in cmd

ssh-keygen -t rsa -b 4096 -C "azure"

this is required to pass location in variables

variable "ssh\_public\_key\_path" {

type = string

**default = "C:/Users/micha/.ssh/id\_rsa.pub" # use forward slashes on Windows – changed after generation**

description = "Path to your SSH public key file"

}

### Commands

terraform fmt

terraform validate

terraform plan

terraform apply

## Add nginx, html file and access for http,

### nsg.tf

Add to nsg.tf to create access for HTTP

security\_rule {

name = "Allow-HTTP"

priority = 1002

direction = "Inbound"

access = "Allow"

protocol = "Tcp"

source\_port\_range = "\*"

destination\_port\_range = "80"

source\_address\_prefix = "\*" # later, tighten to your IP if you want

destination\_address\_prefix = "\*"

}

### vm-extension.tf

Create vm-extension.tf – during create will be installed ngrix and added html ile to display ‘First VM’ message

resource "azurerm\_virtual\_machine\_extension" "nginx" {

name = "nginx-bootstrap"

virtual\_machine\_id = azurerm\_linux\_virtual\_machine.vm.id

publisher = "Microsoft.Azure.Extensions"

type = "CustomScript"

type\_handler\_version = "2.1"

settings = jsonencode({

commandToExecute = "bash -c 'sudo apt-get update -y && sudo apt-get install -y nginx && echo \"<h1>First VM</h1>\" | sudo tee /var/www/html/index.html >/dev/null && sudo systemctl enable nginx && sudo systemctl restart nginx'"

})

}

### Commands

terraform fmt

terraform validate

terraform plan

terraform apply

### Open website

Got to vm

Obraz zawierający tekst, zrzut ekranu, Czcionka

Zawartość wygenerowana przez AI może być niepoprawna.

In network you have public ip address. Copy and go to <http://’publicIpAddress>’.

Result

Obraz zawierający tekst, zrzut ekranu, Czcionka, design

Zawartość wygenerowana przez AI może być niepoprawna.

## Add certificate to https

### Nsg.tf

Add to nsg

# HTTPS

security\_rule {

name = "Allow-HTTPS"

priority = 1003

direction = "Inbound"

access = "Allow"

protocol = "Tcp"

source\_port\_range = "\*"

destination\_port\_range = "443"

source\_address\_prefix = "\*"

destination\_address\_prefix = "\*"

}

### Commands

terraform fmt

terraform validate

terraform plan

terraform apply

### Establish connection via secured shell

Use ssh to connect to vm

ssh -i C:\Users\micha\.ssh\id\_rsa azureuser@<public id> - here we have information where is file with ssh and username from variables.tf

variable "ssh\_public\_key\_path" {

type = string

default = "C:/Users/micha/.ssh/id\_rsa.pub" # use forward slashes on Windows

description = "Path to your SSH public key file"

}

variable "admin\_username" {

type = string

default = "azureuser" # change if you prefer

description = "Linux admin username used for SSH"

}

After connection type **yes** to add access to vm.

Now we can change html file in vm

# (optional) backup the current file

sudo cp /var/www/html/index.html /var/www/html/index.html.bak 2>/dev/null || true

# write the new content

echo "<h1>First VM update</h1>" | sudo tee /var/www/html/index.html >/dev/null

# (usually not needed, but harmless if you do)

sudo systemctl reload nginx

the result it is

Obraz zawierający tekst, Czcionka, zrzut ekranu, Grafika

Zawartość wygenerowana przez AI może być niepoprawna.

At the moment we are sure we can manage installed software on vm so lets install certs.

# install certbot (snap is the recommended path on Ubuntu 22.04)

sudo snap install core; sudo snap refresh core

sudo snap install --classic certbot

sudo ln -s /snap/bin/certbot /usr/bin/certbot

add DNS in pip configuration

Obraz zawierający tekst, linia, zrzut ekranu, Czcionka

Zawartość wygenerowana przez AI może być niepoprawna.

Add cert to vm

sudo certbot --nginx -d <your-dns>.westeurope.cloudapp.azure.com --redirect

and result is

Obraz zawierający tekst, zrzut ekranu, Czcionka

Zawartość wygenerowana przez AI może być niepoprawna.

To close connection just write

exit

### publicId.tf

without manual just add one line to your publicId.tf file

resource "azurerm\_public\_ip" "pip" {

name = "pip-advance"

location = azurerm\_resource\_group.rg.location

resource\_group\_name = azurerm\_resource\_group.rg.name

allocation\_method = "Static"

sku = "Standard"

**# Add only this line this:**

**domain\_name\_label = "myvm-adv" # must be globally unique within the Azure region**

tags = var.tags

}

### Commands

terraform fmt

terraform validate

terraform plan

terraform apply