Jack Roper

Follow 13 Followers About

Understanding Azure Virtual Network Peering — Configuring in Terraform.

Jack Roper Sep 28, 2020 · 3 min read

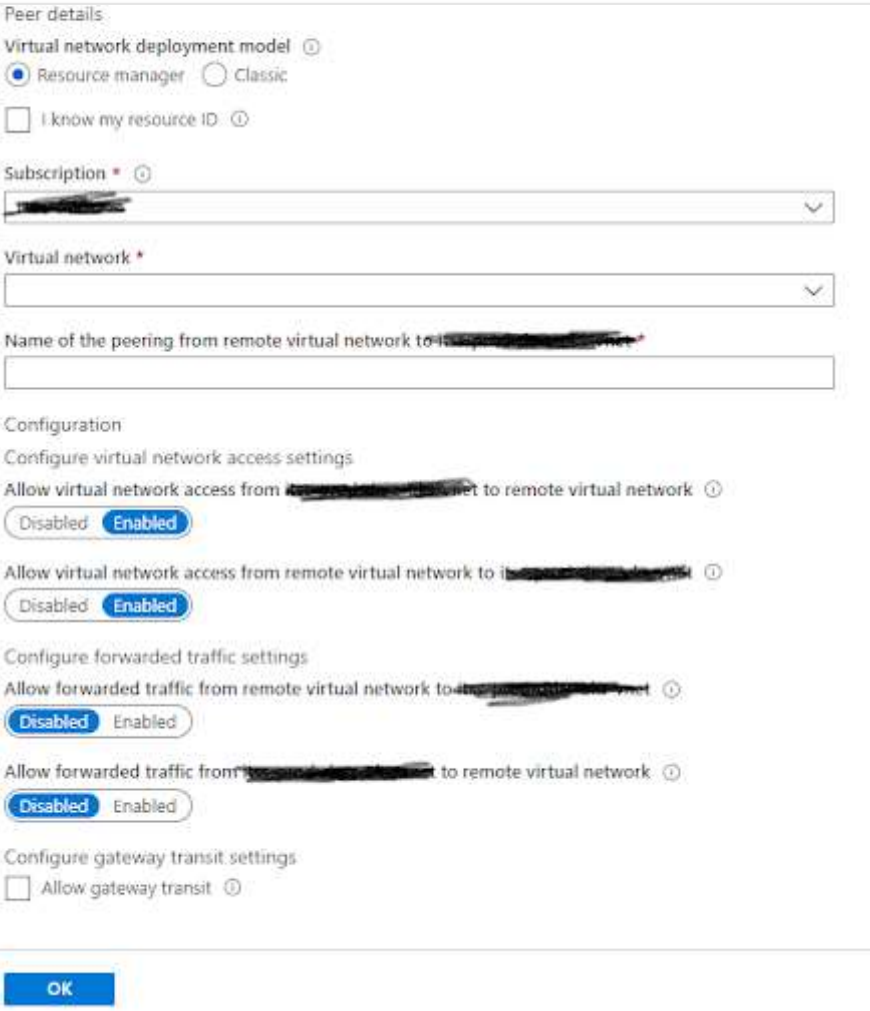
Recently I was involved in a project that required multiple network peerings to be setup between multiple Azure VNETS, with all the settings involved it can get quickly confusing so I thought I'd note down my findings here.

Azure Virtual Network Peering is a way to connect together Virtual networks across the same or different regions to allow traffic to flow between them. Network traffic is kept private and flows across the Microsoft backbone, providing a low-latency and high-bandwidth connection. For a peering to work, a peering link must be created both on the 'local' virtual network (i.e. the source network) and the 'remote' virtual network (i.e. the destination network).

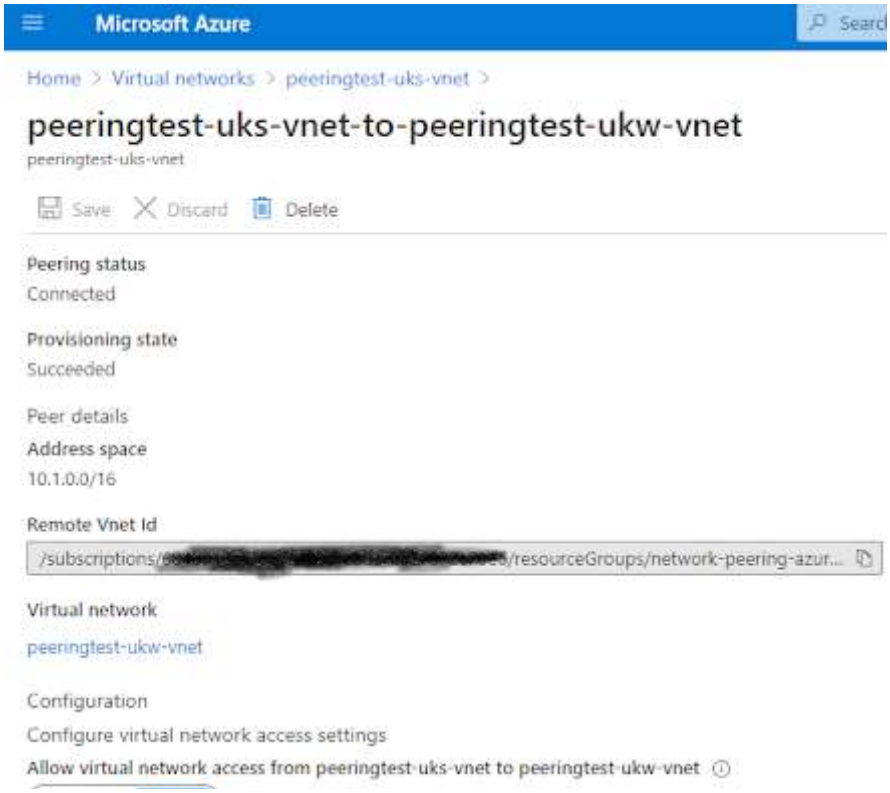
So that's the background out of way, lets get to the nitty gritty.

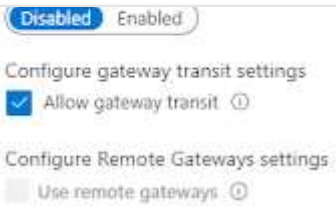
The first thing to know, is if the 2 VNETs you want to connect together have a Gateway subnet or not. If the remote VNET does have a gateway subnet you can use the 'allow gateway transit' setting to allow the traffic to from the local VNET use the remote gateway for transit. If the remote VNET does not have a gateway subnet, this setting cannot be used.

To add a peering through the portal, the following settings must be configured:



Once setup this actually shows as 4 variables in the portal:





In Terraform, 4 variables are of interest in the resource azurerm_virtual_network_peering:

- `allow_virtual_network_access` - (Optional) Controls if the VMs in the remote virtual network can access VMs in the local virtual network. Defaults to true.
- `allow_forwarded_traffic` - (Optional) Controls if forwarded traffic from VMs in the remote virtual network is allowed. Defaults to false.
- `allow_gateway_transit` - (Optional) Controls gatewayLinks can be used in the remote virtual network's link to the local virtual network.
- `use_remote_gateways` - (Optional) Controls if remote gateways can be used on the local virtual network. If the flag is set to `true`, and `allow_gateway_transit` on the remote peering is also `true`, virtual network will use gateways of remote virtual network for transit. Only one peering can have this flag set to `true`. This flag cannot be set if virtual network already has a gateway. Defaults to `false`.

A module can be defined as follows:

modules.tf

```
resource "azurerm_virtual_network_peering" "peering" {
  name                         = "${var.local_vnet_name}-to-${var.remote_vnet_name}"
  resource_group_name          = "${var.local_rg_name}"
  virtual_network_name         = "${var.local_vnet_name}"
  remote_virtual_network_id    = "${var.remote_vnet_id}"
  allow_virtual_network_access = "${var.allow_vnet_access}"
  allow_forwarded_traffic      = "${var.allow_forwarded_traffic}"
  allow_gateway_transit        = "${var.allow_gateway_transit}"
  use_remote_gateways          = "${var.use_remote_gateways}"
}
```

variables.tf

```
}

variable "local_rg_name" {
  description = "The name of the local virtual network rg"
  type        = "string"
}

variable "remote_vnet_name" {
  description = "The name of the remote virtual network"
  type        = "string"
}

variable "remote_vnet_id" {
  description = "The id of the remote virtual network"
  type        = "string"
}

variable "allow_vnet_access" {
  description = "Controls if the VMs in the remote virtual network can access VMs in t
  type        = "string"
}

variable "allow_forwarded_traffic" {
  description = "Controls if forwarded traffic from VMs in the remote virtual network
  type        = "string"
}

variable "allow_gateway_transit" {
  description = "Controls gatewayLinks can be used in the remote virtual network's lin
  type        = "string"
}
variable "use_remote_gateways" {
  description = "Controls if remote gateways can be used on the local virtual network.
  type        = "string"
}
```

Then call the module and pass in the variables:

```
module "vneta-to-vnetb" {
  source = "../modules/module.network.peering"

  providers = {
    azurerm = "azurerm"
  }

  local_vnet_name          = module.vneta.vnet_name
  local_rg_name            = module.vneta.networking_resource_group_name
  remote_vnet_name         = module.vnetb.vnet_name
  remote_vnet_id           = module.vnetb.vnet_id
  allow_vnet_access        = "true"
```

For more detail on the settings, check out the Microsoft docs:

https://docs.microsoft.com/en-gb/azure/virtual-network/virtual-network-manage-peering

And the Terraform docs here:

https://www.terraform.io/docs/providers/azurerm/r/virtual_network_peering.html

There is also a good article on PixelRobots showing how to acheive this using PowerShell:

https://pixelrobots.co.uk/2018/07/step-by-step-guide-on-setting-up-azure-vnet-peering

Thanks for reading and hope it helps clarify VNET peering using Terraform!

Originally published at https://azurelynot.blogspot.com on September 28, 2020.

Azure Azure Vnet Terraform