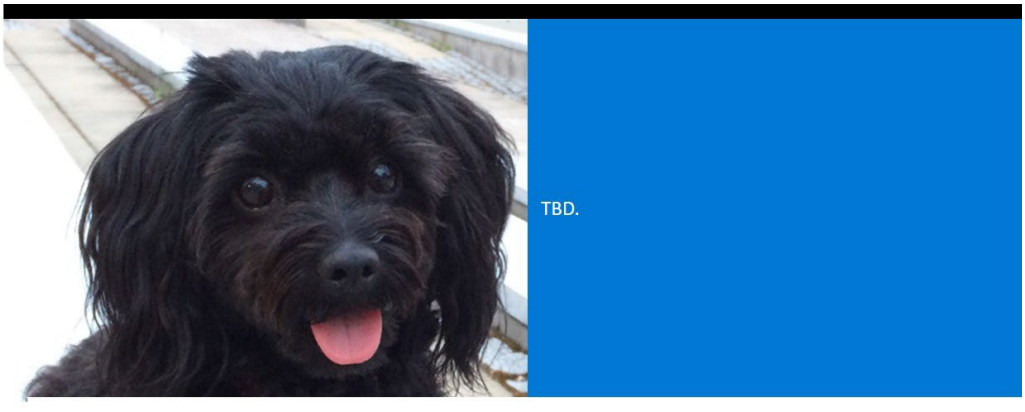
The Network Times
Handbook Series – Part VIII

Azure Networking Fundamentals

A Practical Guide to Understand How to Build a Virtual Datacenter into the AWS Cloud



Toni Pasanen, CCIE#28158

Azure Networking Fundamentals

A Practical Guide to Understand How to Build a Virtual Datacenter into the AWS Cloud

Toni Pasanen, CCIE 28158

Published – xx yy 2022

About the Editor:

Toni Pasanen. CCIE No. 28158 (RS), Distinguished Engineer at Fujitsu Finland. Toni started his IT carrier in 1998 at Tieto, where he worked as a Service Desk Specialist moving via the LAN team to the Data Center team as a 3rd. Level Network Specialist. Toni joined Teleware (Cisco Learning partner) in 2004, where he spent two years teaching network technologies focusing on routing/switching and MPLS technologies. Toni joined Tieto again in 2006, where he spent the next six years as a Network Architect before joining Fujitsu. In his current role, Toni works closely with customers helping them in selecting the right network solutions not only from the technology perspective but also from the business perspective. He is also the author of books:

Virtual Extensible LAN – VXLAN: The Practical Guide to VXLAN Solution –2019

LISP Control-Plane in Campus Fabric. A Practical Guide to Understand the Operation of Campus Fabric– 2020

VXLAN Fabric with BGP EVPN Control-Plane. Design Considerations – 2020

Object-Based Approach to Cisco ACI: The Logic Behind the Application Centric Infrastructure - 2020.

Cisco SD-WAN: A Practical Guide to Understand the Basics of Cisco Viptela Based SD-WAN Solution- 2021.

AWS Networking Fundamentals: A Practical Guide to Understand How to Build a Virtual Datacenter into the AWS Cloud-2021

About This Book

TBD

Disclaimers

The content of this book is based only on the author's own experience and testing results. Its content is neither validated nor accepted by Cisco or any other organization or person. This book is meant to be neither design nor an implementation guide. After reading this book, readers should do their technology validation before using it in a production environment.

Table of Contents

About This Book v

Chapter 1: Azure Virtual Network - VNet 1

Introduction 1
Geography, Region, and Availability Zone 1
Resources and Resource Group 1
Azure Active Directory and Subscription 2
Create Resource Group with Azure Portal 2
Create VNet with Azure Portal 7
Deploy VNet Azure Resource Manager Templates 14
Pre-Tasks 15
Deployment Template for VNet 17
Deployment Parameters for VNet 22
Deploying Process 25
Preferences 31

Chapter 1: Azure Virtual Network - VNet

Introduction

Geography, Region, and Availability Zone

The highlighted areas (*Geography*, *Region*, and *Availability Zones)* in figure 1-1 are related to Microsoft Datacenters and locations. Our example Geography Sweden comprises two Regions: Sweden Central and Sweden South. Sweden Central in Gävle is the primary Region. It hosts Microsoft's Azure, Office 365, and Microsoft Dynamics 365, which all are Microsoft's Software as a Service (SaaS) services. The second one, Sweden South (not shown in figure 1-1) in Staffanstorp, can only be used as a Disaster Recovery site for Sweden Central customers. Each Azure Region comprises three Availability Zones (AZ), which all have one or more DCs with dedicated DC infrastructure. AZs are connected with high-speed connections granting less than 2ms Round-Trip Time (RTT) between AZs. The Regional service is not affected if one of its three AZ is unreachable or down.

Resources and Resource Group

The *Virtual Network* (VNet), our virtual Datacenter, is a *Resource* under the *Networking Resource* category. Each VNet has an associated CIDR Range (Classless Inter-Domain Routing) from which we can allocate subnets. If we create a VNet-to-VNet connection, we need to use VNet-specific unique CIDR Ranges. VNets are Regional and cannot span between Regions. Subnets, in turn, can spread across regional Availability Zones. Each resource has to be associated with *Resource Group*. We can group application-specific resources/services under the same Resource Group. We can tear down the whole application by deleting its logical Resource Group as an example.

Azure Active Directory and Subscription

As I mentioned, Microsoft Azure, Office 365, and Microsoft Dynamics 365 are Software as a Service (SaaS) products. *Azure Active Directory* (AAD) is a system where we define access policies to these services. Our example AAD is The Network Times, where the administrator has created a *Subscription* NWKT for me. Subscription is like my playground. As an Owner, I can manage Resource Groups, Resources, and services under subscription.

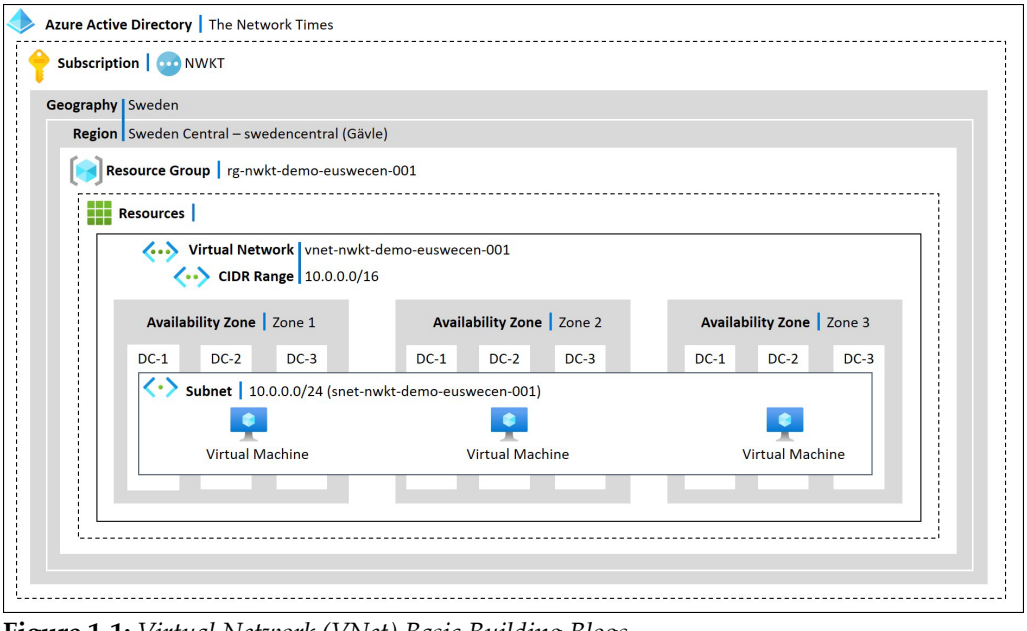


Figure 1-1: *Virtual Network (VNet) Basic Building Blogs.*

Create Resource Group with Azure Portal

Navigate to Azure Portal (portal.azure.com). Log in by giving your user credentials. After successful login, your Azure home page opens. Select the Subscriptions icon (1). In the *Subscription* view, you can see the list of your Subscriptions. In our example, we have only one Subscription NWKT. Next, click the NWKT Subscription hyperlink, and you'll be forwarded to the Subscription page.

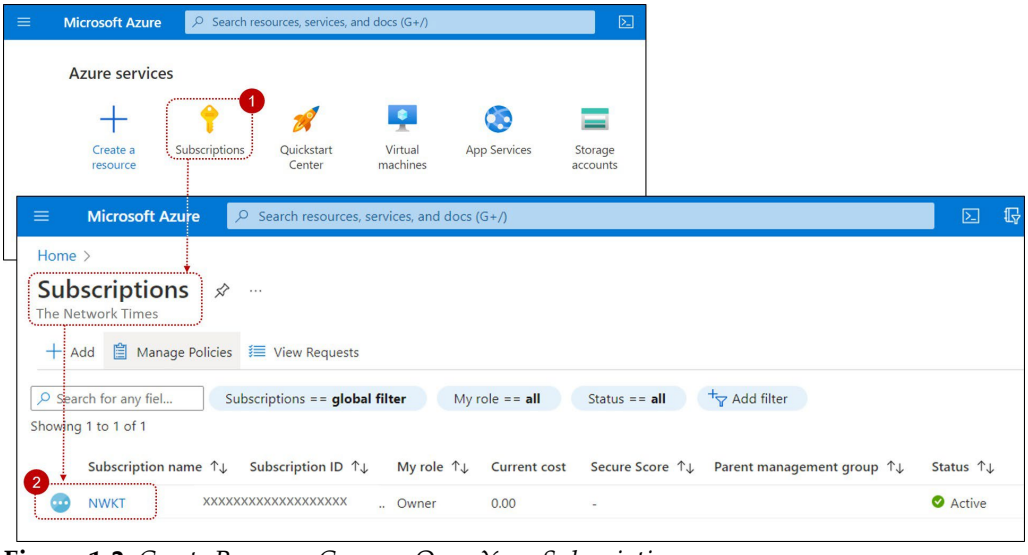


Figure 1-2: *Create Resource Group – Open Your Subscription.*

The Essentials section under the Overview option shows that I am the Owner of the Subscription NWKT and that the Subscription state is Active.

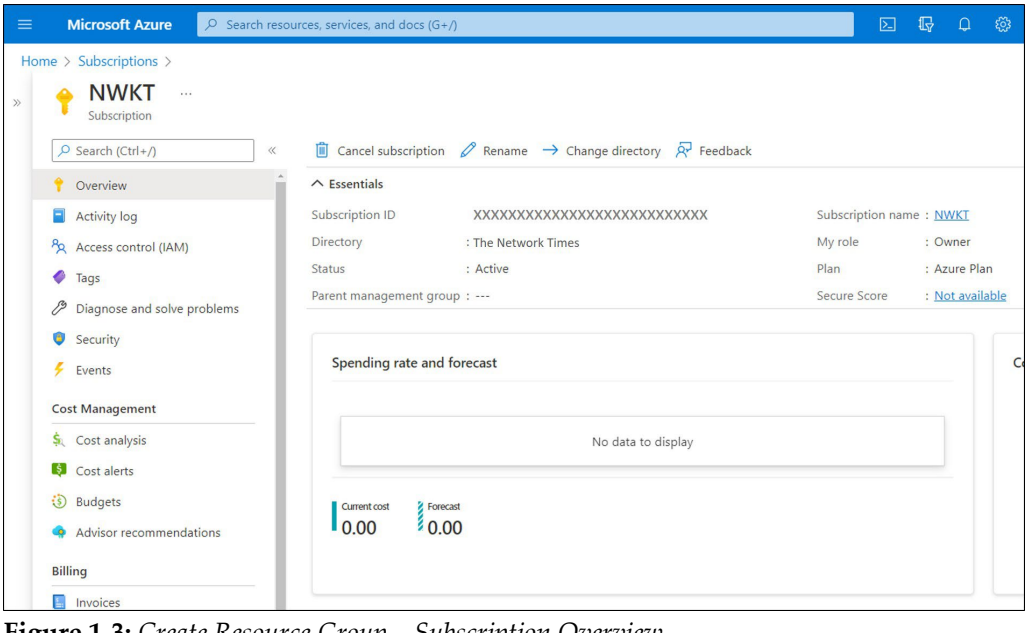


Figure 1-3: *Create Resource Group – Subscription Overview.*

Scroll down to the *Settings* section in the left pane and select *Resource Group* option. Then click the *Create* button on the taskbar.

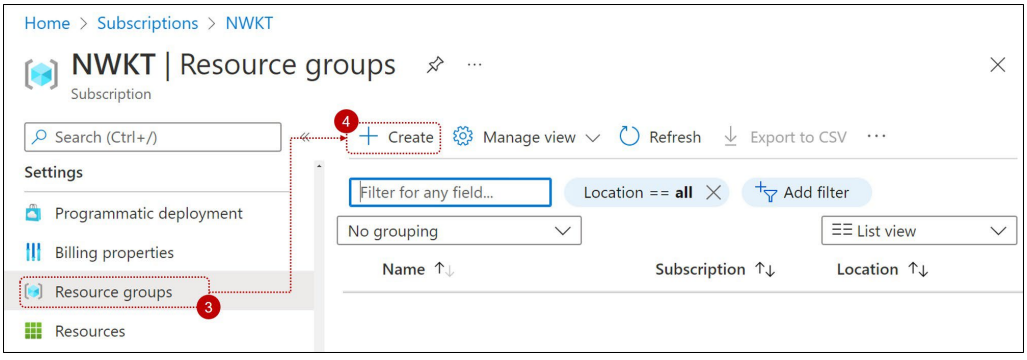


Figure 1-4: *Create Resource Group – Subscription Overview.*

Open the Basics tab in the Create a resource group window. First, Fill in the Project details information. Select your Subscription from the drop-down menu and give the name to Resource Group. I have used the naming convention recommended by Microsoft [1], which divides the name into five sections: Resource Type - Application – Environment - Region - Instance. I have also used recommended resource type abbreviation guideline [2]. I have named our example Resource Group rg-nwkt-demo-euswecen-001. Next, select the region where you want to launch the resource group. If you like to add tags to the resource group, click the Next: Tags > button or select the Tags tab.

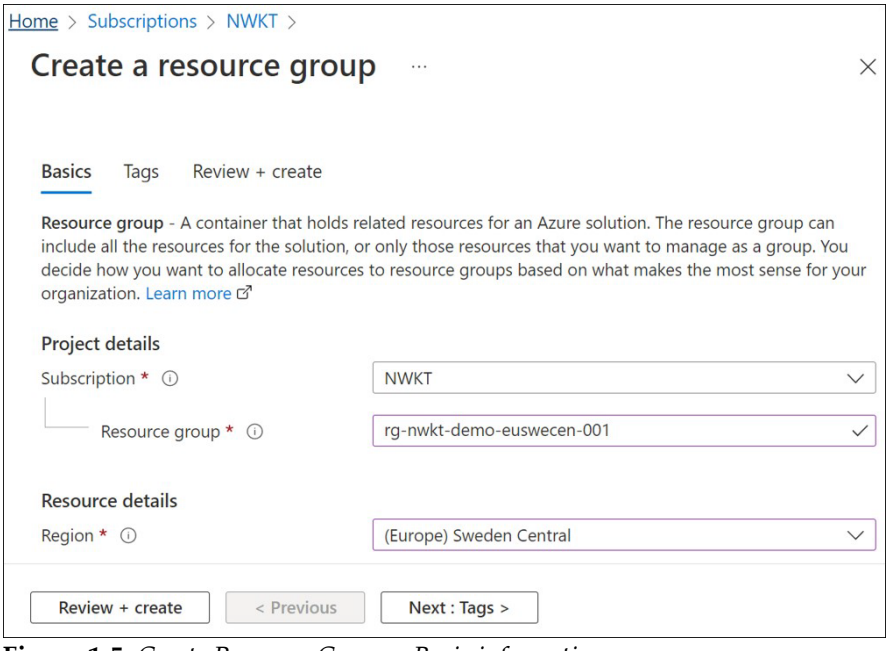


Figure 1-5: *Create Resource Group – Basic information.*

We can group application-specific services and resources under the same resource group. However, we might have a dedicated resource group for the development, testing, and production environment. Using *Tags*, you can give an informative description to resource groups or other Azure resources. You can also view resource-specific billing information based on tags. I have added six Tags to our example resource group: application name (=demo), confidentially (=public), SLA (=none), department (=education), the owner (Toni), and purpose (=education). Note that once you have created tags, you can use them with any of your Azure resources groups or resources. When you have created tags, click the *Next: Review + create* button or select *Review + create* tab.

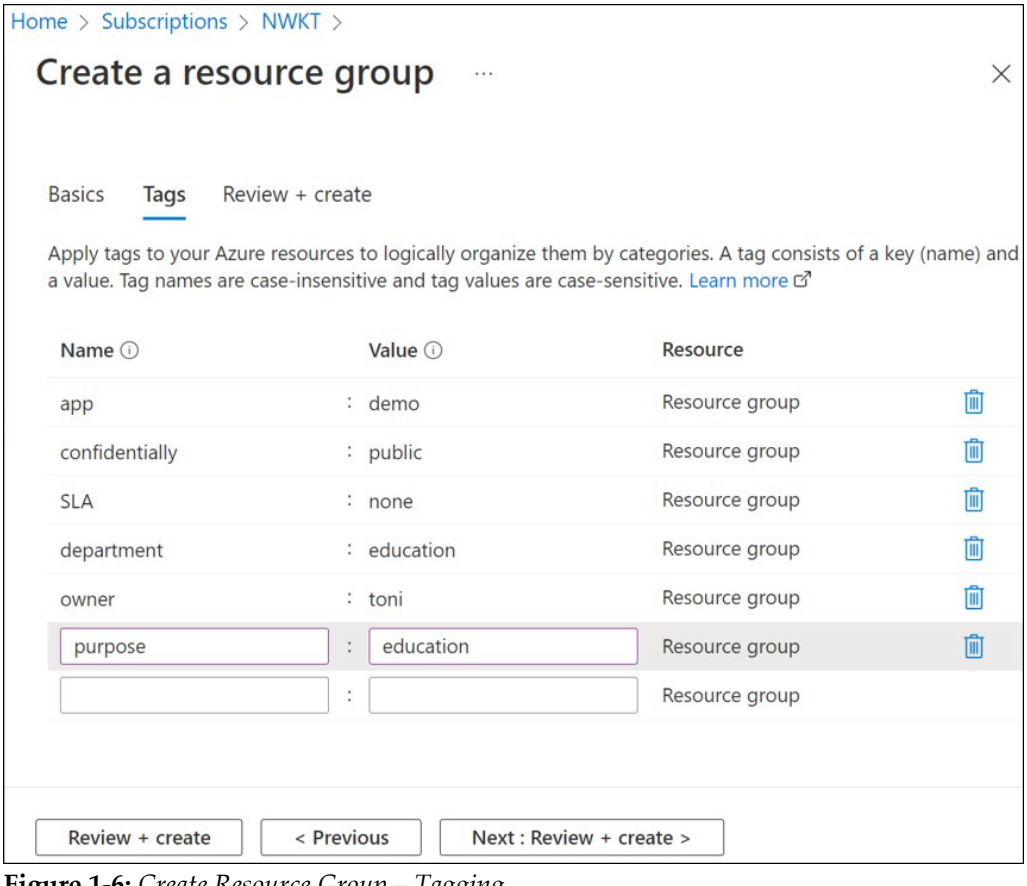


Figure 1-6: *Create Resource Group – Tagging.*

Figure 1-7 shows the Review window. You can download template and parameters files by clicking the hyperlink *Download a template for automation*. The figure below shows the parameters.json file. I will go through the various automation options later. Click the *Create* button for deploying your resource group.

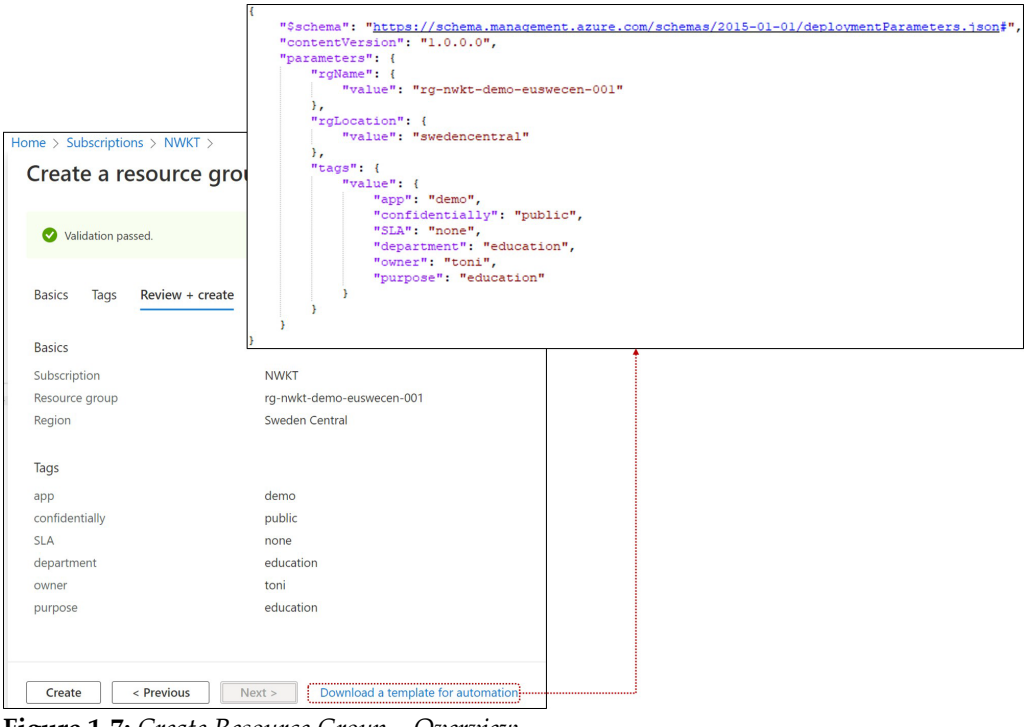


Figure 1-7: *Create Resource Group – Overview.*

Figure 1-8 shows three resource groups under the NWKT subscription. You can filter out unwanted resource groups by using tags. In our example, I have built two filters, which show resource groups with tags: app==demo and purpose==education.

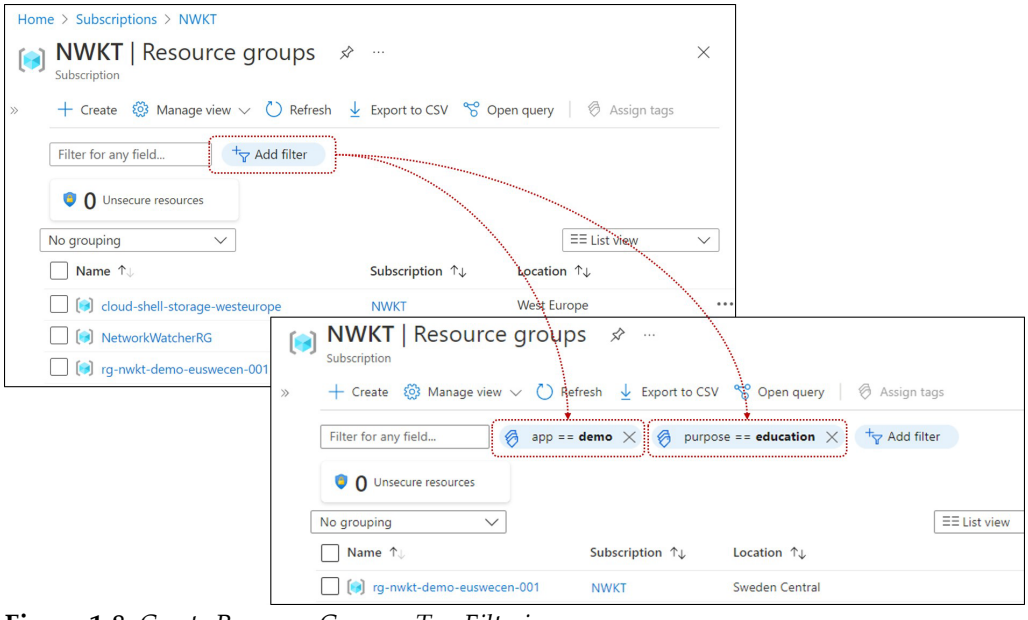


Figure 1-8: *Create Resource Group – Tag Filtering.*

Create VNet with Azure Portal

After creating a resource group, navigate back to the subscription window and select the *Resources* option under the *Settings* section on the left pane. Then click the Create button. Click the *Networking* option from the *Categories* pane on the left pane. Then select Create under Virtual Network.

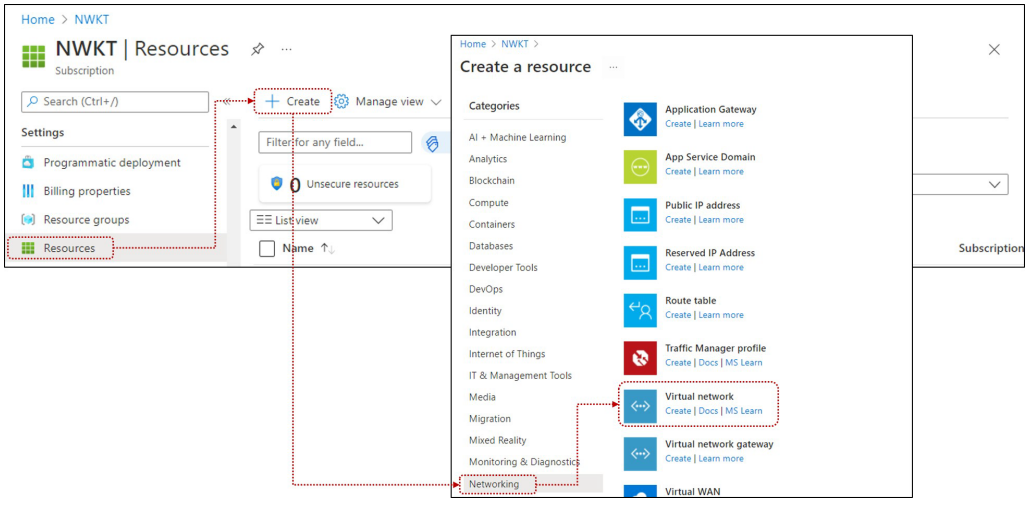


Figure 1-9: *Create Resource – VNet: Step-1.*

Open the Basics tab in the Create virtual network window. Fill in the Project details information. Select your Subscription from the drop-down menu and give the name to Resource Group. These steps are the same that what we did when creating a resource group. You can also create a new resource group in this window. Give the name to VNet and choose the region where you want to launch the resource group. I have used the same naming structure as what I used with the Resource Group (Resource Type - Application – Environment - Region - Instance). Next, click the Next : IP Address button or select it from the tab bar.

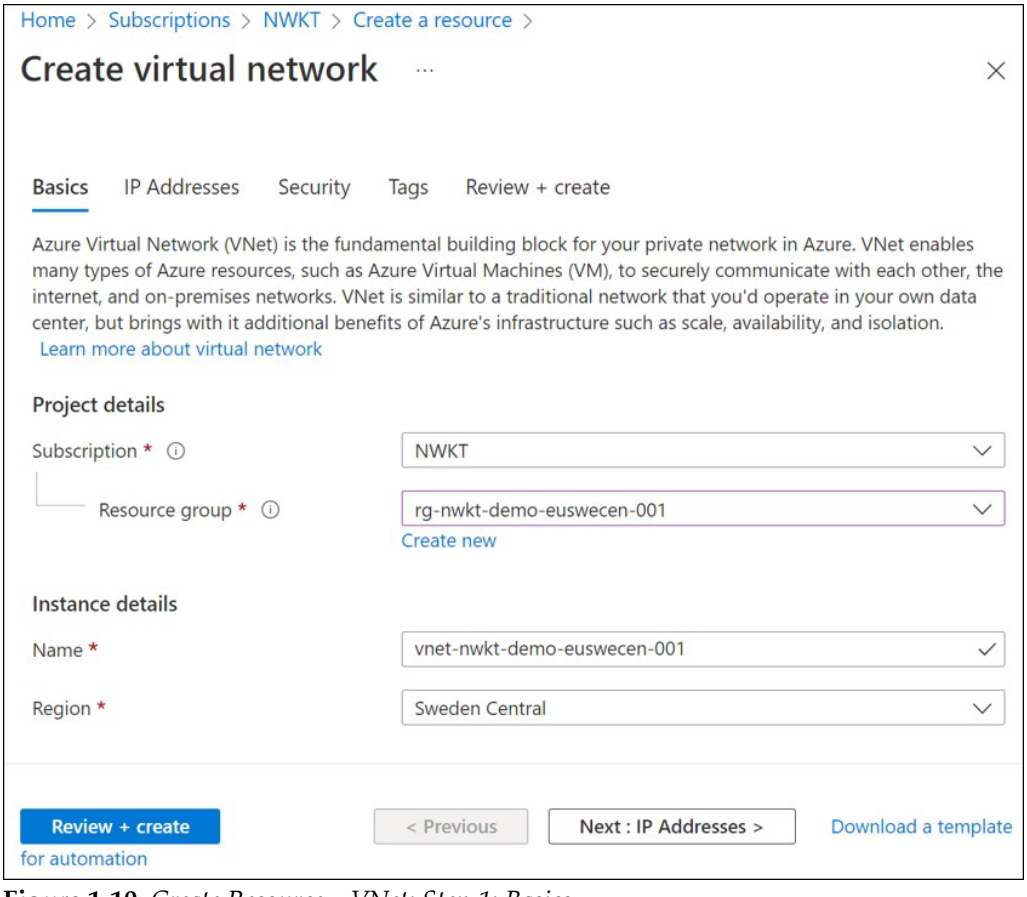


Figure 1-10: *Create Resource – VNet: Step-1: Basics.*

Azure propose the CIDR 10.0.0.0/16 by default. If you have already assigned it to other VNet, Azure propose some other CIDR. The default subnet is the first C-Class subnet with /24 mask. You can edit the name and subnet by clicking the default hyperlink that opens the Edit subnet window. Once again, I have used the same naming as with Resource Group and VNet.

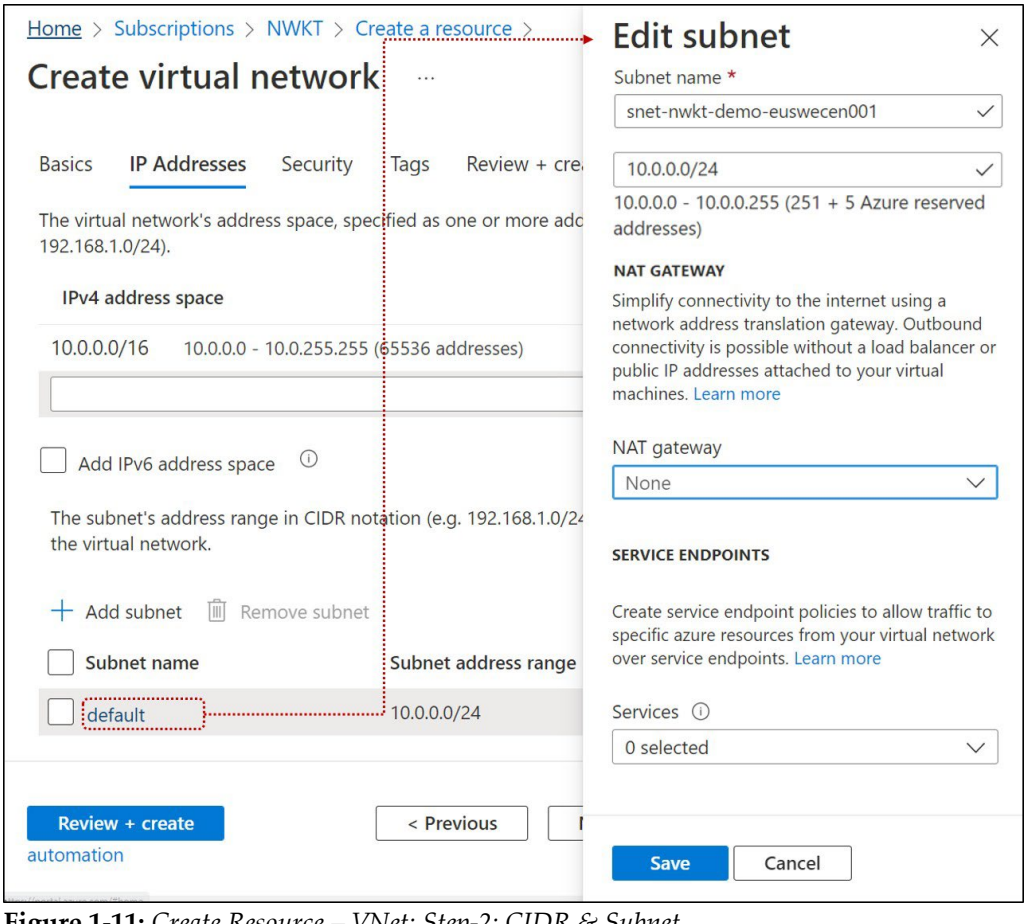


Figure 1-11: *Create Resource – VNet: Step-2: CIDR & Subnet.*

We are not implementing VNet Security features in this example so continue to Tags tab. We can reuse Tags which we created in the Resource Group example. The list of exciting Tags will open when you click the Name field. Select tags you want to use with this VNet. When you have selected the tag, click the Value field and you'll see values associated with the tag. In this example we are using the same tags with VNet as we used with Resource Group. When you have added all necessary Tags, select the Review + create.

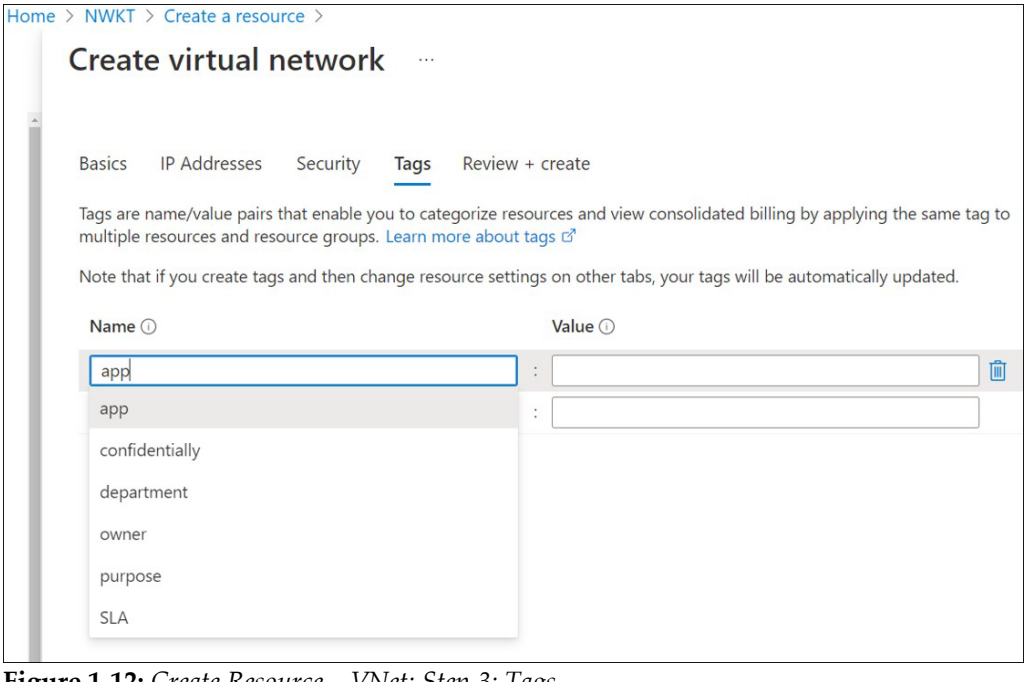


Figure 1-12: *Create Resource – VNet: Step-3: Tags.*

Figure 1-13 shows the Review window. You can download template and parameters files by clicking the hyperlink *Download a template for automation* as we did in the resource Group example. Click the Create button for deploying net VNet.

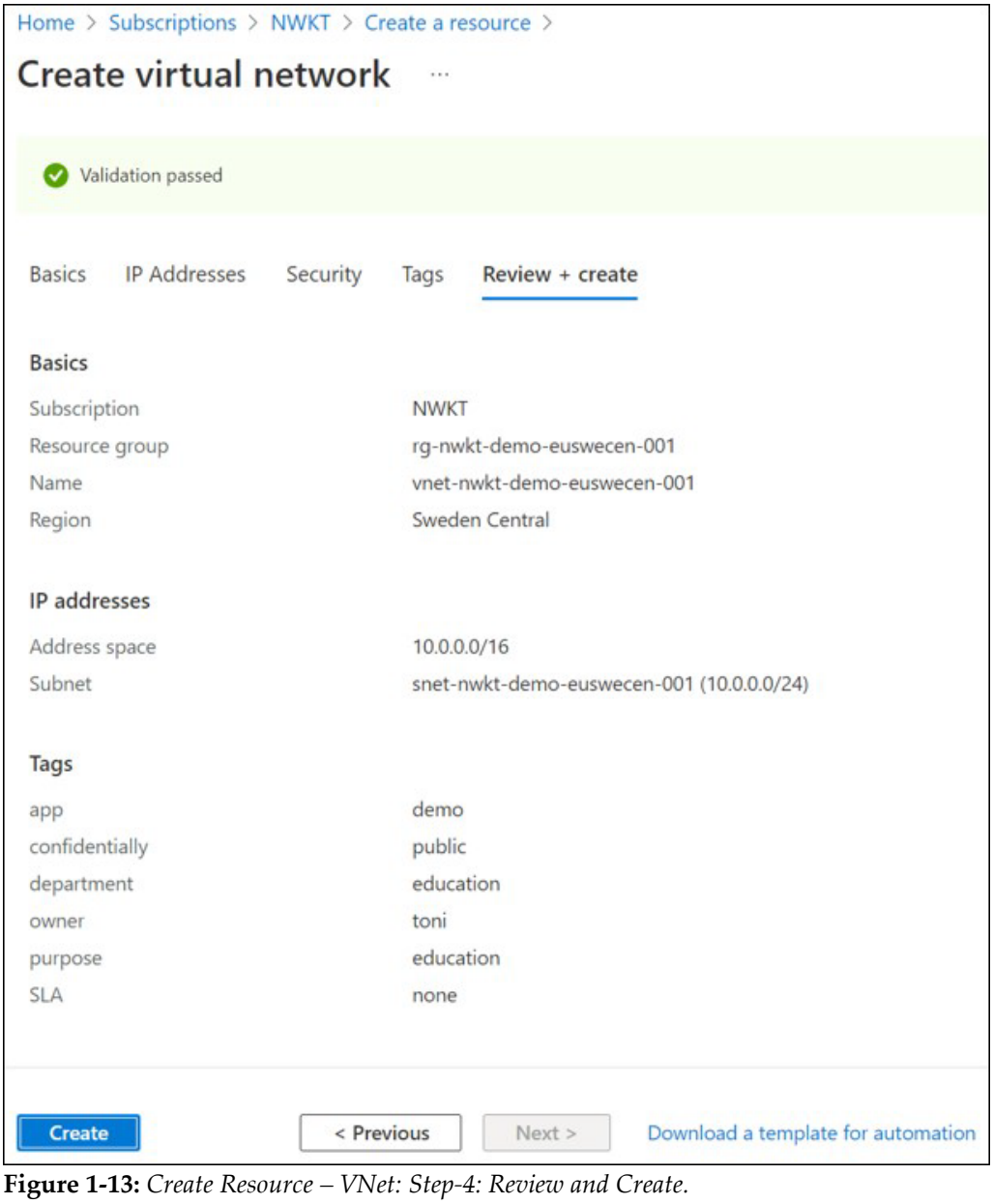


Figure 1-13: *Create Resource – VNet: Step-4: Review and Create.*

Figure 1-14 shows the status of the deployment (complete). You can click the Go to resource button to proceed to vnet-nwkt-euswecen-001 page shown in figure 1-15.

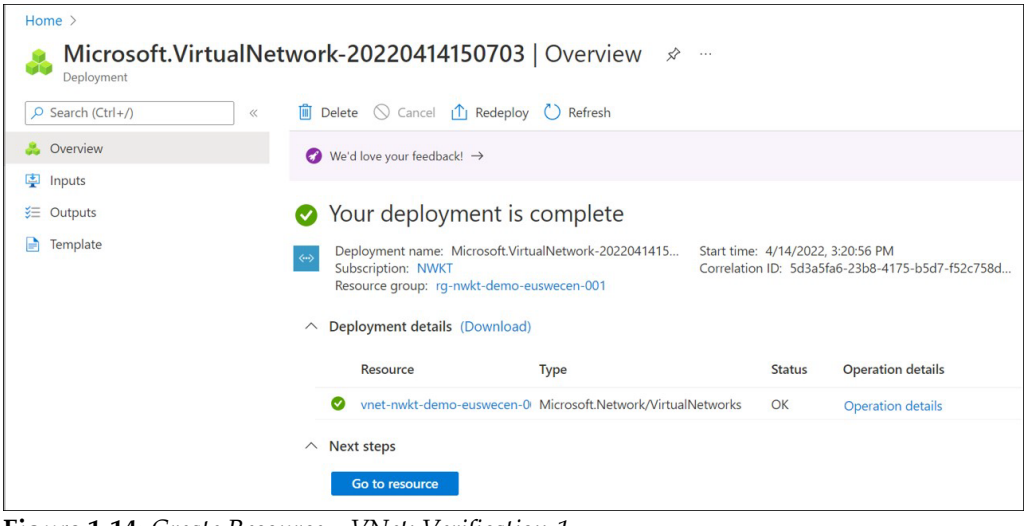


Figure 1-14: *Create Resource – VNet: Verification-1.*

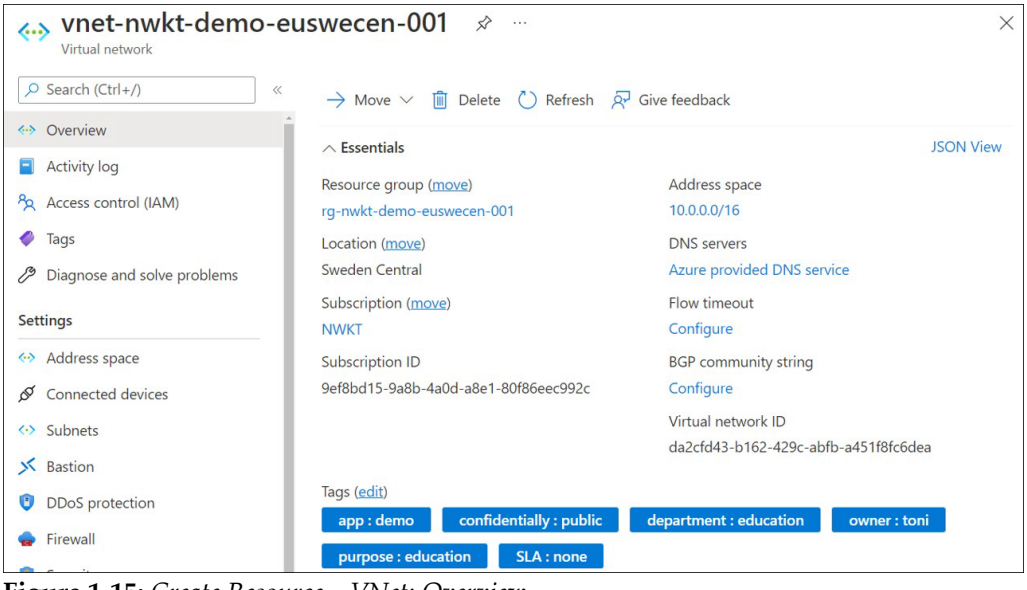


Figure 1-15: *Create Resource – VNet: Overview.*

The Address space option under the Settings section shows the CIDR we assigned to VNet.

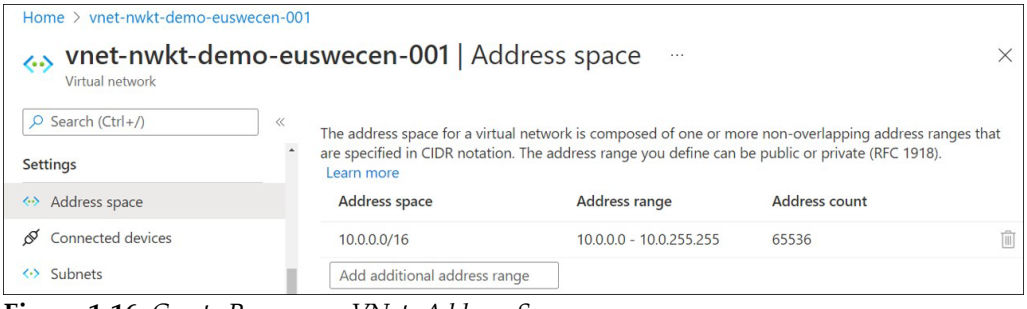


Figure 1-16: *Create Resource – VNet: Address Space.*

The Subnet option under the Settings section shows the subnet 10.0.0.0/24, which we create during the VNet implementation process.

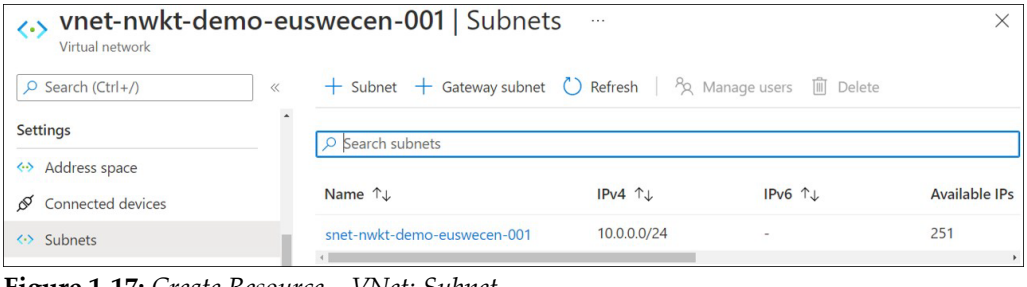


Figure 1-17: *Create Resource – VNet: Subnet.*

Deploy VNet Azure Resource Manager Templates

Azure Resource Manager (ARM) is a management service platform which enables us to manage Azure resources using Azure Portal, Azure CLI, Azure PowerShell and SDK. In this section we focus on how to deploy VNets resources using Visual Studio Code and its ARM Templates extension [8]. ARM Templates describes our resource in JSON format. Figure 1-18 illustrates the workflows and overall building blocks and processes. Our intent is to create two VNets, one for the production workloads and the other one for development workloads. First, we create an ARM Deployment Template (1), which will be a base template for our development and production VNets. In this template we define the name policy for VNet and its subnet. We also define allowed tags associated with VNets. However, we don't describe the actual names for these objects. Next, we create dedicated ARM Deployment Parameters template for production (2a) and development (2b) VNets. These templates define values for objects vnetName, tagName, and snetName. At this phase, we have three ARM templates, one common template for VNets and two VNet-specific parameters templates. When we have done templates, we create VNet-specific Azure PowerShell scripts (3a-b). These scripts create JSON files, where deployed resources are taking from the VNet base template. The Deployment Template and its parameters values, in turn, are taken from the VNet specific Deployment Parameters template. The script itself, defines a resource group and its location. The script deploys these JSON files to ARM. Before converting the deployment file to REST API call, ARM verifies our user credentials (authentication) and check do we have rights to create VNets (authorization). If our deployment passes the security checks, ARM makes a REST API call to Resource Provider (Microsoft.Network) and its specific Service (Virtual Network) to finish the deployment (4a-b). The Deployment Template is also validated before implementation.

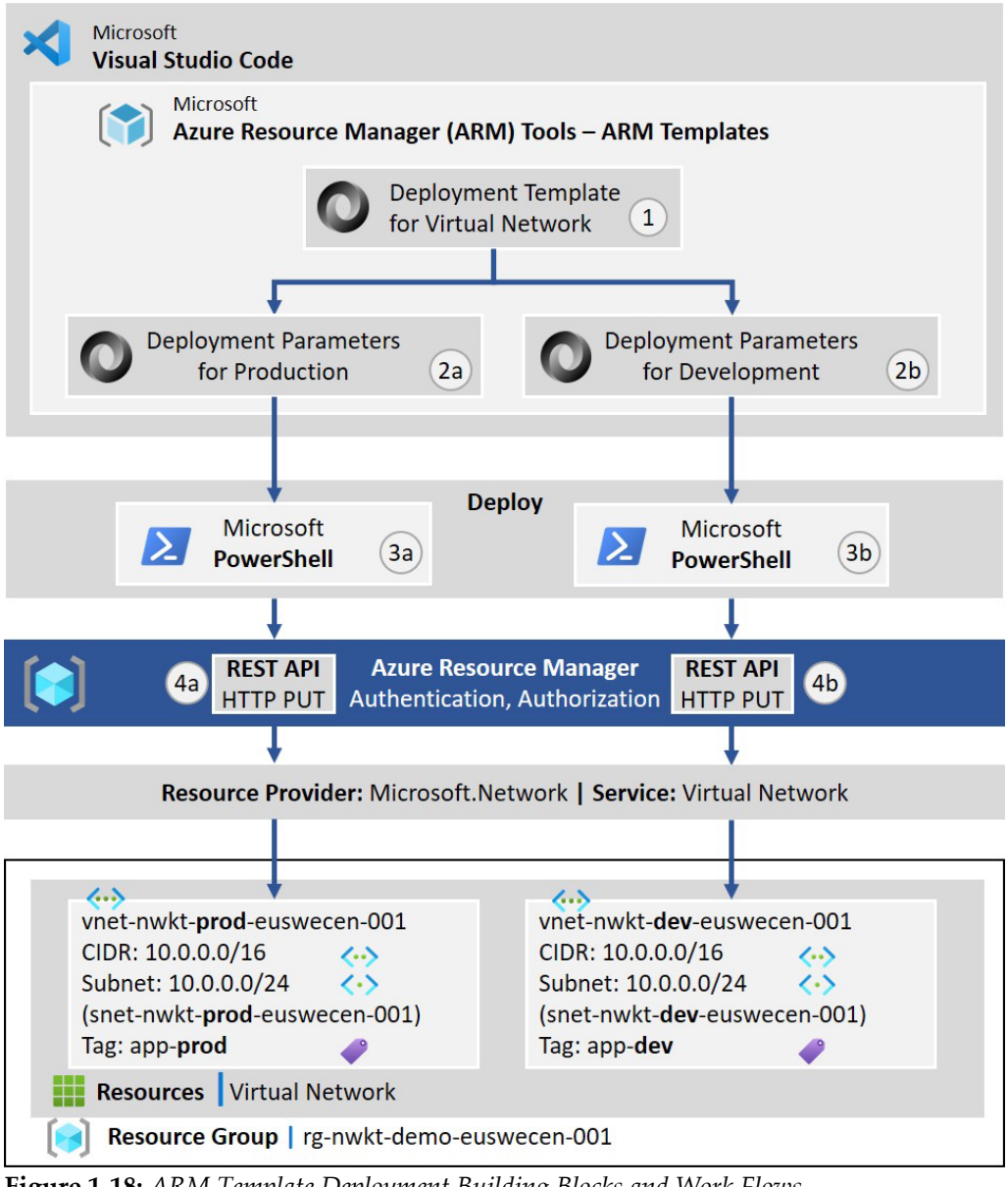


Figure 1-18: *ARM Template Deployment Building Blocks and Work Flows.*

Pre-Tasks

The first thing to do is install the Microsoft Visual Studio Code. You can download the latest version from code.visualstudio.com. You also need to install the Azure Resource Manager Tools (ARM) Tools extension. We are using these tools for creating ARM templates.

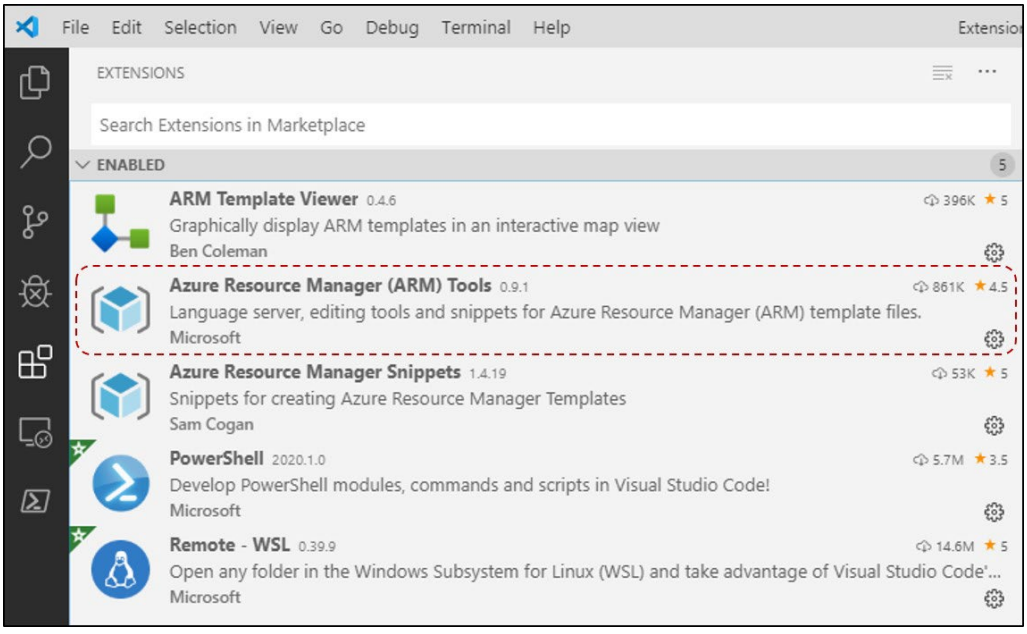


Figure 1-19: *Microsoft Visual Code Studio and the ARM Tools Extension.*

You also need to install the Azure RM PowerShell or the newer one Azure AZ PowerShell module for PowerShell. Examples 1-1 and 1-2 shows how I first log in to my Azure account with PowerShell and then verify that the AzureRM module is installed.

```
PS C:\Users\fitonpasa> Login-AzureRmAccount

Account                SubscriptionName    TenantId      Environment
-------                ----------------    --------      -----------
toni.pasanen@***.com   NWKT                d944c18e-...  AzureCloud
```

Example 1-1: *Login to Azure Account.*

```
PS C:\Users\fitonpasa> Get-InstalledModule -Name AzureRM

Version    Name         Repository         Description
-------    ----         ----------         -----------
6.13.2     AzureRM      PSGallery          Azure Resource Manager Module
```

Example 1-2: *Verifying Installed Modules.*

Deployment Template for VNet

In Visual Studio Code (VSC), click the File tab from the menu bar and select New File. The default file type is text (.txt). In order to use ARM tools extension, you have to save the file in JSON format. I have named the example file as vnet-template.json. When you type the *arm* into first row, you'll get the list of arm-starting options. Select the *arm!,* Azure Resource Manager (ARM) template from the list.

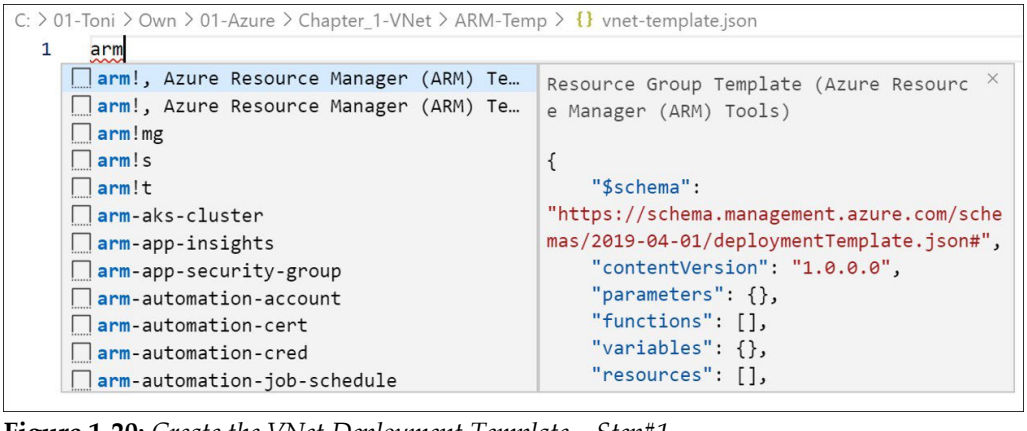


Figure 1-20: *Create the VNet Deployment Template – Step#1.*

The *arm!,* option opens the deployment template that lists basic elements of the JSON schema file. There are three required elements for every JSON file: *$schema, contentVersion,* and *resources*. The *$schema* element describes the location and version of the top-level schema file (schmema.management.azure.com, version: 2019-04-01). The *contentVersion* element tells the version. Though it is a mandatory element, we don't have to use it for versioning. However, it is good practice to update the version numbering when modifying an existing file. The *Resources* array are used for creating resource objects and their attributes. Our focus is on Resource arrays and Parameter objects (not mandatory).

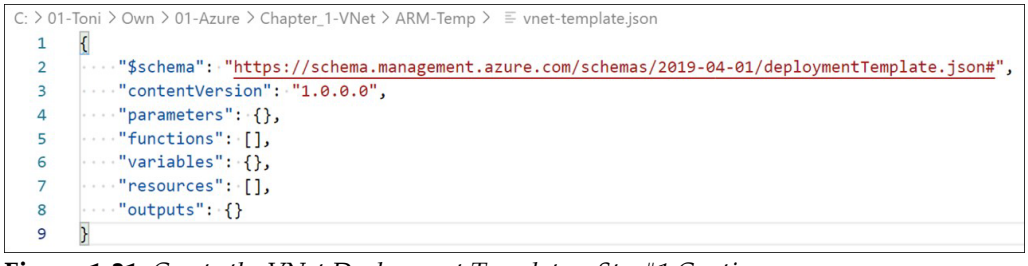


Figure 1-21: *Create the VNet Deployment Template – Step#1 Continues.*

Next, we modify the Resources array. Square brackets [ ] after the Resource states that the following JSON structure is an array. An array contains zero or more JSON values (object, array, number, string, true or false, or null). We are going to create an object for VNet. Point the cursor between square brackets and then press the enter. Type the vne, and you'll get the list of all options that name includes vne. Select the arm-vnet. It is a template for virtual networks.

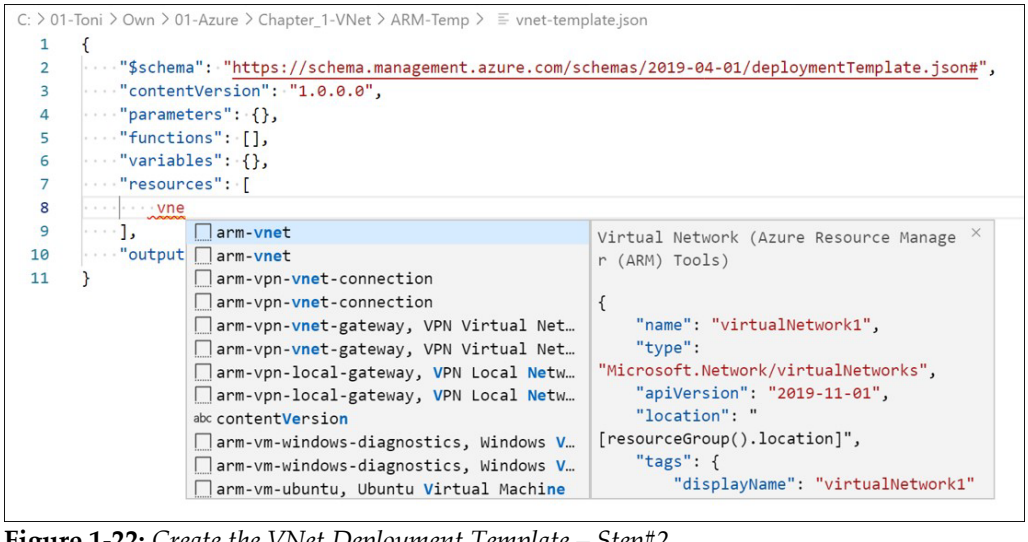


Figure 1-22: *Create the VNet Deployment Template – Step#2.*

JSON object is represented within curly brackets { }. The object includes zero or more name/value pairs. Figure 1-23 shows the *vnet-template.json* template file where all arrays (values) and objects (name/value) are with their defaults. The name/value pair in row 10 describes the Service Provider and its Service. The apiVersion in this template is 2019-11-01. You can find the template format of each API version for Virtual Network resource by navigating to https://docs.microsoft.com/en-us/azure/templates. Then type Virtual Network in the Filter by title field. Select the Virtual Network > Reference > Network > (API versions) option from the list. The name location is the same as the location of the Resource Group where we deploy our VNet. The Properties object defines CIDR Range (as an array) and subnets (as an object).

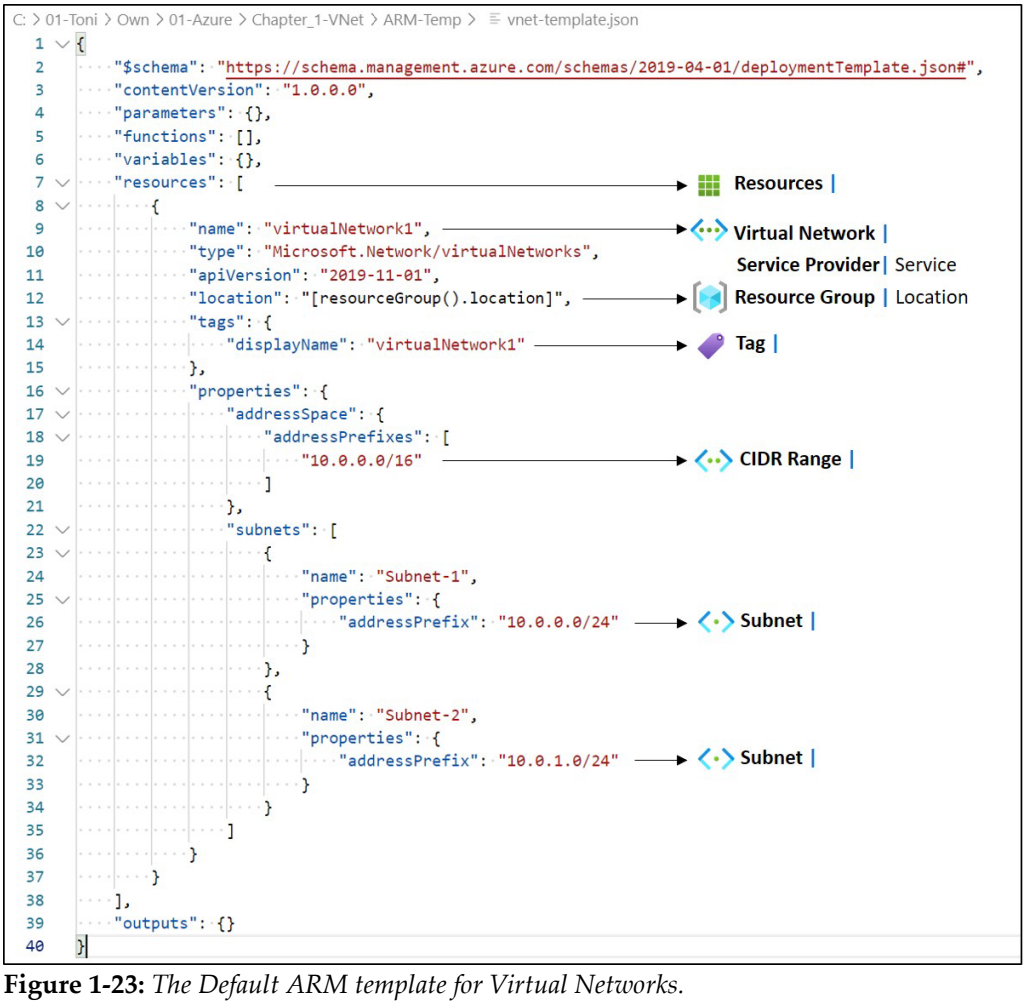


Figure 1-23: *The Default ARM template for Virtual Networks.*

We intend to use *vnet-template* for deploying both development and production VNets. We use the same CIDR range 10.0.0.0/16 and subnet 10.0.0.0/24 with both VNets. However, we use the naming structure, which describes the resource type - application – environment - region – instance. We are also using the tag naming model, which describes the environment. Now, instead of using default values, we are using *functions*. First, replace the *"virtualNetworks"* value with square brackets between the quotation marks *"[ ]"*. Click between square brackets and type *para.* Select the *parameters* function from the list. By doing this, line 25 is changed to ***"name": "[parameters()]",*** add the *'vnetName'* within brackets. At this phase, line 25 is **"name": "[parameters('vnetName')]"**. This function calls the vnetName object (lines 5) under the parameters object (line 4). The vnetName object under the parameters element defines the naming policy, not the name itself. The type of the name is a string, and the min length is 12 characters, and the max length is 24 characters. If we violate this policy, the validation process fails. The function *parameters* under object *tags,* calls *a parameters* element on line 4 and its object *tagName* on line 10. The *allowedValues* array defines values (dev or prod) that we can use. Just for a recap, allowedValues is an array which includes strings without names. If we try to use any other tags than what we have defined in allowedValues, the validation process notices that, and the deployment fails.

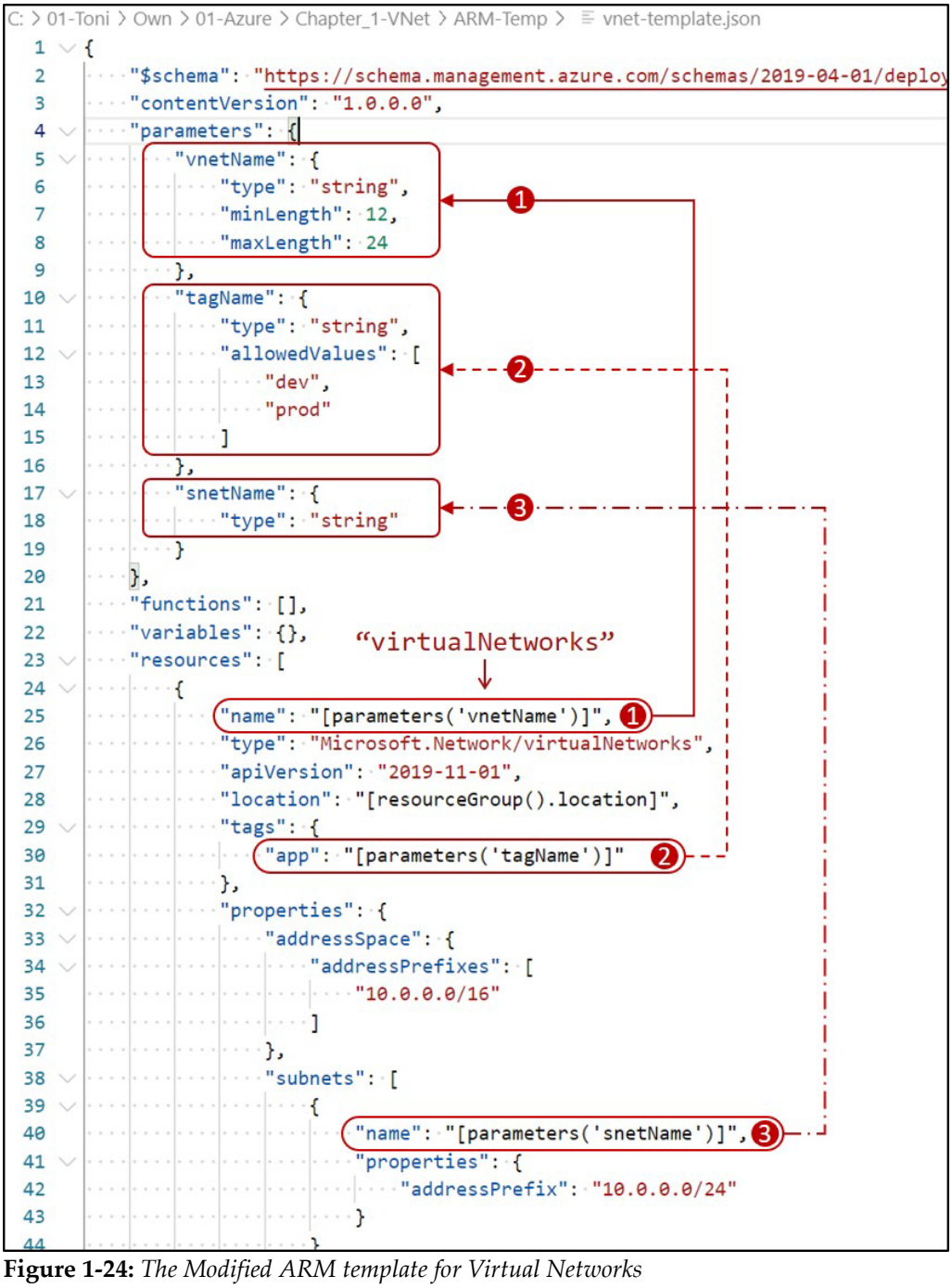


Figure 1-24: *The Modified ARM template for Virtual Networks*

Deployment Parameters for VNet

After building a deployment template JSON file (vnet-template.json), we create separate deployment parameters JSON files for the development and production VNets. Open a new JSON file in Visual Studio Code. Type *armp* on the first line and select *armp!* From the list.

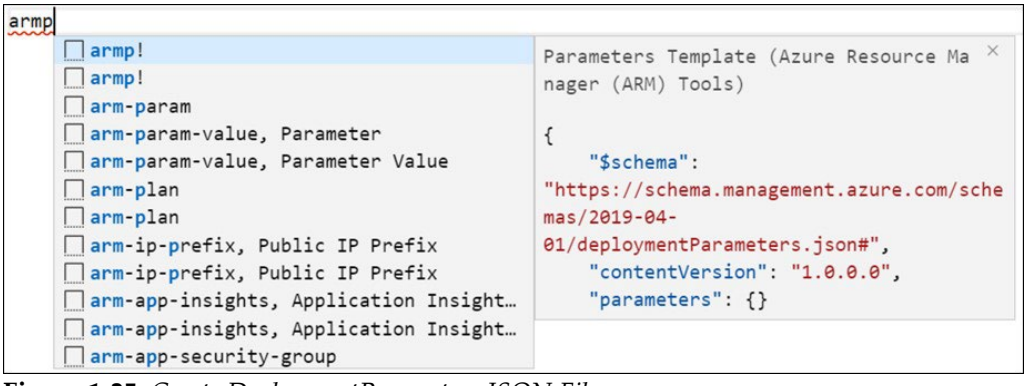


Figure 1-25: *Create DeploymentParameters JSON File.*

Figure 1-26 illustrates the JSON file vnet-parameters-prod.json. It includes three mandatory elements $schema, contentVersion, and parameters. The value of the *vnetName* object describes the vnet name *vnet-nwkt-prod-euswecen-001*. The value of the *tagName* object attaches the tag *prod* to VNet. The value of the *snetName* describes the subnet name *snet-nwkt-prod-euswecen-001*.

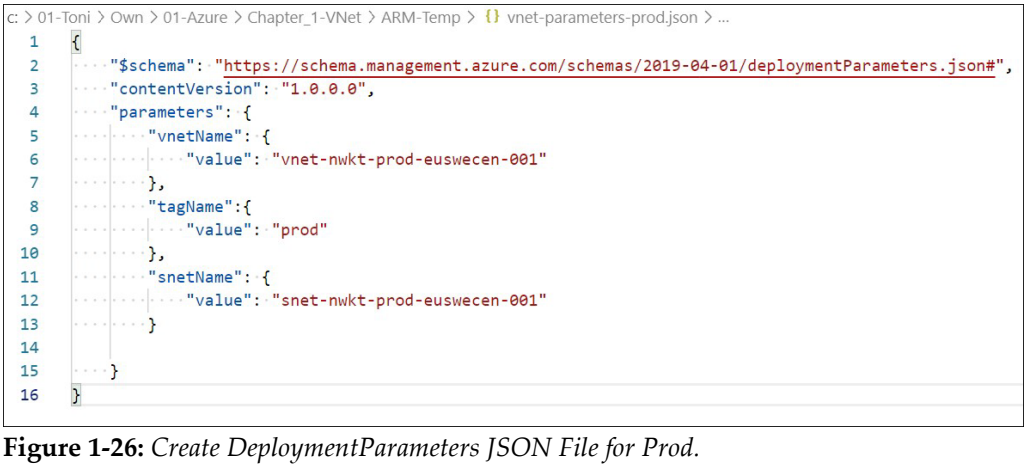


Figure 1-26: *Create DeploymentParameters JSON File for Prod.*

Figure 1-27 shows the JSON file vnet-parameters-dev.json.

C: > 01-Toni > Own > 01-Azure > Chapter_1-VNet > ARM-Temp > {} vnet-parameters-dev.json > {} parameters

```
{
    "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentParameters.json#",
    "contentVersion": "1.0.0.0",
    "parameters": {
        "vnetName": {
            "value": "vnet-nwkt-dev-euswecen-001"
        },
        "tagName":{
            "value": "dev"
        },
        "snetName": {
            "value": "snet-nwkt-dev-euswecen-001"
        }

    }
}
```

Figure 1-27: *Create DeploymentParameters JSON File for Dev.*

When using functions for naming, we have to use equal names for called objects in both Parameter and Deployment JSON files. Figure 1-28 shows that the name value calls the *parameters* object and its *vnetName* object in Deployment Template. These objects have to use the same name.

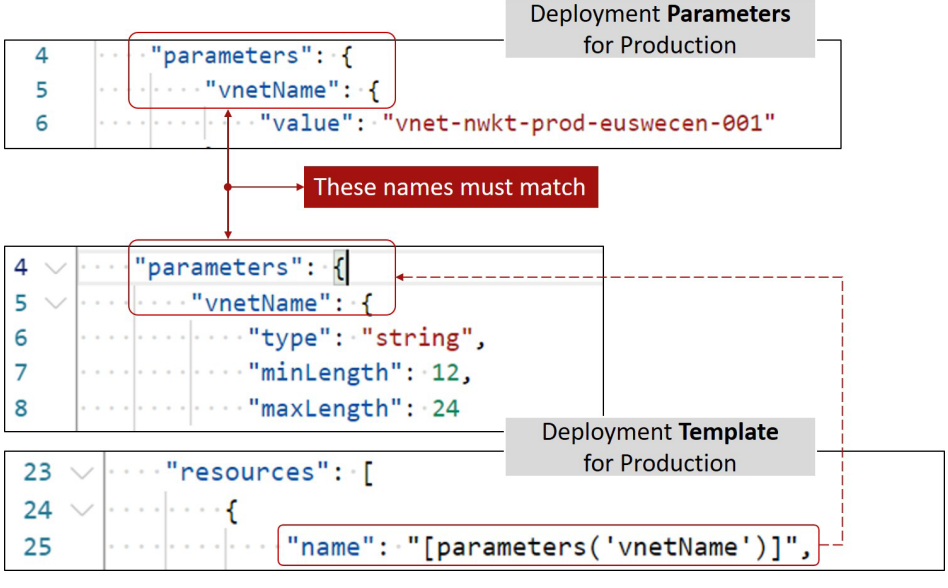


Figure 1-28: *Create DeploymentParameters JSON File for Dev.*

Figure 1-29 lists the Resource Groups we have before we implement new virtual networks.

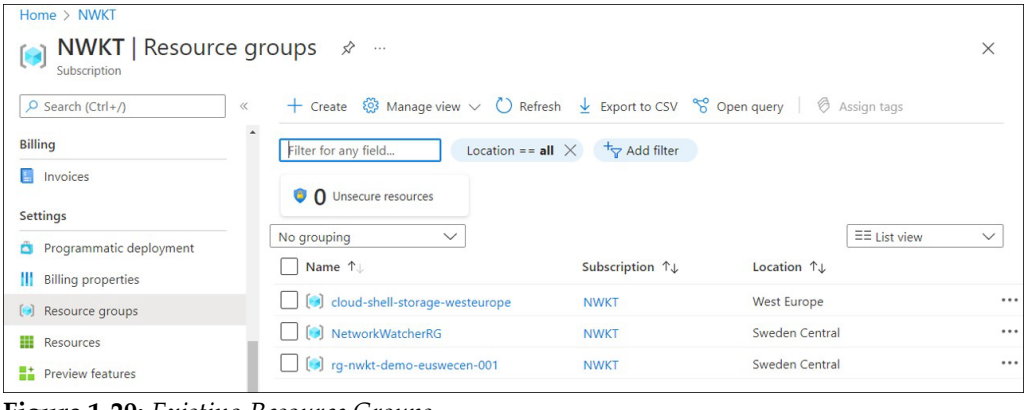


Figure 1-29: *Existing Resource Groups.*

Figure 1-30 verifies that before we implement new virtual networks, there is only previously created VNet.

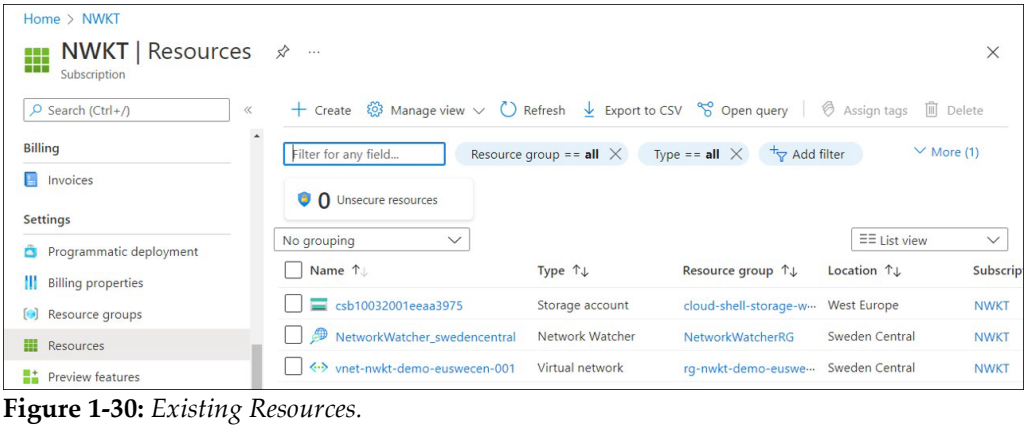


Figure 1-30: *Existing Resources.*

Deploying Process

Example 1-3 shows the PowerShell script for creating the VNet *vnet-nwkt-dev-euswecen-001*. The Azure RM PowerShell command *New-AzureRmResource Group* deploys the Resource Group on Azure Region swedencentral using the name defined on the first row. The *Force* command at the end of the line executes the command without asking for user confirmation. The second command, *New-AzureRmResourceGroupDeployment,* deploys the Resource described in the file *vnet-template.json* using parameters from the file *vnet-parameters-dev.json.* The new resource will be launched in RG rg-arm-demo-euswecen-011. Note that the -Name rg-arm-demo-dev-001 is used as a Deployment name.

```
$rg = 'rg-arm-demo-euswecen-011'
New-AzureRmResourceGroup -Name $rg -Location swedencentral -Force
New-AzureRmResourceGroupDeployment `
    -Name 'rg-arm-demo-dev-011' `
    -ResourceGroupName $rg `
    -TemplateFile 'C:\01-Toni\...\ARM-Temp\vnet-template.json' `
    -TemplateParameterFile 'C:\01-Toni...\ARM-Temp\vnet-parameters-dev.json'
```

Example 1-3: *PowerShell Script for VNet Development Deployment (vnet-deploy-dev.ps1).*

Example 1-3 shows the PowerShell script that creates *vnet-nwkt-prod-euswecen-001* into the same region than *vnet-nwkt-dev-euswecen-001*.

```
$rg = 'rg-arm-demo-euswecen-011'
New-AzureRmResourceGroup -Name $rg -Location swedencentral -Force
New-AzureRmResourceGroupDeployment `
    -Name 'rg-arm-demo-prod-011' `
    -ResourceGroupName $rg `
    -TemplateFile 'C:\01-Toni\...\ARM-Temp\vnet-template.json' `
    -TemplateParameterFile 'C:\01-Toni...\ARM-Temp\vnet-parameters-prod.json'
```

Example 1-4: *PowerShell Script for VNet Production Deployment (vnet-deploy-prod.ps1).*

Example 1-5 illustrates the deployment process. First, we run the *vnet-deploy-dev.ps1* command. Note that you have to use .\ prefix if the executed PowerShell file is in the same directory where you run the command. After a short delay (time depends on the deployment size), Azure gives feedback. The first section shows that we have successfully provisioned a new Resource Group to the swedencentral region. The second section verifies that we have successfully deployed a new VNet. It also describes the parameters associated with resources.

```
PS C:\01-Toni\Own\01-Azure\Chapter_1-VNet\ARM-Temp> .\vnet-deploy-dev.ps1

ResourceGroupName : rg-arm-demo-euswecen-011
Location          : swedencentral
ProvisioningState : Succeeded
Tags              :
ResourceId        : /subscriptions/***/rg-arm-demo-euswecen-011

ResourceGroupName       : rg-arm-demo-euswecen-011
OnErrorDeployment       :
DeploymentName          : rg-arm-demo-011
CorrelationId           : 40731f62-6d40-45e0-822f-5f4c7f9587aa
ProvisioningState       : Succeeded
Timestamp               : 3.5.2022 8.36.51
Mode                    : Incremental
TemplateLink            :
TemplateLinkString      :
DeploymentDebugLogLevel :
Parameters              : {[vnetName,
Microsoft.Azure.Commands.ResourceManager.Cmdlets.SdkModels.DeploymentVariable],
[tagName,
Microsoft.Azure.Commands.ResourceManager.Cmdlets.SdkModels.DeploymentVariable],
[snetName,
Microsoft.Azure.Commands.ResourceManager.Cmdlets.SdkModels.DeploymentVariable]}
ParametersString        :
                          Name             Type             Value
                          ===============  ===============  ==========
                          vnetName         String           vnet-nwkt-dev-euswecen-001
                          tagName          String           dev
                          snetName         String           snet-nwkt-dev-euswecen-001

Outputs                 : {}
OutputsString           :
```

Example 1-5: *Running PowerShell "script vnet-deploy-dev.ps1".*

Example 1-6 shows the deployment state and result when running the PowerShell script vnet-deploy-prod.ps1 script.

```
PS C:\01-Toni\Own\01-Azure\Chapter_1-VNet\ARM-Temp> .\vnet-deploy-prod.ps1

ResourceGroupName : rg-arm-demo-euswecen-011
Location          : swedencentral
ProvisioningState : Succeeded
Tags              :
ResourceId        : /subscriptions/***/resourceGroups/rg-arm-demo-euswecen-011


ResourceGroupName       : rg-arm-demo-euswecen-011
OnErrorDeployment       :
DeploymentName          : rg-arm-demo-011
CorrelationId           : e70bab8b-1a7e-4704-b5f0-c76a7929e646
ProvisioningState       : Succeeded
Timestamp               : 3.5.2022 8.38.34
Mode                    : Incremental
TemplateLink            :
TemplateLinkString      :
DeploymentDebugLogLevel :
Parameters              : {[vnetName,
Microsoft.Azure.Commands.ResourceManager.Cmdlets.SdkModels.DeploymentVariable],
[tagName, Microsoft.Azure.Commands.ResourceManager.Cmdlets.SdkModels.Dep
                          loymentVariable], [snetName,
Microsoft.Azure.Commands.ResourceManager.Cmdlets.SdkModels.DeploymentVariable]}
ParametersString        :
                          Name             Type             Value
                          ===============  ===============  ==========
                          vnetName         String           vnet-nwkt-prod-euswecen-001
                          tagName          String           prod
                          snetName         String           snet-nwkt-prod-euswecen-001

Outputs                 : {}
OutputsString           :
```

Example 1-6: *Running PowerShell "script vnet-deploy-prod.ps1".*

We can also verify from the Azure GUI that we now have a new Recourse Group rg-arm-demo-euswecen-011 (figure 1-31), new resources vnet-nwkt-prod-euswecen-001, and vnet-nwkt-dev-euswecen-001 (figure 1-32) with their associated subnets (figures 1-33, and 1-34).

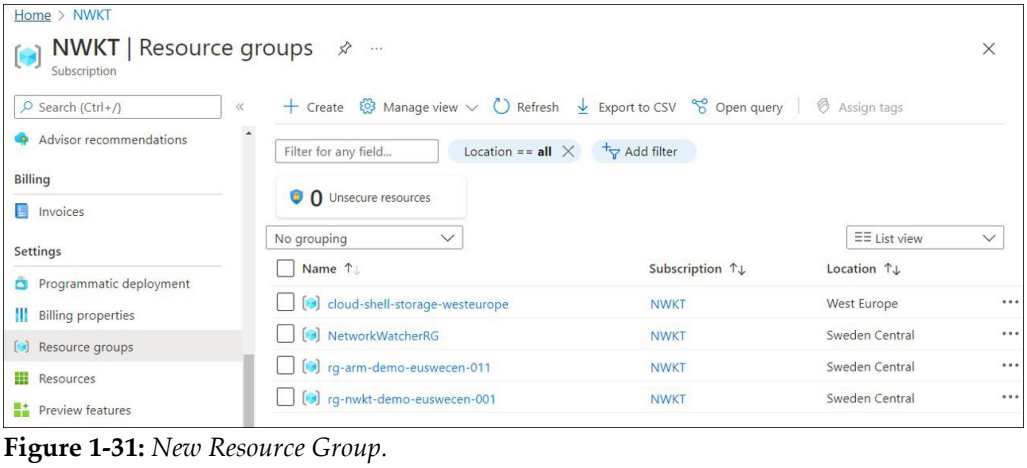


Figure 1-31: *New Resource Group.*

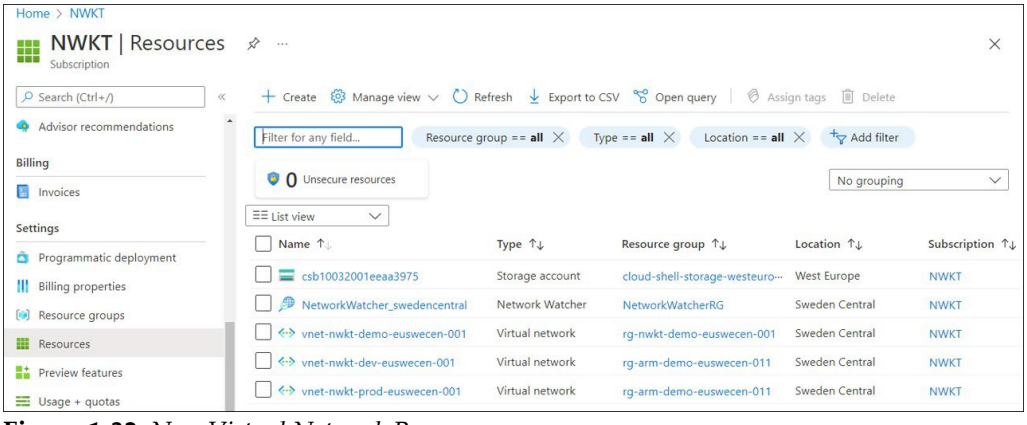


Figure 1-32: *New Virtual Network Resources.*

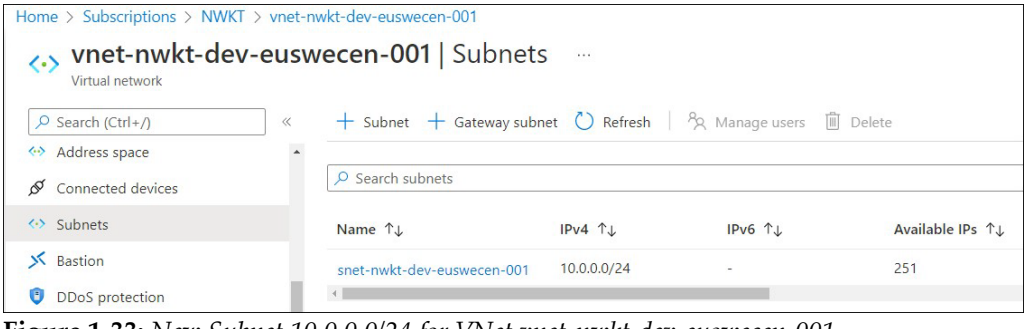


Figure 1-33: *New Subnet 10.0.0.0/24 for VNet vnet-nwkt-dev-euswecen-001.*

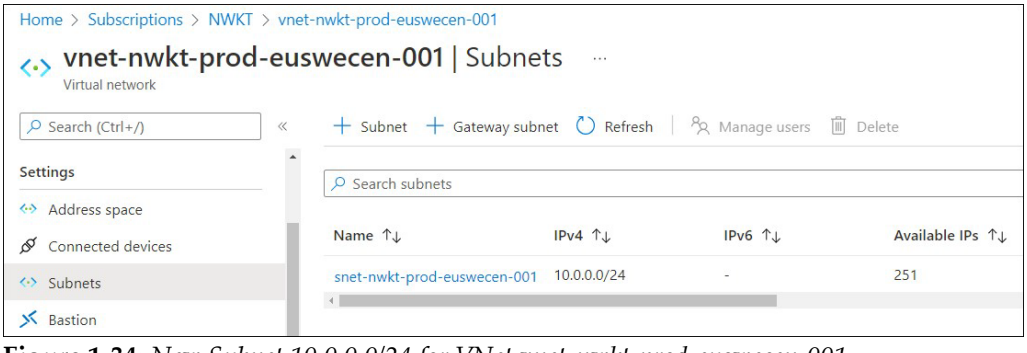


Figure 1-34: *New Subnet 10.0.0.0/24 for VNet vnet-nwkt-prod-euswecen-001.*

Examples 1-7, and 1-8 shows how we can get information about Virtual Networks using Azure AZ PowerShell.

```
PS /home/toni_pasanen> Get-AzVirtualNetwork -Name vnet-nwkt-prod-euswecen-001

Name                   : vnet-nwkt-prod-euswecen-001
ResourceGroupName      : rg-arm-demo-euswecen-011
Location               : swedencentral
Id                     : /subscriptions/***/resourceGroups/rg-arm-demo-euswecen-
011/providers/Microsoft.Network/virtualNetworks/vnet-nwkt-prod-euswecen-001
Etag                   : W/"***"
ResourceGuid           : ***
ProvisioningState      : Succeeded
Tags                   :
                         Name  Value
                         ====  =====
                         app   prod

AddressSpace           : {
                           "AddressPrefixes": [
                             "10.0.0.0/16"
                           ]
                         }
DhcpOptions            : null
FlowTimeoutInMinutes   : null
Subnets                : [
                           {
                             "Delegations": [],
                             "Name": "snet-nwkt-prod-euswecen-001",
                             "Etag": "W/\"***\"",
                             "Id": "/subscriptions/***/resourceGroups/rg-arm-demo-
euswecen-011/providers/Microsoft.Network/virtualNetworks/vnet-nwkt-prod-euswec
                         en-001/subnets/snet-nwkt-prod-euswecen-001",
                             "AddressPrefix": [
                               "10.0.0.0/24"
                             ],
                             "IpConfigurations": [],
                             "ServiceAssociationLinks": [],
                             "ResourceNavigationLinks": [],
                             "ServiceEndpoints": [],
                             "ServiceEndpointPolicies": [],
                             "PrivateEndpoints": [],
                             "ProvisioningState": "Succeeded",
                             "PrivateEndpointNetworkPolicies": "Enabled",
                             "PrivateLinkServiceNetworkPolicies": "Enabled",
                             "IpAllocations": []
                           }
                         ]
VirtualNetworkPeerings : []
EnableDdosProtection   : false
DdosProtectionPlan     : null
ExtendedLocation       : null
```

Example 1-7: *Verification Using Azure AZ PowerShell-1.*

```
PS /home/toni_pasanen> Get-AzVirtualNetwork -Name vnet-nwkt-dev-euswecen-001

Name                   : vnet-nwkt-dev-euswecen-001
ResourceGroupName      : rg-arm-demo-euswecen-011
Location               : swedencentral
Id                     : /subscriptions/***/resourceGroups/rg-arm-demo-euswecen-
011/providers/Microsoft.Network/virtualNetworks/vnet-nwkt-dev-euswecen-001
Etag                   : W/"***"
ResourceGuid           : ***
ProvisioningState      : Succeeded
Tags                   :
                         Name  Value
                         ====  =====
                         app   dev

AddressSpace           : {
                           "AddressPrefixes": [
                             "10.0.0.0/16"
                           ]
                         }
DhcpOptions            : null
FlowTimeoutInMinutes   : null
Subnets                : [
                           {
                             "Delegations": [],
                             "Name": "snet-nwkt-dev-euswecen-001",
                             "Etag": "W/\"***\"",
                             "Id": "/subscriptions/***/resourceGroups/rg-arm-demo-
euswecen-011/providers/Microsoft.Network/virtualNetworks/vnet-nwkt-dev-euswece
                       n-001/subnets/snet-nwkt-dev-euswecen-001",
                             "AddressPrefix": [
                               "10.0.0.0/24"
                             ],
                             "IpConfigurations": [],
                             "ServiceAssociationLinks": [],
                             "ResourceNavigationLinks": [],
                             "ServiceEndpoints": [],
                             "ServiceEndpointPolicies": [],
                             "PrivateEndpoints": [],
                             "ProvisioningState": "Succeeded",
                             "PrivateEndpointNetworkPolicies": "Enabled",
                             "PrivateLinkServiceNetworkPolicies": "Enabled",
                             "IpAllocations": []
                           }
                         ]
VirtualNetworkPeerings : []
EnableDdosProtection   : false
DdosProtectionPlan     : null
ExtendedLocation       : null
```

Example 1-8: *Verification Using Azure AZ PowerShell.*

Preferences

[1] Define your naming convention:
https://docs.microsoft.com/en-us/azure/cloud-adoption-framework/ready/azure-best-practices/resource-naming

[2] Recommended abbreviations for Azure resource types:
https://docs.microsoft.com/en-us/azure/cloud-adoption-framework/ready/azure-best-practices/resource-abbreviations

[3] Resource naming and tagging decision guide
https://docs.microsoft.com/en-us/azure/cloud-adoption-framework/decision-guides/resource-tagging/?toc=/azure/azure-resource-manager/management/toc.json

[4] Define your tagging strategy
https://docs.microsoft.com/en-us/azure/cloud-adoption-framework/ready/azure-best-practices/resource-tagging

[5] Develop your naming and tagging strategy for Azure resources
https://docs.microsoft.com/en-us/azure/cloud-adoption-framework/ready/azure-best-practices/naming-and-tagging

[6] Introducing JSON
https://www.json.org/json-en.html

[7] The JavaScript Object Notation (JSON) Data Interchange Format
T. Bray Dec 2017
https://www.rfc-editor.org/rfc/rfc8259.txt

[8] ECMA-404: The JSON data interchange syntax. 2nd edition Dec 2017
https://www.ecma-international.org/publications-and-standards/standards/ecma-404/

[9] Azure PowerShell Documentation
https://docs.microsoft.com/en-us/powershell/azure/?view=azps-7.5.0

[10] Understand the structure and syntax of ARM templates, 03/18/2022
https://docs.microsoft.com/en-us/azure/azure-resource-manager/templates/syntax

[11] Create Resource Manager parameter file
https://docs.microsoft.com/en-us/azure/azure-resource-manager/templates/parameter-files

[12] Data residency in Azure
https://azure.microsoft.com/en-us/global-infrastructure/data-residency/#overview

[13] Regions and availability zones
https://docs.microsoft.com/en-us/azure/availability-zones/az-overview