Branch Application To Cloud Application Connectivity

Overview

This quickstart guide will provide all the steps to create a secure service between a branch application and/or user and an application hosted in Azure Cloud using NetFoundry Overlay Fabric (NFOF).

> **Important**
>
> Assumption is that the NF Fabric is already up.

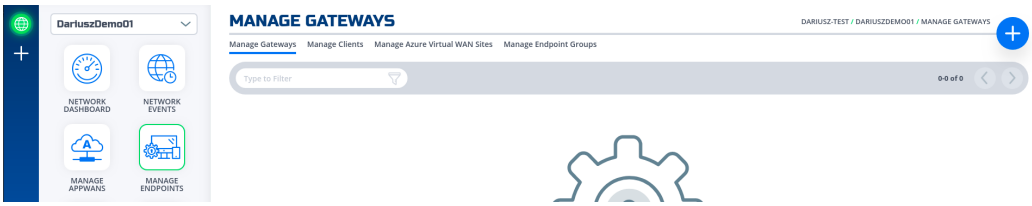
Through NF Web Console UI

Create and Deploy NF Gateway in Branch Datacenter

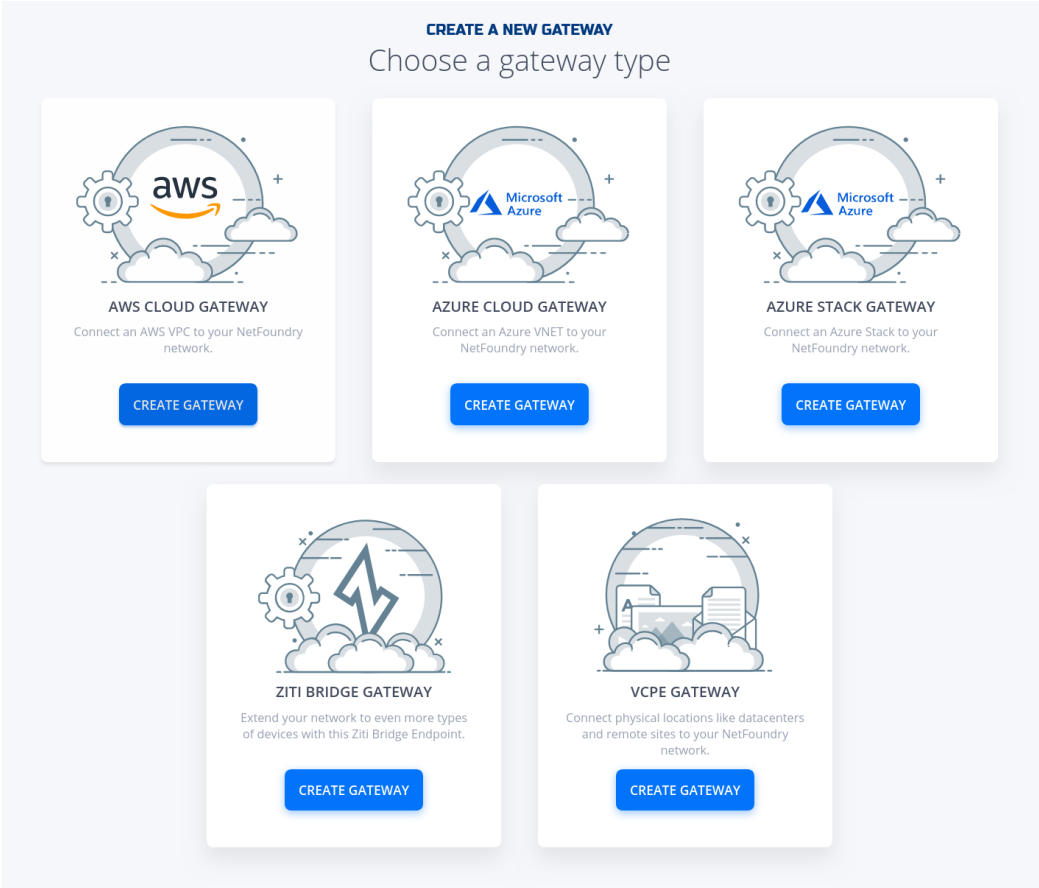
This section will guide a user through the steps on how to create a NF Manage Gateway in the NF Console UI and install it in the Branch Datacenter.

Console UI

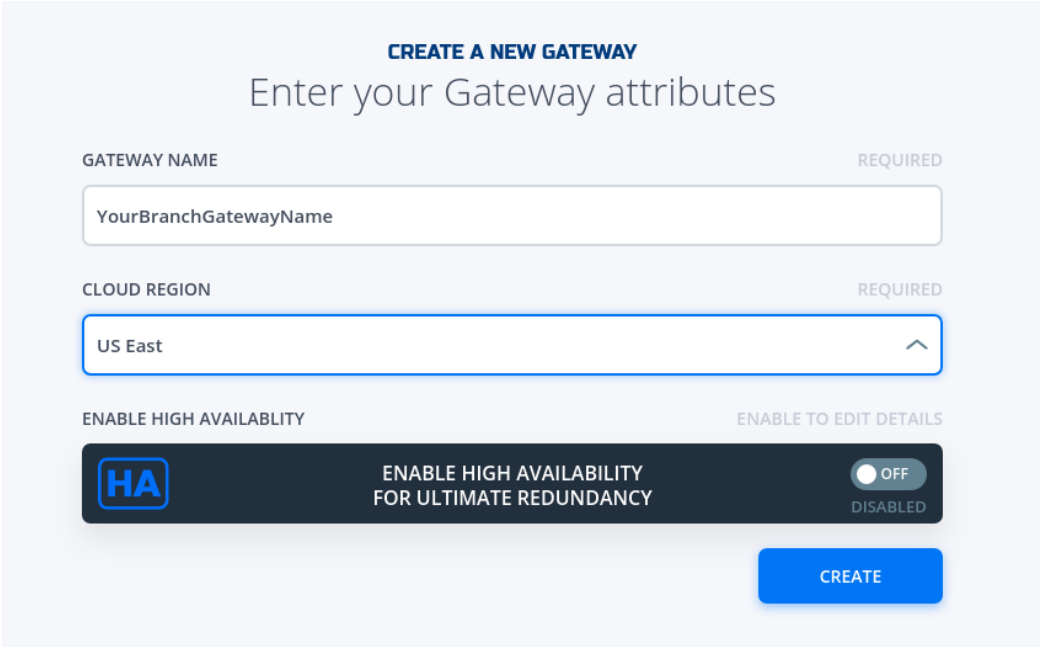
1. Navigate to Manage Gateways Page
2. Click on + sign in the top right corner.



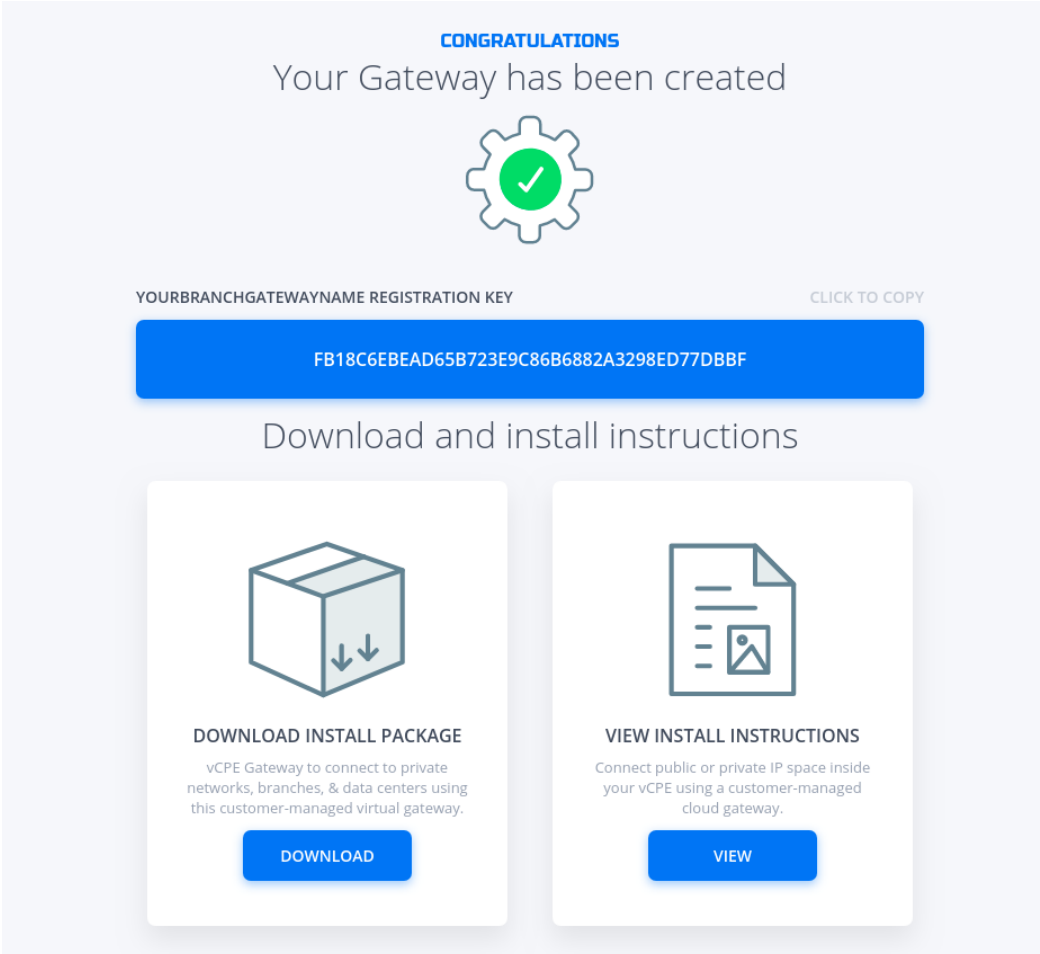
3. Click on "Create Gateway" on the VCPE Gateway Card



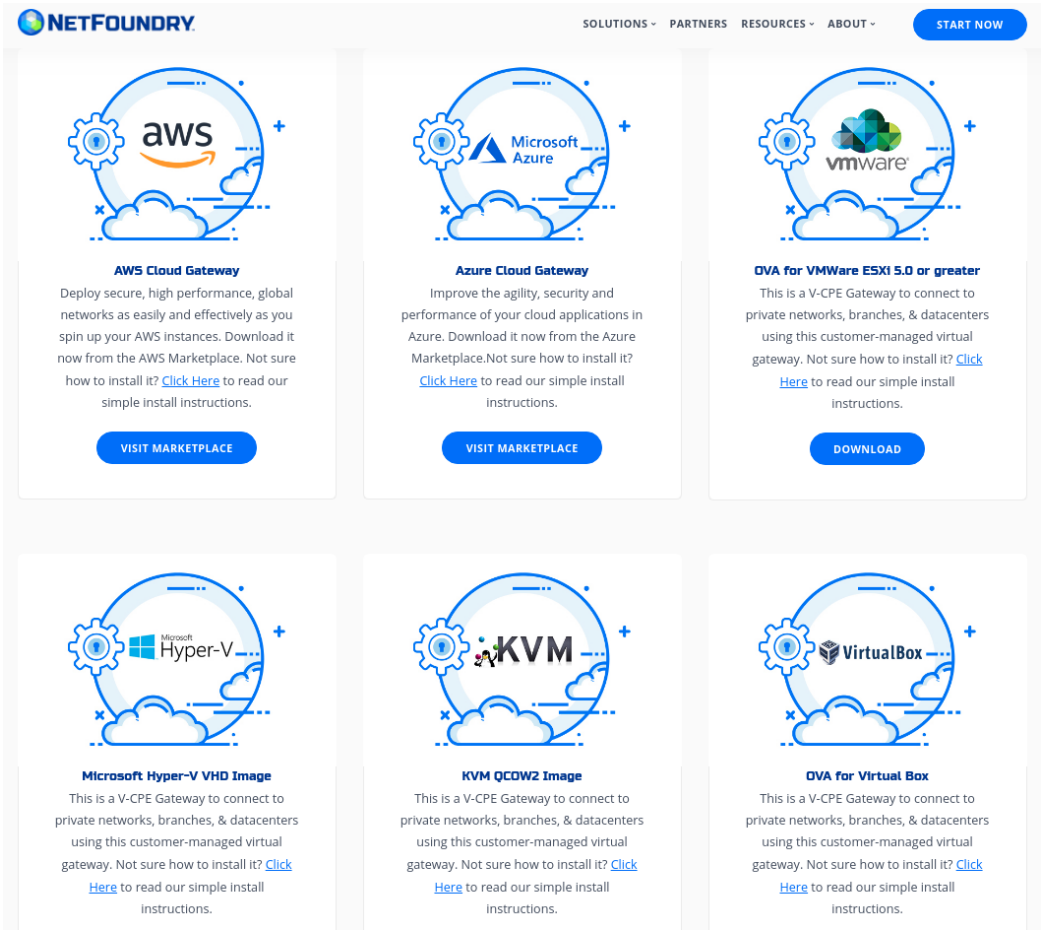
4. Fill in the required information and click on "Create"



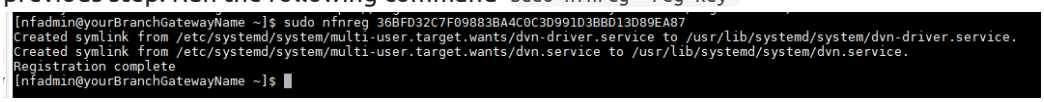
5. Copy the Client Registration Key



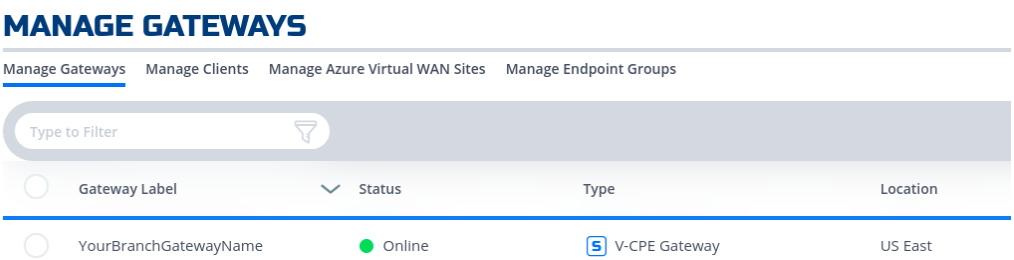
6. Click on "Download" button on the Installation Package Card



7. Download the correct image for the desired Hypervisor.
8. Follow the installation procedure linked in the description of each image type (i.e. "Click Here").
9. Once installed, login into it locally with ssh and register it using the key copied in the previous step. Run the following command `sudo nfnreg "reg key"`



10. Once registered, one should see the gateway status turn to green in NF Console UI



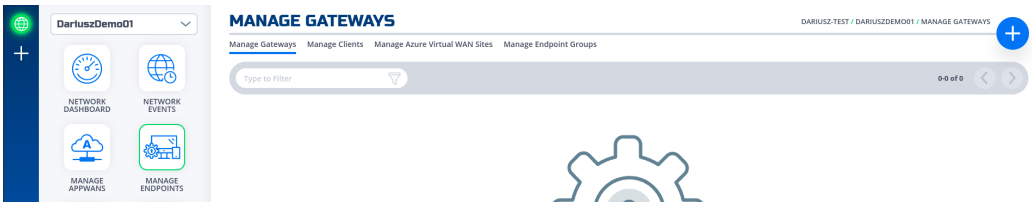
11. Done

Create and Deploy NF Azure Gateway

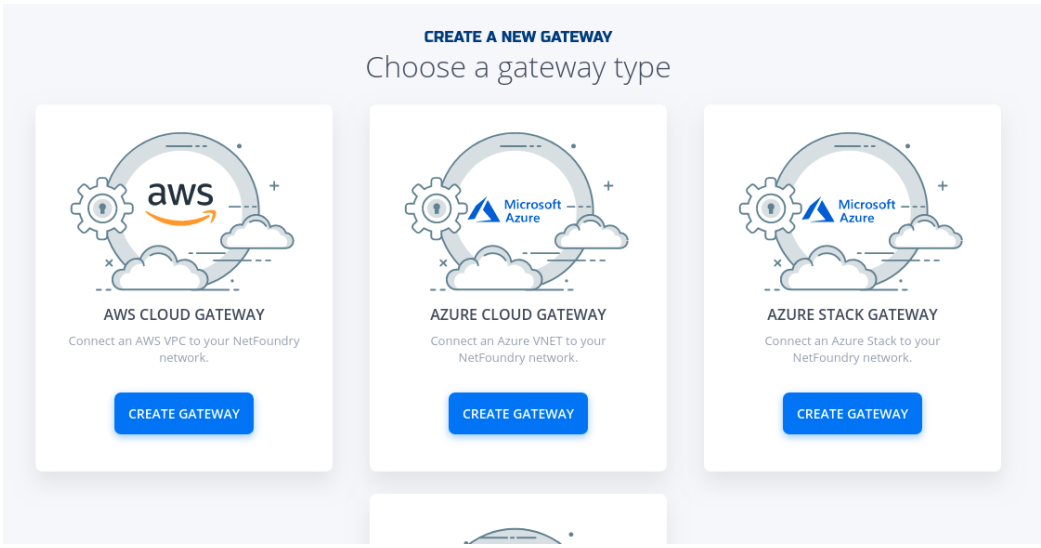
This section will guide a user through the steps on how to create a NF Manage Gateway in the NF Console UI and install it in the Azure vNet.

Console UI

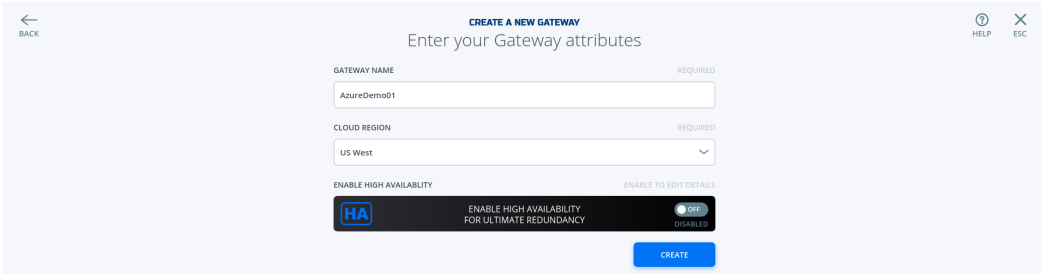
1. Navigate to Manage Gateways Page
2. Click on + sign in the top right corner.



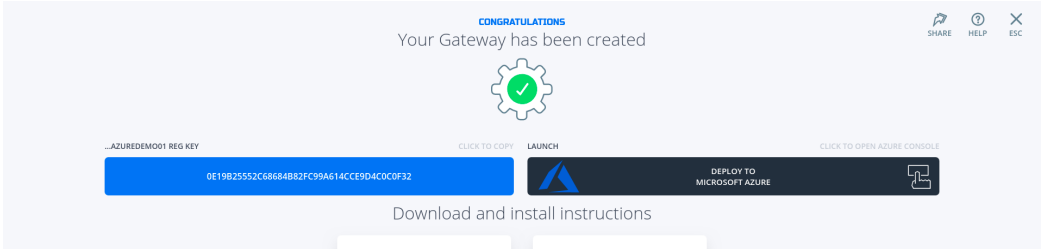
3. Click on "Create Gateway" on the Azure Cloud Gateway Card



4. Fill in the required information and click on "Create"

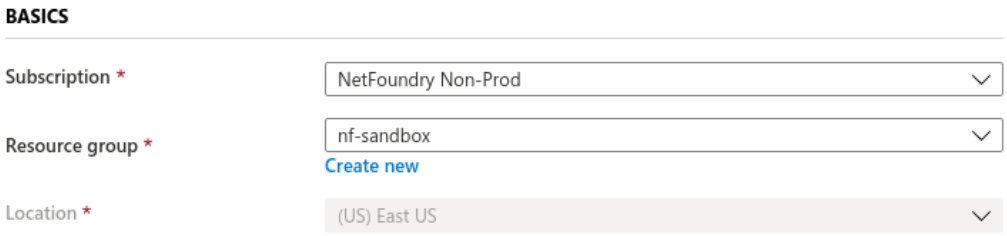


5. Copy the Client Registration Key

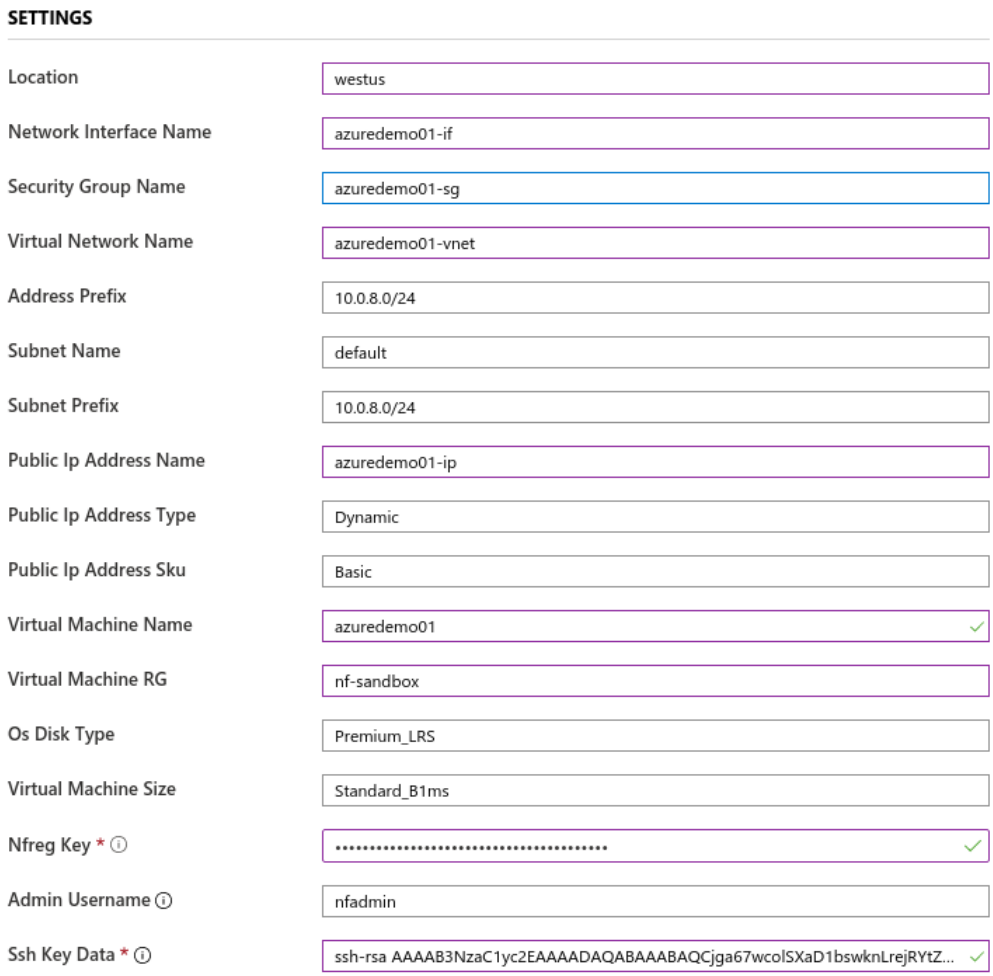


6. Click on "Deploy to Microsoft Azure". It will take you to the Azure Portal and ask you for your login credentials.

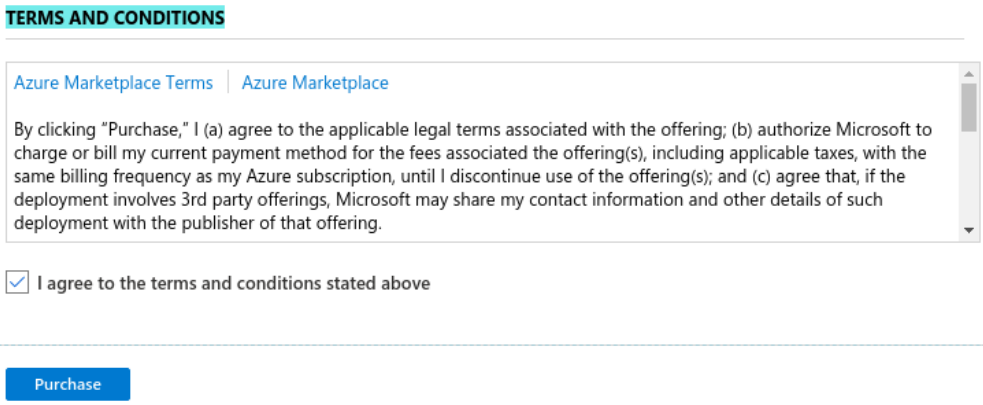
7. You will be presented with the template that needs to be filled. The first section is the Basics regarding your Subscription and Resource Group this gateway will be deployed in.



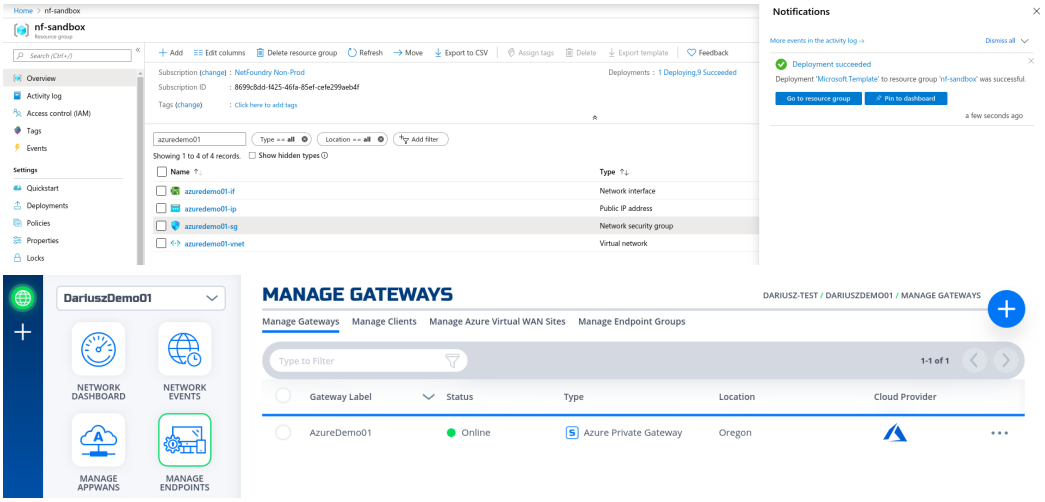
8. The second section related to resources associated with this gateway. e.g. vm name, ip address space, security groups, etc. you will paste the registration key copied in step 5. You will also need the public ssh key to use for access to this gateway remotely.



9. You will need to agree to Azure Marketplace Terms and Conditions and click to "Purchase" to continue.



10. If the NF Gateway was deployed successfully. Here is the view of the Resource Group and NF Conole UI.



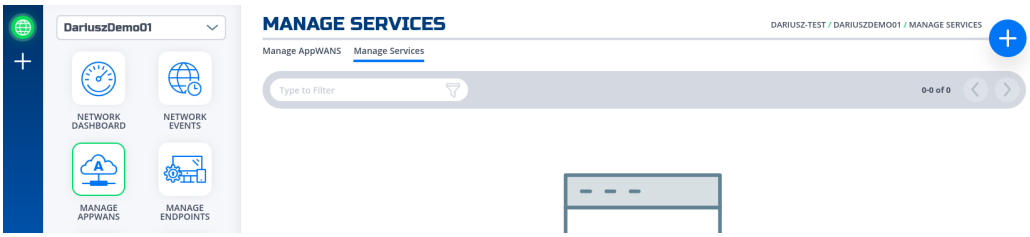
11. Done

Create IP Host Service

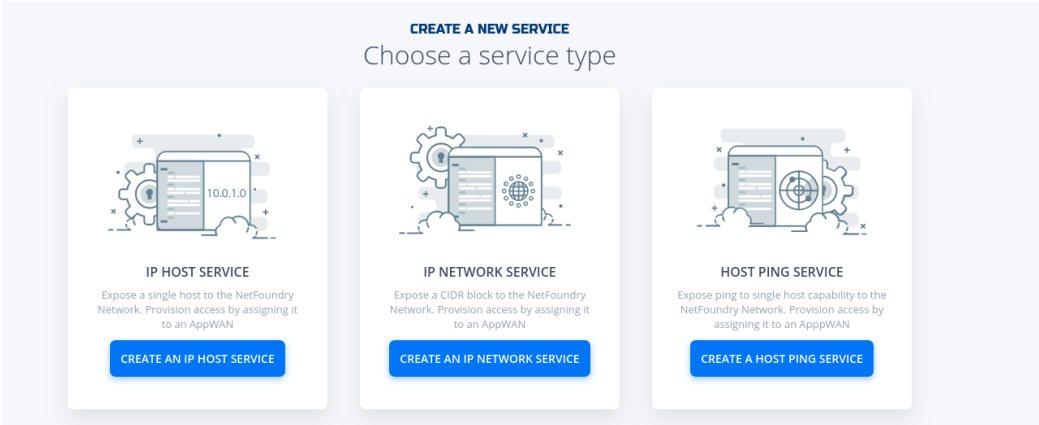
This section will guide a user through the steps on how to create a NF Service.

Console UI

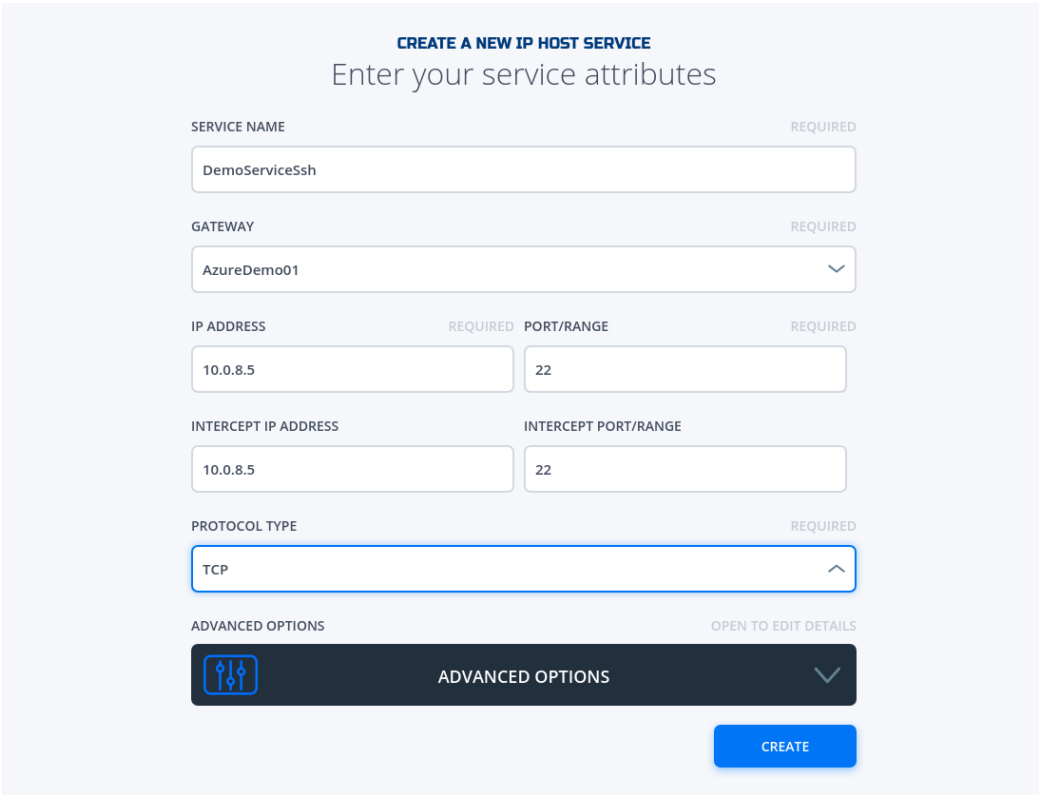
1. Navigate to Manage Services Page under Manage Appwans
2. Click on + sign in the top right corner.



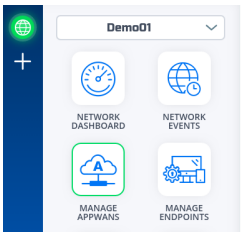
3. Click on "Create an IP Host Service"

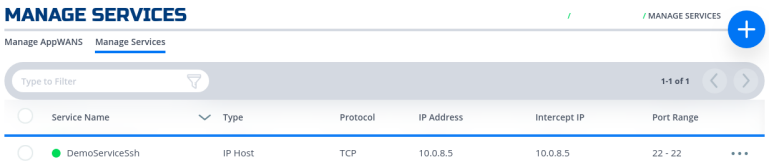


4. Fill in the required information for SSH and click on "Create"



5. If successfully, the service is green.





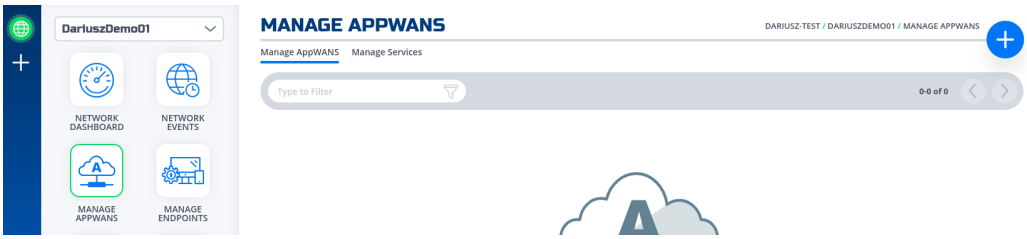
6. Done

Create AppWan

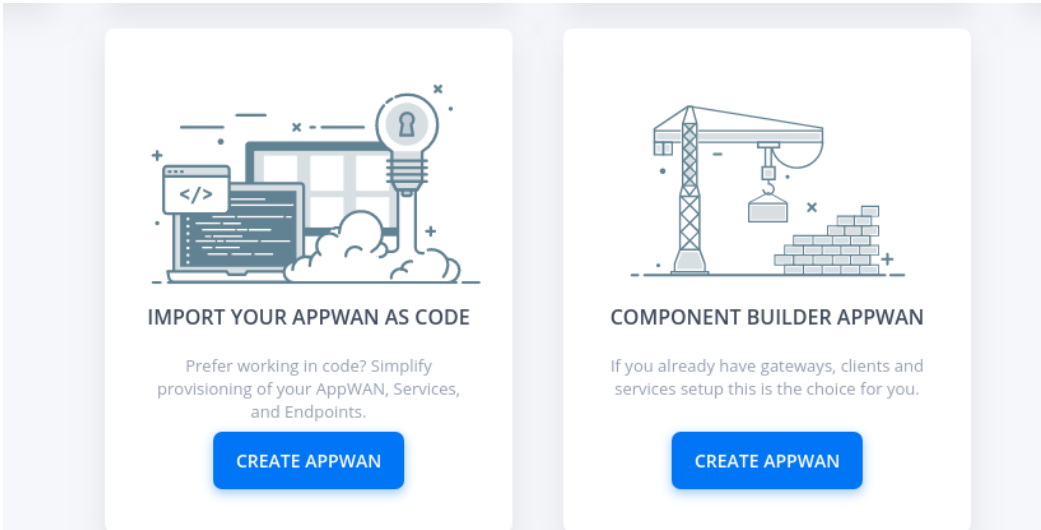
This section will guide a user through the steps on how to enable service connectivity to users by creating an appwan.

Console UI

1. Navigate to Manage AppWANS Page under Manage Appwans
2. Click on + sign in the top right corner.

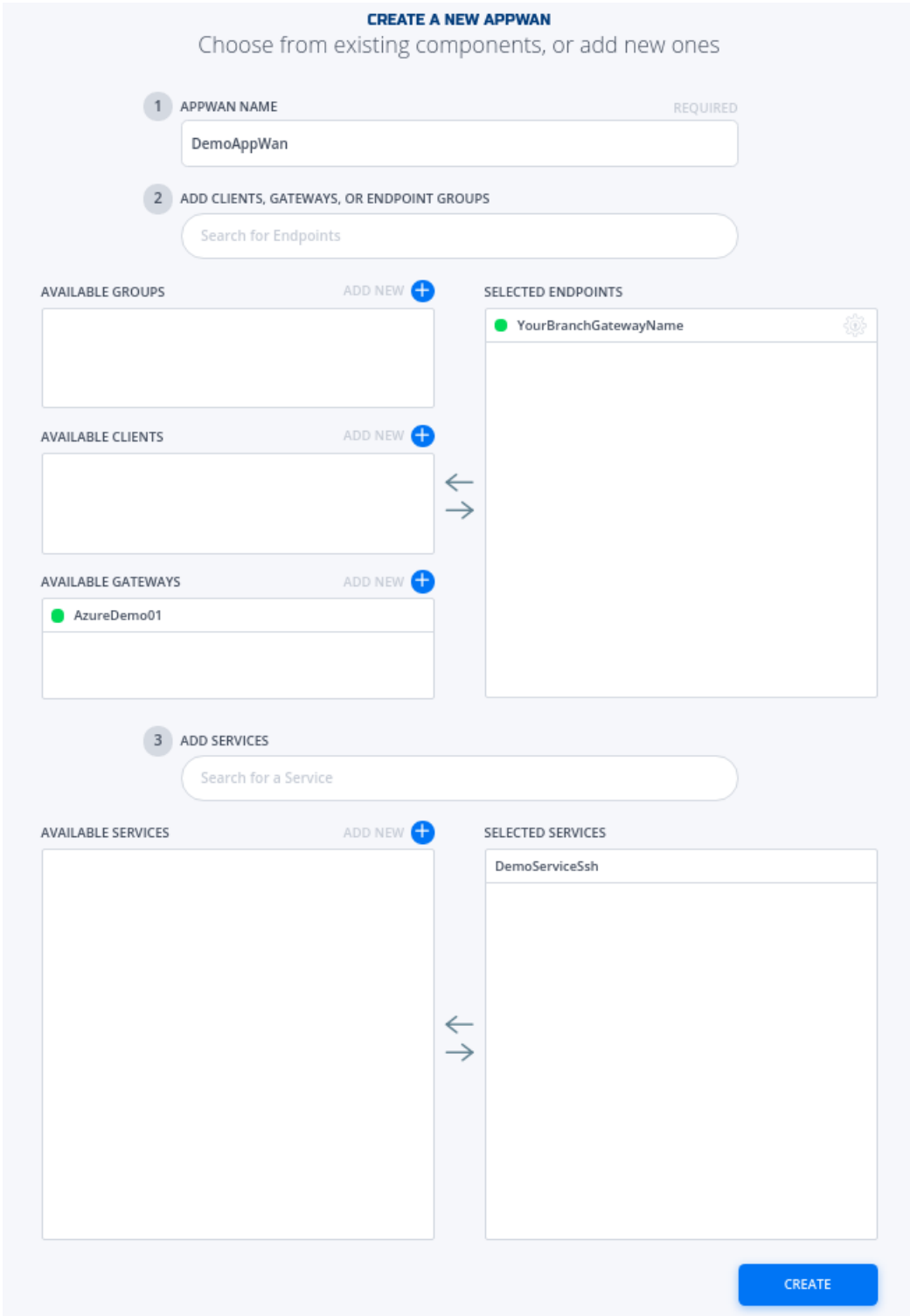


3. Click on "Component Builder Appwan"

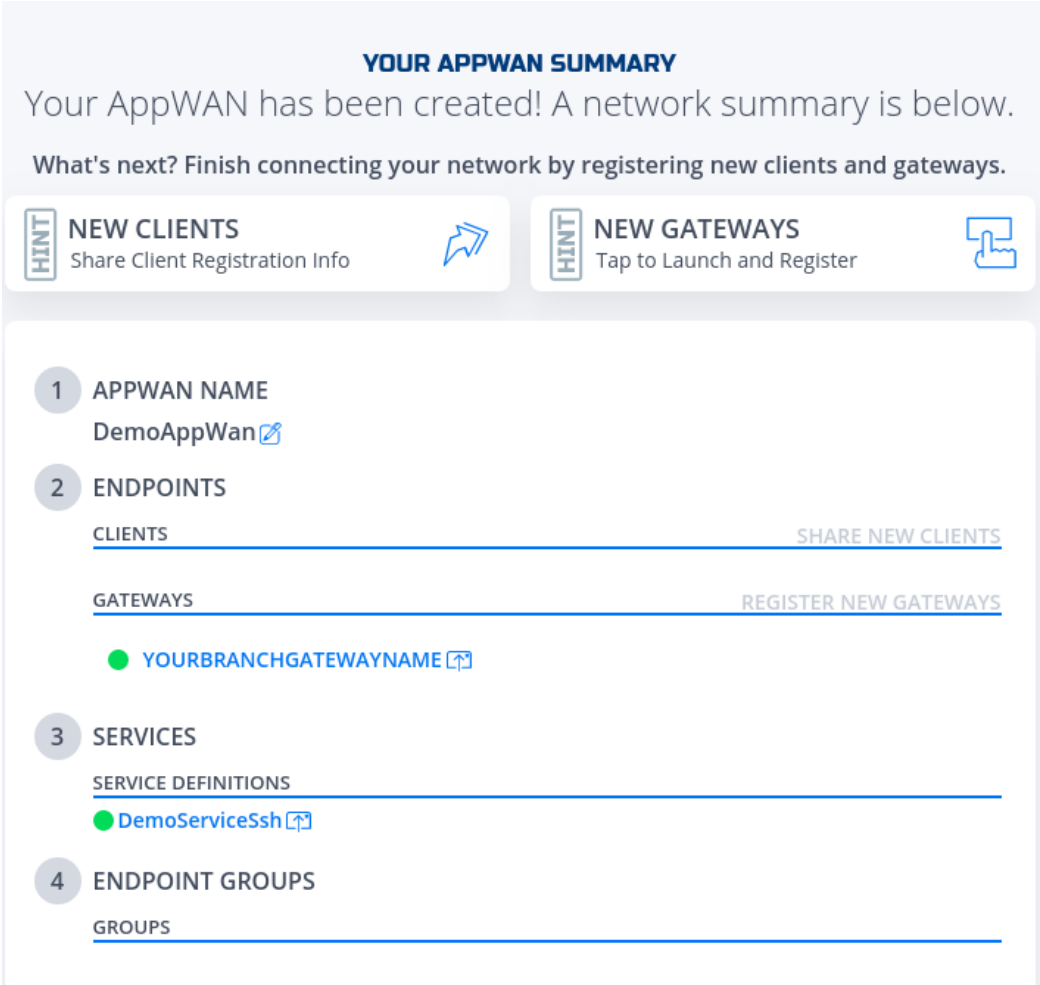


4. Move the desired gateway (e.g. DemoGateway01) from "Available" Gateways to "Selected" Endpoints. Move the desired service (e.g. DemoServiceSsh) from "Available" to

"Selected" Services.



5. Click on "Create".



6. Done

Test Connectivity to Application Server

Route to vNet

The private IP of NF Gateway (e.g. YourBranchGatewayName) needs to be the next hop to reach the vNet in Azure (e.g. 10.0.8.0/24). Thus, a static route will need to be configured in one of the branch routers. NF Gateway can support dynamic routing if needed (e.g.bgp, ospf)

Steps

1. Log in to a Client App Host in Branch DataCenter
2. Run ssh username@privateIpOfServerAppHostInAzure

Programmatically

via Python and Terraform

Python Modules

For the code clarity, we have broken down the code into multiple Python modules

1. NF REST CRUD (Create,Read, Update and Delete) operations
2. Get MOP Session Token
3. Create NF Network
4. Create NF Gateway(s)
5. Create NF Service(s)
6. Create NF AppWan(s)
7. Wrapper Script to Create NF Resources based on Resource yaml file

Environment Setup Requirements

1. ~/.env to store NF Credentials in (e.g. `clientId, clientSecret`) to obtain a session token for NF API
2. Export Azure Credentials (e.g, `export ARM_TENANT_ID, ARM_CLIENT_ID, ARM_CLIENT_SECRET, ARM_SUBSCRIPTION_ID`) to enable resource gateway creation in Azure Resource Group via Terraform.
3. Terraform and Python3 installed in path.

Additional Information:

1. The new Resource Group in Azure is created based on then name provided in Resource yaml, if one does not exists already in the same region (e.g. centralus). The action delete gateway will delete the RG as well even if it was an existing RG. If one does not want to delete the RG, the command `terraform state rm "{tf resource name for RG}"` needs to be run before running the gateway delete step. This will ensure that the RG is not deleted.
2. A new vNet will be created and NF Gateway will be placed in it.
3. Environment means the NF Console Environment used (e.g. production), not Azure.

Steps

1. Clone this repo (git clone https://github.com/netfoundry/mop.git)
2. Update Resource yaml file with the desired options to feed into the wrapper script as described in the following code snippet. All Resource.yml Options
3. Run this from the root folder to create GW in NF Console UI and Azure.

```
python3 quickstarts/docs/python/nf_resources.py --file quickstarts/docs/python/
nf_resources.yml
```

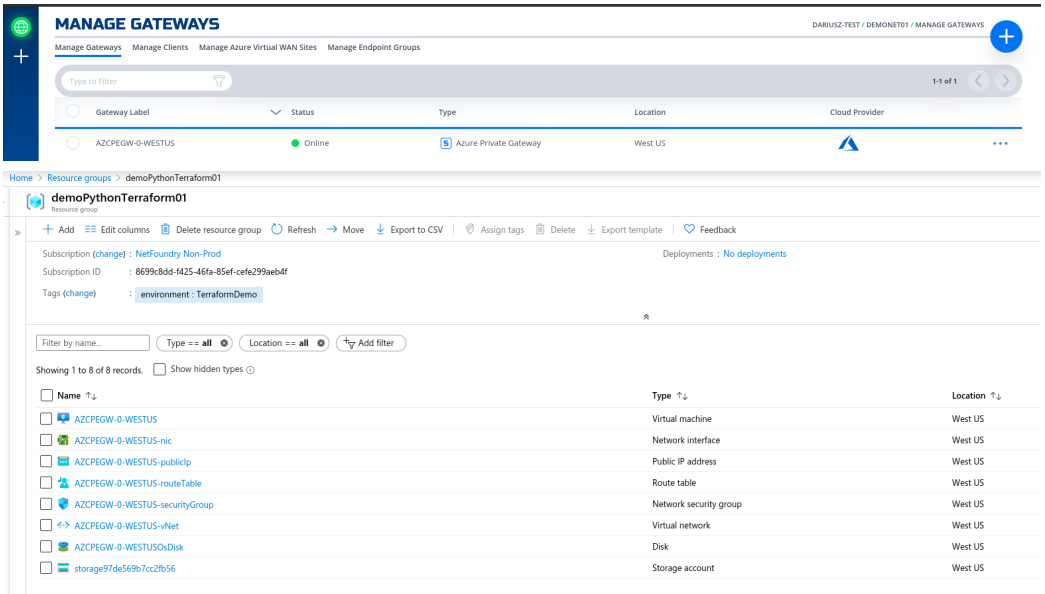
Required Configuration Parameters for Gateway Creation

```
environment: production
network_action: get
network_name: DemoNet01
gateway_list:
- action: create
  cloud: azure
  count: 1
  names: []
  region: westus
  regionalCidr: [10.20.10.0/24]
  regkeys: []
  resourceGroup:
    name: demoPythonTerraform01
    region: centralus
  tag: TerraformDemo
terraform:
  bin: terraform
  output: 'no'
  source: ./quickstarts/docs/terraform
  work_dir: .
```

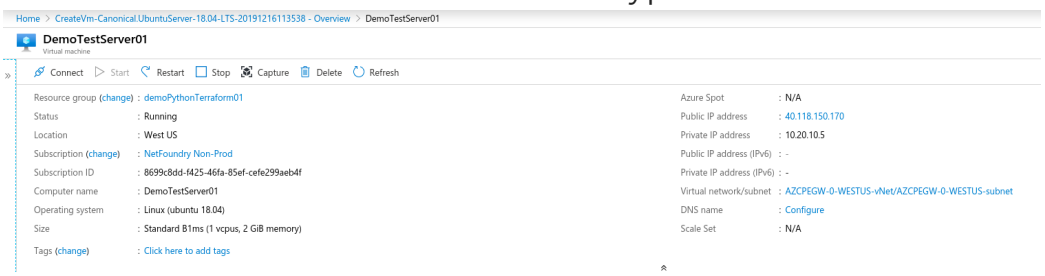
4. After the script is run successfully, one can see that the gateway name and registration key were saved in Resource.yml file. The name is created automatically based on region and gateway type joined with x and gateway count (AZCPEGW means an azure type gateway in NF console). One can create more than one gateway in the same region by increasing the count to more than 1.

```
environment: production
gateway_list:
- action: create
  cloud: azure
  count: 1
  names:
  - AZCPEGWx0xWESTUS
  region: westus
  regionalCidr:
  - 10.20.10.0/24
  regkeys:
  - 21DB86724EC3F31C11C1C9D68CE5ECD6A06F057E
  resourceGroup:
```

```
    name: demoPythonTerraform01
    region: centralus
  tag: TerraformDemo
network_action: get
network_name: DemoNet01
terraform:
  bin: terraform
  output: 'no'
  source: ./quickstarts/docs/terraform
  work_dir: .
```



5. Create a test server vm on the same vNet if not already present.



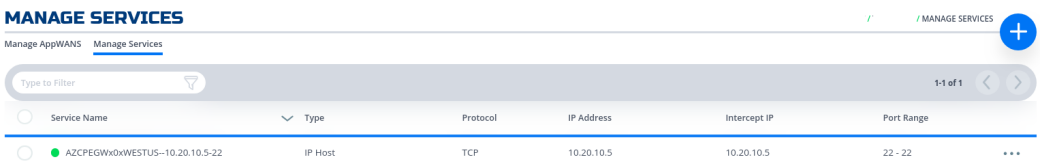
6. Update the Resoure.yaml file to include the Service option to create the NF service on the gateway create in the previous step. Don't forget to change the action on the gateway to "get".

```
environment: production
gateway_list:
- action: get
  cloud: azure
  count: 1
  names:
  - AZCPEGWx0xWESTUS
  region: westus
  regionalCidr:
  - 10.20.10.0/24
  regkeys:
  - 21DB86724EC3F31C11C1C9D68CE5ECD6A06F057E
  resourceGroup:
```

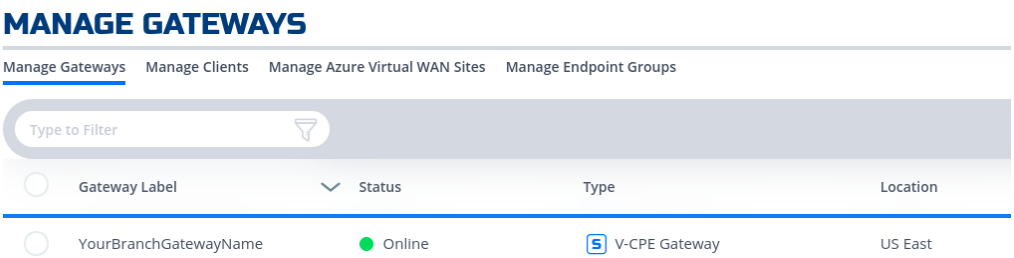
```
    name: demoPythonTerraform01
    region: centralus
  tag: TerraformDemo
network_action: get
network_name: DemoNet01
terraform:
  bin: terraform
  output: 'no'
  source: ./quickstarts/docs/terraform
  work_dir: .
services:
- action: create
  gateway: AZCPEGWx0xWESTUS
  ip: 10.20.10.5
  port: 22
  name:
  type: host
```

7. After the script run again successfully, the service section should have been populated with the service name as so.

```
services:
- action: create
  gateway: AZCPEGWx0xWESTUS
  ip: 10.20.10.5
  name: AZCPEGWx0xWESTUS--10.20.10.5--22
  port: 22
  type: host
```



8. Create a gateway in the branch as the steps in the UI section indicated. We will provide code snippets for private hypervisors deployment through python in later releases (e.g. vSphere)

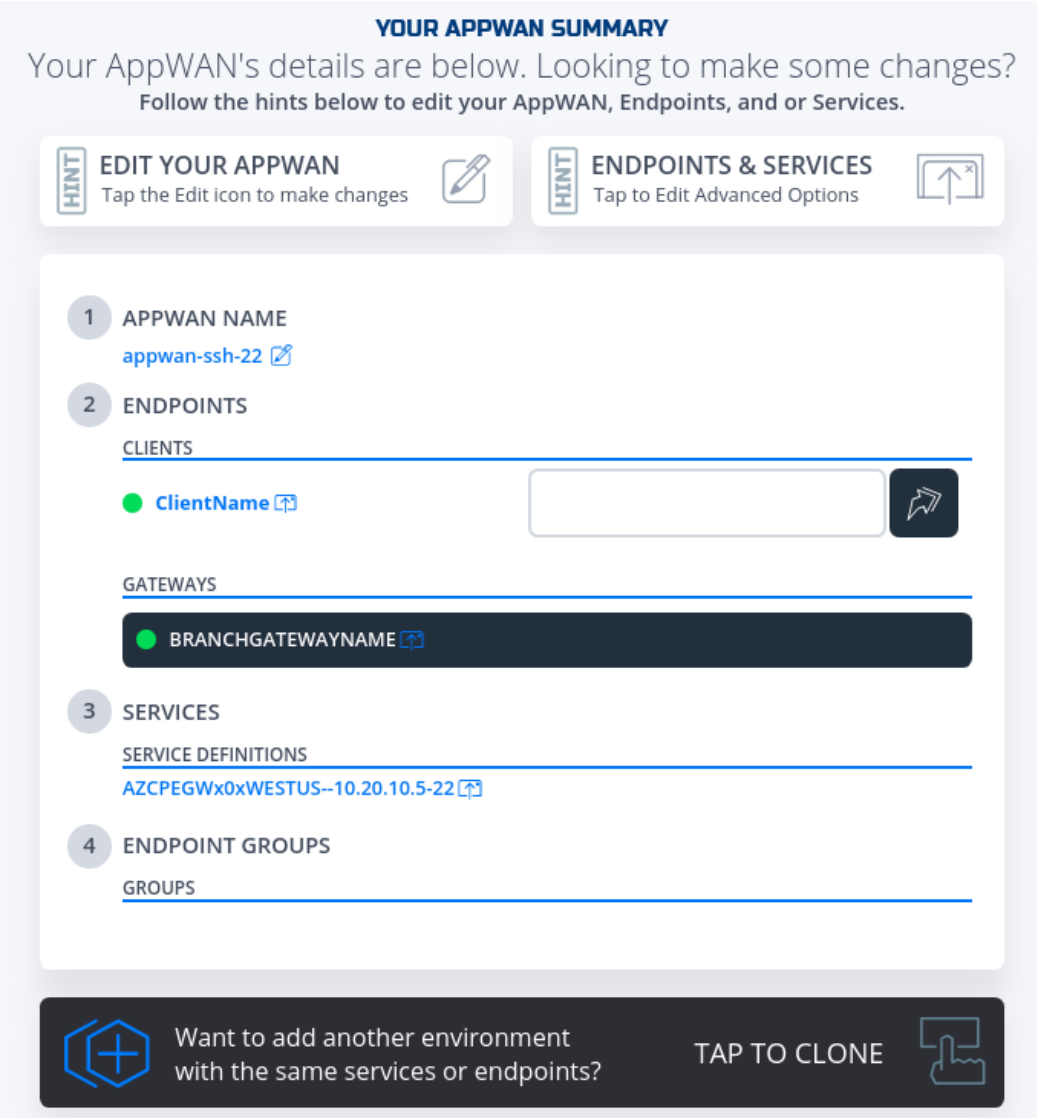


9. Update the Resoure.yaml file to include the AppWan option to create the NF AppWan tying the gateway, client and service created in the previous steps. Don't forget to change the action on the service option to "get".

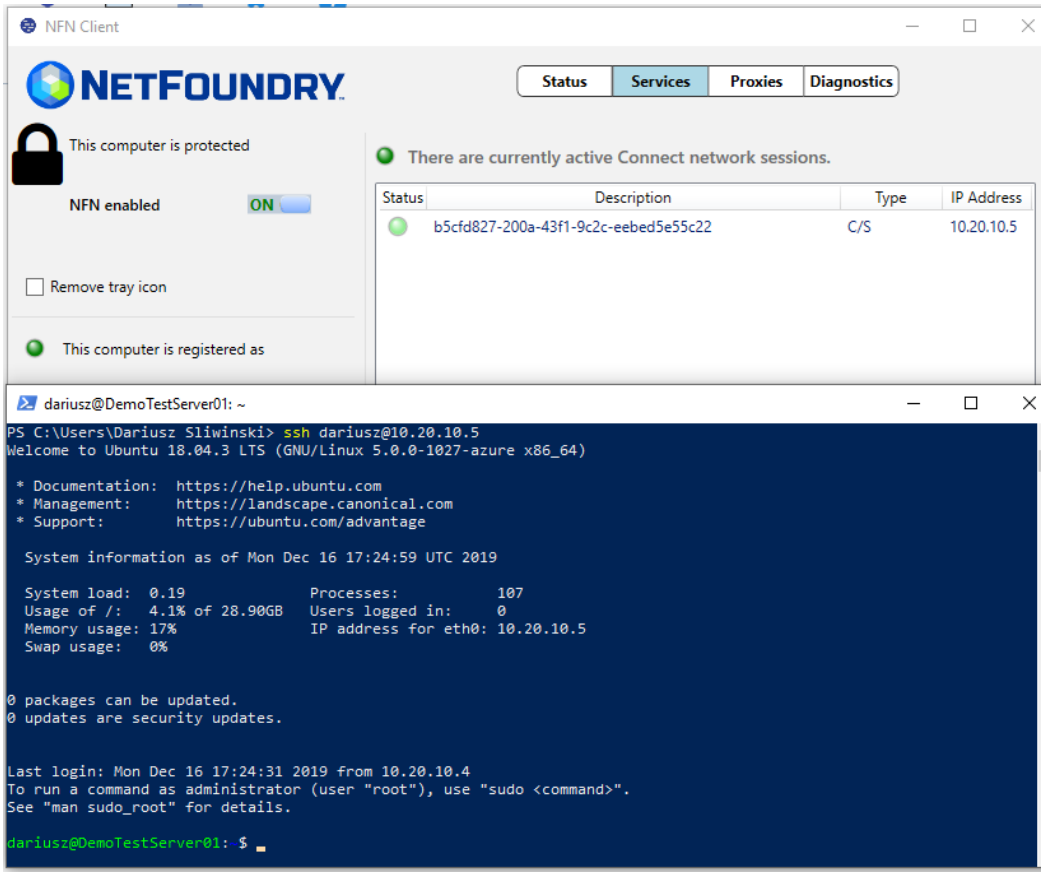
```
environment: production
gateway_list:
- action: get
  cloud: azure
```

```
  count: 1
  names:
  - AZCPEGWx0xWESTUS
  region: westus
  regionalCidr:
  - 10.20.10.0/24
  regkeys:
  - 21DB86724EC3F31C11C1C9D68CE5ECD6A06F057E
  resourceGroup:
    name: demoPythonTerraform01
    region: centralus
  tag: TerraformDemo
network_action: get
network_name: DemoNet01
services:
- action: get
  gateway: AZCPEGWx0xWESTUS
  ip: 10.20.10.5
  name: AZCPEGWx0xWESTUS--10.20.10.5--22
  port: 22
  type: host
terraform:
  bin: terraform
  output: 'no'
  source: ./quickstarts/docs/terraform
  work_dir: .
appwans:
- action: create
  endpoints:
  - BranchGatewayName
  - ClientName
  name: appwan-ssh-22
  services:
  - AZCPEGWx0xWESTUS--10.20.10.5--22
```

10. After the script ran again successfully, the connectivity should have been up.



11. To test connectivity, log in to the Remote Client or Branch App Server and run ssh "username"@"privateIp"



12. To delete resources created, just follow the reverse order. Change the action to delete for AppWans first, then other resources as indicated in the code snippets.

```
appwans:
- action: delete
  endpoints:
  - BranchGatewayName
  - ClientName
  name: null
  services:
  - AZCPEGWx0xWESTUS--10.20.10.5--22
```

13. Services

```
services:
- action: delete
  gateway: AZCPEGWx0xWESTUS
  ip: 10.20.10.5
  name: null
  port: 22
  type: host
```

14. Endpoints - will delete all resources in Azure as well.

    a. `terraform state rm "{tf resource name for RG}" // run this before the python script if Resource Group needs to be preserved` 1.

```
gateway_list:
- action: delete
  cloud: azure
  count: 1
  names: []
  region: westus
  regionalCidr:
  - 10.20.10.0/24
  regkeys: []
  resourceGroup:
    name: demoPythonTerraform01
    region: centralus
  tag: TerraformDemo
```

15. Network

```
environment:  production
network_action: delete
network_name: DemoNet01
```

16. Done

via Jenkins

In this section, we will use Resource yaml along with Jenkinsfile to show how to automate the steps further by creating the Jenkins Job

Coming Soon