OV App Streaming (OVAS) API in Azure - Step by Step Guide

Omniverse

Exported on 09/19/2024

Table of Contents

1 Important Note

This is not official documentation. These are notes/instructions on how we got OVAS up and running in our Azure environment. Use this guide as reference.

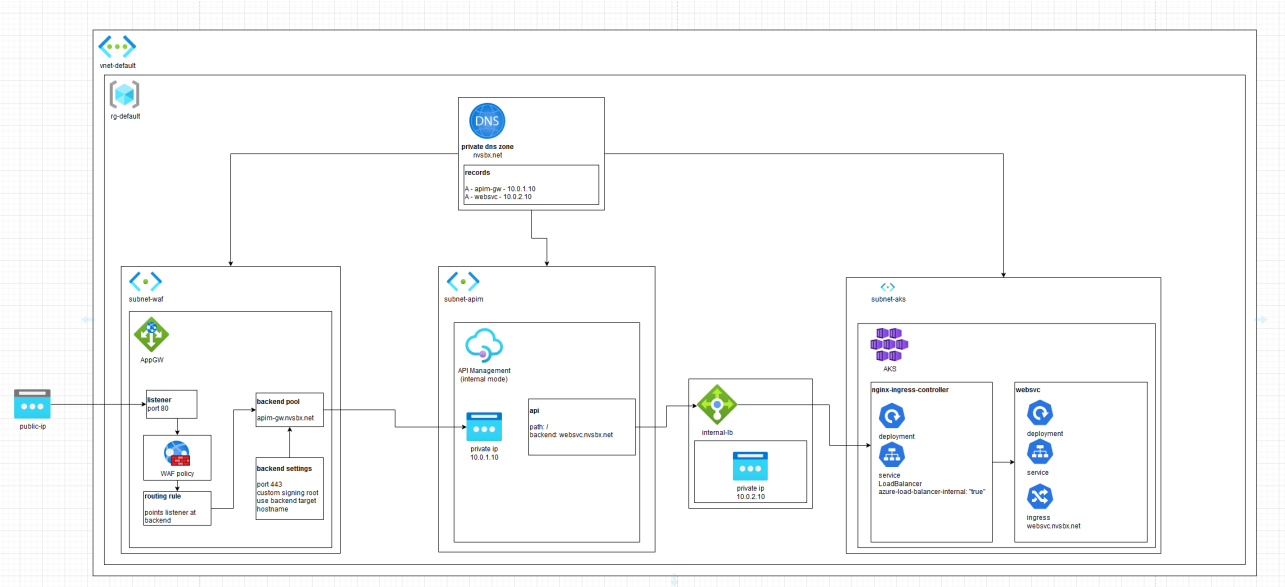
2 Prerequisites

- Unix/Linux Operating system with Bash Shell
- Ensure you are connected to the NVIDIA VPN (needed to pull Docker Image)
- Install Kubectl[1] on your deployment system
- Install Helm[2] on your deployment system
- Install the Azure CLI[3] on your deployment system

1 https://kubernetes.io/docs/tasks/tools/
2 https://helm.sh/docs/intro/quickstart/
3 https://learn.microsoft.com/en-us/cli/azure/install-azure-cli

3 Infrastructure deployment (Manual)

Diagram not complete, example only



3.1 Access subscription

https://portal.azure.com/

> I added an "elitang" prefix/postfix to the names of objects, this is just for personal reference. You can name your objects however you like.

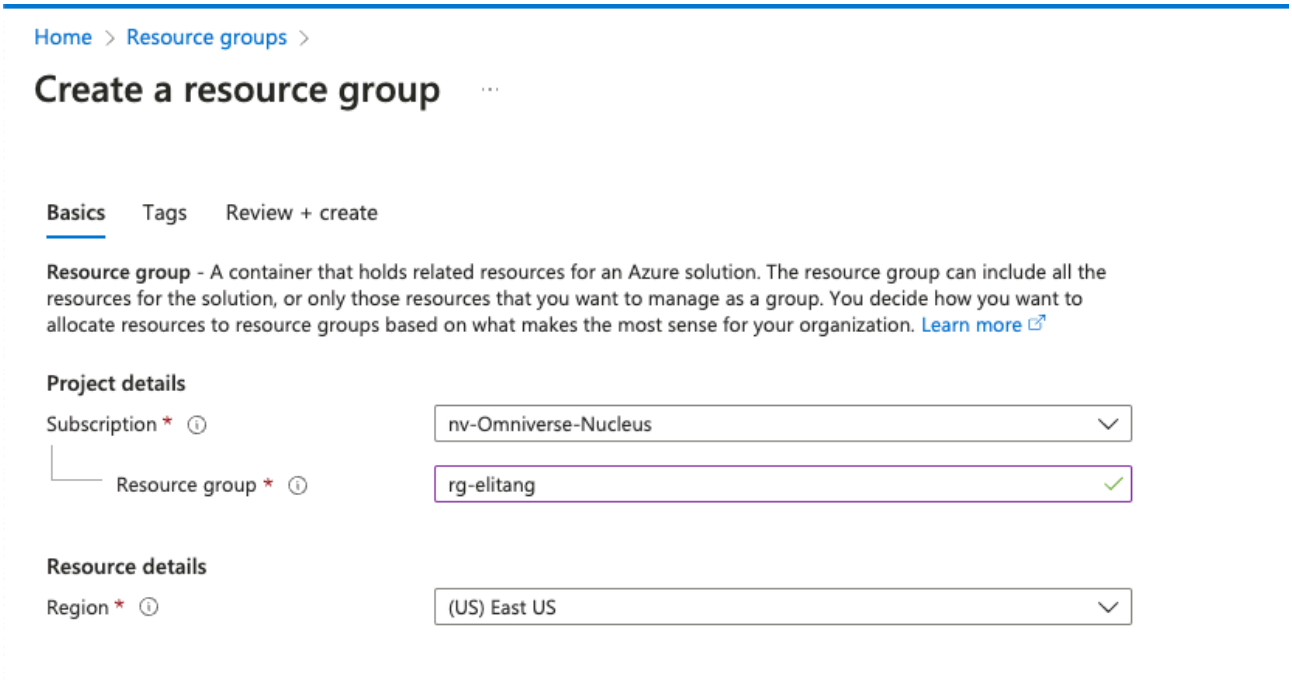
In this section we make the following assumptions:

- All resources are created within a single resource group
- All resources are created in the `eastus` region
- All resource names are for example purposes only and are chosen for consistency purposes in documentation; you may choose your own resource names
- The vnet range and subnet ranges do not overlap (or are taken into consideration) with any planned peered vnets
- **For example purposes only**, the private DNS zone of our deployment is `elitang-ovas-streaming.net`

- **For example purposes only**, The public DNS zone of our deployment is nvsbx.omniverse.nvidia.com[4]
  - The public zone was already created prior to this deployment. In subsequent steps we will be adding records to create the subdomain elitang-ovas-streaming.[5]nvsbx.omniverse.nvidia.com[6]

3.2 Create resource group

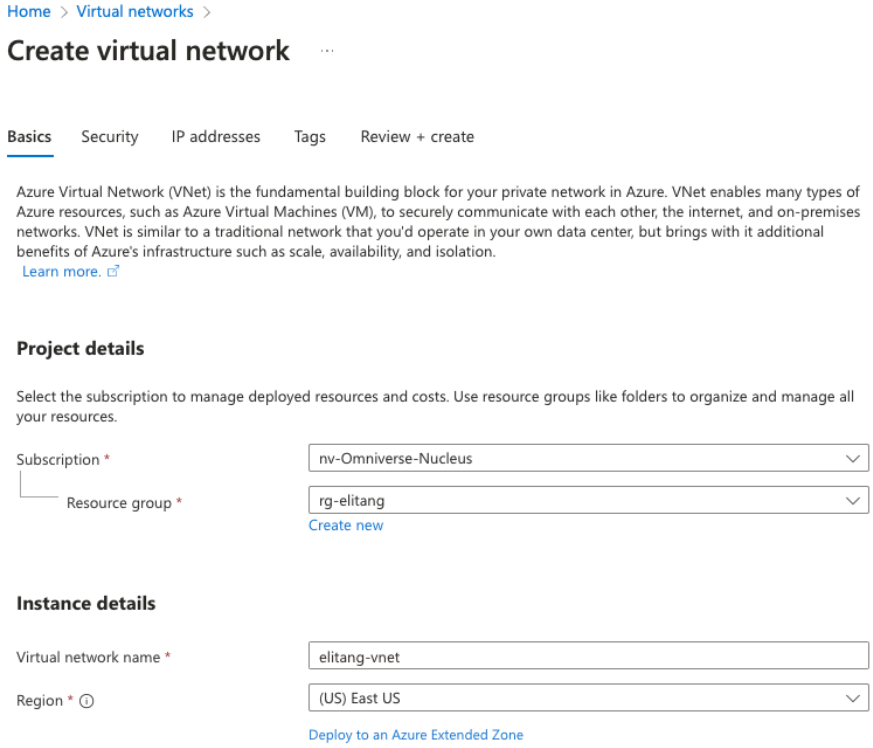
Search for resource groups and create one

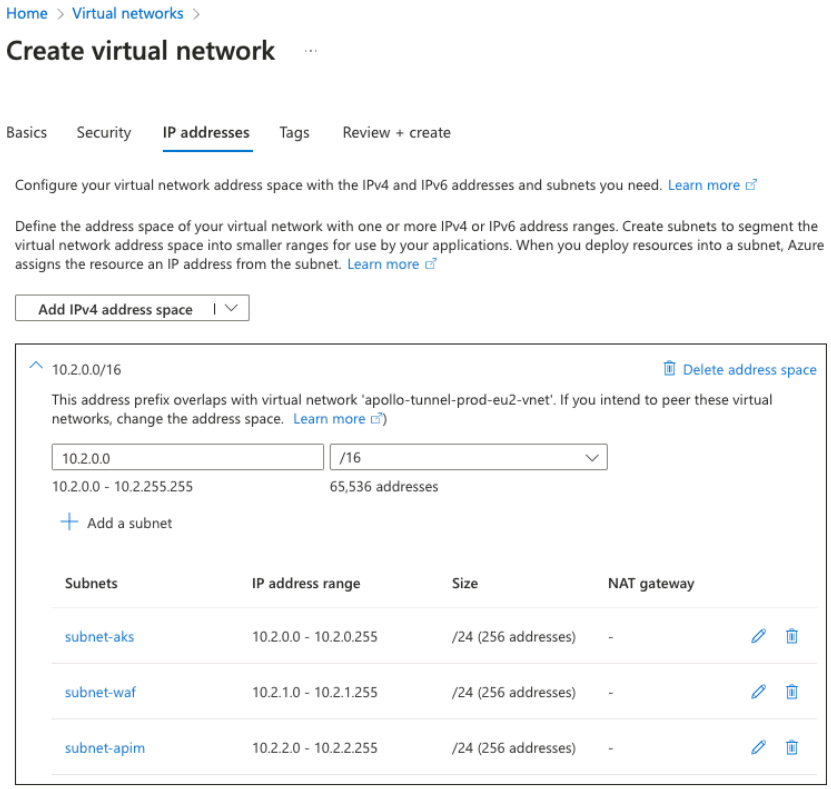


3.3 VNET/Subnets

1. Create a new virtual network
   - **Range**: `10.2.0.0/16`

4 http://elitang-ovas-streaming.nvsbx.omniverse.nvidia.com
5 http://elitang-ovas-streaming.nvsbx.omniverse.nvidia.com
6 http://elitang-ovas-streaming.nvsbx.omniverse.nvidia.com

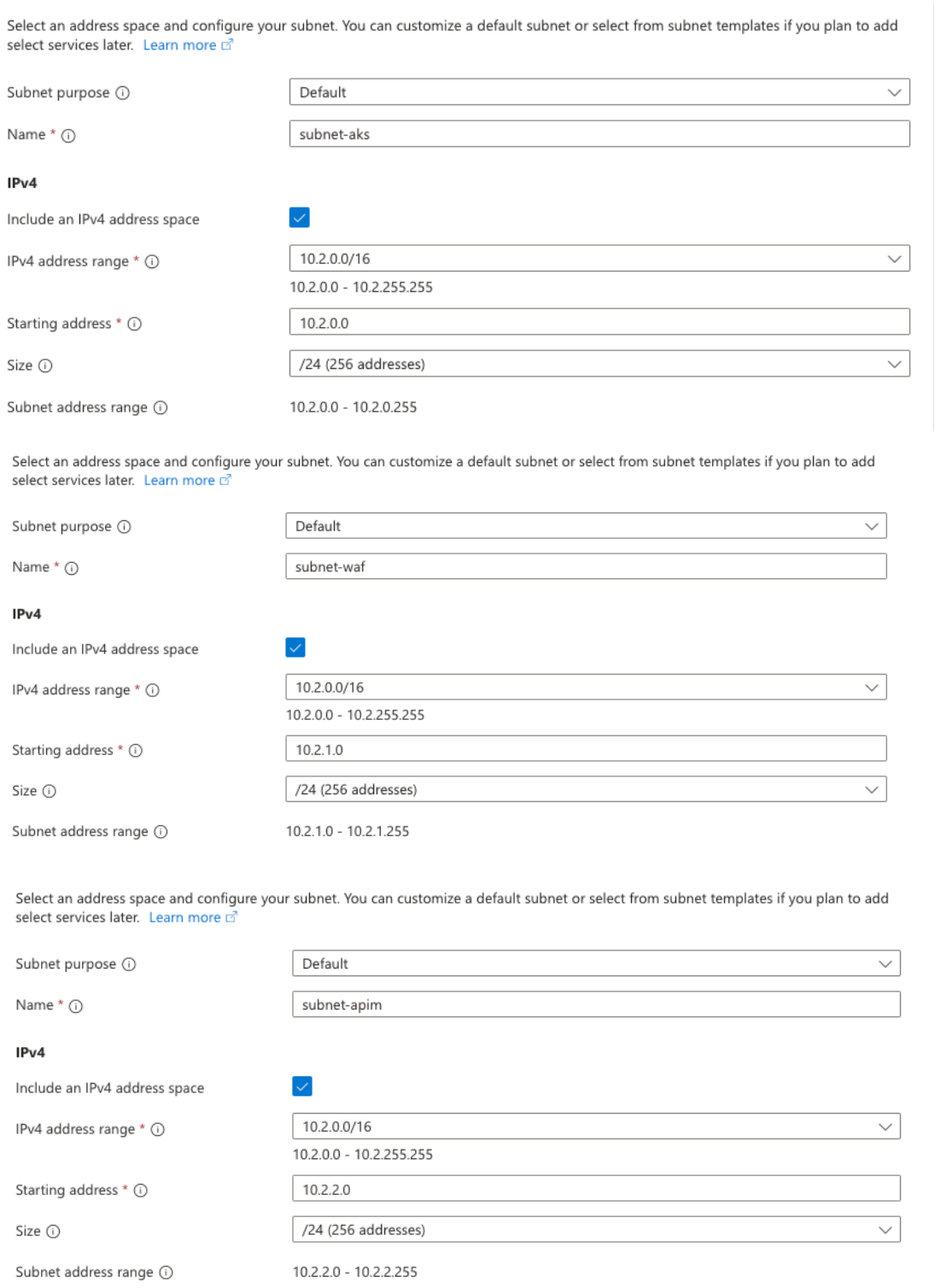


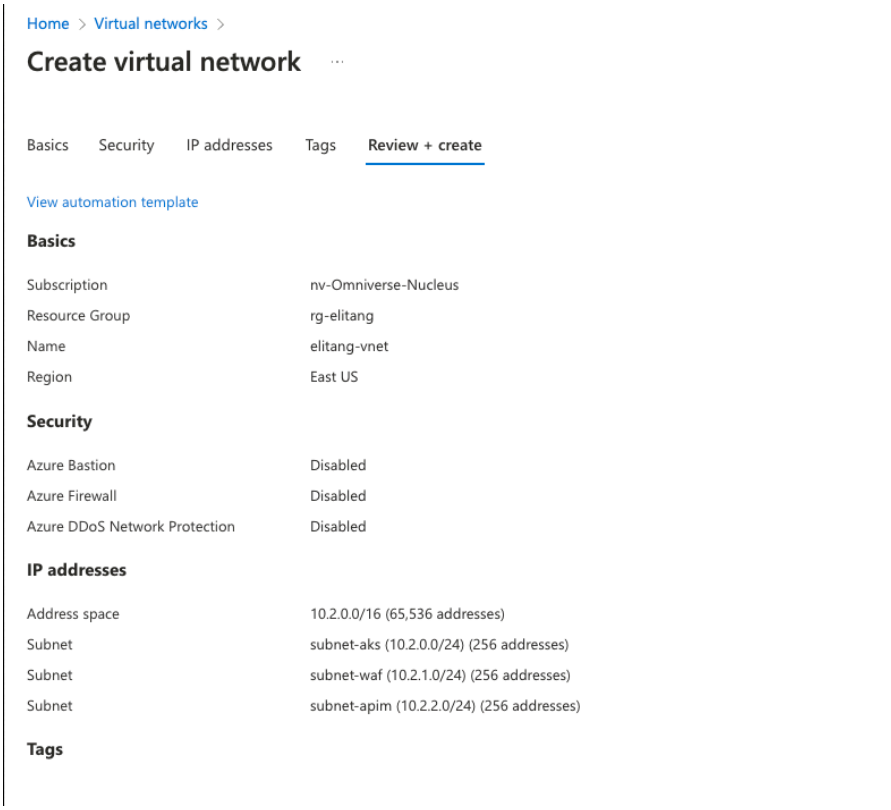


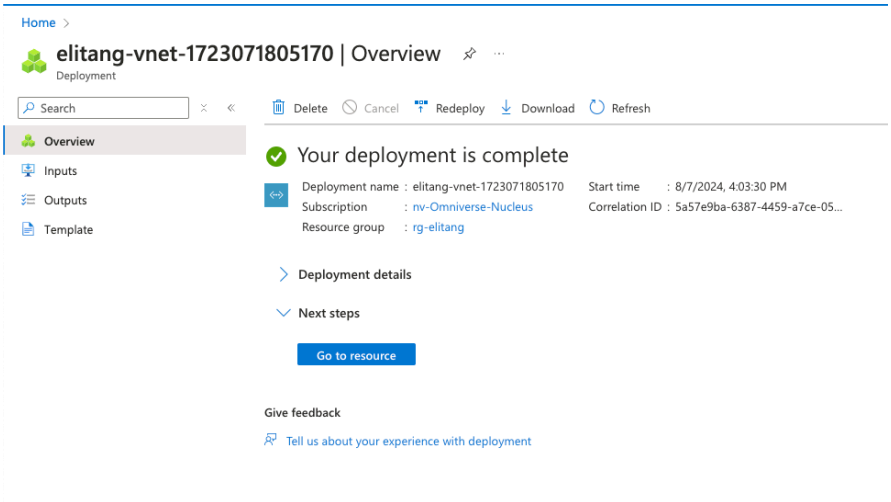
1. Create new subnets

- `subnet-aks` - `10.2.0.0/24`
- `subnet-waf` - `10.2.1.0/24`
- `subnet-apim` - `10.2.2.0/24`

See examples for each subnet below:

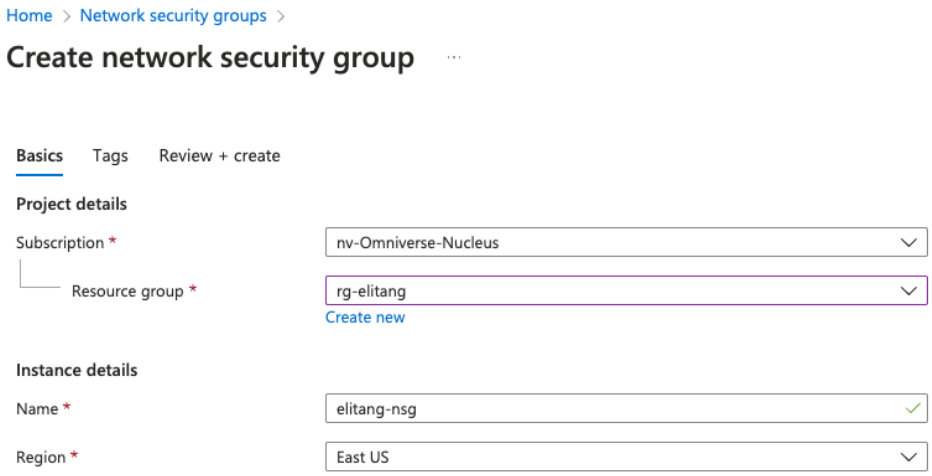

Select an address space and configure your subnet. You can customize a default subnet or select from subnet templates if you plan to add select services later. Learn more
Subnet purpose
Default
Name *
subnet-aks
IPv4
Include an IPv4 address space
IPv4 address range *
10.2.0.0/16
10.2.0.0 - 10.2.255.255
Starting address *
10.2.0.0
Size
/24 (256 addresses)
Subnet address range
10.2.0.0 - 10.2.0.255
Select an address space and configure your subnet. You can customize a default subnet or select from subnet templates if you plan to add select services later. Learn more
Subnet purpose
Default
Name *
subnet-waf
IPv4
Include an IPv4 address space
IPv4 address range *
10.2.0.0/16
10.2.0.0 - 10.2.255.255
Starting address *
10.2.1.0
Size
/24 (256 addresses)
Subnet address range
10.2.1.0 - 10.2.1.255
Select an address space and configure your subnet. You can customize a default subnet or select from subnet templates if you plan to add select services later. Learn more
Subnet purpose
Default
Name *
subnet-apim
IPv4
Include an IPv4 address space
IPv4 address range *
10.2.0.0/16
10.2.0.0 - 10.2.255.255
Starting address *
10.2.2.0
Size
/24 (256 addresses)
Subnet address range
10.2.2.0 - 10.2.2.255





Go to resource. We will need this later.

3.4 Create Network Security Group

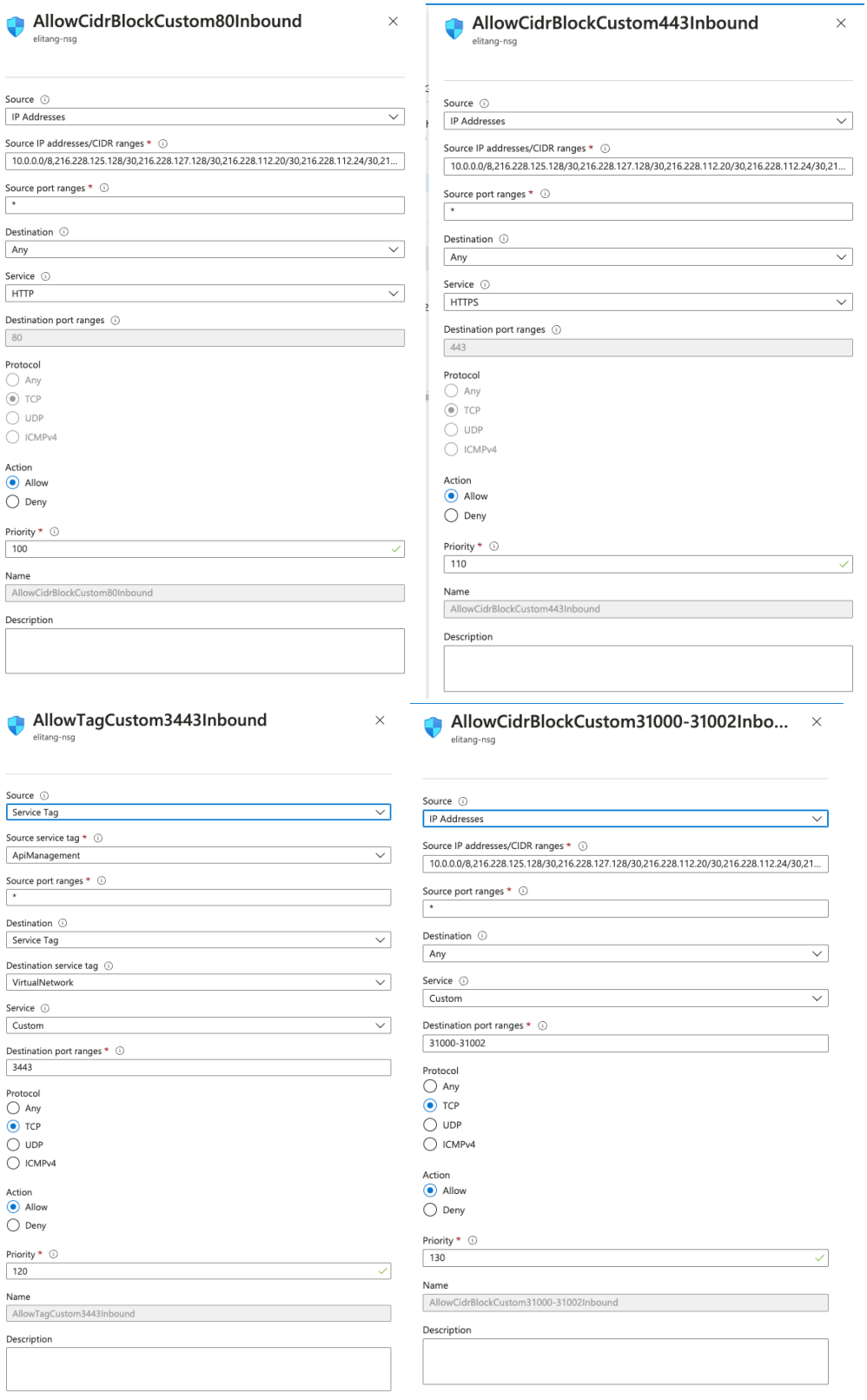


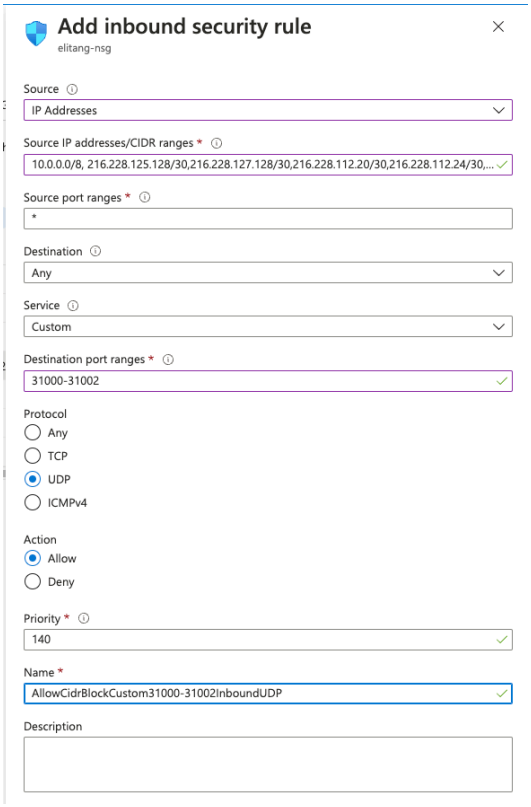
Create the following inbound rules:

- Add new rules
  - Port 80 (http)
    - **Source:** IP Addresses
    - **Source**: ideally your specific VPN CIDR blocks, also `10.0.0.0/8`
    - **Source Port Ranges: ***
    - **Destination:** Any
    - **Service**: HTTP
    - **Protocol**: TCP
  - Port 443 (https)
    - **Source:** IP Addresses
    - **Source**: ideally your specific VPN CIDR blocks, also `10.0.0.0/8`
    - **Source Port Ranges: ***
    - **Destination:** Any
    - **Service**: HTTPS
    - **Protocol**: TCP
  - Port 3443 (APIM management)
    - **Source**: `Service Tag`
    - **Source service tag**: `ApiManagement`

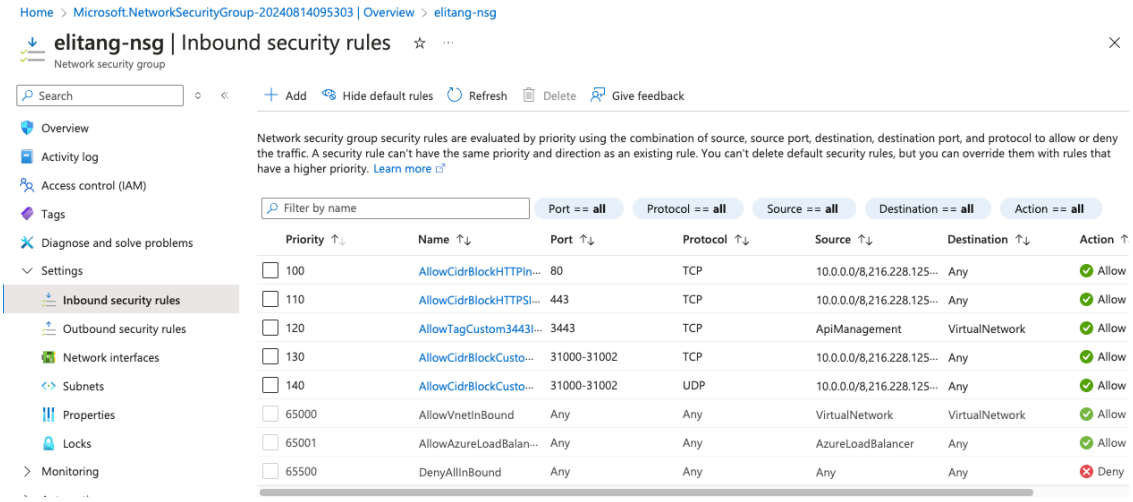
        - **Source port ranges**: `*`
        - **Destination**: `Service Tag`
        - **Destination service tag**: `VirtualNetwork`
        - **Service**: Custom
        - **Protocol**: TCP
        - **Destination port**: `3443`
    - Ports 31000-31002 (streaming)
        - **Source:** IP Addresses
        - **Source**: ideally your specific VPN CIDR blocks, also `10.0.0.0/8`
        - **Source Port Ranges: ***
        - **Destination:** Any
        - **Service**: Custom
        - **Destination Port Ranges**: 31000-31002
        - **Protocol**: TCP and UDP (need a separate rule for each)

See examples for each rule below:


AllowCidrBlockCustom80Inbound
elitang-nsg
Source
IP Addresses
Source IP addresses/CIDR ranges *
10.0.0.0/8,216.228.125.128/30,216.228.127.128/30,216.228.112.20/30,216.228.112.24/30,21...
Source port ranges *
*
Destination
Any
Service
HTTP
Destination port ranges
80
Protocol
Any
TCP
UDP
ICMPv4
Action
Allow
Deny
Priority *
100
Name
AllowCidrBlockCustom80Inbound
Description
AllowCidrBlockCustom443Inbound
elitang-nsg
Source
IP Addresses
Source IP addresses/CIDR ranges *
10.0.0.0/8,216.228.125.128/30,216.228.127.128/30,216.228.112.20/30,216.228.112.24/30,21...
Source port ranges *
*
Destination
Any
Service
HTTPS
Destination port ranges
443
Protocol
Any
TCP
UDP
ICMPv4
Action
Allow
Deny
Priority *
110
Name
AllowCidrBlockCustom443Inbound
Description
AllowTagCustom3443Inbound
elitang-nsg
Source
Service Tag
Source service tag *
ApiManagement
Source port ranges *
*
Destination
Service Tag
Destination service tag
VirtualNetwork
Service
Custom
Destination port ranges *
3443
Protocol
Any
TCP
UDP
ICMPv4
Action
Allow
Deny
Priority *
120
Name
AllowTagCustom3443Inbound
Description
AllowCidrBlockCustom31000-31002Inbo...
elitang-nsg
Source
IP Addresses
Source IP addresses/CIDR ranges *
10.0.0.0/8,216.228.125.128/30,216.228.127.128/30,216.228.112.20/30,216.228.112.24/30,21...
Source port ranges *
*
Destination
Any
Service
Custom
Destination port ranges *
31000-31002
Protocol
Any
TCP
UDP
ICMPv4
Action
Allow
Deny
Priority *
130
Name
AllowCidrBlockCustom31000-31002Inbound
Description

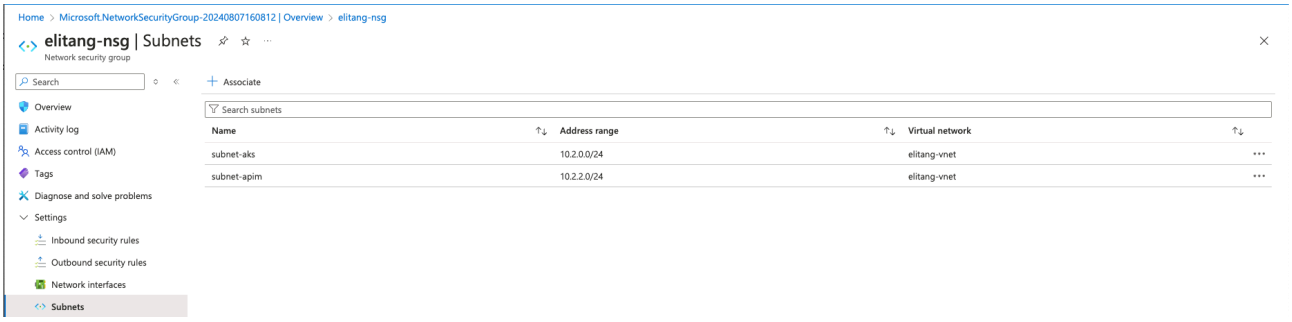


You should see 5 rules when finished.



3.4.1 Assign to `subnet-aks` and `subnet-apim`

- Click Settings > Subnets
- Click Associate and select both.

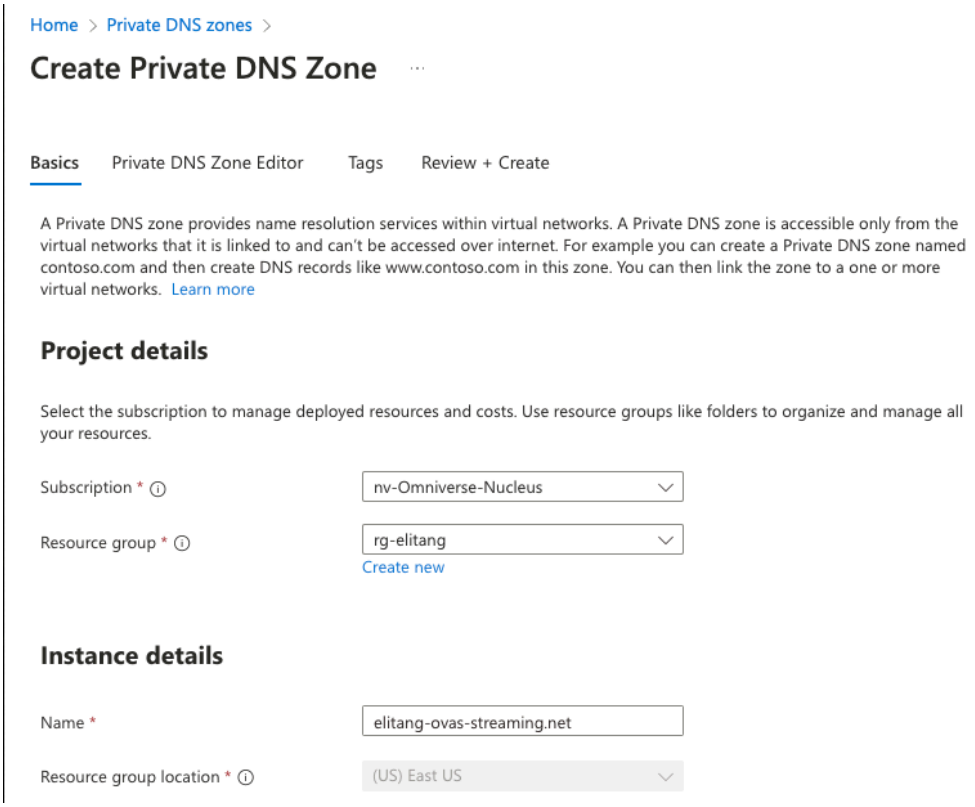


3.5 DNS/Certificates

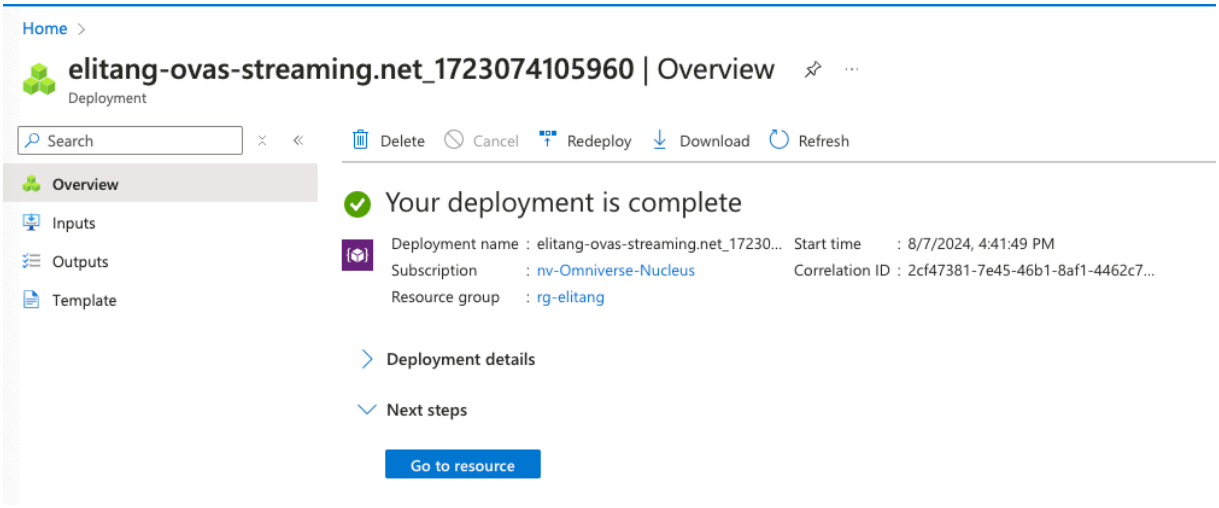
3.5.1 Create private DNS zone

- Add `Private DNS Zones` from the marketplace if not already added
  - `Name` is the name of the zone: `elitang-ovas-streaming.net`
  - Once created, under DNS Management > Virtual Network Links, link to vnet created in step 1

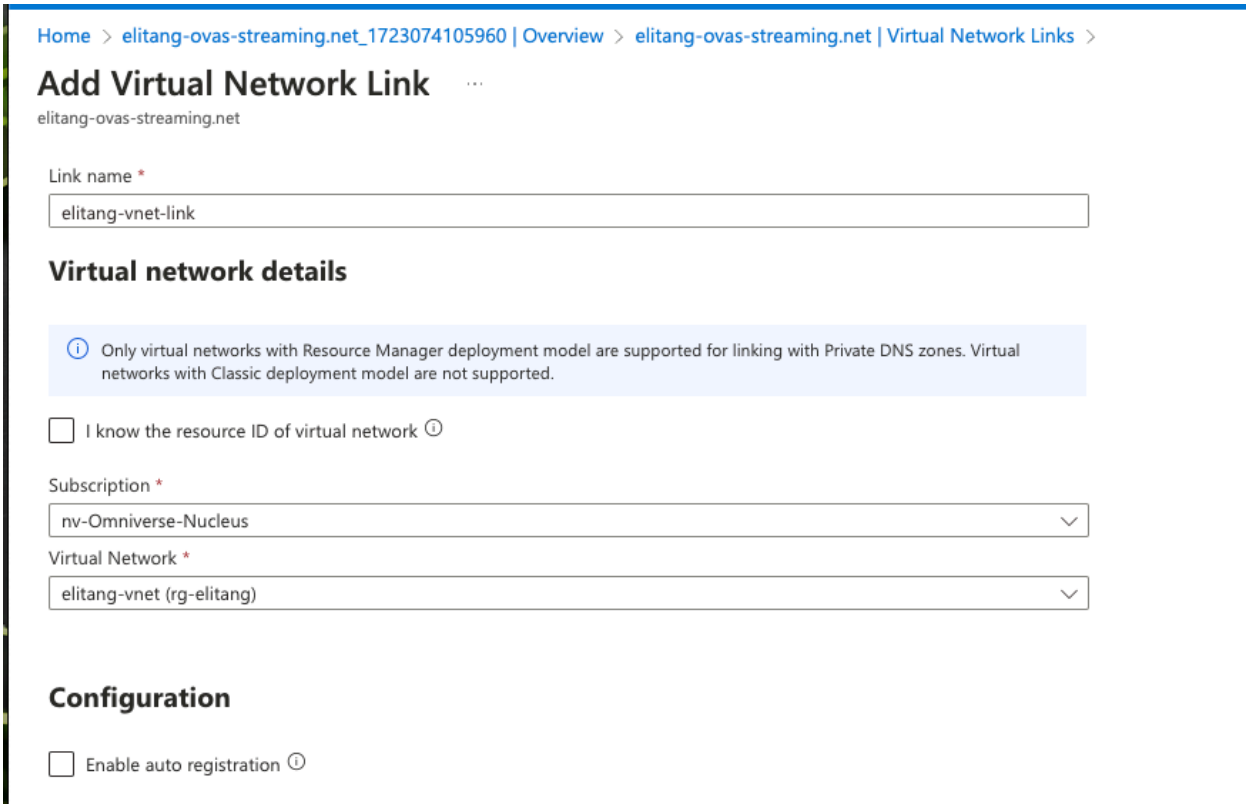
Example screenshots below:



Create and click go to resource when ready.



Add a virtual network link



3.5.2 Create self-signed certificates for private DNS zone

Use following Powershell script to generate both a root and SSL certificate:

1. Execute the following. Make changes to the CN

   a.
```
 # Create the root signing cert
 $root = New-SelfSignedCertificate -Type Custom -KeySpec Signature `
     -Subject "CN=ovas-streaming-net-signing-root" -KeyExportPolicy
Exportable `
     -HashAlgorithm sha256 -KeyLength 4096 `
     -CertStoreLocation "Cert:\CurrentUser\My" -KeyUsageProperty Sign `
     -KeyUsage CertSign -NotAfter (get-date).AddYears(5)
```

2. Execute the following command. Make changes to the -DnsName. Make changes to the CN.

   a.
```
 # Create the wildcard SSL cert.
 $ssl = New-SelfSignedCertificate -Type Custom -DnsName "*.elitang-ovas-
streaming.net","elitang-ovas-streaming.net" `
     -KeySpec Signature `
     -Subject "CN=*.ovas-streaming.net" -KeyExportPolicy Exportable `
     -HashAlgorithm sha256 -KeyLength 2048 `
     -CertStoreLocation "Cert:\CurrentUser\My" `
     -Signer $root
```

3. Execute the following commands to create the two certificates. These will be stored in the directory you are currently in.

   a.
```
 # Export CER of the root and SSL certs
 Export-Certificate -Type CERT -Cert $root -FilePath .\ovas-streaming-
signing-root.cer
 Export-Certificate -Type CERT -Cert $ssl -FilePath .\ovas-streaming-
ssl.cer
```

4. Execute the following commands to create the two certificates. These will be stored in the directory you are currently in. NOTE: Password is required.

   a.
```
# Export PFX of the root and SSL certs
 Export-PfxCertificate -Cert $root -FilePath .\ovas-streaming-signing-
root.pfx `
     -Password (read-host -AsSecureString -Prompt "password")
```

   b.
```
 Export-PfxCertificate -Cert $ssl -FilePath .\ovas-streaming-ssl.pfx `
     -ChainOption BuildChain -Password (read-host -AsSecureString
-Prompt "password")
```

3.5.3 Create LetsEncrypt certificates manually for public DNS zone

Create LetsEncrypt certificates manually for public DNS zone https://certbot.eff.org/

Change the domains to your public DNS.

```
sudo certbot certonly \
    --manual \
    --preferred-challenges=dns \
    --email user@nvidia.com \
    --server https://acme-v02.api.letsencrypt.org/directory \
    --agree-tos \
    -d elitang-ovas-streaming.nvsbx.omniverse.nvidia.com \
    -d '*.elitang-ovas-streaming.nvsbx.omniverse.nvidia.com'
```

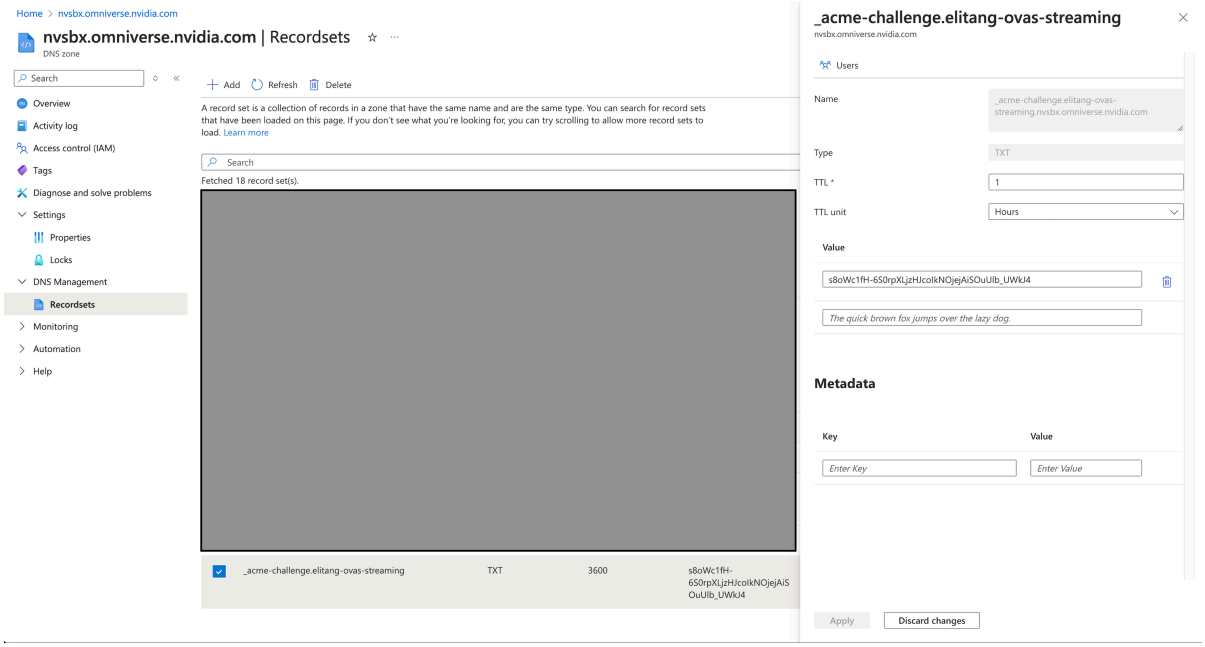
- This process requires access to the zone in order to manually create and verify TXT records
- The certificates will be created in `/etc/letsencrypt/live/ovas-streaming.nvsbx.omniverse.nvidia.com`[7]
- Example output

```
elitang@elitang-mlt ~ % sudo certbot certonly \
        --manual \
        --preferred-challenges=dns \
        --email user@nvidia.com \
        --server https://acme-v02.api.letsencrypt.org/directory \
        --agree-tos \
        -d elitang-ovas-streaming.nvsbx.omniverse.nvidia.com \
        -d '*.elitang-ovas-streaming.nvsbx.omniverse.nvidia.com'
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Requesting a certificate for elitang-ovas-streaming.nvsbx.omniverse.nvidia.com
and *.elitang-ovas-streaming.nvsbx.omniverse.nvidia.com

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Please deploy a DNS TXT record under the name:

_acme-challenge.elitang-ovas-streaming.nvsbx.omniverse.nvidia.com.

with the following value:

s8oWc1fH-6S0rpXLjzHJcoIkNOjejAiSOuUlb_UWkJ4

Before continuing, verify the TXT record has been deployed. Depending on the
DNS
```

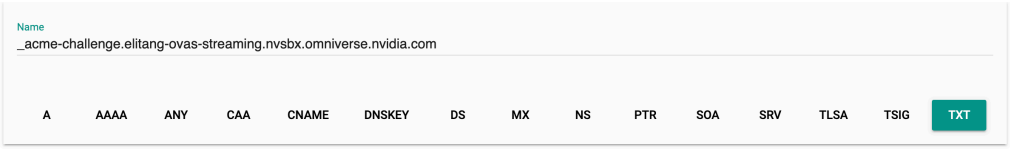
7 http://ovas-streaming.nvsbx.omniverse.nvidia.com

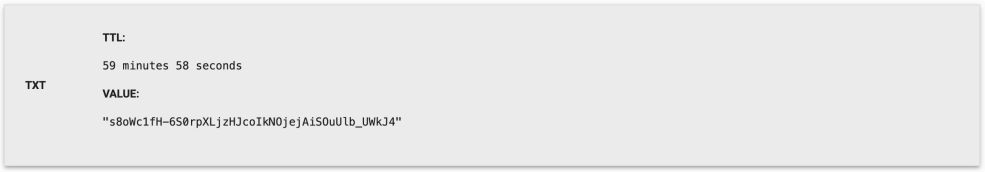
```
provider, this may take some time, from a few seconds to multiple minutes. You
can
check if it has finished deploying with aid of online tools, such as the Google
Admin Toolbox: https://toolbox.googleapps.com/apps/dig/#TXT/_acme-
challenge.elitang-ovas-streaming.nvsbx.omniverse.nvidia.com.
Look for one or more bolded line(s) below the line ';ANSWER'. It should show
the
value(s) you've just added.
```

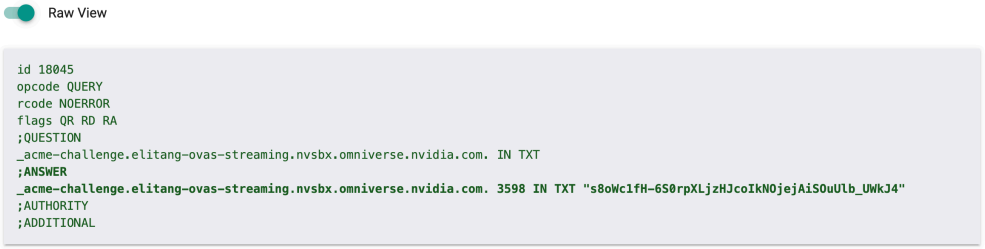
- Add the record in Recordsets



- Verify that the record show up with the link provided in the command output above:







-
- Once you have verified that the TXT entry shows up press Enter
-

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Press Enter to Continue

Successfully received certificate.
Certificate is saved at: /etc/letsencrypt/live/elitang-ovas-
streaming.nvsbx.omniverse.nvidia.com/fullchain.pem
Key is saved at:         /etc/letsencrypt/live/elitang-ovas-
streaming.nvsbx.omniverse.nvidia.com/privkey.pem
This certificate expires on 2024-11-07.
These files will be updated when the certificate renews.

NEXT STEPS:
- This certificate will not be renewed automatically. Autorenewal of --manual
certificates requires the use of an authentication hook script (--manual-auth-
hook) but one was not provided. To renew this certificate, repeat this same
certbot command before the certificate's expiry date.

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
If you like Certbot, please consider supporting our work by:
 * Donating to ISRG / Let's Encrypt:   https://letsencrypt.org/donate
 * Donating to EFF:                    https://eff.org/donate-le
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
elitang@elitang-mlt ~ %
```

If this fails, try running the command again or waiting longer.

3.5.4 Create a `.pfx` from the full chain certificate and private key

Note: When creating this certificate, a password is required.

```
openssl pkcs12 -export -in /etc/letsencrypt/live/elitang-ovas-
streaming.nvsbx.omniverse.nvidia.com/fullchain.pem -inkey /etc/letsencrypt/live/
elitang-ovas-streaming.nvsbx.omniverse.nvidia.com/privkey.pem -out
nvsbx.omniverse.nvidia.com.pfx
```

Your file will be named `nvsbx.omniverse.nvidia.com`[8]`.pfx.` Change the value after `-out` to name it differently. Save the location of this file, we will need to upload it to Azure later.

3.6 AKS Cluster

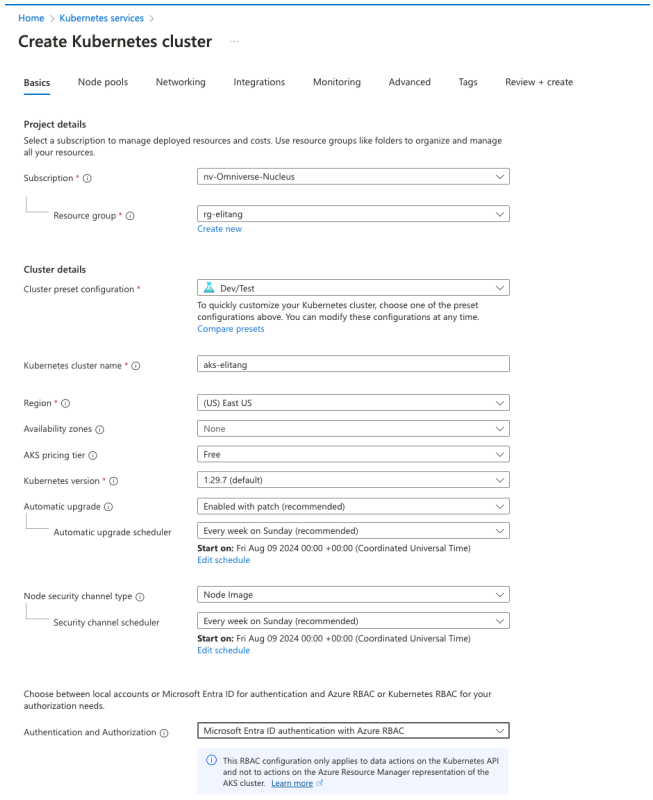
Create new AKS cluster

3.6.1 Basics

- Name
  - `aks-elitang`
- Authentication
  - Microsoft Entra ID with Azure RBAC

Example:

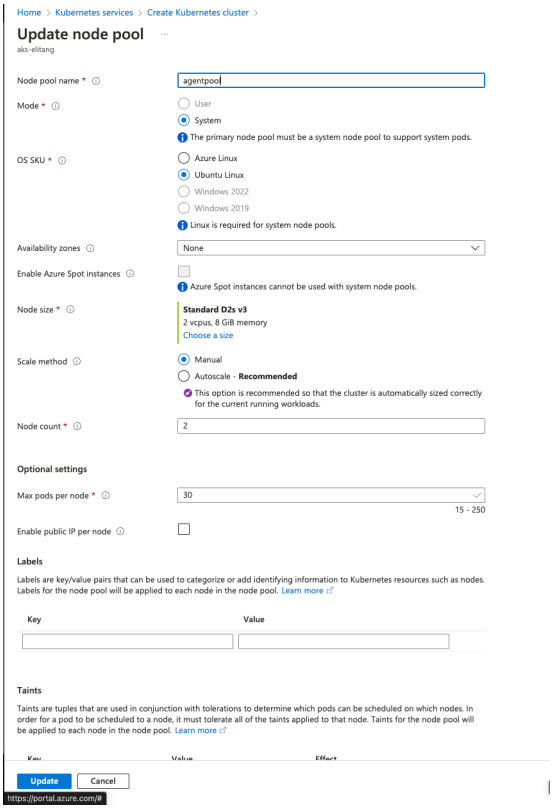
8 http://nvsbx.omniverse.nvidia.com



3.6.2 Nodepools

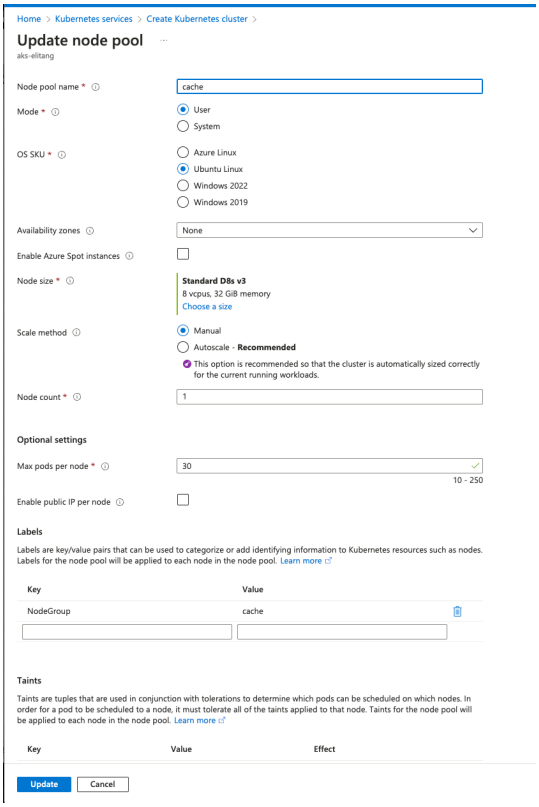
- `agentpool` (Default)
  - Mode `system` (optional)
  - `Standard_D2s_v3`
  - Manual scaling
  - 2-3 nodes
  - Max Pods per node: `30`
  - Note: The name of this node will be applied as a label in AKS. So if you leave it as the default, a label of agentpool=agentpool will be created. If you change the name to agentpool2, the label will be agentpool=agentpool2. Remember this label later, during the OVAS Stack deployment steps.

Example:



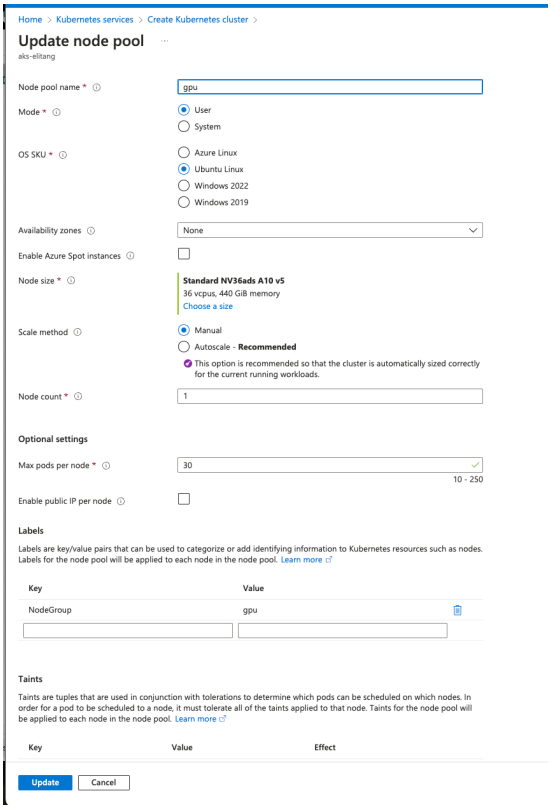
- `cache`
  - `Standard_D8s_v3`
  - Manual scaling
  - 1 node
  - Label: `NodeGroup=cache`
  - Max Pods per node: `30`

Example:



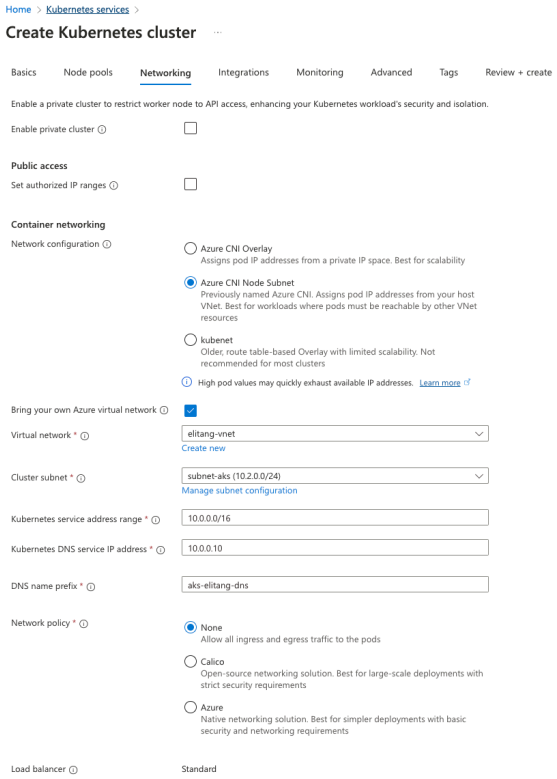
- `gpu`
  - `NV36ads_A10_v5`
  - Manual scaling
  - 1 node
  - Label: `NodeGroup=gpu`
  - Max Pods per node: `30`

Example:

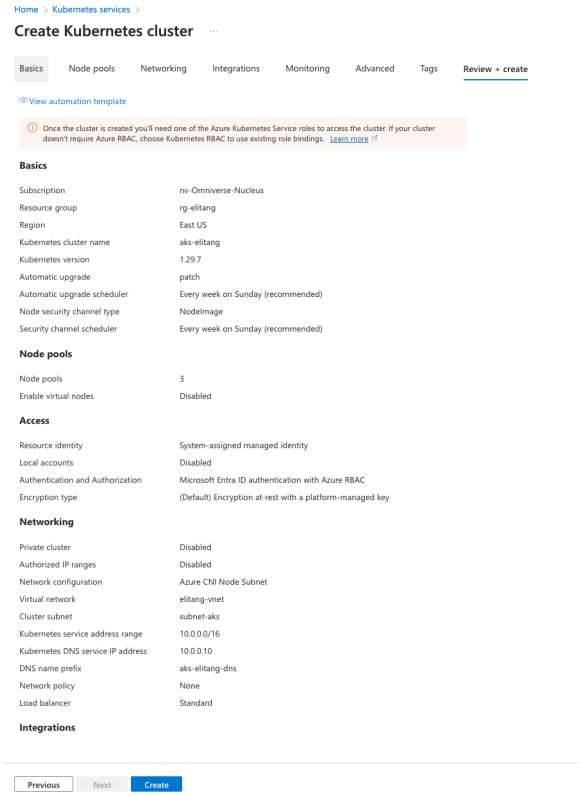


3.6.3 Networking

- **Container configuration**: `Azure CNI Node Subnet`
- Check `Bring your own Azure virtual network`
    - Choose vnet created in step 1
        - It may autofill with something similar to `(New) rg-elitang-vnet` , do not select this one, search for your vnet created before.
    - Choose `subnet-aks` subnet


Home > Kubernetes services >
Create Kubernetes cluster
Basics
Node pools
Networking
Integrations
Monitoring
Advanced
Tags
Review + create
Enable a private cluster to restrict worker node to API access, enhancing your Kubernetes workload's security and isolation.
Enable private cluster
Public access
Set authorized IP ranges
Container networking
Network configuration
Azure CNI Overlay
Assigns pod IP addresses from a private IP space. Best for scalability
Azure CNI Node Subnet
Previously named Azure CNI. Assigns pod IP addresses from your host VNet. Best for workloads where pods must be reachable by other VNet resources
kubenet
Older, route table-based Overlay with limited scalability. Not recommended for most clusters
High pod values may quickly exhaust available IP addresses. Learn more
Bring your own Azure virtual network
Virtual network *
elitang-vnet
Create new
Cluster subnet *
subnet-aks (10.2.0.0/24)
Manage subnet configuration
Kubernetes service address range *
10.0.0.0/16
Kubernetes DNS service IP address *
10.0.0.10
DNS name prefix *
aks-elitang-dns
Network policy *
None
Allow all ingress and egress traffic to the pods
Calico
Open-source networking solution. Best for large-scale deployments with strict security requirements
Azure
Native networking solution. Best for simpler deployments with basic security and networking requirements
Load balancer
Standard

3.6.4 Review and deploy:



Go to resource when completed.

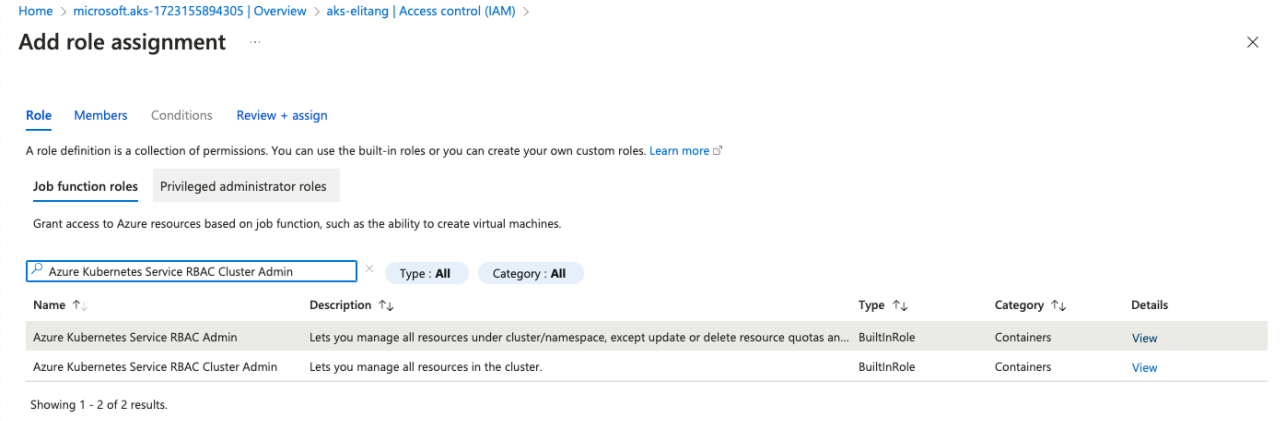
3.7 Post-deployment

3.7.1 Accessing Kubernetes (RBAC Required to access cluster resources)

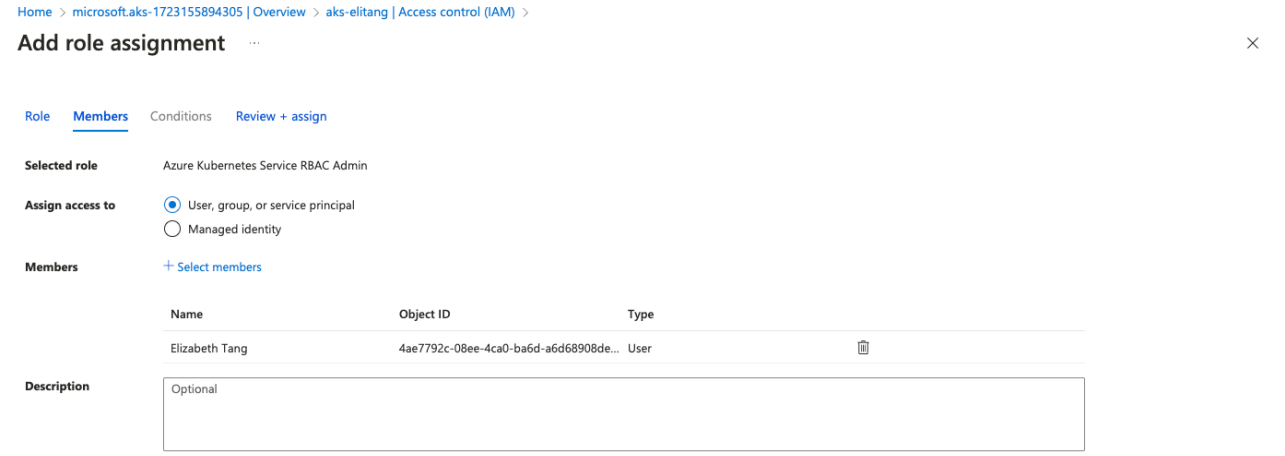
This is required to access the AKS cluster and manage resources inside Kubernetes. Add any user here that might need to access the cluster.

- RBAC
  - Click Access control (IAM)
  - Add self to `Azure Kubernetes Service RBAC Cluster Admin`
  - Add self to `Azure Kubernetes Service RBAC Admin`
  - These are done one at a time

Find role:



- Find your name:



- Review and Assign both roles.

3.7.2 Pull kubeconfig locally and check access

Log in to the `az` CLI with `az login`

Ensure you are in the correct subscription with `az account set --subscription <subscription-id>`

Install kubelogin: https://azure.github.io/kubelogin/install.html

Run the following commands:

```
elitang@elitang-mlt ~ % az aks get-credentials --format azure --resource-group rg-
elitang --name aks-elitang
Merged "aks-elitang" as current context in /Users/elitang/.kube/config
elitang@elitang-mlt ~ % export KUBECONFIG=/Users/elitang/.kube/config
elitang@elitang-mlt ~ % kubelogin convert-kubeconfig
elitang@elitang-mlt ~ % kubectl get nodes
To sign in, use a web browser to open the page https://microsoft.com/devicelogin and
enter the code BV2FPJZ5E to authenticate.
No resources found in default namespace.
elitang@elitang-mlt ~ % kubectl get nodes
NAME                                STATUS   ROLES    AGE   VERSION
aks-agentpool-35741025-vmss000000   Ready    <none>   20m   v1.29.7
aks-agentpool-35741025-vmss000001   Ready    <none>   20m   v1.29.7
aks-cache-35741025-vmss000000       Ready    <none>   20m   v1.29.7
aks-gpu-35741025-vmss000000         Ready    <none>   19m   v1.29.7
```

Check that the nodes you created appear.

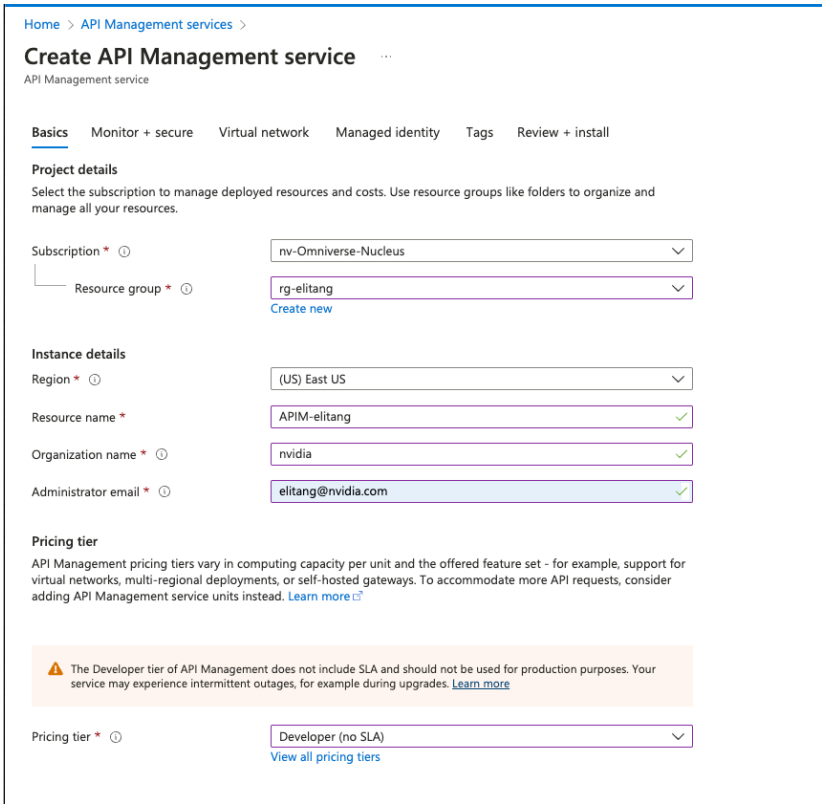
3.8 API Management (APIM) Service

3.8.1 Create the base APIM service:

- **Org name**: `nvidia`
- **Admin email**: `username@nvidia.com`[9]
- **Pricing tier**: `Developer`

Example:

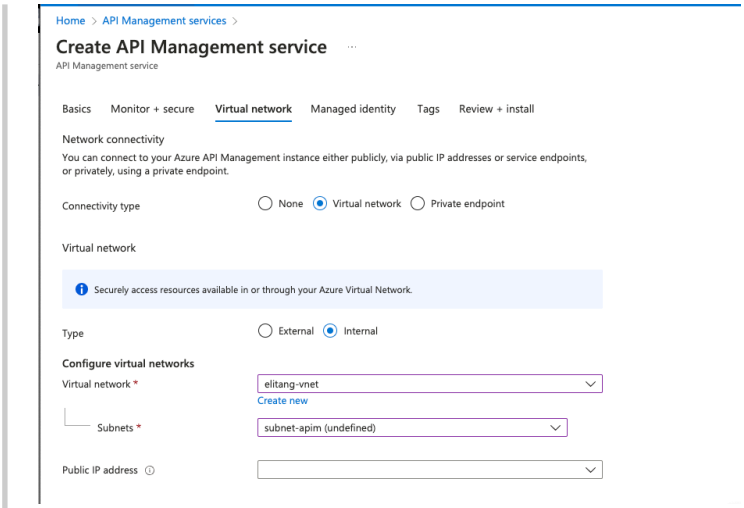
9 mailto:username@nvidia.com



Virtual Network

- **Connectivity type**: `Virtual Network`
- **Type**: `Internal`
- **Vnet**: (created in step 1)
- **Subnet**: `subnet-apim`

Example:



Review and install

> *Note: The APIM service can take up to an hour to deploy*

3.8.2 Post Deployment

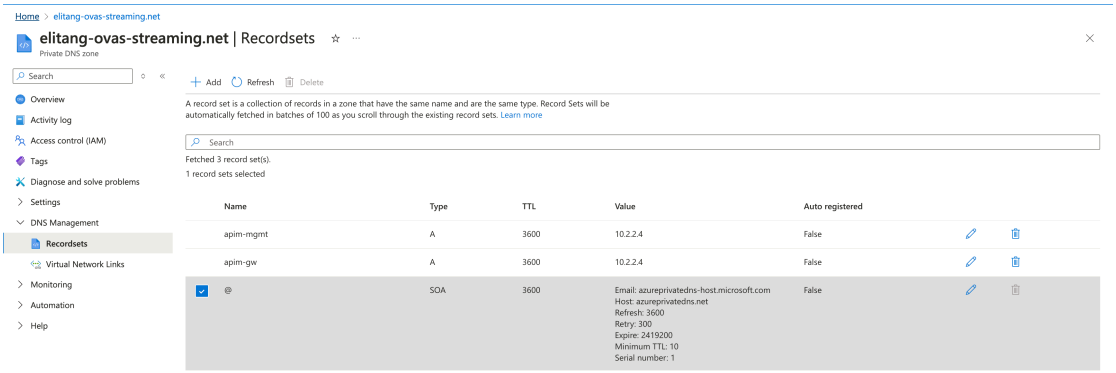
1. Once deployed, add a custom domain:
    - Select `Custom Domains` under `Deployment and Infrastructure`
    - Add a new custom domain for the gateway:
        - **Type**: `Gateway`
        - **Hostname**: `apim-gw.elitang-ovas-streaming.net`[10] ← This should be apim-gw.<your private DNS>
        - **Certificate**: `Custom`
        - **Certificate file**: `ovas-streaming-ssl.pfx`
        - Check `Default SSL binding`
    - Add a new custom domain for the management gateway:
        - **Type**: `Management`
        - **Hostname**: `apim-mgmt.elitang-ovas-streaming.net`[11] ← This should be apim-mgmt.<your private DNS>
        - **Certificate**: `Custom`
        - **Certificate file**: `ovas-streaming-ssl.pfx`

    > *Note: These settings can take up to an hour to apply*

2. Add DNS records to private DNS zone
    - Add two records for `apim-gw` and `apim-mgmt` to previously created private DNS zone pointing to private IP of the APIM instance created above
    - Example:

10 http://apim-gw.ovas-streaming.net

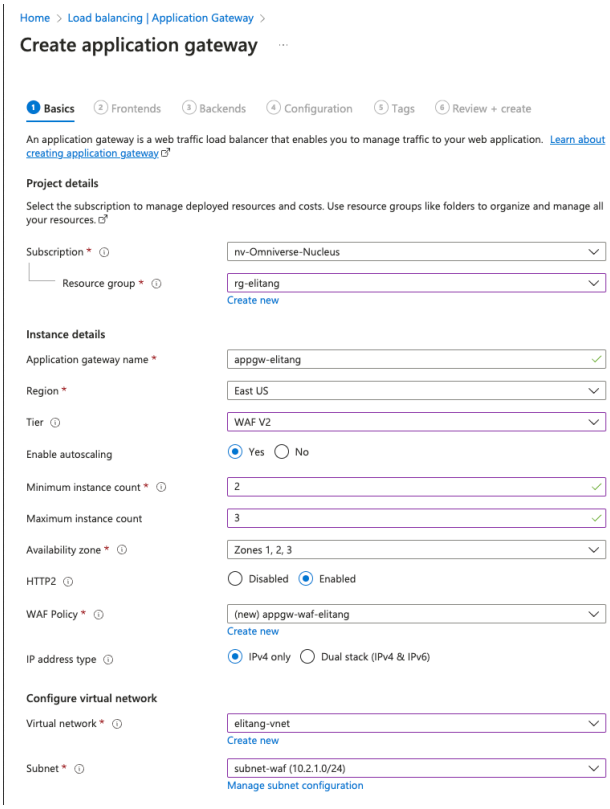
11 http://apim-mgmt.ovas-streaming.net



3.9 Application Gateway (with WAF)

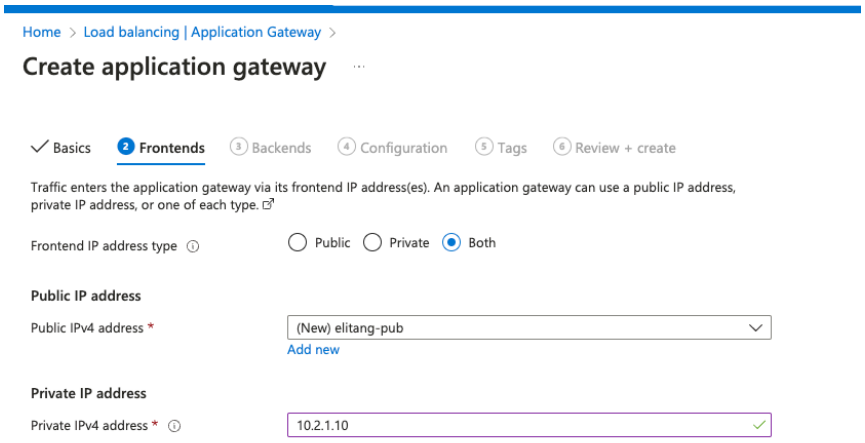
3.9.1 Basics

- **Tier**: WAFv2
- **Autoscaling**: (optional) min 2, max 3
- **Policy**: Create new (can use defaults)
- **Vnet**: Created in step 1
- **Subnet**: `subnet-waf`



3.9.2 Frontends

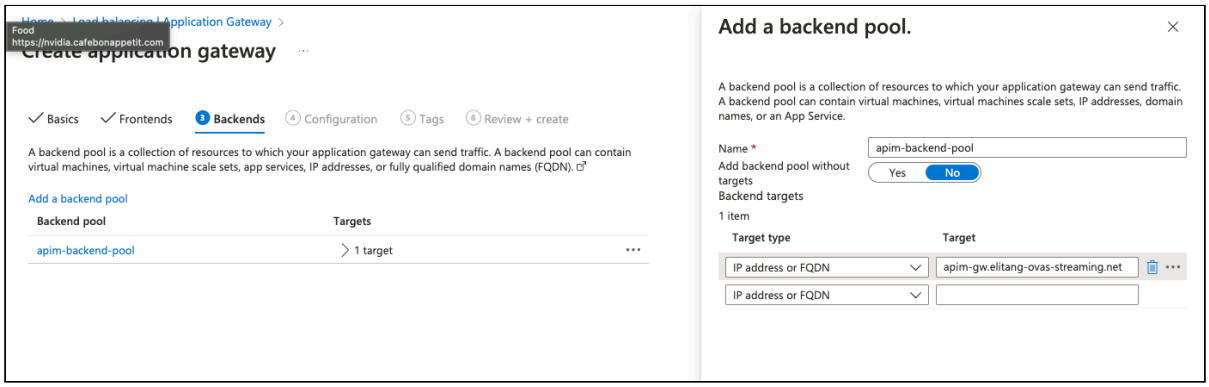
- **Type**: `both`
- **Public IP**: Create new
- **Private IP**: Choose IP in private subnet range (e.g. `10.2.1.10`)



-

3.9.3 Backends

- Create new
- **Name**: `apim`
- **Backend target**: `apim-gw.elitang-ovas-streaming.net`[12] ← This should be apim-gw.<your private DNS>



-

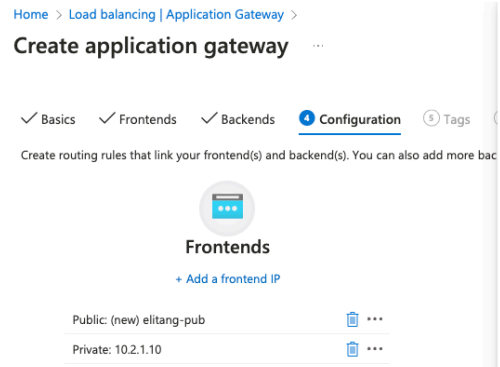
3.9.4 Configuration

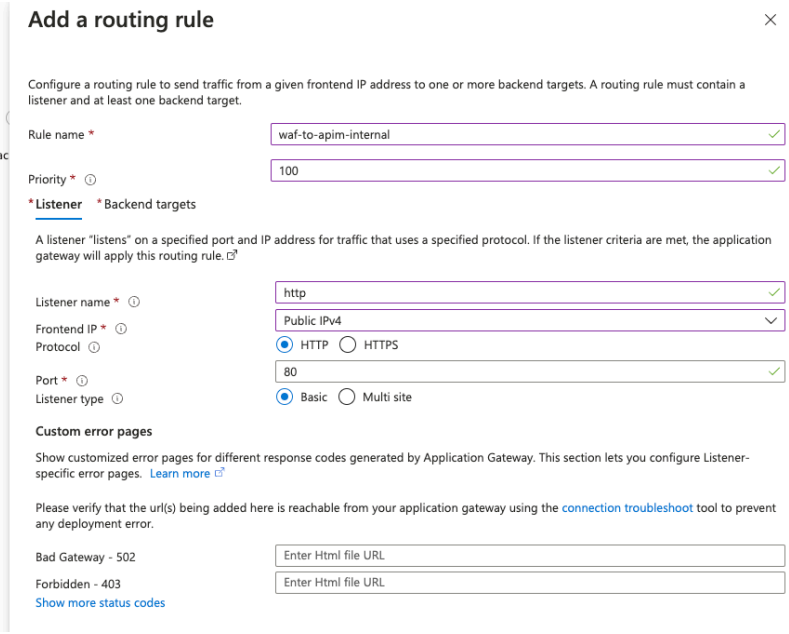
- Add new routing rule

12 http://apim-gw.ovas-streaming.net

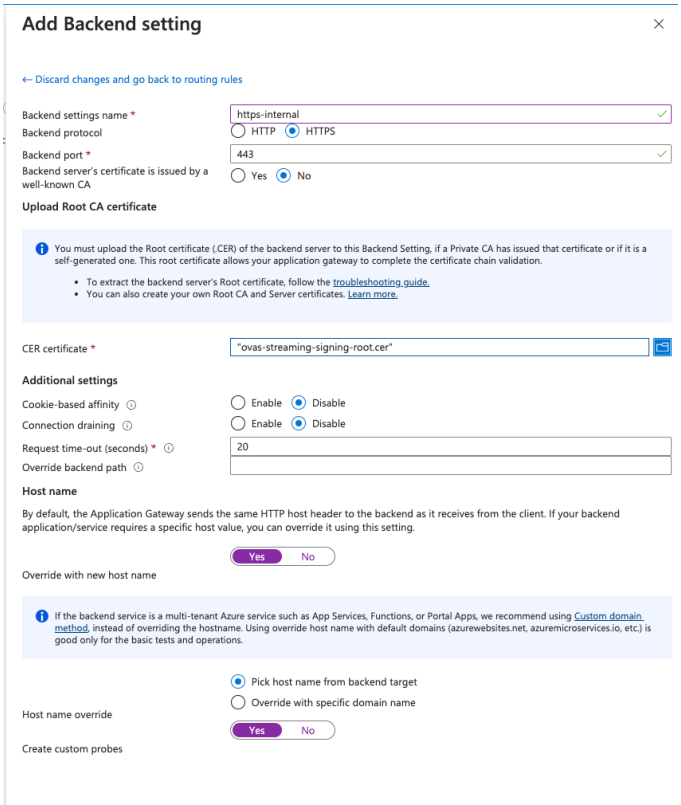
- **Name**: `waf-to-apim-internal`
- Priority `100`
- Listener
    - **Name**: `http`
    - **Frontend IP**: `public IPv4`
    - **Protocol**: `http`
    - **Port**: `80`

Example:





- Backend targets
    - **Target**: `apim` (backend created above)
    - Create new backend setting
        - **Name**: `https-internal`
        - **Protocol**: `https`
        - **Server certificate is signed by well-known authority**: `No`
        - **CER certificate**: `ovas-streaming-signing-root.cer`
        - **Override with new host name**: `Yes`
            - Select `Pick hostname from backend target`
        - **Create custom probes**: `Yes`
- Example

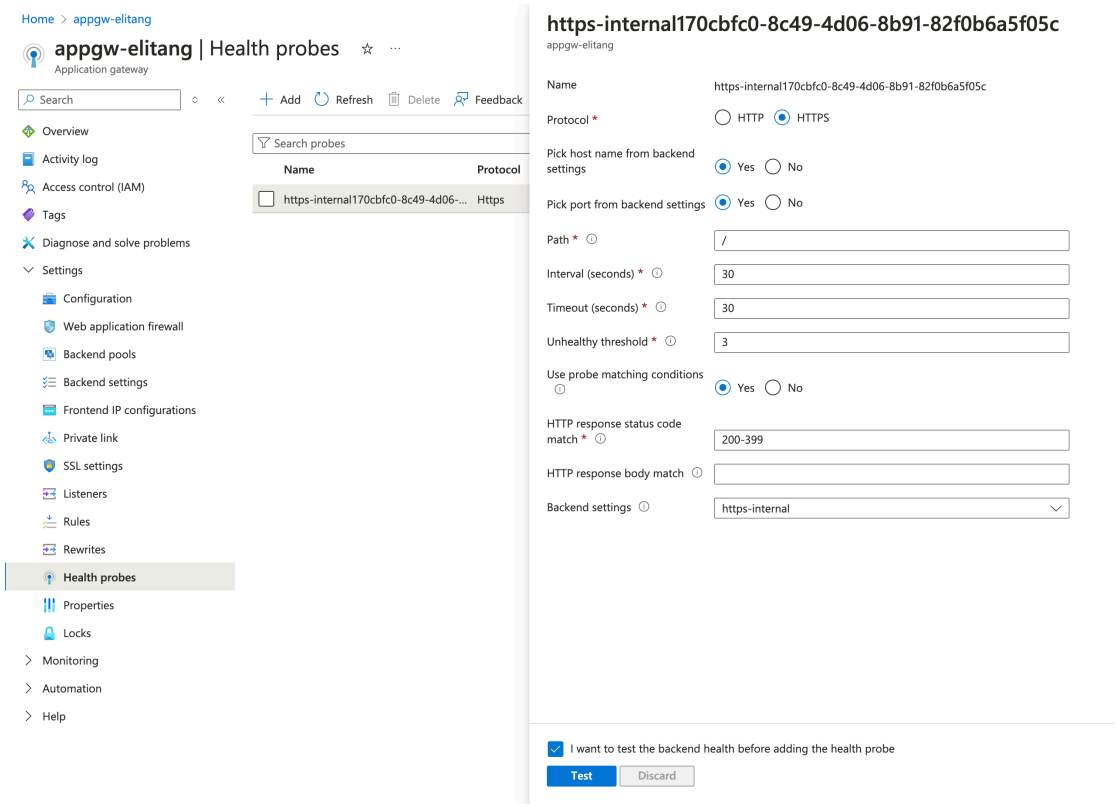


3.9.5 Once deployed go to app gateway:

3.9.5.1 Health Probe

Click Settings > Health Probe. Click the test and check that it works.

- **Name**: `https`
- **Pick hostname from backend settings**: `Yes`
- **Path**: `/status-0123456789abcdef`
- **Use probe matching conditions**: `Yes`
- **Backend settings**: choose existing one
- Test the probe; it should be successful
- Click `Add`

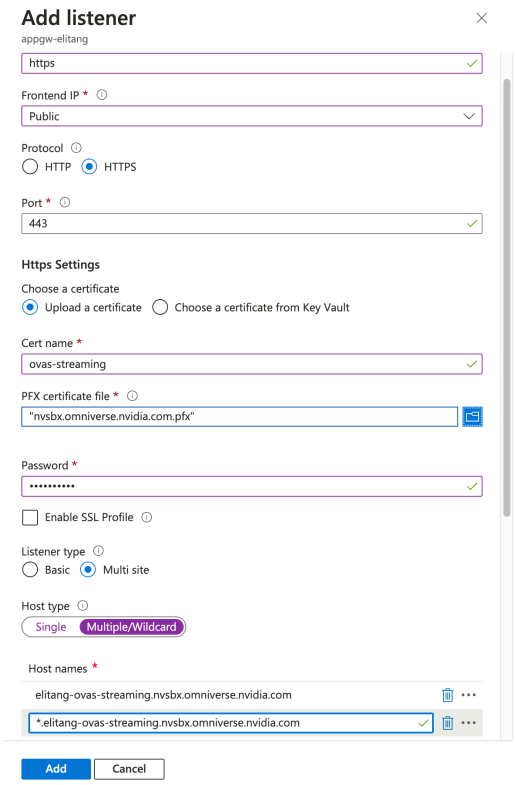


3.9.5.2 HTTPS Listener

Add a new HTTPS listener (optional; adds TLS termination at AppGw)

- Under `Settings > Listeners`, click `+ Add Listener`
  - **Name**: `https`

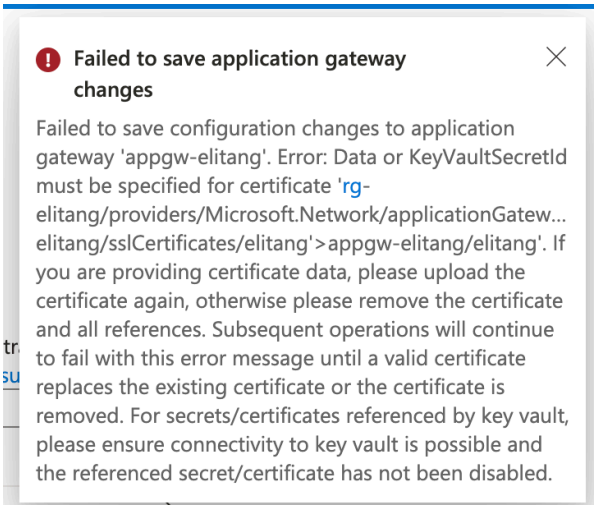
- **Frontend IP**: `public`
- **Protocol**: `https`
- **Certificate**: `Upload a certificate`
- **Name**: `ovas-streaming`
- **PFX Certificate File**: (`.pfx` file created earlier)
- **Password**:
- **Listener type**: `Multi site`
- **Host type**: `Multiple/Wildcard`
- **Host names**:
    - `elitang-ovas-streaming.nvsbx.omniverse.nvidia.com`[13]
    - `*.elitang-ovas-streaming.nvsbx.omniverse.nvidia.com`[14]



- If you get an error like this:

13 http://ovas-streaming.nvsbx.omniverse.nvidia.com

14 http://ovas-streaming.nvsbx.omniverse.nvidia.com

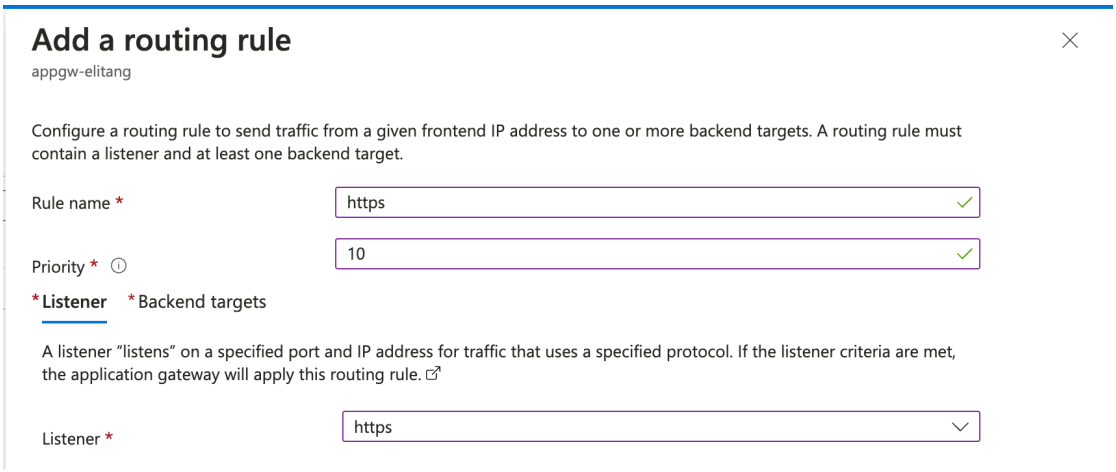


- Change the permissions on the pfx file. An example on Mac/Linux to allow any access to file:

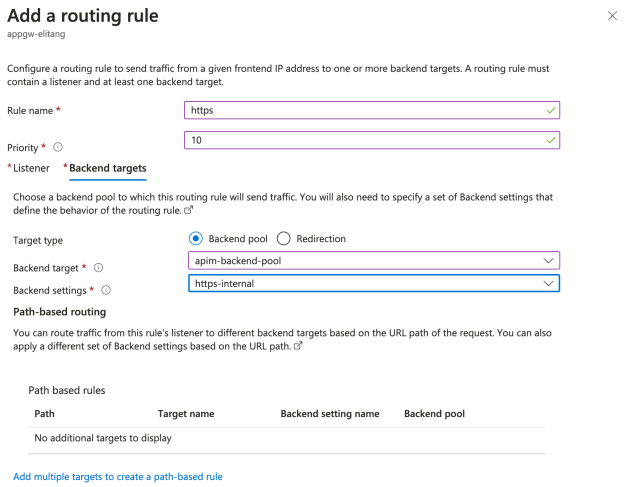
```
elitang@elitang-mlt ~ % sudo chmod 777 nvsbx.omniverse.nvidia.com.pfx
```

3.9.5.3 Routing Rules

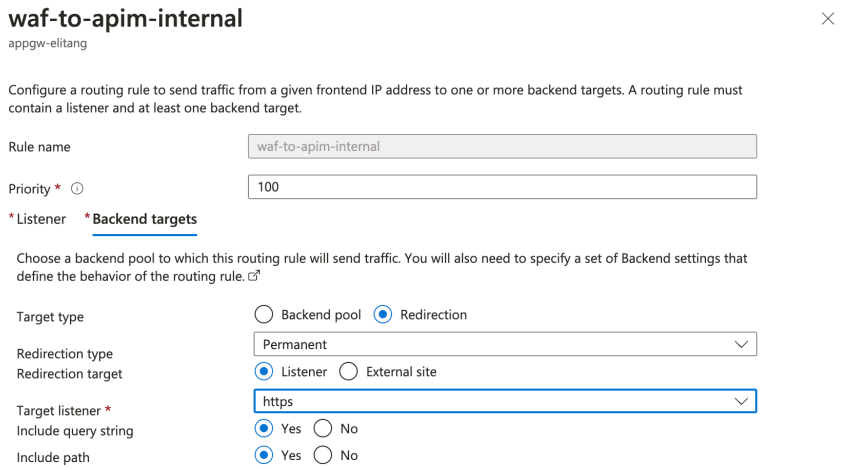
- Under `Settings > Rules`, click `+ Routing rule`
  - **Name**: `https`
  - **Priority**: `10`
  - **Listener**: `https`



  - Backend pool
    - **Backend target**: `apim`
    - **Backend settings**: `https-internal`



- Under `Settings > Rules`, click `waf-to-apim-internal`
  - Backend targets
    - Change `Target type` from `Backend pool` to `Redirection`
    - **Target listener**: `https`



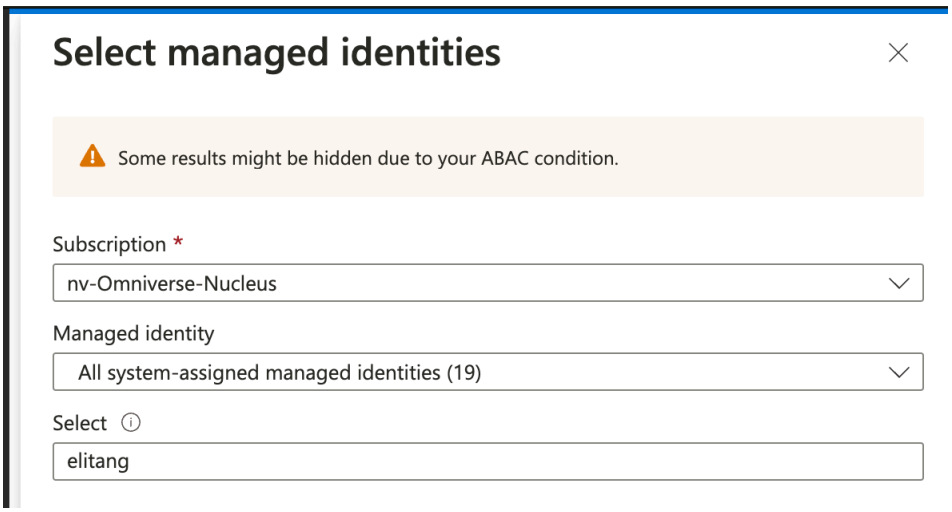
3.9.6 Post Deployment:

If your APIM service is not yet finished deploying with post deployment steps, you will see an error on the AppGW with something like "Unhealthy backend pools". Once AppGW is deployed, you should no longer have this error. Try testing the health probe if this problem continues and make sure that the URLs you added are configured correctly.

3.10 RBAC

1. Assign RBAC permissions to enterprise app registration created by AKS cluster

- Find the vnet created in step 1
- Under `Access control (IAM)`, select the `Network Contributor` role
    - Search for the cluster name under `Managed Identities` and add the managed identity of the AKS cluster



4 Kubernetes Services

4.1 Deploy base helm charts

Clone this repository https://gitlab-master.nvidia.com/omniverse/farm/infrastructure/azure/43083d15-7273-40c1-b7db-39efd9ccc17a/6db1f0c2-73fc-434c-9ebe-7fc0dbcc87a9/ovas-streaming-example#aks-cluster

Or, a more customized repo: https://gitlab-master.nvidia.com/wwfo-sae-omniverse/cloud-testing/aws/team-rdarbha/-/tree/azure

```
git clone https://gitlab-master.nvidia.com/wwfo-sae-omniverse/cloud-testing/aws/team-rdarbha.git
git checkout azure
```

If you do not have access to this, a zip is provided.

4.1.1 Set environment-specific values

- At a minimum, the following values need to be changed to suit your environment:
- Note: Instructions for this are specified in the following steps.
  - `helm/nginx-ingress-controller/values-internal.yaml`
    - `service.beta.kubernetes.io/azure-load-balancer-resource-group`[15]: name of resource group ex: `rg-elitang`
  - `helm/ov-ov-svc-applications/values.yaml`
    - `host`: This is an arbitrary internal hostname of your choosing used to access the swagger web interface for the application service
    - `streaming.serviceConfig.backend_csp_args.base_domain`: This is the public DNS name at which to access the APIs
  - `helm/ov-ov-svc-streaming/values.yaml`
    - `host`: This is an arbitrary internal hostname of your choosing used to access the swagger web interface for the streaming service

15 http://service.beta.kubernetes.io/azure-load-balancer-resource-group

4.1.2 Internal ingress controller (`helm/nginx-ingress-controller`)

Check values file; make sure resource group is correct in annotations. File located at `helm/nginx-ingress-controller/values-internal.yaml`

```
...
    service.beta.kubernetes.io/azure-load-balancer-resource-group: rg-elitang
...
```

```
helm repo add bitnami https://charts.bitnami.com/bitnami
helm repo update
helm upgrade -i nginx-ingress-controller-internal -n nginx-ingress-controller --
create-namespace -f helm/nginx-ingress-controller/values-internal.yaml bitnami/nginx-
ingress-controller
```

- Ensure the `Service` of type `LoadBalancer` is provisioned with a private external IP (i.e. does not say `<Pending>` ; check output of `kubectl get svc -A`)
    - This private IP should be within the range of the `subnet-aks` subnet! If it's not, double-check that the cluster was deployed within your own vnet and not a managed one (see AKS instructions above)

```
kubectl get svc -n nginx-ingress-controller
 NAME                                                TYPE
CLUSTER-IP    EXTERNAL-IP   PORT(S)                      AGE
 nginx-ingress-controller-internal                   LoadBalancer   10.0.3
4.20    10.2.0.120    80:30619/TCP,443:31307/TCP   23h
```

4.1.3 FluxCD (`helm/flux2`)

In your flux values file, change `value: system` to `value: <name of your agentpool>.`

You can find the labels by executing `kubectl get nodes --show-labels | grep agentpool` and looking for the label value.

Example:

```
elmController:
  create: true
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
```

```
        nodeSelectorTerms:
          - matchExpressions:
              - key: agentpool
                operator: In
                values:
                  - agentpool   <-- this should be <name of your agentpool>
  create: false
kustomizeController:
  create: false
notificationController:
  create: false
imageReflectionController:
  create: false
policies:
  create: false
rbac:
  create: true
sourceController:
  create: true
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
          - matchExpressions:
              - key: agentpool
                operator: In
                values:
                  - agentpool        <-- this should be <name of your agentpool>
```

```
helm repo add fluxcd-community https://fluxcd-community.github.io/helm-charts
 helm repo update
 helm upgrade --install fluxcd -n omni-system --create-namespace -f helm/flux2/
values.yaml fluxcd-community/flux2
```

4.1.4 GPU Operator (`helm/gpu-operator`)

```
helm repo add nvidia https://helm.ngc.nvidia.com/nvidia
 helm repo update
 helm upgrade -i gpu-operator -n gpu-operator --create-namespace -f helm/gpu-
operator/values.yaml nvidia/gpu-operator
```

4.1.5 Memcached (helm/memcached)

```
 helm upgrade -i memcached-service-r3 -n omni-streaming --create-namespace bitnami/
memcached --version 7.0.2 -f helm/memcached/values.yml
```

4.1.6 ExternalDNS (scripts/external-dns)

Create a service principal and assign the correct roles via the first script. Edit the `scripts/external-dns/01-create-sp-for-rbac.sh` file with your own:

```
SUBSCRIPTION_ID="YOUR_SUBSCRIPTION_ID"
EXTERNALDNS_NEW_SP_NAME="NvSbxExternalDnsServicePrincipal" # name of the service
principal (This should be unique)
AZURE_DNS_ZONE_RESOURCE_GROUP="fidot_sandbox" # name of resource group where dns zone
is hosted
AZURE_DNS_ZONE="nvsbx.omniverse.nvidia.com" # DNS zone name like example.com or
sub.example.com
```

Execute

```
./scripts/external-dns/01-create-sp-for-rbac.sh
```

Example output:

```
elitang@elitang-mlt ovas-streaming-example % ./scripts/external-dns/01-create-sp-for-
rbac.sh
WARNING: The output includes credentials that you must protect. Be sure that you do
 not include these credentials in your code or check the credentials into your source
control. For more information, see https://aka.ms/azadsp-cli
Client ID: <CLIENT ID HERE>
Client secret: <CLIENT SECRET HERE>
...
...
Copy azure.json.template to azure.json and add the above client ID, secret, resource
group and subscription ID
```

Create `azure.json` file with new credentials (note resource group may differ from your cluster RG):

```
 {
   "tenantId": "<Your-tentent-ID>",
   "subscriptionId": "<your-subscription-id>",
```

```
    "resourceGroup": "<dns-zone-rg>",
    "aadClientId": "<client-id>",
    "aadClientSecret": "<client-secret>"
  }
```

Apply

```
kubectl create secret generic azure-config-file --namespace "default" --from-file ./
scripts/external-dns/azure.json
```

Edit `scripts/external-dns/03-external-dns-manifest.yaml` edit appropriate values for `--domain-filter` and `--azure-resource-group`

```
...
   spec:
      serviceAccountName: external-dns
      containers:
        - name: external-dns
          image: registry.k8s.io/external-dns/external-dns:v0.14.2
          args:
            - --source=service
            - --source=ingress
            - --domain-filter=nvsbx.omniverse.nvidia.com <-- your Public domain here
            - --provider=azure
            - --azure-resource-group=<dns resource group>
            - --txt-prefix=externaldns-
          volumeMounts:
            - name: azure-config-file
              mountPath: /etc/kubernetes
              readOnly: true
...
```

Apply the external-dns manifest

```
kubectl apply -f scripts/external-dns/03-external-dns-manifest.yaml
```

4.1.7 Create required secrets

Get your NGC API token (you must have access to NGC's `nvidian/omniverse` org)

`export NGC_API_TOKEN=<your-token>`

Create the `regcred` secret

```
kubectl create secret -n omni-streaming docker-registry regcred \
--docker-server=nvcr.io \
```

```
 --docker-username='$oauthtoken' \
 --docker-password=$NGC_API_TOKEN \
 --dry-run=client -o json | \
 kubectl apply -f -
```

Create the ngc-omni-user secret

```
kubectl create secret generic -n omni-streaming ngc-omni-user \
--from-literal=username='$oauthtoken' \
--from-literal=password="$NGC_API_TOKEN"
```

Add the required NVIDIA helm repositories

```
helm repo add omniverse https://helm.ngc.nvidia.com/nvidia/omniverse --
username='$oauthtoken' --password=$NGC_API_TOKEN
helm repo update
```

4.2 Deploy OVAS services

4.2.1 Streaming (helm/kit-appstreaming-manager)

Check values file and update DNS names accordingly

```
...
ingress:
    host: api.elitang-ovas-streaming.net.     <--- your private domain here ex:
api.<your-private-domain>
    className: internal-nginx
...
```

To enable/disable WSS follow these directions

```
...
     backend_csp_cls: "nv.svc.streaming._csp.Generic"
    backend_csp_args:
      enable_wss: true    <----- set to true to enable WSS. set to false to
disable WSS
```

```
      base_domain: "ovas-streaming.nvsbx.omniverse.nvidia.com" <--- add
your public domain name here (Leave blank if disabling WSS)
...
```

```
helm upgrade --install \
  --namespace omni-streaming \
  -f helm/kit-appstreaming-manager/values.yaml --version 1.0.0-ea \
  streaming omniverse/kit-appstreaming-manager
```

4.2.2 Applications (helm/kit-appstreaming-applications)

Check values file and update DNS names accordingly

```
...
  ingress:
    host: api.elitang-ovas-streaming.net   <--- your private domain here ex:
api.<your-private-domain>
 ...
```

```
helm upgrade --install \
  --namespace omni-streaming \
  -f helm/kit-appstreaming-applications/values.yaml --version 1.0.0-ea \
  applications omniverse/kit-appstreaming-applications
```

4.2.3 RMCP (helm/kit-appstreaming-rmcp)

Check values file and update DNS names accordingly

Change or comment out affinity from `system` to `agentpool`

```
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
          - matchExpressions:
              - key: agentpool
                operator: In
                values:
                  - agentpool <-- this should be <name of your agentpool>
```

```
helm upgrade --install \
  --namespace omni-streaming \
  -f helm/kit-appstreaming-rmcp/values.yaml --version 1.0.0-ea \
 rmcp omniverse/kit-appstreaming-rmcp
```

4.2.4 Deploy the custom streaming resources (manifests/omniverse/azure)

1. To enable WSS: If there is already a file at `manifests/omniverse/azure/application-profile-wss.yaml`, there is no need to create the file. If there is not, create it.
2. Then run `kubectl apply -n omni-streaming -f application-profile-wss.yaml`

```
apiVersion: omniverse.nvidia.com/v1
kind: ApplicationProfile
metadata:
  name: azurelb
spec:
  name: AzureLB example profile
  description: Default profile - uses an AzureLB per stream
  supportedApplications:
    - name: usd-viewer
      versions:
        - '*'
  chartMappings:
    container: streamingKit.image.repository
    container_version: streamingKit.image.tag
    name: streamingKit.name
  chartValues:
    global:
      imagePullSecrets:
        - name: regcred
    streamingKit:
      envoy:
        enabled: true
        secretRef: stream-tls-secret
        config: |
          static_resources:
            listeners:
            - name: webrtc_signaling_listener
              address:
```

```
                socket_address:
                  address: 0.0.0.0
                  port_value: 49200
              filter_chains:
              - transport_socket:
                  name: envoy.transport_sockets.tls
                  typed_config:
                    "@type": type.googleapis.com/
envoy.extensions.transport_sockets.tls.v3.DownstreamTlsContext
                    common_tls_context:
                      tls_certificates:
                        - certificate_chain: { filename: "/etc/envoy/tls/
tls.crt" }
                          private_key: { filename: "/etc/envoy/tls/tls.key"
}
                filters:
                - name: envoy.filters.network.tcp_proxy
                  typed_config:
                    "@type": type.googleapis.com/
envoy.extensions.filters.network.tcp_proxy.v3.TcpProxy
                    stat_prefix: tcp
                    cluster: service_cluster
                    access_log:
                      - name: envoy.access_loggers.stream
                        typed_config:
                          "@type": type.googleapis.com/
envoy.extensions.access_loggers.stream.v3.StdoutAccessLog
                          log_format:
                            text_format: "[%START_TIME%] \"%PROTOCOL%\"
connection from %DOWNSTREAM_REMOTE_ADDRESS% to %UPSTREAM_HOST%\n"
            - name: health_listener
              address:
                socket_address:
                  address: 0.0.0.0
                  port_value: 8080
              filter_chains:
              - filters:
                - name: envoy.filters.network.http_connection_manager
                  typed_config:
                    "@type": type.googleapis.com/
envoy.extensions.filters.network.http_connection_manager.v3.HttpConnectionM
anager
                    stat_prefix: health_check
                    codec_type: AUTO
                    route_config:
                      name: local_route
                      virtual_hosts:
                      - name: local_service
                        domains: ["*"]
                        routes:
                        - match:
```

```
                            prefix: "/health"
                          direct_response:
                            status: 200
                            body:
                              inline_string: "OK"
                    http_filters:
                    - name: envoy.filters.http.router
            clusters:
            - name: service_cluster
              connect_timeout: 0.25s
              type: STATIC
              lb_policy: ROUND_ROBIN
              load_assignment:
                cluster_name: service_cluster
                endpoints:
                - lb_endpoints:
                  - endpoint:
                      address:
                        socket_address:
                          address: 127.0.0.1
                          port_value: 49100  # Forwarding to the stream
        tls:
          enabled: true
          secretRef: stream-tls-secret
      image:
        repository: nvcr.io/omniverse/prerel/usd-viewer
        pullPolicy: Always
        tag: 0.2.0
      sessionId: session_id
      service:
        signalingPort: 31000
        mediaPort: 31001
        healthPort: 31002
        annotations:
          # NLB configuration
          service.beta.kubernetes.io/azure-load-balancer-internal: "false"
          external-dns.alpha.kubernetes.io/ttl: "5"
          # Security stuff - may need to specify this in
`loadBalancerSourceRanges`
          # service.beta.kubernetes.io/load-balancer-source-ranges: 216.228.1
25.128/30,216.228.127.128/30,206.223.160.26/32,216.228.125.131/32,216.228.1
25.128/30,216.228.127.128/30
          # Health check
          service.beta.kubernetes.io/port_8080_health-probe_protocol: HTTP
          service.beta.kubernetes.io/port_8080_health-probe_port: "8080"
          service.beta.kubernetes.io/port_8080_health-probe_request-path: /
health
        type: LoadBalancer
      name: kit-app
      resources:
        limits:
```

```
            cpu: "3"
            memory: 20Gi
            nvidia.com/gpu: "1"
          requests:
            nvidia.com/gpu: "1"
        env:
          - name: USD_PATH
            value: /app/data/Forklift_A/Forklift_A01_PR_V_NVD_01.usd
```

To disable WSS, make the following changes in `manifests/omniverse/azure/application-profile-azurelb.yaml`

```
      envoy:
        tls:
          enabled: true <---- set this to false
          secretRef: stream-tls-secret
```

Then run

```
kubectl apply -n omni-streaming -f application-profile-azurelb.yaml
```

```
kubectl apply -n omni-streaming -f application.yaml
kubectl apply -n omni-streaming -f application-version.yaml
```

4.2.5 Deploy HelmRepository (manifests/helm-repostiories)

```
kubectl apply -n omni-streaming -f manifests/helm-repositories/ngc-omniverse.yaml
```

This should (eventually) show READY: True in the output of `kubectl get helmrepositories -n omni-streaming:`

```
> kubectl get helmrepositories -n omni-streaming

NAMESPACE        NAME            URL                                          AGE
READY   STATUS
omni-streaming   ngc-omniverse   https://helm.ngc.nvidia.com/nvidia/omniverse   13m
True    stored artifact: revision
'sha256:5766710ed3052d8a247ea8aeb4d093bc9395bd82afbfaedaa41546af2a22d850'
```

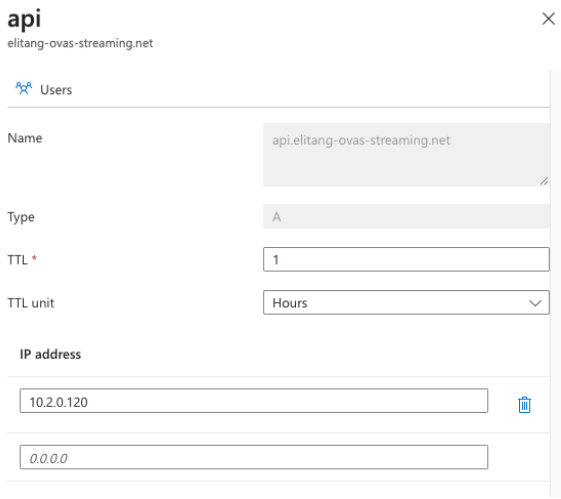
4.2.6 Create private DNS record for ingress controller

Go to the Private DNS Zone you created. Create the following recordset:

`api.elitang-ovas-streaming.net`[16] -> `private external ip of ingress controller LB service` (e.g. `10.2.0.120` shown below)

```
elitang@elitang-mlt ovas-streaming-example % kubectl get svc -n nginx-ingress-
controller
NAME                                                TYPE           CLUSTER-IP
EXTERNAL-IP    PORT(S)                      AGE
nginx-ingress-controller-internal                   LoadBalancer   10.0.100.71
10.2.0.120     80:32710/TCP,443:31053/TCP   13h
nginx-ingress-controller-internal-default-backend   ClusterIP      10.0.166.232
<none>         80/TCP                       13h
elitang@elitang-mlt ovas-streaming-example %
```

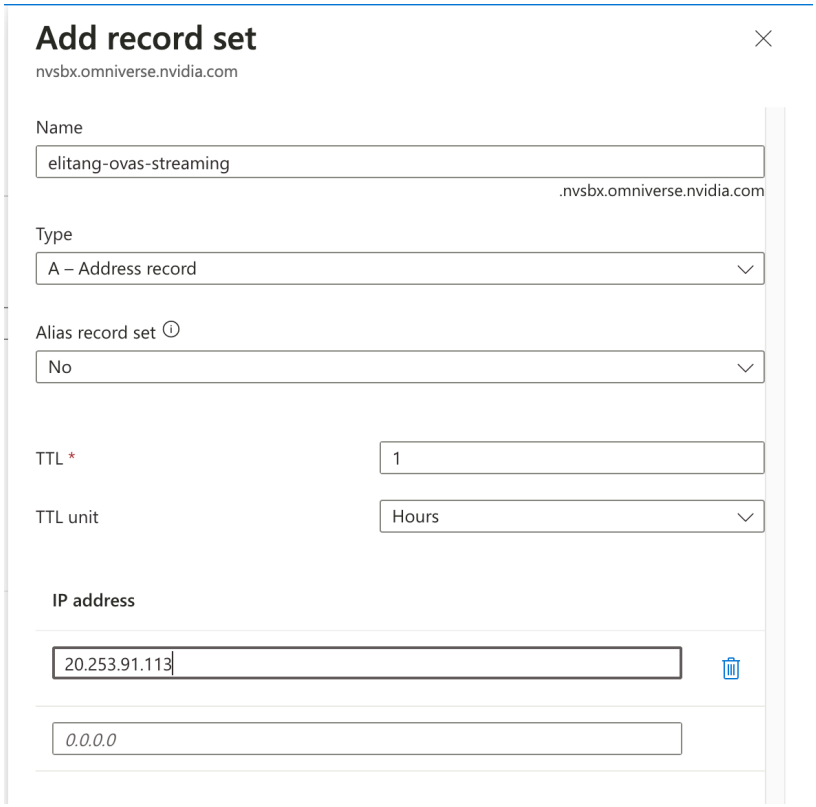
Example of recordset to create:



4.2.7 Create public DNS entry for App Gateway

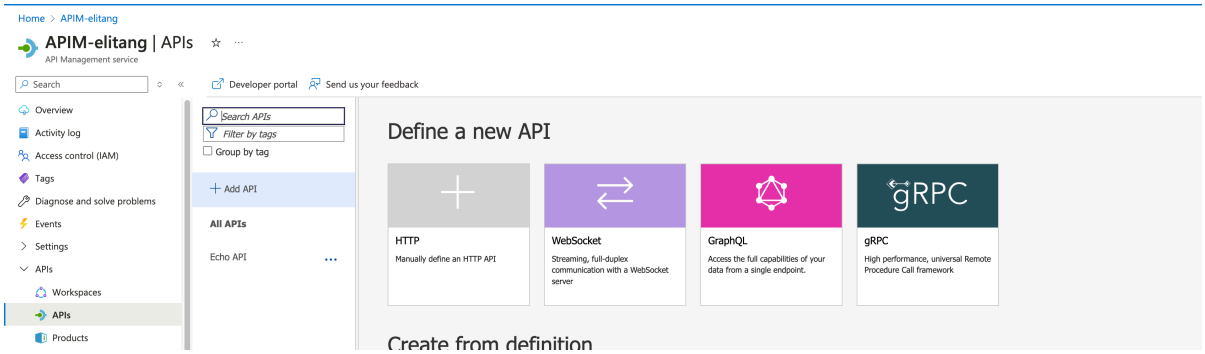
- Navigate to the Public DNS Zone
- Create the following RecordSet
- `elitang-ovas-streaming.nvsbx.omniverse.nvidia.com`
- Points to IP address of APIGW public IP
    - To find this navigate to the APIGW
- Should match the value of `streaming.serviceConfig.backend_csp_args.base_domain` in the streaming service specified previously.

16 http://api.ovas-streaming.net

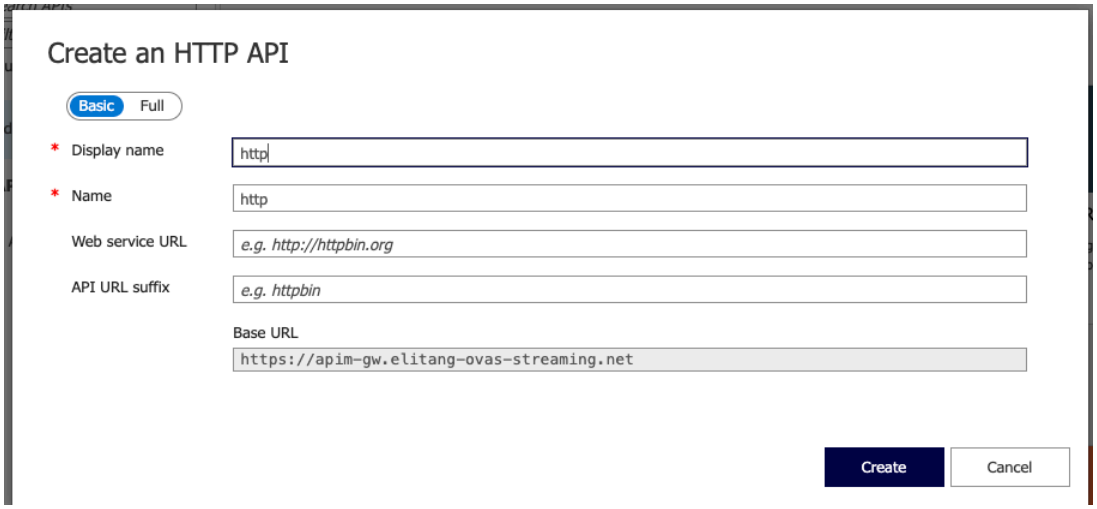


4.2.8 Add API route to ingress controller in APIM

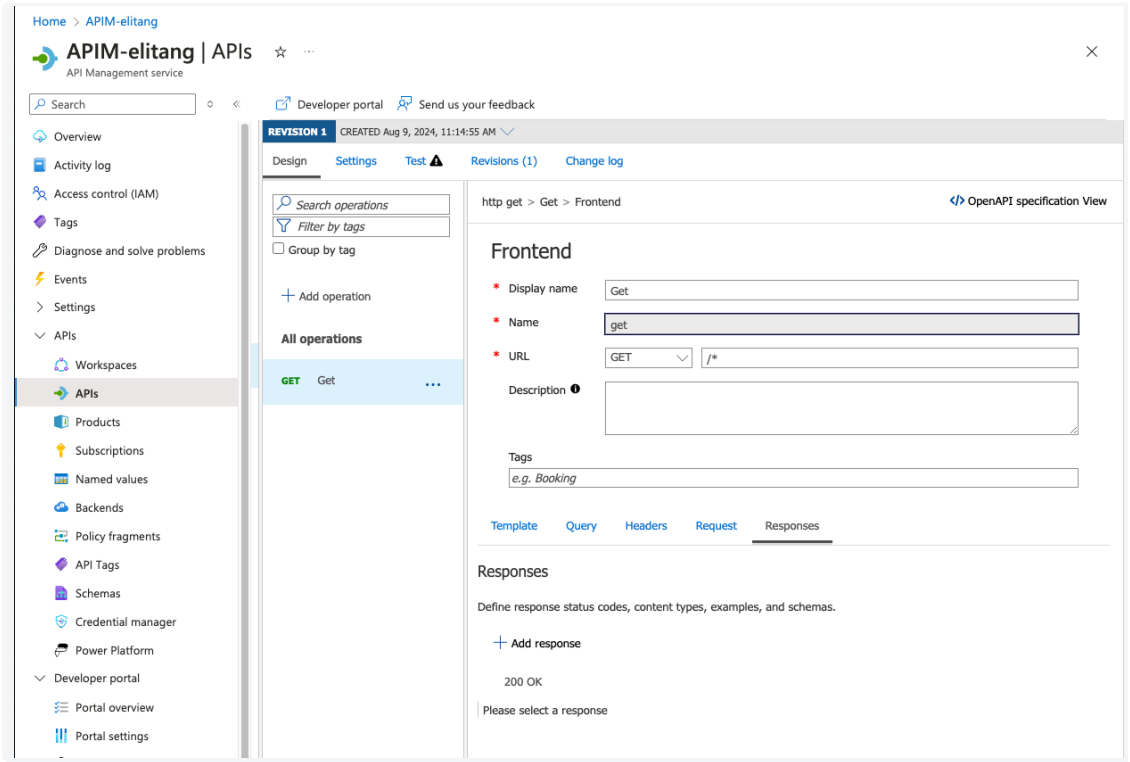
Navigate to APIs in the APIM



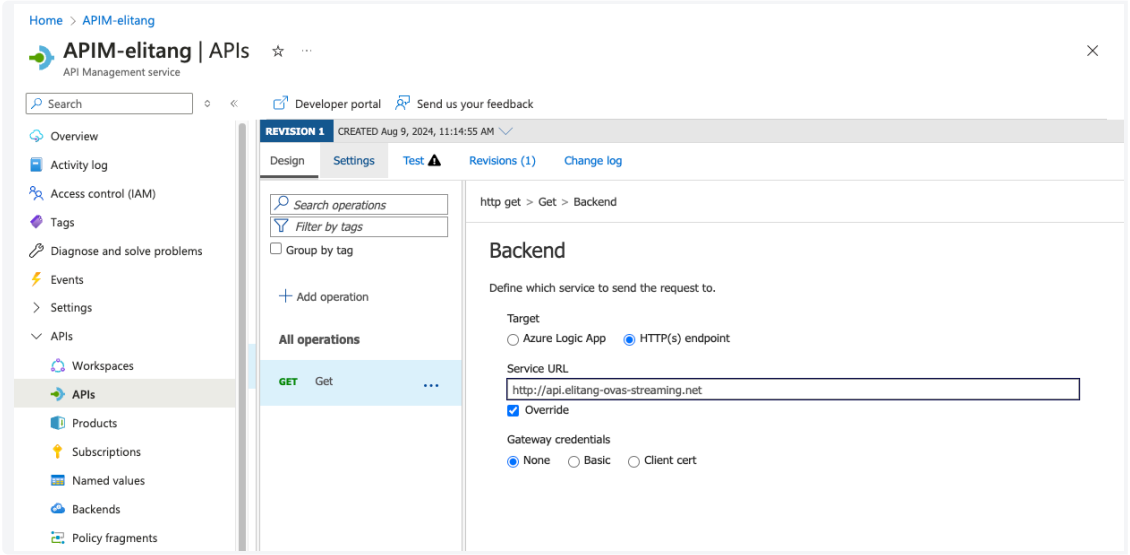
- Create a new `HTTP` API

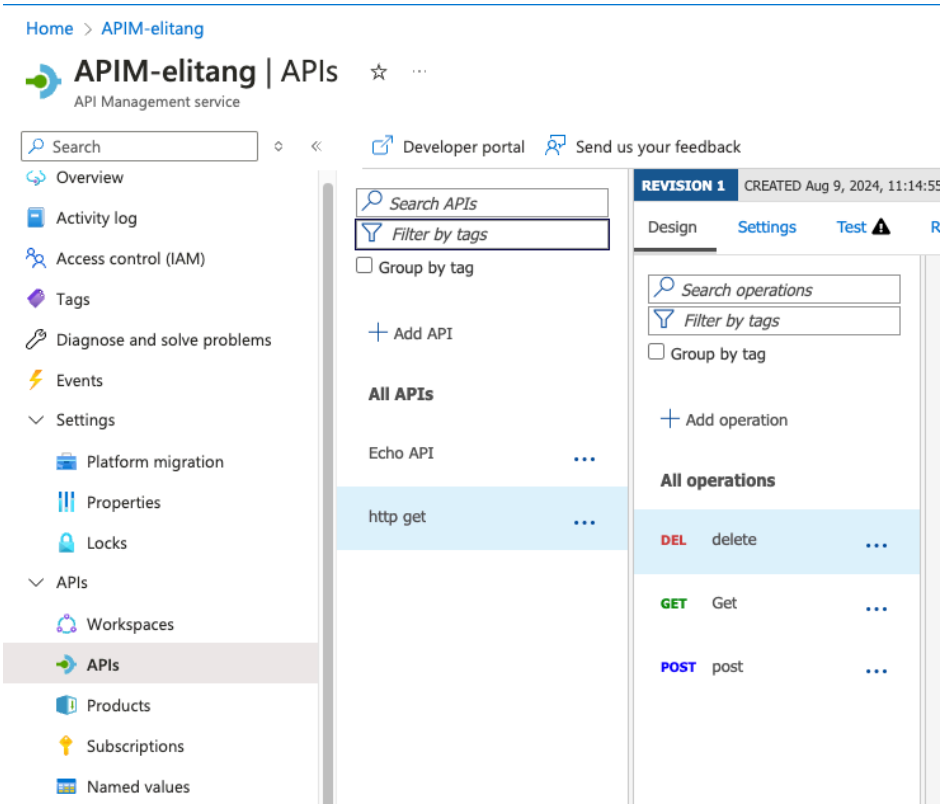


- Create new `GET` operation
  - **URL**: `/*`
  - Under `Responses` tab, add `200`



  - Under `Backend > HTTP(s) endpoint`, add the DNS name given to the streaming services, `api.elitang-ovas-streaming.net`
    - Check the `Override` box next to `Service URL`
    - Select `HTTP(s) endpoint` instead of `Azure logic app`

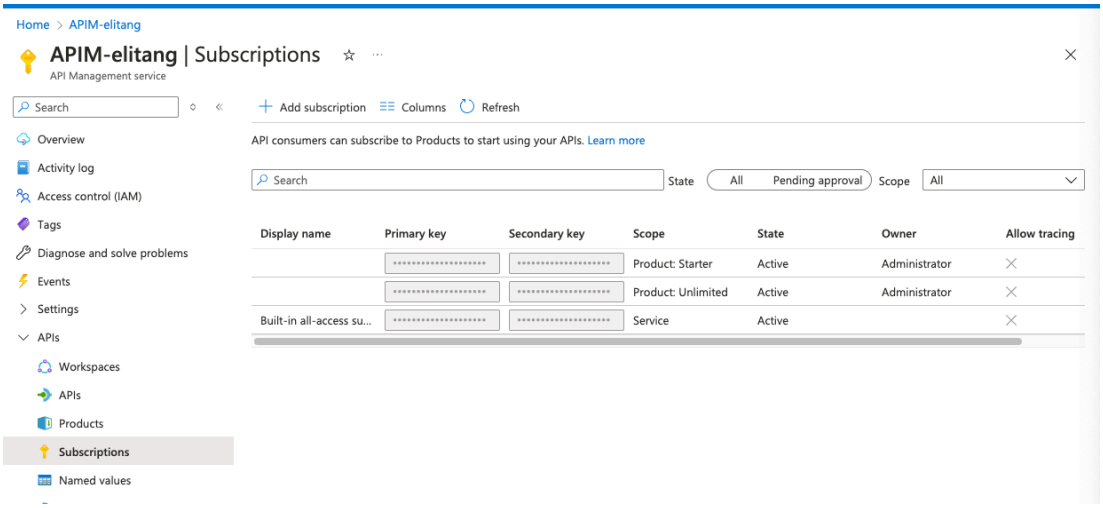
- 

- Repeat this for POST and DELTE operations. Result below:
- 

5 Testing

5.1 Testing From CLI

- From CLI with subscription key required:
  - To get subscription key, go to APIM and navigate to APIs > Subscriptions



  - Right click on the row that has `Built in all-access subscription...`
  - `Click Show/hide keys`
  - Copy primary key

```
curl -H 'Ocp-Apim-Subscription-Key: <your-key>' https://appgw.ovas-streaming.nvsbx.omniverse.nvidia.com/streaming/docs``
curl -H 'Ocp-Apim-Subscription-Key: <your-key>' https://appgw.ovas-streaming.nvsbx.omniverse.nvidia.com/applications/docs
```

  - Example Output:

```
elitang@elitang-mlt ~ % curl -H 'Ocp-Apim-Subscription-Key: <KEY>'
 https://elitang-ovas-streaming.nvsbx.omniverse.nvidia.com/streaming/docs

    <!DOCTYPE html>
    <html>
    <head>
    <link type="text/css" rel="stylesheet" href="https://cdn.jsdelivr.net/npm/swagger-ui-dist@5.9.0/swagger-ui.css">
```

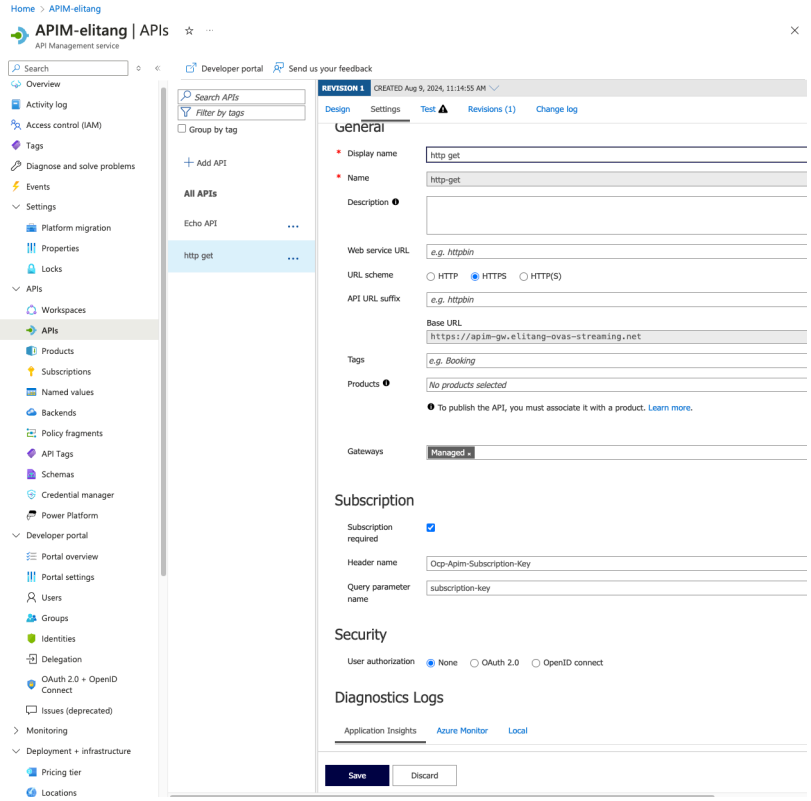
```
    <link rel="shortcut icon" href="https://fastapi.tiangolo.com/img/
favicon.png">
    <title>Application Streaming API - Swagger UI</title>
    </head>
    <body>
    <div id="swagger-ui">
    </div>
    <script src="https://cdn.jsdelivr.net/npm/swagger-ui-dist@5.9.0/
swagger-ui-bundle.js"></script>
    <!-- `SwaggerUIBundle` is now available on the page -->
    <script>
    const ui = SwaggerUIBundle({
        url: '/streaming/openapi.json',
    "dom_id": "#swagger-ui",
"layout": "BaseLayout",
"deepLinking": true,
"showExtensions": true,
"showCommonExtensions": true,
oauth2RedirectUrl: window.location.origin + '/streaming/docs/oauth2-
redirect',
    presets: [
        SwaggerUIBundle.presets.apis,
        SwaggerUIBundle.SwaggerUIStandalonePreset
        ],
    })
    </script>
    </body>
    </html>
    %
elitang@elitang-mlt ~ %
```

5.2 Testing with swagger docs

5.2.1 Disabling Subscription Key

From browser (requires disabling subscription key without a service in front to acquire one): navigate to above URLs and confirm you can get to swagger docs, run commands, etc.

Go to APIM, APIs > APIs > http get > settings. Uncheck the box that says `subscription required`



Swagger docs for Streaming and Application can be accessed here:

Edit with your public domain.

https://elitang-ovas-streaming.nvsbx.omniverse.nvidia.com/streaming/docs[17]

https://elitang-ovas-streaming.nvsbx.omniverse.nvidia.com/application/docs[18]

5.2.2 Creating a stream

Go to streaming swagger, ex: https://elitang-ovas-streaming.nvsbx.omniverse.nvidia.com/streaming/docs[19]

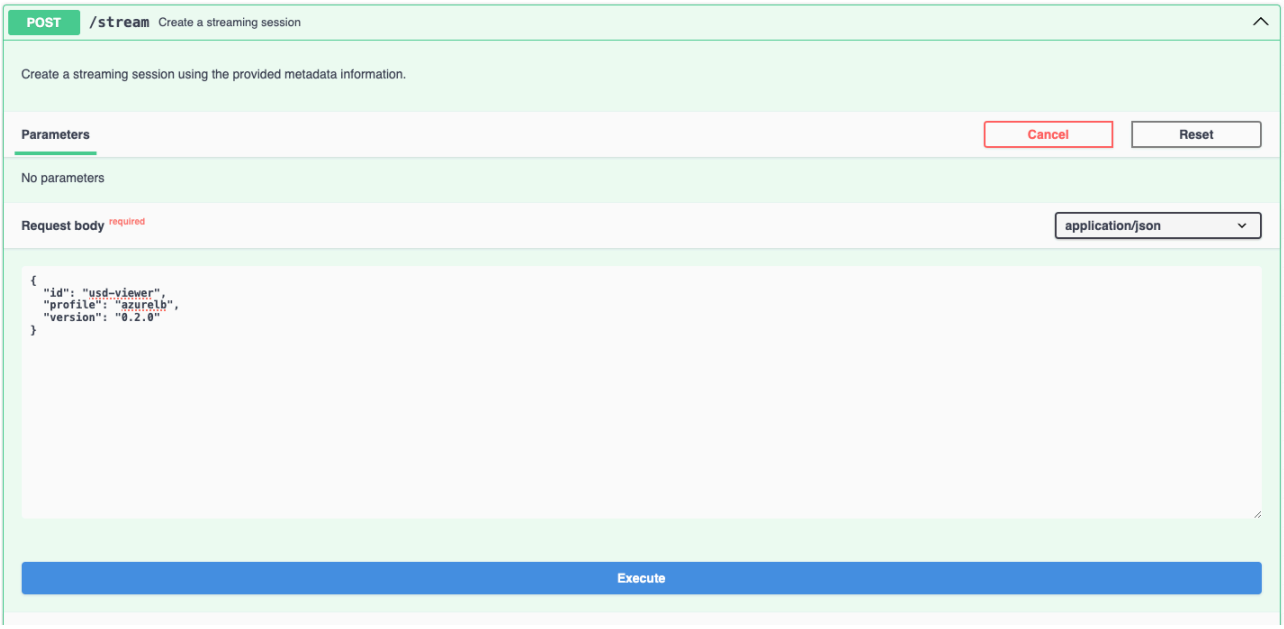
Click the POST /Stream

Click Post, then click `"Try it out"`

Enter the following as the request body:

```
{
  "id": "usd-viewer",
  "profile": "azurelb",
  "version": "0.2.0"
}
```

17 https://elitang-ovas-streaming.nvsbx.omniverse.nvidia.com/streaming/docs#/Streaming%20APIs/start_stream_stream_post
18 https://elitang-ovas-streaming.nvsbx.omniverse.nvidia.com/streaming/docs#/Streaming%20APIs/start_stream_stream_post
19 https://elitang-ovas-streaming.nvsbx.omniverse.nvidia.com/streaming/docs#/Streaming%20APIs/start_stream_stream_post

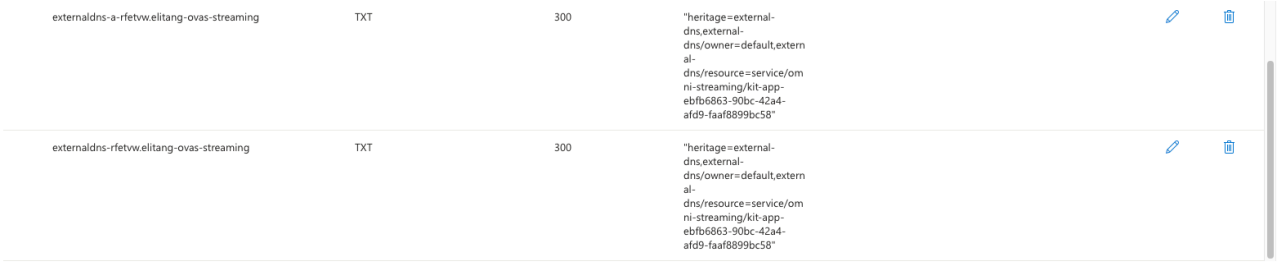


Once executed, you should be able to see a pod being created with `kubectl get pods -A`

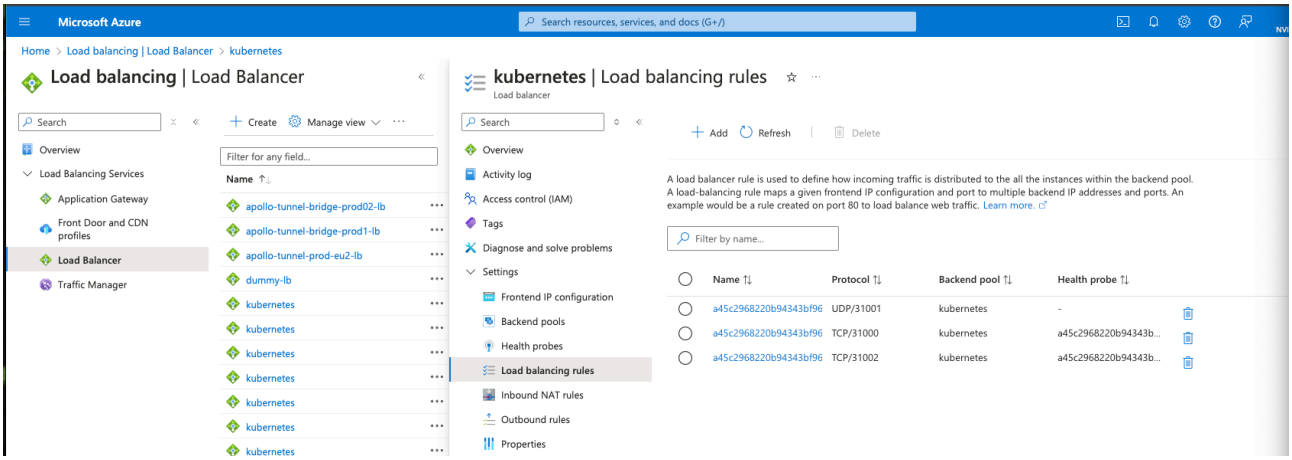
Additionally, you can use the Get /stream operation on the swagger page and see that the stream is created.

5.2.3 Verification

Checking that external DNS created entries in Public DNS Zone



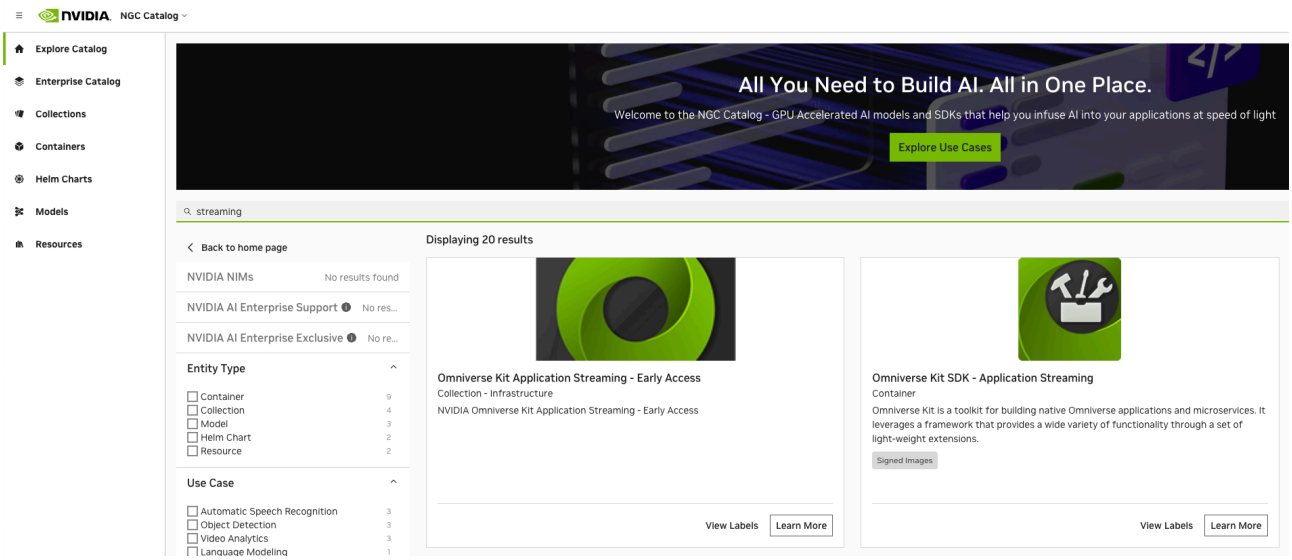
You can also check that a new rule is created on your load balancer.



5.3 Testing with local client app (Locally installed certificates)

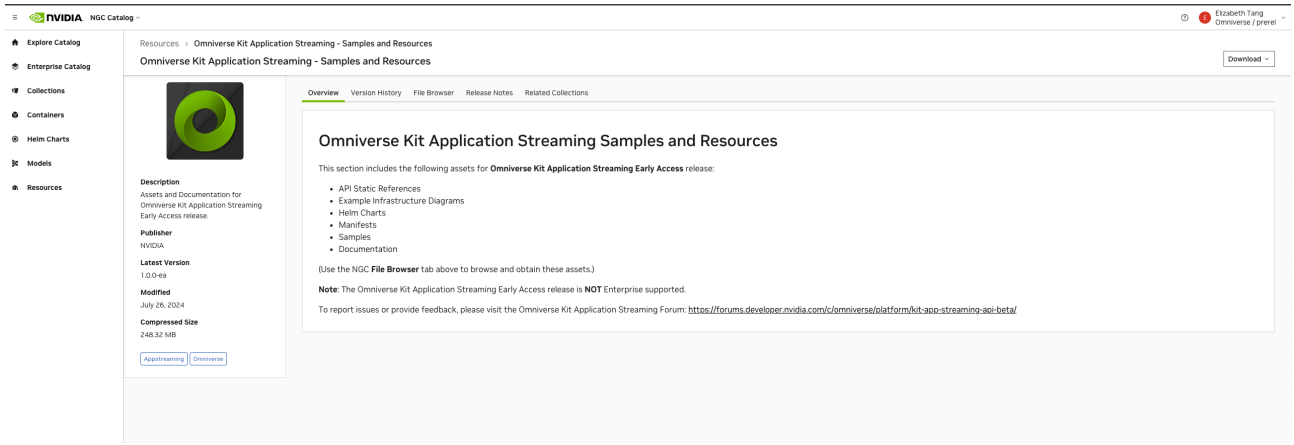
Make sure the the subscription key is disabled in the step above.

Go to NGC and search for streaming



Click on Omniverse Kit Application Streaming - Early Access

Scroll to the bottom and click on Resources



Click File Browser and click `usd-viewer-streaming-sample.zip` to download this zip. Unzip this.

5.3.1 Client Application On Local Machine

Edit stream.config.json with the following file and change endpoints for streaming and applications.

Make sure these are HTTPS endpoints

"appServer": "https://elitang-ovas-streaming.nvsbx.omniverse.nvidia.com/applications[20]", "streamServer": "https://elitang-ovas-streaming.nvsbx.omniverse.nvidia.com/streaming[21]"

```
{
    "usd_stage_uri": "/app/data/Forklift_A/Forklift_A01_PR_V_NVD_01.usd",
    "defaultPrimPath": "/World/Geometry",
    "$comment": "source can be either 'stream', 'gfn' or 'local'",
    "source": "stream",

    "stream": {
        "$comment": "Required props if source is set to 'stream'.",
        "applicationId": "usd-viewer",
        "applicationVersion": "0.2.0",
        "appServer": "https://elitang-ovas-streaming.nvsbx.omniverse.nvidia.com/
applications",
        "streamServer": "https://elitang-ovas-streaming.nvsbx.omniverse.nvidia.com/
streaming"
    },
    "gfn": {
        "$comment": "Required props if source is set to 'gfn'.",
        "catalogClientId": "",
        "clientId": "",
        "cmsId": 0
    },
    "local": {
```

20 http://elitang-ovas-streaming.nvsbx.omniverse.nvidia.com/applications
21 http://elitang-ovas-streaming.nvsbx.omniverse.nvidia.com/streaming

```
        "$comment": "Required props if source is set to 'local'.",
        "server": "127.0.0.1"
    }
}
```

5.3.1.1 Install NPM

Install NPM and specify version (running `apt install npm` may install a version too old or too new)

```
curl -sL https://deb.nodesource.com/setup_18.x[22] -o nodesource_setup.sh
sudo bash nodesource_setup.sh
sudo apt install nodejs
```

Install npm (this may take some time)

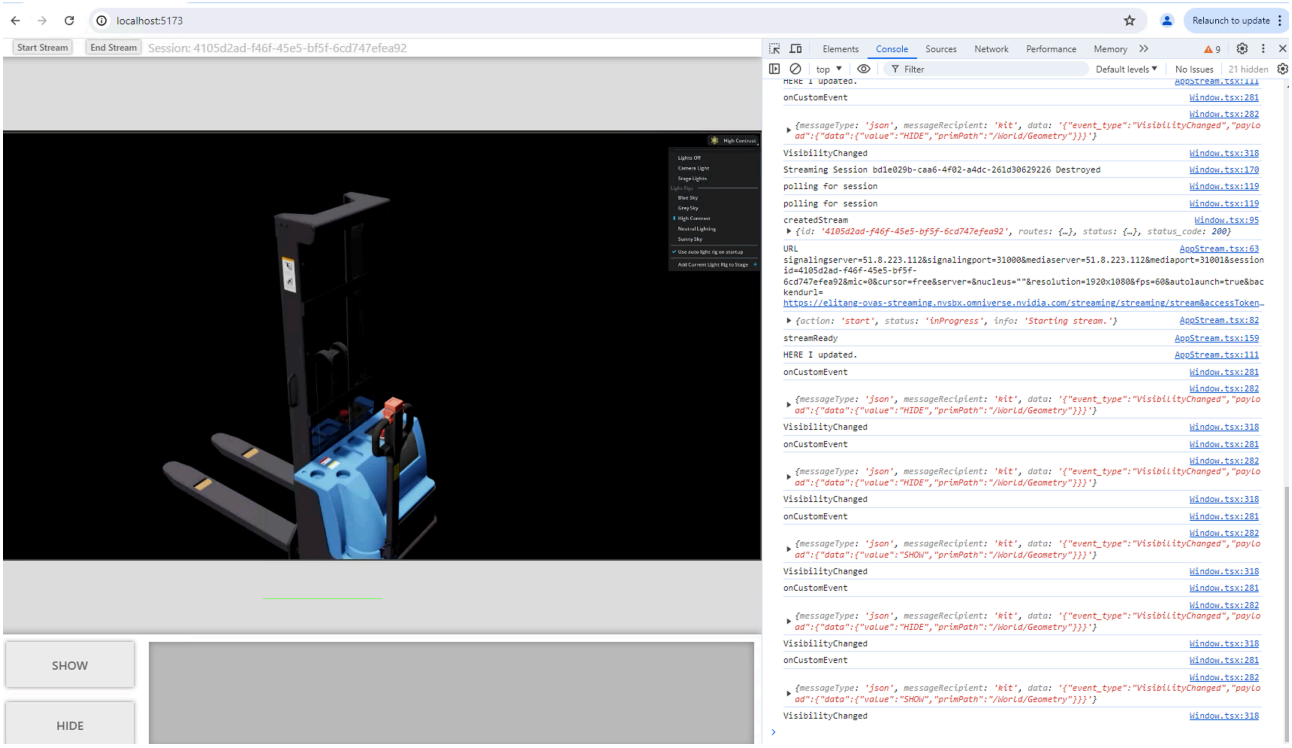
```
npm install
```

```
npm run dev --host
```

If WSS is not enabled, navigate the the URL provided. You should see a forklift load. This may take some time the first stream, as objects need to be loaded into the cache. Wait about 10 minutes. If the stream fails, try once more. Sometimes the stream times out before the objects are loaded. Once you see the forklift, OVAS is fully validated.

If WSS is enabled, the stream will not start, follow steps below.

22 http://deb.nodesource.com/setup_18.x



5.3.2 Enabling WSS:

Download and install Mkcert: https://github.com/FiloSottile/mkcert

Run the command

```
mkcert -install
```

Create a self signed certificate for the base domain specified in your `nv-ov-svc-streaming/values.yaml`

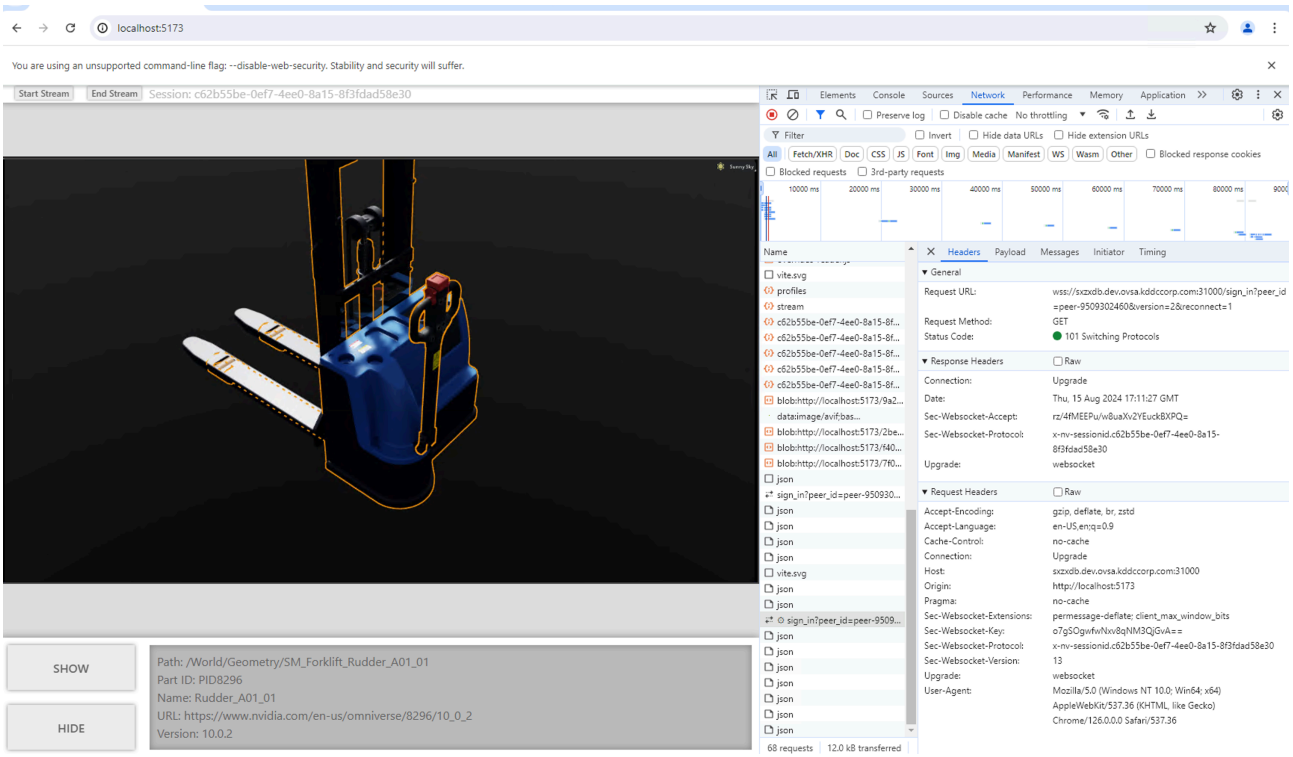
Locate the domain under base_domain. Example:

```
...
    backend_csp_cls: "nv.svc.streaming._csp.Generic"
    backend_csp_args:
      enable_wss: true
      base_domain: "dev.ovsa.kddccorp.com"
...
```

On your local machine, execute this command

```
mkcert *.dev.ovsa.kddccorp.com
```

Now you can access the stream at http://localhost:5173/. You should see in the networking tab, a WSS connection established. Once you see the forklift, OVAS is fully validated with end to end encryption.



6 Client App in Azure

Clone the branch `feat/appstream_noauth_azure_configmap` :https://gitlab-master.nvidia.com/omniverse/samples/web/usd-viewer-streaming-sample/-/tree/feat/appstream_noauth_azure_configmap?ref_type=heads

6.1 Creating a NGINX Ingress Controller

Create a file called `nginx-ingress-external.yaml`

Add the following code to the file:

```
nodeAffinityPreset:
  type: hard
  key: kubernetes.azure.com/mode
  values:
  - system

config:
  proxy-body-size: 0m
  proxy-request-buffering: "off"
  ssl-ciphers: HIGH:!aNULL:!eNULL:!EXPORT:!DES:!RC4:!MD5:!PSK:@SECLEVEL=1
  ssl-protocols: TLSv1.2 TLSv1.3
  use-http2: "false"
metrics:
  enabled: false
  serviceMonitor:
    enabled: false
replicaCount: 2
service:
  annotations:
    service.beta.kubernetes.io/azure-load-balancer-health-probe-request-path: /
healthz
    service.beta.kubernetes.io/azure-load-balancer-resource-group: rg-elitang
  externalTrafficPolicy: Local
```

Apply this file with:

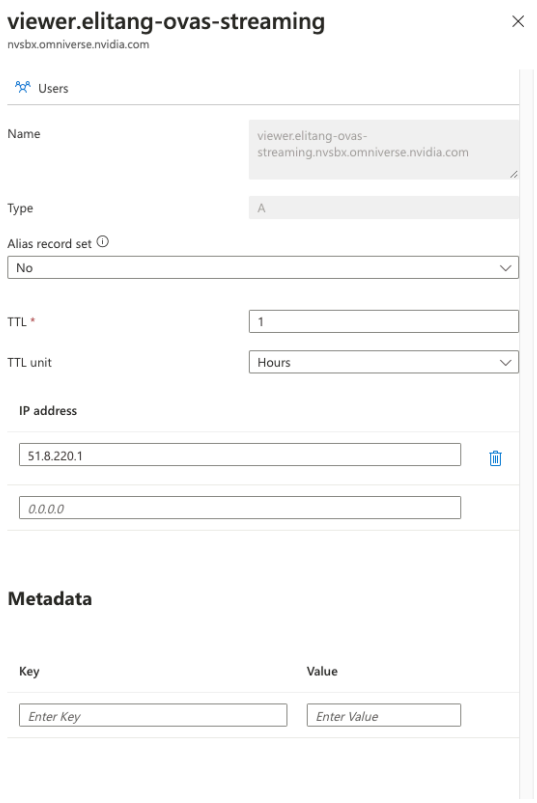
```
 helm upgrade -i nginx-ingress-controller -n nginx-ingress-controller --create-
namespace -f nginx-ingress-external.yaml bitnami/nginx-ingress-controller
```

This will create a new nginx ingress controller, with an exposed Public IP. This may take a few minutes to come up. Find this IP with:

```
elitang@elitang-mlt usd-viewer-streaming-sample 2 % kubectl get svc -n nginx-ingress-
controller
NAME                                                 TYPE           CLUSTER-IP
EXTERNAL-IP   PORT(S)                      AGE
nginx-ingress-controller                             LoadBalancer   10.0.132.23
51.8.220.1    80:30761/TCP,443:31981/TCP   3d18h   <----- Copy the external IP
nginx-ingress-controller-default-backend             ClusterIP      10.0.180.252
<none>        80/TCP                       3d18h
nginx-ingress-controller-internal                    LoadBalancer   10.0.71.172
10.2.0.149    80:32690/TCP,443:32315/TCP   12d
nginx-ingress-controller-internal-default-backend    ClusterIP      10.0.163.197
<none>        80/TCP                       12d
```

6.2 Adding Entry to Public DNS Zone

Create a recordset with a new subdomain prefix ex: `viewer.elitang-ovas-streaming`



Edit the file: `helm/streaming-sample/internal/azure/dev-b/values.yaml`

```
---
```

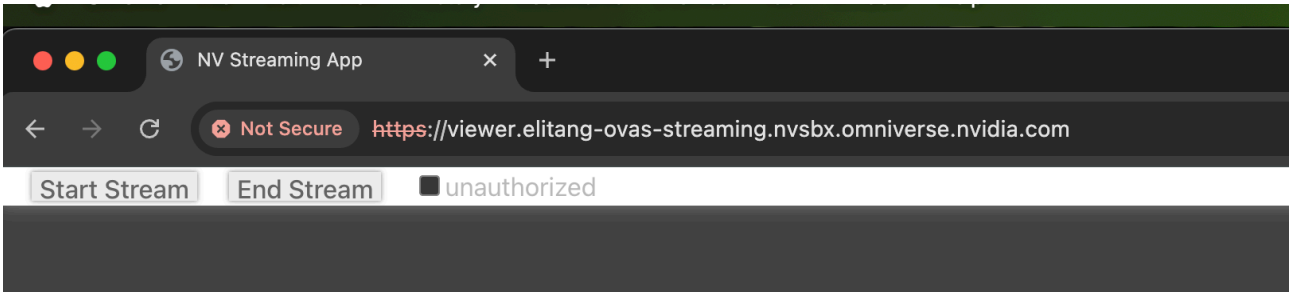
```
image:
  repository: nvcr.io/nvidian/omniverse/streaming-sample-viewer           #<----
point to your client app image
  pullPolicy: Always
  # Overrides the image tag whose default is the chart appVersion.
  tag: 1.0.0-beta.3-azure-configmap
imagePullSecrets:
  - name: regcred                                                         #<---- In
this tutorial regcred is in the namespace omni-streaming. Therefore to access this
 secret, this chart must be deployed with -n omni-streaming

serviceConfig:
  appServerUri: https://elitang-ovas-streaming.nvsbx.omniverse.nvidia.com/
applications           #<---- add your endpoints, these will be added to the
application with a configmap
  streamServerUri: https://elitang-ovas-streaming.nvsbx.omniverse.nvidia.com/
streaming

ingress:
  enabled: true
  className: nginx
  tls:
    - hosts:
        - viewer.elitang-ovas-streaming.nvsbx.omniverse.nvidia.com
      #<---- The public DNS record you created above should match this
      secretName: stream-tls-secret
  hosts:
    - host: viewer.elitang-ovas-streaming.nvsbx.omniverse.nvidia.com
      paths:
        - path: /
          pathType: Prefix
  annotations:
    # Route53 DNS name
    external-dns.alpha.kubernetes.io/ingress-hostname-source: annotation-only
    #external-dns.alpha.kubernetes.io/hostname: viewer.dev-
b.nvsbx.omniverse.nvidia.com
    # Certificate and TSL redirection
    kubernetes.io/ingress.class: nginx
```

Apply this with:

```
helm upgrade -i viewer-sample -f helm/streaming-sample/internal/azure/dev-b/
values.yaml helm/streaming-sample -n omni-streaming
```

If you see that both pods are in a RUNNING state, go to the domain you created. Ex: viewer.elitang-ovas-streaming.nvsbx.omniverse.nvidia.com[23]

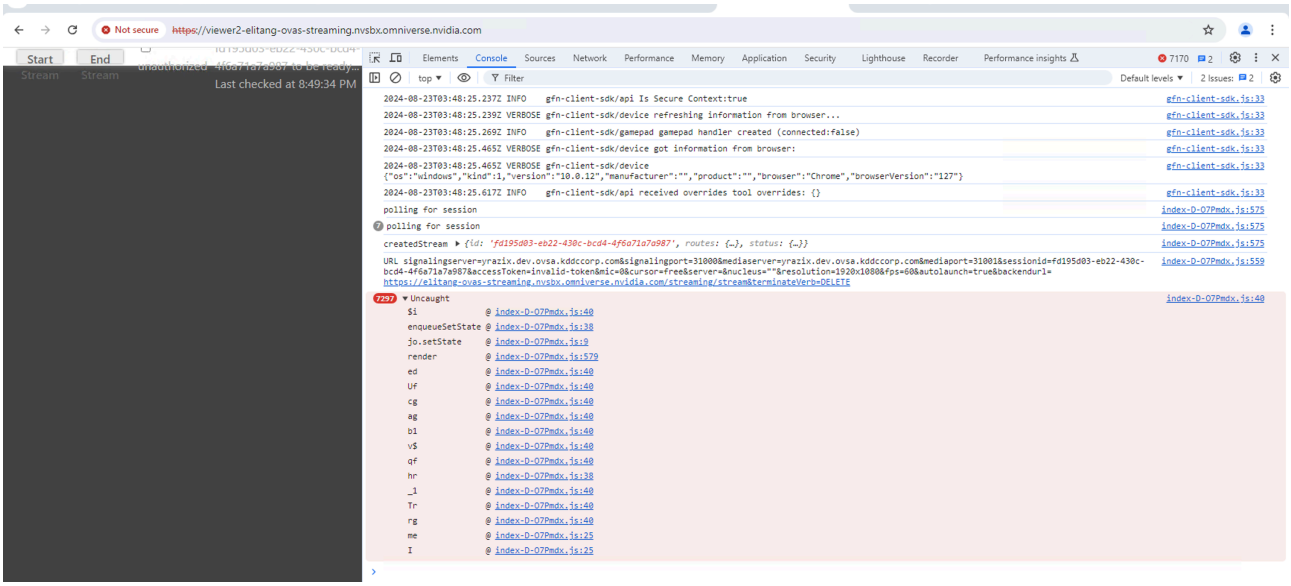
23 http://viewer.elitang-ovas-streaming.nvsbx.omniverse.nvidia.com



If the domain hangs and does not resolve, try troubleshooting with nslookup. Recordsets may not update instantly.

If a NGINX code appears, there may be a problem with your client application.

6.3 Current Bug (Tracked on Jira)



6.4 Changing Application Used

If you need to use a custom application, there may be some changes needed in helm/streaming-sample/templates/configmap.yaml

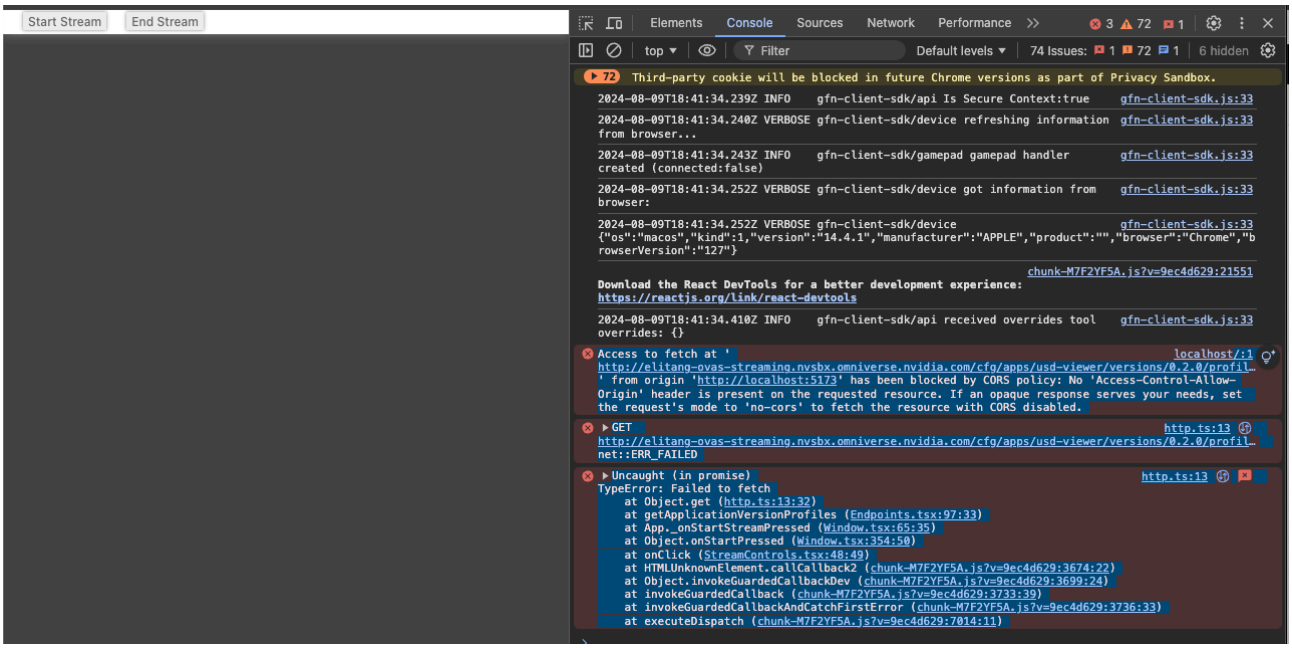
Such as: applicationID, applicationversion, applicatinprofile

7 Deleting resources and environment

Find your resource group and delete the resource group

8 Troubleshooting

8.1 CORS Policy Error



https://stackoverflow.com/questions/3102819/disable-same-origin-policy-in-chrome

On Windows:

1. Search for Run in Windows search
2. Type the command `chrome.exe --user-data-dir="C://Chrome dev session" --disable-web-security`
3. This should open a chrome browser. Access the stream from this browser instead.

8.2 NSG Rules

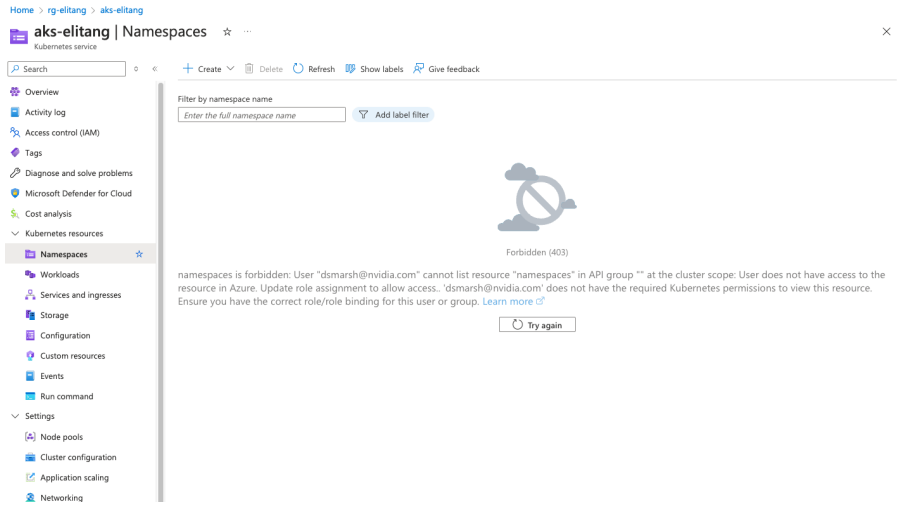
Each tenant may be different and there may be additional NSG rules that must be created. Search for the name of your resources or in NSG on Azure to see if any additional configurations are necessary. If allowed, try opening NSG rules to all temporarily for testing.

8.3 Forbidden access to Kubernetes

If you get an error like

```
dsmarsh@dsmarsh-linux-vm:~$ az aks get-credentials --format azure --resource-group
rg-elitang --name aks-elitang
Merged "aks-elitang" as current context in /home/dsmarsh/.kube/config
dsmarsh@dsmarsh-linux-vm:~$ kubelogin convert-kubeconfig
dsmarsh@dsmarsh-linux-vm:~$ kubectl get pods
Error from server (Forbidden): pods is forbidden: User "dsmarsh@nvidia.com" cannot
list resource "pods" in API group "" in the namespace "default": User does not have
access to the resource in Azure. Update role assignment to allow access.
```

Or you see forbidden in the Azure Portal,



You need to add yourself to the AKS RBAC with these steps:

AKS-RBAC (see page 29)

9 Enabling WSS Troubleshooting (Problem resolving DNS)

One issue in our environment is that there's currently a wildcard DNS entry for `*.nvsbx.omniverse.nvidia.com`[24]. That means if you do an `nslookup` for `<anything>.nvsbx...` it'll resolve to that default ip. The problem is because the client first checks to make sure it can resolve the hostname of the kit app pod (that gets created with external-dns), but the wildcard throws that off and (a) starts trying to connect before the actual record is created and (b) resolves to the wildcard ip instead of the ip of the load balancer.

The workaround implemented here is to switch the external-dns configuration to use a different DNS host (route53 for example), replace the stream-tls-secret with certs for that zone, so that the zone that doesn't have a default entry.

If your environment does not have a wildcard DNS entry, this may not be necessary.

Our alternative zone in Route53 is dev.ovsa.kddccorp.com[25]

Follow the directions from above on creating a certificate for the public DNS zone. Enter the DNS TXT in your zone and obtain the certificate.

9.1 Obtain Public DNS Certificate

```
elitang@elitang-mlt ~ % sudo certbot certonly \
        --manual \
        --preferred-challenges=dns \
        --email user@nvidia.com \
        --server https://acme-v02.api.letsencrypt.org/directory \
        --agree-tos \
        -d dev.ovsa.kddccorp.com \
        -d '*.dev.ovsa.kddccorp.com'
Saving debug log to /var/log/letsencrypt/letsencrypt.log
Requesting a certificate for dev.ovsa.kddccorp.com and *.dev.ovsa.kddccorp.com

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Please deploy a DNS TXT record under the name:

_acme-challenge.dev.ovsa.kddccorp.com.

with the following value:

qlJq309uMTHIRELNLq7gzBgmrsijIc24EdBFqjmb15I

Before continuing, verify the TXT record has been deployed. Depending on the DNS
provider, this may take some time, from a few seconds to multiple minutes. You can
check if it has finished deploying with aid of online tools, such as the Google
```

24 http://nvsbx.omniverse.nvidia.com/
25 http://dev.ovsa.kddccorp.com

```
Admin Toolbox: https://toolbox.googleapps.com/apps/dig/#TXT/_acme-
challenge.dev.ovsa.kddccorp.com.
Look for one or more bolded line(s) below the line ';ANSWER'. It should show the
value(s) you've just added.

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Press Enter to Continue

Successfully received certificate.
Certificate is saved at: /etc/letsencrypt/live/dev.ovsa.kddccorp.com/fullchain.pem
Key is saved at:         /etc/letsencrypt/live/dev.ovsa.kddccorp.com/privkey.pem
This certificate expires on 2024-11-12.
These files will be updated when the certificate renews.

NEXT STEPS:
- This certificate will not be renewed automatically. Autorenewal of --manual
certificates requires the use of an authentication hook script (--manual-auth-hook)
but one was not provided. To renew this certificate, repeat this same certbot command
before the certificate's expiry date.

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
If you like Certbot, please consider supporting our work by:
 * Donating to ISRG / Let's Encrypt:   https://letsencrypt.org/donate
 * Donating to EFF:                    https://eff.org/donate-le
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
elitang@elitang-mlt ~ %
```

Once this certificate is obtained, create a secret in Azure with the following command, specifying where your certs are located.

```
sudo kubectl create secret tls stream-tls-secret --cert=/etc/letsencrypt/live/
dev.ovsa.kddccorp.com/fullchain.pem --key=/etc/letsencrypt/live/
dev.ovsa.kddccorp.com/privkey.pem -n omni-streaming
```

9.2 Change ExternalDNS to Use Alternative Zone

Since our alternative zone existed in AWS, we had to reconfigure externalDNS.

First, remove the previous externalDNS

```
kubectl delete -f /scripts/external-dns/03-external-dns-manifest.yaml
```

9.3 Setting up ExternalDNS (Required for TLS)

External DNS was installed through these docs https://github.com/kubernetes-sigs/external-dns/blob/master/docs/tutorials/aws.md.

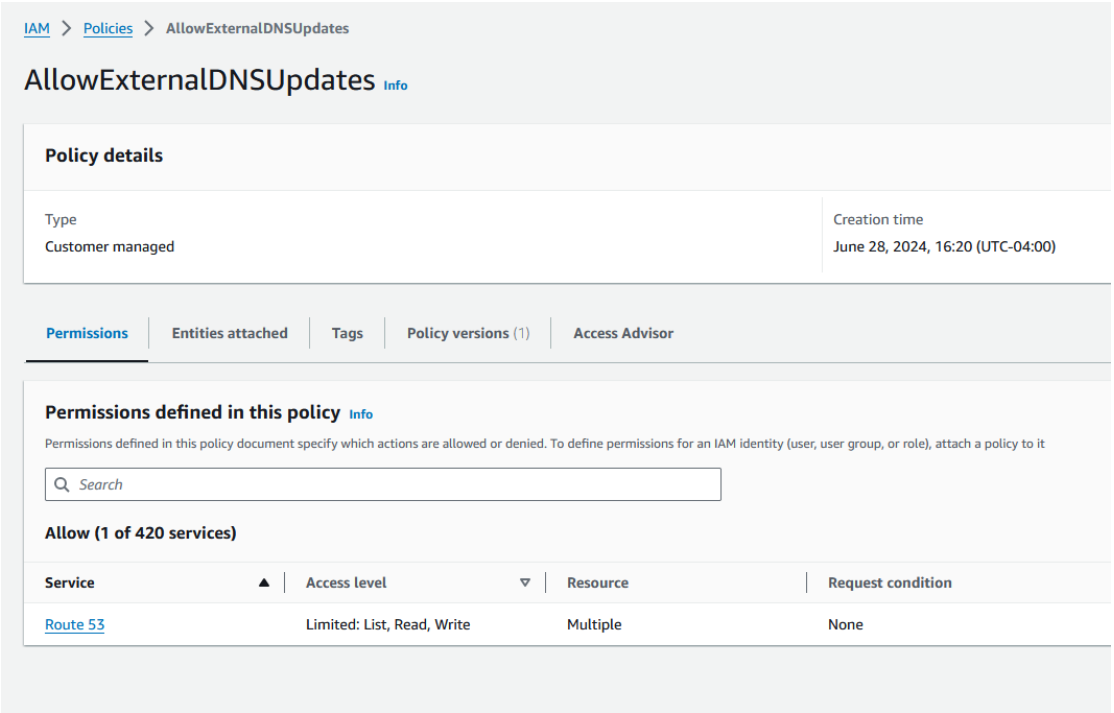
1. Create IAM credentials
    a. Create file with the following contents called `policy.json`

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "route53:ChangeResourceRecordSets"
      ],
      "Resource": [
        "arn:aws:route53:::hostedzone/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "route53:ListHostedZones",
        "route53:ListResourceRecordSets",
        "route53:ListTagsForResource"
      ],
      "Resource": [
        "*"
      ]
    }
  ]
}
```

    b. Apply with

```
aws iam create-policy --policy-name "AllowExternalDNSUpdates" --policy-
document file://policy.json

# example: arn:aws:iam::XXXXXXXXXXXX:policy/AllowExternalDNSUpdates
export POLICY_ARN=$(aws iam list-policies \
 --query 'Policies[?PolicyName==`AllowExternalDNSUpdates`].Arn' --output
text)
```

Verify that this was applied by looking in the AWS Console:



2. Create permissions to modify DNS Zone (static credential steps followed)

a. ### 9.3.1 Create IAM user and attach the policy

```
# create IAM user
aws iam create-user --user-name "externaldns"

# attach policy arn created earlier to IAM user
aws iam attach-user-policy --user-name "externaldns" --policy-arn
$POLICY_ARN
```

b. ### 9.3.2 Create the static credentials

```
SECRET_ACCESS_KEY=$(aws iam create-access-key --user-name "externaldns")
ACCESS_KEY_ID=$(echo $SECRET_ACCESS_KEY | jq -r '.AccessKey.AccessKeyId')

cat <<-EOF > credentials

[default]
aws_access_key_id = $(echo $ACCESS_KEY_ID)
aws_secret_access_key = $(echo $SECRET_ACCESS_KEY | jq -r
'.AccessKey.SecretAccessKey')
```

```
EOF
```

c. 9.3.3 Create Kubernetes secret from credentials

```
kubectl create secret generic external-dns \
  --namespace ${EXTERNALDNS_NS:-"default"} --from-file /local/path/to/
credentials
```

3. Create `externaldns-with-rbac.yaml`. This YAML example file below:
   a. Make sure that the credentials file location matches the location specified in the yaml below. Currently the yaml is expecting this file to be in the local path of `/.aws/credentials`. Create this folder and move the credentials file there. Ex:


```
# comment out sa if it was previously created
apiVersion: v1
kind: ServiceAccount
metadata:
  name: external-dns
  labels:
    app.kubernetes.io/name: external-dns
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: external-dns
  labels:
    app.kubernetes.io/name: external-dns
rules:
  - apiGroups: [""]
    resources: ["services","endpoints","pods","nodes","ingresses"]
    verbs: ["get","watch","list"]
  - apiGroups: ["extensions","networking.k8s.io"]
    resources: ["ingresses"]
    verbs: ["get","watch","list"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: external-dns-viewer
  labels:
    app.kubernetes.io/name: external-dns
```

```
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: external-dns
subjects:
  - kind: ServiceAccount
    name: external-dns
    namespace: default # change to desired namespace: externaldns, kube-addons
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: external-dns
  labels:
    app.kubernetes.io/name: external-dns
spec:
  strategy:
    type: Recreate
  selector:
    matchLabels:
      app.kubernetes.io/name: external-dns
  template:
    metadata:
      labels:
        app.kubernetes.io/name: external-dns
    spec:
      serviceAccountName: external-dns
      containers:
        - name: external-dns
          image: registry.k8s.io/external-dns/external-dns:v0.14.2
          # image: registry.k8s.io/external-dns/external-dns:v1.14.5
          args:
            - --source=service
            - --source=ingress
            - --domain-filter=ovsa.kddccorp.com # will make ExternalDNS see only the
hosted zones matching provided domain, omit to process all available hosted zones
            - --provider=aws
            # - --policy=upsert-only # would prevent ExternalDNS from deleting any
records, omit to enable full synchronization
            - --registry=txt
            - --txt-owner-id=external-dns
          env:
            - name: AWS_DEFAULT_REGION
              value: us-east-1 # change to region where EKS is installed
     # Uncommend below if using static credentials
            - name: AWS_SHARED_CREDENTIALS_FILE
              value: /.aws/credentials
          volumeMounts:
            - name: aws-credentials
              mountPath: /.aws
              readOnly: true
      volumes:
        - name: aws-credentials
```

```
          secret:
             secretName: external-dns
```

There is an pptional args that will restrict `externaldns` from only manipulating the filtered domain to avoid issues where other DNS entries are incorrectly edited. Edit the `externaldns-with-rbac.yaml` file and add this entry in the `args:` section:

```
spec:
...
  template:
...
    spec:
      serviceAccountName: external-dns
      containers:
        - name: external-dns
          image: registry.k8s.io/external-dns/external-dns:v0.14.2
          # image: registry.k8s.io/external-dns/external-dns:v1.14.5
          args:
            - --source=service
            - --source=ingress
###here-->  - --domain-filter=ovsa.kddccorp.com # will make ExternalDNS see only the
hosted zones matching provided domain, omit to process all available hosted zones
            - --provider=aws
            # - --policy=upsert-only # would prevent ExternalDNS from deleting any
records, omit to enable full synchronization
            - --registry=txt
            - --txt-owner-id=external-dns
...
```

Apply with:

```
kubectl apply  --filename externaldns-with-rbac.yaml --namespace $
{EXTERNALDNS_NS:-"default"}
```

Now externalDNS is configured to use the alternative DNS Zone in AWS.

10 Resources Deployed in This Tutorial

10.1 Azure subscription `nv-Omniverse-Nucleus`

Resources deployed in the rg-elitang resource group

NAME	TYPE	LOCATION	
aks-elitang	Kubernetes service	East US	
APIM-elitang	API Management service	East US	
appgw-elitang	Application gateway	East US	
appgw-waf-elitang	Application Gateway WAF policy	East US	
CPU Usage Percentage - aks-elitang	Metric alert rule	Global	
elitang-aks-template	Template spec	East US	
elitang-apim-template	Template spec	East US	
elitang-appgateway-template	Template spec	East US	
elitang-nsg	Network security group	East US	
elitang-ovas-streaming.net[26]	Private DNS zone	Global	
elitang-pub	Public IP address	East US	
elitang-vnet	Virtual network	East US	

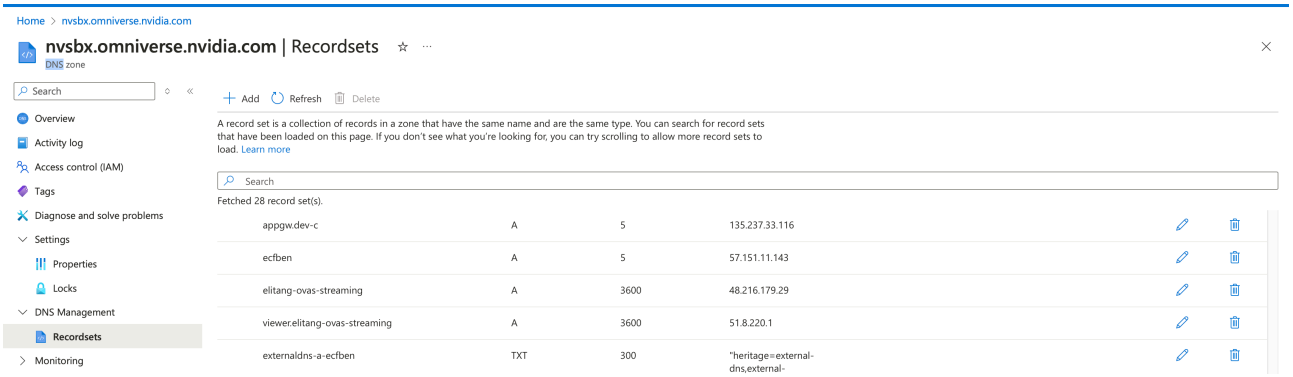
26 http://elitang-ovas-streaming.net

kubernetes-ac3d328687b004cd5a68c4bf9981f6cc	Public IP address	East US	
Memory Working Set Percentage - aks-elitang	Metric alert rule	Global	
RecommendedAlertRules-AG-1	Action group	Global	

In order to access AKS from the CLI or see Kubernetes resources in the portal, you must add RBAC to your user:

To see more go to RBAC for AKS (see page 29)

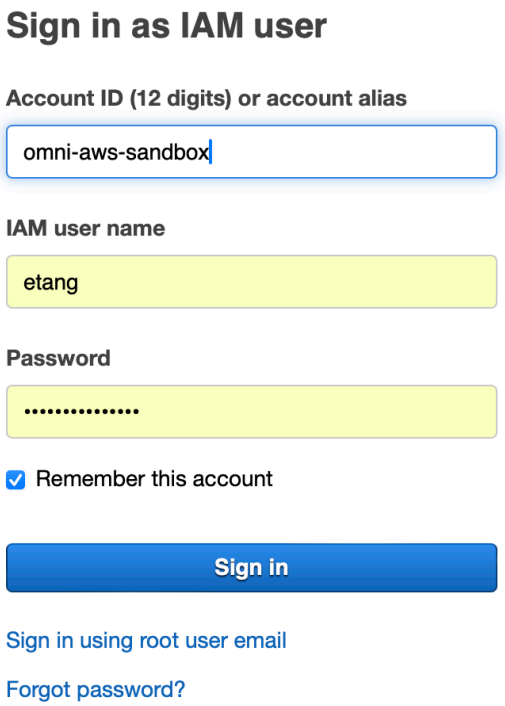
Recordset entries in the nvsbx.omniverse.nvidia.com[27] Public DNS Zone. This belongs to `fidot_sandbox` resource group.

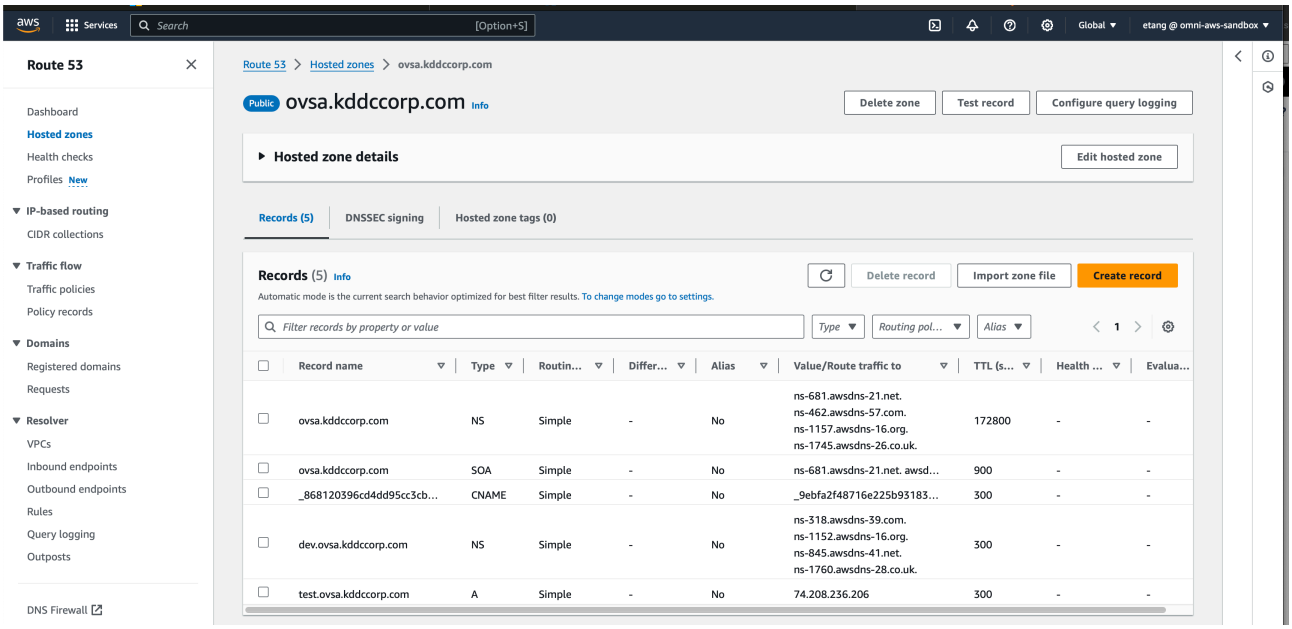


27 http://nvsbx.omniverse.nvidia.com

10.2 AWS

10.2.1 Drew Como's sandbox account: `omni-aws-sandbox`





10.2.2 AWSOS-AD-Admin

- I created a Hosted Zone which uses Drew Como's sandbox domain
- The entry in ovsa.kddccorp.com which is dev.ovsa.kssccorp.com is necessary for the below hosted zone to function. This hosted zone is where my current ExternalDNS makes changes