

Contoso Overview

Microsoft Corporation | One Microsoft Way, Redmond, WA 98052, USA

Azure jumpstart document

NVIDIA: Shashi Bhushan/Ganesh Sivaraman/James Mckenna/MARTIN KARLSSON MICROSOFT: AARON WEISSBART/DRew Dudek/Felipe Becerra/Raj nemani/Rajesh Vasireddy

2024

Contents

[Contoso [Insert Industry Use Case] Overview (Shashi/Aaron/Ganesh/James/Felipe/Martin/Raj) 4](#_Toc182123693)

[Omniverse Kit App Streaming Infrastructure Setup 6](#_Toc182123694)

[Architecture and Technology Stack 6](#_Toc182123695)

[Workflow 7](#_Toc182123696)

[Prerequisites 7](#_Toc182123697)

[Technical Skills required 8](#_Toc182123698)

[Azure Resource Deployment 8](#_Toc182123699)

[Azure Resources Diagram 9](#_Toc182123700)

[Create Resource Group 9](#_Toc182123701)

[Create Virtual Network 10](#_Toc182123702)

[Add Subnets to the Virtual Network 12](#_Toc182123703)

[Create subnet for AKS 12](#_Toc182123704)

[Create subnet for Web Application Firewall (WAF) 13](#_Toc182123705)

[Create subnet for API Management (APIM) Gateway 14](#_Toc182123706)

[Create Network Security Groups 17](#_Toc182123707)

[Assign to subnets subnet-aks and subnet-apim 21](#_Toc182123708)

[DNS and Certificate Configuration 24](#_Toc182123709)

[Create self-signed certificates for private DNS zone 24](#_Toc182123710)

[Create LetsEncrypt certificates manually for public DNS zone 25](#_Toc182123711)

[Create a .pfx from the full chain certificate and private key 27](#_Toc182123712)

[AKS Cluster Configuration 27](#_Toc182123713)

[Create AKS Cluster 29](#_Toc182123714)

[Create Nodepools 30](#_Toc182123715)

[Configure Networking 32](#_Toc182123716)

[Review and Deploy 33](#_Toc182123717)

[RBAC Setup 33](#_Toc182123718)

[Pull kubeconfig locally and check access 33](#_Toc182123719)

[API Management (APIM) Service 35](#_Toc182123720)

[Create APIM Service 35](#_Toc182123721)

[Configure Custom Domain and Private DNS Zone 37](#_Toc182123722)

[Application Gateway Service with Web Application Firewall (WAF) 39](#_Toc182123723)

[Omniverse Kit App Streaming Deployment and Configuration 50](#_Toc182123724)

[Custom Omniverse Kit Application installation and configuration 50](#_Toc182123725)

[Configure ISV Custom Kit App for Deployment 50](#_Toc182123726)

[Upload ISV custom kit app to a Container Registry 51](#_Toc182123727)

[Create Azure Container Registry using the Azure Portal 51](#_Toc182123728)

[Upload Helm Charts etc from NGC recommendation 53](#_Toc182123729)

[Helm Chart Deployment and Configuration 53](#_Toc182123730)

[Set environment-specific values 53](#_Toc182123731)

[Internal ingress controller (helm/nginx-ingress-controller) 54](#_Toc182123732)

[FluxCD (helm/flux2) 54](#_Toc182123733)

[GPU Operator (helm/gpu-operator) 55](#_Toc182123734)

[Memcached (helm/memcached) 55](#_Toc182123735)

[ExternalDNS (scripts/external-dns) 55](#_Toc182123736)

[Create required secrets 57](#_Toc182123737)

[Omniverse Kit App Streaming Services 58](#_Toc182123738)

[Streaming (helm/kit-appstreaming-manager) 58](#_Toc182123739)

[Applications (helm/kit-appstreaming-applications) 58](#_Toc182123740)

[RMCP (helm/kit-appstreaming-rmcp) 59](#_Toc182123741)

[Deploy the custom streaming resources (manifests/omniverse/azure) 59](#_Toc182123742)

[Deploy HelmRepository (manifests/helm-repostiories) 60](#_Toc182123743)

[Create private DNS record for ingress controller 60](#_Toc182123744)

[Create public DNS entry for App Gateway 62](#_Toc182123745)

[Validate Omniverse Kit App Streaming End Points via Swagger UI 62](#_Toc182123746)

[ISV Web App deployment 63](#_Toc182123747)

[Deploying Web Client Application in Azure 64](#_Toc182123748)

[Power BI React Component Configuration 64](#_Toc182123749)

[Azure Apps Registration 64](#_Toc182123750)

[Event Hubs 64](#_Toc182123751)

[Power BI Workspace Settings 65](#_Toc182123752)

[Power BI Data Connections 65](#_Toc182123753)

[Event Hub React Component Configuration 66](#_Toc182123754)

[3D Viewport React Component Configuration 66](#_Toc182123755)

[Azure Static Web Application Deployment 67](#_Toc182123756)

[Deploying a React App to Azure Static Web Apps using Azure Portal 67](#_Toc182123757)

[Create a New Azure Static Web App 67](#_Toc182123758)

[Configure the Static Web App 67](#_Toc182123759)

[Deploy Your React App 67](#_Toc182123760)

[Verify the Deployment 67](#_Toc182123761)

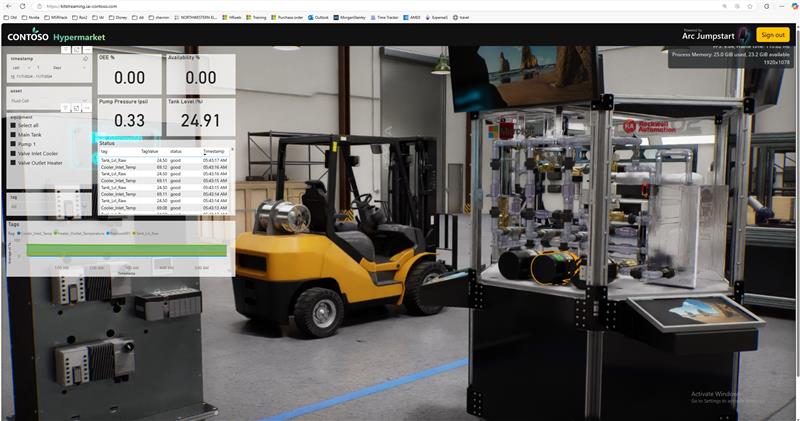
[Final End-to-End Validation (Raj/Rajesh/David/Drew/Rama/Paul) 68](#_Toc182123762)

[3D Viewer Component 68](#_Toc182123763)

[Power BI Component (Drew) 68](#_Toc182123764)

[Validation of Complete ISV Web App Deployment (MSFT) 68](#_Toc182123765)

## Contoso [Insert Industry Use Case] Overview (Shashi/Aaron/Ganesh/James/Felipe/Martin/Raj)



Hook = Business Value Impact – motivate Developers (let’s capture the value broad enough incl IoT but captures production data)

* Industries are concentrating their SMEs to central locations, where they are being asked more than ever to give expert guidance to sites around the world. The combination of real-time machine data, in context and overlaid on 3D visualizations, means can gain situational awareness and react faster than ever.
* Industries are migrating toward a 3D centric workflow
  + Attrition of experts
  + Data vs information
  + ...
* 3D centric workflows allow industries to quickly navigate to areas of interest, discovery issues, concerns, etc., and perform analysis, mitigation
* Microsoft IoT brings together bronze, silver, and gold data together into a unified data lake
* Omniverse Visualization brings the data to life by providing photorealistic and physics accurate representation of the data

Today, developers are building solutions to provide Remote Operations Centers to a variety of industries from factory monitoring/automation to monitoring of physical assets. These developers need mechanisms to quickly unify data from various ingress sources into a single unified environment. From the unified data environment, the developers need to build out applications that display the data in a 3D centric physical representation while maintaining access to the rich underlying data. This enables their users to gain situational awareness for analytics, decision making, maintenance, operations tracking, etc.   
  
Microsoft Fabric platform brings together raw (bronze data), silver (post-processed data) and gold data (interpreted data) into a single data platform for ingress and dissemination. Nvidia Omniverse leverages industry standard OpenUSD to bridge the physical representation of the data with the data platform.

Better Together = Technical Strengths

Fabric + Omniverse diagram of omniverse with authoring and reality capture feeding in, while fabric is the data foundation

Microsoft – AIO – Fabric – 1D 2D Data – Contextualization – Analytics – Dashboarding

NVIDIA - Omniverse – Simulation – Physically Accurate Rendering – Interactive – USD Standards based -

## Omniverse Kit App Streaming Infrastructure Setup

### Architecture and Technology Stack



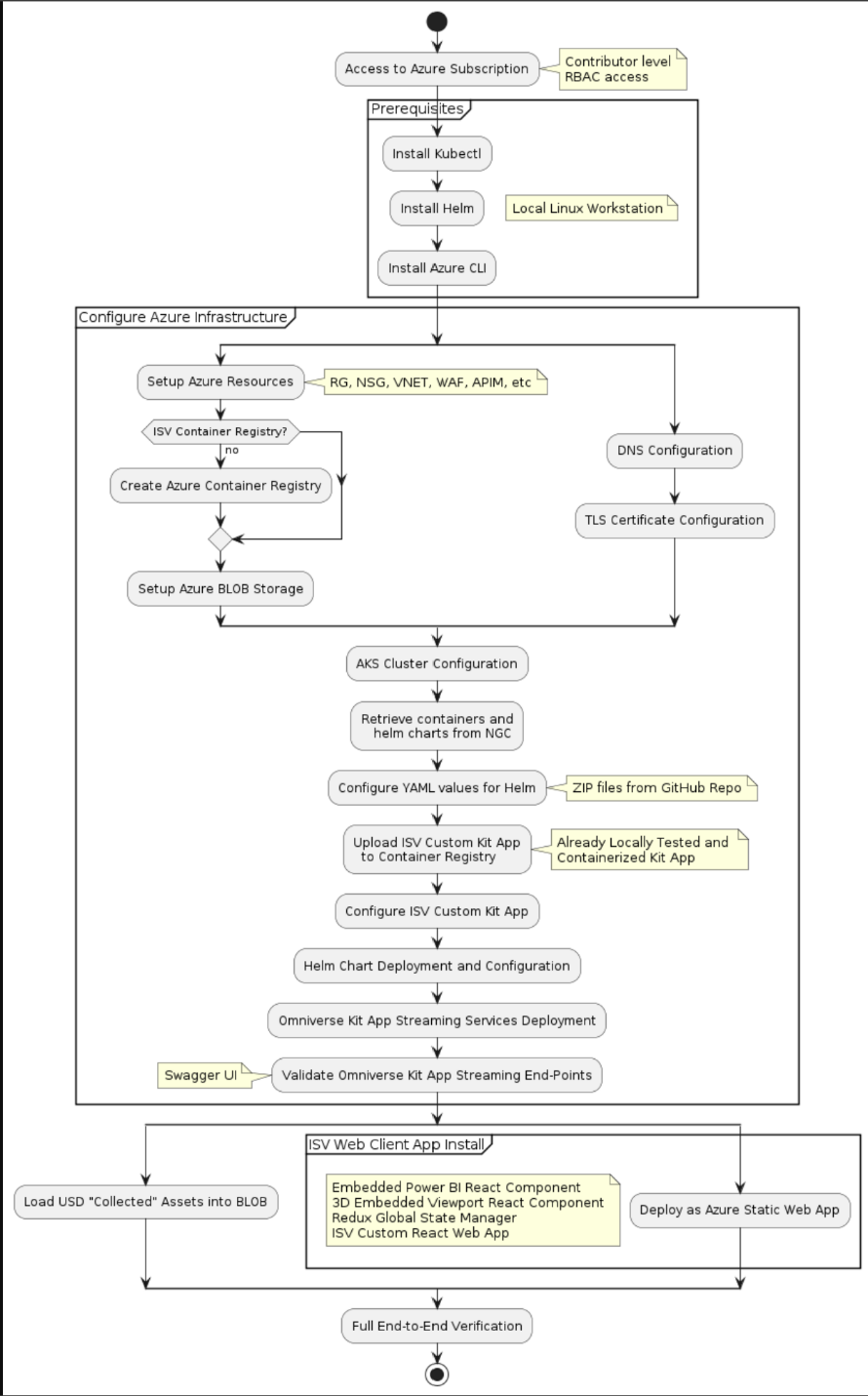
The Azure IoT Operations – Power BI – Omniverse (AIO-PBI-OV) architecture encompasses (from left to right) collecting IoT data from factory floor, efficiently processing and logging it, then updating a Power BI report that is synced with an Omniverse Streaming 3D Viewport to visualize the results in a very high-quality visual interactive digital twin.

The left side of this architectural diagram feeds IoT data via Azure IoT Operations into Event Hubs and Azure Data Explorer (ADX) time-series into a Power BI Report, which then updates the Redux global state with status information that is sent to the 3D Viewport updating the viewport visuals which information such as current operational status.   
  
The 3D Viewport is composed of two parts: 1) an interactive Omniverse streaming client that displays 2) a remotely cloud rendered Omniverse 3D scene via low-latency Omniverse Kit App Streaming. The Omniverse cloud rendering is implemented using Azure Kubernetes Service (AKS) and Omniverse Kit App Streaming that orchestrates a scalable NVIDIA RTX GPU accelerated compute and network infrastructure for an Omniverse Kit App. Both the front-end 3D Viewport client and the backend Omniverse Kit App are expected to be customized by the ISV for their use cases. More information about doing this customization may be found in the associated GitHub Repo [add link here].

This ARC Jumpstart guide provides the essential information needed by an ISV to implement this architecture in their own Azure Commercial Tenant cloud environment.

### Workflow

This Activity Diagram presents a visual overview of the recommended workflow as described in the sections below.



### Prerequisites

Prerequisites required to complete this setup include

* Access to Azure Subscription with Contributor level RBAC access
* Linux Operating system with Bash Shell
* Install [Kubectl](https://kubernetes.io/docs/tasks/tools/) on your deployment system
* Install [Helm](https://helm.sh/docs/intro/quickstart/) on your deployment system
* Install the [Azure CLI](https://learn.microsoft.com/en-us/cli/azure/install-azure-cli) on your deployment system

### Technical Skills required

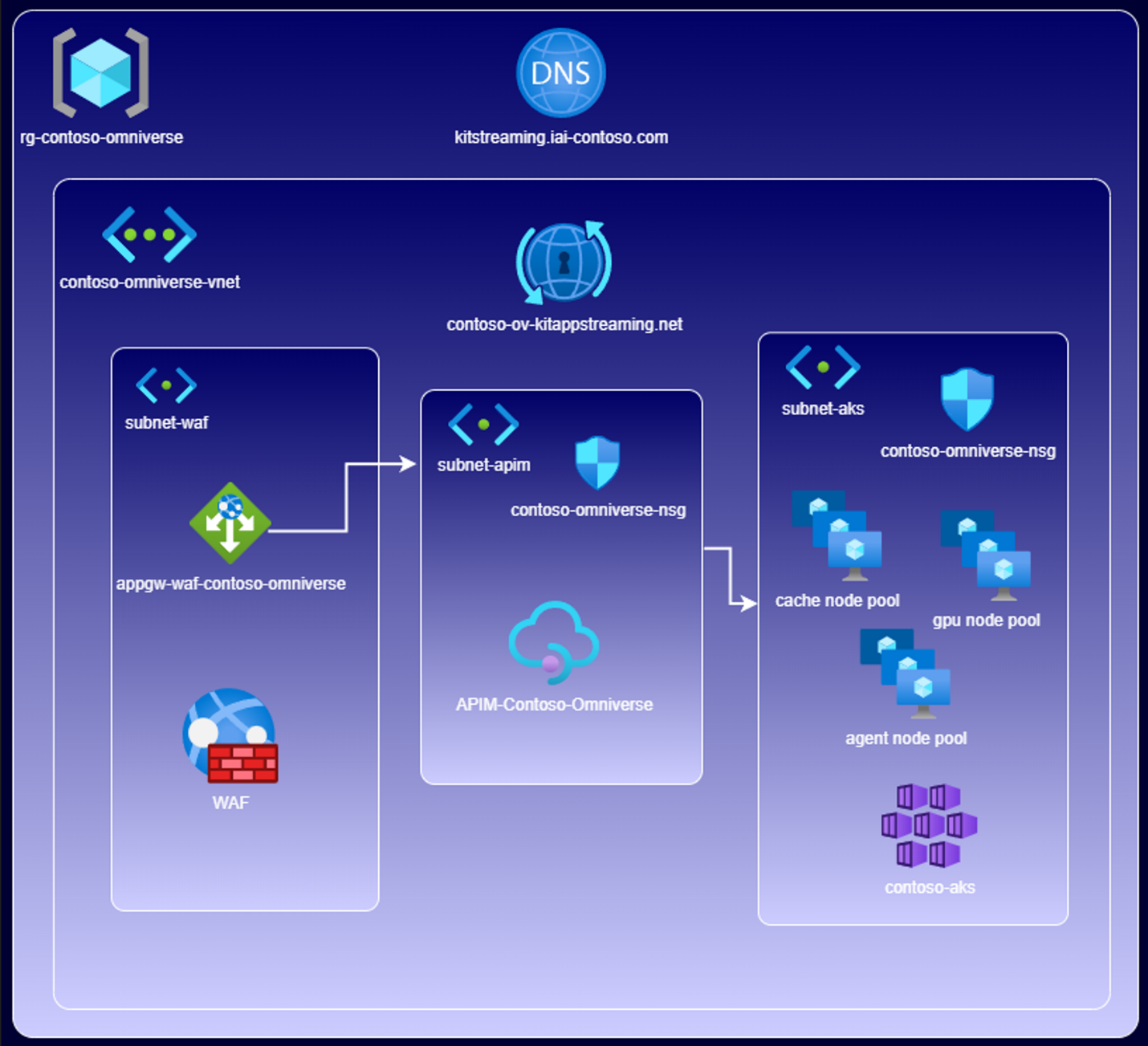
* Azure Cloud Portal
* Using Kubernetes
* Using Helm Charts

### Azure Resource Deployment

To access your subscription login to <http://portal.azure.com> and make sure you are in the correct subscription. From here, the setup steps assume the following.

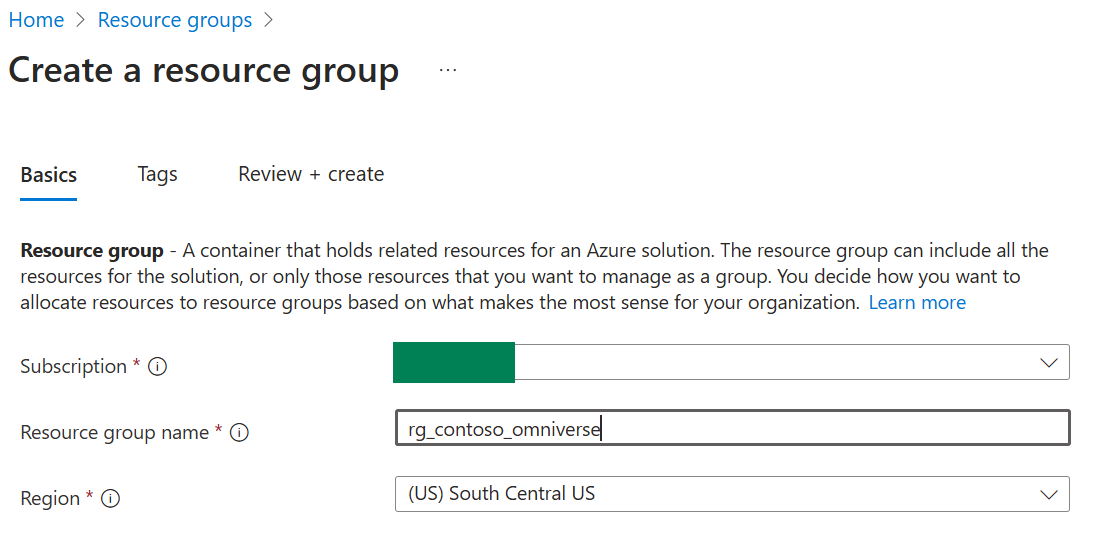
* All resources are created within a single resource group
* All resources are created in the (US) East US region
* All resource names are for example purposes only and are chosen for consistency purposes in documentation, you may choose your own resource names
* The selected vnet range and subnet ranges do not overlap (or are taken into consideration) with any planned peered vnets
* The private DNS zone for our deployment is contoso-ov-kitappstreaming.net
* The public DNS zone for our deployment is iai-contoso.com and we will create a record for subdomain kitstreaming in subsequent steps.

#### Azure Resources Diagram



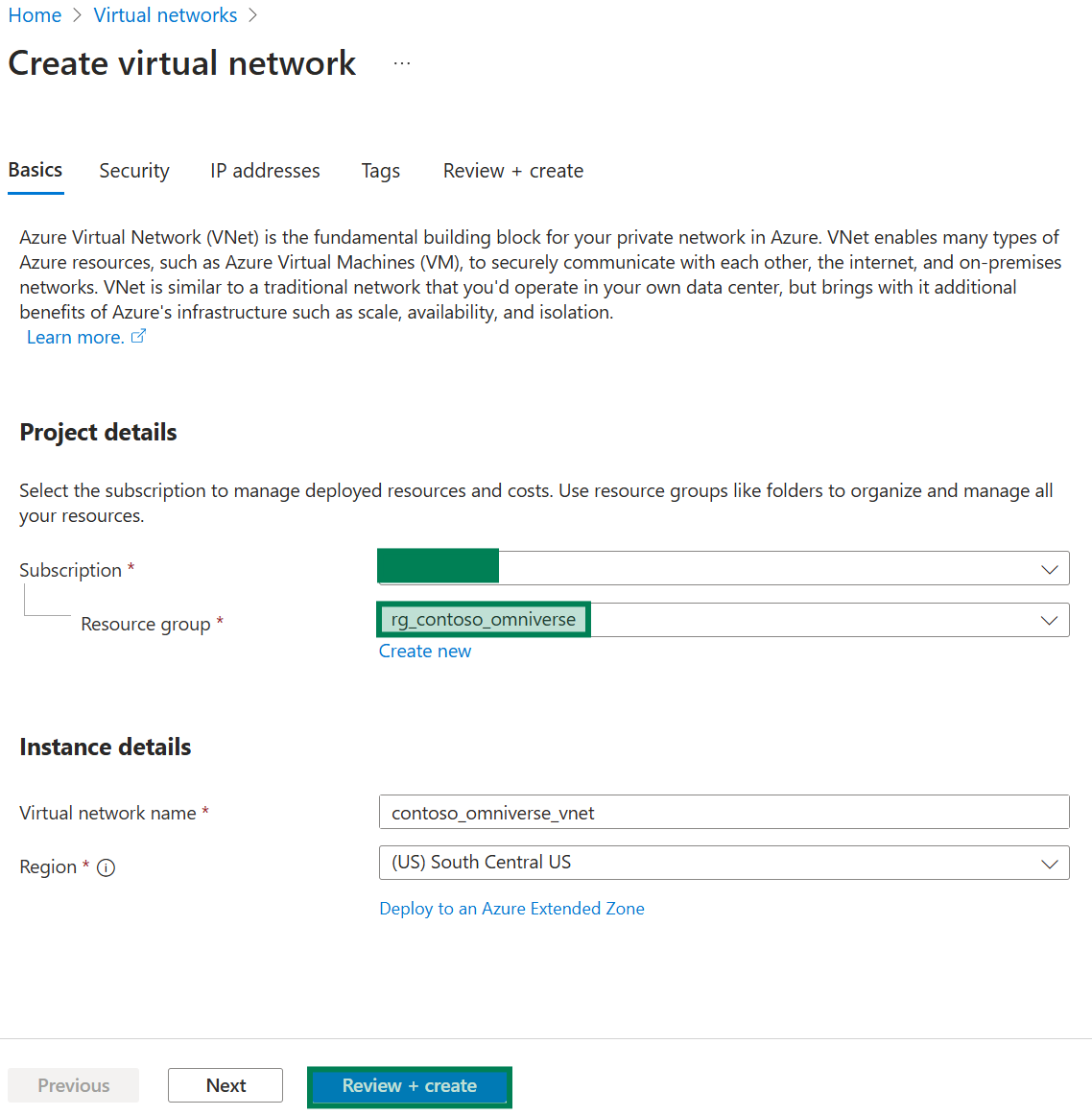
#### Create Resource Group

Using the search tool at the top of the portal home page, search for “Resource Groups”, select the “Resource Groups” option to navigate to “Resource Groups” page. Click on “+ Create” to create a new resource group. Enter the information on the “Create Resource Group” dialog as shown in the picture below. Select the correct subscription from the “Subscription” list. Provide a name for the resource group and select the region where you want this resource group to be created. Click “Review+Create” button, do a final review of the supplied information in the next screen and click “Create”.



#### Create Virtual Network

Using the search tool at the top of the portal home page, search for “Virtual Networks”, select the “Virtual Networks” option from the search results to navigate to “Virtual Networks” page. Click on “+ Create” to create a new virtual network. Enter the information on the “Create virtual network” dialog as shown in the picture below. Select the correct subscription from the “Subscription” list. Select the same resource group that was created earlier. Provide a name for the virtual network and then select the region where you want this virtual network to be created in.



#### Add Subnets to the Virtual Network

In thie section we will add 3 subnets to the Vnet created . Name of the subnet and IP address space for each subnet are shown below.

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

Go to “IP addresses” tab on the “Create Virtual Network” dialog. Change he address space to 10.2.0.0/16 and then add the three subnets as shown in the screenshot below. In the next sectons we will discuss creation of each subnet.

A screenshot of a computer

Description automatically generated

##### Create subnet for AKS

Click “+ Add Subnet” button to navigate to “Add Subnet” dialog. Enter all the required information as showin in the screenshot below and click “Add” to create the subnet.

A screenshot of a computer

Description automatically generated

##### Create subnet for Web Application Firewall (WAF)

Click “+ Add Subnet” button to navigate to “Add Subnet” dialog. Enter all the required information as showin in the screenshot below and click “Add” to create the subnet.

A screenshot of a computer

Description automatically generated

##### Create subnet for API Management (APIM) Gateway

Click “+ Add Subnet” button to navigate to “Add Subnet” dialog. Enter all the required information as showin in the screenshot below and click “Add” to create the subnet.

A screenshot of a computer

Description automatically generated

Finally, when all the subnets are created, click on “Review+Create” button on the “Create Virtual Network” dialog, perform a final review of the supplied information, and created subnets and click on “Create” to create the virtual network.

A screenshot of a computer

Description automatically generated

#### Create Network Security Groups

Using the search tool at the top of the portal home page, search for “Network Security Groups”, select the “Network Security Groups” option from the search results to navigate to “Network Security Groups” page. Click on “+ Create” to create a new network security group. Enter information on the “Create network security group” dialog as shown in the picture below. Select the correct subscription from the “Subscription” list. Select the same resource group that was created earlier. Provide a name for the network security group and then select the region where you want this network security group to be created in.

A screenshot of a computer

Description automatically generated

Create the following inbound rules as shown in the screenshots below.

* 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

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

#### Assign to subnets subnet-aks and subnet-apim

Navigate to the contoso-omniverse-nsg, click on Settings and then Subnets to navigate to “contoso-omniverse-nsg | Subnets” page. Click on “+ Associate”. On the “Associate subnet” dialog box, select “contoso-omniverse-vnet” as the virtual network and “subnet-aks” as the subnet to associate the network security group. Repeat the same process to associate the network security group to subnet “subnet-apm”

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

### DNS and Certificate Configuration

These may be long lead items as they may require additional IT/Networking approvals.

To access from Internet requires permissions to insert DNS ‘A’ Record.

Requires CA signed certificate, as self-signed certificates may not be sufficient

#### Create self-signed certificates for private DNS zone

In the below commands, the private DNS zone is contoso-ov-kitappstreaming.net. Use following Bash script to generate both a root and SSL certificate:

1. Execute the following. Make changes to the CN

# Generate a private key.

openssl genrsa -out contoso-ov-kitappstreaming-net-signing-root.key 40963. 4. # Create a self-signed root certificate.5. openssl req -x509 -new -nodes -key contoso-ov-kitappstreaming-net-signing-root.key \6. -sha256 -days 1825 -out contoso-ov-kitappstreaming-net-signing-root.crt \7. -subj "/CN=contoso-ov-kitappstreaming-net-signing-root"

1. Execute the following commands. Make changes to the DNS Name. Make changes to the CN.

# Create a private key for the wildcard SSL certificate:

openssl genrsa -out contoso-ov-kitappstreaming-ssl.key 20483.

# Create a Certificate Signing Request (CSR):

openssl req -new -key contoso-ov-kitappstreaming-ssl.key -out contoso-ov-kitappstreaming-ssl.csr -subj "/CN=\*.contoso-ov-kitappstreaming.net" -addext "subjectAltName=DNS:contoso-ov-kitappstreaming.net,DNS:\*.contoso-ov-kitappstreaming.net"

# Sign the SSL certificate with the root certificate:

openssl x509 -req -in contoso-ov-kitappstreaming-ssl.csr -CA contoso-ov-kitappstreaming-net-signing-root.crt -CAkey contoso-ov-kitappstreaming-net-signing-root.key -CAcreateserial -out contoso-ov-kitappstreaming-ssl.crt -days 825 -sha256 -extfile <(printf "subjectAltName=DNS:contoso-ov-kitappstreaming.net,DNS:\*.contoso-ov-kitappstreaming.net")

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

# Export CER of the root and SSL certs2.

openssl x509 -in contoso-ov-kitappstreaming-signing-root.crt -out contoso-ov-kitappstreaming-signing-root.cer3.

openssl x509 -in contoso-ov-kitappstreaming-ssl.crt -out contoso-ov-kitappstreaming-ssl.cer

1. Edit the password argument and execute the following commands to create the two certificates. These will be stored in the directory you are currently in. NOTE: Password is required.

# Export PFX of the root and SSL certs

openssl pkcs12 -export -out contoso-ov-kitappstreaming-signing-root.pfx -inkey contoso-ov-kitappstreaming-net-signing-root.key -in contoso-ov-kitappstreaming-net-signing-root.crt -password pass:<password>

openssl pkcs12 -export -out contoso-ov-kitappstreaming-ssl.pfx -inkey contoso-ov-kitappstreaming-ssl.key -in contoso-ov-kitappstreaming-ssl.crt -certfile contoso-ov-kitappstreaming-net-signing-root.crt -password pass:<password>

#### 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. In the following examples, kitstreaming.iai-contoso.com is the public DNS zone.

sudo certbot certonly --manual --preferred-challenges=dns --email [user@nvidia.com](mailto:user@nvidia.com) --server <https://acme-v02.api.letsencrypt.org/directory> --agree-tos -d kitstreaming.iai-contoso.com -d '\*.kitstreaming.iai-contoso.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/<Public DNS Zone>

Example output:

sudo certbot certonly --manual --preferred-challenges=dns –email [user@nvidia.com](mailto:user@nvidia.com) --server <https://acme-v02.api.letsencrypt.org/directory> --agree-tos -d kitstreaming.iai-contoso.com -d '\*.kitstreaming.iai-contoso.com' Saving debug log to /var/log/letsencrypt/letsencrypt.log

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - Would you be willing, once your first certificate is successfully issued, to share your email address with the Electronic Frontier Foundation, a founding partner of the Let's Encrypt project and the non-profit organization that develops Certbot? We'd like to send you email about our work encrypting the web, EFF news, campaigns, and ways to support digital freedom.

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - (Y)es/(N)o: Y

Account registered.

Requesting a certificate for kitstreaming.iai-contoso.com and \*.kitstreaming.iai-contoso.com

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - Please deploy a DNS TXT record under the name: \_acme-challenge.kitstreaming.iai-contoso.comwith the following value:

EdO-qo1kRstJ6lGiwRI9fm-UBByVsmUN9Tfb\_hVOBTs

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

Press Enter to Continue

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

Please deploy a DNS TXT record under the name:

\_acme-challenge.kitstreaming.iai-contoso.com. with the following value:

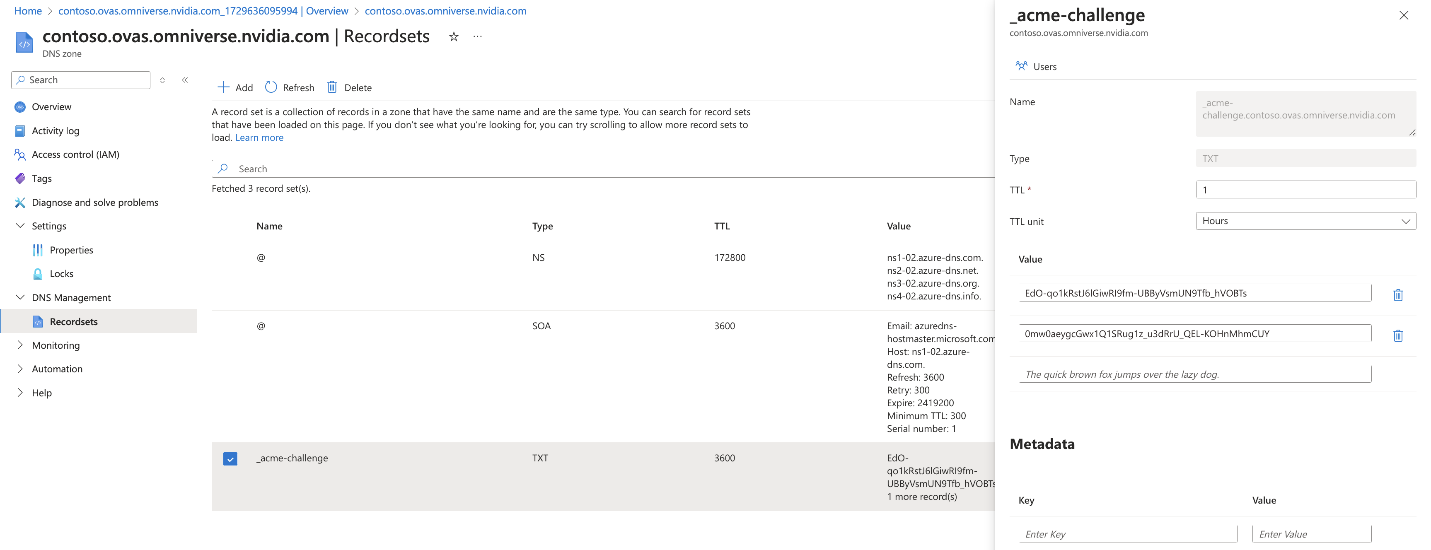
0mw0aeygcGwx1Q1SRug1z\_u3dRrU\_QEL-KOHnMhmCUY

(This must be set up in addition to the previous challenges; do not remove, replace, or undo the previous challenge tasks yet. Note that you might be asked to create multiple distinct TXT records with the same name. This is permitted by DNS standards.)

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 Admin Toolbox: [https://toolbox.googleapps.com/apps/dig/#TXT/\_acme-challenge.kitstreaming.iai-contoso.com.](https://toolbox.googleapps.com/apps/dig/#TXT/_acme-challenge.contoso.ovas.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 DNS Management > Recordsets and verify that the record shows up with the link provided in the command output above.





Once you have verified that the TXT entry shows up press Enter. If this fails, try running the command again or waiting longer.

#### 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/kitstreaming.iai-contoso.com/fullchain.pem -inkey /etc/letsencrypt/live/kitstreaming.iai-contoso.com/privkey.pem -out kitstreaming.iai-contoso.com.pfx

Your file will be named <Public DNS Zone>.pfx. Change the value after -out to name it differently. Save the location of this file, as it will need to be uploaded later.

### AKS Cluster Configuration

Omniverse requires a mix of General Purpose & GPU computation power provisioned as Azure Kubernetes Service node pools. Please ensure your Azure subscription receivedquota to provision Azure virtual machines for the following AKS agent node pool configurations

* Control Plane
  + Node Pool Name: agentpoolds
  + VM SKU: Standard D2s V3
  + Minium Quota Quantity: 2 VMs
* Cache Layer
  + Node Pool Name: cachepool
  + VM SKU: Standard D8S V3
  + Minimum Quota Quantity: 1VM
* GPU Compute
  + Node Pool Name: gpupool
  + VM SKU: Standard NV36ads A10 V5
  + Minimum Quota Quantity: 1 VM

To provision AKS cluster, start by search & select “Kubernetes Services” in the Azure Portal top search box and click “Create” - > “Kubernetes cluster”. Fill the Create Kubernetes cluster form by selecting the pre-provisioned resource group, give a name for the Kubernetes cluster and select “Microsoft Entra ID authentication with Azure RBAC”. The adjacent screenshot shows an example of a filled form. Please do **NOT** click “Review + Create” yet, as we need to pre-configure Node pools and Networking tabs as part of creating the AKS cluster.

A screenshot of a computer

Description automatically generated

#### Create AKS Cluster

#### 

#### Create Nodepools

We will create three node pools as part of the AKS cluster configuration i.e. Agent pool, Cache pool & GPU pool with Azure VM SKU Types described in the introductory segment of AKS cluster configuration.

Sample forms of the node pool configurations provided below for quick reference.

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated Please ensure you have sufficient capacity quota granted to your subscription. If required please open Azure support ticket to seek capacity allocation as described in [this link](https://learn.microsoft.com/en-us/azure/quotas/quickstart-increase-quota-portal).

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

#### 

Once the node pools are configured, move onto the “Networking” tab to connect AKS cluster with the pre-created Virtual Network associated with necessary Network Security group rules.

#### Configure Networking

Please select “Azure CNI Node Subnet” option for the Container Networking - > Network Configuration parameter and select the pre-created Virtual Network from the drop down as shown below.

A screenshot of a computer

Description automatically generated

Review all the configurations across the “Basics”, “Node Pools” & “Networking” tabs of the “Create Kubernetes Cluster” form and when ready, click “Review + Create”. This step will take time as it involves provisioning compute resources as specified in the node pool configurations.

#### Review and Deploy

#### RBAC Setup

To access the AKS Cluster and manage resources inside Kubernetes, proper role assignments within the resource group must be completed. Add any user here that will need access to the cluster by completing the following steps:

* Navigate to Access control (IAM)

A screenshot of a web page

Description automatically generated

* Click Add, and navigate to Add role assignment

A screenshot of a computer

Description automatically generated

* Search Azure Kubernetes Service in the role search
* Add desired user to Azure Kubernetes Service RBAC Cluster Admin
* Add desired user to Azure Kubernetes Service RBAC Admin

#### Pull kubeconfig locally and check access

To begin, log in to the az CLI with az login. Ensure the correct subscription is selected with az account set --subscription <subscription-id>

Install [kubelogin](https://azure.github.io/kubelogin/install.html) and run the following commands:

user@contoso ~ % az aks get-credentials --format azure --resource-group rg\_contoso\_omniverse --name aks-contoso

Merged "aks-contoso" as current context in /Users/user/.kube/config

user@contoso ~ % export KUBECONFIG=/Users/user/.kube/config

user@contoso ~ % kubelogin convert-kubeconfig –l azurecli

user@contoso ~ % 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.

user@contoso ~ % kubectl get nodes

Check that the nodes you created appear.

### API Management (APIM) Service

#### Create APIM Service

To provision the API Management (APIM) service, start by searching for “API Management” in the Azure Portal top search box and select “API Management Services” option from the result list to navigate to “API Management Services” screen. Click “+ Create” on the “API Management Services” screen to navigate to “Create API Management service” screen. Populate “Basics” tab with information as shown in the screenshot. Please do **NOT** click “Review + Create” yet, as we need configure a virtual network for the service in the “Virtual Network” tab.

A screenshot of a computer

Description automatically generated

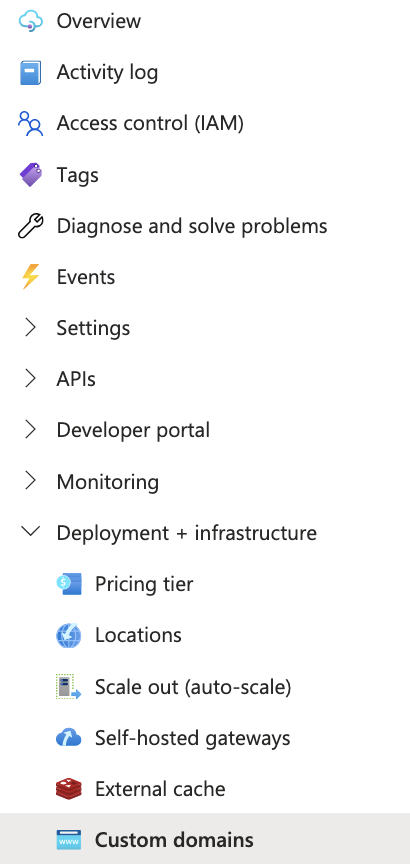
On the virtual network tab, select Connectivity type, Virtual network and subnet options as shown in the example screenshot below.

A screenshot of a computer

Description automatically generated

#### Configure Custom Domain and Private DNS Zone

1. Once deployed, add a custom domain:
   1. Select Custom Domains under Deployment and Infrastructure



* 1. Add a new custom domain for the gateway:
     + **Type**: Gateway
     + **Hostname**: [apim-gw.contoso-ov-kitappstreaming.net](http://apim-gw.ovas-streaming.net/)← This should be apim-gw.<your private DNS>
     + **Certificate**: Custom
     + **Certificate file**: contoso-ov-kitappstreaming-ssl.pfx ← This is the self-signed certificate that was created previously
     + **Check** Default SSL binding

A screenshot of a computer

Description automatically generated

* 1. Add a new custom domain for the management gateway:
     + **Type**: Management
     + **Hostname**: [apim-mgmt.contoso-ov-kitappstreaming.net](http://apim-mgmt.ovas-streaming.net/) ← This should be apim-mgmt.<your private DNS>
     + **Certificate**: Custom
     + **Certificate file**: contoso-ov-kitappstreaming-ssl.pfx ← This should be what the SSL certificate file was named previously.

A screenshot of a computer

Description automatically generated

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

1. Add DNS records to private DNS zone:
   1. Add two **A records** for apim-gw and apim-mgmt to previously created private DNS zone pointing to private IP of the APIM instance created previously.

### Application Gateway Service with Web Application Firewall (WAF)

To provision the Application Gateway service, start by searching for “Application Gateway” in the Azure Portal top search box and select “Application Gateways” option from the result list to navigate to “Load Balancing | Application Gateway” screen. Click on “+ Create” button on this screen and navigate to the “Create application gateway” screen. Provide information in the “Basics” tab as shown in the screenshot example of a filled form. Please do **NOT** click “Review + Create” yet, as we need configure “Frontends”, “Backends”, and “Configuration” tabs in the subsequent steps.

**Basics**

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

A screenshot of a computer

Description automatically generated

**Frontends**

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

A screenshot of a computer

Description automatically generated

**Backends**

Create new

* **Name**: apim
* **Backend target**: apim-gw.contoso-ov-kitappstreaming.net ← This should be apim-gw.<your private DNS>

A screenshot of a computer screen

Description automatically generated

**Configuration**

Add new routing rule

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

A screenshot of a computer

Description automatically generated

**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:** ov-kitappstreaming-signing-root.cer
* **Override with new host name:** Yes
* Select Pick hostname from backend target
* **Create custom probes:** Yes

Example:

A screenshot of a computer

Description automatically generated

**Once deployed go to app gateway:**

**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

A screenshot of a computer screen

Description automatically generated

**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**: contoso-ov-kitappstreaming
  + **PFX Certificate File:** (.pfx file created earlier)
  + **Password**:
  + **Listener type:** Multi site
  + **Host type:** Multiple/Wildcard
  + **Host names**:
    - kitstreaming.iai-contoso.com
    - \*.kitstreaming.iai-contoso.com

A screenshot of a computer

Description automatically generated

If it shows “Failed to save application gateway changes”:

|  |
| --- |
| user@contoso ~ % sudo chmod 777 kitstreaming.iai-contoso.com.pfx |

**Routing Rules**

Under Settings > Rules, click + Routing rule

* **Name**: https
* **Priority**: 10
* **Listener**: https

A screenshot of a computer screen

Description automatically generated

**Backend pool**

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

A screenshot of a computer

Description automatically generated

Under Settings > Rules, click waf-to-apim-internal

**Backend targets**

* Change Target type from Backend pool to Redirection
* **Target listener:** https

A screenshot of a computer screen

Description automatically generated

**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 APIM 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.

**RBAC**

Assign RBAC permissions to enterprise app registration created by AKS cluster

* Find the vnet created in step 1
* Under Access control (IAM), click Check access
  + Search for the cluster name after selecting Managed Identities and add the managed identity of the AKS cluster

A screenshot of a computer screen

Description automatically generated

## Omniverse Kit App Streaming Deployment and Configuration

For an overview of Omniverse Kit App Streaming please see [Overview — Omniverse Application Streaming API latest documentation (nvidia.com)](https://docs.omniverse.nvidia.com/ovas/latest/index.html)

### Custom Omniverse Kit Application installation and configuration

ISV Custom Kit app needs to be substituted for the default kit app in the Kubernetes pod (the pod also contains an Envoy proxy). This has already been done for the sample custom kit app but would need to be reconfigured for ISV specific custom Kit app. See README.md in GitHub repo for creating a Docker container for an ISV custom kit app.

Omniverse Kit is a powerful toolkit for developers to build applications, plugins, or microservices for their own ecosystems. In this document, we describe leveraging Omniverse Kit to build a custom rendering application.

#### Configure ISV Custom Kit App for Deployment

This is where you need to set the YAML files values to pull the specific version of the ISV custom kit app previously uploaded to an accessible Container Registry (e.g. private Azure Container Registry in same Resource Group).

Omniverse Kit App Streaming allows you to register different Kit containers as potential candidates for streaming sessions. To use your custom container, you will need to register a new `application`, `application-version`, and `application-profile`.

* Application - the overall Kit Application that you want to make available for streaming.
* ApplicationVersion - a specific release or version of an application.
* Application Profile - the runtime configuration and deployment settings to use when instantiating a Kit Application stream.

You can read more about integrating and managing containerized Omniverse Kit Applications in the official Omniverse Kit App Streaming documentation here: [<https://docs.omniverse.nvidia.com/ovas/latest/deployments/apps/index.html>]

Specifically, be sure to change the `image` and `imagePullSecrets` values `application-profile.yaml` and `application-version.yaml` before applying the modified helm charts to your cluster.

If your container registry is guarded by a secret, you will need to configure an Image Registry Pull Secret. You can read more about this here: [<https://docs.omniverse.nvidia.com/ovas/latest/deployments/infra/installation.html#create-image-registry-pull-secret>]

```console

kubectl create secret -n omni-streaming docker-registry myregcred\  
 --docker-server=<TODO> \  
 --docker-username=<TODO> \  
 --docker-password=<TODO>\  
 --dry-run=client -o json |  
 kubectl apply -f -  
  
secret/myregcred created

```

More detailed information may be found at [Deploying Omniverse Kit Applications — Omniverse Application Streaming API latest documentation](https://docs.omniverse.nvidia.com/ovas/latest/deployments/apps/index.html)

For the bundled sample kit app the matching configuration values:

appId: 'usd-viewer-msft',

version: '106.1.0',

profile: 'azurelb-wss',

### Upload ISV custom kit app to a Container Registry

The containerized kit app needs to be accessible by Kubernetes to provide the OV kit app streaming functionality, hence the ISV needs to use either one of their own existing container registries or create a private Azure Container Registry for this project that holds the containerized ISV Kit App.

See GitHub Project README.md for guidance on creating a docker container for the ISV custom kit app.

#### Create Azure Container Registry using the Azure Portal

**Create a Private Container Registry**

* Click on the **+ Create a resource** button.
* Search for "Container Registry" in the search bar.
* Select **Container Registry** from the results.
* Click on the **Create** button.
* Fill in the required information:
  + **Registry name**: Enter a unique name for your container registry (e.g., "my-container-registry").
  + **Resource group**: Select the resource group you created in Step 2.
  + **Location**: Choose a location for your container registry (e.g., "East US").
  + **SKU**: Select the desired SKU for your container registry (e.g., "Standard").
  + **Admin user**: Choose whether to enable or disable the admin user.
* Click on the **Create** button.

**Configure the Private Container Registry**

* Go to the **Private Container Registry** resource you created in Step 3.
* Click on the **Settings** tab.
* Configure the following settings as desired:
  + **Repository**: Create a new repository or link to an existing one.
  + **Access policies**: Configure access policies for your registry.
  + **Network policies**: Configure network policies for your registry.
* Click on the **Save** button.

**Create a Service Principal for Authentication**

* Go to the **Azure Active Directory** resource.
* Click on the **App registrations** tab.
* Click on the **+ New registration** button.
* Fill in the required information:
  + **Name**: Enter a unique name for your service principal (e.g., "my-container-registry-sp").
  + **Supported account types**: Select "Accounts in this organizational directory only".
* Click on the **Register** button.
* Go to the **Certificates & secrets** tab.
* Click on the **+ New client secret** button.
* Fill in the required information:
  + **Description**: Enter a description for your client secret (e.g., "my-container-registry-sp-secret").
* Click on the **Add** button.
* Copy the client secret value.

**Configure Docker to Use the Private Container Registry**

* Install Docker on your machine if you haven't already.
* Run the following command to configure Docker to use your private container registry:

docker login <registry-name>.azurecr.io

Replace <registry-name> with the name of your container registry (e.g., "my-container-registry").

* Enter the username and password for your service principal when prompted.

**Push the Docker Image to the Azure Private Container Registry**

* Run the following command to push the Docker image to your ACR:

docker push <registry-name>.azurecr.io/<image-name>

Replace <registry-name> with the name of your ACR (e.g., "my-container-registry") and <image-name> with the name of the Docker image you want to upload (e.g., "hello-world").

**Verify the Private Container Registry**

* Run the following command to verify that your private container registry is working correctly:

docker pull <registry-name>.azurecr.io/<image-name>

### Upload Helm Charts etc from NGC recommendation

*Note: Kubernetes containers and Helm charts are retrieved from NGC.* [Omniverse Application Streaming API | NVIDIA NGC](https://catalog.ngc.nvidia.com/orgs/nvidia/teams/omniverse/collections/kit-appstreaming-collection)

### Helm Chart Deployment and Configuration

#### 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](http://service.beta.kubernetes.io/azure-load-balancer-resource-group): name of resource group ex: rg\_contoso\_omniverse
  + **helm/kit-appstreaming-applications/values.yaml** 
    - host: api.<private DNS zone> ex: api.contoso-ov-kitappstreaming.net
    - Global.ingress.annotations.<>:

# CORS-related annotations

nginx.ingress.kubernetes.io/enable-cors: "true"

nginx.ingress.kubernetes.io/cors-allow-origin: "\*"

nginx.ingress.kubernetes.io/cors-allow-methods: "GET, PUT, POST, DELETE, PATCH, OPTIONS"

nginx.ingress.kubernetes.io/cors-allow-headers: "DNT,User-Agent,X-Requested-With,If-Modified-Since,Cache-Control,Content-Type,Range,Authorization,Accept,Origin"

nginx.ingress.kubernetes.io/cors-expose-headers: "Content-Length,Content-Range"

nginx.ingress.kubernetes.io/cors-allow-credentials: "true"

nginx.ingress.kubernetes.io/cors-max-age: "86400"

# SSL/TLS settings

nginx.ingress.kubernetes.io/ssl-redirect: "true"

nginx.ingress.kubernetes.io/hsts: "true"

nginx.ingress.kubernetes.io/hsts-max-age: "31536000"

nginx.ingress.kubernetes.io/hsts-include-subdomains: "true"

* + - repository: <TODO: kit appstreaming applications container URL>

helm/kit-appstreaming-manager/values.yaml

* + - host: api.<private DNS zone> ex: api.contoso-ov-kitappstreaming.net
    - backend\_csp\_args.base\_domain: <public DNS zone> ex: kitstreaming.iai-contoso.com
    - Global.ingress.annotations.<>:

# CORS-related annotations

nginx.ingress.kubernetes.io/enable-cors: "true"

nginx.ingress.kubernetes.io/cors-allow-origin: "\*"

nginx.ingress.kubernetes.io/cors-allow-methods: "GET, PUT, POST, DELETE, PATCH, OPTIONS"

nginx.ingress.kubernetes.io/cors-allow-headers: "DNT,User-Agent,X-Requested-With,If-Modified-Since,Cache-Control,Content-Type,Range,Authorization,Accept,Origin"

nginx.ingress.kubernetes.io/cors-expose-headers: "Content-Length,Content-Range"

nginx.ingress.kubernetes.io/cors-allow-credentials: "true"

nginx.ingress.kubernetes.io/cors-max-age: "86400"

# SSL/TLS settings

nginx.ingress.kubernetes.io/ssl-redirect: "true"

nginx.ingress.kubernetes.io/hsts: "true"

nginx.ingress.kubernetes.io/hsts-max-age: "31536000"

nginx.ingress.kubernetes.io/hsts-include-subdomains: "true"

#### Internal ingress controller (helm/nginx-ingress-controller)

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

...

service.beta.kubernetes.io/azure-load-balancer-resource-group: rg\_contoso\_omniverse

...

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)

o kubectl get svc -n nginx-ingress-controller

#### 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.

Execute the following:

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

#### GPU Operator (helm/gpu-operator)

Execute the following to deploy the gpu-operator helm chart.

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

#### Memcached (helm/memcached)

Execute the following to deploy the memcached helm chart.

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

#### 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 the desired values:

SUBSCRIPTION\_ID="YOUR\_SUBSCRIPTION\_ID"

EXTERNALDNS\_NEW\_SP\_NAME="OVKitStreamingExternalDnsServicePrincipal" # name of the service principal (This should be unique)

AZURE\_DNS\_ZONE\_RESOURCE\_GROUP="rg\_contoso\_omniverse # name of resource group where dns zone is hosted

AZURE\_DNS\_ZONE="kitstreaming.iai-contoso.com" # DNS zone name like example.com or sub.example.com

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

Example output:

user@contoso ov-kitappstreaming-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](https://aka.ms/azadsp-cli%20Client) 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:

{

"tenantId": "<Your-tentent-ID>",

"subscriptionId": "<your-subscription-id>",

"resourceGroup": "<dns-zone-rg>",

"aadClientId": "<client-id>",

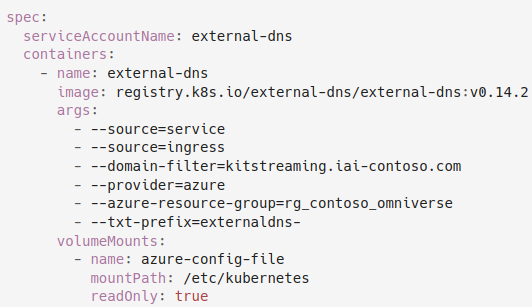
"aadClientSecret": "<client-secret>"

}

Execute the following:

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 and edit appropriate values for --domain-filter and --azure-resource-group.

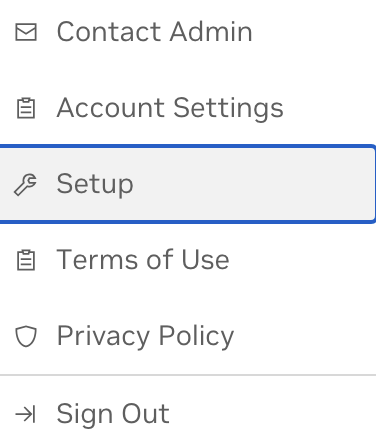


Apply the External DNS Manifest.

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

#### Create required secrets

Get your NGC API token by visiting ngc.nvidia.com and selecting “Setup” near the top-right corner of the webpage.



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

### Omniverse Kit App Streaming Services

#### Streaming (helm/kit-appstreaming-manager)

Check helm/kit-appstreaming-manager/values.yaml file and update DNS names accordingly:

...

ingress:

host: api.<your-private-domain> <--- your private domain here

ex: api.contoso-ov-kitappstreaming.net

className: internal-nginx...

To enable/disable WSS:

...

backend\_csp\_cls: "nv.svc.streaming.\_csp.Generic"

backend\_csp\_args:

enable\_wss: true <----- set to true/false to enable/disable WSS. base\_domain: "kitstreaming.iai-contoso.com" <--- add public DNS here (Leave blank if disabling WSS)

...

Deploy helm/kit-appstreaming-manager by running:

helm upgrade –install --namespace omni-streaming -f helm/kit-appstreaming-manager/values.yaml streaming omniverse/kit-appstreaming-manager

#### Applications (helm/kit-appstreaming-applications)

Check helm/kit-appstreaming-applications/values.yaml file and update DNS names accordingly:

...

ingress:

host: api.contoso-ov-kitappstreaming.net <--- your private domain here

...

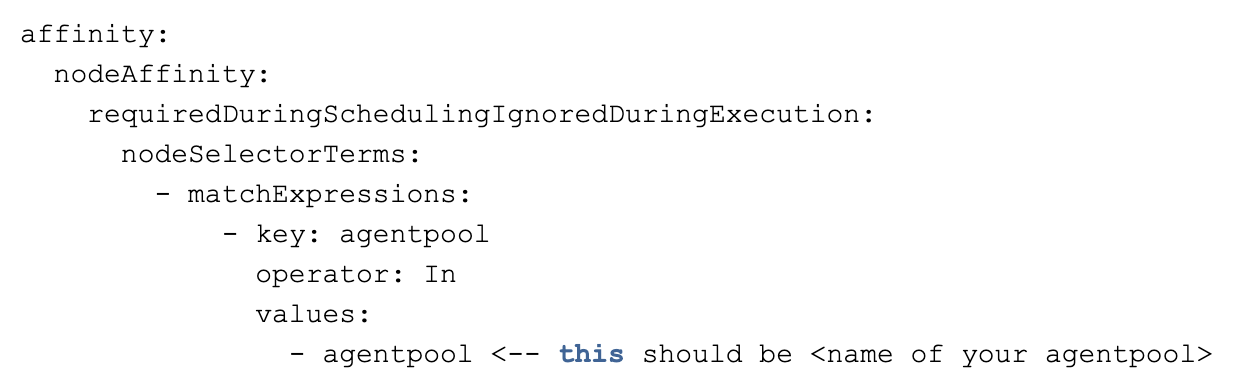
Deploy helm/kit-appstreaming-applications by running:

helm upgrade --install --namespace omni-streaming -f helm/kit-appstreaming-applications/values.yaml applications omniverse/kit-appstreaming-applications

#### RMCP (helm/kit-appstreaming-rmcp)

Check helm/kit-appstreaming-rcmp/values.yaml file and update DNS names accordingly.

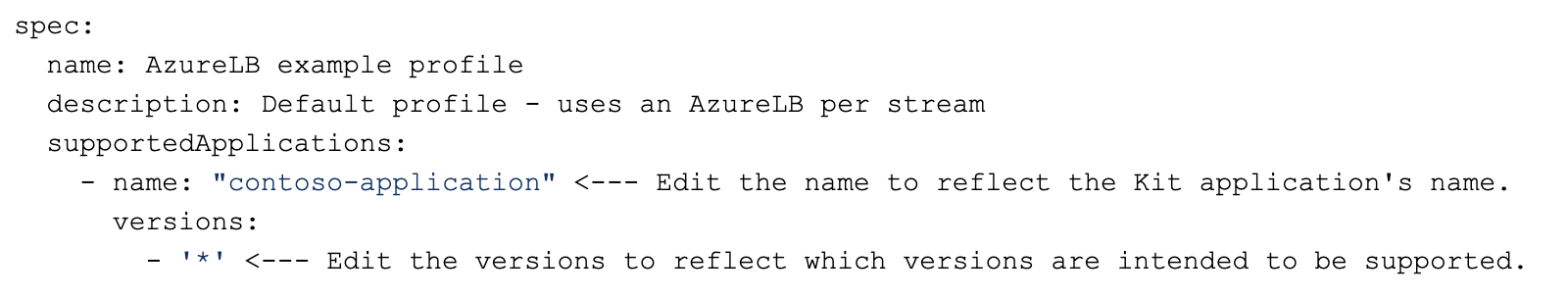
Change or comment out affinity from system to agentpool



helm upgrade –install --namespace omni-streaming -f helm/kit-appstreaming-rmcp/values.yaml rmcp omniverse/kit-appstreaming-rmcp

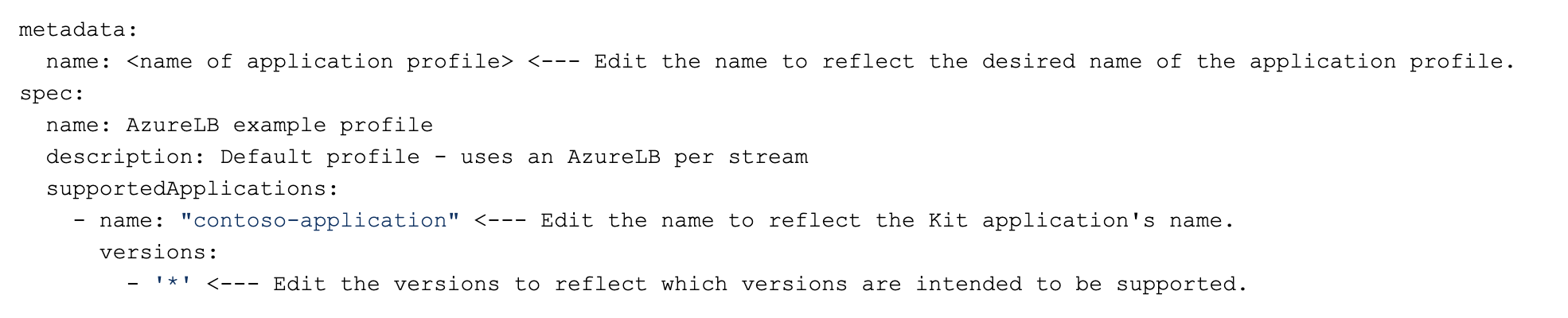
#### Deploy the custom streaming resources (manifests/omniverse/azure)

To enable WSS, open manifests/omniverse/azure/application-profile-wss.yaml and edit the following sections listed below:



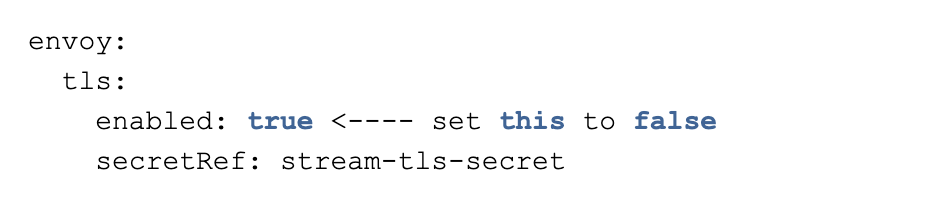
Run kubectl apply -n omni-streaming -f application-profile-wss.yaml.

Make the following changes in manifests/omniverse/azure/application-profile-azurelb.yaml.



To disable WSS, make the following changes in

manifests/omniverse/azure/application-profile-azurelb.yaml:



Then run:

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

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

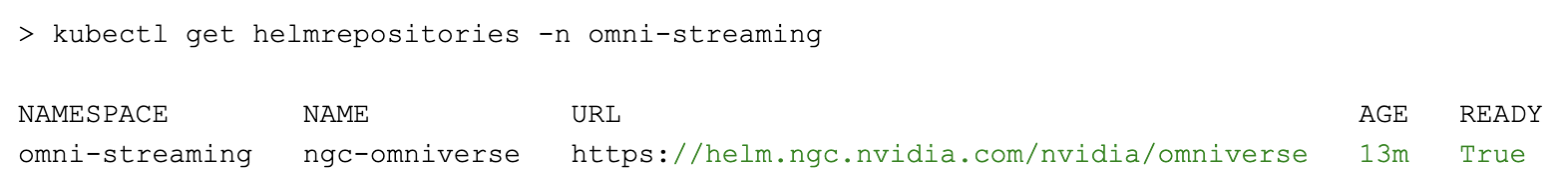
kubectl apply -n omni-streaming -f application-version.yaml

#### Deploy HelmRepository (manifests/helm-repostiories)

Execute the following:

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

This should (eventually) show READY: True in the output of:



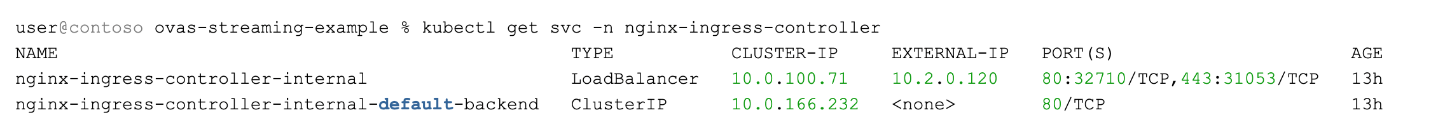
#### Create private DNS record for ingress controller

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

A screenshot of a computer

Description automatically generated

[api.contoso-ov-kitappstreaming.net](http://api.ovas-streaming.net/) -> private external ip of ingress controller LB service (e.g. 10.2.0.120 shown below)



#### Create public DNS entry for App Gateway

Navigate to the Public DNS Zone (ex. kitstreaming.iai-contoso.com) and create an A Recordset that points to the IP address of the Public IP address used in your resource group.

### Validate Omniverse Kit App Streaming End Points via Swagger UI

Before proceeding with installing web front end, let’s validate the backend services are functioning nominally using the [REST API Documentation Tool | Swagger UI.](https://swagger.io/tools/swagger-ui/)

Using the domain name URL where the Omniverse Kit App Streaming is running, append “/application/docs” or “/streaming/docs” to that URL to get respective Swagger UI web page.

For example:

If domain URL is “https://ignite. streaming.iai-contoso.com” then

* ‘appserver’ = https://ignite.streaming.iai-contoso.com/applications/docs
* ‘streamingServer’ = https://ignite.streaming.iai-contoso.com/streaming/docs

Now validate the backend running using the following steps, in the order presented.

1. GET ${appServer}/cfg/apps

=> Expect return list containing an entry ‘app\_id‘ = “omni-viewer-msft"

1. GET ${appServer}/cfg/apps/${appId}/versions
   1. Enter “app\_id” = omni-viewer-msft

=> Expect return of ‘appVersion’ = “106.1.0”

1. GET ${appServer}/cfg/apps/${appId}/versions/${appVersion}/profiles
   1. Enter “app\_id” = “omni-viewer-msft"
   2. Enter ‘appVersion’ = “106.1.0”

=> Expect return of “azurelb-wss”

1. POST ${streamServer}/streaming/stream

Create a stream

=> Returns a ‘session\_id’ in GUID format

1. GET ${streamServer}/streaming/stream

=> Returns streaming sessions

1. GET ${streamServer}/streaming/stream/${sessionId}
   1. Enter ‘session\_id” = <GUID>

=> Returns information about the stream

(may need to poll until returns with ‘condition’ == ‘ready’)

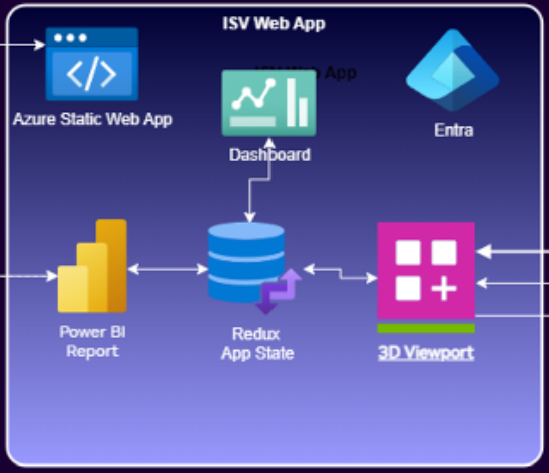
Note: Should have “routes” JSON filled in.

1. DELETE ${streamServer}/streaming/stream
   1. The session\_id “id”: <GUID> needs to be supplied in the JSON request body.
2. GET ${streamServer}/streaming/stream

=> Verify streaming session no longer listed.

## ISV Web App deployment

An opinionated example kit app and web app are provided for reference as a starting point; with ISV developers expected and encouraged to further customize these examples for their own use cases. This section expands on the ISV Web App portion of the overall architecture diagram.



### Deploying Web Client Application in Azure

This section focuses on installing and deploying the bundled sample web client front-end. See the Appendix for information on developing this custom ISV web client sample. For the purposes of this sample, presume that the web client is going to be deployed as an Azure Static Web App (the ‘www’ icon in above diagram).

The ‘Dashboard’ provides the top-level index.html and React ‘App.tsx’ that presents two panels with the P*ower BI Repor*t (IoT) on left overlayed on the Omniverse powered *3D Viewport* on right, which are kept in sync by the Redux based web App State.

There is also a developer focused “test” panel available in ‘debug’ sub-folder for testing end-to-end streaming and messaging functionality between front-end and back-end for both local streaming and remote cloud streaming scenarios. Please see Appendix for more information.

### Power BI React Component Configuration

#### Azure Apps Registration

Add permissions (admin consent not required) required to allow associated services access.

* API Permissions tab
  + Power BI Service
    - Report.Read.All
    - Dataset.Read.All
  + Microsoft.EventHubs
    - user\_impersonation
  + Azure Storage
    - user\_impersonation

Note: resulting value from scope *'*[*https://analysis.windows.net/powerbi/api/Report.Read.All*](https://analysis.windows.net/powerbi/api/Report.Read.All)*'* and *activeAccount* used to retrieve JWT token.

#### Event Hubs

Add role assignments to the Event Hubs Namespace and Event Hub itself via Access Control (IAM):

* Azure Event Hubs Data Receiver
* Azure Events Hub Data Sender

#### Power BI Workspace Settings

Ensure that in Power BI Workspace Settings the License Info -> License Configuration has been assigned for this Azure tenant.

* Current license:  **Fabric capacity**
* License Capacity: *name* with SKU set to number of different users in region using this capacity, e.g. **name: ignitecapacityaio,** **SKU: F4, Region: West US 3**
* Semantic model storage format: **Small semantic model storage format**

#### Power BI Data Connections

Within Power BI workspace connects Power BI custom report to two Azure Data Explorer (ADX) datasets: *digital\_twins*, and *FluidCell*.

The Power BI report was modified to have a transparent background so it overlays what is behind it (e.g. a 3D Viewport).  
  
Information on embedding a custom Power BI report may be found at [Power BI embedded analytics Client APIs | Microsoft Learn](https://learn.microsoft.com/en-us/javascript/api/overview/powerbi/) . The specifics for how this was done for this example may be found in the ‘EmbedPowerBIComponent’ typescript file in this project’s GitHub Repo. Microsoft copilot may also clarify and generate appropriate code.

In the *PageLayout* typescript file, the <EmbedPowerBIComponent> arguments values need to be examined closely.

* *PowerBIVisualReportEmbedURL*
  + This is actual embedded URL copied from the Power BI app “File | Embed” menu item embed content link from the “Securely embed this report in a website or portal” dialog.
  + Remove “&autoAuth=True” from the embed URL before pasting; otherwise, will receive an error.
* Copy the value portion of the “?reportId=value” of the *PowerBIVisualReportEmbedURL* and paste it into the *powerBIVisualReportId.*
* The *powerBITable* value should be “digital\_twins” to match the ADX Data Source name.
* The *powerBIColumn* value should be “module”, which needs to match the Power BI Slicer (filter) column name “module” in the “digital\_twins” ADX Data table for this report.
* The *powerBIVisualName* is more complicated to obtain because it needs to be retrieved at runtime from the “module” Power BI Slicer in the Power BI Report. One way to accomplish this is to add an ‘alert(visual.name)’ statement in the eventHandlers when visual.type is ‘slicer’, then paste the returned value, which looks more like an id, are the value for this argument.

Note; The organization and description of these tables may be found in the associated Azure Data Explorer (ADX) time-series database Query page in the Azure Portal.

### Event Hub React Component Configuration

At the top of the screen selected Event Hub stream values received from the Azure IoT Operations are being updated. This is configured via the <EventHubStreamComponent> argument values:

* The eventHubsResourceName
* The eventHubName
* The consumerGroup

For this to work, a new token needs to be retrieved with the proper scope, which is specified in more detail in EventHubStreamCompoment typescript file in this project’s rep, which subcribe to processInitialize(), processClose(), and processEvents() methods via a WebSocket connection.

Note: The processEvents() callback performs a check on the current pressure and updates the Redux based global state.

For a more general tutorial on how to does this, please see [Send or receive events using JavaScript - Azure Event Hubs | Microsoft Learn](https://learn.microsoft.com/en-us/azure/event-hubs/event-hubs-node-get-started-send?tabs=passwordless%2Croles-azure-portal)

### 3D Viewport React Component Configuration

The 3D Viewport React Component is configured via changing the web app’s global state variables. Hence, update the initial state of the Redux global state to reflect those initial values. The initial state values in the web-app/src/state/slice/serviceSlice.ts file should match the ones set in the ISV Custom Kit App Configuration section above.

appId: 'usd-viewer-msft',

version: '106.1.0',

profile: 'azurelb-wss',

### Azure Static Web Application Deployment

The web-app sub-folder in the GitHub repo includes everything in ISV Web App, including Power BI component, Redux component, and 3D Viewport Component. The Power BI React Component provides the integration between Power BI Reports and the Redux based global state manager via Redux Slices and Reducers. The 3D Viewport Component encapsulates the Omniverse Streaming functionality and likewise provides the integration with the Redux global state manager via Slices and Reducers.

#### Deploying a React App to Azure Static Web Apps using Azure Portal

##### Create a New Azure Static Web App

1. Log in to the Azure Portal at <https://portal.azure.com/>.
2. Click on **Create a resource** and search for **Static Web Apps**.
3. Click on **Static Web Apps** and then click on **Create**.
4. Fill in the required details:
   1. **Subscription**: Select your Azure subscription.
   2. **Resource group**: Create a new resource group or select an existing one.
   3. **Name**: Enter a name for your Static Web App.
   4. **Account plan**: Select a plan that suits your needs.
5. Click on **Review + create** and then **Create**.

##### Configure the Static Web App

1. Once the deployment is complete, navigate to your newly created Static Web App.
2. Click on **Configuration** under the **Settings** section.
3. Configure the following settings:
   1. **Default document**: Set to index.html.
   2. **Default directory**: Set to the root directory of your React app (usually public or build).
   3. **Route rules**: Configure any route rules as needed for your app.

##### Deploy Your React App

1. Click on **Deployment** under the **Settings** section.
2. Click on **Connect to GitHub** (or your preferred repository provider).
3. Follow the prompts to connect your repository to Azure.
4. Select the repository and branch that contains your React app.
5. Azure will automatically detect the build configuration and deploy your app.

##### Verify the Deployment

1. Once the deployment is complete, navigate to the **Overview** section of your Static Web App.
2. Click on the **Site URL** to verify that your React app is deployed and accessible.

## Final End-to-End Validation (Raj/Rajesh/David/Drew/Rama/Paul)

### 3D Viewer Component

This step validates that pod containing and Envoy proxy is running in Kubernetes can be accessed remotely via Azure cloud using Omniverse Kit App Streaming with WebRTC streaming Omniverse Viewport to the bundled React app running in an Edge or Chrome browser. See GitHub Repo “testing” panel.

Web client users should be able to navigate around in the 3D Viewport panel using mouse controls or WSAD movement. [Viewport Navigation — Omniverse Extensions latest documentation](https://docs.omniverse.nvidia.com/extensions/latest/ext_core/ext_viewport/navigation.html) When an object in the 3D Viewport scene is “picked” (left mouse click), that information is sent to the Power BI Report. Note that only pre-specified objects in the scene are pickable.

### Power BI Component (Drew)

*Signing in should present Power BI report, with IoT data streaming into report tables.*

### Validation of Complete ISV Web App Deployment (MSFT)

*How to validate the two panels together in top-level web app. TBD*