

# Configuración de Ingress Controller y MTLS en API Management.

Detalle de recursos Azure Cloud:

Recurso Cloud	Nombre	Ambiente	Descripción
Azure Key Vault	kvprimainfdesa01	Desarrollo	Se generó los certificados para TLS y MTLS.
Azure Key Vault	kvprimainfcert01	Certificación	Se generó los certificados para TLS y MTLS.
Azure Key Vault	kvprimainfprod01	Producción	Se generó los certificados para TLS y MTLS.
Azure Private DNS	primadesa.com.pe	Desarrollo	Dominio privado y se registro los subdominios para los ingress del aks dm y bff.
Azure Private DNS	primacert.com.pe	Certificación	Dominio privado y se registro los subdominios para los ingress del aks dm y bff.
Azure Private DNS	primaprod.com.pe	Producción	Dominio privado y se registro los subdominios para los ingress del aks dm y bff.
Azure API Management	apiMGMT-ApiLayer-Desa	Desarrollo	API Manager para las APIs de Negocio y Experiencia de Prima AFP.
Azure API Management	apiMGMT-ApiLayer-Cert	Certificación	API Manager para las APIs de Negocio y Experiencia de Prima AFP.
Azure API Management	apiMGMT-ApiLayer-Prod	Producción	API Manager para las APIs de Negocio y Experiencia de Prima AFP.

Generación de certificados para https y mtls:

Tener en cuenta que los certificados son diferentes tanto para el AKS DM y BFF.

1. Los certificados son auto firmados usando azure key vault, por lo cual los certificados son generados en Key Vault tanto para https y mtls.
2. Generar key y crt para https, para lo cual puede descargar el certificado en formato pfx:
  - a. Para generar el crt ejecutar el siguiente comando:

```
i. openssl pkcs12 -in kvprimainfprod01-azingressbfftls-  
20220708.pfx -clcerts -nokeys -out aks-ingress-tls.crt
```

- b. Para generar el key ejecutar el siguiente comando:

```
i. openssl pkcs12 -in kvprimainfprod01-azingressbfftls-  
20220708.pfx -out azingressbfftls.pem -nodes  
openssl rsa -in azingressbfftls.pem -out aks-ingress-tls.  
key
```

- c. Crear el secret en kubernetes en base al key y crt generado en el paso anterior, debe crearlo en los namespaces donde se encuentre el ingress de sus servicios:

```
kubectl create secret tls secret-azingressbfftls --namespace  
bff --key=aks-ingress-tls.key --cert=aks-ingress-tls.crt
```

3. Generar el crt para MTLS

- a. Ejecutar el siguiente comando:

```
i. openssl pkcs12 -in kvprimainfprod01-azingressbffmtls-  
20220708.pfx -clcerts -nokeys -out ca.crt
```

- b. Ejecutar el siguiente comando para crear el secret para mtls en kubernetes, debe crear en los namespaces donde sean necesarios:

```
i. kubectl create secret generic ca-secret --from-file=ca.  
crt=ca.crt -n bff
```

## 1. Instalación de componentes de Ingress Controller en AKS

Detalle de los Ingress Controller configurados:

Hostname Ingress Controller	IP	AKS	Ambiente	Helm Chart
azingressbff.primadesa.com.pe	10.104.31.253	aksServiceCanal-Desa	Desarrollo	ingress-nginx-4.0.13
azingressmsdm.primadesa.com.pe	10.105.9.152	aksServiceLayer-Desa	Desarrollo	ingress-nginx-4.0.13
azingressbff.primacert.com.pe	10.100.30.30	aksServiceCanal-Cert	Certificación	ingress-nginx-4.0.13
azingressdm.primacert.com.pe	10.101.41.150	aksServiceLayer-Cert	Certificación	ingress-nginx-4.0.13
azingressbff.primaprod.com.pe	10.96.25.150	aksServiceCanal-Prod	Producción	ingress-nginx-4.0.13
azingressdm.primaprod.com.pe	10.97.41.150	aksServiceLayer-Prod	Producción	ingress-nginx-4.0.13

Añadir nginx repositorio de helm y actualizar el repositorio de helm:

```
helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx  
helm repo update
```

Archivo values.yaml:

**Importante:** Al crear un nuevo ingress controller, debe insertar una ip valida:

```
controller:  
  service:  
    loadBalancerIP: 10.104.31.253  
    annotations:  
      service.beta.kubernetes.io/azure-load-balancer-internal: "true"  
    extraArgs:  
      default-ssl-certificate: "ingress-basic/secret-azingresbff-tls"
```

```
controller:
  service:
    loadBalancerIP: 10.104.31.253
    annotations:
      service.beta.kubernetes.io/azure-load-balancer-internal: "true"
  extraArgs:
    default-ssl-certificate: "ingress-basic/secret-azingresbff-tls"
```

Para instalar el ingress controller ejecutar el siguiente comando:

```
helm install nginx-ingress ingress-nginx/ingress-nginx \
--create-namespace \
--namespace ingress-basic \
--version 3.3.0 \
-f values.yaml \
--set controller.replicaCount=3 \
--set controller.nodeSelector."kubernetes\.io/os"=linux \
--set defaultBackend.nodeSelector."kubernetes\.io/os"=linux \
--set controller.admissionWebhooks.patch.nodeSelector."kubernetes\.io
/os"=linux \
--set controller.service.annotations."service\.beta\.kubernetes\.io
/azure-load-balancer-health-probe-request-path"/=/healthz \
--set controller.watchIngressWithoutClass=false \
--set controller.ingressClassResource.default=true
```

## 2. Actualización de componentes de Ingress Controller en AKS

Para realizar el upgrade mediante Helm, debe tener el siguiente archivo [values.yaml](#) con los valores indicados:

values.yaml AKS Canal Layer

```
controller:
  service:
    loadBalancerIP: 10.96.25.150
    annotations:
      service.beta.kubernetes.io/azure-load-balancer-internal: "true"
  extraArgs:
    default-ssl-certificate: "ingress-basic/secret-azingresbff-tls"
```

values.yaml AKS Service Layer

```

controller:
  service:
    loadBalancerIP: 10.97.41.150
  annotations:
    service.beta.kubernetes.io/azure-load-balancer-internal: "true"
  extraArgs:
    default-ssl-certificate: "ingress-basic/secret-azingressdm-tls"

```

Ejecutar el siguiente comando para añadir nginx al repositorio helm y actualizar:

```

helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx
helm repo update

```

Ejecutar el siguiente comando para realizar el upgrade del ingress controller, tener en cuenta los siguientes argumentos y deben indicar en base al ingress controller existente:

- Release: nginx-ingress
- Namespace: ingress-basic
- Replicate Count: 3

```

helm upgrade nginx-ingress ingress-nginx/ingress-nginx \
--namespace ingress-basic \
--version 4.0.13 \
-f values.yaml \
--set controller.replicaCount=3 \
--set controller.nodeSelector."kubernetes\.io/os"=linux \
--set defaultBackend.nodeSelector."kubernetes\.io/os"=linux \
--set controller.admissionWebhooks.patch.nodeSelector."kubernetes\.io
/os"=linux \
--set controller.service.annotations."service\.beta\.kubernetes\.io
/azure-load-balancer-health-probe-request-path"/=/healthz \
--set controller.watchIngressWithoutClass=false \
--set controller.ingressClassResource.default=true

```

### 3. Validación y Ratificación de Ingress Controller en AKS

Validar que el release de helm fue actualizado, estado **deployed** en la columna **STATUS** y la versión 1.1.1 y contenga versión 4.0.13 en la columna chart.

```
helm ls -n ingress-basic
```

NAME	NAMESPACE	REVISION	UPDATED	STATUS	CHART	APP VERSION
nginx-ingress	ingress-basic	5	2022-07-15 20:46:22.255116287 -0500 -05	deployed	ingress-nginx-4.0.13	1.1.0

Esperar de 2 a 5 minutos a que los nuevos pods del ingress controller esten en estado Running.

```
kubectl get pods -n ingress-basic
```

```
[opc@azure-connect k8s]$ kubectl get pods -n ingress-basic
aks-helloworld-796448fcc4-kplk9           1/1    Running   0      64d
nginx-ingress-ingress-nginx-controller-9d47dfff6-dgdp6  1/1    Running   0      12d
nginx-ingress-ingress-nginx-controller-9d47dfff6-fzqnf  1/1    Running   0      12d
nginx-ingress-ingress-nginx-controller-9d47dfff6-rt8wm  1/1    Running   0      12d
[opc@azure-connect k8s]$
```

Seleccionar unos de los pods del anterior comando y verificar que se actualizó a la versión 1.19.9:

```
kubectl -n ingress-basic exec <pod> -- /nginx-ingress-controller --version
```

```
[opc@azure-connect k8s]$ kubectl -n ingress-basic exec nginx-ingress-ingress-nginx-controller-9d47dfff6-dgdp6 -- /nginx-ingress-controller --version
NGINX Ingress controller
  Release:        v1.1.0
  Build:          cacbee86b6ccc45bde8ffc184521bed3022e7dee
  Repository:     https://github.com/kubernetes/ingress-nginx
  nginx version:  nginx/1.19.9
-----
```

#### 4. Reversión mediante rollback de Ingress Controller en AKS

Se debe listar los chart releases instalados e identificar los ingress controller:

```
helm ls -n ingress-basic
```

Obtener el historial de instalaciones para el correspondiente release:

```
helm history <nombre-release> -n ingress-basic
```

```
[opc@azure-connect k8s]$ helm history nginx-ingress -n ingress-basic
REVISION      UPDATED             STATUS      CHART          APP VERSION      DESCRIPTION
1            Mon Oct  5 12:56:40 2020  superseded  ingress-nginx-3.3.0  0.35.0  Install complete
2            Wed Jun  1 22:45:24 2022  superseded  ingress-nginx-4.0.13  1.1.0   Upgrade complete
3            Thu Jul 14 23:18:41 2022  superseded  ingress-nginx-4.0.13  1.1.0   Upgrade complete
4            Fri Jul 15 09:32:08 2022  superseded  ingress-nginx-4.0.13  1.1.0   Rollback to 2
5            Fri Jul 15 20:46:22 2022  deployed   ingress-nginx-4.0.13  1.1.0   Upgrade complete
[opc@azure-connect k8s]$
```

Identificar el ultimo release instalado y revertir al penúltimo numero de revisión deseado:

```
helm rollback <nombre-release> 4 -n ingress-basic
```

#### 5. Eliminar Ingress Controller en AKS

Ejecutar el siguiente comando:

```
helm uninstall <release> -n ingress-basic
```

## 6. Configurar cabeceras en Ingress Controller en AKS

Crear configmap en el namespace donde se encuentra instalado el ingress controller:

Configmap referencial: **configmap-custom-headers.yaml**

```
apiVersion: v1
data:
  Strict-Transport-Security: "max-age=31536000; includeSubDomains"
  Content-Security-Policy: "default-src 'self'"
  X-XSS-Protection: "1; mode=block"
  X-Frame-Options: "DENY"
  X-Content-Type-Options: "nosniff"
kind: ConfigMap
metadata:
  name: custom-headers
  namespace: ingress-basic
```

```
kubectl apply -f configmap-custom-headers.yaml -n ingress-basic
```

Actualizar configmap ingress-nginx-controller:

```
apiVersion: v1
data:
  add-headers: "ingress-basic/custom-headers"
  ssl-protocols: "TLSv1.2 TLSv1.3"
kind: ConfigMap
metadata:
  name: ingress-nginx-controller
  namespace: ingress-basic
  labels:
    app.kubernetes.io/name: ingress-nginx
    app.kubernetes.io/part-of: ingress-nginx
```

```
kubectl apply -f configmap.yaml -n ingress-basic
```

## 7. Añadir los Backend en el API Management:

1. En el API Management, seleccionar **Backend** en API y **Add**:

The screenshot shows the Azure API Management service interface. The top navigation bar includes a gear icon, the service name "apiMGMT-ApiLayer-Desa | Backends", a star icon, and three dots. Below the navigation is a search bar with placeholder "Search (Ctrl+ /)" and a toolbar with "Add", "Columns", "Refresh", and a feedback link. A descriptive text block states: "Backends let you manage resources representing backend services of your APIs. You can reference backend resources from your API definitions." On the left, a sidebar menu lists various management options: Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Events, Settings (selected), Properties, Locks, APIs (selected), APIs, Products, Subscriptions, Named values, and Backends (highlighted with a yellow box). The main content area displays a table of backends:

Name	Description	Type
api-devprimatestpe		Custom URL
azingressbff	Ingress Controller de los Microservicios B...	Custom URL
azingressdm	PoC	Custom URL
azingressmsdm	Ingress Controller de los Microservicios D...	Custom URL

2. Añadir los detalles del Backend y finalmente click en **Create**

## Backend

...

API Management service

Name \*

azingressdm02

Description

Backend APIs

Type \*

Custom URL Azure resource Service Fabric

Runtime URL \* ⓘ

https://azingressbff.primadesa.com.pe

Validate certificate chain ⓘ

Validate certificate name ⓘ

### Authorization credentials

API Management can use query parameters, client certificates, the Authorization header, or other headers to define the authentication mechanism.

Headers Query Client certificates

Name

Value

Key

Select named value

Schema ⓘ

Authorization header schema. For example, Basic.

Parameter ⓘ

For example, Username:Password

Create

8. Añadir los certificados al API Management para la configuración de MTLS:

1. En API Management, en Security seleccionar Certificates y elegir Certificates y Add:

a.

Subject	Thumbprint	Expires on	Key vault name	Secret name	Key vault fetch status
CN=primatest.pe	63307CE09A5D20F0C7A5ED21...	12/3/2022	kvprimades01	api-dev-primatest-pe	<span style="color: green;">✓ Success</span>
CN=Prima AFP	BBB4320BC5DDA48433B4AB9...	6/2/2023	kvprimainfdesa01	azingressbfffmTLS	<span style="color: green;">✓ Success</span>
CN=Prima Seguros	C4C3D30BDE6E482832755F5D...	5/19/2023	kvprimainfdesa01	azingressdmtls	<span style="color: green;">✓ Success</span>
CN=Prima AFP	B70C838CECACBCFAFB859F48...	5/26/2023	kvprimainfdesa01	azingressmsdmTLS	<span style="color: green;">✓ Success</span>

2. En Client certificates:

- Id: azingressbfffmTLS
- Certificate: Key Vault
- Certificate key vault id: Seleccionar el certificado correspondiente para mTLS según el ingress controller y ambiente.
- Finalmente en Add.
- Home > apiMGMT-ApiLayer-Desa >

## Client certificates

API Management service

**Id \***  
zingressbfffmTLS01

**Certificate**  
 Key Vault  Custom

**Certificate key vault id \***  
kvprimainfdesa01

**Client identity \*** ⓘ  
System assigned identity

**Add** **Cancel**

### Select certificate from Azure ...

Subscription \*  
Prima - Desarrollo

Key vault \*  
kvprimainfdesa01

Create new key vault

Certificate \*  
zingressbfffmTLS (Thumbprint: BBB4320BC5DDA48433B4AB9...)

Create new

**Select** **Cancel**

9. Cambios en manifest de ingress:

1. Cambio en manifest con version v1beta1:

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  annotations:
    nginx.ingress.kubernetes.io/client-body-buffer-size: 24m
    nginx.ingress.kubernetes.io/proxy-body-size: 24m
    nginx.ingress.kubernetes.io/backend-protocol: HTTP
    nginx.ingress.kubernetes.io/use-regex: "true"
    nginx.ingress.kubernetes.io/rewrite-target: /$1
    nginx.ingress.kubernetes.io/auth-tls-verify-client: "on"
    nginx.ingress.kubernetes.io/auth-tls-secret: "account/ca-secret"
  name: dm-account-statement-information-ing
  namespace: account
spec:
  ingressClassName: nginx
  tls:
    - hosts:
      - azingressdm.primaprod.com.pe
      secretName: secret-azingressdm-tls
  rules:
    - host: azingressdm.primaprod.com.pe
      http:
        paths:
          - backend:
              serviceName: dm-account-statement-information-service
              servicePort: 8080
            path: /dm-account-statement-information/(.*)
```

2. Cambio en manifest con version v1:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  annotations:
    nginx.ingress.kubernetes.io/auth-tls-secret: authentication/ca-secret
    nginx.ingress.kubernetes.io/auth-tls-verify-client: "on"
    nginx.ingress.kubernetes.io/backend-protocol: HTTP
    nginx.ingress.kubernetes.io/client-body-buffer-size: 24m
    nginx.ingress.kubernetes.io/proxy-body-size: 24m
    nginx.ingress.kubernetes.io/rewrite-target: /$1
    nginx.ingress.kubernetes.io/use-regex: "true"
  name: dm-authentication-ing
  namespace: authentication
spec:
  ingressClassName: nginx
  rules:
  - host: azingressdm.primacert.com.pe
    http:
      paths:
      - backend:
          service:
            name: dm-authentication-service
            port:
              number: 8080
            path: /dm-authentication/(.*)
            pathType: ImplementationSpecific
      tls:
      - hosts:
        - azingressdm.primacert.com.pe
        secretName: secret-azingressdm-tls
```