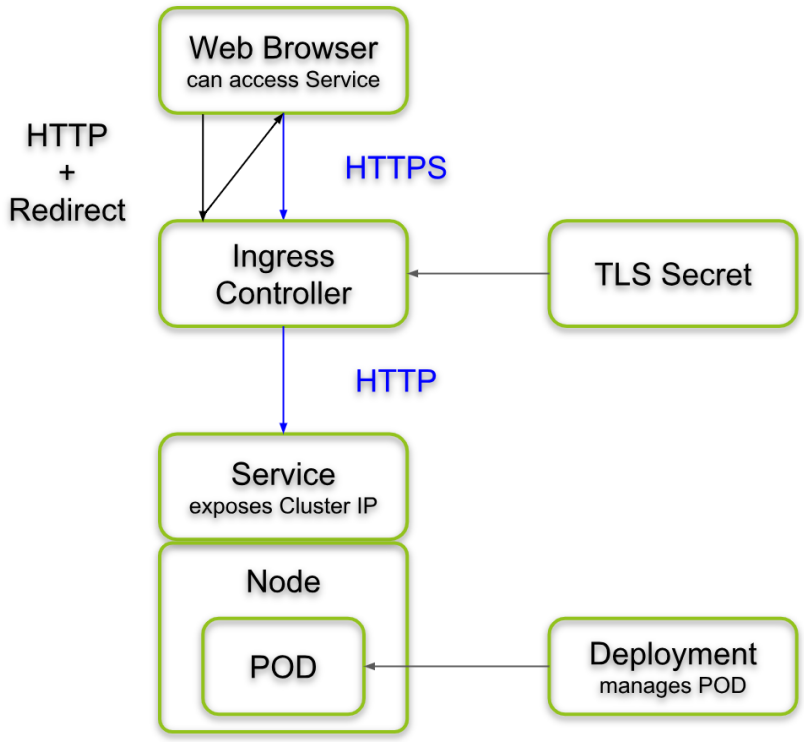
Adição de Certificado HTTPS Letsencript no AKS

Descrição

Este artigo descreve a adição de um "Ingress Cntroler" como LB para as aplicações e o "Cert-Manager" para a utilização do certificado gratuito "Let's Encrypt", como renovação automática.

Ilustração Simplificada da arquitetura:



Passo a passo

Pré-requisitos:

- az-cli
- kubectl
- helm cli

Instalação Centos e RedHat

```
rpm --import https://packages.microsoft.com/keys/microsoft.asc
sh -c 'echo -e "[azure-cli]\nname=Azure CLI\nbaseurl=https://packages.microsoft.com/yumrepos/azure-
cli\nenabled=1\ngpgcheck=1\ngpgkey=https://packages.microsoft.com/keys/microsoft.asc" > /etc/yum.repos.d/azure-
cli.repo'
yum install azure-cli
az aks install-cli
wget https://get.helm.sh/helm-v2.14.1-linux-amd64.tar.gz
tar -zxvf helm-v2.*.tar.gz
mv linux-amd64/helm /usr/bin/helm
```

Criando uma IP estático para ser usado do Ingress(LB):

```
1 - Captura do "Resource Group name" com os recursos do Kubernetes em uma variável.
MC_myResourceGroup_myAKSCluster_eastus=$(az group list|grep MC|grep name|awk '{print $2}'|cut -c2-| sed 's/",
//g'|grep $RESOURCEGROUP)

2 - Criação Do IP publico estático.
az network public-ip create --resource-group $MC_myResourceGroup_myAKSCluster_eastus --name $AKSPublicIPName --
allocation-method static --query publicIp.ipAddress -o tsv

3 - Captura do endereço IP criado em uma variável.
AKSPublicIP=$(az network public-ip list -g $MC_myResourceGroup_myAKSCluster_eastus --subscription
$SUBSCRIPTIONID --query "[?dnsSettings.domainNameLabel=='$AKSPublicIPName']"|grep ipAddress|awk '{print
$2}'|cut -c2-| sed 's/",//g')

4 - Captura do ID do IP em uma variavel.
PUBLICIPID=$(az network public-ip list --query "[?ipAddress!=null]|[?contains(ipAddress, '$AKSPublicIP')].[id]"
--output tsv)
```

Criando o Ingress Controller utilizando o Helm.

NAMESPACE=<nome_da_namespace_sendo_usada>

Ingress baseado em Nginx

```
helm install stable/nginx-ingress \
  --namespace $NAMESPACE \
  --set controller.replicaCount=2 \
  --set controller.nodeSelector."beta\.kubernetes\.io/os"=linux \
  --set defaultBackend.nodeSelector."beta\.kubernetes\.io/os"=linux \
  --set controller.service.loadBalancerIP="$AKSPublicIP" \
  --set controller.service.externalTrafficPolicy=Local
```

Obs: crie o Ingress no mesmo namespace das aplicações para evitar problemas.
Nossa Implantação utiliza-se do "replica set" para subir 2 replicas do Ingress Controller.

O resultado da implantação pode ser observado com o comando "kubectl get service -l app=nginx-ingress --namespace $NAMESPACE"

Instalando o cert-manager.

```
1 - Instala o "CustomResourceDefinition resources separately"
kubectl apply --validate=false -f https://raw.githubusercontent.com/jetstack/cert-manager/release-0.12/deploy
/manifests/00-crds.yaml

2 - Cria um namespace para o cert-manager
kubectl create namespace cert-manager

3 - Label o namespace do cert-manager para desabilitar o "resource validation".
kubectl label namespace cert-manager cert-manager.io/disable-validation=true

4 - Adiciona o Jetstack Helm repo
helm repo add jetstack https://charts.jetstack.io

5 - Update do cache do helm repo
helm repo update

6 - Instalar o cert-manager Helm chart
helm install \
--name cert-manager \
--namespace cert-manager \
--version v0.12.0 \
jetstack/cert-manager
```

Criando a autoridades de certificação (CAs).

ClusterIssuer

```
cat <<EOF > yamls/cluster-issuer-dev.yaml
apiVersion: cert-manager.io/v1alpha2
kind: ClusterIssuer
metadata:
  name: letsencrypt-prod
  namespace: eugenio-apps
spec:
  acme:
    server: https://acme-v02.api.letsencrypt.org/directory
    email: br.softwareservices@la.logicalis.com
    privateKeySecretRef:
      name: letsencrypt-prod
    solvers:
    - http01:
        ingress:
          class: nginx
EOF


kubectl apply -f yamls/cluster-issuer-dev.yaml
```

Criando rotas de entrada.

```
cat <<EOF > yamls/$PROJECTNAME-ingress.yaml
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: eugenio-apps-dev-ingress
  annotations:
    kubernetes.io/ingress.class: nginx
    cert-manager.io/cluster-issuer: letsencrypt-prod
    nginx.ingress.kubernetes.io/rewrite-target: /$1
spec:
  tls:
  - hosts:
    - meudominio.com
    secretName: tls-secret
  rules:
  - host: meudominio.com
    http:
      paths:
      - backend:
          serviceName: meuapp-service
          servicePort: 80
        path: /(.*)
EOF

kubectl apply -f yamls/$PROJECTNAME-ingress.yaml
```

Criando o objeto de certificado.

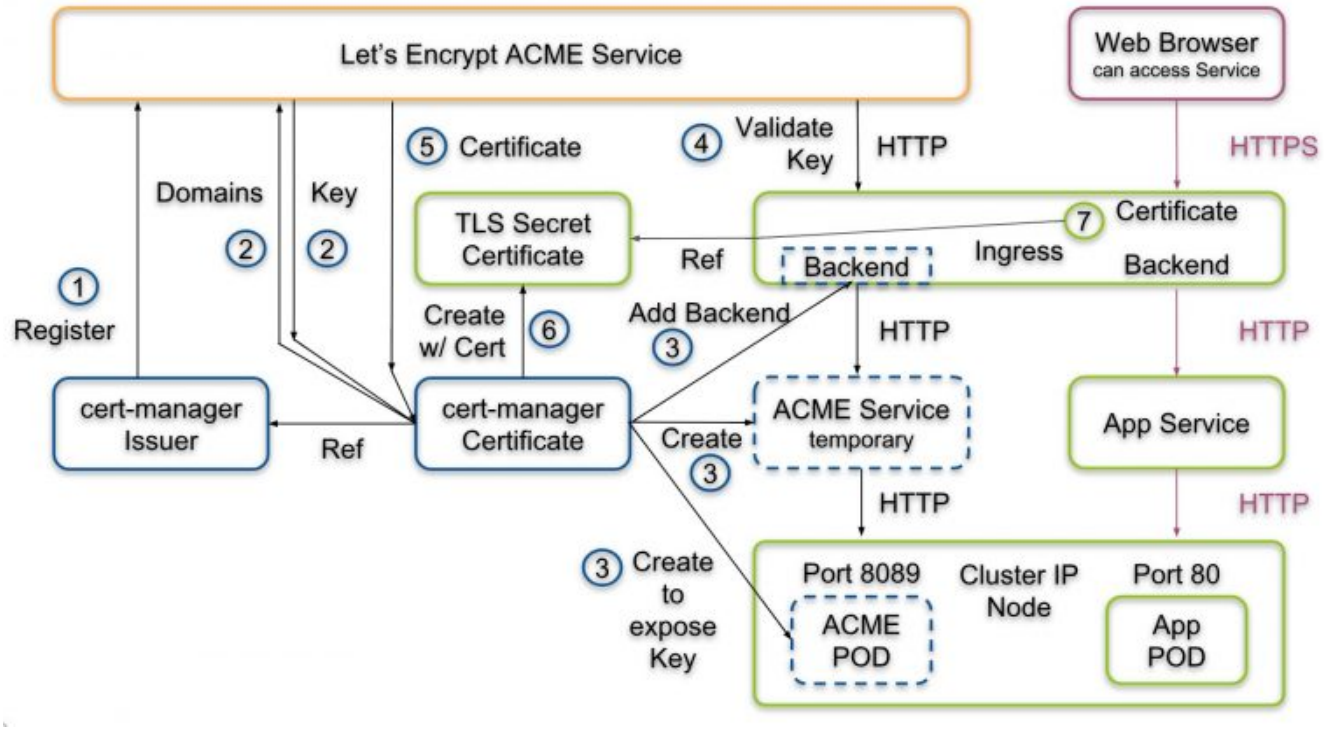
```
cat <<EOF > yamls/certificates.yaml
apiVersion: cert-manager.io/v1alpha2
kind: Certificate
metadata:
  name: tls-secret
  namespace: eugenio-apps
spec:
  secretName: tls-secret
  dnsNames:
  - meudominio.com
  acme:
    config:
    - http01:
        ingressClass: nginx
      domains:
      - meudominio.com
  issuerRef:
    name: letsencrypt-prod
    kind: ClusterIssuer
EOF

kubectl apply -f yamls/certificates.yaml
```

Obs: Após esta implantação, o DNS deve ser apontado para o IP criado neste passo a passo "echo $AKSPublicIP". O certificado só será gerado após o apontamento do DNS, sendo assim, no caso de uma aplicação já em produção, este processo pode ser realizado e testado apontando o IP no /etc/host da maquina local, mesmo com erro de certificado, e então agendar uma janela para o apontamento DNS.

Os YAML deste arquivo relacionados a rota e certificado precisam ser alterados com os domínios a serem usados, podendo ser adicionados novos domínios quando necessário, seguindo a sintaxe correta do arquivo.

Visão detalhada de como a magica acontece:



1 - O emissor do gerente de certificação se registra no serviço Let's Encripty.
2 - Após o registro bem-sucedido, o cert-manager enviará uma lista de domínios. O Let's Encripty enviará um caminho de URL e uma chave para o cert-manager.
3 - Imediatamente depois, o cert-manager cria três objetos temporários com a finalidade de validação do ACME:
- um POD ACME escutando na porta 8089
- um serviço ACME acessando o POD
- uma entrada de back-end no controlador de ingresso que aponta para a porta 8090

4 - Depois que os três objetos temporários estiverem totalmente funcionais, o serviço Let's Encripty ACME pode acessar o POD no domínio na URL e receberá a chave do POD. Com isso, o Let's Encripty verifica que a pessoa que solicitou uma assinatura de certificado tem controle total do domínio listado.
5 - Portanto, o domínio válido e o serviço ACME enviarão um certificado assinado para o cert-manager.
6 - Após o recebimento do certificado assinado, o cert-manager encapsulará o certificado e a chave privada em um segredo TLS.
7 - O controlador de ingresso agora pode acessar o certificado e a chave e usá-lo para finalizar as sessões HTTPS da Internet.

Referências

https://docs.microsoft.com/pt-br/azure/aks/ingress-static-ip

https://cert-manager.io/docs/installation/kubernetes/