# POC on Azure AD integration with Angular Application

## Changes at Angular end

* Follow instructions provided on this page to create angular app with azure AD integration:

<https://docs.microsoft.com/en-us/azure/active-directory/develop/scenario-spa-app-registration>

* After registration go to the quick start and choose the type of application then download the demo code.
* Copy you application id of your app which you have created and paste it in app.module.ts file .

*app.module.ts*

export function MSALInstanceFactory(): IPublicClientApplication {

  return new PublicClientApplication({

    auth: {

      clientId: '3099e36d-71e6-4ee5-ac58-f3497dc7cf1c',

      authority: 'https://login.microsoftonline.com/common',

      redirectUri: 'https://ng-ad-app2.eastus.cloudapp.azure.com/'

    },

    cache: {

      cacheLocation: BrowserCacheLocation.LocalStorage,

      storeAuthStateInCookie: isIE, // set to true for IE 11

    },

    system: {

      loggerOptions: {

        loggerCallback,

        logLevel: LogLevel.Info,

        piiLoggingEnabled: false

      }

    }

  });

}

*clientId*, *authority* and *redirectUri* should be mentioned as per your application.

## Create HTTPS ingress controller on AKS

1. #Import the images used by the Helm chart into your ACR

REGISTRY\_NAME=<Your REGISTRY NAME>

SOURCE\_REGISTRY=k8s.gcr.io

CONTROLLER\_IMAGE=ingress-nginx/controller

CONTROLLER\_TAG=v1.0.4

PATCH\_IMAGE=ingress-nginx/kube-webhook-certgen

PATCH\_TAG=v1.1.1

DEFAULTBACKEND\_IMAGE=defaultbackend-amd64

DEFAULTBACKEND\_TAG=1.5

az acr import --name $REGISTRY\_NAME --source $SOURCE\_REGISTRY/$CONTROLLER\_IMAGE:$CONTROLLER\_TAG --image $CONTROLLER\_IMAGE:$CONTROLLER\_TAG

az acr import --name $REGISTRY\_NAME --source $SOURCE\_REGISTRY/$PATCH\_IMAGE:$PATCH\_TAG --image $PATCH\_IMAGE:$PATCH\_TAG

az acr import --name $REGISTRY\_NAME --source $SOURCE\_REGISTRY/$DEFAULTBACKEND\_IMAGE:$DEFAULTBACKEND\_TAG --image $DEFAULTBACKEND\_IMAGE:$DEFAULTBACKEND\_TAG

1. Follow these steps to create ingress controller:

# Get the resource group name of the AKS cluster

az aks show --resource-group sm\_resources1 --name sm\_k8s\_cluster1 --query nodeResourceGroup -o tsv

# TEMPLATE - Create a public IP address with the static allocation

az network public-ip create --resource-group <output of previous command> --name <any name for AKS Public IP for Ingress> --sku Standard --allocation-method static --query publicIp.ipAddress -o tsv

e.g, MC\_sm\_resources1\_sm\_k8s\_cluster1\_eastus is output of previous command and myAKSPublicIPForIngress is name for AKS public IP.

This command returns public IP.

# Install Helm3 (if not installed)

brew install helm

# Add the official stable repository

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

helm repo add stable https://charts.helm.sh/stable

helm repo update

# Create a namespace for your ingress resources e.g., ingress-basic

kubectl create namespace ingress-basic

# Use Helm to deploy an NGINX ingress controller

helm install ingress-nginx ingress-nginx/ingress-nginx \

--namespace ingress-basic \

--set controller.replicaCount=2 \

--set controller.nodeSelector."beta\.kubernetes\.io/os"=linux \

--set defaultBackend.nodeSelector."beta\.kubernetes\.io/os"=linux \

--set controller.service.externalTrafficPolicy=Local \

--set controller.service.loadBalancerIP="<public IP returned from above command>"

Check if ingress controller created successfully:

# List Services with labels

kubectl get service -l app.kubernetes.io/name=ingress-nginx --namespace ingress-basic

# List Pods

kubectl get pods -n ingress-basic

kubectl get all -n ingress-basic

# Access Public IP from browser

http://<Public-IP-created-for-Ingress>

it should return 404

1. Generate domain for your external IP: either create custom domain or configure FQDN.

Configure FQDN:

Your FQDN will be of the form <CUSTOM LABEL>.<AZURE REGION NAME>.cloudapp.azure.com

follow these steps for the same:

# Public IP address of your ingress controller

IP=<Public-IP-created-for-Ingress>

# Name to associate with public IP address

DNSNAME="<your application’s DNS name>"

e.g., ng-ad-app2

# Get the resource-id of the public ip

PUBLICIPID=$(az network public-ip list --query "[?ipAddress!=null]|[?contains(ipAddress, '$IP')].[id]" --output tsv)

# Update public ip address with DNS name

az network public-ip update --ids $PUBLICIPID --dns-name $DNSNAME

# Display the FQDN

az network public-ip show --ids $PUBLICIPID --query "[dnsSettings.fqdn]" --output tsv

1. Create SSL certificate for domain:

openssl req -x509 -nodes -days 365 -newkey rsa:2048 \

-out aks-ingress-tls.crt \

-keyout aks-ingress-tls.key \

-subj "/CN=<your application domain name> /O=aks-ingress-tls"

e.g., CN= ng-ad-app2.eastus.cloudapp.azure.com

# Create Kubernetes secret for the TLS certificate

kubectl create secret tls aks-ingress-tls \

--namespace ingress-basic \

--key aks-ingress-tls.key \

--cert aks-ingress-tls.crt

1. Create app registration in Azure Active directory:

* Given any name to registration.
* Select ID tokens (used for implicit and hybrid flows).
* Select Accounts in any organizational directory (Multitenant).
* Give redirect URI as https://<your domain name>

e.g., https://ng-ad-app2.eastus.cloudapp.azure.com/

* Redirect URI should be eligible for Auth Code flow with PKCE.

1. Create Docker file as given below:

FROM node:12.14-alpine AS build

WORKDIR /usr/src/app

COPY package.json package-lock.json ./

RUN npm install

COPY . .

RUN npm run build

### STAGE 2: Run ###

FROM nginx:1.17.1-alpine

COPY nginx.conf /etc/nginx/nginx.conf

COPY --from=build /usr/src/app/dist/ng-ad-app /usr/share/nginx/html

1. Create service, deployment and ingress as given below:

apiVersion : apps/v1

kind: Deployment

metadata:

  name: ng-ad-app-deploy

  labels:

    app: ng-ad-app

spec:

  replicas: 1

  selector:

    matchLabels:

      app: ng-ad-app

  template:

    metadata:

      labels:

        app: ng-ad-app

    spec:

      containers:

        - name: ng-ad-app

          image: "smregistry1.azurecr.io/ng-ad-app"

          ports:

          - containerPort: 80

---

apiVersion: v1

kind: Service

metadata:

    name: ng-ad-app-svc

    labels:

      app: ng-ad-app

spec:

    type: ClusterIP

    ports:

      - port: 80

        targetPort: 80

    selector:

        app: ng-ad-app

---

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

  name: ng-ad-app-ingress

  annotations:

    kubernetes.io/ingress.class: nginx

spec:

  tls:

  - hosts:

    - ng-ad-app2.eastus.cloudapp.azure.com

    secretName: aks-ingress-tls

  rules:

  - host: ng-ad-app2.eastus.cloudapp.azure.com

    http:

      paths:

      - path: /

        pathType: Prefix

        backend:

          service:

            name: ng-ad-app-svc

            port:

              number: 80

Create service, deployment, and ingress with following command:

kubectl create -f service-deployment.yaml –namespace ingress-basic

Access your domain from browser:

https://<your domain name>

Graphical user interface, application

Description automatically generated

Now, try Login:

Graphical user interface, application, Teams

Description automatically generated

Once login is successful, next scree looks like this:

Graphical user interface, application

Description automatically generated

