Auth0 IdP を利用した Qlik Sense Enterprise on Kubernetes の設定

Qlik Sense Enterprise マルチクラウド演習

目次

ここではまず、Azure の環境に Qlik Sense Enterprise on Kubernetes の展開を行う事前に、それらの作業を行うローカルの Windows PC 上で以下の準備を行います。

- Azure アカウントの取得
- Azure CLI の導入
- kubectl の導入
- helm の導入
- IdP プロバイダの準備

1.Azure アカウントの取得

事前に取得済みのアカウント情報をご準備頂くか、アカウントをお持ちでない場合には以下のサイトからトライアルアカウントを作成してください。

https://azure.microsoft.com/ja-jp/

2.Azure CLI の導入

以下のサイトから Azure CLI のインストーラーをダウンロードし、ローカル Windows PC にインストールを行います。

https://docs.microsoft.com/ja-jp/cli/azure/?view=azure-cli-latest

3.Kubectl の導入

以下のサイトの説明に従って kubectl をローカル Windows PC に導入します。

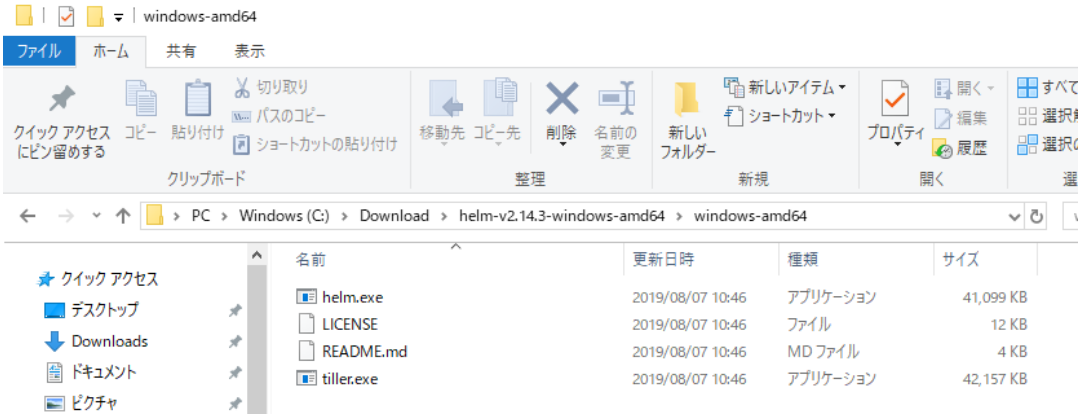
https://kubernetes.io/docs/tasks/tools/install-kubectl/#install-kubectl-on-windows

4.Helm の導入

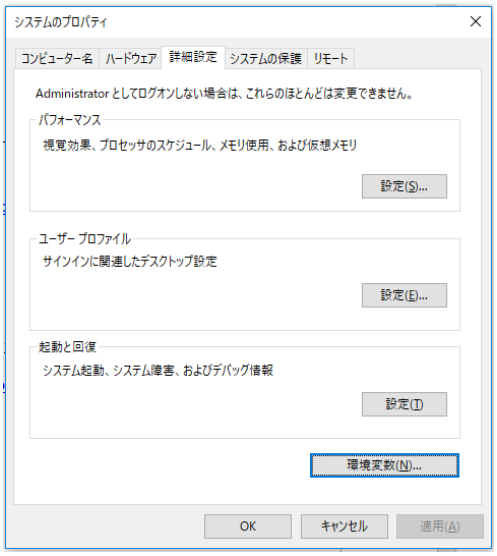
1) 以下のサイトから helm の最新版をダウンロードします。

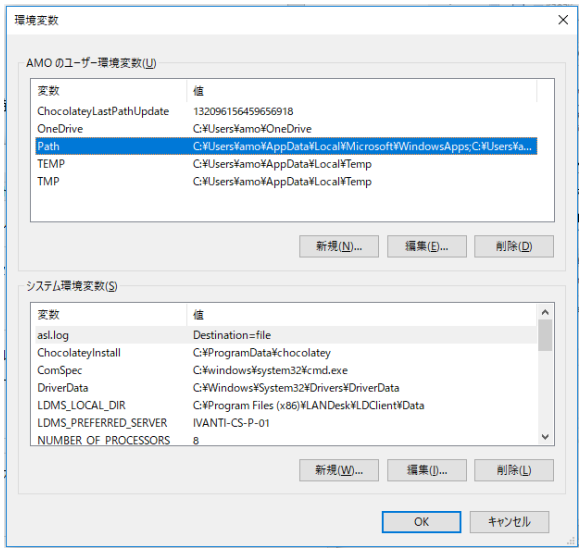
   https://github.com/helm/helm/releases

2) 任意の場所にダウンロードした Zip ファイルを解凍します。



3) helm.exe ファイルが存在するフォルダへ PATH を通します。(Windows 左下の検索ボックスに「システムの詳細」と入力し、検索結果として表示される「システムの詳細設定の表示」を選択します。「システムのプロパティ」ウインドウ上で「環境変数」をクリックしてフォルダへのパスを追加します。)



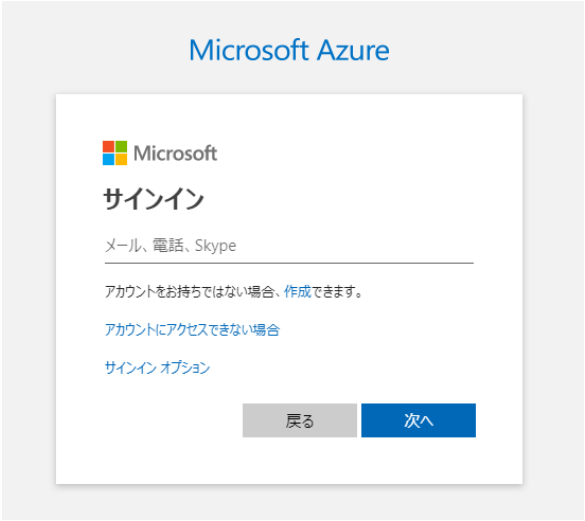


5.Qlik Sense Enterprise on Kubernetes の展開

1) PowerShell を開いて以下のコマンドを実行して Azure へログインします。

```
az login
```

2) Azure への認証情報を入力してログオンを完了し、ブラウザを閉じます。



3) Azure アカウントの Azure Kubernetes Service(AKS)を有効化します。

```
az provider register -n Microsoft.ContainerService
az provider register -n Microsoft.Compute
```

4) AKS の有効化には数分かかることがありますので、以下のコマンドで「RegistrationState」がそれぞれ「Registered」になるまで待ちます。

```
az provider show -n Microsoft.ContainerService | Select-String registrationState
az provider show -n Microsoft.Compute | Select-String registrationState
```

```
PS C:¥Users¥amo> az provider register -n Microsoft.ContainerService
PS C:¥Users¥amo> az provider register -n Microsoft.Compute
PS C:¥Users¥amo> az provider show -n Microsoft.ContainerService | Select-String registrationState

  "registrationState": "Registered",


PS C:¥Users¥amo> az provider show -n Microsoft.Compute | Select-String registrationState

  "registrationState": "Registered",
```

5) 次に Kubernetes クラスタを作成します。ここではまずクラスタ名とリソースグループ名を変数に格納します。

```
$AKS_CLUSTER_NAME="AKSCluster"
$AKS_RES_GROUP=$AKS_CLUSTER_NAME
```

6) 次のコマンドを実行してリソースグループを作成します。

```
az group create --name $AKS_RES_GROUP --location japaneast
```

※ここでは japaneast を利用していますが、その他の利用可能なリージョンの一覧は以下から参照が可能です。
https://docs.microsoft.com/ja-jp/azure/aks/quotas-skus-regions#region-availability

7) 次のコマンドを実行してクラスタを作成します。作成には十数分かかり、作成中は Running と表示されますが、完了すると Succeeded になります。(必要に応じてノード数や VM サイズなどを変更する必要があります。)

```
az aks create --resource-group $AKS_RES_GROUP --name $AKS_CLUSTER_NAME --
node-count 3 --generate-ssh-keys --node-vm-size Standard_DS2_v2
```

※ 仮想マシンのサイズと数によって利用料金が発生しますのでご注意ください。
https://azure.microsoft.com/ja-jp/pricing/details/virtual-machines/linux/

8) 以下を実行してクラスタ接続のための認証情報を取得します。このコマンドを実行するとユーザーの.kube ディレクトリに接続情報が保存され、デフォルトのコンテキストでこの AKS へ接続する設定となります。

```
az aks get-credentials --resource-group=$AKS_RES_GROUP --
name=$AKS_CLUSTER_NAME
```

9) 以下を実行して現在のコンテキストが「AKSCluster」となっていることを確認します。

```
kubectl config current-context
```

6.Role Based Access Control(RBAC)の設定

1) PowerShell のカレントディレクトリ上で、以下の内容の「rbac-config.yaml」という名称のファイルを作成します。

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: tiller
  namespace: kube-system
---
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRoleBinding
metadata:
  name: tiller
roleRef:
  apiGroup: rbac. y
```

2) 次のコマンドを実行して Tiller のサービスアカウントを作成し、ClusterRole にバインドします。

```
kubectl create -f rbac-config.yaml
```

3) 新しく作成されたサービスアカウントを利用して helm を初期化します。

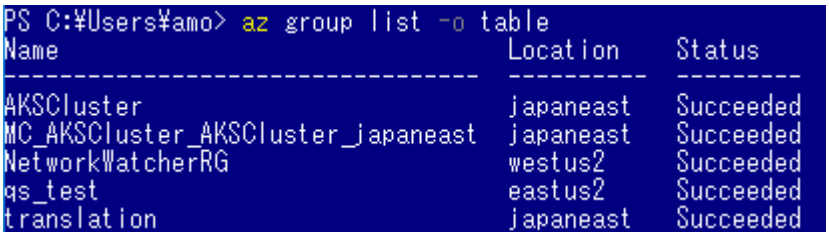
```
helm init --upgrade --service-account tiller
```

7.ストレージアカウントの作成

1) ファイルを格納するには、ストレージ アカウントをクラスターに追加する必要があります。リソースグループを検索するには、次の操作を行います。

```
az group list -o table
```

以下のように$AKS_RES_GROUP と MC_$AKS_RES_GROUP・・の 2 つの名称のリソースグループが作成されていることが確認できます。ここでは「MC」で始まるリソースグループを利用します。



2) ストレージアカウントの名称を指定します。Azure 上で一意の名称である必要があるため、任意の名称に変更してください。(文字数: 3～24、半角英数字が利用可能)

```
$AKS_STORAGE_ACT_NAME = "qsaksstorage"
```

3) 次のコマンドでストレージアカウントを作成します。

```
az storage account create -g
MC_${AKS_RES_GROUP}_${AKS_CLUSTER_NAME}_japaneast -n
$AKS_STORAGE_ACT_NAME --sku Standard_LRS
```

4) AKS は 2 種類のファイル ストレージを提供しますが、デフォルト (Azure Disk) は、Qlik Sense Enterprise on Kubernetes のクラスター内の複数のノードへ永続ストレージを提供するために必要な ReadWriteMany をサポートしていません。従って、Azure File Share を使用する必要があります。ストレージクラスとしてクラスターにストレージを追加する必要があります。

```
kubectl create clusterrole system:azure-cloud-provider --verb=get,create --
resource=secrets
```

5) ロールをバインドします。

```
kubectl create clusterrolebinding system:azure-cloud-provider --
clusterrole=system:azure-cloud-provider --serviceaccount=kube-system:persistent-
volume-binder
```

6) PowerShell のカレントディレクトリ上で、以下の内容の「azure-sc.yaml」という名称のファイルを作成します。

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: azurefile
provisioner: kubernetes.io/azure-file
mountOptions:
  - dir_mode=0777
  - file_mode=0777
```

```
  - uid=63400
  - gid=63400
parameters:
  skuName: Standard_LRS
```

7) 次のコマンドでストレージクラスを作成します。

```
kubectl apply -f azure-sc.yaml
```

8.Qlik Sense Enterprise on Kubernetes の導入(共通作業)

1) Qlik の Helm チャートレポジトリを追加します。

```
helm repo add qlik https://qlik.bintray.com/stable
```

2) 以下を実行してレポジトリの一覧を確認します。

```
helm repo list
```

3) 以下を実行します。

```
helm install --name qliksense-init qlik/qliksense-init
```

9.Qlik Sense Enterprise on Kubernetes の導入(外部 IdP を利用しないテスト環境の構築)

1) まずリソースグループ名を変数に格納します。Azure 上で一意の名称である必要があるため、名称は任意の名称に変更してください。

```
$APP_NAME="qs-soidc"
```

2) 次のコマンドを実行してリソースグループを作成します。

```
az group create --name $APP_NAME --location "Japan East"
```

3) サービスプランを作成します。

```
az appservice plan create --name $APP_NAME-ServicePlan --resource-group
$APP_NAME  --sku B1 --is-linux
```

4) ウェブアプリを作成します。

```
az webapp create --resource-group $APP_NAME --plan $APP_NAME-ServicePlan --name $APP_NAME-webapp --deployment-container-image-name qlik/simple-oidc-provider
```

5) ウェブアプリの設定を行います。

- REDIRECTS: 信頼できるリダイレクト URL を設定(コンマ区切り)
- PORT: アプリケーションポートの設定
- WEBSITES_PORT: Web サイト ポートを設定すると、Azure は https トラフィックをこのポートにリダイレクトします。

```
az webapp config appsettings set --resource-group $APP_NAME --name $APP_NAME-webapp --settings PORT=8000 WEBSITES_PORT=8000 REDIRECTS=https://elastic.example/login/callback
```

6) Discovery URL を取得します。取得した URL をメモします。

```
echo https://$APP_NAME-webapp.azurewebsites.net/.well-known/openid-configuration
```

```
PS C:¥Users¥amo¥azure> echo https://$APP_NAME-webapp.azurewebsites.net/.well-known/openid-configuration
https://qs-soidc-webapp.azurewebsites.net/.well-known/openid-configuration
```

7) エディタを利用して以下の内容の values_simple_oidc.yaml ファイルを作成します。以下の黄色でハイライトされている DiscoveryURL は前の手順で取得した URL で置き換えてください。

```
#This setting enables dev mode to include a local MongoDB install
devMode:
  enabled: true

#This setting accepts the EULA for the product
engine:
  acceptEULA: "yes"

global:
  persistence:
    storageClass: "azurefile"

identity-providers:
  secrets:
    idpConfigs:
```

```
    - discoveryUrl: "https://qs-soidc-webapp.azurewebsites.net/.well-known/openid-configuration"
      clientId: "foo"
      clientSecret: "bar"
      realm: "simple"
      hostname: "elastic.example"
      claimsMapping:
        sub: ["sub", "client_id"]
```

8) 以下を実行して Qlik Sense Enterprise on Kubernetes を展開します。

```
helm upgrade --install qliksense qlik/qliksense -f values_simple_oidc.yaml
```

9) 以下を実行し、展開した Pod の状況を確認します。STATUS が ContainerCreating から全て Running となるまで待ちます。

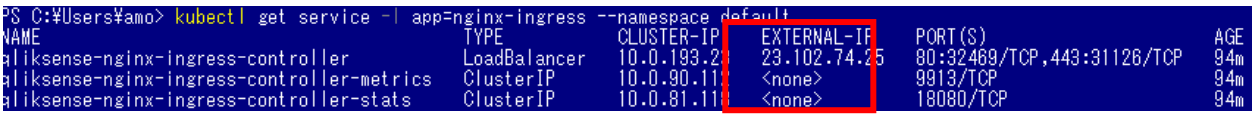
```
kubectl get pods
```

以下のコマンドを実行すると定期的な繰り返し処理で上記のコマンドを実行することも可能です。

```
while (1) {kubectl get pods; sleep 2; cls }
```

10) 外部 IP アドレスを取得するため以下のコマンドを実行します。

```
kubectl get service -l app=nginx-ingress --namespace default
```



11) Windows ホストマシン情報 Host ファイルの変更します。以下のフォルダ情報 host.txt を開きます。

C:¥Windows¥System32¥drivers¥etc

12)　以下の形で先ほど取得した Azure 上の Qlik Sense Enterprise on Kubernetes の IP アドレスへ elastic.example のホスト名でアクセスを行えるための項目を追加します。(既に elastic.example のエントリが存在する場合にはコメントアウトして無効化します。)

```
# localhost name resolution is handled within DNS itself.
#    127.0.0.1       localhost
#    ::1             localhost
192.168.56.120 drt
192.168.56.102 qv03
192.168.56.103 qs03
192.168.56.20   qvws01
192.168.56.21   qvws02
174.129.44.242  qs01.qlik.local
192.168.56.101  qs01
192.168.56.201  qmi-qs-qabdi
192.168.56.202  qmi-qabdi
192.168.56.74 qmi-attunity-en
23.102.74.25 elastic.example
#192.168.56.100 elastic.example
```

13)　Windows のホストコンピューター上でブラウザを開き、以下の URL にアクセスします。

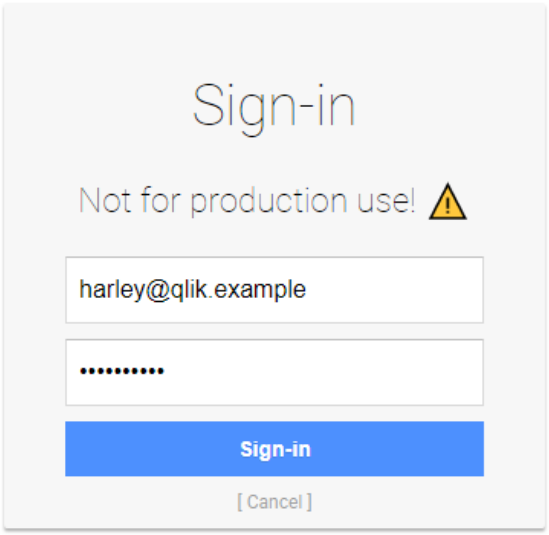
https://elastic.example/

14)　以下のような警告画面が表示されますので、そのままアクセスを行います。

15)　サインインの画面が表示されますので、以下のユーザーID・パスワードを使ってログインします。

ユーザーID: harley@qlik.example

パスワード: Password1!



※ デフォルトで以下のようなテストユーザーが登録されています。

メールアドレス	パスワード	グループ
harley@qlik.example	Password1!	'Everyone', 'Sales'
barb@qlik.example	Password1!	'Everyone', 'Support'
quinn@qlik.example	Password1!	'Everyone', 'Accounting'
sim@qlik.example	Password1!	'Everyone', 'Accounting'
phillie@qlik.example	Password1!	'Everyone', 'Marketing', 'Sales'
peta@qlik.example	Password1!	'Everyone', 'Engineering'
marne@qlik.example	Password1!	'Everyone', 'Marketing'
sibylla@qlik.example	Password1!	'Everyone', 'Accounting'
evan@qlik.example	Password1!	'Everyone', 'Engineering'
franklin@qlik.example	Password1!	'Everyone', 'Sales'

16) 以下のようなハブのトップ画面が表示されます。この状態では IdP を使わず、テスト利用目的に内部的に用意されたユーザーを使ってログオンしている状況となります。

この環境でライセンスを適用してそのまま利用するには別紙「03_Qlik Sense Enterprise on Kubernetes 上の管理コンソールでのライセンス有効化」の説明に従ってライセンスの適用を行います。ただし、ライセンスを適用したユーザーが RootAdmin としての管理者権限が付与されますので、一旦ライセンスを適用後に認証を IdP での認証に切り替えると、この RootAdmin のユーザーとしてログオンが出来なくなってしまいますので注意が必要です。次章の Auth0 を使った認証設定に進むには、ライセンスの有効化を行わず、このまま次章に進んでください。

10. Qlik Sense Enterprise on Kubernetes の導入 (Auth0 を利用した環境の構築)

手順は別紙「02_Auth0 IdP を利用した Qlik Sense Enterprise on Kubernetes の設定」をご参照ください。尚、Qlik Sense Enterprise on Kub
ernetes を展開する際に利用する YAML ファイルは以下となります。また、黄色でハイライトされている「discoveryUrl」、「clientId」、「clientSecret」、「postLogoutRedirectUri」の値は前段の手順でメモした Auth0 上のアプリケーションの「Domain」、「Client ID」、「Client Secret」の値に基づいて書き換える必要があります。

```
#This setting enables dev mode to include a local MongoDB install
devMode:
  enabled: true

#This setting accepts the EULA for the product
engine:
  acceptEULA: "yes"

global:
  persistence:
    storageClass: "azurefile"

identity-providers:
  secrets:
    idpConfigs:
      - discoveryUrl: "https://<Domain>/.well-known/openid-configuration"
        clientId: "DxLSmVzaacCdm42rlFaHWIWA9kCIipDT"
        clientSecret : "N7mpe7xkrC-omyTFZb6iIu-
EfGB7KsyPXCI8J9XOo2HcvyLZZcPjMu7u-3-I7Ncc"
        realm: "Auth0"
        postLogoutRedirectUri: "https://<Domain>/v2/logout/"
```

```
    hostname: "elastic.example"
    claimsMapping:
      client_id: [ "client_id", "azp" ]
      groups: "/https:~1~1qlik.com~1groups"
      sub: ["/https:~1~1qlik.com~1sub","sub"]
```

11. 作成した環境のクリーンアップ

簡易認証用に作成したリソースを削除するには以下のコマンドを実行します。

```
az group delete --name $APP_NAME
```

作成した QSEoK の環境を削除するには以下を実行します。

```
helm del --purge qliksense
helm del --purge qliksense-init
```

以下のコマンドは Azure 上のリソースを削除します。

```
az group delete --name $AKS_RES_GROUP
```

About Qlik

Qlik is on a mission to create a data-literate world, where everyone can use data to solve their most challenging problems. Only Qlik's end-to-end data management and analytics platform brings together all of an organization's data from any source, enabling people at any skill level to use their curiosity to uncover new insights. Companies use Qlik products to see more deeply into customer behavior, reinvent business processes, discover new revenue streams, and balance risk and reward. Qlik does business in more than 100 countries and serves over 48,000 customers around the world.

qlik.com