**Kubernetes - LDAP authentication with Dex**

**This deployment follows Dex by CoreOS & Kubernetes Documentations:**

1. [**Kubernetes OIDC Doc**](https://kubernetes.io/docs/reference/access-authn-authz/authentication/#option-1---oidc-authenticator)
2. **Dex by CoreOS**
3. **Login App**

**Requirements:**

1. **DNS entries**: (Since this configuration uses NodePort, these can be CNAMEs to your kubernetes nodes)

* **dex.k8s.opsmx.com** --> Dex OIDC provider
* **login.k8s.opsmx.com** --> Custom Login Application

1. **Kubernetes cluster available with the following requirements:**
   1. **RBAC enabled**
   2. **OIDC authentication enabled. API server configuration:**

* - --oidc-issuer-url=https://dex.k8s.opsmx.com/dex: **External Dex endpoint**
* - --oidc-client-id=loginapp: **ID for our Login Application**
* - --oidc-ca-file=/etc/kubernetes/pki/ca.pem**: CA file generated using gencert.sh below**
* - --oidc-username-claim=uid**: Map to nameAttr Dex configuration. This will be used by Kubernetes RBAC to authorize users based on their name.**
* oidc-groups-claim=grou- --oidc-groups-claim=groups: **This will be used by Kubernetes RBAC to authorize users based on their groups.**
  1. **An available LDAP server**, it is 35.230.138.249 which was used for testing.

# Login App Application

**Create a name space called, ‘auth’.**

$ kubectl create ns auth

**Create required SSL certs and secrets (make sure to update alt\_names to match your domain):**

$ ./gencert.sh

$ kubectl create secret tls login.k8s.opsmx.com.tls --cert=ssl/cert.pem --key=ssl/key.pem -n auth

$ kubectl create secret tls dex.k8s.opsmx.com.tls --cert=ssl/cert.pem --key=ssl/key.pem -n auth

Create resources:

# CA configmap, Update 'ca-cm.yml' with the content of ca.pem(generated by gencert.sh)

$ kubectl create -f ca-cm.yml

# Login App configuration

$ kubectl create -f loginapp-cm.yml

# Login App service

$ kubectl create -f loginapp-ing-svc.yml

# Login App Deployment

$ kubectl create -f loginapp-deploy.yml

\* It should fail because Dex is not deployed.

# Dex App installation

* + 1. **Custom Resource Definitions:**

We will use Kubernetes Custom Resource Definitions thru ‘dex-crd.yml’ as Dex storage backend:

$ kubectl create -f dex-crd.yml

* + 1. **Deployment**

Create Dex resources:

# Dex configuration

$ kubectl create -f dex-cm.yml

# Dex service

$ kubectl create -f dex-ing-svc.yml

# Dex deployment

$ kubectl create -f dex-deploy.yml

Now assuming that you setup the DNS, this should work: try <https://login.k8s.opsmx.com:32002>, login and retrieve k8s configuration.

You can decode the id\_token to verify the returned claims using: https://jwt.io/

* + 1. **Create RBAC resource (assgin a group called "admins" cluster admin role):**

$ kubectl create -f rbac.yml

* + 1. **Copy below Certificate files:**

$ sudo cp ssl/cert.pem /etc/ssl/certs

$ sudo cp ssl/ca-key.pem /etc/ssl/private

# Manifests

The following are the manifests used for deploying **loginapp** and **dex**:

$ **kubectl -n auth edit cm ca**

# Please edit the object below. Lines beginning with a '#' will be ignored,

# and an empty file will abort the edit. If an error occurs while saving this file will be

# reopened with the relevant failures.

#

apiVersion: v1

data:

ca.pem: |

-----BEGIN CERTIFICATE-----

MIIC9zCCAd+gAwIBAgIJAPQfiXrgDVt8MA0GCSqGSIb3DQEBCwUAMBIxEDAOBgNV

BAMMB2t1YmUtY2EwHhcNMTkwODE1MTAxMTI2WhcNMjIwNTExMTAxMTI2WjASMRAw

DgYDVQQDDAdrdWJlLWNhMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA

0T9OlhRE0A3pgGZ0YTHCGObiqtS6994EufCJ9nVBVCuKsJhYigDnMfWQDMkdsl/W

M8MlZJxXqGwYOEfEZ0LFeaXemv3YFpR8CY4hbObLg5MQopFWB6HYRI8p/87cnwN0

z+uqlq8FmXlvmzFMfuXlGlF0bllozM6PD9pFOBoIp4gbCy+JeTzMh5kyVwj8iaWW

SEW+vjqi6xeiKEFXtoZNf90Dhtu5K0logC6iJDCcHenRQxDRhqh/JUybuXR6bDFG

DoseFT6H0XarS1NTtO4G4yNgwbOvdGb73z9sxR99QXFO31pllvvOtL2DEy6EVksV

bvBpiZVmLxmIBSBwAeoPKwIDAQABo1AwTjAdBgNVHQ4EFgQUa/ruKb+nY2txoKfN

ddclBVdJmDQwHwYDVR0jBBgwFoAUa/ruKb+nY2txoKfNddclBVdJmDQwDAYDVR0T

BAUwAwEB/zANBgkqhkiG9w0BAQsFAAOCAQEAYfMOiSoK5FhOi78f8D/esg8Uij9v

XHKd4xCWnqMeIIPW9ThUHCTWHnIr2myuotMD5tuHJ93TmBEx6QofKoicgB54JVEy

yowN2f+vnJUCxVwrtOphjbJOMtcZ9shBEjYce/W03clZcQCHjq/K9wygVaQ8q9qC

7G3zGJ8x+mQPFY07oot8hG/JW8xgsihfklhTXZgEzUd27ruSfExTlip18LFd7RPI

/w4HQEZXsq1k8Ssr1vFoK4foWNeDyqMtjpTRozeRPA2UTwvuZxfZXS1s8DXiMzo5

n677SQ89hJDwrAgroaAokGHY2lNMeLd69SdIo1EaeNi8RrpV6aLxdmlrXA==

-----END CERTIFICATE-----

kind: ConfigMap

metadata:

creationTimestamp: "2019-08-15T10:16:05Z"

name: ca

namespace: auth

resourceVersion: "700773"

selfLink: /api/v1/namespaces/auth/configmaps/ca

uid: dcff33b0-0140-42a1-a5fe-fca3679efecc

==================================================================================

$ **kubectl -n auth edit cm loginapp**

# Please edit the object below. Lines beginning with a '#' will be ignored,

# and an empty file will abort the edit. If an error occurs while saving this file will be

# reopened with the relevant failures.

#

apiVersion: v1

data:

config.yaml: |

debug: false

client\_id: "loginapp"

client\_secret: 4TORGiNV9M54BTk1v7dNuFSaI6hUjfjr

issuer\_url: "https://dex.k8s.opsmx.com:32000/dex"

issuer\_root\_ca: "/etc/ssl/ca.pem"

redirect\_url: "https://login.k8s.opsmx.com:32002/callback"

listen: "https://0.0.0.0:5555"

tls\_cert: "/etc/loginapp/tls/tls.crt"

tls\_key: "/etc/loginapp/tls/tls.key"

disable\_choices: false

extra\_scopes: "groups"

app\_name: "Kubernetes Auth"

kind: ConfigMap

metadata:

creationTimestamp: "2019-08-15T10:19:09Z"

name: loginapp

namespace: auth

resourceVersion: "701793"

selfLink: /api/v1/namespaces/auth/configmaps/loginapp

uid: 2c6eb6a4-1f14-40d8-a3fd-8d8d708b99b2

==================================================================================

**$ kubectl -n auth edit deploy loginapp**

# Please edit the object below. Lines beginning with a '#' will be ignored,

# and an empty file will abort the edit. If an error occurs while saving this file will be

# reopened with the relevant failures.

#

apiVersion: extensions/v1beta1

kind: Deployment

metadata:

annotations:

deployment.kubernetes.io/revision: "2"

creationTimestamp: "2019-08-15T10:22:53Z"

generation: 2

labels:

app: loginapp

name: loginapp

namespace: auth

resourceVersion: "703791"

selfLink: /apis/extensions/v1beta1/namespaces/auth/deployments/loginapp

uid: a2ee0224-7557-4c7e-8603-ae0c3954581c

spec:

progressDeadlineSeconds: 2147483647

replicas: 1

revisionHistoryLimit: 2147483647

selector:

matchLabels:

app: loginapp

strategy:

rollingUpdate:

maxSurge: 1

maxUnavailable: 1

type: RollingUpdate

template:

metadata:

creationTimestamp: null

labels:

app: loginapp

spec:

containers:

- image: objectiflibre/login-app:latest

**imagePullPolicy: IfNotPresent**

name: loginapp

ports:

- containerPort: 5555

hostPort: 5555

name: http

protocol: TCP

resources: {}

terminationMessagePath: /dev/termination-log

terminationMessagePolicy: File

volumeMounts:

- mountPath: /etc/ssl/

name: ca

- mountPath: /app/

name: config

- mountPath: /etc/loginapp/tls

name: tls

**dnsPolicy: ClusterFirstWithHostNet**

**hostNetwork: true**

restartPolicy: Always

schedulerName: default-scheduler

securityContext: {}

terminationGracePeriodSeconds: 30

volumes:

- configMap:

defaultMode: 420

items:

- key: ca.pem

path: ca.pem

name: ca

name: ca

- configMap:

defaultMode: 420

items:

- key: config.yaml

path: config.yaml

name: loginapp

name: config

- name: tls

secret:

defaultMode: 420

secretName: login.k8s.opsmx.com.tls

status:

availableReplicas: 1

conditions:

- lastTransitionTime: "2019-08-15T10:22:53Z"

lastUpdateTime: "2019-08-15T10:22:53Z"

message: Deployment has minimum availability.

reason: MinimumReplicasAvailable

status: "True"

type: Available

observedGeneration: 2

readyReplicas: 1

replicas: 1

updatedReplicas: 1

==================================================================================

**$ kubectl -n auth edit svc loginapp**

# Please edit the object below. Lines beginning with a '#' will be ignored,

# and an empty file will abort the edit. If an error occurs while saving this file will be

# reopened with the relevant failures.

#

apiVersion: v1

kind: Service

metadata:

creationTimestamp: "2019-08-15T10:20:39Z"

name: loginapp

namespace: auth

resourceVersion: "701108"

selfLink: /api/v1/namespaces/auth/services/loginapp

uid: 88371086-745f-4328-9d22-dd1be1f2c81e

spec:

clusterIP: 10.102.27.60

externalTrafficPolicy: Cluster

ports:

- name: loginapp

nodePort: 32002

port: 5555

protocol: TCP

targetPort: 5555

selector:

app: loginapp

sessionAffinity: None

type: NodePort

status:

loadBalancer: {}

==================================================================================

**$ kubectl -n auth edit cm dex**

# Please edit the object below. Lines beginning with a '#' will be ignored,

# and an empty file will abort the edit. If an error occurs while saving this file will be

# reopened with the relevant failures.

#

apiVersion: v1

data:

config.yaml: |

issuer: https://dex.k8s.opsmx.com:32000/dex

storage:

type: kubernetes

config:

inCluster: true

web:

https: 0.0.0.0:5556

tlsCert: /etc/dex/tls/tls.crt

tlsKey: /etc/dex/tls/tls.key

logger:

level: "debug"

format: text

connectors:

- type: ldap

# Required field for connector id.

id: ldap

# Required field for connector name.

name: LDAP

config:

# Host and optional port of the LDAP server in the form "host:port".

# If the port is not supplied, it will be guessed based on "insecureNoSSL",

# and "startTLS" flags. 389 for insecure or StartTLS connections, 636

# otherwise.

host: **<LDAP-SERVER-IP>:<PORT>**

# Following field is required if the LDAP host is not using TLS (port 389).

# Because this option inherently leaks passwords to anyone on the same network

# as dex, THIS OPTION MAY BE REMOVED WITHOUT WARNING IN A FUTURE RELEASE.

#

insecureNoSSL: true

# If a custom certificate isn't provide, this option can be used to turn on

# TLS certificate checks. As noted, it is insecure and shouldn't be used outside

# of explorative phases.

#

insecureSkipVerify: true

# When connecting to the server, connect using the ldap:// protocol then issue

# a StartTLS command. If unspecified, connections will use the ldaps:// protocol

#

# startTLS: true

# Path to a trusted root certificate file. Default: use the host's root CA.

#rootCA: /etc/dex/ldap.ca

# A raw certificate file can also be provided inline.

#rootCAData:

# The DN and password for an application service account. The connector uses

# these credentials to search for users and groups. Not required if the LDAP

# server provides access for anonymous auth.

# Please note that if the bind password contains a `$`, it has to be saved in an

# environment variable which should be given as the value to `bindPW`.

bindDN: CN=**<USERNAME>**,CN=Users,DC=local,DC=opsmx,DC=com

bindPW: **<PASSWORD>**

# User search maps a username and password entered by a user to a LDAP entry.

userSearch:

# BaseDN to start the search from. It will translate to the query

# "(&(objectClass=person)(uid=<username>))".

baseDN: CN=Users,DC=local,DC=opsmx,DC=com

# Optional filter to apply when searching the directory.

#filter: "(objectClass=posixAccount)"

# username attribute used for comparing user entries. This will be translated

# and combine with the other filter as "(<attr>=<username>)".

username: name

# The following three fields are direct mappings of attributes on the user entry.

# String representation of the user.

idAttr: name

# Required. Attribute to map to Email.

emailAttr: name

# Maps to display name of users. No default value.

nameAttr: name

# Group search queries for groups given a user entry.

groupSearch:

# BaseDN to start the search from. It will translate to the query

# "(&(objectClass=group)(member=<user uid>))".

baseDN: CN=Users,DC=local,DC=opsmx,DC=com

# Optional filter to apply when searching the directory.

filter: "(objectClass=posixGroup)"

# Following two fields are used to match a user to a group. It adds an additional

# requirement to the filter that an attribute in the group must match the user's

# attribute value.

userAttr: uid

groupAttr: memberUid

# Represents group name.

nameAttr: cn

oauth2:

skipApprovalScreen: true

staticClients:

- id: loginapp

redirectURIs:

- 'https://login.k8s.opsmx.com:32002/callback'

name: 'Login Application'

secret: 4TORGiNV9M54BTk1v7dNuFSaI6hUjfjr

kind: ConfigMap

metadata:

creationTimestamp: "2019-08-15T10:26:55Z"

name: dex

namespace: auth

resourceVersion: "708342"

selfLink: /api/v1/namespaces/auth/configmaps/dex

uid: c9dc75eb-657c-4fc2-ad8c-1539d8f47375

==================================================================================

**$ kubectl -n auth edit deploy dex**

# Please edit the object below. Lines beginning with a '#' will be ignored,

# and an empty file will abort the edit. If an error occurs while saving this file will be

# reopened with the relevant failures.

#

apiVersion: extensions/v1beta1

kind: Deployment

metadata:

annotations:

deployment.kubernetes.io/revision: "1"

creationTimestamp: "2019-08-15T10:27:40Z"

generation: 1

labels:

app: dex

name: dex

namespace: auth

resourceVersion: "708385"

selfLink: /apis/extensions/v1beta1/namespaces/auth/deployments/dex

uid: 442331f2-dba0-41a2-b1d6-a6cb6ab55b7c

spec:

progressDeadlineSeconds: 2147483647

replicas: 1

revisionHistoryLimit: 2147483647

selector:

matchLabels:

app: dex

strategy:

rollingUpdate:

maxSurge: 1

maxUnavailable: 1

type: RollingUpdate

template:

metadata:

creationTimestamp: null

labels:

app: dex

spec:

containers:

- command:

- dex

- serve

- /etc/dex/cfg/config.yaml

image: quay.io/coreos/dex:v2.9.0

imagePullPolicy: IfNotPresent

name: dex

ports:

- containerPort: 5556

name: http

protocol: TCP

resources: {}

terminationMessagePath: /dev/termination-log

terminationMessagePolicy: File

volumeMounts:

- mountPath: /etc/dex/cfg

name: config

- mountPath: /etc/dex/tls

name: tls

dnsPolicy: ClusterFirst

restartPolicy: Always

schedulerName: default-scheduler

securityContext: {}

serviceAccount: dex

serviceAccountName: dex

terminationGracePeriodSeconds: 30

volumes:

- configMap:

defaultMode: 420

items:

- key: config.yaml

path: config.yaml

name: dex

name: config

- name: tls

secret:

defaultMode: 420

secretName: dex.k8s.opsmx.com.tls

status:

availableReplicas: 1

conditions:

- lastTransitionTime: "2019-08-15T10:27:40Z"

lastUpdateTime: "2019-08-15T10:27:40Z"

message: Deployment has minimum availability.

reason: MinimumReplicasAvailable

status: "True"

type: Available

observedGeneration: 1

readyReplicas: 1

replicas: 1

updatedReplicas: 1

==================================================================================

**$ kubectl -n auth edit svc dex**

# Please edit the object below. Lines beginning with a '#' will be ignored,

# and an empty file will abort the edit. If an error occurs while saving this file will be

# reopened with the relevant failures.

#

apiVersion: v1

kind: Service

metadata:

creationTimestamp: "2019-08-15T10:27:10Z"

name: dex

namespace: auth

resourceVersion: "701644"

selfLink: /api/v1/namespaces/auth/services/dex

uid: 1a4e935a-785d-478a-bd58-0a3dbd6c4068

spec:

clusterIP: 10.107.44.112

externalTrafficPolicy: Cluster

ports:

- name: dex

nodePort: 32000

port: 5556

protocol: TCP

targetPort: 5556

selector:

app: dex

sessionAffinity: None

type: NodePort

status:

loadBalancer: {}

**$ cat /etc/hosts**

127.0.0.1 localhost

127.0.0.1 dex.k8s.opsmx.com

127.0.0.1 login.k8s.opsmx.com

# The following lines are desirable for IPv6 capable hosts

::1 ip6-localhost ip6-loopback

fe00::0 ip6-localnet

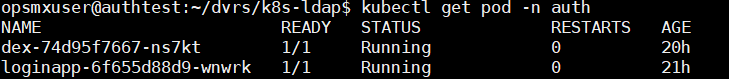
Update ‘**hosts’** file in ‘**C:\Windows\System32\drivers\etc’** location in my local machine to access loginapp through browser:

**<Public-IP of VM> dex.k8s.opsmx.com**

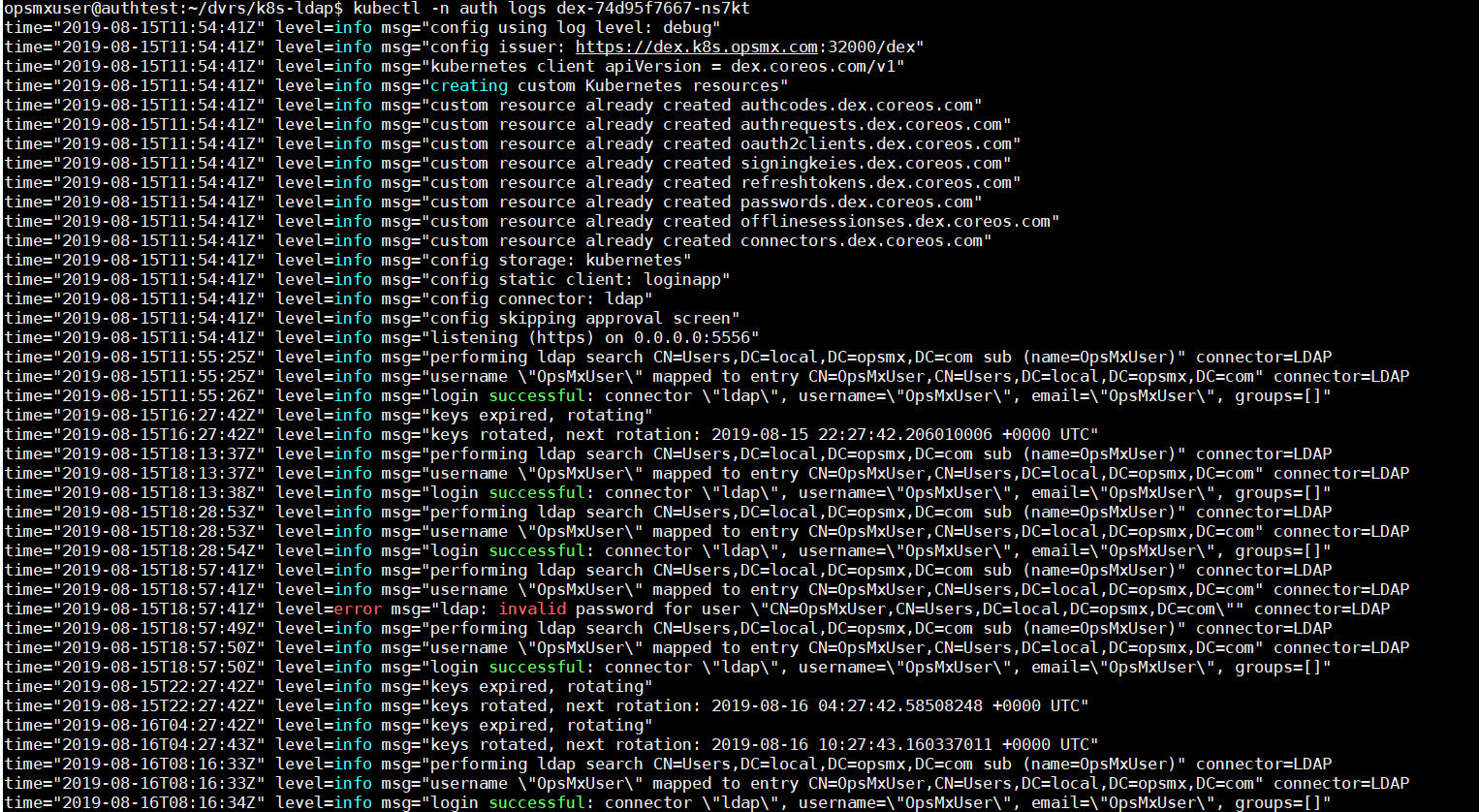
**<Public-IP of VM> login.k8s.opsmx.com**

# Verification

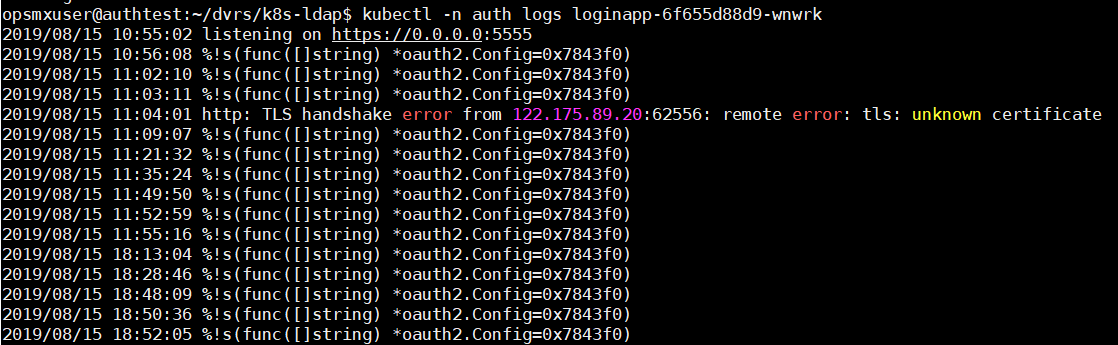
* 1. Running **loginapp**and **pod**status:



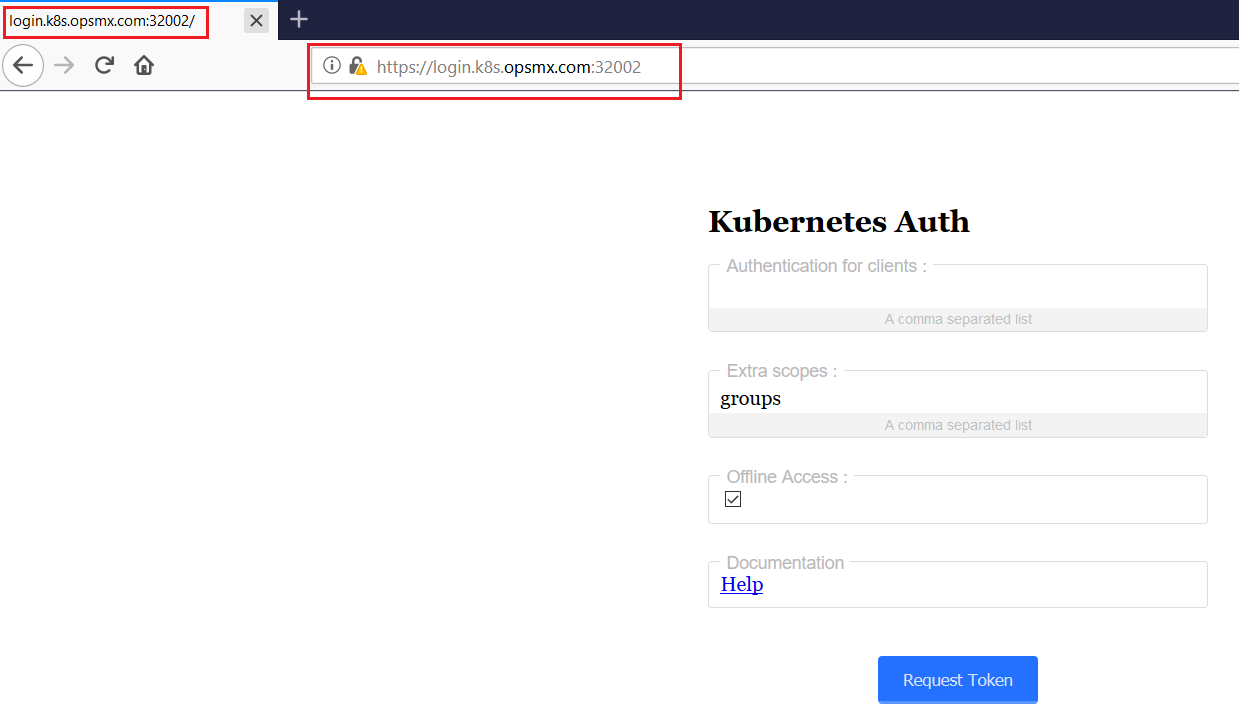
* 1. The below is **dex**app logs after several attempts of LDAP configuration:



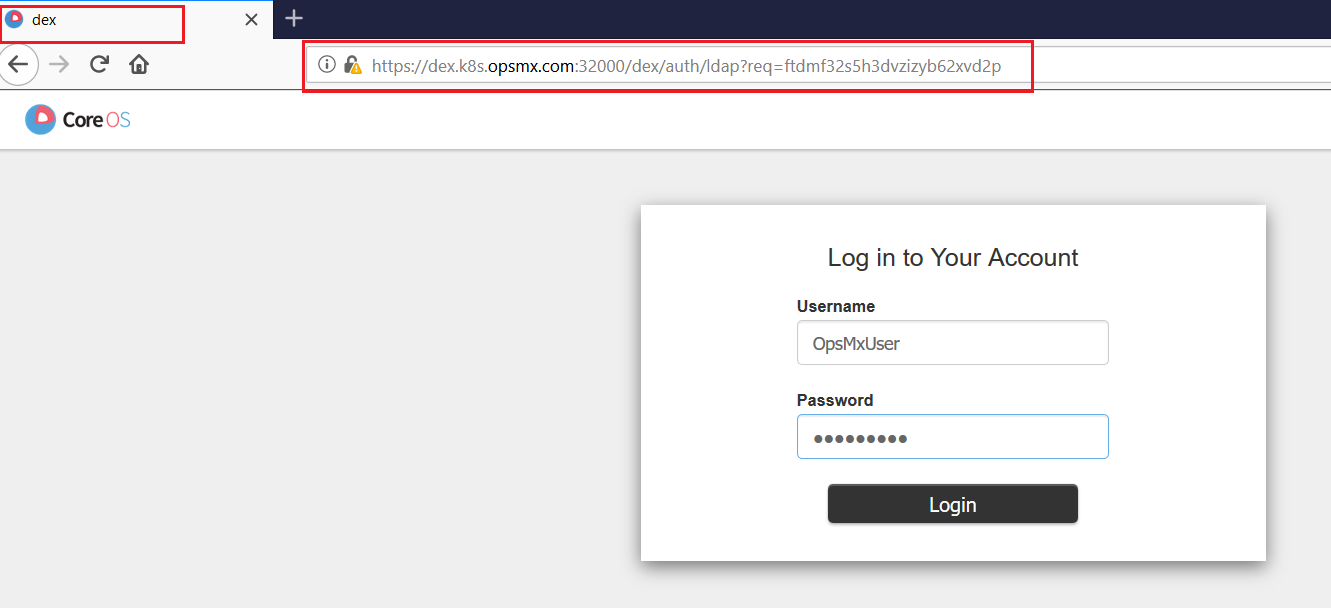
* 1. The below are the **loginapp** logs:



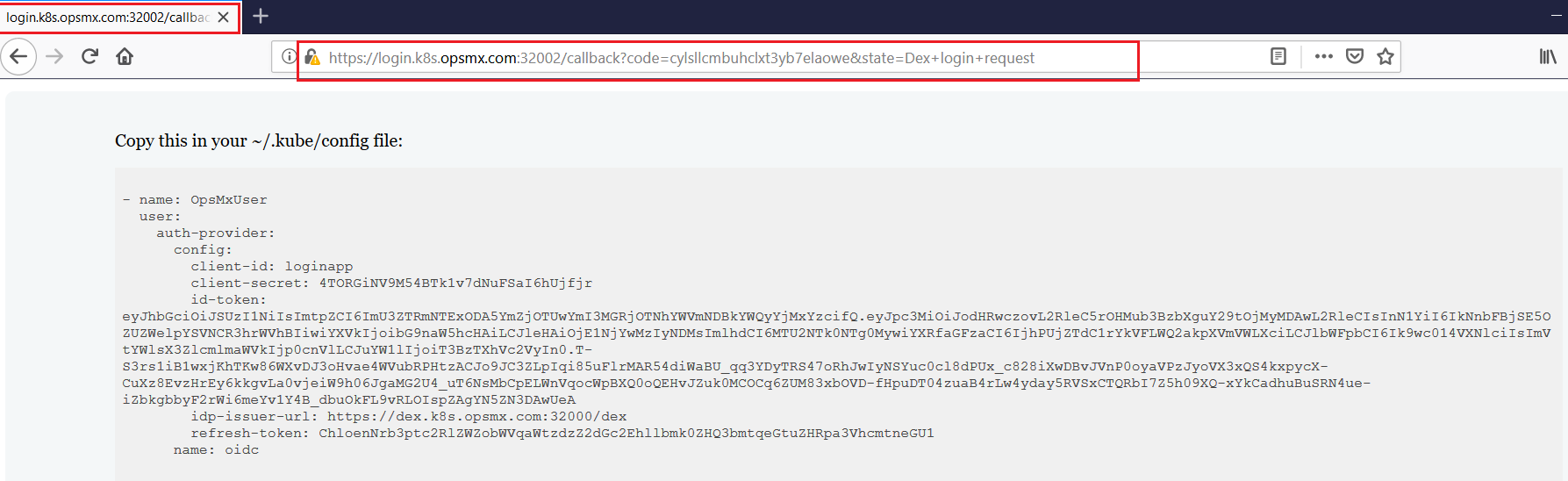
* 1. Checked login URL, [https://login.k8s.opsmx.com:32002](https://login.k8s.opsmx.com:32002/) from browser and click on 'Request Token' button.



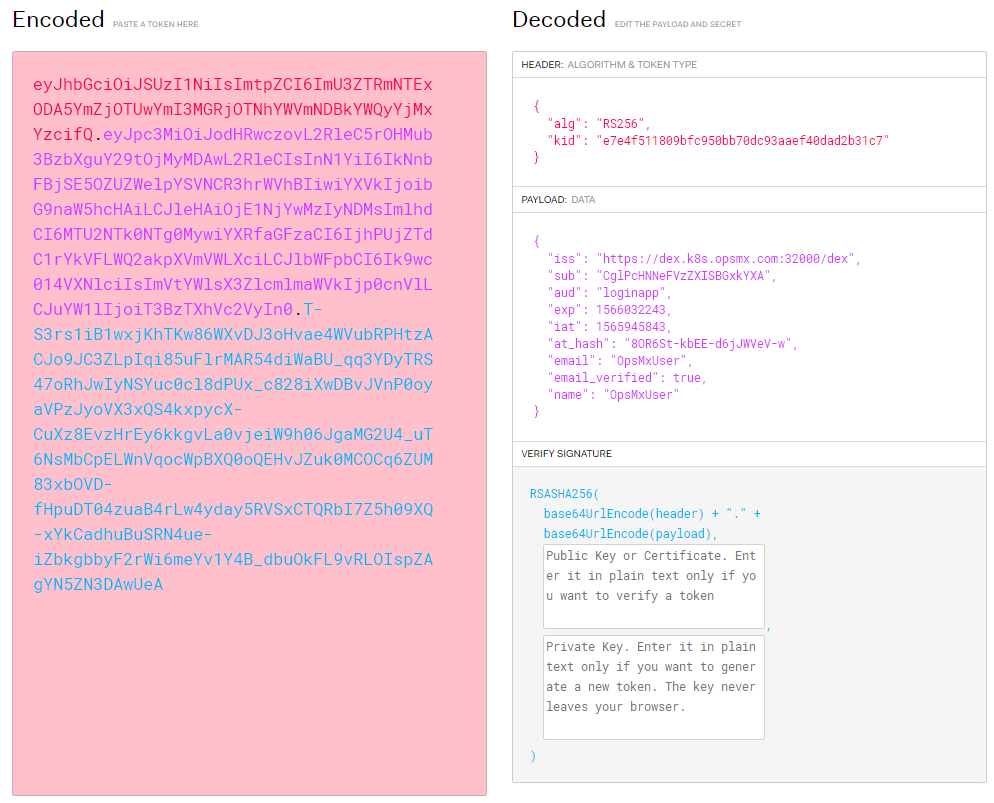
* 1. Then it is redirected to 'dex' app and seeking the login details of LDAP user as shown below:



* 1. Entered the login details of a LDAP user('OpsMxUser') and reached to the below callback:



* 1. You can decode the id\_token to verify the returned claims using: <https://jwt.io/>



* 1. **Kubernetes Server side**: In the Kubernetes cluster, Create **ClusterRoleBinding** which allows ‘OpsMxUser’ to grant admin/edit/view the objects in the default or custom namespace.

$ cat ldapcrb.yml

kind: ClusterRoleBinding

apiVersion: rbac.authorization.k8s.io/v1

metadata:

name: testedit

subjects:

- kind: User

name: https://dex.k8s.opsmx.com:32000/dex#OpsMxUser

apiGroup: rbac.authorization.k8s.io

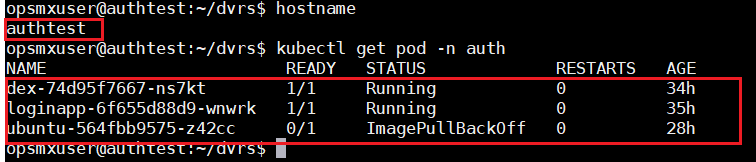
roleRef:

kind: ClusterRole

name: edit

apiGroup: rbac.authorization.k8s.io

$ kubectl create –f ldapcrb.yml



* 1. **Client VM side:** Connect any client VM and build a ‘**config’** file in ‘.kube’ directory with the loginapp config which was received on the browser. Define clusters with the k8s server details and contexts with a specific user, ‘OpsMxUser’ in our case.



$ cat $HOME/.kube/config

apiVersion: v1

clusters:

- cluster:

certificate-authority-data: 

server: https://10.160.0.9:6443

name: kubernetes

contexts:

- context:

cluster: kubernetes

user: OpsMxUser

name: opsmx

current-context: opsmx

kind: Config

preferences: {}

users:

- name: OpsMxUser

user:

auth-provider:

config:

client-id: loginapp

client-secret: 4TORGiNV9M54BTk1v7dNuFSaI6hUjfjr

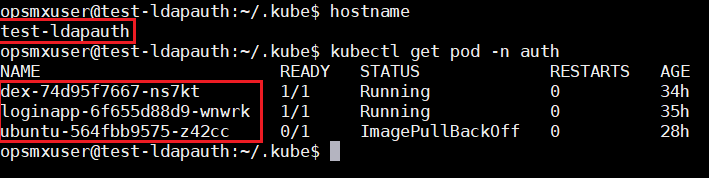
id-token: eyJhbGciOiJSUzI1NiIsImtpZCI6ImE2NzY0NDMwMTk5MTY3YzRlOGZlMGQ0NWYzNDI4NzNiODJiYTE5ODcifQ.eyJpc3MiOiJodHRwczovL2RleC5rOHMub3BzbXguY29tOjMyMDAwL2RleCIsInN1YiI6IkNnbFBjSE5OZUZWelpYSVNCR3hrWVhBIiwiYXVkIjoibG9naW5hcHAiLCJleHAiOjE1NjYwNTI2MDMsImlhdCI6MTU2NTk2NjIwMywiYXRfaGFzaCI6IndSdHBMMHdET25mMmNrU2oxbERKSmciLCJlbWFpbCI6Ik9wc014VXNlciIsImVtYWlsX3ZlcmlmaWVkIjp0cnVlLCJuYW1lIjoiT3BzTXhVc2VyIn0.nqhW0QUkuwiZWLzIgqcApzj0HBIRKoxv2ap9scxsetm8rw7A7Df50\_ML6gGgwdIc8WrU5aiLh7TaLmLd0EaHjY1jOHbvLEe746cpZaWy1PBTWNDiwmbSBKJdyWKEnCDUdJqYOkZ6\_U7TbsgmLLhSwrE2kLZn0kLYN8ktZlIsZ6wkVO\_LkMuXCdbiBH\_D2r5t1INZQ2\_ysxme\_AXNQggOe0OxGEf6o65yVrE2tevQYgfZmKFI9eVfvJ6jorrfiVXB3GDybiHnQXmzZuHAaj707QtozAMD8U8EyNOHH5Qu6KeCu2ThX99ujTVdZ9pTkXdAcIHGAPPhbXZtSAZnK\_h31g

idp-issuer-url: https://dex.k8s.opsmx.com:32000/dex

refresh-token: Chl0dWEycHozNTV2ZG1tazRpYWg1bWV6ZXF3Ehl2ZDdsZWhxcHJ5dDZ6M2NocmhsM2s3cmR1

name: oidc

And then check the access of running pods with kubectl command which should display the pods which are running in the Kubernetes server as shown below in client VM called 'test-ldapauth':



# References

<https://github.com/krishnapmv/k8s-ldap>