Kubernetes Followers using Docker Desktop with Kubernetes

Install Docker Desktop

A screenshot of a computer

Description automatically generated

Choose Preferences:

A screenshot of a cell phone

Description automatically generated

Select Kubernetes:

A screenshot of a cell phone screen with text

Description automatically generated

Make sure Enable Kubernetes has been selected. Apply and Restart if necessary.

First, download the images needed for the deployment:

#conjur-appliance image

docker pull captainfluffytoes/dap:latest

#or download the conjur-appliance image from the SFE and install it:

docker load -i conjur-appliance\_11.4.0.tar.gz

#kubernetes seed fetcher image

docker pull captainfluffytoes/seed-fetcher:latest

#conjur cli image

docker pull cyberark/conjur-cli:latest

Next, deploy the Master container:

docker container run -d --name conjur-appliance --restart unless-stopped --security-opt=seccomp:unconfined -p "443:443" -p "5432:5432" -p "1999:1999" registry.tld/conjur-appliance:11.4.0

Configure the Master:

docker exec conjur-appliance evoke configure master --accept-eula --hostname $MasterFQDN --master-altnames="$MasterVIP,$Standby1FQDN,$Standby2FQDN" -p Cyberark1 $environmentName

$MasterFQDN = fqdn address of the master node

$MasterVIP = fqdn address of the master vip (if no vip, use the master node address)

$Standby1FQDN = fqdn address of the 1st standby node (if no standby, you can enter fake fqdn)

$Standby2FQDN = fqdn address of the 2nd standby node (if no standby, you can enter fake fqdn)

$environmentName= the name of the conjur account (just a label but important as it is referenced throughout the configuration as Conjur Account)

My command looks like this:

docker exec conjur-appliance evoke configure master --accept-eula --hostname mymacbook.myorg.local --master-altnames="mymacbook.myorg.local,Standby1,Standby2" -p MyCyber@rk01 myorg

mymacbook.myorg.local is an entry I added to my /etc/hosts file for my local IP address

Standby1 and Standby2 are fake hostnames because I have no standby nodes

MyCyber@rk01 is my admin account password (due to password requirements for 11.4.0)

myorg is my Conjur Account

If successfully configured as a master, you will see:

--------------------------------------------------------------------------------

You have accepted the **CyberArk End User License Agreement**.

Use `evoke eula show` in the future to display the EULA.

--------------------------------------------------------------------------------

Configuration successful. Conjur master up and running.

Now add the conjur-cli container:

docker container run -d --name=conjur-cli --restart unless-stopped --entrypoint "" cyberark/conjur-cli:5 sleep infinity

Next, add entries to the /etc/hosts files of your laptop, conjur-cli and conjur-appliance containers:

echo your-ip-address your-fqdn >> /etc/hosts

To edit the /etc/hosts files within the containers:

docker exec -it conjur-appliance bash

echo your-ip-address your-fqdn >> /etc/hosts

docker exec -it conjur-cli bash

echo your-ip-address your-fqdn >> /etc/hosts

to test the edit to the /etc/hosts file:

curl -k <https://your-fqdn/health>

you should see the health output:

root@a1424849a611:/# curl -k https://mymacbook.myorg.local/health

{

"services": {

"ui": "ok",

"ldap-sync": "disabled",

"possum": "ok",

"ok": true

},

"database": {

"ok": true,

"connect": {

"main": "ok"

},

"free\_space": {

"main": {

"kbytes": 49755884,

"inodes": 3635763

}

},

"replication\_status": {

"pg\_current\_xlog\_location": "0/1825FC0",

"pg\_current\_xlog\_location\_bytes": 25321408

}

},

"ok": true

Next lets initialize Conjur:

Type: conjur init

Enter the URL of your Conjur service:

SHA1 Fingerprint=1D:D0:DD:B7:8D:53:C6:BB:5A:7E:16:5A:5F:08:DA:14:25:8A:99:18

Please verify this certificate on the appliance using command:

openssl x509 -fingerprint -noout -in ~conjur/etc/ssl/conjur.pem

Trust this certificate (yes/no):yes

(type Yes)

Enter your organization account name:myorg

(enter your Conjur Account from the configure master command)

Wrote certificate to /root/conjur-myorg.pem

Wrote configuration to /root/.conjurrc

Now you need to login to Conjur:

Type: conjur authn login

Enter your username to log into Conjur: (admin)

Please enter your password (it will not be echoed): MyCyber@rk01 (this is the password you specified in the master configure command)

Logged in

root@a1424849a611:/#

Now you have a master deployed, and the conjur-cli container logged in to the master for configuration and policy management.

Next, open another terminal window for the Kubernetes configuration.

In customer deployments, the Kubernetes Administrator will be doing most of this side of the configuration.

What you will need for Kubernetes:

Conjur-role.yml

KubernetesFollower.yml

What you will need for Conjur:

Host-authn-k8s.yml

Authn-k8s.yml

Seedservice.yml

Seedfetcher.yml

Conjur-role.yml:

# conjur-role.yml

# Service Account: dap-authn-service

# Namespace: dap-namespace

# Authenticator: dap-authenticator

# App Label: app-conjur

---

apiVersion: v1

kind: Namespace

metadata:

name: dap-namespace #must match the namespace in the seedfetcher.yml file

labels:

name: dap-namespace #must match the namespace in the seedfetcher.yml file

---

apiVersion: v1

kind: ServiceAccount

metadata:

name: dap-authn-service #must match the service account in the seedfetcher.yml file

namespace: dap-namespace #must match the namespace in the seedfetcher.yml file

---

apiVersion: rbac.authorization.k8s.io/v1

kind: ClusterRole

metadata:

name: dap-authenticator #must match the authenticator in the seedfetcher.yml file

rules:

- apiGroups: [""]

resources: ["pods", "serviceaccounts"]

verbs: ["get", "list"]

- apiGroups: ["extensions"]

resources: [ "deployments", "replicasets"]

verbs: ["get", "list"]

- apiGroups: ["apps"]

resources: [ "deployments", "statefulsets", "replicasets"]

verbs: ["get", "list"]

- apiGroups: [""]

resources: ["pods/exec"]

verbs: ["create", "get"]

---

kind: ClusterRoleBinding

apiVersion: rbac.authorization.k8s.io/v1

metadata:

name: dap-authenticator #must match the authenticator in the seedfetcher.yml file

subjects:

- kind: ServiceAccount

name: dap-authn-service #must match the service account in the seedfetcher.yml file

namespace: dap-namespace #must match the namespace in the seedfetcher.yml file

roleRef:

kind: ClusterRole

name: dap-authenticator #must match the authenticator in the seedfetcher.yml file

apiGroup: rbac.authorization.k8s.io

host-authn-k8s.yml:

---

- &hosts

- !host

id: authn-k8s

...

Authn-k8s.yml:

# =================================================

# Creates variables, webservice and consumers role for K8s authentication

# Note 2-space indentation – this is created “under” the conjur policy

# Service Account: dap-authn-service

# Namespace: dap-namespace

# Authenticator: dap-authenticator

#

# =================================================

- !policy

id: conjur/authn-k8s/prod #Must match AUTHENTICATOR\_ID in KubernetesFollower.yml

owner: !host /authn-k8s #Must match CONJUR\_AUTHENTICATORS in KubernetesFollower.yml

annotations:

description: authn-k8s defs for the cluster

body:

# variables that must be populated in Conjur

- !variable kubernetes/service-account-token

- !variable kubernetes/ca-cert

- !variable kubernetes/api-url

# vars for CA for this authenticator ID

- !variable ca/cert

- !variable ca/key

- !webservice

annotations:

description: authn service for cluster

# Hosts that can authenticate become members of the

# `consumers` group.

- !group consumers

# Grant consumers group role authentication privileges

- !permit

role: !group consumers

privilege: [ read, authenticate ]

resource: !webservice

seedservice.yml:

# =================================================

# Create webservice and consumers role for the Seed Service

# Note 4-space indentation – this is created “under” the authn-k8s policy

# =================================================

- !policy

id: conjur/seed-generation

body:

- !webservice

# Hosts that generate seeds become members of the

# `consumers` layer.

- !group consumers

# Authorize `consumers` to request seeds

- !permit

role: !group consumers

privilege: [ "execute" ]

resource: !webservice

seedfetcher.yml:

# =================================================

# == Create seed-service client identity for Follower initialization

# =================================================

# Define Follower host identity for authentication service in CyberArk namespace

- !host

id: dap-authn-service #Must match CONJUR\_AUTHN\_LOGIN in KubernetesFollower.yml

annotations:

authn-k8s/namespace: dap-namespace #Must match conjur-role.yml

authn-k8s/service\_account/name: dap-authn-service # Must match conjur-role.yml

authn-k8s/authentication-container-name: dap-authenticator # Must match conjur-role.yml

# Grant roles that gives Follower host identity permission to:

# - authenticate to the cluster authn-k8s endpoint

# - execute the seed-generation webservice

- !grant

roles:

- !group conjur/authn-k8s/prod/consumers

- !group conjur/seed-generation/consumers

members:

- !host dap-authn-service

KubernetesFollower.yml:

---

apiVersion: apps/v1

kind: Deployment

metadata:

name: dap-follower

namespace: dap-namespace #Must match conjur-role.yml and seedfetcher.yml

labels:

app: dap-follower

spec:

replicas: 2

selector:

matchLabels:

role: access

template:

metadata:

labels:

role: access

spec:

serviceAccountName: dap-authn-service #Must match conjur-role.yml and seedfetcher.yml

volumes:

- name: seedfile

emptyDir:

medium: Memory

- name: conjur-token

emptyDir:

medium: Memory

initContainers:

- name: dap-authenticator # Must match conjur-role.yml and seedfetcher.yml

image: cyberark/dap-seedfetcher:latest. #must match seedfetcher image in local repo

imagePullPolicy: IfNotPresent

env:

- name: CONJUR\_SEED\_FILE\_URL

value: https://mymacbook.myorg.local/configuration/myorg/seed/follower



- name: SEEDFILE\_DIR

value: /tmp/seedfile

- name: FOLLOWER\_HOSTNAME

value: dap-follower

- name: AUTHENTICATOR\_ID

value: prod #Must match authn-k8s.yml file

- name: CONJUR\_ACCOUNT

value: myorg #Must match Conjur Account from configure master command

- name: CONJUR\_SSL\_CERTIFICATE

valueFrom:

configMapKeyRef:

name: server-certificate #Must match configmap

key: ssl-certificate

- name: MY\_POD\_NAME

valueFrom:

fieldRef:

fieldPath: metadata.name

- name: MY\_POD\_NAMESPACE

valueFrom:

fieldRef:

fieldPath: metadata.namespace

- name: MY\_POD\_IP

valueFrom:

fieldRef:

fieldPath: status.podIP

- name: CONJUR\_AUTHN\_LOGIN

value: "host/dap-authn-service" #must match seedfetcher.yml

volumeMounts:

- name: seedfile

mountPath: /tmp/seedfile

- name: conjur-token

mountPath: /run/conjur

containers:

- name: node

imagePullPolicy: IfNotPresent

image: registry.tld/conjur-appliance:11.4.0 #Must match image in local repo

command: ["/tmp/seedfile/start-follower.sh"]

env:

- name: CONJUR\_AUTHENTICATORS

value: "authn-k8s/prod" #Must match authn-k8s.yml

- name: SEEDFILE\_DIR

value: /tmp/seedfile

ports:

- containerPort: 443

protocol: TCP

- containerPort: 5432

protocol: TCP

- containerPort: 1999

protocol: TCP

readinessProbe:

httpGet:

path: /health

port: 443

scheme: HTTPS

initialDelaySeconds: 15

timeoutSeconds: 5

volumeMounts:

- name: seedfile

mountPath: /tmp/seedfile

readOnly: true

resources:

requests:

memory: "1Gi"

cpu: "250m"

limits:

memory: "2Gi"

cpu: "500m"

---

apiVersion: v1

kind: Service

metadata:

name: access

namespace: dap-namespace #Must match conjur-role.yml and seedfetcher.yml

spec:

ports:

- name: https

port: 443

targetPort: 443

selector:

role: access

type: ClusterIP

---

apiVersion: autoscaling/v1

kind: HorizontalPodAutoscaler

metadata:

name: dap-follower

namespace: dap-namespace #Must match conjur-role.yml and seedfetcher.yml

spec:

scaleTargetRef:

apiVersion: apps/v1

kind: Deployment

name: dap-follower

minReplicas: 1

maxReplicas: 10

targetCPUUtilizationPercentage: 10

Now it’s time to apply the configuration. We will follow this order when deploying the followers:

1. Apply conjur-role.yml manifest in Kubernetes
2. Load the host-authn-k8s.yml policy in Conjur CLI
3. Load the authn-k8s.yml policy in Conjur CLI
4. Enable the authenticator
5. Load the seedservice.yml policy in Conjur CLI
6. Load the seedfetcher.yml policy in Conjur CLI
7. Create configMap for /opt/conjur/etc/ssl/conjur.pem in Kubernetes
8. Populate variables in Conjur:

kubernetes/service-account-token

kubernetes/ca-cert

kubernetes/api-url

ca/cert

ca/key

1. Apply the conjur-role.yml manifest in Kubernetes

Kubectl apply -f conjur-role.yml

1. Load the host-authn-k8s.yml policy in Conjur CLI

Conjur policy load root host-authn-k8s.yml

1. Load the authn-k8s.yml policy in Conjur CLI

Conjur policy load root authn-k8s.yml

1. Enable the authenticator #Matches the authn-k8s.yml policy id

docker exec conjur-appliance evoke variable set CONJUR\_AUTHENTICATORS authn,authn-k8s/prod

1. Load the seedservice.yml policy in Conjur CLI

Conjur policy load root seedservice.yml

1. Load the seedfetcher.yml policy in Conjur CLI

Conjur policy load root seedfetcher.yml

1. Create a ConfigMap for the /opt/conjur/etc/ssl/conjur.pem in Kubernetes

(Matches the config map in KubernetesFollower.yml)

kubectl create configmap server-certificate --from-file=ssl-certificate=/tmp/conjur.pem -n dap-namespace

(First you must copy the conjur.pem file to the /tmp folder from the /opt/conjur/etc/ssl directory of the conjur master)

1. Apply the KubernetesFollowers.yml Manifest in Kubernetes

Kubectl apply -f KubernetesFollowers.yml

1. Populate the variables:
2. kubernetes/service-account-token:

conjur variable values add conjur/authn-k8s/prod/kubernetes/service-account-token

1. kubernetes/ca-cert:

conjur variable values add conjur/authn-k8s/prod/kubernetes/ca-cert

1. kubernetes/api-url:

conjur variable values add conjur/authn-k8s/prod/kubernetes/api-url “<https://kubernetes.docker.internal:6443>”

1. ca/cert:

conjur variable values add conjur/authn-k8s/prod/ca/cert

1. ca/key:

conjur variable values add conjur/authn-k8s/prod/ca/key

1. To get the Kubernetes/service-account-token

First, list the secrets in your namespace:

Kubectl get secrets -n dap-namespace

Which displays:

dap-authn-service-token-4tmtr kubernetes.io/service-account-token 3 27m

default-token-qqkdw kubernetes.io/service-account-token 3 27m

(since dap-authn-service is the service account, dap-authn-service-token-4tmtr is the service account token)

Kubectl get secret -n dap-namespace dap-authn-service-token-4tmtr -o json

Which displays:

Kubectl get secret -n dap-namespace dap-authn-service-token-4tmtr -o json

{

"apiVersion": "v1",

"data": {

"ca.crt": "",

"namespace": "ZGFwLW5hbWVzcGFjZQ==",

"token": ""

},

"kind": "Secret",

"metadata": {

"annotations": {

"kubernetes.io/service-account.name": "dap-authn-service",

"kubernetes.io/service-account.uid": "cf9a3707-2bee-42b1-8022-6e0e9958f986"

},

"creationTimestamp": "2020-06-30T15:38:02Z",

"name": "dap-authn-service-token-4tmtr",

"namespace": "dap-namespace",

"resourceVersion": "9607",

"selfLink": "/api/v1/namespaces/dap-namespace/secrets/dap-authn-service-token-4tmtr",

"uid": "7b33c019-2be9-4354-9143-81a29553cee3"

},

"type": "kubernetes.io/service-account-token"

}

The highlighted portion is the service-account-token that must be base64 decoded. You can copy the highlighted text to a file and then run this command:

Base64 -d filename >> decodedfilename

Then you can copy the contents of decodedfilename to the Kubernetes/service-account-token variable

conjur variable values add conjur/authn-k8s/prod/kubernetes/service-account-token “contents of decodedfilename”

1. To get the Kubernetes/ca-cert:

Kubectl get secret -n dap-namespace dap-authn-service-token-4tmtr -o json

Which displays:

Kubectl get secret -n dap-namespace dap-authn-service-token-4tmtr -o json

{

"apiVersion": "v1",

"data": {

"ca.crt": "",

"namespace": "ZGFwLW5hbWVzcGFjZQ==",

"token": ""

},

"kind": "Secret",

"metadata": {

"annotations": {

"kubernetes.io/service-account.name": "dap-authn-service",

"kubernetes.io/service-account.uid": "cf9a3707-2bee-42b1-8022-6e0e9958f986"

},

"creationTimestamp": "2020-06-30T15:38:02Z",

"name": "dap-authn-service-token-4tmtr",

"namespace": "dap-namespace",

"resourceVersion": "9607",

"selfLink": "/api/v1/namespaces/dap-namespace/secrets/dap-authn-service-token-4tmtr",

"uid": "7b33c019-2be9-4354-9143-81a29553cee3"

},

"type": "kubernetes.io/service-account-token"

}

The highlighted portion is the ca-cert that must be base64 decoded. You can copy the highlighted text to a file and then run this command:

Base64 -d ca-filename >> decoded-ca

Then you can copy the contents of decoded-ca to the Kubernetes/ca-cert variable

conjur variable values add conjur/authn-k8s/prod/kubernetes/ca-cert “contents of decoded-ca”

1. To get the api-url: kubectl config view --minify -o json

(address under cluster server)

Kubectl config view --minify -o json

{

"kind": "Config",

"apiVersion": "v1",

"preferences": {},

"clusters": [

{

"name": "docker-desktop",

"cluster": {

"server": "https://kubernetes.docker.internal:6443",

"certificate-authority-data": "DATA+OMITTED"

}

}

],

"users": [

{

"name": "docker-desktop",

"user": {

"client-certificate-data": "REDACTED",

"client-key-data": "REDACTED"

}

}

],

"contexts": [

{

"name": "docker-desktop",

"context": {

"cluster": "docker-desktop",

"user": "docker-desktop"

}

}

],

"current-context": "docker-desktop"

1. To get the ca/cert, go to the conjur-appliance container in the /opt/conjur/etc/ssl directory and copy the contents of the ca.pem file:

Cat /opt/conjur/etc/ssl/ca.pem

Which displays:

root@c7b93adfe32f:/# cat /opt/conjur/etc/ssl/ca.pem

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

MIIDwjCCAqqgAwIBAgIUesv70zFZBl57ObbkOnNOdP2lGS4wDQYJKoZIhvcNAQEL

BQAwRDEOMAwGA1UECgwFbXlvcmcxEjAQBgNVBAsMCUNvbmp1ciBDQTEeMBwGA1UE

AwwVbXltYWNib29rLm15b3JnLmxvY2FsMB4XDTIwMDYzMDE0MzY1N1oXDTMwMDYy

ODE0MzY1N1owRDEOMAwGA1UECgwFbXlvcmcxEjAQBgNVBAsMCUNvbmp1ciBDQTEe

MBwGA1UEAwwVbXltYWNib29rLm15b3JnLmxvY2FsMIIBIjANBgkqhkiG9w0BAQEF

AAOCAQ8AMIIBCgKCAQEA0jtCIxK6mufGL+lRrK4I14wWYoLckCyAeUBfyxfTy1IJ

udFmmgiwpv3g4sM8RSyh7SuiygTe7BO6Uwgt7sS2HQkmt8wHERiKhTH9IrPJ2nwc

gjuAXSrJEtGnNNn+7uO2oDJ6EjSIspoUYw9IzaOjltkV/hKvVkuhZ1a8BDXKkooS

DjkuNuxPeVHQMX9ITNnG11Ar2sIBiRtNRcCSkUN28QPVf8+CQ3TJHmMfpkIGughS

jZWdgus0J997AKbMZenrCZN7bls/Voe3gAvnZ7M3JzCF3l+Dz7fexo9sEBqCdEjF

zfRtZ21+CBj7FIPs5kCeXy96wqqHoTCfj8erAu4ZaQIDAQABo4GrMIGoMEsGA1Ud

EQREMEKCFW15bWFjYm9vay5teW9yZy5sb2NhbIIVbXltYWNib29rLm15b3JnLmxv

Y2FsgghTdGFuZGJ5MYIIU3RhbmRieTIwHQYDVR0OBBYEFChiyMHQ0fvJnKfNtvAZ

s/wG7s7TMB8GA1UdIwQYMBaAFChiyMHQ0fvJnKfNtvAZs/wG7s7TMAwGA1UdEwQF

MAMBAf8wCwYDVR0PBAQDAgHmMA0GCSqGSIb3DQEBCwUAA4IBAQCy1tcfJapNhNEK

l6/GehTgE/98wGC+F/YpwdlqiV3zz06F/lJfFrczupySsFADWd0J25RCNUOYETpJ

N8yFhsF5K3zp/B31G2nC6Bg3vBLrd9Skhf+fuHcT8XwN3GFh0rqMek75mOGvsw4X

lQobLPjIfW7sl0RO2SdTp4LmmNoK/0BjYDlN2JUyl2K3Rj/wxqnsXVNDl0pZB/KI

h/t/7g8CgVSJjBNCEoS94DGMylglH1RqOEVYuVy+F+bByV4jj/FiC8Gvnw+6549a

EUhL6no/XX5B0mpICE6gPFLdzZh+N/mbaENqB7bLQmhnU6ZCLHFnCAm5bAcUxNjD

RdY9EXJ0

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

Then copy the highlighted text and populate the variable with:

conjur variable values add conjur/authn-k8s/prod/ca/cert “paste highlighted text”

1. To get the ca/key, go to the conjur-appliance container in the /opt/conjur/etc/ssl directory and copy the contents of the ca.key file:

Cat /opt/conjur/etc/ssl/ca.key

Which displays:

root@c7b93adfe32f:/# cat /opt/conjur/etc/ssl/ca.key

-----BEGIN PRIVATE KEY-----

MIIEvwIBADANBgkqhkiG9w0BAQEFAASCBKkwggSlAgEAAoIBAQDSO0IjErqa58Yv

6VGsrgjXjBZigtyQLIB5QF/LF9PLUgm50WaaCLCm/eDiwzxFLKHtK6LKBN7sE7pT

CC3uxLYdCSa3zAcRGIqFMf0is8nafByCO4BdKskS0ac02f7u47agMnoSNIiymhRj

D0jNo6OW2RX+Eq9WS6FnVrwENcqSihIOOS427E95UdAxf0hM2cbXUCvawgGJG01F

wJKRQ3bxA9V/z4JDdMkeYx+mQga6CFKNlZ2C6zQn33sApsxl6esJk3tuWz9Wh7eA

C+dnszcnMIXeX4PPt97Gj2wQGoJ0SMXN9G1nbX4IGPsUg+zmQJ5fL3rCqoehMJ+P

x6sC7hlpAgMBAAECggEALcDV0PpHgGUG3fCCZPlvVTHYwVHHnN0CjOD0uCeUYYCW

O0mWcdmIUJ+aLN/eoxNmnk5L3HmY5noYp+90x60qEUGnmoKd2xujMf/CHqZQPOub

7L/0XNJt9lmIcVksQWNvoIe7x3CFe5FK9JAkARUnCjJutd/tJvPIycu3mZb3YlKi

9CPKe0nBt4pOhDgRG6j8nRJLUe5kNeVus3oCgDCFYdwO/xbrV+xG6L8tZLSchnCa

/NnG23jzoMsa815HSwu/KpsLti7rkE6/XMa9voENXvc1KFG8o+WoKiaGjcuhvf4v

aOI4/toFYR5qAGwsy/wmpl4B4HUuoNn+VRMubICFgQKBgQD7S8NjfTerBzvDpDbj

Z4IJBOgRhRciMmqjINpKNopwPhLltjB6nZCC8YZRlMszyudWbiYxH2HXvUIZNybq

ZZMQ1tvj0c3cpd2QNtXHbv8oE5szZ5IFXmkA6mAD1xEdscjSfK3vfLjR/eSdatlw

W9ngUexpvIqAw/yorxvBCrXksQKBgQDWKrW2o7t7DQ2qge3ItmxuZgj39YpPrGIA

34QDyZKR9HIGb5xdn9T3j2qYjDtpADtA+6+c09N3/XBIKoHc/OR2Zd3e6r4jZkMr

sSGwqw/HNdGXUrphI6OwuRVZ5OP/RHwDM2EzkpiYq76/N0tx7h/VkrKTjifwyWmb

95CNKdSOOQKBgQDauPzdkVVspSUdicbEc4k49G5RIXgbKaqrRaL5yTMBsuWbztGv

R4lKj5ILW+jpsYddMQ5I3zm4H2BythBHIBW7KGfHiQsXK78HENNNfIzVgwso2NK9

nYjjZRwg+4ZIRD+5BRAXTIVJMSVU5nVPWILwvh15gsNChFNbNt5/TSU4YQKBgQCV

iBCDyuIV0lFeYYVihGxE0J0mc5v0cQmkTmtgMZdSrPgIIllszTHm/42PoSnB8FVu

08wTlZqqxptk57jwUOo6+HjnMcM7+CSiFvTM40MK9jq2ix5ywG2D8FZGA5hlHK3x

ligh2BMPVarC25qjm31Co0Bn4fNlHVvNeBbHWRN9kQKBgQDQlh0YNAyGroiROzEp

B4OzLw21E5UEDgJb2p9QnIoRmQTxjxtLGvDY6dQdHw+ZVwAdD0ICFCmMwPkXkMou

I3ZaXJWVmxoAGNGvhIZx7EVxTcSs1HwwLjcsbEgzcA5al7vvGNVppRGnLC1NIOND

crDQeuktbA9erOL3Y+cJPnuRww==

-----END PRIVATE KEY-----

Then copy the highlighted text and populate the variable with:

conjur variable values add conjur/authn-k8s/prod/ca/key “paste highlighted text”

To test the status of the replicas that you have configured:

Kubectl get pods -n dap-namespace

If successful, you will eventually (after about a minute) see this output:

JOSHUA-M-MAC:MyK8s\_Files jmoses$ kubectl get pods -n dap-namespace

NAME READY STATUS RESTARTS AGE

dap-follower-fbf9b5864-l72ds 1/1 Running 0 28m