CyberArk Conjur Enterprise and Openshift CRC LAB

By huy.do@cyberark.com

PART I: SETING UP ENVIRONMENT

1. LAB Prerequisites

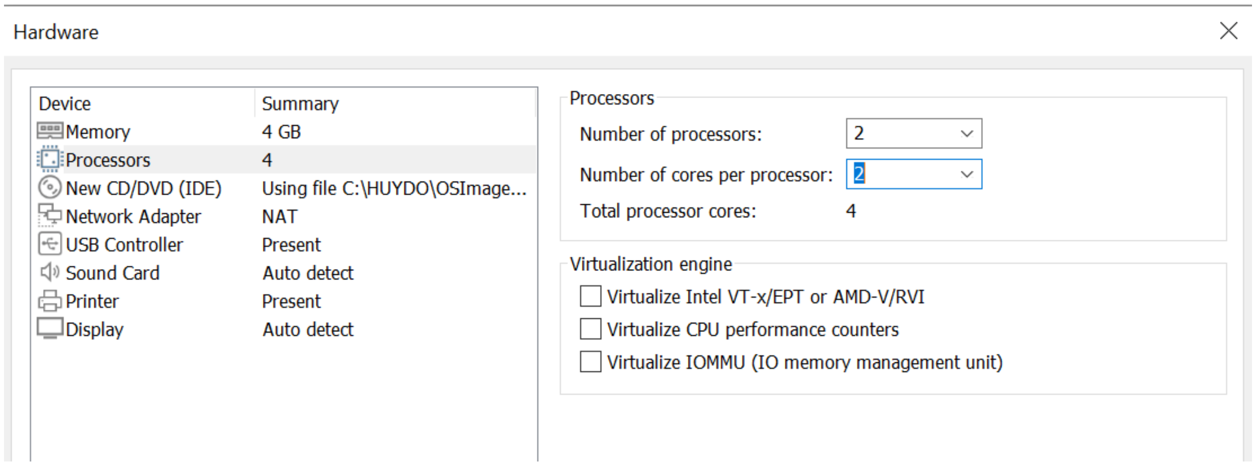
- ESXI server or VMWorkstation to create 02 lab VMs as below:
  - Conjur Master VM (ConjurVM)
    - 4GB RAM
    - 2 vCore CPU
    - 60GB HDD
    - CentOS 8 base OS (Minimal Install)
      - Hostname: conjur-master.cyberarkdemo.local
      - DNS server enabled with dnsmasq
      - Static IP (eg 172.16.100.61/24)
  - Openshift CRC VM (CRCVM)
    - 16GB RAM
    - 4 vCores CPU with Hardware virtualization enabled
    - 120 GB HDD
    - CentOS 8 base OS (Minimal Install)
      - Hostname: ocp-crc.cyberarkdemo.local
      - Static IP (eg 172.16.100.62/24)
- Conjur docker images & utilities:
  - Contact CyberArk local representative for following docker images
    - conjur-appliance_12.4.0.tar.gz
    - conjur-authn-k8s-client_0.22.0.tar.gz
    - dap-seedfetcher_0.3.1.tar.gz
    - secretless-broker_1.7.8.tar.gz
  - Conjur CLI v7: https://github.com/cyberark/conjur-api-python3/releases
- Redhat Account to login to https://console.redhat.com/openshift (register for free)
- Download CodeReady Containers and Pull Secret
  - Download CRC for Linux https://console.redhat.com/openshift/create/local
  - Dowload pull secret and store in file: pull-secret.txt

The IP addresses in this document is using from current lab environment. Please replace the ***172.16.100.62*** *by your* ***CRCVM****'s and* ***172.16.100.61*** *by your* ***ConjurVM****'s IPs*.

2. VMs Preparation

Step1: Creating ConjurVM and CRCVM

Hardware configuration for ConjurVM



Hardware

Device	Summary
Memory	4 GB
Processors	4
New CD/DVD (IDE)	Using file C:\HUYDO\OSImage...
Network Adapter	NAT
USB Controller	Present
Sound Card	Auto detect
Printer	Present
Display	Auto detect

Processors

Number of processors: 2

Number of cores per processor: 2

Total processor cores: 4

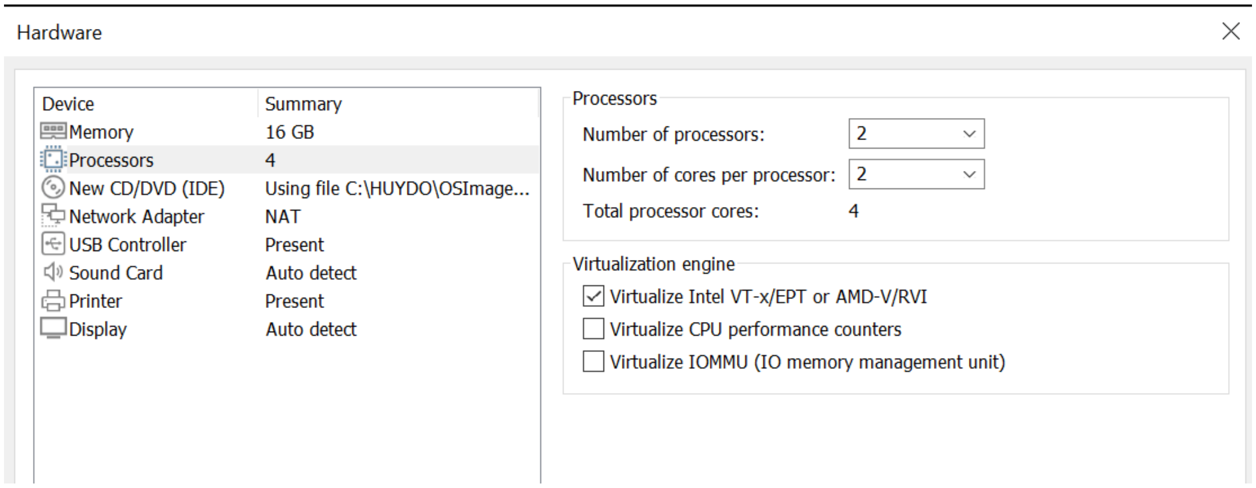
Virtualization engine

☐ Virtualize Intel VT-x/EPT or AMD-V/RVI

☐ Virtualize CPU performance counters

☐ Virtualize IOMMU (IO memory management unit)

For CRCVM, check the Virtualize options to enable virtualization capability



Hardware

Device	Summary
Memory	16 GB
Processors	4
New CD/DVD (IDE)	Using file C:\HUYDO\OSImage...
Network Adapter	NAT
USB Controller	Present
Sound Card	Auto detect
Printer	Present
Display	Auto detect

Processors

Number of processors: 2

Number of cores per processor: 2

Total processor cores: 4

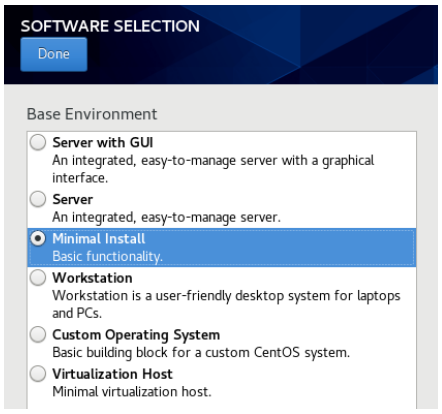
Virtualization engine

☑ Virtualize Intel VT-x/EPT or AMD-V/RVI

☐ Virtualize CPU performance counters

☐ Virtualize IOMMU (IO memory management unit)

Installing OS with minimal packages



SOFTWARE SELECTION

Done

Base Environment

○ **Server with GUI**
An integrated, easy-to-manage server with a graphical interface.

○ **Server**
An integrated, easy-to-manage server.

● **Minimal Install**
Basic functionality.

○ **Workstation**
Workstation is a user-friendly desktop system for laptops and PCs.

○ **Custom Operating System**
Basic building block for a custom CentOS system.

○ **Virtualization Host**
Minimal virtualization host.

Step2: Installing docker in both ConjurVM and CRCVM

Login to both VMs as root, checking for Internet connection and running below commands to install docker engine and creating new user for lab environment

```
yum remove docker \
                  docker-client \
                  docker-client-latest \
                  docker-common \
                  docker-latest \
                  docker-latest-logrotate \
                  docker-logrotate \
                  docker-engine
yum install -y yum-utils
yum-config-manager \
    --add-repo \
    https://download.docker.com/linux/centos/docker-ce.repo
yum -y update
yum install -y docker-ce docker-ce-cli containerd.io
yum install -y git

systemctl enable docker
systemctl start docker
```

Step3: Copying necessary files to installation folder

Login to both machines as root and do following commands

```
mkdir -p /opt/lab/setup_files
chmod 777 /opt/lab/setup_files
```

Copying all images and files to ~/setup_files folder. Below are files that will be needed for this lab.

ConjurVM:

- Conjur docker image: conjur-appliance_12.4.0.tar.gz

CRCVM:

- Conjur docker image: conjur-appliance_12.4.0.tar.gz
- Conjur k8s authentication client: conjur-authn-k8s-client_0.22.0.tar.gz
- Conjur seedfetcher: dap-seedfetcher_0.3.1.tar.gz
- Conjur secretless broker: secretless-broker_1.7.8.tar.gz
- CRC Linux installation file: crc-linux-amd64.tar.xz
- CRC pullsecret file: pull-secret.txt

Step4: Clone installation scripts and change config.sh data

Login to both machines as root and do following commands

```
cd /opt/lab
git clone https://github.com/huydd79/conjur-openshift-lab.git
```

Edit /opt/lab /conjur-openshif-lab/conjur-vm/00.config.sh and /opt/lab /conjur-openshif-lab/crc-vm/00.config.sh, change the parameters to meet your local lab (IP addresses, domain, setup_files folder…). Set READY=true when done.

Step5: Setting DNS server on ConjurVM

Login to ConjurVM as root. Running below script to install DNS service on ConjurVM

```
cd /opt/lab/conjur-openshift-lab/conjur-vm/
./01.configure-dns-server.sh
```

Change the DNS server configuration on ConjurVM (/etc/resolve.conf) to use 127.0.0.1 as primary DNS server. Also change your console's DNS server to ConjurVM's IP and test for the DNS

```
nslookup abc.apps-crc.testing 127.0.0.1
nslookup anything.crc.testing 127.0.0.1
nslookup conjur-master.$LAB_DOMAIN 127.0.0.1
nslookup ocp-crc.$LAB_DOMAIN 127.0.0.1
```

Make sure you are still able to resolve other public internet domains such as google.com or cyberark.com

3. Setting up mysql container for LAB DB

Login to ConjurVM, run following scripts to set up and run mysql container. Default password to access to database is Cyberark1.

```
cd /opt/lab/conjur-openshift-lab/conjur-vm
./02.running-mysql-db.sh
```

If you want to test the mysql connection, run below commands

```
yum install -y mysql
mysql -u cityapp -p -h conjur-master.$LAB_DOMAIN
```

4. Setting up Conjur Master

Step1: Loading conjur appliance image

Copy conjur appliance image to ConjurVM, run and configure it as master

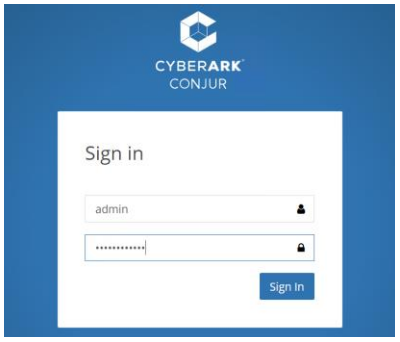
```
cd /opt/lab/conjur-openshift-lab/conjur-vm
./03.loading-conjur-images.sh
```

Step2: Start conjur container as master

Start the conjur container using below commands

```
cd /opt/lab/conjur-openshift-lab/conjur-vm
./04.start-conjur-container.sh
./05.configure-conjur-master.sh
```

Open browser and login to https://conjur-master.$LAB_DOMAIN with user admin and password set in 00.config.sh file



admin
••••••••••••

Login and review the conjur UI to complete this step

Step3: Installing ConjurCLI and loading demo data

Login to ConjurVM as root. Run below script to install and setup Conjur CLI in ConjurVM

```
cd /opt/lab/conjur-openshift-lab/conjur-vm
./06.install-conjur-cli.sh
```

Run below script to load demo data to conjur environment

```
cd /opt/lab/conjur-openshift-lab/conjur-vm
./07.loading-demo-data.sh
```

Step4: Enabling K8s authenticator

Login to ConjurVM as root and run follow script to enable K8s authenticator

```
cd /opt/lab/conjur-openshift-lab/conjur-vm
./08.enable-k8s-authenticator.sh
```

Using curl to doublecheck for the enabled authenticators

```
curl -k https://conjur-master.$LAB_DOMAIN/info
```

5. Setting up Openshift CRC Environment

Step1: CRC Installing and starting up

Login to CRCVM as crcuser, running following command to setup CRC environment

```
cd /opt/lab/conjur-openshift-lab/crc-vm
./01.installing-crc.sh
```

Su or relogin to CRCVM using crcuser and run below script to install openshift crc

```
su crcuser
cd /opt/lab/conjur-openshift-lab/crc-vm
./02.crc-setup.sh
```

This process will take few minutes (up to hours depends on Internet connection). About 11GB of data will need to be downloaded from Internet and installed to CRCVM.

After setup process completed successfully, run below script to start CRC environment. Take a copy of pullscret text in screen, press Enter when done and paste to runtime environment as requested.

```
cd /opt/lab/conjur-openshift-lab/crc-vm
./03.crc-start.sh
```

There will have some error notification related to ssh connection or pull secret, they are normal. If the starting process got failed, try to run crc delete and crc start again.

After the starting process completed successfully, take note of the kubeadmin and developer ‘s credential and login url. Checking for crc status and login to openshift crc cli console with below commands (note that there have an additional dot before a script path to pull result to environment)

```
cd /opt/lab/conjur-openshift-lab/crc-vm
. ./04.crc-login-admin.sh
```

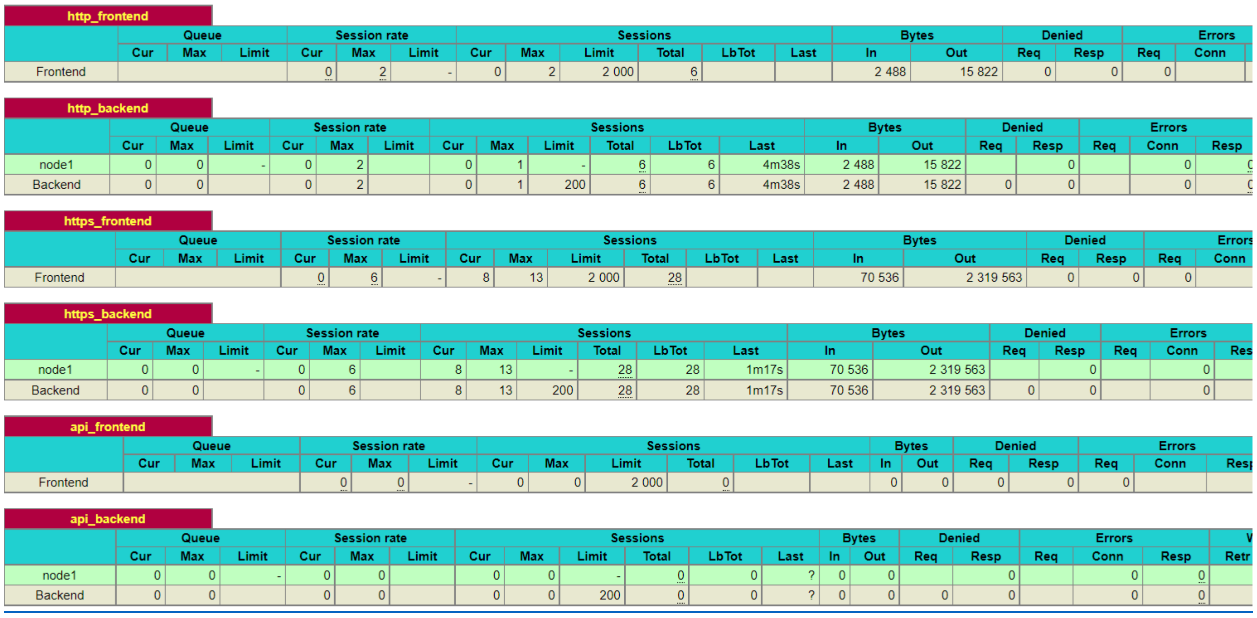
Step3: Installing HAProxy in CRCVM

Login to CRCVM as crcuser. Install and configure haproxy using below script

```
cd /opt/lab/conjur-openshift-lab/crc-vm
./05.install-haproxy.sh
```

Open browser and check for haproxy status:

http://console-openshift-console.apps-crc.testing:9000/stats



http_frontend

	Queue			Session rate			Sessions						Bytes		Denied		Errors	
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn
Frontend				0	2	-	0	2	2 000	6			2 488	15 822	0	0	0	

http_backend

	Queue			Session rate			Sessions						Bytes		Denied		Errors		
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp
node1	0	0	-	0	2		0	1	-	6	6	4m38s	2 488	15 822		0		0	0
Backend	0	0		0	2		0	1	200	6	6	4m38s	2 488	15 822	0	0		0	0

https_frontend

	Queue			Session rate			Sessions						Bytes		Denied		Errors	
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn
Frontend				0	6	-	8	13	2 000	28			70 536	2 319 563	0	0	0	

https_backend

	Queue			Session rate			Sessions						Bytes		Denied		Errors		
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Res
node1	0	0	-	0	6		8	13	-	28	28	1m17s	70 536	2 319 563		0		0	
Backend	0	0		0	6		8	13	200	28	28	1m17s	70 536	2 319 563	0	0		0	

api_frontend

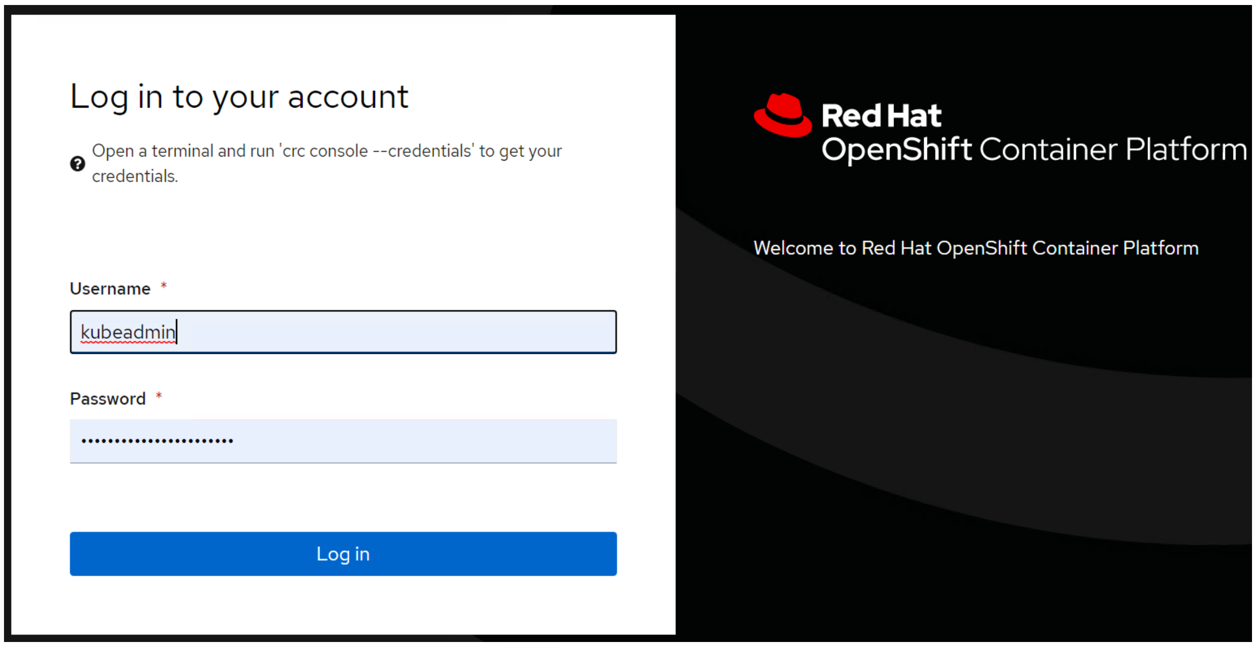
	Queue			Session rate			Sessions						Bytes		Denied		Errors		
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Res
Frontend				0	0	-	0	0	2 000	0			0	0	0	0	0		

api_backend

	Queue			Session rate			Sessions						Bytes		Denied		Errors			W
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out	Req	Resp	Req	Conn	Resp	Retr
node1	0	0	-	0	0		0	0	-	0	0	?	0	0		0		0	0	
Backend	0	0		0	0		0	0	200	0	0	?	0	0	0	0		0	0	

Login to CRC Web Console

https://console-openshift-console.apps-crc.testing



Log in to your account

Open a terminal and run 'crc console --credentials' to get your credentials.

Username *

kubeadmin

Password *

•••••••••••••••••••••••

Log in

Red Hat
OpenShift Container Platform

Welcome to Red Hat OpenShift Container Platform

6. Deploying Conjur Follower with Seed Fetcher

Step1: Installing Conjur CLI on CRCVM

Login to CRCVM as crcuser and run below script to install and setup conjur CLI environment.

```
cd /opt/lab/conjur-openshift-lab/crc-vm
./06.install-conjur-cli.sh
```

Step2: Pushing conjur docker images to OCP

Login to CRCVM as crcuser and run below commands to push all necessary images to Openshift CRC

```
cd /opt/lab/conjur-openshift-lab/crc-vm
./07.push-dap-images.sh
```

Checking crc registry result by using below command:

```
oc get is
```

Step3: Adding policies and variables for follower seed fetcher

Run below commands to add policies and values for variables

```
cd /opt/lab/conjur-openshift-lab/crc-vm
./08.configure-seedfetcher.sh
```

Login to conjur-master GUI and doublecheck for the value of below three variables

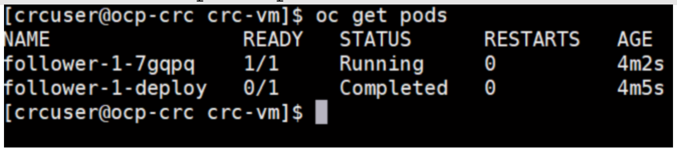
```
conjur/authn-k8s/okd/kubernetes/ca-cert
conjur/authn-k8s/okd/kubernetes/service-account-token
conjur/authn-k8s/okd/kubernetes/api-url
```

Step4: Deploy conjur follower with auto seed fetcher

```
cd /opt/lab/conjur-openshift-lab/crc-vm
./09.deploy-follower.sh
```

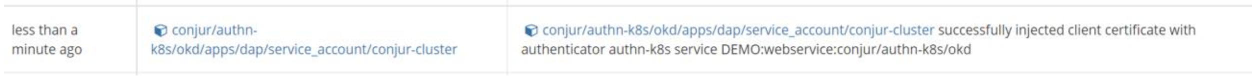
This command will load the conjur appliance image for the first time so it will take few minutes to complete pod starting up process. Using following command to check for result

```
oc get pods
oc describe pod <pod-id>
```



```
[crcuser@ocp-crc crc-vm]$ oc get pods
NAME                READY   STATUS      RESTARTS   AGE
follower-1-7gqpq    1/1     Running     0          4m2s
follower-1-deploy   0/1     Completed   0          4m5s
[crcuser@ocp-crc crc-vm]$
```

Checking dashboard in conjur GUI for authentication event using K8s



less than a minute ago | conjur/authn-k8s/okd/apps/dap/service_account/conjur-cluster | conjur/authn-k8s/okd/apps/dap/service_account/conjur-cluster successfully injected client certificate with authenticator authn-k8s service DEMO:webservice:conjur/authn-k8s/okd

Testing follower service using curl

```
curl -k https://follower-dap.apps-crc.testing/info
```

Note that this follower will not provide service outside of cluster because its certificate's subject is local name follower.dap.svc.cluster.local

PART II: CITYAPP USE CASES

7. CityApp Project

Cityapp is a demo container which run a ruby application to connect to mysql database and select a random data record of city population to show to web UI. Application will get information from environmental variables to connect to database server as below:

- DBAddress: IP or hostname of mysql server
- DBPort: mysql port
- DBName: name of database
- DBUsername: username to connect to database server. If this value is empty, application use ruby API to connect to conjur and get the username value
- DBPassword: password to connect to database server. If this value is empty, application use ruby API to connect to conjur and get the password value

Cityapp will need other environmental variables for conjur connection and secret lookup

- CONJUR_APPLIANCE_URL
- CONJUR_ACCOUNT
- CONJUR_AUTHN_TOKEN_FILE
- DBUsername_CONJUR_VAR
- DBPassword_CONJUR_VAR

To build Cityapp images for docker and push it to CRC environment, login to CRCVM as crcuser and run below script

```
cd /opt/lab/conjur-openshift-lab/crc-vm
./10.load-cityapp.sh
```

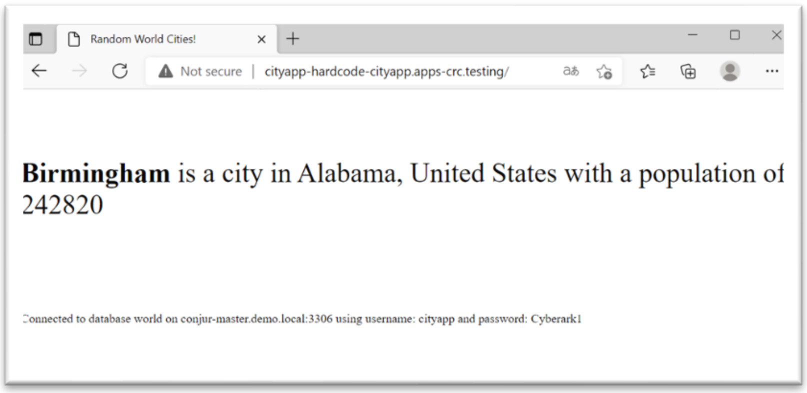
8. CityApp HardCode

Cityapp hardcode is container which will run with the database connection information hardcoded in the configuration file and pushed to environmental variable when running. Login to CRCVM as crcuser and run below steps to build the cityapp docker image and start application with hardcoded credential

To start CityApp hardcode pod, login to CRCVM as crcuser and run below scrip

```
cd /opt/lab/conjur-openshift-lab/crc-vm
./11.run-cityapp-hardcode.sh
```

Wait until cityapp-hardcode pods started and run, doublecheck the application status by using curl or web browser to open url: http://cityapp-hardcode-cityapp.apps-crc.testing/



Birmingham is a city in Alabama, United States with a population of 242820

Connected to database world on conjur-master.demo.local:3306 using username: cityapp and password: Cyberark1

Testing the result by using curl or open URL in browser

```
curl http://cityapp-hardcode-cityapp.apps-crc.testing/
```

9. CityApp Summon Init

Cityapp summon init use summon tool to retrieve DBUsername and DBPassword as secrets from conjur. Secrets are sent directly from summon to application over memory variables.

Summon take the access token from share drive to authenticate with conjur follower. The access token is created by initContainer using authen-k8s-client

Step1: Configuring authn-k8s-client

Login to CRCVM as crcuser and run below script to configure authn-k82-client

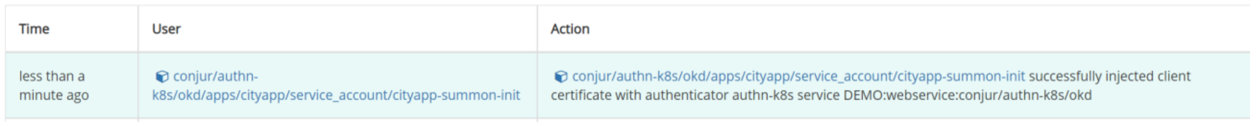
```
cd /opt/lab/conjur-openshift-lab/crc-vm
./13.config-authn-k8s-client.sh
```

Step2: Start Cityapp Summon Init

Login to CRCVM as crcuser and run below script to start cityapp summon init pods

```
cd /opt/lab/conjur-openshift-lab/crc-vm
./14.run-cityapp-summon-init.sh
```

Watching the conjur's GUI dashboard and waiting for authentication event as below



Time	User	Action
less than a minute ago	conjur/authn-k8s/okd/apps/cityapp/service_account/cityapp-summon-init	conjur/authn-k8s/okd/apps/cityapp/service_account/cityapp-summon-init successfully injected client certificate with authenticator authn-k8s service DEMO:webservice:conjur/authn-k8s/okd

Wait until all pods are up and running. Using browser or curl to check for app's url:

http://cityapp-summon-init-cityapp.apps-crc.testing/