



# CYBERARK DYNAMIC ACCESS PROVIDER INTEGRATION WORKSHOP

## Table of Contents

Introduction .....	2
Lab 1 – Configure Existing Jenkins Freestyle Project to Use DAP Secrets.....	2
Purpose: .....	2
Steps:.....	2
Lab 2 - Configure Jenkins Pipeline.....	8
Purpose: .....	8
Steps:.....	8
Lab 3 – Integrate your app to use Native K8's secrets.....	9
Purpose: .....	9
Steps:.....	9
Lab 4 – Deploy an app using continuous authentication.....	12
Purpose: .....	12
Steps:.....	12
Lab 5 – Deploy the Secretless Broker for an application .....	14
Purpose: .....	14
Steps:.....	15

## Introduction

This workshop is designed to show developers how to integrate their applications with CyberArk's Dynamic Access Provider (DAP). The workshop consists of a series of labs demonstrating how to convert existing applications, and deploy new applications using the CyberArk Dynamic Access Provider. The workshop will also show the integrations between the CyberArk Enterprise Password Vault and Dynamic Access Provider environment.

The Training environment consists of an administration workstation with Docker installed, a Kubernetes master server, two Kubernetes worker machines, a CyberArk Vault Server, and a CyberArk Components server. The CyberArk components server has a Central Policy Manager, Privileged Vault Web Access, and CyberArk Conjur Vault Synchronizer configured and installed. Please reference the table below for IP Addresses, and authentication information to the necessary environments.

Machine	IP	OS	NodePort	Admin User	Admin Password
k8s-master	10.0.0.3	Ubuntu		cybradmin	Cyberark1
k8s-worker1	10.0.0.4	Ubuntu		cybradmin	Cyberark1
k8s-worker2	10.0.0.5	Ubuntu		cybradmin	Cyberark1
admin-wkstn	10.0.0.9	Ubuntu Desktop		cybradmin	Cyberark1
DAP	10.0.0.9:443	Container		Admin	Cyberark1
Jenkins	10.0.0.9:8080	Container		cybradmin	Cyberark1
K8s Dashboard	<ul style="list-style-type: none"><li>https://k8s-master/{{NodePort}}</li><li><a href="https://k8s-worker1/{{NodePort}}">https://k8s-worker1{{NodePort}}</a></li><li>https://k8s-worker2{{NodePort}}</li></ul>	Container	32673	"skip"	

## Lab 1 – Configure Existing Jenkins Freestyle Project to Use DAP Secrets

### Purpose:

In Lab 1 you will convert an existing Jenkins Freestyle project to use Conjur secrets rather than the built-in Jenkins Credential Plugin. The CyberArk Vault Conjur Synchronizer has been setup for you, and the secret is already stored in the vault. There are two tasks. The first is to grant Jenkins access to secrets through policy in DAP. Second, we will configure the freestyle project to use the DAP secret, rather than a native Jenkins secret.

### Steps:

#### *Inspect current Freestyle Project*

1. Login to the Jenkins website by launching the Firefox Browser, clicking the Jenkins shortcut in the toolbar, then "Sign In". Use the "cybradmin" credentials that have been saved in the browser.

2. Click **Lab1\_DBLogin**, then click **Configure**
  - a. Briefly review the Bindings and Build sections to see what the project does:

**Bindings**

Username and password (separated)

Username Variable

Password Variable

Credentials ☒ Specific credentials ☐ Parameter expression

☐ Abort the build if it's stuck

☐ Add timestamps to the Console Output

☐ Inspect build log for published Gradle build scans

☐ With Ant

**Build**

Execute shell

Command `mysql --host=10.0.0.9 --user=$USER --password=$PASSWORD -e "show databases;"`

3. Click **Save**, then click **Build Now** to see the project work with the existing SSH Key.
4. Click on the **#1** in the left part of the page, then click **Console Output** in the left part of the page to see the hostname of the machine the project returns. You should see output like below:

## Console Output

```
Started by user cybradmin
Running as SYSTEM
Building in workspace /var/jenkins_home/workspace/LAB1_DBLogin
[LAB1_DBLogin] $ /bin/sh -xe /tmp/jenkins2331182559259405060.sh
+ mysql --host=10.0.0.9 --user=**** --password=**** -e show databases;
Database
information_schema
mysql
performance_schema
sys
Finished: SUCCESS
```

*Trigger a Password Change for the root MySQL credential in CyberArk EPV*

5. In Firefox, open a new browser window, then click the **Password Vault** shortcut in the toolbar, then login with the credentials saved in the browser.
6. Once logged in, find the root credential, and click anywhere on the line. Once the management pane appears, click **Change**, to trigger a change to the root MySQL account.

☆	Status	Username	Address
★ ⚡		root	10.0.0.9

Platform: MySQL Server Safe: database\_account

Overview Details Activities Version

Compliance Status **Compliant**

1 Days ago

Changed by PasswordManager  
Oct 23, 2019 2:32 PM

Reconcile Change

*Grant Jenkins Access to retrieve the new MySQL credential from DAP*

- Open a terminal of your choice on the admin workstation
- Type **docker container ls** to view the existing containers:

```
cybradmin@admin-wkstn:~$ docker container ls
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS
050229892e8f   mysql     "docker-entrypoint.s..." 41 minutes ago Up 41 minutes 0.0.0.0:3306->3306/tcp, 0.0.0.0:33060->33060/tcp
2878251df6e4   mysql     jenkins/jenkins:lts       42 minutes ago Up 42 minutes 0.0.0.0:8080->8080/tcp, 0.0.0.0:50000->50000/tcp
34996b699a27   jenkins   cyberark/conjur-cli:5-latest "sleep infinity"      6 hours ago   Up 6 hours
782d398bb16e   dap-cli   conjur-appliance:5.6.0    "/bin/keyctl session..." 6 hours ago   Up 6 hours 0.0.0.0:443->443/tcp, 0.0.0.0:1999->1999/tcp, 0.0.0.0:5432->5432/tcp
dap-master
```

All commands will be ran against the dap-cli container to avoid direct interaction with the master container.

- Run **docker exec dap-cli conjur list** to see current objects in the DAP environment. We need to grant the host jenkins-master access to the synced secrets for lab1. Although it is possible to grant access to the secret variables explicitly, it is much more efficient to grant the host access to the consumers group for the safe & policy. Your results should look similar to the screenshot below:

```
cybradmin@admin-wkstn:~$ docker exec dap-cli conjur list
[
  "acme:policy:root",
  "acme:policy:jenkins",
  "acme:group:epv-admins",
  "acme:host:Sync_HOST-1",
  "acme:policy:epv",
  "acme:group:epv/lab1-admins",
  "acme:policy:epv/lab1",
  "acme:group:epv/dbaccess-admins",
  "acme:policy:epv/dbaccess",
  "acme:group:epv/dbaccess/database_accounts-admins",
  "acme:policy:epv/dbaccess/database_accounts",
  "acme:policy:epv/dbaccess/database_accounts/delegation",
  "acme:group:epv/dbaccess/database_accounts/delegation/consumers",
  "acme:variable:epv/dbaccess/database_accounts/lab1/password",
  "acme:variable:epv/dbaccess/database_accounts/lab1/username",
  "acme:group:epv_admins",
  "acme:policy:lab1",
  "acme:policy:k8s-api",
  "acme:policy:lab3",
  "acme:host:jenkins/jenkins-master"
```

10. Review the policy already created for you to grant the Jenkins host access to synced secrets by typing **docker exec dap-cli cat /policy/lab1.yml**

```
cybradmin@admin-wkstn:~$ docker exec dap-cli cat /policy/lab1.yml
---
- !grant
  role: !group epv/dbaccess/database_accounts/delegation/consumers
  member: !host jenkins/jenkins-master
```

11. Apply the policy by typing **docker exec dap-cli conjur policy load root /policy/lab1.yml** Your output should look like the following:

```
cybradmin@admin-wkstn:~$ docker exec dap-cli conjur policy load root /policy/lab1.yml
Loaded policy 'root'
{
  "created_roles": {
  },
  "version": 6
}
```

*Trigger another build in Jenkins resulting in a failed build*

12. Back in the Jenkins console, trigger a build of Lab1, and notice that the job now fails. Let's inspect the Console Output again to see why it failed.
13. If not already in the LAB1\_JenkinsCredential management page, click **LAB1\_DBLogin**
14. Click the **red dot #2** on the left
15. Click **Console Output** on the left and inspect the results of the build. Notice the permission has been denied because the root password has changed.

## Console Output

```
Started by user cybradmin
Running as SYSTEM
Building in workspace /var/jenkins_home/workspace/LAB1_DBLogin
[LAB1_DBLogin] $ /bin/sh -xe /tmp/jenkins5709753201660654941.sh
+ mysql --host=10.0.0.9 --user=**** --password=**** -e show databases;
ERROR 1045 (28000): Access denied for user ****@'172.19.0.1' (using password: YES)
Build step 'Execute shell' marked build as failure
Finished: FAILURE
```

*Configure Jenkins to use secrets from DAP*

- At the home page for Jenkins, click **Credentials, System**, then **Global Credentials**. You'll see the root credential that was previously used, and a credential for Jenkins to authenticate to the Dynamic Access Provider.

### Global credentials (unrestricted)

Credentials that should be available irrespective of domain specification to requirements matching.

	Name	Kind
	<a href="#">root/*****</a>	Username with password
	<a href="#">host/jenkins/jenkins-master/*****</a>	Username with password

Icon: [S](#) [M](#) [L](#)

- Click **Add Credentials** on the left side of the page, then for kind, select **Conjur Secret Credential**, then enter the path for the lab1 password, give it an ID, then click OK:

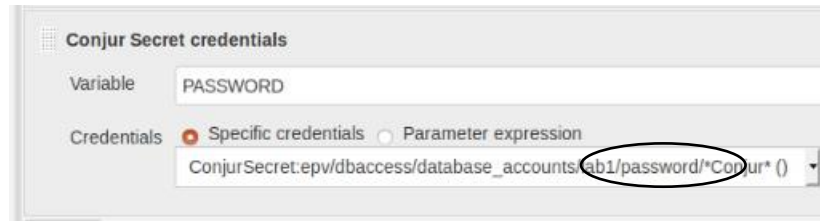
Kind	Conjur Secret Credential
Scope	Global (Jenkins, nodes, items, all child items, etc)
Variable Path	epv/dbaccess/database_accounts/lab1/password
ID	lab1-password
Description	

- Do the same for the lab1 username:

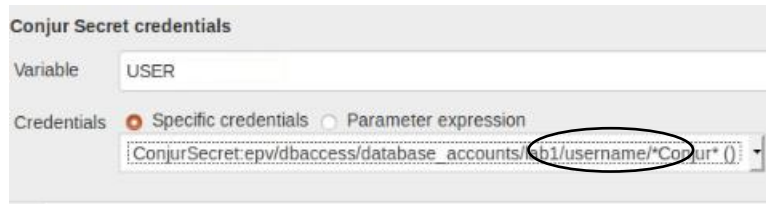
Kind	Conjur Secret Credential
Scope	Global (Jenkins, nodes, items, all child items, etc)
Variable Path	epv/dbaccess/database_accounts/lab1/username
ID	lab1-username
Description	

- Navigate to the Jenkins homepage, then **LAB1\_DBLogin, Configure**

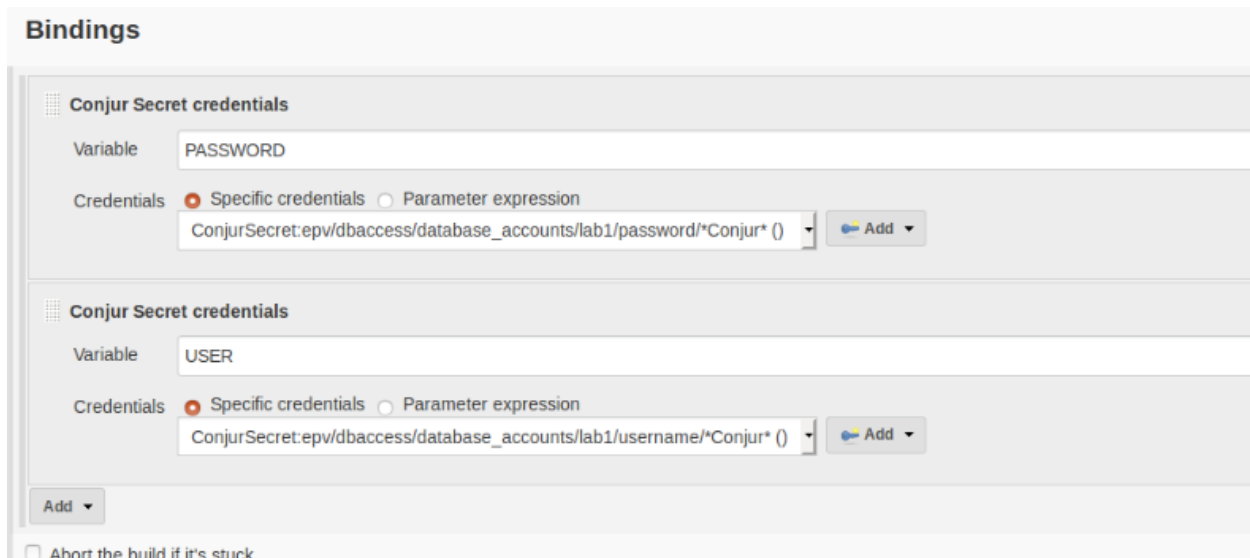
20. In the bindings section, below the Username and Password credential binding, click the **Add** button and select **Conjur Secret Credentials**. Ensure the selected credential in the dropdown is the password, then change the Variable Name to **PASSWORD**:



21. Repeat step 5, this time selecting the username, and setting the Variable Name to **USER**:



22. Delete the old binding for the root credential. Your Bindings section should look like this:



23. Click **Save**. Once the page reloads, click **Build Now** on the left. You should now see a successful build. You can verify the output by clicking on the final build number, then **Console Output**:



## Console Output

```
Started by user cybradmin
Running as SYSTEM
Building in workspace /var/jenkins_home/workspace/LAB1_DBLogin
[LAB1_DBLogin] $ /bin/sh -xe /tmp/jenkins8154225259688209612.sh
+ mysql --host=10.0.0.9 --user=**** --password=**** -e show databases;
Database
information_schema
mysql
performance_schema
sys
Finished: SUCCESS
```

## Lab 2 - Configure Jenkins Pipeline

### Purpose:

The purpose of this lab is to show you how to create a Jenkins Pipeline that uses DAP Secrets. For this lab, we will be logging into the Kubernetes API to get information about a Kubernetes deployment. The Kubernetes deployment has already been created for you, and the secret stored in the Dynamic Access Provider for you as well.

### Steps:

#### Configure the DAP secret in Jenkins

1. At the home page for Jenkins, click **Credentials, System**, then **Global Credentials**. You'll see the credentials that were configured in the previous Lab.
2. Click **Add Credentials** on the left side of the page, then for kind, select **Conjur Secret Credential**, then enter the path for the lab2 connection token, give it an ID of **lab2-token**, then click OK:

Scope	Global (Jenkins, nodes, items, all child items, etc)
Variable Path	lab2/lab2-connection-token
ID	lab2-token
Description	

#### Create the Pipeline

3. At the Jenkins homepage, click **New Item**, enter **LAB2\_K8sAPIAccess**, and select **Pipeline** then click **OK**
4. Scroll to the bottom and paste the pipeline information into the Pipeline Script Box. Review the Credentials section with the lab instructor, then click **Save**:

```
node{
  stage('Get Namespace Info'){
    withCredentials([conjurSecretCredential(credentialsId: 'lab2-token', variable: 'TOKEN')]){
      sh 'curl https://10.0.0.3:6443/apis/apps/v1/namespaces/lab2/deployments -k -H "Authorization: Bearer $TOKEN"'
    }
  }
}
```

- On the left side of the screen, click **Build Now**. Once the build completes, hover over the green box and click Logs to see the results of the pipeline:

Stage Logs (Get Namespace Info)

Shell Script -- curl https://10.0.0.3:6443/apis/apps/v1/namespaces/lab2/deployments -k -H "Authorization: Bearer \$TOKEN" (self time 302ms)

```
+ curl https://10.0.0.3:6443/apis/apps/v1/namespaces/lab2/deployments -k -H Authorization: Bearer
****
```

% Total	% Received	% Xferd	Average Speed	Time	Time	Time	Current
			Dload	Upload	Total	Spent	Left
0	0	0	0	0	0	0	0
100	3897	100	3897	0	0	82768	0

```

{
  "kind": "DeploymentList",
  "apiVersion": "apps/v1",
  "metadata": {
    "selfLink": "/apis/apps/v1/namespaces/lab2/deployments",
    "resourceVersion": "405147"
  },
  "items": [
    {
      "metadata": {
        "name": "lab2",
        "namespace": "lab2",

```

## Lab 3 – Integrate your app to use Native K8's secrets

### Purpose:

The purpose of this lab is to walk you through the deployment of a Dynamic Access Provider namespace deployment in Kubernetes, then we will integrate an existing application that uses Native Kubernetes Secrets with the Dynamic Access Provider.

### Steps:

#### Deploy the initial manifest

- Review the lab3.yml manifest with your instructor.
- At the command line, type **kubectl apply -f lab3/lab3.yml** to deploy the initial lab3 manifest. You should see output similar to:

```
cybradmin@admin-wkstn:~/dap-workshop$ kubectl apply -f lab3/lab3.yml
namespace/lab3 created
serviceaccount/lab3-svc created
secret/lab3-user-pass created
deployment.apps/simple-app created
configmap/k8s-app-ssl created
```

- Launch the Kubernetes Dashboard using the shortcut in Firefox, then click **skip**. In the namespace dropdown on the left, select the **lab3** namespace. Click the pod hyperlink pictured below:

Cluster

Cluster Roles

Namespaces

Nodes

Persistent Volumes

Storage Classes

Namespace

lab3

Overview

Workloads

Cron Jobs

Daemon Sets

Deployments

Jobs

Pods

Replica Sets

Workloads

Deployments

Name	Labels	Pods
simple-app	app: lab3	1 / 1

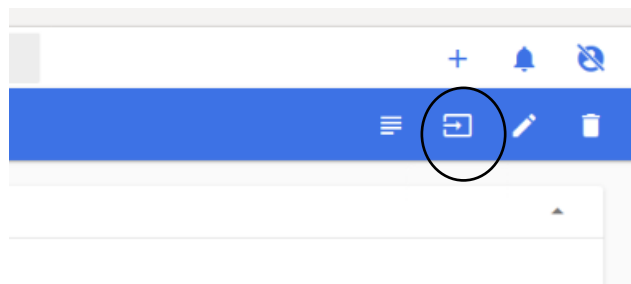
Pods

Name	Labels	Node	Status
simple-app-5dcf96779b-zd58h	app: lab3 pod-template-hash: 5dcf96779b	k8s-worker1	Running

Replica Sets

Name	Labels	Pods
------	--------	------

- Click the icon at the top right of the page to exec into the pod:



- In the shell, type **cat scripts/local-k8s-secret.sh** then press enter. The script is designed to read the mapped Kubernetes secrets discussed in previous steps:

```
[root@simple-app-b4c479c7c-c9ctt /]# cat scripts/local-k8s-secret.sh
#!/bin/bash
while ($true)
do
    echo "username: "$(cat /etc/secrets/username)
    echo "password: "$(cat /etc/secrets/password)
    echo "-----"
    sleep 3s
done[root@simple-app-b4c479c7c-c9ctt /]#
```

- In the shell, type **./scripts/local-k8s-secret.sh** then press enter. You'll notice the secrets have been decoded and you'll see the value of the secrets:

```
[root@simple-app-5dcf96779b-d82s8 /]# ./scripts/local-k8s-secret.sh
username: lab3user
password: Cyberark1
-----
username: lab3user
password: Cyberark1
-----
```

### Deploy the Kubernetes Authenticator to the Kubernetes Cluster

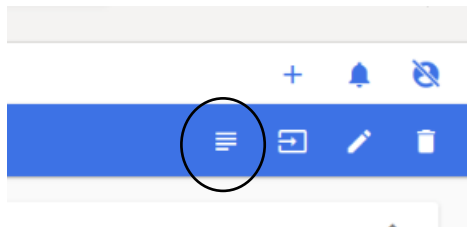
7. Review the **dapAccessManifest.yml** file and **config.sh** scripts with your instructor located in the **dap-workshop/lab3** folder
8. At the command line, type **kubectl apply -f lab3/dapAccessManifest.yml** to deploy the initial lab3 manifest. You should see output similar to:

```
cybradmin@admin-wkstn:~/dap-workshop$ kubectl apply -f lab3/dapAccessManifest.yml
namespace/dap unchanged
serviceaccount/conjur-cluster unchanged
clusterrole.rbac.authorization.k8s.io/conjur-authenticator unchanged
clusterrolebinding.rbac.authorization.k8s.io/conjur-authenticator unchanged
Warning: kubectl apply should be used on resource created by either kubectl create --save-config or kubectl apply
configmap/k8s-app-ssl configured
deployment.apps/dap-access-node unchanged
service/access unchanged
```

9. Type type **./lab3/config.sh** to populate the config map and necessary data to allow for follower auto-enrollment:

```
cybradmin@admin-wkstn:~/dap-workshop$ ./lab3/config.sh
configmap/k8s-app-ssl replaced
configmap/k8s-app-ssl replaced
configmap/k8s-app-ssl replaced
Value added
Value added
Value added
pod "dap-access-node-78b5849879-snjr4" deleted
```

10. In your web portal, select the **dap** namespace, then click the pod hyperlink, as previously done for the initial container deployment in this lab. Then click the logs button at the top right:



11. Review the logs with your instructor:

```
version='4', pub date='2018-09-25'
<46>1 2019-10-28T21:15:11.357+00:00 dap-access-node-78b5849879-mxscx syslog-ng 19 - [meta sequenceId="2"] Accepting connections; addr='AF_UNIX
<46>1 2019-10-28T21:15:11.357+00:00 dap-access-node-78b5849879-mxscx syslog-ng 19 - [meta sequenceId="3"] Accepting connections; addr='AF_INET
<45>1 2019-10-28T21:15:11.357+00:00 dap-access-node-78b5849879-mxscx syslog-ng 19 - [meta sequenceId="4"] Configuration reload request received
<45>1 2019-10-28T21:15:11.357+00:00 dap-access-node-78b5849879-mxscx syslog-ng 19 - [meta sequenceId="5"] Configuration reload finished;
<45>1 2019-10-28T21:15:11.360+00:00 dap-access-node-78b5849879-mxscx syslog-ng 19 - [meta sequenceId="6"] Syslog connection established; fd='3
local='AF_INET(0.0.0.0:0)'
Configuration successful. Conjur follower up and running.
<86>1 2019-10-28T21:17:01.000+00:00 dap-access-node-78b5849879-mxscx CRON 914 - [meta sequenceId="1"] pam_unix(cron:session): session opened for
<78>1 2019-10-28T21:17:01.000+00:00 dap-access-node-78b5849879-mxscx CRON 916 - [meta sequenceId="2"] (root) CMD ( cd / && run-parts --report
<86>1 2019-10-28T21:17:01.000+00:00 dap-access-node-78b5849879-mxscx CRON 914 - [meta sequenceId="3"] pam_unix(cron:session): session closed for
```

### Integrate the application to work with the Dynamic Access Provider

12. Review the **lab3-dap.yml** deployment manifest with your instructor
13. At the command line, type **kubectl apply -f lab3/lab3-dap.yml** to deploy the integrated application manifest.

```
cybradmin@admin-wkstn:~/dap-workshop$ kubectl apply -f lab3/lab3-dap.yml
namespace/lab3 unchanged
secret/lab3-user-pass configured
serviceaccount/lab3-svc unchanged
rolebinding.rbac.authorization.k8s.io/secrets-access-binding created
rolebinding.rbac.authorization.k8s.io/lab3-conjur-authenticator-role-binding-dap created
clusterrole.rbac.authorization.k8s.io/secrets-access unchanged
deployment.apps/simple-app configured
```

14. In your web portal, select the **lab3** namespace, then click the pod hyperlink, as previously done for the initial container deployment in this lab. Then click the logs button at the top right and select the **authenticator** pod. Review the logs with your instructor.

```
/acme/host%2Fconjur%2Fauthn-k8s%2Fk8s-follower%2Fapps%2Flab3%2F%2A%2F%2A/authenticate
INFO: 2019/10/31 02:08:39 authenticator.go:256: CAKC001I Successfully authenticated
INFO: 2019/10/31 02:08:39 k8s_secrets_client.go:53: CSPFK004I Creating Kubernetes client...
INFO: 2019/10/31 02:08:39 k8s_secrets_client.go:22: CSPFK005I Retrieving Kubernetes secret 'lab3-user-pass' from namespace 'lab3'
INFO: 2019/10/31 02:08:39 conjur_secrets_retriever.go:11: CSPFK003I Retrieving following secrets from Conjur: [lab3/username lab3/secret]
INFO: 2019/10/31 02:08:39 conjur_client.go:21: CSPFK002I Creating Conjur client...
INFO: 2019/10/31 02:08:39 k8s_secrets_client.go:53: CSPFK004I Creating Kubernetes client...
INFO: 2019/10/31 02:08:39 k8s_secrets_client.go:40: CSPFK006I Patching Kubernetes secret 'lab3-user-pass' in namespace 'lab3'
```

15. In the breadcrumb at the top, click the pod name, then exec into the app container as previously done in this lab. Type `./scripts/local-k8s-secret.sh` then press enter, and you will see that the password has changed in the output.

## Lab 4 – Deploy an app using continuous authentication

### Purpose:

This lab's purpose is to teach you how to integrate new applications with the CyberArk Dynamic Access Provider using the traditional CyberArk integration. We will leverage dual accounts from the Enterprise Password Vault to deliver secrets to a container that will change without requiring any changes or rescheduling of the containers. This lab leverages Dual Accounts from CyberArk EPV, which your instructor will explain to you. We will also use bulk-retrieval, showcasing the ability for multiple secret retrievals with CyberArk's Dynamic Access Provider with one API call.

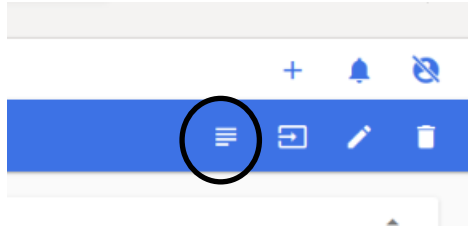
### Steps:

#### Deploy the Lab 4 manifest

1. Review the lab4.yml manifest with your instructor.
2. At the command line, type `kubectl apply -f lab4/lab4.yml` to deploy the lab4 manifest. You should see output similar to:

```
cybradmin@admin-wkstn:~/dap-workshop$ kubectl apply -f lab4/lab4.yml
namespace/lab4 created
serviceaccount/lab4-svc created
configmap/k8s-app-ssl created
deployment.apps/simple-app created
rolebinding.rbac.authorization.k8s.io/application-conjur-authenticator-role-binding-dap created
```

3. In your web portal, select the **lab4** namespace, then click the pod hyperlink, as previously done for the initial container deployment in this workshop. Then click the logs button at the top right:



4. Review the logs with your instructor:

```
INFO: 2019/10/31 02:34:47 authenticator.go:193: CAKC002I Logged in
INFO: 2019/10/31 02:34:47 authenticator.go:176: CAKC008I Cert expires: 2019-11-03 02:34:24 +0000 UTC
INFO: 2019/10/31 02:34:47 authenticator.go:177: CAKC009I Current date: 2019-10-31 02:34:47.416634032 +0000 UTC
INFO: 2019/10/31 02:34:47 authenticator.go:178: CAKC010I Buffer time: 30s
INFO: 2019/10/31 02:34:47 requests.go:46: CAKC012I Authn request to: https://access.dap.svc.cluster.local/api/authn-k8s/k8s-follower
/acme/host%2Fconjur%2Fauthn-k8s%2Fk8s-follower%2Fapps%2Fflab4%2F%2A%2F2A/authenticate
INFO: 2019/10/31 02:34:47 authenticator.go:256: CAKC001I Successfully authenticated
INFO: 2019/10/31 02:34:47 main.go:61: CAKC013I Waiting for 6m0s to re-authenticate
```

5. In the breadcrumb at the top, click the pod name, then exec into the app container as previously done in this lab. Type `cat scripts/dap-k8s-secret.sh` then press enter. Review the script with your instructor:

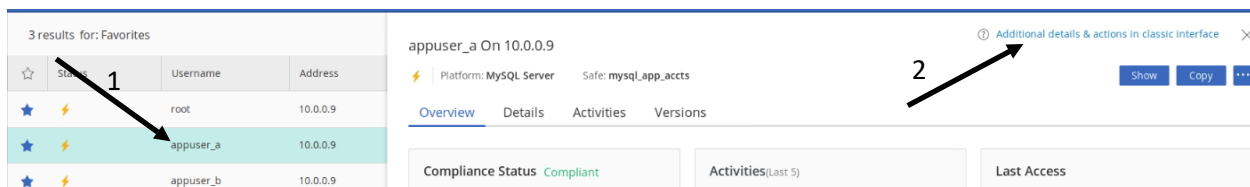
```
#!/bin/bash
URL_USERNAME=$(echo "$USERNAME_PATH" | sed 's/\./%2F/g' )
URL_PASSWORD=$(echo "$SECRET_PATH" | sed 's/\./%2F/g' )

while ($true)
do
    token=$(cat /run/conjur/access-token | base64 | tr -d '\r\n')
    output=$(curl -k -s -X GET -H "Authorization: Token token=\"$token\"" $CONJUR_APPLIANCE_URL/secrets?variable_ids=$CONJUR_ACCOUNT:variable:$URL_USERNAME,$CONJUR_ACCOUNT:variable:$URL_PASSWORD)
    echo -n "username = "
    echo $output | jq -r '["'$CONJUR_ACCOUNT':variable:'$USERNAME_PATH']'
    echo -n "password = "
    echo $output | jq -r '["'$CONJUR_ACCOUNT':variable:'$SECRET_PATH']'
    echo "-----"
    sleep 3s
done
```

6. Type `./scripts/dap-k8s-secret.sh` then press enter. Allow this script to continue scrolling:

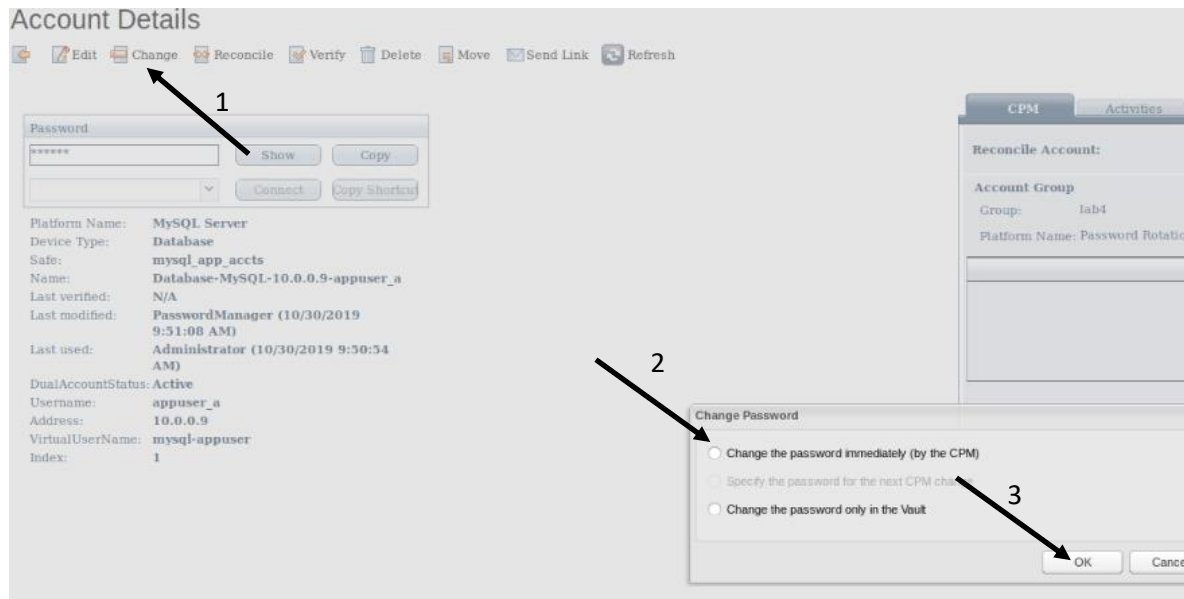
```
[root@simple-app-667bb46888-qzwnr /]# ./scripts/dap-k8s-secret.sh
username = appuser_a
password = XMRuTf2I
-----
username = appuser_a
password = XMRuTf2I
-----
```

7. In your browser, open a new tab and login to the Password Vault. Click anywhere on `appuser_a`'s line, then click **Additional details & actions in classic interface**:



8. Click **Change**, then select the top radio button, then click **OK**. The Message Box will increase in size once your click the top radio button. That is expected behavior:





9. Your instructor will explain the process of dual accounts to you. Once the change process is complete, you will see your output change to a different username and password:

```
username = appuser_a
password = XMRuTf2I
-----
username = appuser_a
password = XMRuTf2I
-----
username = appuser_b
password = PXRv36Yq
-----
username = appuser_b
password = PXRv36Yq
-----
username = appuser_b
password = PXRv36Yq
-----
username = appuser_b
```

## Lab 5 – Deploy the Secretless Broker for an application

### Purpose:

This lab's purpose is to teach you how to integrate applications with the CyberArk Dynamic Access Provider using the Secretless broker. CyberArk's Secretless broker completely isolates the application container from the secret. This allows developers to develop without ever knowing the application secrets or username. The broker will handle all of the authentication and secret injection for the application. See <https://secretless.io> for more detailed information on how the Secretless broker works, documentation, and FAQ.

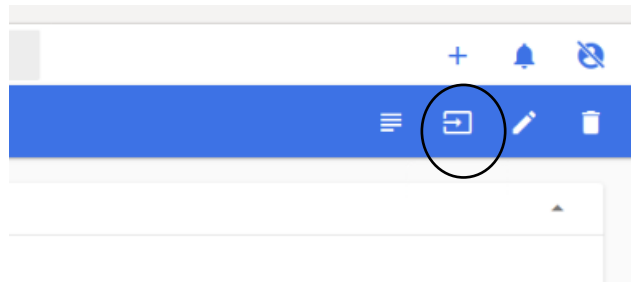
## Steps:

### Deploy the initial lab 5 manifest

1. Review the lab5.yml manifest with your instructor.
2. At the command line, type **kubectl apply -f lab5/lab5.yml** to deploy the initial lab5 manifest. You should see output similar to:

```
cybradmin@admin-wkstn:~/dap-workshop$ kubectl apply -f lab5/lab5.yml
namespace/lab5 unchanged
serviceaccount/lab5-svc unchanged
secret/lab5-user-pass unchanged
deployment.apps/simple-app created
configmap/k8s-app-ssl unchanged
```

3. In your web portal, select the **lab5** namespace, then click the pod hyperlink, as previously done for the initial container deployment in this workshop. Then click the icon at the top right of the pod to exec into the pod:



4. In the shell, type **cat scripts/mysql-performers.sh** then press enter. At this point, we are contacting the database directly using the hard-coded secrets in the manifest:

```
[root@simple-app-7fb4bd55d-zg4dz /]# cat scripts/mysql-performers.sh
#!/bin/bash
mysql --host=$LAB5_DB_HOST --user=$LAB5_USER --password=$LAB5_PASSWORD -e "select * from performers.comedians;"
```

5. In the shell, type **./scripts/mysql-performers.sh** then press enter. You'll see a list of comedians stored in the database:

```
[root@simple-app-7fb4bd55d-zg4dz /]# ./scripts/mysql-performers.sh
mysql: [Warning] Using a password on the command line interface can be insecure.
+-----+-----+
| first_name | last_name |
+-----+-----+
| Tiffany    | Haddish   |
| Dave       | Chappelle |
| Bert       | Kreisler  |
| Pete       | Davidson  |
| Tina       | Fey       |
| Wanda      | Sykes     |
+-----+-----+
```

### Change the password for the MySQL account used in the lab

6. In Firefox, open a new browser window, then click the **Password Vault** shortcut in the toolbar, then login with the credentials saved in the browser.



- Once logged in, find the `appuser_lab5` credential, and click anywhere on the line. Once the management pane appears, click **Change**:

The screenshot shows the Vault interface. On the left, a table lists 4 results for 'Favorites'. The table has columns: Status, Username, and Address. The rows are: root (10.0.0.9), appuser\_a (10.0.0.9), appuser\_b (10.0.0.9), and appuser\_lab5 (10.0.0.9). The appuser\_lab5 row is highlighted in light blue. A blue arrow points to this row. On the right, the management pane for 'appuser\_lab5 On 10.0.0.9' is shown. It includes a lightning bolt icon, 'Platform: MySQL Server', and 'Safe: mysql\_app\_accts'. Below this are tabs for Overview, Details, Activities, and Versions. The Overview tab is active, showing a 'Compliance Status' of 'Compliant' with a green circle and '0 Days ago'. It also shows 'Changed by PasswordManager Mar 23, 2020 3:27 PM' and buttons for 'Reconcile' and 'Change'. A blue arrow points to the 'Change' button.

Status	Username	Address
-	root	10.0.0.9
-	appuser_a	10.0.0.9
-	appuser_b	10.0.0.9
-	appuser_lab5	10.0.0.9

appuser\_lab5 On 10.0.0.9

Platform: MySQL Server Safe: mysql\_app\_accts

Overview Details Activities Versions

Compliance Status **Compliant**

0 Days ago

Changed by PasswordManager Mar 23, 2020 3:27 PM

Reconcile Change

- A window will pop up telling you the password will be changed to a random password. Click **Change**:

The screenshot shows a dialog box titled 'Change password for account appuser\_lab5 on undefined'. It contains the text 'The CPM will change the password to a new random password' and 'More options'. Below this is a link 'Change the password only in the Vault'. At the bottom are 'Cancel' and 'Change' buttons. A blue arrow points to the 'Change' button.

Change password for account **appuser\_lab5** on undefined

The CPM will change the password to a new random password

More options

[Change the password only in the Vault](#)

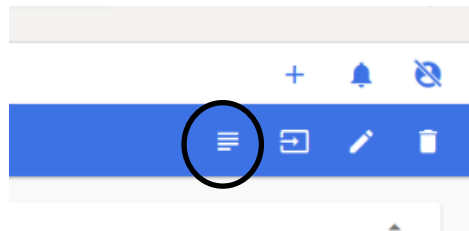
Cancel Change

*Integrate the application to work with the Secretless Broker*

- Review the `lab5-dap.yml` deployment manifest with your instructor
- At the command line, type `kubectl apply -f lab5/lab5-dap.yml` to deploy the integrated application manifest.

```
cybradmin@admin-wkstn:~/dap-workshop$ kubectl apply -f lab5/lab5-dap.yml
namespace/lab5 unchanged
serviceaccount/lab5-svc unchanged
rolebinding.rbac.authorization.k8s.io/application-conjur-authenticator-role-binding-dap unchanged
configmap/secretless-config unchanged
secret/lab5-user-pass unchanged
deployment.apps/simple-app configured
configmap/k8s-app-ssl unchanged
```

11. In your web portal, select the **lab5** namespace, then click the pod hyperlink, as previously done for the initial container deployment in this workshop. Then click the logs button at the top right:

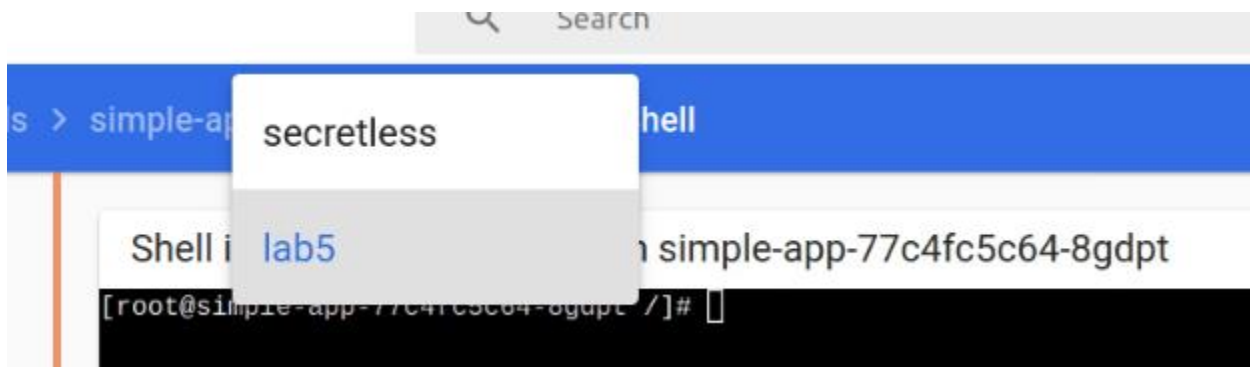


12. Review the logs with your instructor:

Logs from secretless ▾ in simple-app-77c4fc5c64-8gdpt ▾

```
2020/03/23 19:53:29 Secretless v1.5.2-b9bf4e4 starting up...
2020/03/23 19:53:29 Initializing health check on :5335...
2020/03/23 19:53:29 Initialization of health check done. You can access the endpoint at '/live' and '/ready'.
2020/03/23 19:53:29 [WARN] Plugin hashes were not provided - tampering will not be detectable!
2020/03/23 19:53:29 Trying to load configuration file: /etc/secretless/secretless.yml
2020/03/23 19:53:29 [INFO] Configuration found. Loading...
2020/03/23 19:53:29 [INFO] Validating config against available plugins: ssh,ssh-agent,mssql,pg,mysql,aws,basic_auth,conjur,generic_http
2020/03/23 19:53:29 Registering reload signal listeners...
2020/03/23 19:53:29 [INFO] Starting TCP listener on 0.0.0.0:3306...
2020/03/23 19:53:29 [INFO] mysql_tcp: Starting service
```

13. In the breadcrumb at the top, click the pod name, then exec into the app container as previously done in this lab. Once at the container CLI screen, change the shell dropdown and select **lab5**:



14. Type **cat scripts/mysql-performers.sh** then press enter. Notice the script hasn't changed, but we have triggered a password change from the CyberArk Password vault.
15. Type **./scripts/mysql-performers.sh** then press enter. Even though we have changed the password and did not update those values in the Kubernetes store, we are able to authenticate to the database and retrieve the same list of performers.