q

Cyberark dynamic access provider integration workshop

Table of Contents

[Introduction 2](#_Toc37076342)

[Lab 1 – Configure Existing Jenkins Freestyle Project to Use DAP Secrets 2](#_Toc37076343)

[Purpose: 2](#_Toc37076344)

[Steps: 2](#_Toc37076345)

[Lab 2 - Configure Jenkins Pipeline 8](#_Toc37076346)

[Purpose: 8](#_Toc37076347)

[Steps: 8](#_Toc37076348)

[Lab 3 – Integrate your app to use Native K8’s secrets 9](#_Toc37076349)

[Purpose: 9](#_Toc37076350)

[Steps: 9](#_Toc37076351)

[Lab 4 – Deploy an app using continuous authentication 12](#_Toc37076352)

[Purpose: 12](#_Toc37076353)

[Steps: 12](#_Toc37076354)

[Lab 5 – Deploy the Secretless Broker for an application 14](#_Toc37076355)

[Purpose: 14](#_Toc37076356)

[Steps: 15](#_Toc37076357)

# 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 | * https://K8s-master/{{NodePort}} * <https://k8s-worker1>{{NodePort}} * https://k8s-worker2{{NodePort}} | 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**
   1. Briefly review the Bindings and Build sections to see what the project does:

A screenshot of a cell phone

Description automatically generated

1. Click **Save,** then click **Build Now** to see the project work with the existing SSH Key.
2. 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:

A screenshot of a cell phone

Description automatically generated

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

1. 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.
2. 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.

A screenshot of a cell phone

Description automatically generated

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

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

A close up of a screen

Description automatically generated

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

1. 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:

A screenshot of a cell phone

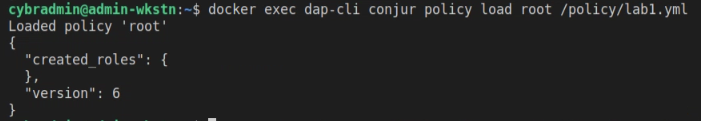
Description automatically generated

1. 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**

A screenshot of a cell phone

Description automatically generated

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



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

1. 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.
2. If not already in the LAB1\_JenkinsCredential management page, click **LAB1\_DBLogin**
3. Click the **red dot #2** on the left
4. 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.

A screenshot of a cell phone

Description automatically generated

#### Configure Jenkins to use secrets from DAP

1. 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.

A screenshot of a social media post

Description automatically generated

1. 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:

A screenshot of a cell phone

Description automatically generated

1. Do the same for the lab1 username:

A screenshot of a cell phone

Description automatically generated

1. Navigate to the Jenkins homepage, then **LAB1\_DBLogin**, **Configure**
2. 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**:

A screenshot of a cell phone

Description automatically generated

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

A screenshot of a cell phone

Description automatically generated

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

A screenshot of a cell phone

Description automatically generated

1. 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**:

A screenshot of a cell phone

Description automatically generated

# 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

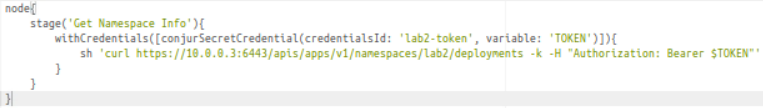
1. At the home page for Jenkins, click **Credentials, System,** then **Global Credentials**. You’ll see the credentials that were configured in the previous Lab.
2. Click **Add Credentials** on the left side of the page, then for kind, select **Conjur Secret Credential**, then enter the path for the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

A screenshot of a cell phone

Description automatically generated

#### Create the Pipeline

1. At the Jenkins homepage, click **New Item**, enter **LAB2\_K8sAPIAccess**, and select **Pipeline** then click **OK**
2. 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**:



1. 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:

A screenshot of a social media post

Description automatically generated

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

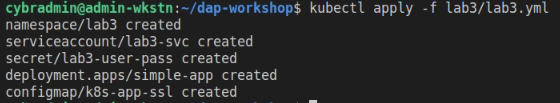
## Purpose:

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

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



1. 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:

A screenshot of a cell phone

Description automatically generated

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

A screenshot of a cell phone

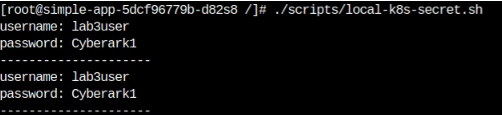
Description automatically generated

1. 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:

A screen shot of a person

Description automatically generated

1. 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:



#### Deploy the Kubernetes Authenticator to the Kubernetes Cluster

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

A screenshot of a cell phone

Description automatically generated

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

A picture containing text

Description automatically generated

1. 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:

A screenshot of a cell phone

Description automatically generated

1. Review the logs with your instructor:

A screen shot of a computer keyboard

Description automatically generated

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

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

A screenshot of a cell phone

Description automatically generated

1. 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.

A picture containing text, bottle

Description automatically generated

1. 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

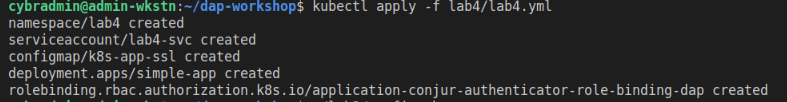
## Purpose:

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:



1. 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:

A screenshot of a cell phone

Description automatically generated

1. Review the logs with your instructor:

A close up of a keyboard

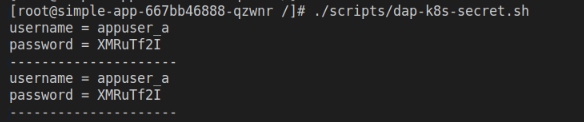
Description automatically generated

1. 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:

A screenshot of a cell phone

Description automatically generated

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



1. 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**:

A screenshot of a cell phone

Description automatically generated

1

2

1. 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:

A screenshot of a social media post

Description automatically generated

1

2

3

1. 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:

A close up of text on a black background

Description automatically generated

# Lab 5 – Deploy the Secretless Broker for an application

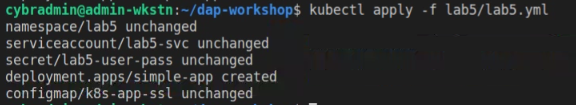
## Purpose:

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:



1. 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 page to exec into the pod:

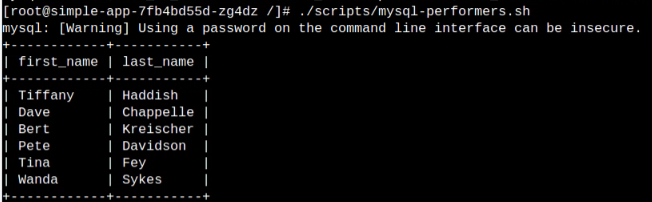
A screenshot of a cell phone

Description automatically generated

1. 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:

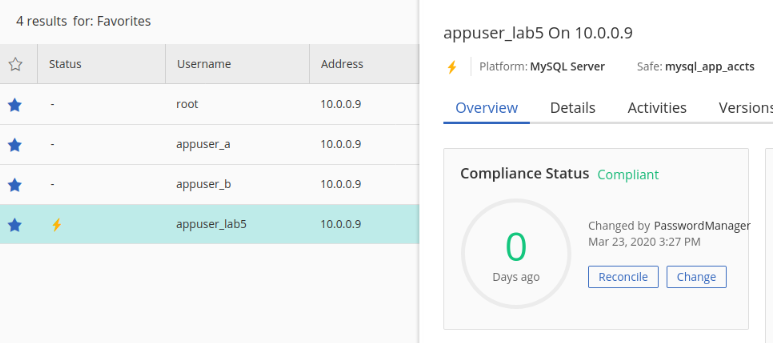


1. In the shell, type **./scripts/ysql-performers.sh** then press enter. You’ll see a list of comedians stored in the database:

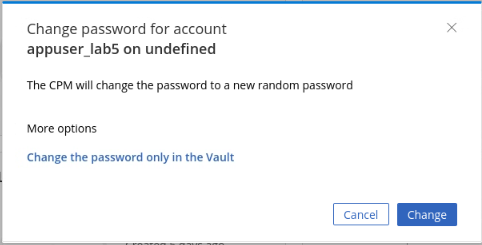


#### Change the password for the MySQL account used in the lab

1. 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.
2. Once logged in, find the appuser\_lab5 credential, and click anywhere on the line. Once the management pane appears, click **Change**:

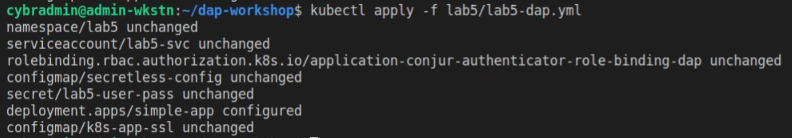


1. A window will pop up telling you the password will be changed to a random password. Click **Change**:



#### Integrate the application to work with the Secretless Broker

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

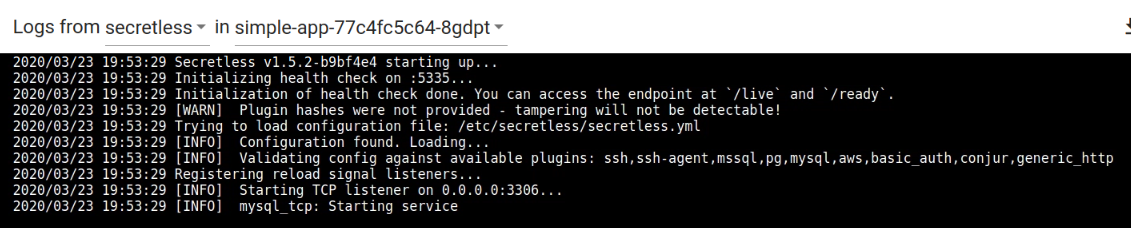


1. In your web portal, select the **lab5** namespace, then click the pod hyperlink, as previously done for the initial container deployment in this workshop. Then click the logs button at the top right:

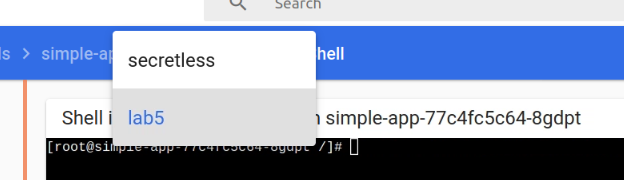
A screenshot of a cell phone

Description automatically generated

1. Review the logs with your instructor:



1. 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**:



1. 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.
2. 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.