

Lab: Securing Secrets for a VM Deployment

# Training Objective

Learn how to encrypt secrets in server conﬁgurations and synapse conﬁgurations for a VM deployment.

# High Level Steps

* Create your integration artifacts.
* Conﬁgure the server conﬁgurations and synapse conﬁgurations that should be encrypted.
* Test the scenario.

# Detailed Instructions

## **Step 1: Defining secrets**

### **Static secrets**

Static secrets are sensitive data that are specified directly in configurations. The secret can be a plain-text value or the alias of an encrypted value.

You must first list the plain-text secrets in the deployment.toml file under the secrets header. Note that we have specified an **alias** for the secret followed by the actual plain-text secret.

[secrets]

server\_secret = "[secret\_1]"

synapse\_secret = "[secret\_2]"

### **Dynamic secrets**

Dynamic secrets are specified in configurations as environment variables, system properties, Docker secrets, or Kubernetes secrets. The actual secrets is then encrypted using the WSO2 API Controller, **apictl** and injected to the environment.

1. First, list the dynamic secrets in the deployment.toml file under the [secrets] section. However, unlike for static secrets, specify the secret value as an environment variable or system property.

**Note**

In this example, dynamic\_secret is a placeholder for the secret. You will use this placeholder as the secret's alias when you encrypt the plain-text secret using the apictl (in the next step).

Environment Variable

[secrets]

server\_secret = "$env{dynamic\_secret}"

System Property

[secrets]

server\_secret = "$sys{dynamic\_secret}"

1. Now, encrypt a plain-text secret for the dynamic\_secret alias by using the WSO2 API Controller.

# Encrypting Secrets with apictl

**WSO2 API Controller (apictl)** allows you to encrypt a plain-text secret. You can use this feature to export secrets as environment variables, system properties, Docker secrets, or Kubernetes secrets.

## **Initialize apictl with a key store**

**Note**

Secret encryption supports only JKS Key Stores.

**Note**

Key Store used in this step needs to be the same Key Store which is used by the WSO2 API Manager (WSO2 API-M) or WSO2 Micro Integrator (WSO2 MI) to decrypt secrets.

Run the following command to initialize the apictl with the Key Store used to encrypt the secrets. It will prompt you to input the following,

|  |  |
| --- | --- |
| Key Store location | Path to the Key Store used by the WSO2 API-M or WSO2 MI to decrypt secrets |
| Key Store password | The password of the Key Store |
| Key alias | The alias of the key used to encrypt the secrets |
| Key password | The password of the key used to encrypt the secrets |

* **Command**
* **apictl** secret init

**Example**

apictl secret init

Enter Key Store location: /home/wso2mi-4.2.0/repository/resources/security/wso2carbon.jks

Enter Key Store password:

Enter Key alias: wso2carbon

Enter Key password:

* **Response**
* **Key** Store initialization completed

## **Encrypt secrets**

**Note**

Secret encryption supports only **RSA/ECB/OAEPWithSHA1AndMGF1Padding** (default) or **RSA/ECB/PKCS1Padding** as encryption algorithm.

**Note**

Encrypting algorithm used in this step needs to be the same algorithm used by the WSO2 API-M or WSO2 MI to decrypt secrets.

Run the following command to encrypt secrets with the apictl,

* **Command**
* **apictl** secret create

**Info**

**Flags:**

* + Optional :  
    --cipher or -c : Encryption algorithm (default is RSA/ECB/OAEPWithSHA1AndMGF1Padding)  
    --output or -o : Get the output in yaml (k8) or properties (file) format. By default the output is printed to the console  
    --from-file or -f : Path to the properties file which contains secrets to be encrypted
  + Encrypt a secret and get output on console

**Example**

apictl secret **create**

Enter plain **alias** **for** secret:db\_password

Enter plain text secret:

**Repeat** plain text secret:

* + Response
  + **db\_password** : eKALmLVA+HFVl7vqLUUhm6o0Vwsap+L6czwyEKWKomX+AcRmOCAHmiujPXPAZUboWJlZi4k0CwZYAvwD4BflbU8j5CCrtESzOlOrkJaJPormf/ViixRbftae2RqaDozPSEp9zSnfDKlKDXRq==
  + Encrypt secrets defined in a properties file

**Example**

apictl secret **create** -f ./**keys**/secrets.properties

* + Response
  + db\_password : JVlyw8j9TQqoPFTQUnKxNoGJn9p4+gGCHkkyHt2jXGVZoe60xndi2GjBJ1roR6667dlynhABXbcv434DFjz3ZI0iRjg1QhJLoXNtttSFl7KtyNDk5VtRMPDqAckheJAJe02KjWgdZXszEzjtBd6o2mY1nipsWBat3cOq0kt==
  + admin\_password : gPImOAX1zwHu3malMHm0+Zy5WEcfKpUSmxJ2ZXfI3bi1yIZbHjrHUxiY+MKurTWRN8GJ6+EVL==
  + Encrypt secrets defined in a properties file and get a .yaml file

**Example**

apictl secret **create** -o k8 -f ./**keys**/secrets.properties

* + Response
  + Kubernetes secret file created in apictl/security/wso2-secrets.yaml **with** **default** **name** **and** namespace

You can **change** the **default** **values** **as** **required** **before** applying.

## **Step 2: Running the Cipher Tool**

Running the Cipher Tool will first encrypt any [static secrets](https://apim.docs.wso2.com/en/4.2.0/install-and-setup/setup/mi-setup/security/encrypting_plain_text/#static-secrets) defined in the [secrets] section, and then enable all the secrets (static as well as dynamic) in the environment.

### **In a VM environment**

In a **VM environment**, you need to manually run the Cipher Tool as follows:

**Tip**

If you are using **Windows**, you need to have [Ant](http://ant.apache.org/) installed before using the Cipher Tool.

1. Open a terminal, navigate to the <MI\_HOME>/bin/ directory.
2. Execute the -Dconfigure command with the cipher tool script as shown below.

On Linux

./ciphertool.sh -Dconfigure

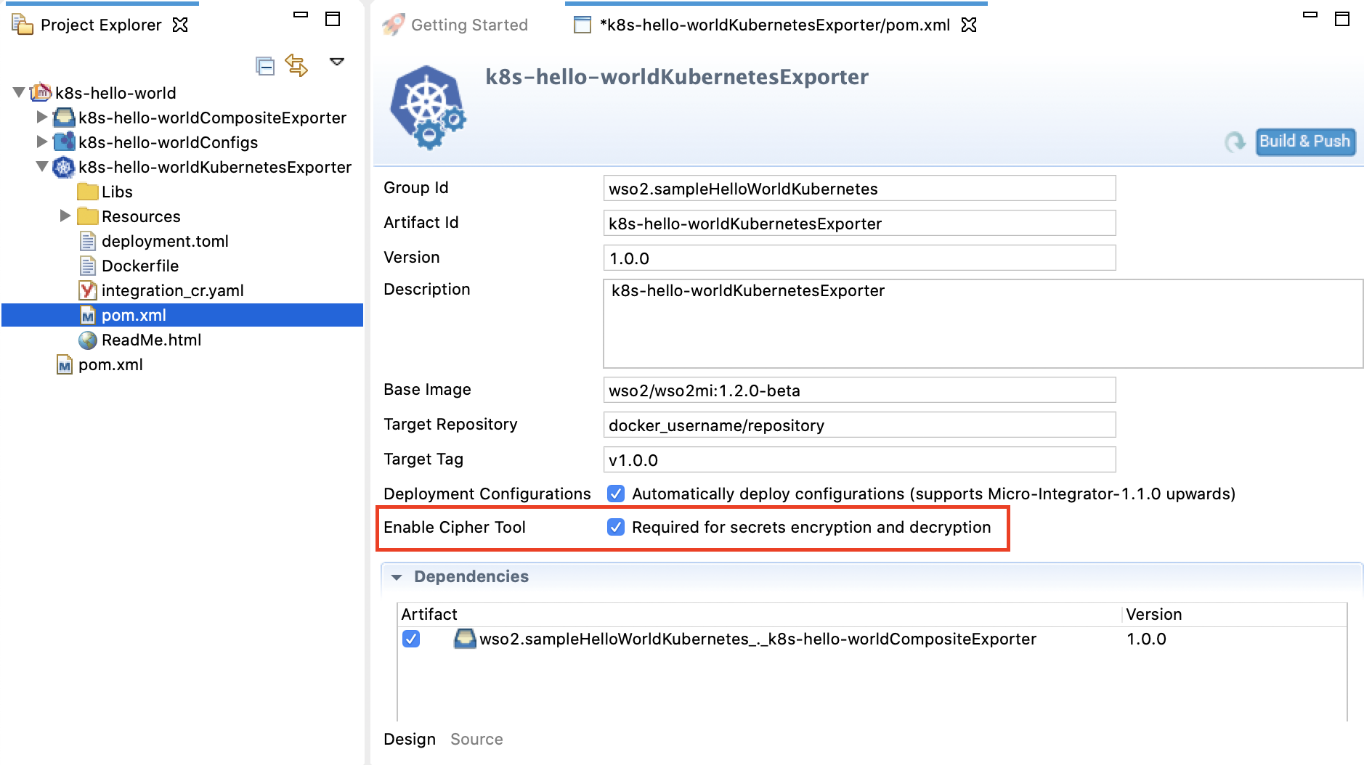
On Windows

./ciphertool.bat -Dconfigure

### **In a Kubernetes environment**

In a **Kubernetes environment**, you don't need to manually run the Cipher tool. Follow the steps given below.

1. Open your Integration Project in WSO2 Integration Studio, which contains all the integration artifacts and the Kubernetes Exporter.
2. Open the pom.xml of the Kubernetes Exporter module and select the **Enable Cipher Tool** check box as show below.



1. When you build the Docker image from your Kubernetes exporter, the secrets will get encrypted and enabled in the environment.

If you specified any [static secrets](https://apim.docs.wso2.com/en/4.2.0/install-and-setup/setup/mi-setup/security/encrypting_plain_text/#static-secrets), go back to the deployment.toml file and see that the secrets are encrypted.

[secrets]

keystore\_password = "encrypted\_pass\_3"

key\_password = "encrypted\_pass\_4"

truststore\_password = "encrypted\_pass\_5"

## **Step 3: Accessing secrets**

### **In server configurations**

You can refer an encrypted secret in your server configurations by using the $secret{alias} function in place of the plain-text secret as shown below. Replace alias with the secret's alias that is defined under the [secrets] section.

**Note**

You can use encrypted [static secrets](https://apim.docs.wso2.com/en/4.2.0/install-and-setup/setup/mi-setup/security/encrypting_plain_text/#static-secrets) as well as [dynamic secrets](https://apim.docs.wso2.com/en/4.2.0/install-and-setup/setup/mi-setup/security/encrypting_plain_text/#dynamic-secrets).

[keystore.primary]

password = "$secret{server\_secret}"

alias = "$secret{server\_secret}"

key\_password = "$secret{server\_secret}"

[truststore]

password = "$secret{server\_secret}"

### **In synapse configurations**

You can refer an encrypted secret in your synapse configurations by using the **vault lookup** function ({wso2:vault-lookup('alias')) in place of the plain-text secret as shown below. Replace alias with the secret's alias that is defined under the [secrets] section.

**Note**

You can only use encrypted [static secrets](https://apim.docs.wso2.com/en/4.2.0/install-and-setup/setup/mi-setup/security/encrypting_plain_text/#static-secrets) here.

<**log** level="custom">

<**property** expression="wso2:vault-lookup('synapse\_secret')" name="secret"/>

</**log**>

## **Step 4: Populating dynamic secrets**

If you have defined [dynamic secrets in your configurations](https://apim.docs.wso2.com/en/4.2.0/install-and-setup/setup/mi-setup/security/encrypting_plain_text/#dynamic-secrets), you must populate the secret into the relevant environment as required.

### **In a VM environment**

For an instance, in the case of environment variables, you can populate them with the export command as follows:

**export** env\_carbon\_sec=<**ENCRYPTED\_VALUE**>

### **In a Kubernetes environment**

If you are in a Kubernetes enviroment, you should have generated a .yaml file with the encrypted secrets using the WSO2 API Controller, **apictl**.

### **Start server as a background job**

If you start the Micro Integrator as a background job, you will not be able to provide password values on the command line. Therefore, you must start the server in "daemon" mode as explained below.

1. Create a new file in the MI\_HOME directory. The file should be named according to your OS as explained below.
   * For Linux: The file name should be password-tmp.
   * For Windows: The file name should be password-tmp.txt.

**Note**

When you start the server, the keystore password will be picked from this new file. Note that this file is automatically deleted from the file system after the server starts. Therefore, the admin has to create a new text file every time the server starts.

Alternatively, if you want to retain the password file after the server starts, the file should be named as follows:

* + For Linux: The file name should be password-persist
  + For Windows: The file name should be password-persist.txt

1. Add the primary keystore password (which is "wso2carbon" by default) to the new file and save. By default, the password provider assumes that both private key and keystore passwords are the same. If not, the private key password must be entered in the second line of the file.
2. Now, start the server as a background process by running the following command.

./micro-integrator.sh **start**

 [CC BY 4.0](http://creativecommons.org/licenses/by/4.0/)