WSO2 API Manager 4.2.0 Advanced - Integration Profile

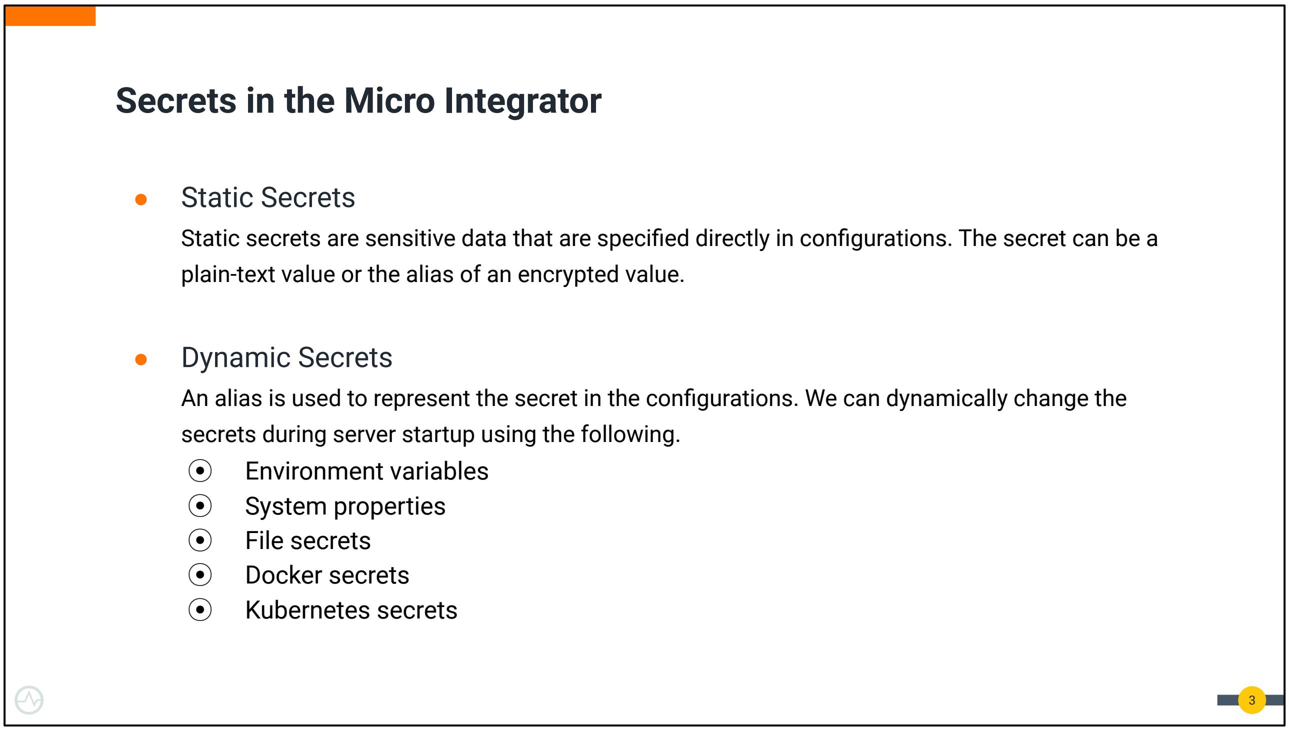
Applying Security

WSO2 Training

Module Objective

At the end of this module, attendees will be able to:

- Understand the different types of secrets that can be used in the Micro Integrator.
- Understand how to use encrypted secrets in different environments.
- Understand how secure vault implementation is used in the Micro Integrator for securing secrets.
- Understand how WS Security policies and various security handlers are used to apply security.



Secrets in the Micro Integrator

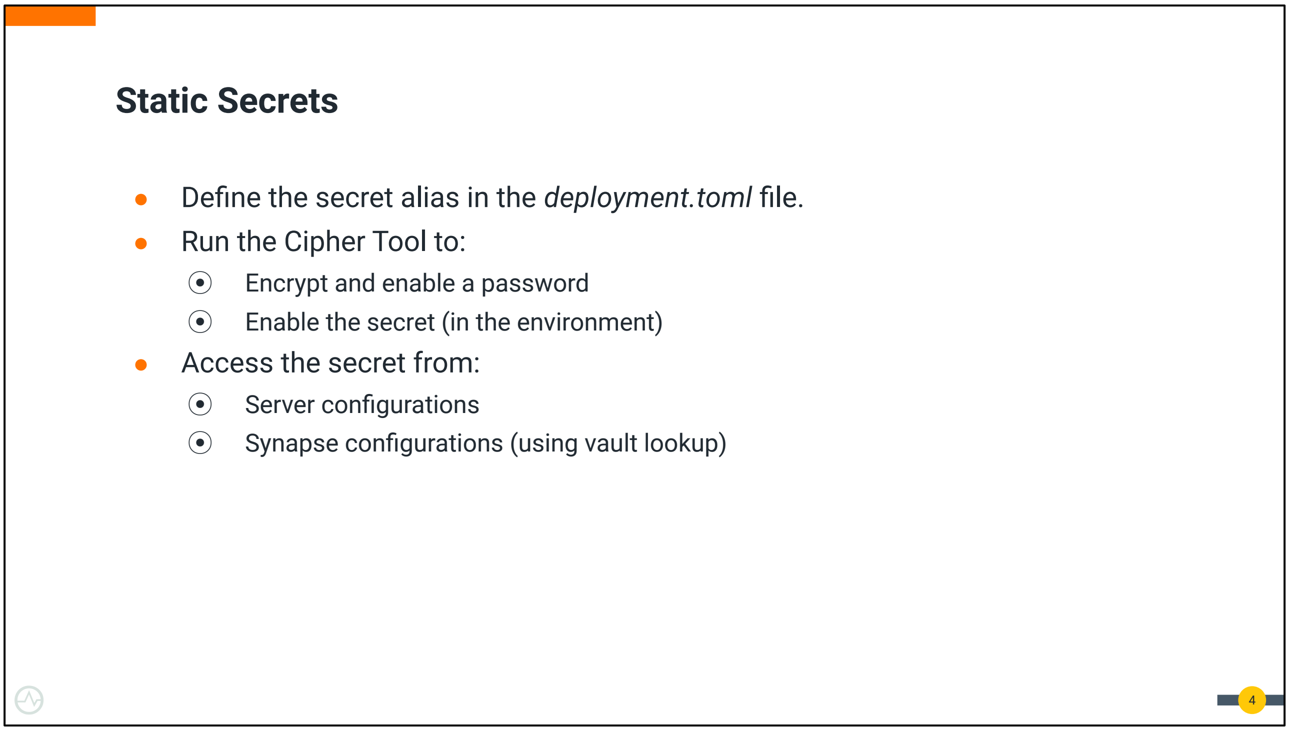
- Static Secrets

  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.

- Dynamic Secrets

  An alias is used to represent the secret in the configurations. We can dynamically change the secrets during server startup using the following.
  - Environment variables
  - System properties
  - File secrets
  - Docker secrets
  - Kubernetes secrets

https://apim.docs.wso2.com/en/4.2.0/install-and-setup/setup/mi-setup/security/encrypting_plain_text/



Static Secrets

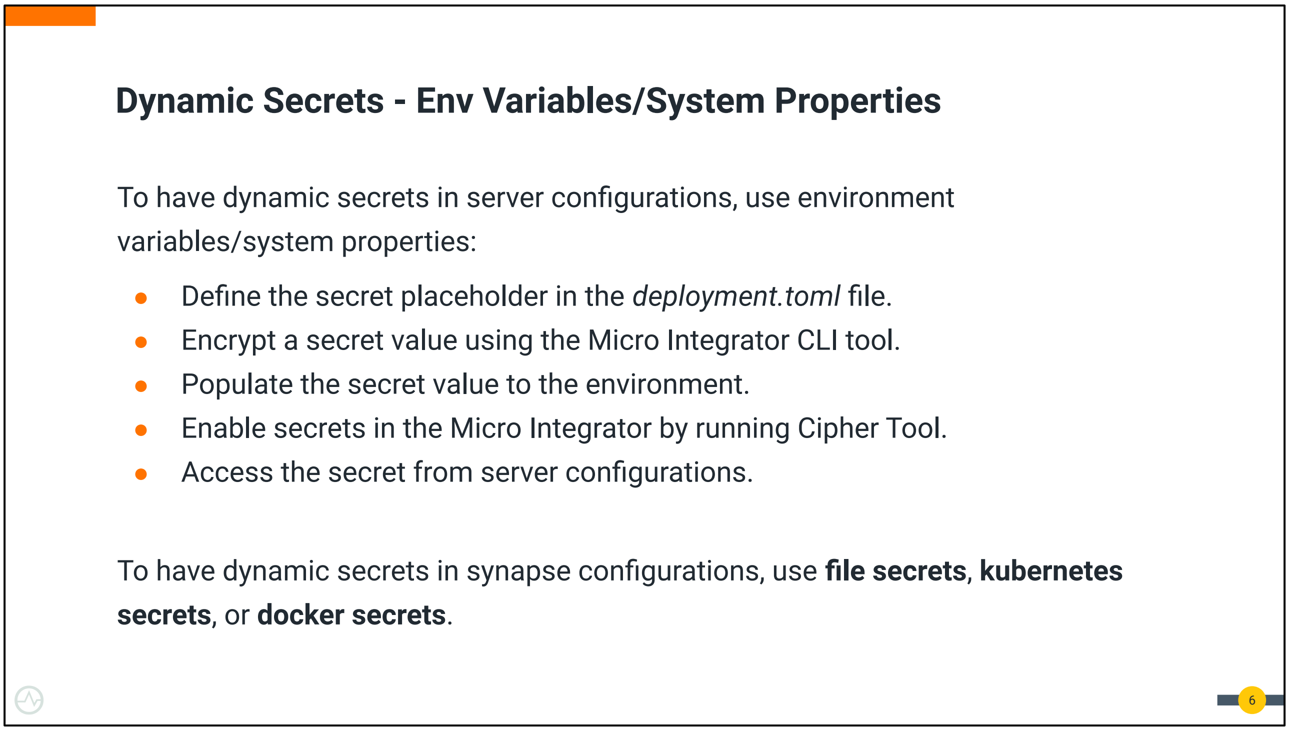
- Define the secret alias in the *deployment.toml* file.
- Run the Cipher Tool to:
  - Encrypt and enable a password
  - Enable the secret (in the environment)
- Access the secret from:
  - Server configurations
  - Synapse configurations (using vault lookup)

https://apim.docs.wso2.com/en/4.2.0/install-and-setup/setup/mi-setup/security/encrypting_plain_text/

Dynamic Secrets

The following table indicates whether the dynamic secrets can be referred to from server configurations, synapse configurations, etc.

	Server Configurations	Synapse Configurations	VM Environment	Containerized Environment
Environment Variable	✔	✔	✔	✔
System Properties	✔	✔	✔	✔
File Secrets	-	✔	✔	✔
Docker Secrets	-	✔	-	✔
Kubernetes Secrets	-	✔	-	✔



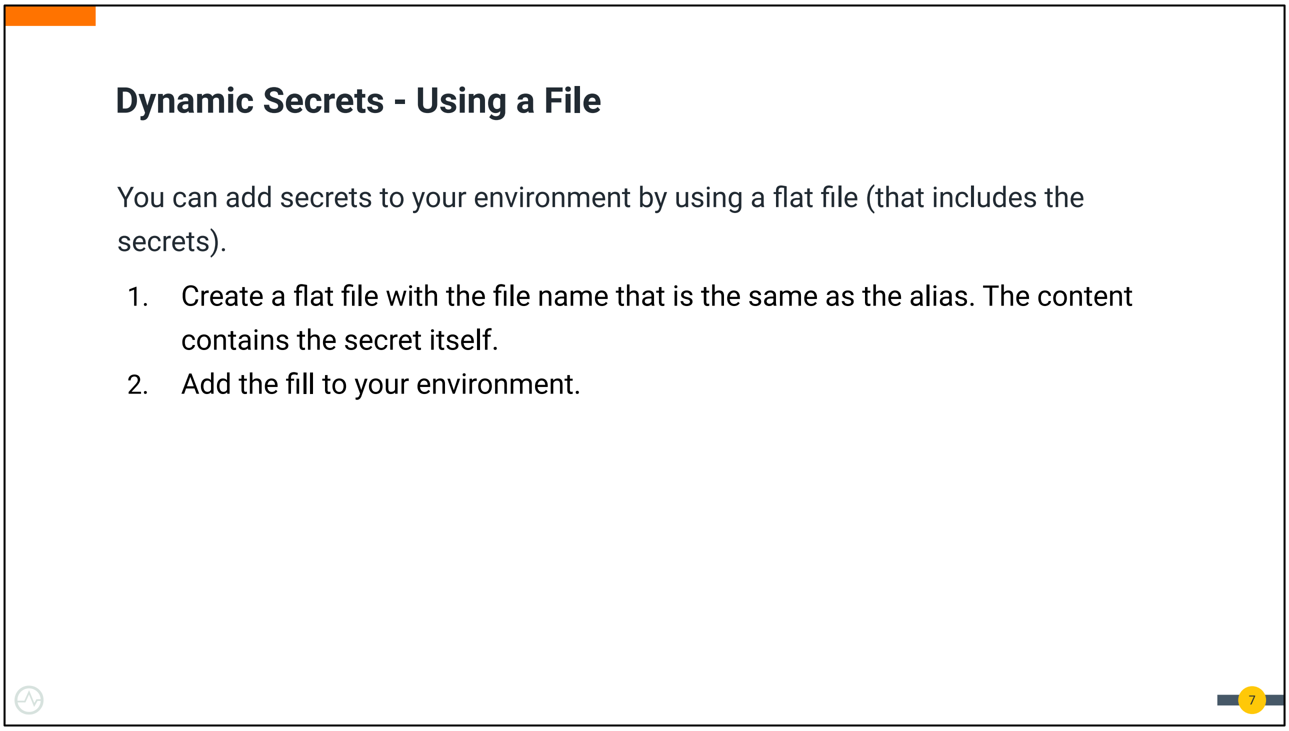
Dynamic Secrets - Env Variables/System Properties

To have dynamic secrets in server configurations, use environment variables/system properties:

- Define the secret placeholder in the *deployment.toml* file.
- Encrypt a secret value using the Micro Integrator CLI tool.
- Populate the secret value to the environment.
- Enable secrets in the Micro Integrator by running Cipher Tool.
- Access the secret from server configurations.

To have dynamic secrets in synapse configurations, use **file secrets**, **kubernetes secrets**, or **docker secrets**.

https://apim.docs.wso2.com/en/4.2.0/install-and-setup/setup/mi-setup/security/encrypting_plain_text/

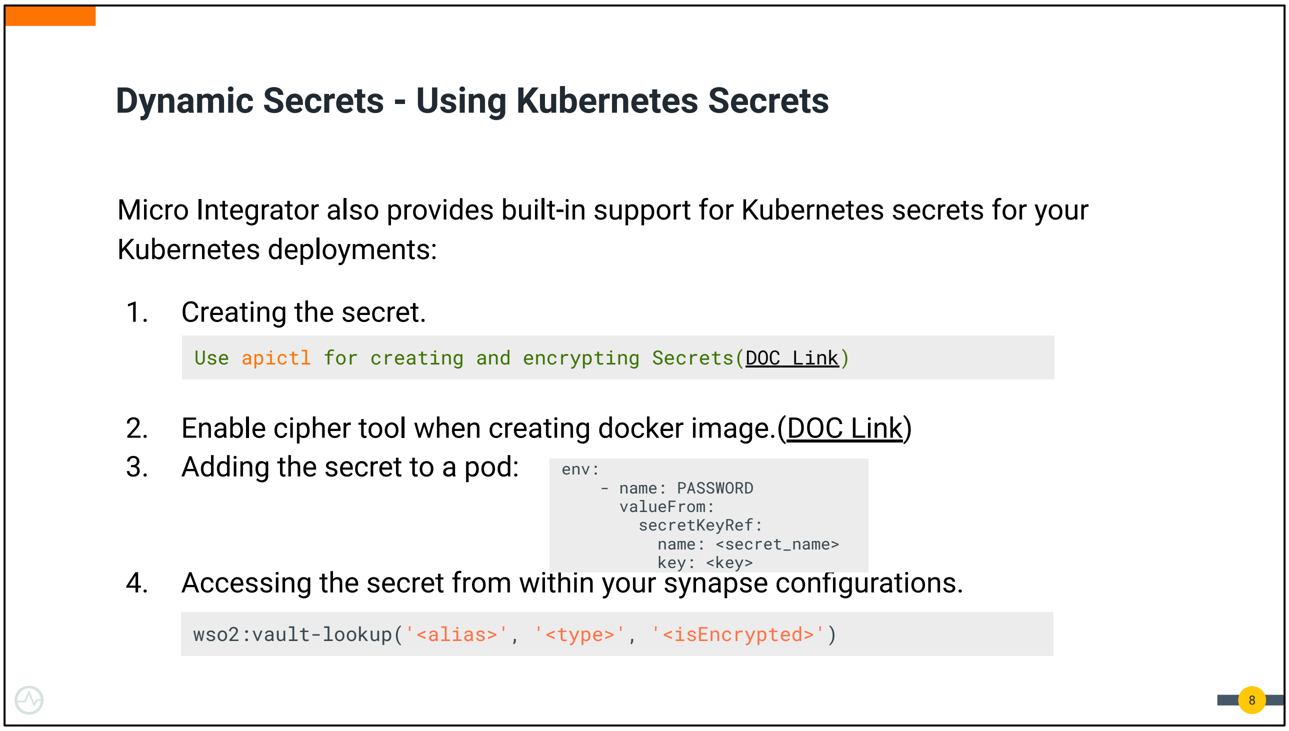


Dynamic Secrets - Using a File

You can add secrets to your environment by using a flat file (that includes the secrets).

1. Create a flat file with the file name that is the same as the alias. The content contains the secret itself.
2. Add the fill to your environment.

https://apim.docs.wso2.com/en/4.2.0/integrate/develop/creating-artifacts/using_docker_secrets/



Dynamic Secrets - Using Kubernetes Secrets

Micro Integrator also provides built-in support for Kubernetes secrets for your Kubernetes deployments:

1. Creating the secret.

```
Use apictl for creating and encrypting Secrets(DOC Link)
```

2. Enable cipher tool when creating docker image.(DOC Link)
3. Adding the secret to a pod:

```
env:
    - name: PASSWORD
      valueFrom:
        secretKeyRef:
          name: <secret_name>
          key: <key>
```

4. Accessing the secret from within your synapse configurations.

```
wso2:vault-lookup('<alias>', '<type>', '<isEncrypted>')
```

https://apim.docs.wso2.com/en/4.2.0/integrate/develop/creating-artifacts/using_k8s_secrets/

Dynamic Secrets - Using Docker Secrets

Micro Integrator also provides built-in support for Docker secrets for your Docker deployments.

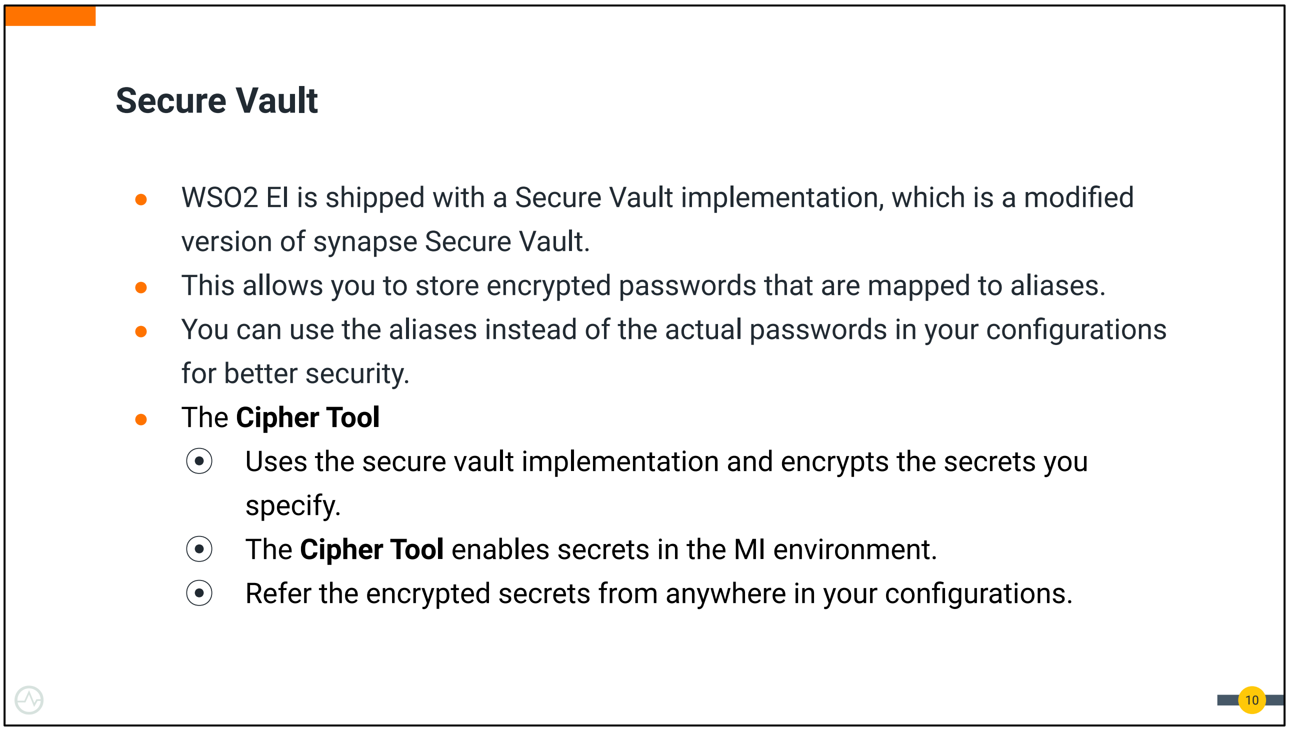
1. Adding the secret to your Docker environment.

```
echo "dockersecret123456" | docker secret create testdockersecret -
```

2. Accessing the secret from within your synapse configurations.

```
wso2:vault-lookup('<alias>', '<type>', '<isEncrypted>')
```

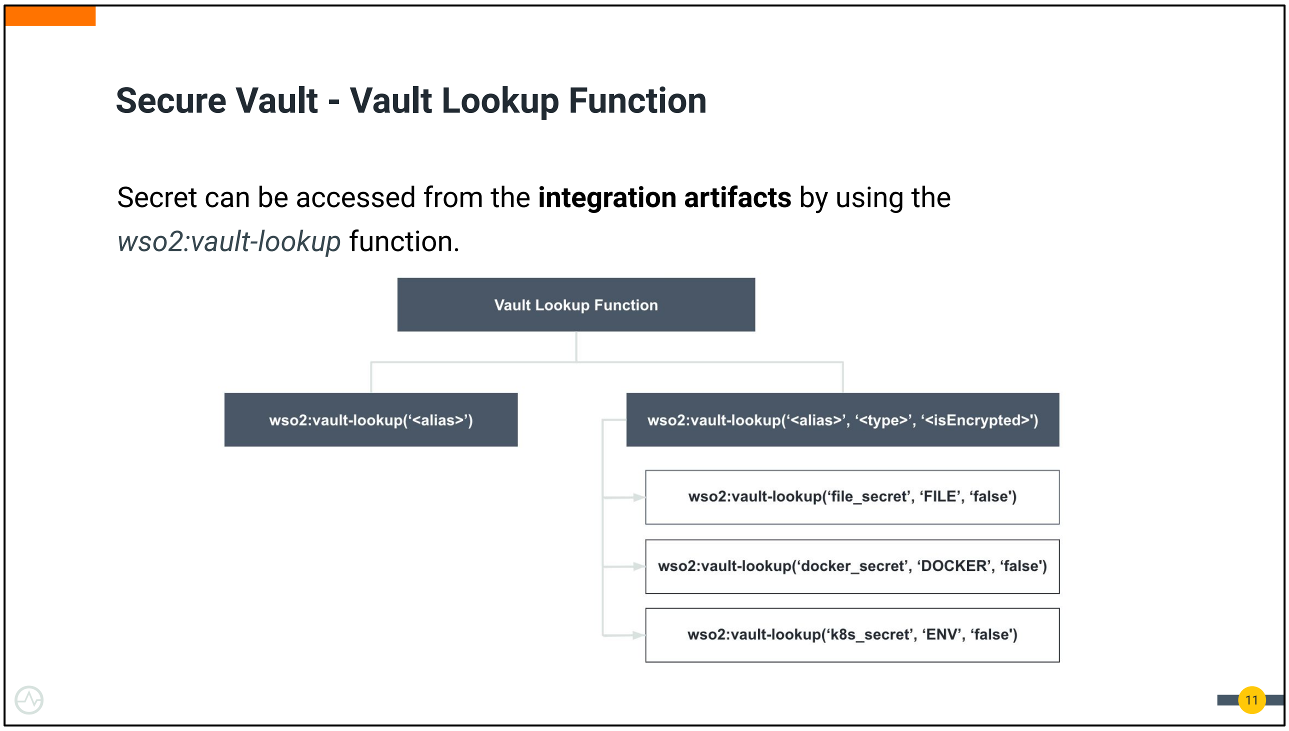
https://apim.docs.wso2.com/en/4.2.0/integrate/develop/creating-artifacts/using_docker_secrets/



Secure Vault

- WSO2 EI is shipped with a Secure Vault implementation, which is a modified version of synapse Secure Vault.
- This allows you to store encrypted passwords that are mapped to aliases.
- You can use the aliases instead of the actual passwords in your configurations for better security.
- The **Cipher Tool**
  - Uses the secure vault implementation and encrypts the secrets you specify.
  - The **Cipher Tool** enables secrets in the MI environment.
  - Refer the encrypted secrets from anywhere in your configurations.

- https://apim.docs.wso2.com/en/4.2.0/install-and-setup/setup/mi-setup/security/encrypting_plain_text/
- https://apim.docs.wso2.com/en/4.2.0/reference/mi-security-reference/customizing-secure-vault/



Secure Vault - Vault Lookup Function

Secret can be accessed from the **integration artifacts** by using the *wso2:vault-lookup* function.

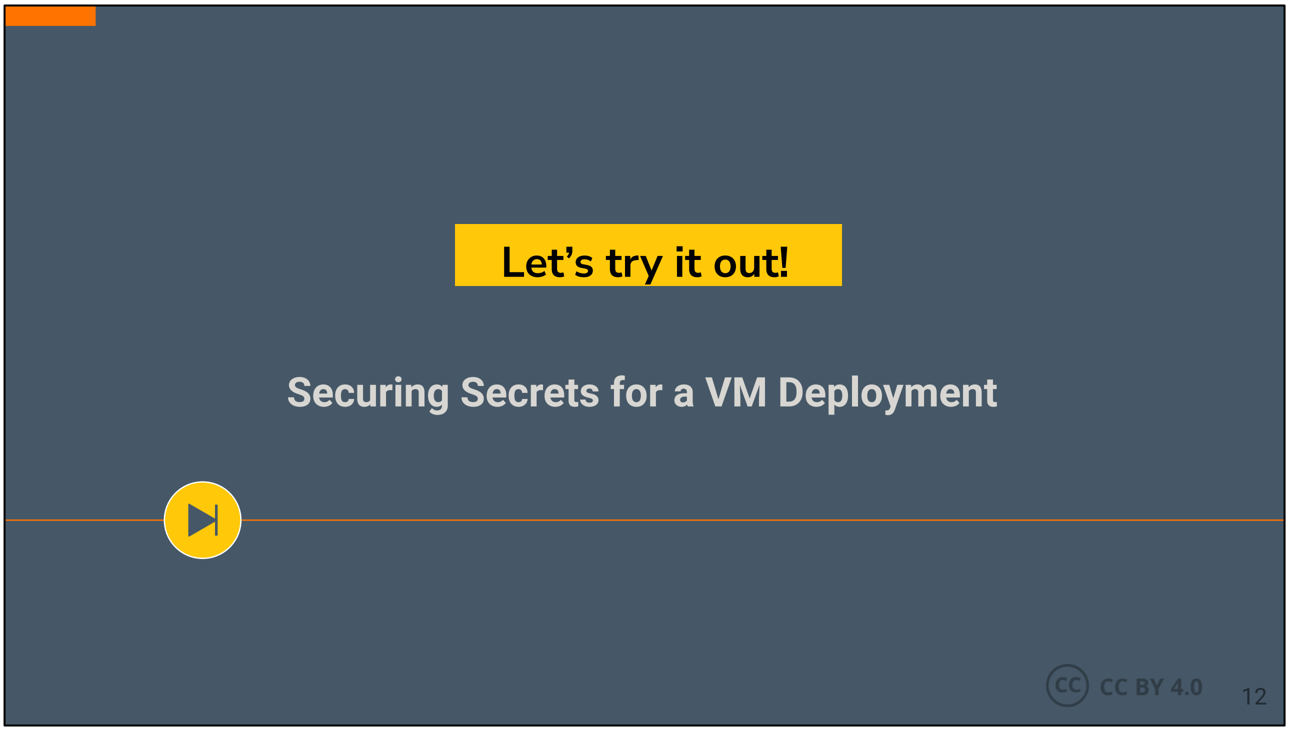
Vault Lookup Function
wso2:vault-lookup('<alias>')
wso2:vault-lookup('<alias>', '<type>', '<isEncrypted>')
wso2:vault-lookup('file_secret', 'FILE', 'false')
wso2:vault-lookup('docker_secret', 'DOCKER', 'false')
wso2:vault-lookup('k8s_secret', 'ENV', 'false')

In a **VM environment**, specify the values for the alias:

- `wso2:vault-lookup('synapse_secret')`

In a **container environment**, specify values for the following three parameters:

- `<alias>`: Name of the secret.
- `<type>`: Set this to `DOCKER`
- `<isEncrypted>`: Set this to `true` or `false` to specify whether the secret is encrypted.



Let’s try it out!

Securing Secrets for a VM Deployment

Let’s try it out!

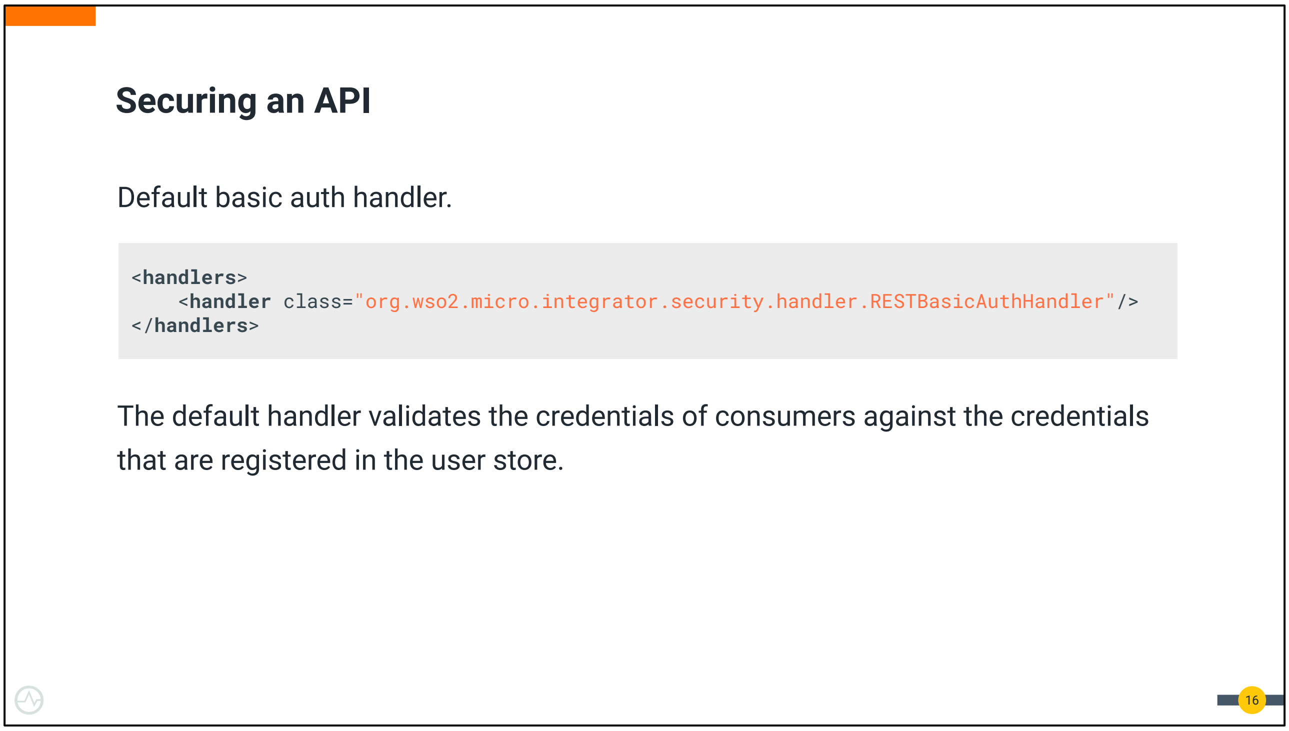
Securing Secrets for a Container Deployment

WS Security

- WSO2 Micro Integrator supports WS-Security, WS-Policy, and WS-Security Policy specifications.
- Possibility to define custom security policies.
- Commonly-used security scenarios.
- Apply service-specific security.

Let’s try it out!

Applying Security to a Proxy Service



Securing an API

Default basic auth handler.

```
<handlers>
    <handler class="org.wso2.micro.integrator.security.handler.RESTBasicAuthHandler"/>
</handlers>
```

The default handler validates the credentials of consumers against the credentials that are registered in the user store.

https://apim.docs.wso2.com/en/4.2.0/integrate/develop/advanced-development/applying-security-to-an-api/#using-a-basic-auth-handler

Securing an API

Custom Basic Auth handlers and other security implementations:

- OAuth
- Transforming Basic Auth to WS-Security

REST API Security implementations.

Let's try it out!

Applying Security to an API