WSO2 API Manager 4.1.0 Fundamentals - Integration Profile

Artifact Deployment

WSO2 Training

Module Objective

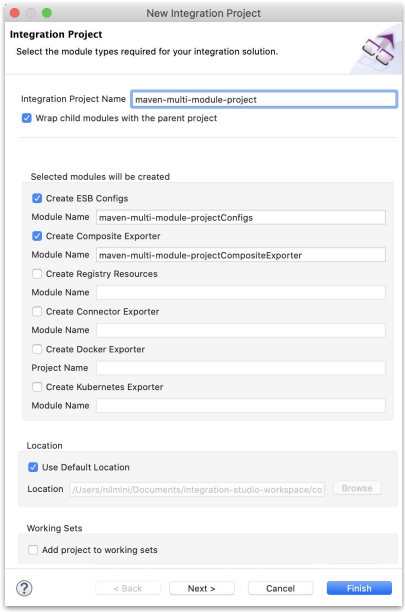
At the end of this module, attendees will be able to:

- Understand how integration artifacts are packaged for deployment.
- Understand how artifacts are deployed in a VM environment.
- Understand how artifacts are deployed in container environments.
- Externalize parameters in integration artifacts using environment variables.
- Manage environment-specific artifacts across different environments.
- Understand the basics of using a CICD pipeline for the WSO2 Micro Integrator on Kubernetes.

Integration Project

Integration Project

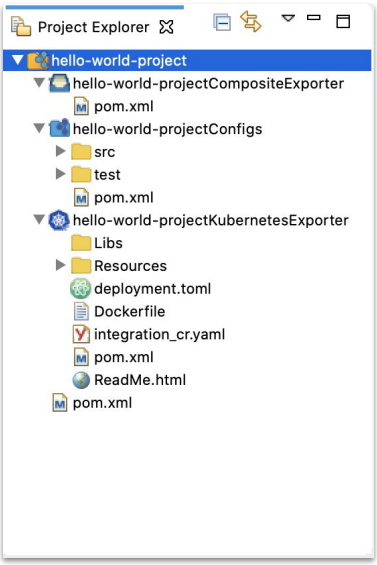
- The Integration Project you create from WSO2 Integration Studio is a **Maven Multi Module** project.
- Use this project's POM file to build the integrations solutions for your deployment.
  - Build a **composite application** for a VM deployment.
  - Build **Docker images** for a container deployment.



Integration Project

This project includes:

- Modules with integration artifacts.
  - ESB Configs
  - Connector Configs
  - Registry Resources
- Composite Exporter that packages the integration solutions in a CApp.
- Modules with deployment resources for Docker and/or Kubernetes.



Integration Project

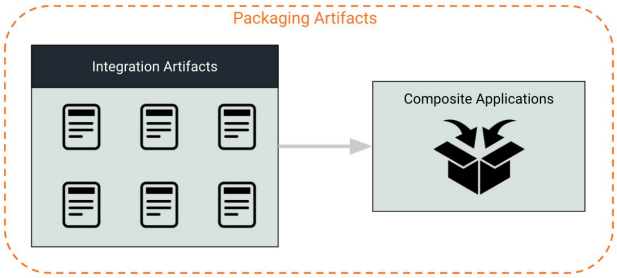
You can build individual maven profiles for the required deployment.

Profile Name	Description
Solution	Builds the integration artifacts stored in the **ESB Config** and **Composite Exporter** sub projects.
Docker	Builds the integration artifacts stored in the **ESB Config. Composite Exporter** and **Docker** sub projects.
Kubernetes	Builds the integration artifacts stored in the **ESB Config, Composite Exporter** and **Kubernetes** sub projects.

Centralized Deployment

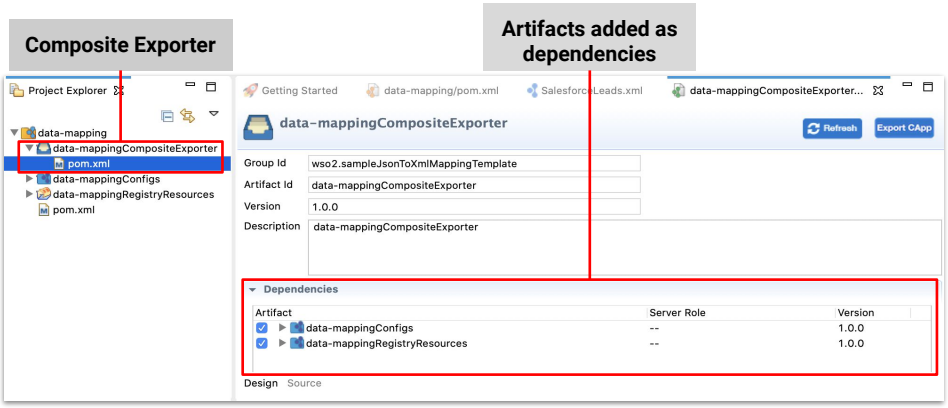
Packaging Integration Artifacts

Integration artifacts stored in your ESB Config module and supporting modules (Connector Exporter and/or Registry Resources) should be package in a **Composite Exporter** for deployment.



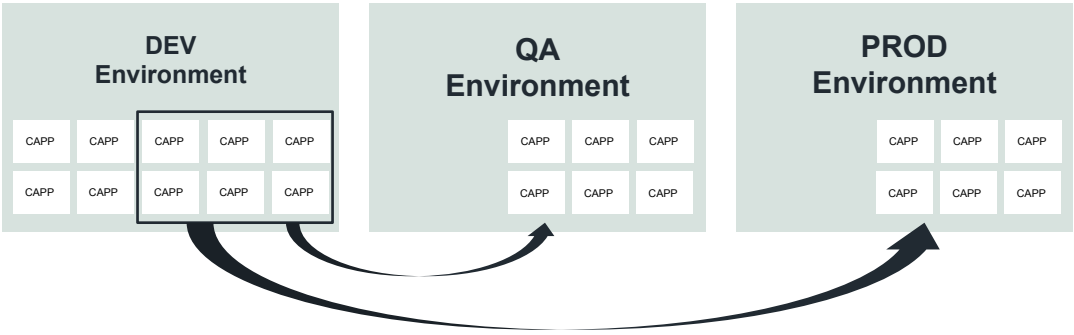
Composite Application

The composite exporter module in the integration project includes the packaged composite application of your integration solution.



Deploy in a VM

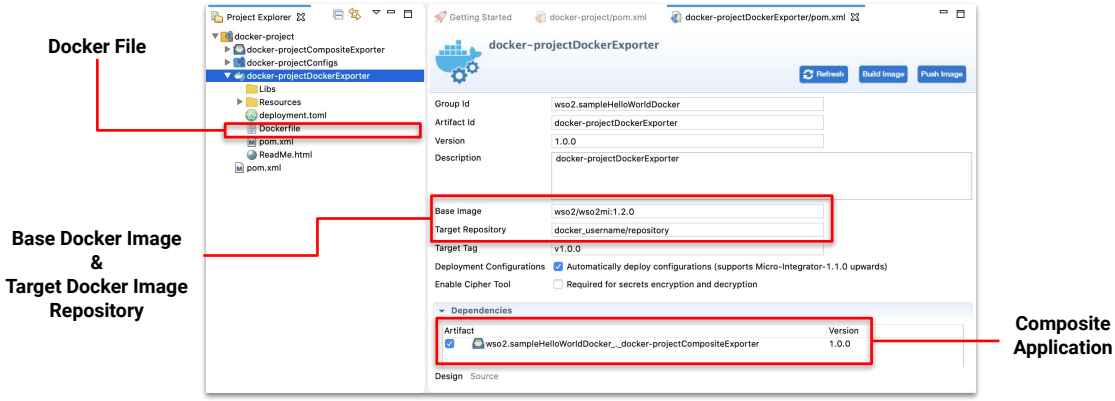
Integration solutions are deployed to VM environments by deploying the composite applications (CApps) that include the relevant artifacts.



Microservices Deployment

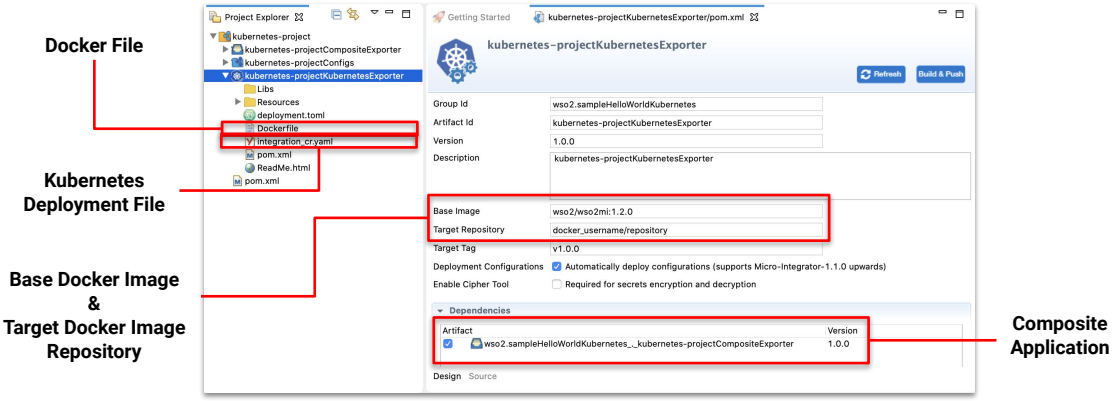
Docker Exporter

The Docker exporter module generates the Docker image with the Integration solution. This is required for your HELM-based container deployment.



Kubernetes Exporter

The Kubernetes Exporter is used to generate the Docker image and also to create the deployment in a Kubernetes cluster.



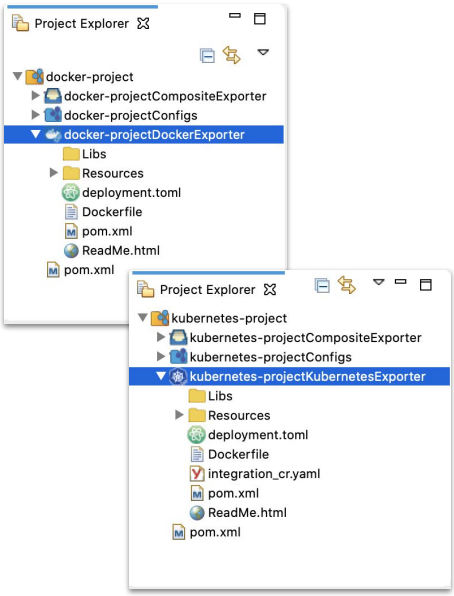
Deploying Microservices in Containers

A microservices deployment can be done in two ways:

- Using Helm Charts
- Using the EI-Kubernetes Operator

Using the EI-Kubernetes Operator

- Use the Kubernetes Exporter in WSO2 Integration Studio to create a portable Docker image.
- Use a base Micro Integrator image from the WSO2 Docker registry.
- Use the **integration_cr.yaml** file in the Kubernetes Exporter to specify the Kubernetes deployment.
- Use the EI-K8s operator to create the deployment in Kubernetes.



Using HELM Charts

- Helm charts for a WSO2 Micro Integrator deployment are available in the *WSO2 Helm Repository*.
  - https://artifacthub.io/packages/helm/wso2/micro-integrator

- Use the **Docker Exporter** in WSO2 Integration Studio to create a portable Docker image.
- Use the **values.yaml** of the HELM package to specify the Kubernetes deployment.

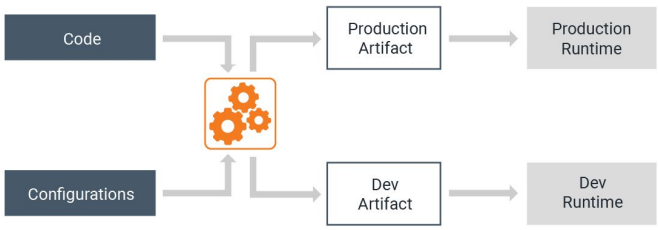
Let’s try it out!

Deploying on Kubernetes using the El-K8s Operator

Artifact Governance

Managing Artifact Configurations

Configuration parameters such as endpoint URLs change from environment to environment. This requires a new build and a new test cycle for every change.



Using Environment Variables and Files

- Use environment variables or files to externalize parameter values.
- Specially useful when deploying in containerized environments.
- Dynamically inject the parameter values to container environments.

Injecting Endpoint Parameters

1. Parameterize the config using $SYSTEM:{variable name}:

```
<?xml version="1.0" encoding="UTF-8"?>
<endpoint xmlns="http://ws.apache.org/ns/synapse" name="JSON_EP">
  <address uri="$SYSTEM:VAR"/>
</endpoint>
```

   Use $FILE:{variable name} if you want to inject endpoint parameters via a file.

2. Define variable name as an environment variable.

```
export VAR=http://localhost:61616/...
```

Supported Parameters

The following list of configuration parameters support this feature:

https://apim.docs.wso2.com/en/4.1.0/integrate/develop/injecting-parameters/#supported-parameters

Let’s try it out!

Injecting Parameters Dynamically

Continuous Integration
Continuous Deployment

Continuous Integration/Continuous Delivery (CICD)

What is CICD?

- **Continuous Integration (CI)** is the process of automatically detecting, pulling, building, and running unit testing as your integration solutions are periodically changed.
- **Continuous Delivery (CD)** is the overall chain of processes (pipeline) that automatically detects changes and runs them through build, test, packaging, and related operations to produce a deployable integration solutions.

VM CICD Pipeline

Integration Project Build Job

- Maintain one Jenkins job per Integration Project repository.
- The build phase will build the project and run the unit tests.
- The release phase of the job will publish the CApps to the Nexus repository and create a release tag in GitHub.

Deployment Descriptor Build Job

- Maintain one Jenkins job per Environment.
- There will be descriptor files for each project inside a separate folder.
- Use a new version/rollback to a previous version by defining the change inside the descriptor and commit to the branch.
- This job contains only the build phase.

Kubernetes CICD Pipeline

Integration Project Build Job

- Maintain one Jenkins job per Integration Project repository.
- The build phase will build the Integration project and run the unit tests.
- The release phase generates Docker images, name and project version and pushes the image to the configured Docker registry and creates a release tag in GitHub.

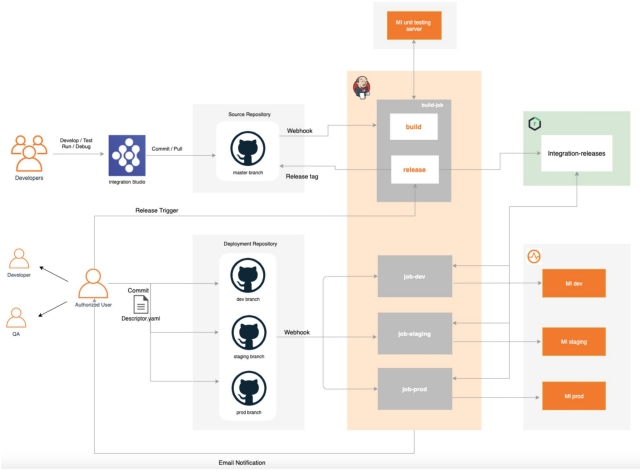
Deployment Descriptor Build Job

- Maintain one Jenkins job per Environment.
- There will be descriptor files generated for each project inside a separate folder.
- To use a new version/rollback to a previous version, define the version inside the `integration_cr.yaml` or `integration_k8s.yaml` and commit to the branch.
- This job contains only the build phase.

VM Micro Integrator CICD Pipeline

Tools in the VM CICD Pipeline of WSO2 Micro Integrator:

- **CI/CD Tool**:
  - Jenkins
- **Artifact Repository**:
  - Nexus



Kubernetes Micro Integrator CICD Pipeline

Tools in the Kubernetes CICD Pipeline of WSO2 Micro Integrator:

- **CI/CD Tool**:
  - Jenkins
- **Artifact Repository**:
  - Docker Registry