INTERNAL

2022-01-24

Continuous Integration and Delivery Best Practices Guide

THE BEST RUN

Content

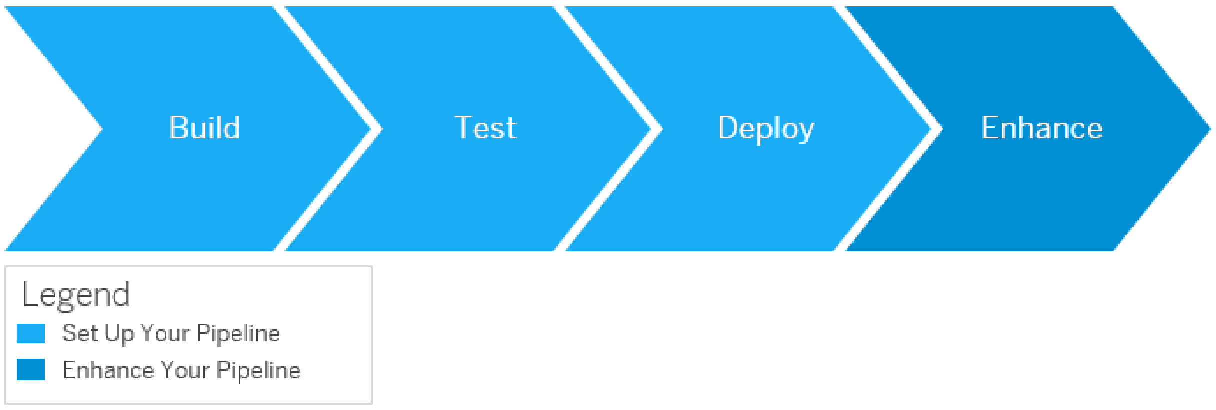
1 Continuous Integration and Delivery Best Practices Guide

Implement continuous integration for SAP-specific development projects.

What Is This Guide About?

The Continuous Integration and Delivery Best Practices Guide provides simple procedures to implement continuous delivery (CD) pipelines on any CI/CD stack. It does not describe fully fledged pipelines, but rather demonstrates how to apply the principles of CI/CD to SAP-specific technologies. For more information about the concepts and principles of CI/CD, have a look at the Continuous Integration and Delivery Introduction Guide.

In this guide, each procedure describes one specific SAP scenario and either focuses on the core stages of a pipeline (build, test, and deploy) or helps you enhance your existing pipeline with SAP BTP services. See Procedures for CI/CD Pipelines [page 5].



Set Up Your Pipeline vs. Enhance Your Pipeline

As they are supported by most CI tools, the Continuous Integration and Delivery Best Practices Guide proposes implementations using Bash.

Is This Guide for You?

The Continuous Integration and Delivery Best Practices Guide addresses **customers who want to use their existing CI infrastructure** (exept for Jenkins, see the following tip) and **experts in CI/CD who want to have full flexibility when implementing their pipelines**.

→ Tip

If you use Jenkins or plan to use it, consider working with project "Piper" , instead.

For a full overview of the different solutions SAP provides for CI/CD, see SAP Solutions for Continuous Integration and Delivery.

What You Should Consider Before Getting Started

To keep the setup of your CI environment and the installation of tools used in your pipelines as simple as possible, in our procedures, we use Docker containers whenever applicable. However, in our scripting examples, we also provide alternatives without using Docker.

For more information on using Docker and its advantages, see SAP Solutions for Continuous Integration and Delivery.

⚠ Caution

Please check with your IT and security departments how to handle Docker images from public sources.

2 Procedures for CI/CD Pipelines

Depending on your scenario, choose one of the listed procedures.

> → Tip
>
> If you use Jenkins or plan to use it, have a look at **project "Piper"** , instead.

For a full overview of the different solutions SAP provides for CI/CD, see SAP Solutions for Continuous Integration and Delivery.

Set Up Your Pipeline

- **Apply CI/CD to SAP Fiori Development on SAP BTP** [page 6]
  Implement a CI/CD pipeline for the development of SAPUI5/SAP Fiori applications on SAP BTP in the Neo environment.
- **Apply CI/CD to SAP Fiori Development on an SAP Fiori Front-End Server** [page 11]
  Implement a CI/CD pipeline for the development of SAPUI5/SAP Fiori applications on an SAP Fiori front-end server .
- **Apply CI/CD to SAP HANA Extended Application Services, Advanced Model Development** [page 25]
  Implement a CI/CD pipeline for the development of SAP HANA extended application services, advanced model applications.

Enhance Your Pipeline

- **Integrate SAP Cloud Transport Management into Your CI/CD Pipeline** [page 30]
  Add an enterprise-ready change and release management process to your CI/CD pipieline and enable the transport of cloud-based applications on SAP BTP.
- **Integrate Change Control Management with SAP Solution Manager into Your CI/CD Pipeline** [page 35]
  Add change control management processes to your CI/CD pipeline and facilitate the synchronization of changes made both on-premise and on SAP BTP.

2.1 Apply CI/CD to SAP Fiori Development on SAP BTP

Implement a CI/CD pipeline for the development of SAPUI5/SAP Fiori applications on SAP BTP in the Neo environment.

> → Tip
>
> If you use Jenkins or plan to use it, have a look at the **project "Piper"** scenario Build and Deploy SAPUI5/SAP Fiori Applications on SAP BTP, instead.

For a full overview of the different solutions SAP provides for CI/CD, see SAP Solutions for Continuous Integration and Delivery.

Context

This procedure explains how to wrap your SAPUI5/SAP Fiori application into a multitarget application (MTA) to produce an archive that is deployable on SAP BTP. As of today, many CI tools provide Bash scripting support and you can implement the corresponding Bash commands in a CI tool of your choice.

A multitarget application (MTA) is a package that comprises multiple application and resource modules, which share a common lifecycle even though they might have been created with different technologies and deployed to different runtimes. To create a MTA, you bundle the modules, describe them along with their interdependencies to other modules, services, and interfaces, and package them. MTAs can be deployed to SAP HANA Extended Application Services, Advanced Model, Cloud Foundry, and Neo environments. In this procedure, you deploy the resulting .mtar file to the SAP BTP Neo environment. For more information, see The Multitarget Application Model.

Prerequisites

- As it prevents clutter on your build system and improves both maintainability and flexibility, we recommend working with Docker . Refer to the documentation of your CI tool to check whether it supports the usage of Docker containers in its pipelines.

  > ⚠ Caution
  >
  > Please check with your IT and security departments how to handle Docker images from public sources.

- You have an SAPUI5/SAP Fiori application as source in a source code management system of your choice. To build it, you can choose between two different tooling options:
  - UI5 Tooling
  - Grunt

  > i Note
  >
  > If you generate your project in SAP Web IDE, depending on its version, the builder is preconfigured to either Grunt or UI5 tooling.

Depending on your tooling, expand one of the following sections and make sure that you meet the described prerequisites:

Additional Prerequisites When Using the UI5 Tooling

On the top level of your directory, you have:

- An mta.yaml which contains the following script:

  Sample Code

  ```
  ID: openui5-sample-app
  version: 1.0.0
  _schema-version: '2.1'
  modules:
    - name: webapp
      type: html5
      path: .
      parameters:
        disk-quota: 256M
        memory: 256M
      build-parameters:
        builder: npm
  ```

- A package.json file which contains the following build script:

  ```
  "scripts": {
    "build": "ui5 build --a",
    "test": "uiveri5"
    "linting": "eslint ."
  }
  ```

  i Note

  The build script executes the UI5 command-line tool. In this example, the test script runs UIVeri5. You can, however, substitute `uiveri5` with any other automated tests you have implemented.

Additional Prerequisites When Using Grunt

On the top level of your directory, you have:

- A mta.yaml file which contains the following script:

  Sample Code

  ```
  ID: openui5-sample-app
  version: 1.0.0
  _schema-version: '2.1'
  modules:
    - name: webapp
      type: html5
      path: .
      parameters:
        disk-quota: 256M
        memory: 256M
      build-parameters:
        builder: npm
  ```

- A package.json file which contains the following build script:

  ```
  "scripts": {
  ```

```
   "build": "ui5 build --a",
   "test": "uiveri5"
   "linting": "eslint ."
 }
```

i Note

The build script executes the UI5 command-line tool. In this example, the test script runs UIVeri5. You can, however, substitute `uiveri5` with any other automated tests you have implemented.

- A local .npmrc file that contains the following entry to download the `@sap/grunt-sapui5-bestpractice-build`:

```
registry=https://registry.npmjs.org/
@sap:registry=https://npm.sap.com/
```

i Note

If you use our recommended Docker image, this reference already exists.

- A Grunt file which contains the following:

```
module.exports = function (grunt) {
            "use strict";
            grunt.loadNpmTasks("@sap/grunt-sapui5-bestpractice-build");
            grunt.config.merge({ compatVersion: "1.38" });
            grunt.registerTask("default", [
                        "clean",
                        "lint",
                        "build"
            ]);
            grunt.loadNpmTasks("@sap/grunt-sapui5-bestpractice-test");
            grunt.registerTask("unit_and_integration_tests", ["test"]);
            grunt.config.merge({
                        coverage_threshold: {
                                    statements: 0,
                                    branches: 100,
                                    functions: 0,
                                    lines: 0
                        }
            });
};
```

For more information on how to set up your Grunt build in SAP Web IDE Full-Stack, see Grunt Build in SAP Web IDE Full-Stack.

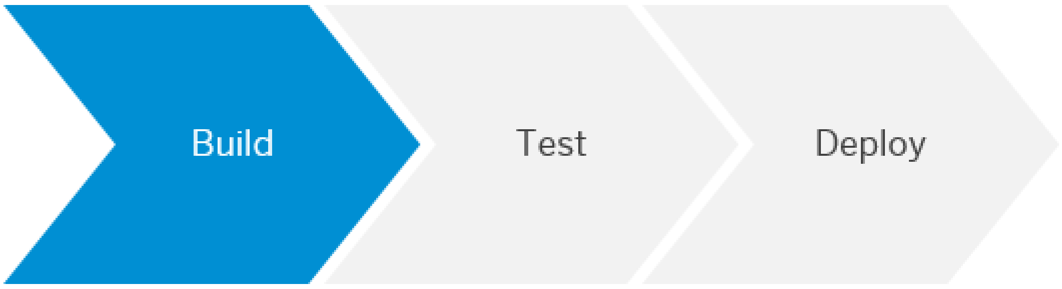
Procedure

In this procedure, we focus on the core stages of a pipeline: build, test, and deploy.

→ Tip

You can combine and enhance these simple build, test, and deploy steps to implement more complex pipelines.

1. **Build**



The build optimizes and packages your project sources. Use the Cloud MTA Build Tool to orchestrate the technical build steps using Node Package Manager (npm), which are defined in the package.json. See Cloud MTA Build Tool . Depending on your configuration, you can also add linting and unit tests. The build produces an out.mtar file, which is used in the following deployment stage. See step 3: **Deploy**.
For more information on how to download, set up, and run the Cloud MTA Build Tool, see Setting Up and Using the Multitarget Application Archive Builder.
To run the build, choose one of the following options:

- **With Docker:**
  Execute the following command in the directory that contains your project sources:

  ```
  docker run -v "${PWD}":/project devxci/mbtci:latest mbt --mtar out.mtar --
  platform CF build
  ```

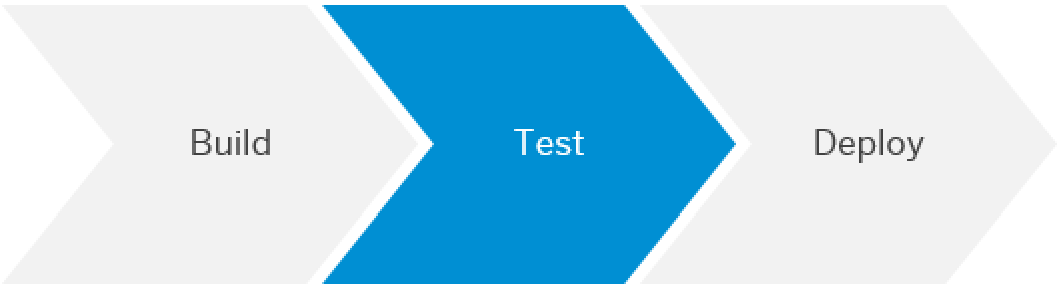
  > i Note
  >
  > As many CI tools automatically mount your workspace that contains the project sources into your Docker container, you may omit the `-v` command, which mounts your current working directory.

  > → Tip
  >
  > If you change the build target to `CF` or `XSA`, you can reuse this call for other scenarios.

- **Without Docker:**
  If you provide all dependencies on the build server, you can build your project without Docker by placing the multitarget application archive builder directly on your build system. See https://tools.hana.ondemand.com/#cloud.
  Execute the following command in the directory that contains your project sources:

  ```
  java -jar /path/to/mta.jar --mtar out.mtar --build-target NEO build
  ```

2. **Test**



Depending on your requirements and the setup of your project, consider adding automated tests to it. In your package.json, you have already configured them with the following lines:

```
   "test": "uiveri5"
```

```
    "linting": "eslint ."
```

For SAPUI5/ SAP Fiori, we recommend implementing UIVeri5 and OPA5 tests. See, for example Add Automated System Tests with the SAPUI5 Test Recorder to Your CI/CD Pipeline and Testing.
To add automated tests, choose one of the following options:

- **With Docker:**
  From the Docker community, use a standard image using Node.js in a version that fits your requirements and execute the following commands.

  ```
  docker run -v "${PWD}":/project node:latest npm run-script test
  ```

  ```
  docker run -v "${PWD}":/project node:latest npm run-script linting
  ```

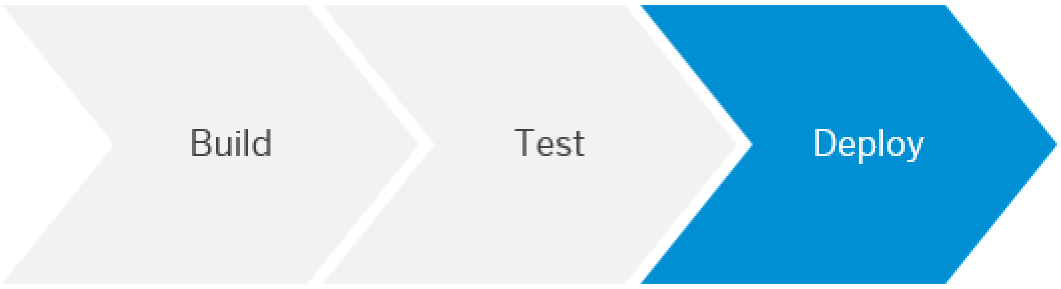
  > **i Note**
  >
  > As many CI tools automatically mount your workspace that contains the project sources into your Docker container, you may omit the `-v` command, which mounts your current working directory.

- **Without Docker:**
  In your shell, execute the following commands:

  ```
  npm run-script test
  ```

  ```
  npm run-script linting
  ```

3. **Deploy**



Depending on whether you work or do not work with Docker, use one of the following commands to deploy your application to the Neo environment. In a shell environment, provide the required variables: SAP BTP host, account, and user credentials. Please refer to the documentation of your specific CI tool for how to store variables.
To upload the resulting out.mtar file from the build step to SAP BTP, use the `deploy-mta` command of the Console Client for the Neo environment, which you can download from https://tools.hana.ondemand.com/#cloud and which is part of any SAP BTP SDK for the Neo environment. For more information, see Console Client for the Neo Environment.

> **⚠ Caution**
>
> Don't put credentials in the call but use the credential mechanism of your CI tool, instead.

- **With Docker:**
  Adapt the following command by replacing the placeholders in brackets with your actual values and execute it.

  ```
  docker run -v "${PWD}":/project ppiper/neo-cli:latest neo.sh deploy-mta --
  host ${HOST} --account ${ACCOUNT} --source out.mtar --synchronous --user $
  {USER} --password ${PASS}
  ```

  > i Note
  >
  > As many CI tools automatically mount your workspace that contains the project sources into your Docker container, you may omit the `-v` command, which mounts your current working directory.

- **Without Docker:**
  Place the SDK for the Neo environment of your choice on the build machine and call it.

  ```
  /path/to/cloud-platform-sdk/tools/neo.sh deploy-mta --host ${HOST} --
  account ${ACCOUNT} --source out.mtar --synchronous --user ${USER} --
  password ${PASS}
  ```

For more options, see Console Client Commands.

Result

You have built a basic CI/CD pipeline, which you can enhance according to your needs, for example, by adding additional tests and manual release steps.

> i Note
>
> This guide also provides other procedures to enhance your finished CI/CD pipelines. See Procedures for CI/CD Pipelines [page 5].

2.2 Apply CI/CD to SAP Fiori Development on an SAP Fiori Front-End Server

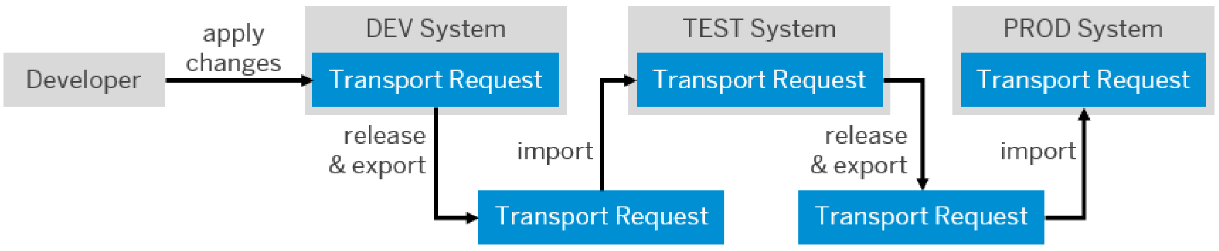
Implement a CI/CD pipeline for the development of SAPUI5/SAP Fiori applications on an SAP Fiori front-end server.

> → Tip
>
> If you use Jenkins or plan to use it, have a look at **project "Piper"** , instead.

For a full overview of the different solutions SAP provides for CI/CD, see SAP Solutions for Continuous Integration and Delivery.

Context

The most common ABAP system setup is a staging landscape that consists of three different systems: One for development (DEV), one for acceptance testing (TEST), and a productive one (PROD). In this order, they are linked to each other through transport routes. Following these routes, transport requests are exported from one system and imported into the next. The following graphic illustrates this procedure:



Transport Requests in a Typical ABAP System Setup

Your staging landscape must at least consist of two different systems: DEV and PROD. Depending on your needs, however, you can combine as many systems as you like.

> **i Note**
>
> The following procedure describes the transport from one system to another. Depending on how many systems you use, repeat parts of it as often as necessary. For more information, see section **The Transport Management System**.

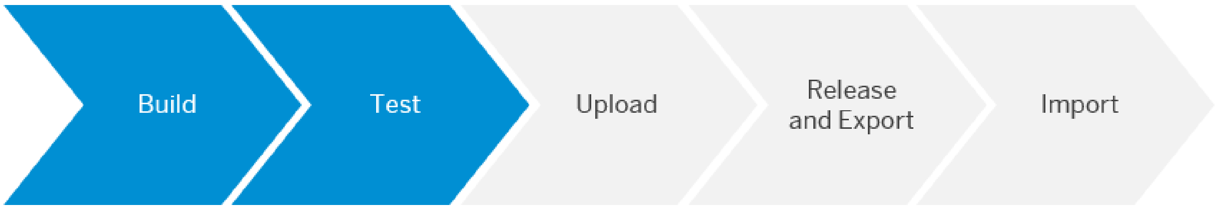
When combining continuous integration with the ABAP life-cycle management, the CI process ensures the quality of the code implemented outside the ABAP system. Through minification and the generation of a preload, it converts the sources into the correct format, runs automated tests and code checks, and produces an artifact that is ready to be uploaded to the ABAP development system. From there, the ABAP system takes over and transports the changes to the test system, and finally to the productive one.

Therefore, the described workflow consists of two distinct parts:

> **→ Tip**
>
> Expand the following sections for more detailed information.

The Continuous Integration Process

The CI process outside the ABAP system contains the following steps:



In detail, this process works as follows:

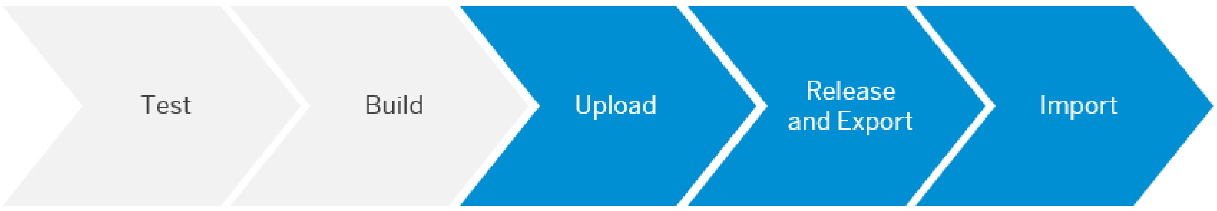
1. In a source code management tool of your choice, apply changes to your SAPUI5/SAP Fiori project.
2. Merge your changes with the master branch. Thereby, the CI process is triggered.

3. The CI process runs. It includes the following tasks:
    - Static code checks for JavaScript
    - Automated tests
    - Minification, which means that all comments and white spaces are removed from the JavaScrip sources to reduce the load on the network
    - Generating a preload, which means that all JavaScript files are merged into a single file to reduce the number of requests from the browser to the server
    - Packaging the application into a ZIP file

    The last three tasks of the CI process constitute the **Build** step.

The Transport Management System

The mainly automated delivery process with the Transport Management System contains the following steps:



In detail, this process works as follows:

1. Immediately after a successful CI build, the build scheduler is triggered. It performs the following tasks:
    - Creating a new, individual transport request in the ABAP development system
    - Uploading the application to the ABAP development system
    - Releasing the transport request
2. Manually import the transport request into the following ABAP system.

> i Note
>
> If in your ABAP system setup, you use more than two different systems, repeat the **Release and Export** and **Import** steps as often as necessary.

OData vs. RFC

To orchestrate the Transport Management System, your CI/CD pipeline directly communicates with the ABAP development system. Depending on which version of SAP NetWeaver you use, choose between the following protocols:

- **Open Data Protocol (OData)**
  OData is a REST-based web protocol for querying and updating data as well as for applying and building on web technologies. It is used by SAP Gateway and recommended for SAP NetWeaver systems **>= 7.5 SPS12**. For more information, see SAP Gateway, REST and OData.
- **Remote Function Call (RFC)**
  RFC calls a function to be executed in a remote system and is the standard SAP interface for communication between SAP NetWeaver systems. For more information, see RFC.

i Note

Although you can use RFC for all SAP NetWeaver systems, we recommend OData for newer ones (>= 7.5 SPS12).

Depending on which protocol you use, choose one of the following procedures:

- **Apply CI/CD to SAP Fiori Development on an SAP Fiori Front-End Server with OData [page 14]**
- **Apply CI/CD to SAP Fiori Development on an SAP Fiori Front-End Server with RFC [page 18]**

For an overview of all procedures in this guide, see Procedures for CI/CD Pipelines [page 5].

2.2.1 Apply CI/CD to SAP Fiori Development on an SAP Fiori Front-End Server with OData

Implement a CI/CD pipeline for the development of SAPUI5/SAP Fiori applications on an SAP Fiori front-end server with OData.

→ Tip

If you use Jenkins or plan to use it, have a look at Project "Piper" , instead.

For a full overview of the different solutions SAP provides for CI/CD, see SAP Solutions for Continuous Integration and Delivery.

Context

This procedure allows you to implement a CI/CD pipeline to develop a SAPUI5/SAP Fiori application locally and to run it on an SAP Fiori front-end server using an OData service.

The **SAP Fiori front-end server** is an add-on product for SAP NetWeaver Application Server for ABAP (AS ABAP). See SAP Fiori front-end server.

The **OData service** is an interface of the Change and Transport System (CTS) with which you can manage transport requests and upload your SAPUI5/SAP Fiori application.

Prerequisites

- You have an SAPUI5/SAP Fiori application as source in a source code management system of your choice.
- You have installed the SAP component SAP_UI 7.53 or higher on your ABAP system.
- You have enabled the OData service to load data to the SAPUI5 ABAP repository.
- You have the S_DEVELOP authorization to perform operations in your SAPUI5 ABAP repository.

Procedure

Use the following procedure to create a pipeline in a CI/CD tool of your choice. The pipeline consists of two distinct parts: a **continuous integration** process and an **ABAP life-cycle management** process.

It uses **npm** as a

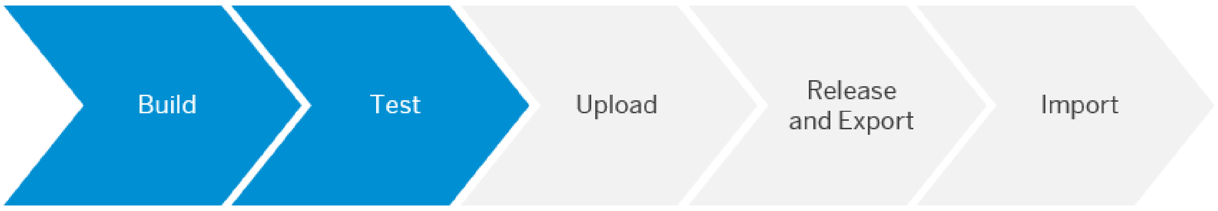
- package manager to set up the required environment on your build system,
- task runner to run and automate the pipeline steps.

For more information, see npm .

→ Recommendation

Use the containerization platform **Docker** to avoid scrambling your build system. See Docker .

Continuous Integration



The continuous integration process comprises the following steps:

- The **Build** step, which builds your SAPUI5/SAP Fiori application.
- The **Test** step, which runs tests on your SAPUI5/SAP Fiori application.

Preparing Your Build System

To prepare your build system, proceed as follows:

1. If you don't want to use Docker, install the following dependencies on your build machine:
   - Node.js
   - CM Client
2. Create a `package.json` file in the top-level directory of your project, if it doesn't exist yet. See package.json .
   Add the following lines:

```
{
   ...
   "scripts": {
      "lint": "eslint webapp",
      "uiveri5": "uiveri5",
      "test": "npm run lint && npm run uiveri5",
      "build": "ui5 build --clean-dest",
      "build-self-contained": "ui5 build self-contained -a --clean-dest",
      "start": "ws --compress -d dist"
   },
    ...
   "devDependencies": {
      "@ui5/cli": "^2.11.2",
      "eslint": "^7.29.0",
      "local-web-server": "^4.2.1",
   }
   ...
}
```

Build

To build your SAPUI5/SAP Fiori application, proceed as follows:

- In your shell, execute the following command:

  ```
  docker run -v "${PWD}":/project --workdir "/project" node:latest bash -c "npm
  run-script build"
  ```
- If you don't use Docker, run the npm script in your shell:

  ```
  npm run-script build
  ```

Test

To run tests on your SAPUI5/SAP Fiori application, proceed as follows:

- In your shell, execute the following command:

  ```
  docker run -v "${PWD}":/project --workdir "/project" node:latest bash -c "npm
  run-script test"
  ```
- If you don't use Docker, run the npm script in your shell:

  ```
  npm run-script test
  ```

  The test script executes **lint** and **uiVeri5** tests.

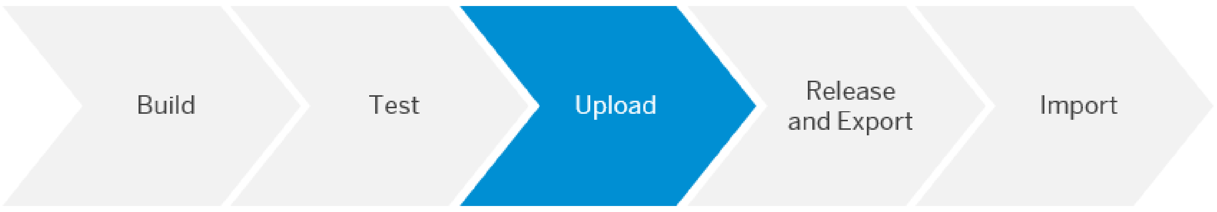
  > i Note
  >
  > For more information about how to create a uiVeri5 test, see UIVeri5 . You can, however, substitute UIVeri5 tests with any other automated tests you have implemented.

The ABAP Life-Cycle Management

Run the following steps to deliver and release your SAPUI5/SAP Fiori application:

- The **Upload** step, which uploads your SAPUI5/SAP Fiori application into a transport request of your developer system.
- The **Release & Export** step, which releases and exports your transport request to be imported into the next system of the transport route.
- The **Import** step, which imports your exported transport request into the next system of the transport route.
  For more information, see Transport Requests in a Typical ABAP System Setup

Upload



To upload a file into your transport request, proceed as follows:

1. Create a transport request, if it is not already existing:
   - Use the **CM Client** if you want to automate the creation. See CM Client .
     - In your shell, execute the following command:

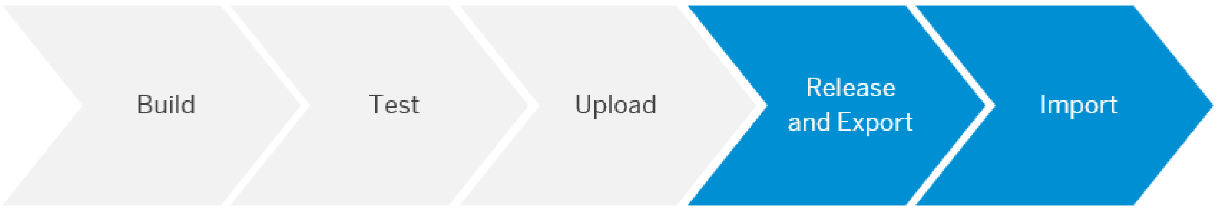
       ```
       docker run ppiper/cm-client:latest cmclient --endpoint "<ENDPOINT>" --
       user "<USER>" --password "<PASSWORD>" --backend-type CTS  create-
       transport --target-system "<TARGET SYSTEM>" --transport-type
       "<TRANSPORT TYPE>"  --owner "<OWNER>" --description "<DESCRIPTION>"
       ```
     - Without Docker, run the CM Client in your shell:

       ```
       cmclient --endpoint "<ENDPOINT>" --user "<USER>" --password
       "<PASSWORD>" --backend-type CTS  create-transport --target-system
       "<TARGET SYSTEM>" --transport-type "<TRANSPORT TYPE>"  --owner
       "<OWNER>" --description "<DESCRIPTION>"
       ```
   - Use the Transport Organizer Web UI of the **SAP Transport Management System** if you want to manually create the transport request.
2. Use **SAP Fiori Tools** to upload your SAPUI5/SAP Fiori application. See SAP Fiori Tools .
   - In your shell, execute the following command:

     ```
     docker run --env ABAP_USER="<USER>" --env ABAP_PASSWORD="<PASS>" -v "$
     {PWD}":/project --workdir "/project" node:latest bash -c "npm run-script
     upload -- --noConfig -f -y --username ABAP_USER --password ABAP_PASSWORD --
     url <URL_ENDPOINT> --client <CLIENT> --transport <TRANSPORT_ID> --package
     <PACKAGE> --name <APP_NAME> --description \"<DESCRIPTION>\""
     ```
   - Without Docker, run the npm script in your shell:

     ```
     export ABAP_USER=<USER>
     export ABAP_PASSWORD=<PASS>
     npm run-script upload -- --noConfig -f -y --username ABAP_USER --password
     ABAP_PASSWORD --url <URL_ENDPOINT> --client <CLIENT> --transport
     <TRANSPORT_ID> --package <PACKAGE> --name <APP_NAME> --description
     "<DESCRIPTION>"
     ```

   For more information about the command and its parameters, see fiory deploy .



Release & Export

To deliver and release your SAPUI5/SAP Fiori application, use SAP Logon and proceed as follows:

1. In SAP Logon, open the transaction `stms`.
2. Choose *Transport Organizer Web UI*.
3. Select your transport request and choose *Release*.

Import

To import your transport request into the following system, use SAP Logon and proceed as follows:

1. In SAP Logon, open the transaction `stms`.

2. Choose *Import Overview*.
3. Choose your target system and choose *Display Import Queue*.
4. To import the waiting request, select it and choose *OK*.

Result

You have combined the ABAP life-cycle management with a basic CI pipeline, which you can enhance according to your needs.

> **i Note**
>
> This guide also provides procedures to enhance your finished CI/CD pipelines. See Procedures for CI/CD Pipelines [page 5].

2.2.2 Apply CI/CD to SAP Fiori Development on an SAP Fiori Front-End Server with RFC

Implement a CI/CD pipeline for the development of SAPUI5/SAP Fiori applications on an SAP Fiori front-end server with RFC.

> **→ Tip**
>
> If you use Jenkins or plan to use it, have a look at **project "Piper"** , instead.

For a full overview of the different solutions SAP provides for CI/CD, see SAP Solutions for Continuous Integration and Delivery.

Prerequisites

- You have an SAPUI5/SAP Fiori application as source in a source code management system of your choice.
- In the top level directory of your project, you have a file called `Gruntfile.js`, whicht contains the following entries:

```
module.exports = function (grunt) {
            "use strict";
            grunt.loadNpmTasks("@sap/grunt-sapui5-bestpractice-build");
            grunt.config.merge({ compatVersion: "1.38" });
            grunt.registerTask("default", [
                        "clean",
                        "lint",
                        "build"
            ]);
            grunt.loadNpmTasks("@sap/grunt-sapui5-bestpractice-test");
            grunt.registerTask("unit_and_integration_tests", ["test"]);
            grunt.config.merge({
                        coverage_threshold: {
```

```
                                    statements: 0,
                                    branches: 100,
                                    functions: 0,
                                    lines: 0
                        }
            });
};
```

For more information on how to set up your Grunt build in SAP Web IDE Full-Stack, see Grunt Build in SAP Web IDE Full-Stack.

- In the top level directory of your project, you have a file called `package.json`, which contains the following lines:

```
"devDependencies": {
  "@sap/grunt-sapui5-bestpractice-build": "1.4.1",
  "@sap/grunt-sapui5-bestpractice-test": "2.0.1"
},
"scripts": {
  "build": "ui5 build --a",
  "test": "uiveri5"
  "linting": "eslint ."
}
```

> **i Note**
>
> The build script executes the UI5 command line tool. In this example, the test script runs UIVeri5. You can, however, substitute `uiveri5` with any other automated tests you have implemented.

- As it prevents clutter on your build system and improves both maintainability and flexibility, we recommend working with Docker . Refer to the documentation of your CI tool to check whether it supports the usage of Docker containers in its pipelines.

> **⚠ Caution**
>
> Please check with your IT and security departments how to handle Docker images from public sources.

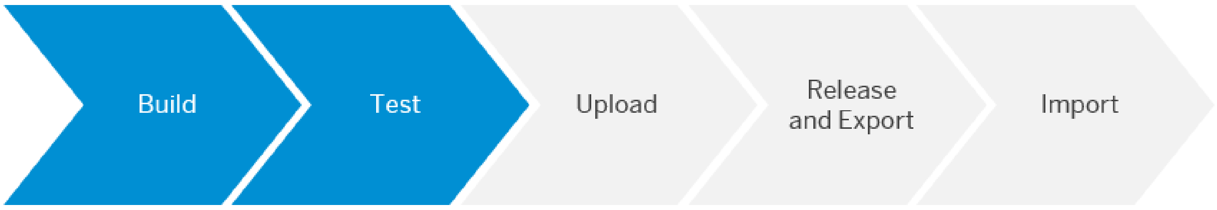
Depending on whether you want or don't want to work with Docker, make sure that you meet the following additional prerequisites:

- **With Docker:**
  You have built the RFC CTS+ Dockerfile .
- **Without Docker:**
  - You have installed NodeJS on your build machine.
  - You have downloaded and installed the SAP NetWeaver RFC Library.

Procedure

Use the following procedure to create a pipeline in a CI/CD tool of your choice. It consists of two distinct parts: The continuous integration process and the ABAP life-cycle management. For more information, see Apply CI/CD to SAP Fiori Development on an SAP Fiori Front-End Server [page 11].

The Continuous Integration Process



1. To install all needed dependencies for your application, call Grunt, and execute the tasks in your `gruntfile.js`, choose one of the following options:
   - **With Docker:**
     In your shell, execute the following commands:

     ```
     docker run -v "${PWD}":/project node:latest npm install
     ```

     ```
     docker run -v "${PWD}":/project node:latest node_modules/grunt-cli/bin/
     grunt default createZip
     ```

     i Note

     The `-v` command mounts your current working directory. As many CI tools automatically mount the workspace, which contains the project sources into your Docker container, you may omit it.
   - **Without Docker:**
     In your shell, execute the following commands:

     ```
     npm install
     ```

     ```
     node_modules/grunt-cli/bin/grunt default createZip
     ```
2. To add automated tests to your pipeline, choose one of the following options:
   - **With Docker:**
     Use a standard node image that fits your requirements from the Docker Hub. In your shell, execute the following commands:

     ```
     docker run -v "${PWD}":/project node:latest npm run-script test
     ```

     ```
     docker run -v "${PWD}":/project node:latest npm run-script linting
     ```
   - **Without Docker:**
     In your shell, execute the following commands:

     ```
     npm run-script test
     ```

     ```
     npm run-script linting
     ```

   → Tip

   For SAPUI5/ SAP Fiori, we recommend implementing UIVeri5 and OPA5 tests. See, for example Add Automated System Tests to Your CI/CD Pipeline.
3. Trigger your pipeline to test the CI build.

The ABAP Life-Cycle Management

Build
Test
Upload
Release and Export
Import

1. To define the logic for RFC connections, create a Gruntfile with the name `run-rfc-task.js` on the top level of your directory, and copy the following code into it:

```
"use strict";
var rfc = require("node-rfc");
var fs = require("fs");
module.exports = function(grunt) {
    // Project specific variables
    var abapDevelopmentUser = process.env.ABAP_DEVELOPMENT_USER;
    var abapDevelopmentPassword = process.env.ABAP_DEVELOPMENT_PASSWORD;
    var abapDevelopmentServer = process.env.ABAP_DEVELOPMENT_SERVER;
    var abapDevelopmentInstance = process.env.ABAP_DEVELOPMENT_INSTANCE;
    var abapDevelopmentClient = process.env.ABAP_DEVELOPMENT_CLIENT;
    var abapApplicationName = process.env.ABAP_APPLICATION_NAME;
    var abapApplicationDesc = process.env.ABAP_APPLICATION_DESC;
    var abapPackage = process.env.ABAP_PACKAGE;
    var zipFileURL = process.env.ZIP_FILE_URL;
    var codePage = process.env.CODE_PAGE;
    var acceptUnixStyleLineEndings = process.env.ABAP_ACCEPT_UNIX_STYLE_EOL;
    var transportDescription = process.env.TRANSPORT_DESCRIPTION;
    var targetDir = process.env.SAPDATADIR;
    var verbose = process.env.VERBOSE;
    var failUploadOnWarning = process.env.FAIL_UPLOAD_ON_WARNING;
    // Global Variables
    var ctsDataFile = targetDir + "/CTS_Data.txt";
    // Project configuration.
    var abapConn = {
        user: abapDevelopmentUser,
        passwd: abapDevelopmentPassword,
        ashost: abapDevelopmentServer,
        sysnr: abapDevelopmentInstance,
        client: abapDevelopmentClient
    };
    grunt.initConfig({
        pkg: grunt.file.readJSON("package.json"),
        createTransportRequest: {
            options: {
                conn: abapConn,
                author: abapDevelopmentUser,
                description: transportDescription,
                verbose: verbose
            }
        },
        uploadToABAP: {
            options: {
                conn: abapConn,
                zipFileURL: zipFileURL,
                codePage: codePage,
                acceptUnixStyleLineEndings: acceptUnixStyleLineEndings,
                verbose: verbose,
                failUploadOnWarning: failUploadOnWarning
            }
        },
        releaseTransport: {
            options: {
                conn: abapConn,
                verbose: verbose
            }
        }
    });
    var rfcConnect = function(functionModule, importParameters, gruntContext)
{
        return new Promise(function(resolve, reject) {
            var verbose = gruntContext.options().verbose
            var conn = gruntContext.options().conn;
            var client = new rfc.Client(conn);
            grunt.log.writeln("RFC client lib version:", client.version);
            client.connect(function(err) {
                if (err) { // check for login/connection errors
                    grunt.log.errorlns("could not connect to server", err);
                    return reject();
                }
                // invoke remote enabled ABAP function module
                grunt.log.writeln("Invoking function module", functionModule);
                client.invoke(functionModule,
                    importParameters,
                    function(err, res) {
                        if (err) { // check for errors (e.g. wrong parameters)
                            grunt.log.errorlns("Error invoking",
functionModule, err);
                            return reject();
                        }
                        client.close();
                        if(verbose == 'true') {
                            grunt.log.writeln("Result:", res);
                        }
                        return resolve(res);
                    });
            });
        });
    };
    grunt.registerTask("createTransportRequest", "Creates an ABAP Transport
Request", function() {
        grunt.log.writeln("Creating Transport Request");
        var importParameters = {
            AUTHOR: this.options().author,
            TEXT: this.options().description
        };
        var done = this.async();
        rfcConnect("BAPI_CTREQUEST_CREATE", importParameters, this)
            .then(
            function(returnValue) {
                if  (returnValue.RETURN.TYPE == "E" ||
returnValue.RETURN.TYPE == "W") {
                    grunt.log.errorlns("Error invoking
BAPI_CTREQUEST_CREATE.");
                    grunt.log.writeln("Return:", returnValue);
                    done(false);
                    return;
                }
                if (returnValue.REQUESTID == "") {
                    grunt.log.errorlns("Error invoking
BAPI_CTREQUEST_CREATE.");
                    grunt.log.errorlns("Transport request could not be
created.");
                    grunt.log.errorlns(returnValue.RETURN.MESSAGE);
                    done(false);
                    return;
                }
                grunt.log.writeln("Transport request", returnValue.REQUESTID,
"created.");
                if (fs.existsSync(targetDir) === false) {
                    fs.mkdirSync(targetDir);
                }
                fs.writeFile(ctsDataFile,
                    JSON.stringify(
                        { REQUESTID: returnValue.REQUESTID }
                    ),
                    function(err) {
                        if (err) {
                            grunt.log.errorlns("Error Creating file:", err);
                            done(false);
                            return;
                        }
                        grunt.log.writeln("Created file:", ctsDataFile);
                        done();
                    }
                )
            },
            function() {
                done(false);
            });
    });
    grunt.registerTask("uploadToABAP", "Uploads the application to the ABAP
System", function(transportRequest) {
        grunt.log.writeln("Uploading to ABAP");
        if (!transportRequest) {
            grunt.log.errorlns("No Transport request specified.");
            return (false);
        }
        grunt.log.writeln("Transport request:", transportRequest);
        var url = this.options().zipFileURL;
        var verbose = this.options().verbose;
        var failUploadOnWarning = this.options().failUploadOnWarning;
        var importParameters = {
            IV_URL: url,
            IV_SAPUI5_APPLICATION_NAME: abapApplicationName,
            IV_SAPUI5_APPLICATION_DESC: abapApplicationDesc,
            IV_PACKAGE: abapPackage,
            IV_WORKBENCH_REQUEST: transportRequest,
            IV_TEST_MODE: "-",
            IV_EXTERNAL_CODE_PAGE: this.options().codePage,
            IV_ACCEPT_UNIX_STYLE_EOL:
this.options().acceptUnixStyleLineEndings
        };
        var done = this.async();
        grunt.log.writeln("Uploading application from", url);
        rfcConnect("/UI5/REPO_LOAD_FROM_ZIP_URL", importParameters, this)
            .then(
            function(returnValue) {
                if (returnValue.EV_SUCCESS == "E" || (failUploadOnWarning !=
"false" && returnValue.EV_SUCCESS == "W")) {
                    grunt.log.errorlns("Error invoking", "/UI5/
REPO_LOAD_FROM_ZIP_URL");
                    grunt.log.writeln("Return:", returnValue);
                    done(false);
                    return;
                } else if (returnValue.EV_SUCCESS == 'S') {
                    grunt.log.writeln("Application uploaded.");
                    done();
                } else {
                    grunt.log.writeln("Invalid return status (EV_SUCCESS): "
+ returnValue.EV_SUCCESS);
                    done(false);
                    return;
                }
                if(verbose == 'true' ) {
                    grunt.log.writeln("Return:", returnValue);
                }
                grunt.log.writeln("Application uploaded.");
                done();
            },
            function() {
                done(false);
            });
    });
    grunt.registerTask("releaseTransport", "Releases an ABAP Transport
Request", function(transportRequest) {
        grunt.log.writeln("Releasing Transport Request");
        if (!transportRequest) {
            grunt.log.errorlns("No Transport request specified.");
            return (false);
        }
        grunt.log.writeln("Transport request:", transportRequest);
        var importParameters = {
            REQUESTID: transportRequest,
            COMPLETE: "X",
            BATCH_MODE: "X"
        }
        var done = this.async();
        rfcConnect("BAPI_CTREQUEST_RELEASE", importParameters, this)
            .then(
            function(returnValue) {
            if (returnValue.RETURN.TYPE == "E" || returnValue.RETURN.TYPE ==
"W") {
                    grunt.log.errorlns("Error invoking",
"BAPI_CTREQUEST_RELEASE");
                    grunt.log.writeln("Return:", returnValue);
                    done(false);
                    return;
                }
                grunt.log.writeln("Transport request released.");
                done();
            },
            function() {
                done(false);
            });
    });
};
```

This Gruntfile implements the following tasks, which are passed as parameters to the `grunt` command in the next step:

Tasks in the run-rfc-task.js

Task	Function
`createTransportRequest`	Create a transport request in the ABAP system.
`uploadToABAP`	Upload the application as ZIP file to the ABAP system.
`releaseTransport`	Release the transport request with all its transport tasks.

2. To execute the tasks from the `run-rfc-task.js`, choose one of the following options:
   - **With Docker:**
     In your shell, execute the following commands:

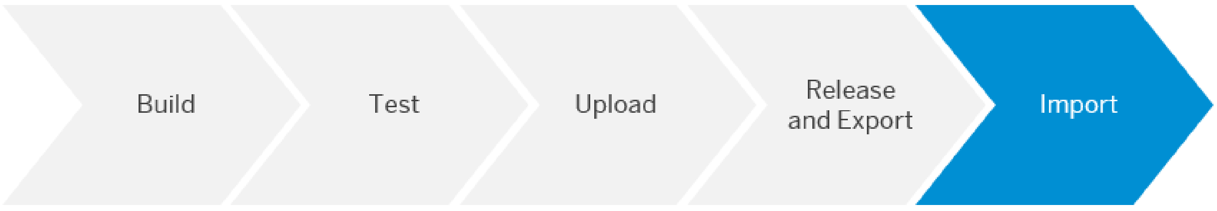
     ```
     docker run -v "${PWD}":/project node:latest npm install
     ```

```
docker run -v "${PWD}":/project node:latest node_modules/grunt-cli/bin/
grunt  --gruntfile run-rfc-task.js createTransportRequest uploadToABAP
releaseTransport
```

- **Without Docker:**
  In your shell, execute the following commands:

```
npm install
```

```
node_modules/grunt-cli/bin/grunt  --gruntfile run-rfc-task.js
createTransportRequest uploadToABAP releaseTransport
```



Use SAP Logon to manually import your transport request into the following system:

1. In SAP Logon, open the transaction `stms`.
2. Choose *Import Overview*.
3. Select your target system and choose *Display Import Queue*.
4. To import the waiting request, select it and choose *OK*.

Result

You have combined the ABAP life-cycle management with a basic CI pipeline, which you can enhance according to your needs.

i Note

This guide also provides procedures to enhance your finished CI/CD pipelines. See Procedures for CI/CD Pipelines [page 5].

2.3 Apply CI/CD to SAP HANA Extended Application Services, Advanced Model Development

Implement a CI/CD pipeline for the development of SAP HANA extended application services, advanced model applications.

→ Tip

If you use Jenkins or plan to use it, have a look at **project "Piper"** , instead.

For a full overview of the different solutions SAP provides for CI/CD, see SAP Solutions for Continuous Integration and Delivery.

Context

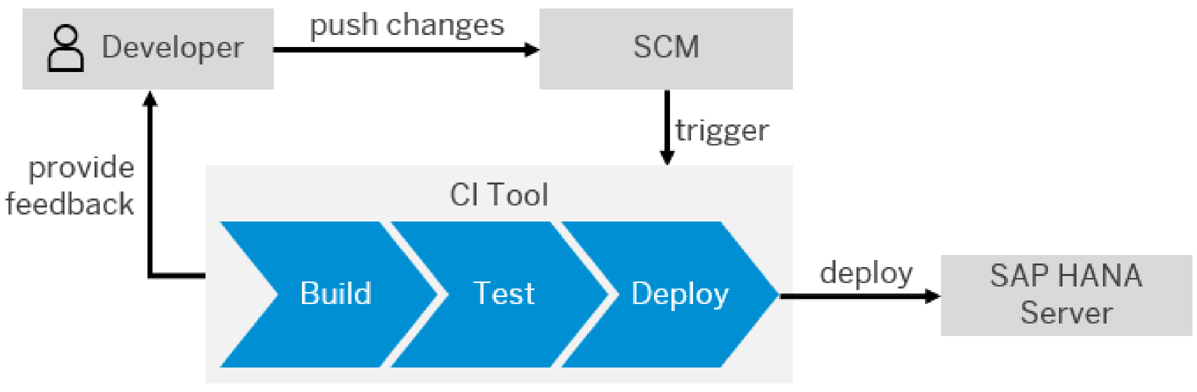
This procedure implements a CI/CD process for a multitarget application that runs on an SAP HANA extended application services, advanced model server.

An SAP HANA extended application services, advanced model application can consist of a couple of modules (for both UI and backend). To bundle and deploy these modules, the multitarget application archive format is used. A multitarget application (MTA) is a package that comprises multiple application and resource modules, which share a common lifecycle even though they might have been created with different technologies and deployed to different runtimes. To create an MTA, you bundle the modules, describe them along with their interdependencies to other modules, services, and interfaces, and package them. For more information, see The Multitarget Application Model.

A CI/CD pipeline for SAP HANA extended application services, advanced model comprises the following steps:

1. Push your code changes to a source code management (SCM) tool of your choice. The push event to the SCM system triggers the CI process.
2. In the CI build, the Cloud MTA Build Tool (MBT) triggers the technology-specific compilers for the respective modules contained in the MTA. For more information, see Cloud MTA Build Tool .
3. The Cloud MTA Build Tool packages the artifacts from each module into one archive file with the extension `.mtar`.
4. The build result is automatically deployed into an environment for automated testing during the CI build. The CI build may contain different tests, such as static code checks for the JavaScript sources (ESLint) and automated user interface tests (for example, Add Automated System Tests with the SAPUI5 Test Recorder to Your CI/CD Pipeline).
5. The MTA archive is deployed to the production environment.

The following graphic illustrates this procedure:



CI/CD in SAP HANA Extended Application Services, Advanced Model Development

Prerequisites

- As it prevents clutter on your build system and improves both maintainability and flexibility, we recommend working with Docker. Refer to the documentation of your CI tool to check whether it supports the usage of Docker containers in its pipelines.

  > ⚠ Caution
  >
  > Please check with your IT and security departments how to handle Docker images from public sources.

- If you don't want to work with Docker, make sure that you have installed the XS Advanced Command-Line Client.

In your MTA, you can have various modules. Depending on which module type you intend to use in them, choose from the following sets of additional prerequisites:

> → Tip
>
> Expand the section that fits your scenario.

SAP Fiori

- You have an SAPUI5/SAP Fiori application as module in the sources of your MTA application. For building it, use the UI5 Tooling .
- On the top level of your directory, you have a file named `mta.yaml`, which contains the following entry in the modules section:

```
modules:
  - name: webapp
    type: html5
    path: .
    parameters:
      disk-quota: 256M
      memory: 256M
    build-parameters:
      builder: npm
```

- On the top level of your directory, you have a file named `package.json`, which contains the following build script:

```
"scripts": {
  "build": "ui5 build --a",
  "test": "uiveri5"
  "linting": "eslint ."
}
```

  > i Note
  >
  > The build script executes the UI5 command-line tool. In this example, the test script runs UIVeri5. You can, however, substitute `uiveri5` with any other automated tests you have implemented.

Java

- You have a Java application as module in the sources of your MTA application.
- On the top level of your directory, you have a file named `mta.yaml` which contains the following:

Sample Code

```
modules:
  - name: webapp
    type: java
    path: .
    parameters:
      disk-quota: 256M
      memory: 1024M
    build-parameters:
      build-result: 'target/*.jar'
      builder: mvn
```

- On the top level of your Java project, you have a pom.xml, which controls the Maven build.

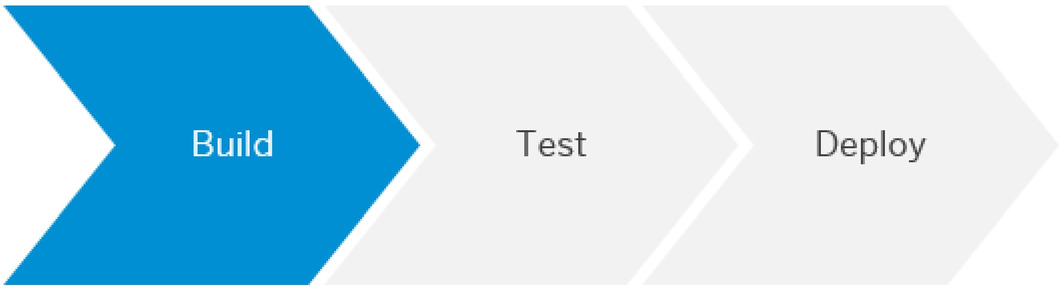
Procedure

In this procedure, we focus on the core stages of a CI/CD pipeline: build, test, and deploy.

→ Tip

You can combine and extend these simple build, test, and deploy steps to implement more complex pipelines.

1. **Build**



The build optimizes and packages your project sources. Use the Cloud MTA Build Tool (MBT) to orchestrate the technical build steps using either npm or Maven. Depending on your configuration, you can also add linting and unit tests. The build produces an out.mtar file, which is used in the following deployment stage. See step 3: **Deploy**.
For more information on how to download, set up, and run the Cloud MTA Build Tool, see Cloud MTA Build Tool .
To run the build, choose one of the following options:

   - **With Docker:**
     Execute the following command in the directory that contains your project sources:

```
docker run -v "${PWD}":/project devxci/mbtci:latest mbt build -p xsa
```

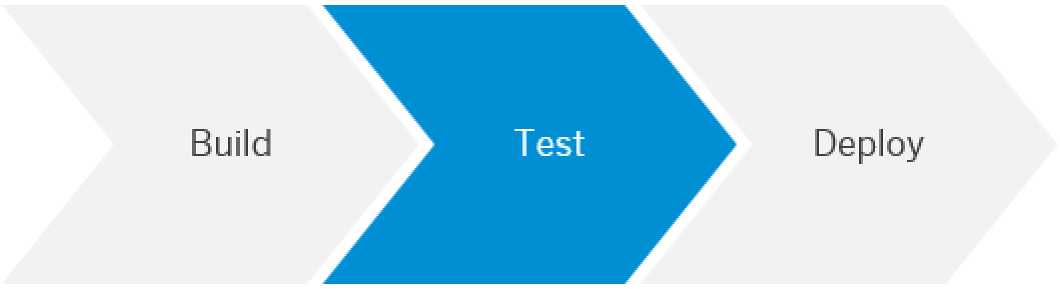
i Note

As many CI tools automatically mount your workspace that contains the project sources into your Docker container, you may omit the `-v` command, which mounts your current working directory.

- **Without Docker:**
  If you provide all dependencies on the build server, you can build your project without Docker by placing the Cloud MTA Build Tool directly on your build system. See Cloud MTA Build Tool .
  Execute the following command in the directory that contains your project sources:

  ```
  mbt build -p xsa
  ```

2. **Test**



Depending on your requirements and the setup of your project, consider adding automated tests to it. For SAPUI5/ SAP Fiori, we recommend implementing UIVeri5 and OPA5 tests. See, for example, Add Automated System Tests with the SAPUI5 Test Recorder to Your CI/CD Pipeline.
To add automated tests, choose one of the following options:

- **With Docker:**
  Use a Docker image that fits your requirements, for example, node for NodeJS.

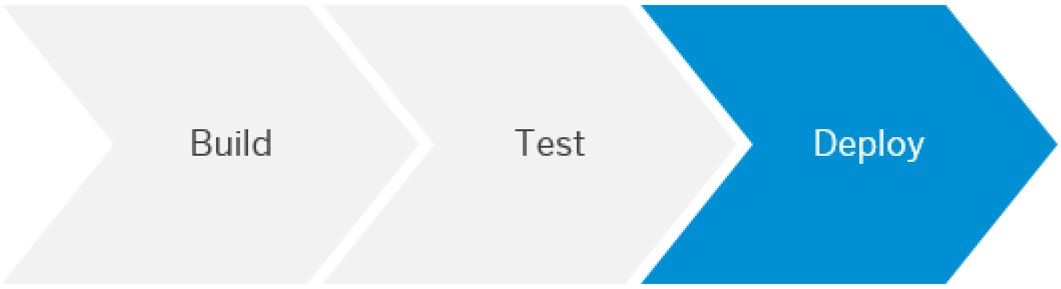
  ```
  docker run -v "${PWD}":/project node:latest <execute your build script>
  ```

  i Note

  As many CI tools automatically mount your workspace that contains the project sources into your Docker container, you may omit the -v command, which mounts your current working directory.

- **Without Docker:**
  In your shell, execute your test script.

3. **Deploy**



Depending on whether you work or do not work with Docker, use one of the following commands to deploy your application to SAP HANA.

⚠ Caution

Don't write credentials into a file but use the credential mechanism of your CI tool, instead.

- **With Docker:**
  If you want to use our XS command-line Dockerfile, you have to build it locally, first. See XS Command-Line Client Dockerfile .

Adapt the following command with your actual values and execute it:

```
docker run -v "${PWD}":/home ppiper/xs-cli:latest xs api <YOUR SAP HANA
ENDPOINT> && xs login -u <USERNAME> -p <PASSWORD> && xs push
```

- **Without Docker:**
  Adapt the following commands and in this order, execute them:
  - `xs api <YOUR SAP HANA ENDPOINT>`
  - `xs login -u <USERNAME> -p <PASSWORD>`
  - `xs bg-deploy`

Result

You have created a basic CI/CD pipeline, which you can extend according to your needs, for example, by adding additional tests and manual release steps.

> i Note
>
> This guide also provides procedures to enhance your finished CI/CD pipelines. See Procedures for CI/CD Pipelines [page 5].

2.4 Integrate SAP Cloud Transport Management into Your CI/CD Pipeline

Add an enterprise-ready change and release management process to your CI/CD pipeline and enable the transport of cloud-based applications on SAP BTP between several stages.

> → Tip
>
> If you use Jenkins or plan to use it, have a look at the **project "Piper"** scenario Integrate SAP Cloud Transport Management Into Your CI/CD Pipeline , instead.

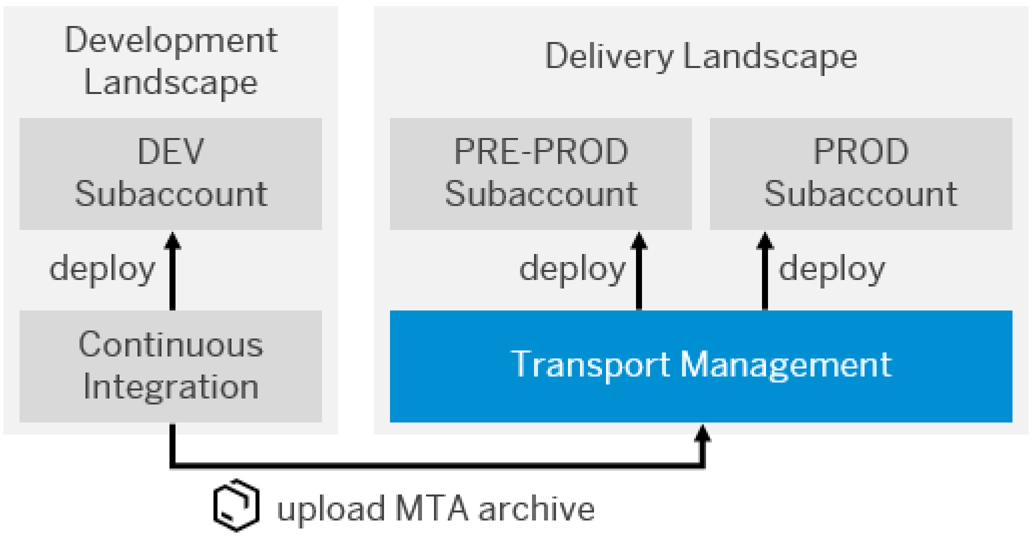
For a full overview of the different solutions SAP provides for CI/CD, see SAP Solutions for Continuous Integration and Delivery.

Context

This procedure explains how to upload a multitartget application out of a CI/CD pipeline to SAP Cloud Transport Management and then import it into its target environment.

SAP Cloud Transport Management allows you to manage the transport of development artifacts and application-specific content between different SAP BTP accounts. It adds transparency to the audit trail of changes so that you get information about who performed which changes in your production accounts and when they did it. At the same time, the Transport Management service enables a separation of concerns: For example, a developer of an application or SAP BTP content artifacts can trigger the propagation of changes, while the resulting transport is handled by a central operations team. For more information, see SAP Cloud Transport Management.

The following graphic provides an overview about the interplay between continuous integration and SAP Cloud Transport Management:



Interplay of Continuous Integration and SAP Cloud Transport Management

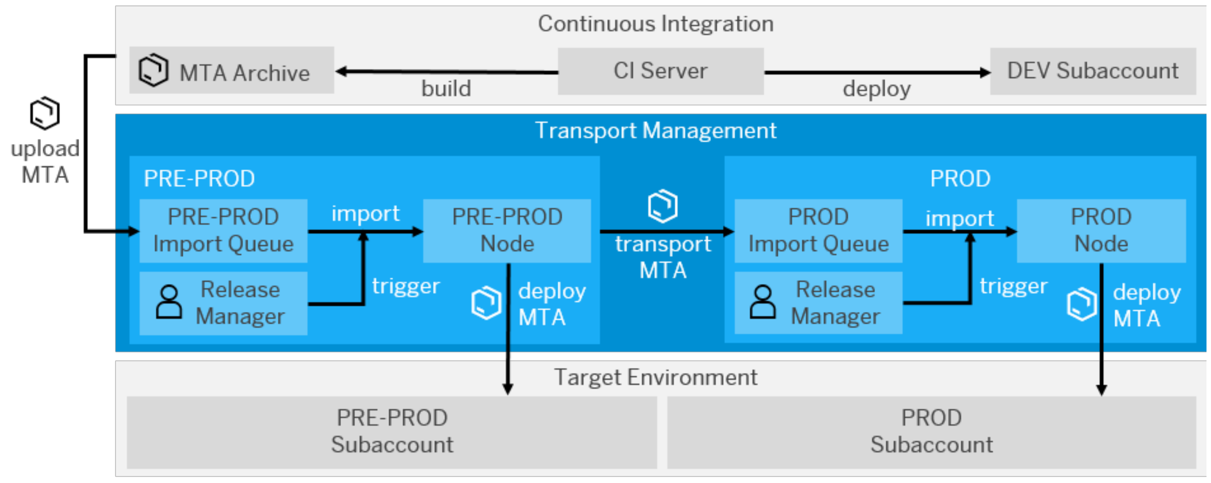
Prerequisites

- You have an existing CI pipeline, which you want to enhance with SAP Cloud Transport Management.
- You have an MTA project and the folder structure of its sources corresponds to the standard MTA structure. For more information, see The Multitarget Application Model.
- You have access to SAP Cloud Transport Management. See Provide Access to SAP Cloud Transport Management.
- You have set up SAP Cloud Transport Management and created a service key. See Set Up the Environment to Transport Content Archives directly in an Application.
- You have configured your Transport Management landscape. See Configuring the Landscape.

Procedure

This procedure belongs to the **Enhance Your Pipeline** category, which means that you can use it to enhance any CI process that meets the prerequisites, for example, the one described in Apply CI/CD to SAP Fiori Development on SAP BTP [page 6].

The following graphic shows an example of the detailed procedure when combining continuous integration and SAP Cloud Transport Management:



Detailed Procedure When Combining CI and SAP Cloud Transport Management

The process flow contains the following steps:

1. The CI server builds a multitarget application (MTA) archive.
2. The MTA is uploaded into the import queue of the target node, which is specified in the CI pipeline (in this example, PRE-PROD).
3. The release manager manually triggers or schedules the import, which results in the physical deployment of the MTA archive into the corresponding subaccount (in this example, PRE-PROD).
4. As soon as the import is executed, a transport is triggered along the defined transport route so that the MTA archive reaches the import queue of the next node (in this example, PROD).
5. There, the physical import into the corresponding subaccount can be either triggered manually by the release manager or automatically by using the scheduling mechanisms of SAP Cloud Transport Management.

To enhance your exisiting CI/CD pipeline with SAP Cloud Transport Management, execute the following tasks:

1. **Upload**

   1. Copy the SAP Cloud Transport Management service key and store it in the secret store of your CI server.

2. To authenticate yourself against Transport Management, adapt and use this API call:

API Call for Authenticating against Transport Management

HTTP Method	`POST`
URL	`https://<serviceKey.uaa.url>/oauth/token/?grant_type=client_credentials&response_type=token` Use a JSON parser to extract the value `uaa.url` from the service key.
Headers	○ **Key**: `Username` **Value**: `<serviceKey.uaa.clientid>` ○ **Key**: `Password` **Value**: `<serviceKey.uaa.clientsecret>` Use a JSON parser to extract the values `uaa.clientid` and `uaa.clientsecret` from the service key.
Response	You get an **access token**, which is needed for following Transport Management API calls.

3. To upload your MTA to SAP Cloud Transport Management, adapt and use this API call:

API Call for Uploading an MTA to Transport Management

HTTP Method	`POST`
URL	`https://<serviceKey.uri>/v2/files/upload` Use a JSON parser to extract the value `uri` from the service key.
Header	**Key**: `Authorization` **Value**: `bearer <access_token>`
Body	○ **Key**: `File` **Value**: `<file with application content>` ○ **Key**: `namedUser` **Value**: `<id of the user performing the request>`
Response	You get a **file ID**, which is needed for further Transport Management API calls.

4. To upload the MTA archive to a transport node, adapt and use this API call:

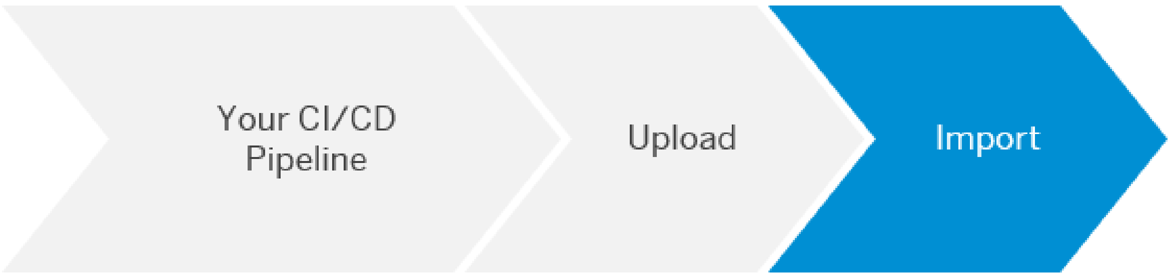
API Call for Uploading an MTA Archive to a Transport Node

HTTP Method	`POST`
URL	`https://<serviceKey.uri>/v2/nodes/upload`
Header	**Key**: Authorization **Value**: `bearer <access_token>`
Body	Sample Code

Sample Code

```
{
  "nodeName": "PRE-PROD",
  "contentType": "MTA",
  "description": "<your
description>",
  "storageType": "FILE",
  "namedUser": "<id of the user
performing the request>",
  "entries": [
    {
       "uri": "<fileId>"
    }
  ]
}
```

2. **Import**



After successfully uploading your MTA archive to SAP Cloud Transport Management, you can use the import mechanism of Transport Management to distribute it within your transport landscape. See Using the Import Queue.
Choose between the following options:
- To manually import your artifacts, see Import Transport Requests.
- To schedule automated imports, see Schedule Imports.

Result

You have enhanced your CI/CD pipeline with SAP Cloud Transport Management.

i Note

Now, you can also add change request management with SAP Soluiton Manager to your pipeline. See Integrate Change Request Management with SAP Solution Manager into Your CI/CD Pipeline [page 38].

For an overview of all procedures in this guide, see Procedures for CI/CD Pipelines [page 5].

2.5 Integrate Change Control Management with SAP Solution Manager into Your CI/CD Pipeline

Add change control management processes to your CI/CD pipeline and facilitate the synchronization of changes made both on-premise and on SAP BTP.

→ Tip

If you use Jenkins or plan to use it, have a look at the **project "Piper"** scenario Build and Deploy Hybrid Applications with Jenkins and SAP Solution Manager , instead.

For a full overview of the different solutions SAP provides for CI/CD, see SAP Solutions for Continuous Integration and Delivery.

Context

Change control management features help you create change requests, edit them, if necessary, and implement changes. You can choose between two different options, which you can integrate into your CI/CD pipeline:

- **Quality Gate Management**
  Quality Gate Management is an out-of-the-box solution that helps you monitor all software change processes, distribute software across systems and technology stacks, and get an overview of the implementation of changes to your SAP software solution. For more information, see Quality Gate Management.
- **Change Request Management**
  Change Request Management allows you to manage your projects in SAP Solution Manager from end to end: Plan your change management and project, manage your resources, and physically transport changes from the development environment into the productive one. Change Request Management with SAP Solution Manager is highly customizable. For more information, see Change Request Management.

Depending on which option you want to use, choose one of the following scenarios:

- **Integrate Quality Gate Management with SAP Solution Manager into Your CI/CD Pipeline** [page 36]
- **Integrate Change Request Management with SAP Solution Manager into Your CI/CD Pipeline** [page 38]

2.5.1 Integrate Quality Gate Management with SAP Solution Manager into Your CI/CD Pipeline

Add change management processes that are compliant with the Information Technology Infrastructure Library (ITIL) to your CI/CD pipeline and facilitate the synchronization of changes made both on-premise and on SAP BTP.

→ Tip

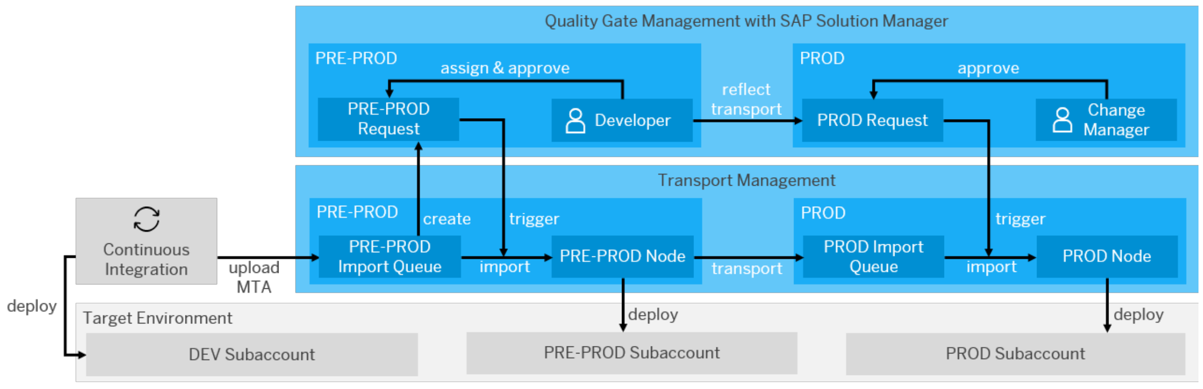
If you use Jenkins or plan to use it, have a look at the **project "Piper"** scenario Build and Deploy Hybrid Applications with Jenkins and SAP Solution Manager , instead.

For a full overview of the different solutions SAP provides for CI/CD, see SAP Solutions for Continuous Integration and Delivery.

Context

This scenario explains how to add Quality Gate Management to your CI/CD pipeline with an SAP Cloud Transport Management integration. For more information about Quality Gate Management with SAP Solution Manager, see Quality Gate Management.

The following graphic provides an overview about the interplay between continuous integration, SAP Cloud Transport Management, and Quality Gate Management:



Detailed Procedure When Combining CI and Quality Gate Management

The interplay comprises the following steps:

1. As a result of the continuous intgeration build, a multitarget application (MTA) is created, uploaded to SAP Cloud Transport Management, and attached to a new transport request in the import queue of the PRE-PROD node.
2. The Transport Management transport requests are visible in SAP Solution Manager.
3. The developer assigns the request to a change document.
4. The developer approves the change document, which in the transport management service, triggers the import of the MTA into the PRE-PROD node.
5. Through the import, the deployment of the MTA to the PRE-PROD subaccount is triggered.

6. At the same time, the import forwards the transport request with the MTA to the PROD import queue in SAP Cloud Transport Management.
7. This transport is also reflected in SAP Solution Manager, where the change document is transported to the PROD requests.
8. The change manager approves the change document for production, which in the transport management service, triggers the import of the MTA into the PROD node.
9. The import triggers the deployment of the MTA to the PROD subaccount.

Prerequisites

- You have enhanced your CI/CD pipeline with SAP Cloud Transport Management, as described in Integrate SAP Cloud Transport Management into Your CI/CD Pipeline [page 30].
- You have access to an SAP Solution Manager system in version 7.2 SP10 or higher.

Procedure

1. From Setting up SAP BTP TMS for Change Control Management, execute the tasks in the following sections:
    - Set-up Steps in the Change Control Management
    - Set-up Steps in the Landscape Management Database (LMDB)
    - Set-up Steps in the Solution Administration (SLAN)
2. To prevent manual imports into the nodes that you want to be controlled by Quality Gate Management, set the flag *Controlled by SAP Solution Manager* in their configurations in SAP Cloud Transport Management.

Result

SAP Cloud Transport Management is now visible in the landscape view of SAP Solution Manager. If you use your continuous delivery pipeline to upload an MTA (or another content archive) to the transport management service, the content is now attached to the import queue of the first node that is controlled by SAP Cloud Transport Management.

In SAP Solution Manager, you can now search for transport requests that are waiting in the import queue and assign them to the correct change request in the change control management. Whenever in a change request, the change reaches an appropriate state (for example, *To be tested*), you can trigger its import into the node.

2.5.2 Integrate Change Request Management with SAP Solution Manager into Your CI/CD Pipeline

Add change management processes that are compliant with the Information Technology Infrastructure Library (ITIL) to your CI/CD pipeline and facilitate the synchronization of changes made both on-premise and on SAP BTP.

→ Tip

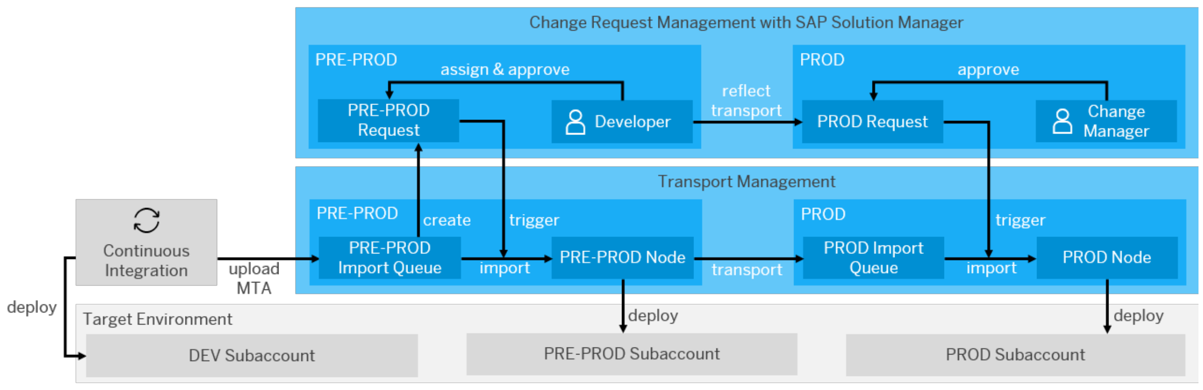
If you use Jenkins or plan to use it, have a look at the **project "Piper"** scenario Build and Deploy Hybrid Applications with Jenkins and SAP Solution Manager , instead.

For a full overview of the different solutions SAP provides for CI/CD, see SAP Solutions for Continuous Integration and Delivery.

Context

This scenario explains how to add Change Request Management to your CI/CD pipeline with an SAP Cloud Transport Management integration. For more information about Change Request Management with SAP Solution Manager, see Change Request Management.

The following graphic provides an overview about the interplay between continuous integration, SAP Cloud Transport Management, and Change Request Management:



Detailed Procedure When Combining CI and Change Request Management

The interplay comprises the following steps:

1. As a result of the continuous intgeration build, a multitarget application (MTA) is created, uploaded to SAP Cloud Transport Management, and attached to a new transport request in the import queue of the PRE-PROD node.
2. The Transport Management transport requests are visible in SAP Solution Manager.
3. The developer assigns the request to a change document.
4. The developer approves the change document, which in the transport management service, triggers the import of the MTA into the PRE-PROD node.
5. Through the import, the deployment of the MTA to the PRE-PROD subaccount is triggered.

6. At the same time, the import forwards the transport request with the MTA to the PROD import queue in SAP Cloud Transport Management.
7. This transport is also reflected in SAP Solution Manager, where the change document is transported to the PROD requests.
8. The change manager approves the change document for production, which in the transport management service, triggers the import of the MTA into the PROD node.
9. The import triggers the deployment of the MTA to the PROD subaccount.

Prerequisites

- You have enhanced your CI/CD pipeline with SAP Cloud Transport Management, as described in Integrate SAP Cloud Transport Management into Your CI/CD Pipeline [page 30].
- You have access to an SAP Solution Manager system in version 7.2 SP10 or higher.
- You have configured Change Request Management with SAP Solution Manager, as described in Configuring Change Request Management.

Procedure

1. From Setting up SAP BTP TMS for Change Control Management, execute the tasks in the following sections:
    - Set-up Steps in the Change Control Management
    - Set-up Steps in the Landscape Management Database (LMDB)
    - Set-up Steps in the Solution Administration (SLAN)
2. To prevent manual imports into the nodes that you want to be controlled by Change Request Management, set the flag *Controlled by SAP Solution Manager* in their configurations in SAP Cloud Transport Management.

Result

SAP Cloud Transport Management is now visible in the landscape view of SAP Solution Manager. If you use your continuous delivery pipeline to upload an MTA (or another content archive) to the transport management service, the content is now attached to the import queue of the first node that is controlled by SAP Cloud Transport Management.

In SAP Solution Manager, you can now search for transport requests that are waiting in the import queue and assign them to the correct change request in the change control management. Whenever in a change request, the change reaches an appropriate state (for example, *To be tested*), you can trigger its import into the node.

Important Disclaimers and Legal Information

Hyperlinks

Some links are classified by an icon and/or a mouseover text. These links provide additional information.
About the icons:

- Links with the icon : You are entering a Web site that is not hosted by SAP. By using such links, you agree (unless expressly stated otherwise in your agreements with SAP) to this:
  - The content of the linked-to site is not SAP documentation. You may not infer any product claims against SAP based on this information.
  - SAP does not agree or disagree with the content on the linked-to site, nor does SAP warrant the availability and correctness. SAP shall not be liable for any damages caused by the use of such content unless damages have been caused by SAP's gross negligence or willful misconduct.
- Links with the icon : You are leaving the documentation for that particular SAP product or service and are entering a SAP-hosted Web site. By using such links, you agree that (unless expressly stated otherwise in your agreements with SAP) you may not infer any product claims against SAP based on this information.

Videos Hosted on External Platforms

Some videos may point to third-party video hosting platforms. SAP cannot guarantee the future availability of videos stored on these platforms. Furthermore, any advertisements or other content hosted on these platforms (for example, suggested videos or by navigating to other videos hosted on the same site), are not within the control or responsibility of SAP.

Beta and Other Experimental Features

Experimental features are not part of the officially delivered scope that SAP guarantees for future releases. This means that experimental features may be changed by SAP at any time for any reason without notice. Experimental features are not for productive use. You may not demonstrate, test, examine, evaluate or otherwise use the experimental features in a live operating environment or with data that has not been sufficiently backed up.
The purpose of experimental features is to get feedback early on, allowing customers and partners to influence the future product accordingly. By providing your feedback (e.g. in the SAP Community), you accept that intellectual property rights of the contributions or derivative works shall remain the exclusive property of SAP.

Example Code

Any software coding and/or code snippets are examples. They are not for productive use. The example code is only intended to better explain and visualize the syntax and phrasing rules. SAP does not warrant the correctness and completeness of the example code. SAP shall not be liable for errors or damages caused by the use of example code unless damages have been caused by SAP's gross negligence or willful misconduct.

Bias-Free Language

SAP supports a culture of diversity and inclusion. Whenever possible, we use unbiased language in our documentation to refer to people of all cultures, ethnicities, genders, and abilities.

www.sap.com/contactsap

THE BEST RUN SAP