Deploy to Cloud Foundry

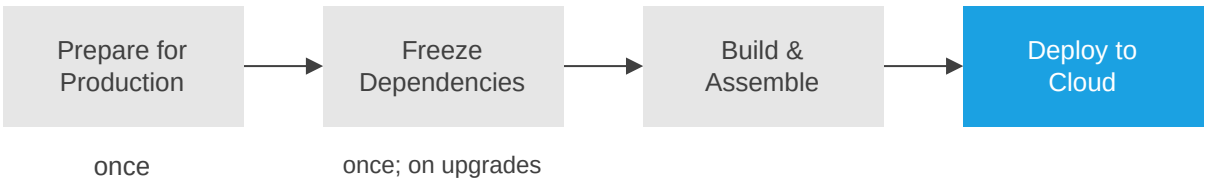
A comprehensive guide on deploying applications built with SAP Cloud Application Programming Model (CAP) to SAP BTP Cloud Foundry environment.

Table of Contents

Intro & Overview

After completing the functional implementation of your CAP application by following the Getting Started or Cookbook guides, you would finally deploy it to the cloud for

production. The essential steps are illustrated in the following graphic:



First, you apply these steps manually in an ad-hoc deployment, as described in this guide. Then, after successful deployment, you automate them using CI/CD pipelines.

▶ *This guide is available for Node.js and Java.*

Prerequisites

The following sections are based on a new project that you can create like this:

```sh
cds init bookshop --add sample
cd bookshop
```

▶ *Alternatively, download or clone the sample repository*

In addition, you need to prepare the following:

1. SAP BTP with SAP HANA Cloud Database Up and Running

- Access to SAP BTP, for example a trial
- An SAP HANA Cloud database running in your subaccount
- Entitlement for `hdi-shared` service plan for your subaccount
- A Cloud Foundry space

> **Starting the SAP HANA database takes several minutes**

> Therefore, we recommend doing these steps early on. In trial accounts, you need to start the database **every day**.

2. Latest Versions of *@sap/cds-dk*

Ensure you have the latest version of *@sap/cds-dk* installed globally:

```sh
npm -g outdated       #> check whether @sap/cds-dk is listed
npm i -g @sap/cds-dk  #> if necessary
```

Likewise, ensure the latest version of *@sap/cds* is installed in your project:

```sh
npm outdated          #> check whether @sap/cds is listed
npm i @sap/cds        #> if necessary
```

3. Cloud MTA Build Tool

- Run *mbt* in a terminal to check whether you've installed it.
- If not, install it according to the MTA Build Tool's documentation .
- For macOS/Linux machines best is to install using *npm* :

  ```sh
  npm i -g mbt
  ```

- For Windows, please also install *GNU Make* .

4. Cloud Foundry CLI w/ MTA Plugins

- Run *cf -v* in a terminal to check whether you've installed version **8** or higher.
- If not, install or update it according to the Cloud Foundry CLI documentation .
- In addition, ensure to have the MTA plugin for the Cloud Foundry CLI installed.

  ```sh
  cf add-plugin-repo CF-Community https://plugins.cloudfoundry.org
  cf install-plugin -f multiapps
  cf install-plugin -f html5-plugin
  ```

Prepare for Production

If you followed CAP's grow-as-you-go approach, you've developed your application with an in-memory database and basic (mocked) authentication. In the cloud, you typically use production-grade services like SAP HANA and authentication providers.

The `cds add <facets>` command ensures required services are configured correctly and their dependencies are added to your *package.json*.

1. SAP HANA Database

While we used SQLite as a low-cost stand-in during development, we're using an SAP HANA Cloud database for production:

```sh
cds add hana
```

↳ *Learn more about using SAP HANA for production.*

2. Authorization/Authentication

Configure your app for XSUAA-based authentication:

```sh
cds add xsuaa
```

> **This will also generate an `xs-security.json` file**
>
> The roles/scopes are derived from authorization-related annotations in your CDS models. Ensure to rerun `cds compile --to xsuaa`, as documented in the *Authorization* guide whenever there are changes to these annotations.

↳ *Learn more about SAP Authorization and Trust Management/XSUAA.*

3. MTA-Based Deployment

We'll be using the Cloud MTA Build Tool to execute the deployment. The modules and services are configured in an `mta.yaml` deployment descriptor file, which we generate

with:

```sh
cds add mta
```

↳ *Learn more about MTA-based deployment.*

4. App Router as Gateway

The *App Router* acts as a single point-of-entry gateway to route requests to. In particular, it ensures user login and authentication in combination with XSUAA.

Two deployment options are available:

- **Managed App Router**: for SAP Build Work Zone, the Managed App Router provided by SAP Fiori Launchpad is available.
- **Custom App Router**: for custom scenarios without SAP Fiori Launchpad, the App Router needs to be deployed along with your application. In this case, use the following command to enhance the application configuration:

  ```sh
  cds add approuter
  ```

↳ *Learn more about the SAP BTP Application Router.*

5. User Interfaces

Option A: SAP Cloud Portal

If you intend to deploy **multi-tenant** user interface applications, you also need to set up the HTML5 Application Repository in combination with the SAP Cloud Portal service :

```sh
cds add portal
```

Option B: SAP BTP Application Frontend beta

For **single-tenant** applications, you can use the new SAP BTP Application Frontend service:

```sh
cds add app-front
```

6. Optional: Multitenancy

To enable multitenancy for production, run the following command:

```sh
cds add multitenancy
```

> **You're set!**
>
> The previous steps are required *only once* in a project's lifetime. With that done, we can repeatedly deploy the application.

Build and Deploy

Make sure you are logged in to Cloud Foundry and target the space you want to deploy to:

```sh
cf login --sso  # to log on with SAP Universal ID
cf target
```

↳ *Learn more about* `cf login`

If your project already includes a *package-lock.json*, freeze your updated dependencies:

```sh
npm install --package-lock-only
```

You can now build and deploy the application:

```sh
cds up
```

▶ *Essentially, this automates the following steps...*

▶ *Test with* `cds build`

↳ *Got errors? See the troubleshooting guide.*

↳ *Learn how to reduce the MTA archive size **during development**.*

This process can take some minutes and finally logs an output like this:

```log
[…]
Application "bookshop" started and available at
"[org]-[space]-bookshop.<landscape-domain>.com"
[…]
```

You can use this URL to access the App Router as the entry point of your application.

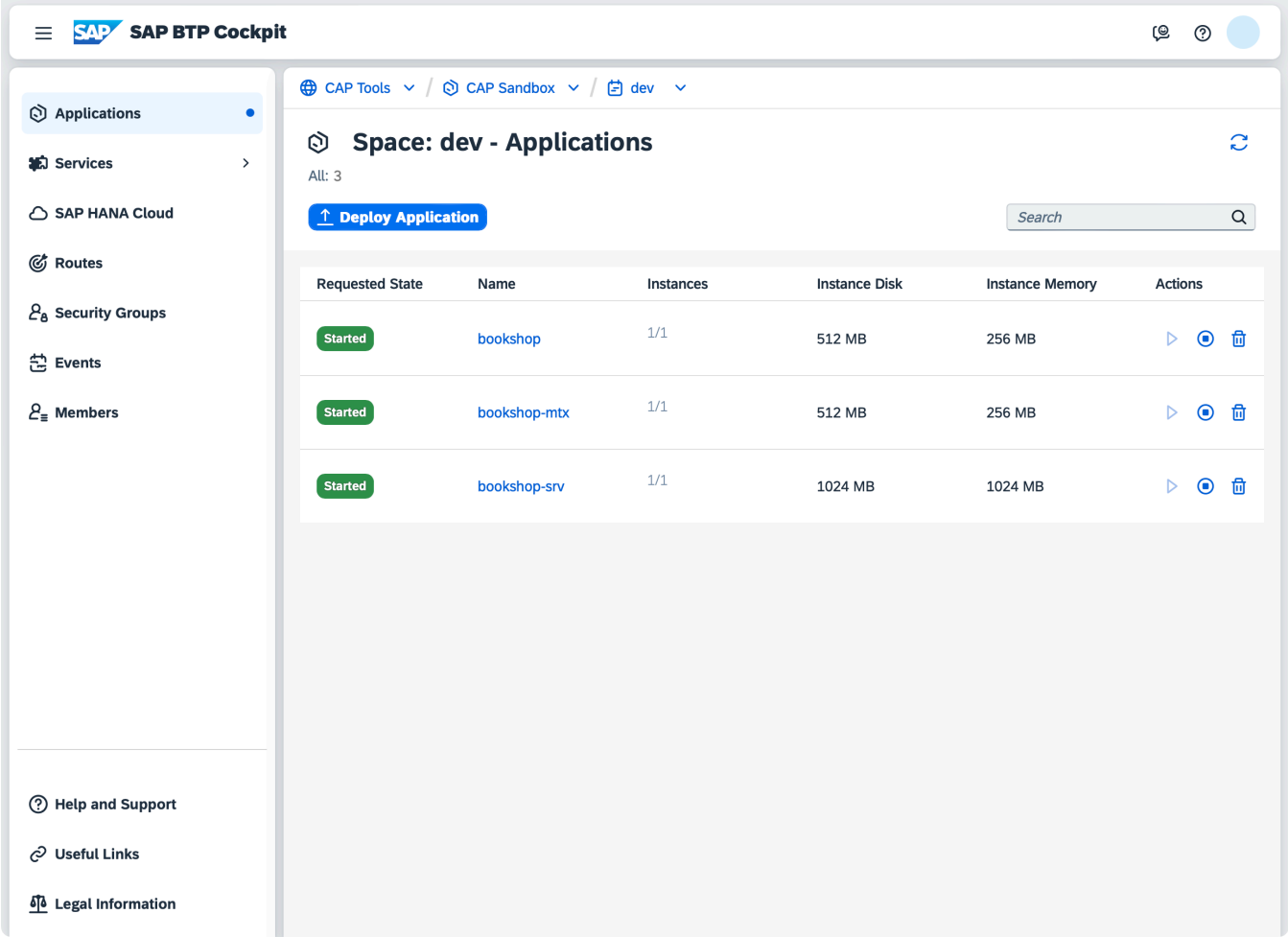
For **multitenant applications**, you have to subscribe a tenant first. The application is accessible via a tenant-specific App Router URL after subscription.

> **SaaS Extensibility**
>
> Share the generic App-Router URL with SaaS consumers for logging in as extension developers using `cds login` or other extensibility-related commands.

> **No index page and SAP Fiori preview in the cloud**
>
> The default index page and SAP Fiori preview, that you are used to seeing during local development, are only meant for the development profile and not available in the cloud. For productive applications, you should add a proper SAP Fiori elements application through on of the user interface options outlined before.

Inspect Apps in BTP Cockpit

Visit the "Applications" section in your SAP BTP cockpit to see the deployed apps:



> **Next up: Assign the *admin* role**
>
> In order to access the admin APIs you need to assign the *admin* role required by `AdminService`. Create a role collection and assign the role and your user to get access.

↳ *Got errors? See the troubleshooting guide.*

Keep Dependencies Up-to-date

Deployed applications should freeze all their dependencies, including transient ones. Therefore, on first execution, `cds up` creates a *package-lock.json* file for all application modules.

It is **essential to regularly update dependencies** to consume latest bug fixes and improvements. Not doing so will increase the risk of **security vulnerabilities**, expose your

application to **known bugs**, and make future upgrades significantly harder and more time-consuming.

We recommend setting up Dependabot , Renovate or similar automated solutions to update dependencies **one-by-one** to easily identify breaking changes, minimize risks, and ensure continuous compatibility and **stability of your application**.

Next Up...

You would then set up your CI/CD for automating deployments, for example after merging pull requests.