Set up a local development environment and initial development project

When you set up and configure an Adobe Experience Manager Forms as a Cloud Service environment, you set up development, staging, and production environments on cloud. In addition, you can also set up and configure a local development environment.

You can use the local development environment to create forms and related assets (themes, templates, custom Submit Actions, and more) and convert PDF forms to Adaptive Forms without logging in to cloud development environment. After an Adaptive Form or related assets are ready on the local development instance, you can export the Adaptive Form and related assets from the local development environment to a Cloud Service environment for further testing and publishing.

You can also develop and test custom code like custom components and prefill service on the local development environment. When the custom code is tested and ready, you can use the Git repository of your Cloud Service development environment to deploy the custom code.

To set up a new local development environment and use it to develop for activities, perform the following actions in listed order:

- Set up development tools
- Setup local Author and Publish instances
- Add Forms archive to local development instances and configure users
- Set up local development environment for microservices
- Set up a development project
- Set up local Dispatcher tools

Prerequisites

You require the following software to set up a local development environment. Download these before starting to setup the local development environment:

Software	Description	Download links
Adobe Experience Manager Forms as a Cloud Service SDK	SDK includes Adobe Experience Manager QuickStart and Dispatcher tools	Download from Software Distribution
Adobe Experience Manager Formsfeature archive	Tools to create, style, and optimize Adaptive Forms and other Adobe Experience Manager Formsfeatures	Download from Software Distribution

Set up development tools for AEM Projects

The Adobe Experience Manager Forms project is a custom code base. It contains code, configurations, and content that is deployed via Cloud Manager to Adobe Experience Manager as a Cloud Service. The AEM Project Maven Archetype provides the baseline structure for the project.

Setup the following development tools to use for your Adobe Experience Manager project for development:

- Java™
- Git
- Node.js (npm)
- Maven

For detailed instructions to set up previously mentioned development tools, see Set up development tools.

Set up local Experience Manager environment for development

The Cloud Service SDK provides a QuickStart file. It runs a local version of Experience Manager. You can run either the Author or Publish instances locally.

While the QuickStart provides a local development experience, it does not have all features available in Adobe Experience Manager as a Cloud Service. So, always test your features and code with Adobe Experience Manager as a Cloud Service development environment before moving the features to stage or production.

To install and configure local Experience Manager environment, perform the following steps:

- Download and extract the Adobe Experience Manager as a Cloud Service SDK
- Set up an Author instance
- Set up a Publish instance

Add Forms archive to local Author and Publish instances and configure Forms-specific users

Perform the following steps in the listed order to add Forms archive to Experience Manager instances and configure forms-specific users:

Install the latest Forms add-on feature archive

Adobe Experience Manager Forms as a Cloud Service feature archive provides tools to create, style, and optimize Adaptive Forms on the local development environment. Install the package to create an Adaptive Form and use various other features of AEM Forms. To install the package:

1. Download and extract the latest AEM Forms archive for your operating system from Software Distribution.
2. Navigate to the crx-quickstart/install directory. If the folder does not exist, create it.
3. Stop your AEM instance, place the AEM Forms add-on feature archive, `aem-forms-addon-<version>.far`, in the install folder, and restart the instance.

Configure users and permissions

Create users like Form Developer and Form Practitioner and add these users to pre-defined forms groups to provide them required permissions. The table below lists all types of users and pre-defined groups for each type of forms users:

User Type	AEM Group
Form practitioner /	forms-users (AEM Forms Users), template-authors, workflow-users, workflow-editors, and fdm-authors
Form developer	forms-users (AEM Forms Users), template-authors, workflow-users, workflow-editors, and fdm-authors
Customer Experience Lead or UX designer	forms-users, template-authors
AEM administrator	aem-administrators, fd-administrators
End user	When a user must login to view and submit an Adaptive Form, add such users to forms-users group. When no user authentication is required to access Adaptive Forms, do not assign any group to such users.

Set up local development environment for microservices

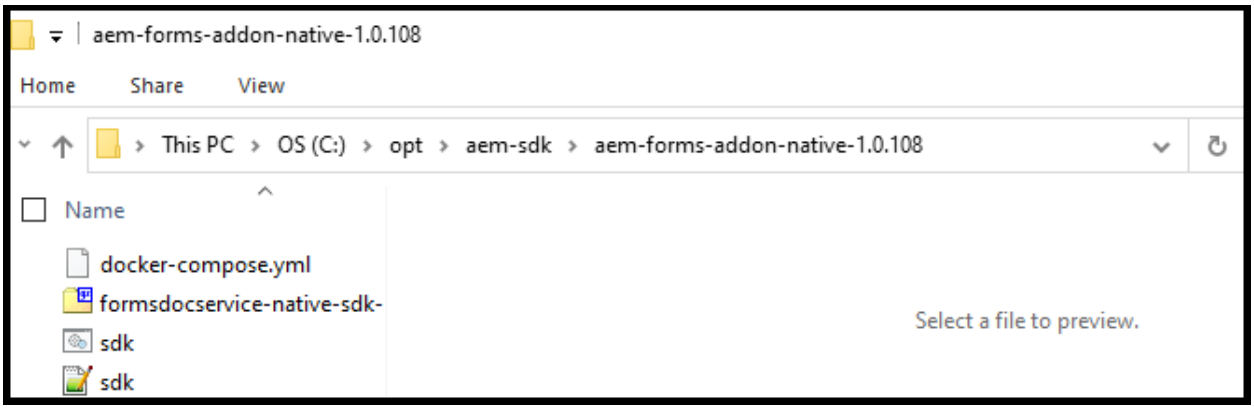
AEM Forms as a Cloud Services provides a docker-based SDK environment for easier development of Document of Record and for using other microservices. It frees you from manually configuring platform specific binaries and adaptations. To setup the environment:

1. Install Docker Desktop. It configures Docker Engine and docker-compose on your machine.

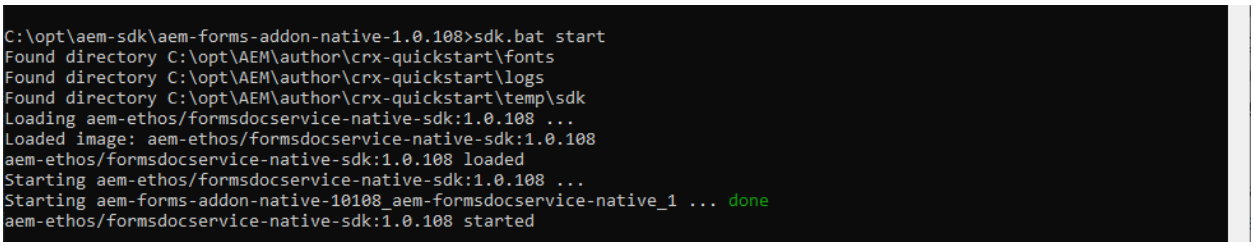
   NOTE

   - Docker for Mac whitelists a few folders (and their subfolders) for bind mounting. Expliciyly configure mounting a folder outside these well known defaults.
   - Docker Desktop for Windows supports two backends, Hyper-V (legacy) and WSL2 (modern). File sharing is automatically managed by Docker when using WSL2 (modern). You have to explicitly configure file sharing while using Hyper-V (legacy).

2. Create a folder, say aem-sdk, in parallel to your author and publish instances. For example /opt/aem-sdk.
3. Extract the `aem-forms-addon-<version>.zip\aem-forms-addon-native-<version>.zip` file.



4. Create an environment variable AEM_HOME and point to local AEM Author installation. For example opt/aem/author.
5. Open sdk.bat or sdk.sh for editing. Set the AEM_HOME to point to local AEM Author installation. For example opt/aem/author.
6. Open command prompt and navigate to the `aem-forms-addon-native-<version>` folder.
7. Run the following command to start the SDK:
    - (on Microsoft Windows) sdk.bat start
    - (on Linux or Apple Mac OS) sdk.sh start



You can now use the local development envionment to render Document of Record. To test, upload an XDP file to your environment and render it. for example, http://localhost:4502/libs/xfaforms/profiles/default.print.pdf?template=crx:///content/dam/formsanddocuments/cheque-request.xdp converts the XDP file to the PDF document.

Set up a development project for Forms based on Experience Manager archetype

Use this project to create Adaptive Forms, deploy configuration updates, overlays, create custom Adaptive Form components, test, and custom code on local Experience Manager Forms SDK.

After testing locally, you can deploy the project to Experience Manager Forms as a Cloud Service production and non-production environments. To set up the project:

1. **Clone Cloud Manager Git repository on your local development instance:** Your Cloud Manager Git repository contains a default AEM project. It is based on AEM Archetype. Clone your Cloud Manager Git Repository using Self-Service Git Account Management from Cloud Manager UI to bring the project on your local development environment. For details to accessing the repository, see Accessing Git.

1. **Create an Experience Manager Forms as a [Cloud Service] project:** Create an Experience Manager Forms as a [Cloud Service] project based on AEM Archetype 27 or later. The archetype help developers easily start developing for AEM Forms as a Cloud Service. It also includes some sample themes and templates to help you started quickly.

   Open the command prompt and run the below command to create an Experience Manager Forms as a Cloud Service project. To include Forms specific configurations, themes, and templates, set `includeForms=y`.

   ```
   mvn -B archetype:generate -DarchetypeGroupId=com.adobe.aem -
   DarchetypeArtifactId=aem-project-archetype -DarchetypeVersion=27 -
   DaemVersion="cloud" -DappTitle="My Site" -DappId="mysite" -
   DgroupId="com.mysite" -DincludeForms="y"
   ```

1. Also, change `appTitle`, `appId`, and `groupId`, in the above command to reflect your environment.
2. Deploy the project to your local development environment. You can use the following command to deploy to your local development environment

   ```
   mvn -PautoInstallPackage clean install
   ```

   For the complete list of commands, see Building and Installing

3. Deploy the code to your AEM Forms as a Cloud Service environment.

Set up local Dispatcher tools

Dispatcher is an Apache HTTP Web server module that provides a security and performance layer between the CDN and AEM Publish tier. Dispatcher is an integral part of the overall Experience Manager architecture and should be part of local development environment.

Perform the following steps to configure local Dispatcher and then add Forms-specific rules to it:

Set up local Dispatcher

The Experience Manager as a Cloud Service SDK includes the recommended Dispatcher Tools version, that facilitates configuring, validating, and simulating Dispatcher locally. Dispatcher

Tools are Docker-based and provide command-line tools to transpile Apache HTTP Web Server and Dispatcher configuration files into a compatible format and deploy them to Dispatcher running in the Docker container.

Caching on Dispatcher allows AEM Forms to prefill Adaptive Forms at a client. It improves rendering speed of prefilled forms.

For detailed instructions to set up Dispatcher, see Set up local Dispatcher tools

Add Forms specific rules to Dispatcher

Perform the following steps to configure Dispatcher cache for Experience Manager Forms as a Cloud Service:

1. Open your AEM Project and navigate to `\src\conf.dispatcher.d\available_farms`
2. Create a copy of the `default.farm` file. For example, `forms.farm`.
3. Open the newly created `forms.farm` file for editing and replace the following code:

```
#/ignoreUrlParams {
#/0001 { /glob "*" /type "deny" }
#/0002 { /glob "q" /type "allow" }
#}
```

with

```
/ignoreUrlParams {
/0001 { /glob "*" /type "deny" }
/0002 { /glob "dataRef" /type "allow" }
}
```

3.
4. Save and close your file.
5. Go to `conf.d/enabled_farms` and create a symbolic link to the `forms.farm` file.
6. Compile and deploy the project to your AEM Forms as a Cloud Service environment.

Considerations about caching

- Dispatcher caching allows AEM Forms to prefill Adaptive Forms at a client. It improves rendering speed of prefilled forms.
- Caching secured content features is disabled, by default. To enable the feature, you can perform the instructions provided in the Caching Secured Content article
- The Dispatcher can fail to invalidate some Adaptive Forms and related Adaptive Forms. To resolve such issues, see AEM Forms Caching in troubleshooting section.
- Caching localized Adaptive Forms:
    - Use URL format `http://host:port/content/forms/af/<afName>.<locale>.html` to request a localized version of an Adaptive Form instead of `http://host:port/content/forms/af/afName.html?afAcceptLang=<locale>`

    - Browser Locale option is disabled, by default. To change browser locale setting,
- When you use URL Format `http://host:port/content/forms/af/<adaptivefName>.html`, and Use Browser Locale in configuration manager is disabled, the non-localized version of the Adaptive Form is served. The non-localized language is the language used while developing the Adaptive Form. The locale configured for your browser (browser locale) is not considered and a non-localized version of the Adaptive Form is served.
- When you use URL Format `http://host:port/content/forms/af/<adaptivefName>.html`, and Use Browser Locale in configuration manager is enabled, a localized version of the Adaptive Form is served, if available. The language of the localized Adaptive Form is based on the locale configured for your browser (browser locale). It can lead to [caching only first instance of an Adaptive Form]. To prevent the issue from happening on your instance, see only first instance of an Adaptive Form is cached in troubleshooting section.

Your local development environment is ready.

Upgrade your local development environment

Upgrading the SDK to a new version requires replacing the entire local development environment, resulting in a loss of all code, configuration, and content in the local repositories. Ensure that any code, config, or content that should not be destroyed is safely committed to Git, or exported from the local Experience Manager instances as CRX-Packages.

How to avoid content loss when upgrading the SDK

Upgrading the SDK is effectively creating a brand new Author and Publish instances, including a new repository (Set up AEM project), meaning any changes made to a prior SDK's repository are lost. For viable strategies for aiding in persisting content between SDK upgrades, see How to avoid content loss when upgrading the AEM SDK

Back up and import Forms-specific content to a new SDK environment

To back up and move assets from existing SDK to a new SDK environment:

- Create a backup of your existing content.
- Set up a fresh SDK environment.
- Import the backup to your new SDK environment.

Create a backup of your existing content

Back up your Adaptive Forms, templates, form data model, theme, configurations, and custom code. You can perform the following action to create backup:

1. Download Adaptive Forms, themes, and PDF forms.

2. Export Adaptive Form templates.
3. Download Form Data Models
4. Export editable templates, cloud configurations, and workflow model. To export all the previously mentioned items from your existing SDK, create an CRX-Package with the following filters:
    - /conf/ReferenceEditableTemplates
    - /conf/global/settings/cloudconfigs
    - /conf/global/settings/wcm
    - /var/workflow/models
    - /conf/global/settings/workflow
5. Export email configurations, submit, and prefill actions code from your local development environment. To export these settings and configuration, create a copy of the following folders and files on your local development environment:
    - `[Archetype Project in Cloud Service Git]/core/src/main/java/com/<program name>/core/service`
    - `[Archetype Project in Cloud Service Git] /core/src/main/java/com/<program name>/core/servlets/FileAttachmentServlet.java`
    - `[Archetype Project in Cloud Service Git]/ui.apps/src/main/content/jcr_root/apps/<program name>/config`

Import the backup to your new SDK environment

Import Adaptive Forms, templates, form data model, theme, configurations, and custom code to your fresh environment. You can perform the following action to import backup:

1. Import Adaptive Forms, themes, and PDF forms to new SDK environments.
2. Import Adaptive Form templates to new SDK environment.
3. Upload Form Data Models to new SDK environment.
4. Import editable templates, cloud configurations, and workflow model. To import all the previously mentioned items yo your new SDK environment, import the CRX-Package containing these items to your new SDK environment.
5. Import email configurations, submit, and prefill actions code from your local development environment. To import these settings and configuration, place the following files from your old Archetype project to your new Archetype project:
    - `[Archetype Project in Cloud Service Git]/core/src/main/java/com/<program name>/core/service`
    - `[Archetype Project in Cloud Service Git] /core/src/main/java/com/<program name>/core/servlets/FileAttachmentServlet.java`
    - `[Archetype Project in Cloud Service Git]/ui.apps/src/main/content/jcr_root/apps/<program name>/config`

Your new environment now has forms and related assets of old environment.