ACTIVITY GUIDE

Adobe Experience Manager Technical Basics

Adobe Digital Learning Services

Technical Basics

07192024

Contents

Adobe Experience Manager Architecture

Introduction

Adobe Experience Manager (AEM) is a modern application for experience management that accelerates the delivery of omnichannel personalization throughout the customer journey. It optimizes marketer and developer workflows for the entire content lifecycle informed by data insights.

In order to use AEM capabilities effectively, you should first understand the architecture of AEM.

AEM Applications

AEM provides an easy-to-use solution to create, manage, publish, and update complex digital forms while integrating with back-end processes, business rules, and data.

Sites

AEM Sites provides the digital foundation you need to swiftly create, manage, and deliver personalized, engaging content to every customer who visits your site. An intuitive drag-and-drop interface, out-of-the-box components, and an easy-to-use template editor help marketers deliver content quickly with minimal effort.

Forms

AEM Forms combine form authoring, management, and publishing along with correspondence management capabilities, document security, and integrated analytics to create engaging end-to-end experiences. Designed to work across web and mobile channels, AEM Forms can be efficiently integrated into your business processes, reducing paper processes and errors while improving efficiency.

Assets

AEM Assets is an application on the AEM Platform that allows our customers to manage their digital assets (images, videos, documents and audio clips) in a web-based repository. AEM Assets includes metadata support, Renditions, the Digital Asset Management Finder and the AEM Assets Administration UI.

Cloud Manager

Cloud Manager enables organizations to self-manage Experience Manager in the cloud. It includes a continuous integration and continuous delivery (CI/CD) framework that lets IT teams and implementation partners expedite the delivery of customizations or updates without compromising performance or security.

AEM Architecture

AEM is a web-based client-server system, made up of several infrastructure-level and application-level functions. These functions are used to build relevant applications.

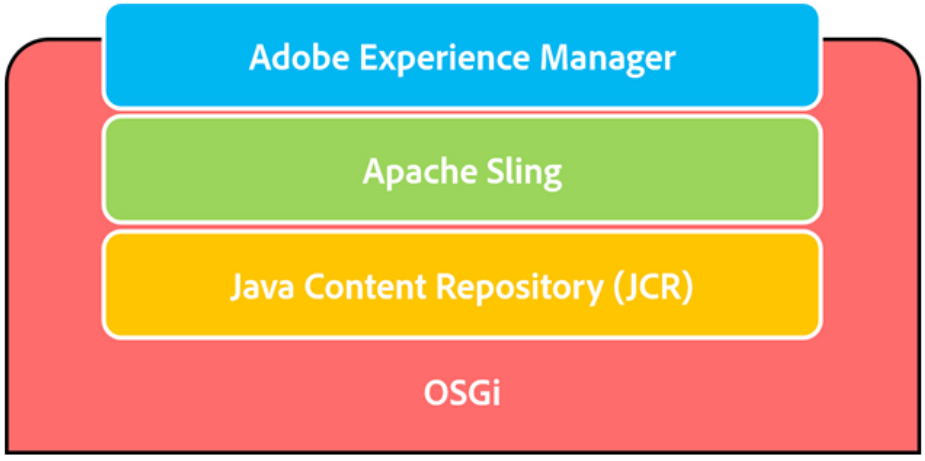
To learn more about AEM Architecture, go to:

https://docs.adobe.com/content/help/en/experience-manager-cloud-service/core-concepts/architecture.html.

Basics of AEM Architecture Stack

AEM architecture stack is based on technologies such as OSGi, Java Content Repository (JCR), and Apache Sling.

The following diagram depicts a high-level view of the AEM architecture stack:



Granite Platform

Granite is a general-purpose platform for building robust scalable applications. It is Adobe's open web stack and forms the technical foundation on which AEM is built. Granite supports an open architecture, which is based on both open standards (JCR and OSGi) and open source projects (Apache Sling and Apache Jackrabbit).

Note: Granite is an open development project within Adobe but not an open source project.

Technically, at the core, Granite provides:

- **An application launcher** for a standalone Java or Web application archive for deployment in the existing servlet containers or application servers.
- An **OSGi Framework** into which all applications are deployed.
- **OSGi Compendium Services** to support building applications, such as Log Service, Http Service, Event Admin Service, Configuration Admin Service, Declarative Services, and Metatype Service.
- A comprehensive **Logging Framework** providing various logging APIs, such as SLF4J, Log4F, Apache Commons Logging, and OSGi Log Service.
- A repository based on **Apache Jackrabbit Oak** and **JSR-283**.
- The **Apache Sling Web Framework**.

Granite UI

The consoles in the main navigation, tools, and editors of AEM are built using the Granite UI. The Granite UI provides a foundational UI framework for:

- UI widgets
- Extensible and plugin-based admin UI

It also adheres to the following requirements:

- Mobile first (designing an online experience for mobile before designing it for the desktop)
- Extensible
- Easy to override

Note: The Granite UI is based on Coral 3, which is Adobe's universal UI across all products.

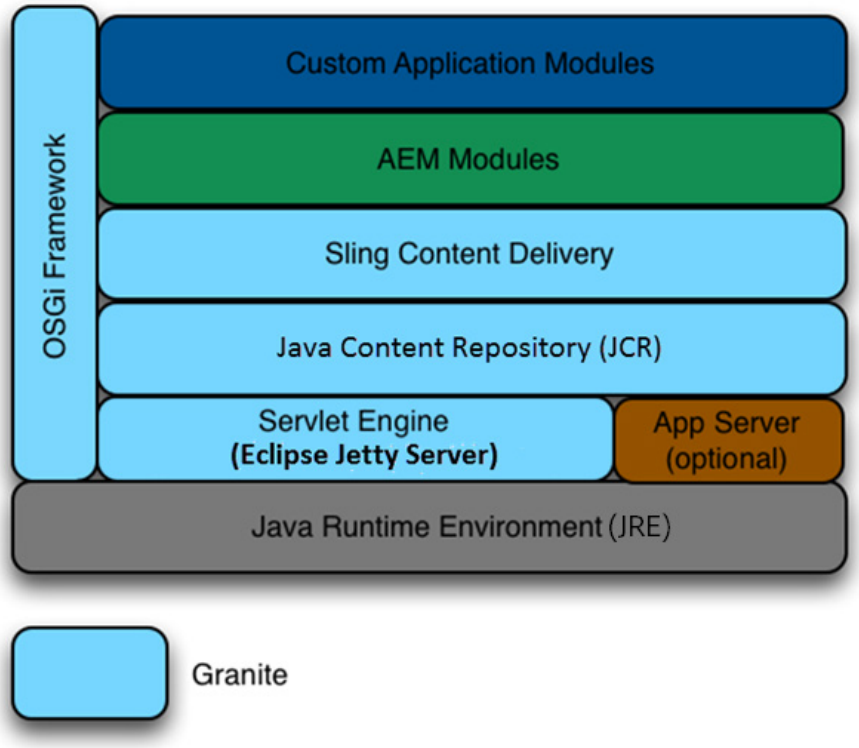
OSGi Framework

OSGi enables a collaborative and modular environment, where each application may be built and implemented as a small bundle. Each bundle is a collection of tightly coupled, dynamically loadable classes, JAR files, and configuration files that explicitly declare their external dependencies.

Note: OSGi used to stand for Open Service Gateway Initiative, but that name has been discontinued, and it is now officially no longer an abbreviation. It is just known in the industry as OSGi.

All content is stored in the content repository, which means backup is done at the repository level. OSGi runtime hosts Java applications that can access the repository by using the JCR API. As part of the application runtime, you get Apache Sling, a RESTful web application framework that exposes the full repository content using HTTP and other protocols.

The following diagram depicts the AEM platform foundation of Granite and the OSGi framework:



Apache Felix

Apache Felix is an open source implementation of OSGi for the AEM framework. It provides a dynamic runtime environment, where the code and content bundles can be loaded, unloaded, and reconfigured at runtime.

Note: You can learn more about OSGi, by visiting the link in the **References** section at the end of this module.

JCR

The JCR, specifically Java Specification Request-283 (JSR- 283), is a database that supports structured and unstructured content, versioning, and observation. In other words, it is a database that resembles a file system.

Note: The Adobe implementation of JSR-283 was known as the Content Repository eXtreme (CRX). Hence, you may see CRX in some tools and interfaces in AEM. However, the CRX as a feature is being phased out. In its place, AEM uses the Granite platform and Apache Jackrabbit Oak.

All data pertaining to AEM, such as HTML, HTML Template Language (HTL), CSS, JavaScript/Java, images, and videos are stored in the JCR object database. JCR is built with Apache Jackrabbit Oak, an open-source project.

The advantages of using JCR are:

- It provides a generic application data store for structured and unstructured content. While file systems provide excellent storage for unstructured and hierarchical content, the databases provide storage for structured data. This way, JCR provides the best of both the data storage architectures.
- It supports namespaces. Namespaces prevent naming collisions among items and node types that come from different sources and application domains. JCR namespaces are defined with a prefix, delimited by a single colon (:). For example, jcr:title. This means that this title property is defined in the jcr namespace.
- It provides one interface to interact with text and binary data. This helps with easy access and management of data in comparison to storing it in multiple places.

Note: You can learn more about JCR, by visiting the link in the **References** section at the end of this module.

Apache Sling

Apache Sling is a web application framework for content-centric applications and uses a JCR, such as Apache Jackrabbit Oak, to store and manage content.

The key features of Apache Sling include:

- It is Apache open source.
- It is based on REST principles and helps build applications as a series of OSGi bundles.
- It is resource-oriented (every resource has a URI) and maps to JCR nodes.

A request URL is first resolved to a resource, and then based on the resource, Apache Sling selects the Servlet or script to handle that request. Servlets and scripts are handled as resources and are accessible by a resource path. This means every script, Servlet, filter, and error handler is available from the Resource Resolver just like normal content—providing data to be rendered on request.

Note: You can learn more about Apache Sling, by visiting the link in the **References** section at the end of this module.

Infrastructure of AEM

When it comes to underlying infrastructure for AEM, there are three different solutions available, Cloud Service, Managed Services, and On-premise.

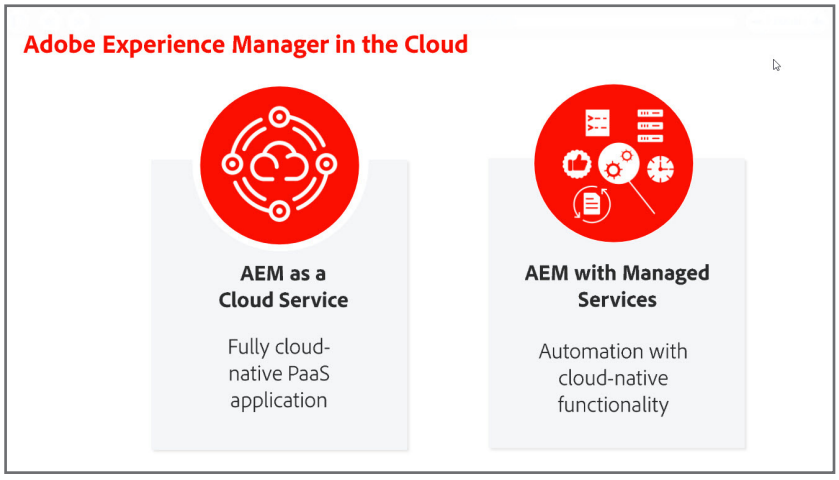
AEM as a Cloud Service

AEM as a Cloud Service is the cloud-native way of leveraging the AEM applications.
It enables you to provide your customers with personalized, content-led experiences, by combining the power of the AEM Content Management System with AEM Digital Asset Management. The solution has been entirely designed for the cloud and is scalable, secure, always available and up-to-date.

AEM as a Cloud Service has resulted in changes to the architecture. AEM as a Cloud Service now has a dynamic architecture with a variable number of AEM images.

The architecture of AEM as a Cloud Service:

- Is scaled based on the actual traffic and actual activity.
- Has individual instances that only run when needed.
- Uses modular applications.
- Has an author cluster as default; this avoids downtime for maintenance tasks.



Managed Services

AEM Managed Services is a complete solution for Digital Experience management. It provides benefits of experience delivery solution in the cloud while retaining all the control, security, and customization benefits of an on-premise deployment. AEM Managed Services enables customers to launch faster by deploying on the cloud and also by leaning on the best practices and support from Adobe. Organizations and business users can engage customers in minimal time, drive market share, and focus on creating innovative marketing campaigns while reducing the burden on IT.

AEM On-premise

AEM deployed and managed in your corporate environment. You can install AEM on servers in your Corporate environment. Typical installation instances include: Development, Testing, and Publishing environments.

References

Use the following links for more information on Architecture:

- AEM as a Cloud Service Architecture: https://docs.adobe.com/content/help/en/experience-manager-cloud-service/core-concepts/architecture.html
- OSGi: http://www.osgi.org
- JCR 2.0 Specification: http://jackrabbit.apache.org/jcr/jcr-api.html
- JSR-283: https://jcp.org/en/jsr/detail?id=283
- Apache Sling: https://sling.apache.org/
- Apache Felix: https://felix.apache.org/

Start Adobe Experience Manager Locally

Introduction

Development on Adobe Experience Manager (AEM) typically occurs on a local author service with the quickstart jar. Typical AEM deployments consist of two services: an author service and a publish service. These services are identical in terms of installed software, but an author service is for internal content creation and a publish service is for public content delivery. To develop for AEM, you must install a local author service and, optionally, a publish service.

Activity 1: Start Your AEM Instance

As an AEM developer or administrator, you will need to work on your AEM project locally. To work locally, you will need to start a local AEM instance that was created from an AEM quickstart JAR file.

Note: ReadyTech has AEM preinstalled for your convenience. You can view the installations under:
- **C:/adobe/aem-6.5/** for AEM 6.5
- **C:/adobe/aem-sdk/** for AEM Cloud Service

Prerequisite: AEM installed on your machine.

This activity includes the following tasks:

1. Start your AEM instance (Cloud Service only).
2. Start your AEM instance (6.5 only).
3. Verify author is running.

Task 1: Start Your AEM Instance (Cloud Service Only)

Note: If you are not using ReadyTech, then you need to install AEM first. Installation instructions can be found in the activities under "Local Install -Additional Learning section".

1. Open a command prompt and navigate to **C:\adobe (**if you are not already in c:\adobe):

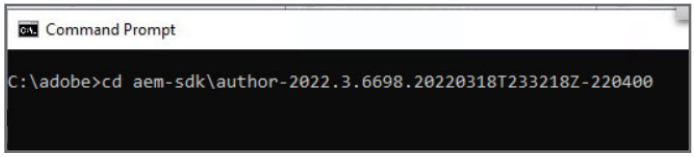
   ```
   cd C:\adobe
   ```

2. Navigate to the AEM installation folder.

a. `$ cd aem-sdk\author-`

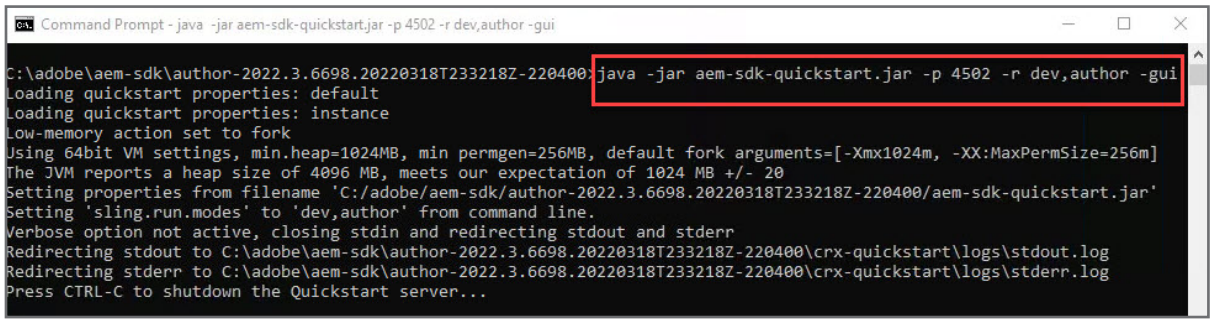
b. Press <TAB>. This will auto-complete the **author-<NNNN>** folder name for you. Press <Enter> to navigate to the folder.

c. `$ cd aem-sdk\author-<nnnn>`



3. Start the AEM instance using the Cloud Service start command:

```
$ java -jar aem-sdk-quickstart.jar -r author,dev -p 4502 -gui
```



Note: In the above command:

-p specifies the port number for the local aem quickstart

-r specifies the run mode that you are running locally

-gui opens the GUI window on your desktop

If no port number is specified, the first available port from the following list is used:
1) 4502, 2) 8080, 3) 8081, 4) 8082, 5) 8083, 6) 8084, or a random port.

Task 2: Start Your AEM Instance (6.5 Only)

1. Open a command prompt and navigate to **C:\adobe (**if you are not already in c:\adobe):

   ```
   cd C:\adobe
   ```

2. Navigate to the AEM installation folder.
   a. `$ cd aem-6.5\author-6.5.19`

3. Start the AEM instance using the AEM 6.5 start command:

   ```
   $ java -jar aem-65-quickstart.jar -r author,dev -p 4502 -gui
   ```

Note: In the above command:

-p specifies the port number for the local aem quickstart
-r specifies the run mode that you are running locally
-gui opens the GUI window on your desktop

If no port number is specified, the first available port from the following list is used:
1) 4502, 2) 8080, 3) 8081, 4) 8082, 5) 8083, 6) 8084, or a random port.

Task 3: Verify Author Is Running

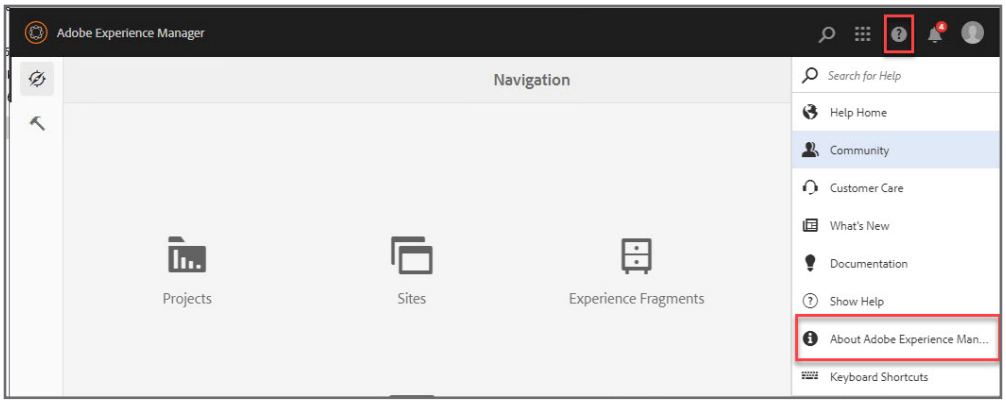
1. Verify your author service has started. After the author service starts successfully, the start-up screen (the GUI) changes to something similar to the following:

   In addition, after AEM starts, your default browser automatically opens to the start URL (where the port number is the one you defined on installation). For example: http://localhost:4502 .

   A sign-in screen is displayed.

2. Enter your user name and password and click **Sign In**. If you are using a ReadyTech machine or local installation, use the following credentials to sign in:
   a. User name: **admin**
   b. Password: **admin**

3. Go back to http://localhost:4502/aem/start.html and in the top right corner, click **?** > **About Adobe Experience Manager**, as shown:



4. Verify you have the right version of Adobe Experience Manager. For example:

Local Install

Additional Learning

Important: You can skip this Additional Learning section if you want to follow the Fast Track approach. This Additional Learning section and Activities are optional.

Installing the Quickstart Jar

Install the quickstart jar using the command-line method. The installation takes approximately 5-7 minutes the first time, depending on your system's capabilities.
When you use the command-line method to install and start AEM, you can provide additional performance-tuning parameters to the Java Virtual Machine (JVM) and perform other administrative tasks. On Windows, MacOS X or *x, you can increase the Java heap size during the installation, which improves performance.

Prior to installation, you may want to know which parameters are available to configure quickstart. Type the following command in the command prompt to display a complete list of optional parameters:

```
java -jar aem-sdk-quickstart.jar -h
```

The quickstart installer shows all the available command-line options without starting the server.

Some command-line options:

-debug <port>
Enable Java Debugging on port number; forces forking

-gui
Show GUI if running on a terminal

-nobrowser (-quickstart.nobrowser)
Do not open browser at startup

-unpack
Unpack installation files only, do not start the server (implies -verbose)

-v (-verbose)
Do not redirect stdout/stderr to files and do not close stdin

-nofork

Do not fork the JVM, even if not running on a console

-fork

Force forking the JVM if running on a console, using recommended default memory settings for the forked JVM

Activity 2: Install the SDK Quickstart Jar

As a developer or administrator, you need to work with Adobe Experience Manager Cloud service locally using the SDK quickstart JAR file. In this activity, you will install and start an author dev service on port 4502. Optionally, if you would like to start a publish instance, go through this entire activity again and replace "author" with "publish".

Prerequisites:

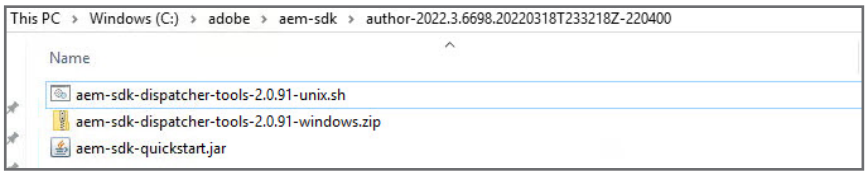
- Java 11 installed
- Latest AEM SDK downloaded: https://downloads.experiencecloud.adobe.com/content/software-distribution/en/aemcloud.html.

Note: To use this URL, your organization needs to be provisioned with Cloud Service and your IMS user needs to be added to the product profile in the Admin console. If you do not have access to this location, your instructor will provide the SDK.

To install AEM SDK on your local machine:

1. Locate the aem-sdk-<nnnn>.zip file and extract it into a folder of your choice. For example: C:/adobe/aem-sdk/
2. Rename the folder **aem-sdk-<nnnn>** to **author-<nnnn>**.
3. Open the renamed **author-<nnnn>** folder to find the quickstart jar **aem-sdk-quickstart-<nnnn>.jar**.

4. Rename the file **aem-sdk-quickstart-<nnnn>.jar** to **aem-sdk-quickstart.jar.** For simplicity, remove the build number from the jar file.
   - aem = Application
   - sdk-quickstart = Local Cloud quickstart
   - 2023.12.14697.20231215T125030Z-231200 = the build of this SDK

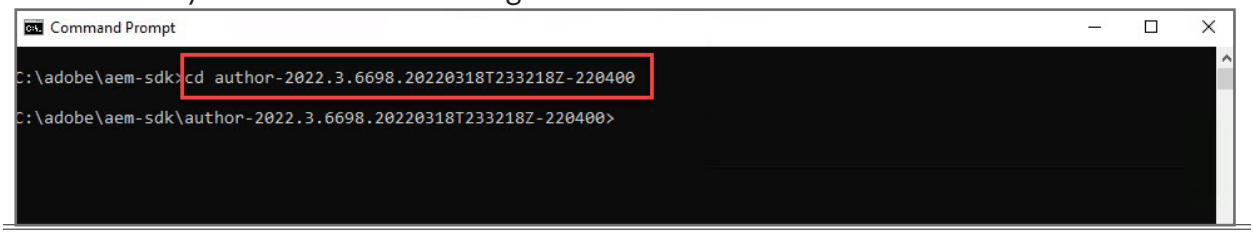


Note: Change the jar file name for convenience only. Optionally you could keep the build number in the file name. Keeping the build number helps quickly understand what build you are currently using. In this training, we keep the build number in the parent folder.

5. Open a command prompt and navigate to the directory on your machine where the SDK exists:

```
cd C:\adobe\aem-sdk
```

6. Start typing **cd author** and press <TAB>. This will auto-complete the **author-<NNNN>** folder name for you. Press <Enter> to navigate to the folder.

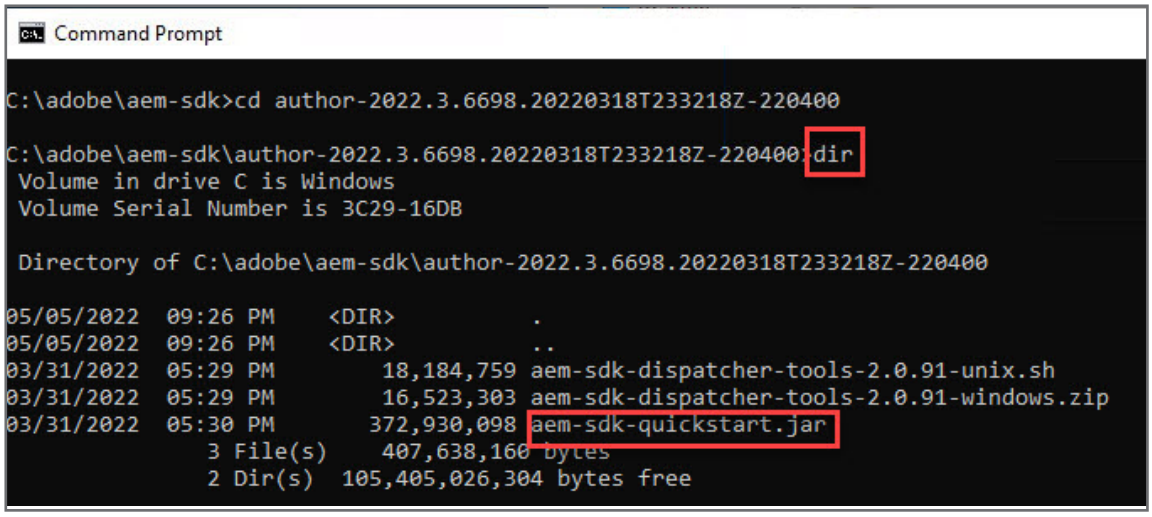


Note: If you are using a ReadyTech Service, this directory should be **C:\adobe\aem-sdk\author-<nnnn>** where nnnn is the build number.

7. Execute the following command to verify you are in the correct directory where the SDK quickstart jar resides.

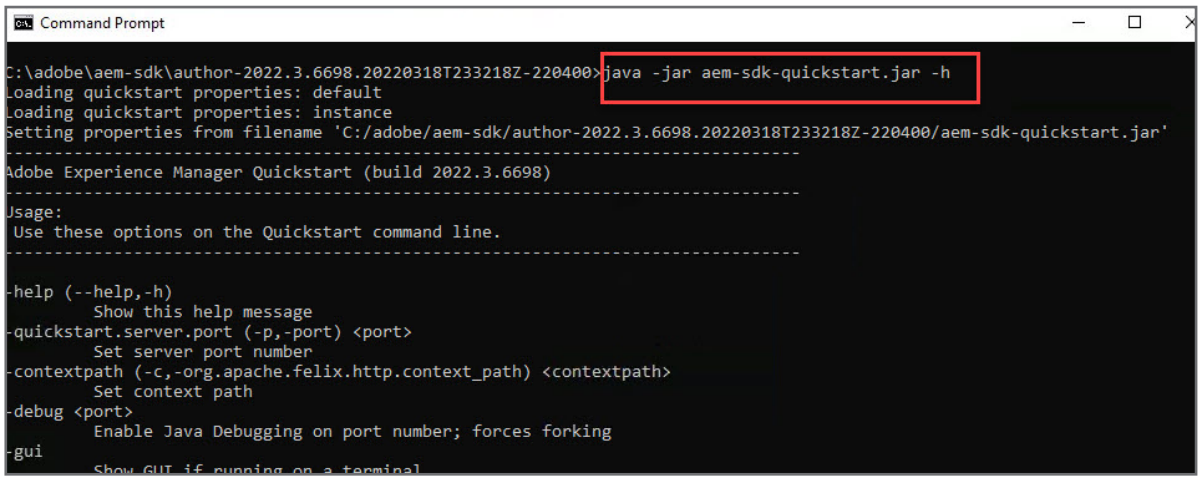
   Windows: **dir**

   Mac: **ls**



8. In the command line, run the following command to see the different parameters that you can use to run the quickstart:

```
java -jar aem-sdk-quickstart.jar -h
```



9. To install and start your author service, use the following command:

```
java -jar aem-sdk-quickstart.jar -p 4502 -r author,dev -gui
```

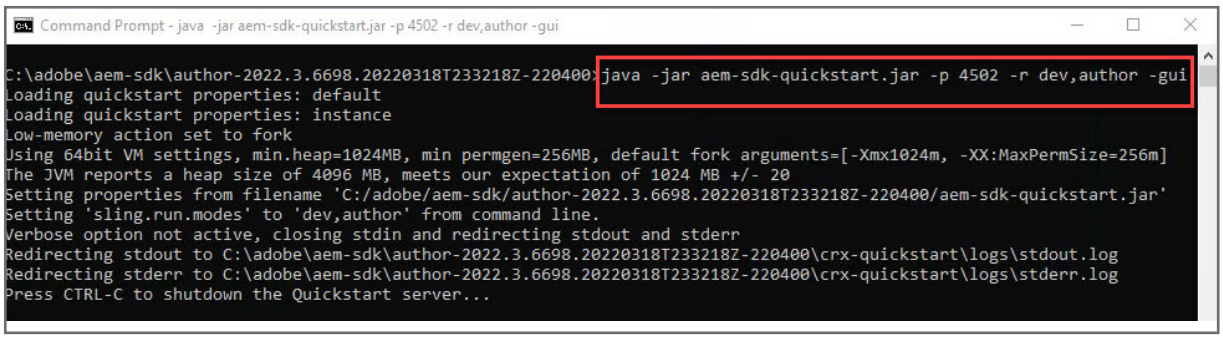
Note: In the above command:

-p specifies the port number for the local AEM quickstart

-r specifies the run mode that you are running locally

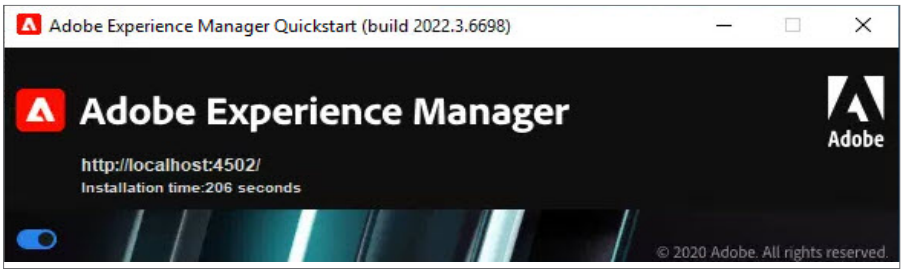
-gui opens the GUI window on your desktop

If no port number is specified, the first available port from the following list is used:
1) 4502, 2) 8080, 3) 8081, 4) 8082, 5) 8083, 6) 8084, or a random port.



Tip: Be patient, as it may take up to two minutes for the installation GUI window to appear.

10. Verify your author service has started. The command window shows startup details and the GUI window shows progress. Installation takes approximately 5–7 minutes depending on your system's capabilities. After the author service has started successfully, the start-up screen (the GUI) changes to something similar to the following:



You have now completed the local install of Adobe Experience Manager as a Cloud Service.

Note: To stop the service, click the **ON/OFF** toggle icon at the lower left corner of the server GUI.

11. Each time you want to run your author service, use the same command in step 9 to start it. However, after the first time, the startup will be as fast as one minute or less, as the initial installation task has been performed.

    In addition, after AEM starts, your default browser automatically opens to the start URL (where the port number is the one you defined on installation). For example: http://localhost:4502

    A sign-in screen is displayed as shown here:

12. Enter your user name and password, and click **Sign In**. If you are using a ReadyTech machine or local installation, use the following credentials to sign in:
    a. User name: **admin**
    b. Password: **admin**
13. Instead of using the GUI to stop your author service, use the command window you used to initially launch the quickstart. For Windows, type **CTRL+C** in the command window to stop your author service.

Note: If you would like to view server logs, see "Activity 4: Observe the Local Log Files" on page 36.

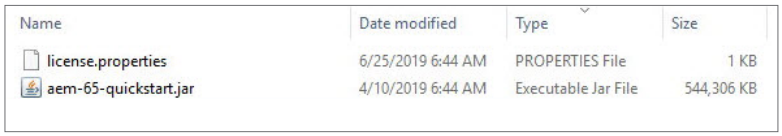
Activity 3: Install 6.5 Quickstart Jar and Service Pack

As a developer or administrator, you need to work with Adobe Experience Manager 6.5 locally using both the quickstart JAR file and the command line method. If you are attending a vILT class using ReadyTech, steps 1 through 3 were completed for you. Skip to step 4. You can also skip steps to install service pack (steps 13-20).

In this activity, you will install the 6.5 quickstart jar and a service pack. You will then start your author instance on port 4502. Optionally, if you would like to start a publish instance, go through this entire activity again and replace "author" with "publish".

To install an author Instance:

1. Create a folder structure on your file system where you will store, install, and start your author instance. For example, in:
   - Windows: **C:/adobe/aem-6.5/author-6.5.19**
   - MacOS X: **/Applications/adobe/aem-6.5/author-6.5.19** or ***x: /opt/adobe/aem-6.5/author-6.5.19**
2. If not already present in the /author folder, copy the **aem-quickstart-6.5.0.jar** and **license.properties** files, from the location provided by your instructor to your newly created directory.
3. Rename the **aem-quickstart-6.5.0.jar** file to **aem-65-quickstart.jar:**



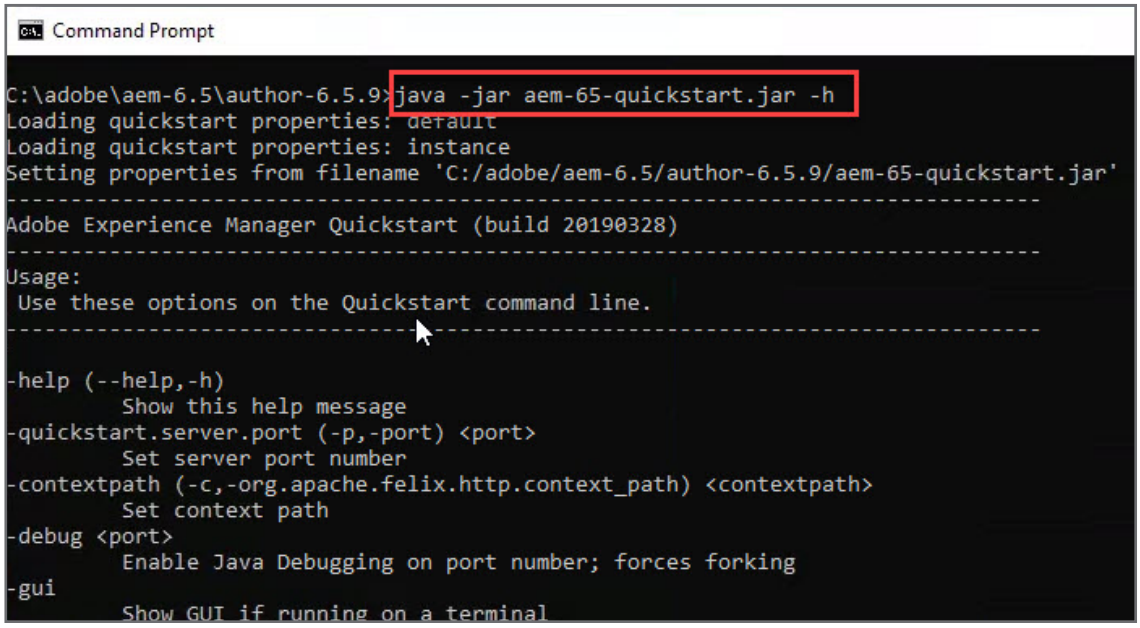
Use the command line to install AEM:

4. Open a command prompt and navigate to the directory that contains the jar and license file:

```
cd C:\adobe\aem-6.5\author-6.5.19
```

5. In the command line, run the following command to see the different parameters that you can use to run the quickstart:

```
java -jar aem-65-quickstart.jar -h
```



6. To start your AEM quickstart, use the following command:

```
java -jar aem-65-quickstart.jar -r author,dev -gui
```

Note: In the above command:

-p specifies the port number for the local AEM quickstart

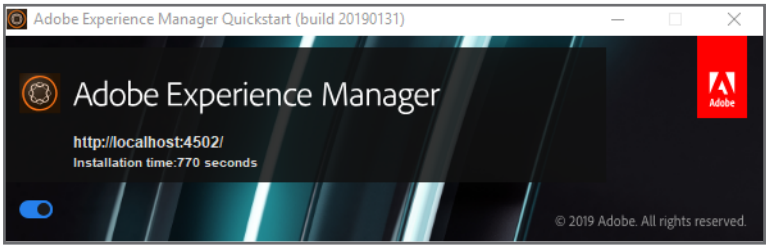
-r specifies the run mode that you are running locally

-gui opens the GUI window on your desktop

If no port number is specified, the first available port from the following list is used: 1) 4502, 2) 8080, 3) 8081, 4) 8082, 5) 8083, 6) 8084, or a random port.

Note: You can also double-click the **aem-65-quickstart.jar** file to start the author instance, but double-clicking does not add the dev runmode to the instance.

7. The command line input contained the -gui parameter, which displays the installation GUI. At this point you should see it installing the quickstart.



8. Installing base 6.5 will take anywhere from 5-7 minutes. The browser UI opens, and you will see the installation loading in the GUI that appears. At this point the service pack is installed. This will take an additional 5-7 minutes. In total, your author instance should be ready to use in 10-14 minutes.

Note: To stop the author instance, click the **ON/OFF** toggle icon at the lower left corner of the server GUI.

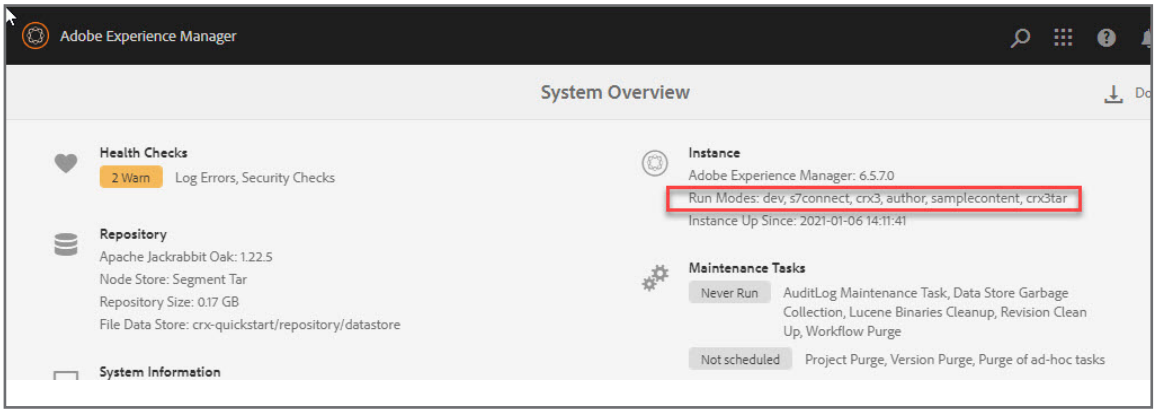
9. Each time you want to run your author instance, use the same command in step 6 to start it. However, after the first time, the startup will be as fast as one minute or less, as the initial installation task has been performed.

When the login screen appears, your author instance is ready. To log in:

10. Enter your user name and password, and click **Sign In**. If you are using a ReadyTech machine or local installation, use the following credentials to sign in:
    a. User name: **admin**
    b. Password: **admin**
11. Navigate to **Tools** > **Operations** > **System Overview** in your author instance. The **System Overview** screen appears.

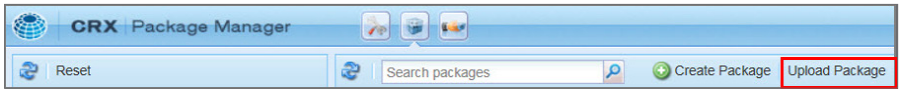
Note: A Product Navigation pop-up window might appear. If it appears, select the "Don't show this again" checkbox and click **Close** to ensure it no longer appears when you are on this page.

12. Verify the Run Modes for the service in the **Instance** area, as shown:



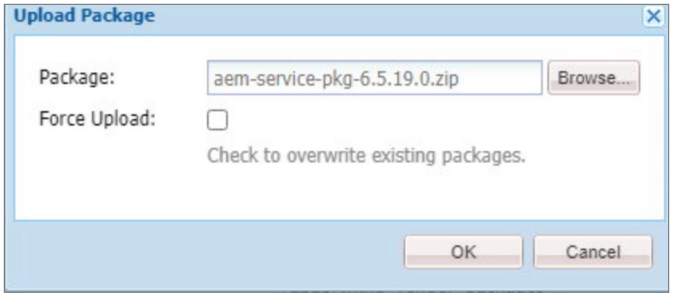
To include the service pack for AEM 6.5:

13. Navigate to http://localhost:4502/crx/packmgr/index.jsp. This takes you to the AEM Package Manager tool.
14. Click **Upload Package.**

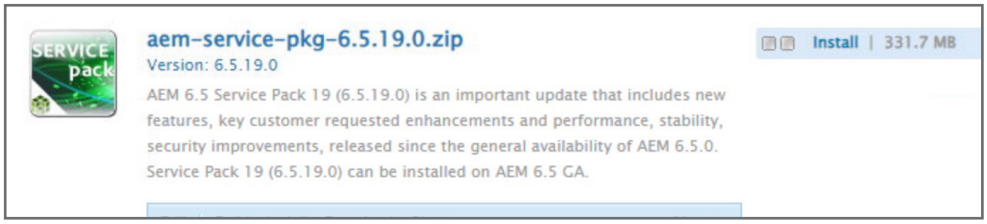


15. In the **Upload Package** dialog box, click **Browse** and select the **aem-service-pkg-6.5.19.zip** from the folder `C:\adobe\aem-6.5\author-6.5.19.`

16. Click **Open** and then click **OK** as shown:

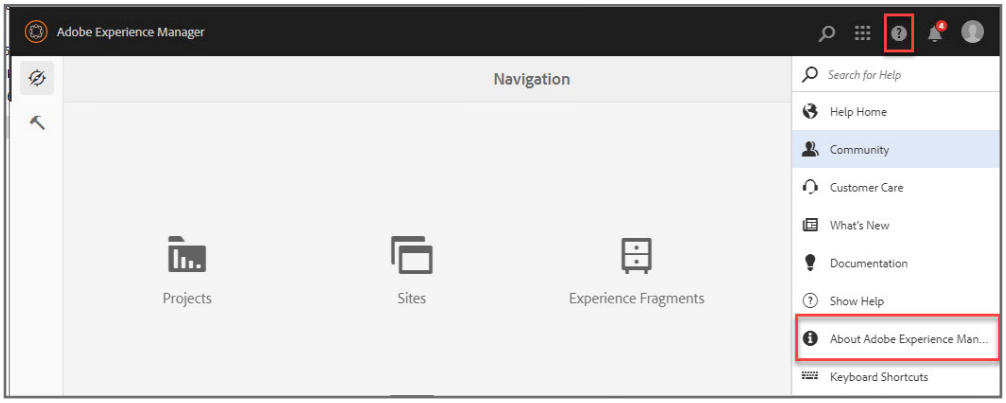


17. Your uploaded package is now available in AEM Package Manager. Click **Install**, as shown.

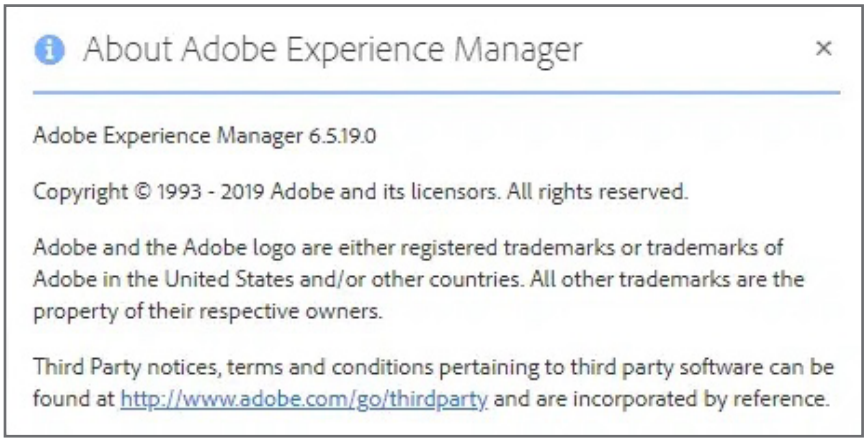


18. The **Install Package** dialog box appears. Ignore the **Advanced Settings** area and click **Install**.

19. Go back to http://localhost:4502/aem/start.html and in the top right corner, click **?** > **About Adobe Experience Manager**, as shown:

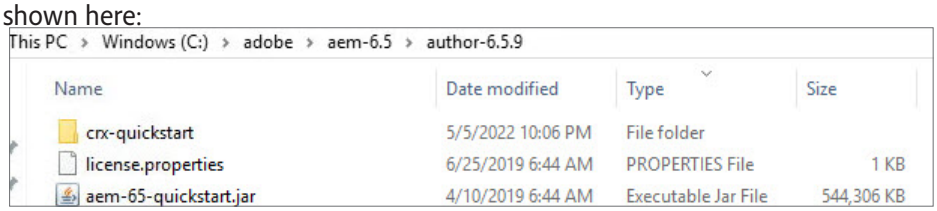


20. Verify the version of the service pack installed, as shown:



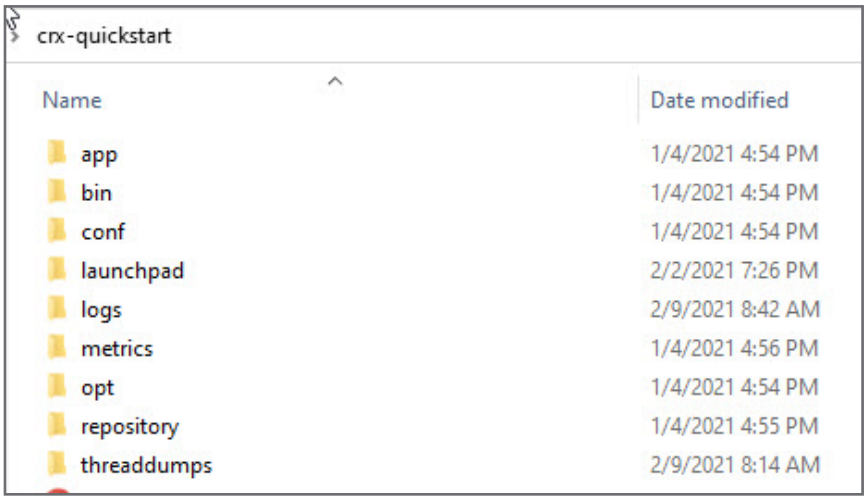
21. Instead of using the GUI to stop your author instance, use the command window you used to initially launch AEM. For Windows, type **CTRL+C** in the command window to stop your author instance.

Note: In the installation folder, notice how a **crx-quickstart** directory is also created on your computer, as shown here:



This is the extracted repository that is created upon installation.

22. Open the author instance folder and navigate to **crx-quickstart** folder. Notice all the files/folders written by the server:



Local Log Files

Additional Learning

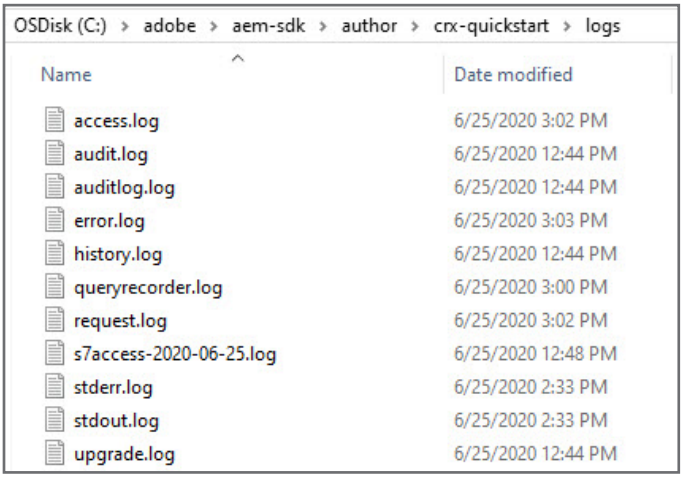
The logging framework in AEM is based on Apache Sling. The root logger defines the global settings for the logging system. The root logger is configured by using Apache Sling Logging Configuration. The org.apache.slng.commons.log bundle manages the logging system. This bundle helps:

- Implement the OSGi log service specification and register the LogService and LogReader services
- Export four commonly used APIs:
  - Apache Commons Logging
  - Simple Logging Façade for Java (SLF4J)
  - Log4j
  - Java.util.logging

Types of log files available:

- Audit.log: Registers all modern actions
- Error.log: Registers all error messages
- Request.log: Registers all access requests along with their responses
- Stderr.log: Holds error messages generated during startup
- Upgrade.log: Provides a log of all upgrade operations
- Access.log: Registers all access requests sent to AEM and the repository

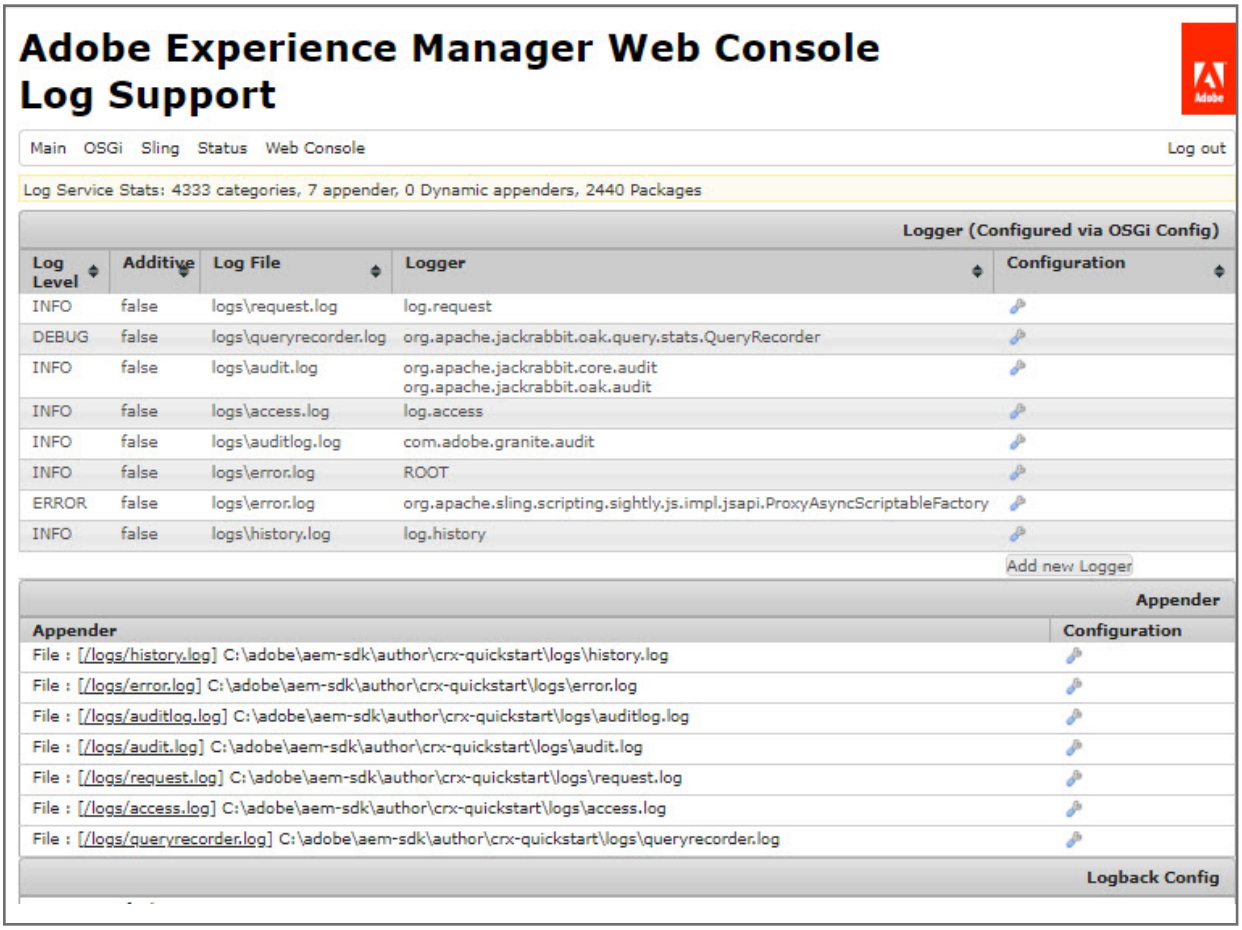
These files are available in the **/crx-quickstart/logs** installation directory.



By default, the error, access, history, and request logs rotate once per day. When this occurs, the existing log files are appended with a timestamp and a new file is created.

If you are doing local development, you can also view the log files through the Web console at:

http://localhost:4502/system/console/slinglog



Adobe Experience Manager Web Console
Log Support

Main OSGi Sling Status Web Console Log out

Log Service Stats: 4333 categories, 7 appender, 0 Dynamic appenders, 2440 Packages

Logger (Configured via OSGi Config)

Log Level	Additive	Log File	Logger	Configuration
INFO	false	logs\request.log	log.request	
DEBUG	false	logs\queryrecorder.log	org.apache.jackrabbit.oak.query.stats.QueryRecorder	
INFO	false	logs\audit.log	org.apache.jackrabbit.core.audit org.apache.jackrabbit.oak.audit	
INFO	false	logs\access.log	log.access	
INFO	false	logs\auditlog.log	com.adobe.granite.audit	
INFO	false	logs\error.log	ROOT	
ERROR	false	logs\error.log	org.apache.sling.scripting.sightly.js.impl.jsapi.ProxyAsyncScriptableFactory	
INFO	false	logs\history.log	log.history	

Add new Logger

Appender

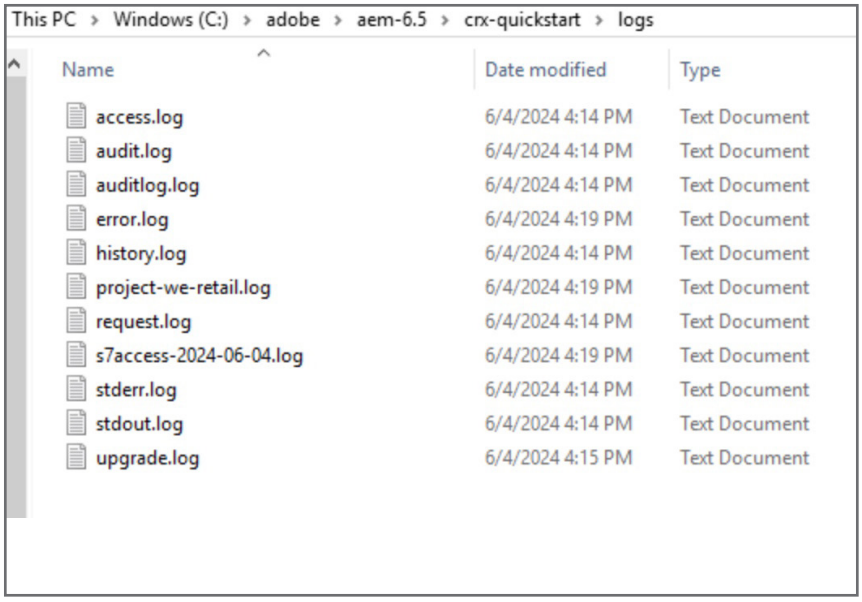
Appender	Configuration
File : [/logs/history.log] C:\adobe\aem-sdk\author\crx-quickstart\logs\history.log	
File : [/logs/error.log] C:\adobe\aem-sdk\author\crx-quickstart\logs\error.log	
File : [/logs/auditlog.log] C:\adobe\aem-sdk\author\crx-quickstart\logs\auditlog.log	
File : [/logs/audit.log] C:\adobe\aem-sdk\author\crx-quickstart\logs\audit.log	
File : [/logs/request.log] C:\adobe\aem-sdk\author\crx-quickstart\logs\request.log	
File : [/logs/access.log] C:\adobe\aem-sdk\author\crx-quickstart\logs\access.log	
File : [/logs/queryrecorder.log] C:\adobe\aem-sdk\author\crx-quickstart\logs\queryrecorder.log	

Logback Config

Activity 4: Observe the Local Log Files

In this activity, you will observe the local log files created after the installation..

1. Open the author instance folder and navigate to **\crx-quickstart\logs**. The list of log files is shown here:



2. Observe the log files.

Note: During development, any logs your code produces will automatically be filtered into the error.log file. Simply filter for your project name to find the logging results.

References

- Use this link for more information on Installation

Developer Tools

Introduction

Adobe Experience Manager (AEM) has common developer tools that are used to develop your Java Content Repository (JCR), Apache Sling, or AEM applications.

Developer Tools

The three most common developer tools available in AEM for developers and administrators are:

- CRXDE Lite
- Web Console
- Package Manager

Local Development

- For local development, Developers have full access to CRXDE Lite (/crx/de) and the AEM Web Console (/system/console).

Note: On local development (using the cloud-ready quickstart), /apps and /libs can be written to directly, which is different from Cloud environments where those top-level folders are immutable.

AEM as a Cloud Service Development Tools

- Customers can access CRXDE lite on the development environment but not stage or production. The immutable repository (/libs, /apps) cannot be written to at runtime, so attempting to do so will result in errors.
- Customers have access to package manager for author instances (and not publish) in Cloud Service. They can only upload packages containing mutable content.
- A set of tools for debugging AEM as a Cloud Service developer environments are available in the Developer Console for dev, stage, and production environments. The URL can be determined by adjusting the author or publish service URLs as follows:
  `https://dev-console>-<namespace>.<cluster>.dev.adobeaemcloud.com`
- As a shortcut, you can use the following Cloud Manager CLI command to launch the developer console based on an environment parameter described below:
  `aio cloudmanager:open-developer-console <ENVIRONMENTID> --programId <PROGRAMID>`

Activity 1: Install a Package

As an AEM developer or administrator of development operations, you need to have sample content to reference and base your code off of. In this task, you will install the WKND content package so you can reference the WKND implementation during development. The WKND content package is a container content package. This means the only thing that is contained in this package is other content packages. This allows for a single content package for a project but still maintains the separation of mutable and immutable content.

This activity includes the following tasks:

1. Install a container package.
2. Verify the newly installed site.

Task 1: Install a Container Package

In this task, you will validate and install a container package using the Package Manager and then validate that the mutable and immutable content packages extracted were successfully installed.

Download the Container Package:

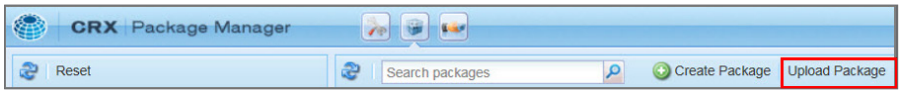
1. If your AEM instance is not running, start it.
2. If you are using ReadyTech, WKND can be found under
   - SDK- **c:/adobe/aem-sdk/**
   - 6.5- **c:/adobe/aem-6.5/**

Note: Your instructor will inform you which WKND version to use for class. WKND is already downloaded on ReadyTech under the folder that correlates with the AEM type you are using.
Cloud Service: C:\adobe\aem-sdk\aem-guides-wknd.all-#.#.#.zip
6.5: C:\adobe\aem-sdk\aem-guides-wknd.all-#.#.#-classic.zip

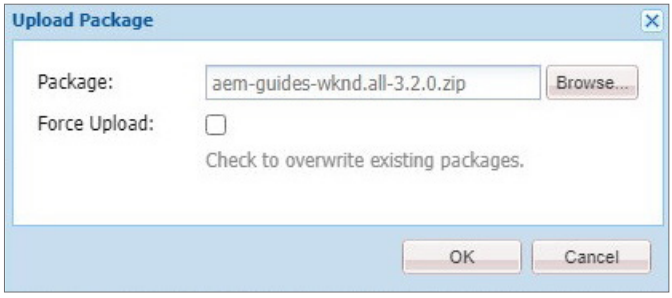
Note: Alternatively, you can download the WKND from the release page: https://github.com/adobe/aem-guides-wknd/releases.

To install your package:

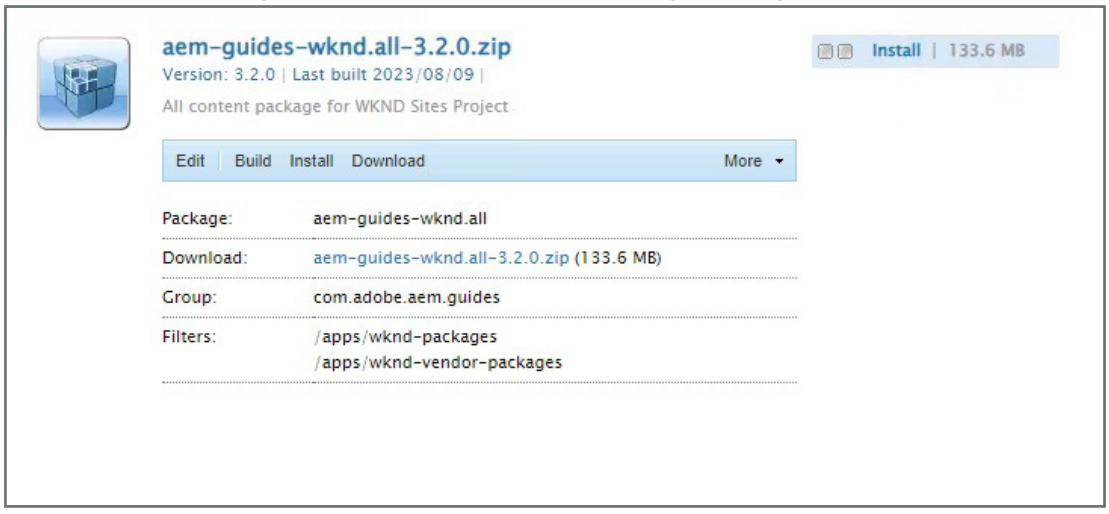
3. Verify you are logged on to your AEM author service on port 4502 (http://localhost:4502).
4. Navigate to http://localhost:4502/crx/packmgr/index.jsp. This will take you to the AEM Package Manager tool.
5. Click **Upload Package.**



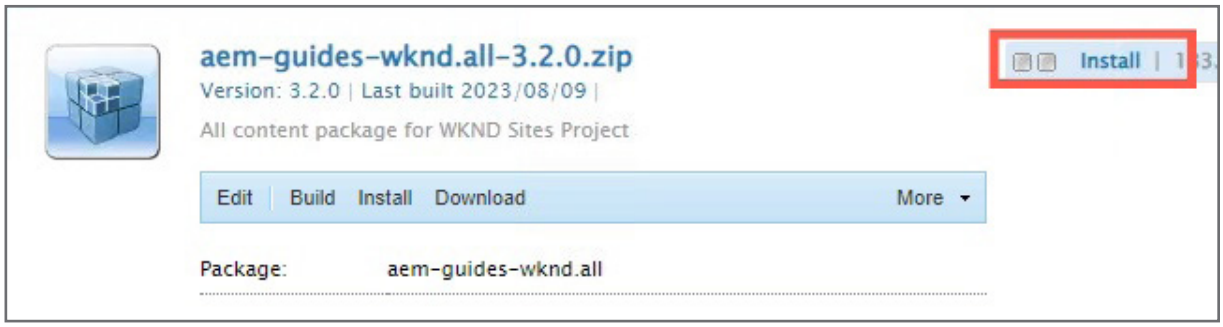
6. In the **Upload Package** dialog box, click **Browse** and select the **aem-guides-wknd.all-#.#.#.zip** package that you downloaded in Step 1.
7. Click **Open** and then click **OK** as shown:



Your uploaded package is now available in AEM Package Manager, as shown.

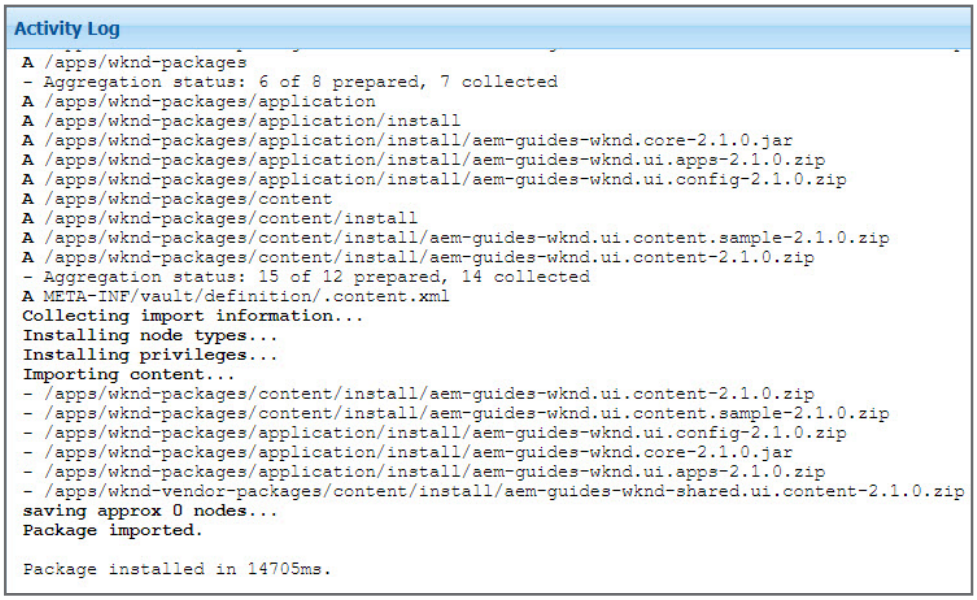


8. Click **Install**, as shown.



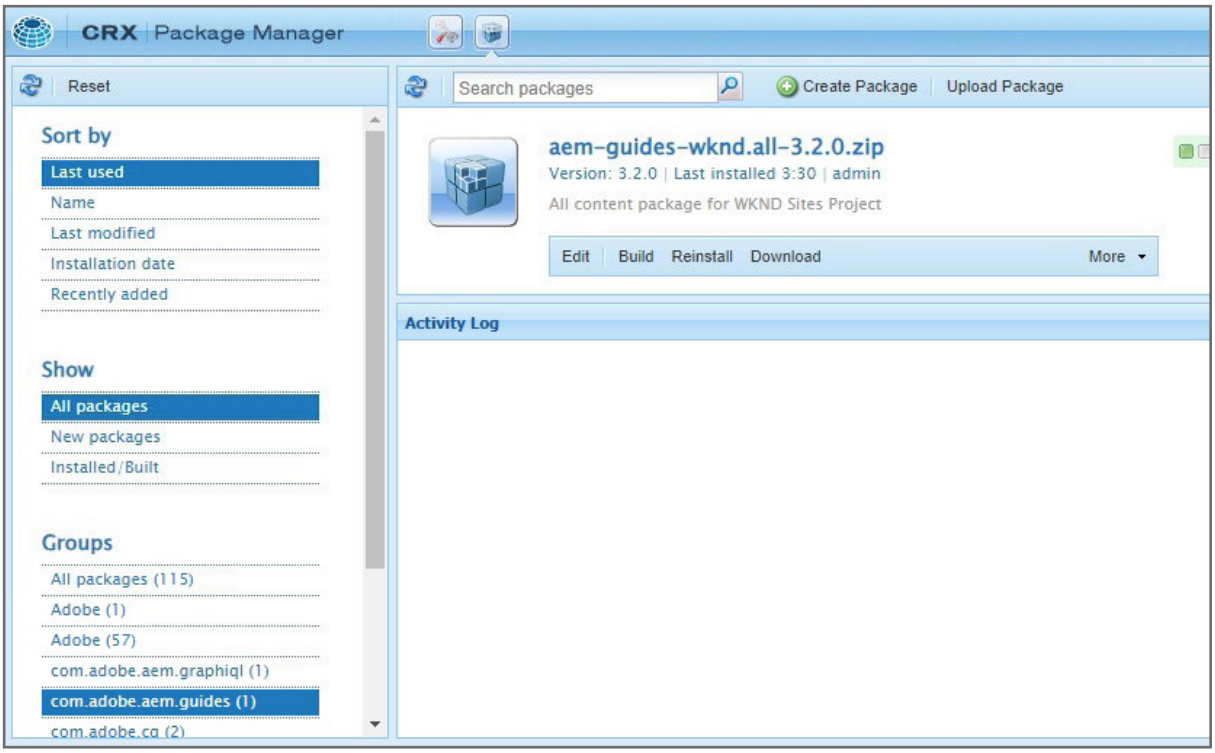
The **Install Package** dialog box appears.

9. Ignore the **Advanced Settings** area and click **Install**.
10. Check the **Activity Log**. You can see that the content was added from the package.



Note: This container content package that you installed contains mutable and immutable content packages.

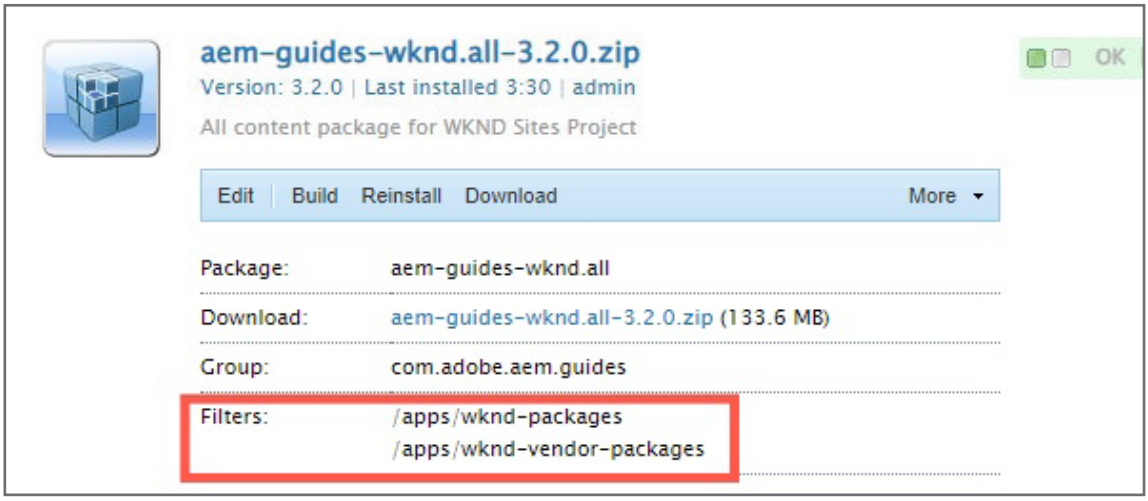
11. In the left navigation pane, under **Groups**, click on **com.adobe.aem.guides** and verify the container package, as shown:



Note: If you do not see **com.adobe.aem.guides (1)** in the left navigation pane (under **Groups**) OR, after clicking the **com.adobe.aem.guides (1)** link, you do not see the two packages (referenced in Step 11) in the main Packages area, refresh the Package Manager. Sometimes, it takes time for the screen to refresh so you can see the updated links (on the left side) as well as see the corresponding packages associated with the links on the right side.

Note: A container package can ONLY include other content packages (.zip) and bundles (.jars).

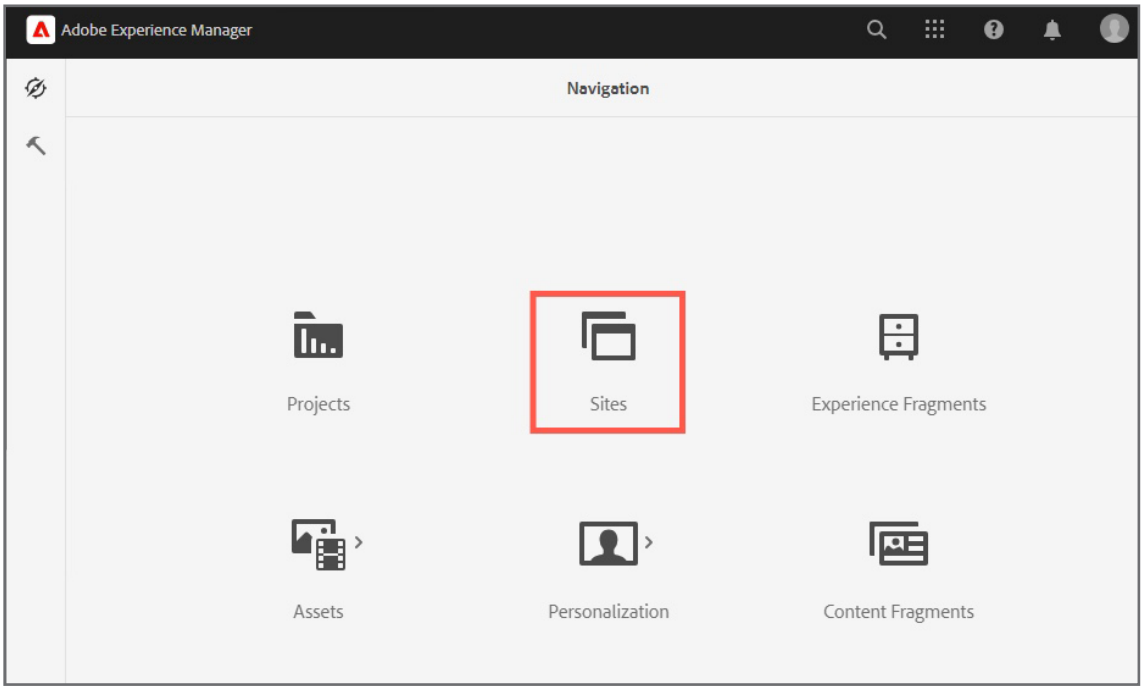
12. The **aem-guides-wknd.all-#.#.#.zip** contains all of the mutable content for the WKND project. Click the **aem-guides-wknd.all-#.#.#.zip** link and scroll down to the **Filters** section:



Task 2: Verify the Newly Installed Site

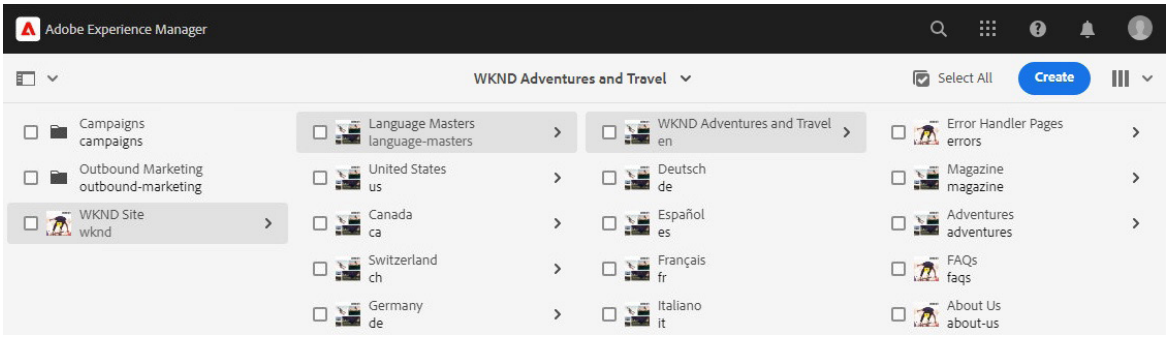
In this task, you will navigate to Sites and verify the newly installed site.

1. Click the browser tab with the Adobe author service.
2. Click **Adobe Experience Manager** in the upper left.
3. In the Navigation page area, click **Sites**, as shown.



Note: You might see the **Product Navigation** tutorial dialog guiding you through the navigation. You may click **Next** to proceed through the tutorial and learn the basic AEM UI elements and navigation, or you may click **Close** to hide the tutorial.

4. In the column view, verify the new **WKND Site**. Navigate to **WKND Site** > **Language Masters**, as shown.



TIP: If you see a checkmark on the WKND site thumbnail, as shown here, it means you have *selected* WKND site for editing and/or managing the page. Click the thumbnail again to clear the selection and click the right-pointing arrow instead.

You have successfully installed the WKND container package, which contains both mutable and immutable content packages.

Note: If you would like to learn how to create, build, and download a package, see the next activity, "Activity 2: Create, Build, and Download Packages" on page 46.

Activity 2: Create, Build, and Download Packages

In this activity, you will learn how to create a content package. Locally, you can create content packages for mutable and immutable content. However, on an AEM service, only immutable content packages can be created and managed because mutable code (/apps) can only be installed to an AEM service via Cloud manager. Your instructor might provide you with a course-filter.txt file; you can use those filters rather than the example filters in this activity.

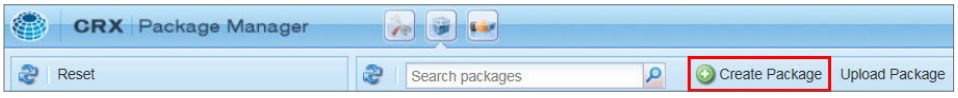
This activity includes the following tasks:

1. Create an immutable package.
2. Create a mutable package.

Task 1: Create an Immutable Package

In this task, you will validate and install a package using the Package Manager.

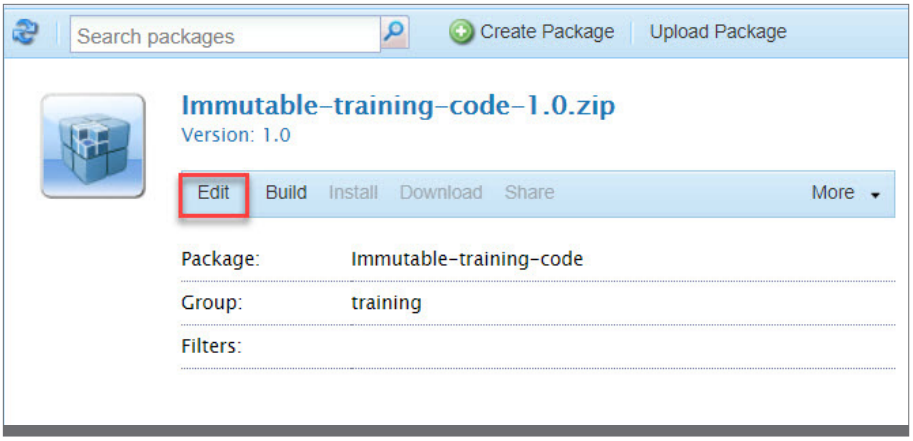
1. In your AEM **Package Manager** tool, click **Create Package** as shown:



2. In the **New Package** dialog box, type the following details:
   a. Package Name: **Immutable-training-code**
   b. Version: **1.0**

c. Group: **training**

3. Click **OK**. The **Immutable-training-code** package is created.
4. Click **Edit** on the newly created package, as shown.



5. To add filters to the package, click the **Filters** tab and then click **Add Filter**.

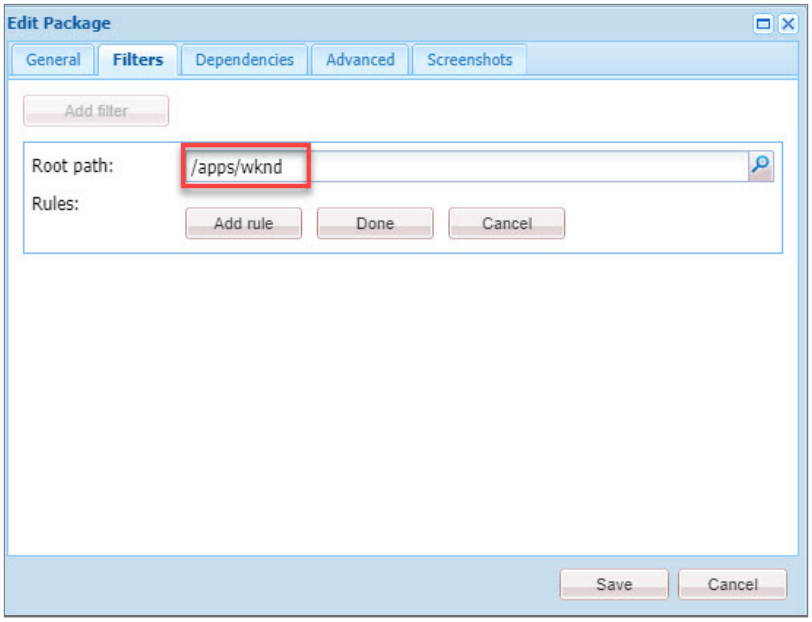
Note: Filters are the mechanism used to add content to packages. Here, you specify the paths that contain the content from the JCR you want to include in a package. Before adding filters, your package is completely empty. You may also restrict file types added to a package using filter rules, such as excluding all *.txt files.

6. For the **Root** path, type: **/apps/wknd**

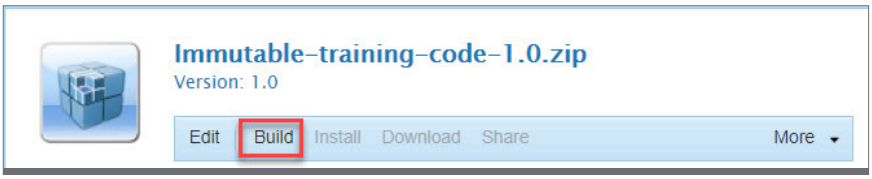
TIP: As you type in the Root path field, the field will auto-complete.

Note: If your instructor has provided you with a **course-filter.txt** file, use the immutable paths noted in this file.

7. Click **Done**, as shown.



8. Click **Save**.
9. Click **Build** to build the package, as shown.



10. Click **Build** again in the confirmation dialog box. The package is now ready for download.

Note: As the package is being built, the activity log is running at the bottom of the screen. This area is known as the Activity Log. The log shows a series of "A" actions. "A" denotes a node is being added to the content package.

11. Click the **Download** link to download a copy of the package to your computer. The package is downloaded to your computer's default **Downloads** folder for the browser.

Note: You should see a list of all packages in your service organized by group on the left side menu. Note that there is now one new package, indicated by **(1)**, in the **training** group that you just created for **Immutable-training-code**. Therefore, you can now use the group name to categorize and organize your packages in AEM.

You have successfully created an immutable package.

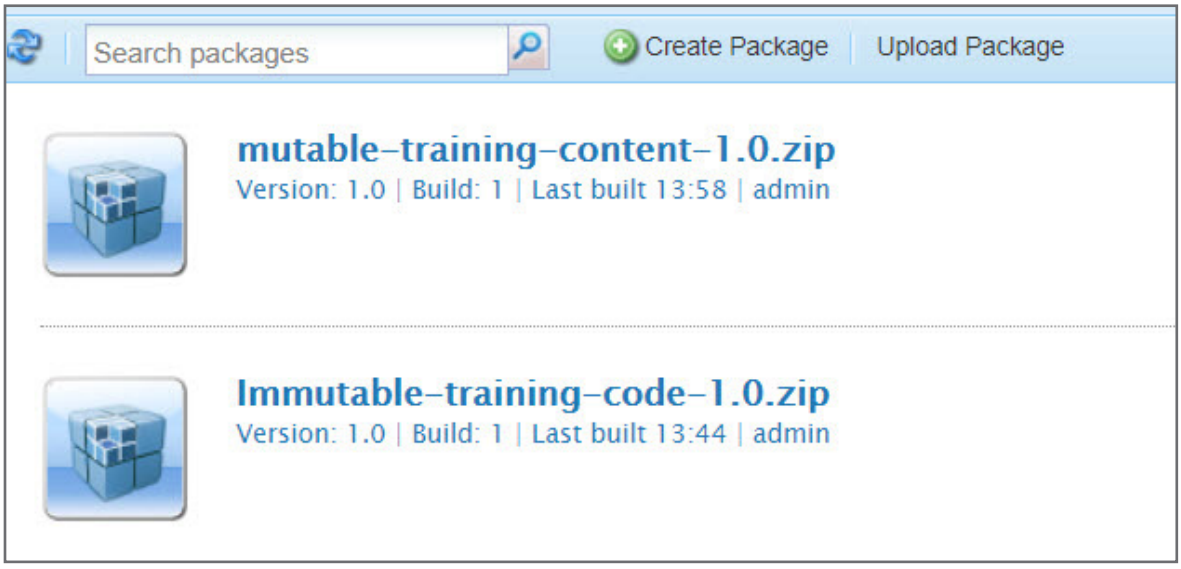
Task 2: Create a Mutable Package

1. In your **CRXDE Lite** browser tab, click the **Package Manager** icon. The **Package Manager** screen appears.
2. Click **Create Package**.
3. In the **New Package** dialog box, type the following details:
   a. Package Name: **mutable-training-content**
   b. Version: **1.0**
   c. Group: **training**
4. Click **OK**. The **mutable-training-content** package is created.

5. Click **Edit** on the newly created package.
6. To add filters to the package, click the **Filters** tab and then click **Add Filter**.
7. For the **Root** path, enter **/conf/wknd**.

Note: If your instructor has provided you with a **course-filter.txt** file, use the mutable paths noted in this file.

8. Click **Done** and click **Save**.
9. Click **Build** to build the package. Click **Build** again in the confirmation dialog box. The package is now ready for download.
10. Click the **Download** link to download a copy of the package to your computer. The package is downloaded to your computer's default **Downloads** folder for the browser.
11. On the left side menu, click the **training** group link and notice the packages under the group.



You have successfully created a mutable package.

References

Bookmark/"favorite" these sites for a local author installation (development environment):

- CRXDE Lite: http://localhost:4502/crx/de/index.jsp
- Web Console: http://localhost:4502/system/console
- Package Manager: http://localhost:4502/crx/packmgr/index.jsp

Sites Authoring Basics

Introduction

In Adobe Experience Manager (AEM), the author service allows you to create, update, and review content before publishing it. These authoring functions are made available to you through the AEM UI.

AEM Sites

A page in AEM is similar to a webpage and contains **components** such as text, images, and videos. These pages are created from **templates** that define the structure of the page—including where each component can be placed.

Reference Site: WKND

The WKND Site is a fully functional site created with AEM. The WKND is a fictional online magazine and blog that focuses on nightlife, activities, and events in several international cities. This site is built with the following best practices of AEM:

- Localized site structure, with language masters live-copied into country-specific sites
- Content fragments
- Core components
- Responsive layout for all pages
- All editable templates
- HTML Template Language (HTL) for all components

Note: While WKND illustrates an adventure and relaxation site, it is not specific to a leisure site. The site is set up in a way that it can be applied to any vertical.

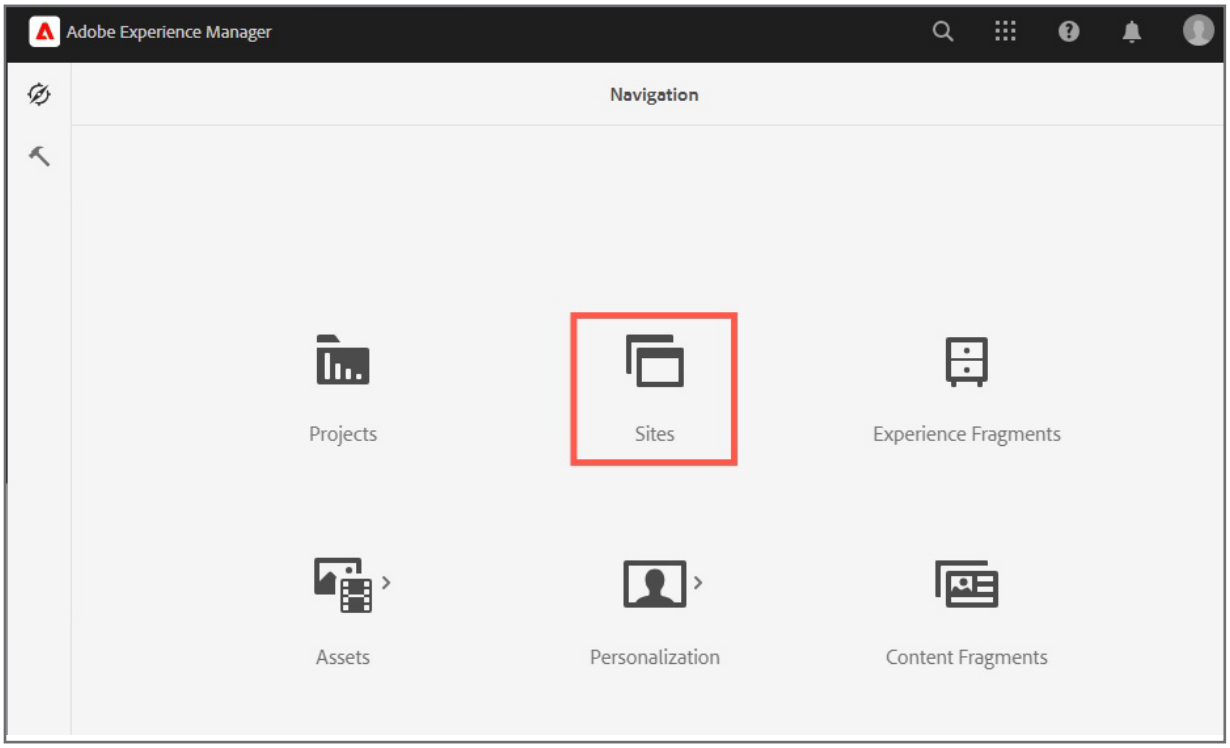
Activity 1: Create a Page in AEM

As an author, you need to create and edit a page in AEM Sites under the built-in WKND site hierarchy.

In this activity, you will create a demo page using the content template available in AEM.

To create a new page:

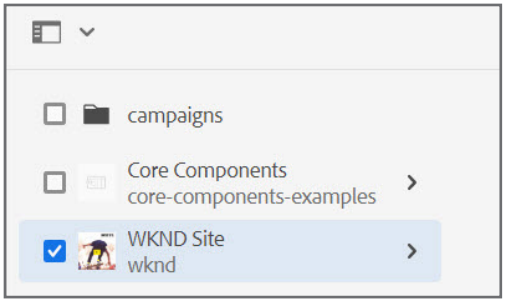
1. Ensure you have started and logged on to your AEM author service on port 4502.
2. The Navigation page is displayed by default in the content area. Click **Sites**, as shown.



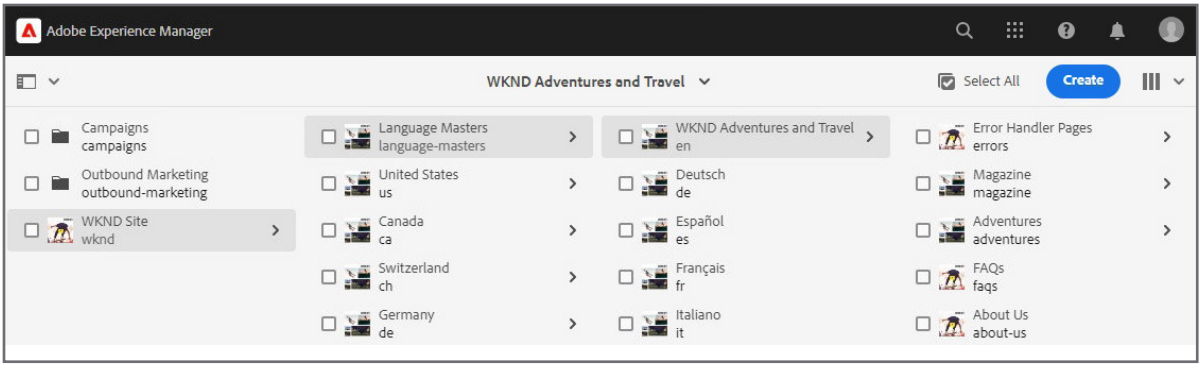
The column view is displayed.

Note: You might see the **Product Navigation** tutorial dialog guiding you through the navigation. Click **Next** to proceed through the tutorial and learn the basic AEM UI elements and navigation, or click **Close** to hide the tutorial.

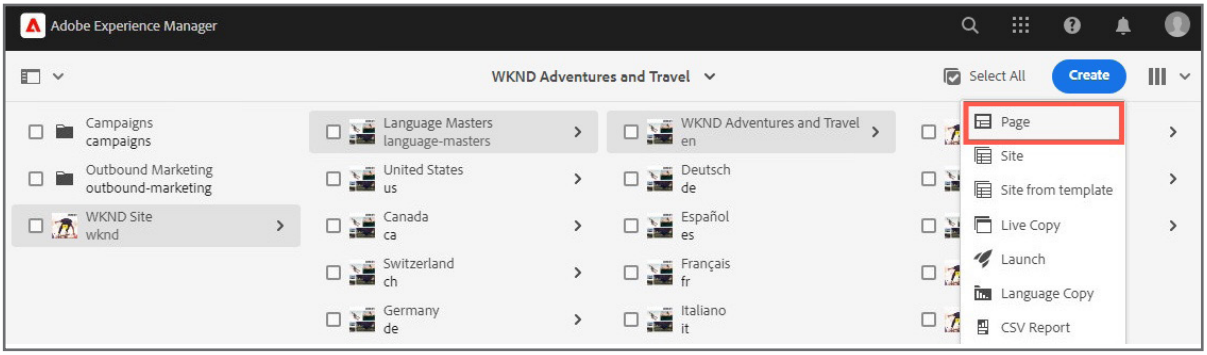
TIP: If you see a checkmark on the WKND site thumbnail as shown below, it means you have *selected* WKND site for editing and/or managing the page. Click the thumbnail again to clear the selection and click the right-pointing arrow instead.



3. In the column view, click the right-pointing arrow next to **WKND Site** and navigate to **WKND Site** > **Language Masters** > **WKND Adventures and Travel,** as shown:

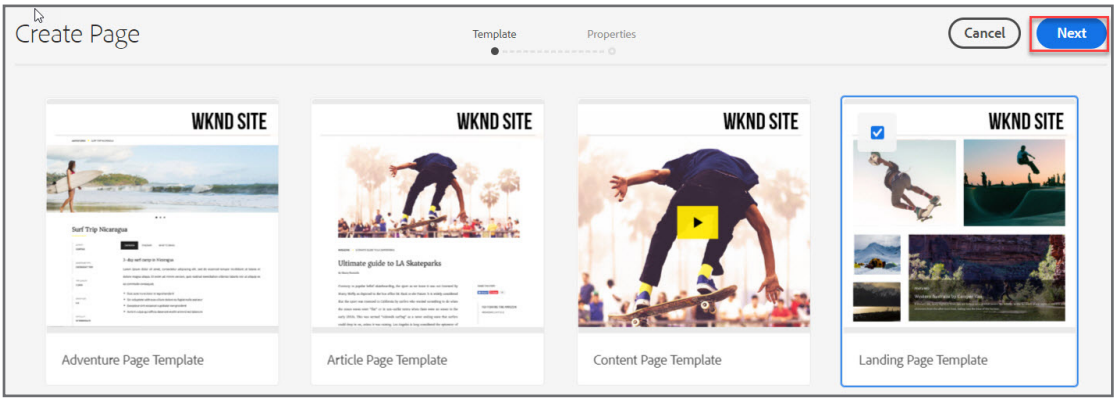


4. Click **Create** > **Page**, as shown.

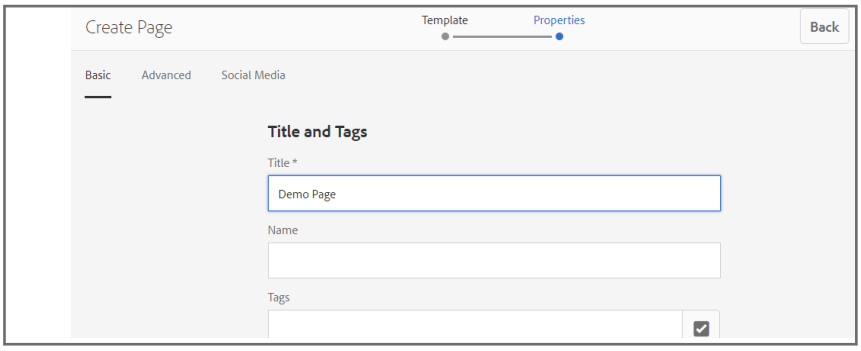


After you click **Page**, a wizard appears where you need to select a template for your page.

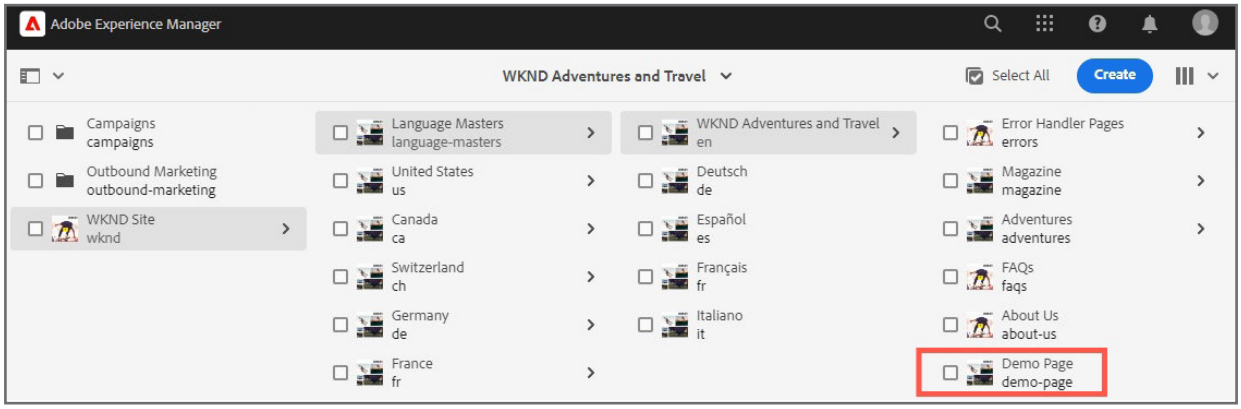
5. Click the **Landing Page Template** to select it and then click **Next**, as shown.



6. In the **Properties** step of the page creation wizard, enter **Demo Page** in the title field, as shown:



7. Click **Create** in the upper-right corner to create the page. A **Success** dialog box is displayed with a message that your page has been created.
8. Click **Done**. The new page appears as a child page of WKND Site. The page title (**Demo Page**) appears in the column view. You can see the page name **demo-page** below the page title.

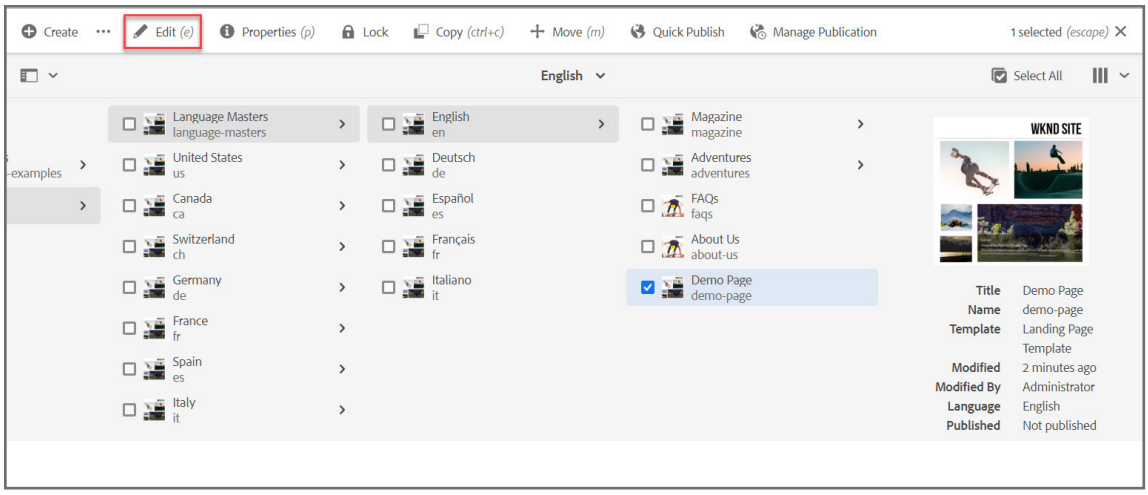


You can create subpages of any page using the same process.

Activity 2: Edit a Page in AEM

In this activity, you will edit the page you just created by adding text and image components.

1. Click checkbox for the **Demo Page** (thumbnail) you just created to select it, and click **Edit *(e)*** from the actions bar. Be patient as it may take a few minutes to load the AEM page editor the first time.



Note: You will notice the shortcut keys in the header bar. After you have selected a page, you may use the keyboard shortcuts such as **e** to edit the page, **p** to view page properties, and **Ctrl+C** to copy the page.

The Demo Page opens in a new tab in your browser. A Modes window may display, click **Close** to close it.

Ensure the page is open in edit mode by checking in the upper-right corner to see if the **Edit** mode is highlighted, as shown:

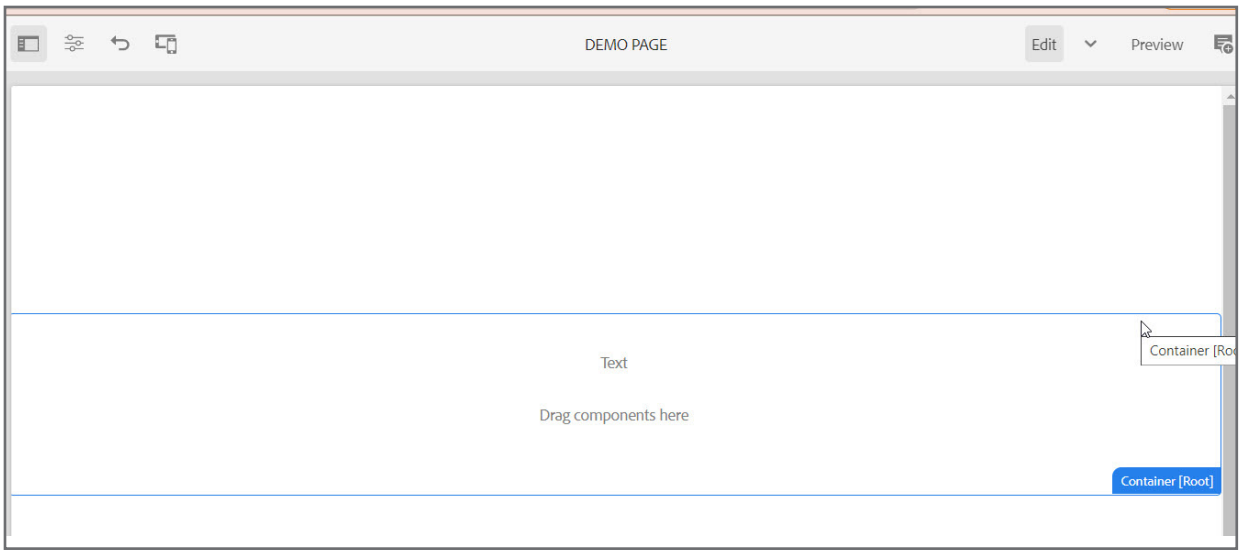
To add a text component to the page:

2. Click the **Toggle Side Panel** icon in the upper-left corner, as shown.

The side panel is displayed.

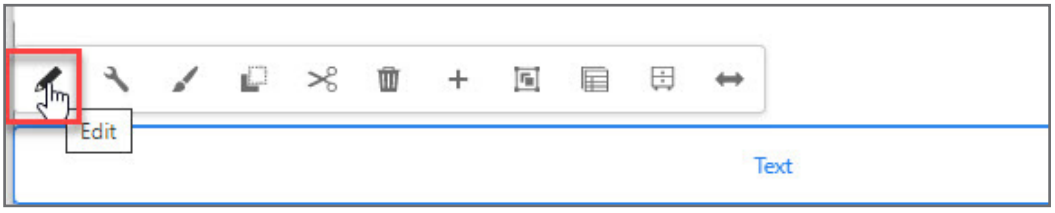
3. Click the **Components** icon.
4. In the **Filter** field, type **text** and press **Enter** to search for the **Text** component. The search yields results that contain the word "text".
5. Drag the **Text (WKND.Content)** component onto the **Drag components here** box, as shown:

Your editor should look like the following:



Tip: If the ordering is not correct, just drag the text component again to the **Drag components here** box.

6. Click the **Text** component, and then click **Edit** (**pencil** icon) from the component toolbar as shown. A text editor opens in the same tab.

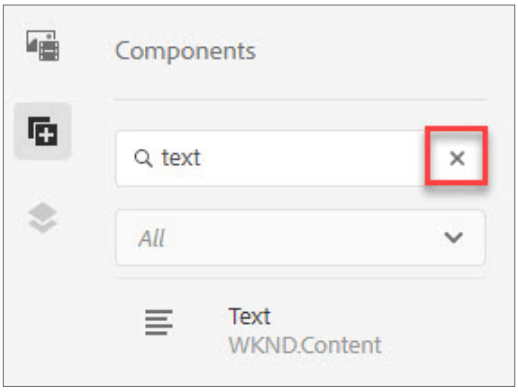


7. Add sample text of your choice in the text editing form that appears and click **Done** (**checkmark** icon) to save the changes. The text is added to the page.

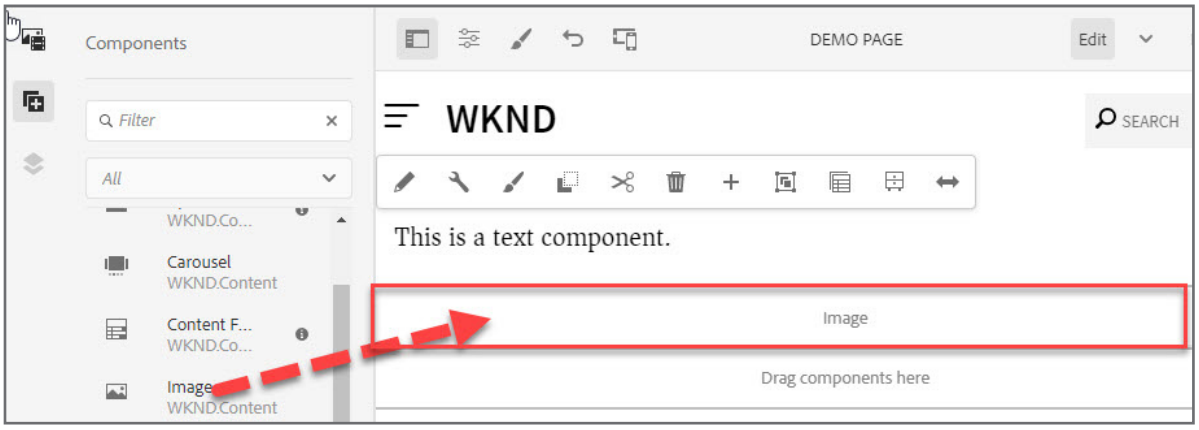
You have now added the first component to your page!

To add the image component to your page:

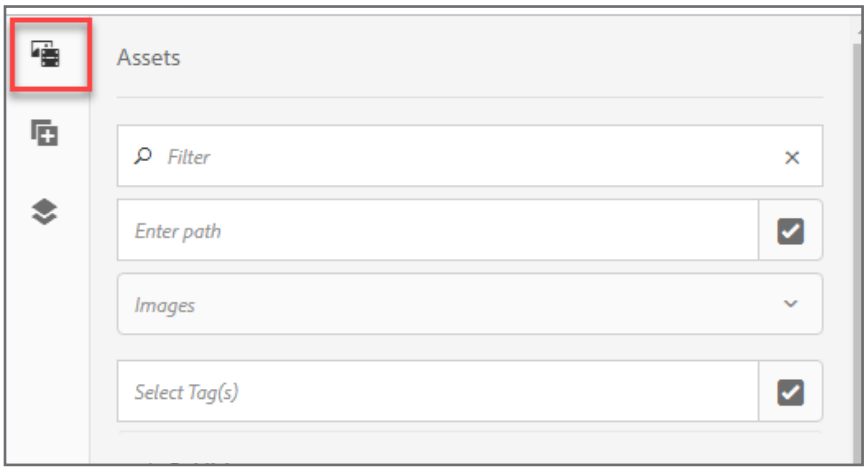
8. In the side panel, click **X** to clear the search, as shown.:



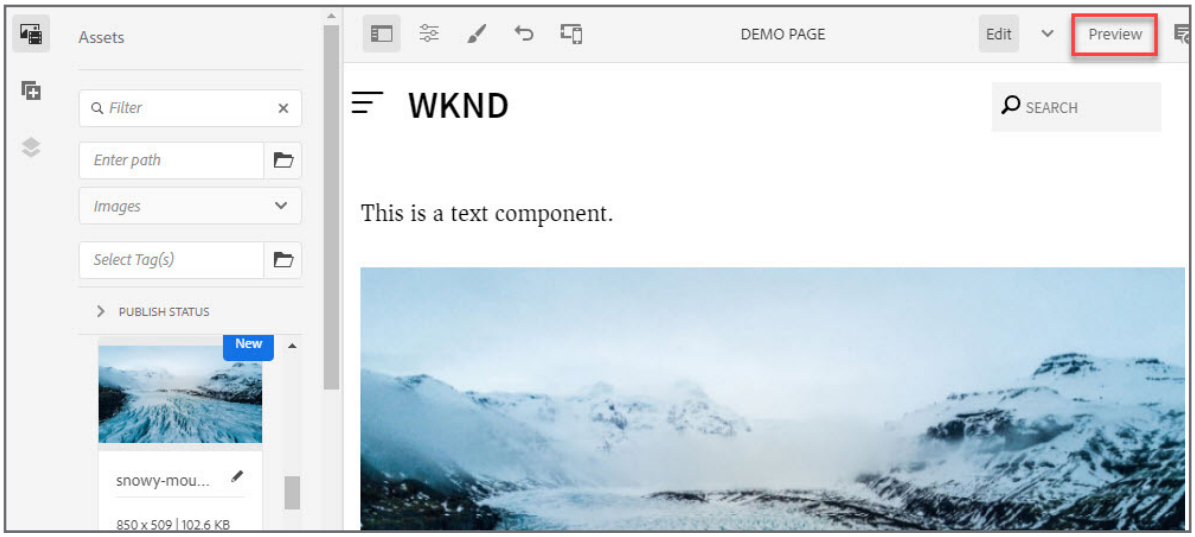
9. Scroll down and drag the **Image (WKND.Content)** component onto the **Drag components here** box as shown:



10. Click the **Assets** icon in the side panel as shown:



11. Drag any image from the **Assets** tab onto the **Image** component. AEM will add the image to the page.
12. Click **Preview** in the upper-right corner to preview the changes to your page, as shown.



13. Press **CTRL+SHIFT+M** to go back to the edit mode. You can use this method to toggle between **Preview** and **Edit** mode, as you may need to switch often when adding and testing page content.

Note: This keyboard shortcut is AEM-specific and does not depend on the operating system or the browser. In other words, this shortcut is universal to AEM. The only variation to this is the use of ⌘ .This key is specific to macOS, and is equivalent to the **CTRL** key in Windows.

Note: At this time, your page has been edited, but is not yet live. In order to push content changes to the web, your page must be published ("activated") to a publish service of AEM.

Author with Core Components

Additional Learning

In AEM, Core components:

- Are production-ready components that Adobe provides as a base for site creation
- Provide robust and extensible base components built on the latest technology and best practices adhering to accessibility guidelines
- Are compliant with the Web Content Accessibility Guidelines (WCAG) 2.0 AA standard

To learn more about Core Components, visit: https://experienceleague.adobe.com/en/docs/experience-manager-core-components/using/introduction

Note: As a developer, understanding the Core components authoring capabilities and leveraging the Core components is vital for rapid project development. After you know the authoring features of the Core components, you can create a development plan for designing and customizing the components further, if required. To test the authoring capabilities of the Core components, you must create proxy components and add the Core component client libraries to your project to test the authoring capabilities.

Activity 3: Explore Core Component Authoring

Core components provide the toolset to quickly start your projects with minimal component development. In this activity, you will explore the authoring capabilities of different core components. WKND uses the core components through proxy components, which follows best practices. This is a good activity to come back to after completing this class to fully understand the capabilities of some of the core components.

This activity includes the following tasks:

1. Create a Page.
2. Explore different components.

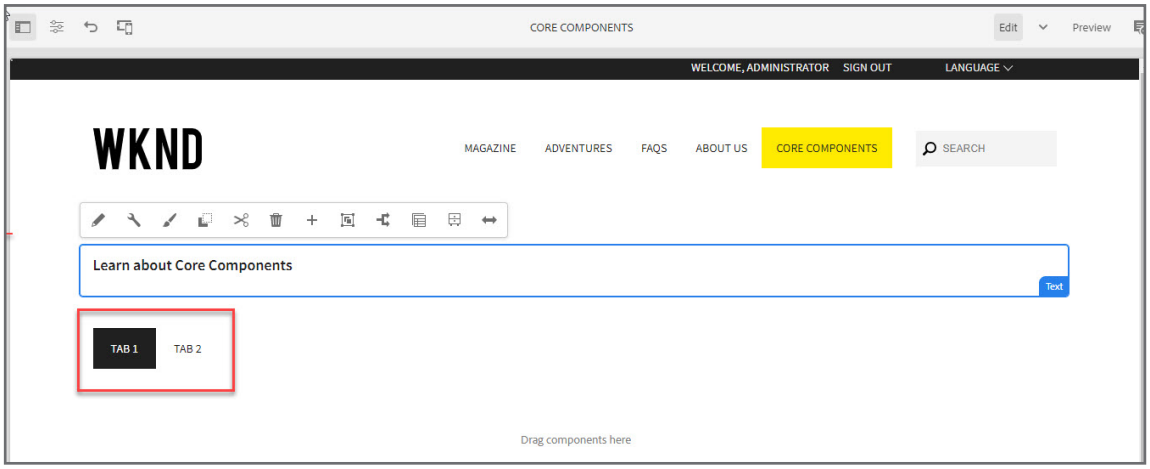
Task 1: Create a Page

1. In your AEM author instance, navigate to the **Sites** > **WKND Site** > **United States** > **WKND Adventures and Travel** page.
2. Click **Create** > **Page**. The **Create Page Template** wizard opens.
3. Select the **Content Page Template** and click **Next**. The **Create Page Properties** wizard opens.
4. In the **Title** field, type **Core Components**, and click **Create**. The **Success** dialog box is displayed.
5. Click **Open** to open the page.

Task 2: Explore Different Components

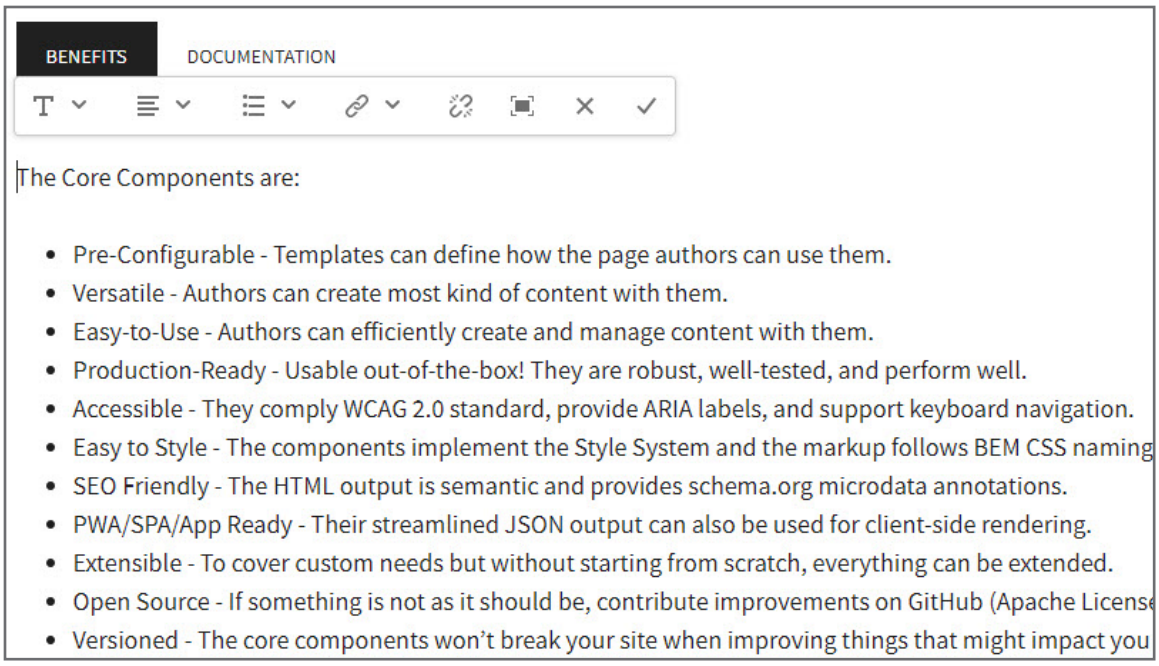
1. On the page you created in the previous task, ensure you are in the **Edit** mode, and click the **Toggle Side Panel**.
2. Click the **Components** icon in the left panel. The list of available components is displayed.
3. Drag the **Title** component onto the **Drag components here** area.
4. Double-click the **Title** component you added to edit it. The **Title dialog** box opens.
5. In the **Title** field, type **Learn about Core Components**, and click the **Done** (checkmark) icon in the upper right. The title is updated.

6. From the Components browser in the left panel, drag the **Tabs** component onto the **Drag components here** area. Two tabs, Tab 1 and Tab 2, are added, as shown:



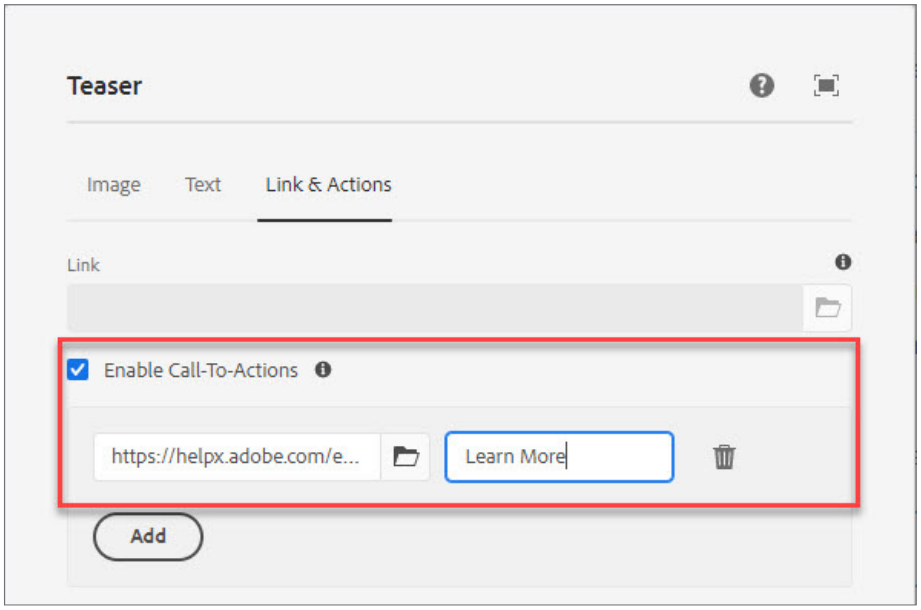
7. Double-click the tabs component. The **Tabs** dialog box opens.
8. On the default **Items** tab, rename **Tab 1** as **Benefits**.
9. Rename **Tab 2** as **Documentation**.
10. Click **Done** to save the changes.
11. On the **Benefits** tab, click the **Drag components here** area.
12. Click the **Insert component** icon (the + icon) and add a **Text** component.
13. Navigate to the **Sites Authoring Basics** activity files provided to you and copy the content from \Activity_Files\training-files\Sites_Authoring_Basics**Benefits.txt**.
14. On the **Components** page, double-click the **Text** component. The **Text** dialog box opens.

15. Paste the content and apply the bullet list format to the text, as shown:

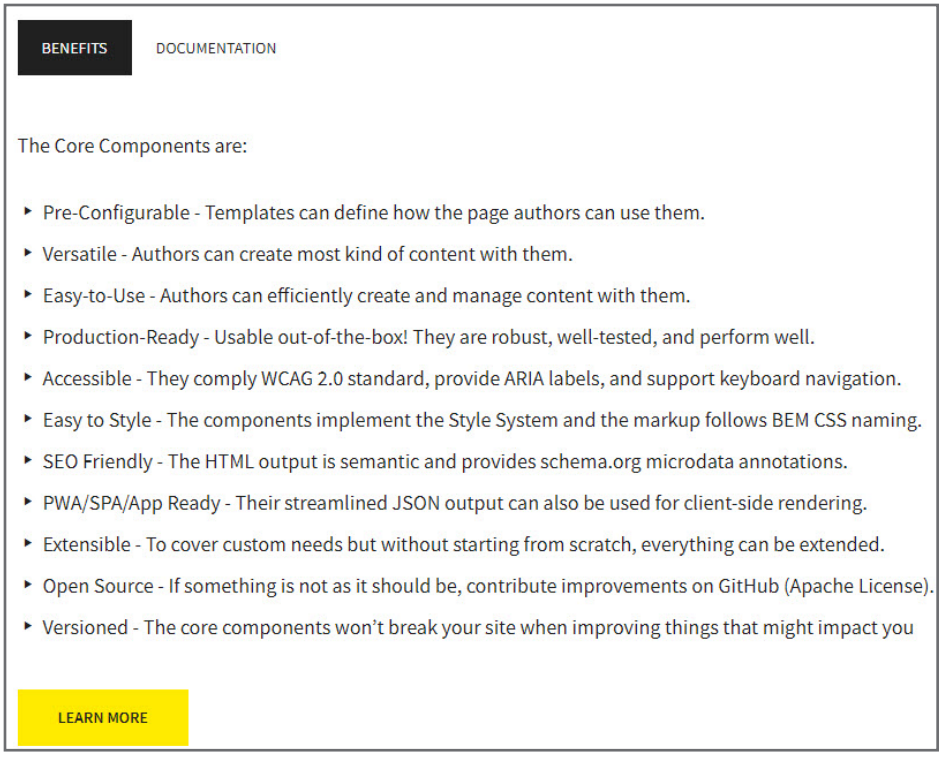


16. Click **Done** to save the changes. The text is added to the **Benefits** tab.
17. Click the **Drag components here** area below the text you added in the previous step.
18. Click the **Insert component** icon (the + icon), and add a **Teaser** component.
19. Click the **Teaser** component, and click the **Configure** icon (the wrench icon). The **Teaser** dialog box opens.
20. On the **Link & Actions** tab, select the **Enable Call-to-Actions** checkbox.
21. In the **Link** field, type https://helpx.adobe.com/experience-manager/core-components/using/introduction.html.

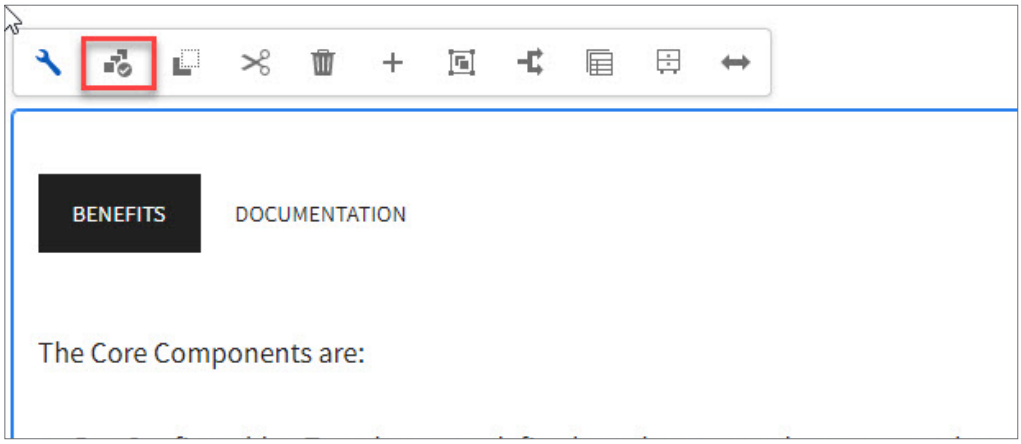
22. In the **Text** field, type **Learn More**. The **Teaser** dialog should look, as shown:



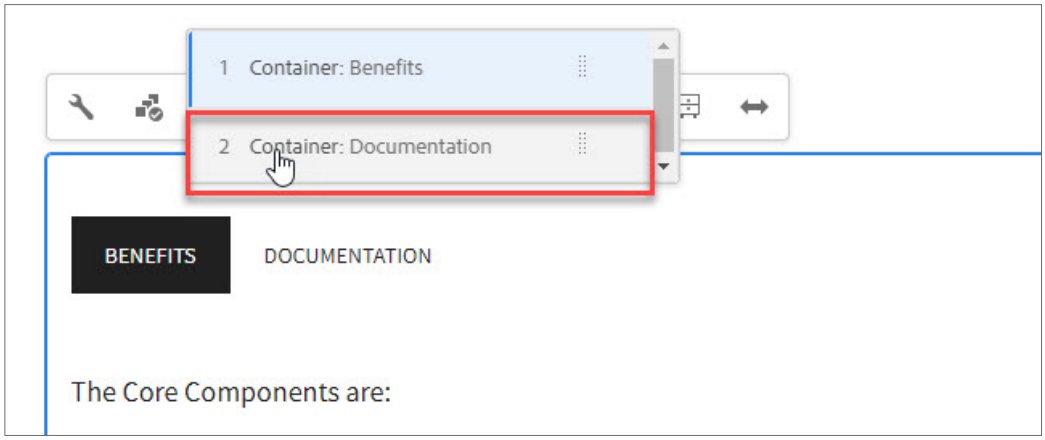
23. Click **Done** (checkmark icon) to save the changes. The **Learn More** button is added to the page. The Benefits tab should look, as shown:



24. To add components and content on the **Documentation** tab, click the **Tabs** component and click **Select Panel**, as shown:



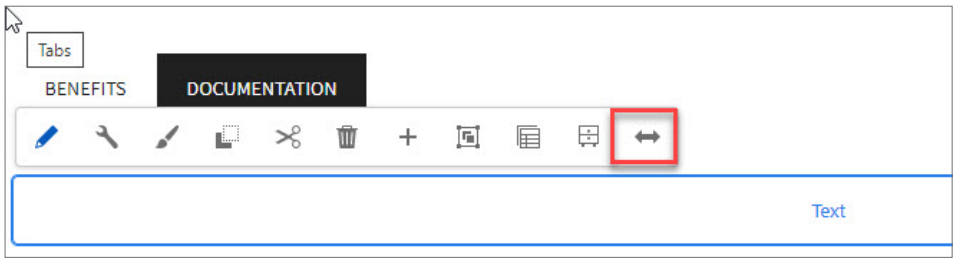
25. Click **Container: Documentation**, as shown. The components and content you insert now will be added under Documentation.



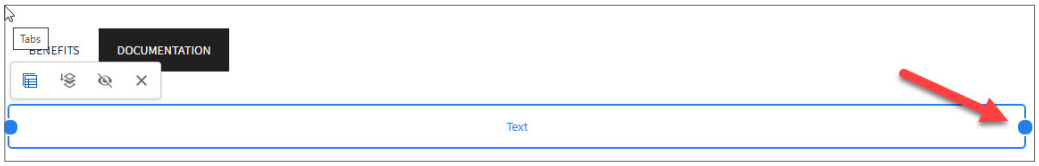
26. On the **Documentation** tab, click the **Drag components here** area.

27. Click the **Insert component** icon (the + icon), and add a **Text** component.

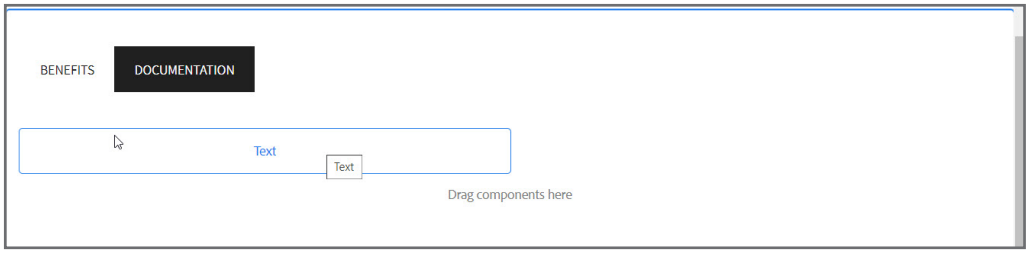
28. Click the **Text** component, and click **Layout**, as shown. The layout is now editable.



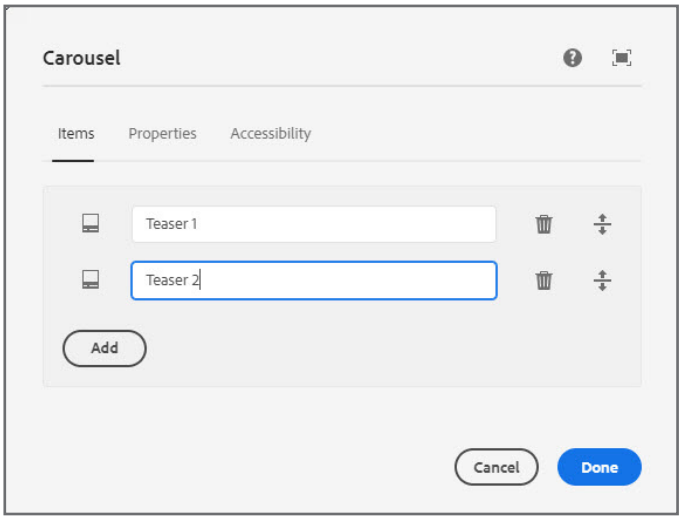
29. Drag the blue circle and adjust the layout to 6/12 columns, as shown:



30. Click **Close**. The component should look, as shown:



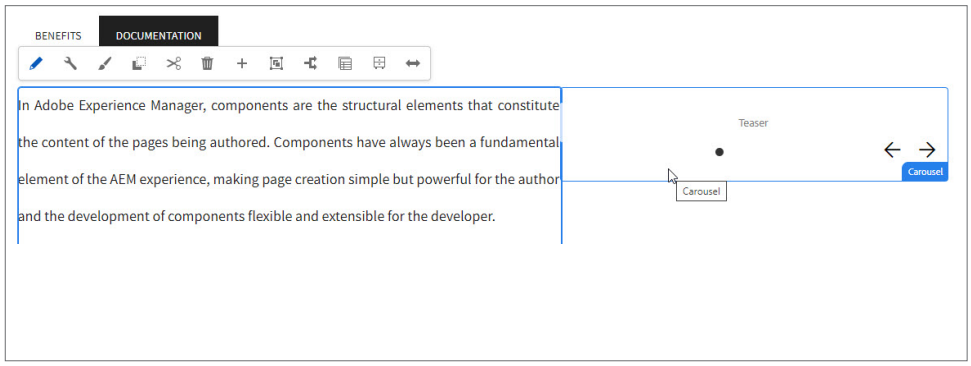
31. Navigate to the **Sites Authoring Basics** activity files provided to you and copy the content from \Activity_Files\Sites_Authoring_Basics**Documentation.txt**.
32. Click the **Text** component in AEM, and click the **Edit icon** (pencil icon) to add content.
33. Paste the content you copied in step 30 and click **Done**. The text is added under the **Documentation** tab.
34. Add a **Carousel** component under the Text component in the **Documentation** tab.
35. Select the **Carousel** component and click the configure icon.
36. Under the **Items** tab, add two Teasers.
37. Name them as **Teaser 1** and **Teaser 2** respectively, as shown:



38. Click **Done**.

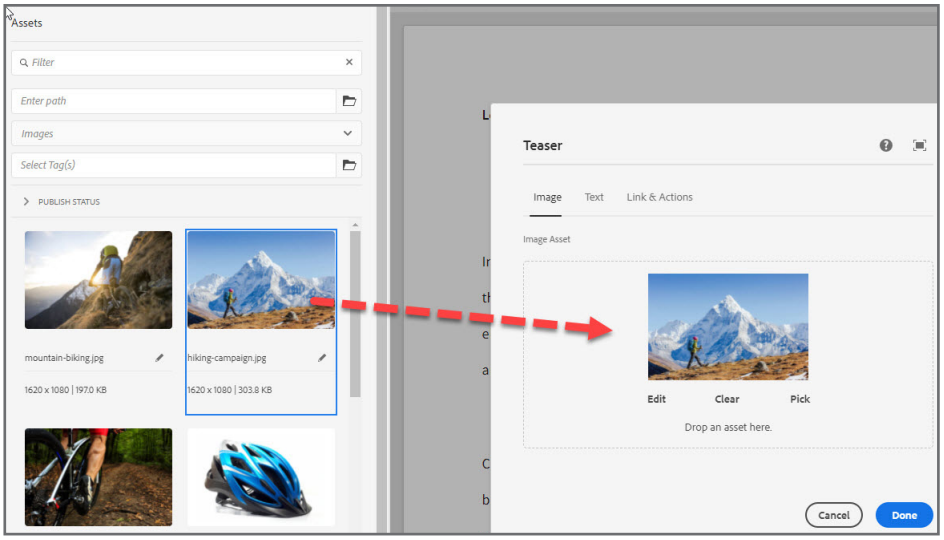
39. Click the **Carousel** component, and click **Layout**. The layout is now editable.

40. Drag the blue circle and adjust the layout to 6/12 columns. The two components are now placed next to each other, as shown:

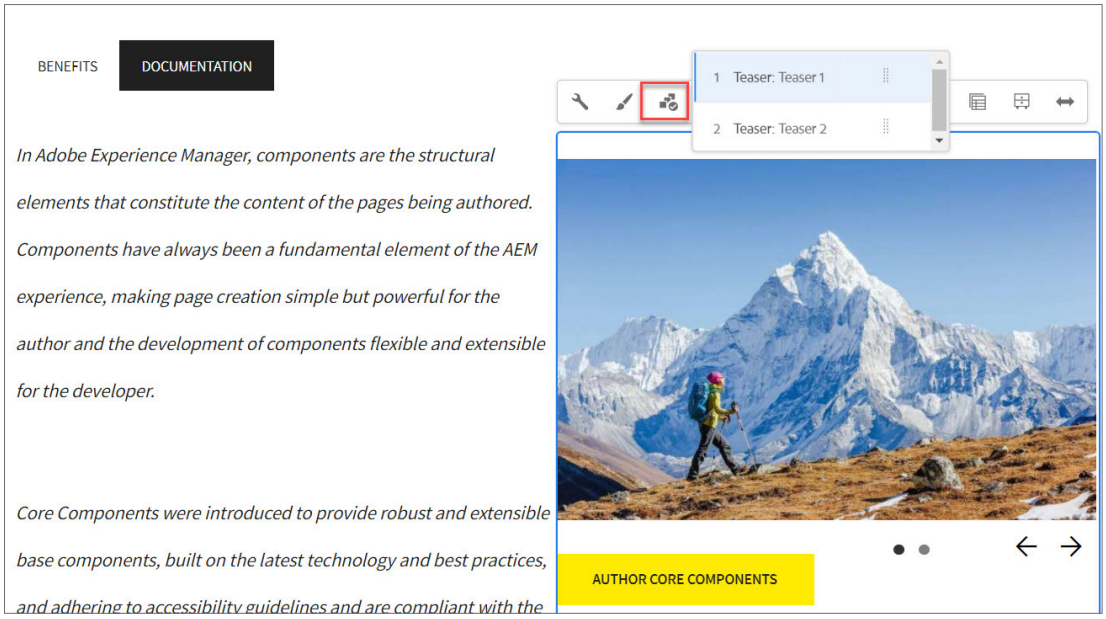


41. Select the **Teaser 1** dialog box and click **Configure**. The **Teaser** dialog box opens.

42. On the **Image** Tab, drag an image from the **Assets** tab onto the **Drop an asset here** area. The image is added.



43. On the **Text** Tab, select the **Get title from linked page** checkbox.
44. On the **Links & Actions** tab, select the **Enable Call-to-Actions** checkbox.
45. In the **Link** field, enter https://helpx.adobe.com/experience-manager/core-components/using/authoring.html
46. In the **Text** field, enter **Author Core Components**.
47. Click **Done**.
48. Select the **Carousel** component and click the **Select Panel** icon, as shown:

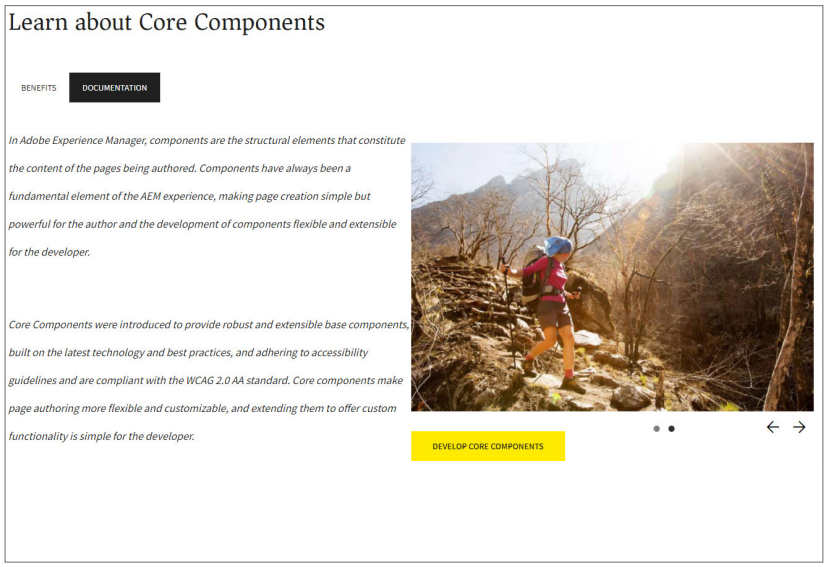


49. Click **Teaser: Teaser 2** to add content under Teaser 2.
50. Click the **Toggle Side Panel** icon to access the options in the left panel.
51. Select **Teaser 2** dialog and click the **Configure** icon. The **Teaser** dialog box opens.
52. From the **Assets** tab, drag an image onto the **Drop an asset here** area.
53. On the **Text** tab, select the **Get title from linked page** checkbox.
54. On the **Links & Actions** tab, select the **Enable Call-to-Actions** checkbox.
55. Click the **Done** icon (checkmark in the upper right).

56. In the **Link** field, type https://helpx.adobe.com/experience-manager/core-components/using/developing.html

57. In the **Text** field, type **Develop Core Components**.

58. Click **Done** to save the changes. The **Documentation** tab should look, as shown:



59. To view your completed work, click **Preview** in the upper-right corner of the page. Click all tabs and links to check if all the elements are working correctly.

60. Optional: The AEM core components also has set of form components. You can enable these on the WKND Site by going into the Template Editor and modifying the Container Policy to include **WKND.FORM** component group.

Note: You can see the completed version of this activity by uploading and installing the **core-component-authoring-in-xf.zip** content package from the activity files. This content package has an Experience Fragment called **Core Components**. You can view this fragment by adding an Experience Fragment component onto the page and dragging the Core Components fragment onto the page.

References

You can use the following links for more information on:

- Authoring Pages:
  https://docs.adobe.com/content/help/en/experience-manager-cloud-service/sites/authoring/getting-started/quick-start.html
- You can download the latest release of the Wknd site using the following link:
  https://github.com/adobe/aem-guides-wknd/releases
- Core components introduction:
  https://docs.adobe.com/content/help/en/experience-manager-core-components/using/introduction.html
- Component library:
  http://opensource.adobe.com/aem-core-wcm-components/library.html
- Core components versions:
  https://docs.adobe.com/content/help/en/experience-manager-core-components/using/versions.html
- Author with Core components:
  https://docs.adobe.com/content/help/en/experience-manager-core-components/using/get-started/authoring.html
- Using Core components:
  https://docs.adobe.com/content/help/en/experience-manager-core-components/using/get-started/using.html

Create a Project Using Maven

Introduction

Apache Maven is an open source tool for managing software projects by automating builds and providing quality project information. It is the recommended build management tool for AEM projects. This third-party tool helps a developer to comprehend the development effort easily and accurately.

AEM Archetype

AEM archetype creates a minimal Adobe Experience Manager project as a starting point for your own projects. The properties that must be provided when using this archetype allow you to name all parts of this project as required.

This project has a number of features that are intended to offer a convenient starting point for new projects:

- Best Practice: Bootstrap your site with all of Adobe's latest recommended practices.
- Low-Code: Edit your templates, create content, deploy your CSS, and your site is ready for go-live.
- Cloud-Ready: If desired, use AEM as a Cloud Service to go-live in few days and ease scalability and maintenance.
- Dispatcher: A project is complete only with a Dispatcher configuration that ensures speed and security.
- Multi-Site: If needed, the archetype generates the content structure for a multi-language and multi-region setup.
- Core Components: Authors can create nearly any layout with our versatile set of standardized components.
- Editable Templates: Assemble virtually any template without code, and define what the authors are allowed to edit.
- Responsive Layout: On templates or individual pages, define how the elements reflow for the defined breakpoints.
- Header and Footer: Assemble and localize them without code, using the localization features of the components.
- Style System: Avoid building custom components by allowing authors to apply different styles to them.
- Front-End Build: Front-end developers can mock AEM pages and build client libraries with Webpack, TypeScript, and SASS.

- WebApp-Ready: For sites using React or Angular, use the SPA SDK to retain in-context authoring of the app.
- Commerce Enabled: For projects that want to integrate AEM Commerce with commerce solutions like Magento using the Commerce Core Components.
- Example Code: Checkout the HelloWorld component, and the sample models, servlets, filters, and schedulers.
- Open Sourced: If something is not as it should, contribute your improvements!

Below are the common modules for an AEM Project:

- all - A container package that includes all artifacts of a project, including any third-party dependencies
- ui.apps - Contains all the immutable code in /apps
- core - Contains the OSGi bundle
- ui.content - Contains all mutable content and configuration
- ui.config - Contains all OSGi configurations for an application
- it.launcher - Used for deploying the sling testing framework
- it.tests - Contains sling tests
- ui.apps.structure - Module that contains project roots
- dispatcher - (required for Managed Services and Cloud Service) Module used to configure the static web server in front of AEM that caches content
- ui.frontend - (optional) Module that contains a front end project built for Webpack, React, or AngularJS and installed as an AEM client library.

AEM Project's Content Package Structure

In this section you will learn about an AEM project structure based on the AEM Archetype. This project structure is required for all modern projects in AEM. Developers should follow this packaging pattern closely when implementing their own projects.

AEM application deployments must be comprised of a single Experience Manager package. This package should in turn contain subpackages that comprise everything required by the application to function, including code, configuration, and any supporting baseline content.

Experience Manager requires a separation of content and code, which means a single content package cannot deploy to both /apps and runtime-writable areas (/content, /conf, /home, and so forth) of the repository. Instead, the application must separate code and content into discrete packages for deployment into Experience Manager.

Mutable vs. Immutable Content

/apps and /libs are considered immutable areas of Experience Manager as they cannot be changed (create, update, delete) after deployment into the AEM Cloud Service (for example, at runtime). Any attempt to change an immutable area at runtime will fail.

Everything else in the repository, /content, /conf, /var, /home, /etc, /oak:index, /system, /tmp, and so forth are all mutable areas, meaning they can be changed at runtime.

Note: /libs remains off-limits. Only Adobe product code may deploy to /libs.

Note: To learn more about project structure, see https://experienceleague.adobe.com/docs/experience-manager-cloud-service/content/implementing/developing/aem-project-content-package-structure.html?lang=en#aem-project-structure.

POM Files

POM files are XML files that contain the identity and the structure of the project, build configuration, and other dependencies. When executing a task or goal, Maven looks for the POM in the current directory. Maven reads the POM, gets the needed configuration information, and then executes the goal. A typical POM file includes:

- General project information: This section includes the project name, website URL, and the organization name. It can also include a list of developers and contributors along with the license for a project.
- Build settings: This section includes the directory settings of source and tests, plugins, plugin goals, and site-generation parameters.
- Build environment: This section includes the profiles that can be used in different environments. The build environment customizes the build settings for a specific environment and is often supplemented by a custom settings.xml file in the Maven repository. A Maven repository is a directory that stores all project files, including JAR files, plugins, artifacts, and other dependencies.

If you use a Maven Repository Manager, such as Sonatype Nexus, Apache, or JFrog Artifactory, you will need to:

- Add the configuration to reference the Maven repository manager
- Add Adobe's Maven Repository to your repository manager

Adobe provides a special jar file to reduce the number of dependencies needed for an AEM project.

This API jar file contains:

- All public Java APIs exposed by AEM
- Limited external libraries—all public APIs available in AEM, which comes from Apache Sling, Apache Jackrabbit, Apache Lucene, Google Guava, and two libraries used for image processing
- Interfaces and classes exported by an OSGi bundle in AEM
- A MANIFEST.MF file with the correct package export versions for all exported packages

AEM SDK API for Cloud Service

AEM as a Cloud Service uses a dependency called the AEM SDK API. To use the SDK jar file, you must add the following elements to the pom.xml file.
Dependency element: To add the actual dependency to your project, this code is used:

```
<dependency>
  <groupId>com.adobe.aem</groupId>
  <artifactId>aem-sdk-api</artifactId>
  <version>2020.01.1850.20200109T110957Z-191201</version>
  <scope>provided</scope>
</dependency>
```

UberJar for AEM 6.5 and earlier

AEM 6.5 and earlier uses a dependency called the UberJar. To use the UberJar file, you must add the following elements to the pom.xml file.

Dependency element: To add the actual dependency to your project, this code is used:

```
1230    <!-- Adobe AEM Dependencies -->
1231
1232    <dependency>
1233
1234      <groupId>com.adobe.aem</groupId>
1235
1236      <artifactId>uber-jar</artifactId>
1237
1238      <version>6.5.13</version>
1239
```

Note: The UberJar version should be the same as the AEM version and service pack you develop with locally.

Install to AEM with Maven Profiles

Archetype 21+ introduced two new Maven profiles. autoInstallSinglePackage and autoInstallSinglePackagePublish. Now, all Maven profiles are located:

Profiles in parent POM

- autoInstallPackage: – Used in ui.apps or ui.content module to install on author
- autoInstallPackagePublish: - Used in ui.apps or ui.content module to install on publish
- autoInstallbundle: – Used in core module to install just the bundle on author
- adobe-public

Profiles in all POM

- autoInstallSinglePackage: – Used to install ui.apps, ui.content, core, and all vendor dependencies including core components on the author service
- autoInstallSinglePackagePublish - Used to install ui.apps, ui.content, core, and all vendor dependencies including core components on the publish service

autoInstallSinglePackage is used by the cloud manager pipeline to install a customer's code base in a single content package, <your-project>-all-1.0-SNAPSHOT.zip. Installing this content package results in:

- /apps/<your-project>-packages
    - › application/install
        - » ui.apps.package.here.zip (also contains core bundle)

    - content/install
        - ui.content.package.here.zip
- /apps/<your-project>-vendor-packages (for example, Core components)
    - application/install
        - ui.apps.vendor.package.here.zip
    - content/install
        - ui.content.vendor.package.here.zip

Note: Any content package or bundle in /apps/*/install will be auto installed by package manager.

Example Maven Commands:

- Install the entire project
```
$ mvn clean install -PautoInstallSinglePackage
```
- Install the entire project to the publish server
```
$ mvn clean install -PautoInstallSinglePackagePublish
```
- Install only the java bundle
```
$ mvn clean install -PautoInstallBundle
```
- Install only immutable content (ui.apps)
```
$ cd ui.apps
$ mvn clean install -PautoInstallPackage
```

Activity 1: Create an AEM Project

In this activity you will create an AEM Project using the AEM Maven archetype. This archetype maintains best practice for AEM projects and is the recommended starting point for all new AEM projects.

Prerequisite: Maven is installed

Note: If you do not have Maven installed, see "Activity 3: Install Build Tools (Local only)" on page 117.

This activity includes the following tasks:

1. (Optional) Configure the profile for Maven.
2. Create a project using Maven.

Note: If you are using a ReadyTech Service, you can skip Task 1 as these steps have already been configured for you.

Task 1: (Optional) Configure the Profile for Maven

1. Navigate to **C:\Users\<user>\.m2** directory.
2. If you already have a **settings.xml**, make a copy of the file **settings.xml** and rename the file to **settings-orig.xml.**

3. Open **settings.xml** using a text editor and replace the Profile settings with the content, as shown:

```
<profile>
   <id>archetype</id>

   <activation>
     <activeByDefault>true</activeByDefault>
    </activation>

   <properties>

      <releaseRepository-Id>adobe-public-releases</releaseRepository-Id>
       <releaseRepository-Name>Adobe Public
        Releases</releaseRepository-Name>

       <releaseRepository-URL>http://repo.adobe.com/nexus/content/groups/public</releaseRepository-URL>
    </properties>

    <repositories>
        <repository>
            <id>adobe-public-releases</id>
            <name>Adobe Basel Public Repository</name>
            <url>http://repo.adobe.com/nexus/content/groups/public</url>
            <releases>
                <enabled>true</enabled>
                <updatePolicy>never</updatePolicy>
            </releases>
            <snapshots>
                <enabled>false</enabled>
            </snapshots>
        </repository>
    </repositories>

   <pluginRepositories>
        <pluginRepository>
            <id>adobe-public-releases</id>
            <name>Adobe Basel Public Repository</name>
            <url>http://repo.adobe.com/nexus/content/groups/public</url>
            <releases>
                <enabled>true</enabled>
                <updatePolicy>never</updatePolicy>
            </releases>
            <snapshots>
                <enabled>false</enabled>
            </snapshots>
        </pluginRepository>
    </pluginRepositories>
</profile>
```

4. Save the changes.

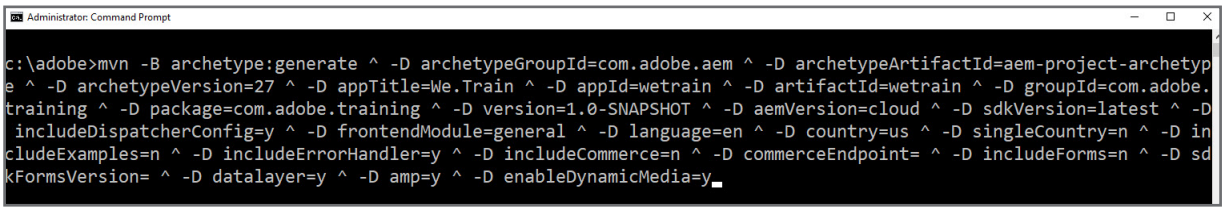
Note: If you do not have a **settings.xml** file, download the file from https://experienceleague.adobe.com/docs/experience-cloud-kcs/kbarticles/KA-17454.html?lang=en and place it in the user's .m2 directory.

Task 2: Create an AEM Project

1. Open a **Command Prompt** window.
2. In your Command Prompt window, navigate to the **adobe** directory which will be your working directory for this training.
3. Navigate to **Activity_Files\training-files\archetype-scripts** and copy the content from the file specified by your instructor. For example: **generate-archetypexx-xxxxxx-cloud.bat** on Windows (For Cloud Service on Mac, use **generate-archetypexx-xxxxxx-cloud.sh**. For 6.5, use **generate-archetypexx-xxxxxx-6.5.bat** on Windows and **generate-archetypexx-xxxxxx-6.5.sh** on Mac):

```
mvn -B org.apache.maven.plugins:maven-archetype-plugin:3.2.1:generate ^
 -D archetypeGroupId=com.adobe.aem ^
 -D archetypeArtifactId=aem-project-archetype ^
 -D archetypeVersion=39 ^
 -D appTitle=We.Train ^
 -D appId=wetrain ^
 -D artifactId=wetrain ^
 -D groupId=com.adobe.training ^
 -D package=com.adobe.training ^
 -D version=1.0-SNAPSHOT ^
 -D aemVersion=cloud ^
 -D sdkVersion=2022.10.9398.20221020T071514Z-220800 ^
 -D includeDispatcherConfig=n ^
 -D frontendModule=general ^
 -D language=en ^
 -D country=us ^
 -D singleCountry=n ^
 -D includeExamples=n ^
 -D includeErrorHandler=y ^
 -D includeCommerce=n ^
 -D commerceEndpoint= ^
 -D includeForms=n ^
 -D includeFormscommunications=n ^
 -D includeFormsenrollment=n ^
 -D sdkFormsVersion= ^
 -D datalayer=y ^
 -D amp=n ^
 -D enableDynamicMedia=y ^
 -D enableSSR=n ^
 -D precompiledScripts=n ^
 -D includeFormsheadless=n
```

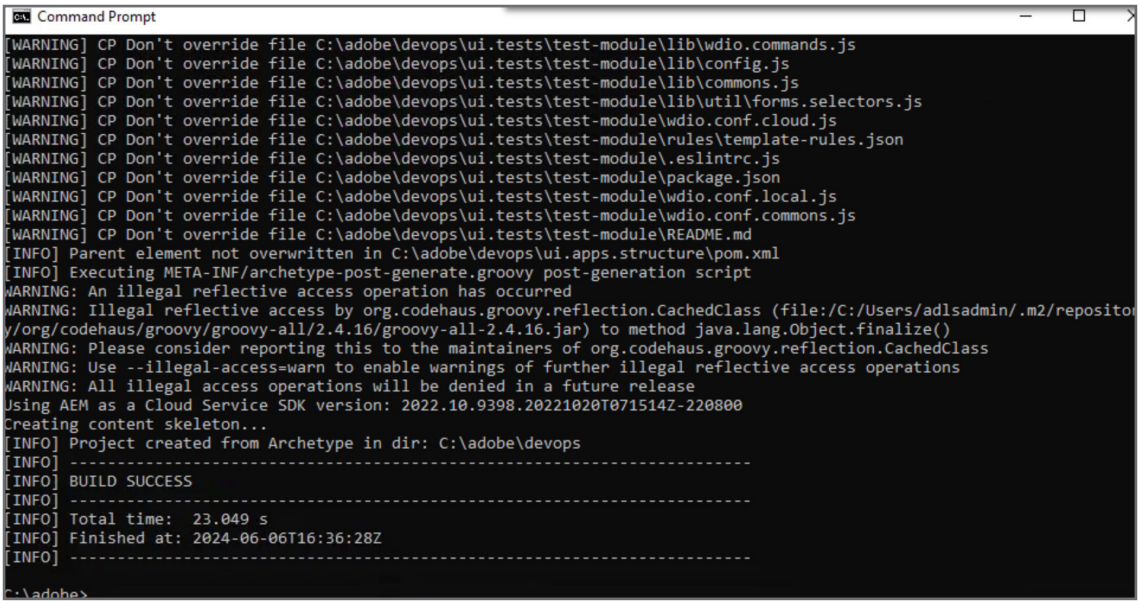
4. Paste the contents in the Command Prompt Window and press <Enter>:



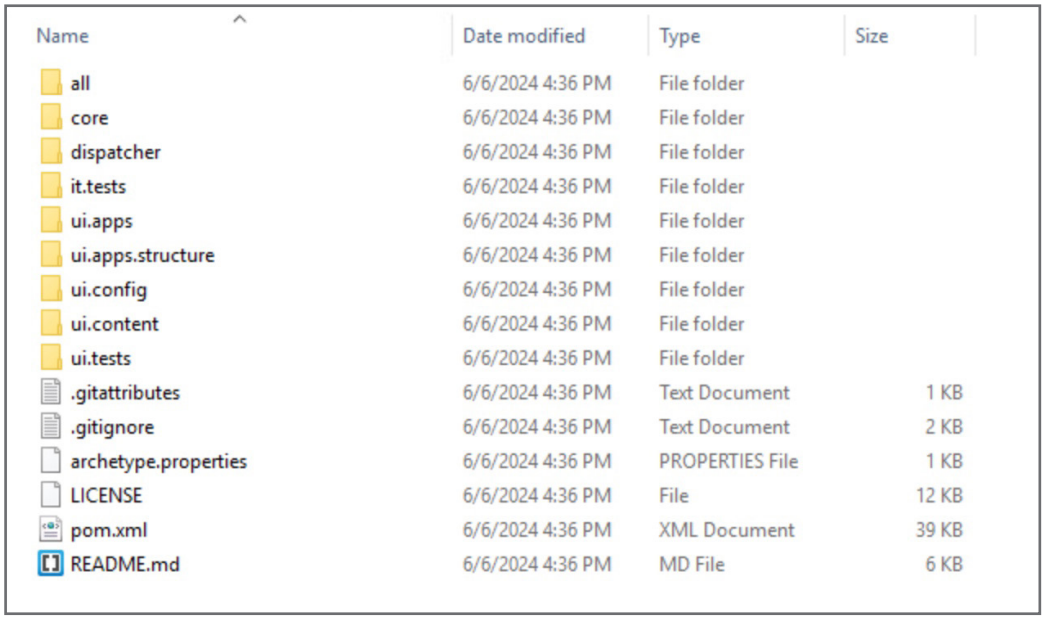
The command runs, which may take a few minutes. The project starts to build.

Note: If you are prompted to confirm the parameters type **y** and press <Enter>.

5. Verify Build Success, as shown:



6. Navigate to the project folder you just created …\adobe\<AEM Project>, as shown:



Name	Date modified	Type	Size
all	6/6/2024 4:36 PM	File folder	
core	6/6/2024 4:36 PM	File folder	
dispatcher	6/6/2024 4:36 PM	File folder	
it.tests	6/6/2024 4:36 PM	File folder	
ui.apps	6/6/2024 4:36 PM	File folder	
ui.apps.structure	6/6/2024 4:36 PM	File folder	
ui.config	6/6/2024 4:36 PM	File folder	
ui.content	6/6/2024 4:36 PM	File folder	
ui.tests	6/6/2024 4:36 PM	File folder	
.gitattributes	6/6/2024 4:36 PM	Text Document	1 KB
.gitignore	6/6/2024 4:36 PM	Text Document	2 KB
archetype.properties	6/6/2024 4:36 PM	PROPERTIES File	1 KB
LICENSE	6/6/2024 4:36 PM	File	12 KB
pom.xml	6/6/2024 4:36 PM	XML Document	39 KB
README.md	6/6/2024 4:36 PM	MD File	6 KB

You have successfully created an AEM project.

Activity 2: Install the Project into AEM

In this activity, you will install the project that you created in the previous activity into AEM.

Prerequisite: AEM author service is up and running

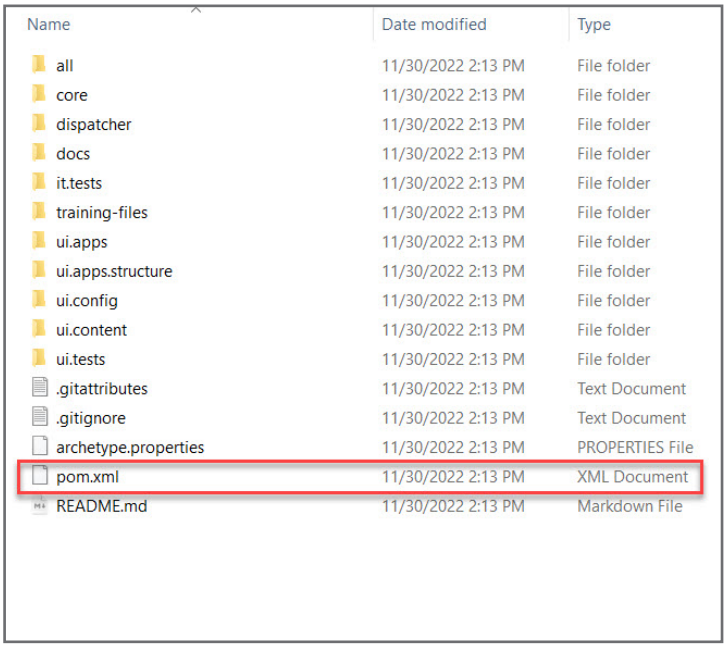
This activity includes the following tasks:

1. (Cloud Service only) Update dependencies.
2. (6.5 only) Update dependencies.
3. Install your project into AEM.
4. Verify the installed content packages.

Task 1: (Cloud Service Only) Update Dependencies

When working with any AEM Maven project, dependency management is critical to begin development. Your project needs to reflect the same dependencies and plugins that your development environment supports. Typically, most dependencies are initially setup by the Project Owner, Team Lead, Consultants or other SMEs and stay largely unchanged throughout the life of the project. Some dependencies though, need to be updated when developing new features and functions. In this task, you will investigate and optionally update common dependencies and plugins for Cloud Service.

1. Navigate to your newly created Maven project **<AEM project>** and open the parent **pom.xml** in a text editor (such as Notepad++):



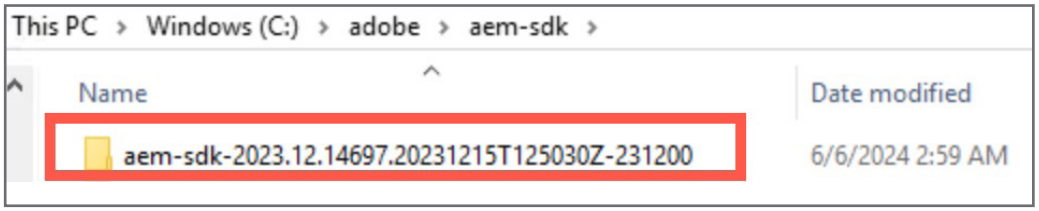
aem-sdk-api Version:

When you create a new project from the archetype, it sets the aem-sdk-api version as a property for you. When developing your application, you want to make sure the aem-sdk-api in your POM matches the SDK build you're developing with. This is important because the aem-sdk-api is a dependency that contains all dependencies used within AEM. Using the same aem-sdk-api that you're developing with ensures there won't be any dependency collisions or dependencies missing from your project. In general, to develop locally, you would:

- Download the latest (or desired) version of the AEM SDK from the Software Distribution portal
- Copy the SDK version. Ex: 2022.10.9398.20221020T071514Z-220800
- Open your Parent pom and paste the version into the <aem.sdk.api> property
- Develop normally

Verify/update the aem-sdk-api version:

1. Navigate to the location of the AEM SDK quickstart jar. In **ReadyTech** navigate to **C:/adobe/aem-sdk/**.



2. In your opened parent POM, find the property value <aem.sdk.api> (near the top) and verify (or paste) the build number that you copied from the SDK, as shown:

```
<properties>
  <aem.host>localhost</aem.host>
  <aem.port>4502</aem.port>
  <aem.publish.host>localhost</aem.publish.host>
  <aem.publish.port>4503</aem.publish.port>
  <sling.user>admin</sling.user>
  <sling.password>admin</sling.password>
  <vault.user>admin</vault.user>
  <vault.password>admin</vault.password>
  <frontend-maven-plugin.version>1.12.0</frontend-maven-plugin.version>
  <core.wcm.components.version>2.19.0</core.wcm.components.version>
  <bnd.version>5.1.2</bnd.version>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
  <aem.sdk.api>2022.10.9398.20221020T071514Z-220800</aem.sdk.api>
  <aemanalyser.version>1.4.10</aemanalyser.version>
  <componentGroupName>DevOps Project</componentGroupName>
</properties>
```

Note: To learn more about features and functions added with each release you can check the monthly Release Notes.

aemanalyser-maven-plugin Version:
The analyser plugin does a series of checks on a maven project to ensure it is Cloud Service compatible. Keeping up with the latest plugin is beneficial to ensure your project will continue to pass through the CICD process of Cloud Manager.

Verify/update the aemanalyser-maven-plugin version:

1. Find the property value <aemanalyser.version> and verify the version is greater than 1.3.0. If it is not, update it to **1.4.10**.

```
81
82      <aem.sdk.api>2022.10.9398.20221020T071514Z-220800</aem.sdk.api>
83
84      <aemanalyser.version>1.4.10</aemanalyser.version>
85
86      <componentGroupName>We.Train</componentGroupName>
```

2. This property variable is for the AEM Analyser plugin that runs on every module each time your project is built.

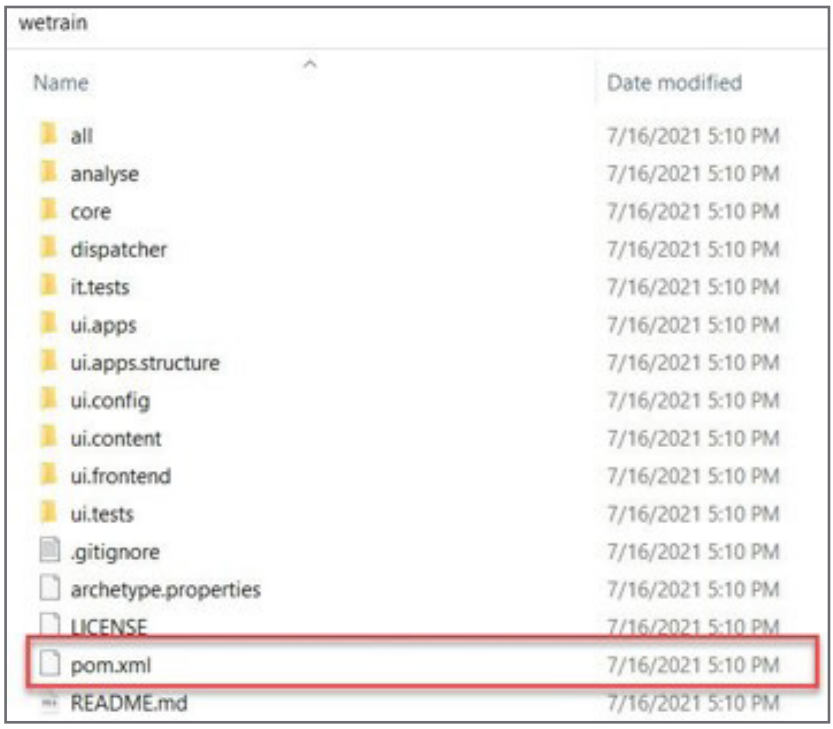
```
549             <!-- AEM Analyser Plugin -->
550
551             <plugin>
552
553               <groupId>com.adobe.aem</groupId>
554
555               <artifactId>aemanalyser-maven-plugin</artifactId>
556
557               <version>${aemanalyser.version}</version>
558
559               <extensions>true</extensions>
560
561             </plugin>
```

3. Save the file.
4. Continue on to Task 3 to install your project.

Task 2: (AEM 6.5 Only) Update Dependencies

When working with any AEM maven project, dependency management is critical to begin development. Your project needs to reflect the same dependencies and plugins that your development environment supports. Typically, most dependencies are initially setup by the Project Owner, Team Lead, Consultants or other SMEs and stay largely unchanged throughout the life of the project. Some dependencies though, need to be updated when developing new features and functions. In this task you will investigate and optionally update common dependencies and plugins for 6.5.x versions.

1. Navigate to your newly created Maven project <AEM project> and open the parent pom.xml in a text editor (such as Notepad++):



uber-jar Version:

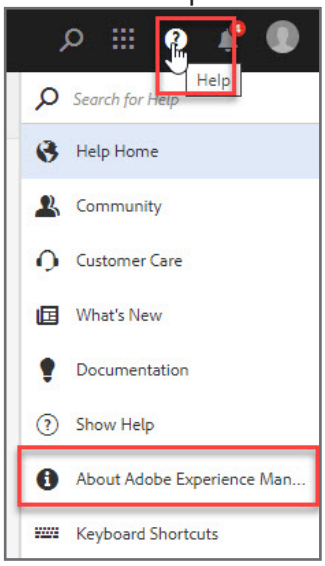
When you create a new project from the archetype, it sets the uber-jar version as a property for you. When developing your application, you want to make sure the uber-jar in your POM matches the 6.5 Service Pack you're developing with. This is important because the uber-jar is a dependency that

contains all dependencies used within AEM. Using the same uber-jar that you're developing with ensures there won't be any dependency collisions or dependencies missing from your project. In general, to develop locally, you would:

- Download the latest (or desired) Service Pack of the AEM 6.5 from the Software Distribution portal
- Open your Parent pom and update the uber-jar dependency with the Service Pack. Ex 6.5.13
- Develop normally

Update Uber Jar:

1. If you are unsure about the version of AEM Service Pack you are using, open AEM in the browser: http://localhost:4502/
2. Click on the Help icon on the top right corner > About Adobe Experience Manager



3. Take note of the AEM version:

4. Navigate to your newly created maven project and open the **parent pom.xml** in a text editor (such as Notepad++).
5. Find the dependency with the **<artifactId>uber-jar</artifactId>** (near line 646) and update the <version> with the correct service pack number, as shown:

Note: Based on the Release Notes for 6.5.14, you should continue using 6.5.13 for the uber-jar since no dependencies changed. Attempting to find a 6.5.14 uber-jar in the maven repository will result in error.

core.wcm.components Version

AEM 6.5.0 initially installs a version of the Core Components released in 2019. Since Core Components are open source, it is up to developers to update the core components for AEM 6.5 projects. Service Packs do not update Core Components. AEM 6.5 projects must include Core Components artifacts to install the desired version needed for the project.

Note: Cloud Service projects do not need to specify the core component version since Cloud Service builds always have the latest released Core Components version.

Verify/update the core.wcm.components Version

Our training project relies on the WKND reference website. WKND 6.5 (classic) installs a specific version of the Core Components, and our project should use the same version:

- WKND 1.0.0 uses Core Components 2.16.4
- WKND 1.1.0 uses Core Components 2.17.2
- WKND 2.0.0-2.12 uses Core Components 2.19.2

1. Navigate to the parent POM and find the property <core.wcm.components.version> and update it to the correct Core Components version that correlates with the WKND version you are using.

```
<properties>
  <aem.host>localhost</aem.host>
  <aem.port>4502</aem.port>
  <aem.publish.host>localhost</aem.publish.host>
  <aem.publish.port>4503</aem.publish.port>
  <sling.user>admin</sling.user>
  <sling.password>admin</sling.password>
  <vault.user>admin</vault.user>
  <vault.password>admin</vault.password>
  <frontend-maven-plugin.version>1.12.0</frontend-maven-plugin.version>
  <core.wcm.components.version>2.19.0</core.wcm.components.version>
  <bnd.version>5.1.2</bnd.version>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
  <aem.sdk.api>2022.10.9398.20221020T071514Z-220800</aem.sdk.api>
  <aemanalyser.version>1.4.10</aemanalyser.version>
  <componentGroupName>DevOps Project</componentGroupName>
</properties>
```

2. Once the version is set as a property, Core Components are installed as embedded third-party artifacts in the all/POM.

3. Open the **all/pom.xml** and scroll down to the dependencies section and find the core component artifacts using the version property from the parent pom.

```
208         <dependency>
209             <groupId>com.adobe.cq</groupId>
210             <artifactId>core.wcm.components.content</artifactId>
211             <type>zip</type>
212         </dependency>
213         <dependency>
214             <groupId>com.adobe.cq</groupId>
215             <artifactId>core.wcm.components.config</artifactId>
216             <type>zip</type>
217         </dependency>
```

4. Near the top of the **all/pom.xml** file, you will find the embedded section that includes the core component artifacts:

```
80              <embedded>
81                  <groupId>com.adobe.cq</groupId>
82                  <artifactId>core.wcm.components.content</artifactId>
83                  <type>zip</type>
84                  <target>/apps/wetrain-vendor-packages/application/install</target>
85              </embedded>
86              <embedded>
87                  <groupId>com.adobe.cq</groupId>
88                  <artifactId>core.wcm.components.core</artifactId>
89                  <target>/apps/wetrain-vendor-packages/application/install</target>
90              </embedded>
91              <embedded>
92                  <groupId>com.adobe.cq</groupId>
93                  <artifactId>core.wcm.components.config</artifactId>
94                  <type>zip</type>
95                  <target>/apps/wetrain-vendor-packages/application/install</target>
96              </embedded>
```

5. Continue on to Task 3 to install your project.

Task 3: Install Your Project into AEM

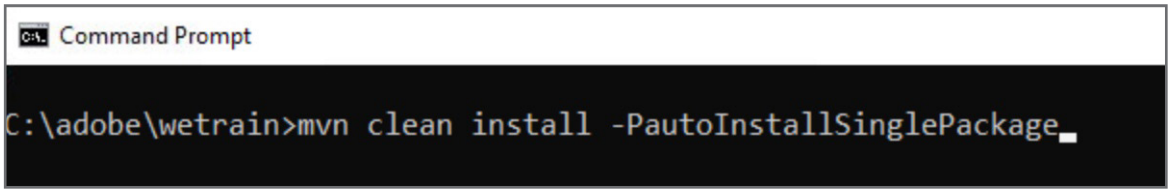
Install via Maven:

1. Open a Command Prompt window and navigate to your newly created Maven project <AEM Project>.

Note: This is the project directory <AEM Project> that you created in the previous activity.

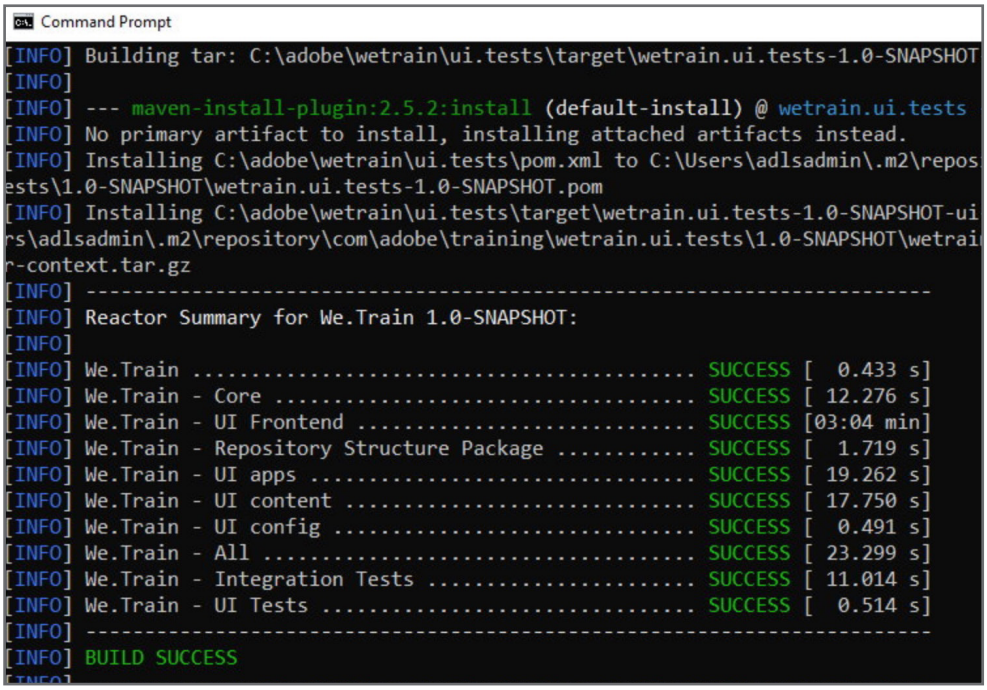
2. Run the following command and press **Enter**. The project starts to build.

```
mvn clean install -PautoInstallSinglePackage
```



Note: The project build may take a few minutes.

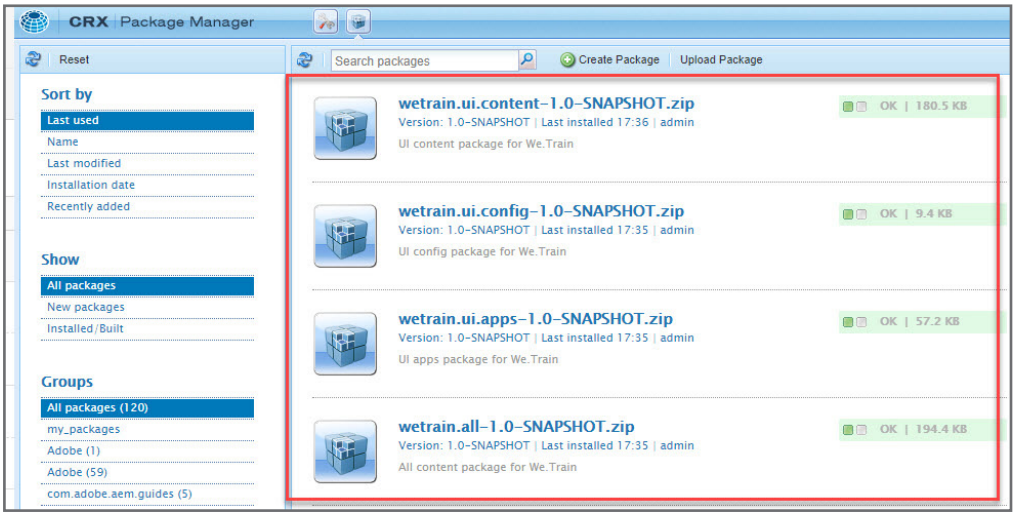
3. Verify the build is success, as shown:



Task 4: Verify the Installed Content Packages

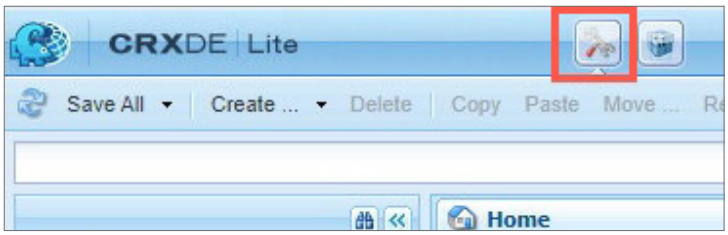
In this task, you will navigate to CRXDE Lite and verify the newly installed content package.

1. Verify you are logged on to your AEM author service on port 4502 (http://localhost:4502).
2. Navigate to **Tools** > **Deployment** > **Packages** in your AEM author service. The **Package Manager** opens.
3. Verify the three newly installed <AEM Project> packages, as shown:



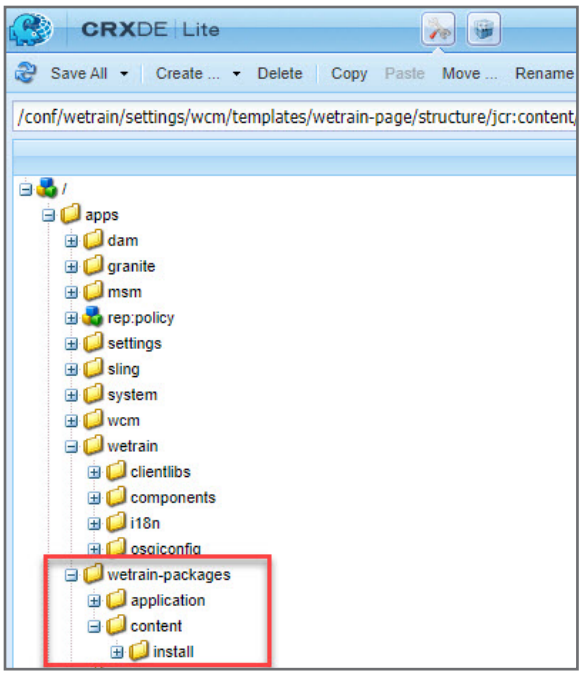
Note: The profile autoInstallSinglePackage installed the container package all. This all package then installed all of the mutable and immutable content packages.

4. Click the Develop icon as shown:

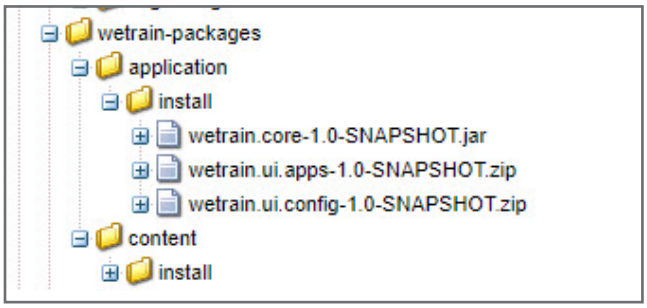


This takes you back to **CRXDE Lite**.

5. Navigate to the **/apps** folder and verify the content packages, as shown:



6. Navigate to the **/apps/<AEM Project-packages>/application/install** folder and notice the install of **<AEM Project>.ui.apps-1.0-SNAPSHOT.zip** which is the mutable content, as shown:

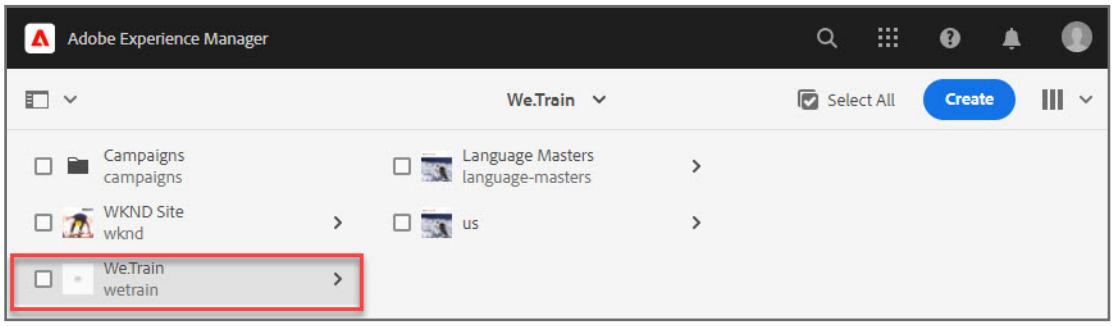


7. Similarly, navigate to the **/apps/<AEM Project>-packages/content/install** folder and notice the install of the **<AEM Project>.ui.content-1.0-SNAPSHOT.zip** which is the immutable content.

Note: Modern AEM projects organize the content into mutable and immutable content. Mutable contents are content that can typically change at runtime (/content,/conf/,/var, etc.). Immutable content can only be updated via the CICD pipeline (/apps, /libs).

---

8. Go back to the browser tab that is open with the author service.
9. Click **Adobe Experience Manager** in the upper left.
10. In the Navigation page area, click **Sites**. The column view is displayed.
11. In the column view, verify the new **<AEM Project>** site. Navigate to **<AEM Project> > us**, as shown.



You have successfully installed your project into AEM.

Activity 3: Install Build Tools (Local Only)

Additional Learning

In this activity, you will install and configure Maven.
This activity includes the following tasks:

1. Install Maven.
2. Configure Environment Variables.
3. Install Node Version Manager (NVM).

Note: You can skip this activity if you use a ReadyTech environment because Maven is already installed as part of the image.

Task 1: Install Maven 3.6.x

Windows

1. Download Maven directly from their website: https://maven.apache.org/download.cgi
2. Navigate to the directory where you extracted the contents of the Maven installation zip file. For example, navigate to **C:\Program Files\apache-maven-3.x.x** on Windows.

Mac

1. Open a terminal window and type **mvn -version**. If you get "mvn: command not found" then you need to install Maven.
2. To install Maven, follow this online tutorial https://crunchify.com/how-to-install-maven-on-mac-os-x-manually-fix-unsupportedclassversionerror-orgapachemavenclimavencli/
   or follow the generic steps below.
3. Open a a terminal window and type the following command, which will download Maven to /Users/<user>/.
   ```
   wget http://mirrors.koehn.com/apache/maven/maven-3/3.6.1/binaries/
    apache-maven-3.6.1-bin.zip
   ```
4. Type:
   ```
   unzip apache-maven-3.6.1-bin.zip
   ```

5. You should now have a **/Users/<user>/apache-maven-3.6.1** directory. Now add Maven to your bash profile:

```
sudo vi ~/.bash_profile
```

6. Add the following two lines and save the file:

```
export M2_Home=/Users/<user>/apache-maven-3.6.1

export PATH=$PATH:$M2_HOME/bin
```

7. Reload your bash_profile:

```
source ~/.bash_profile
```

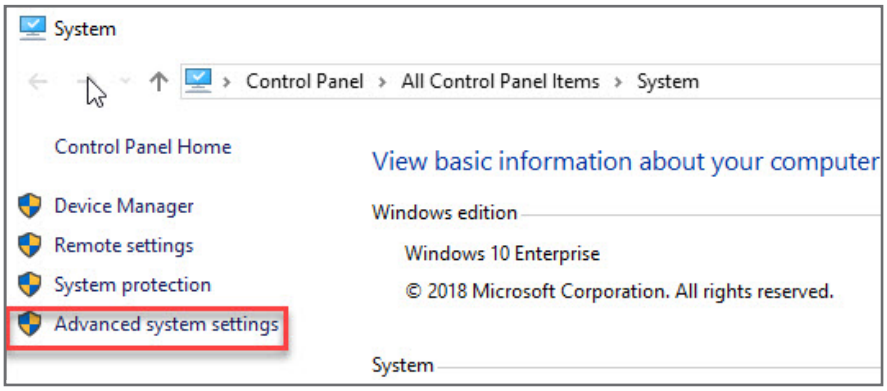
8. Test to see if you can successfully run Maven:

```
mvn -version
```

Task 2: Configure Environment Variables (Windows Only)

In this task, you will configure the environment variable.

1. In Windows, go to **Advanced System Settings,** as shown:

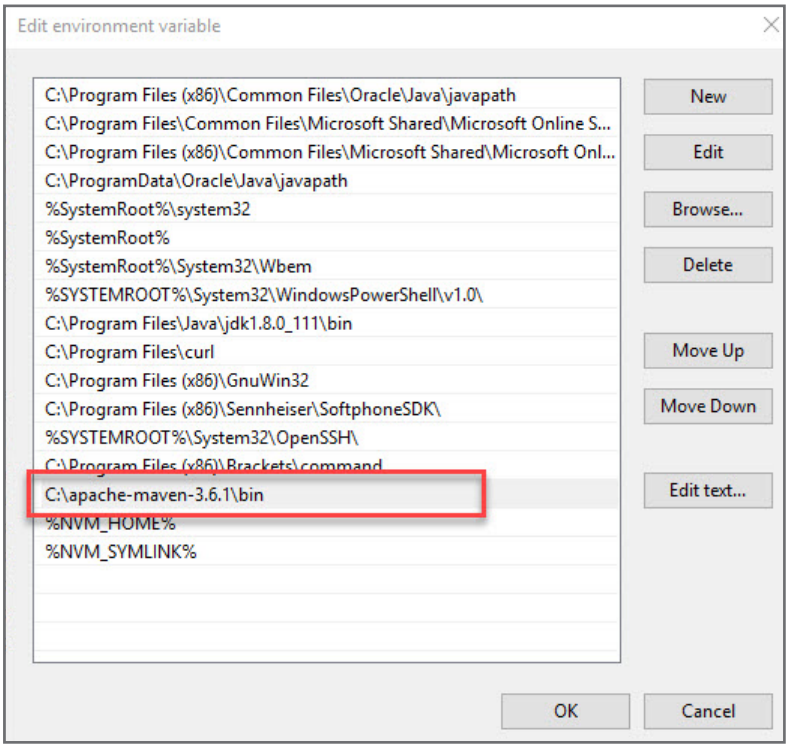


The **System Properties** window opens.

2. Click **Environment Variables** and select **Path** under **System variables**.

Note: To set up Maven on Mac, go to https://www.mkyong.com/maven/install-maven-on-mac-osx/.

3. Click **Edit** and add the Maven directory to the **PATH** variable, as shown:



4. Click **OK** to close the **Edit environment variable** window.
5. Click **OK** (again) to close the **Environment Variables** window.
6. Open a command prompt on Windows and type **mvn -version** to ensure Maven is set up properly.

```
C:\Users\prpurush>mvn -version
Apache Maven 3.6.1 (d66c9c0b3152b2e69ee9bac180bb8fcc8e6af555; 2019-04-04
Maven home: C:\apache-maven-3.6.1\bin\..
Java version: 1.8.0_111, vendor: Oracle Corporation, runtime: C:\Program
Default locale: en_US, platform encoding: Cp1252
OS name: "windows 10", version: "10.0", arch: "amd64", family: "windows"
```

Congratulations! You have set up Maven successfully on your machine.

Task 3: Install Node Version Manager (NVM)

In this task, you will install Node.js and NVM.

Note: If you have already installed NVM and Node.js, skip this task.

1. Download the latest Long Term Support (LTS) installer, for your operating system, from the node.js org site (or obtain the installer files from your instructor).
   https://nodejs.org/en/download/

2. Install Node.js.

- **Windows**
  - Double-click on the .msi file (example node-v10.16.3-x86.msi) to start the installation and follow the instructions in the installer.
- **Mac OSX**
  - Double-click on the .pkg file (example node-v10.16.3.pkg) to start the installation and follow the instructions in the installer.

Congratulations! You have installed Node.js and NPM and are now ready to install aemfed.

References

You can use the following links for more information on:

- Development Tools for AEM Projects
- AEM Archetype
- Angular frontend module
- Commerce Integration Framework (CIF) core components
- AEM Forms add-on
- Adobe Data layer
- AMP support
- Core components enabled with Dynamic Media
- Example Core Component Library

Develop with Visual Studio Code

Introduction

Visual Studio Code is a great choice for front-end developers who will primarily be writing CSS/LESS and JavaScript code to create AEM client libraries.

VSCode AEM Sync is an extension for Visual Studio Code to automatically sync changes to files in the Adobe Experience Manager server. It is used for local development.

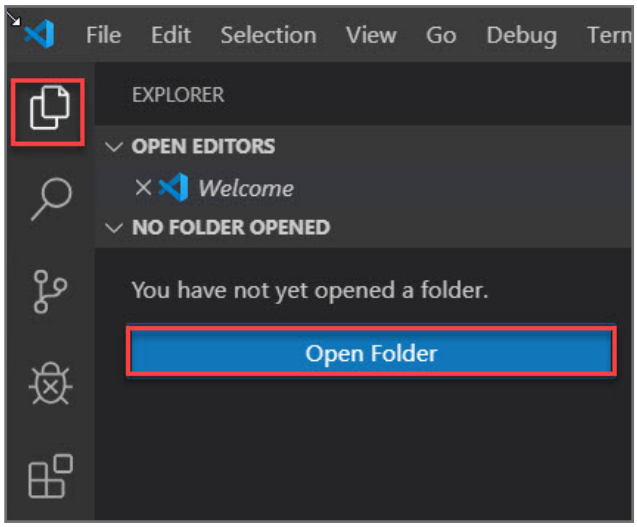
Activity 1: Import an AEM Project

In this activity, you will import an AEM project into Visual Studio Code.

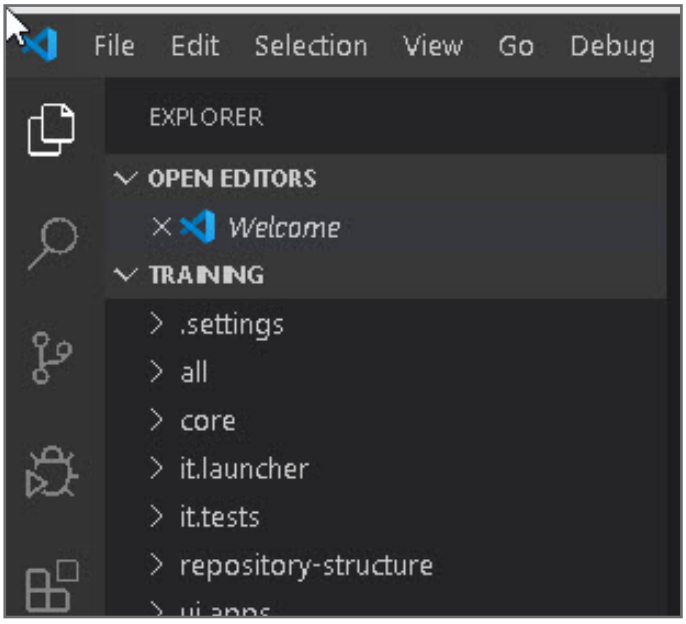
Note: This activity assumes you have a local Maven project. <AEM Project Directory> represents the location of this local Maven project.

Note: You can skip this task if you have already installed your AEM project.

1. Confirm that your AEM service is running.
2. In **Visual Studio Code**, click the Explorer icon on the left (OR you can press Ctrl+Shift+E) and click **Open Folder**, as shown:



3. Navigate to **<AEM Project Directory>** and click **Select Folder** to open. The folder opens in **Visual Studio Code**, as shown:

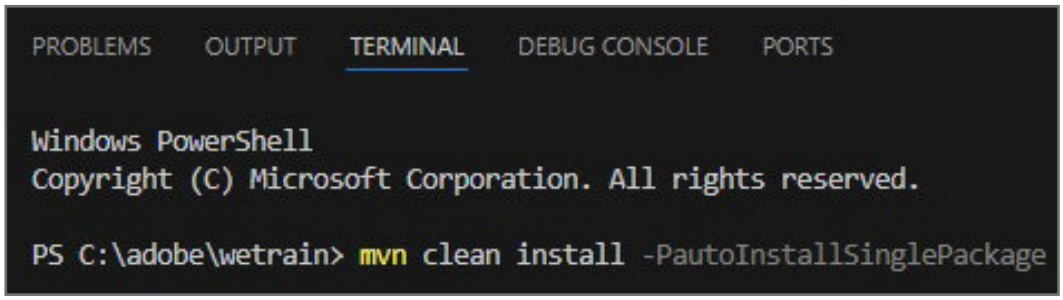


To use the Integrated Terminal:

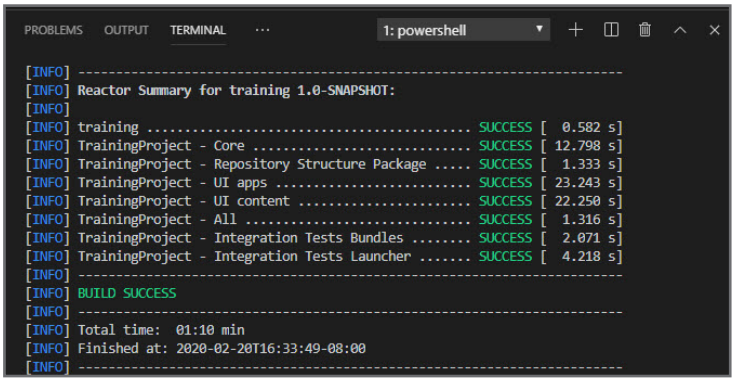
4. Select the **Terminal** tab in the window at the bottom, as shown.

5. You can deploy your AEM project from the Integrated Terminal, as shown:

```
$mvn clean install -PautoInstallSinglePackage
```



6. Verify the install is successful, as shown:



Congratulations! You have successfully imported an AEM project.

Additional Learning

Visual Studio Code

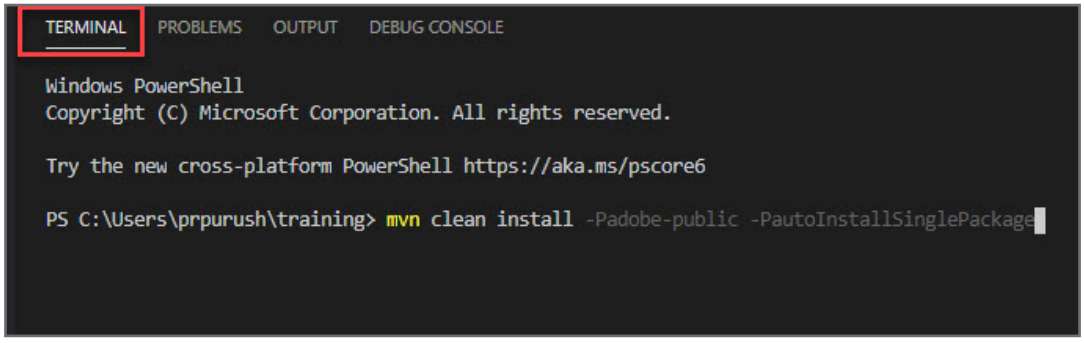
Visual Studio Code is a free source-code editor developed by Microsoft for Windows, Linux and macOS. It comes with built-in support for JavaScript, TypeScript and Node.js and has a rich ecosystem of extensions for other languages (such as C++, C#, Java, Python, PHP, and Go) and runtimes (such as .NET and Unity). Visual Studio Code also includes support for debugging, embedded Git control and GitHub, syntax highlighting, intelligent code completion, snippets, and code refactoring.

VS Integrated Terminal:

In Visual Studio Code, you can open an integrated terminal, initially starting at the root of your workspace. This can be convenient as you do not have to switch windows or alter the state of an existing terminal to perform a quick command-line task.

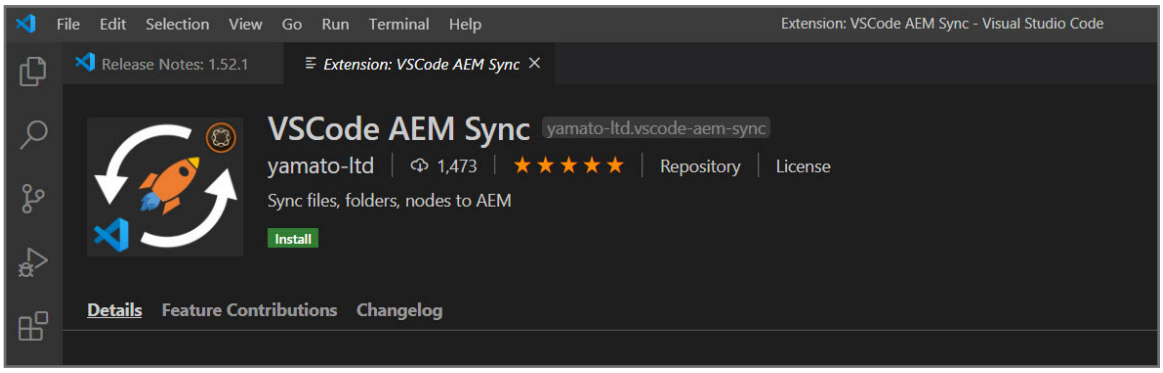
To open the terminal:

- Use the **Ctrl+`** keyboard shortcut with the backtick character.
- Use the **View** > **Termina**l menu command.
- From the Command Palette (Ctrl+Shift+P), use the View: Toggle Integrated Terminal command.



VSCode AEM Sync

An extension for Visual Studio Code that allows AEM Developer to sync their code updates smoothly. You can sync files, and also .content.xml, dialog.xml, etc. with one click. This sync feature is ported from AEM Brackets Extension.

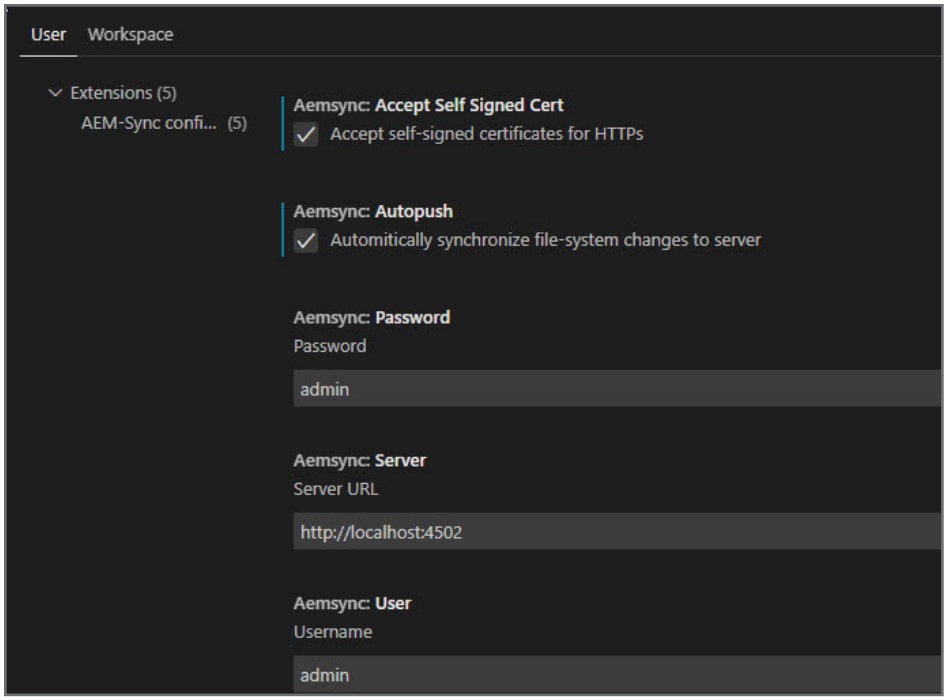


Settings

The menu command File > Preferences > Settings (Code > Preferences > Settings on Mac) provides entry to configure user and workspace settings.
The following settings can be configured:

- aemsync.server - Server URL
- aemsync.user - Username
- aemsync.password - Password
- aemsync.autopush - Automatically synchronize file-system changes to server
- aemsync.acceptSelfSignedCert - Accept self-signed certificates for HTTPs

If auto-push feature is enabled, Export to AEM Server is executed automatically, when you save.



Activity 2: Install Visual Studio Code (Local Only)

In this activity, you will install Visual Studio Code.

Note: You can skip this activity if you use a ReadyTech environment because AEM is already installed as part of the image.

1. Download the latest VS Code installer, for your operating system using the link below: https://code.visualstudio.com/download/

Note: If you do not have the ability to download Visual Studio from the Internet, you can use the installer in the distribution-files folder provided to you by your instructor.

2. Install Visual Studio Code.

- **Windows**
  - Double-click on the .exe file, for example VSCodeUserSetup-ia32-1.38.1.exe, to start the installation and follow the instructions in the installer.
- **Mac OSX**
  - Double-click on the .zip file, for example VSCode-darwin-stable.zip, to start the installation and follow the instructions in the installer.

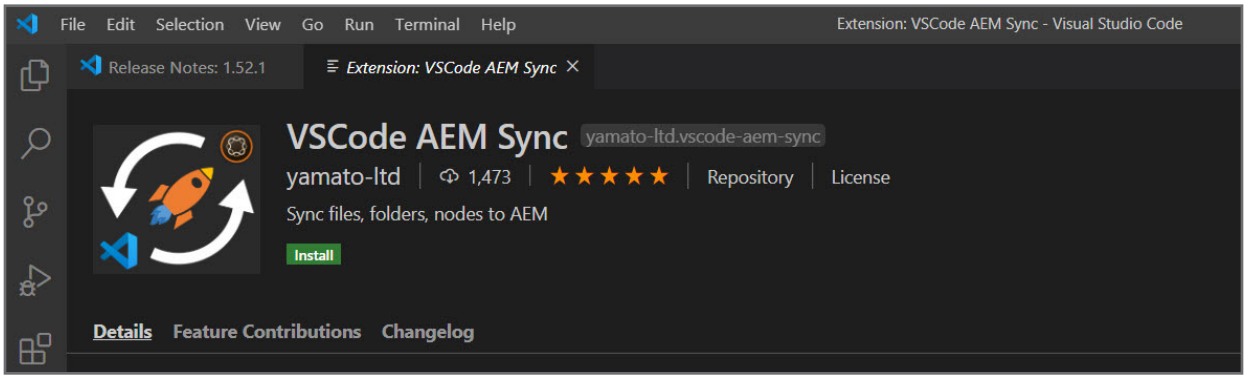
Congratulations! You have installed Visual Studio Code and are now ready to install the VSCode AEM Sync extension.

Activity 3: Install VSCode AEM Sync (Local Only)

In this activity, you will install VSCode AEM Sync.

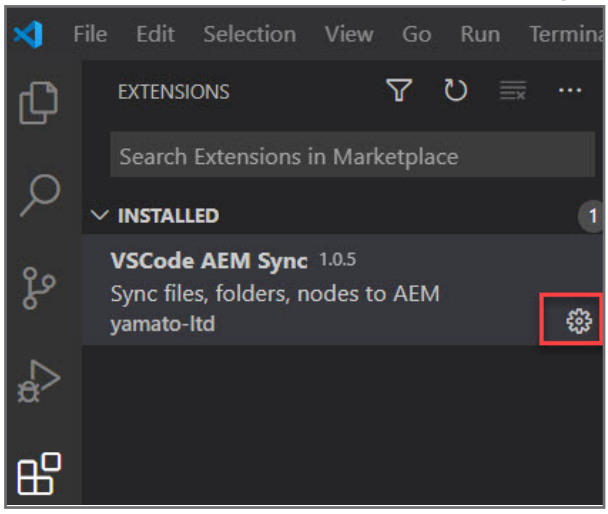
Note: You can skip this activity if you use a ReadyTech environment because AEM is already installed as part of the image.

1. Open Visual Studio Code.
2. In a browser, go to https://marketplace.visualstudio.com/items?itemName=Yinkai15.aemsync
3. Click the **Install** button. **VSCode AEM Sync** Readme will open in the Visual Studio Code window, as shown:

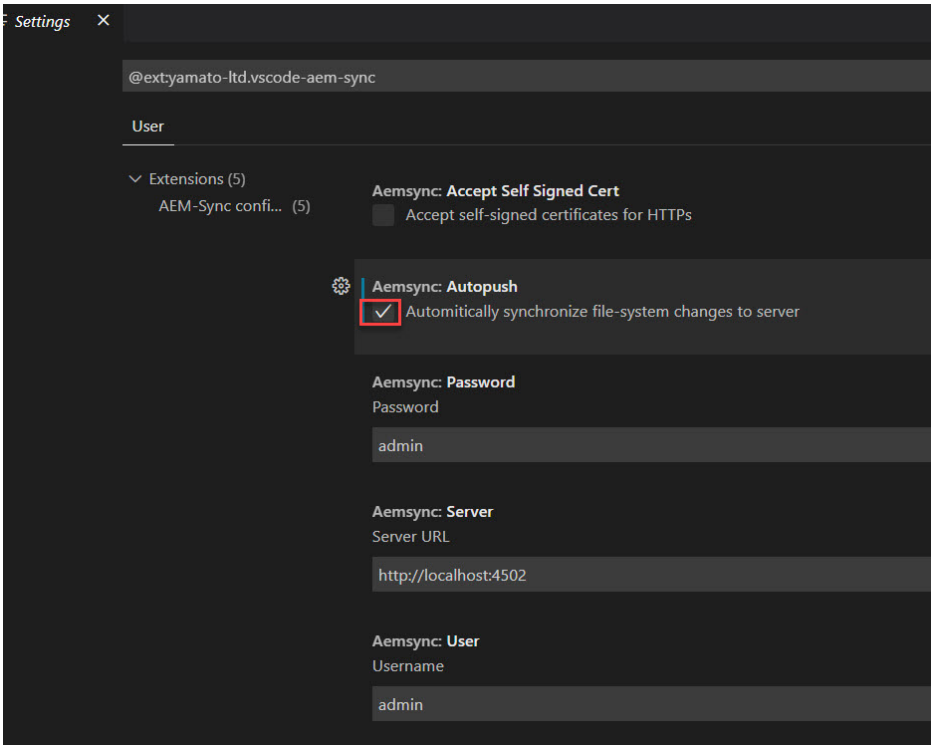


4. Click **Install** to install the extension.
5. In **Visual Studio Code**, select **View** > **Extensions** from the menu bar. The **VSCode AEM Sync Extension** is displayed and enabled in the **EXTENSIONS** area.

6. Click **VSCode AEM Sync** and click the Manage icon (gear icon), as shown:



7. In the Menu, click **Extension Settings.** The **Settings** tab opens.
8. Verify that the extension is properly configured, as shown:
   - Aemsync: Autopush - Enabled
   - Password
   - Server
   - User



9. Close the **Settings** tab by clicking **X**.

You have installed the VSCode AEM Sync Extension for Visual Studio Code.

Activity 4: Synchronization Tools for VSCode

In this activity, you will verify auto push is enabled and sync the AEM content.

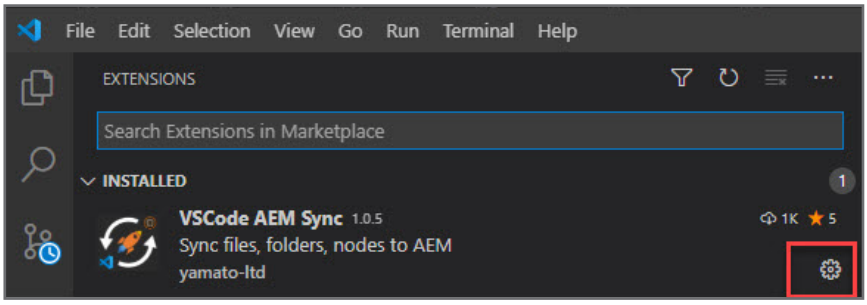
Note: Make sure you have Visual Studio already installed and an AEM author service running locally on port 4502. This activity already assumes you have a Maven project created from the latest archetype. If you do not have any of these things, refer to earlier sections of this guide.

This activity consists of the following tasks:

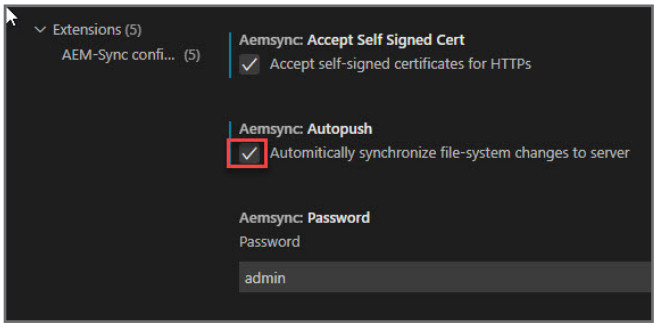
1. Verify auto push is enabled.
2. Sync AEM content.

Task 1: Verify Auto Push Is Enabled

1. In **Visual Studio Code**, select **View** > **Extensions** from the menu bar. The **VSCode AEM Sync Extension** is displayed and enabled in the **EXTENSIONS** area.
2. Click **VSCode AEM Sync** and click the Manage icon (gear icon), as shown:



3. In the Menu, click **Extension Settings.** The **Settings** tab opens.
4. Verify **Autopush** is enabled, as shown:



Task 2: Sync AEM Content

If your project doesn't have the Helloworld component, select a component of your choice to complete this activity.

1. In VS Code Explorer, navigate to: **<AEM project>.ui.apps** > **src / main /content/jcr_root > apps> training > components > helloworld**.
2. Double-click **helloworld.html**. The HTML page opens.
3. Add the following line at the end of the **<pre>** tag in the code: "**This is a change in VS Code**" .

```
*/-->
<div class="cmp-helloworld" data-cmp-is="helloworld">
    <h2 class="cmp-helloworld__title">Hello World Component</h2>
    <div class="cmp-helloworld__item" data-sly-test="${properties.text}">
        <p class="cmp-helloworld__item-label">Text property:</p>
        <pre class="cmp-helloworld__item-output" data-cmp-hook-helloworld="property">${properties.text}</pre>
    </div>
    <div class="cmp-helloworld__item" data-sly-use.model="com.adobe.training.core.models.HelloWorldModel" data-sly-test="${model.message}">
        <p class="cmp-helloworld__item-label">Model message:</p>
        <pre class="cmp-helloworld__item-output"data-cmp-hook-helloworld="model">${model.message}"This is a change in VS Code"</pre>
    </div>
</div>
```

4. Save the changes.
5. Verify the success message "Successfully exported: \apps\<AEM project>\components\.." at the bottom of the screen, as shown:

Note: When you save helloworld.html, the AEM Server connection in Visual Studio Code automatically exports helloworld.html to the JCR.

6. In **CRXDE Lite**, browse to **apps** > **training** > **components** > **content** > **helloworld** > **helloworld.html** and double-click **helloworld.html.** The HTML page opens.

7. Verify the change in CRXDE Lite:

```
<div class="cmp-helloworld" data-cmp-is="helloworld">
    <h2 class="cmp-helloworld__title">Hello World Component</h2>
    <div class="cmp-helloworld__item" data-sly-test="${properties.text}">
        <p class="cmp-helloworld__item-label">Text property:</p>
        <pre class="cmp-helloworld__item-output" data-cmp-hook-helloworld="property">${properties.text}</pre>
    </div>
    <div class="cmp-helloworld__item" data-sly-use.model="com.adobe.training.core.models.HelloWorldModel" data-sly-test="${model.message}">
        <p class="cmp-helloworld__item-label">Model message:</p>
        <pre class="cmp-helloworld__item-output"data-cmp-hook-helloworld="model">${model.message "This is a change in VS Code"</pre>
    </div>
</div>
```

8. Next, we will make a change in CRXDE Lite and import it back into our Visual Studio Code Project. In **CRXDE Lite**, navigate to **apps** > **training** > **components** > **helloworld**.

9. Right-click the **helloworld** component node, and select **Create** > **Create File**.

10. Enter the name as **test.html.**

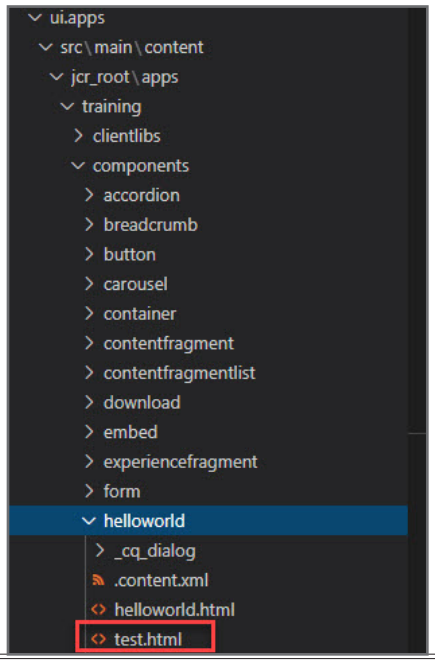
11. Click **OK**. The file is created. Click **Save All** in the upper left to save the changes.

Note: Ensure to click Save All after every change in CRXDE lite.

12. In **Visual Studio Code**, under **<AEM project>.ui.apps**, select **/src/main/content/jcr_root/apps/training/components/helloworld**, right-click and select **Import from AEM Server.**

13. Verify the success message "Successfully imported: \apps\<AEM project>\components\..". The selected node and its children are imported from the server.

14. Under **<AEM project>.ui.apps**, navigate to **/apps/training/components/helloworld** and verify **test.html** appears in your project, as shown:



Tip: If you create something in CRXDE Lite that you want to keep, you must sync it back to **Visual Studio Code** using the process above.

References

You can use the following links for more information on:

- https://code.visualstudio.com/download
- https://code.visualstudio.com/docs/?dv=win32user
- https://marketplace.visualstudio.com/items?itemName=yamato-ltd.vscode-aem-sync

Assets Authoring Basics

Introduction

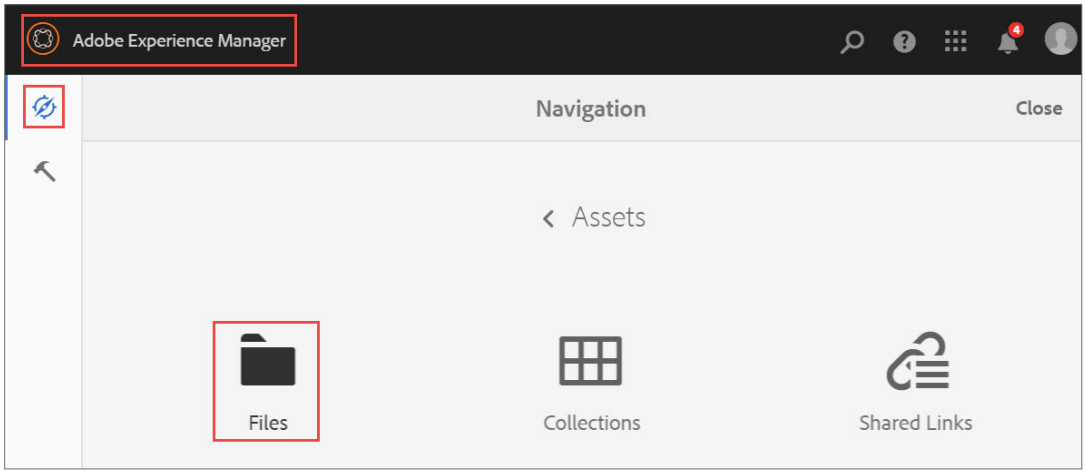
Adobe Experience Manager (AEM) Assets gives you automation and smart tools to rapidly source, adapt, and deliver your assets across audiences and channels. The Assets console of AEM helps import, manage, and share digital assets, such as images, videos, documents, audio files, and content fragments across different channels.

Activity 1: Create a Folder and Upload Assets to It

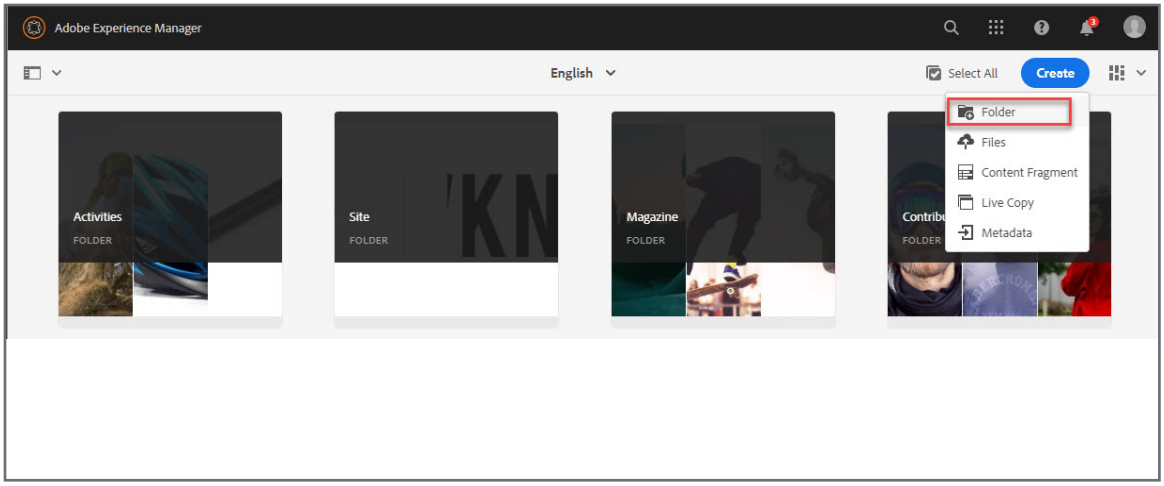
For performing this activity, you will use the WKND site asset folder that comes with AEM.

To create a folder and upload assets to it:

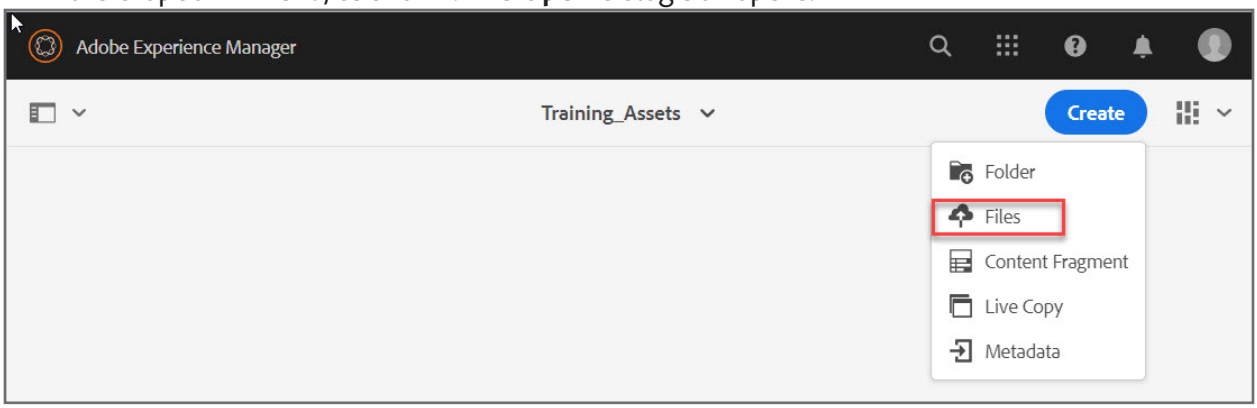
1. Go back to the tab in your browser that has the AEM author service open.
2. Click the **Adobe Experience Manager** icon.
3. Click **Navigation** > **Assets**, and then click **Files**, as shown. If a **Product Navigation** dialog box appears, click **Don't show this again**, and then click **Close**, if you do not want to go through the Product Navigation step-by-step wizard for information.



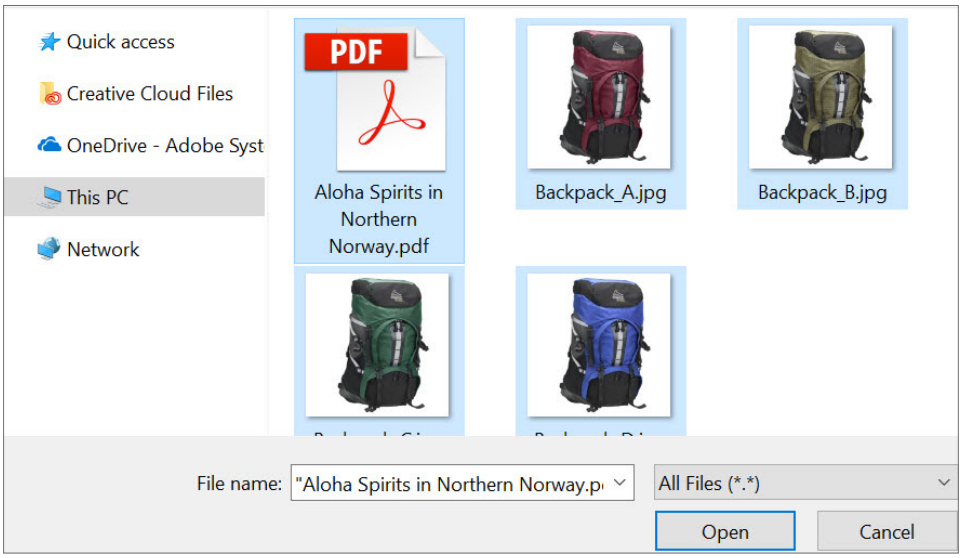
4. Navigate to **WKND Site** > **English**, and then click **Create** from the actions bar, and select **Folder** from the dropdown menu as shown. The **Create Folder** dialog box opens.



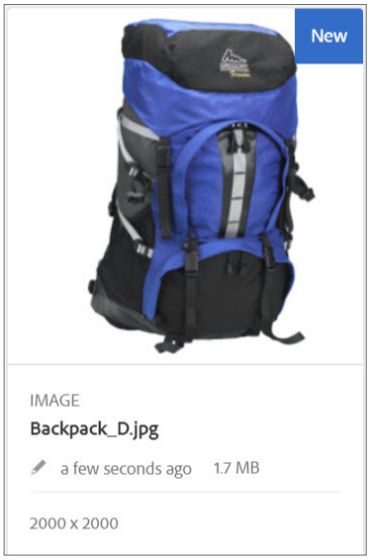
5. Type **Training_Assets** as the title of the folder. The name of the folder is automatically populated from the title.
6. Select **Private** to ensure only you have access to this folder but do not select the **Orderable** checkbox.
7. Click **Create**, as shown. Notice how the **Training_Assets** folder is added to the **WKND site** folder. Refresh the browser, if you are unable to view the new folder.
8. Click the **Training_Assets** folder. Click **Create** from the actions bar, and then select **Files** from the dropdown menu, as shown. The **Open** dialog box opens.



9. From your local file system, navigate to the **Activity_Files/training-files/Assets_Authoring_Basics/** folder, select the **Aloha Spirits in Northern Norway.PDF** as well as all other images by pressing and holding the **Ctrl** key from your keyboard as you click each image, and then click **Open** as shown. The **Upload Assets** dialog box appears, with the images (assets) you are about to upload to the folder.



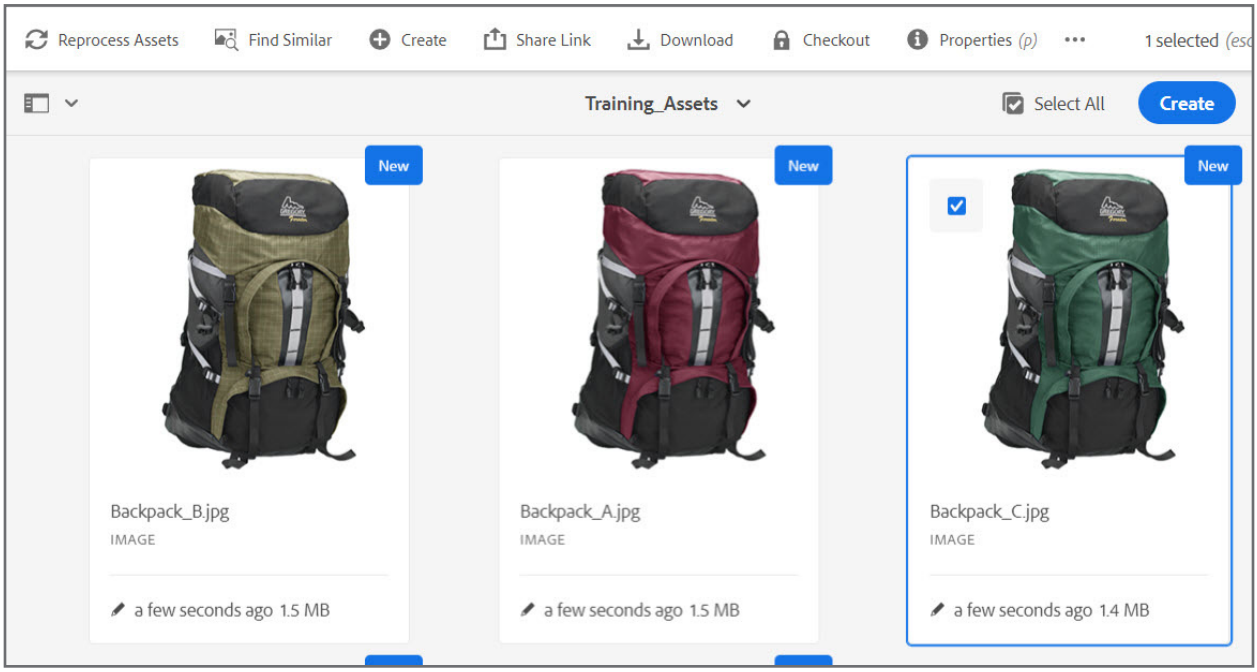
10. Click **Upload**. The assets are being uploaded to the folder. The blue processing bar underneath the corresponding images indicate the upload progress. Once uploaded, a blue "New" tab will appear in the upper-right corner of each image as shown.



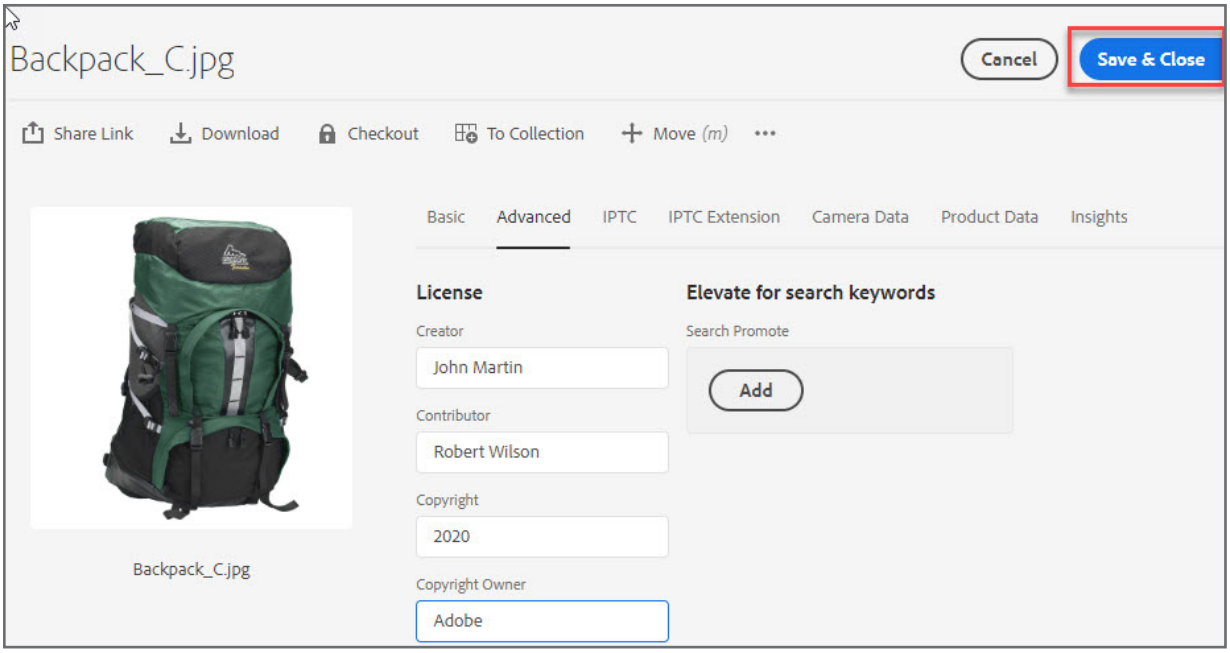
Activity 2: Add Metadata to an Asset

To add the metadata of an asset:

1. Click the **Adobe Experience Manager** icon.
2. Click **Navigation** > **Assets** > **Files**.
3. Navigate to the **WKND site** > **English** > **Training_Assets** folder.
4. Select the **Backpack_C** asset from the folder and click **Properties *(p)*** in the actions bar, as shown. The properties screen opens.

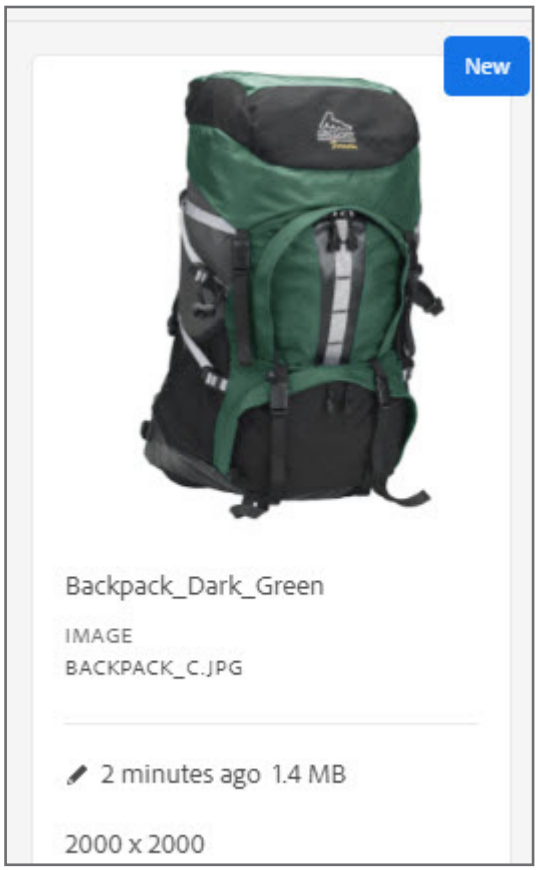


5. On the **Basic** tab, type the following details, as shown:
    a. Title: **Backpack_Dark_Green**
    b. Description: **Backpacks for tough running experience**
    c. Select **English (United States)** from the **Language** dropdown menu.
6. On the **Advanced** tab:
    a. In the **License** section, type the following details:
        i. Creator: **John Martin** (you can add your name)
        ii. Contributor: **Robert Wilson**
        iii. Copyright: **2020**
        iv. Copyright Owner: **Adobe** (your organization name)
7. Click **Save & Close**, as shown. You are taken back to the **Training_Assets** folder.



8. Notice a green message displays at the top of the page, indicating **The form has been submitted successfully**.

9. Verify the changes you made are visible on the asset card, as shown:



References

Use the following link for more information on Assets:

- https://experienceleague.adobe.com/docs/experience-manager-65/assets/home.html?lang=en

Develop Using Eclipse

Introduction

Eclipse can be used to configure the development environment for Adobe Experience Manager (AEM). You can use the Eclipse IDE to develop AEM projects with the help of Maven, Git, and other plugins from the Eclipse marketplace. Along with these plugins, the Eclipse AEM plugin can be used for synchronizing code to a local AEM repository.

Activity 1: Import an AEM Project

In this activity, you will import an AEM project using the AEM Archetype.

This activity includes the following task:

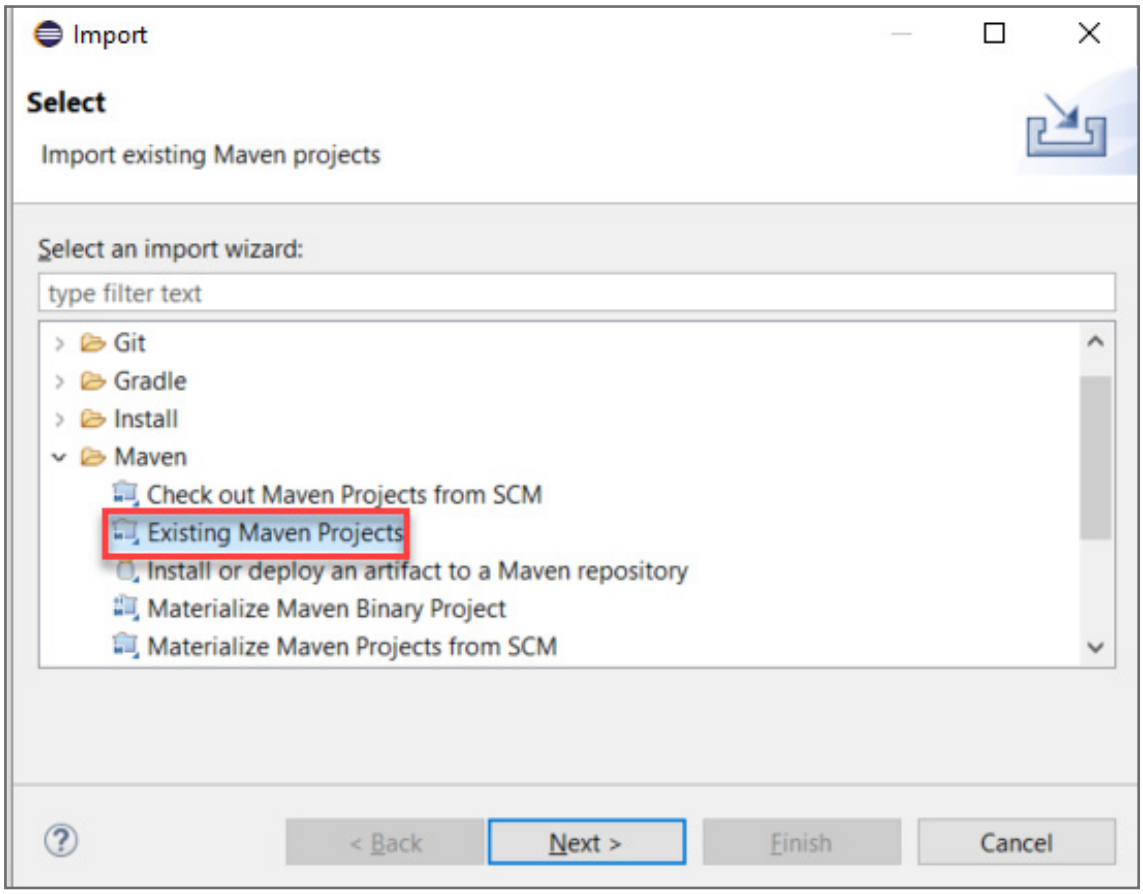
1. Import an AEM project.
2. Configure jcr_root folders.

Task 1: Import an AEM Project

In this task, you will import an AEM project.

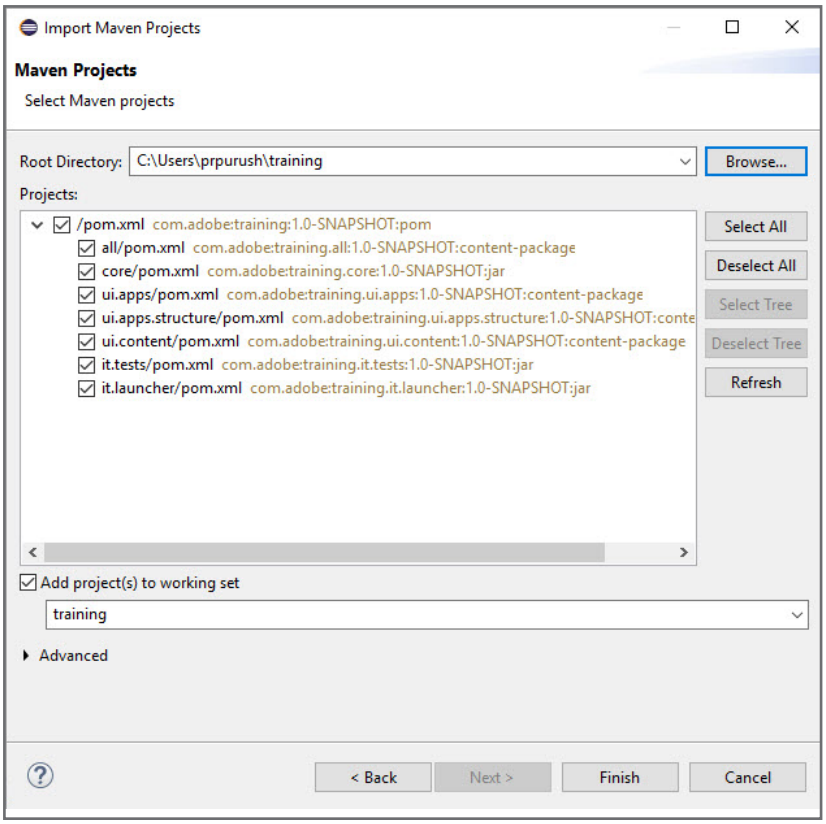
1. Right-click the **Eclipse** icon, and select **Open** from the list. The **Eclipse Launcher** opens .
2. Click **Launch**. The eclipse-workspace – Eclipse IDE window opens.
3. Click **File** > **Import**. The **Import** window opens.
4. Click the down arrow symbol beside **Maven** to expand it.

5. Select **Existing Maven Projects**, and click **Next**, as shown. The **Import Maven Projects** window opens.



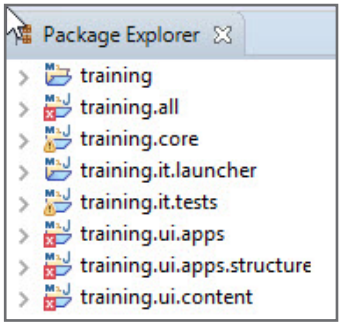
6. Click the **Browse** button beside the **Root Directory** field. The **Select Root Folder** window opens.
7. In the **Select Root Folder** window, navigate to the path to your Maven project **/<AEM project>** and click **Ok**.

8. Select all the POM files, as shown: :



Note: Your AEM project might have a different name.

9. Ensure that the Add project(s) to working set checkbox is selected.
10. Click **Finish**. The AEM project is imported in the Project Explorer, as shown :



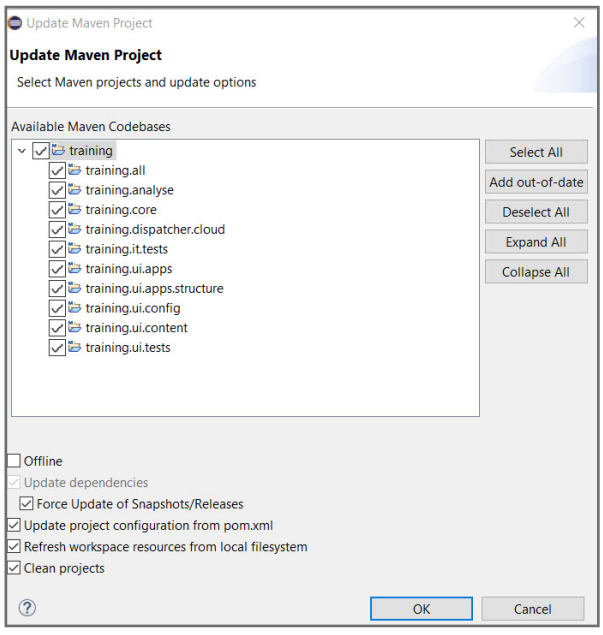
Note: Just like the screenshot above, your project will probably contain errors. These errors can be from a variety of things and you can optionally ignore these in the Eclipse preferences.

11. A **Setup maven plugin connectors** window pops up. Click **Resolve All Later** and then **Finish**. Click **OK** on the **Incomplete Maven Goal Execution** popup.
12. On the **Project Explorer** tab, right-click the parent folder and select **Maven** > **Update Project.** The **Update Maven Project** window opens.
13. Verify your codebase is selected as **training**.

Note: Your Maven codebase might have a different name.

---

14. Select the **Force Update of Snapshots/Releases** checkbox, and click **OK**, as shown. Your project is now updated.

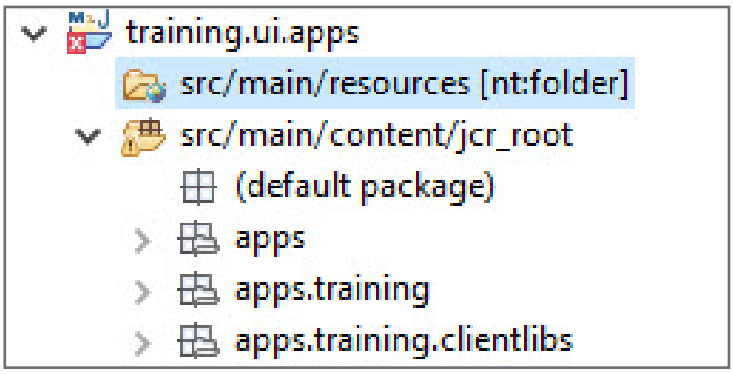


15. Ignore any errors. You will fix them later.

You have successfully imported an AEM project.

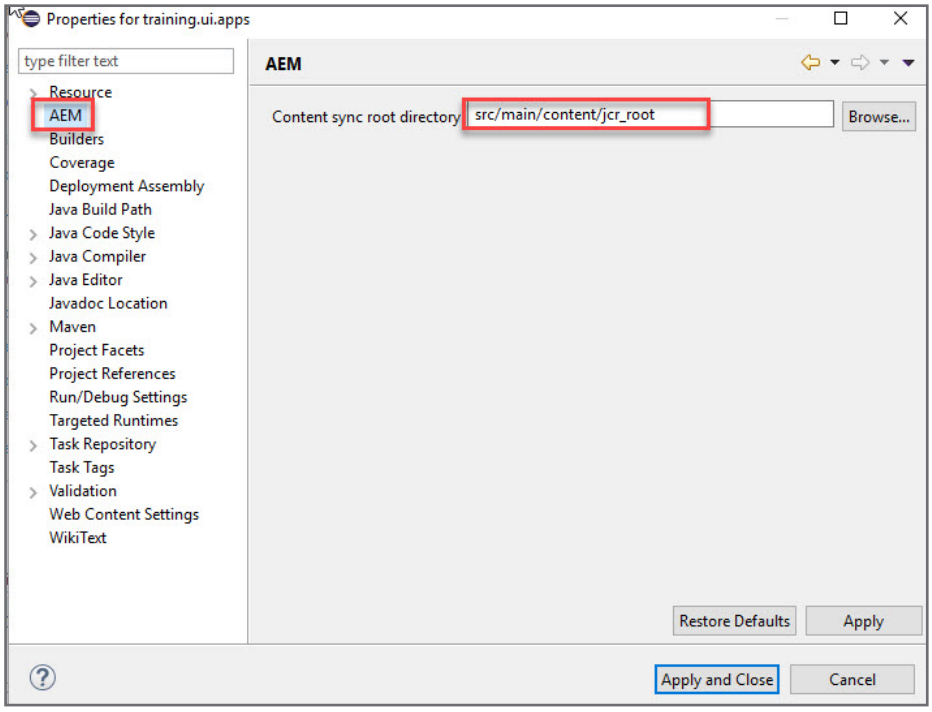
Task 2: Configure jcr_root Folders

1. In Eclipse Project Explorer, navigate to: **<AEM project>.ui.apps** and notice **src/main/resources** is synchronized to AEM, as shown:



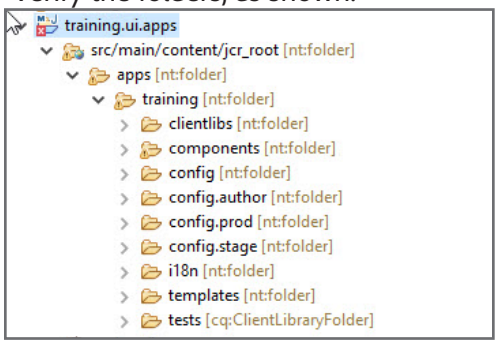
Note: AEM project is the name of the project you created and imported. For example **training**. The name of your project might be different from the screenshot shown.

2. To Synchronize **src/main/content/jcr_root** folder, right-click **<AEM project>.ui.apps** folder and choose **Properties** from the menu.
3. Select **AEM**.
4. Click **Browse** and select the content sync root directory: **src/main/content/jcr_root**, as shown:



5. Click **Ok**.

6. Click **Apply** and **Apply and Close.**
7. Verify the folders, as shown:



8. Similarly, right-click **<AEM project>.ui.content** folder, choose **Properties** from the menu and repeat steps 3-6.
9. Similarly, right-click **<AEM project>.ui.config** folder, choose **Properties** from the menu and repeat steps 3-6.

You have successfully configured your jcr_root folders.

Installing and Configuring Eclipse

Additional Learning

Eclipse is an open source Integrated Development Environment (IDE) used to edit your project source locally on your file system. For AEM projects, you must ensure Eclipse has the following plugins:

- Maven Integration for Eclipse (M2E)
- AEM plugin for Eclipse

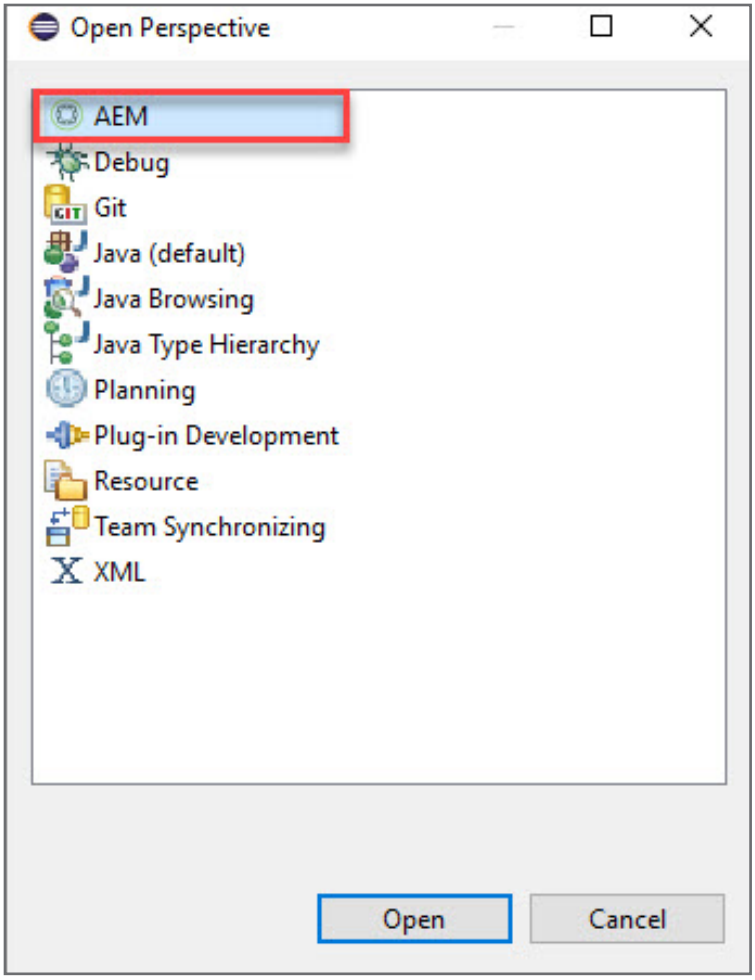
Benefits of AEM Plugin for Eclipse

The AEM plugin has the following benefits:

- Synchronizes both content and OSGi bundles
- Supports debugging and code hot swapping
- Includes a project creation wizard to simplify bootstrapping of AEM projects
- Integrates with AEM Services seamlessly through the Eclipse Server Connector
- Easy JCR properties edition

AEM Perspective

The AEM perspective offers complete control over all your AEM projects and Services.



Using AEM Perspective, you can configure an AEM Server to which Eclipse will connect.
The AEM perspective enables you to add and modify nodes and properties in your AEM project through the AEM Console and the JCR properties view.

Activity 2: Install and Configure Eclipse (Local Only)

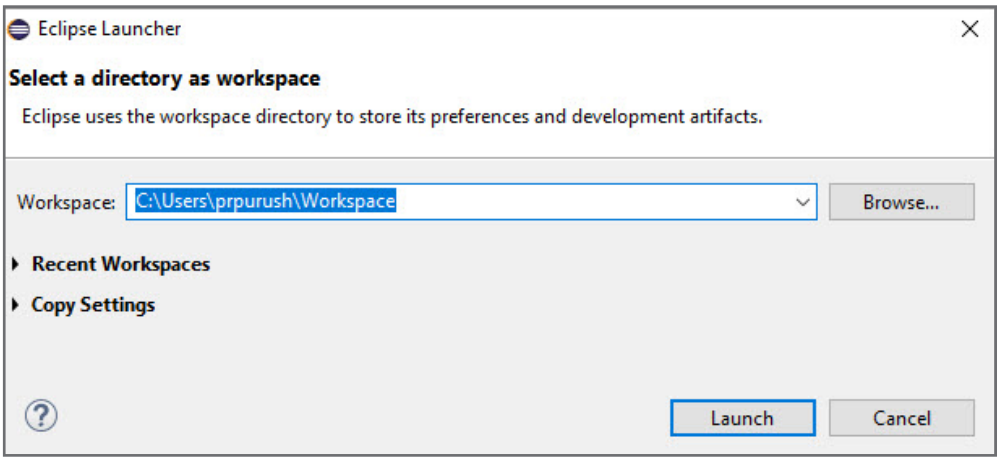
In this activity, you will install and configure Eclipse.
This activity includes the following tasks:

1. Install Eclipse
2. Configure Eclipse

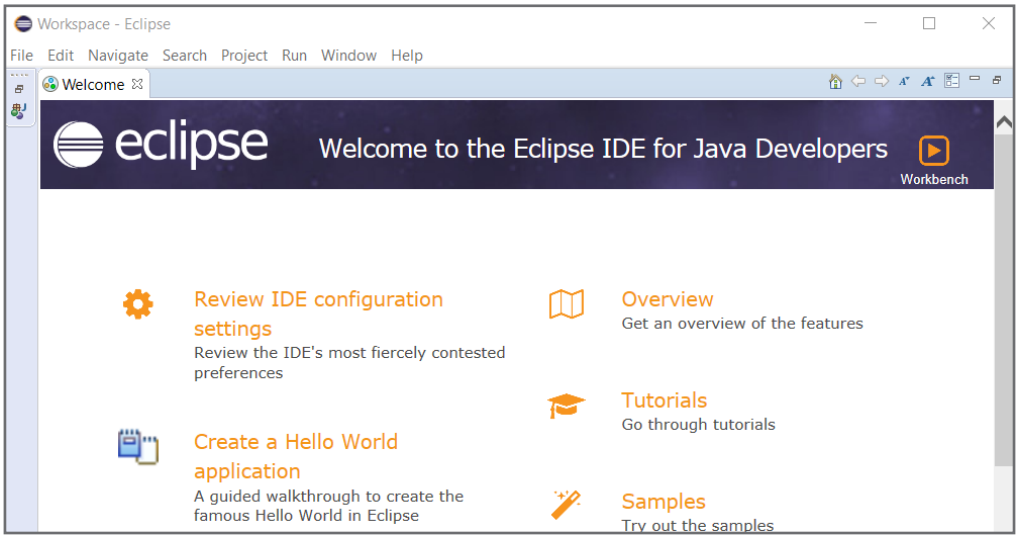
Note: You can skip this activity if you use a ReadyTech environment because AEM is already installed as part of the image.

Task 1: Install Eclipse

1. You can download Eclipse directly from their website: https://www.eclipse.org/downloads/.
2. Navigate to the directory where you extracted the contents of the Eclipse installation zip file. For example, navigate to **C:\Program Files\Eclipse** on Windows or **Applications/Eclipse** on Mac.
3. Double-click **eclipse.exe** (or **eclipse.app**) to start Eclipse. The Eclipse Launcher opens, as shown:



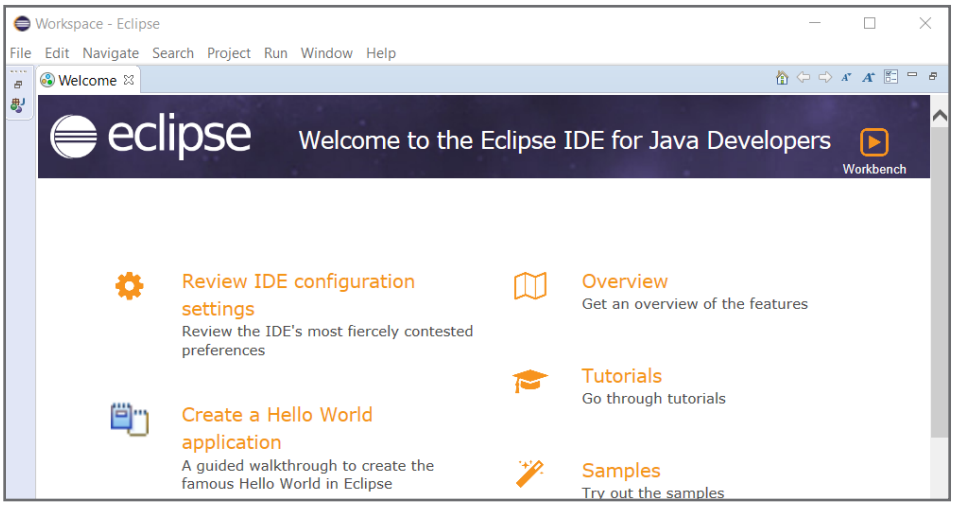
4. Accept the default workspace and click **Launch**. The Eclipse Development Environment opens, as shown:



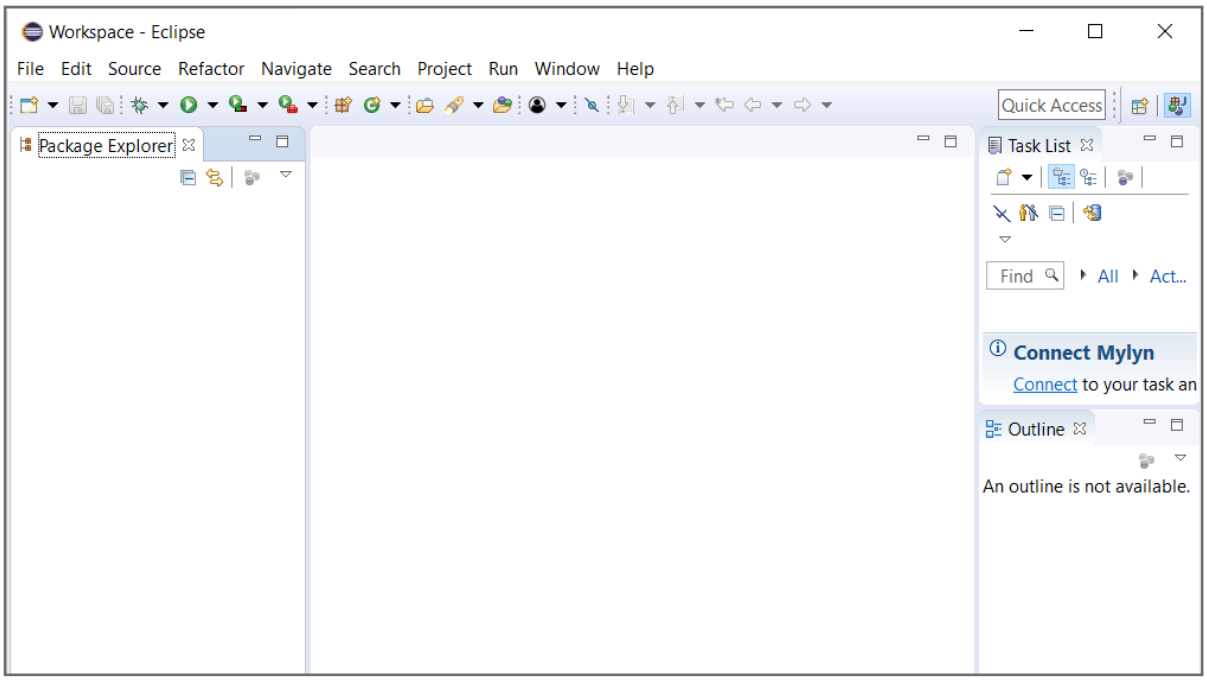
Task 2: Configure Eclipse

In this task, you will configure Eclipse.

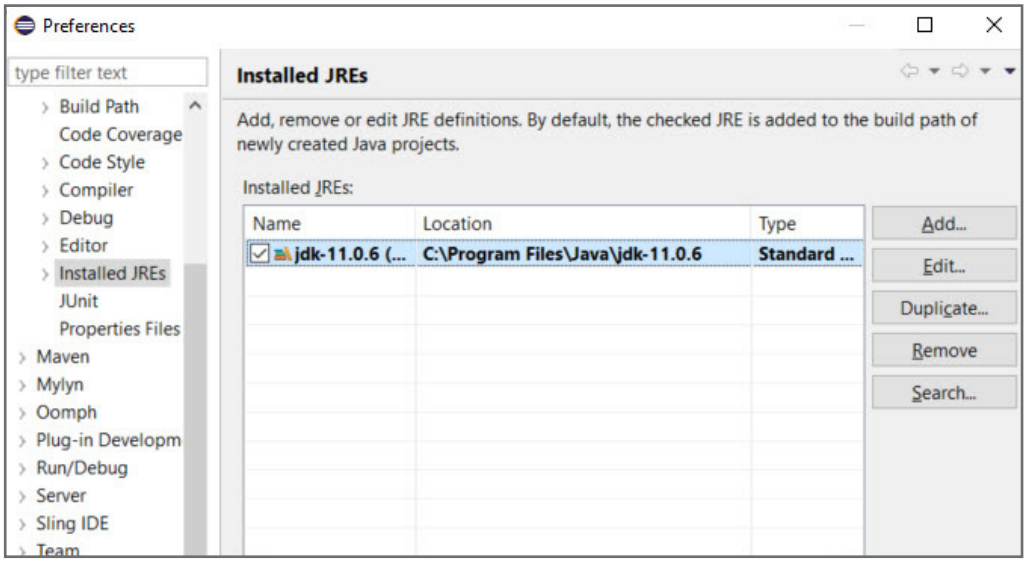
1. Click the **Workbench** logo in the upper-right corner, as shown, to close the Welcome screen.



The Workbench opens, as shown:



2. Verify Eclipse's JRE is set to a JDK by clicking **Window** > **Preferences** > **Java** > **Installed JREs**. You should see a path similar to the one shown in the following screenshot. Otherwise, you must provide the correct directory path to your JDK.



3. Click X in the upper right or click **Cancel** in the lower right to close the Preferences window.

Activity 3: Install the Eclipse AEM Plugin (Local Only)

In this activity, you will install and configure Eclipse AEM plugin.

Note: You can skip this activity if you use a ReadyTech environment because AEM is already installed as part of the image.

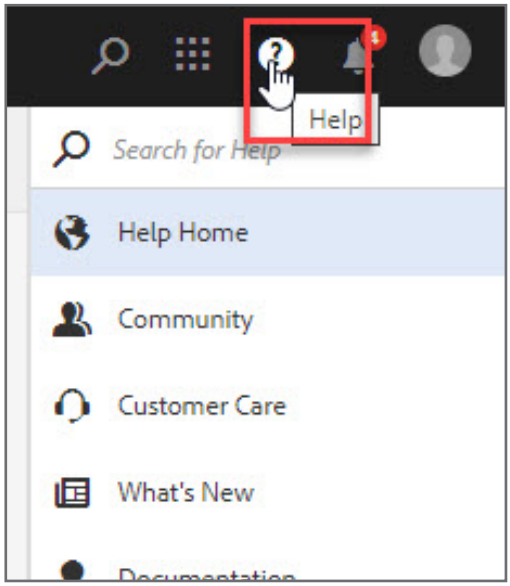
1. Open the URL https://eclipse.adobe.com/aem/dev-tools/.

Note: There are two ways to install the plugin:Online:
1. You will provide the link to install the plugin in Eclipse.Offline.
2. You will provide the downloaded plugin in Eclipse.

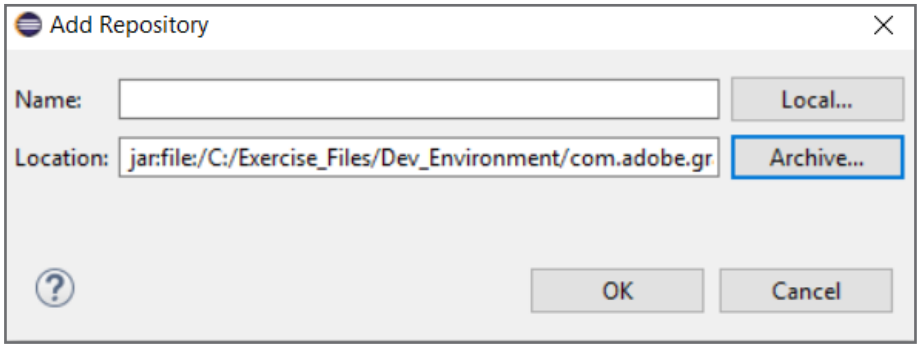
You will perform the offline method in this activity.

To install your package:

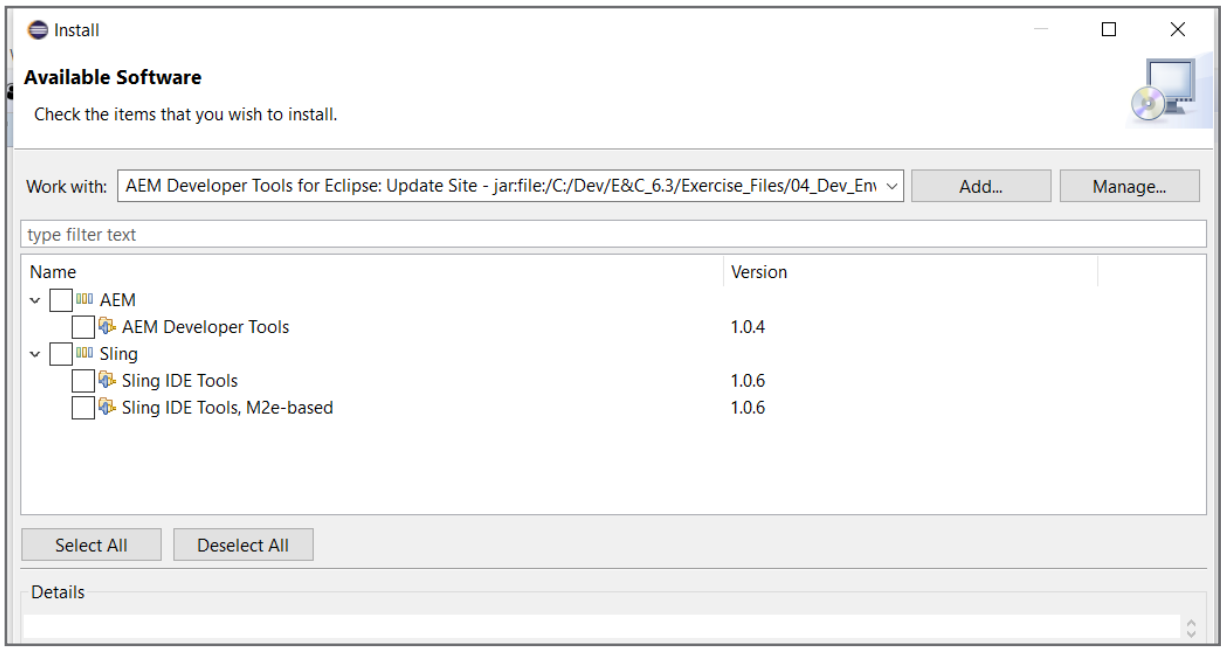
2. Double-click **eclipse.exe** (or eclipse.app) to start Eclipse. The Eclipse Development Environment opens:
3. Select **Help** > **Install New Software,** as shown. The Install window opens.



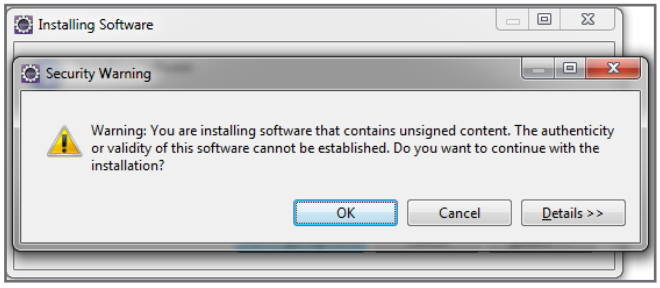
4. Click **Add**. The Add Repository dialog box opens.
5. Click **Archive**.
6. Navigate to the repository archive and select the zip file (**com.adobe.granite.ide.p2update-1.3.0.zip**) provided for the plugin from **Activity_Files_TB/Dev_Environment/**.
7. Click **Open** to add the location to the **Location** field in the **Add Repository** window.
8. Click **Add**. The location of the repository is added, as shown:



9. The **Available Software** window opens displaying the items you want to install, as shown:



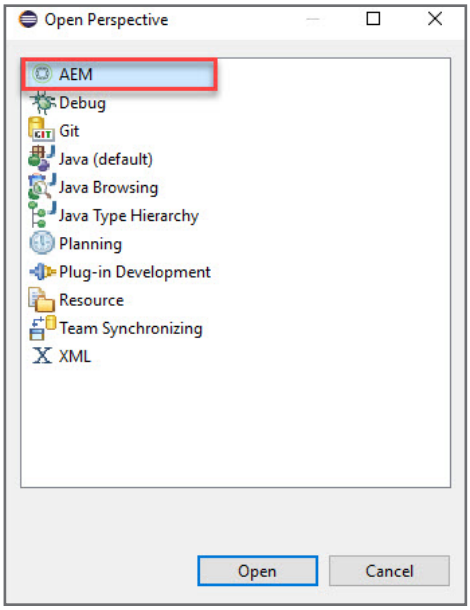
10. Click **Select All** to select AEM and Sling.
11. Click **Next**. The Install Details screen opens.
12. Click **Next**. The Review Licenses screen opens.
13. Click the **I accept the terms of the license agreements** option and then click **Finish**. The **Installing Software** dialog box with a progress bar opens. The installation may take a couple of minutes. You can see the progress of the installation in the lower-right corner of the Eclipse workspace.
14. If a **Security Warning** window pops up, as shown, click **Install anyway** to continue the installation. The **Software Updates** dialog box will re-open until the installation is completed.



15. Click **Restart Now** to restart Eclipse to load the newly installed tools.

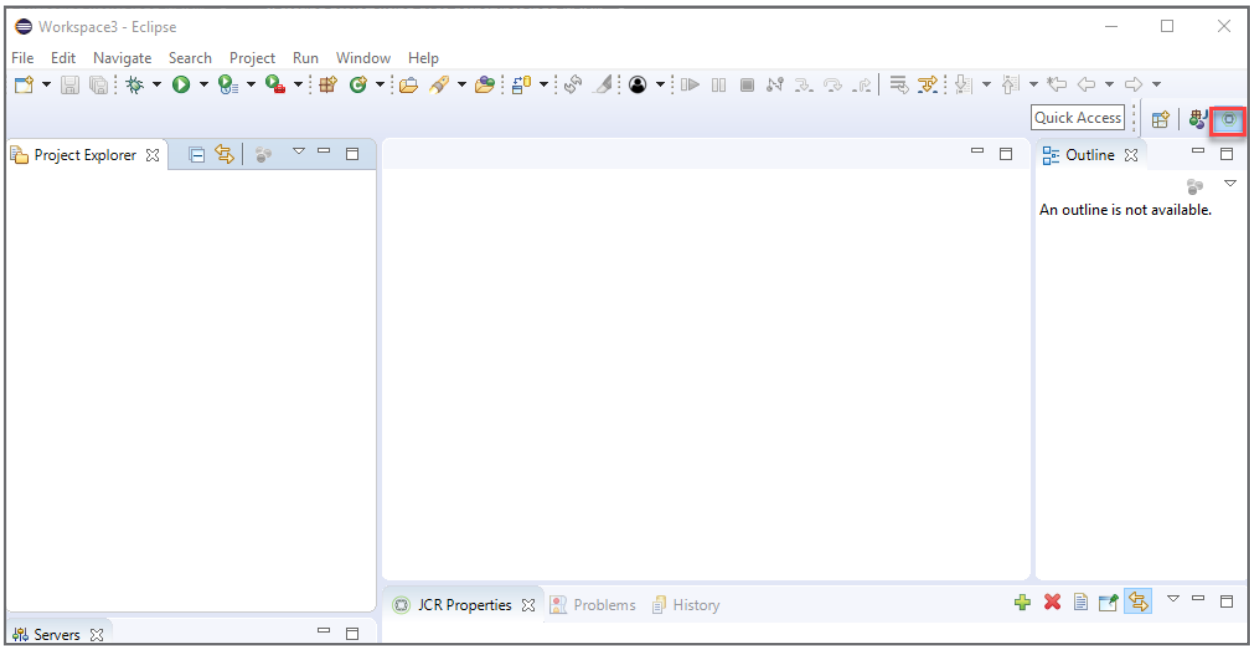
Note: This process takes about a minute. If you see a dialog box that asks if you want to keep the current location of your Eclipse install, click **OK**. Eclipse opens.

16. Click **Workbench** in the upper-right corner. The Workspace opens.
17. A new AEM perspective becomes available in Eclipse. To verify the AEM perspective was added, click **Window** > **Perspective** > **Open Perspective** > **Other...**, which opens the **Open Perspective** window, as shown:



18. Click **AEM** and then click **Open**. This closes the **Open Perspective** window.

19. Notice the AEM perspective icon is visible at the top, as shown:



Note: Keep Eclipse open for the next activity.

Activity 4: Synchronization Tools for Eclipse

The AEM plugin allows you to connect to an AEM server to auto-push changes made in the project into the JCR. This can be done to synchronize content in the ui.apps and ui.content modules with the JCR, but also for hot code swap (such as updating the bundle without Maven builds) on changes made in the core module. The synchronization happens when saving a file in those modules, but can also be manually triggered (as well as changes done in the repository can be manually imported into the project).

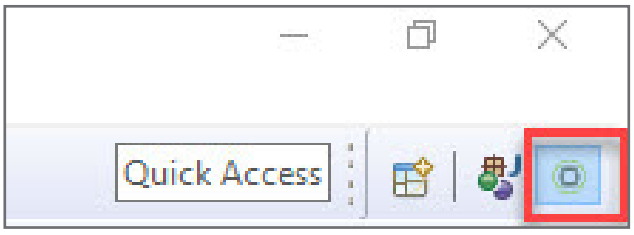
The AEM server connection allows you to synchronize modules individually, by using the add/remove resources function.
This activity includes the following tasks:

1. Configure the AEM server
2. Sync AEM content

Task 1:Configure the AEM Server

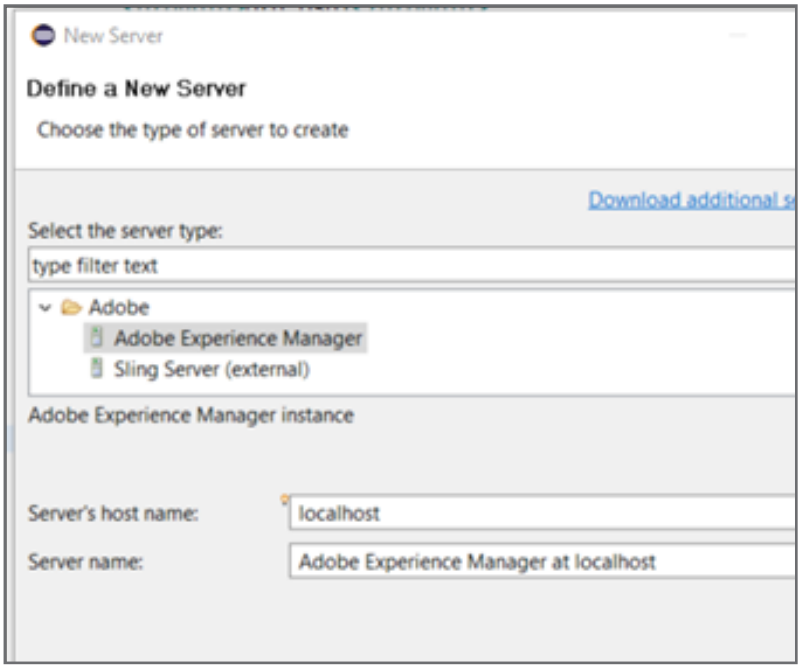
In this task, you will configure a connection to the AEM server to enable content synchronization for the ui.apps and ui.content modules, but NOT the core module, since you are using Maven to build and install the code (preferred method).

1. In Eclipse, verify **AEM Perspective** is selected in the upper-right corner.



2. On the left-hand side of the Eclipse Workspace below the **Project Explorer tab**, notice the **Servers** tab. Click the **No Servers are available. Click this link to create a new server...** link. The **Define a New Server** dialog box opens.

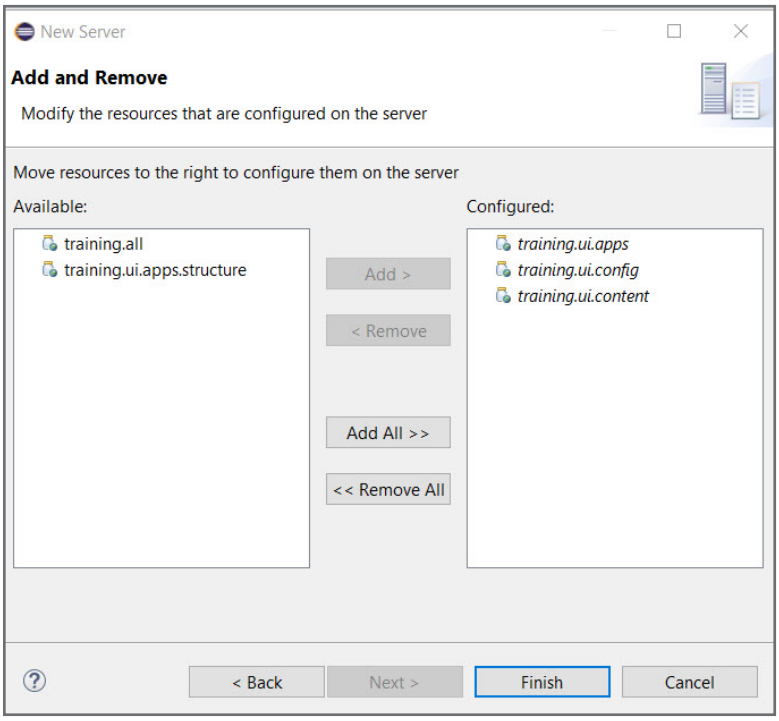
3. Select **Adobe** > **Adobe Experience Manager**, as shown:



4. Accept the default settings for the **Server's host name** and **Server name** fields.

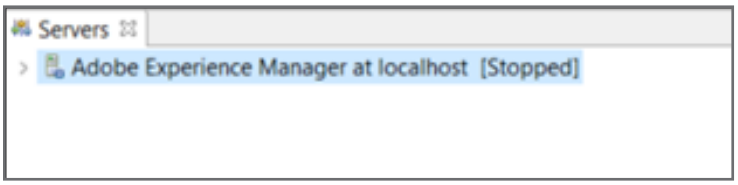
5. Click **Next**. The **Add and Remove resources** dialog box opens.

6. Select **<AEM project>.ui.apps, <AEM project>.ui.config** and **<AEM project>.ui.content** one by one, and click the **Add >** button to move the specified resources from the **Available** section to the **Configured** column, as shown:



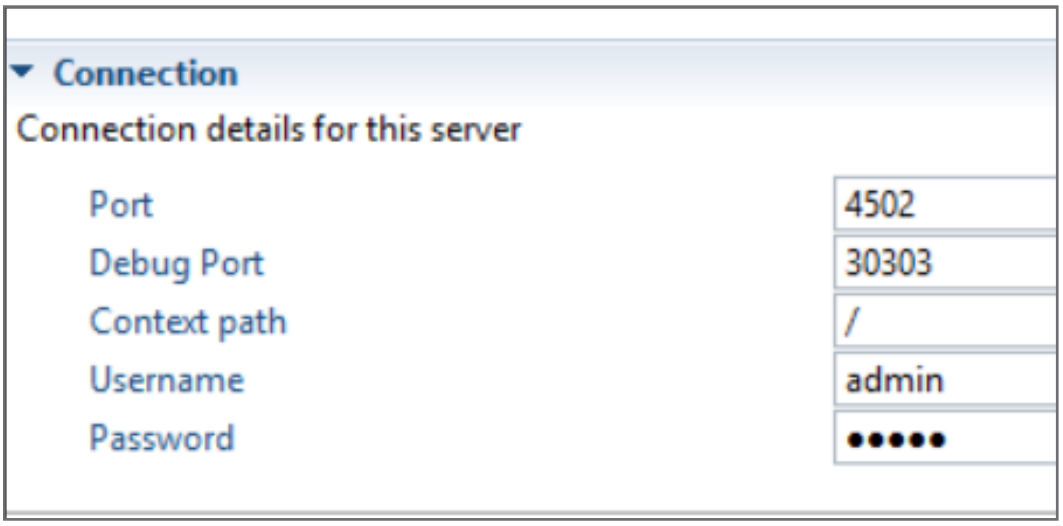
Note: AEM project is the name of the project you created and imported. For example **training**. The name of your project might be different from the screenshot shown.

7. Click **Finish** on the **New Server** screen. The AEM Server is now defined.
8. On the left-hand side of the workspace (below the **Project Explorer** tab), click the **Servers** tab and note how **Adobe Experience Manager at localhost [Stopped]** is now available, as shown:



9. To modify the configuration for the AEM Server, double-click the **Adobe Experience Manager at localhost [Stopped]** to open it in the editor.

10. In the **Connection** area, change the current port (listed in the **Port** field, in the **Connection** area) to **4502**, as shown:

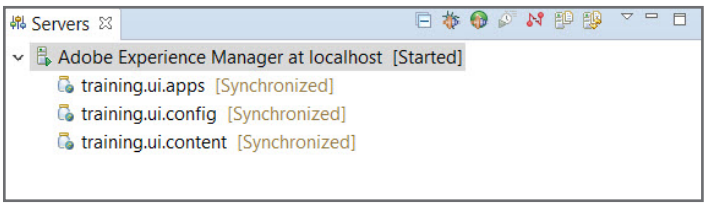


11. Save the changes (**File** > **Save** OR **Ctrl+S)**.

You have now created a connection from Eclipse to the AEM server on your computer. You will now start the AEM server.

12. Right-click **Adobe Experience Manager at localhost** on the **Servers** tab, and click **Start**. The server is started.

13. Verify the Server has started, as shown:



The contents are now in synch with the AEM server.

Task 2: Sync AEM Content

If your project doesn't have the Helloworld component, select a component of your choice to complete this activity.

1. In Eclipse Project Explorer, navigate to: **<AEM project>.ui.apps** > **src / main /content/jcr_root > apps> training > components > helloworld**.
2. Double-click **helloworld.html**. The HTML page opens.
3. Add the following line at the end of the **<pre>** tag in the code: "**This is a change in Eclipse**" .

```
17      <h2 class="cmp-helloworld__title">Hello World Component</h2>
18      <div class="cmp-helloworld__item" data-sly-test="${properties.text}">
19          <p class="cmp-helloworld__item-label">Text property:</p>
20          <pre class="cmp-helloworld__item-output" data-cmp-hook-helloworld="property">${properties.text}</pre>
21      </div>
22      <div class="cmp-helloworld__item" data-sly-use.model="com.adobe.training.core.models.HelloWorldModel" data-sly-test="${model.mess
23          <p class="cmp-helloworld__item-label">Model message:</p>
24          <pre class="cmp-helloworld__item-output"data-cmp-hook-helloworld="model">${model.message}"This is a change in Eclipse"</pre>
25      </div>
26  </div>
27
```

4. Save the changes.

Note: When you save helloworld.html, the AEM Server connection in Eclipse exports helloworld.html to the JCR.

5. In **CRXDE Lite**, browse to **apps** > **training** > **components** > **content** > **helloworld** > **helloworld.html** and double-click **helloworld.html.** The HTML page opens.
6. Verify the change in CRXDE Lite:

```
    Unless required by applicable law or agreed to in writing, software
    distributed under the License is distributed on an "AS IS" BASIS,
    WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
    See the License for the specific language governing permissions and
    limitations under the License.
*/-->
<div class="cmp-helloworld" data-cmp-is="helloworld">
    <h2 class="cmp-helloworld__title">Hello World Component</h2>
    <div class="cmp-helloworld__item" data-sly-test="${properties.text}">
        <p class="cmp-helloworld__item-label">Text property:</p>
        <pre class="cmp-helloworld__item-output" data-cmp-hook-helloworld="property">${properties.text}</pre>
    </div>
    <div class="cmp-helloworld__item" data-sly-use.model="com.adobe.training.core.models.HelloWorldModel" data-sly-test="${model.message}">
        <p class="cmp-helloworld__item-label">Model message:</p>
        <pre class="cmp-helloworld__item-output"data-cmp-hook-helloworld="model">${model.message}"This is a change in Eclipse"</pre>
    </div>
</div>
```

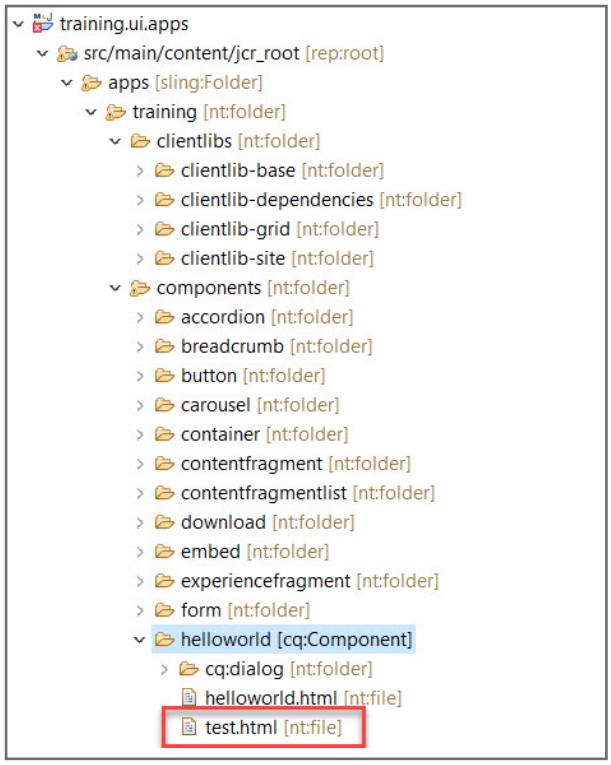
Note: Any files saved in your project will auto sync with AEM based on the filter.xml file in each module that is configured with the AEM connection.

7. You will now sync changes made in AEM back into your Maven project. In **CRXDE Lite**, navigate to **apps** > **training** > **components** > **helloworld**.
8. Right-click the **helloworld** component node, and select **Create** > **Create File**.
9. Enter the name as **test.html.**

10. Click **OK**. The file is created. Click **Save All** in the upper left to save the changes.

Note: Ensure to click Save All after every change in CRXDE lite.

11. In **Eclipse**, under **<AEM project>.ui.apps**, select **/src/main/content/jcr_root/apps/training/components/helloworld**, right-click and select **Import from Server.**
12. Accept the default settings and click **Finish**. The selected node and its children are imported from the server.
13. Under **<AEM project>.ui.apps**, navigate to **/apps/training/components/helloworld** and verify **test.html** appears in your project, as shown:



Note: Step 11 is a very typical process in AEM Java development to sync new content from the JCR to Eclipse. Remember that Eclipse is our master repository locally and anything created in the JCR that is a part of our project must be pulled back down into Eclipse. This is a very common process with config nodes, dialog structures, components, and clientlibs.

Tip: If you create something in CRXDE Lite that you want to keep, you must sync it back to Eclipse using the process above.

References

- https://docs.adobe.com/content/help/en/experience-manager-learn/cloud-service/local-development-environment-set-up/development-tools.html

Front-End Development Using aemfed

Introduction

aemfed is an open-source, command-line tool that can be used to speedup front-end development.

aemfed

aemfed is an open-source, Command line interface (CLI) that can be used to speed-up front-end development. It is powered by aemsync, Browsersync, and the Apache Sling Log Tracer.

Using aemsync, aemfed listens to file changes and automatically syncs the changes to a running AEM service. aemfed also determines which clientlibs are affected by the uploaded changes.

aemfed is also built to work with the Sling Log tracer to automatically display any server-side errors directly in the terminal window. It captures errors related to:

- Clientlibs
  - Less compilation
  - Javascript minification
- HTL templates
- JSP files
- .content.xml

If the error messages contain references to nodes in the JCR, it attempts to translate them back to the files on your local file system.

Activity 1: Install aemfed (Local Only)

In this activity, you will install aemfed.

Prerequisite: You need to have NVM installed on your machine to complete this activity. For instructions on how to install NVM, refer to **Module 6 - Creating project using Maven - Activity 1 > Task 3**.

Note: You can skip this activity if you use a ReadyTech environment because aemfed is already installed as part of the image.

1. Open a terminal window in your User directory.

Note: This global installation will require administrative privileges. On Windows, open command window using Run as admin option.

2. Install aemfed.

- Mac

```
$ npm install aemfed
```

- Windows

```
$ npm install aemfed --global
```

Note: The previous command will install aemfed globally. If you have a package.json for your AEM project, you can modify the package.json to add aemfed as a dev dependency:

$ npm install aemfed --save-dev

3. Show available options by entering the following command:

```
$ aemfed -h
```

```
C:\Program Files\nodejs>aemfed -h
Usage: aemfed [OPTIONS]
Options:
  -t targets           Default is http://admin:admin@localhost:4502
  -p proxy_port        Default is 3000
  -w path_to_watch     Default is current
  -e exclude_filter    Anymatch exclude filter; disabled by default
  -i sync_interval     Update interval in milliseconds; default is 100
  -o open_page         Browser page to be opened after successful launch; def
t is "false".
  -b browser           Browser where page should be opened in; this parameter
 platform dependent; for example, Chrome is "google chrome" on OS X, "google-
ome" on Linux and "chrome" on Windows; default is "google chrome"
  -q load_qr           Enable QR code plugin for connected browsers; default
"true".
  -h                   Displays this screen
  -v                   Displays version of this package
```

Congratulations! You have installed aemfed.

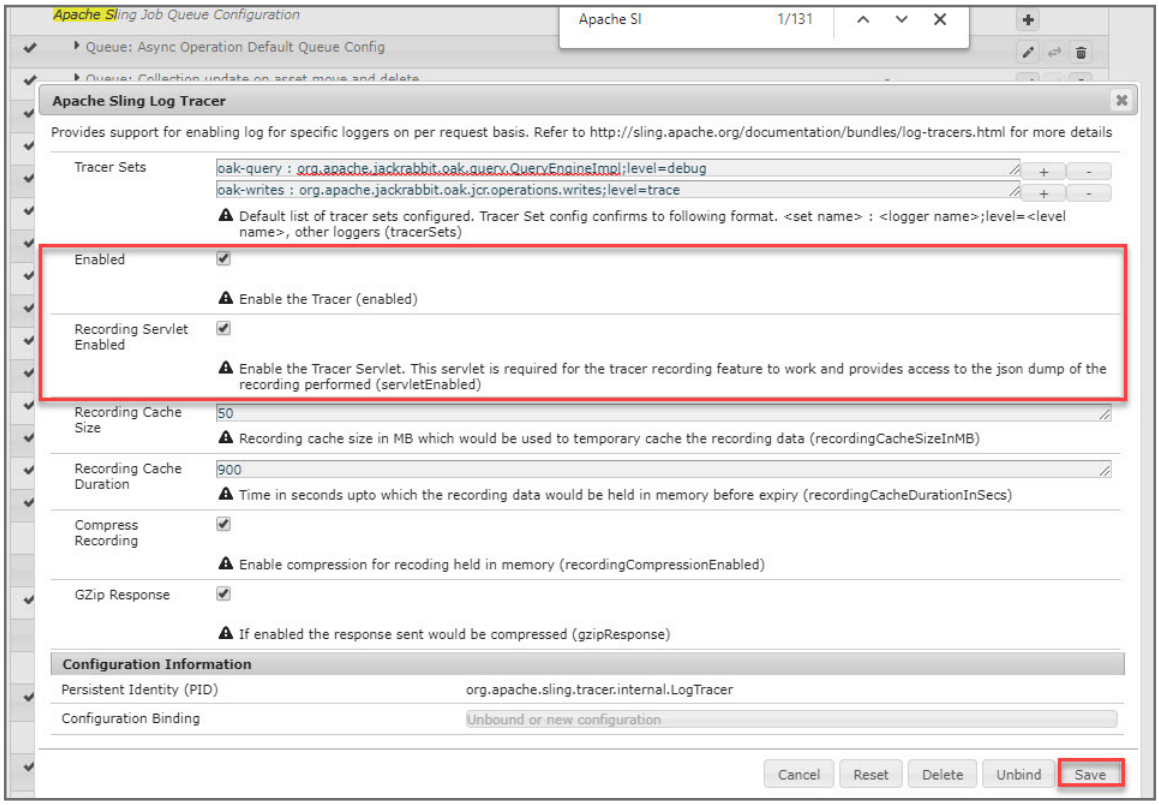
Activity 2: Configure and Run aemfed

In this activity you will enable the Apache Sling Log Tracer in AEM.

1. Enable the Apache Sling Log Tracer in AEM
2. Start aemfed.

Task 1: Enable the Apache Sling Log Tracer in AEM

1. Navigate to http://localhost:4502/system/console/configMgr.
2. Search for **Apache Sling Log Tracer** and open the configuration for the **Apache Sling Log Tracer**.
3. Update the configuration with the following values selected and **Save**.
   - **Enabled**
   - **Recording Servlet Enabled**



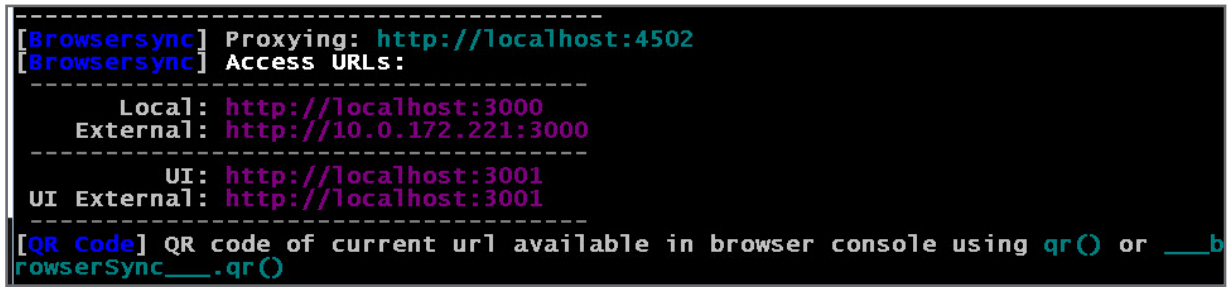
Task 2: Start aemfed

1. Open a new terminal window and navigate to your <AEM project directory>. To start aemfed and configure the AEM target service:

```
$ aemfed -t "http://admin:admin@localhost:4502" -w "ui.apps/src/
main/content/jcr_root/"
```

Note: If you get a Windows Security Alert, click Allow Access.

Note: aemfed will return proxy URLs that you can use to see automated clientlib changes.



Caution: Do not close this terminal window. It is controlling aemfed.

Activity 3: Install Content Locally (Optional)

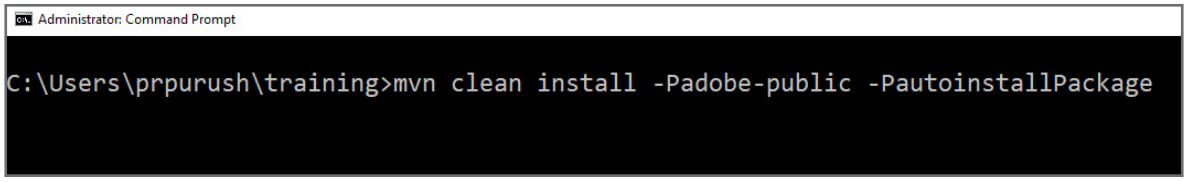
In this activity, you will install the local Maven project into AEM.

Note:This activity assumes you have a local Maven project. <AEM Project Directory> represents the location of this local Maven project.

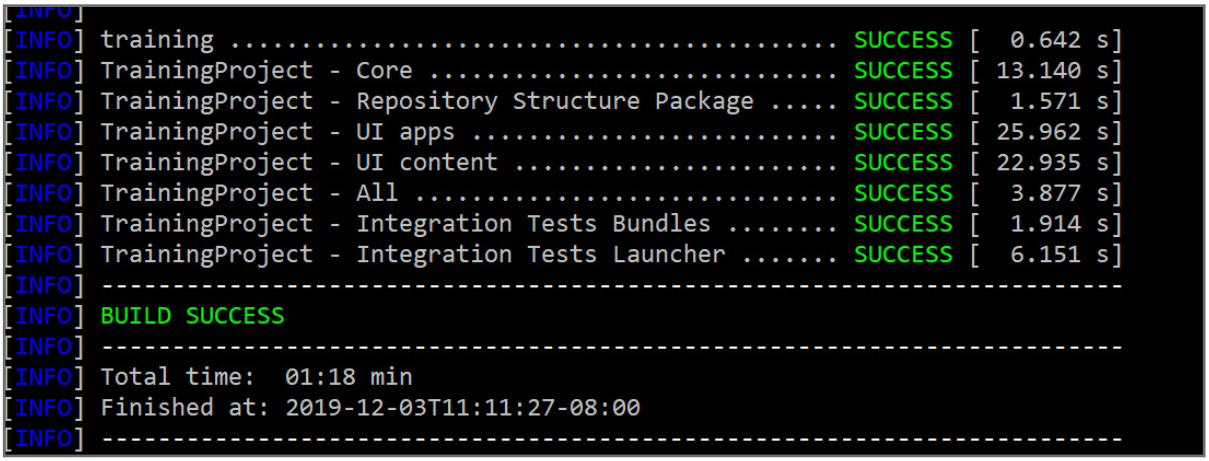
Note: You can skip this activity if you have already installed your AEM project.

1. Confirm that your AEM service is running.
2. Open a terminal window and navigate to your <AEM project directory>. Run the following command to deploy your AEM project

```
$mvn clean install -Padobe-public -PautoInstallSinglePackage
```



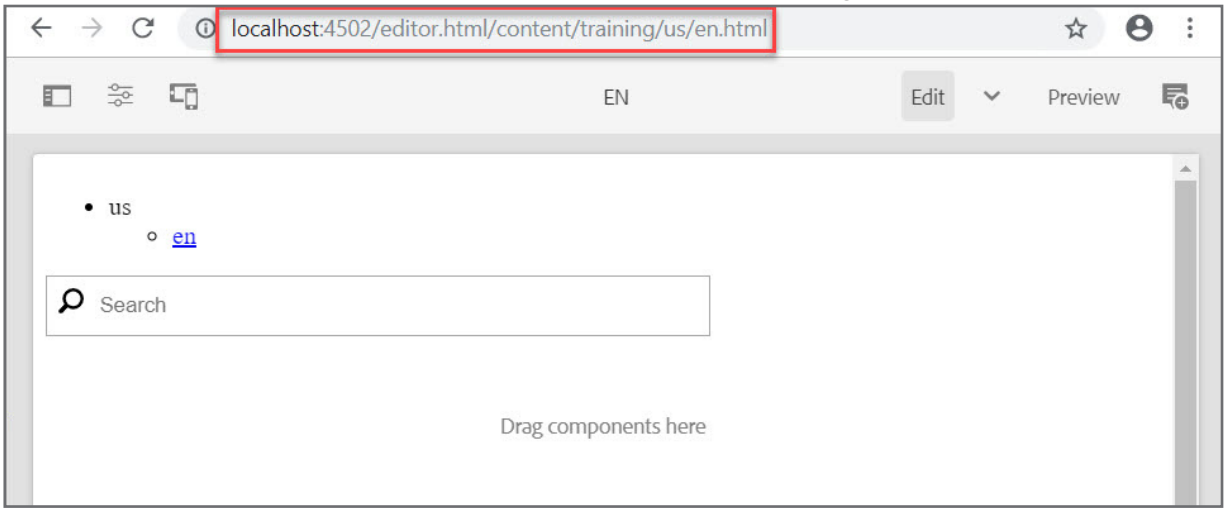
3. Verify the install is successful, as shown:



Activity 4: Add a Component Style Using aemfed

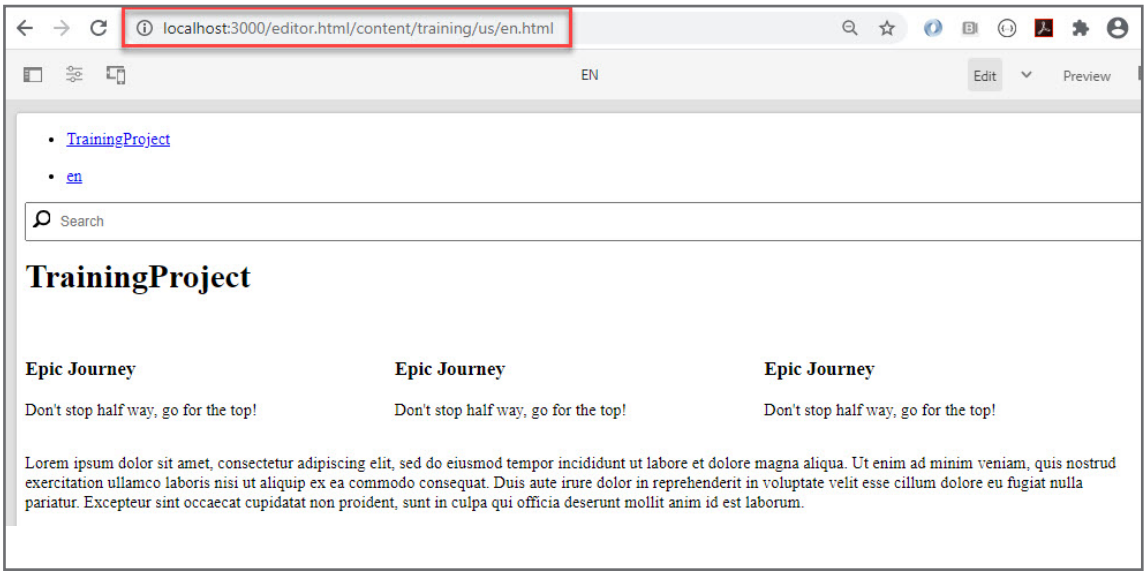
You can use aemfed to connect a local AEM server to a content package. In this activity, you will use a project created from the archetype, but you could connect with any unzipped content package.

1. Using your browser, navigate to **Sites** > **training** > **us** . Click the **en** page (thumbnail) to select it, and then click **Edit *(e)*** from the actions bar. This open the en page.



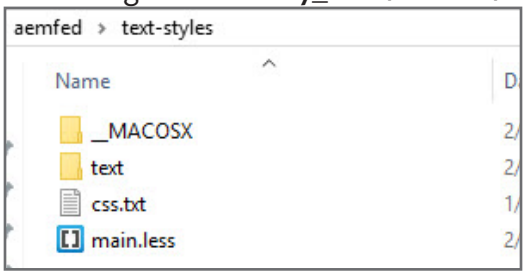
2. Drag a text component to the **en** Page and add some dummy text.

3. Modify the URL to point to the proxy server that is running. If you are using default ports, change 4502 (AEM) to port 3000 (aemfed) and refresh the browser.



You should now see the same page content as your browser tab that displays content from **localhost:4502** but proxied via port **3000** (or whichever port the proxy is using).

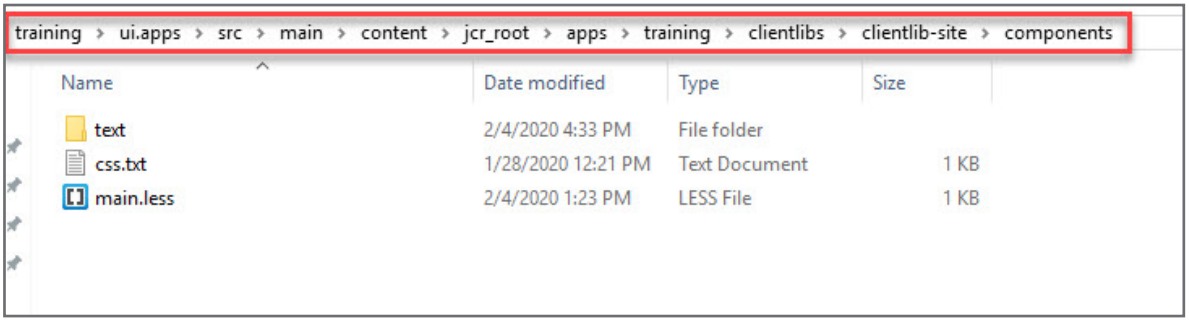
4. Navigate to **Activity_Files/aemfed/** and extract the contents of **text-styles.zip,** as shown:



5. Copy the folder **text.**

6. In a file window, navigate to /**<AEM Project Directory>/ui.apps/src/main/content/jcr_root/apps/training/clientlibs/clientlib-site**

7. Paste the **text** folder from **Activity_Files**> **aemfed** into the **clientlib-site/components** folder (create **components** folder if needed):

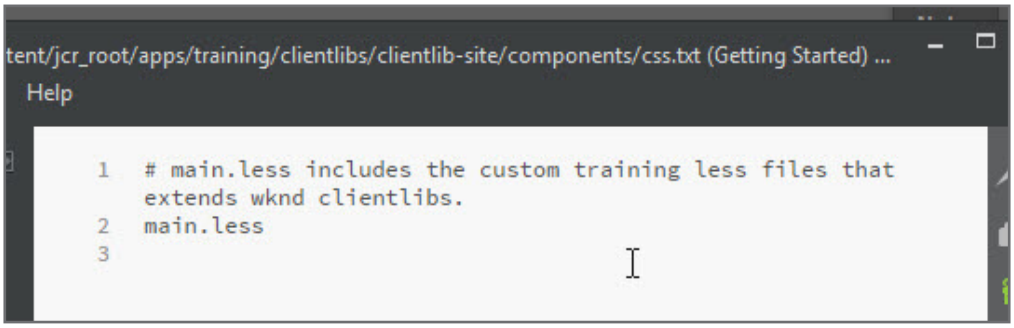


8. Verify you see messages in the aemfed terminal window similar to the following screenshot:

```
[localhost:4502] Special paths were changed, so rebuild clientlib tree
[localhost:4502] Get data from server: 252 ms
[localhost:4502] Process data: 18 ms
[localhost:4502] Clientlib tree: 271 ms
[localhost:4502] Rebuild clientlib tree: 271 ms
[Browsersync] Reloading Browsers...
ADD jcr_root/apps/trainingproject/clientlibs/.DS_Store
  Deploying to [localhost:4502] in 43 ms at 2019-08-23T15:41:39.645Z: OK
[Browsersync] Reloading Browsers...
```

9. Even though initial text style has been loaded, it needs to be added to the design. If you already have a working design in your project, skip to step 11.

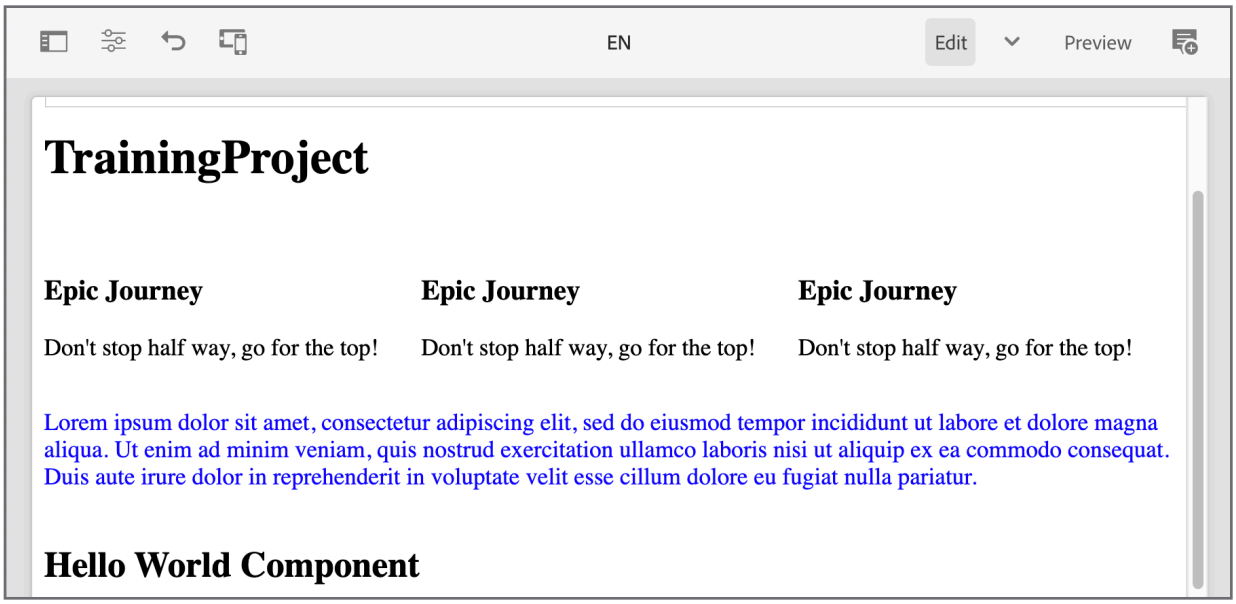
10. Using the IDE of your choice, navigate into the **/clientlib-site/** folder and open **css.txt** and add the line **main.less.**



11. Save **css.txt** and close.
12. Locate the **main.less** file under **clientlibs/clientlib-site**. If the file does not exist, create it.
13. Open **main.less** and add the following line:

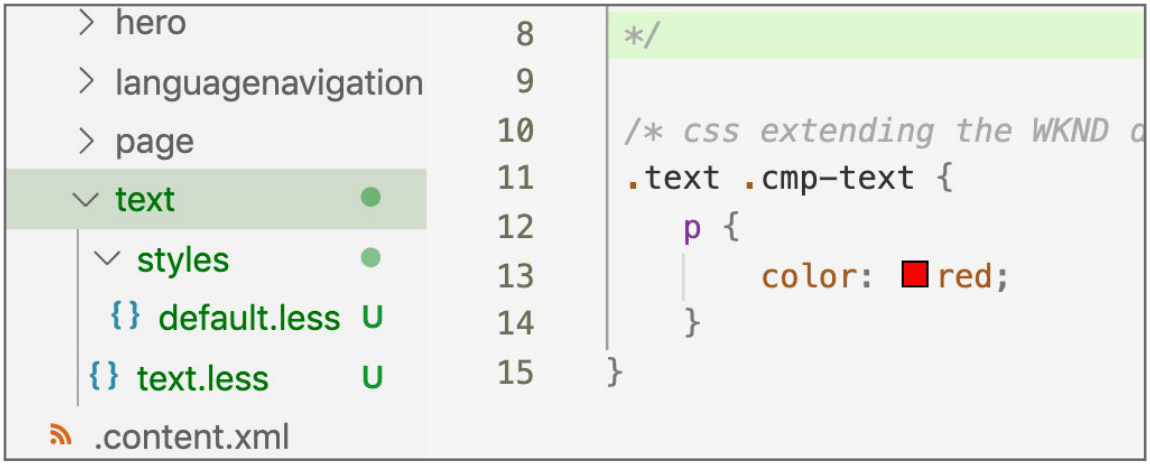
```
@import "components/text/text.less";
```

14. Save **main.less** and close.
15. Go back to the browser tab http://localhost:3000/editor.html/content/training/us/en.html and observe the text color has changed to blue ,as shown:



You should see the browser showing the proxy page change and messages regarding the styling change in the aemfed window.

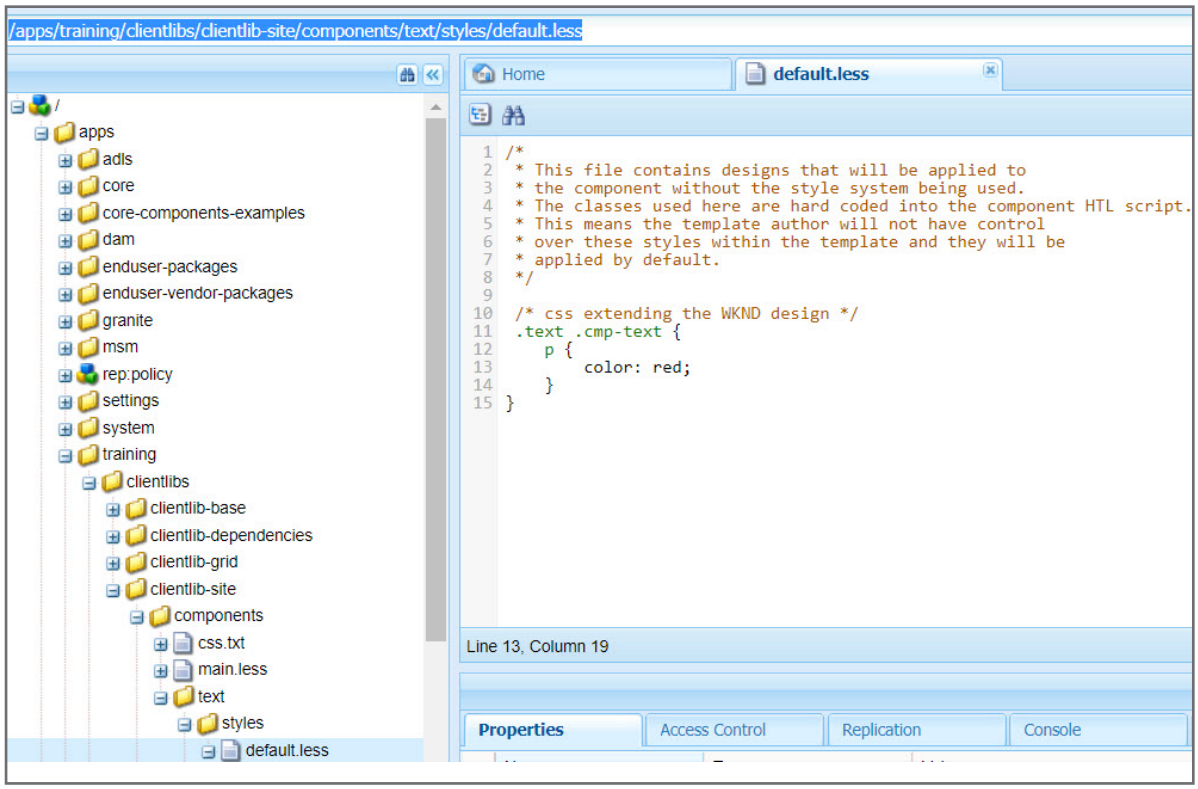
16. Using the IDE of your choice, navigate to **clientlib-site/components/text/styles** and open **default.less**. Update the style to a color of your choice, such as **red**.



You will notice messages in the aemfed window each time you open or modify a file.

```
[Browsersync] Reloading Browsers...
ADD jcr_root/apps/trainingproject/clientlibs/clientlib-site/components/text/text.less
  Deploying to [localhost:4502] in 82 ms at 2019-08-23T15:49:22.486Z: OK
[localhost:4502] Only styling was changed, try to inject
[Browsersync] File event [change] : /apps/trainingproject/clientlibs/clientlib-base.css
[Browsersync] File event [change] : /apps/trainingproject/clientlibs/clientlib-site.css
ADD jcr_root/apps/trainingproject/clientlibs/clientlib-site/components/text/styles/default.less
  Deploying to [localhost:4502] in 43 ms at 2019-08-23T15:49:24.228Z: OK
[localhost:4502] Only styling was changed, try to inject
[Browsersync] File event [change] : /apps/trainingproject/clientlibs/clientlib-base.css
[Browsersync] File event [change] : /apps/trainingproject/clientlibs/clientlib-site.css
```

17. Using **CRXDE Lite**, navigate to **/apps/training/clientlibs/clientlib-site/components/text/styles/default.less** and verify that the files were uploaded to AEM.



You should see the browser showing the proxy page change and messages regarding the styling change in the aemfed window.

TIP: If you did not see the expected behavior try the following:
Clear browser cache:
1. Open Settings in the browser. In Chrome, it is the three vertical dots at the end of the address bar and then choose Settings from the menu.
2. In the left rail, click **Advanced** to open the menu.
3. Click **Privacy and Security** from the Advanced menu.
4. Click **Clear browsing data** from the **Privacy and security** dialog.
5. From the list in the popup window, check the checkbox for **Cached images and files** and click **Clear data**.
6. Navigate to: http://localhost:4502/content/training/en.html. You should see the text rendered in the desired color.
7. You can also check by right-clicking on the page and selecting View source. Search for <link rel="stylesheet" href="/etc.clientlibs/training/clientlibs/clientlib-base.css" type="text/css"> in the source output.
8. Click on the css link.The concatenated css file should open.
9. Search for the element that you set in **default.less** and verify the color value.

Clear Server cache:
1. Using **CRXDE Lite**, verify that the training site is included in the dependency property for **/apps/training/clientlibs/clientlib-base**.
2. In your browser, navigate to: http://localhost:4502/libs/granite/ui/content/dumplibs.html
3. At the top of the page, select Click **here** for rebuilding all libraries.
4. Click **Invalidate Caches** to mark the client library caches invalid and force a recompile.

Caution: Do not click Rebuild Libraries. This is an unnecessary and extensive process and will take a long time.
Tip: To force a single library to rebuild, rename the cq:clientLibraryFolder node, in this case the /apps/training/clientlibs/clientlib-base node.
5. Navigate to http://localhost:4502/content/training/en.html. You should see the text rendered in the desired color. You should also see the correct color in the aemfed proxy rendered page.

Now, test the Sling Log Tracer.

18. Using your IDE, change the font color to green and remove the semicolon at the end of the line to force an error. Save the file.

```
[Browsersync] File event [change] : /apps/trainingproject/clientlibs/clientlib-base.css
[Browsersync] File event [change] : /apps/trainingproject/clientlibs/clientlib-site.css
[localhost:4502] Tracer output for [/etc.clientlibs/trainingproject/clientlibs/clientlib-base.css?browsersync=1566576729776] (daf7a936-ceca-43ac-ace6
-8a7b02436c93)
[ERROR] LessCompilerImpl: failed to compile less /apps/trainingproject/clientlibs/clientlib-site/main.less: ParseError: Unrecognised input in /apps/t
rainingproject/clientlibs/clientlib-site/components/text/styles/default.less on line 6, column 5:
5
6     p {
7         color: green

Local source: /Adobe/Archetype19/training/ui.apps/src/main/content/jcr_root/apps/trainingproject/clientlibs/clientlib-site/components/text/styles/def
ault.less:6:5
```

Note: The Sling Log Tracer outputs the less compiler error message and displays the offending line number in the file. There is no need to go digging through the error log to find the problem.

TIP: If you did not see the expected behavior, try the following:
Rebuild the client libraries:
If the recompile is not attempted, the error in default.less will not be caught. In this case, you can try two options:
1. Rename the cq:clientLibraryFolder node; (in this case: /apps/training/clientlibs/clientlib-base) to force a recompile.
2. If that does not produce the desired results, navigate to http://localhost:4502/libs/granite/ui/content/dumplibs.html.
3. At the top of the page select Click here for rebuilding all libraries.
4. Click Rebuild Libraries. Now, if you change the default.less file again and deliberately introduce an error, the system should attempt a recompile and the error will be reported by the Sling Log Tracer and logged to the aemfed command window.

19. Using your IDE, correct the error and notice the font color change.

Congratulations! You have used aemfed to assist with AEM front-end development.

References

You can use the following links for more information on:

- aemfed: https://aemfed.io/