Adobe

Student Workbook

Adobe Experience Manager Technical Basics

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.

Objectives

After completing this course, you will be able to:

- Describe the applications in Adobe Experience Manager
- Describe the architecture of AEM
- Understand Cloud Manager for AEM
- Explain how AEM fits into the Adobe Experience Cloud

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

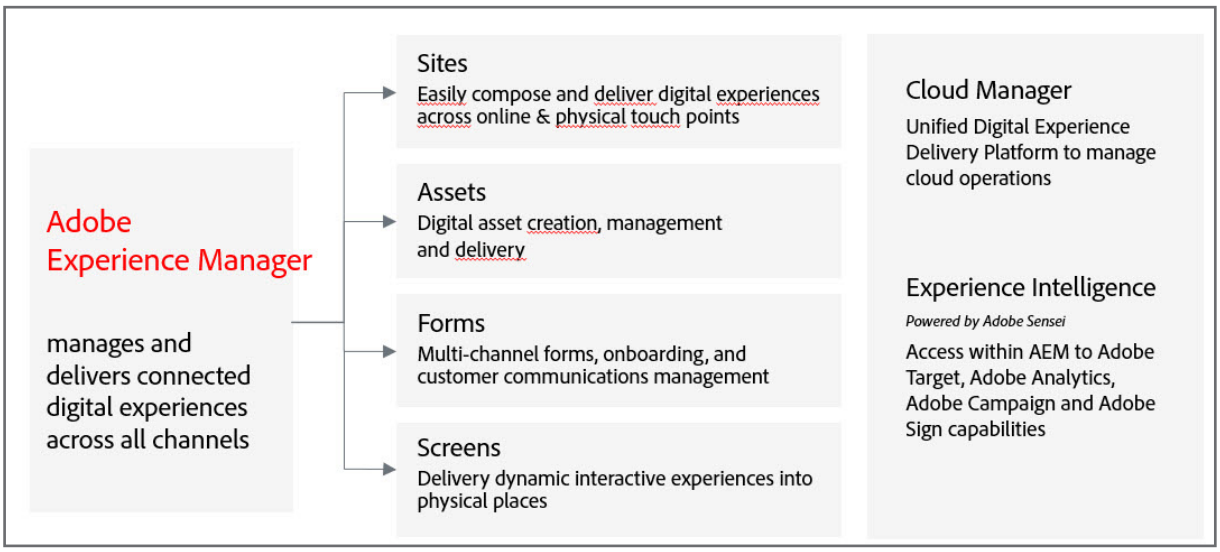
AEM Forms combine form authoring, management, and publishing along with correspondence management capabilities, document security, and integrated analytics to create engaging end-to-end experiences. Designed to work across web and mobile channels, AEM Forms can be efficiently integrated into your business 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 as a Cloud Service introduces the next generation of the AEM product line. It builds on top of two decades of constant investments, preserving and extending all AEM use cases and functionalities, while turning the product into a cloud native solution.

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.

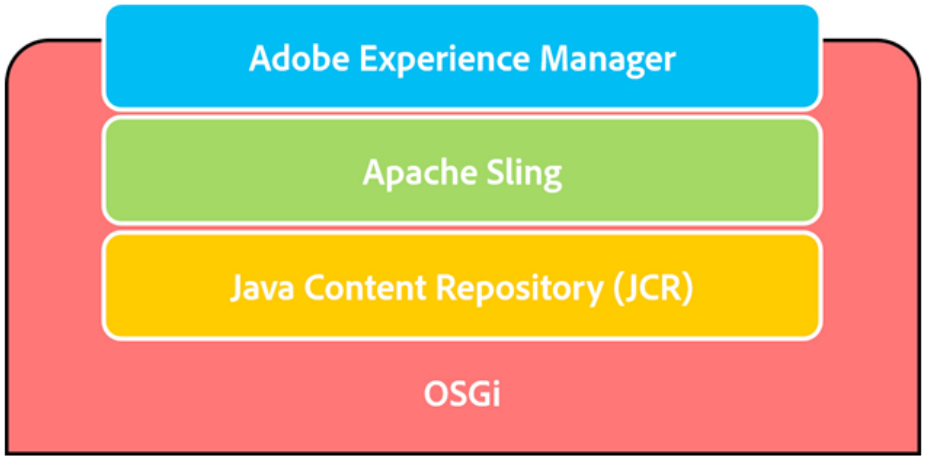
To learn more about AEM Architecture, go to:

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 plug-in-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.

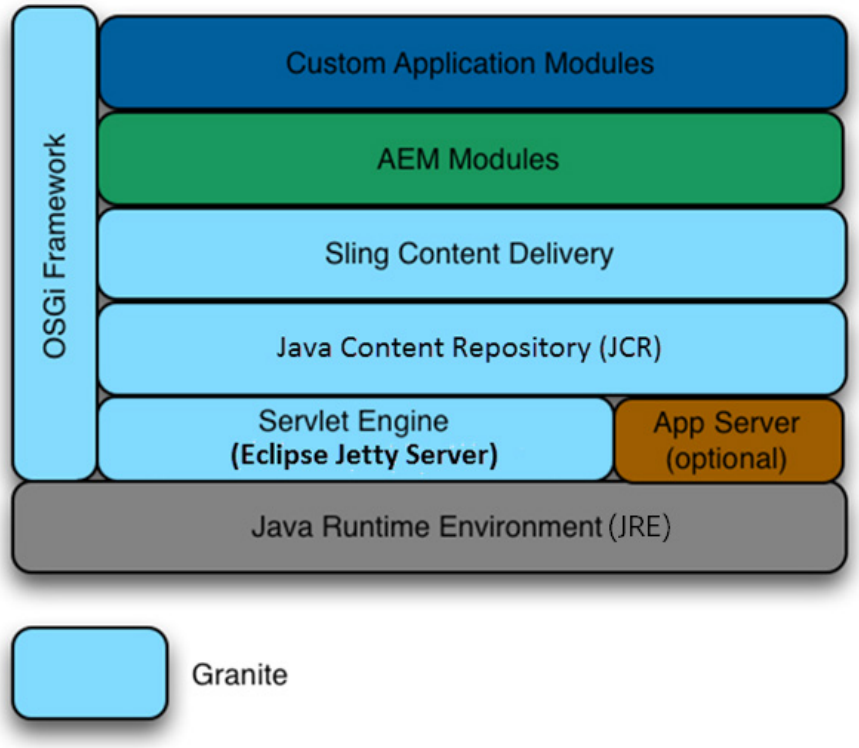
OSGi Framework

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.

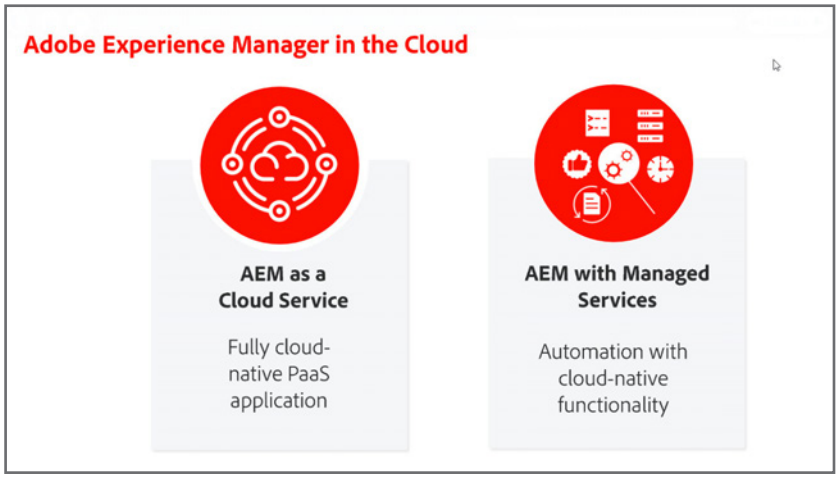
AEM as a Cloud Service

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

You can use the following links for more information on:

- Architecture:
  - Dispatcher in the Cloud: https://docs.adobe.com/content/help/en/experience-manager-cloud-service/implementing/dispatcher/overview.html
  - Manage Environments - Cloud Service: https://docs.adobe.com/content/help/en/experience-manager-cloud-service/implementing/using-cloud-manager/manage-environments.html
  - 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/

Install 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.

Objectives

After completing this course, you will be able to:

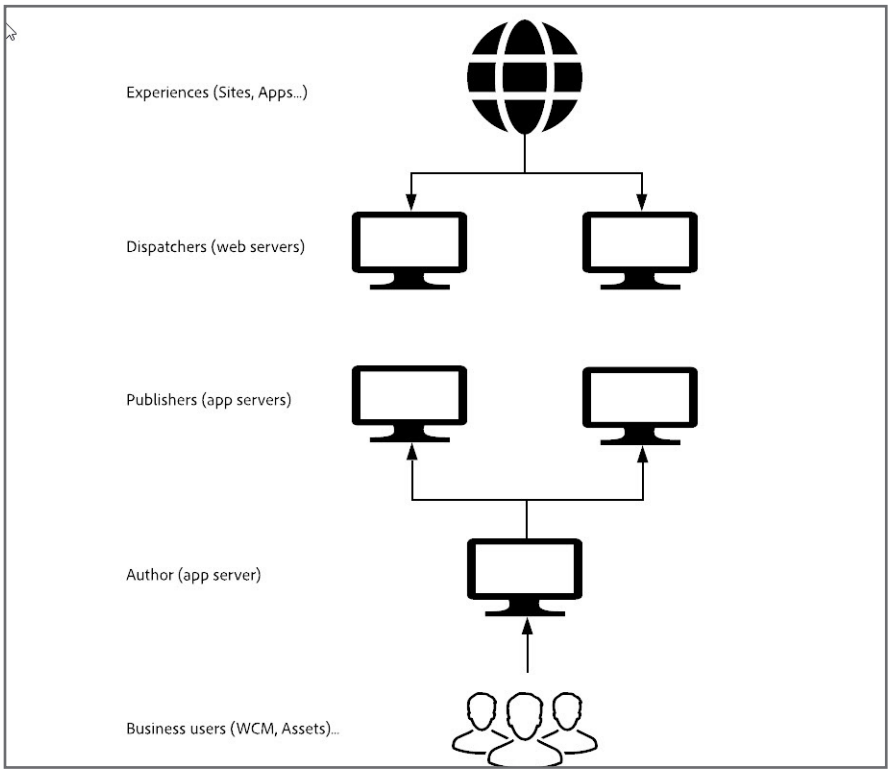
- Understand local installation fundamentals
- Install AEM as a Cloud Service Locally
- Install AEM 6.5.x Locally
- Install a service pack

Local Install

Local quickstart runtime: The quickstart file runs a local version of AEM. You can use this quickstart jar to run either an Author Service or Publish Service locally.

- **Author**: A Service used to create, upload, and edit content and administer the website. After the content is ready to go live, it is replicated to the publish Service. A developer might install a publish service to check component layout and design on the final endpoint for content.
- **Publish**: A Service that serves the published content to the public.

The following graphic depicts an example AEM implementation:



Note: The Dispatcher is a static web server, such as Apache httpd and Microsoft IIS, augmented with the Dispatcher module. It caches webpages produced by the publish Service to improve performance.

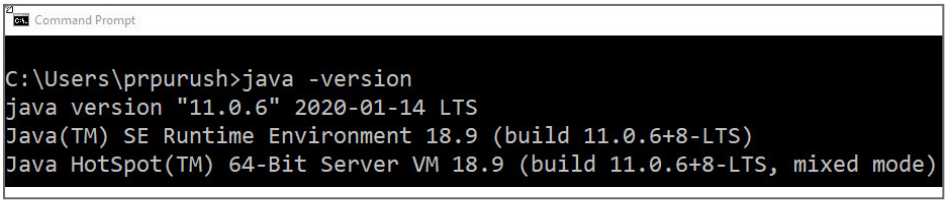
Installation Prerequisites

To install locally, you need:

- AEM quickstart JAR file
- A valid AEM license key properties file (only needed for AEM 6.5 and earlier)
- Java 11
- Approximately 4 GB of free space per Service
- Minimum 4 GB of RAM

During installation, you will notice that the JAR file creates a root folder on your system called crx-quickstart.

Note: Windows users may need to ensure their system's environment variables are set appropriately to run Java 11 before installing AEM. Open a command prompt and type **java -version** to ensure Java is set up properly. When you run the command, it should show the following result (java version "11.0.6") or higher:



You can download Java 11 for your system from the following link:

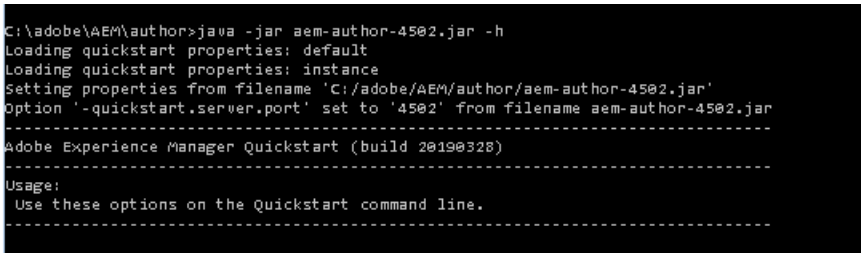
- https://downloads.experiencecloud.adobe.com/content/software-distribution/en/general.html

Installing the quickstart jar

Installation is done using 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 will show all the available command-line options without starting the server.

Few 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.

In addition, you can optionally add java parameters depending on your local server requirement needs. You can start your Service (author or publish) by using the following parameters:

-Xms --> assigns the initial heap size

Default value	64 MB for a JVM running on 32-bit machines, or 83 MB for 64-bit machines
Recommended	Specific to physical memory available and expected traffic
Syntax	-Xms512m (sets the initial heap size to 512 MB)

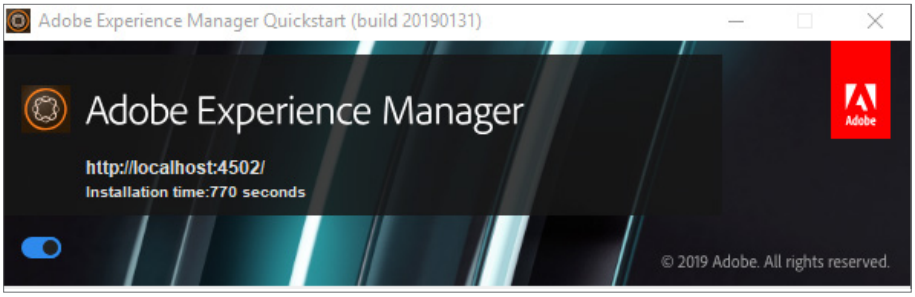
-Xmx --> assigns the maximum size to which the heap can grow

Default value	64 MB for a JVM running on 32-bit machines, or 83 MB for 64-bit machines
Recommended	Specific to physical memory available and expected traffic, but should be equal or greater than the initial size. To run AEM, it is recommended to allocate at least 1024 MB of heap size.
Syntax	-Xmx1024m (sets the maximum size for the heap. In the example, we are letting it grow to 1024 MB. However, in production, this should be higher because AEM consumes a lot of resources).

You can start AEM by using the following command:

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

A dialog box similar to the following (also known as the GUI) will pop up:



After AEM starts, your default browser will open a new tab automatically, pointing to the start URL (where the port number is the one you defined on installation).

Runmodes

A run mode is a collection of configuration parameters that allow you to tune your Service for a specific purpose. Run modes that are defined typically include the service (author and publish) and the environment (dev, stage, prod).

Examples:

- Author Service for a Dev environment
  `-r author, dev`
- Author Service for a Stage environment
  `-r author, prod`
- Publish Service for a Dev environment
  `-r publish, dev`
- All Author Services
  `-r author`

The syntax to specify multiple run modes is:

```
-r runmode1, runmode2, …
```

Examples:

```
java -jar aem-sdk-quickstart.jar -r author,dev -gui
java -jar aem-sdk-quickstart.jar -r author,prod -gui
java -jar aem-sdk-quickstart.jar -r publish,prod -gui
```

Note the use of the -r parameter. This starts the quickstart jar with the specified runmode. The -gui option turns on the GUI mode that shows local server window on your system.

Updating the QuickStart Jar for Cloud Service

Adobe Experience Manager as a Cloud Service releases daily updates and, thus, it is important for local development to be developing against QuickStart Jar version that is near to what is deployed to the Adobe Experience Manager as Cloud Service environments. It is recommended the local QuickStart JAR used for local development be refreshed at least biweekly to match the version of Experience Manager on the Production environment.

Exercise 1: Install the SDK Quickstart Jar

Note: You will perform this exercise only if you are using AEM Cloud Service.

Scenario: As a developer or administrator, you need to work with Adobe Experience Manager locally, using the quickstart JAR file and license.properties.
In this exercise, you will install and start an author dev service on port 4502. In typical development, you will want to download the latest build of the SDK from the public repo.
Location of SDK builds: https://downloads.experiencecloud.adobe.com/content/software-distribution/en/aemcloud.html

Note: In order 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**.

Prerequisite: Make sure you have Java 11 installed and set up on your system. The SDK requires Java 11 SDK to support the development tooling.

Note: You can repeat the exact steps below to also start a Publish Service. Simply replace "author" with "publish" throughout this exercise.

1. If you are using ReadyTech, the SDK has already been downloaded for you under **C:/adobe/aem-sdk/.**

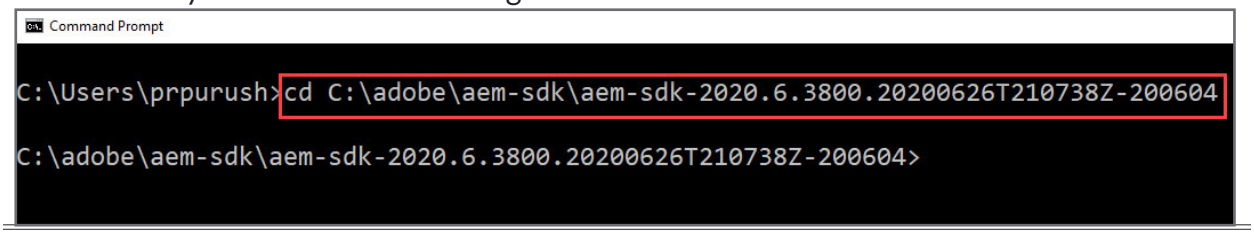
Note: If you are using local machine, then after downloading/receiving the SDK zip, move the .zip file to a location from where you want to start your AEM service. For example, in:
Windows: **C:/adobe/aem-sdk/**
MacOS X: **\Applications\adobe\aem-sdk** or ***x: \opt\adobe\aem-sdk**

2. Extract the Quickstart jar file from the AEM SDK zip file.
   For example: **C:/adobe/aem-sdk/aem-sdk-<nnnn>/**

3. Rename the folder **aem-sdk-<nnnn>** to **author-<nnnn>**.
4. Open the renamed **author-<nnnn>** folder to find the quickstart jar **aem-sdk-quickstart-<nnnn>.jar**.
5. Rename the file **aem-sdk-quickstart-<nnnn>.jar** to **aem-sdk-quickstart.jar.** For simplicity, remove the build number from the jar file.
   a. aem = Application
   b. sdk-quickstart = Local Cloud quickstart
   c. 2020.6.3800.20200626T210738Z-200604 = the build of this SDK

Note: Only change the jar file name for convenience. 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.

6. Open a Command Prompt and navigate to the directory on your machine where you unzipped the SDK and your new folder exists:
   `cd C:\adobe\aem-sdk`
7. Start typing **cd aem** and press <TAB>. This will auto-complete the **aem-sdk-<NNNN>** folder name for you. Press <Enter> to navigate to the folder.

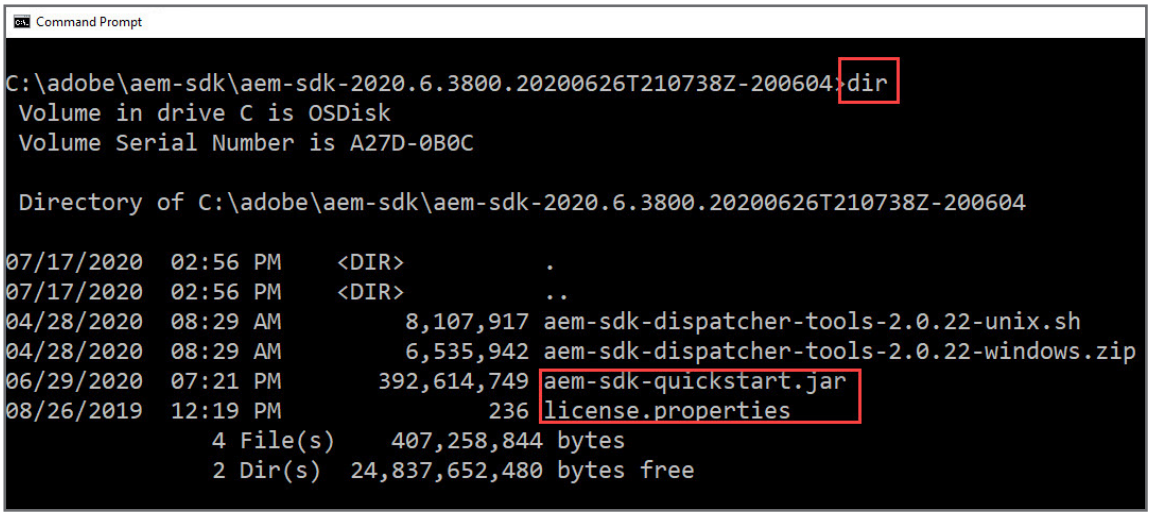


Note: If you are using a ReadyTech Service, this directory should be **C:\adobe\aem-sdk\aem-sdk-<nnnn>** where nnnn is the build number.

8. Execute the following command to verify you are in the correct directory where the SDK quickstart jar and the licence.properties file resides.

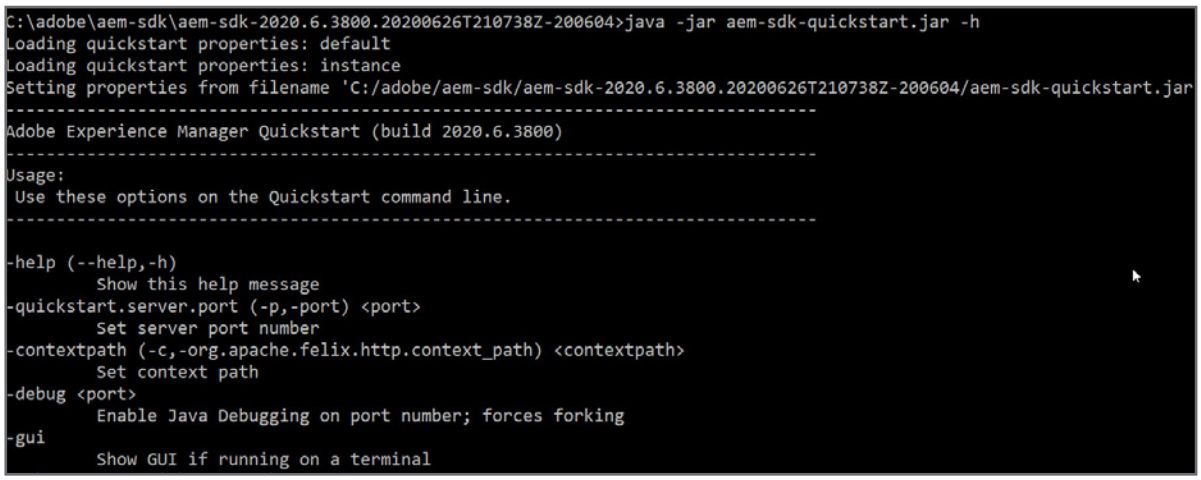
   Windows: **dir**

   Mac: **ls**



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

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



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

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

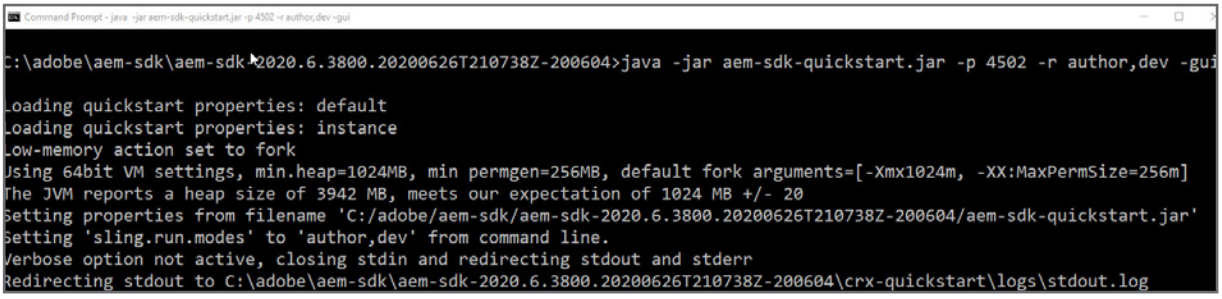
Note: In the above command:

-p specifies the port number for the local aem quickstart

-r specifies the runmode that you are running locally

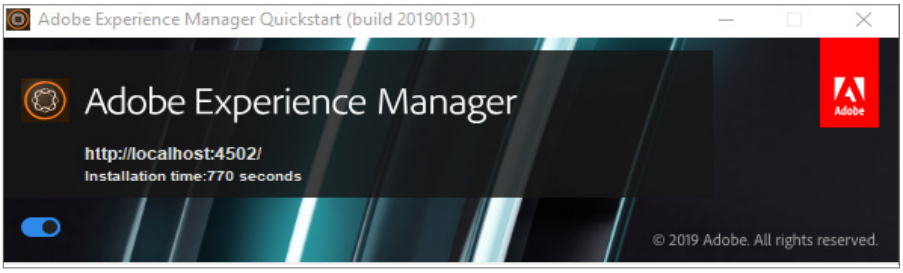
-gui opens the GUI window on your desktop.

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.

11. Verify your author service has started. The command window will show startup details and the GUI window will show progress. Installation will take approximately 5–7 minutes depending on your system's capabilities. After the Author Service has started successfully, the start-up screen (the GUI) will change 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.

12. Each time you want to run your author Service, use the same command in step 9 to start it. However, this 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 will automatically open 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 below:

13. 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**
14. 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.

Exercise 2: Install 6.5 Quickstart Jar and Service Pack

Note: You will perform this exercise only if you are using AEM 6.5.

Scenario: As a developer or administrator, you need to work with Adobe Experience Manager locally, using both the quickstart JAR file and command line method.

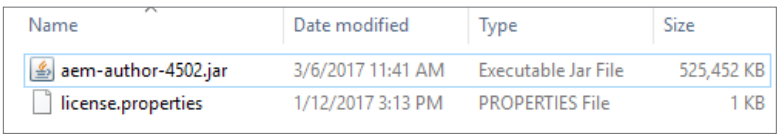
In this exercise, you will install the 6.5 Quickstart jar and a Service Pack. You will then start your author Instance on port 4502.

Note: If you are attending a v/ILT class using ReadyTech, steps 1 through 3 below were completed for you. Skip ahead to step 4.

Note: You can repeat the exact steps below to also start a Publish Instance. Simply replace "author" with "publish" in the directions below to install a publish instance.

To install an author Instance:

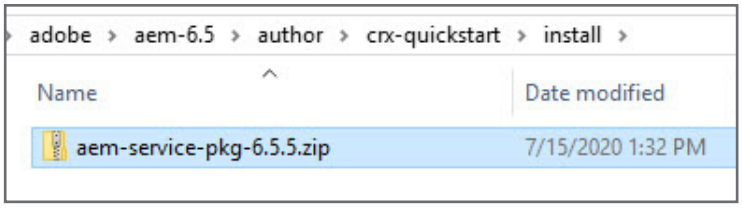
1. Create a folder structure on your file system where you will store, install, and start your author instance. For example, in:
   a. Windows: **C:/adobe/aem-6.5/author**
   b. MacOS X: **/Applications/adobe/aem-6.5/author** or ***x: /opt/adobe/aem-6.5/author**
2. If not already present in the /author 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-author-4502.jar:**
   a. aem = Application
   b. author = Web Content Management (WCM) mode AEM will run in (in this case, author)
   c. 4502 = Port AEM will run in



You can, therefore, control the way AEM is installed by defining properties in a filename.

To include the Service Pack for AEM 6.5:

4. Navigate to **/adobe/aem-6.5/author folder**, create **crx-quickstart** within the author folder.
5. Open the **crx-quickstart** folder and create **install** folder within it.
6. Place the package into **../crx-quickstart/install** folder, as shown:



Note: Once AEM is installed, it recognizes this install directory and automatically installs the service pack..

Use the command line to install AEM:

7. Open a Command Prompt and navigate to the directory that contains the jar and license file:
   `cd C:\adobe\aem-6.5\author`
8. In the command line, run the command below to see the different parameters that you can use to run the quickstart:

```
java -jar aem-author-4502.jar -h
```

```
C:\adobe\aem-6.5>java -jar aem-author-4502.jar -h
Loading quickstart properties: default
Loading quickstart properties: instance
Setting properties from filename 'C:/adobe/aem-6.5/aem-author-4502.jar'
Option '-quickstart.server.port' set to '4502' from filename aem-author-4502.jar
--------------------------------------------------------------------------------
Adobe Experience Manager Quickstart (build 20190328)
--------------------------------------------------------------------------------
Usage:
 Use these options on the Quickstart command line.
--------------------------------------------------------------------------------

-help (--help,-h)
        Show this help message
-quickstart.server.port (-p,-port) <port>
        Set server port number
-contextpath (-c,-org.apache.felix.http.context_path) <contextpath>
        Set context path
-debug <port>
        Enable Java Debugging on port number; forces forking
```

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

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

Note: In the above command:

-p specifies the port number for the local aem quickstart

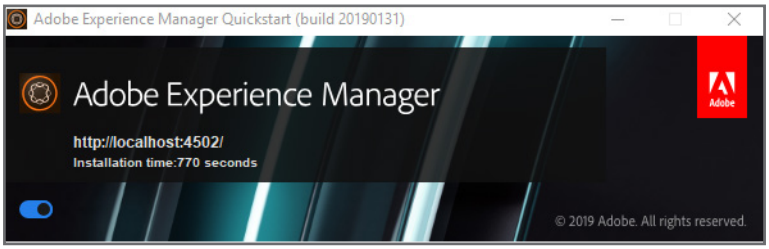
-r specifies the runmode that you are running locally

-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-author-4502.jar** file to start the author instance. But double-clicking does not add the dev runmode to the instance.

10. The command line input contained -gui parameter that shows the installation GUI. At this point you should see it installing the quickstart.



11. Installing base 6.5 will take anywhere from 5-7 minutes. You will see this loading on the Gui that appears and the browser UI open. 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.

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

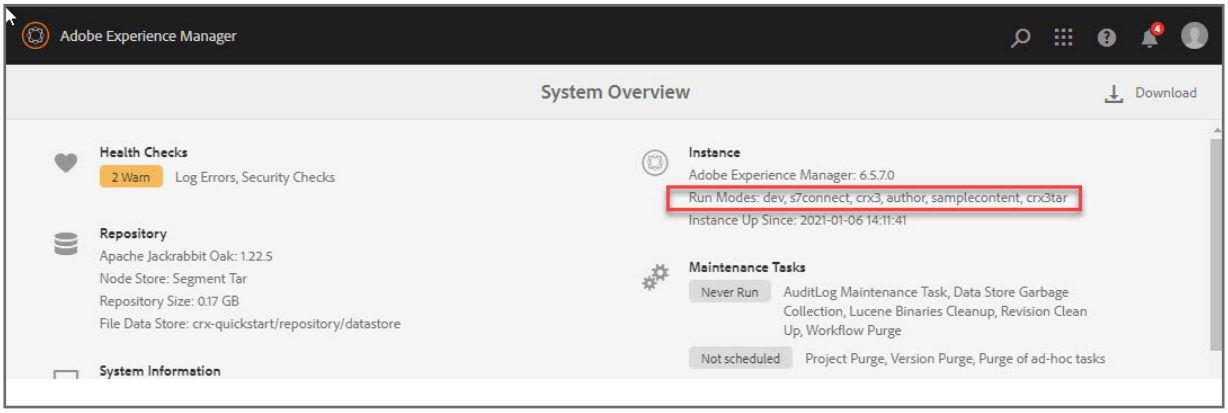
Note: Each time you want to run your author instance, use the same command in step 6 to start it. However, this time, the startup will be as fast as one minute or less, as the initial installation task has been performed.

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

13. 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**
14. Navigate to **Tools** > **Operations** > **System Overview** in your author instance. The **System Overview** screen appears.

Note: A Product Navigation pop-up window may 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.

15. Verify the Run Modes for the Service in the **Instance** area, as shown:



16. 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.

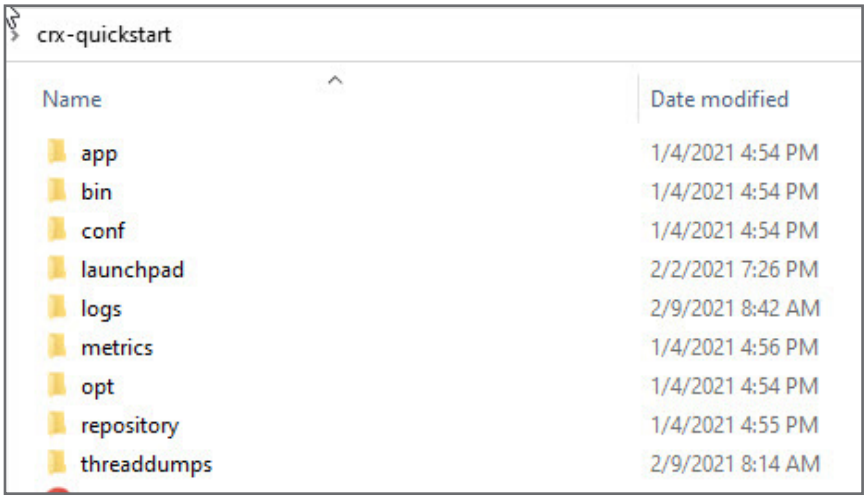
Note: In the installation folder, notice how a **crx-quickstart** directory is also created on your computer, as shown below:

Name	Date modified	Type	Size
crx-quickstart	3/6/2017 1:40 PM	File folder	
aem-author-4502.jar	3/6/2017 11:41 AM	Executable Jar File	525,452 KB
license.properties	1/12/2017 3:13 PM	PROPERTIES File	1 KB

This is the extracted repository that is created upon installation.

---

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



Local Log Files

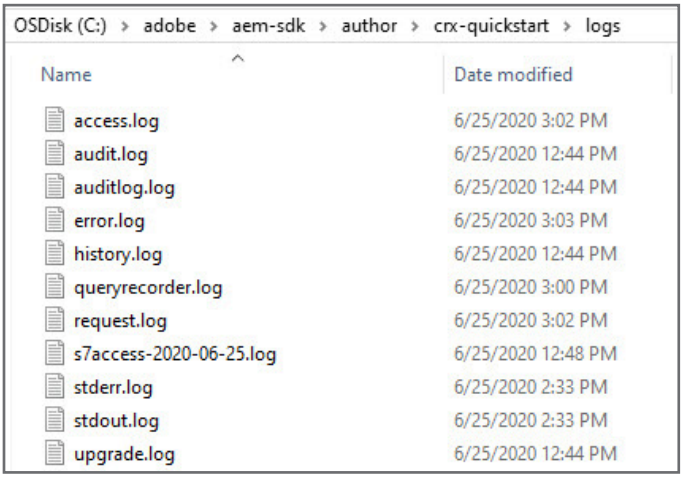
The logging framework in AEM is based on Apache Sling. The root logger defines the global settings for the logging system. The root logger is configured by using Apache Sling Logging Configuration. The org.apache.slng.commons.log bundle manages the logging system. This bundle helps:

- 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

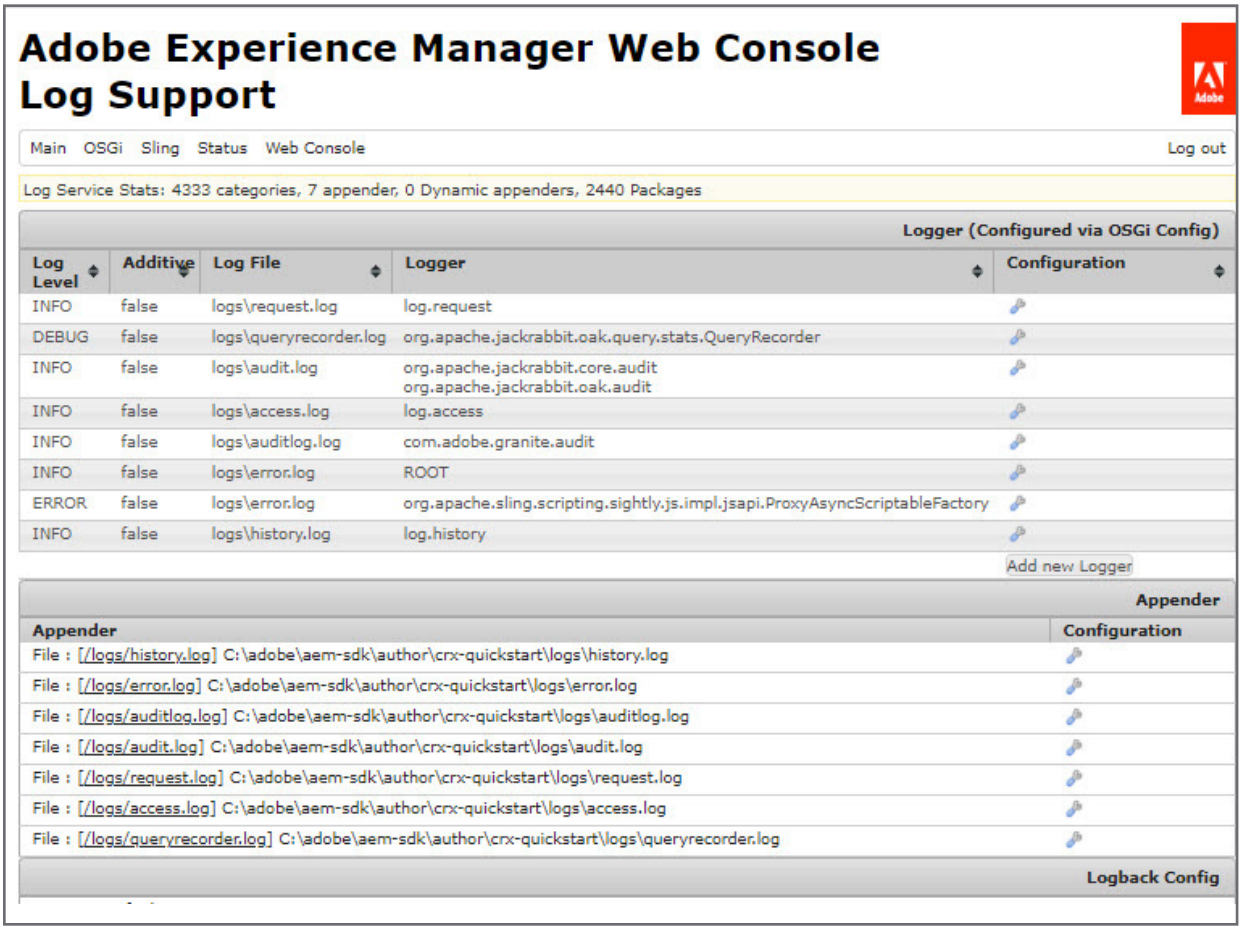
These files are available in the **/crx-quickstart/logs** installation directory.



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

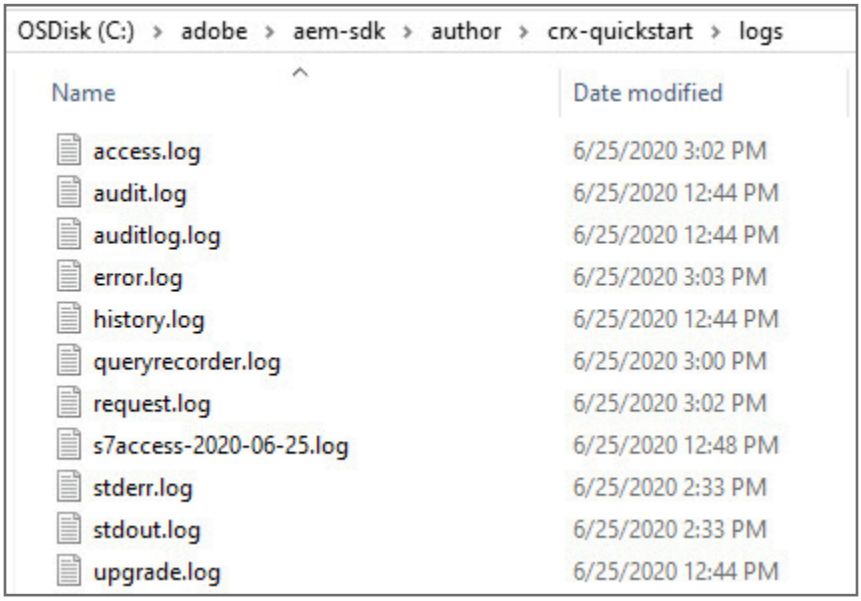
Appender	Configuration
File : [/logs/history.log] C:\adobe\aem-sdk\author\crx-quickstart\logs\history.log	
File : [/logs/error.log] C:\adobe\aem-sdk\author\crx-quickstart\logs\error.log	
File : [/logs/auditlog.log] C:\adobe\aem-sdk\author\crx-quickstart\logs\auditlog.log	
File : [/logs/audit.log] C:\adobe\aem-sdk\author\crx-quickstart\logs\audit.log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

Exercise 3: Observe the local log files

In this exercise, 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:



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

You can use the following 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.

Objectives

After completing this course, you will be able to:

- Explain the features and UI elements of the AEM developer tools
- Install a package in AEM
- Create, build, and download packages
- Describe a content package structure
- Create an immutable package
- Create a mutable package

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>`

CRXDE Lite

CRXDE Lite is embedded into AEM, which allows you to perform common development and administration tasks within the browser. It is a light Integrated Development Environment (IDE) for quick access to the JCR. Because it is embedded in the server and is always available, CRXDE Lite is often the preferred tool for administrators and developers for working with nodes and properties in the JCR. It provides you with quick and direct access to the repository for monitoring, configuration, and development.

You can access CRXDE Lite directly by navigating to http://localhost:4502/crx/de/index.jsp or by navigating to **Tools** > **CRXDE Lite** within AEM.
For local development, Developers have full access to CRXDE Lite (/crx/de) and the AEM Web Console (/system/console). Note that 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.
Customers can access CRX/DE lite on the development environment (but not stage or production). Note that the immutable repository (/libs, /apps) cannot be written to at runtime, so attempting to do so will result in errors.
Developer Tools under /system/console will not be available in Skyline. Instead, a raw data dump tool will be exposed at a dedicated URL that outputs information to either the screen or as a file download.

With CRXDE Lite, you can access the Java Content Repository (JCR) of the AEM service and validate any value in the repository. CRXDE Lite can be used in development in combination with an IDE such as Eclipse, Visual Studio, or similar IDEs. You can also use it for quick developmental tests and training purposes.
CRXDE Lite is useful for:

- Code/content validation
- Product training
- Operations debugging
- Overlays from /libs

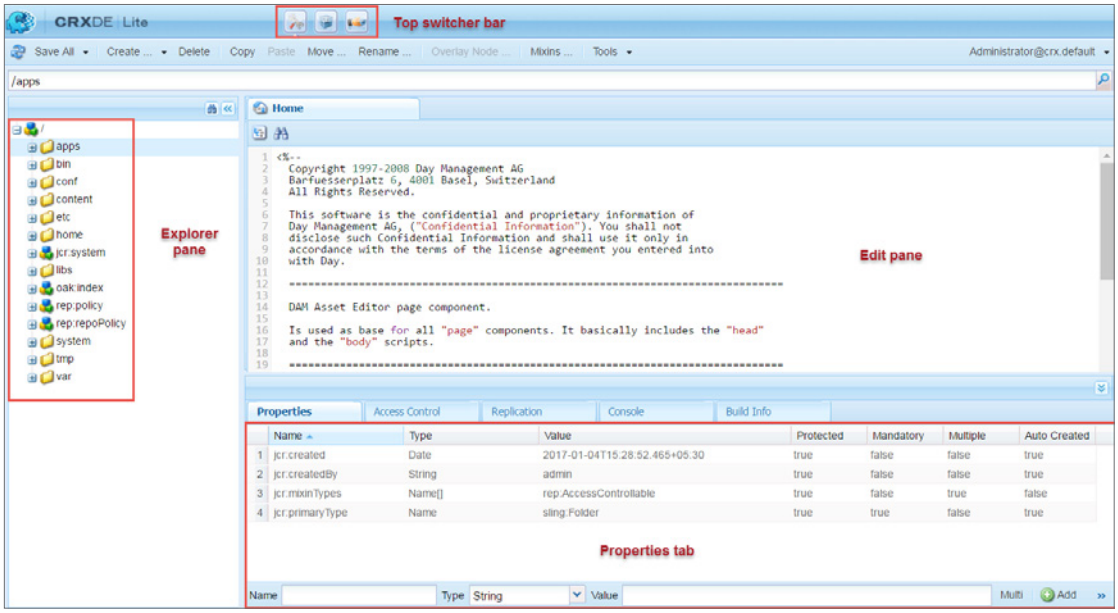
The CRXDE Lite UI contains:

- **Top switcher bar**: Enables you to quickly switch between CRXDE Lite, Package Manager, and Package Share.
- **Explorer pane**: Displays a folder tree structure of all the nodes in the repository.

You can perform the following actions on a node in the tree:

- Select the node and view its properties in the **Properties** tab. Examine all the JCR properties of different nodes.

- Right-click the node and perform an action on it, such as renaming the node, creating a new node, creating a folder, and creating a file.
- Edit the code. The **Edit** pane allows you to double-click a file, such as a .jsp or a .html file, in the **Explorer** pane to display its content. You can then modify the code and save the changes.
- View node properties. On the **Properties** tab, you can display and view the properties of the node that you selected. You can also add new properties or delete existing ones.



Tip: Bookmark the **CRXDE** Lite URL (http://localhost:4502/crx/de/index.jsp) in your browser to access this tool, as you will use it often in your training as well as in your role as an AEM developer or administrator.

Web Console

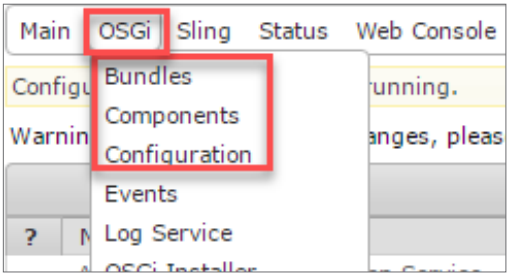
The Web Console in AEM is based on the Apache Felix Web Management Console, and is used to manage OSGi bundles and configurations. Any changes made through this console are automatically applied to the running system, without the need to restart the service. OSGi is where you manage your Java classes in the form of OSGi components and services.

You can access the console at http://localhost:4502/system/console.

Note: The Web console can only be accessed for local development.

The Web Console contains a selection of tabs for maintaining menu selections under the OSGi tab. These tabs include:

- **Bundles**: Used for installing and managing bundles.
- **Components**: Used for managing and controlling the status of components required for AEM.
- **Configuration**: Used for configuring an OSGi bundle and is the underlying mechanism for configuring AEM parameters.



Package Management in AEM

A package is a *.zip file that holds AEM repository content in the form of a file-system serialization called Vault serialization. Vault is provided by Apache Jackrabbit.

Packages provide an easy-to-use-and-edit representation of files, such as pages, assets, and folders. They also enable you to import and export repository content from one service or environment to another.

A package can contain:

- Page-related content
- Project-related content
- Assets-related content
- Vault meta information, such as filter definitions and import configuration information
- Other package information such as package settings, package filters, package screenshots, and package icons

You can use packages to perform any of the following tasks:

- Install new functionality
- Transfer content between services
- Export content to the local file system

You can work with packages by using Package Manager or Package Share. You can access this through the following link: http://localhost:4502/crx/packmgr/index.jsp.

Package Manager

Package Manager is used to import or export content on your service, transfer content between services, and back up repository content. With Package Manager, you can perform the following common tasks:

- Create, build, and download content packages
- Upload, validate, and install packages
- Modify existing packages
- View package information

You can also use filters to create a package containing page content or project-related content.

Creating and Building New Packages

When creating packages using Package Manager, you can apply rules and filters to determine the content a package should extract from the repository. After you define the content, you can build it. A package is created in a .zip file, which you can download to your local file system. You can also test the contents of the package *before* building it.

Package Manager provides you with many options to work with packages:

- **Rebuild**: Helps rebuild the package if there is a change in the repository content.
- **Edit**: Helps edit filters or rules applied to the package.
- **Test**: Helps perform a dry run of the installation.
- **Rewrap**: Helps recreate the package with additional information such as thumbnails and icons.

Downloading Packages to the File System

You can download a package by clicking the **Download** link. This link is displayed when the package details are expanded. After downloading the package, you can unzip the contents of the package to your local system.

Typically, an unzipped (extracted) content package contains the following folders:

- **jcr_root**: Contains files and folders that are serialized nodes and properties from the JCR.
- **META-INF**: Contains metadata regarding node definitions and the filter.xml file that gives directions to Vault about the paths to include.

Package Share

Package Share is a centralized server where public packages are made available. These packages may include hotfixes, new functionality, updates, or documentation. You can search, download, and install any package either to your service or your local file system.

Within the Package Share, you have access to the following:

- AEM packages provided by Adobe (for example, new functionalities such as updated core components, hotfixes, and service packs)
- Shared packages provided by other organizations and made public by Adobe

You can access this console through the following link: http://localhost:4502/crx/packageshare/index.html

You can also directly access Package Share through the following link (after signing in using your Adobe ID): https://experience.adobe.com/#/downloads/content/software-distribution/en/aem.html

Deploying content packages via Cloud Manager and Package Manager

Customers deploy custom code to cloud environments through Cloud Manager. It should be noted that Cloud Manager transforms locally assembled content packages into an artifact conforming to the Sling Feature Model, which is how an AEM service application is described when running in a cloud environment.

Content packages written for AEM Cloud applications must have a clean separation between immutable and mutable content. If this separation does not exist, Cloud Manager will enforce it by failing the build.

Immutable content packages

All code persisted in the immutable repository (/apps) must be checked into Git and deployed through Cloud Manager. In other words, unlike current AEM solutions, code is never deployed directly to a running AEM service. This ensures that the code running for a given release in any Cloud environment is identical, which eliminates the risk of unintentional code variation on production. As an example, OSGI configuration should be committed to source control rather than managed at runtime via the AEM Web Console's configuration manager.

Mutable content packages

Content such as folder path hierarchies, service users, and ACLs are typically committed into a Maven archetype-based AEM project. Techniques include exporting from AEM or writing directly as XML. During the build and deployment process, Cloud Manager generates the resulting mutable content package and installs it, writing out all the previous content nodes.

The full list of mutable content includes:

- service users (add, modify, remove)
- service user ACLs (add, modify, remove)
- folders (add, modify, remove)
- node types (add, modify, remove)
- index definitions (add, modify, remove)
- editable templates (add, modify, remove)
- script (packages can trigger Install hooks at various stages of the install process of package installation)

Content packages are deployed to all environment types (dev, stage, prod); however, it is not possible to limit deployment to a specific environment.

Also, there is no mechanism to rollback the mutable content package changes after they have been applied. If customers detect a problem, they can choose to fix it in their next code release or, as a last resort, restore the entire system to a point in time *before* the deployment.

Exercise 1: Install a package

Scenario: As an AEM developer or administrator/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 that 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 exercise 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:

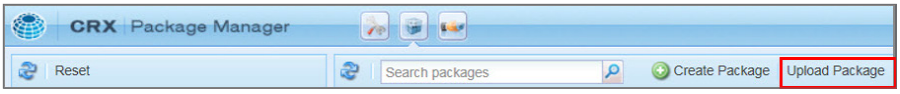
1. If your AEM instance is not running, start it.
2. Download the aem-guides-wknd.all-#.#.#.zip content package using the URL: https://github.com/adobe/aem-guides-wknd/releases

Note: The #.## indicates the version of the WKND site you will use. The instructor will provide this information to you. The version in the screenshots might be different than the version you are given to use.

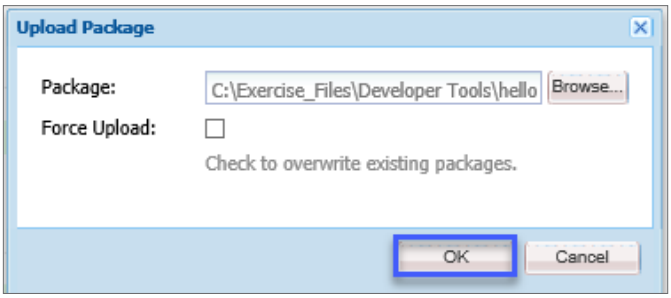
Note: If you are using ReadyTech, **aem-guides-wknd.all-#.#.#.zip** can be found on the desktop. If you are working locally and cannot download from the link above, your instructor can provide this zip for you.

To validate a package:

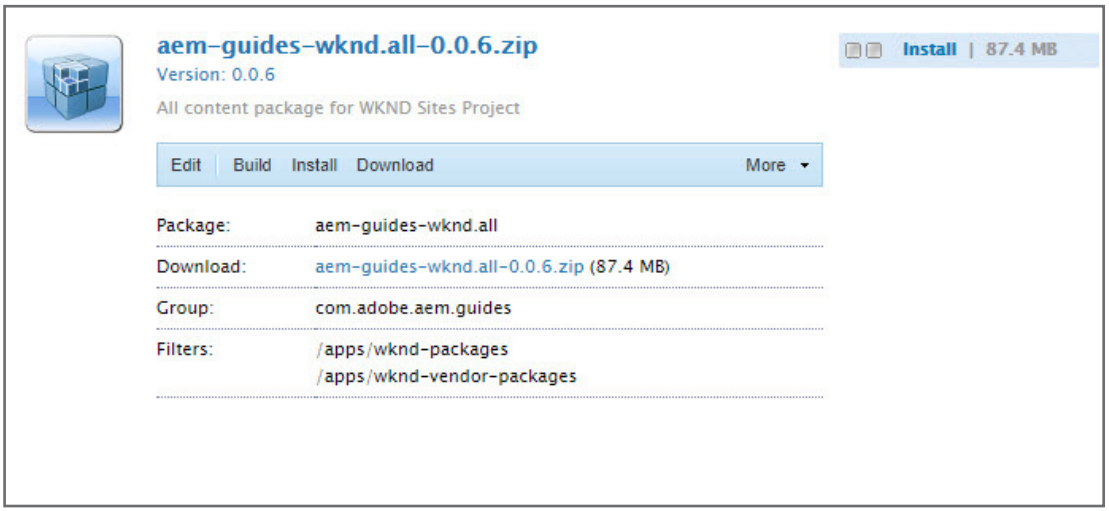
3. Verify you are logged on to your AEM author service on port 4502 (http://localhost:4502).
4. Navigate to http://localhost:4502/crx/packmgr/index.jsp. This will take you to the AEM Package Manager tool.
5. Click **Upload Package.**



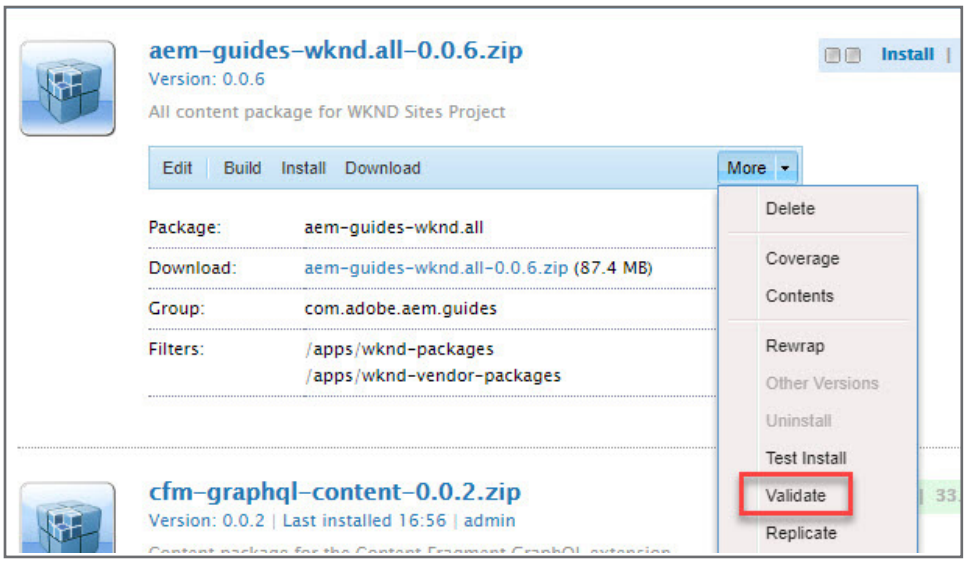
6. In the **Upload Package** dialog box, click **Browse** and select the **aem-guides-wknd.all-#.#.#.zip** package that you downloaded in Step 1.
7. Click **Open** and then click **OK** as shown:



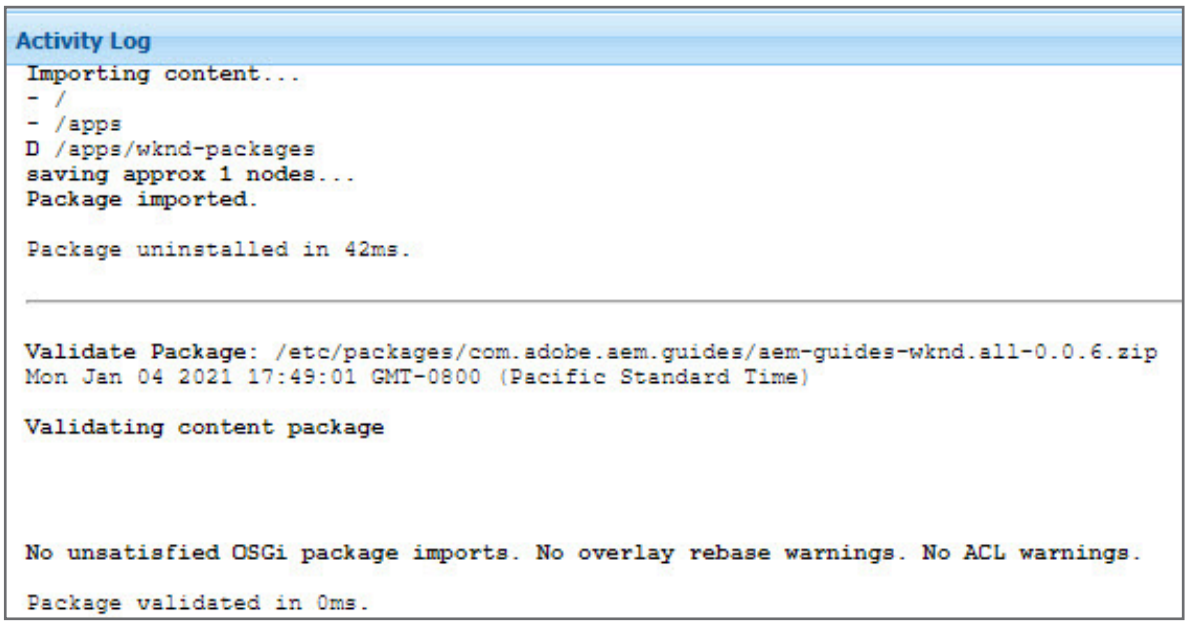
Your uploaded package is now available in AEM Package Manager, as shown.



8. Click **More** > **Validate** as shown:



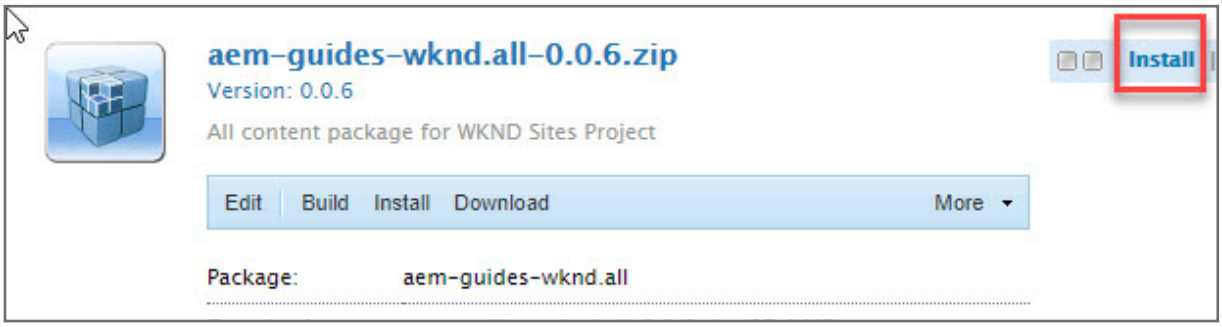
9. Leave all options selected as-is and click **Validate** in the dialog box. In the **Activity Log** below the **aem-guides-wknd.all-#.#.#.zip**, notice there are no issues with your package:



Note: The validation utility will notify you of any issues that impact overlaid JCR resources in /apps, any unsatisfied bundles, and any Access Control Lists (ACL) conflicts. In other words, this will prevent any problems with OSGi bundles that are unable to start after installing a package, any impacts on permissions (ACLs), and files in /libs that impact overlaid files in /apps.

To install your package:

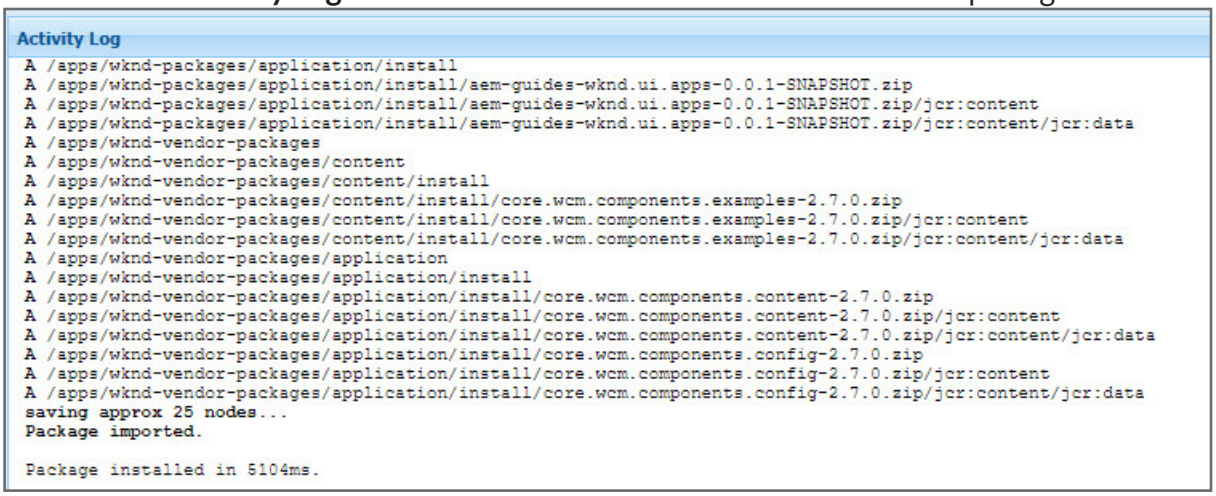
10. Click **Install**, as shown.



The **Install Package** dialog box appears.

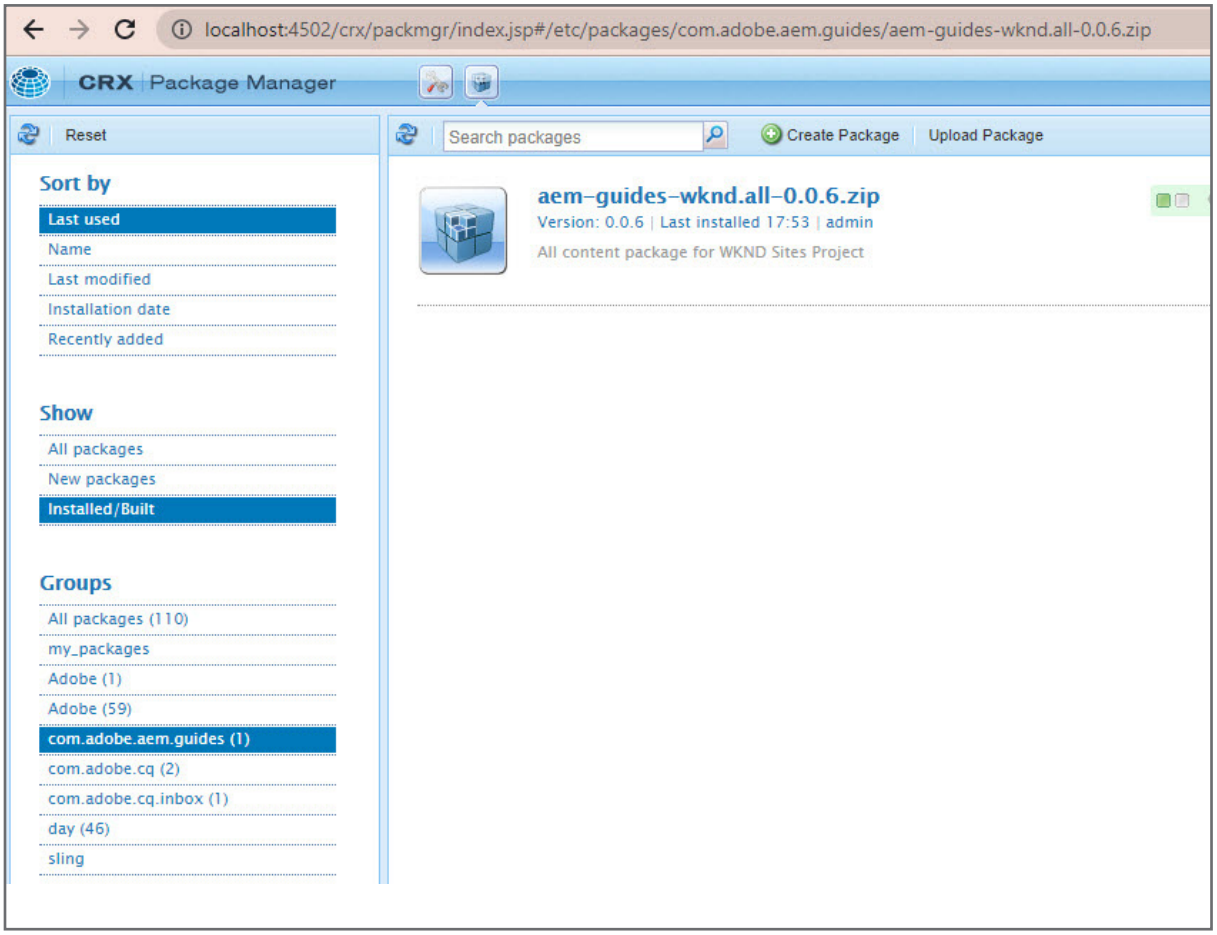
11. Ignore the **Advanced Settings** area and click **Install**.

12. 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.

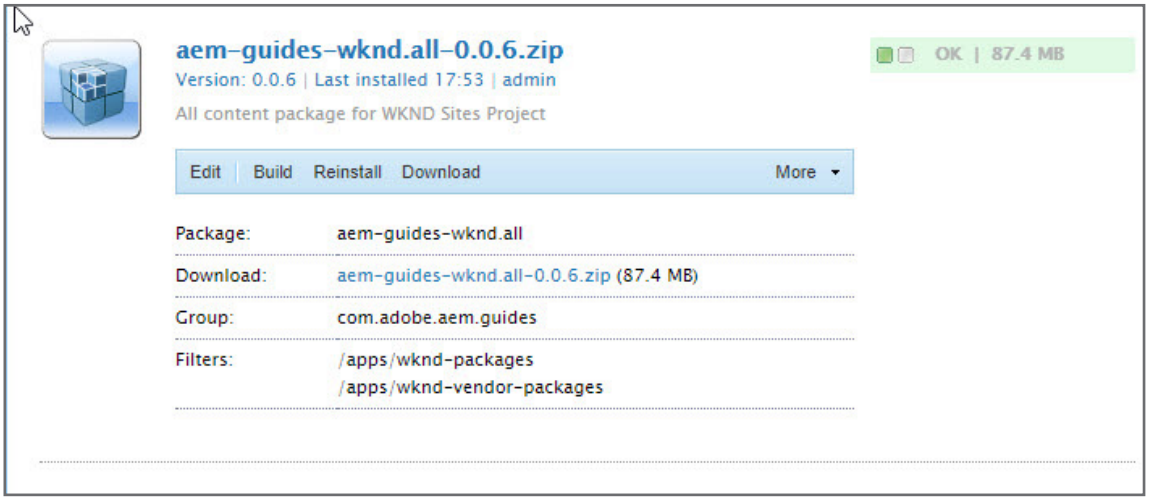
13. In the left navigation pane, under **Groups**, click on **com.adobe.aem.guides** and verify the container package, as shown:



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 in order for you to be able to see updated links (on the left side) as well as see the corresponding packages associated with the links on the left side.

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

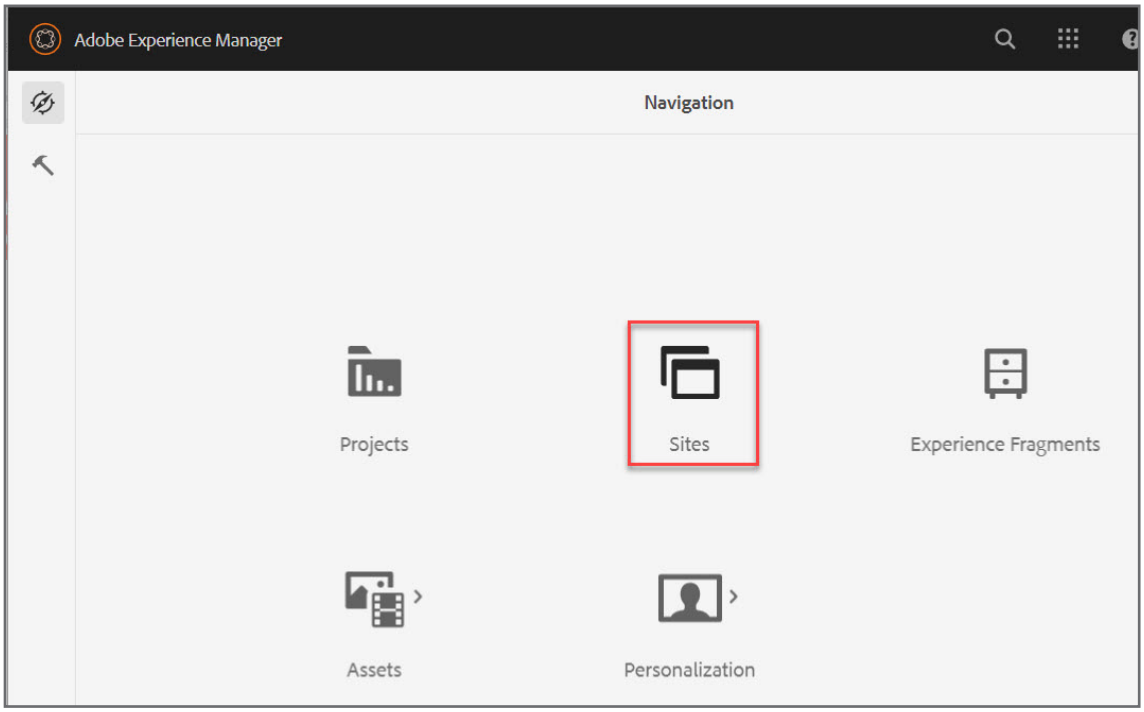
14. The **aem-guides-wknd.all-#.#.#.zip** contains all of the mutable content for the WKND project. Click the **aem-guides-wknd.all-#.#.#.zip** link and scroll down to the **Filters** section:



Task 2: Verify the newly installed site

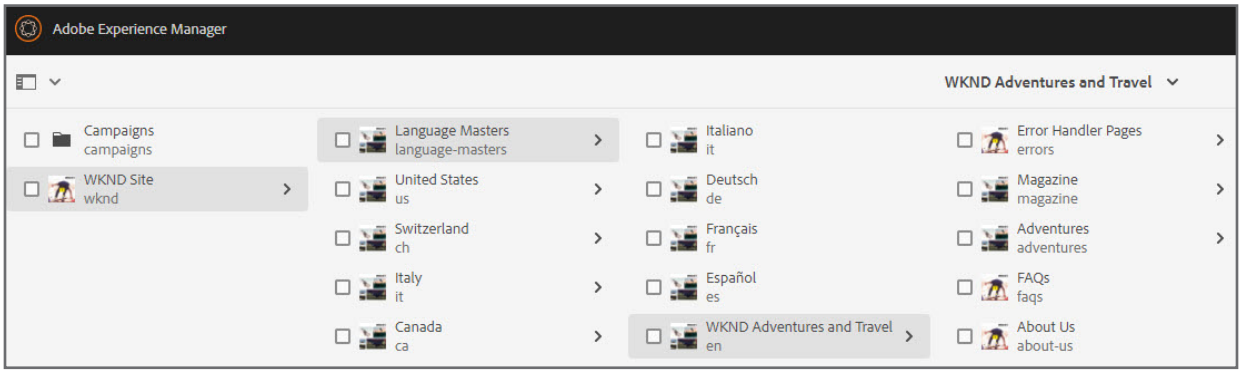
In this task, you will navigate to Sites and verify the newly installed site.

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.

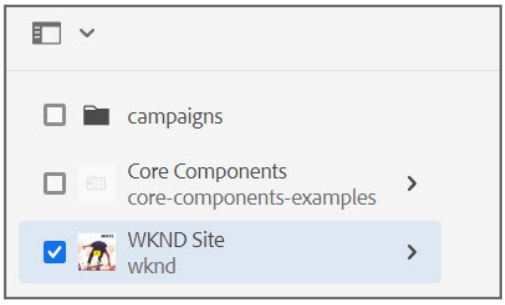


Note: You may see the **Product Navigation** tutorial dialog guiding you through the navigation. You may click **Next** to proceed through the tutorial and learn the basic AEM UI elements and navigation, or you may click **Close** to hide the tutorial.

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



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.



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

Exercise 2: Create, build, and download packages

In this exercise, 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 since mutable code (/apps) can only be installed to an AEM service via Cloud manager. Your instructor may provide you with a course-filter.txt file; you can use those filters rather than the example filters in this exercise..

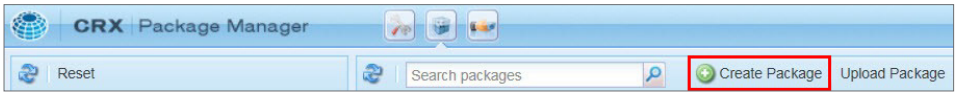
This exercise includes the following tasks:

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

Task 1: Create an immutable package

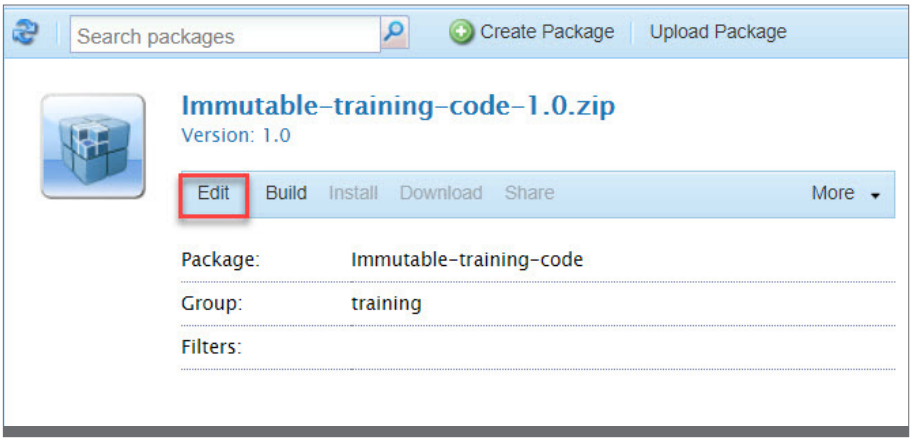
In this task, you will validate and install a package using the Package Manager.

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**.

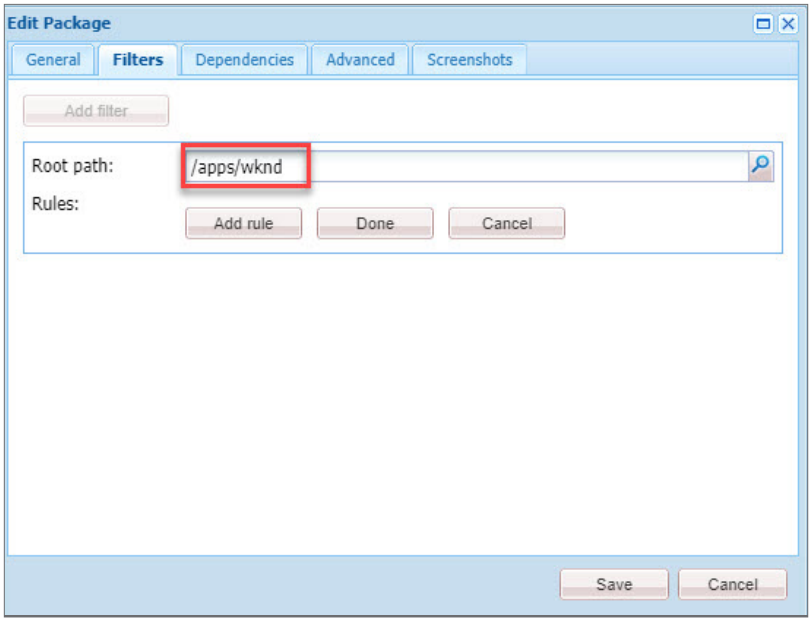
Note: Filters are the mechanism used to add content to packages. Here, you specify the paths that contain the content from the JCR you want to include in a package. Before adding filters, your package is completely empty. You may also restrict file types added to a package using filter rules, such as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

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

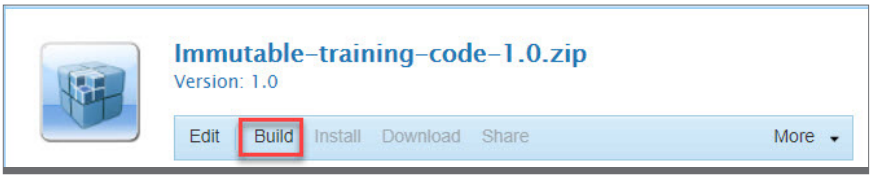
TIP: As you type in the Root path field, the field will auto-complete.

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 notice that you may use the group name to categorize and organize your packages in AEM.

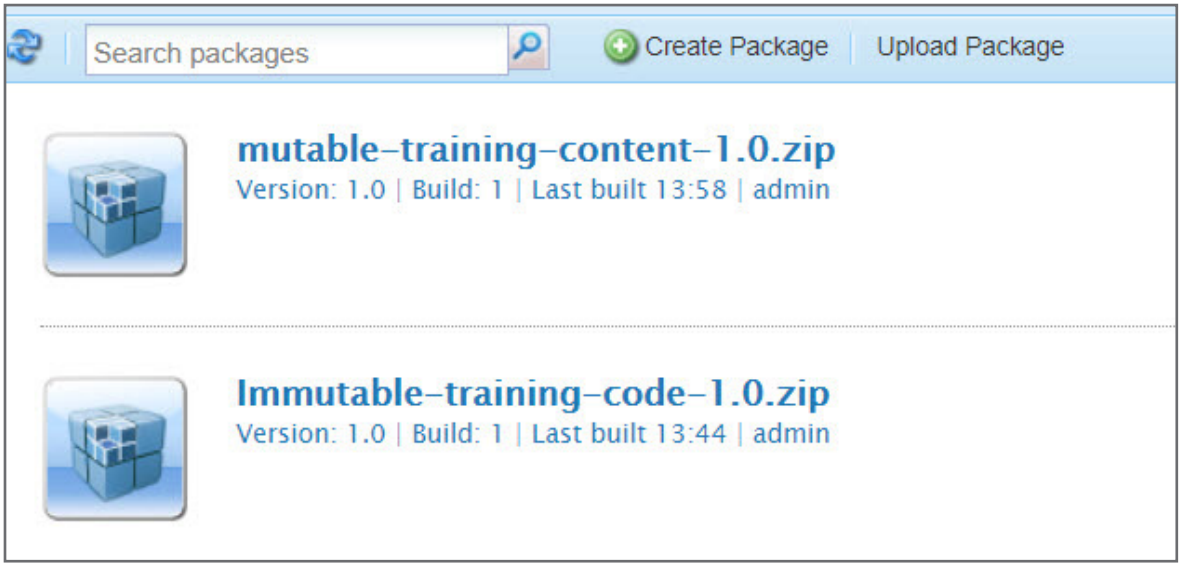
You have successfully created an immutable package.

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.

Objectives

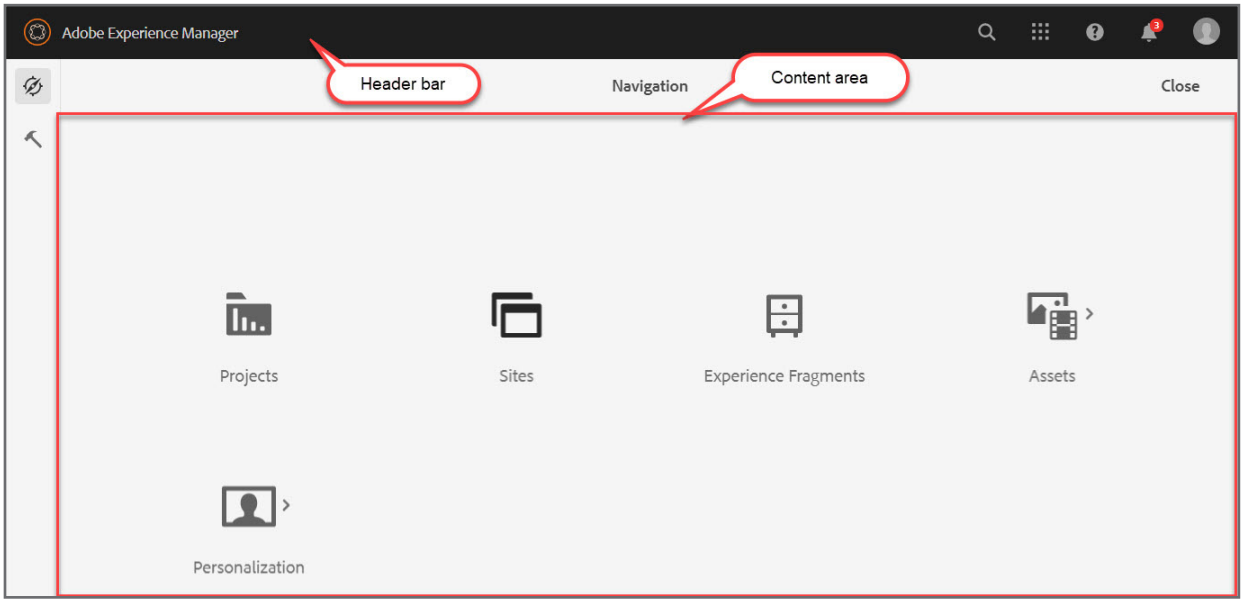
After completing this course, you will be able to:

- Navigate through the AEM UI
- Create and edit pages
- Explore core component authoring

AEM UI Features

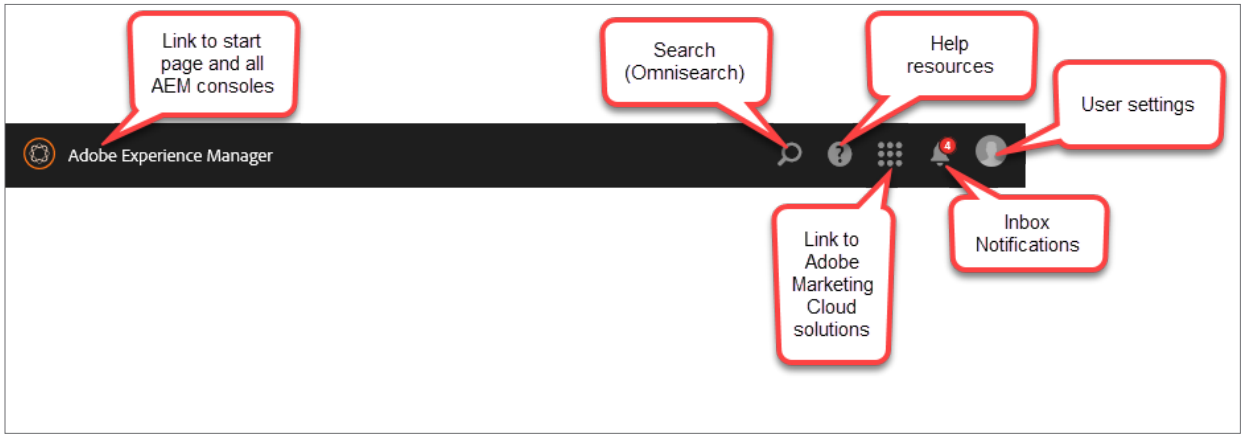
The AEM UI combines the advantages of a web interface with the fluidity and responsiveness that is usually associated with desktop applications. It is touch-optimized for authoring across desktop and mobile devices.

When you sign in to AEM, you are presented with a start screen that includes the header bar and the content area.



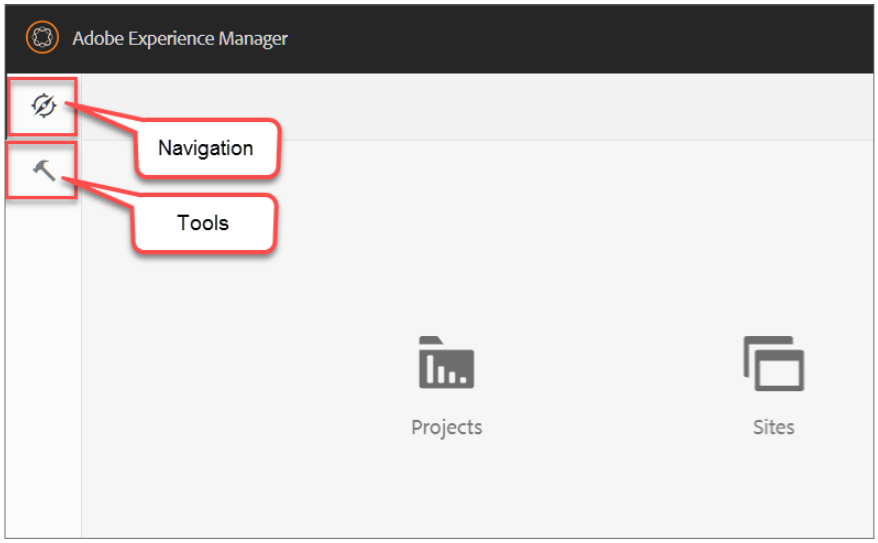
The content area, by default, displays all applications and capabilities of AEM, such as Projects, Sites, Experience Fragments, and Assets.

The header bar displays the default options, as shown below, and changes depending on the process or item you have selected:



Navigation and Tools

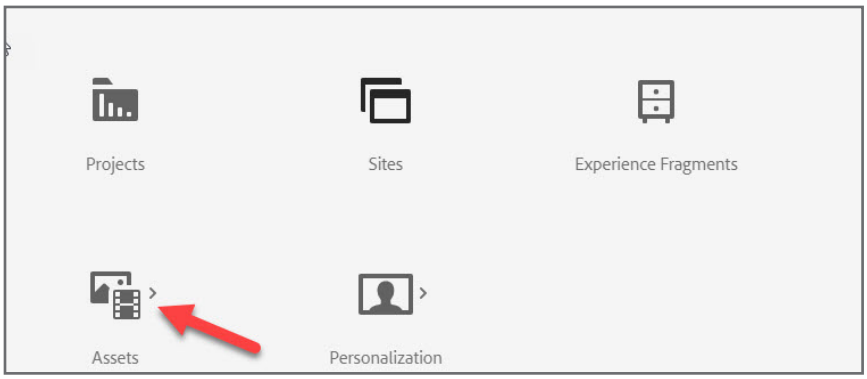
There are two main sections in the AEM UI: Navigation and Tools.



Navigation

The Navigation section is represented by a compass icon in the AEM UI. By default, when you first sign in, the Navigation section loads in the content area. All applications of AEM, such as Projects, Sites, Experience Fragments, Assets, and Forms, are available in this section. This is also the main section relevant to AEM authors to manage and build content for AEM Sites or leverage assets.

Note: The applications within the Navigation section are arranged like folders in a hierarchy. The applications with a carat symbol, as shown below in the following diagram, have "child" subfolders available, such as **Assets** > **Files**.



Tools

The Tools section is represented by a hammer icon in the AEM UI. This section displays all AEM administrative consoles, developer tools, and other technical consoles available.

AEM developers and administrators use this section to develop and administer websites, digital assets, and other aspects of the content repository.

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.

Creating Pages

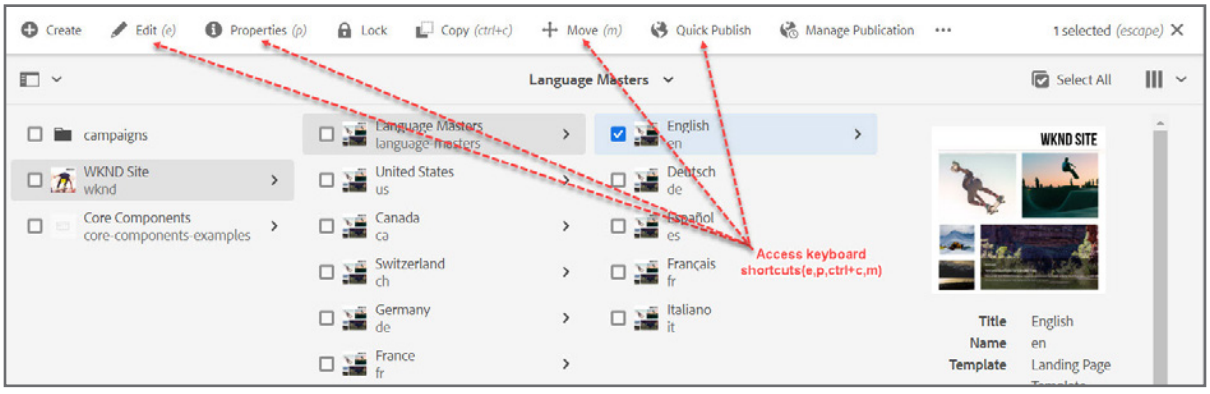
To create a page in AEM Sites, you need to follow a simple wizard where you specify the page template from a list of available templates, and then provide the page with a title and a name. The title is displayed to the user (who views the page in a browser) and the name is used to generate the page URL. The name can be derived from the title, if a name is not initially given when completing the required steps in the page creation wizard.

Editing Pages

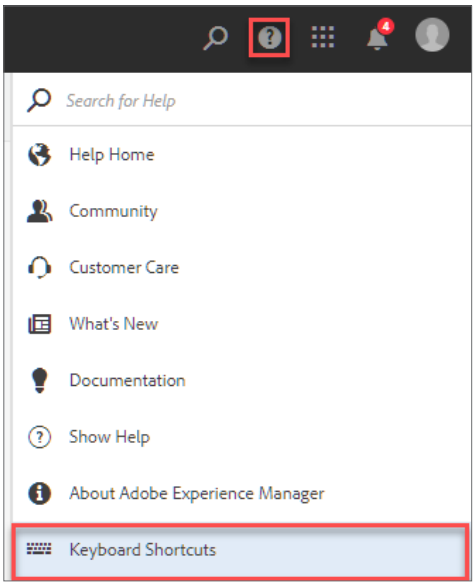
After you create a page, you can edit and add content to it. You can add content by dragging the components available in the side panel onto your page in the page editor.

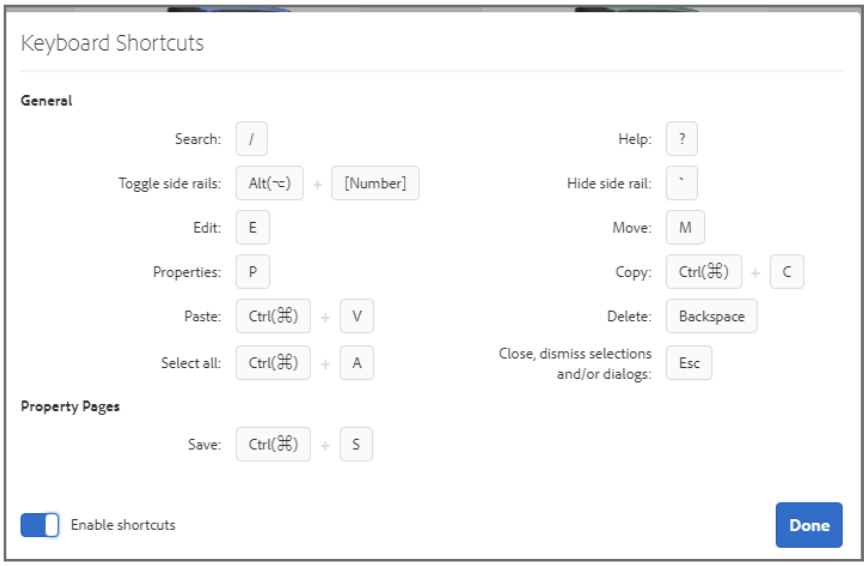
AEM Keyboard shortcuts

When you select an object in AEM such as a page, the header bar changes to show a menu of options to work with the object. To aid in editing and managing pages, there are a collection of keyboard shortcuts you can use to quickly access common tasks and functions, as shown below. By default, each user's keyboard shortcuts are enabled.



You can customize or enable/disable shortcuts by navigating to **Help** > **Keyboard Shortcuts**.





Note: For accessibility reasons, shortcuts may be disabled if they conflict with screen readers. You can also opt to disable the shortcuts yourself using the Enable shortcuts toggle switch in the **Keyboard Shortcuts** dialog box.

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

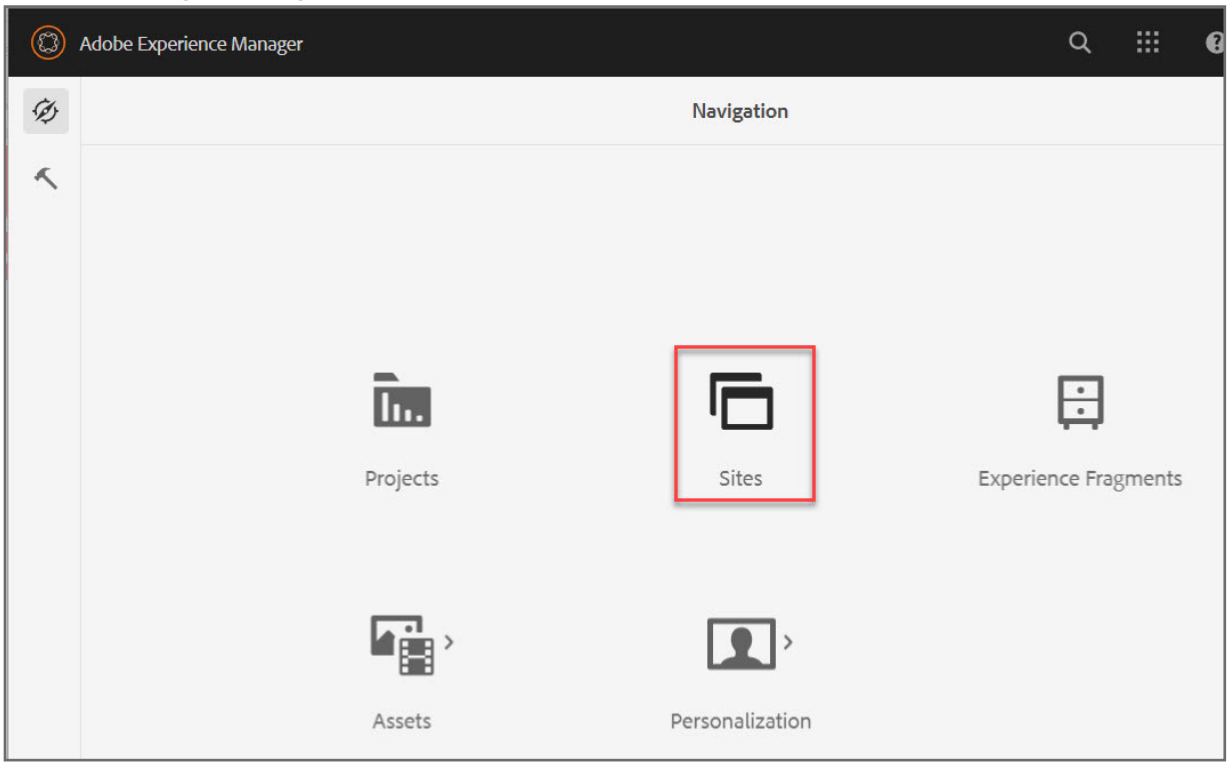
Note: While WKND illustrates an adventure and relaxation site, it is not specific to a leisure site. The site is set up in a way that it can be applied to any vertical.

Exercise 1: Create a page in AEM

Scenario: As an author, you need to create and edit a page in AEM Sites under the built-in WKND site hierarchy.
In this exercise, you will create a demo page using the content template available in AEM.

To create a new page:

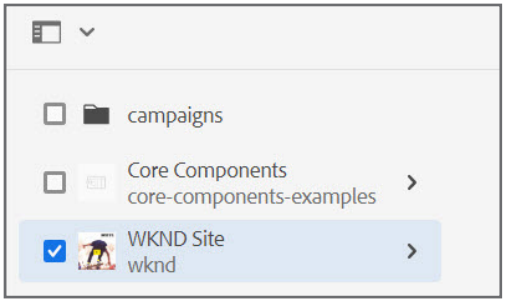
1. Ensure you have started and logged on to your AEM author service on port 4502.
2. The Navigation page is displayed by default in the content area. Click **Sites**, as shown.



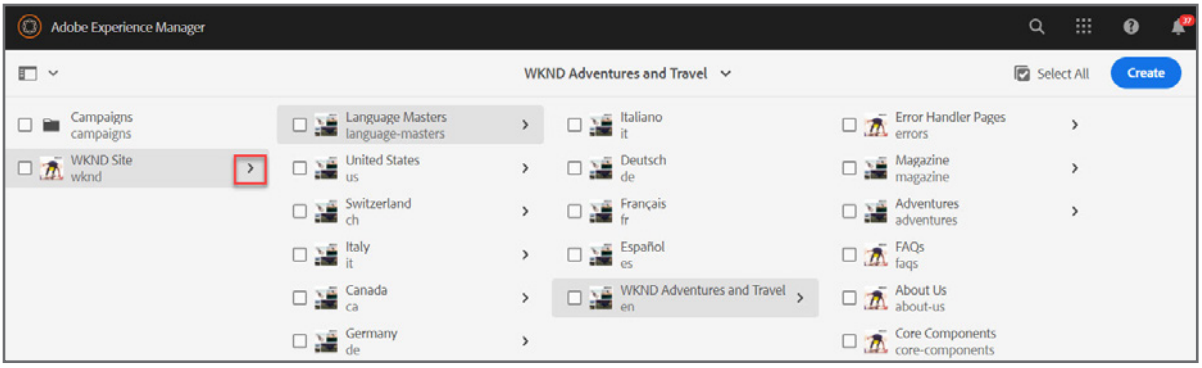
The column view is displayed.

Note: You may 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.

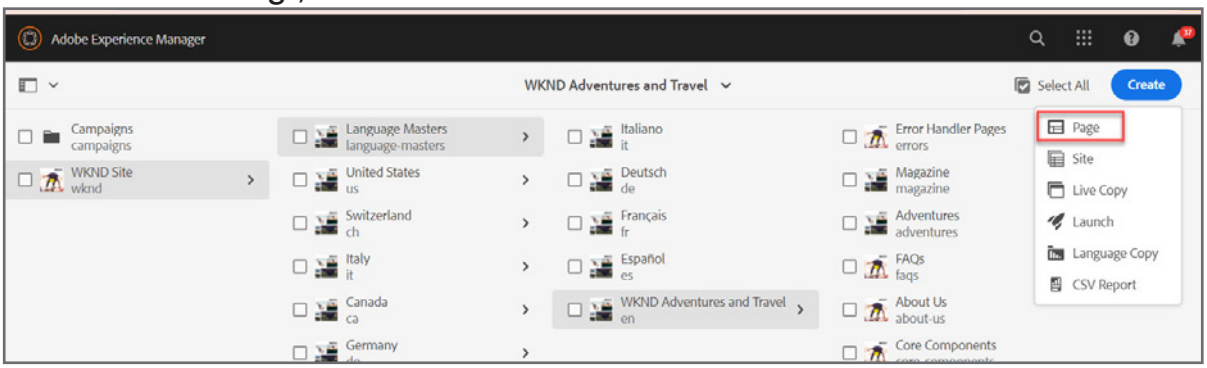
TIP: If you see a checkmark on the WKND site thumbnail as shown below, it means you have *selected* WKND site for editing and/or managing the page. Click the thumbnail again to clear the selection and click the right-pointing arrow instead.



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:

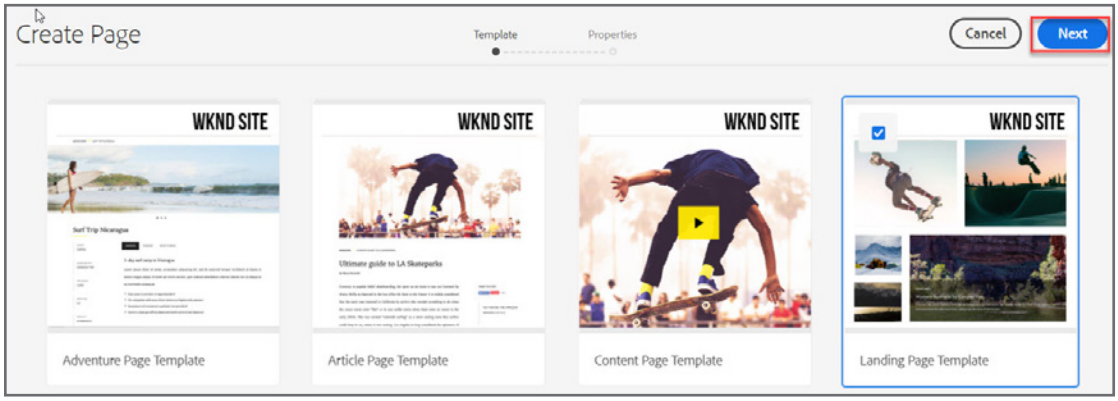


4. Click **Create** > **Page**, as shown.

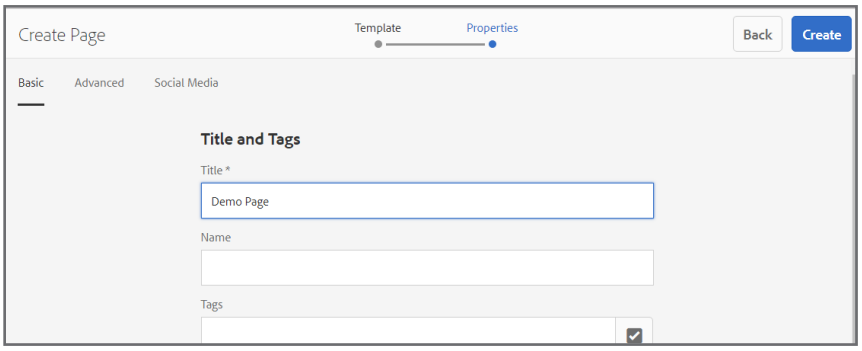


After you click **Page**, a wizard appears where you need to select a template for your page.

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

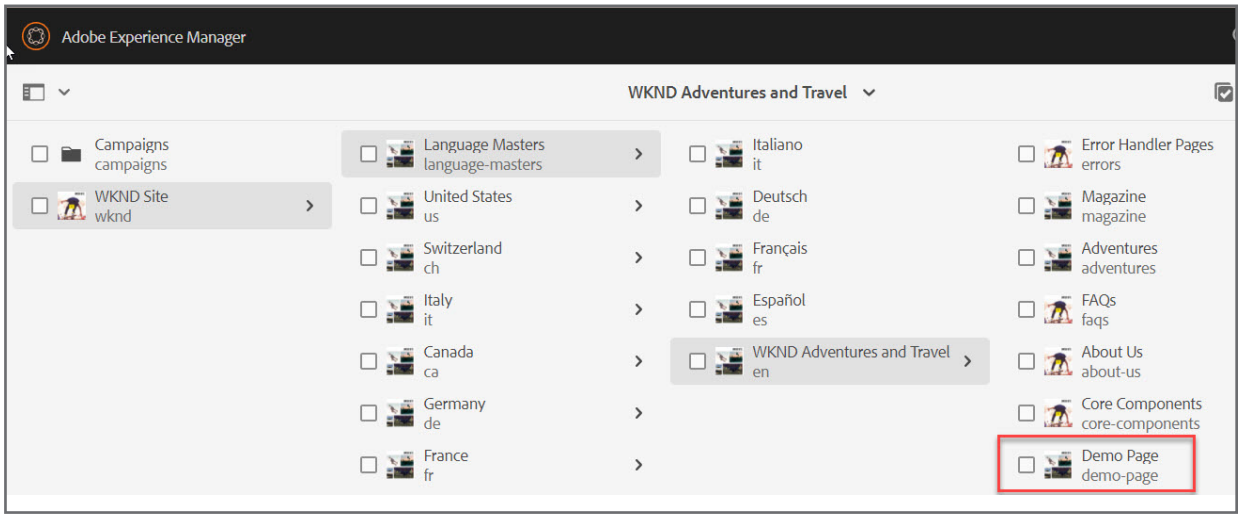


6. In the **Properties** step of the page creation wizard, provide the following value for the corresponding field:
   a. Title: **Demo Page**



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.

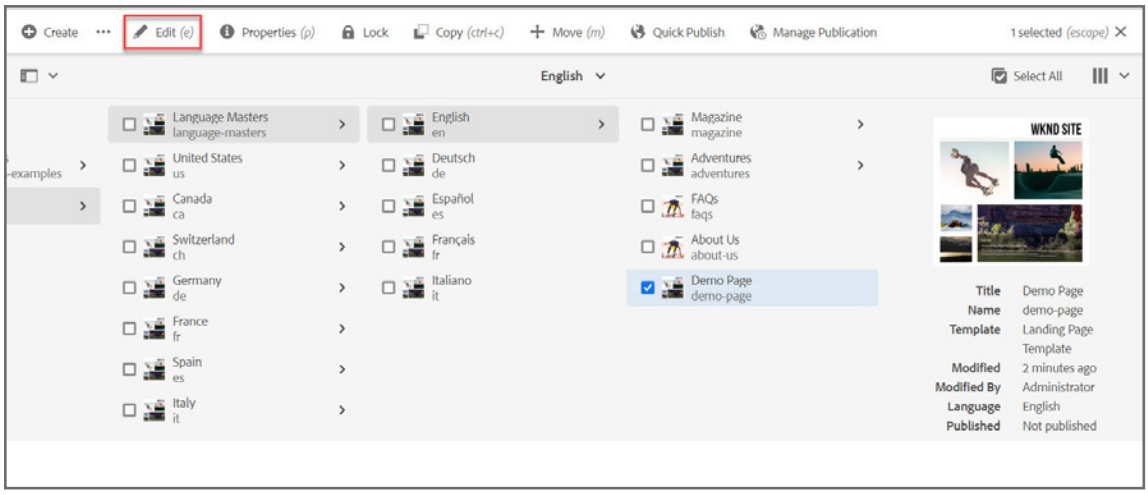


You can create subpages of any page using the same process.

Exercise 2: Edit a page in AEM

In this exercise, 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:

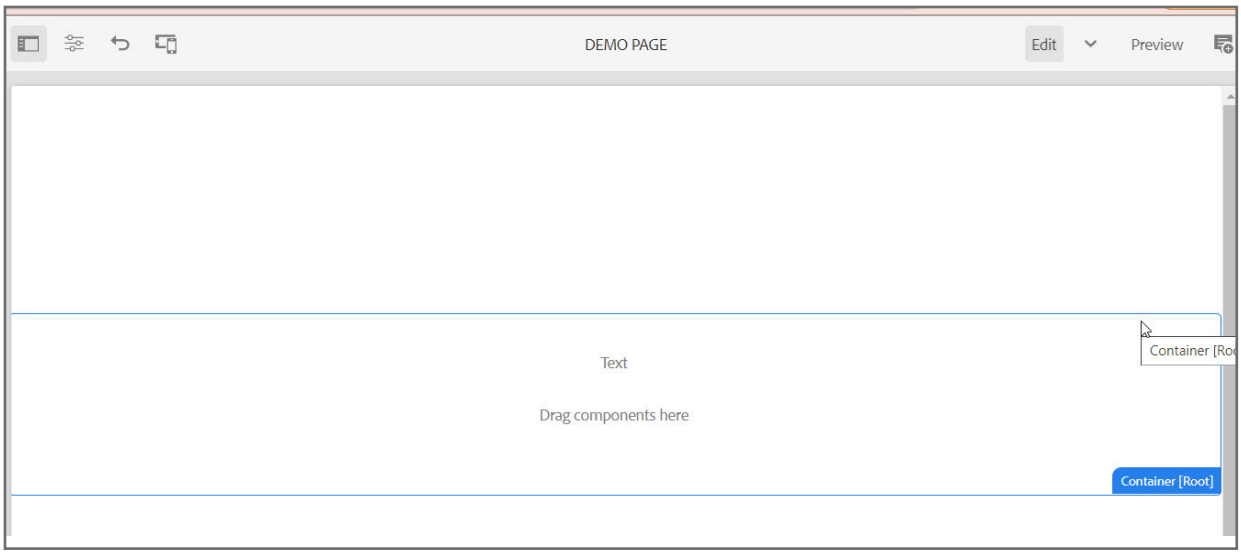
To add a text component to the page:

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

The side panel is displayed.

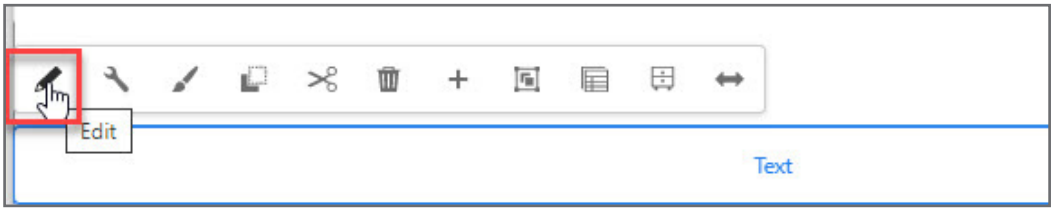
3. Click the **Components** icon.
4. In the **Filter** field, type **text** and press **Enter** to search for the **Text** component. The search yields results that contain the word "text".
5. Drag the **Text (WKND.Content)** component onto the **Drag components here** box, as shown:

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.

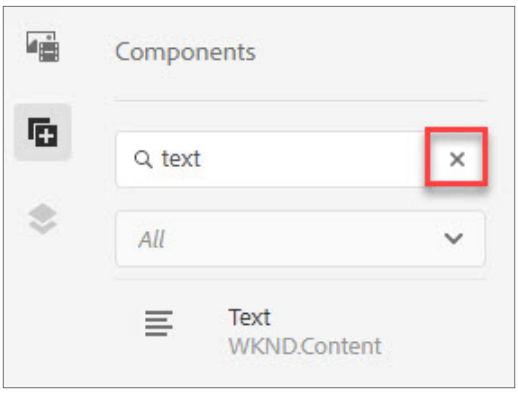


7. Add sample text of your choice in the text editing form that appears and click **Done** (**checkmark** icon) to save the changes. The text is added to the page.

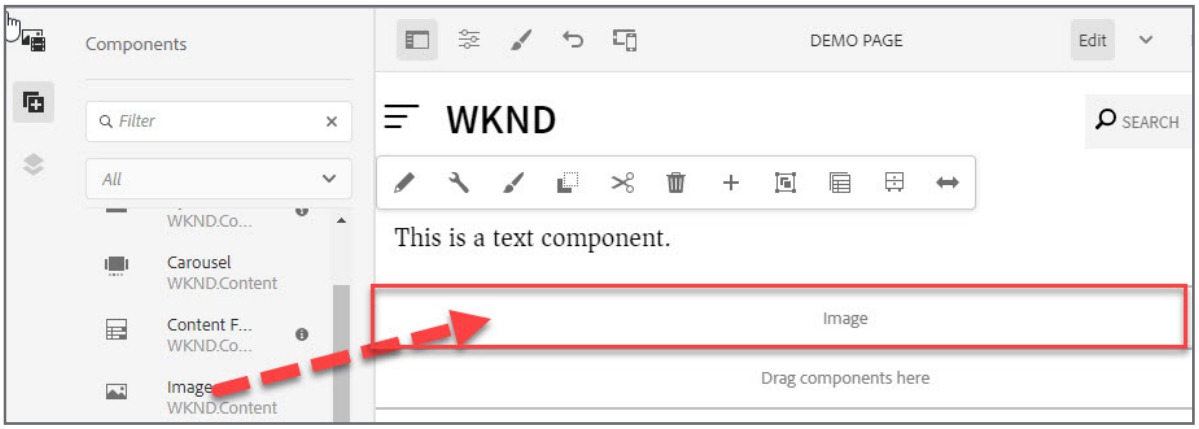
You have now added the first component to your page!

To add the image component to your page:

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

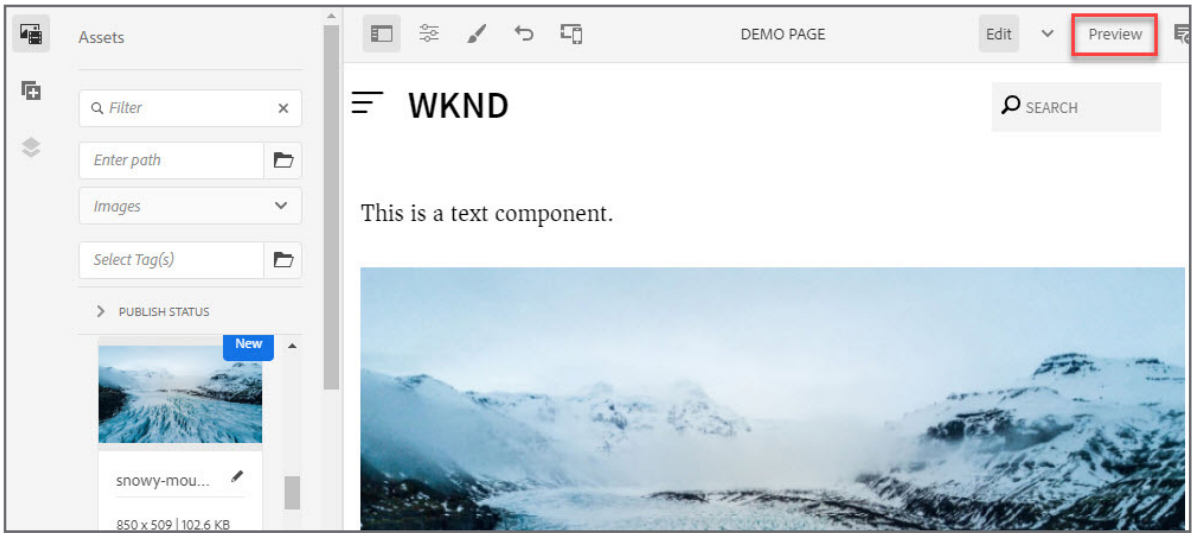


9. Scroll down and drag the **Image (WKND.Content)** component onto the **Drag components here** box as shown:



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

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

You can find that the code on Core component development exists outside of the AEM code base. This enables you to update and enhance the components more often than the product releases. Developers can extend core components to offer custom functionality.

Core components provide the rich-authoring functionality and several advantages to authors when adding content to pages. Components are available on the **Components** tab of the side panel of the page editor when editing a page.

Components are grouped according to categories called component groups to easily organize and filter the components. The component group name is displayed along with the component in the component browser, and it is also possible to filter components by group to easily find the right component.

Pre configuring Core Components

With Core components, a template author can configure a number of capabilities through the Template Editor or in the design mode. For example, if an image component should not allow image upload from the file system or if a text component should only allow certain paragraph formatting features can be enabled or disabled with a simple click.

As the Core components can be pre-configured by template authors to define what options are allowed as part of a template, and further configured by the page author to define actual page content, each component can have options in two different dialogs:

- Edit dialog: Provides the options that a page author can modify during page editing for the placed components. For example, formatting of content text and rotating an image on a page.
- Design dialog: Provides the options that a template author can modify when configuring a page template. For example, text formatting and image in-place editing options.

The styles of most Core components can be defined by using the AEM style system.

- A template author can define which styles are available for a particular component in the design dialog of that component.
- The content author can then choose which styles to apply when adding the component and creating content.

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.

Exercise 3: Explore core component authoring (Optional)

Core components provide the toolset to quickly start your projects with minimal component development. In this exercise, 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 exercise to come back to after completing this class to fully understand the capabilities of some of the core components.

This exercise includes the following tasks:

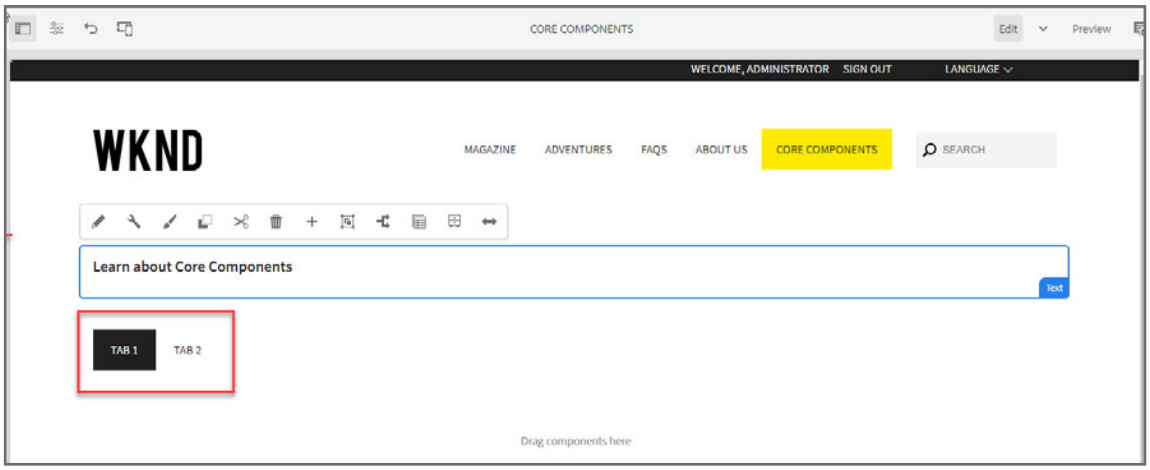
1. Create a Page
2. Explore different components

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.

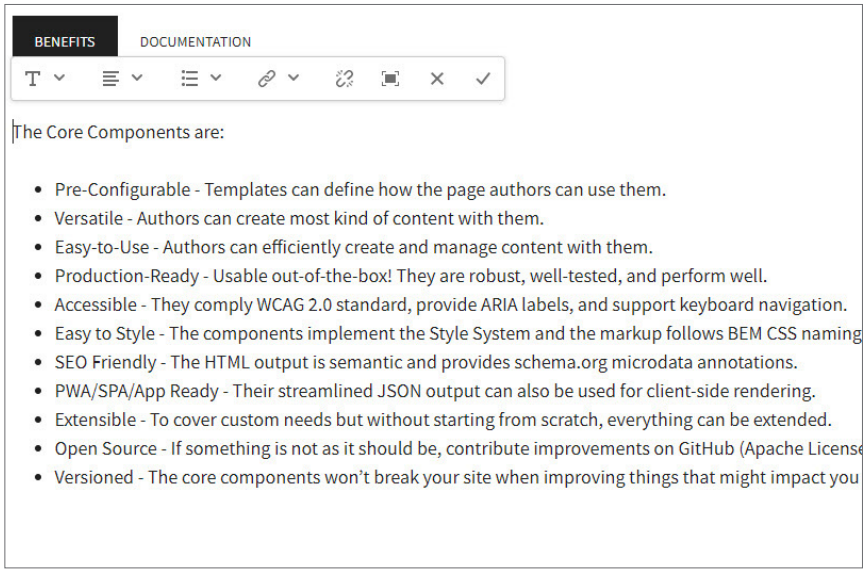
Task 2: Explore different components

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:



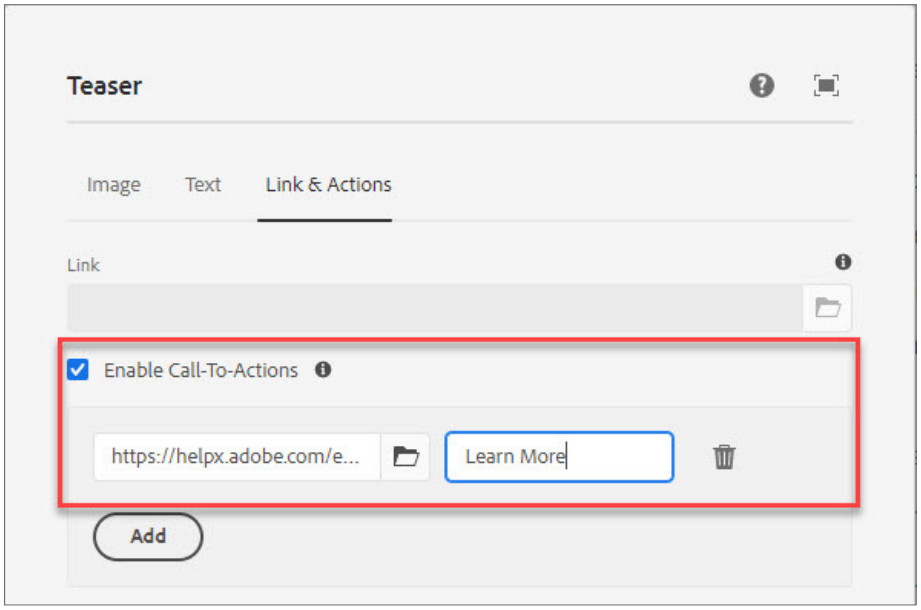
7. Double-click the tabs component. The **Tabs** dialog box opens.
8. On the default **Items** tab, rename **Tab 1** as **Benefits**.
9. Rename **Tab 2** as **Documentation**.
10. Click **Done** to save the changes.
11. On the **Benefits** tab, click the **Drag components here** area.
12. Click the **Insert component** icon (the + icon) and add a **Text** component.

13. Navigate to the **Sites Authoring Basics** exercise files provided to you and copy the content from \Exercise_Files_TB\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:

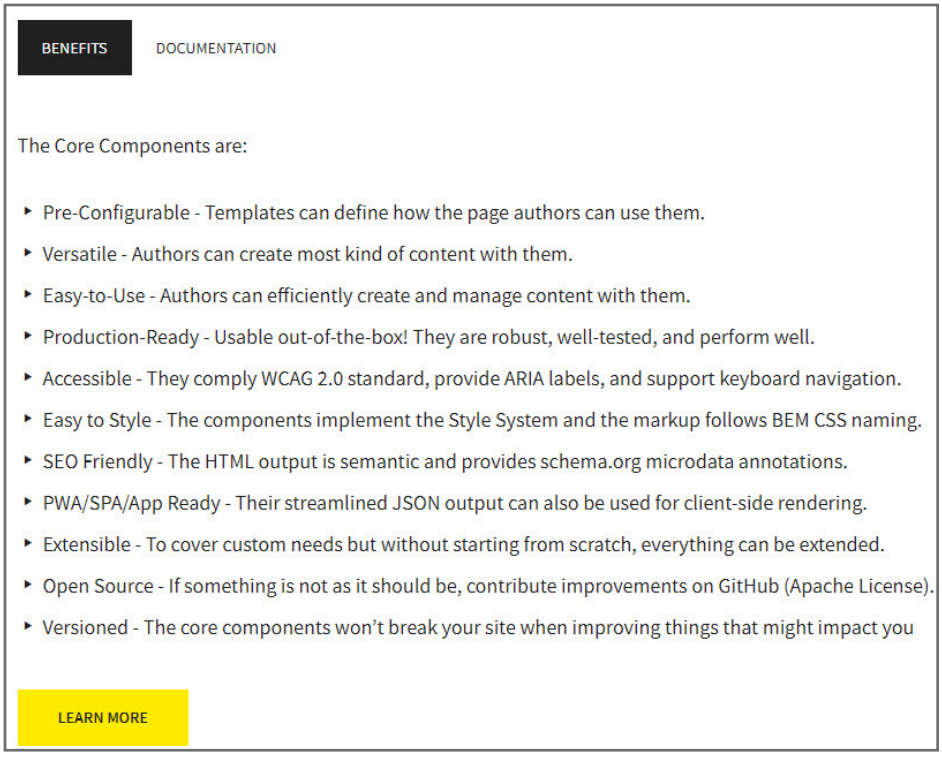


16. Click **Done** to save the changes. The text is added to the **Benefits** tab.
17. Click the **Drag components here** area below the text you added in the previous step.
18. Click the **Insert component** icon (the + icon), and add a **Teaser** component.
19. Click the **Teaser** component, and click 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.

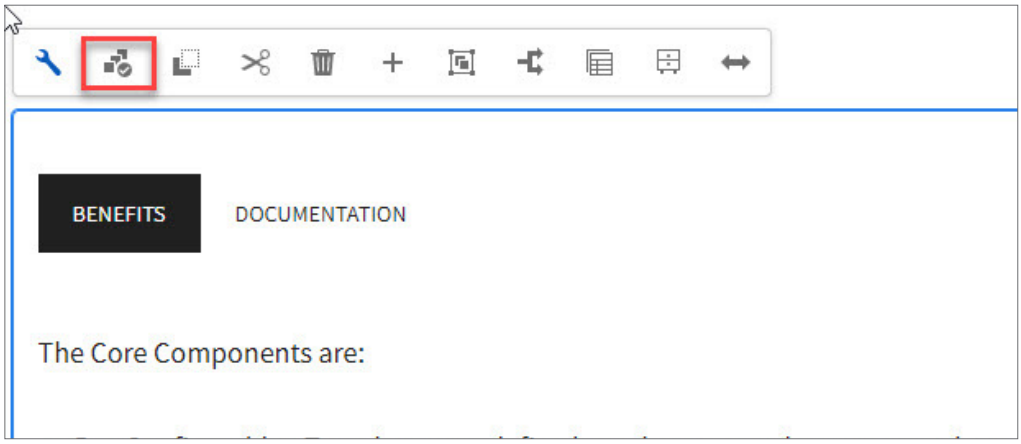
22. In the **Text** field, type **Learn More**. The **Teaser** dialog should look, as shown:



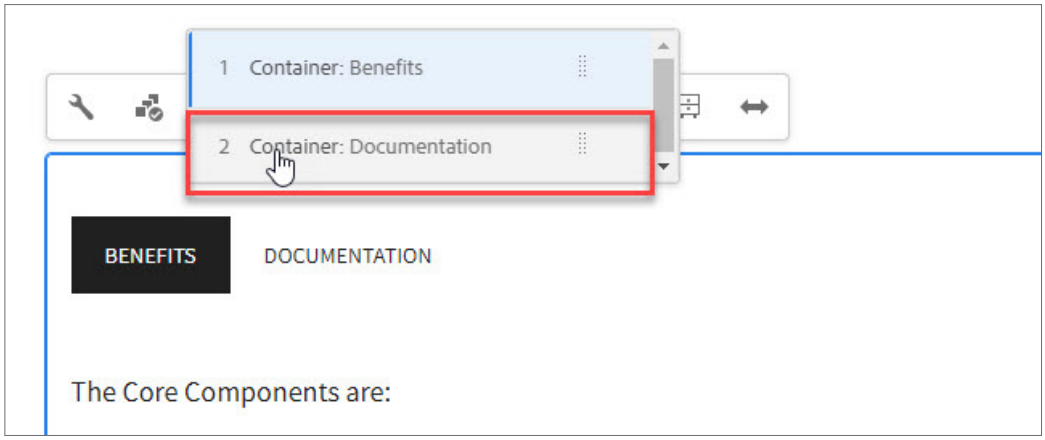
23. Click **Done** (checkmark icon) to save the changes. The **Learn More** button is added to the page. The Benefits tab should look, as shown:



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



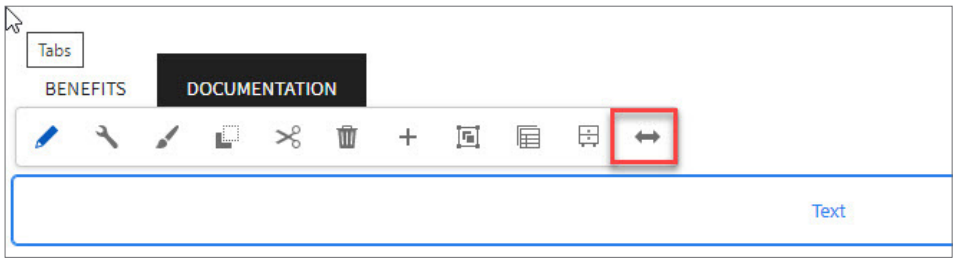
25. Click **Container: Documentation**, as shown. The components and content you insert now will be added under Documentation.



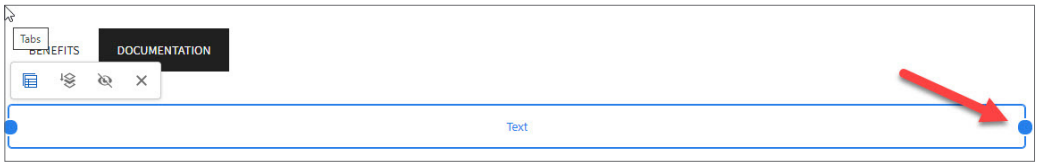
26. On the **Documentation** tab, click the **Drag components here** area.

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

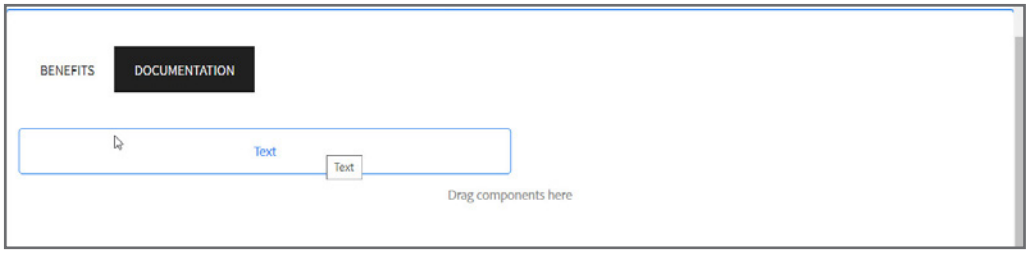
28. Click the **Text** component, and click **Layout**, as shown. The layout is now editable.



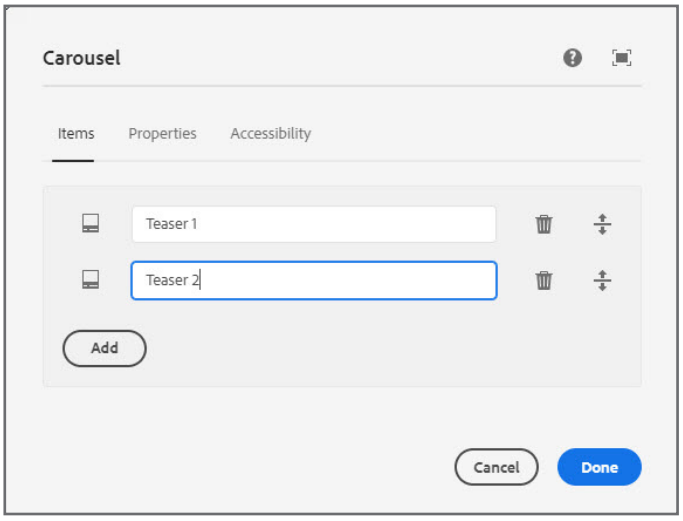
29. Drag the blue circle and adjust the layout to 6/12 columns, as shown:



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



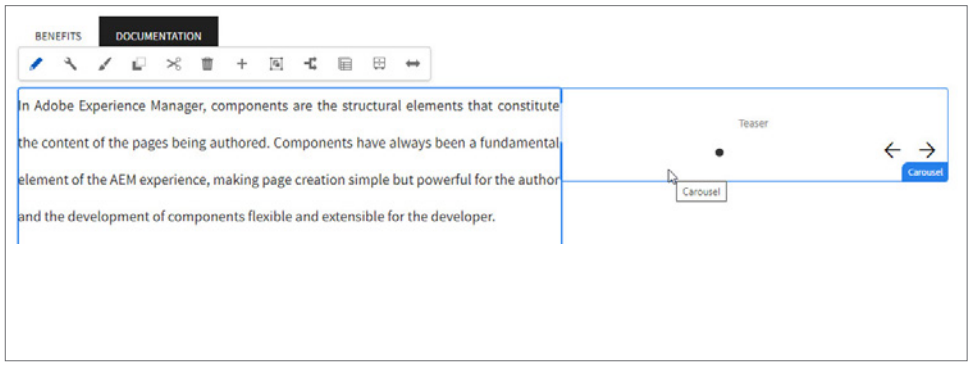
31. Navigate to the **Sites Authoring Basics** exercise files provided to you and copy the content from \Exercise_Files_TB\Sites_Authoring_Basics**Documentation.txt**.
32. Click the **Text** component in AEM, and click the **Edit icon** (pencil icon) to add content.
33. Paste the content you copied in 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**.

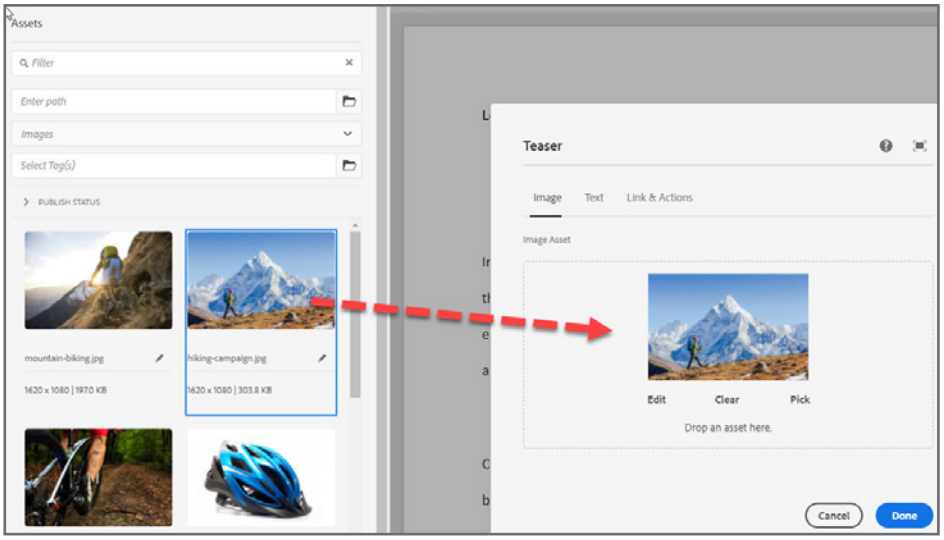
39. Click the **Carousel** component, and click **Layout**. The layout is now editable.

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.

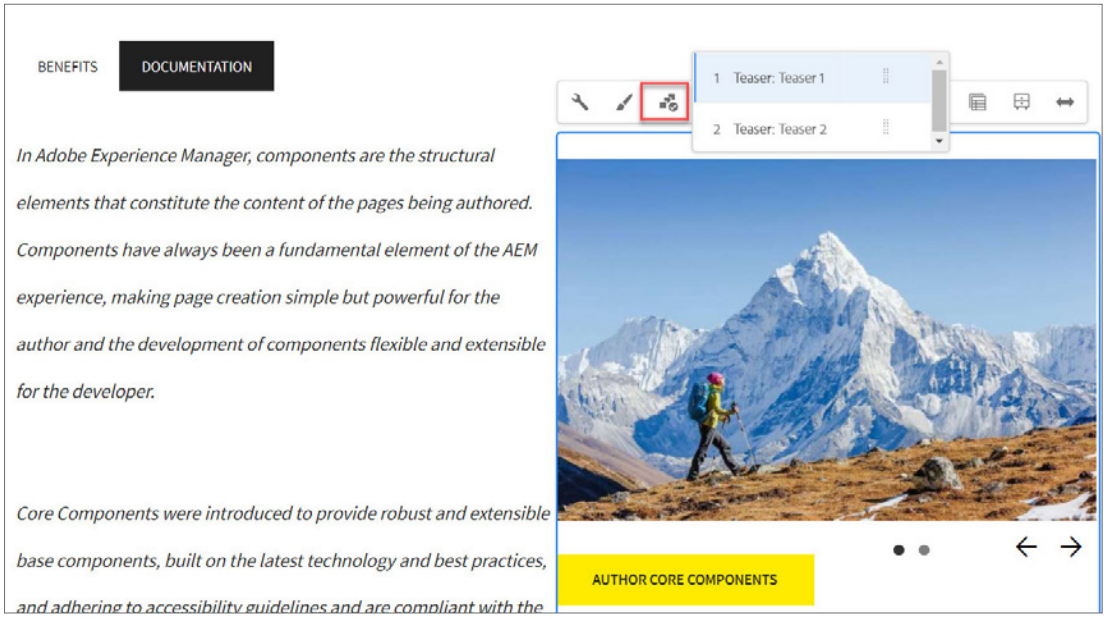
44. On the **Links & Actions** tab, select the **Enable Call-to-Actions** checkbox.

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.

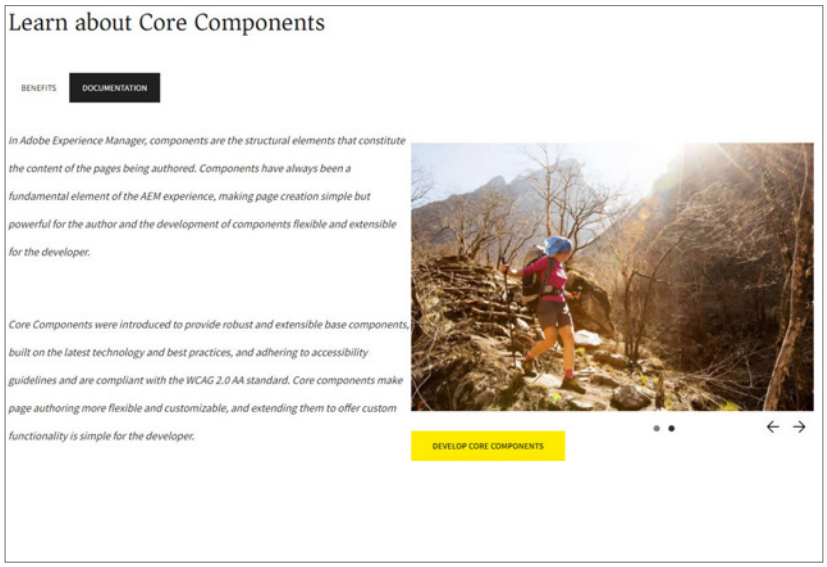
54. On the **Links & Actions** tab, select the **Enable Call-to-Actions** checkbox.

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 exercise by uploading and installing the **core-component-authoring-in-xf.zip** content package from the exercise 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

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.

Objectives

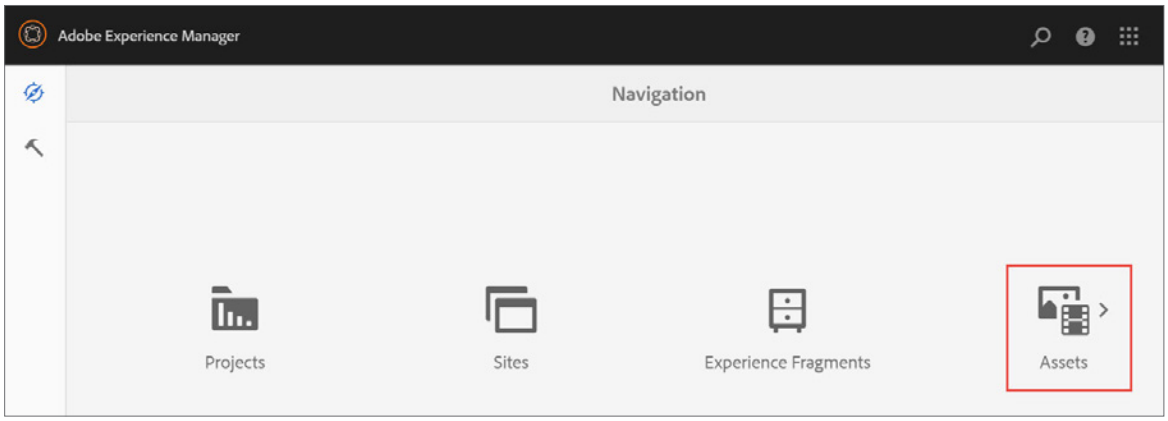
After completing this course, you will be able to:

- Navigate through the Assets console
- Create folders
- Upload assets to a folder
- Explain the basics of asset metadata
- Add metadata to an asset

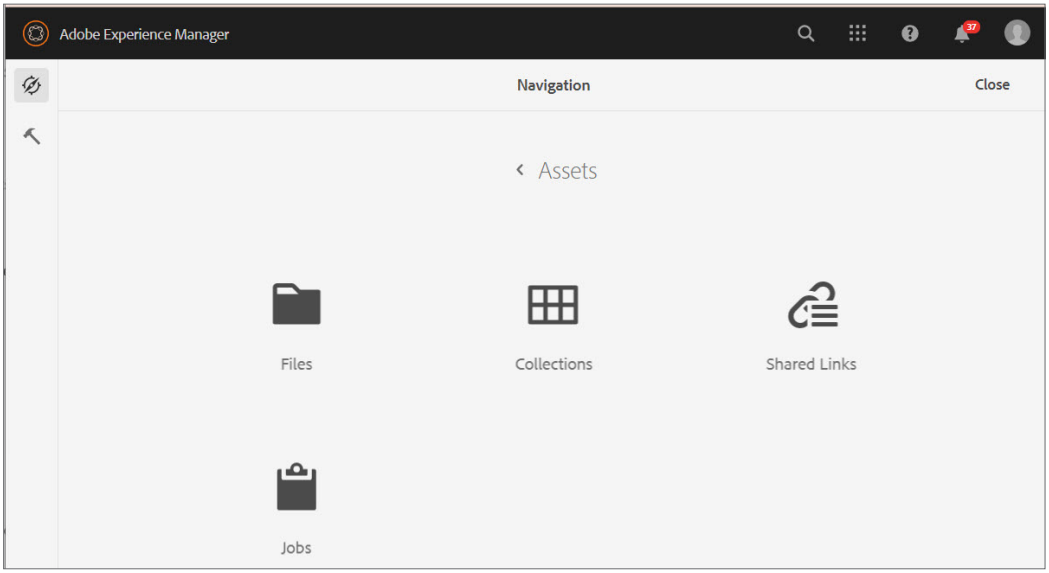
Assets Console

From the Assets console, you can create a collection of assets and catalogs, and then use them on any websites running on the AEM author service.

You can access the Assets console from Navigation as shown:



The Assets console contains different subconsoles for Files, Collections, Shared Links, and Jobs, as shown:

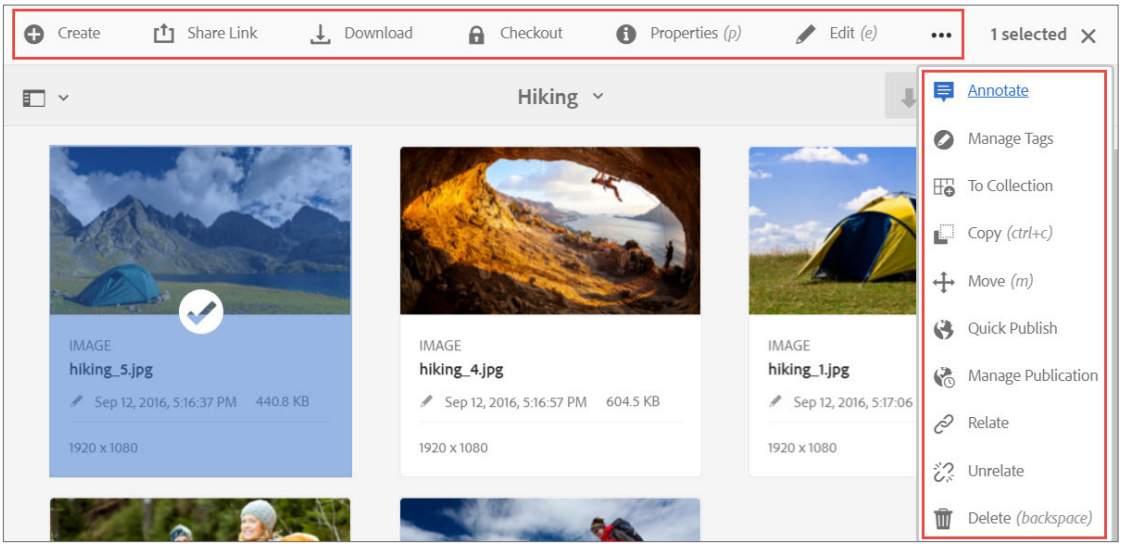


The following table describes the subconsoles of the Assets console and their purposes:

Subconsoles	Purpose
Files	Contains different asset folders and helps organize the assets as per websites
Collections	Contains different types of collections (set of assets) such as Lightbox and Smart collections
Shared Links	Provides the details, such as users and expiry date, of a shared asset
Jobs	Helps monitor the status of asynchronous operations on assets

Actions on Assets

In the selection mode (after you select an asset), the actions bar provides options to perform various actions on assets.



The following table describes the actions available for assets:

Actions	Description
Create	Helps initiate a workflow on the asset and create an asset version
Share Link	Helps share assets, such as a URL, with the members of your organization and external entities, including partners and vendors
Download	Helps download an asset to your local computer
Checkout	Locks the asset to avoid overriding other users' work
Properties	Helps view and edit the asset properties
Edit	Resizes (crop, rotate, and flip) an asset
Annotate	Adds comments to an asset
Manage Tags	Adds tags and smart tags to an asset
To Collection	Adds an asset to a collection
Copy	Creates copies of an asset in multiple folders (keeping the asset --- source of the copied version --- in the original folder while having a copy of the asset in another folder)
Move	Moves an asset from one folder to another (removing the selected asset from the original folder to another folder)
Quick Publish	Publishes an asset that is ready to be delivered to the web application immediately
Manage Publication	Offers more publish options, such as inclusion of child pages, customization of the references, starting any applicable workflows, and publishing assets at a later date
Relate	Links assets that are similar, have some relationship, or are derived from another asset
Unrelate	Unlinks or removes the relation from the asset
Delete	Deletes an asset from a folder and also from the repository

Organizing Assets

AEM Assets supports multiple methods of organizing assets. In the Assets console, you can organize assets in hierarchical or ad hoc ways.

Hierarchical

In the hierarchical method, folders are used to impose a consistent storage structure for digital assets. In the hierarchical method, you can:

- Create public or private folders
- Organize and group asset folders related to a specific task
- Create asset collections, so all related assets are available in one directory
- Assign different tasks to users that they can perform on asset folders based on privilege levels

Ad hoc

In the ad hoc method, you can edit asset properties and add tags to assets. You can search for assets by using the tags and then save the search results as a smart collection.

Creating Asset Folders

In the Assets console, you can create folders to organize your assets. To create a folder in the Assets console, click the **Create** button and select **Folder** from the drop-down menu.

The key fields used when creating an asset (private) folder are:

- Title: This displays to the user in the console and is at the top of the content page when editing. This field is mandatory.
- Name: This is used to generate the Uniform Resource Identifier (URI). The user input for this field is optional. If not specified, the name is derived from the title.

Asset folders have two categories:

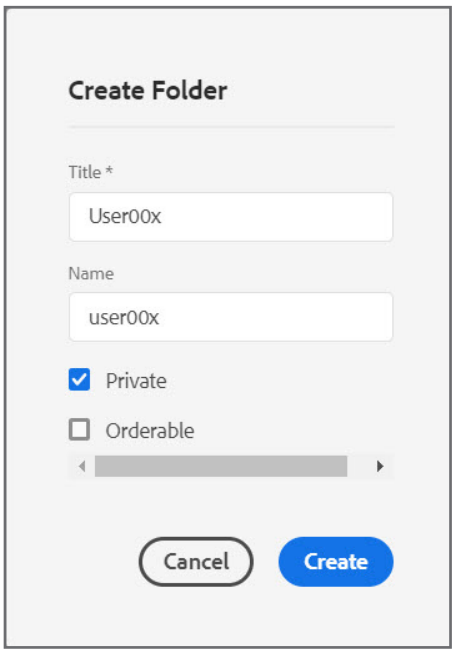
- Private
- Public

Private Folder

A private folder is accessible to only the owner who creates it. You can also share your private folder with other users and assign different privileges to them. Based on the privilege level, users can perform various tasks on the folder, such as viewing assets within the folder or editing them.

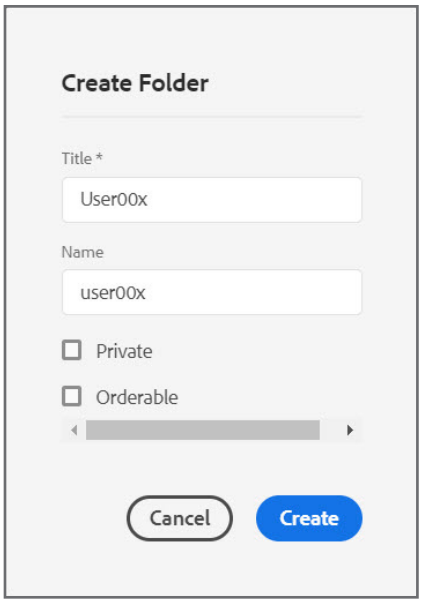
Note: Administrators can access the private folders created by all AEM users.

To make a folder private, you must select the **Private** option from the **Create Folder** dialog box, as shown:



Public Folder

All users in the organization can access a public folder and perform common tasks on its assets. To make a folder public, you must clear the **Private** option from the **Create Folder** dialog box. By default, the **Private** option is not selected.



Note: The Orderable option helps reorder/reorganize the folders within the Assets console.

Supported Assets Files and Formats

AEM Assets supports different files and formats of assets. Before uploading an asset, you must verify the file and format is supported by AEM.

The following table lists the files and formats supported by AEM Assets:

Files	Supported Formats
Images	PNG, GIF, TIFF, JPEG, BMP, PNM, PGM, PBM, PPM, .PSD*, .EPS, DNG, RAW, PICT, and PSB
Documents	AI, DOC, DOCX, ODT, PDF, HTML, RTF, TXT, XLS, XLSX, ODS, PPT, PPTX, ODP, INDD, PS, QXP, and EPUB
Multimedia	AAC, MIDI, 3GP, MP3, M4A, MPG, OGA, OGG, RA, WAV, WMA, DVI, FLV, M4V, MPEG, OGV, MOV, WMV, and SWF
Videos	MP4, MOV.QT, FLV, F4V, WMV, ASF, MPG, VOB, M2V, MP2, M4V, AVI, WEBM, OGV, OGG, MXF, MTS, MKV, R3D RM, RAM RM, FLAC, and MJ2
Archive	TGZ, JAR, RAR, TAR, and ZIP
Others	SVG

Uploading Assets

You can upload different types of assets, even large files (more than 50 Mb), from your local folder or a network drive to AEM Assets. You can pause the upload and resume again, if required. If you cancel the upload operation, AEM will delete the partially uploaded file(s). You can upload assets to a folder from the Assets console or drag the asset from your local computer into a folder.

There are a few different ways to upload assets to a folder:

- Using the Assets console
- Dragging assets
- Using the AEM Desktop App

Uploading Assets from the Assets Console

From the Assets console, you can upload assets by clicking **Create** > **Files** and then selecting the assets you want to upload from your computer.

Dragging Assets

You can upload multiple assets simultaneously by dragging the selected assets from your local folder into the destination folder.

Note: In AEM Assets, serial upload of assets is enabled by default.

Uploading Assets by Using the Desktop App

You can upload individual assets and asset folders to AEM Assets by using the AEM Desktop app.

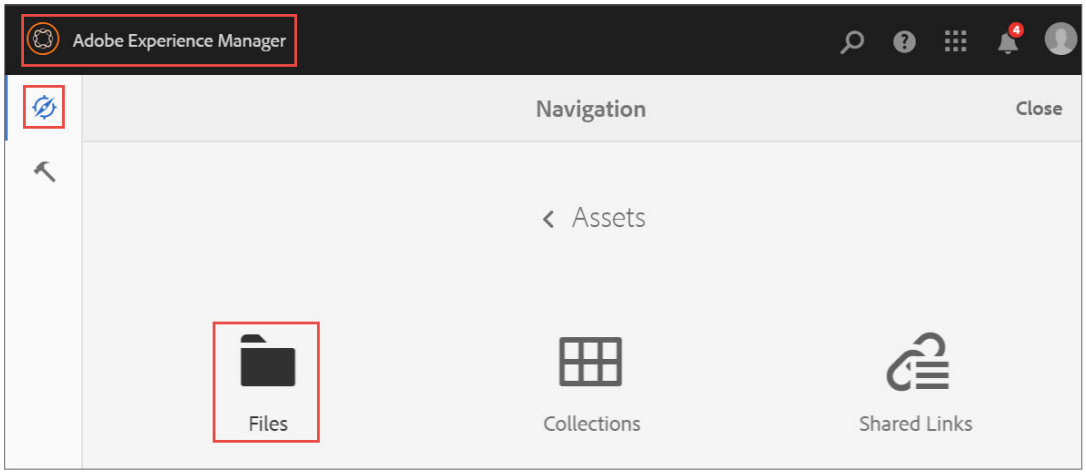
Note: To upload asset folders from your local folder system through the AEM Desktop app, you must enable and configure the Desktop app with AEM Assets.

Exercise 1: Create a folder and upload assets to it

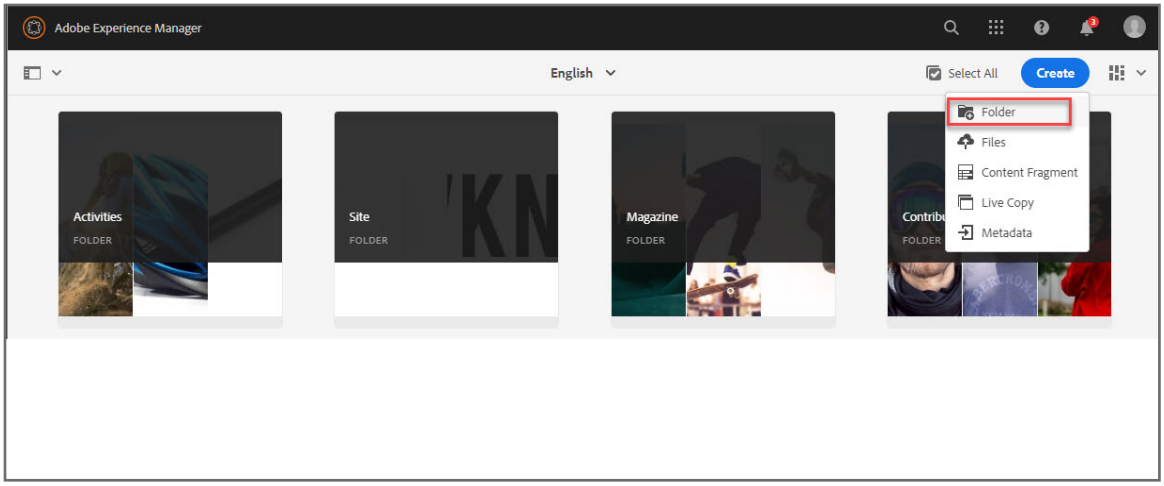
For performing this exercise, you will use the WKND site asset folder that comes with AEM.

To create a folder and upload assets to it:

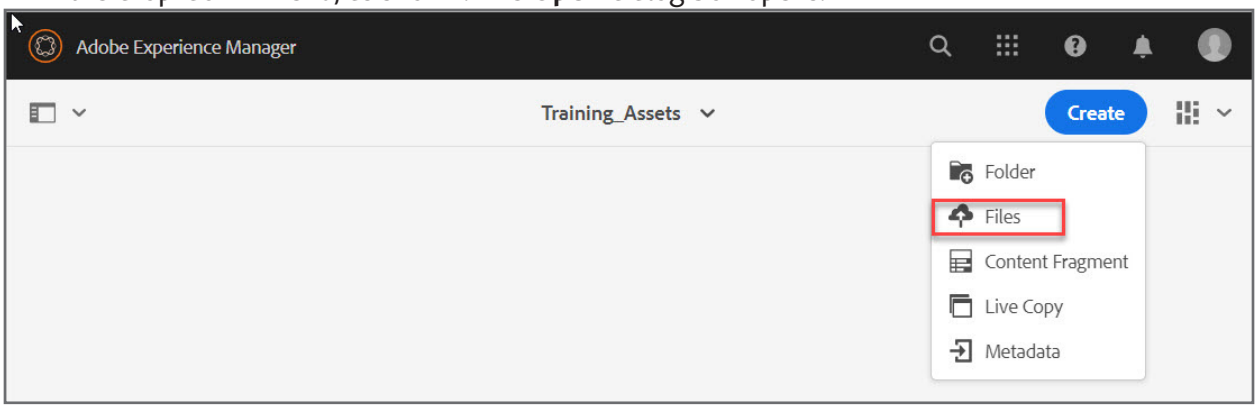
1. Go back to the tab in your browser that has the AEM author service open.
2. Click the **Adobe Experience Manager** icon.
3. Click **Navigation** > **Assets**, and then click **Files**, as shown. If a **Product Navigation** dialog box appears, click **Don't show this again**, and then click **Close**,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



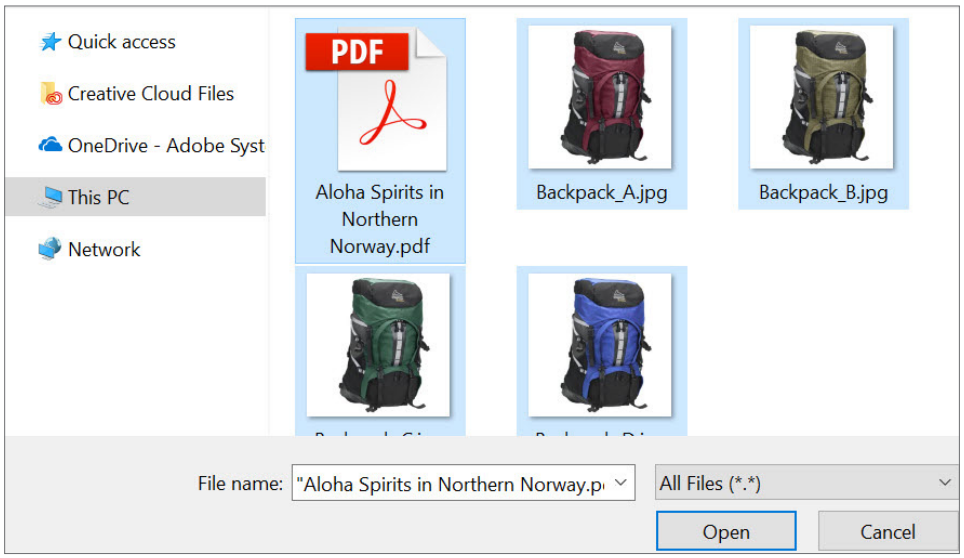
4. Navigate to **WKND Site** > **English**, and then click **Create** from the actions bar, and select **Folder** from the drop-down menu as shown. The **Create Folder** dialog box opens.



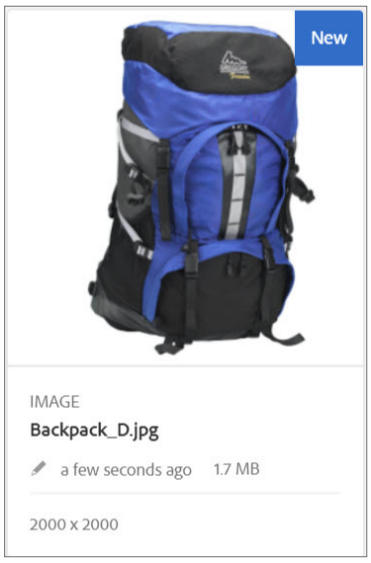
5. Type **Training_Assets** as the title of the folder. The name of the folder is automatically populated from the title.
6. Select **Private** to ensure only you have access to this folder but do not select the **Orderable** checkbox.
7. Click **Create**, as shown. Notice how the **Training_Assets** 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 drop-down menu, as shown. The **Open** dialog box opens.



9. From your local file system, navigate to the **Exercise_Files_TB/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 image(s) indicates the upload progress. Once uploaded, a blue "New" tab will appear in the upper-right corner of each image as shown.



Metadata: An Overview

AEM Assets store metadata of every asset. Metadata is data about data. For example, metadata here can be a collection of all data, such as the image name and size, and time and date of modification.

Metadata helps categorize, organize, and search assets. You can add more high-level data to digital assets, such as the:

- Type (Is it an image, video, audio clip, or document?)
- Owner
- Title
- Description
- Tags assigned to an asset

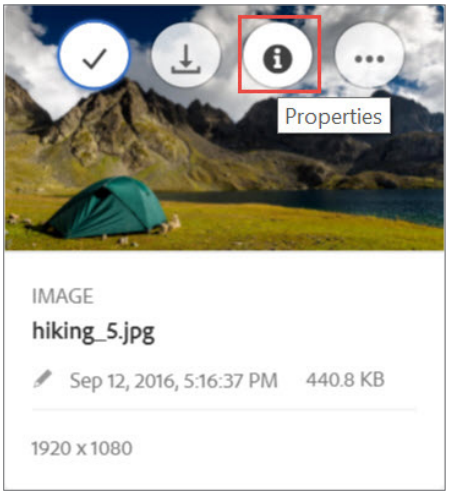
A good amount of metadata helps you further categorize assets. As metadata is added to assets, the asset becomes more accessible and easier to manage. For example, you can find assets with the same properties and apply changes to them easily.

Default Metadata

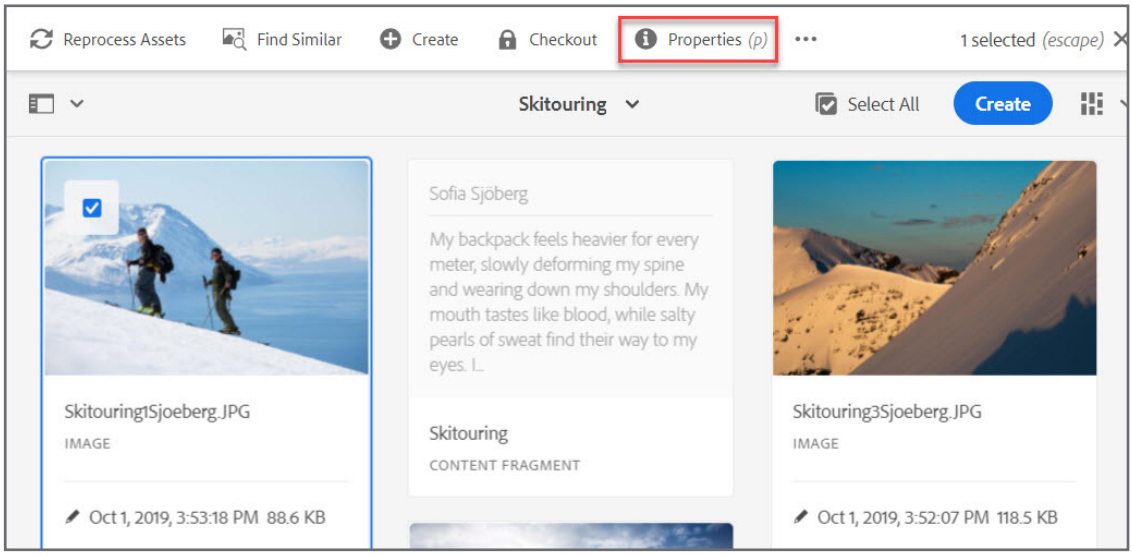
After you upload assets to AEM Assets, you can view and edit an asset's metadata from the Properties screen.

You can view the properties of an asset through quick actions and the actions bar.

- Quick actions: Hover the cursor over an asset card and click **Properties** from the quick actions, as shown. The properties wizard of the selected asset opens.



- Actions bar: Select an asset and click **Properties** from the actions bar, or press **p** from the keyboard, as shown. The properties screen of the selected asset opens.



All asset properties are categorized into the following groups:

- Basic
- Advanced
- IPTC (International Press Telecommunications Council)
- IPTC Extension
- Camera Data
- Product Data
- Insights

The following table describes the tabs of the asset properties screen:

Tab	Description
Basic	Provides the asset's data, such as title, description, tags, type, scheduled activation date, and links to the resources using the asset.
Advanced	Provides details about the creator, contributor, copyright, expiry date, and the rating assigned to an asset.
IPTC and IPTC Extension	Defines the people, locations, and products shown in an image. It also provides dates, names, and identifiers regarding the creation of the image, and a flexible way to express the rights information.
Camera Data	Provides information about the camera and the shot details.
Product Data	Specifies the product data details if the asset is associated with any product.
Insights	Displays rating scores for assets to indicate their popularity.

Encoding Standards

AEM Assets supports all the relevant standards for metadata management. These standards help you create and manage metadata easily, reducing manual involvement.

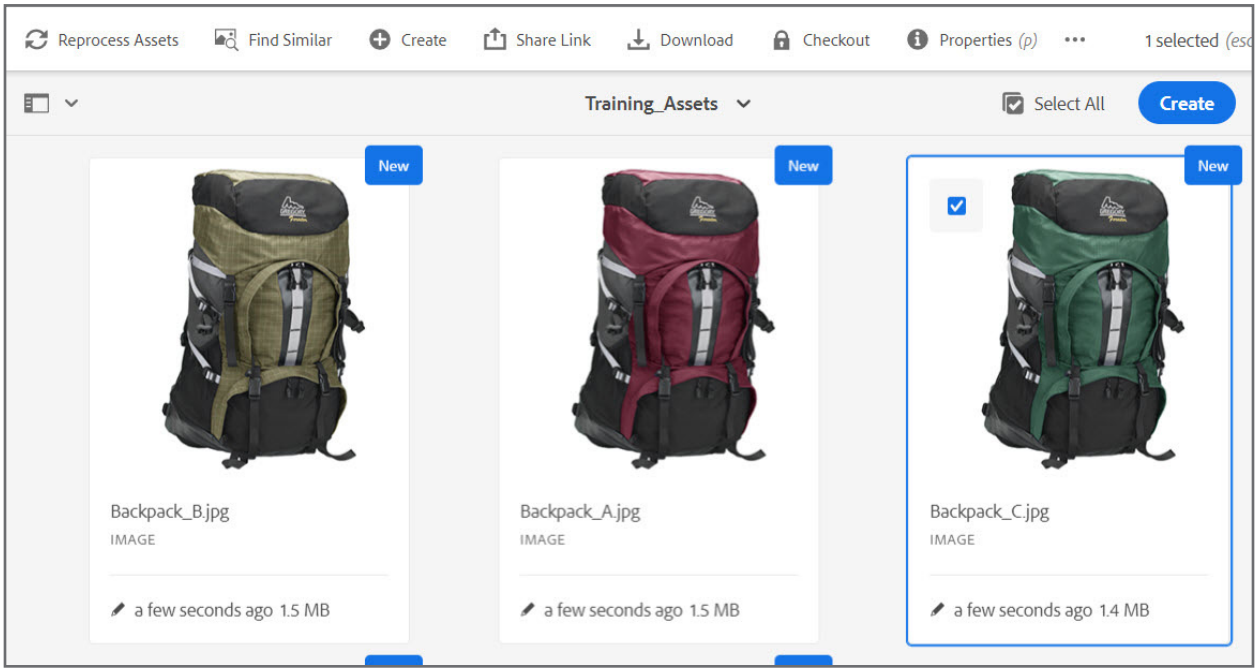
There are many ways to embed metadata into files. AEM supports the following encoding standards:

- Extensible Metadata Platform (XMP): Offers metadata encoding that can be embedded into all file formats.
- Iterative Dichotomiser 3 (ID3): Includes tags that can be stored in your audio and video files.
- Exchangeable Image File Formats (EXIF): Includes tags, which are created by digital cameras.
- Other metadata: Includes Microsoft Word, PowerPoint, and Excel.

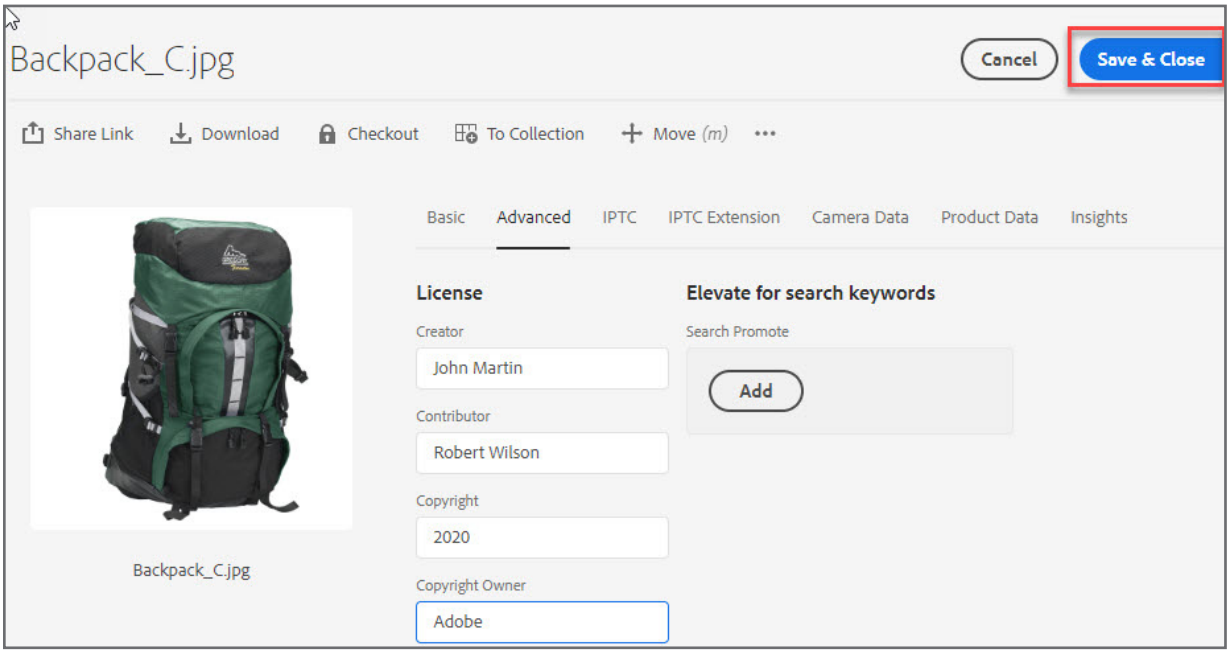
Exercise 2: Add metadata to an asset

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.

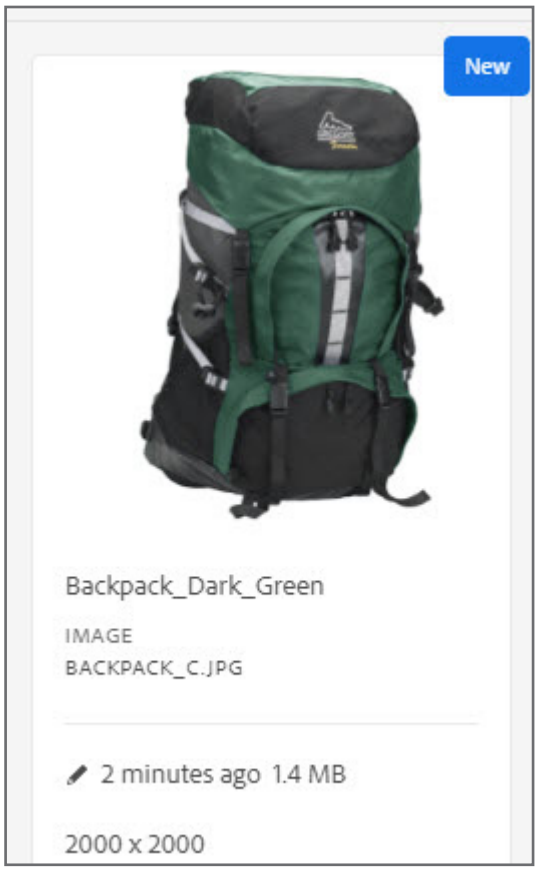


5. On the **Basic** tab, type the following details, as shown:
   a. Title: **Backpack_Dark_Green**
   b. Description: **Backpacks for tough running experience**
   c. Select **English (United States)** from the **Language** drop-down menu.
6. On the **Advanced** tab:
   a. In the **License** section, 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

You can use the following links for more information on:

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

Creating 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.

Objectives

After completing this course, you will be able to:

- Describe Maven
- Install Build Tools for AEM
- Explain AEM Archetype
- Create an AEM project
- Install project into AEM

Working with Maven

Maven helps build and manage Java-based projects. Maven specifies how software is built and describes the dependencies of the software. Maven has a central repository from which Maven can dynamically download Java libraries and plugins and store them in a local cache. This cache can also be updated by the developers with artifacts created by local projects. Public repositories can also be updated. Maven projects are configured using a Project Object Model (POM) stored in a pom.xml file.

Building your AEM project based on Maven offers you the following benefits:

- Integration Development Environment (IDE)-agnostic development environment
- Usage of Maven Archetype and Artifacts provided by Adobe
- Usage of Apache Sling and Apache Felix tool sets for Maven-based development setups
- Ease of project import into an IDE
- Easy integration with Continuous Integration Systems

Working with Node.js and NPM

Node.js (often shortened to Node), built on Chrome's V8 JavaScript engine, is an open-source development platform for executing JavaScript code outside of a browser. Node.js enables developers to use JavaScript to write command line (CLI) tools and for server-side scripting. Node.js is designed to build network applications that require a persistent connection from the browser. The most common example of a Node.js application is a web server.

Node Package Manager (NPM) is a CLI package manager that manages dependencies for Node.js packages. NPM manages an online repository of open source node.js packages. NPM makes installing Node.js packages fast and easy. NPM is installed when you install Node.js.
Node.js and NPM will be used for multiple purposes throughout this module.

If you already use a Maven Repository Manager, such as Sonatype Nexus, Apache Archiva, or JFrog Artifactory, add the appropriate configuration to your project. You can then reference the Maven repository manager that you are using and add Adobe's Maven Repository to your repository manager.

Exercise 1: Install build tools (Local only)

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

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

Note: You can skip this exercise 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 terminal window and type, as shown. The below command 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
```

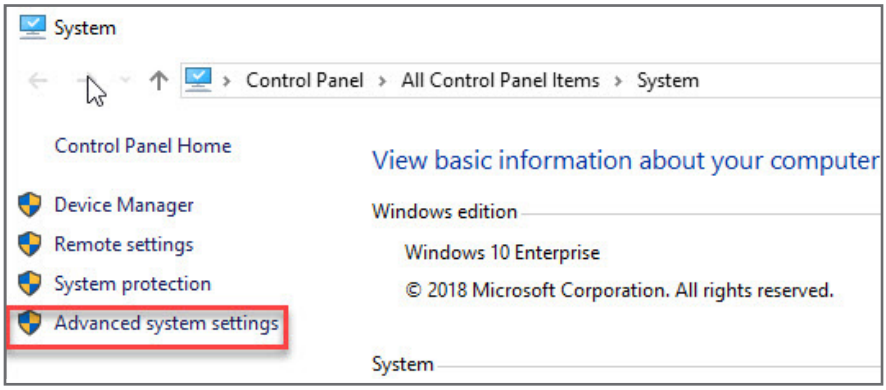
8. Test to see if you can successfully run Maven:

```
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:

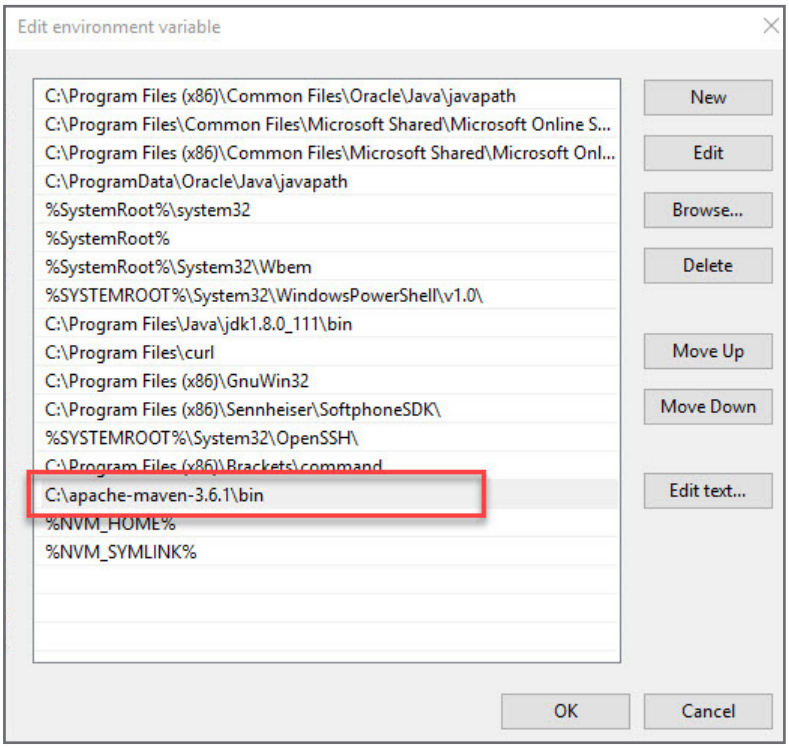


The **System Properties** window opens.

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, please 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.

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 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!

Common modules for an AEM Project:

- all - A container package that includes all artifacts of a project, including any 3rd 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.

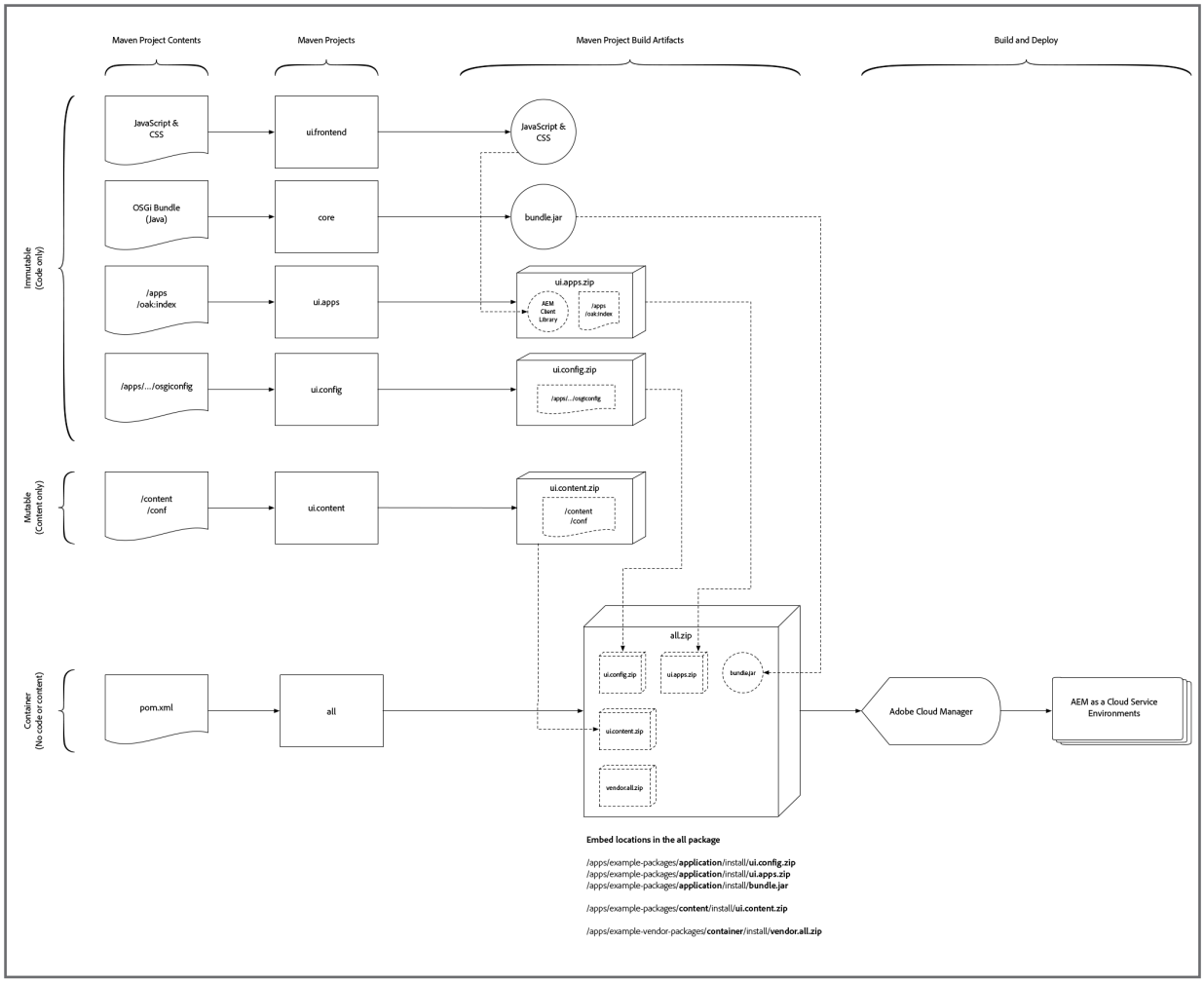
Mutable vs. Immutable content

/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.

Recommended Package Structure:



Code Packages / OSGi Bundles

- The OSGi bundle Jar file is generated, and directly embedded in the all project.
- The ui.apps package contains all the code to be deployed and only deploys to /apps. Common elements of the ui.apps package include, but are not limited to:
  - Component definitions and HTL scripts
    - /apps/my-app/components
  - JavaScript and CSS (via Client Libraries)
    - /apps/my-app/clientlibs
  - Overlays of /libs
    - /apps/cq, /apps/dam/, etc.
  - Fallback context-aware configurations
    - /apps/settings
  - ACLs (permissions)
    - Any rep:policy for any path under /apps

- The ui.config package, contains all OSGi configurations:
  - Organizational folder containing run mode specific OSGi config definitions
    - /apps/my-app/osgiconfig
  - Common OSGi configuration folder containing default OSGi configurations that apply to all target AEM as a Cloud Service deployment targets
    - /apps/my-app/osgiconfig/config
  - Run mode-specific OSGi configuration folders that contains default OSGi configurations that apply to all target AEM as a Cloud Service deployment targets
    - /apps/my-app/osgiconfig/config.<author|publish>.<dev|stage|prod>
  - Repo Init OSGi configuration scripts
    - Repo Init is the recommended way to deploy (mutable) content that is logically part of the AEM application. The Repo Init OSGi configurations should be places in the appropriate config.<runmode> folder as outlined above, and be used to define:

    i. Baseline content structures
    ii. Users
    iii. Service Users
    iv. Groups
    v. ACLs (permissions)
- The all package is a container package that ONLY includes the ui.apps and ui.content packages as embeds. The "all" package must not have any content of its own, but rather delegate all deployment to the repository to its subpackages.

Content Packages

- The ui.content package contains all content and configuration. The Content Package, contains all the node definitions not in the ui.apps or ui.config packages, or in other words, anything not in /apps or /oak:index. Common elements of the ui.content package include, but are not limited to:
  - Context-aware configurations
    - /conf
  - Required, complex content structures (ie. Content build-out that is builds on and extends past Baseline content structures defined in Repo Init.)
    - /content, /content/dam, etc.
  - Governed tagging taxonomies
    - /content/cq:tags
  - Legacy etc nodes (Ideally, migrate these to non-/etc locations)
    - /etc

Container Packages

- The all package is a container package that ONLY includes deployable artifacts, the OSGI bundle Jar file, ui.apps, ui.config and ui.content packages as embeds. The all package must not have any content or code of its own, but rather delegate all deployment to the repository to its sub-packages or OSGi bundle Jar files.
  Packages are now included using the Maven FileVault Package Maven plugin's embeddeds configuration, rather than the <subPackages> configuration.
  For complex Experience Manager deployments, it may be desirable to create multiple ui.apps, ui.config and ui.content projects/packages that represent specific sites or tenants in AEM. If this is done, ensure the split between mutable and immutable content is respected, and the required content packages and OSGi bundle Jar files are embedded as sub-packages in the all container content package.
  For example, a complex deployment content package structure might look like this:
- all content package embeds the following packages, to create a singular deployment artifact
  - common.ui.apps deploys code required by both site A and site B
  - site-a.core OSGi bundle Jar required by site A
  - site-a.ui.apps deploys code required by site A
  - site-a.ui.config deploys OSGi configurations required by Site A
  - site-a.ui.content deploys content and configuration required by site A
  - site-b.core OSGi bundle Jar required by site B
  - site-b.ui.apps deploys code required by site B
  - site-b.ui.config deploys OSGi configurations required by site B
  - site-b.ui.content deploys content and configuration required by site B

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:

```
01.  <dependency>
02.    <groupId>com.adobe.aem</groupId>
03.    <artifactId>uber-jar</artifactId>
04.    <version>6.4.0</version>
05.    <classifier>apis</classifier>
06.    <scope>provided</scope>
07.  </dependency>
```

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

Exercise 2: Create an AEM project

In this exercise 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

This exercise 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 are 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 from **Exercise_Files_TB\Project_using_Maven\adobe-archetype-repo.xml**, 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, then copy **Exercise_Files_TB\Project_using_Maven\adobe-archetype-repo.xml** to \<user>**.m2** directory and rename the file to **settings.xml**.

Task 2: Create an AEM project

1. Open a **Command Prompt** window by right-clicking the **Command Prompt** app/icon and choose as **Run as administrator.**
2. In your **Command Prompt** window, navigate to **\adlsadmin** directory by issuing the command:

```
cd c:\adobe
```

Note: In order to create the AEM project in step 2, you can use the appropriate create script provided in the **Exercise_Files_TB** folder or you can use the create script (6.5/Cloud Service/React/Angular) specified by your instructor for you to use in this class.

3. Navigate to **Exercise_Files_TB\Project_using_Maven** and copy the content from the file specified by your instructor. For example: **create-CS-project-25.bat** on Windows (For Cloud Service on Mac, use **create-CS-project-25.sh**. For 6.5, use **create-6.5-project-25.bat** on Windows and **create-6.5-project-25.sh** on Mac):

```
mvn -B archetype:generate ^
 -D archetypeGroupId=com.adobe.aem ^
 -D archetypeArtifactId=aem-project-archetype ^
 -D archetypeVersion=25 ^
 -D appTitle=TrainingProject ^
 -D appId=training ^
 -D artifactId=training ^
 -D groupId=com.adobe ^
 -D package=com.adobe.training ^
 -D version=1.0-SNAPSHOT ^
 -D aemVersion=cloud ^
 -D sdkVersion=latest ^
 -D includeDispatcherConfig=y ^
 -D frontendModule=none ^
 -D language=en ^
 -D country=us ^
 -D singleCountry=y ^
 -D includeExamples=y ^
 -D includeErrorHandler=n ^
 -D includeCommerce=n ^
 -D commerceEndpoint= ^
 -D includeForms=n ^
 -D sdkFormsVersion= ^
```

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

```
Command Prompt
Microsoft Windows [Version 10.0.18362.959]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\prpurush>mvn archetype:generate ^ -DarchetypeGroupId=com.adobe.granite.archetypes ^ -Darchetype
ArtifactId=aem-project-archetype ^ -DarchetypeVersion=23 ^ -DgroupId=com.adobe ^ -DartifactId=training ^
 -Dversion=1.0-SNAPSHOT ^ -Dpackage=com.adobe.training ^ -DappId=training ^ -DappTitle=TrainingProject ^
 -DaemVersion=cloud ^ -DsdkVersion=latest ^ -DlanguageCountry=en_us ^ -DincludeExamples=y ^ -DincludeErr
orHandler=n ^ -DfrontendModule=none ^ -DsingleCountry=y ^ -DincludeDispatcherConfig=y
```

The command runs, which may take a few minutes.

Note: Make sure to run the command as administrator in order to successfully execute the command.

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:

```
Command Prompt
cr_root\conf\training\settings\wcm\templates\.content.xml
[WARNING] CP Don't override file C:\Users\prpurush\training\ui.content\src\main\content\j
cr_root\conf\training\settings\wcm\templates\_rep_policy.xml
[WARNING] CP Don't override file C:\Users\prpurush\training\ui.content\src\main\content\j
cr_root\conf\training\settings\wcm\templates\content-page\initial\.content.xml
[WARNING] CP Don't override file C:\Users\prpurush\training\ui.content\src\main\content\j
cr_root\conf\training\settings\wcm\templates\content-page\.content.xml
[WARNING] CP Don't override file C:\Users\prpurush\training\ui.content\src\main\content\j
cr_root\conf\training\settings\wcm\templates\content-page\policies\.content.xml
[WARNING] CP Don't override file C:\Users\prpurush\training\ui.content\src\main\content\j
cr_root\conf\training\settings\wcm\templates\content-page\thumbnail.png
[WARNING] CP Don't override file C:\Users\prpurush\training\ui.content\src\main\content\j
cr_root\conf\training\settings\wcm\templates\content-page\structure\.content.xml
[WARNING] CP Don't override file C:\Users\prpurush\training\ui.content\src\main\content\j
cr_root\conf\training\.content.xml
[INFO] Parent element not overwritten in C:\Users\prpurush\training\it.tests\pom.xml
[INFO] Parent element not overwritten in C:\Users\prpurush\training\it.launcher\pom.xml
[INFO] Executing META-INF/archetype-post-generate.groovy post-generation script
[INFO] Project created from Archetype in dir: C:\Users\prpurush\training
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  01:48 min
[INFO] Finished at: 2019-08-29T14:31:25-07:00
[INFO] ------------------------------------------------------------------------
```

6. Navigate to **...\adobe** and verify the project training has been created, as shown:

You have successfully created an AEM project.

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
```

Exercise 3: Install the project into AEM

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

Prerequisite:

- AEM author service is up and running

This exercise includes the following tasks:

1. Update the POM file
2. Install the project into AEM
3. Verify the installed content packages

Task 1: Update the POM file

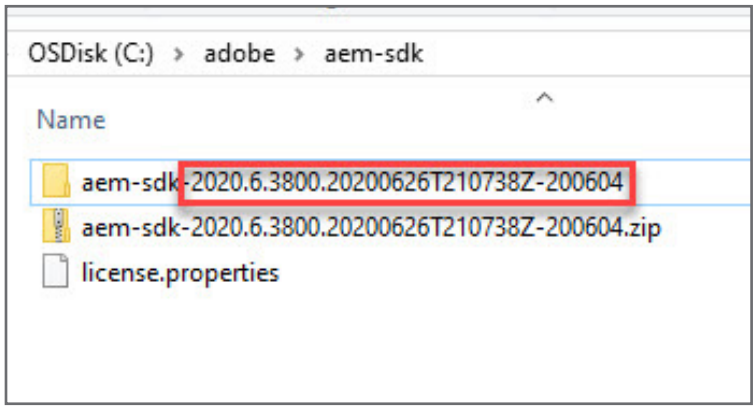
When you create a new project from the archetype, it automatically sets the SDK or Uberjar version for you. When developing your application, you want to make sure the build in your POM matches the SDK build or 6.5 Service Pack number you're using. Depending how often you download and update your local development environment, you might need to update the build in your POM.
In this task you will update the dependency version in your POM to match either the SDK build if your using Cloud Service or the AEM Service pack on the Uberjar if you're using 6.5.
Depending on the type of AEM deployment you have, you will need to update a different dependency:

- AEM Cloud Service needs to have the SDK API dependency updated with the same SDK build number.
- AEM 6.5 and lower need to have the uberjar dependency updated with the same AEM Service Pack number

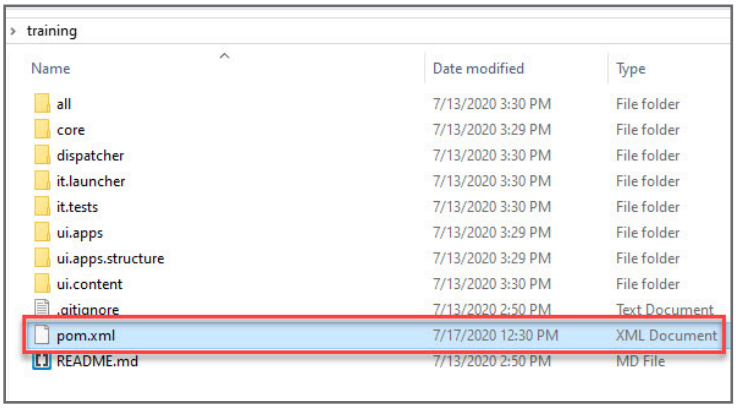
Follow these steps for AEM as a Cloud Service:

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

2. The SDK zip contains the build number: **aem-sdk-<buildNumber>.zip**. Copy this number:



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

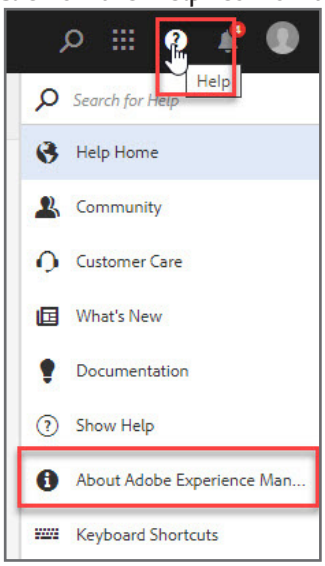


4. Find the property value <aem.sdk.api> (near the top) and paste the build number that you copied from the SDK zip, as shown:

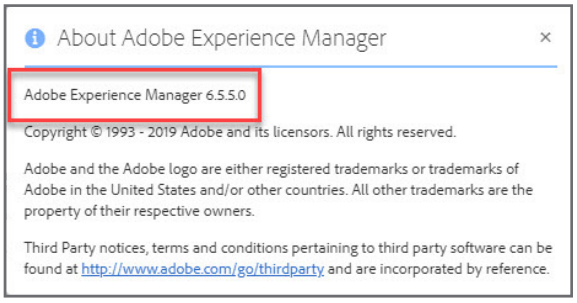
5. Save the file.

Follow these steps for AEM 6.5 and lower Only:

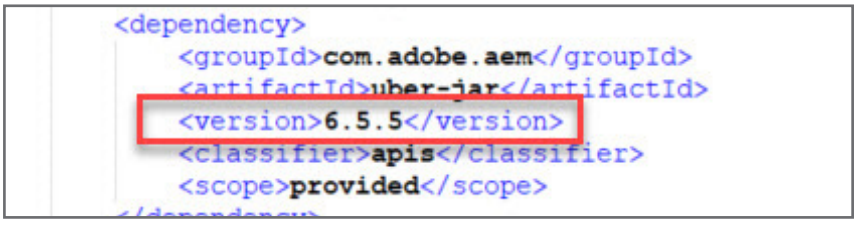
1. If you are unsure about the version of AEM Service Pack you are using, open AEM in the browser: http://localhost:4502/
2. Click on the Help icon on the top right corner > About Adobe Experience Manager



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:



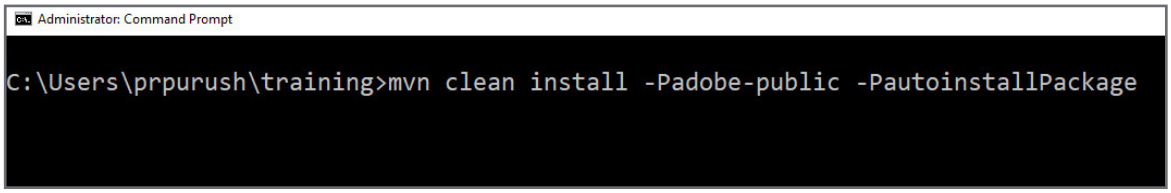
Task 2: Install the project into AEM

1. Open a Command Prompt window and navigate to your newly created Maven project **training**.

Note: This is the project directory **training** that you created in the previous Exercise.

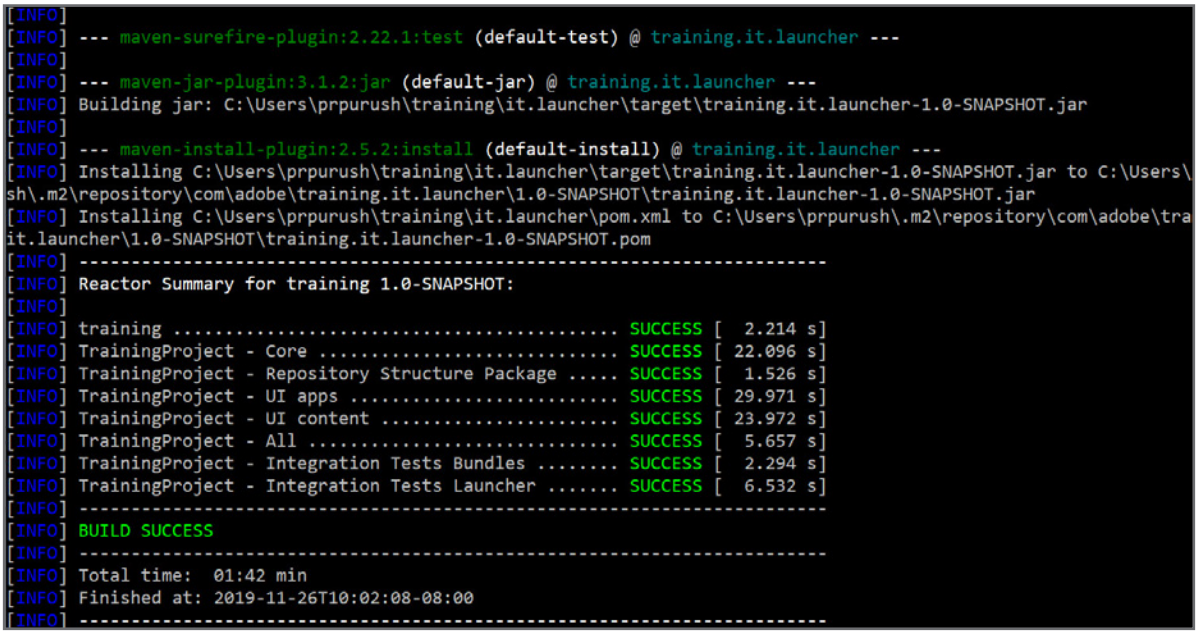
2. Run the following command and press **Enter**. The project starts to build.

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



Note: The project build may take a few minutes.

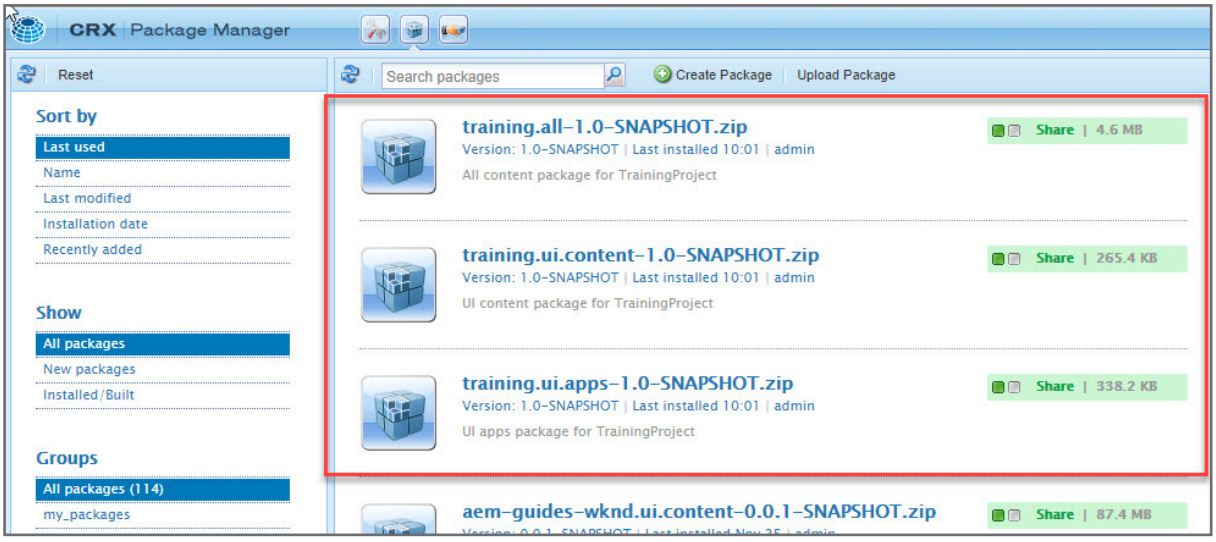
3. Verify the build is success, as shown:



Task 3: Verify the installed content packages

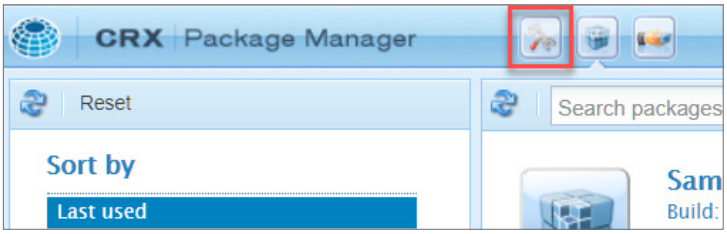
In this task, you will navigate to CRXDE Lite and verify the newly installed content package.

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 training packages, as shown:



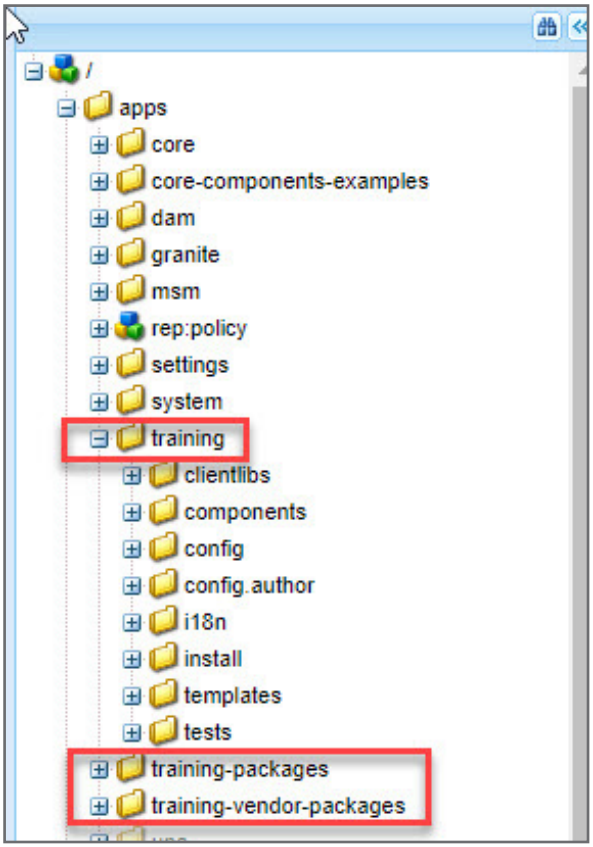
Note: The profile autoInstallSinglePackage installed the container package all. This all package then installed all of the mutable and immutable content packages.

4. Click the Develop icon as shown:

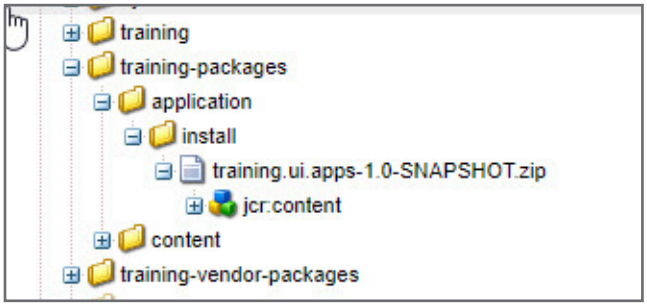


This takes you back to **CRXDE Lite**.

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



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



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

Note: Modern AEM projects organize the content into mutable and immutable content. Mutable content 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 **training** site. Navigate to **training** > **us**, as shown.

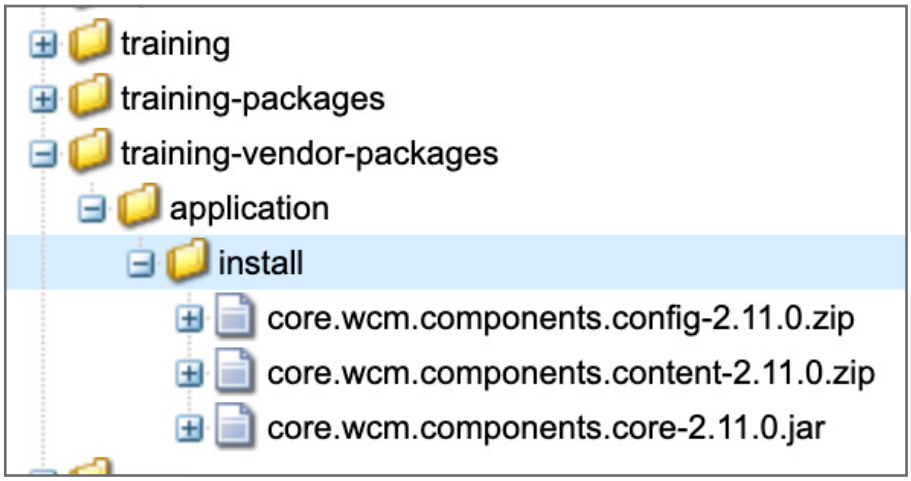
Congratulations! You have successfully installed your project into AEM.

Task 4: Core Components Inclusion (6.5 only)

Scenario

AEM as a Cloud Service uses an 'always up to date' model which means the SDK quickstart will always contain the latest core components and they shouldn't be installed with your project. AEM 6.5 and lower projects must include core components to indicate what release you would like to use. A project created from the archetype will result in a project that includes core components by default. To view core components being installed:

1. In CRXDE Lite, navigate to the **/apps/training-vendor-packages/content/install** folder and notice the install of **the core components**, as shown:



Note: Because the AEM project you installed depends on core components, you will notice core components under application. Under content, you will notice the examples core components.

Core components are installed based on the version in the parent POM and then embedded as 3rd party artifacts in the all pom.

2. Open up the **parent pom.xml** and find the property <core.wcm.components.version> to view the version that's apart of your project:

```
perties>
<aem.host>localhost</aem.host>
<aem.port>4512</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>
<core.wcm.components.version>2.11.0</core.wcm.components.version>
<bnd.version>5.1.2</bnd.version>
<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
```

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:

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

References

You can use the following links for more information on:

- Development Tools for AEM Projects
- AEM Archetype
- React frontend module
- 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 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.

Objectives

After completing this course, you will be able to:

- Install and configure Eclipse
- Install and configure the Eclipse AEM plugin
- Import an AEM project in Eclipse
- Synchronize AEM content

Installing and Configuring Eclipse

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 plug-in for Eclipse

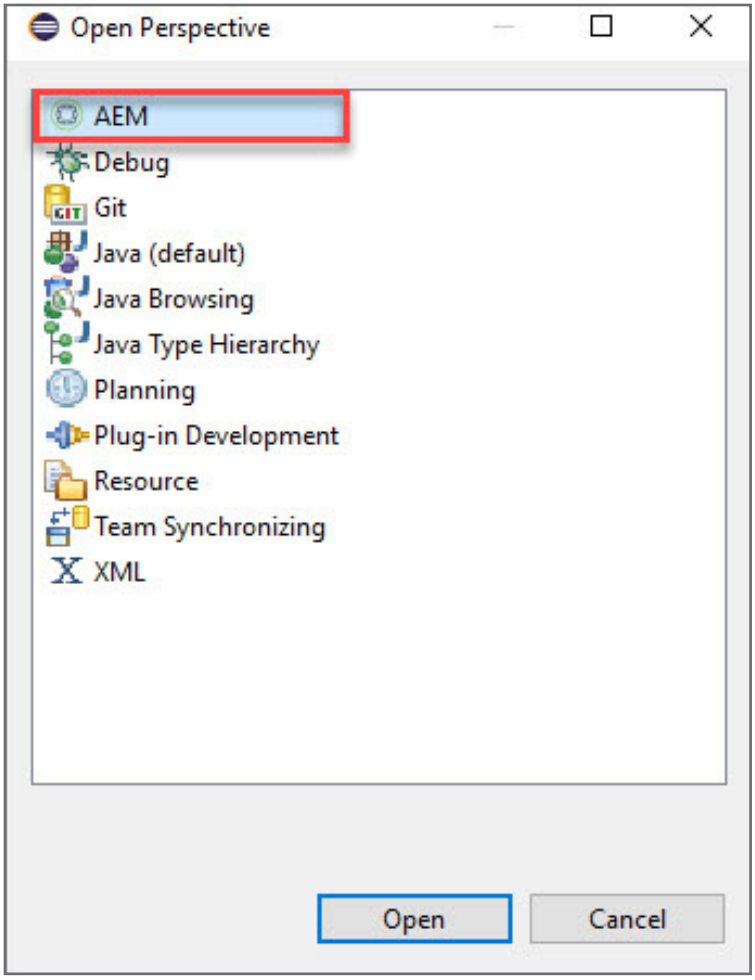
Benefits of AEM Plug-in for Eclipse

The AEM plug-in 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.

Exercise 1: Install and configure Eclipse (Local only)

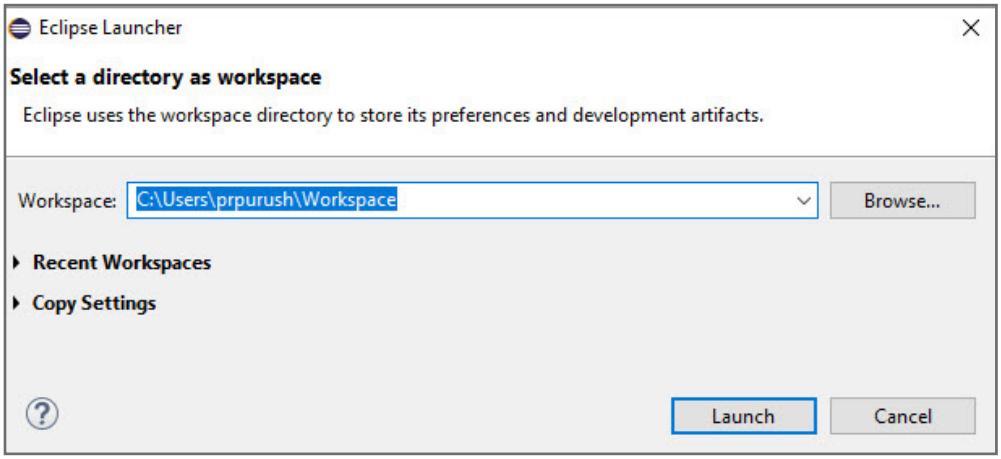
In this exercise, you will install and configure Eclipse.
This exercise includes the following tasks:

1. Install Eclipse
2. Configure Eclipse

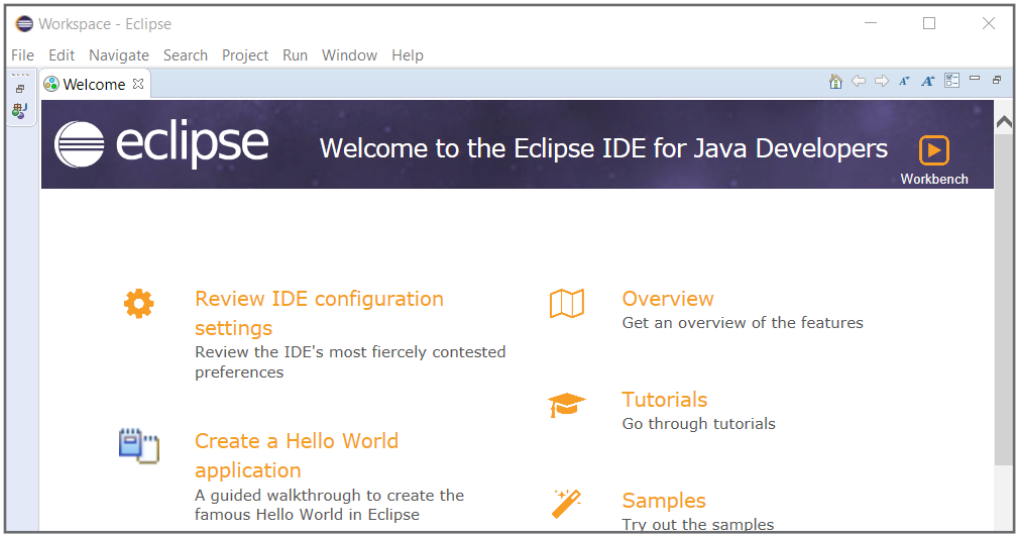
Note: You can skip this exercise if you use a ReadyTech environment because AEM is already installed as part of the image.

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:



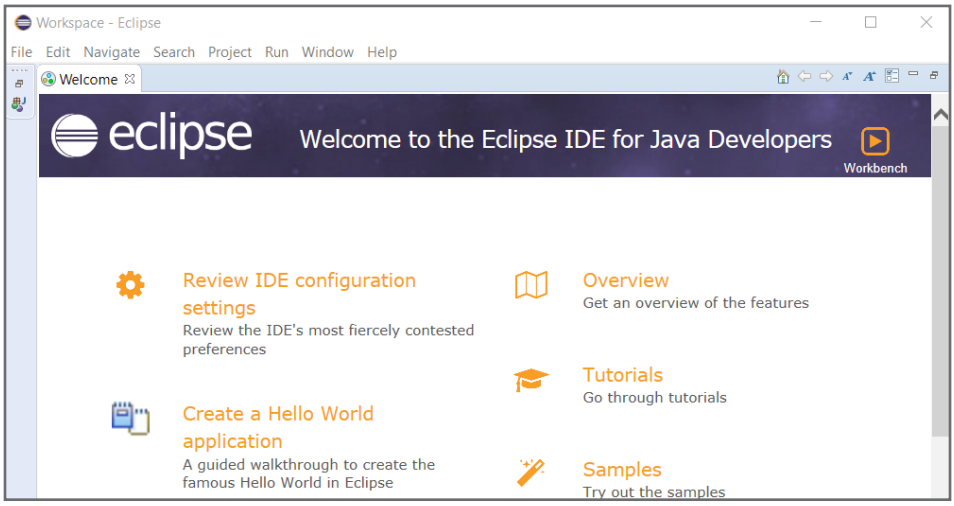
4. Accept the default workspace and click **Launch**. The Eclipse Development Environment opens, as shown:



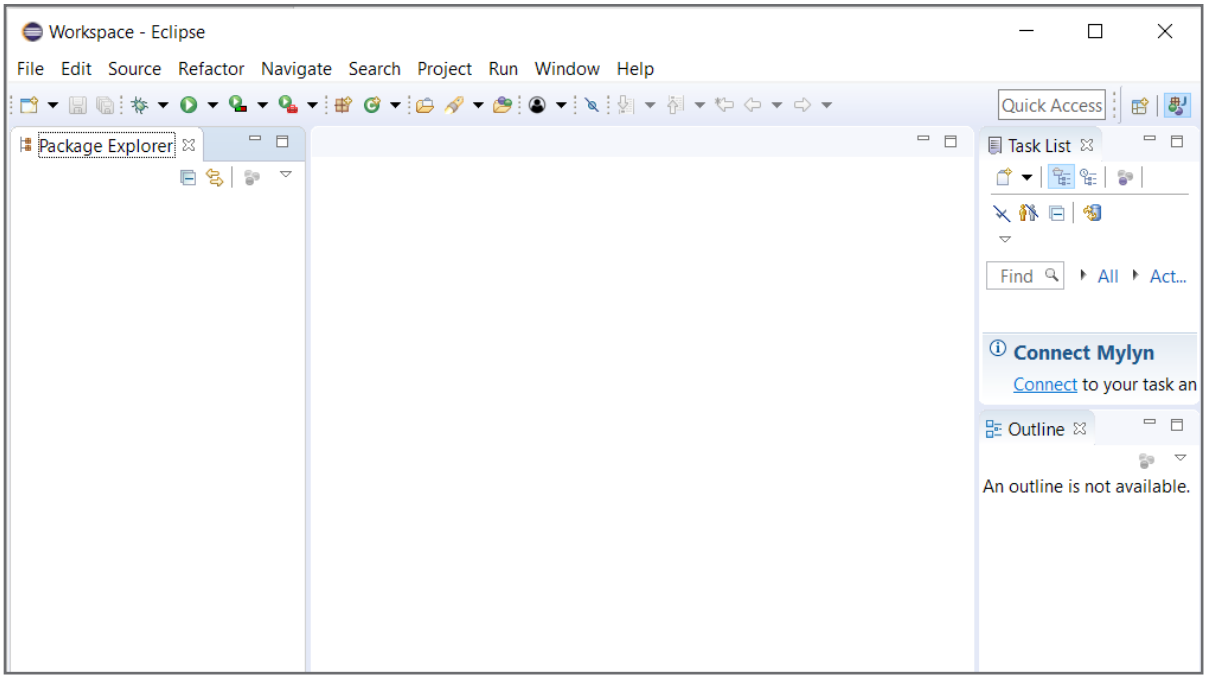
Task 2: Configure Eclipse

In this task, you will configure Eclipse.

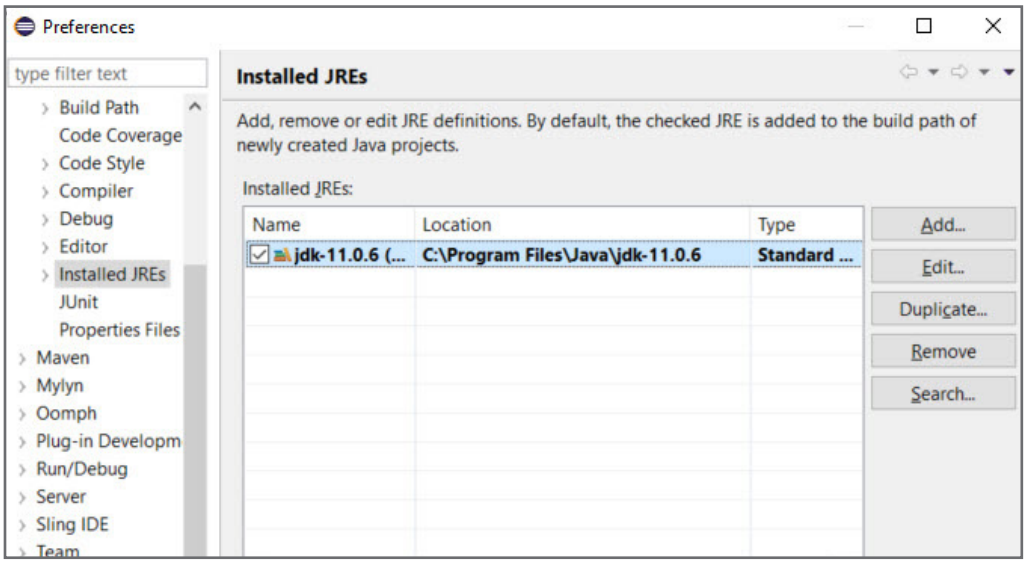
1. Click the **Workbench** logo in the upper-right corner, as shown, to close the Welcome screen.



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.

Exercise 2: Install the Eclipse AEM plug-in (Local only)

In this exercise, you will install and configure Eclipse AEM plug-in.

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

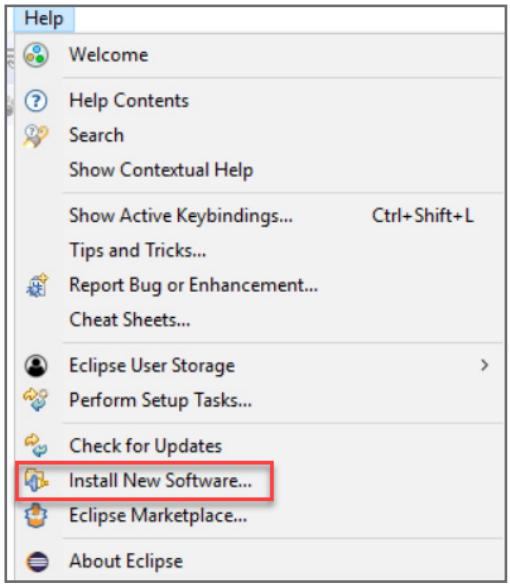
1. Open the URL https://eclipse.adobe.com/aem/dev-tools/ or use the file provided in the **Exercise_Files_TB**.

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

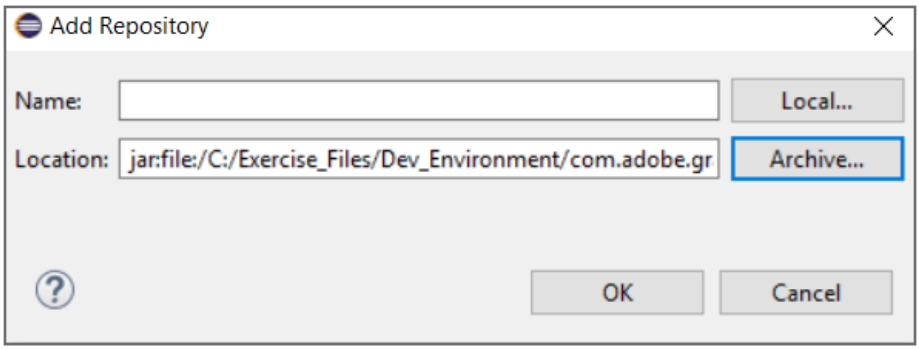
You will perform the offline method in this exercise.

To install your package:

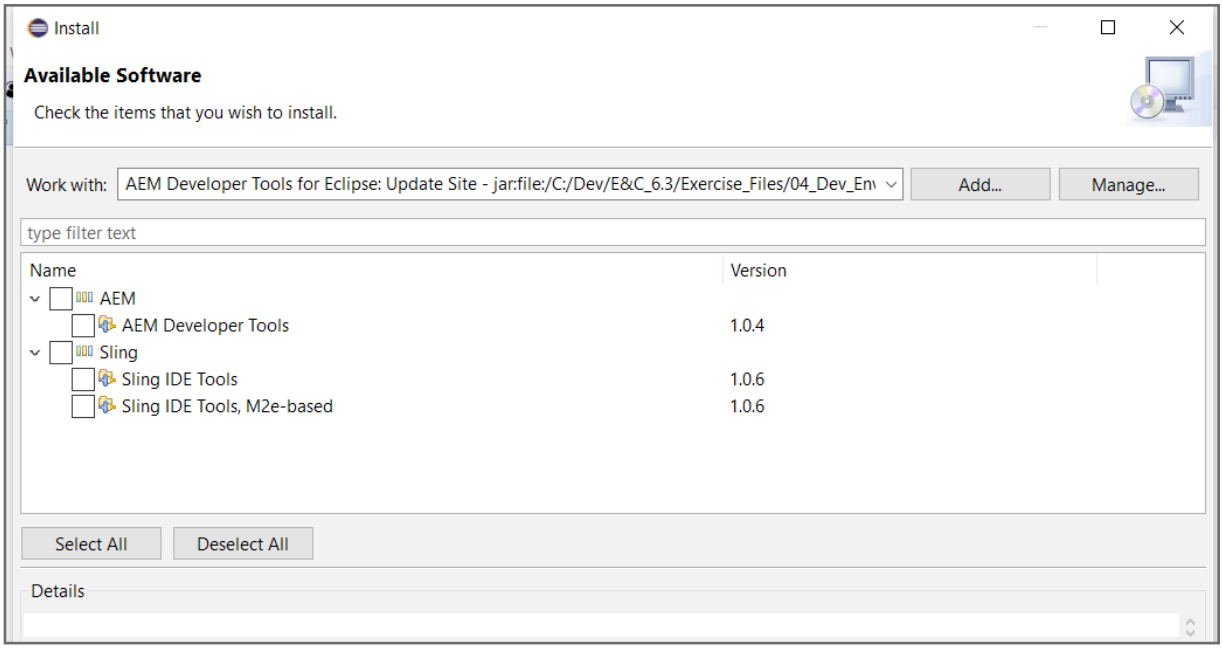
2. Double-click **eclipse.exe** (or eclipse.app) to start Eclipse. The Eclipse Development Environment opens:
3. Select **Help** > **Install New Software,** as shown. The Install window opens.



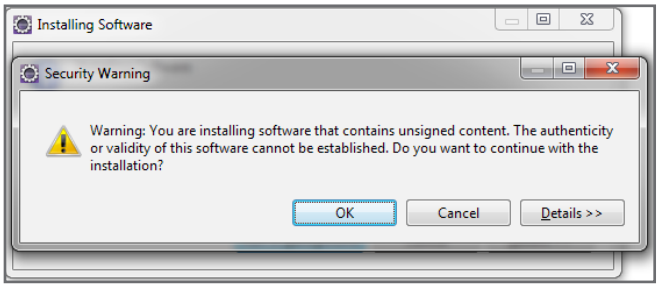
4. Click **Add**. The Add Repository dialog box opens.
5. Click **Archive**.
6. Navigate to the repository archive and select the zip file (**com.adobe.granite.ide.p2update-1.3.0.zip**) provided for the plug-in from **Exercise_Files_TB/Dev_Environment/**.
7. Click **Open** to add the location to the **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:



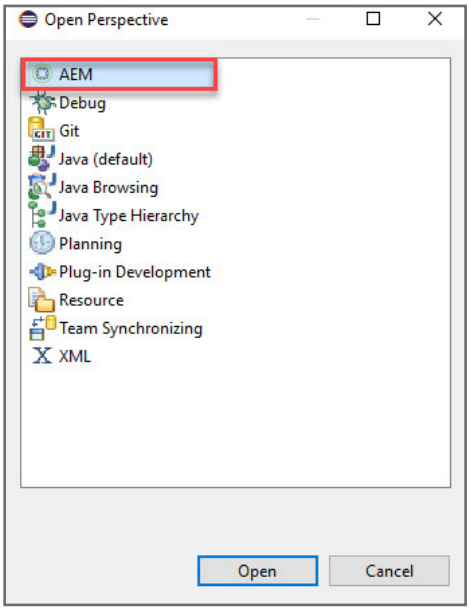
10. Click **Select All** to select AEM and Sling.
11. Click **Next**. The Install Details screen opens.
12. Click **Next**. The Review Licenses screen opens.
13. Click the **I accept the terms of the license agreements** option and then click **Finish**. The **Installing Software** dialog box with a 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.

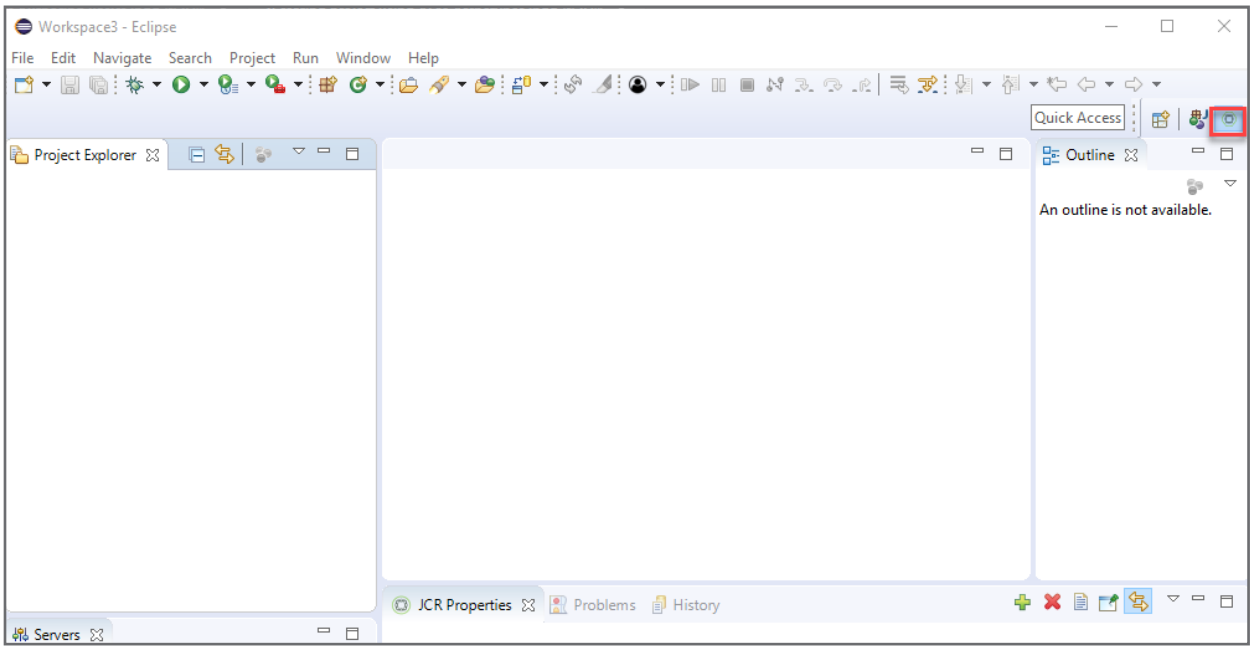
Note: This process takes about a minute. If you see a dialog box that asks if you want to keep the current location of your Eclipse install, click **OK**. Eclipse opens.

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 exercise.

Exercise 3: Import an AEM project

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

This exercise includes the following task:

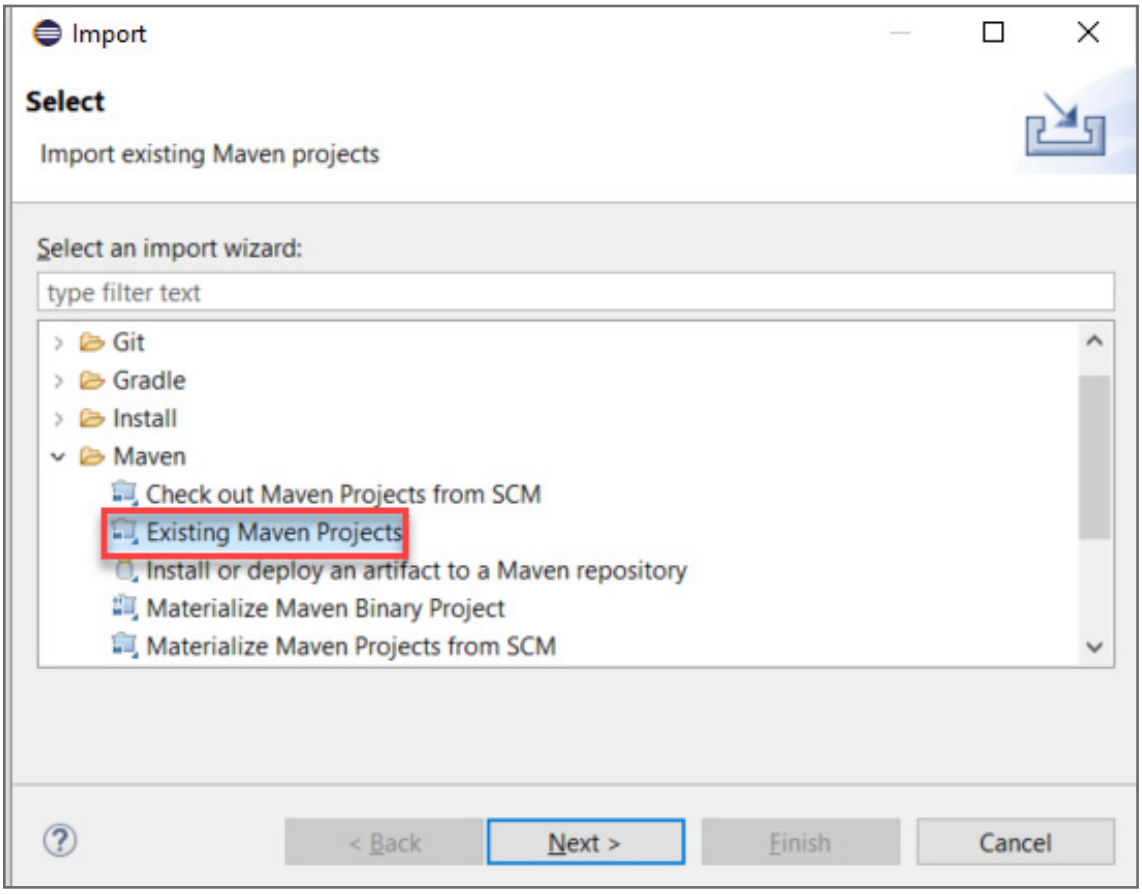
1. Import an AEM project
2. Configure jcr_root folders

Task 1: Import an AEM project

In this task, you will import an AEM project.

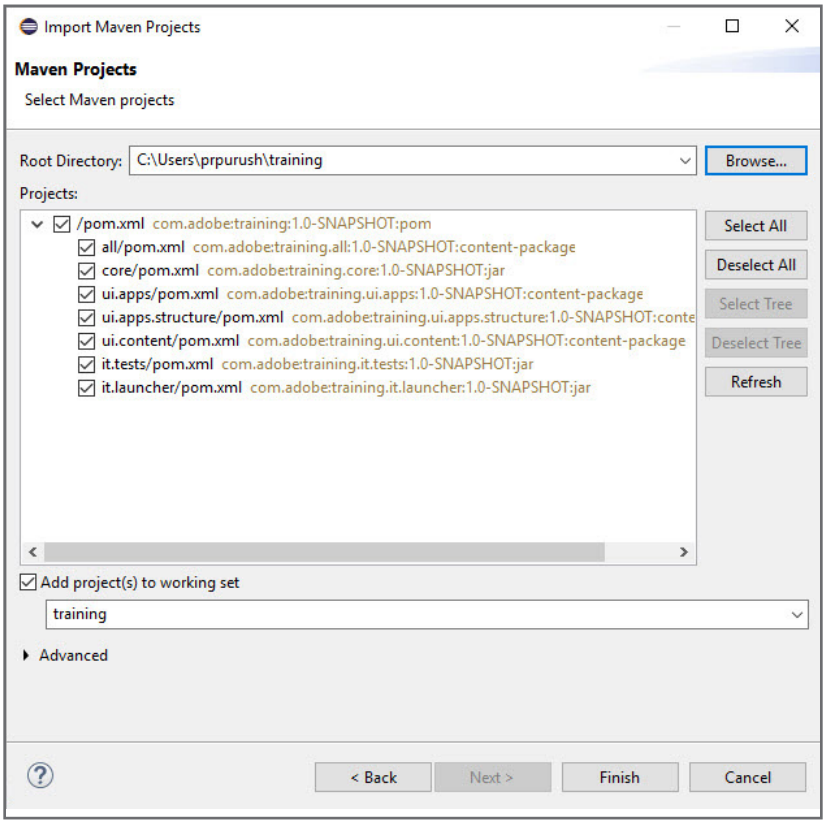
1. Right-click the **Eclipse** icon, and select **Open** from the list. The **Eclipse Launcher** opens .
2. Click **Launch**. The eclipse-workspace – Eclipse IDE window opens.
3. Click **File** > **Import**. The **Import** window opens.

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**.

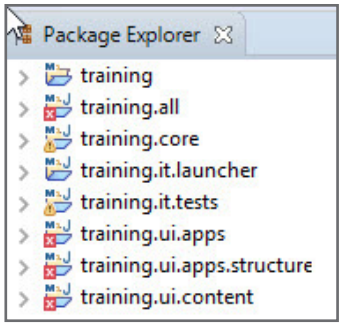
8. Select all the POM files, as shown: :



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 :



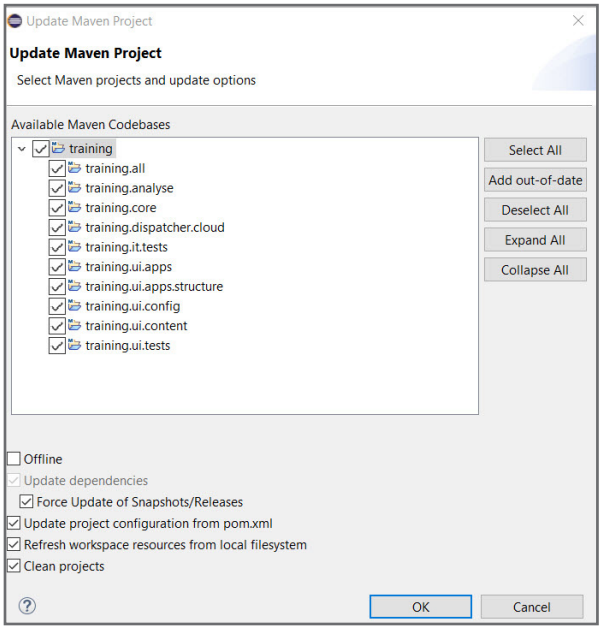
Note: Just like the screenshot above, your project will probably contain errors. These errors can be from a variety of things and you can optionally ignore these in the Eclipse preferences.

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.

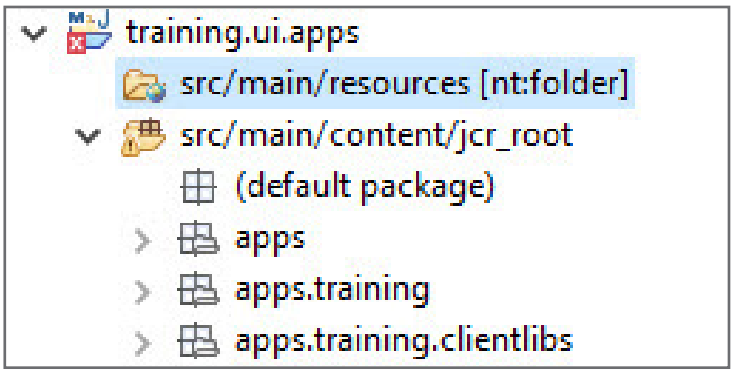


15. Ignore any errors. You will fix them later.

You have successfully imported an AEM project.

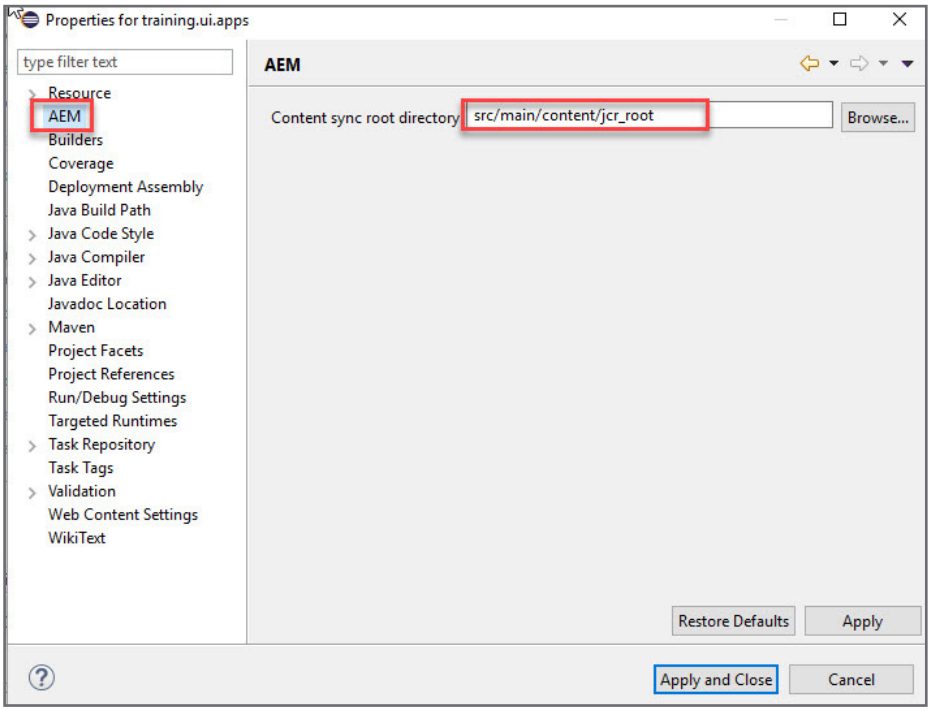
Task 2: Configure jcr_root folders

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

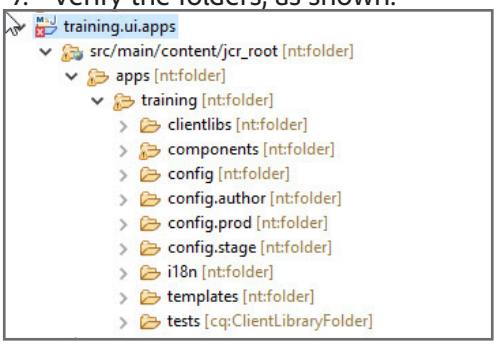


Note: AEM project is the name of the project you created and imported. For example **training**. The name of your project might be different from the screenshot shown.

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.

Exercise 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 exercise includes the following tasks:

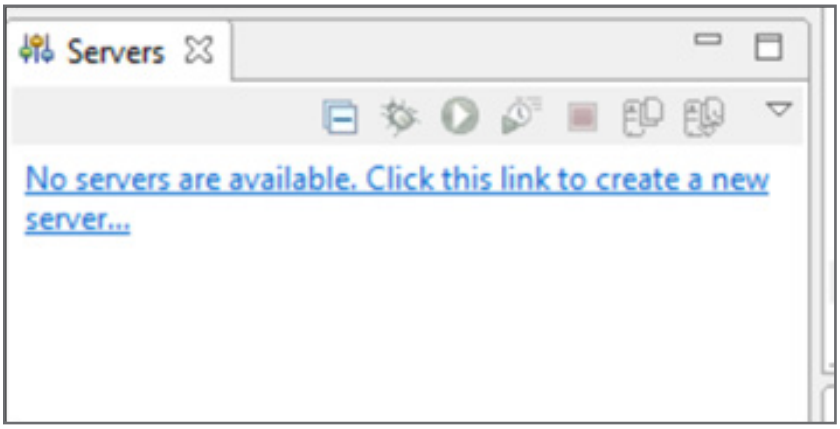
1. Configure the AEM server
2. Sync AEM content

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).

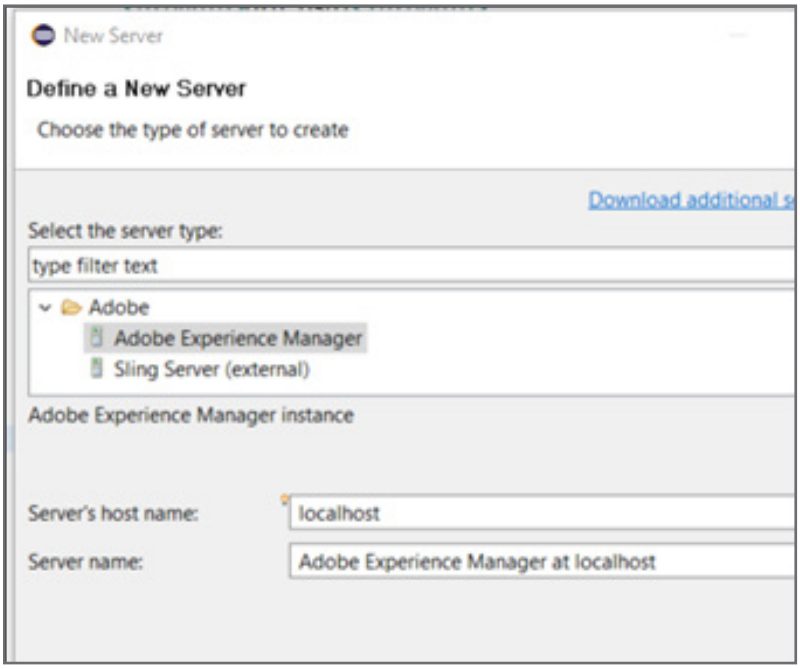
1. In Eclipse, verify **AEM Perspective** is selected in the upper-right corner.

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, as shown:



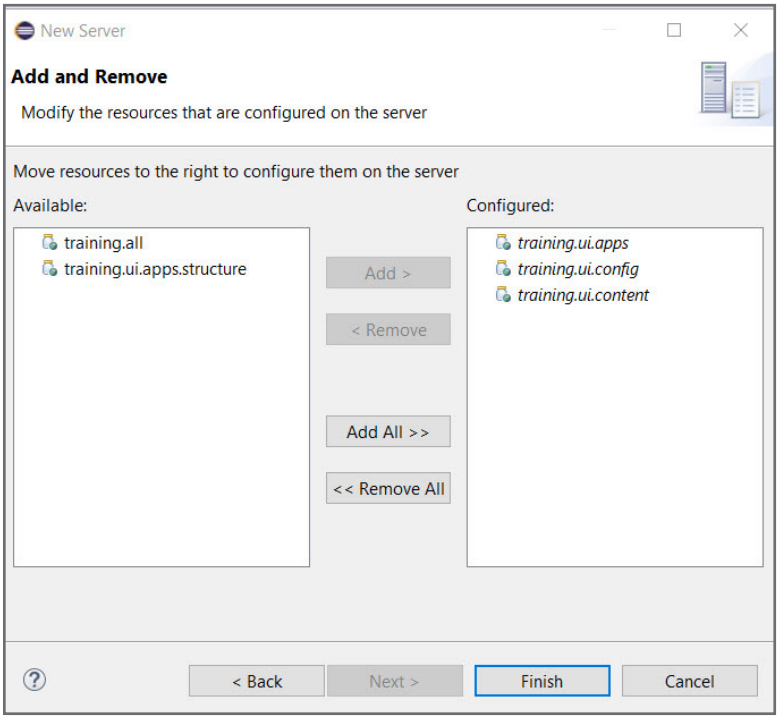
The **Define a New Server** dialog box opens.

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



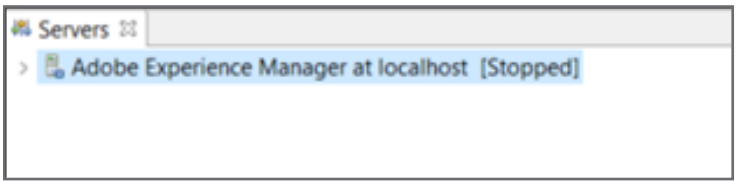
4. Accept the default settings for the **Server's host name** and **Server name** fields.
5. Click **Next**. The **Add and Remove resources** dialog box opens.

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:



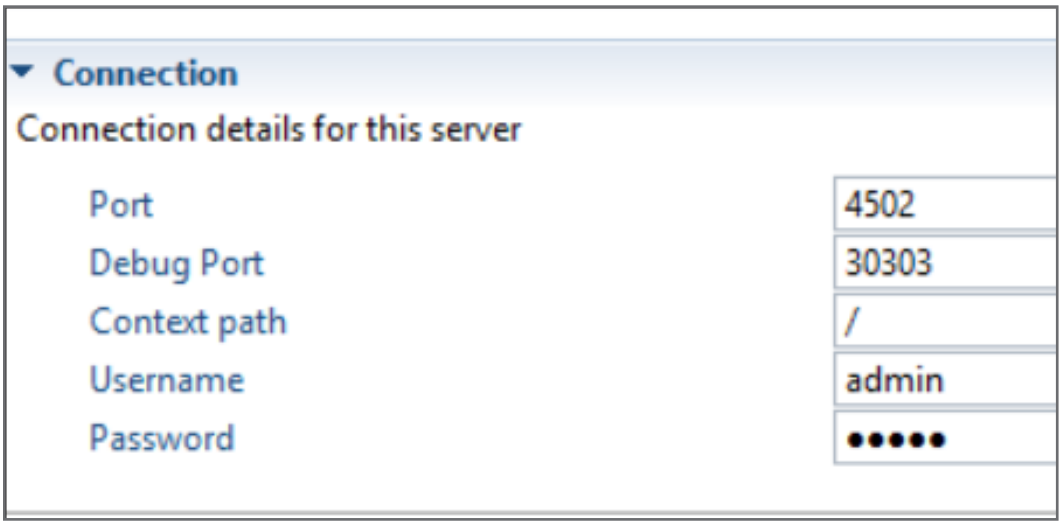
Note: AEM project is the name of the project you created and imported. For example **training**. The name of your project might be different from the screenshot shown.

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:

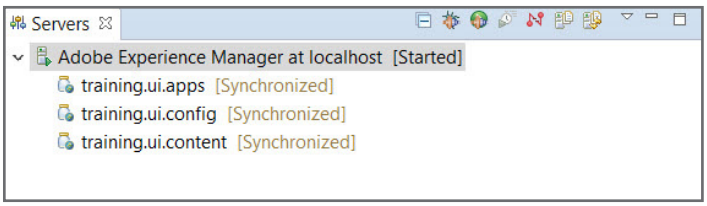


11. Save the changes (**File** > **Save** OR **Ctrl+S)**.

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 exercise.

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>
```

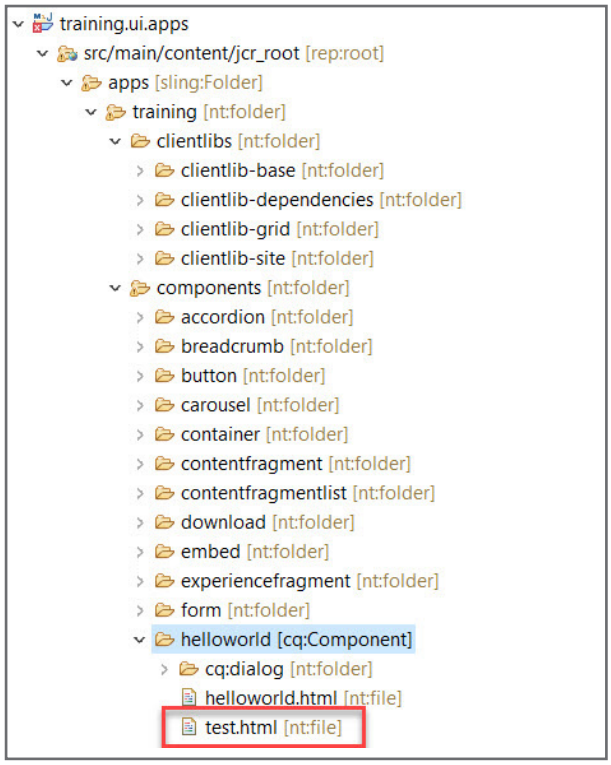
Note: Any files saved in your project will auto sync with AEM based on the filter.xml file in each module that is configured with the AEM connection.

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

You can use the following links for more information on:

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

Development 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.

Objectives

After completing this course, you will be able to:

- Install Visual Studio Code
- Describe VS Integrated Terminal
- Describe VSCode AEM Sync
- Install VSCode AEM Sync
- Test VSCode AEM Sync Extension

Visual Studio Code

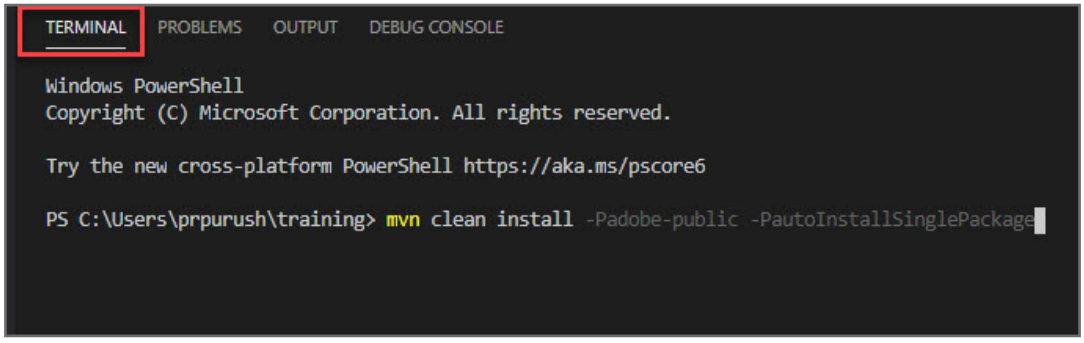
Visual Studio Code is a free source-code editor developed by Microsoft for Windows, Linux and macOS. It comes with built-in support for JavaScript, TypeScript and Node.js and has a rich ecosystem of extensions for other languages (such as C++, C#, Java, Python, PHP, and Go) and runtimes (such as .NET 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.

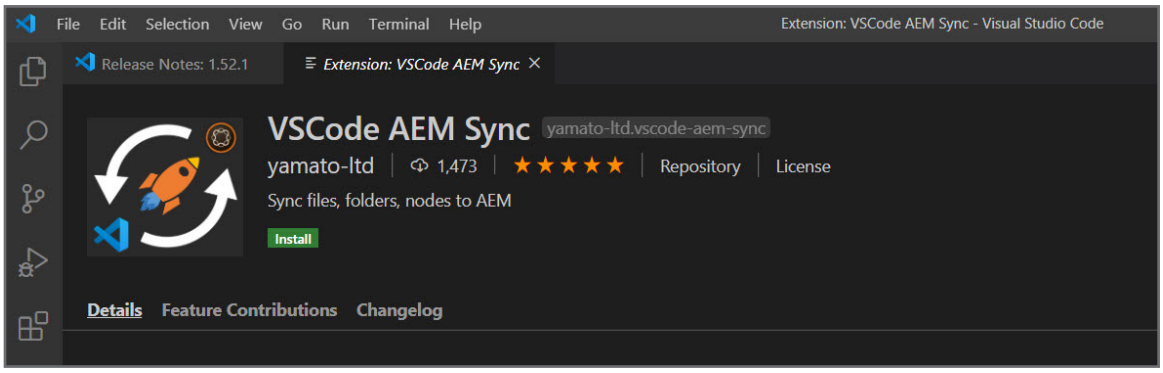
To open the terminal:

- 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 not only files also .content.xml, dialog.xml, etc., can be synced by just one click. This sync feature is ported from AEM Brackets Extension.

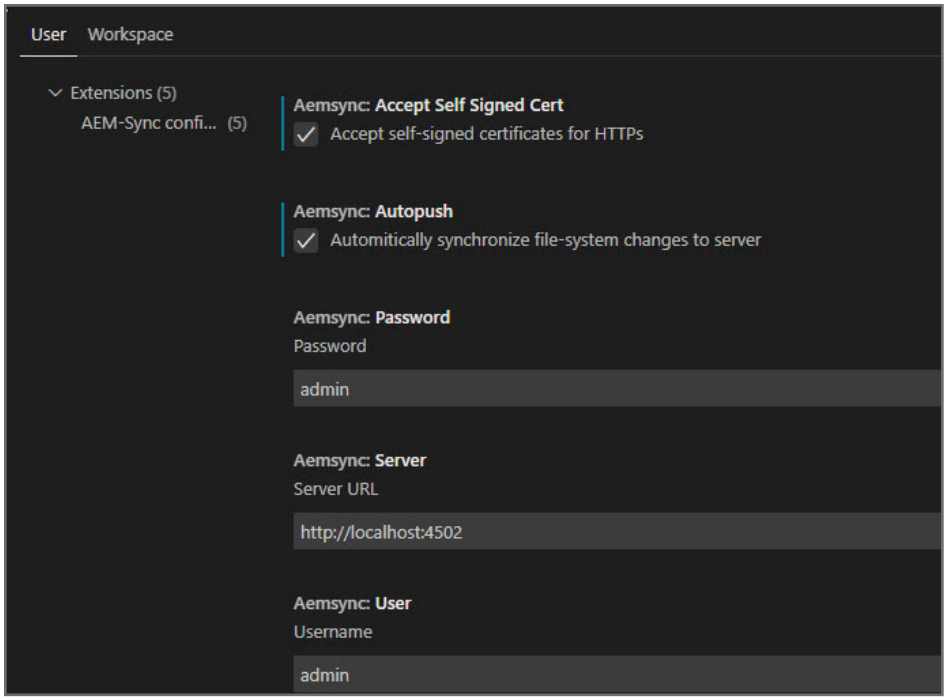


Settings

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.



Exercise 1: Install Visual Studio Code (Local only)

In this exercise, you will install Visual Studio Code.

Note: You can skip this exercise 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.

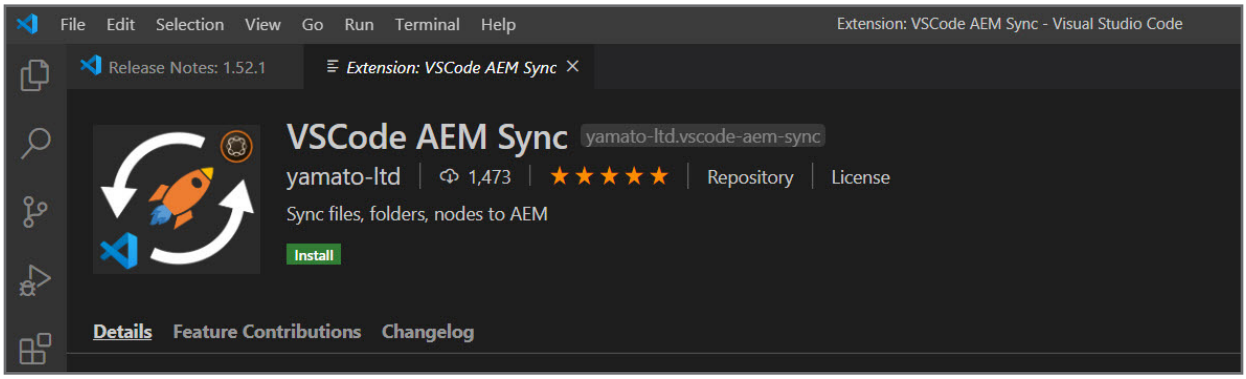
Congratulations! You have installed Visual Studio Code and are now ready to install the VSCode AEM Sync extension.

Exercise 2: Install VSCode AEM Sync (Local only)

In this exercise, you will install VSCode AEM Sync.

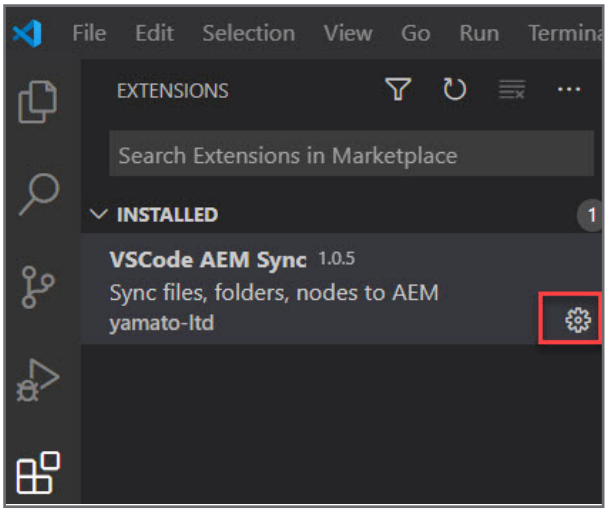
Note: You can skip this exercise if you use a ReadyTech environment because AEM is already installed as part of the image.

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

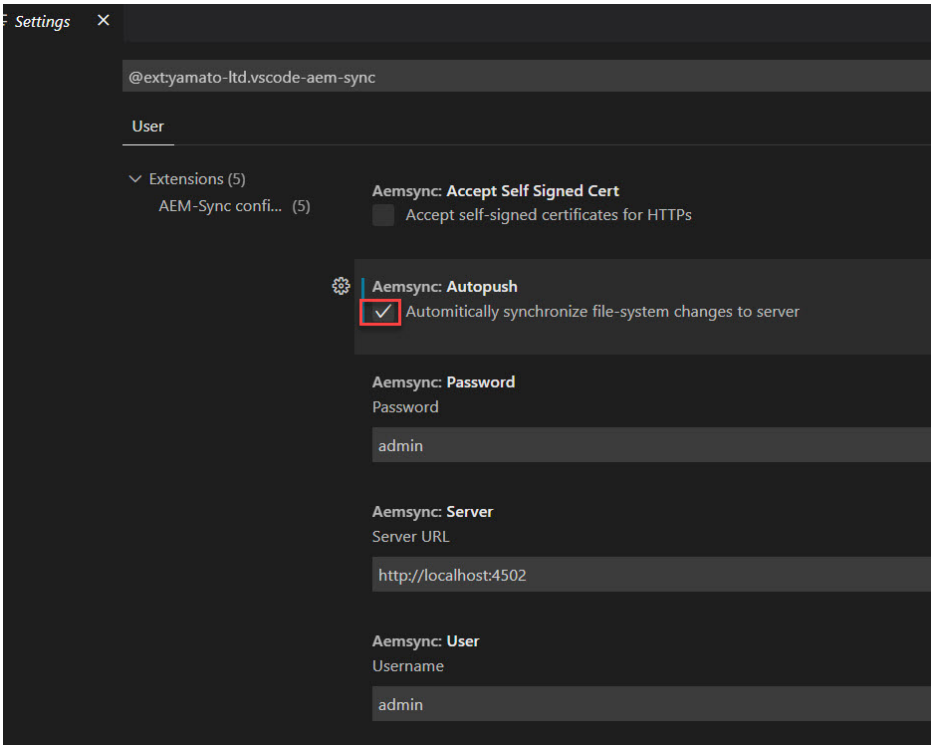


4. Click **Install** to install the extension.
5. In **Visual Studio Code**, select **View** > **Extensions** from the menu bar. The **VSCode AEM Sync Extension** is displayed and enabled in the **EXTENSIONS** area.

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.

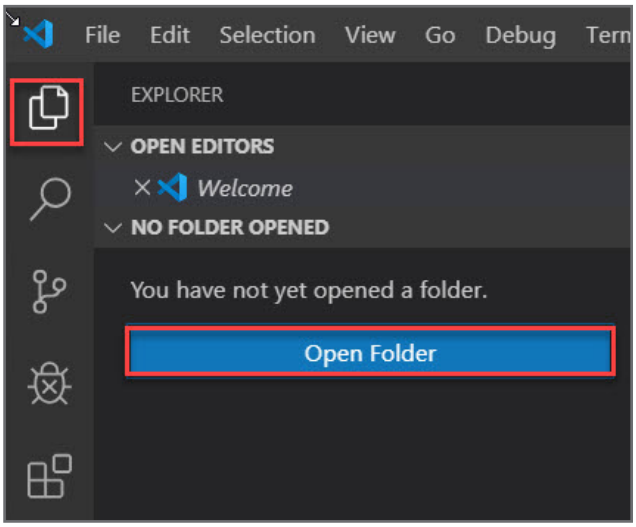
Exercise 3: Import an AEM project

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

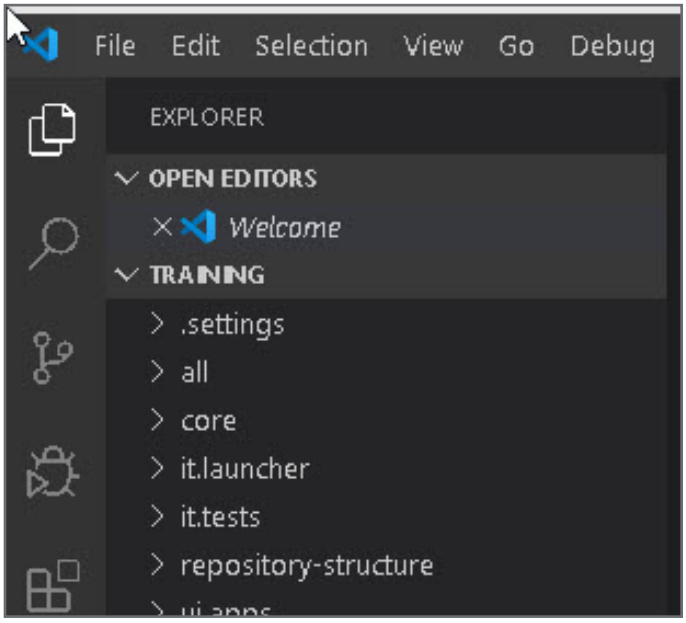
Note: This exercise assumes you have a local Maven project. <AEM Project Directory> represents the location of this local Maven project.

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:

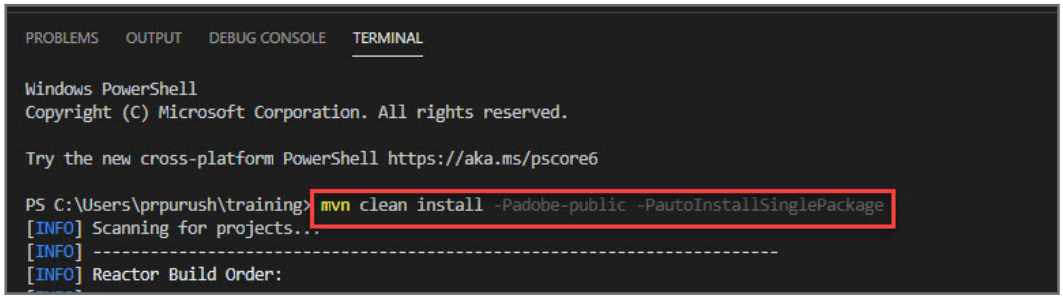


To use the Integrated Terminal:

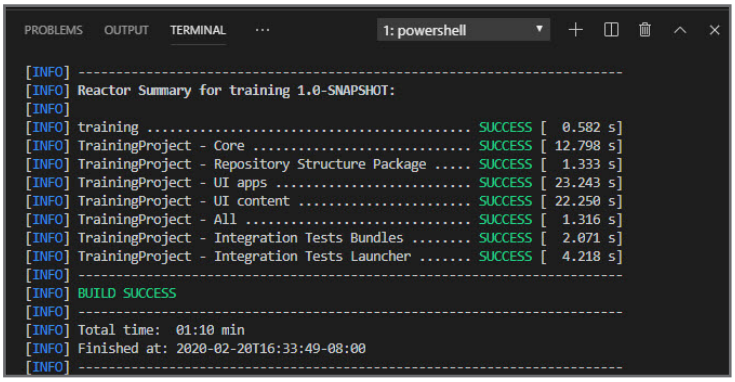
4. Select the **Terminal** tab in the window at the bottom, as shown.

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

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



6. Verify the install is successful, as shown:



Congratulations! You have successfully imported an AEM project.

Exercise 4: Synchronization tools for VSCode

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

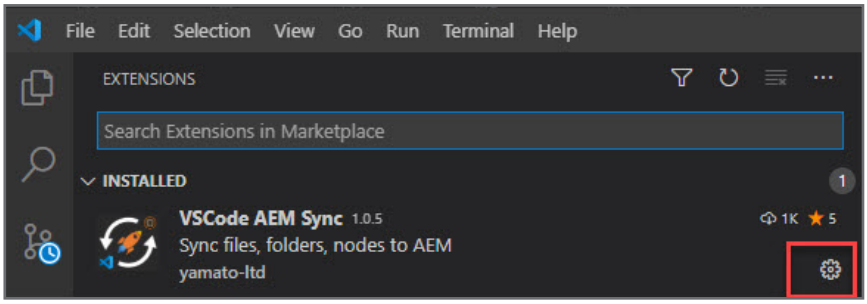
Note: Make sure you have Visual Studio already installed and an AEM author service running locally on port 4502. This exercise already assumes you have a Maven project created from the latest archetype. If you do not have any of these things, refer to earlier sections of this guide.

This exercise consists of the following tasks:

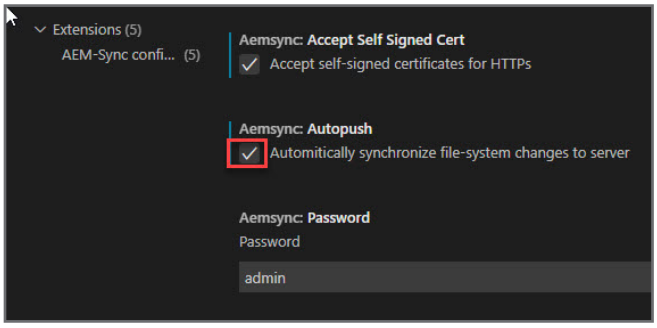
1. Verify auto push is enabled
2. Sync AEM Content

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 exercise.

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.**

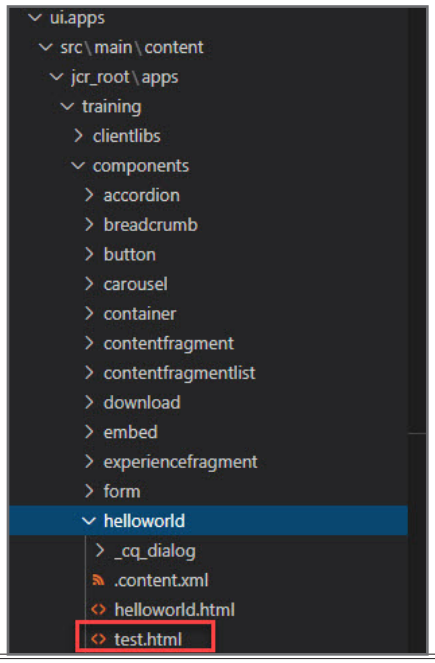
11. Click **OK**. The file is created. Click **Save All** in the upper left to save the changes.

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

Using Brackets for Development

Introduction

Adobe Experience Manager provides Integrated Development Environment (IDE) support for any type of IDEs with Maven support. Brackets is a text editor tool that allows for quick edits to HTML, JS, and CSS files without having to use Maven for synchronization. Developers can ultimately choose their preferred IDE, but Adobe Recommends using the Brackets AEM Extension for front-end developers.

Objectives

After completing this course, you will be able to:

- Explain the features of Brackets
- Install Brackets and the AEM Brackets Extension
- Make changes to the repository by using Brackets

Brackets

Brackets is an open source code editor for HTML, HTL, CSS, and JavaScript developed by Adobe.

The features of Brackets are:

- Live preview
- In-line editors
- Pre-processor support
- Add-on extensions

The plugins available for AEM Developers are:

- AEM Brackets Extension
- Extract for Brackets

You can download the latest version of Brackets from the following link:
http://brackets.io/

Installing the AEM Brackets Extension

You can install the AEM Brackets extension within Brackets by using the Extension Manager. The AEM Brackets extension helps front-end developers build the components with HTML Template Language (HTL).

The AEM Brackets extension provides a smooth workflow to edit AEM components and client libraries and leverages the Brackets code editor to access Photoshop files and layers. The easy synchronization provided by the extension (no Maven or File Vault required) increases developer efficiency and helps front-end developers with limited AEM knowledge to work in projects. This extension also provides some support for the HTL, which makes component development easier and more secure.

The features of AEM Brackets extension are:

- Automated synchronization of changed files to the AEM development Service
- Manual bidirectional synchronization of files and folders
- Full content-package synchronization of the project
- HTL code completion for expressions and data-sly-* block statements

Configuring Your Project

You already learned how to create a package using the Package Manager. You can import the same package to Brackets and begin working on it. If you already have a project package, ensure it has a:

- jcr_root folder
- filter.xml file

To set up your Brackets project based on an existing front-end development project in your repository:

1. Generate a package of your project work in the CRX Package Manager.
2. Extract the contents of the package.
3. Open the jcr_root folder of the package.
4. Configure the project settings to connect to AEM development author Service

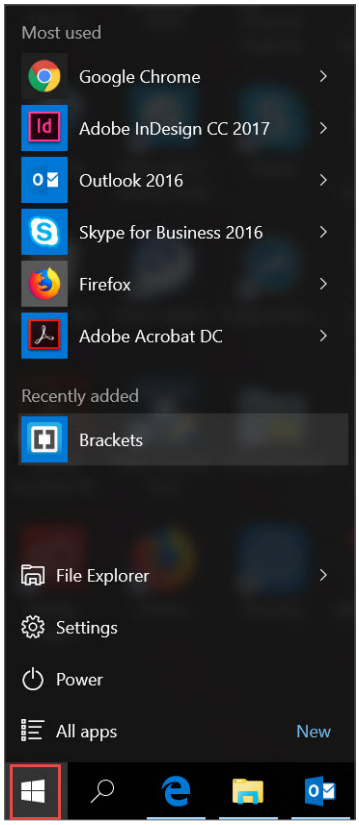
Exercise 1: Install Brackets and the AEM Brackets extension

To install Brackets:

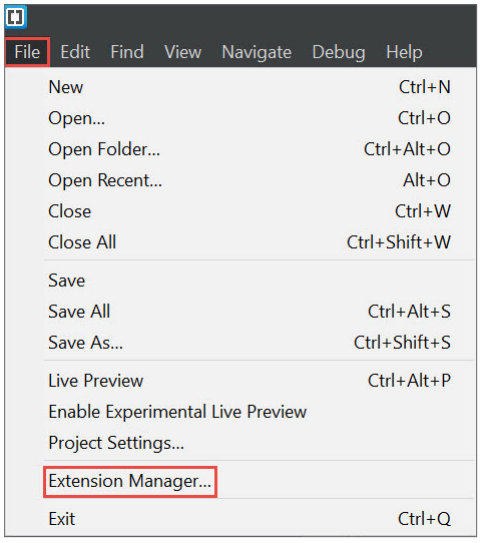
1. Download the latest version of Brackets from: http://brackets.io/. If you do not have network connectivity, you can obtain the standalone installation files from your Instructor.
2. Based on your Operating System (OS), double-click the **Brackets.Release.1.14.msi** or **Brackets.Release.1.14.dmg** installation file. The Brackets Installer dialog box opens.
3. Ensure both checkboxes are selected and click **Next**.
4. Click **Install**. The **Progress** bar appears. The installation may take 3-5 minutes to complete.
5. Click **Finish** to complete the installation.

Note: If you are attending a VILT class using ReadyTech, the installation of Brackets has already been performed for you. You may skip ahead to step #2 of Exercise 2 : Make changes to the repository using Brackets.

6. Click the **Windows** logo on your task bar, click All apps, and then click **Brackets** as shown. The **index.html (Getting Started) - Brackets** page opens.



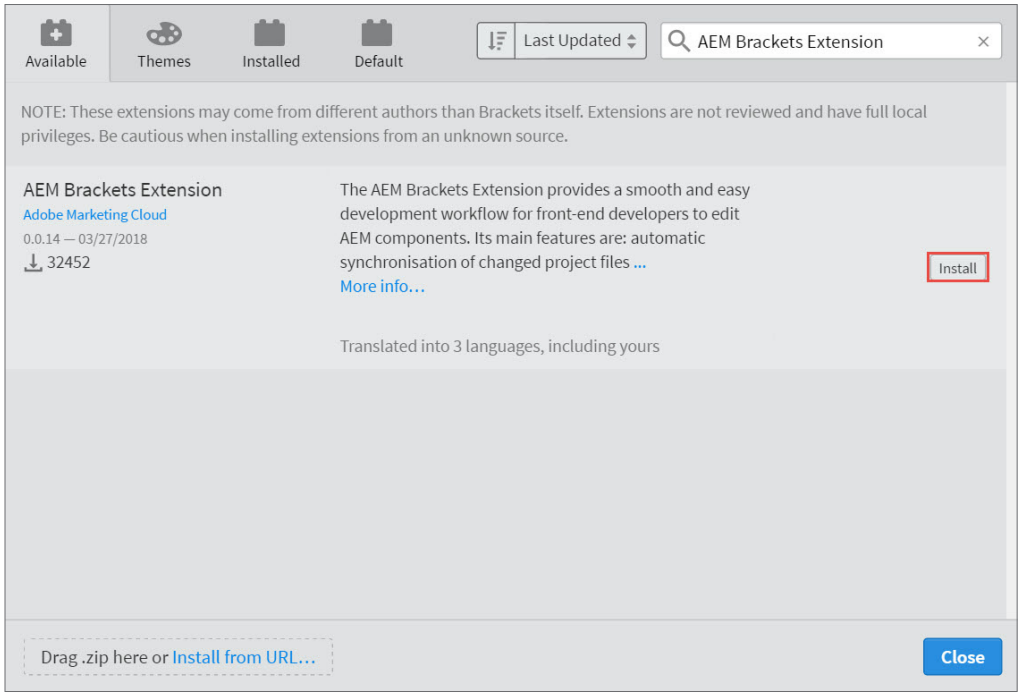
7. Click **File** and select **Extension Manager** from the list, as shown. A new wizard opens.



8. Ensure you are on the **Available** tab.
9. Type **AEM Brackets Extension** in the **Search** field and press **Enter**.

Note: You can also drag the extension zip file (**aem-brackets-extension-master.zip**) from the **Distribution_Files** to the area displayed at the bottom of the pop-up window. Notice, you need to drag the file only if you do not have Internet connectivity.

10. Click **Install** as shown. The **Install Extension** dialog box opens.



11. Wait for the installation to complete, which may take 1-2 minutes. The **Installation Extension** dialog box appears to indicate a successful installation. Click **Close**.
12. Click **Close** in the lower right to go back to the Brackets instance.
13. Click **X** in the upper-right corner to close the Brackets instance.

Exercise 2: Make changes to the repository using Brackets

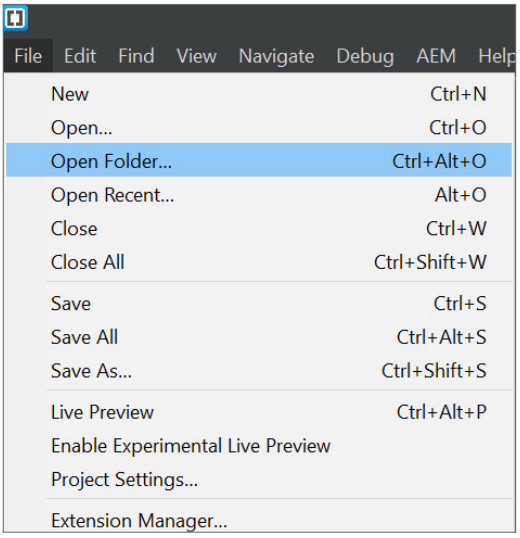
In this exercise, you will edit the contents of your project in Brackets. After completing this task, you can update the package content and share it with team members so they also see the same content in their AEM Services.

You will import the code content package into Brackets to observe how your repository can be configured in Brackets, and then examine the changes propagated to the repository in one of your training pages.

1. Verify your AEM author service is running.

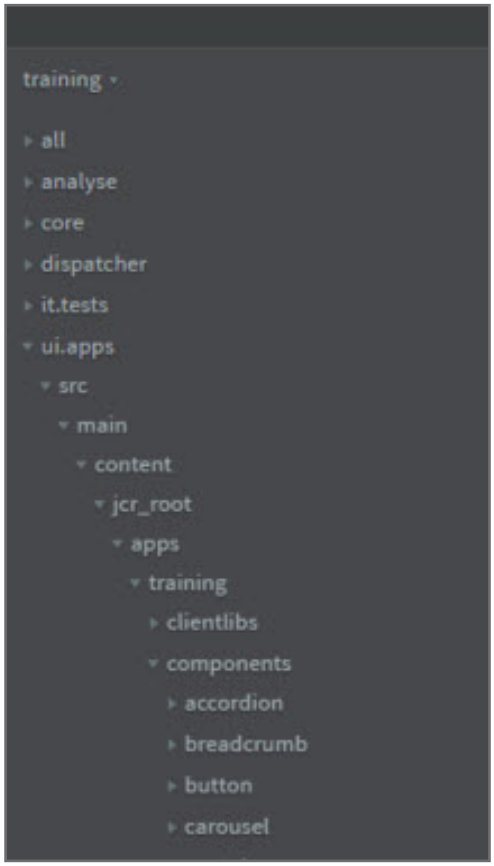
IMPORTANT! You must have an AEM author service running to make changes to the repository that then get saved to AEM.

2. Open the Brackets instance.
3. Click **File** and select **Open Folder** from the list, as shown. The **Choose a folder** dialog box opens.

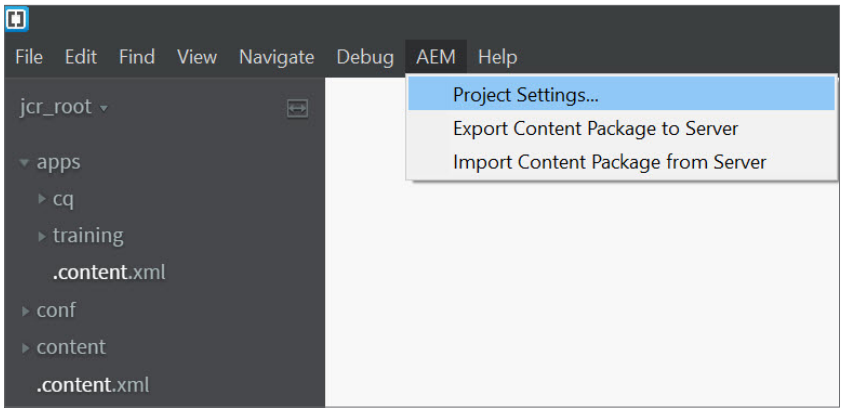


4. Locate the **<AEM project>** folder, and click **Select Folder** (or click **Open** on a Mac system). The contents of the folder are listed in the left panel.

5. Select **ui.apps\src\main\content\jcr_root** and then expand the **apps** folder, as shown:



6. Click **AEM** and select **Project Settings** from the list as shown. The **Project Settings** dialog box opens.



7. Provide the following details in their corresponding fields:

Field	Value
ServerURL	http://localhost:4502
Username	admin
Password	admin

8. Select the **Automatically synchronize file-system changes to server** checkbox and click **OK**

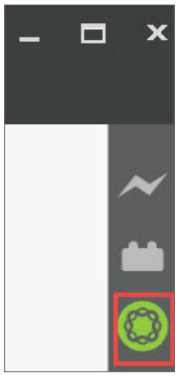
These details help Brackets to interact with the AEM author development Service.

9. In the left panel, expand the apps node and navigate to **/apps/wknd/components/helloworld/helloworld.html**. The file opens on the right panel.

10. Add a message to the file, as shown:

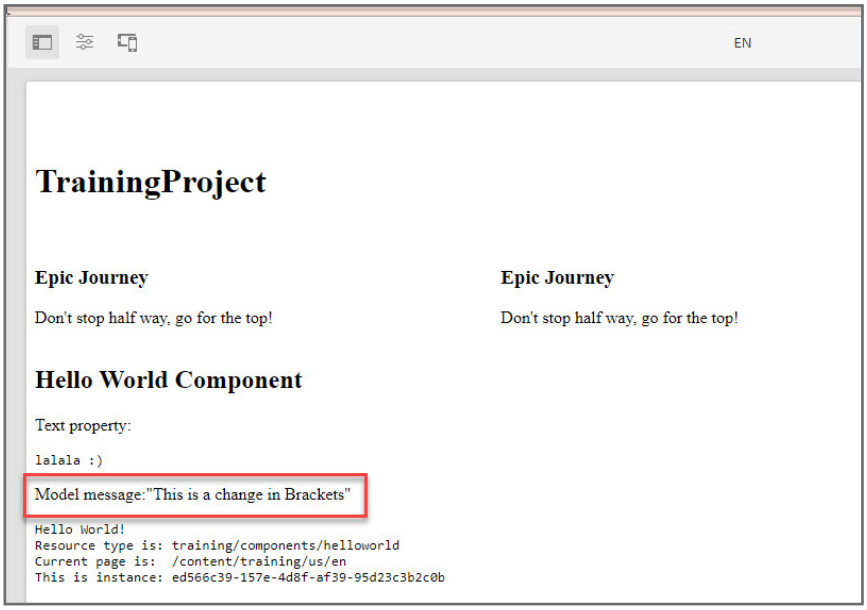
```
    distributed under the License is distributed on an "AS IS" BASIS,
    WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
    See the License for the specific language governing permissions and
    limitations under the License.
*/--><
<div class="cmp-helloworld" data-cmp-is="helloworld">
    <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="${model.message}">
        <p class="cmp-helloworld__item-label">Model message:"This is a change in Brackets"</p>
        <pre class="cmp-helloworld__item-output" data-cmp-hook-helloworld="model">${model.message}</pre>
    </div>
</div>
```

11. Press **Ctrl+S** (Windows)/**Cmd+S** (Mac) or click **File** and select **Save** from the list. Notice the green AEM logo as shown. This confirms that the files are synchronized with AEM successfully.



To test the synchronization, you need to open a training page that is using the page component you just modified:

12. Navigate to the tab where the AEM author service is running.
13. Open a new tab in your browser, type http://localhost:4502/editor.html/content/training/us/en.html in the address bar, and then press **Enter**. The **English** page opens.
14. Notice the text that you added in **helloworld.html** now appears on the page, as shown:



You have successfully synched the files.

Front-End Development using aemfed

Introduction

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

Objectives

After completing this course, you will be able to:

- Install aemfed
- Explain development using aemfed

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.

Exercise 1: Install aemfed (Local only)

In this exercise, you will install aemfed.

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

Note: You can skip this exercise 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.

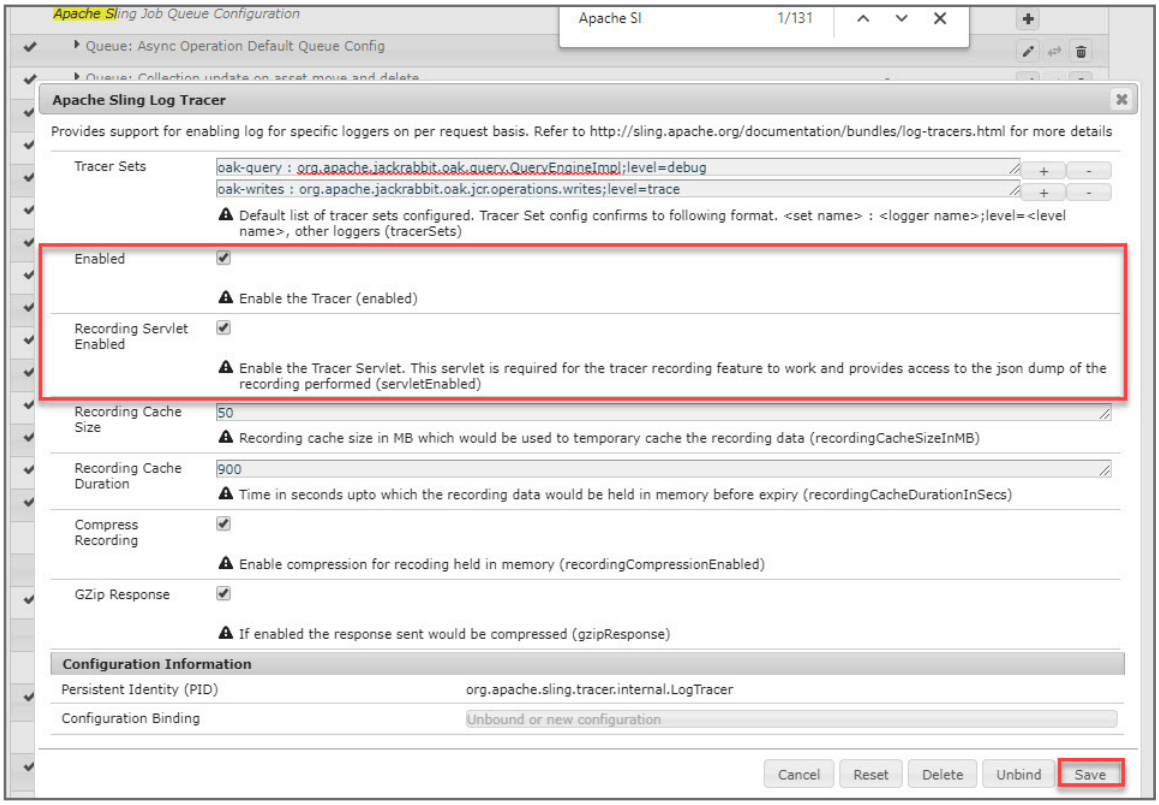
Exercise 2: Configure and run aemfed

In this exercise 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**



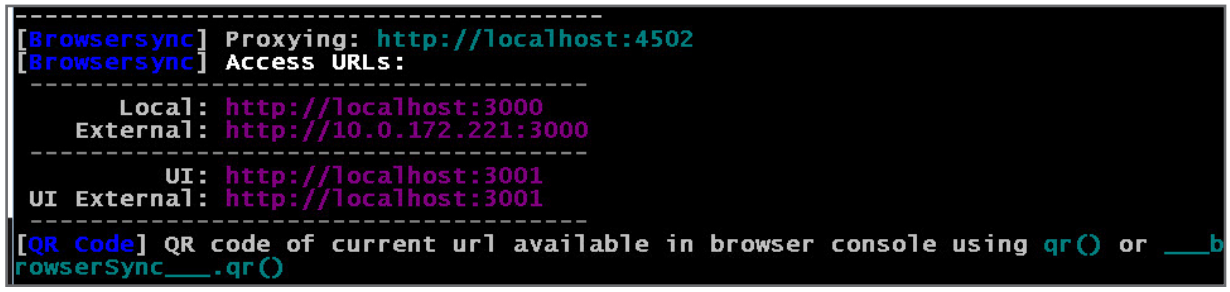
Task 2: Start aemfed

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.

Exercise 3: Install content locally (Optional)

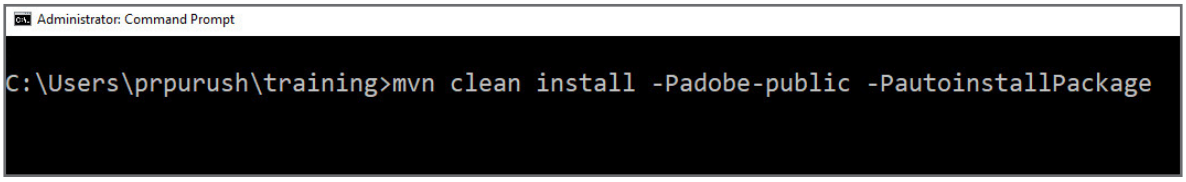
In this exercise, you will install the local Maven project into AEM.

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

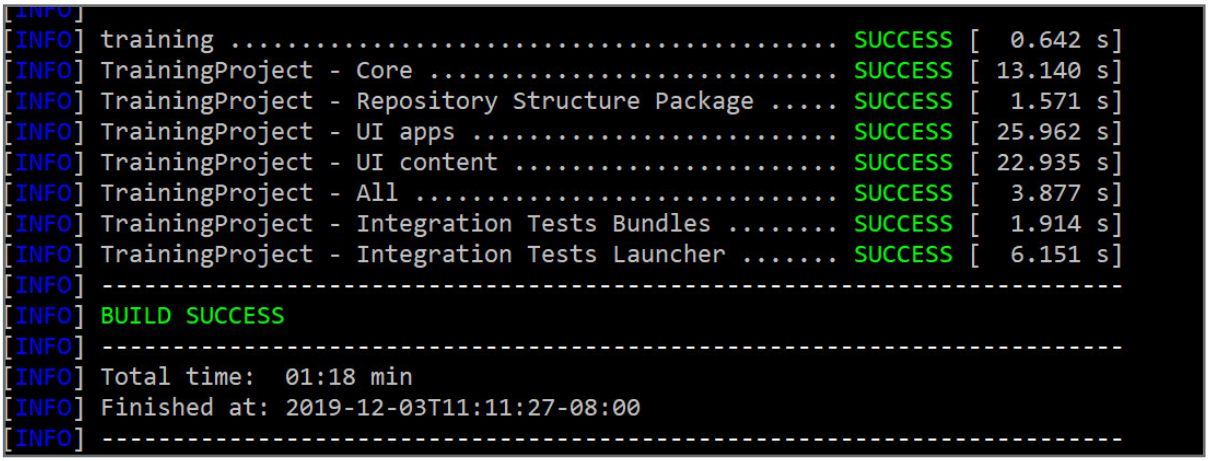
Note: You can skip this exercise if you have already installed your AEM project.

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
```



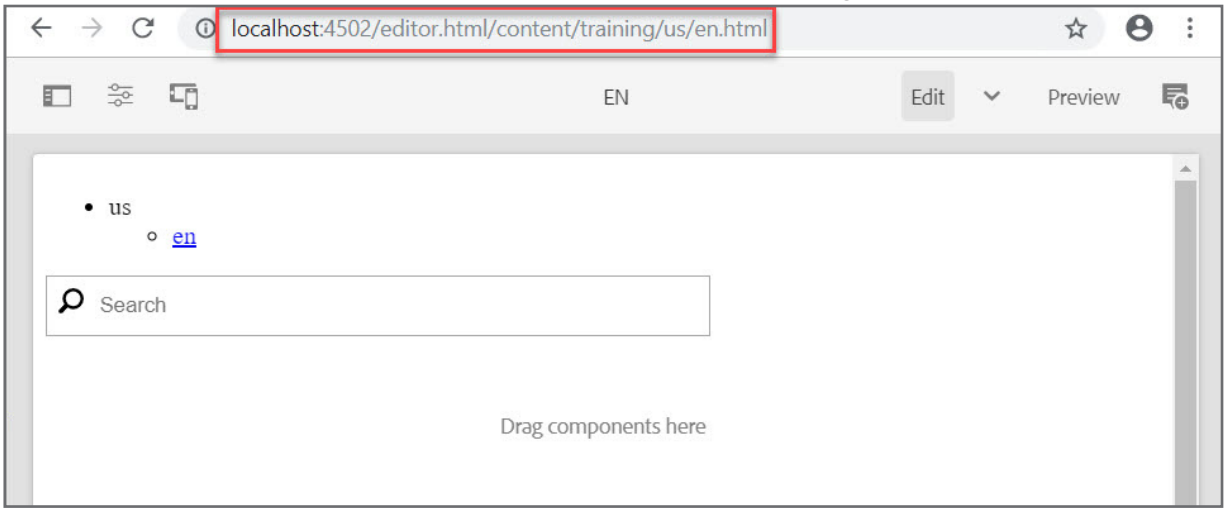
3. Verify the install is successful, as shown:



Exercise 4: Add a component style using aemfed

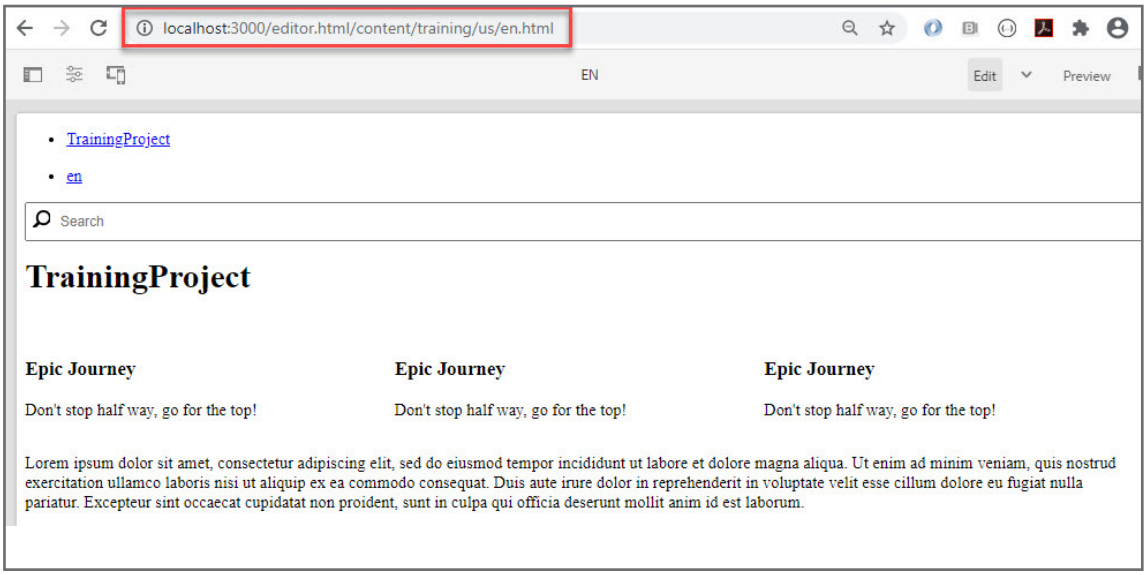
aemfed connects a local AEM server to a content package. In this exercise we will use a project created from the archetype, but you could connect with any unzipped content package.

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.



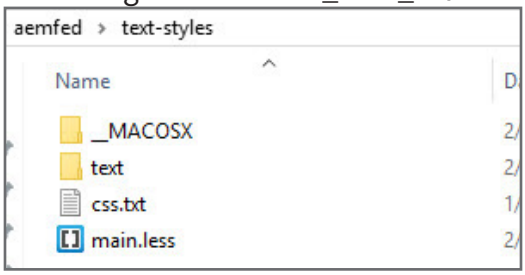
2. Drag a text component to the **en** Page and add some dummy text.

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).

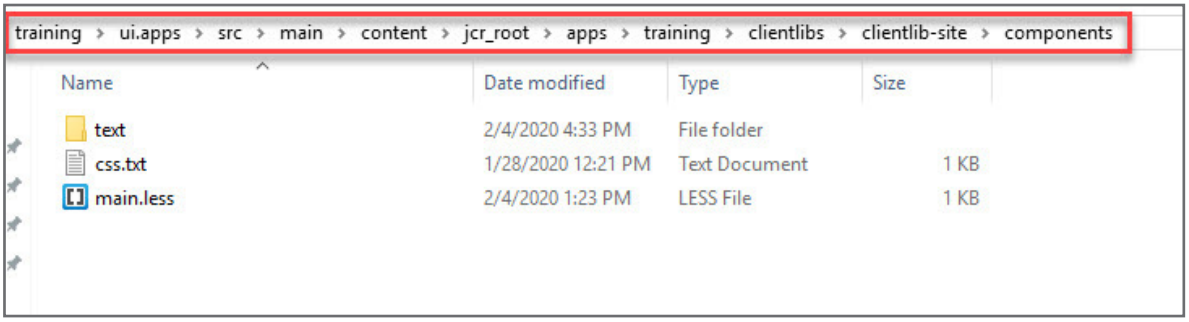
4. Navigate to **Exercise_Files_TB/aemfed/** and extract the contents of **text-styles.zip,** as shown:



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 **Exercise_Files_TB** > **aemfed** into the **clientlib-site/components** folder (create **components** folder if needed):

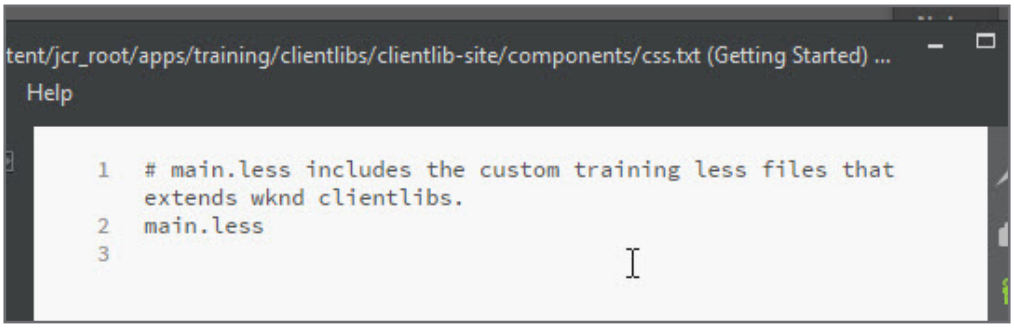


8. Verify you see messages in the aemfed terminal window similar to the following screenshot:

```
[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.

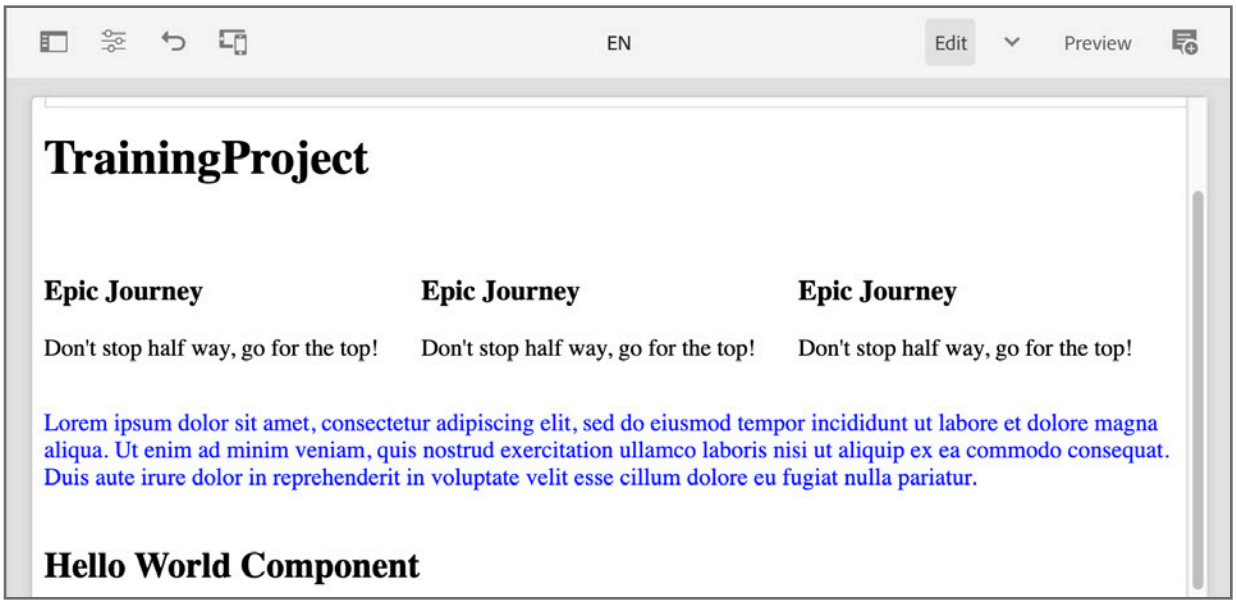
10. Using the IDE of your choice, navigate into the **/clientlib-site/** folder and open **css.txt** and add the line **main.less.**



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:**

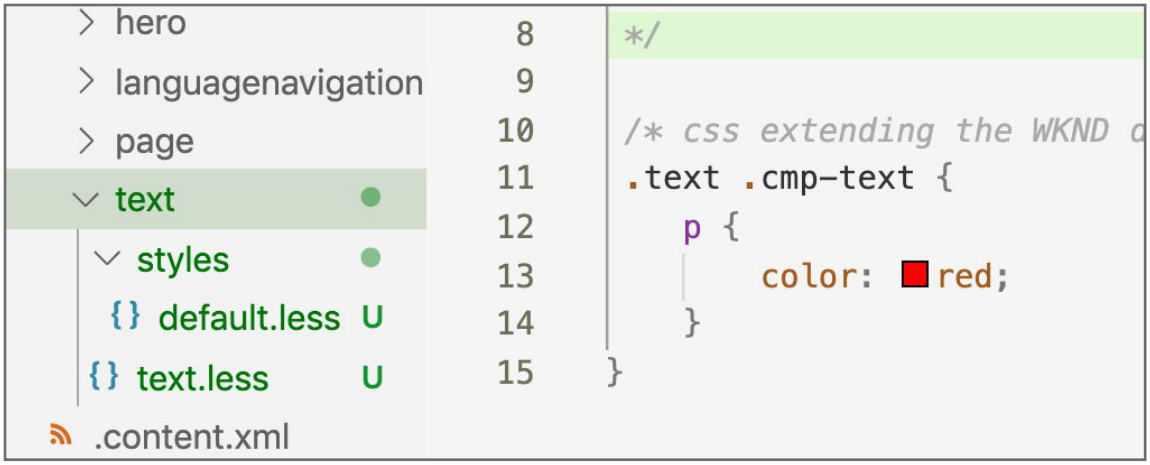
```
@import "components/text/text.less";
```

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.

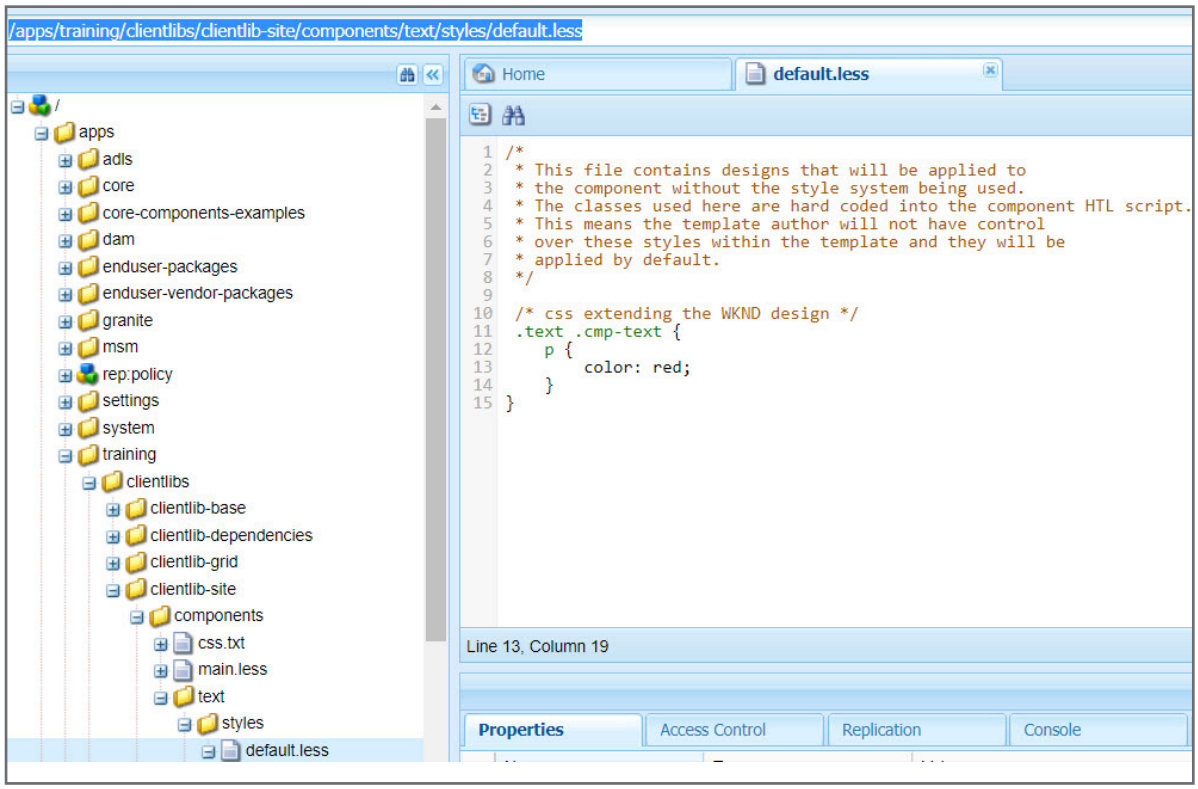
16. Using the IDE of your choice, navigate to **clientlib-site/components/text/styles** and open **default.less**. Update the style to a color of your choice, such as **red**.



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 quite a while.

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/

Cloud Manager Basics

Introduction

Cloud Manager for Adobe Experience Manager allows customers to build, test and deploy AEM applications hosted by Adobe. Cloud Manager enables customers to manage their custom code deployments on their AEM-managed cloud environments with manageable pipeline automation and complete flexibility for their deployments timing or frequency.

Objectives

After completing this course, you will be able to:

- Describe Cloud Manager
- Define basic terminology used in Cloud Manager
- Set up Program
- Define Pipeline with existing repository
- Explain custom code quality rules

Cloud Manager

Cloud Manager enables organizations to self-manage AEM applications. It includes the integration and continuous delivery (CI/CD = Continuous Integration/Continuous Deployment) framework which expedites the delivery of customizations or updates without compromising performance or security.
The following are the key features of Cloud Manager:

- Self service Interface

  The User Interface (UI) for Cloud Manager enables customers to easily access and manage the cloud environment and CI/CD pipeline for their Experience Manager applications.

  Customers define application-specific Key Performance Indicators (KPIs) - peak page views per minute and expected response time for a page load, that ultimately form the basis for measuring a successful deployment. Roles and permissions for different team members can be easily defined.

- CI/CD Pipeline

  CI/CD pipeline helps to speed up the delivery of custom code or updates such as adding new components on the website.

  Through the Cloud Manager UI, customers can configure and kick off their CI/CD pipeline. During this pipeline, a thorough code scan is executed to ensure that only high-quality applications pass through to the production environment.

- Flexible Deployments

  Cloud Manager offers customers flexible and configurable deployment modes so they can deliver experiences according to changing business demands.

  The code is automatically deployed to an environment based on specific events such as code commit. You can also schedule code deployments during specified time frames, even outside business hours.

- Extension and Automation through Adobe IO

  Adobe IO is a single-entry point for communication with all Adobe solutions that's easy, secure, and extensible. You can access this entry point via an HTTP API and command line tool. Using Adobe IO, you can automate pipeline executions, deletions, approvals, and other pipeline tasks. You can also manage environments, access logs, and the developer console as well. Adobe IO also offers an eventing system that allows you to send events to other applications to notify about pipelines and environment tasks that have completed..

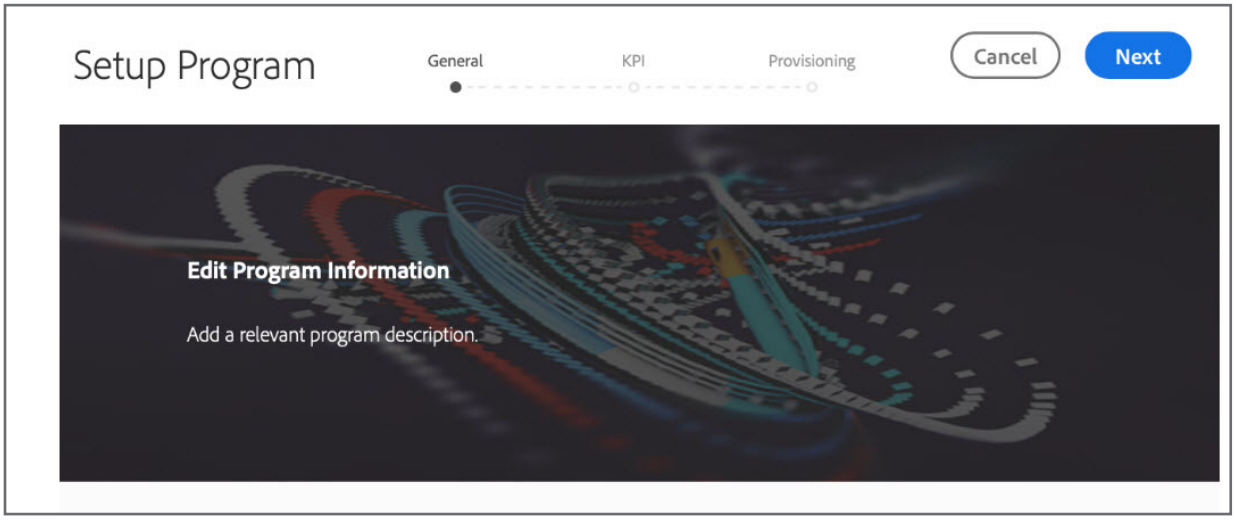
Setup a Program

A program is a set of environments that support a logical grouping of customer initiatives, usually corresponding to a purchased Service Level Agreements (SLA). Each program has exactly one production environment and may have many non-production environments.

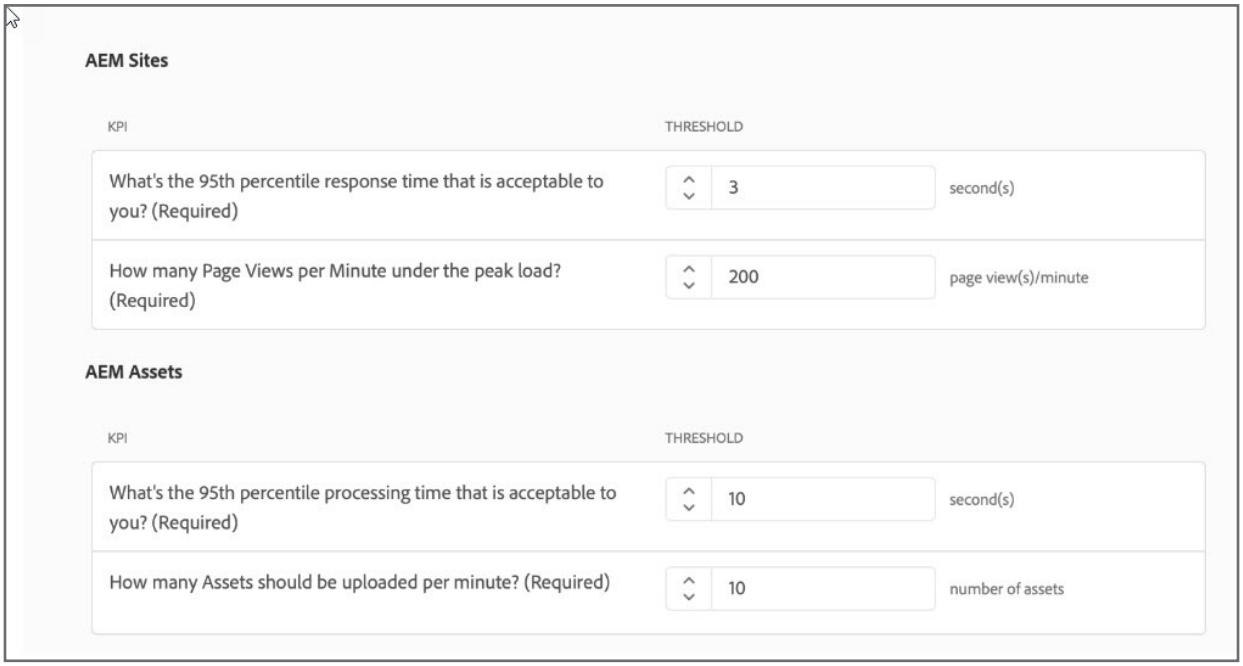
A Program is a high-level entity that contains production and non-production environments which have author, publish and dispatcher services. You can also have program-level configurations, which correspond to your SLA and APIs.

Setting up a Program involves setting the program description and defining the Key Performance Indicators (KPIs) that will be used for performance testing. Optionally, a thumbnail can be uploaded. Additionally, the business owner can configure environment provisioning while setting up the program. The KPIs defined serves as a baseline for performance testing which is passed each time the pipeline executes.

There are three options when you click Setup Program. They are General, KPI and Provisioning. On the General tab, you can upload a thumbnail and add a description to your program.



Under KPI, you can define your two KPIs. Separate KPIs are defined for AEM Sites and AEM Assets, respectively. You will be able to specify the KPIs for the products you have licensed.



Under Provisioning, you can view or edit the provisioning configuration for production and non-production environments in your program. If autoscaling has been turned on for the program, you will see Autoscale is on.

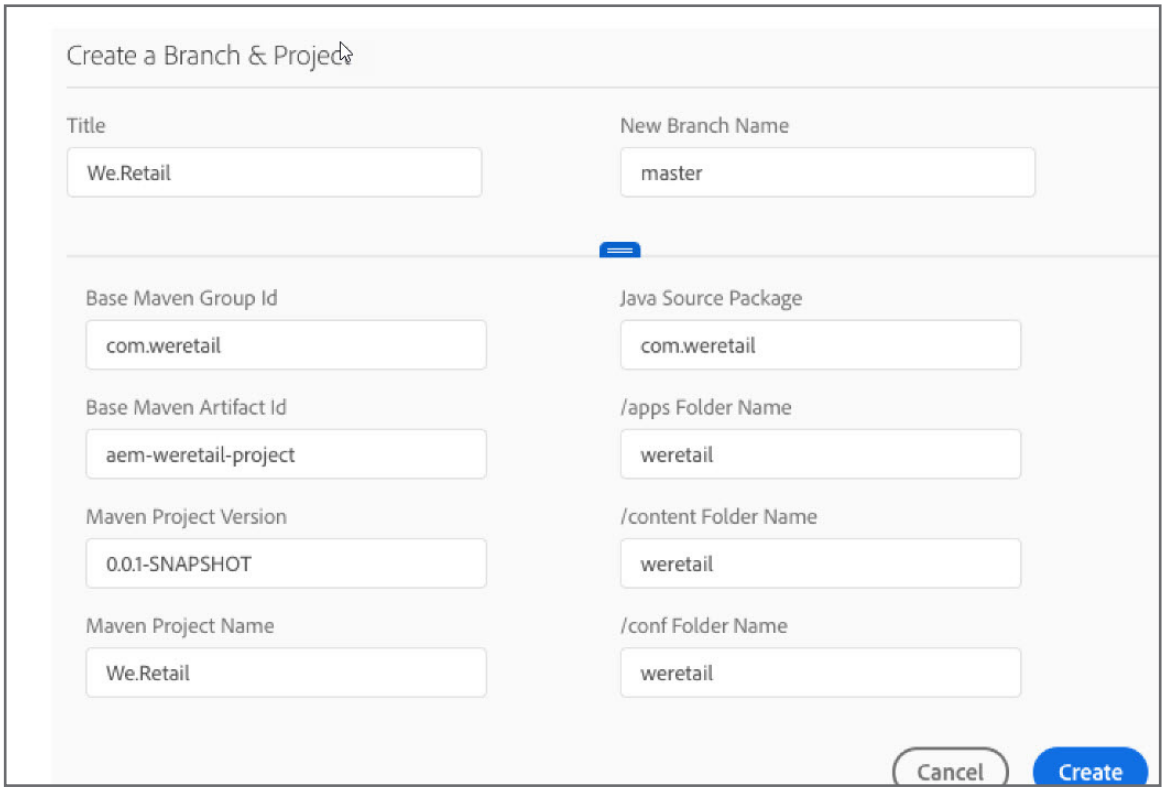
You can also edit the program once the initial program has already been set up.

Creating an AEM Application

Cloud Manger is able to create a minimal AEM project as a starting point for new customers. This process is based on the AEM Project Archetype .
Following are the steps to create an AEM application project:

1. Login to **Cloud Manager**. Once the basic program setup is complete, a special call to action card is shown:

2. Click **Create** to navigate to the Pipeline Setup screen.
3. Click **Create** and in the dialog box, provide the parameters required by the AEM Project Archetype.



4. Click **Create** in the preceding step to create the starter project by using the archetype and commit to the named git branch. Once this is done, you can set up the pipeline.

Setting up Project

In order to be built and deployed successfully with Cloud Manager, existing AEM projects need to adhere to some basic rules:

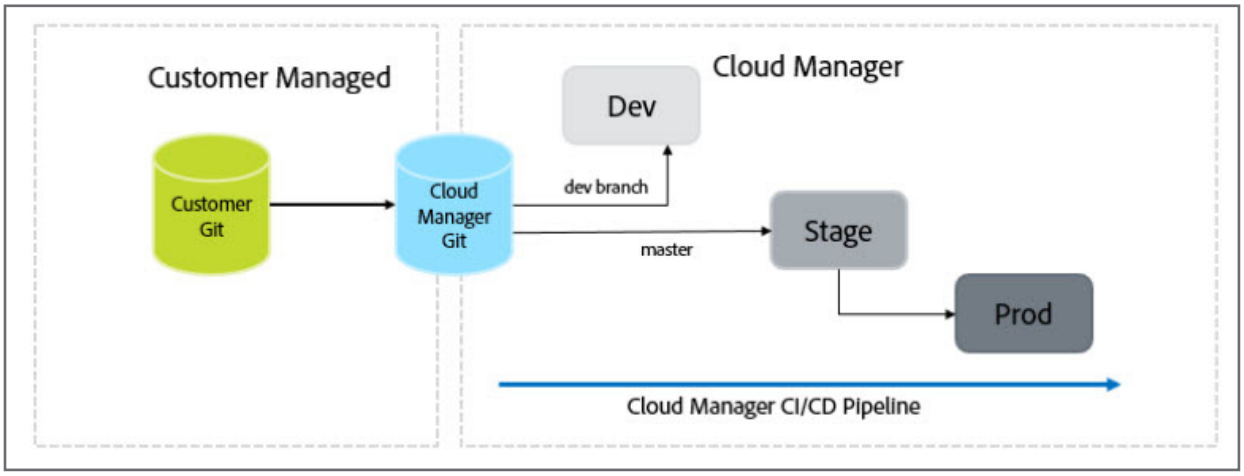
- Projects must be built using Apache Maven.
- There must be a pom.xml file in the root of the Git repository. This pom.xml file can refer to as many submodules (which in turn may have other submodules, and so forth) as necessary.
- You can add references to additional Maven artifact repositories in your pom.xml files. However, access to password-protected or network-protected artifact repositories is not supported.
- Deployable content packages are discovered by scanning for content package zip files which are contained in a directory named target . Any number of submodules may produce content packages.
- Deployable Dispatcher artifacts are discovered by scanning for zip files (again, contained in a directory named target) which have directories named conf and conf.d .
- If there is more than one content package, the ordering of package deployments is not guaranteed. Should a specific order be needed, content package dependencies can be used to define the order.

Building Environment Details

Cloud Manager builds and tests your code using a specialized build environment. The environment must be built with some special attributes.
To learn more about the attributes, click the link Cloud Manager under the References section.

Managing your code

Adobe Cloud Manager comes provisioned with a single Git repository that is used to deploy code using Cloud Manager's CI/CD pipelines. Customers can use the Cloud Manager's Git repository out of the box. Customers also have the option of integrating an on-premise or customer-managed Git repository with Cloud Manager.



Cloud Manager Repository:

AEM Managed Services subscription will include a source code repository provisioned and managed by Adobe. Each customer's program is assigned a single and unique Git Repository , where your associated code will be stored and secured.

This git repository is directly tied to exactly 1 program. You cannot connect multiple git repositories to a single program. If a custom needs to have multiple repositories, they might consider refactoring their code to a multi-project git repo or working with Adobe to gain access to more programs.

Customer Managed Repository

It is recommended that customers maintain their own separate git repository outside of Cloud Manager. Although this is not required and you can use Cloud Manager git as your primary source control, Cloud Manager git is limited to a command line based HTTPS connection. There is no UI or other project management tools associated with it. You can use Git's supported feature for multiple remote repositories. Day-to-day development would continue to happen in your Git Repository. When a release branch is ready for a deployment to production, you will push your latest code to the Cloud Manager's Git repository and trigger the Cloud Manager CI/CD pipeline.

Integrating Git Repository with Cloud Manager Repository

Adobe Cloud Manager comes provisioned with a single Git repository that is used to deploy code using Cloud Manager's CI/CD pipelines. Customers can use the Cloud Manager's git repository out-of-the-box. Customers also have the option of integrating an on-premise or customer-managed Git repository with Cloud Manager.

CI/CD Pipeline

Cloud Manager includes a Continuous Integration (CI) and Continuous Delivery (CD) framework which allows implementation teams to quickly test and deliver new or updated code. For example, implementation teams can set up, configure, and start an automated CI/CD pipeline that leverages Adobe coding best practices to perform a thorough code scan and ensure the highest code quality.

The CI/CD pipeline also automates unit and performance testing processes to increase deployment efficiency and proactively identify critical issues that are expensive to fix after deployment. Implementation teams can access a comprehensive code performance report to gain visibility into potential impact on KPIs and critical security validations if the code gets deployed to production.

CI/CD Production Pipeline:

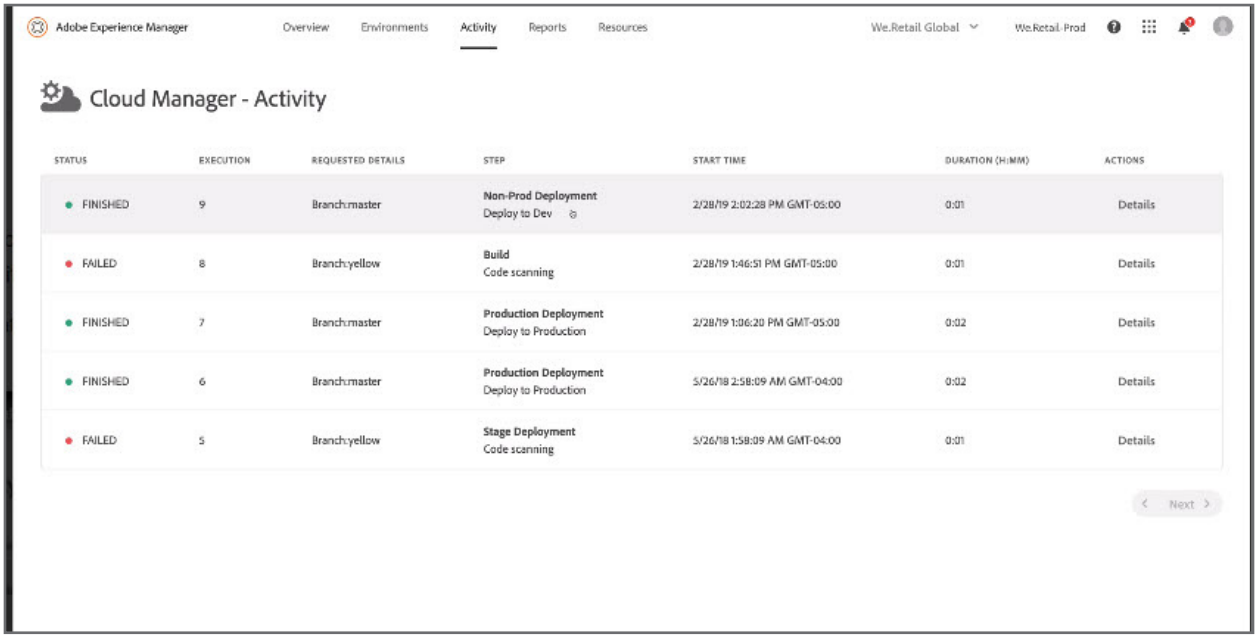
The CI/CD Production Pipeline configuration defines the trigger that will initiate the pipeline, parameters controlling the production deployment and performance test parameters. The CI/CD Production Pipeline is used to build and deploy code through Stage to the Production environment, decreasing time to value.

CI/CD Non-Production Pipeline:

CI/CD Non-production pipelines are broken into two categories, Code Quality pipelines, and Deployment pipelines. Code Quality pipelines all code from a Git branch to build and be evaluated against Cloud Manager's code quality scan. Deployment pipelines support the automated deployment of code from the Git repository to any non-production environment, meaning any provisioned AEM environment that is not Stage or Production.

Cloud Manager Activity

Cloud Manager provides a consolidated view into a Program's activity, listing all CI/CD Pipeline executions, both production and non-production, allowing visibility into the past and present activity, and any activity's Details can be reviewed.



Pipeline configuration in Cloud Manager

Pipeline is configured using the Pipeline Settings tile in the Cloud Manager UI. This is done by the deployment manager. First, you will select a branch from the Git Repository.
The following are the steps involved in Pipeline configuration:

- Define the trigger that will start the pipeline.
- Define the parameters controlling the production deployment.
- Configure the performance test parameters.

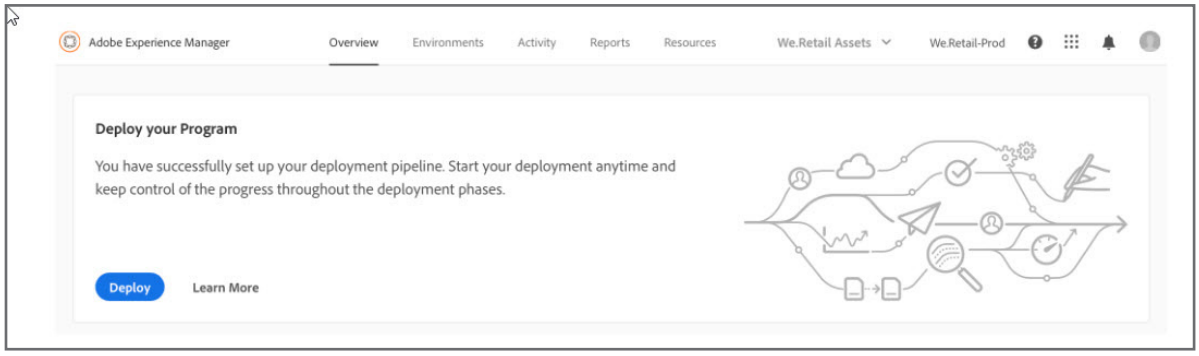
Once you configure your CI/CD Pipeline, you can deploy your code.

The Stage Deployment , involves the following steps:

1. Build & Unit Testing
2. Code Scanning
3. Deploy to Stage

The Production Deployment , involves the following steps:

1. Application for Approval (if enabled)
2. Schedule Production Deployment (if enabled)
3. CSE Support (if enabled)
4. Deploy to Production



Logs through UI

From the Environment card or the Environments screen, users will be able to access a list of available log files for the selected environment. You can download these files through the UI, either from the Overview page or the Environments page.

Logs through API

In addition to downloading logs through the UI, logs will be available through the API and the Command Line Interface. For example, to download the log files for a specific environment, you need to use the following command:

```
$ aio cloudmanager:download-logs --programId 5 <environment Id> author
   aemerrorn
```

To tail logs:

```
$ aio cloudmanager:tail-log --programId 5 <environment Id> author
   aemerror
```

In order to obtain the environment ID and the available service / log name options you can use:

```
$ aio cloudmanager:list-environments
```

SonarQube

SonarQube is a web-based open source platform used to measure and analyze the source code quality. It is written in java but can analyze and manage code of more than 20 programming languages including c/c++, PL/SQL, HTML etc. SonarQube offers reports on duplicated code, coding standards, unit tests, code coverage, code complexity, comments, bugs, and security vulnerabilities.

In SonarQube, analyzers contribute rules which are executed on source code to generate issues. There are four types of rules:

- Code Smell (Maintainability domain)
- Bug (Reliability domain)
- Vulnerability (Security domain)
- Security Hotspot (Security domain)

For Code Smells and Bugs, zero false-positives are expected. At least this result is the target so that developers do not have to wonder if a fix is required.

For Vulnerabilities, the target is to have more than 80% of issues be true-positives.
Security Hotspot rules draw attention to code that is security-sensitive. It is expected that more than 80% of the issues will be quickly resolved as "Reviewed" after review by a developer.
The Rules page is the entry point where you can discover all the existing rules or create new ones based on provided templates.

Custom Code Quality rules for Adobe Cloud Manager

In the Adobe Cloud Manager CICD process, when you move your code into the cloud it runs through standard SonarQube rules and beyond these rules. There are also custom code quality rules for AEM, as shown:

	Code Quality Rules Downloaded from	https://docs.adobe.com/content/help/en/experience-manager-cloud-manager/using/how-to-use/understand-your-test-results.html			
2	Rule Key	Description	Type	Severity	Tags
3	squid:S2068	Credentials should not be hard-coded	Vulnerability	Blocker	cert,cwe,owas
4	squid:S2258	"javax.crypto.NullCipher" should not be used for anything other than testing	Vulnerability	Blocker	cwe,owasp-a6
5	squid:S2278	Neither DES (Data Encryption Standard) nor DESede (3DES) should be used	Vulnerability	Blocker	cert,cwe,owas
6	squid:S3369	Security constraints should be defined	Vulnerability	Blocker	cwe,jee,owas
7	squid:S3374	Struts validation forms should have unique names	Vulnerability	Blocker	cwe,struts
8	squid:S2070	SHA-1 and Message-Digest hash algorithms should not be used	Vulnerability	Critical	cwe,owasp-a6
9	squid:S2076	Values passed to OS commands should be sanitized	Vulnerability	Critical	cwe,owasp-a1
10	squid:S2077	SQL binding mechanisms should be used	Vulnerability	Critical	cert,cwe,hibe
11	squid:S2078	Values passed to LDAP queries should be sanitized	Vulnerability	Critical	cert,cwe,owas
12	squid:S2089	HTTP referers should not be relied on	Vulnerability	Critical	cwe,owasp-a2
13	squid:S2245	Pseudorandom number generators (PRNGs) should not be used in secure contexts	Vulnerability	Critical	cert,cwe,owas
14	squid:S2254	"HttpServletRequest.getRequestedSessionId()" should not be used	Vulnerability	Critical	cwe,owasp-a2
15	squid:S2257	Only standard cryptographic algorithms should be used	Vulnerability	Critical	cwe,owasp-a3
16	squid:S2277	Cryptographic RSA algorithms should always incorporate OAEP (Optimal Asymmetric Encryption Padding)	Vulnerability	Critical	cwe,owasp-a3
17	squid:S2653	Web applications should not have a "main" method	Vulnerability	Critical	cert,cwe,jee
18	squid:S2658	Classes should not be loaded dynamically	Vulnerability	Critical	cwe,owasp-a1
19	squid:S2976	"File.createTempFile" should not be used to create a directory	Vulnerability	Critical	owasp-a9
20	squid:S3355	Defined filters should be used	Vulnerability	Critical	injection,owa
21	CQRules:CWE-134	Don't use format strings that may be externally-controlled	Vulnerability	Major	cqsecurity
22	CQRules:CWE-676	Use of Potentially Dangerous Function	Vulnerability	Major	cqsecurity
23	findbugs:PT_ABSOLUTE_PATH_TRAV	Security - Absolute path traversal in servlet	Vulnerability	Major	cwe
24	findbugs:PT_RELATIVE_PATH_TRAVE	Security - Relative path traversal in servlet	Vulnerability	Major	cwe

Example of non-compliant vs compliant code:

Non-Compliant Code

```
public void dontDoThis() {
  try {
    someMethodThrowingAnException();
  } catch (Exception e) {
    logger.debug(e.getMessage(), e);
  }
}
```

Compliant Code

```
public void doThis() {
  try {
    someMethodThrowingAnException();
  } catch (Exception e) {
    logger.error("Unable to do something", e);
  }
}
```

Exercise 1: Observe the custom code quality rules

In this exercise, you will observe the custom code quality rules output for a non-compliant Java file. You will then observe the code quality rules created for code, which is fully compliant with the code rules.

1. In Explorer, navigate to **/Exercise_Files_TB/Cloud_Manager/** and open the file **old-AutoAssignACL.java** using a text editor.

2. Examine the **old-AutoAssignACL.java** class:

```
1.  package com.adobe.training.core;
2.
3.  import javax.jcr.security.AccessControlList;
4.  import javax.jcr.security.AccessControlManager;
5.  import javax.jcr.security.Privilege;
6.  import javax.jcr.Session;
7.
8.  import org.osgi.service.component.annotations.Component;
9.
10. import org.apache.jackrabbit.api.security.user.Authorizable;
11. import org.apache.jackrabbit.api.security.user.UserManager;
12. import org.apache.jackrabbit.commons.jackrabbit.authorization.AccessControlUtils;
13. import org.apache.sling.jcr.base.util.AccessControlUtil;
14. import org.osgi.framework.Constants;
15. import org.slf4j.Logger;
16. import org.slf4j.LoggerFactory;
17.
18. import com.adobe.granite.workflow.WorkflowException;
19. import com.adobe.granite.workflow.WorkflowSession;
20. import com.adobe.granite.workflow.exec.WorkItem;
21. import com.adobe.granite.workflow.exec.WorkflowProcess;
22. import com.adobe.granite.workflow.metadata.MetaDataMap;
23. import com.adobe.granite.workflow.exec.WorkflowData;
24.
25. /**
26.  * This is a project workflow process step. When starting a workflow from a project, you can specify the:
27.  *
28.  * Payload: The page that will have ‹modify› permissions added/removed from its ACL
29.  * User: The user attached to the ACL
30.  *
31.  * This process will add/remove jcr privileges that correlate to ‹Modify, Create, Delete› in the /useradmin
32.  * for the given user on the payload. An argument is used to specify to add or remove ‹modify› permissions.
33.  * The argument must follow permission=<add || remove>
34.  */
35. @Component(service = WorkflowProcess.class,
36.         property = {Constants.SERVICE_DESCRIPTION + "=A sample workflow process implementation.",
37.                 Constants.SERVICE_VENDOR + "=Adobe Digital Learning Services",
38.                 "process.label=Auto Assign Edit Permissions"
39.         })
40.
41. public class AutoAssignACL implements WorkflowProcess{
42.
43.     private final Logger logger = LoggerFactory.getLogger(getClass());
44.
45.     /**
46.      * @param item - Holds the payload and user
47.      * @param workflowSession - Session with service user (workflow-process-service) that will be modifying the permissions
48.      * @param workflowArgs - permission=add will add ‹modify› permissions. permission=remove will remove ‹modify› permissions
```

```
49.      */
50.     @Override
51.   ute(WorkItem item, WorkflowSession workflowSession, MetaDataMap workflowArgs) throws WorkflowException {
52.
53.         WorkflowData workflowData = item.getWorkflowData(); //get the workflow properties
54.         String userID = workflowData.getMetaDataMap().get("assignee", String.class);
55.         logger.info("User to add permissions: " + userID);
56.         String payload = workflowData.getPayload().toString();
57.         logger.info("Content path to add permissions: " + payload);
58.
59.         UserManager uM;
60.         try {
61.             Session jcrSession = workflowSession.adaptTo(Session.class);
62.             //The user that actually modifies the permissions is the workflow-process-service
63.             //The workflow-process-service must have Read ACL and Write ACL permissions for the payload
64.             logger.info("Service user to change permissions: " + jcrSession.getUserID());
65.
66.             uM = AccessControlUtil.getUserManager(jcrSession);
67.             //Get the Authorizable object for the new user
68.             Authorizable authorizable = uM.getAuthorizable(userID); //this might be a user or a group
69.
70.             AccessControlManager accessControlManager = jcrSession.getAccessControlManager();
71.
72.             //JCR privileges that encompass AEM Permissions: "Modify, Create, Delete" in the /useradmin
73.             Privilege[] privileges = {accessControlManager.privilegeFromName(Privilege.JCR_MODIFY_PROPERTIES),
74.                     accessControlManager.privilegeFromName(Privilege.JCR_LOCK_MANAGEMENT),
75.                     accessControlManager.privilegeFromName(Privilege.JCR_VERSION_MANAGEMENT),
76.                     accessControlManager.privilegeFromName(Privilege.JCR_REMOVE_CHILD_NODES),
77.                     accessControlManager.privilegeFromName(Privilege.JCR_REMOVE_NODE),
78.                     accessControlManager.privilegeFromName(Privilege.JCR_ADD_CHILD_NODES),
79.                     accessControlManager.privilegeFromName(Privilege.JCR_NODE_TYPE_MANAGEMENT)};
80.
81.             //Get the ACL for the payload
82.             AccessControlList acl = AccessControlUtils.getAccessControlList(jcrSession, payload);
83.             //Add the user/privileges to the payload ACL
84.             acl.addAccessControlEntry(authorizable.getPrincipal(), privileges);
85.
86.             //Based on the process step arguments, permissions are either added or removed
87.             //Assumes arguments are given as: permission=add
88.             String arguments = workflowArgs.get("PROCESS_ARGS", "");
89.             if(arguments.contains("permission")){
90.                 String permission = arguments.split("=")[1];
91.                 if(permission.equals("add")){ //Adds permissions to edit the payload
92.                     logger.info("Adding permissions");
93.                     accessControlManager.setPolicy(payload, acl);
94.                     logger.info("Added modify permissions to " + payload + " for " + userID);
95.                 }else if(permission.equals("remove")){ //Removes permissions to edit the payload
96.                     logger.info("Removing permissions");
97.                     accessControlManager.removePolicy(payload, acl);
98.                     logger.info("Removed modify permissions to " + payload + " for " + userID);
99.                 }
100.            }else{
101.                //do nothing, if the workitem argument is not set
102.                logger.info("No arguments given for workitem");
103.            }
104.
105.            jcrSession.save();
106.        } catch (Exception e) {
107.            logger.error(e.getMessage(), e);
108.            e.printStackTrace();
109.        }
110.    }
111. }
```

Note: The **old-AutoAssignACL.java** Java file was created for AEM 6.5 which is non-compliant.

3. In Explorer, navigate to **/Exercise_Files_TB/Cloud_Manager/** and open the file **build_project_issue-Backend65-fail1.csv**.

4. Observe the **Issue**, **Severity**, **Line number** and **Rules** in the Excel output generated for the file **AutoAssignACL.java**, as shown:

	A	B	C	D	E	F	G
1	File Location	Line Num	Issue	Type	Severity	Effort	Rule
2	com.adobe:training.core:src/main/java/com/adobe/training/core/AutoAssignACL.java	106	Catch a list of specific exception subty	Code Sme	Major	15min	squid:S2221
3	com.adobe:training.core:src/main/java/com/adobe/training/core/AutoAssignACL.java	107	Do not use Exception.getMessage() as	Code Sme	Minor	15min	CQRules:CQBP-44---ExceptionGetMessageIsFir
4	com.adobe:training.core:src/main/java/com/adobe/training/core/AutoAssignACL.java	108	Do not use Exception.printStackTrace(	Code Sme	Minor	15min	CQRules:CQBP-44---ExceptionPrintStackTrace
5	com.adobe:training.core:src/main/java/com/adobe/training/core/listeners/ReplicationListener.jav	48	Catch a list of specific exception subty	Code Sme	Major	15min	squid:S2221
6	com.adobe:training.core:src/main/java/com/adobe/training/core/listeners/ReplicationListener.jav	50	Do not use Exception.printStackTrace(	Code Sme	Minor	15min	CQRules:CQBP-44---ExceptionPrintStackTrace
7	com.adobe:training.core:src/main/java/com/adobe/training/core/listeners/TitlePropertyListener.ja	48	A "NullPointerException" could be thr	Bug	Major	10min	squid:S2259
8	com.adobe:training.core:src/main/java/com/adobe/training/core/listeners/TitlePropertyListener.ja	40	Use constant JCR_TITLE from interface	Code Sme	Minor		AEM Rules:AEM-2
9	com.adobe:training.core:src/main/java/com/adobe/training/core/ReplicationLogger.java	50	Do not use Exception.printStackTrace(	Code Sme	Minor	15min	CQRules:CQBP-44---ExceptionPrintStackTrace
10	com.adobe:training.core:src/main/java/com/adobe/training/core/schedulers/StockImportSchedule	141	Use try-with-resources or close this "J	Bug	Blocker	5min	squid:S2095
11	com.adobe:training.core:src/main/java/com/adobe/training/core/schedulers/StockImportSchedule	135	HttpClient instances should always ha	Bug	Critical	15min	CQRules:ConnectionTimeoutMechanism
12	com.adobe:training.core:src/main/java/com/adobe/training/core/schedulers/StockImportSchedule	199	Use constant NT_SLING_ORDERED_FOI	Code Sme	Minor		AEM Rules:AEM-2
13	com.adobe:training.core:src/main/java/com/adobe/training/core/schedulers/StockImportSchedule	204	Use constant NT_UNSTRUCTURED from	Code Sme	Minor		AEM Rules:AEM-2
14	com.adobe:training.core:src/main/java/com/adobe/training/core/schedulers/StockImportSchedule	203	Use constant NT_SLING_ORDERED_FOI	Code Sme	Minor		AEM Rules:AEM-2
15	com.adobe:training.core:src/main/java/com/adobe/training/core/servlets/CSVPageCreator.java	58	Make "logger" transient or serializable	Code Sme	Critical	30min	squid:S1948
16	com.adobe:training.core:src/main/java/com/adobe/training/core/servlets/CSVPageCreator.java	61	Make "replicator" transient or serializa	Code Sme	Critical	30min	squid:S1948
17	com.adobe:training.core:src/main/java/com/adobe/training/core/servlets/CSVPageCreator.java	94	Define and throw a dedicated exceptio	Code Sme	Major	20min	squid:S00112
18	com.adobe:training.core:src/main/java/com/adobe/training/core/servlets/CSVPageCreator.java	190	Catch a list of specific exception subty	Code Sme	Major	15min	squid:S2221
19	com.adobe:training.core:src/main/java/com/adobe/training/core/servlets/PageCreator.java	50	Make "replicator" transient or serializa	Code Sme	Critical	30min	squid:S1948
20	com.adobe:training.core:src/main/java/com/adobe/training/core/servlets/PageCreator.java	55	Remove this useless assignment to loc	Code Sme	Major	15min	squid:S1854
21	com.adobe:training.core:src/main/java/com/adobe/training/core/servlets/PageCreator.java	126	Catch a list of specific exception subty	Code Sme	Major	15min	squid:S2221
22	com.adobe:training.core:src/main/java/com/adobe/training/core/servlets/SearchServlet.java	73	Catch a list of specific exception subty	Code Sme	Major	15min	squid:S2221
23	com.adobe:training.core:src/main/java/com/adobe/training/core/servlets/SearchServlet.java	74	Do not use Exception.printStackTrace(	Code Sme	Minor	15min	CQRules:CQBP-44---ExceptionPrintStackTrace
24	com.adobe:training.core:src/main/java/com/adobe/training/core/servlets/TitleSlingServlet.java	44	Don't use format strings that may be e	Vulnerabi	Major	15min	CQRules:CWE-134
25	com.adobe:training.core:src/main/java/com/adobe/training/core/servlets/TitleSlingServlet.java	19	Do not use Sling servlet paths to regist	Code Sme	Major	30min	CQRules:CQBP-75

5. Notice the error occured around line number 106 which is the catch block in the Java class.

Note: The reason for the failure is the above code **AutoAssignACL.java** was created for AEM 6.5 and was non-compliant.

6. Compare the old Java class with your new class /**Exercise_Files_TB/core/src/test/java/com/adobe/training/core/servlets/AutoAssignACL.java** used in this course and observe how the catch block has been updated.

Note: The code **AutoAssignACL.java** that has been used in this class is fully compliant and hence it passed the CICD pipeline.

References

You can use the following links for more information on:

- Cloud Manager: https://docs.adobe.com/content/help/en/experience-manager-cloud-manager/using/introduction-to-cloud-manager.html
- Key concepts: https://docs.adobe.com/content/help/en/experience-manager-cloud-manager/using/overview/key-concepts.html
- Cloud Manager API: https://www.adobe.io/apis/experiencecloud/cloud-manager/docs.html
- Key Features of Cloud Manager: https://helpx.adobe.com/experience-manager/kt/platform-repository/using/cloud-manager-feature-video-understand.html
- Manage Environments - Cloud Service: https://docs.adobe.com/content/help/en/experience-manager-cloud-service/implementing/using-cloud-manager/manage-environments.html
- SonarQube: https://docs.sonarqube.org/latest/user-guide/rules/
- Code quality rules: https://docs.adobe.com/content/help/en/experience-manager-cloud-manager/using/how-to-use/custom-code-quality-rules.html
- Functional testing docs: https://experienceleague.adobe.com/docs/experience-manager-cloud-service/implementing/using-cloud-manager/test-results/functional-testing.html
- UI Tests: https://experienceleague.adobe.com/docs/experience-manager-cloud-service/implementing/using-cloud-manager/test-results/ui-testing.html
- Testing documentation: https://docs.adobe.com/content/help/en/experience-manager-cloud-manager/using/how-to-use/understand-your-test-results.html
- AEM Rules for SonarQube: https://github.com/wttech/AEM-Rules-for-SonarQube
- CQ: https://docs.adobe.com/content/help/en/experience-manager-cloud-manager/using/how-to-use/custom-code-quality-rules.html
- Pipelines: https://experienceleague.adobe.com/docs/experience-manager-cloud-manager/using/how-to-use/configuring-pipeline.html
- Pipelines - Non-production: https://experienceleague.adobe.com/docs/experience-manager-cloud-manager/using/how-to-use/configuring-pipeline.html?lang=en#non-production-%26-code-quality-only-pipelines
- Managing Code