



Student Workbook

**Adobe Experience Manager
Technical Basics for Cloud**

©2020 Adobe. All rights reserved.

If this guide is distributed with software that includes an end user agreement, this guide, as well as the software described in it, is furnished under license and may be used or copied only in accordance with the terms of such license. Except as permitted by any such license, no part of this guide may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, recording, or otherwise, without the prior written permission of Adobe. Please note that the content in this guide is protected under copyright law even if it is not distributed with software that includes an end user license agreement.

The content of this guide is furnished for informational use only, is subject to change without notice, and should not be construed as a commitment by Adobe. Adobe assumes no responsibility or liability for any errors or inaccuracies that may appear in the informational content contained in this guide.

Please remember that existing artwork or images that you may want to include in your project may be protected under copyright law. The unauthorized incorporation of such material into your new work could be a violation of the rights of the copyright owner. Please be sure to obtain any permission required from the copyright owner.

Any references to company names in sample templates are for demonstration purposes only and are not intended to refer to any actual organization.

Adobe, the Adobe logo, Acrobat, the Creative Cloud logo, and the Adobe Marketing Cloud logo are either registered trademarks or trademarks of Adobe in the United States and/or other countries.

All other trademarks are the property of their respective owners.

Adobe, 345 Park Avenue, San Jose, California 95110, USA.

Notice to U.S. Government End Users. The Software and Documentation are "Commercial Items," as that term is defined at 48 C.F.R. §2.101, consisting of "Commercial Computer Software" and "Commercial Computer Software Documentation," as such terms are used in 48 C.F.R. §12.212 or 48 C.F.R. §227.7202, as applicable. Consistent with 48 C.F.R. §12.212 or 48 C.F.R. §§227.7202-1 through 227.7202-4, as applicable, the Commercial Computer Software and Commercial Computer Software Documentation are being licensed to U.S. Government end users (a) only as Commercial Items and (b) with only those rights as are granted to all other end users pursuant to the terms and conditions herein. Unpublished-rights reserved under the copyright laws of the United States. Adobe agrees to comply with all applicable equal opportunity laws including, if appropriate, the provisions of Executive Order 11246, as amended, Section 402 of the Vietnam Era Veterans Readjustment Assistance Act of 1974 (38 USC 4212), and Section 503 of the Rehabilitation Act of 1973, as amended, and the regulations at 41 CFR Parts 60-1 through 60-60, 60-250, and 60-741. The affirmative action clause and regulations contained in the preceding sentence shall be incorporated by reference.

07/22/2020

Contents

Module 1	Adobe Experience Manager Architecture	6
	AEM Architecture	7
	Granite UI	8
	Fluid Experiences	12
	References	14
Module 2	Installation	15
	AEM Installation	16
	Exercise 1: Install the AEM SDK Quickstart Jar	21
	AEM Local Log Files	27
	Exercise 2: Observe the local log files	29
	References	30
Module 3	Developer Tools	31
	Developer Tools	32
	Creating and Building New Packages	36
	Downloading Packages to the File System	37
	Exercise 1: Install a package	39
	Exercise 2: Create, build, and download packages	47
	References	53
Module 4	Sites Authoring Basics	54
	AEM UI Features	55
	Navigation	56
	Tools	57
	AEM Sites as a Cloud Service	58
	Exercise 1: Create a page in AEM	62
	Exercise 2: Edit a page in AEM	66
	References	71
Module 5	Assets Authoring Basics	72
	Assets Console	73
	Exercise 1: Create a folder and upload assets to it	80
	Metadata: An Overview	83
	Exercise 2: Add metadata to an asset	86

Module 6	Creating a Project using Maven	89
	Working with Maven	90
	Working with Node.js and NPM	91
	(Optional) Exercise 1: Install build tools	92
	Exercise 2: Create an AEM project	97
	AEM Archetype	102
	Archetype Explanation:	103
	AEM as a Cloud Service Project's Content Package Structure	104
	Maven Profiles	109
	Exercise 3: Install the project into AEM	110
	References	116
Module 7	Develop using Eclipse	117
	Installing and Configuring Eclipse	118
	Exercise 1: Install and configure Eclipse (Optional)	120
	Exercise 2: Install the Eclipse AEM plug-in (Optional)	124
	Exercise 3: Import an AEM project	128
	Exercise 4: Build and deploy the project to AEM	134
	Exercise 5: Synchronize with AEM and Eclipse	142
	References	149
Module 8	Development with Visual Studio Code	150
	Visual Studio Code	151
	AEM Sync	152
	Exercise 1: Install Visual Studio Code and AEM Sync	153
	Exercise 2: Add styling using Visual Studio Code	156
	References	164
Module 9	Using Brackets for Development	165
	Brackets	166
	Exercise 1: Install Brackets and the AEM Brackets extension	168
	Exercise 2: Make changes to the repository using Brackets	171
	References	175

Module 10	Front-End Development using aemfed	176
	aemfed	177
	Exercise 1: Install aemfed (Optional)	178
	Exercise 2: Configure and run aemfed	180
	Exercise 3: Install content locally (Optional)	182
	Exercise 4: Add a component style using aemfed	183
	References	190
Module 11	Cloud Manager Basics	191
	Cloud Manager	192
	Setup a Program	193
	Managing your code	197
	CI/CD Pipeline	199
	SonarQube	202
	Exercise 1: Observe the custom code quality rules	204
	References	207

Adobe Experience Manager Architecture



Introduction

Adobe Experience Manager (AEM) is a modern, cloud 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:

- Explain fluid experiences in AEM
- Explain the architecture of AEM

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.

Adobe Experience Manager (AEM) as a Cloud Service has resulted in changes to the architecture.

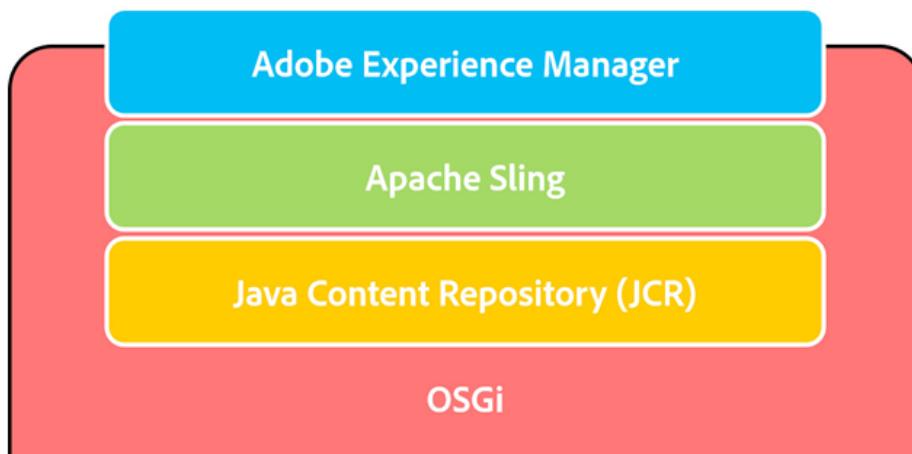
To learn more about AEM Architecture, go to:

<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.

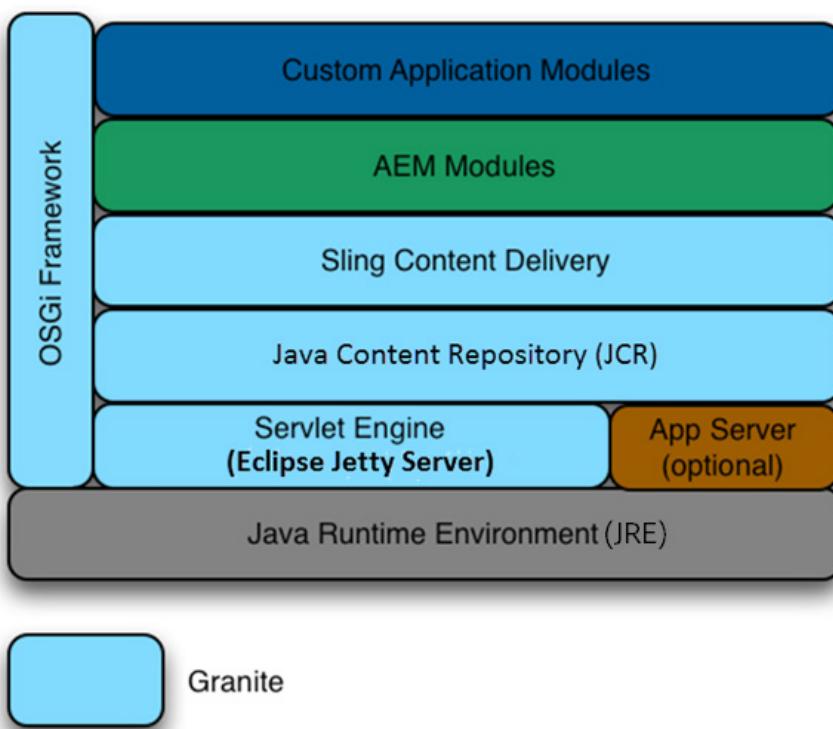
OSGi Framework

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.

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.

AEM also works with other JCR repositories, such as Apache Jackrabbit 2.x, and with a number of non-JCR data stores through connectors. Therefore, it is capable of pulling content from its built-in Oak repository and any JCR-compliant source, such as a third-party repository (for example, Jackrabbit 2.x) or a connector that exposes the legacy storage through the JCR.

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.

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.

Application-Level Functions

At the application-level, AEM has the following functions:

- AEM Sites (Web Experience Management, Digital Signage (Screens), e-Commerce, and Online Communities)
- AEM Assets (Digital Asset Management, Assets Sharing and Dynamic Media Delivery)
- AEM Forms (Online Forms Management and Dynamic Customer Communications)
- Managed Services (Adobe-hosted operations for AEM deployments)
- Content Services
- Multi-Site Manager
- Workflows

The above application functions/modules sit on top of the Granite platform. They share the same infrastructure and UI framework. These modules are encapsulated within the OSGi container and are very tightly integrated with each other.

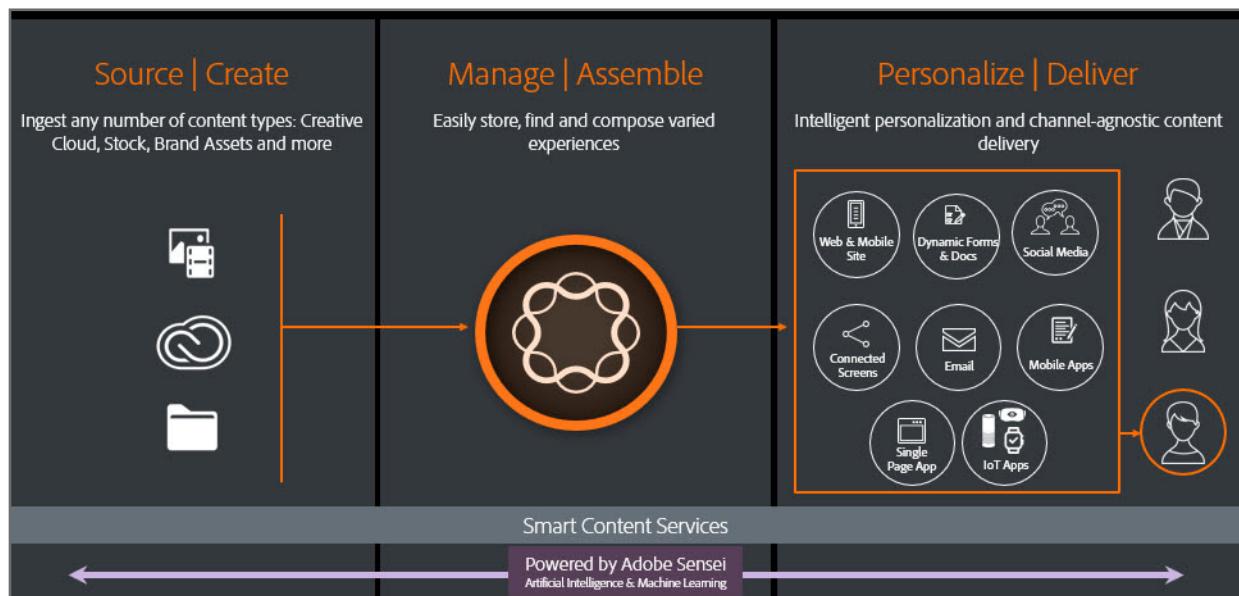
Fluid Experiences

AEM enables your organization to create fluid experiences for your customers. This means no matter where your customer is at, you will be there offering a seamless user experience that is consistent and connected across all channels and devices.

Throughout the customer journey, customers use both offline and online touch points to engage with an organization's products and services. It is important that organizations provide a fluid experience across these multiple touch points as customer engagement increases through the use of various channels.

AEM helps you deliver these fluid experiences by supporting true omnichannel experiences across owned, earned, and paid channels. Content goes beyond the traditional "web" form and is available in a multitude of end points – screens, dynamic adaptive forms, documents, email, social, apps, and more.

The Content Services feature in AEM enables you to reuse web content in mobile app, single page applications, and custom applications.



AEM provides rich omni-channel support for fast, personalized experiences, anywhere:

- Central asset management: Centrally manage all assets across all enterprise and departmental needs with robust admin, workflows, governance, and access rights management.
- Content as a service: Use content as a service in any app, service, site, or system to ensure consistent and connected experiences across all channels.
- Experiences at scale: Create modern experiences with any environment and easily reuse content to scale your mobile or web applications and physical experiences using APIs with low-code/no-code.
- Dynamic Asset Optimization: Maximize asset use and ROI by ensuring your images and videos can automatically adapt for any experience, channel or touchpoint.

With these capabilities, users can model, author and deliver content on any level (raw content fragment, fragment in page, experience fragment, and page) via API, for consumption in any channel.

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/>

Installation

Introduction

Development on Adobe Experience Manager (AEM) typically occurs on a local author service with the AEM SDK. 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:

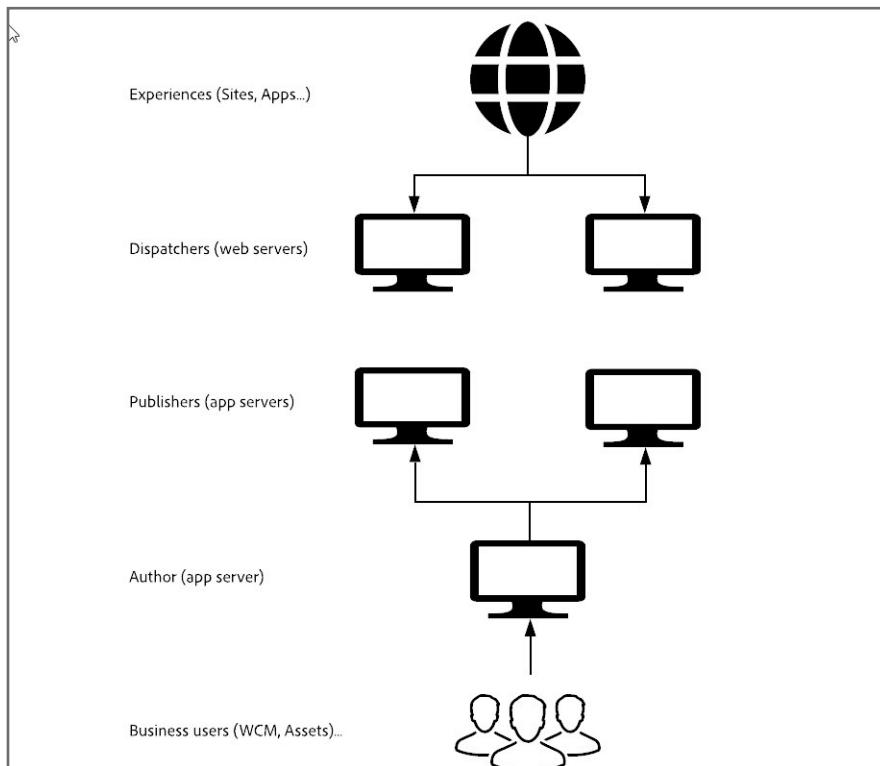
- Explain the different types of AEM Services
- Install and run AEM Services by using the command line method

AEM Installation

Local AEM Runtime: The AEM as a Cloud Service SDK provides a QuickStart Java Archive (JAR) file that runs a local version of AEM. You can use the Quickstart .jar to run either the AEM Author Service or AEM Publish Service locally.

- **Author:** An AEM 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.
- **Publish:** An AEM Service that serves the published content to the public.

The following graphic depicts a typical AEM implementation:



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

Installation Prerequisites

To install AEM, you need:

- AEM installation and startup JAR file (also known as the cloud-ready quickstart file)
- A valid AEM license key properties file
- 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: After the installation is complete, the quickstart file is referred to as the AEM startup file.



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:

```
C:\Users\pprpush>java -version
java version "11.0.6" 2020-01-14 LTS
Java(TM) SE Runtime Environment 18.9 (build 11.0.6+8-LTS)
Java HotSpot(TM) 64-Bit Server VM 18.9 (build 11.0.6+8-LTS, mixed mode)
```

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 AEM SDK Quickstart

AEM 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
```

```
C:\adobe\AEM\author>java -jar aem-author-4502.jar -h
Loading quickstart properties: default
Loading quickstart properties: instance
Setting properties from filename 'C:/adobe/AEM/author/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.
-----
```

The AEM quickstart installer will show all the available command-line options without starting the server.

In addition, you need to tune the JVM used for running AEM. Tuning the JVM is an important and delicate task and requires a more realistic environment in terms of resources (hardware and the operating system) and workload (content and requests). 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)

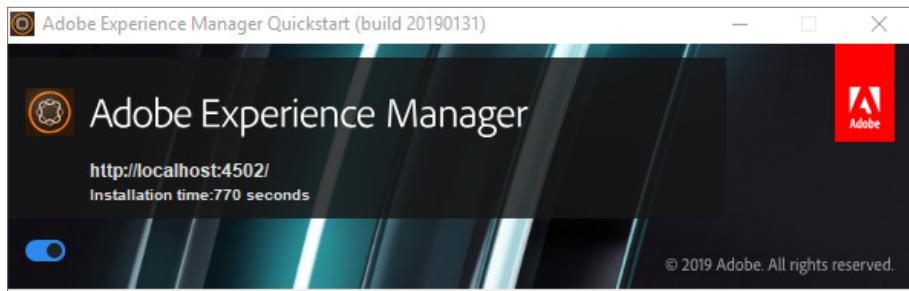
-Xmx --> assigns the maximum size to which the heap can grow

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 -Xms512m -Xmx1024m -jar aem-sdk-quickstart.jar
```

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 AEM start URL (where the port number is the one you defined on installation).

Start AEM with Runmodes

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

The supported runmode configurations are:

- config (The default, applies to all AEM Services)
- config.author (Applies to all AEM Author service)
- config.author.dev (Applies to AEM Dev Author service)
- config.author.stage (Applies to AEM Staging Author service)
- config.author.prod (Applies to AEM Production Author service)
- config.publish (Applies to AEM Publish service)
- config.publish.dev (Applies to AEM Dev Publish service)
- config.publish.stage (Applies to AEM Staging Publish service)
- config.publish.prod (Applies to AEM Production Publish service)
- config.dev (*Applies to AEM Dev services)
- config.stage (*Applies to AEM Staging services)
- config.prod (*Applies to AEM Production services)

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 tells AEM to set this Service as the specified runmode. The `-gui` option turns on the GUI mode that shows the AEM icon window on your system.

Start AEM with scripts:

After you install AEM using the command line method, you can use the built-in batch files in the **crx-quickstart** directory to start AEM. You can also configure the startup and additional options using these batch files.

Navigate to your install directory and then to the **crx-quickstart** directory that was created as part of your installation. Within the `\crx-quickstart\bin` subdirectory, there are batch files (on Windows systems) available for you to start or stop AEM, and get status information on the server. The server parameters provided in these batch files are documented using comments or commented-out as appropriate in the case of options that you may want to set for your Services.

Changing environments, `dev`, `stage`, and `prod`, can be useful for developers to ensure environment-specific configurations are correctly defined and resolved by AEM. It is recommended that local development primarily be done against the default `dev` environment run mode.

Updating the QuickStart Jar

AEM 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 AEM 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 AEM SDK Quickstart Jar

Scenario: As a developer or administrator, you need to install and start/stop a development Service of AEM on a local machine, using the quickstart JAR file and license.properties.

In this exercise, you will install and start an AEM 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 AEMaaCS 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 **AEM SDK**.

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

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

1. 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:
 - a. Windows: C:/adobe/aem-sdk/
 - b. MacOS X: \Applications\adobe\aem-sdk\ or *x: \opt\adobe\aem-sdk\

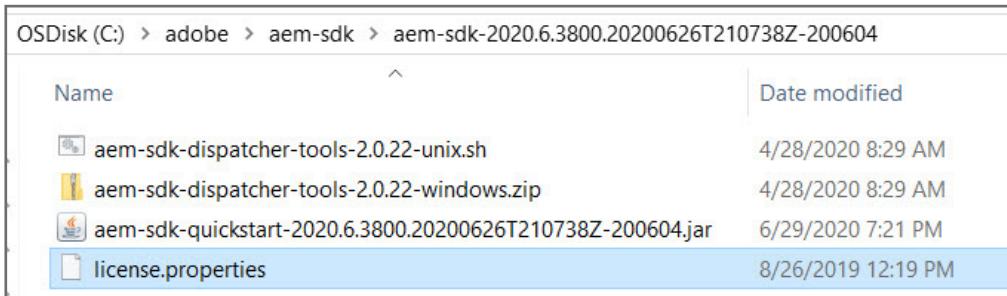
 **Note:** If you are using ReadyTech, the SDK has already been downloaded for you under C:/adobe/aem-sdk/.

2. Extract the Quickstart jar file from the downloaded AEM SDK zip file.

For example: C:/adobe/aem-sdk/aem-sdk-<nnnn>/

Name	Date modified	Type	Size
aem-sdk-dispatcher-tools-2.0.22-unix.sh	4/28/2020 8:29 AM	Shell Script	7,918 KB
aem-sdk-dispatcher-tools-2.0.22-windows.zip	4/28/2020 8:29 AM	Compressed (zipp...)	6,383 KB
aem-sdk-quickstart-2020.6.3800.20200626T210738Z-200604.jar	6/29/2020 7:21 PM	Executable Jar File	383,413 KB

3. Copy the **license.properties** files from **Exercise_Files_TB\AEM_Installation** to your newly created directory.



Name	Date modified
aem-sdk-dispatcher-tools-2.0.22-unix.sh	4/28/2020 8:29 AM
aem-sdk-dispatcher-tools-2.0.22-windows.zip	4/28/2020 8:29 AM
aem-sdk-quickstart-2020.6.3800.20200626T210738Z-200604.jar	6/29/2020 7:21 PM
license.properties	8/26/2019 12:19 PM

 **Note:** If you are using ReadyTech, you can just copy the license file from **C/adobe/aem-sdk** to your newly created folder.

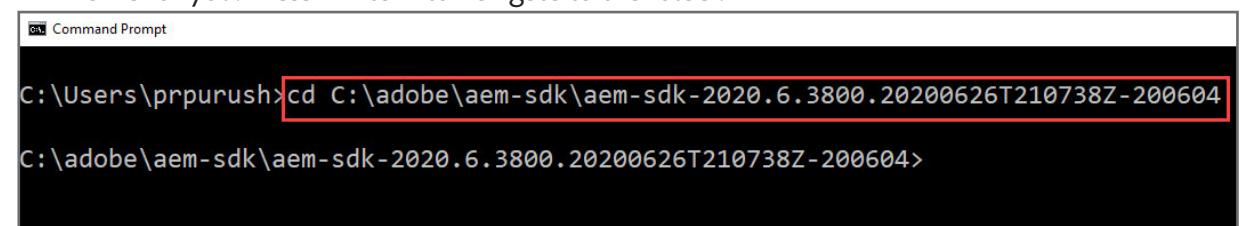
4. Rename the **aem-sdk-quickstart-<nnnn>.jar** file to **aem-sdk-quickstart.jar**. For simplicity, remove the build number from the jar file.
 - a. aem = Application
 - b. sdk-quickstart = Local Cloud quickstart
 - c. 2020.6.3800.20200626T210738Z-200604 = the build of this SDK

 **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.

5. 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-adk
```

6. 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.



```
Command Prompt
C:\Users\prpurush>cd C:\adobe\...
C:\adobe\...-2020.6.3800.20200626T210738Z-200604>
```

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

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

Windows: **dir**

Mac: ls

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-sdk-quickstart.jar -h
```

```
C:\adobe\adobe-aem-sdk\adobe-aem-sdk-2020.6.3800.20200626T210738Z-200604>java -jar aem-sdk-quickstart.jar -h
Loading quickstart properties: default
Loading quickstart properties: instance
Setting properties from filename 'C:/adobe/aem-sdk/aem-sdk-2020.6.3800.20200626T210738Z-200604/aem-sdk-quickstart.jar'
-----
Adobe Experience Manager Quickstart (build 2020.6.3800)
-----
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
--gui
      Show GUI if running on a terminal
```

9. To start your AEM quickstart, 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.

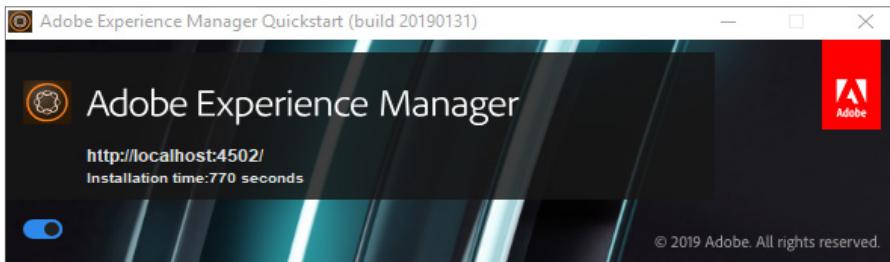
If no port number is specified, AEM will select the first available port from the following list in this specific order: 1) 4502, 2) 8080, 3) 8081, 4) 8082, 5) 8083, 6) 8084, or a random port.

```
C:\adobe\adt\sdk>java -jar aem-sdk-quickstart.jar -p 4502 -r author,dev -gui  
Loading quickstart properties: default  
Loading quickstart properties: instance  
Low-memory action set to fork  
Using 64bit VM settings, min.heap=1024MB, min permgen=256MB, default fork arguments=[-Xmx1024m, -XX:MaxPermSize=256m]  
The JVM reports a heap size of 3942 MB, meets our expectation of 1024 MB +/- 20  
Setting properties from filename 'C:/adobe/aem-sdk/aem-sdk-2020.6.3800.20200626T210738Z-200604/aem-sdk-quickstart.jar'  
Setting 'sling.run.modes' to 'author,dev' from command line.  
Verbose option not active, closing stdin and redirecting stdout and stderr  
Redirecting stdout to C:\adobe\adt\sdk>
```



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

10. Verify your AEM service has started up, with a command window available to view details of the startup. In addition, the GUI window will also be available. Installation will take approximately 5–7 minutes depending on your system's capabilities. After the AEM Author Service has started successfully, the start-up screen (the GUI) will change to something similar to the following:



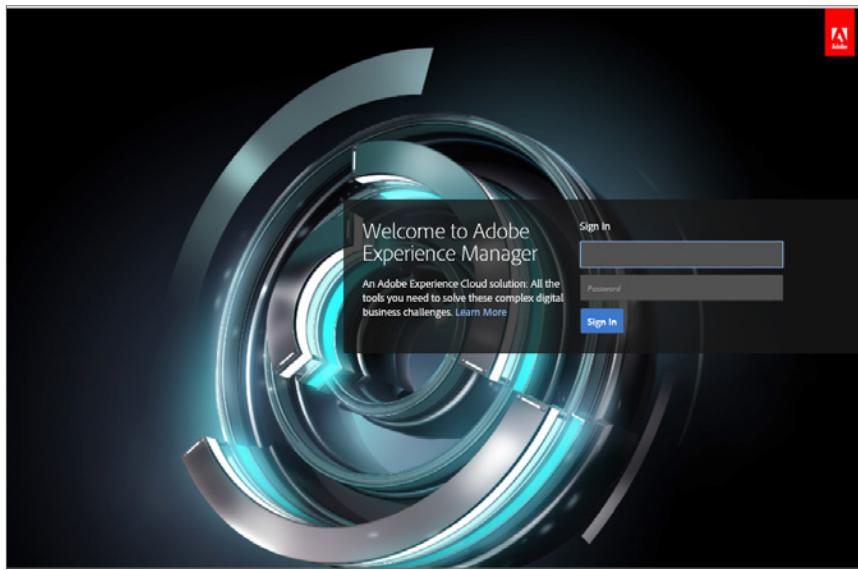
You have now completed the installation of AEM.



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

11. Each time you want to run your AEM 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 AEM 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:



12. Enter your user name and password, and click **Sign In**. If you are using a ReadyTech machine or local installation, use the following credentials to sign in:
- User name: **admin**
 - Password: **admin**
13. Navigate to **Tools > Operations > System Overview** in your AEM author service. 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.

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

The screenshot shows the 'System Overview' page in the AEM admin interface. On the left, there's a 'Health Checks' panel with three sections: 'System Maintenance' (1 Critical), 'Security Checks' (1 Health Check Error), and 'Log Errors, Sling/Granite Content Access Check' (2 Warn). Below that is a 'Maintenance Tasks' panel with one failed task: 'AuditLog Maintenance Task, Workflow Purge'. On the right, the 'Instance' panel displays system details: 'Adobe Experience Manager: 6.6.0', 'Run Modes: dev, s7connect, crx3, author, samplecontent, crx3tar' (which is highlighted with a red box), and 'Instance Up Since: 2019-09-25 14:28:02'. Below the instance panel is a 'Repository' section showing 'Apache Jackrabbit Oak: 1.16.0-R1866730', 'Node Store: Segment Tar', 'Repository Size: 0.16 GB', and 'File Data Store: crx-quickstart/repository/datastore'. At the bottom of the page, there's a note: 'Data Store Garbage Collection: Lucene Binaries'.

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

Pro Tips:

1. Using the command line without the **-Xmx** option will allocate the highest heap that the system can provide.
2. Explicitly setting the **-Xmx** option will allocate the highest heap that the system can provide.
3. Verbose mode is quite useful when stopping the server because it allows you to see the shutdown log in the command window, which allows you to know when all the OSGi services are terminated without monitoring the **stdout** and **stderr** logs.

AEM Local Log Files

The logging framework in AEM is based on Apache Sling. The root logger defines the global settings for the logging system in AEM. The root logger is configured by using Apache Sling Logging Configuration. The org.apache.sling.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 in AEM

- 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.

OSDisk (C:) > adobe > aem-sdk > author > crx-quickstart > logs	
Name	Date modified
access.log	6/25/2020 3:02 PM
audit.log	6/25/2020 12:44 PM
auditlog.log	6/25/2020 12:44 PM
error.log	6/25/2020 3:03 PM
history.log	6/25/2020 12:44 PM
queryrecorder.log	6/25/2020 3:00 PM
request.log	6/25/2020 3:02 PM
s7access-2020-06-25.log	6/25/2020 12:48 PM
stderr.log	6/25/2020 2:33 PM
stdout.log	6/25/2020 2:33 PM
upgrade.log	6/25/2020 12:44 PM

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 using AEM locally, you can also view the log files through the Web Console at:

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

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](#)

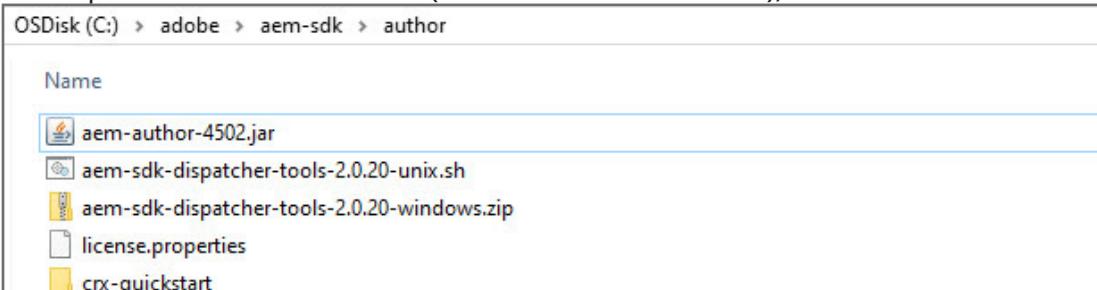
Appender		Configuration
Appender		
File : [/logs/history.log]	C:\adobe\aec-sdk\author\crx-quickstart\logs\history.log	
File : [/logs/error.log]	C:\adobe\aec-sdk\author\crx-quickstart\logs\error.log	
File : [/logs/auditlog.log]	C:\adobe\aec-sdk\author\crx-quickstart\logs\auditlog.log	
File : [/logs/audit.log]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

[Logback Config](#)

Exercise 2: Observe the local log files

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

1. Open the AEM service folder (**C:\adobe\aecm-sdk\author**), as shown:



2. Navigate to **\crx-quickstart\logs**. The list of log files is shown:

OSDisk (C:) > adobe > aem-sdk > author > crx-quickstart > logs	
Name	Date modified
access.log	6/25/2020 3:02 PM
audit.log	6/25/2020 12:44 PM
auditlog.log	6/25/2020 12:44 PM
error.log	6/25/2020 3:03 PM
history.log	6/25/2020 12:44 PM
queryrecorder.log	6/25/2020 3:00 PM
request.log	6/25/2020 3:02 PM
s7access-2020-06-25.log	6/25/2020 12:48 PM
stderr.log	6/25/2020 2:33 PM
stdout.log	6/25/2020 2:33 PM
upgrade.log	6/25/2020 12:44 PM

3. Observe the log files.

References

You can use the following link for more information on:

- Installation:

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

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.
- 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.adobeacmcloud.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

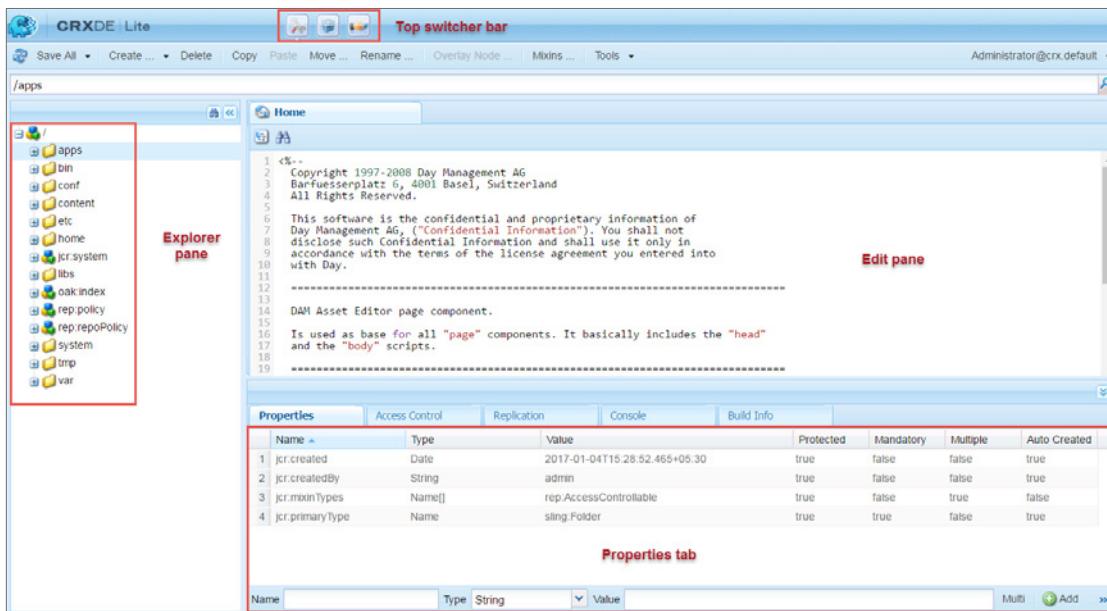
The CRXDE Lite UI contains:

- **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.



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.

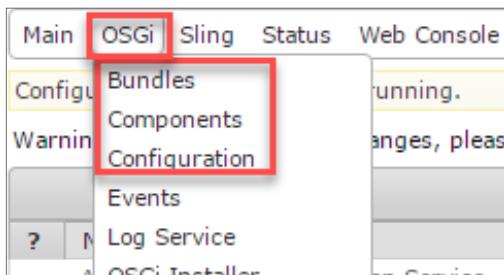
You can access the console at <http://localhost:4502/system/console>.



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.

To validate a package:

1. Verify you are logged on to your AEM author service on port 4502 (<http://localhost:4502>).
2. Navigate to <http://localhost:4502/crx/packmgr/index.jsp>. This will take you to the AEM Package Manager tool.

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.

3. Click **Upload Package**.

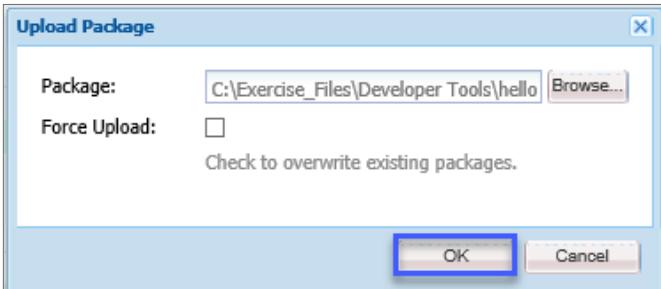


4. In the **Upload Package** dialog box, click **Browse** and select the **aem-guides-wknd.all-0.0.2.zip** package from the **Exercise_Files_TB**(in the \Developer Tools folder) provided to you.



Note: You can also go to the link <https://github.com/adobe/aem-guides-wknd/releases> to download the latest release of the WKND reference site.

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



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

aem-guides-wknd.all-0.0.2.zip
Version: 0.0.2 | Last built 2019/12/20 | dgordon
All content package for WKND Sites Project

Edit Build Install Download Share More ▾

Package: aem-guides-wknd.all
Download: aem-guides-wknd.all-0.0.2.zip (91.8 MB)
Group: wknd
Filters: /apps/wknd-packages
/apps/wknd-vendor-packages

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

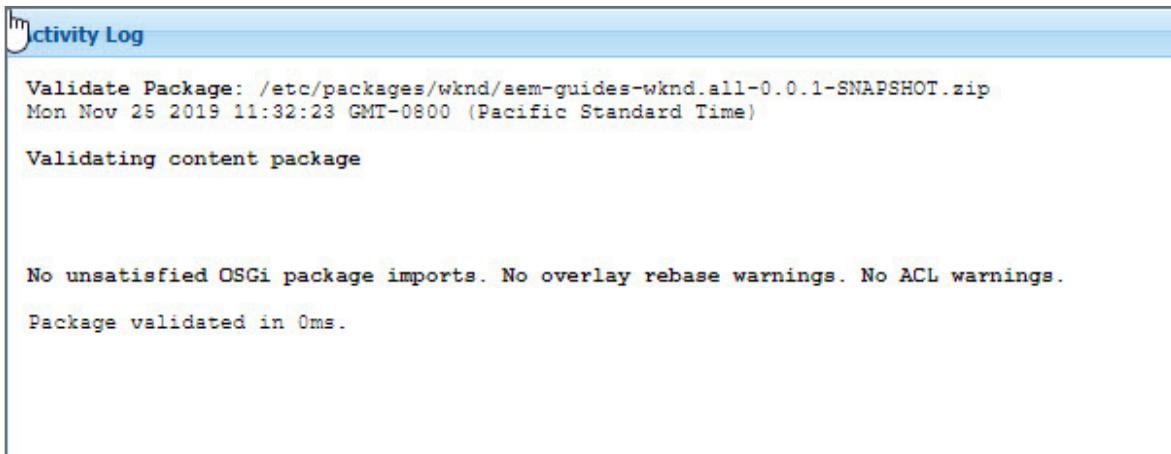
aem-guides-wknd.all-0.0.2.zip
Version: 0.0.2 | Last built 2019/12/20 | dgordon
All content package for WKND Sites Project

Edit Build Install Download Share More ▾

Package: aem-guides-wknd.all
Download: aem-guides-wknd.all-0.0.2.zip (91.8 MB)
Group: wknd
Filters: /apps/wknd-packages
/apps/wknd-vendor-packages

cq-remotedam-client-ui-content-1.1.8.zip
Version: 1.1.8 | Last installed 10:52 | admin

7. Leave all options selected as-is and click **Validate** in the dialog box. In the **Activity Log** below the **aem-guides-wknd.all-0.0.2.zip**, notice there are no issues with your package:

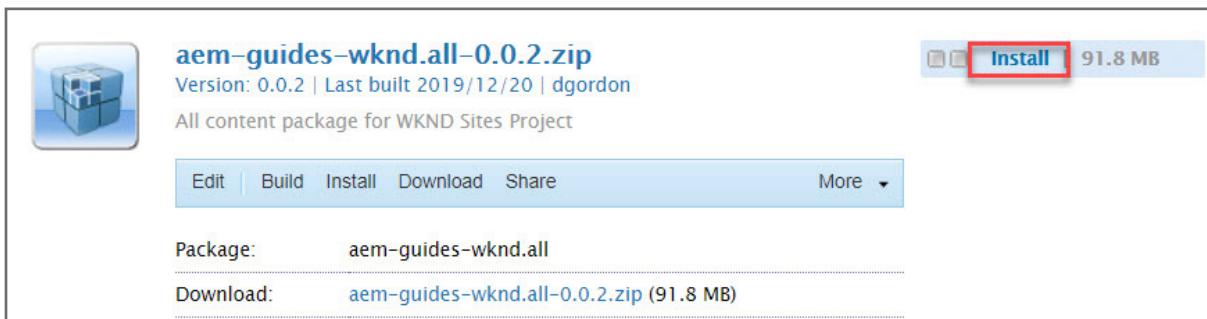


The screenshot shows the 'Activity Log' interface. It displays the command 'Validate Package: /etc/packages/wknd/aem-guides-wknd.all-0.0.1-SNAPSHOT.zip' and the date 'Mon Nov 25 2019 11:32:23 GMT-0800 (Pacific Standard Time)'. Below this, it says 'Validating content package'. The log concludes with 'No unsatisfied OSGi package imports. No overlay rebase warnings. No ACL warnings.' and 'Package validated in Oms.'

 **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:

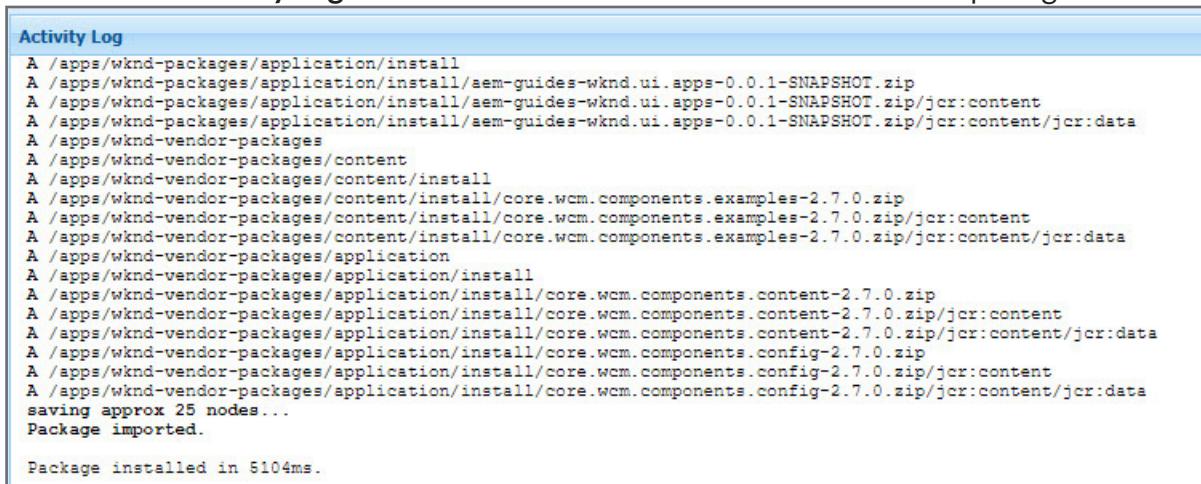
8. Click **Install**, as shown.



The **Install Package** dialog box appears.

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

10. Check the **Activity Log**. You can see that the content was added from the package.



```

Activity Log
A /apps/wknd-packages/application/install
A /apps/wknd-packages/application/install/aem-guides-wknd.ui.apps-0.0.1-SNAPSHOT.zip
A /apps/wknd-packages/application/install/aem-guides-wknd.ui.apps-0.0.1-SNAPSHOT.zip/jcr:content
A /apps/wknd-packages/application/install/aem-guides-wknd.ui.apps-0.0.1-SNAPSHOT.zip/jcr:content/jcr:data
A /apps/wknd-vendor-packages
A /apps/wknd-vendor-packages/content
A /apps/wknd-vendor-packages/content/install
A /apps/wknd-vendor-packages/content/install/core.wcm.components.examples-2.7.0.zip
A /apps/wknd-vendor-packages/content/install/core.wcm.components.examples-2.7.0.zip/jcr:content
A /apps/wknd-vendor-packages/content/install/core.wcm.components.examples-2.7.0.zip/jcr:content/jcr:data
A /apps/wknd-vendor-packages/application
A /apps/wknd-vendor-packages/application/install
A /apps/wknd-vendor-packages/application/install/core.wcm.components.content-2.7.0.zip
A /apps/wknd-vendor-packages/application/install/core.wcm.components.content-2.7.0.zip/jcr:content
A /apps/wknd-vendor-packages/application/install/core.wcm.components.content-2.7.0.zip/jcr:content/jcr:data
A /apps/wknd-vendor-packages/application/install/core.wcm.components.config-2.7.0.zip
A /apps/wknd-vendor-packages/application/install/core.wcm.components.config-2.7.0.zip/jcr:content
A /apps/wknd-vendor-packages/application/install/core.wcm.components.config-2.7.0.zip/jcr:content/jcr:data
saving approx 25 nodes...
Package imported.

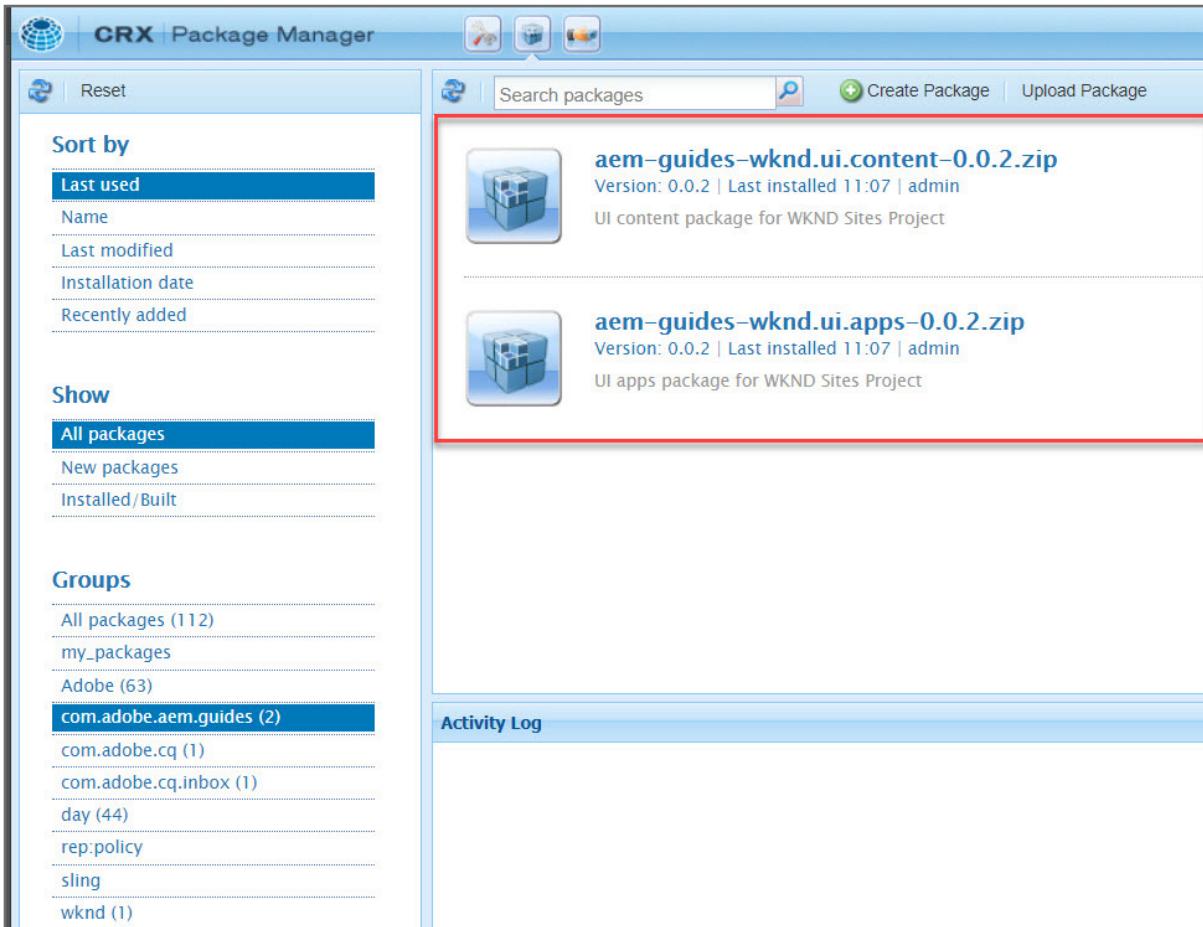
Package installed in 5104ms.

```



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

11. In the left navigation pane, under **Groups**, click on **com.adobe.aem.guides** and verify there are two packages, as shown:



The screenshot shows the CRX Package Manager interface. On the left, the navigation pane displays groups like 'All packages (112)', 'my_packages', 'Adobe (63)', and 'com.adobe.aem.guides (2)'. The 'com.adobe.aem.guides' group is selected. The main panel lists two packages:

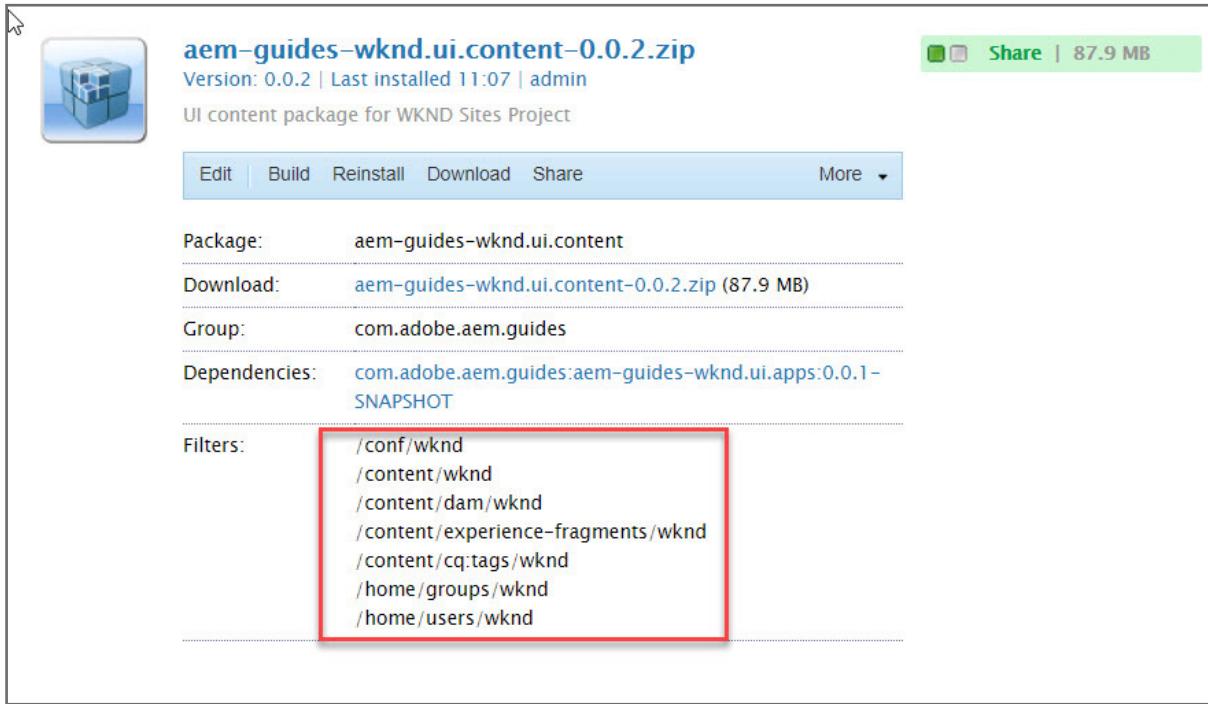
- aem-guides-wknd.ui.content-0.0.2.zip**: Version 0.0.2 | Last installed 11:07 | admin. Description: UI content package for WKND Sites Project.
- aem-guides-wknd.ui.apps-0.0.2.zip**: Version 0.0.2 | Last installed 11:07 | admin. Description: UI apps package for WKND Sites Project.

 Both packages are highlighted with a red border.

 **Note:** If you do not see **com.adobe.aem.guides (2)** in the left navigation pane (under **Groups**) OR, after clicking the **com.adobe.aem.guides (2)** 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:** The container content package contained one mutable content package and one immutable content package. These two content packages can be seen under com.adobe.aem.guides on the left side.

12. The **aem-guides-wknd.ui.content-0.0.2.zip** contains all of the mutable content for the WKND project. Click the **aem-guides-wknd.ui.content-0.0.2.zip** link and scroll down to the **Filters** section:



The screenshot shows the AEM Package Manager interface. At the top, there's a header with the package name **aem-guides-wknd.ui.content-0.0.2.zip**, its version **0.0.2**, the last installed time **11:07**, and the administrator **admin**. To the right is a **Share** button and a size indicator **87.9 MB**. Below the header, the package is described as a **UI content package for WKND Sites Project**. A toolbar below the header includes **Edit**, **Build**, **Reinstall**, **Download**, **Share**, and a **More** dropdown. The main content area displays the following information:

Package:	aem-guides-wknd.ui.content
Download:	aem-guides-wknd.ui.content-0.0.2.zip (87.9 MB)
Group:	com.adobe.aem.guides
Dependencies:	com.adobe.aem.guides:aem-guides-wknd.ui.apps:0.0.1-SNAPSHOT
Filters:	<ul style="list-style-type: none">/conf/wknd/content/wknd/content/dam/wknd/content/experience-fragments/wknd/content/cq:tags/wknd/home/groups/wknd/home/users/wknd

13. The **aem-guides-wknd.ui.apps-0.0.2.zip** contains all of the immutable content for the WKND project. Click the **aem-guides-wknd.ui.content-0.0.2.zip** link and scroll down to the **Filters** section:

The screenshot shows a UI.apps package page for the 'aem-guides-wknd.ui.apps-0.0.2.zip' package. The page includes a navigation bar with 'Edit', 'Build', 'Reinstall', 'Download', 'Share', and 'More'. Below the navigation, there are sections for 'Package', 'Download', 'Group', 'Dependencies', and 'Filters'. The 'Filters' section is highlighted with a red box and contains two entries: '/apps/wknd' and '/apps/msm/wknd_blueprint'.

aem-guides-wknd.ui.apps-0.0.2.zip
Version: 0.0.2 | Last installed 11:07 | admin
UI apps package for WKND Sites Project

Edit Build Reinstall Download Share More ▾

Package: aem-guides-wknd.ui.apps

Download: aem-guides-wknd.ui.apps-0.0.2.zip (510.1 KB)

Group: com.adobe.aem.guides

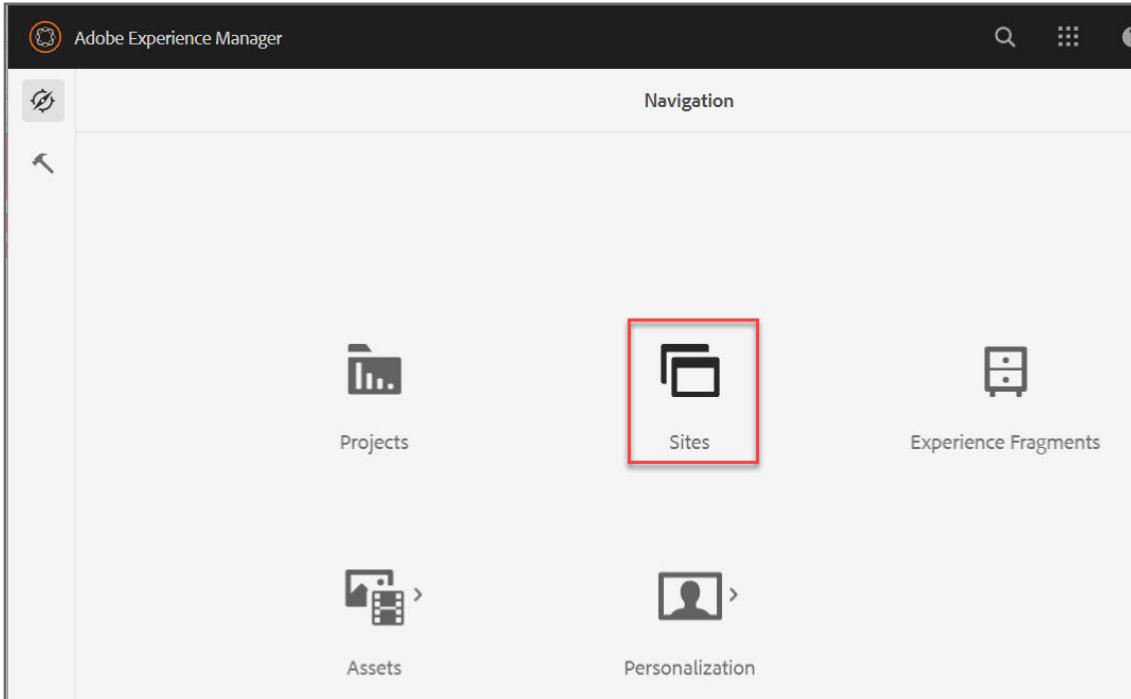
Dependencies: adobe/cq60:core.wcm.components.content:2.8.0
adobe/cq60:core.wcm.components.config:2.8.0

Filters: /apps/wknd
/apps/msm/wknd_blueprint

Task 2: Verify the newly installed site

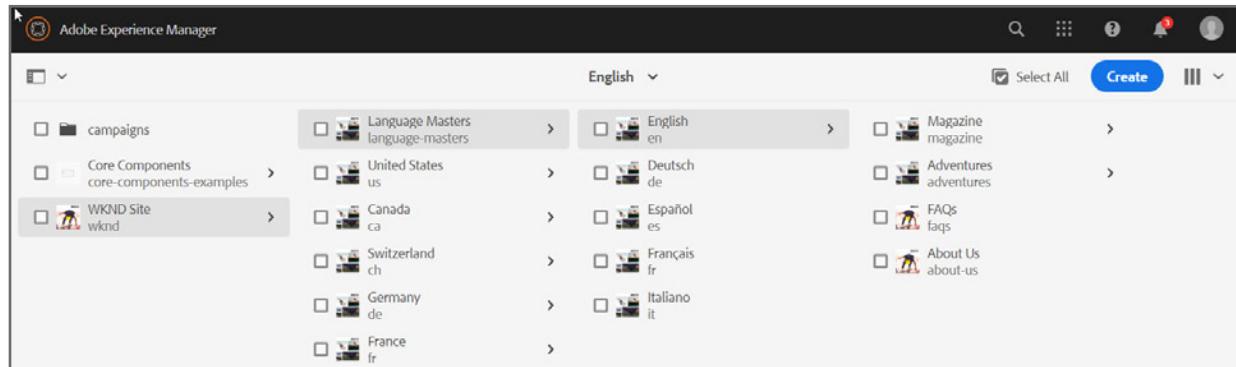
In this task, you will navigate to Sites and verify the newly installed site.

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.



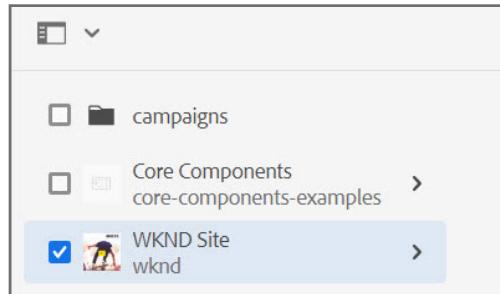
 **Note:** You may see the **Product Navigation** tutorial dialog guiding you through the navigation. You may click **Next** to proceed through the tutorial and learn the basic AEM UI elements and navigation, or you may click **Close** to hide the tutorial.

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



The screenshot shows the Adobe Experience Manager interface in column view. The left sidebar lists 'campaigns', 'Core Components core-components-examples', and 'WKND Site wknd'. The main area shows a tree structure under 'Language Masters': 'United States us', 'Canada ca', 'Switzerland ch', 'Germany de', 'France fr', 'English en' (which is selected, indicated by a grey background), 'Deutsch de', 'Español es', 'Français fr', and 'Italiano it'. To the right of the tree are buttons for 'Select All' and 'Create'.

 **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.



The screenshot shows the Adobe Experience Manager interface in column view. The left sidebar lists 'campaigns', 'Core Components core-components-examples', and 'WKND Site wknd'. The 'WKND Site' node has a checkmark next to its thumbnail, indicating it is selected. The right side of the screen is mostly blank.

Congratulations! 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..

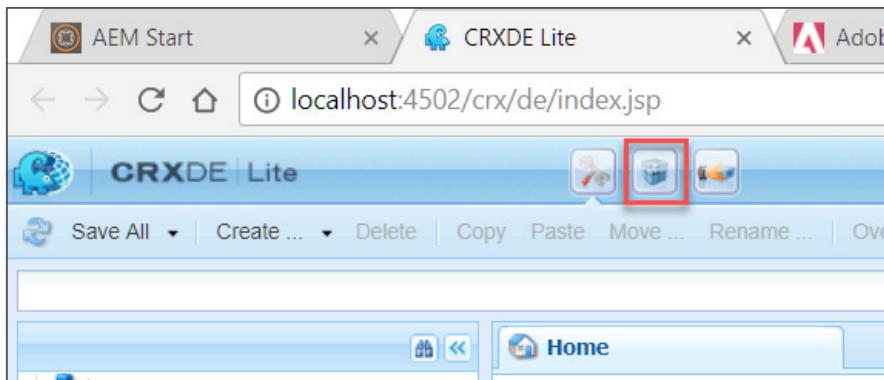
This exercise includes the following tasks:

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 CRXDE Lite browser tab, click the **Package Manager** icon, as shown.

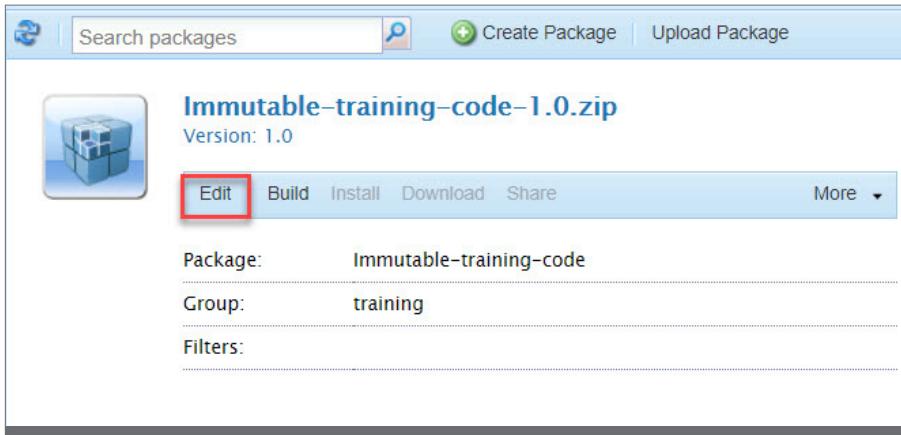


The **Package Manager** screen appears.

2. Click **Create Package** as shown:



3. In the **New Package** dialog box, type the following details:
 - a. Package Name: **Immutable-training-code**
 - b. Version: **1.0**
 - c. Group: **training**
4. Click **OK**. The **Immutable-training-code** package is created.
5. Click **Edit** on the newly created package, as shown.



The screenshot shows the AEM package management interface. At the top, there are buttons for 'Search packages', 'Create Package', and 'Upload Package'. Below this, a package named 'Immutable-training-code-1.0.zip' is listed with its version '1.0'. A red box highlights the 'Edit' button in the action bar below the package name. The package details section shows 'Package: Immutable-training-code', 'Group: training', and an empty 'Filters:' field.

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

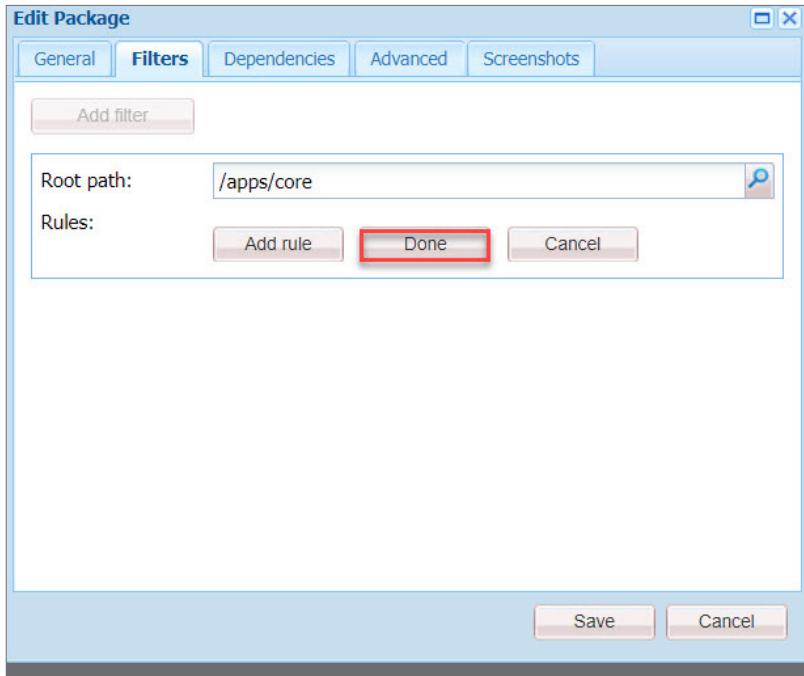
 **Note:** Filters are the mechanism used to add content to packages. Here, you specify the paths that contain the content from the JCR you want to include in a package. Before adding filters, your package is completely empty. You may also restrict file types added to a package using filter rules, such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

7. For the **Root** path, type: **/apps/core**

 **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.

8. Click **Done**, as shown.



9. Click **Save**.

10. Click **Build** to build the package, as shown.



 **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 **Build** again in the confirmation dialog box. The package is now ready for download.
12. 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 now 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 may use the group name to categorize and organize your packages in AEM.

Groups
All packages (108)
Adobe (62)
com.adobe.cq.inbox (1)
day (42)
my_packages
rep:policy
sling
training (1)

Congratulations! 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**.
-



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

8. Click **Done**.
9. Click **Save**.
10. Click **Build** to build the package.
11. Click **Build** again in the confirmation dialog box. The package is now ready for download.
12. 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.

13. On the left side menu, click the **training** group link and notice the packages under the group.

The screenshot shows a software interface for managing packages. At the top, there is a navigation bar with icons for search, create package, and upload package. Below the navigation bar, there are two package entries listed:

- mutable-training-content-1.0.zip**
Version: 1.0 | Build: 1 | Last built 13:58 | admin
- Immutable-training-code-1.0.zip**
Version: 1.0 | Build: 1 | Last built 13:44 | admin

Each package entry includes a small thumbnail icon on the left.

Congratulations! 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

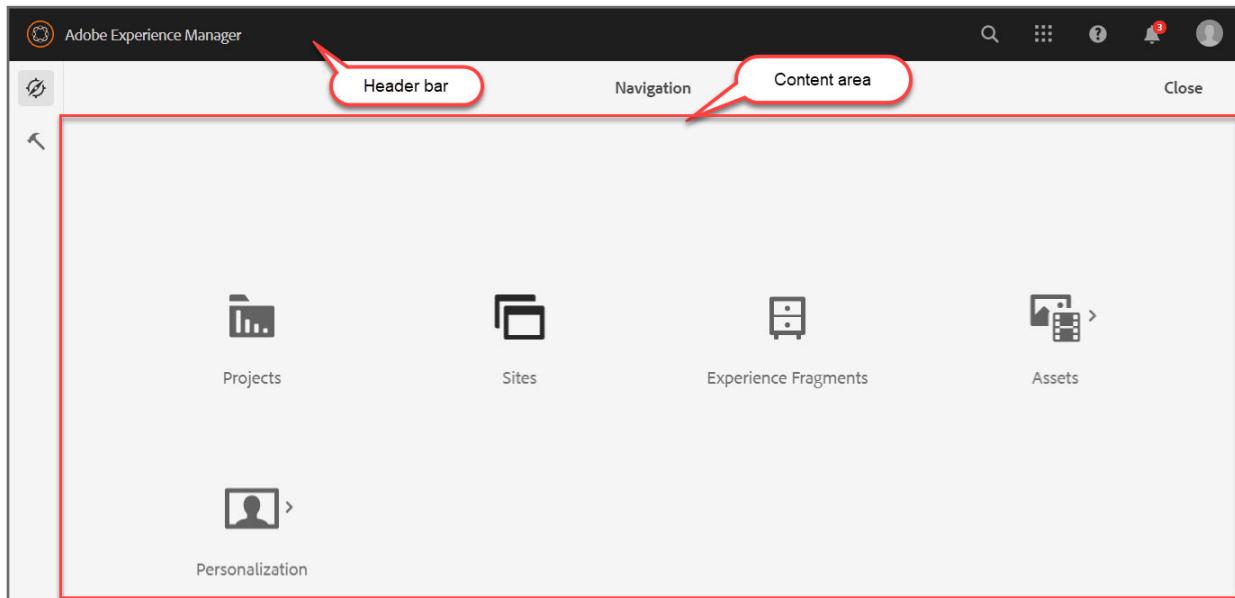
After completing this course, you will be able to:

- Navigate through the AEM UI
- Create and edit pages

AEM UI Features

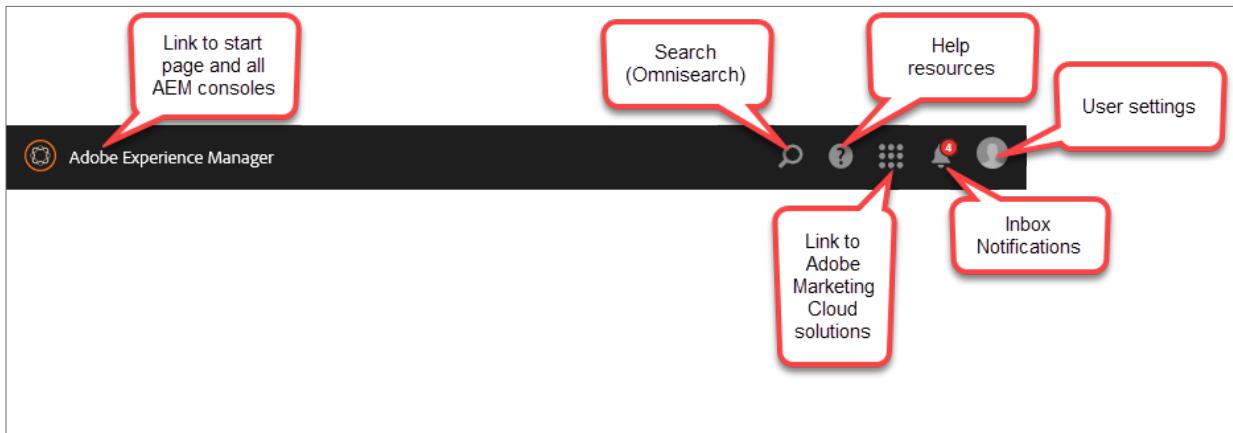
The AEM UI combines the advantages of a web interface with the fluidity and responsiveness that is usually associated with desktop applications. It is touch-optimized for authoring across desktop and mobile devices.

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



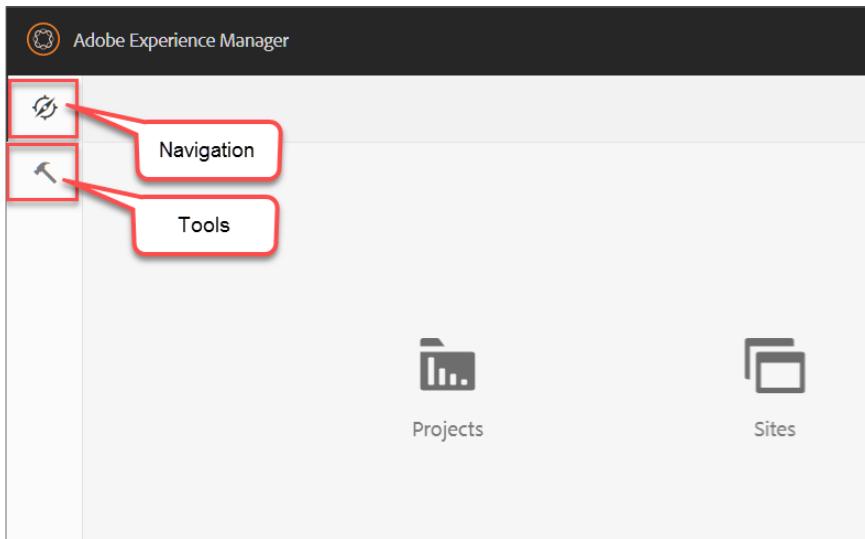
The content area, by default, displays all applications and capabilities of AEM, such as Projects, Sites, Experience Fragments, and Assets.

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



Navigation and Tools

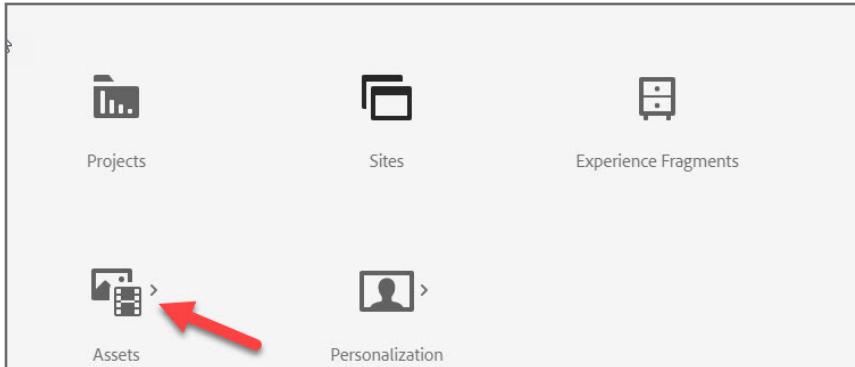
There are two main sections in the AEM UI: Navigation and Tools.



Navigation

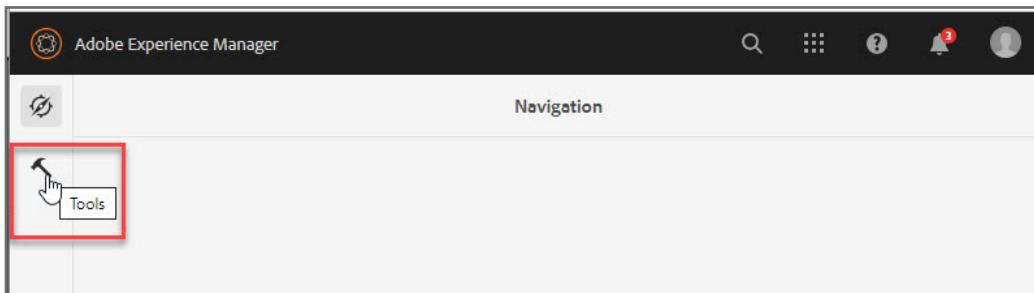
The Navigation section is represented by a compass icon in the AEM UI. By default, when you first sign in, the Navigation section loads in the content area. All applications of AEM, such as Projects, Sites, Experience Fragments, Assets, and Forms, are available in this section. This is also the main 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 as a Cloud Service

AEM Sites as a Cloud Service provides experience management capabilities as part of the cloud-native AEM as a Cloud Service platform. In addition to core benefits of AEM as a Cloud Service, such as cloud-native scalability, uptime, and always being up-to-date, AEM Sites as a Cloud Service also provides a number of Sites-specific changes and additions.

Changes and additions in AEM Sites as a Cloud Service are as follows:

- Asynchronous Page Operations
- New Reference Site and Tutorial

Asynchronous Page Operations

In AEM Cloud service, operations that traditionally have blocked the UI have been broken down into smaller tasks that run in the background.

- Move pages
- Roll-out pages

New Reference Site and Tutorial

WKND , a new AEM reference site, has been updated and published to reflect best practices for building a website with AEM, and the comprehensive set of capabilities, components, and deployment models that are available in AEM. The new reference site and accompanying tutorial covers fundamental topics like project setup, Core Components, Editable Templates, client libraries, and component development with Adobe Experience Manager 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

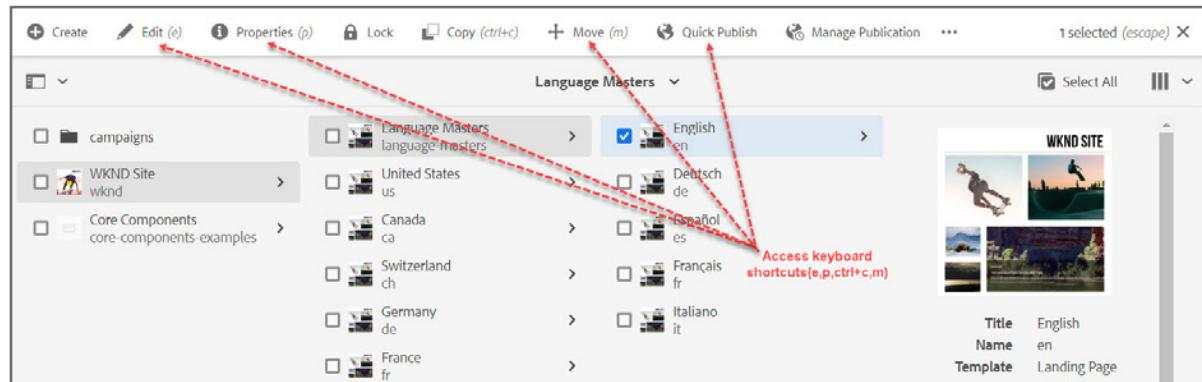
To create a page in AEM Sites, you need to follow a simple wizard where you specify the page template from a list of available templates, and then provide the page with a title and a name. The title is displayed to the user (who views the page in a browser) and the name is used to generate the page URL. 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

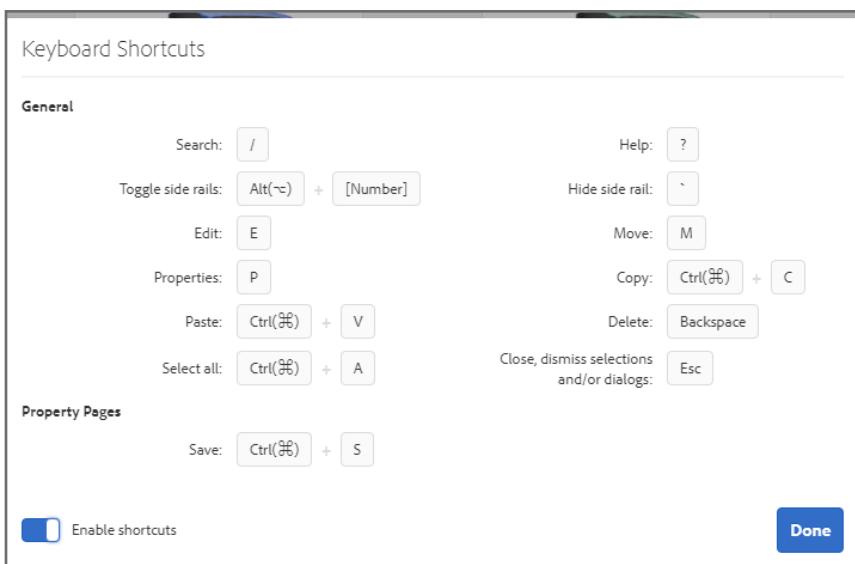
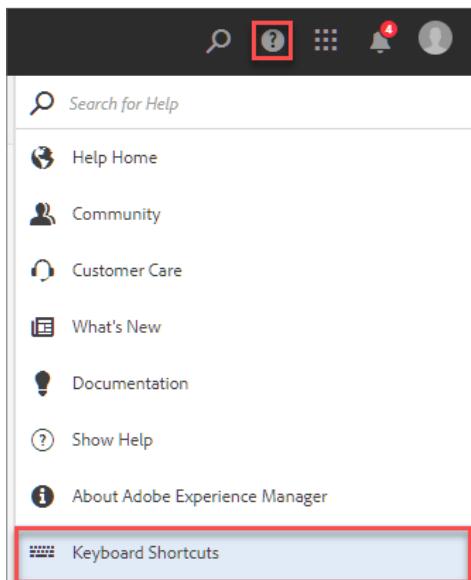
After you create a page, you can edit and add content to it. You can add content by dragging the components available in the side panel onto your page in the page editor.

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.

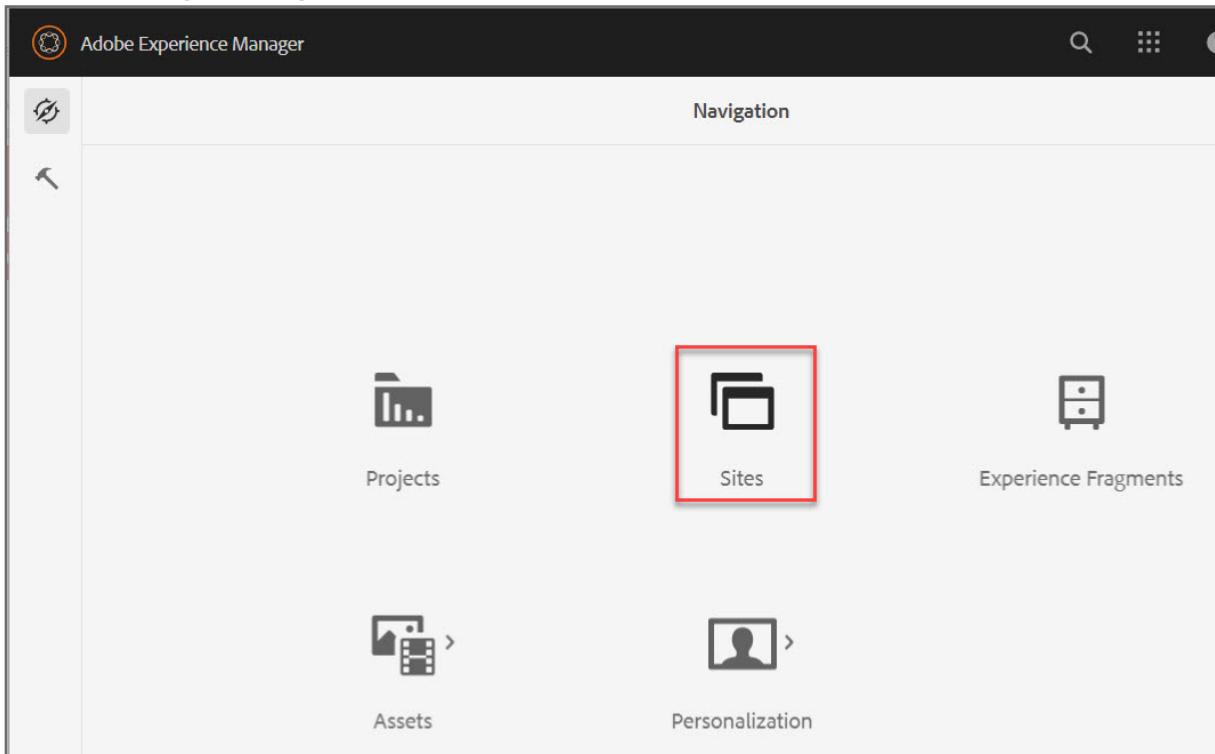
Exercise 1: Create a page in AEM

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.

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

The screenshot shows the Adobe Experience Manager interface in column view. The navigation path is as follows: Campaigns > Core Components > WKND Site > Language Masters > English. The 'WKND Site' node has a red box around its thumbnail and a red arrow pointing to the right next to it, indicating it was selected for navigation.

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.

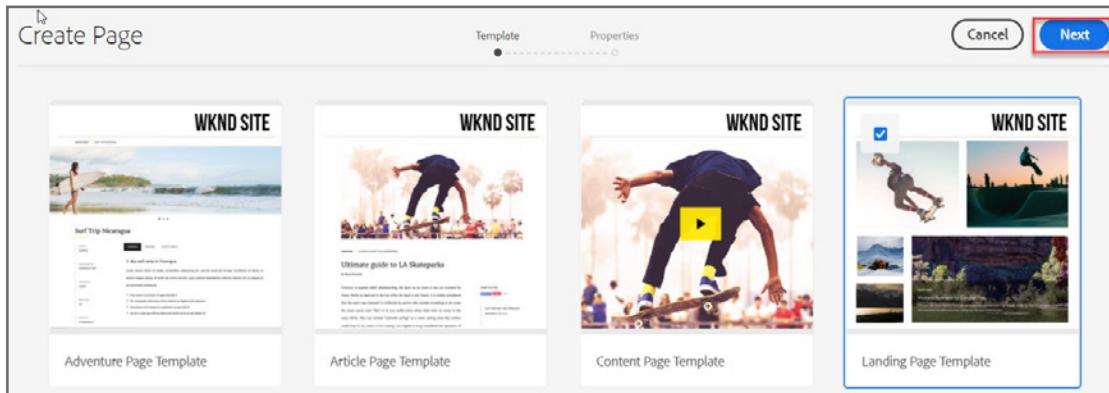
This screenshot shows the same navigation path as above, but the 'WKND Site' node now has a blue checkmark icon next to its thumbnail, indicating it is currently selected.

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

The screenshot shows the 'Create' button in the top right corner being clicked, which opens a dropdown menu. The 'Page' option in this menu is highlighted with a red box.

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**

Basic	Advanced	Social Media
Title and Tags Title * <input type="text" value="Demo Page"/> Name <input type="text"/> Tags <input type="checkbox"/>		

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.

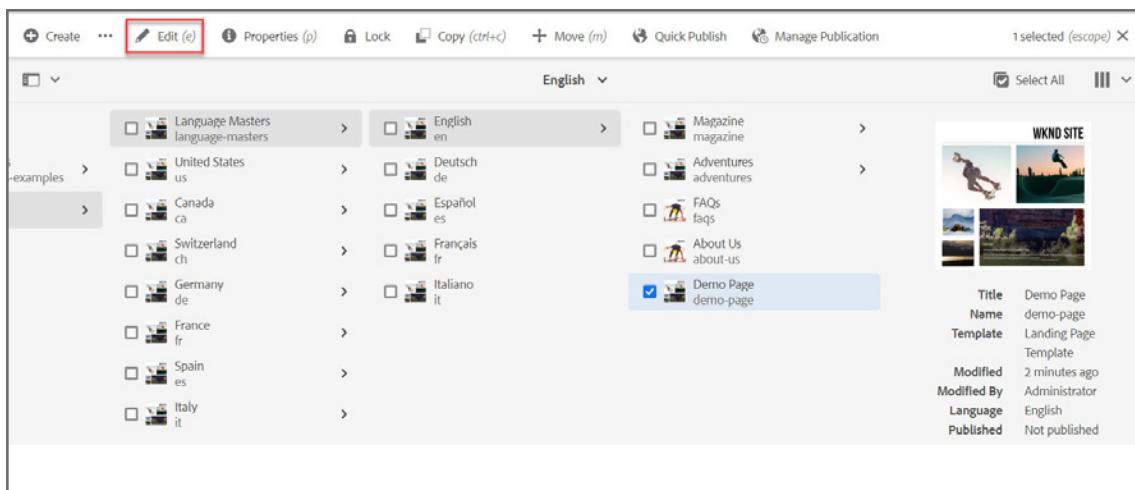
The screenshot shows the AEM Content Editor interface. The left sidebar lists site structures like 'campaigns', 'Core Components core-components-examples', and 'WKND Site'. The main area shows a tree structure for the 'English' language. Under 'Language Masters', there are nodes for 'United States', 'Canada', 'Switzerland', and 'Germany'. Under each of these are further language nodes: 'English en', 'Deutsch de', 'Espaniol es', 'Francais fr', and 'Italiano it'. On the far right, there are other nodes like 'Magazine magazine', 'Adventures adventures', 'FAQs faqs', 'About Us about-us', and 'Demo Page demo-page'. The 'Demo Page' node is highlighted with a red box.

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.



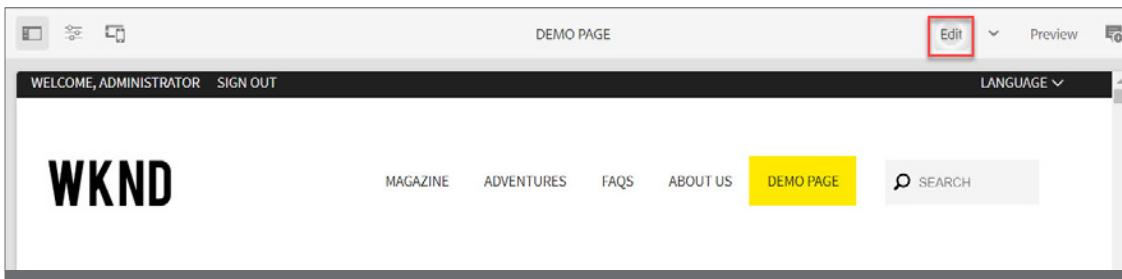
The screenshot shows the AEM Content Tree interface. On the left, there's a tree view of site structures under 'examples'. In the center, a detailed view of the 'Demo Page' node is shown. The node has a thumbnail image of a person riding a skateboard, a title 'WKND SITE', and several sub-components like 'Magazine', 'Adventures', 'FAQs', and 'About Us'. On the right, a properties panel displays the following details:

Title	Demo Page
Name	demo-page
Template	Landing Page Template
Modified	2 minutes ago
Modified By	Administrator
Language	English
Published	Not published

 **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:



The screenshot shows the AEM page editor interface. At the top, there's a toolbar with icons for 'Create', 'Edit (e)', 'Properties (p)', 'Lock', 'Copy (ctrl+c)', 'Move (m)', 'Quick Publish', and 'Manage Publication'. The 'Edit (e)' button is highlighted with a red box. Below the toolbar, the page content area shows the 'WKND' logo and a navigation menu with 'MAGAZINE', 'ADVENTURES', 'FAQS', 'ABOUT US', and 'DEMO PAGE'. The 'DEMO PAGE' item is highlighted with a yellow box. At the bottom right of the content area, there's a 'SEARCH' bar. The overall status bar at the bottom shows 'WELCOME, ADMINISTRATOR' and 'SIGN OUT'.

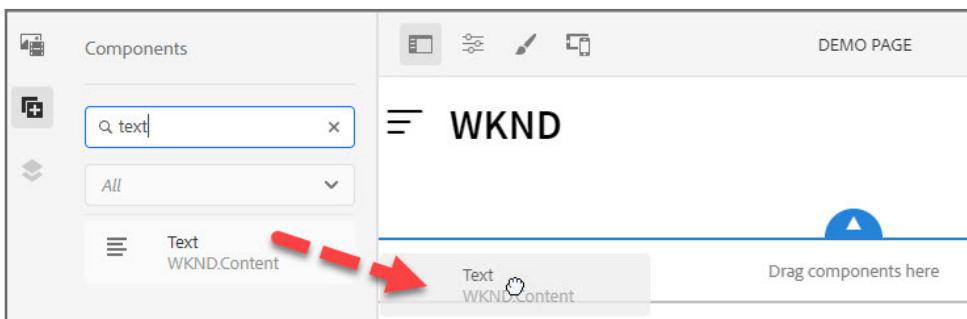
To add a text component to the page:

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

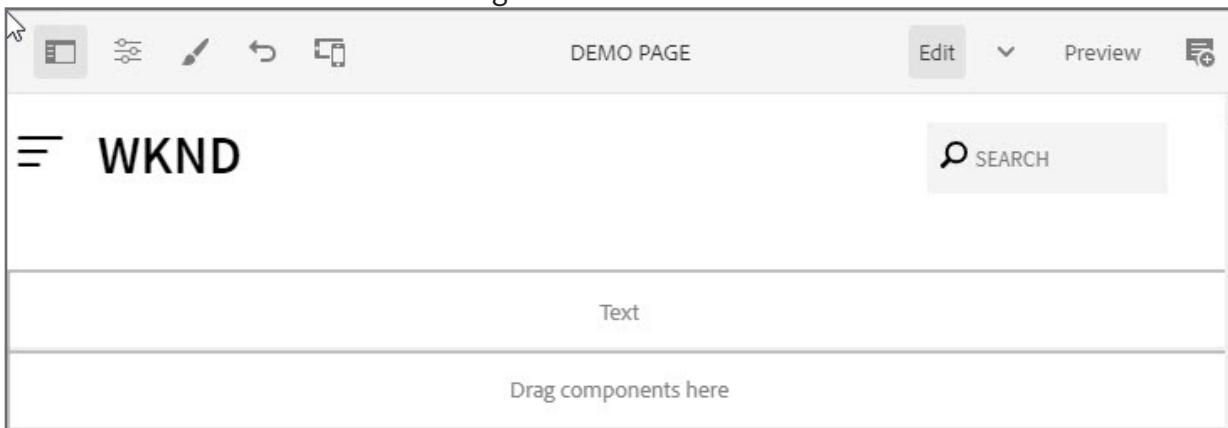


The side panel is displayed.

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:

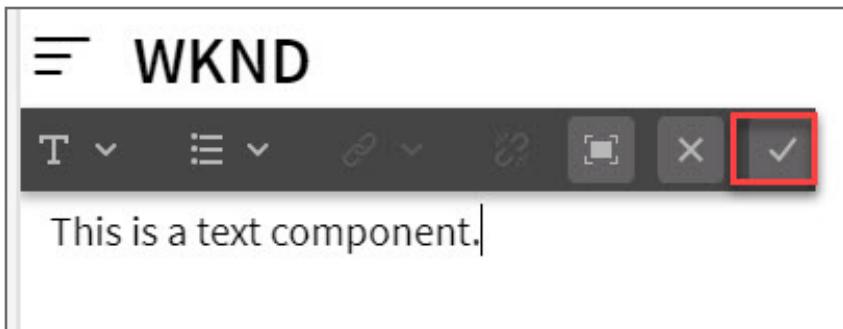


Tip: If the ordering is not correct, just drag the text component again to the **Drag components here** box.

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.



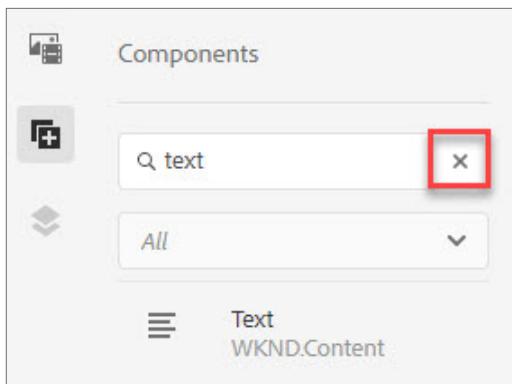
7. Add sample text of your choice in the text editing form that appears and click **Done** (checkmark icon) to save the changes. The text is added to the page.



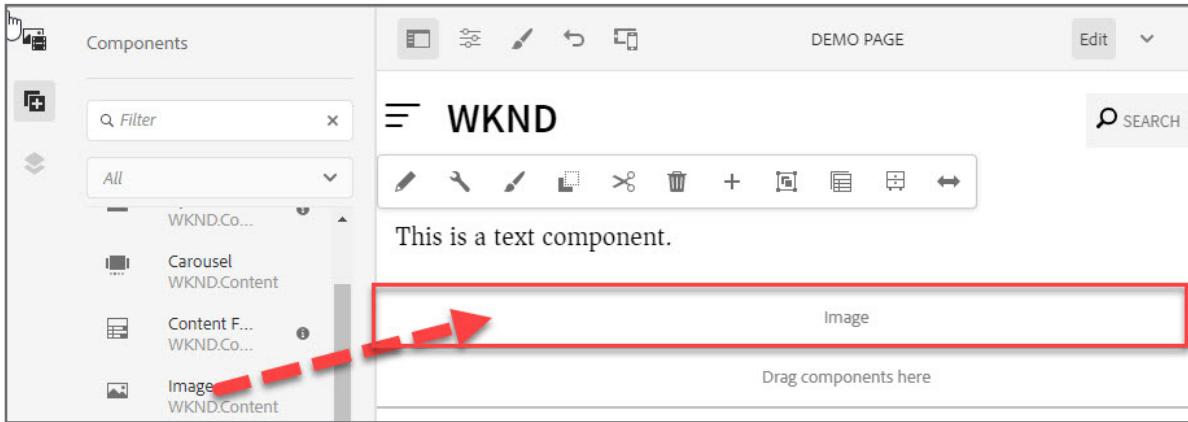
You have now added the first component to your page!

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:

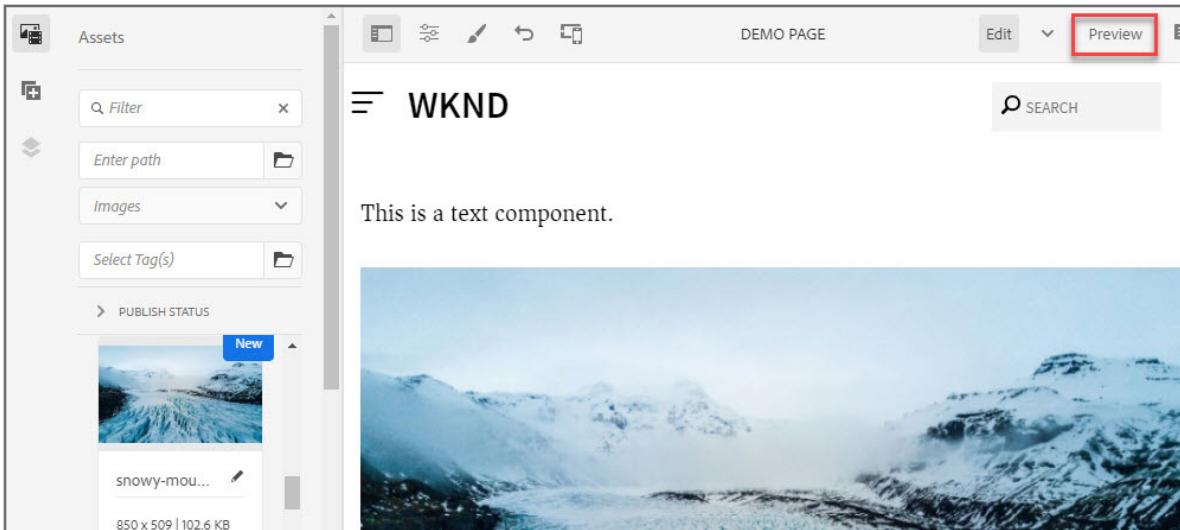


10. Click the **Assets** icon in the side panel as shown:



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 that 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.

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>

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

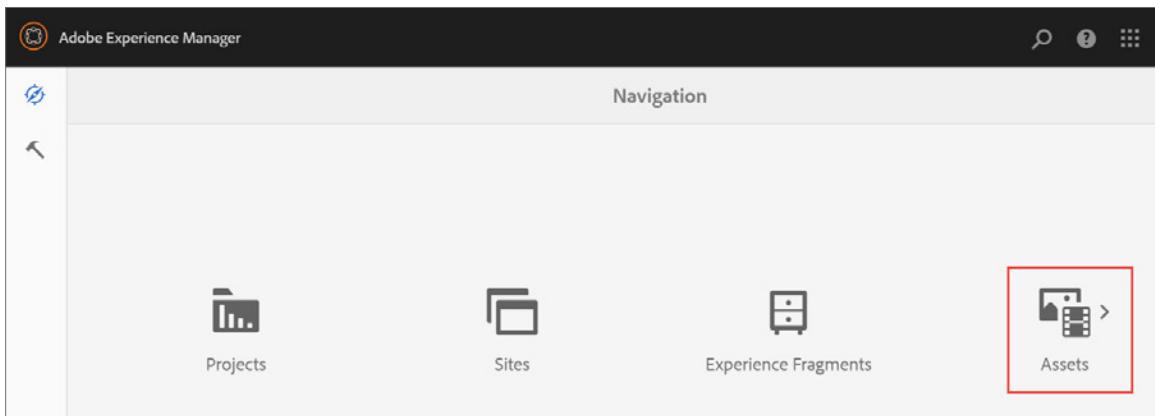
After completing this course, you will be able to:

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

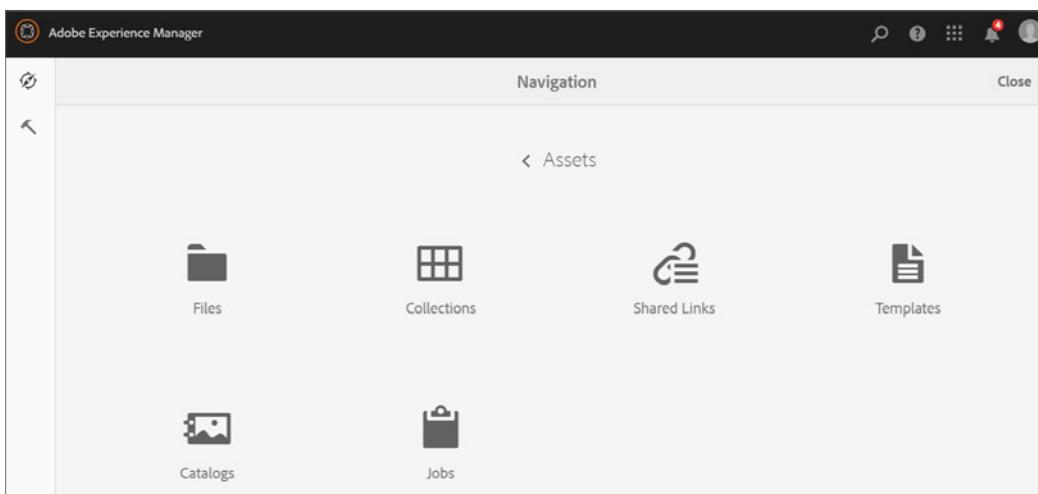
Assets Console

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, Templates, Catalogs, and Jobs, as shown:

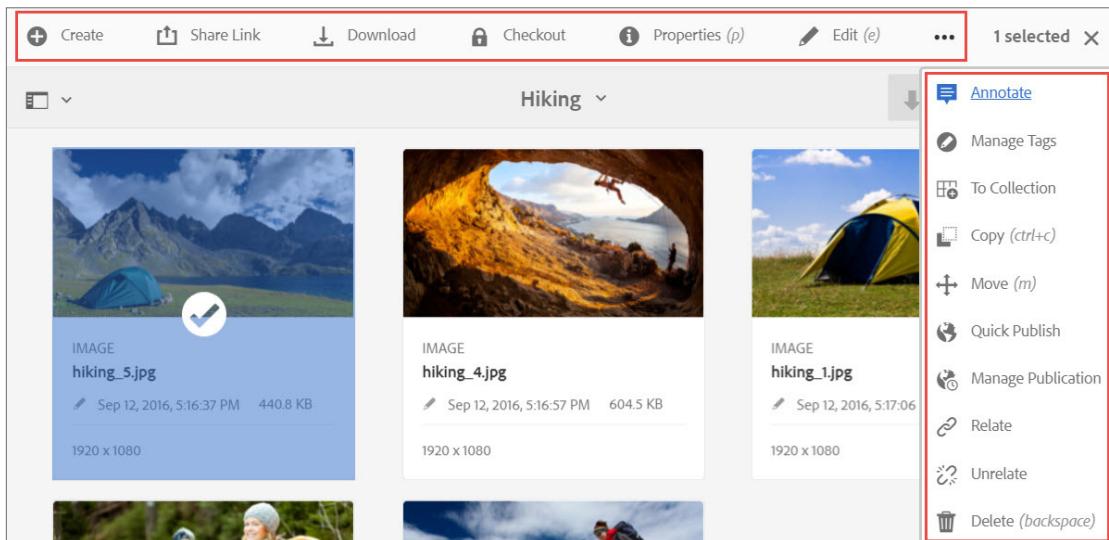


The following table describes the subconsoles of the Assets console and their purposes:

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
Templates	Contains the templates of different print media, such as brochures, flyers, and postcards
Catalogs	Helps create product catalogs based on assets and product information
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:

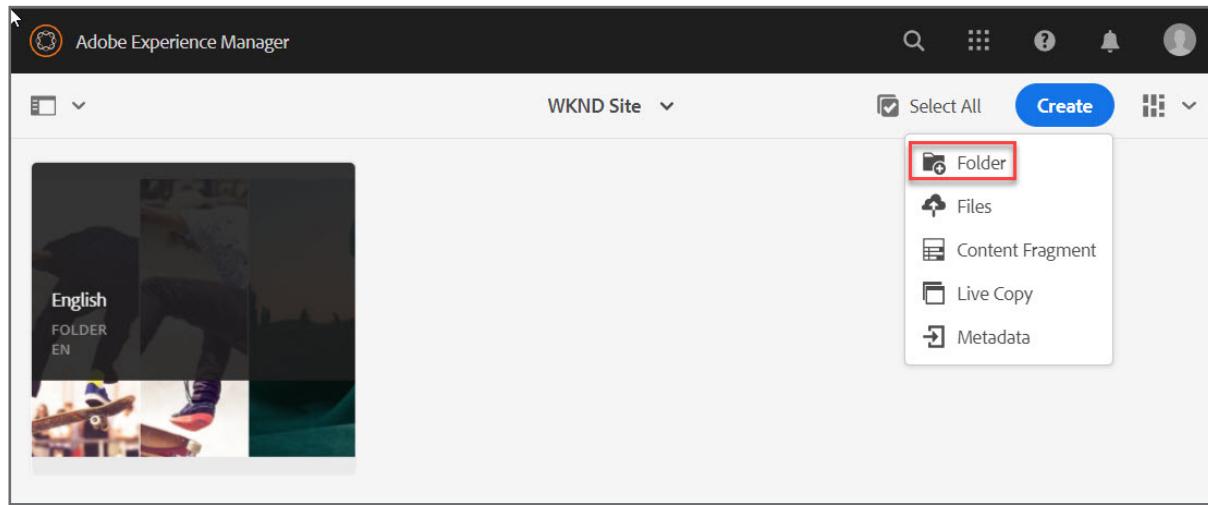
- Create public or private folders
- Organize and group asset folders related to a specific task
- Create asset collections, so all related assets are available in one directory
- Assign different tasks to users that they can perform on asset folders based on privilege levels

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

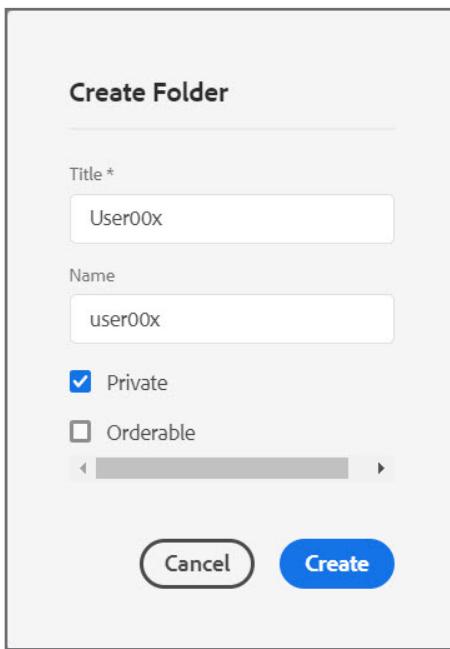
Private Folder

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.



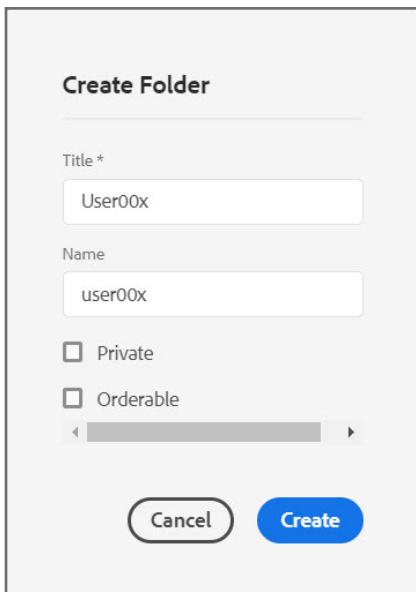
Note: Administrators can access the private folders created by all AEM users.

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.



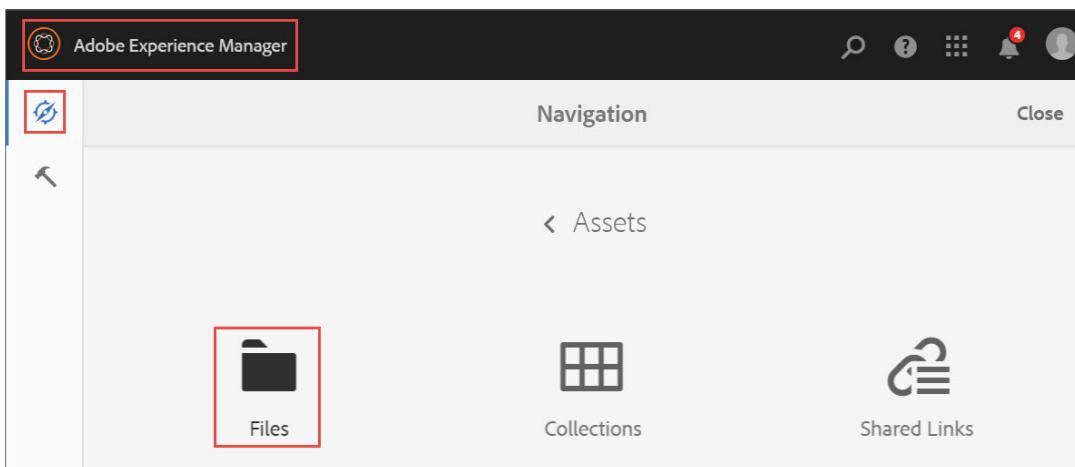
Note: To upload asset folders from your local folder system through the AEM Desktop app, you must enable and configure the Desktop app with AEM Assets.

Exercise 1: Create a folder and upload assets to it

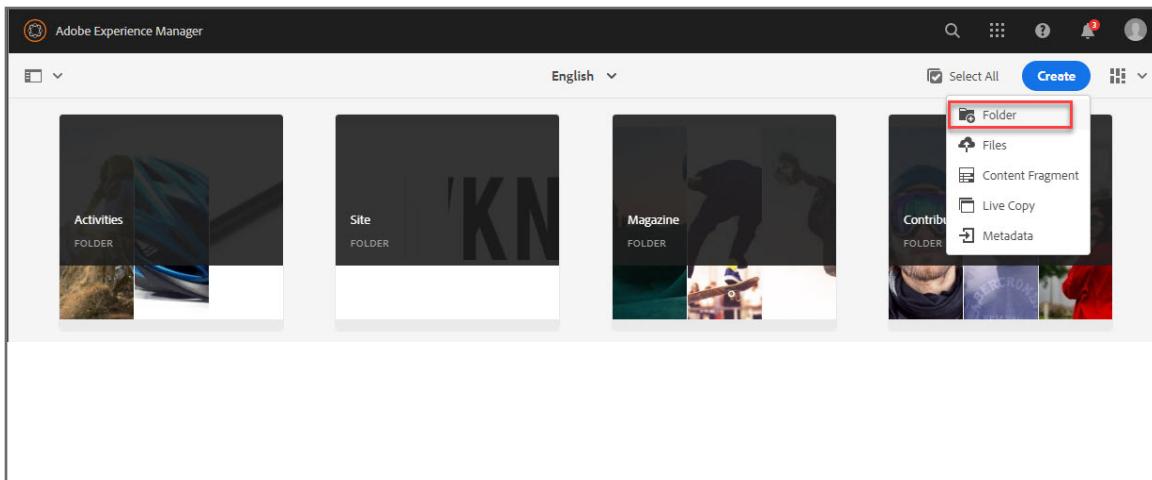
For performing this exercise, you will use the WKND site asset folder that comes with AEM.

To create a folder and upload assets to it:

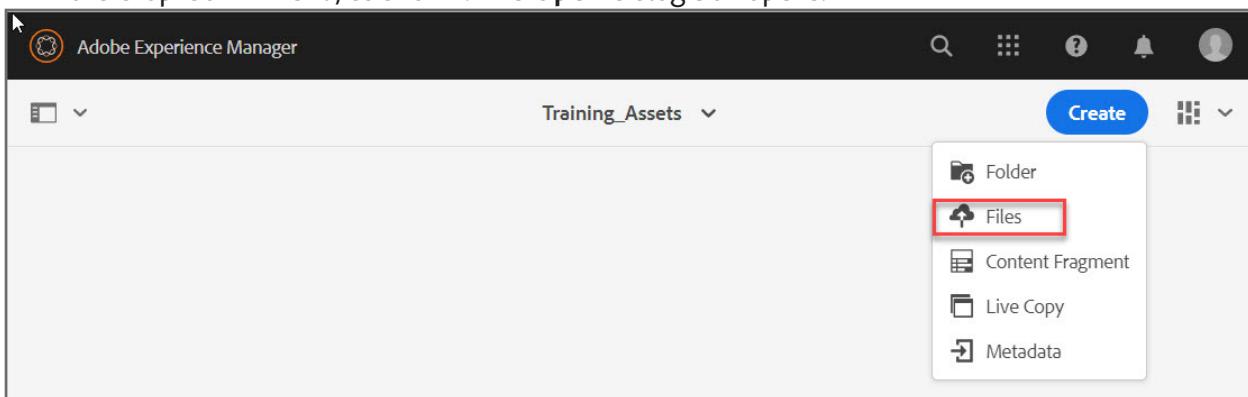
1. Go back to the tab in your browser that has the AEM author service open.
2. Click the **Adobe Experience Manager** icon.
3. Click **Navigation > Assets**, and then click **Files**, as shown. If a **Product Navigation** dialog box appears, click **Don't show this again**, and then click **Close**,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



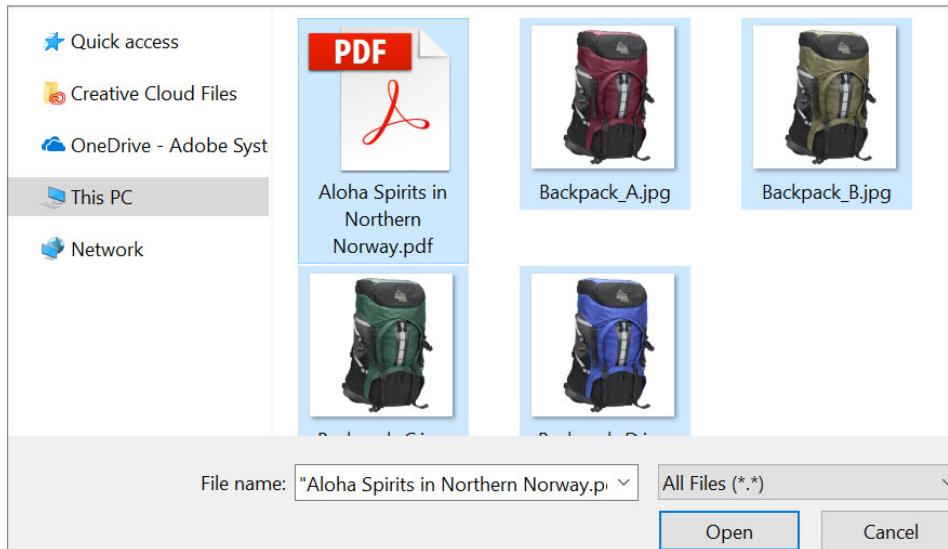
4. Navigate to **WKND Site > English**, and then click **Create** from the actions bar, and select **Folder** from the drop-down menu as shown. The **Create Folder** dialog box opens.



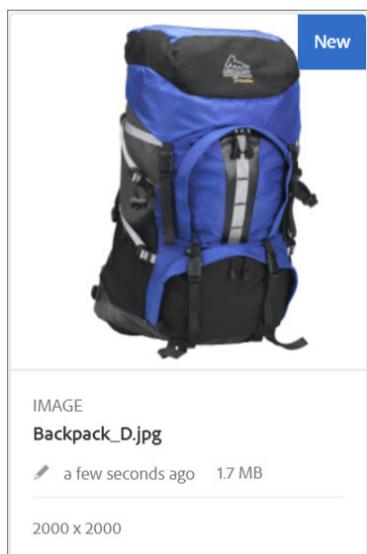
5. Type **Training_Assets** as the title of the folder. The name of the folder is automatically populated from the title.
6. Select **Private** to ensure only you have access to this folder but do not select the **Orderable** checkbox.
7. Click **Create**, as shown. Notice how the **Training_Assets**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

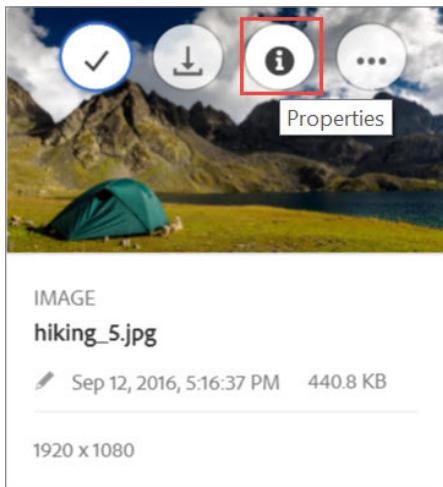
A good amount of metadata helps you further categorize assets. As metadata is added to assets, the asset becomes more accessible and easier to manage. For example, you can find assets with the same properties and apply changes to them easily.

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.

A screenshot of the AEM Assets interface showing the properties screen for a selected asset. The top navigation bar includes "Reprocess Assets", "Find Similar", "Create", "Checkout", "Properties (p)" (highlighted with a red box), and "Select All". The main area displays a list of assets under the heading "Skitouring". One asset is selected, showing its details: "Sofia Sjöberg" with the description "My backpack feels heavier for every meter, slowly deforming my spine and wearing down my shoulders. My mouth tastes like blood, while salty pearls of sweat find their way to my eyes. I...". Another asset, "Skitouring", is listed as a "CONTENT FRAGMENT". On the right, a preview image of a snowy mountain slope is shown with the file name "Skitouring3Sjoeberg.JPG", type "IMAGE", and details "Oct 1, 2019, 3:53:18 PM 88.6 KB".

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.

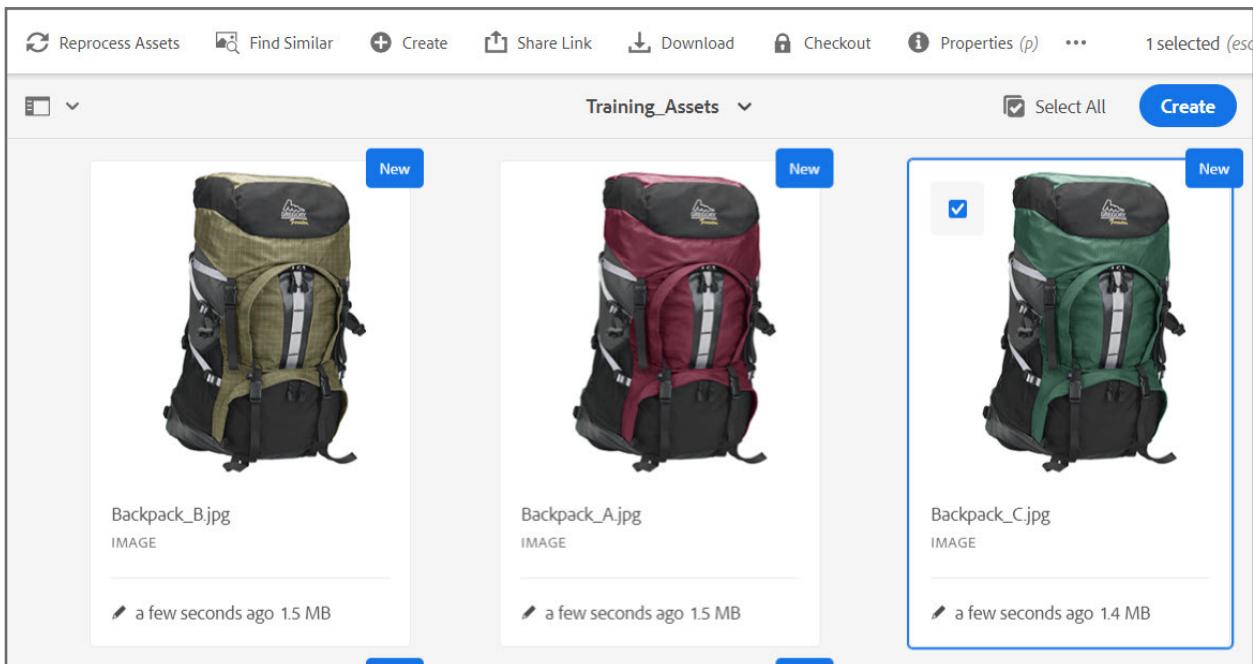
There are many ways to embed metadata into files. AEM supports the following encoding standards:

- 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.

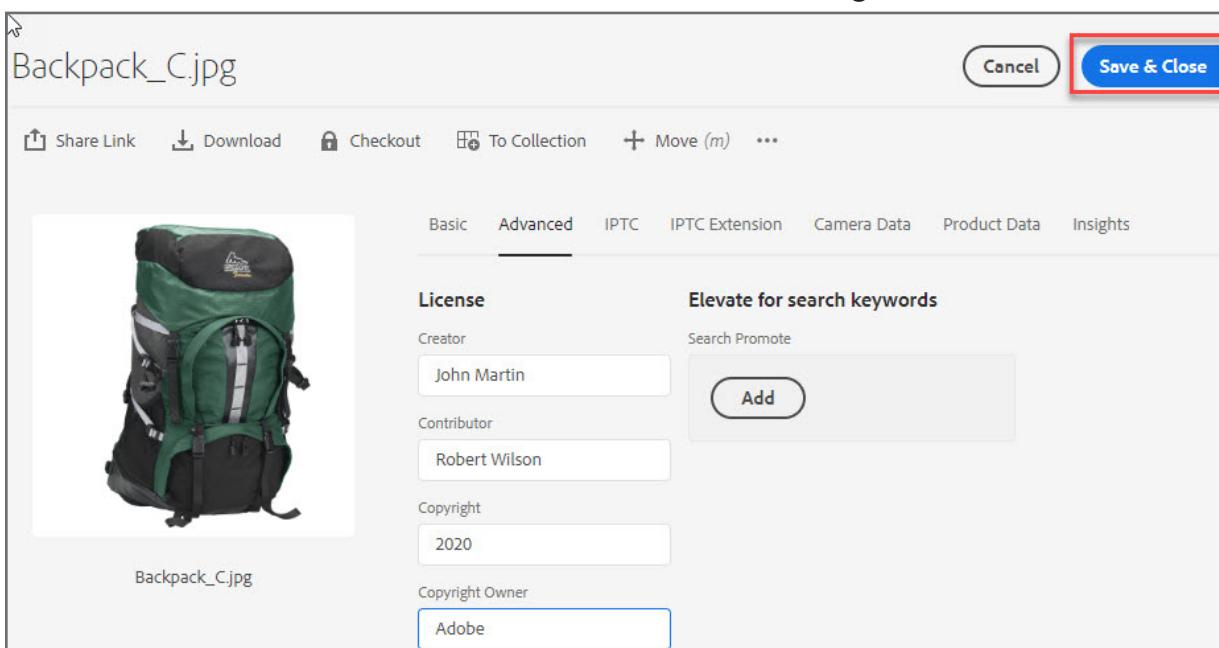
Exercise 2: Add metadata to an asset

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.

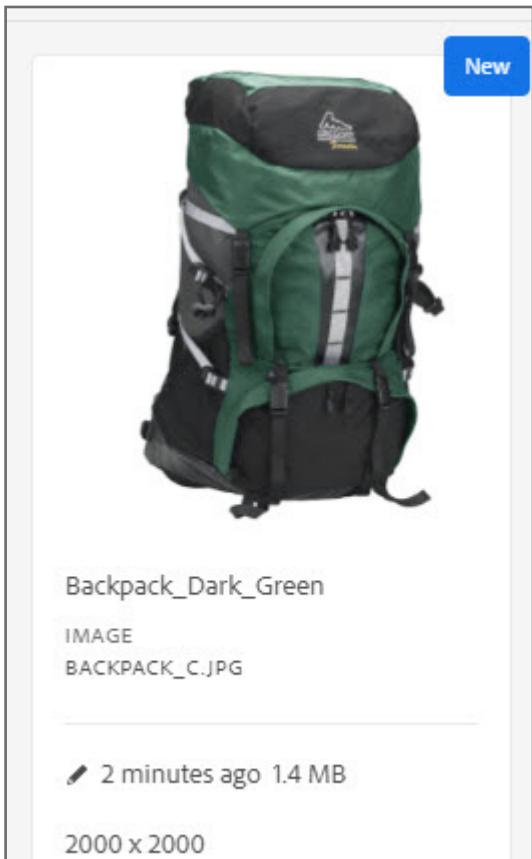


5. On the **Basic** tab, type the following details, as shown:
 - a. Title: **Backpack_Dark_Green**
 - b. Description: **Backpacks for tough running experience**
 - c. Select **English (United States)** from the **Language** drop-down menu.
6. On the **Advanced** tab:
 - a. In the **License** section,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



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 Maven command
- Install Node.js
- Install NPM
- 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.

(Optional) Exercise 1: Install build tools

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. Also, if you do not want to install Maven and create your own project, your instructor can provide a pre-created maven project for you to use during the course. If you are using ReadyTech, this project can be found in the Eclipse workspace under the Sample Projects folder.

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/](https://crunchify.com/how-to-install-maven-on-mac-os-x-manually-fix-unsupported-class-version-error-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>/.

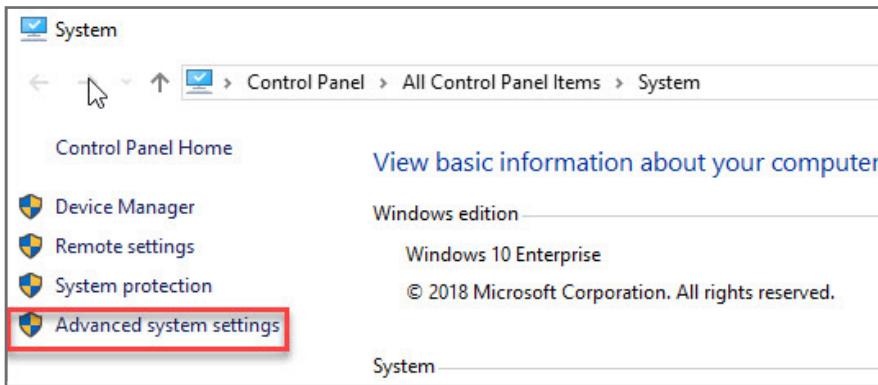
```
wget http://mirrors.koehn.com/apache/maven/maven-3/3.6.1/binaries/
apache-maven-3.6.1-bin.zip
```

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:

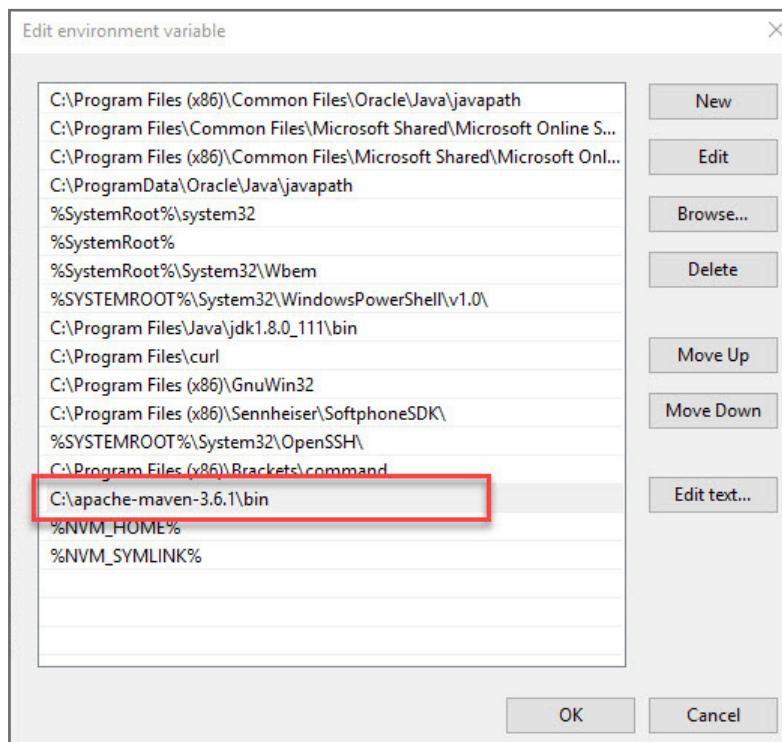


The **System Properties** window opens.

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 exercise.

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.

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

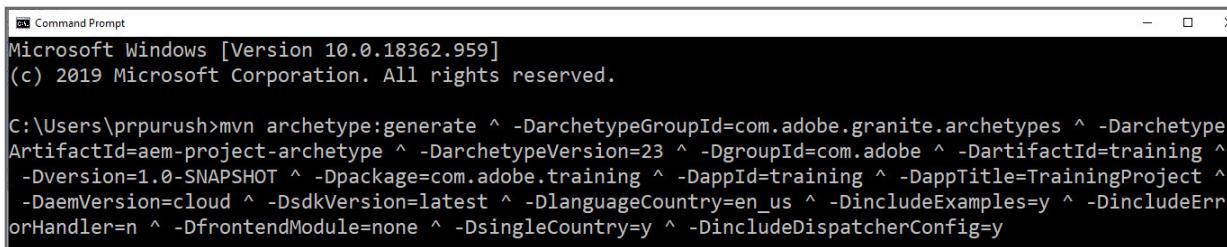
1. Open a **Command Prompt** window by right-clicking the **Command Prompt** app/icon and choose as **Run as administrator**.

 Note: In order to create the AEM project in step 2, you can use the create script provided in the **Exercise_Files_TB** folder or you can use the create script specified by your instructor.

2. Navigate to **Exercise_Files_TB\Project_using_Maven** and copy the following content from the file **create-cloud-project-23.bat** on Windows (For Mac, use **create-cloud-project-23.sh**):

```
svn archetype:generate ^
-DarchetypeGroupId=com.adobe.granite.archetypes ^
-DarchetypeArtifactId=aem-project-archetype ^
-DarchetypeVersion=23 ^
-DgroupId=com.adobe ^
-DartifactId=training ^
-Dversion=1.0-SNAPSHOT ^
-Dpackage=com.adobe.training ^
-DappId=training ^
-DappName=TrainingProject ^
-DaemVersion=cloud ^
-DsdkVersion=latest ^
-DlanguageCountry=en_us ^
-DincludeExamples=y ^
-DincludeErrorHandler=n ^
-DfrontendModule=none ^
-DsingleCountry=y ^
-DincludeDispatcherConfig=y|
```

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

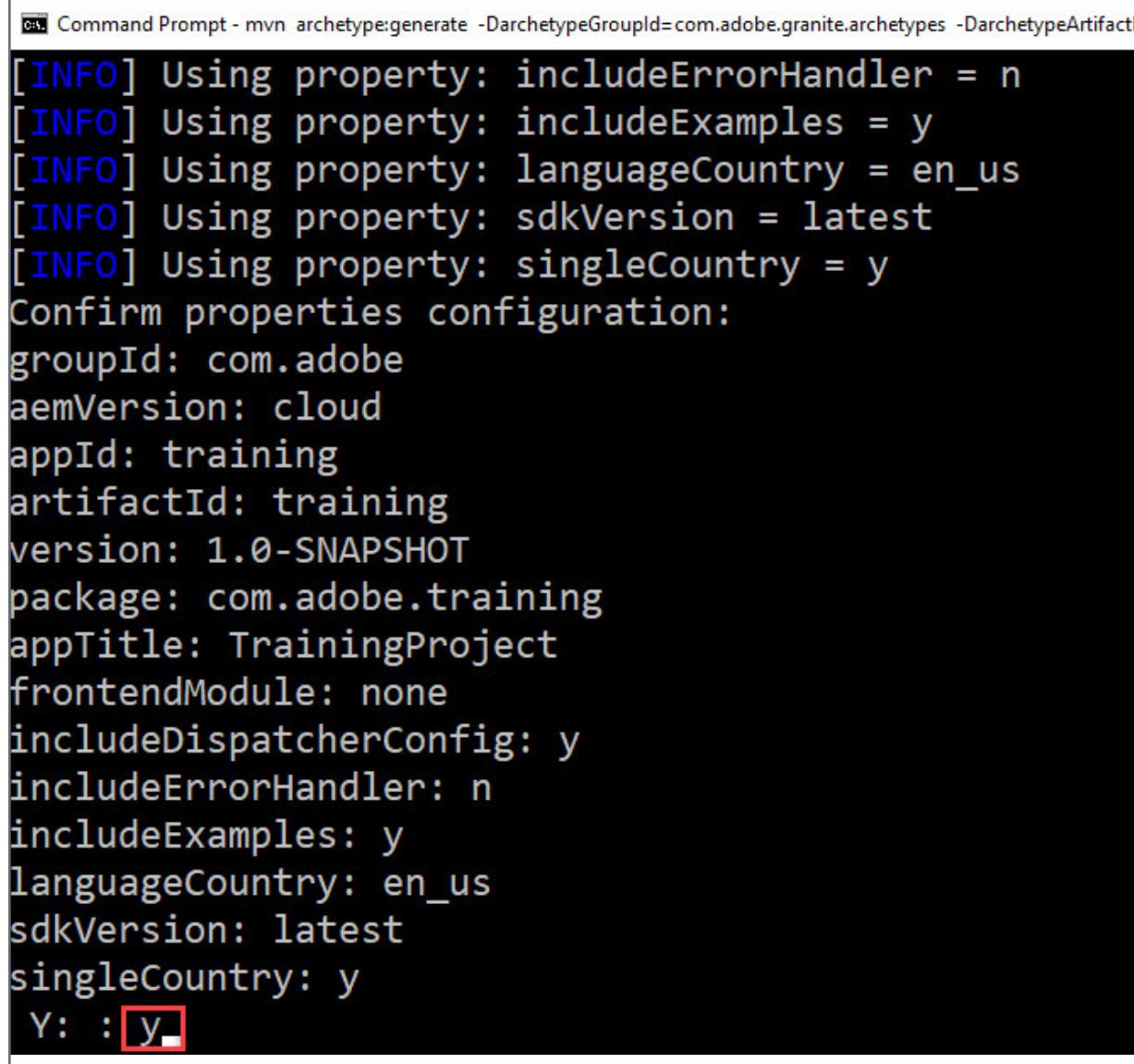


```
PS C:\Users\prpurush>mvn archetype:generate ^ -DarchetypeGroupId=com.adobe.granite.archetypes ^ -DarchetypeArtifactId=aem-project-archetype ^ -DarchetypeVersion=23 ^ -DgroupId=com.adobe ^ -DartifactId=training ^ -Dversion=1.0-SNAPSHOT ^ -Dpackage=com.adobe.training ^ -DappId=training ^ -DappTitle=TrainingProject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

The command runs, which may take a few minutes, and asks for confirmation at the end.

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

4. Type Y and press **Enter** to confirm the settings, as shown:



```
[INFO] Using property: includeErrorHandler = n
[INFO] Using property: includeExamples = y
[INFO] Using property: languageCountry = en_us
[INFO] Using property: sdkVersion = latest
[INFO] Using property: singleCountry = y
Confirm properties configuration:
groupId: com.adobe
aemVersion: cloud
appId: training
artifactId: training
version: 1.0-SNAPSHOT
package: com.adobe.training
appTitle: TrainingProject
frontendModule: none
includeDispatcherConfig: y
includeErrorHandler: n
includeExamples: y
languageCountry: en_us
sdkVersion: latest
singleCountry: y
Y: :y.
```

The project starts to build.

5. Verify Build Success, as shown:

```

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
[TINFO] -----

```

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

Name	Date modified
.settings	3/18/2020 10:45
all	3/18/2020 10:54
core	3/18/2020 10:53
it.launcher	3/18/2020 10:54
it.tests	3/18/2020 10:54
ui.apps	3/18/2020 10:53
ui.apps.structure	3/18/2020 10:53
ui.content	3/18/2020 10:54
.gitignore	3/18/2020 10:13
.project	3/18/2020 10:44
pom.xml	3/18/2020 10:13

Congratulations! You have successfully created an AEM project.

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:

- Language Master / Country structure
 - › A language master structure and country top-level page
 - › Live copy of the top-level page in the specified language
 - › Blueprint
- Content and Experience fragment templates based on the editable template feature
 - › Example content policy
- Page component
 - › Based on the AEM Sites Core Component page
 - › customfooterlibs.html and customheaderlibs.html snippet to load additional JS and CSS clientlibs according to the {cssId} property
- Content Components
 - › Proxy components for all AEM Sites Core Components of the latest released version. For instance accordion, carousel, text and much more. For a complete overview see AEM Sites Core Components
 - › Example: helloworld example of custom HTL component with Sling Models for the logic
- Frontend module
 - › Build process based on Webpack with support for Sass and TypeScript / ES6
 - › Automatic AEM Clientlib generation
 - › CSS class names follow BEM naming conventions
 - › Component-specific styles stored within each component
- Configurations
 - › Device emulators displayed in the authoring interface
 - › Allow direct drag & drop of assets from the content finder into layout container (6.3 TouchUI)
 - › Dictionary structure for internationalizing hardcoded strings
- Bundle with some examples
 - › Models: Models for more complex business logic of components

- › Servlets: Rendering the output of specific requests
- › Filters: Applied to the requests before dispatching to the servlet or script
- › Schedulers: Cron-job like tasks
- Tests
 - › Unit tests
 - › Integration tests
 - › Client-side Hobbes tests within developer mode

Archetype Explanation:

- all - A container package that includes the ui.apps and ui.content packages as embeds
- ui.apps - Contains all the immutable code in /apps and embeds the OSGi bundle
- core - Contains the OSGi bundle
- ui.content - Contains all mutable content and configuration
- it.launcher - Used for deploying the sling testing framework
- it.tests - Contains sling tests
- ui.apps.structure - Cloud specific module that contains project roots
- dispatcher - (optional) Module used to configure the static web server in front of AEM that caches content
- ui.frontend - (optional) Module that contains the artifacts required to use the Webpack-based front-end build module.

AEM as a Cloud Service Project's Content Package Structure

In this chapter, you will learn about the changes required to Experience Manager projects in order to make Experience Manager as a Cloud Service compatible, ensuring the projects respect the split of mutable and immutable content, requisite dependencies are established to create non-conflicting, deterministic deployments, and are packaged in a deployable structure.

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

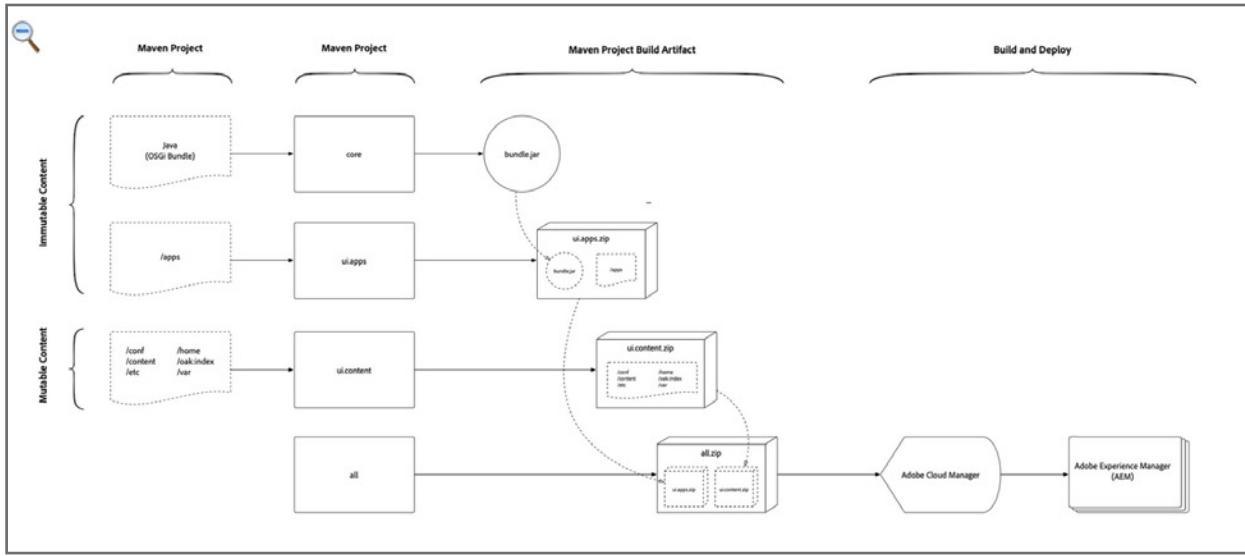
/apps and /libs are considered immutable areas of Experience Manager as they cannot be changed (create, update, delete) after deployment into the AEM Cloud Service (for example, at runtime). Any attempt to change an immutable area at runtime will fail.

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:



The recommended application deployment structure is as follows:

- The **ui.apps** package, or Content 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:
 - › OSGi bundles
 - » **/apps/my-app/install**
 - › OSGi configurations
 - » **/apps/my-app/config**
 - › HTML scripts
 - » **/apps/my-app/components**
 - › JavaScript and CSS (via Client Libraries)
 - » **/apps/my-app/clientlibs**
 - › Overlays of **/libs**
 - » **/apps/cq, /apps/dam/, and so on**
 - › Fallback context-aware configurations
 - » **/apps/settings**
 - › ACLs (permissions)
 - » Any **rep:policy** for any path under

- The ui.content package, or Code Package, contains all content and configuration. Common elements of the ui.content package include, but are not limited to:
 - › Context-aware configurations
 - » /conf
 - › Baseline content structures (Asset folders, Sites root pages)
 - » /content,/content/dam, and so on
 - › Governed tagging taxonomies
 - » /content/cq:tags
 - › Service users
 - » /home/users
 - › User groups
 - » /home/groups
 - › Oak indexes
 - » /oak:indexes
 - › Etc legacy nodes
 - » /etc
 - › ACLs (permissions)
 - » Any rep:policy for any path not under /apps
- 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.

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

AEM Java SDK

The AEM Java SDK is a special JAR file provided by Adobe to reduce the number of dependencies.

The SDK 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

Using the AEM Java SDK File

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>
```

- Core components dependency element

```
<dependency>
    <groupId>com.adobe.cq</groupId>
    <artifactId>core.wcm.components.core</artifactId>
    <version>${core.wcm.components.version}</version>
</dependency>
<dependency>
    <groupId>com.adobe.cq</groupId>
    <artifactId>core.wcm.components.content</artifactId>
    <type>zip</type>
    <version>${core.wcm.components.version}</version>
</dependency>
<dependency>
    <groupId>com.adobe.cq</groupId>
    <artifactId>core.wcm.components.config</artifactId>
    <type>zip</type>
    <version>${core.wcm.components.version}</version>
</dependency>
<dependency>
    <groupId>com.adobe.cq</groupId>
    <artifactId>core.wcm.components.examples</artifactId>
    <type>zip</type>
    <version>${core.wcm.components.version}</version>
</dependency>
```

- Core components version used:

```
<properties>
    <aem.host>localhost</aem.host>
    <aem.port>4502</aem.port>
    <aem.publish.host>localhost</aem.publish.host>
    <aem.publish.port>4503</aem.publish.port>
    <sling.user>admin</sling.user>
    <sling.password>admin</sling.password>
    <vault.user>admin</vault.user>
    <vault.password>admin</vault.password>
    <core.wcm.components.version>2.7.0</core.wcm.components.version>
    <bnd.version>4.2.0</bnd.version>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
</properties>
```

Maven Profiles

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

Training (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

training.all (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.

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:

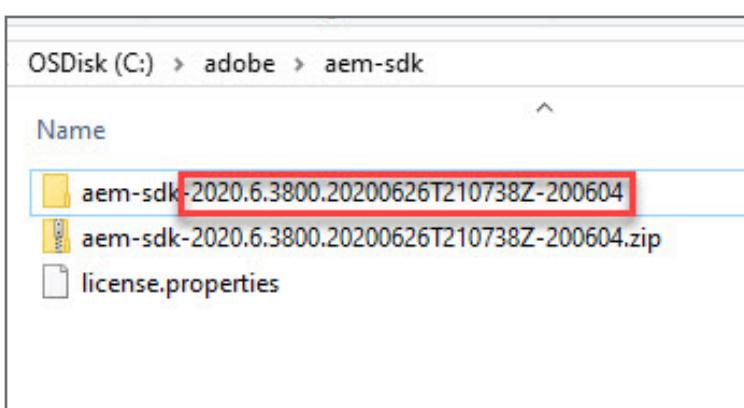
1. Update the POM file
2. Install the project into AEM
3. Verify the installed content packages

Task 1: Update the POM file

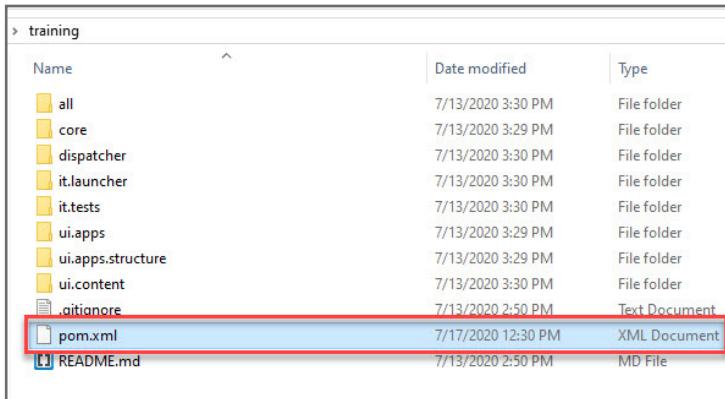
When you create a new cloud project from the archetype, it automatically adds the latest AEM SDK build to your parent POM file. When developing your application, you want to make sure the build in your POM matches the build of the quickstart jar you're using. Depending how often you download and update the quickstart jar, you might need to update the build in your POM.

In this task you will update the build in your POM to match the quickstart jar build to reduce the risk of dependency mismatching.

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**:



Name	Date modified	Type
all	7/13/2020 3:30 PM	File folder
core	7/13/2020 3:29 PM	File folder
dispatcher	7/13/2020 3:30 PM	File folder
it.launcher	7/13/2020 3:30 PM	File folder
it.tests	7/13/2020 3:30 PM	File folder
ui.apps	7/13/2020 3:29 PM	File folder
ui.apps.structure	7/13/2020 3:29 PM	File folder
ui.content	7/13/2020 3:30 PM	File folder
.gitignore	7/13/2020 2:50 PM	Text Document
pom.xml	7/17/2020 12:30 PM	XML Document
README.md	7/13/2020 2:50 PM	MD File

4. Open the POM file using a text editor (such as Notepad++) and paste the build number that you copied from the SDK zip file in step 2, as shown:



```

<properties>
    <aem.host>localhost</aem.host>
    <aem.port>4502</aem.port>
    <aem.publish.host>localhost</aem.publish.host>
    <aem.publish.port>4503</aem.publish.port>
    <sling.user>admin</sling.user>
    <sling.password>admin</sling.password>
    <vault.user>admin</vault.user>
    <vault.password>admin</vault.password>
    <core.wcm.components.version>2.8.0</core.wcm.components.version>
    <bnd.version>5.0.0</bnd.version>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
    <aem.sdk.api>2020.6.3800.20200626T210738Z-200604</aem.sdk.api>
</properties>

```

5. Save the file.

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
```

```
C:\Users\prpurush\training>mvn clean install -Padobe-public -PautoInstallSinglePackage
```



Note: The project build may take a few minutes.

3. Verify the build is success, as shown:

```
[INFO] --- maven-surefire-plugin:2.22.1:test (default-test) @ training.it.launcher ---
[INFO]
[INFO] --- maven-jar-plugin:3.1.2:jar (default-jar) @ training.it.launcher ---
[INFO] Building jar: C:\Users\prpurush\training\it.launcher\target\training.it.launcher-1.0-SNAPSHOT.jar
[INFO]
[INFO] --- maven-install-plugin:2.5.2:install (default-install) @ training.it.launcher ---
[INFO] Installing C:\Users\prpurush\training\it.launcher\target\training.it.launcher-1.0-SNAPSHOT.jar to C:\Users\sh\.m2\repository\com\adobe\training\it.launcher\1.0-SNAPSHOT\training.it.launcher-1.0-SNAPSHOT.jar
[INFO] Installing C:\Users\prpurush\training\it.launcher\pom.xml to C:\Users\prpurush\.m2\repository\com\adobe\tra
it.launcher\1.0-SNAPSHOT\training.it.launcher-1.0-SNAPSHOT.pom
[INFO] -----
[INFO] Reactor Summary for training 1.0-SNAPSHOT:
[INFO]
[INFO] training ..... SUCCESS [ 2.214 s]
[INFO] TrainingProject - Core ..... SUCCESS [ 22.096 s]
[INFO] TrainingProject - Repository Structure Package ..... SUCCESS [ 1.526 s]
[INFO] TrainingProject - UI apps ..... SUCCESS [ 29.971 s]
[INFO] TrainingProject - UI content ..... SUCCESS [ 23.972 s]
[INFO] TrainingProject - All ..... SUCCESS [ 5.657 s]
[INFO] TrainingProject - Integration Tests Bundles ..... SUCCESS [ 2.294 s]
[INFO] TrainingProject - Integration Tests Launcher ..... SUCCESS [ 6.532 s]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 01:42 min
[INFO] Finished at: 2019-11-26T10:02:08-08:00
[INFO] -----
```

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:

The screenshot shows the CRX Package Manager interface. On the left, there are filters for 'Sort by' (Last used), 'Show' (All packages), and 'Groups' (All packages (114)). The main area lists three packages:

- training.all-1.0-SNAPSHOT.zip**: Version: 1.0-SNAPSHOT | Last installed 10:01 | admin. Description: All content package for TrainingProject. Share button: 4.6 MB.
- training.ui.content-1.0-SNAPSHOT.zip**: Version: 1.0-SNAPSHOT | Last installed 10:01 | admin. Description: UI content package for TrainingProject. Share button: 265.4 KB.
- training.ui.apps-1.0-SNAPSHOT.zip**: Version: 1.0-SNAPSHOT | Last installed 10:01 | admin. Description: UI apps package for TrainingProject. Share button: 338.2 KB.

A red box highlights the three packages listed in the center.

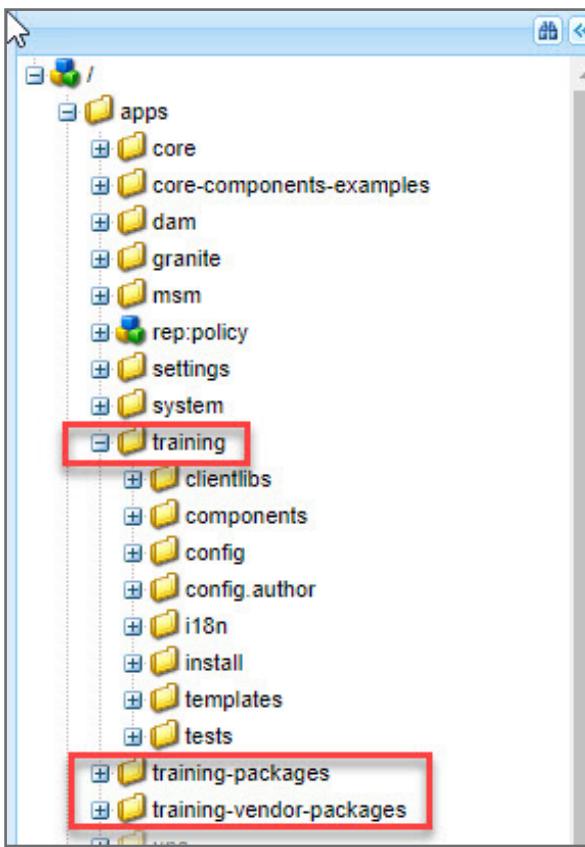
Note: With Archetype 23, the install has organized the content into mutable and immutable content.

4. Click the Develop icon as shown:

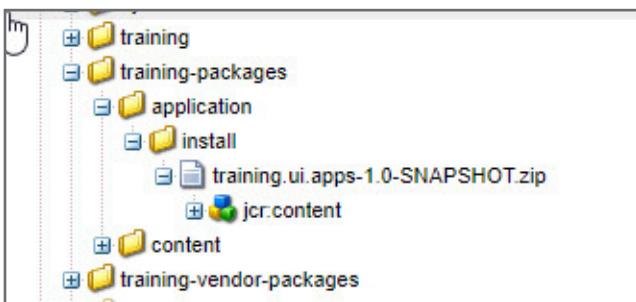
The screenshot shows the CRX Package Manager interface. The top toolbar includes icons for 'Search packages', 'Create Package', 'Upload Package', and a 'Develop' icon (highlighted with a red box). Below the toolbar, there are filters for 'Sort by' (Last used) and a search bar.

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:

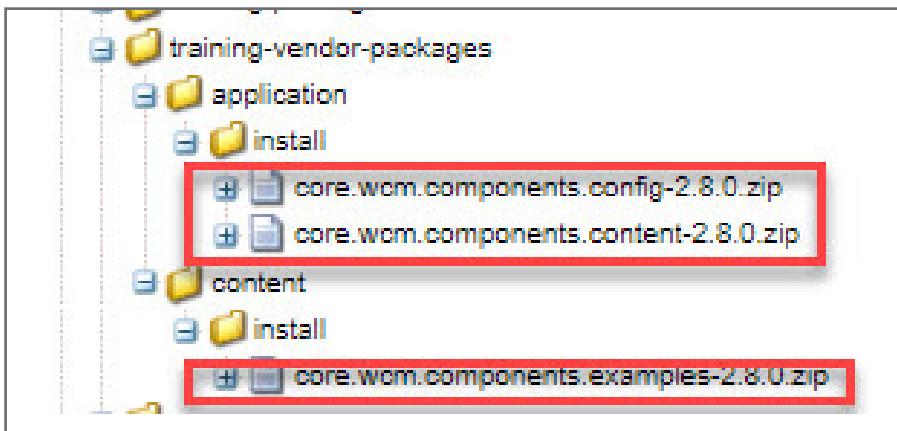


7. Similarly, navigate to the `/apps/training-packages/content/install` folder and notice the install of the `training.ui.content-1.0-SNAPSHOT.zip` which is the immutable content.



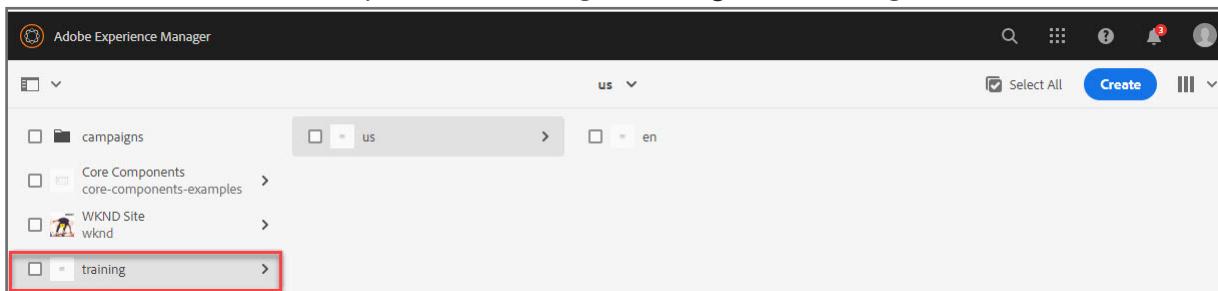
Note: With Archetype 23, the install has organized the content into mutable and immutable content. Mutable content are content that can typically change at runtime(/content,/conf/,/var, etc..). Immutable content can only be updated via the CICD pipeline (/apps, /libs).

8. 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.

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



Congratulations! You have successfully installed your project into AEM.

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>

- To learn more about the latest AEM Archetype, go to:

<https://github.com/adobe/aem-project-archetype>

<https://docs.adobe.com/content/help/en/experience-manager-core-components/using/developing/archetype/overview.html>

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
- Set up AEM for Eclipse
- Build and deploy an AEM project

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

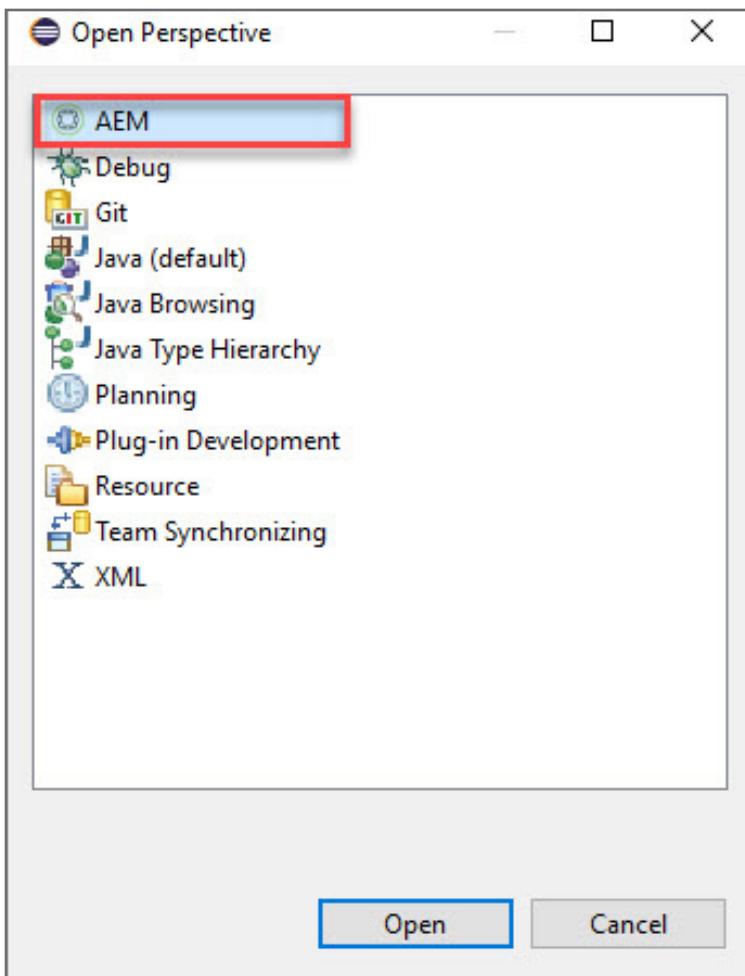
Benefits of AEM Plug-in for Eclipse

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 (Optional)

In this exercise, you will install and configure Eclipse.

This exercise includes the following tasks:

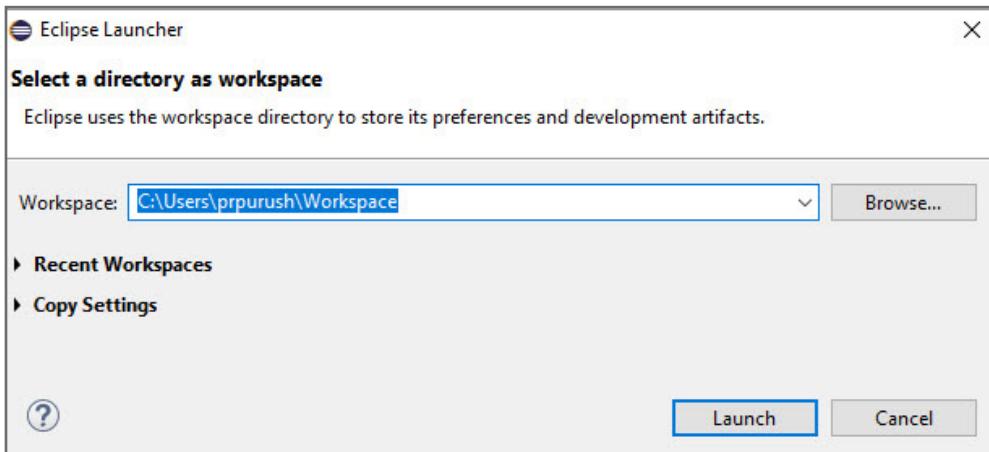
1. Install Eclipse
2. Configure Eclipse



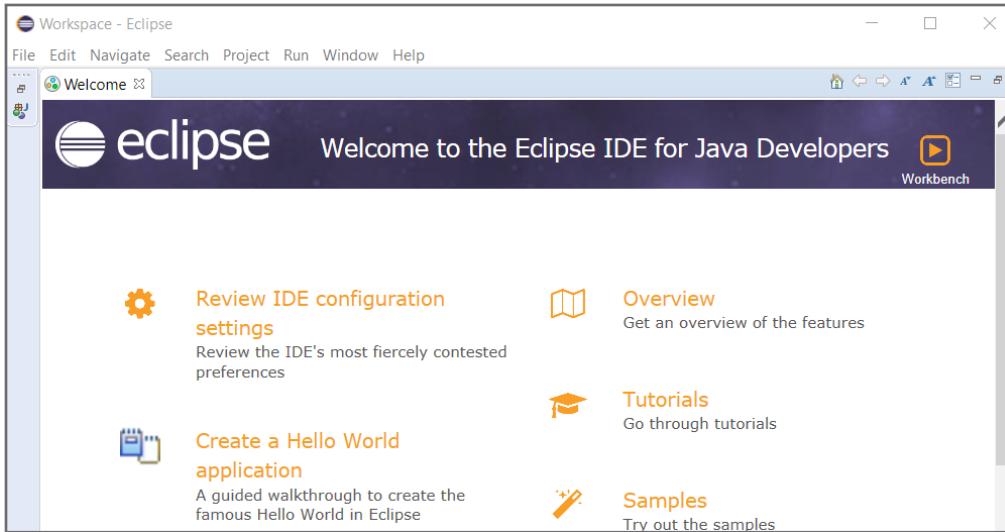
Note: You can skip this exercise if you use a ReadyTech environment because AEM is already installed as part of the image.

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:



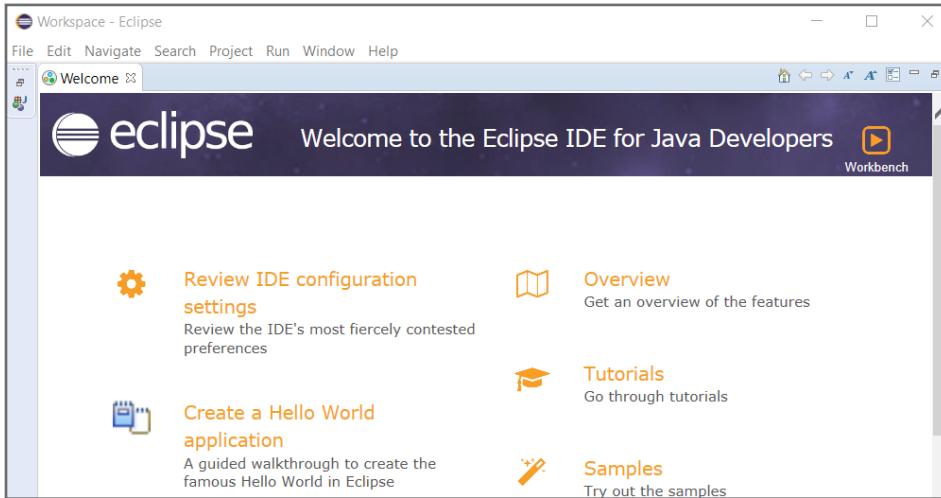
4. Accept the default workspace and click **Launch**. The Eclipse Development Environment opens, as shown:



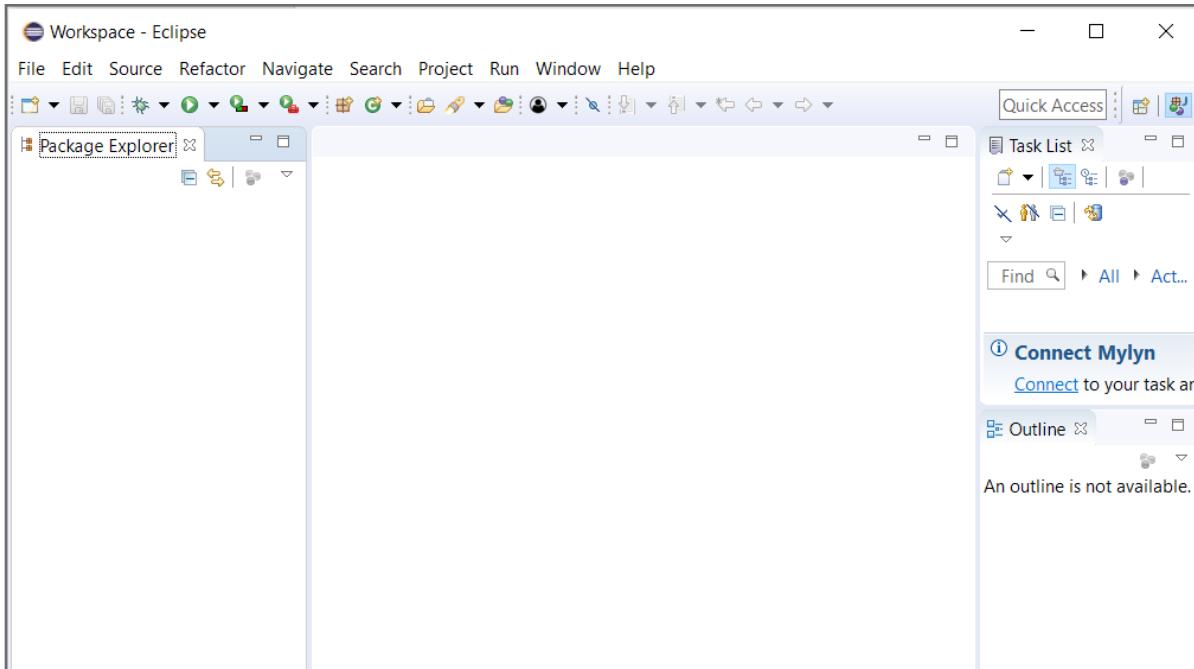
Task 2: Configure Eclipse

In this task, you will configure Eclipse.

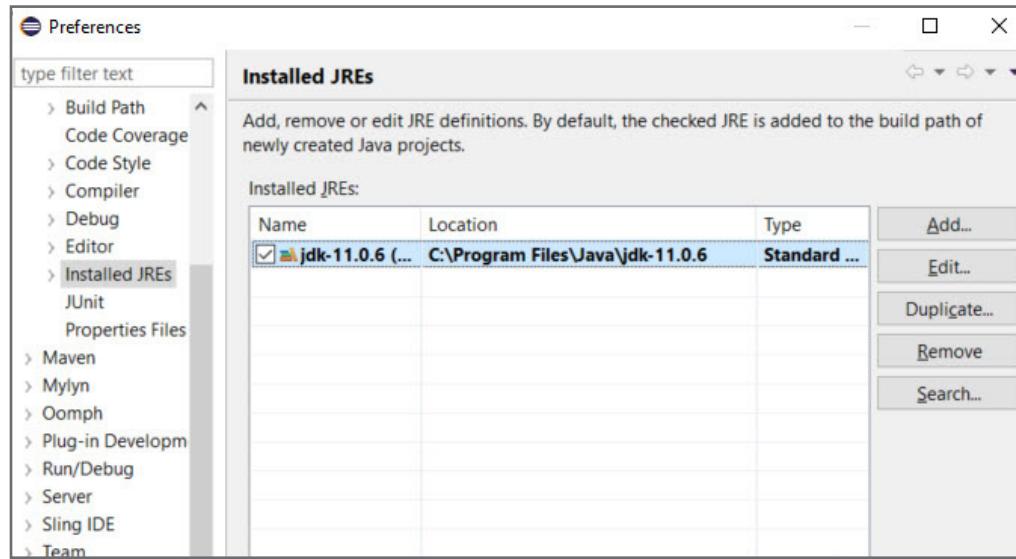
1. Click the **Workbench** logo in the upper-right corner, as shown, to close the Welcome screen.



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 (Optional)

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](#).

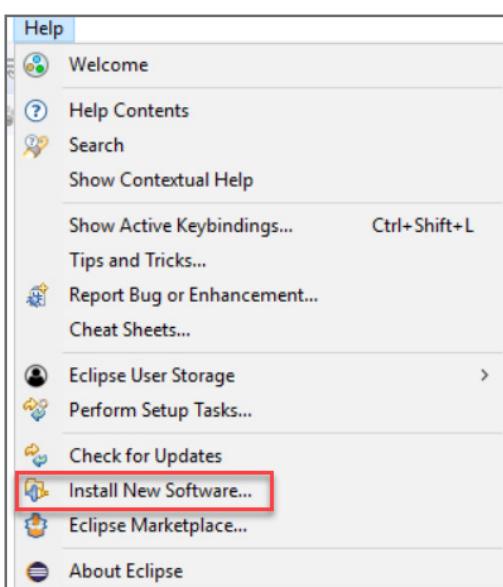
 **Note:** There are two ways to install the plug-in:Online:

1. You will provide the link to install the plug-in in Eclipse.Offline.
2. You will provide the downloaded plug-in in Eclipse.

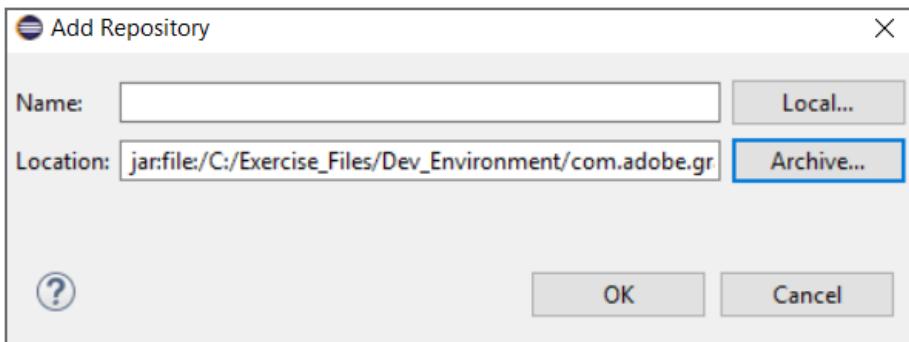
You will perform the offline method in this exercise.

To install your package:

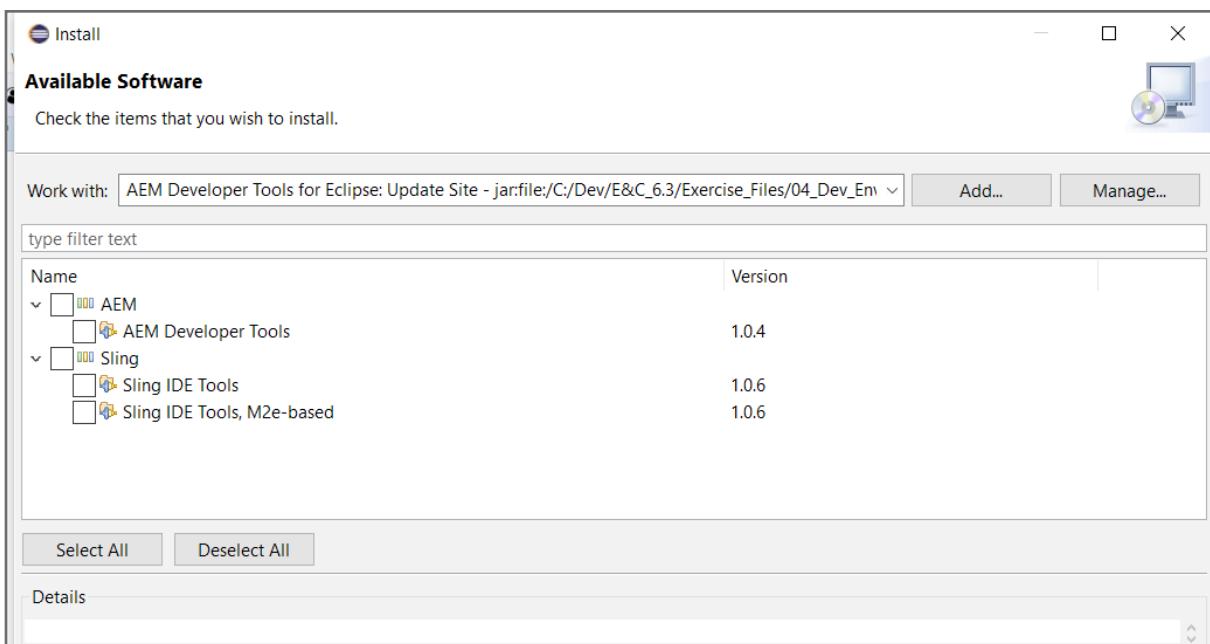
2. Double-click **eclipse.exe** (or **eclipse.app**) to start Eclipse. The Eclipse Development Environment opens:
3. Select **Help > Install New Software**, as shown. The Install window opens.



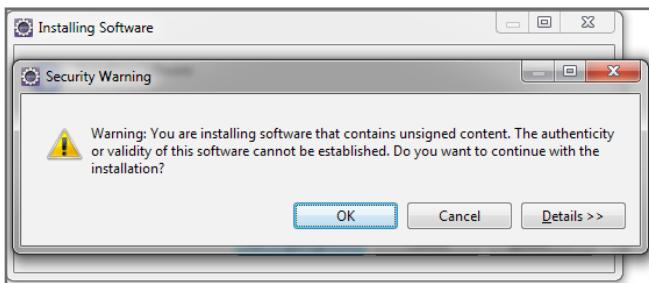
4. Click **Add**. The Add Repository dialog box opens.
5. Click **Archive**.
6. Navigate to the repository archive and select the zip file (**com.adobe.granite.ide.p2update-1.3.0.zip**) provided for the plug-in from **Exercise_Files_TB/Dev_Environment/**.
7. Click **Open** to add the location to the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



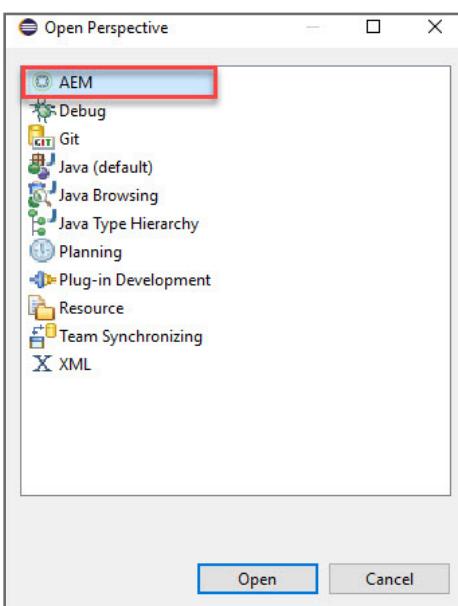
10. Click **Select All** to select AEM and Sling.
11. Click **Next**. The Install Details screen opens.
12. Click **Next**. The Review Licenses screen opens.
13. Click the **I accept the terms of the license agreements** option and then click **Finish**. The **Installing Software** dialog box with a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

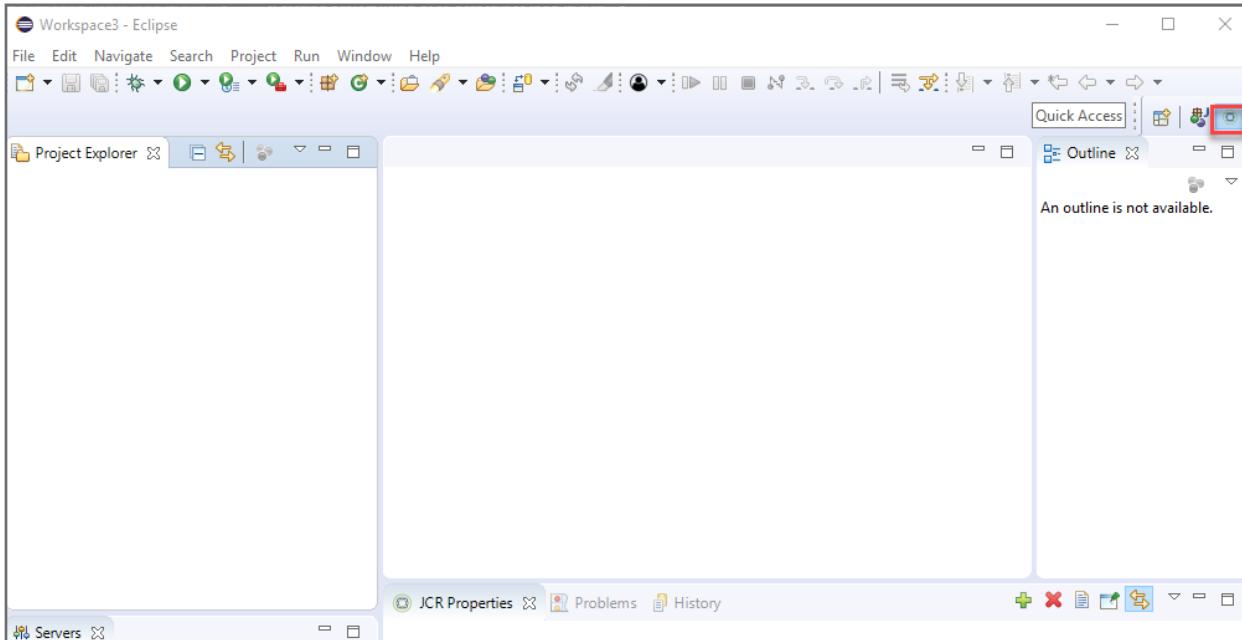
 **Note:** This process takes about a minute. If you see a dialog box that asks if you want to keep the current location of your Eclipse install, click **OK**. Eclipse opens.

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:

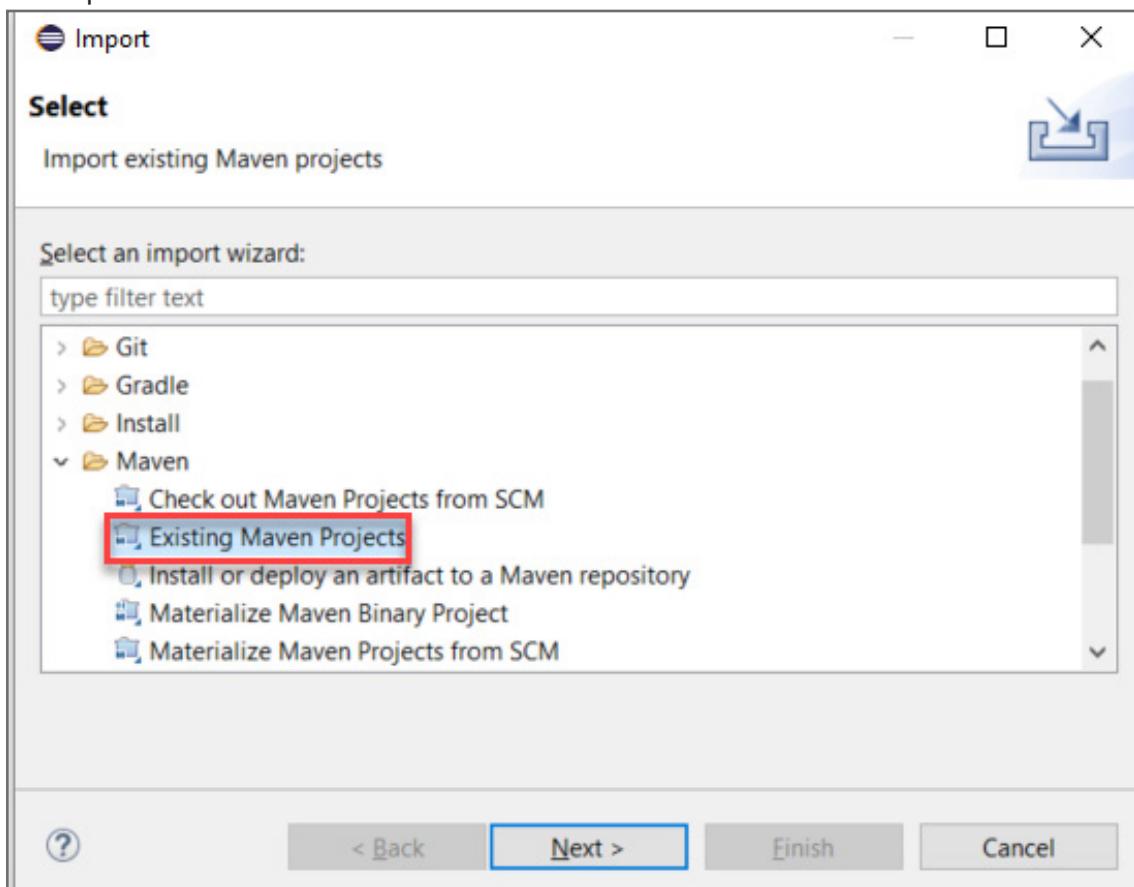
1. Import an AEM project
2. Configure jcr_root folders

Task 1: Import an AEM project

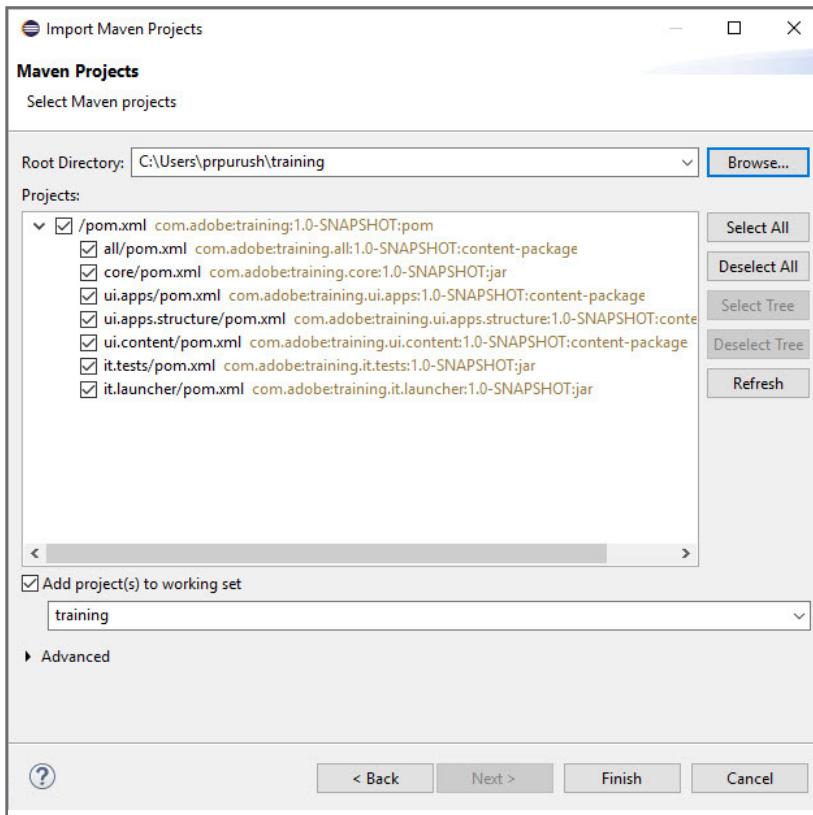
In this task, you will import an AEM project.

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.

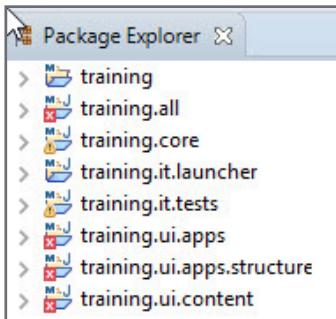


6. Click the **Browse** button beside the Root Directory field and navigate to the path/to your Maven project /<AEM project> and select all the POM files, as shown:



 **Note:** Your AEM project might have a different name.

7. Ensure that the Add project(s) to working set checkbox is selected.
 8. 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.

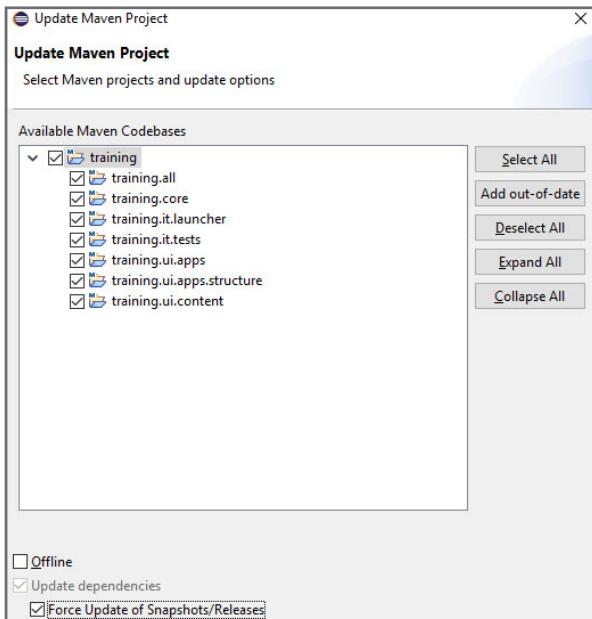
9. On the **Project Explorer** tab, right-click the parent folder and select **Maven > Update Project**.
The **Update Maven Project** window opens.

10. Verify your codebase is selected as **training**.



Note: Your Maven codebase might have a different name.

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

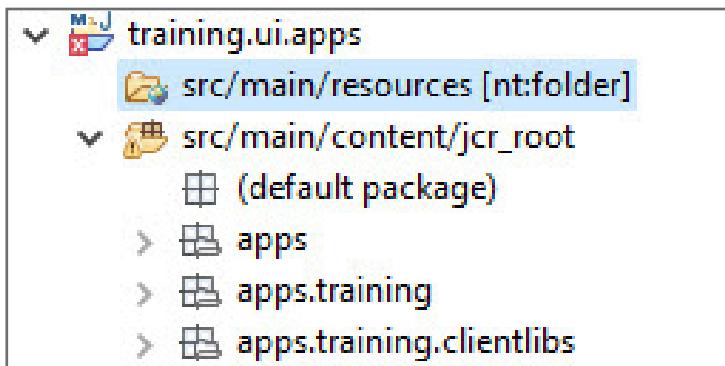


12. Ignore any errors. You will fix them later.

Congratulations! You have successfully imported an AEM project.

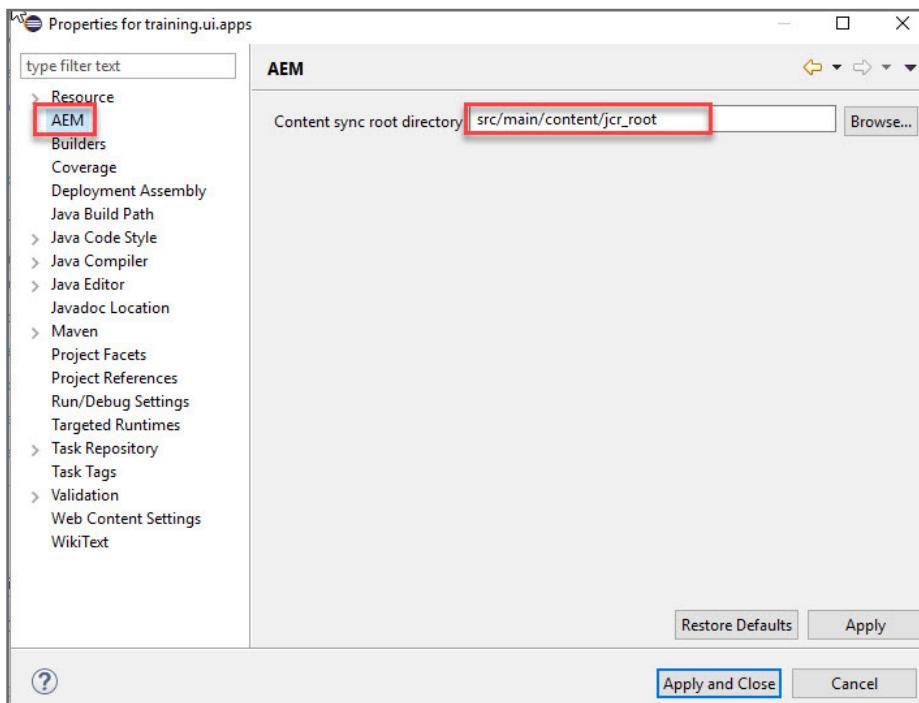
Task 2: Configure jcr_root folders

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



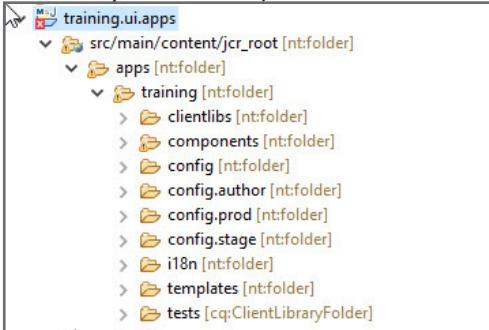
 **Note:** AEM project is the name of the project you created and imported. For example **training**. The name of your project might be different from the screenshot shown.

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



5. Click **Apply** and **Apply and Close**.

6. Verify the folders, as shown:



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

Congratulations you have successfully configured your jcr_root folders!

Exercise 4: Build and deploy the project to AEM

Now that Eclipse is setup, you can install your Maven project through the command line (see previous module) or you can use the Eclipse M2E plugin for Maven commands. In this exercise you will learn how to build and deploy using the M2E plugin.

This exercise includes the following tasks:

1. Select Maven profiles for projects
2. Deploy the project

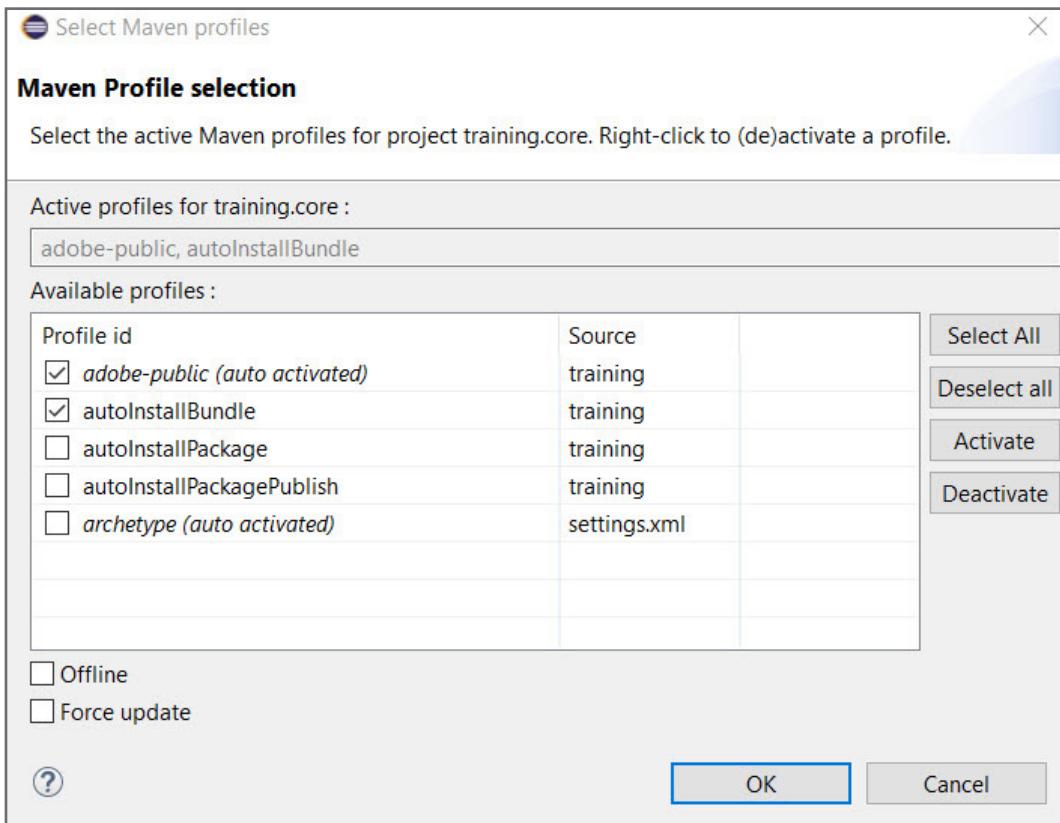
Task 1: Select Maven profiles for projects

1. Right-click <AEM project>.core, and select **Maven > Select Maven Profiles**. The Maven Profile selection window opens.



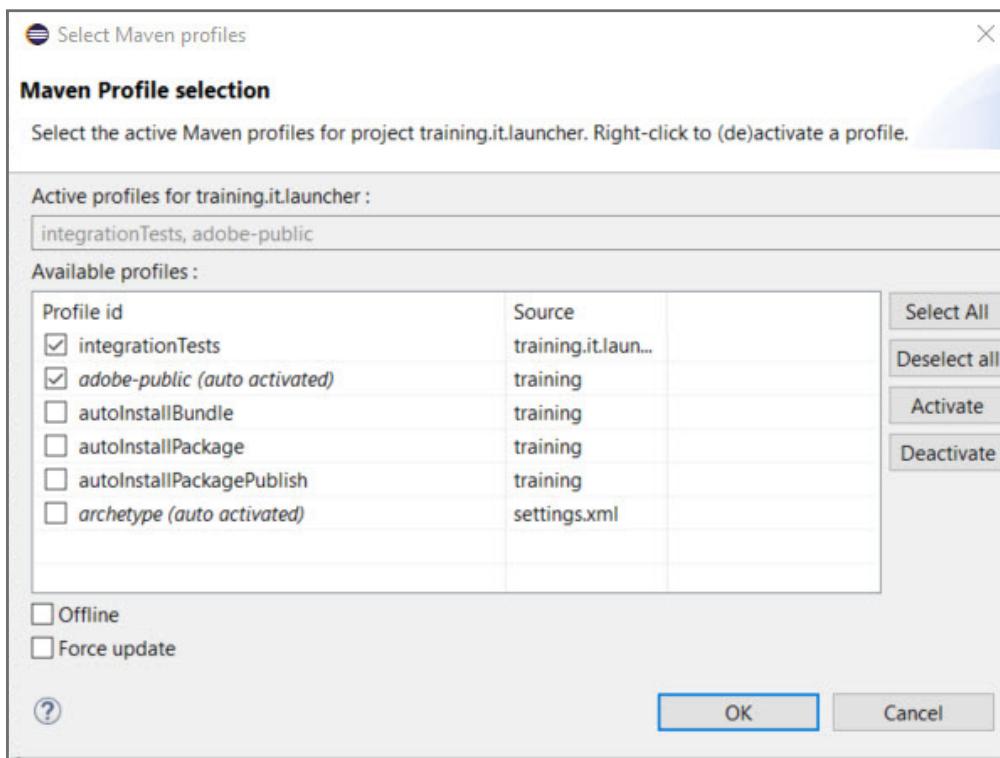
Note: AEM project is the name of the project you created and imported. For example **training**. The name of your project might be different.

2. In the **Select Maven profiles** window, select **autoInstallBundle** and **adobe-public** from the Available profiles section, and click **OK**, as shown. The **autoInstallBundle** for training.core is activated.



3. Right-click <AEM project>.it.launcher, and select **Maven > Select Maven Profiles**. The Maven Profile selection window is displayed.

4. Select **integrationTests** and **adobe-public** from the Available profiles section, and click **OK**, as



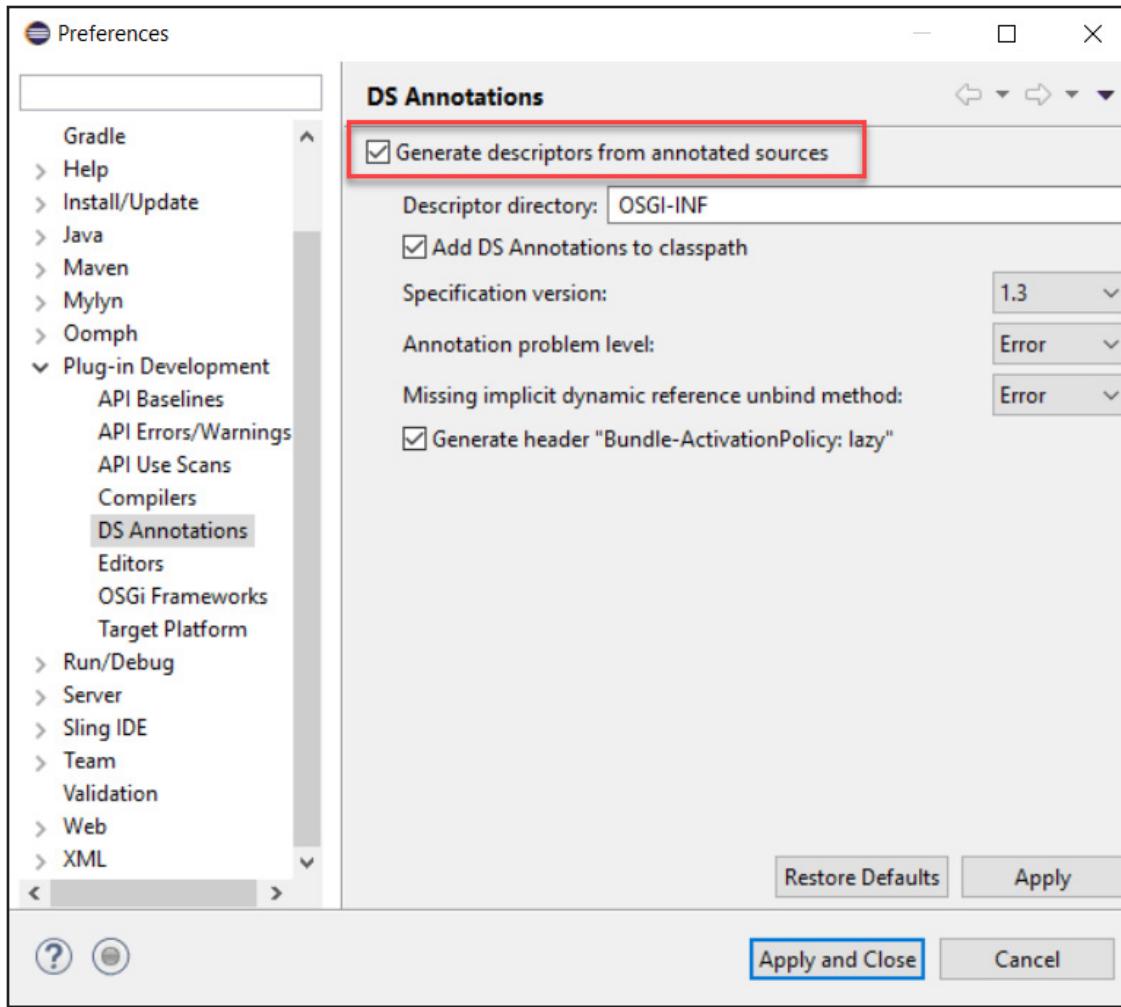
5. Right-click <AEM project>.ui.apps, and click **Maven > Select Maven Profiles**. The Maven Profile selection window is displayed.
6. Select **autoInstallPackage** and **adobe-public** from the Available profiles section, and click **OK**. The **autoInstallPackage** for the **training.ui.apps** is activated.
7. Right-click <AEM project>.ui.content and activate the profiles **autoInstallPackage** and **adobe-public**.

You have now set up your Maven profiles.



Note: If you see **No DS descriptor found at path target/classes/OSGI-INF/com.adobe.training.core...**, it is because Eclipse looking for files that you do not need. You can remove these messages by configuring Eclipse so that it will create the missing DS descriptors files automatically.

8. (Optional) - To remove the messages mentioned in the above note, go to **Window > Preferences > Plug-in Development > DS Annotations**, select **Generate descriptors from annotated sources** and click **Apply and Close**:



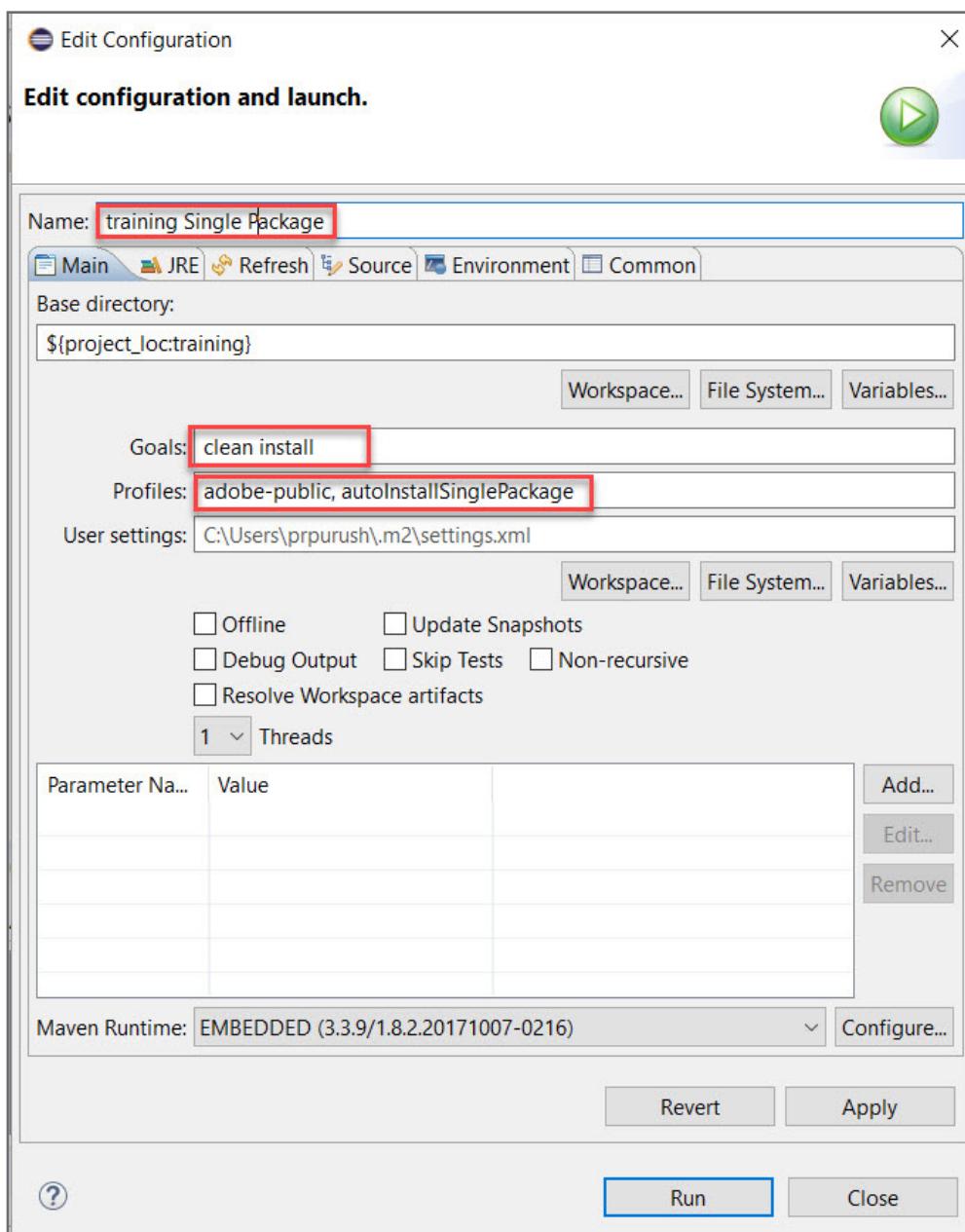
Then, update the Maven project (Alt+F5).

Task 2: Deploy the AEM project

In this task, you will deploy the AEM project.

Option 1 (using Eclipse):

1. Right-click <AEM project> (parent directory), and select **Run As > 1 Maven build...** The **Edit configuration and launch** dialog box opens.
2. In the **Edit configuration and launch** dialog box, type the following, as shown:
 - > Name: **training Single Package**
 - > Goals: **clean install**
 - > Profiles: **adobe-public, autoInstallSinglePackage**





Note: You should run the **autoInstallSinglePackage** profile to ensure you have the correct content pages built within the repository. If you run **autoInstallBundle** first, the build will fail to upload the bundle into Apache Felix.

- Click **Apply**, and then click **Run**. The Success message appears:

```
[INFO] Downloading from : https://repo.maven.apache.org/maven2/org/apache/maven/maven-archiver/3.1.1/maven-archiver-3.1.1.jar
[INFO] Downloaded from : https://repo.maven.apache.org/maven2/org/apache/maven/maven-archiver/3.1.1/maven-archiver-3.1.1.jar (24
[INFO] Downloading from : https://repo.maven.apache.org/maven2/org/apache/maven/shared/maven-shared-utils/3.0.1/maven-shared-util
[INFO] Downloaded from : https://repo.maven.apache.org/maven2/org/apache/maven/shared/maven-shared-utils/3.0.1/maven-shared-util
[INFO] Downloading from : https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-io/2.7.1/plexus-io-2.7.1.jar
[INFO] Downloaded from : https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-io/2.7.1/plexus-io-2.7.1.jar (86 kB at 4
[INFO] Building jar: C:\Users\prpurush\.m2\training\it.launcher\target\training.it.launcher-1.0-SNAPSHOT.jar
[INFO]
[INFO] --- maven-install-plugin:2.5.2:install (default-install) @ training.it.launcher ---
[INFO] Installing C:\Users\prpurush\.m2\training\it.launcher\target\training.it.launcher-1.0-SNAPSHOT.jar to C:\Users\prpurush\.m2\repository\com\adobe\training\it.la
[INFO] Installing C:\Users\prpurush\.m2\training\it.launcher\pom.xml to C:\Users\prpurush\.m2\repository\com\adobe\training\it.la
[INFO]
[INFO] Reactor Summary for training 1.0-SNAPSHOT:
[INFO]
[INFO] training ..... SUCCESS [ 0.483 s]
[INFO] Training Project - Core ..... SUCCESS [ 13.384 s]
[INFO] TrainingProject - UI apps ..... SUCCESS [ 23.911 s]
[INFO] TrainingProject - UI content ..... SUCCESS [ 4.556 s]
[INFO] TrainingProject - Integration Tests Bundles ..... SUCCESS [ 2.287 s]
[INFO] TrainingProject - Integration Tests Launcher ..... SUCCESS [ 56.354 s]
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 01:44 min
[INFO] Finished at: 2019-09-04T10:42:48-07:00
[INFO]
```



Note: When the AEM project is first run, Maven will download all required dependencies locally to the user's .m2 directory. Depending on your Internet connection, this can take about 15 minutes. Downloading dependencies is a one-time action if you do not add any new dependencies. Deploying the project to AEM after the initial deployment only takes a few seconds.

Option 2 (using Maven):

4. Open a Command Prompt window and navigate to your newly created Maven project called <AEM project directory>..



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

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

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

To verify the installed packages:

6. Log in to AEM with the admin credentials.
7. Navigate to Package Manager (**Adobe Experience Manager > Tools > CRXDE Lite**).
8. In the Search box at the top of Package manager, type training and press **Enter** to search for training.
9. Verify you see the installed packages, as shown:

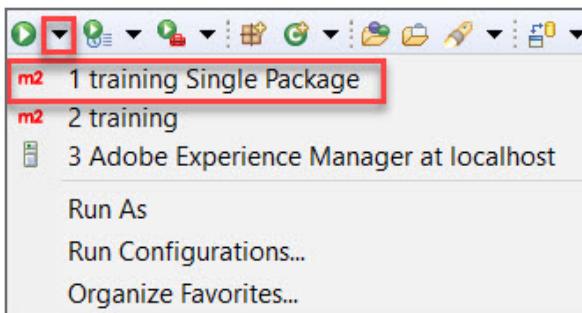
Package Name	Description	Size
training.ui.content-1.0-SNAPSHOT.zip	UI content package for TrainingProject	128.1 KB
training.ui.apps-1.0-SNAPSHOT.zip	UI apps package for TrainingProject	3.4 MB

10. Go to the **Web Console** (<http://localhost:4502/system/console/configMgr>), and select **OSGI > Bundles**.

11. Perform a search using Ctrl+F to find training, as shown. Notice that the highest ID is **TrainingProject--Core (training.core)**.

Bundle information: 517 bundles in total - all 517 bundles active	
Id ▾ Name	
517	► Apache Sling Tooling Support Install (<i>org.apache.sling.tooling.support.install</i>)
516	► Apache Sling Tooling Support Source (<i>org.apache.sling.tooling.support.source</i>)
515	▼ TrainingProject - Core (<i>training.core</i>) Symbolic Name: training.core Version: 1.0.0.20200205221623183 Bundle Location: jcrinstall:/apps/training/install/training.core-1.0-SNAPSHOT.jar

 **Note:** If you want to rerun the project, go back to **Eclipse** and click the Play button and select **training Single Package**, as shown:



Exercise 5: Synchronize with AEM and 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.

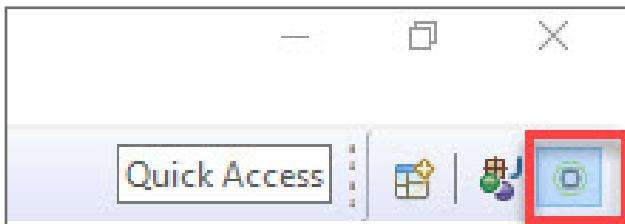
This exercise includes the following tasks:

1. Configure the AEM server
2. Import content to Local project
3. Export content from Eclipse

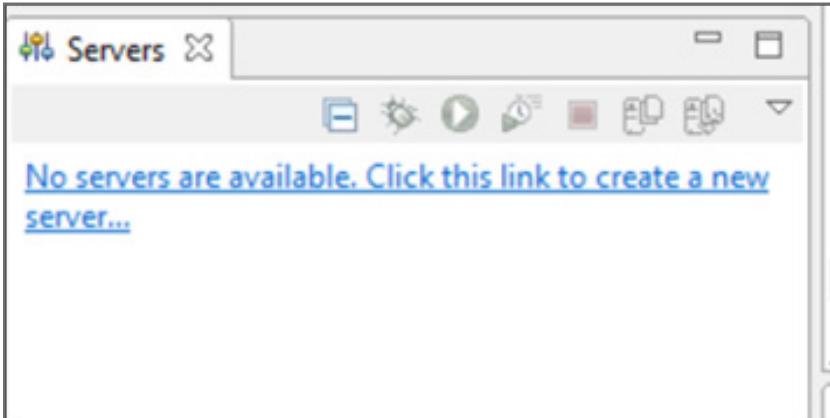
Task 1:Configure the AEM server

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.

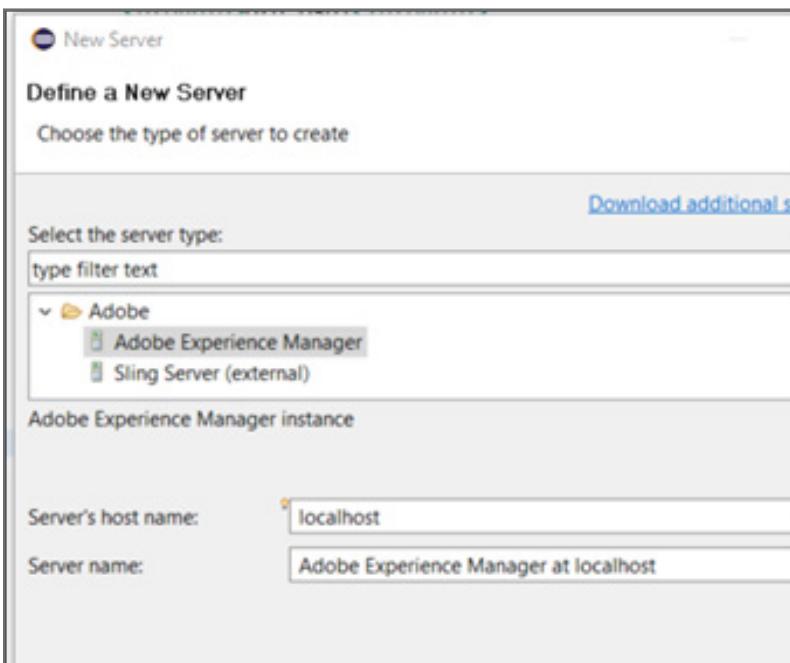


2. On the left-hand side of the Eclipse Workspace below the **Project Explorer tab**, notice the **Servers** tab. Click the **No Servers are available. Click this link to create a new server..** link, as shown:



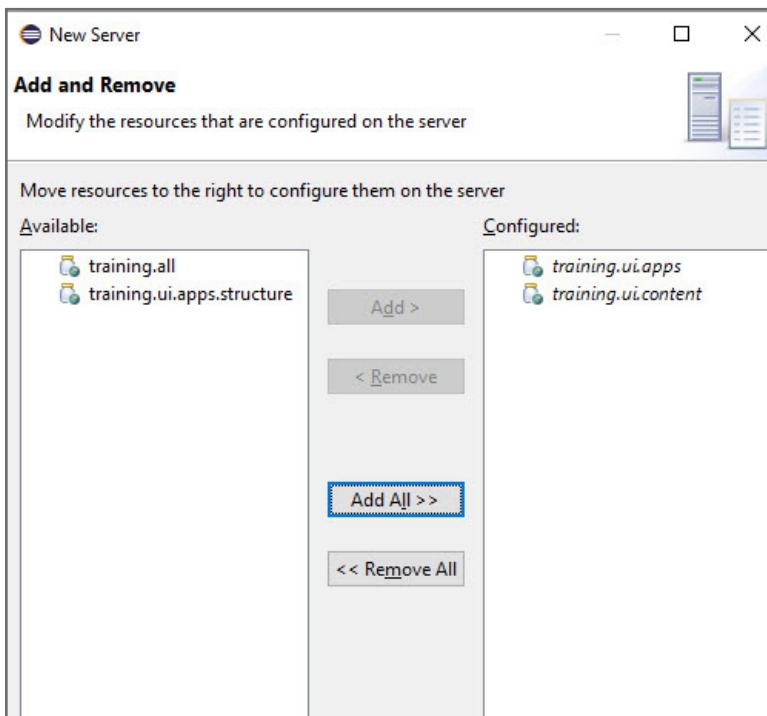
The **Define a New Server** dialog box opens.

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



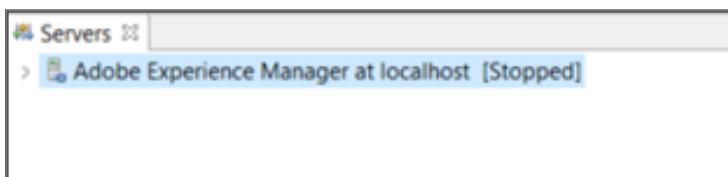
4. Accept the default settings for the **Server's host name** and **Server name** fields.
5. Click **Next**. The **Add and Remove resources** dialog box opens.

6. Select <AEM project>.ui.apps 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:

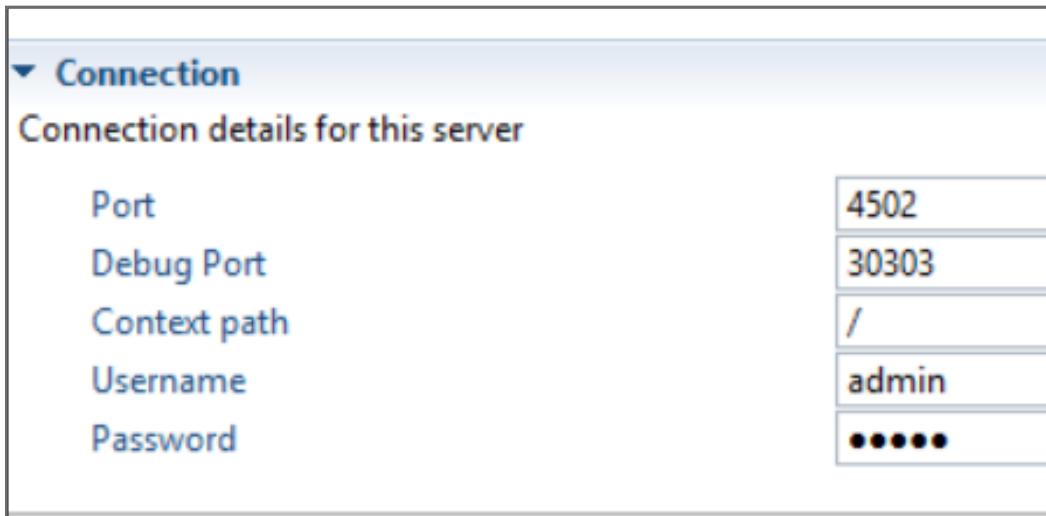


 **Note:** AEM project is the name of the project you created and imported. For example **training**. The name of your project might be different from the screenshot shown.

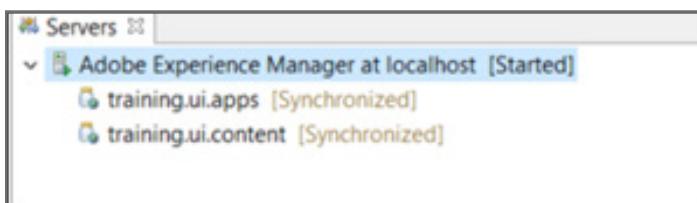
-
7. Click **Finish** on the **New Server** screen. The AEM Server is now defined.
8. On the left-hand side of the workspace (below the **Project Explorer** tab), click the **Servers** tab and note how **Adobe Experience Manager at localhost [Stopped]** is now available, as shown:



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: Import content to Local project

In this task, you will observe the filters for both ui.apps and ui.content content packages and then import any changes from AEM to your local project.

1. The ui.apps module is for all the immutable code for your project. Navigate to: <AEM project>.ui.apps > src/main/content/META-INF/vault > filter.xml



2. Double-click **filter.xml** to open it and view the current filter that will be synchronized with this module.



Note: If your instructor has provided a **course-filter.txt** file as part of the Exercise-Folder, ensure the ui.apps filters are included.



Note: If you had any changes in AEM (based on filter.xml), those nodes and properties are imported to your project by performing the steps 3 and 4.

Skip steps 3 and 4 , if you do not have any changes made to filter.xml.

3. Right click on <AEM project>.ui.apps > src/main/content/jcr_root and select **Import from server**.
4. Click **Finish**.
5. The ui.content module is for all of the mutable content for your project. Navigate to: <AEM project>.ui.content > src/main/content/META-INF/vault > filter.xml



Note: If your instructor has provided a **course-filter.txt** file as part of the Exercise-Folder, ensure the ui.content filters are included.

6. View the current filter that will be synchronized with this module.
7. Right click on <AEM project>.ui.content > src/main/content/jcr_root and select **Import from server**.
8. Click **Finish**.

Task 3: Export content from Eclipse

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="${model.message}">
23    <p class="cmp-helloworld_item-label">Model message:</p>
24    <pre class="cmp-helloworld_item-output" data-cmp-hook-helloworld="model">${model.message}<pre>"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:

```

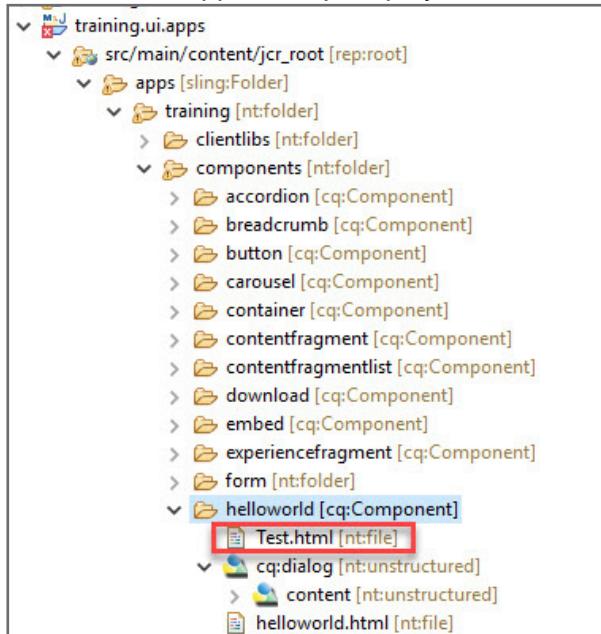
0 Unless required by applicable law or agreed to in writing, software
1 distributed under the License is distributed on an "AS IS" BASIS,
2 WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
3 See the License for the specific language governing permissions and
4 limitations under the License.
5 *-->
6 <div class="cmp-helloworld" data-cmp-is="helloworld">
7   <h2 class="cmp-helloworld_title">Hello World Component</h2>
8   <div class="cmp-helloworld_item" data-sly-test="${properties.text}">
9     <p class="cmp-helloworld_item-label">Text property:</p>
10    <pre class="cmp-helloworld_item-output" data-cmp-hook-helloworld="property">${properties.text}</pre>
11  </div>
12  <div class="cmp-helloworld_item" data-sly-use.model="com.adobe.training.core.models.HelloWorldModel" data-sly-test="${model.message}">
13    <p class="cmp-helloworld_item-label">Model message:</p>
14    <pre class="cmp-helloworld_item-output" data-cmp-hook-helloworld="model">${model.message}<pre>"This is a change in Eclipse"</pre>
15  </div>
16 </div>

```

7. Next, we will make a change in CRXDE Lite and import it back into our Eclipse 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.

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 AEM Sync
- Install AEM Sync
- Test 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.

AEM Sync

AEM Sync, an extension for Visual Studio Code, automatically syncs file changes to an Adobe Experience Manager service. It is intended to be used for local development.

The AEM Sync extension does NOT make use of Filevault or Filevault's underlying APIs. It relies entirely on the Restful APIs that AEM provides. Therefore, these APIs must be enabled for this extension to work. They should be enabled by default for local development.

Exercise 1: Install Visual Studio Code and AEM Sync

In this exercise, you will install Visual Studio Code and AEM Sync.

1. Install Visual Studio Code.
2. Install AEM Sync.

Task 1: Install Visual Studio Code

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



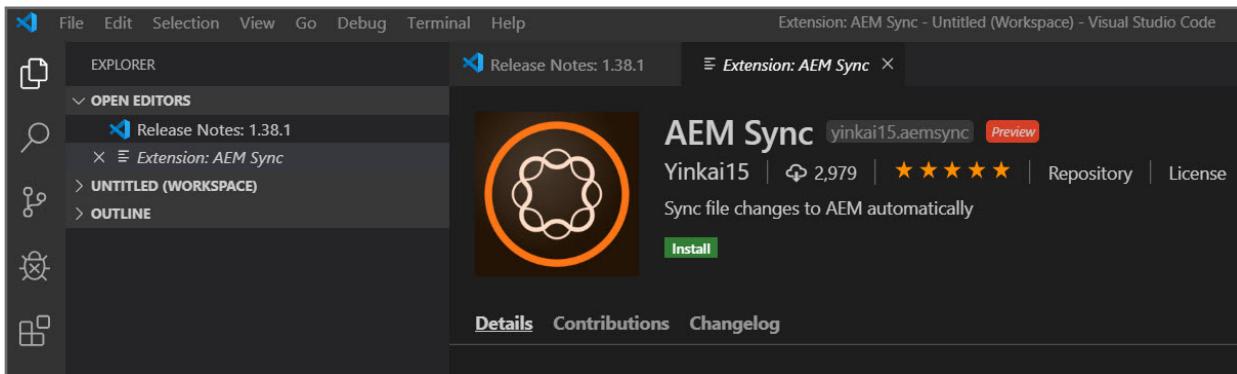
Note: If you do not have the ability to download Visual Studio from the Internet, you can use the installer in the distribution-files folder provided to you by your instructor.

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.

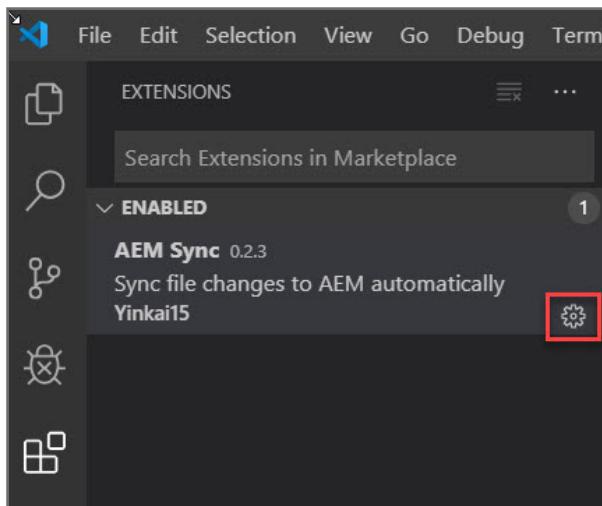
Congratulations! You have installed Visual Studio Code and are now ready to install the AEM Sync extension.

Task 2: Install AEM Sync

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



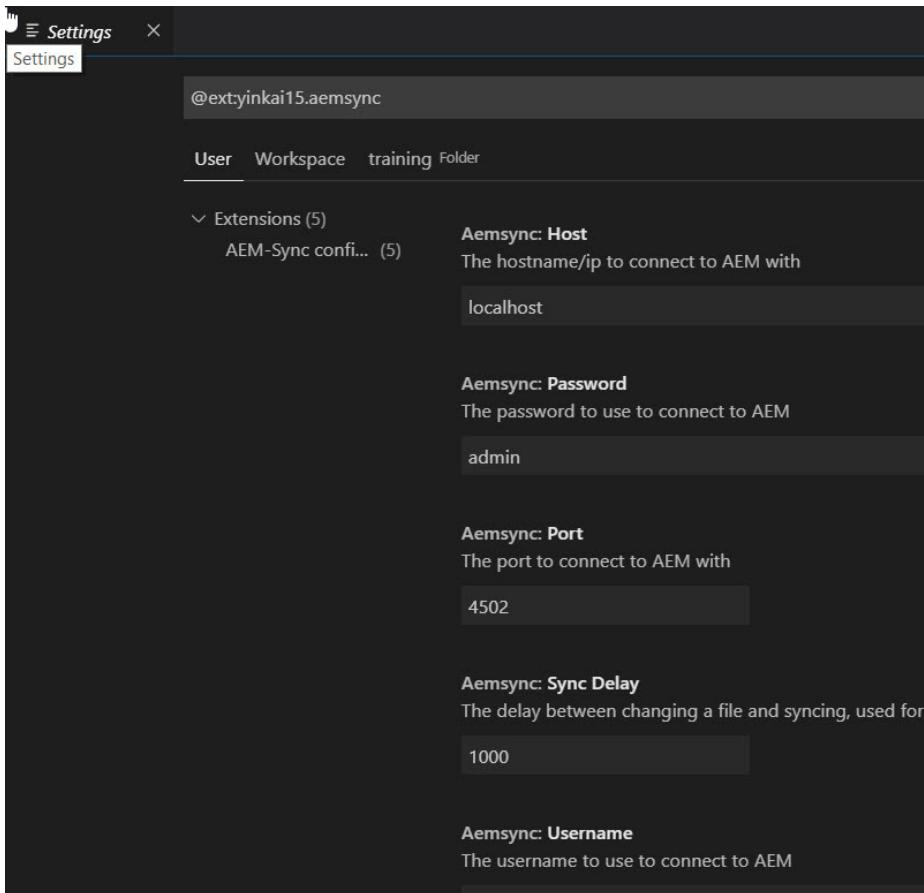
4. Click **Install** to install the extension.
5. In **Visual Studio Code**, select **View > Extensions** from the menu bar. The **AEM Sync Extension** is displayed and enabled in the **EXTENSIONS** area.
6. Click **AEM Sync** and click the Manage icon (gear icon), as shown:



7. In the Menu, click **Extension Settings**. The **Settings** tab opens.

8. Verify that the extension is properly configured, as shown:

- › Host
- › Port
- › Username
- › Password



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

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

Exercise 2: Add styling using Visual Studio Code

In this exercise, you will install AEM Sync and test the AEM Sync Extension.

 **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:

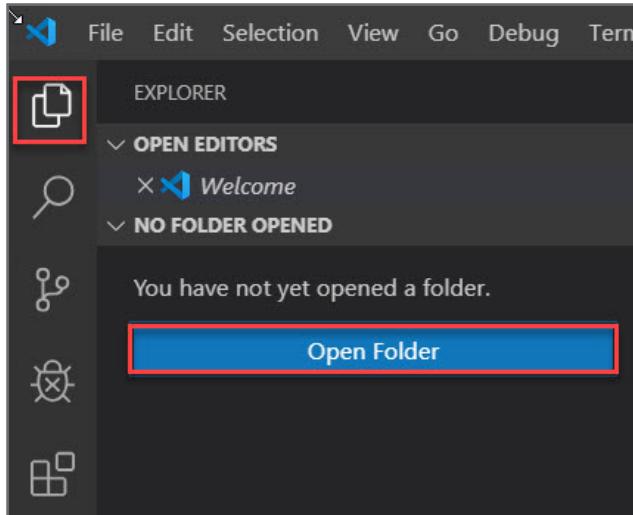
1. Install content locally (Optional)
2. Test AEM Sync Extension

Task 1: Install content locally (Optional)

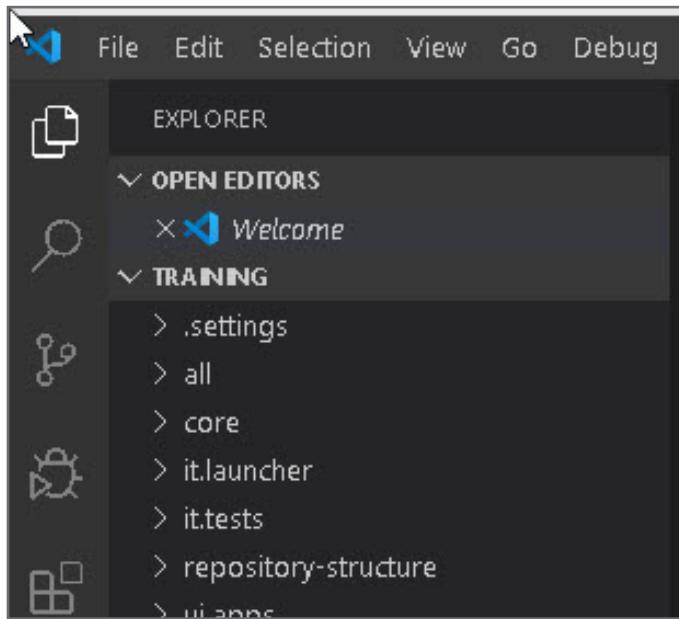
 **Note:** This exercise assumes you have a local Maven project. <AEM Project Directory> represents the location of this local Maven project.

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

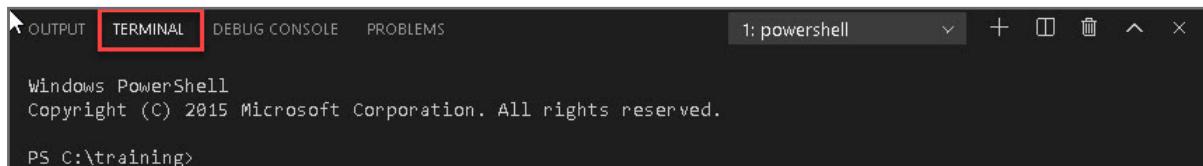
1. Confirm that your AEM service is running.
2. In **Visual Studio Code**, click the Explorer icon on the left (OR you can press Ctrl+Shift+E) and click **Open Folder**, as shown:



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

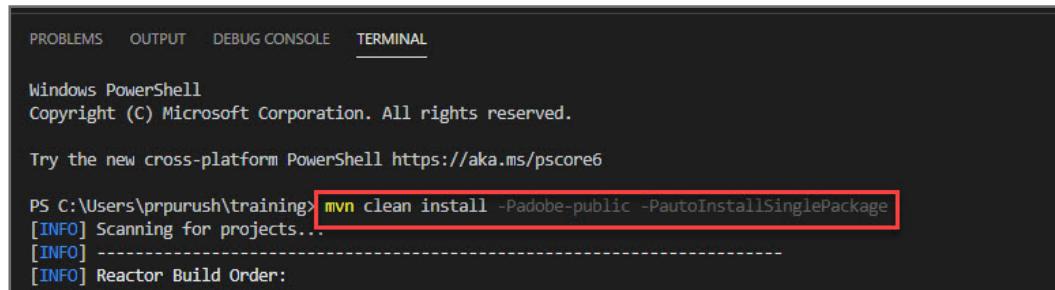


4. Select the Terminal tab in the window at the bottom, as shown:

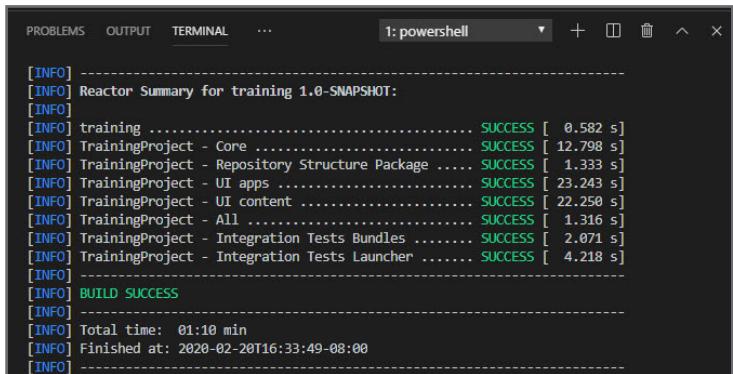


5. Run the following command to deploy your AEM project:

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



6. Verify the install is successful, as shown:



A screenshot of a terminal window titled "1: powershell". The window displays a build log for a project named "TrainingProject". The log shows various components being built, each with a "SUCCESS" status and a duration in parentheses. The log concludes with a "BUILD SUCCESS" message and ends with the total time taken and the finished date and time.

```
[INFO] -----
[INFO] Reactor Summary for training 1.0-SNAPSHOT:
[INFO]
[INFO] training ..... SUCCESS [ 0.582 s]
[INFO] TrainingProject - Core ..... SUCCESS [ 12.798 s]
[INFO] TrainingProject - Repository Structure Package ..... SUCCESS [ 1.333 s]
[INFO] TrainingProject - UI apps ..... SUCCESS [ 23.243 s]
[INFO] TrainingProject - UI content ..... SUCCESS [ 22.250 s]
[INFO] TrainingProject - All ..... SUCCESS [ 1.316 s]
[INFO] TrainingProject - Integration Tests Bundles ..... SUCCESS [ 2.871 s]
[INFO] TrainingProject - Integration Tests Launcher ..... SUCCESS [ 4.218 s]
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 01:10 min
[INFO] Finished at: 2020-02-20T16:33:49-08:00
[INFO] -----
```

Task 2: Test AEM Sync Extension



Note: This task assumes you have a maven project already created on your machine.

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

aemfed > text-styles		
Name		D
__MACOSX	2/	
text	2/	
css.txt	1/	
main.less	2/	

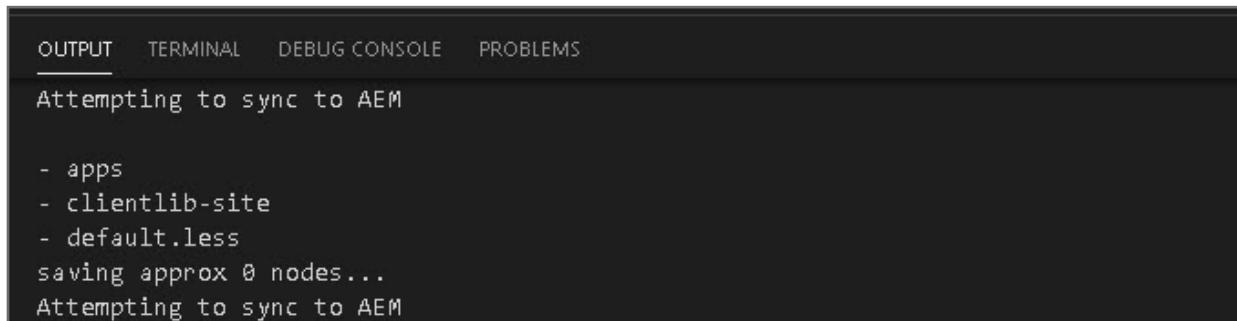
2. Copy the folder **text**.
3. In a file window, navigate to **/<AEM Project Directory>/ui.apps/src/main/content/jcr_root/apps/training/clientlibs/clientlib-site**

training > ui.apps > src > main > content > jcr_root > apps > training > clientlibs		
Name	Date modified	Type
clientlib-base	12/3/2019 10:56 A...	File folder
clientlib-dependencies	12/3/2019 10:56 A...	File folder
clientlib-grid	12/3/2019 10:56 A...	File folder
clientlib-site	12/3/2019 10:56 A...	File folder

4. Paste the **text** folder from **Exercise_Files_TB > VisualStudioCode** into the **/clientlib-site/components** folder (create **components** folder if needed):

training > ui.apps > src > main > content > jcr_root > apps > training > clientlibs > clientlib-site > components			
Name	Date modified	Type	Size
text	2/4/2020 4:33 PM	File folder	
css.txt	1/28/2020 12:21 PM	Text Document	1 KB
main.less	2/4/2020 1:23 PM	LESS File	1 KB

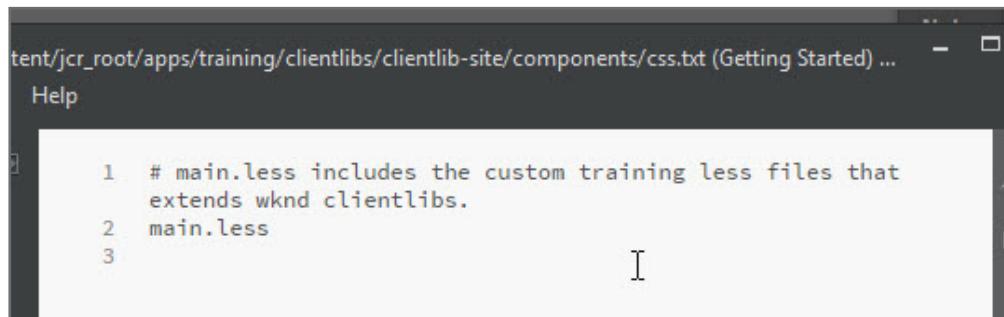
5. Verify you see messages in the **Visual Studio Code** output tab:



The screenshot shows the Visual Studio Code interface with the 'OUTPUT' tab selected. The output window displays log messages related to syncing files to AEM:

```
Attempting to sync to AEM
- apps
- clientlib-site
- default.less
saving approx 0 nodes...
Attempting to sync to AEM
```

6. 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 9.
7. In **Visual Studio Code**, navigate into the **/clientlib-site/** folder and open **css.txt** and add the line **main.less**.



The screenshot shows the Visual Studio Code interface with the 'css.txt' file open in the editor. The file contains the following content:

```
1 # main.less includes the custom training less files that
  extends wknd clientlibs.
2 main.less
3
```

8. Save **css.txt** and close.
9. Create a new file under **clientlibs/clientlib-site** called **main.less**.
10. Open **main.less** and add the following line:
`@import "components/text/text.less";`
11. Save **main.less** and close.

12. In **Visual Studio Code Explorer**, navigate to **ui.apps > src/main/content/jcr_root/apps/training/ clientlibs/clientlib-site/components/text/styles** and open **default.less**, as shown:

The screenshot shows the Visual Studio Code interface. The left sidebar displays the project structure under 'EXPLORER'. It includes a tree view of 'TRAINING' and 'ui.apps' folders, with 'src\main\content' expanded to show 'jcr_root\apps', 'training', 'clientlibs', 'clientlib-base', 'clientlib-dependencies', 'clientlib-grid', and 'clientlib-site'. Within 'clientlib-site', there are 'components\text-styles', '_MACOSX', 'text', and 'styles' sub-folders. The 'default.less' file is selected and highlighted with a red border at the bottom of the list.

The right pane shows the code content of 'default.less'. The code is as follows:

```

1  /*
2   * This file contains designs that will be applied to
3   * the component without the style system being used.
4   * The classes used here are hard coded into the component HTML
5   * This means the template author will not have control
6   * over these styles within the template and they will be
7   * applied by default.
8   */
9
10 /* css extending the WKND design */
11 .text .cmp-text {
12   p {
13     color: blue;
14   }
15 }

```

13. Change the value of the color element to **red** and save the file.
14. See the notifications in log window at the bottom, recording the changes and the saving of the nodes.

The screenshot shows the 'OUTPUT' tab in the Visual Studio Code terminal. The log output is as follows:

```

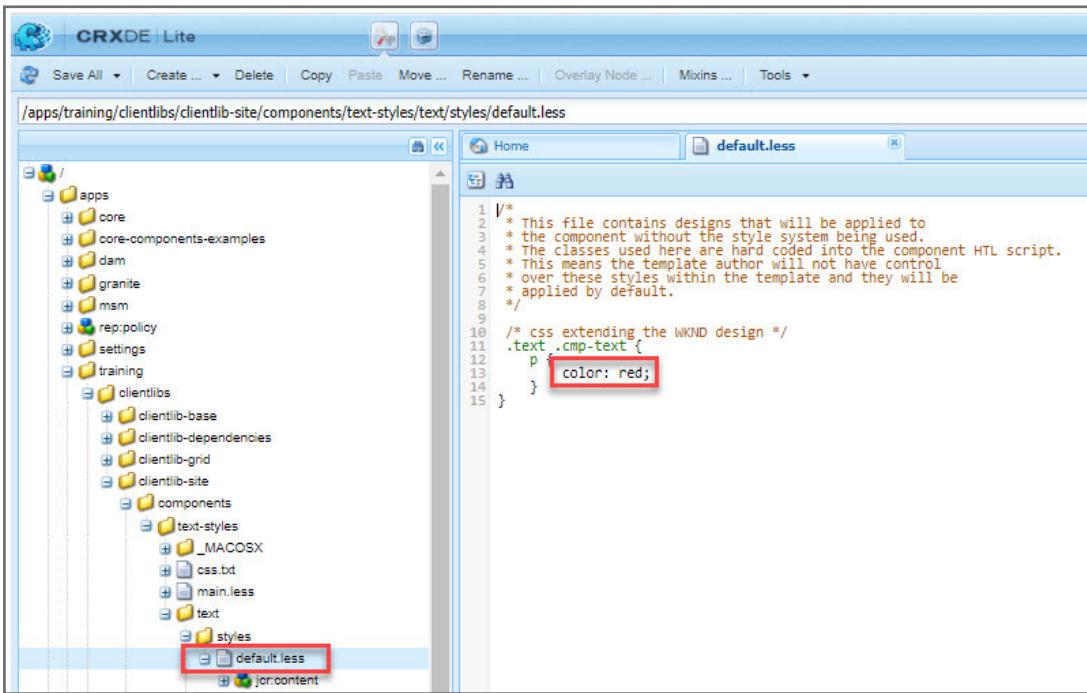
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
AEM Sync started... Searching for jcr_root
AEM Sync now watching for changes inside c:\Users\prpurush\.m2\training\ui.apps\src\main\content\jcr_root
Attempting to sync to AEM

-
- apps
- clientlib-site
A components
A text
A styles
A default.less
A jcr:content
A jcr:data
saving approx 6 nodes...

```

15. Using CRXDE Lite, click the Develop icon and navigate to `/apps/training/clientlibs/clientlib-site/components/text/styles` and double-click `default.less` to open it.

16. Notice the value of the color element has changed, as shown:

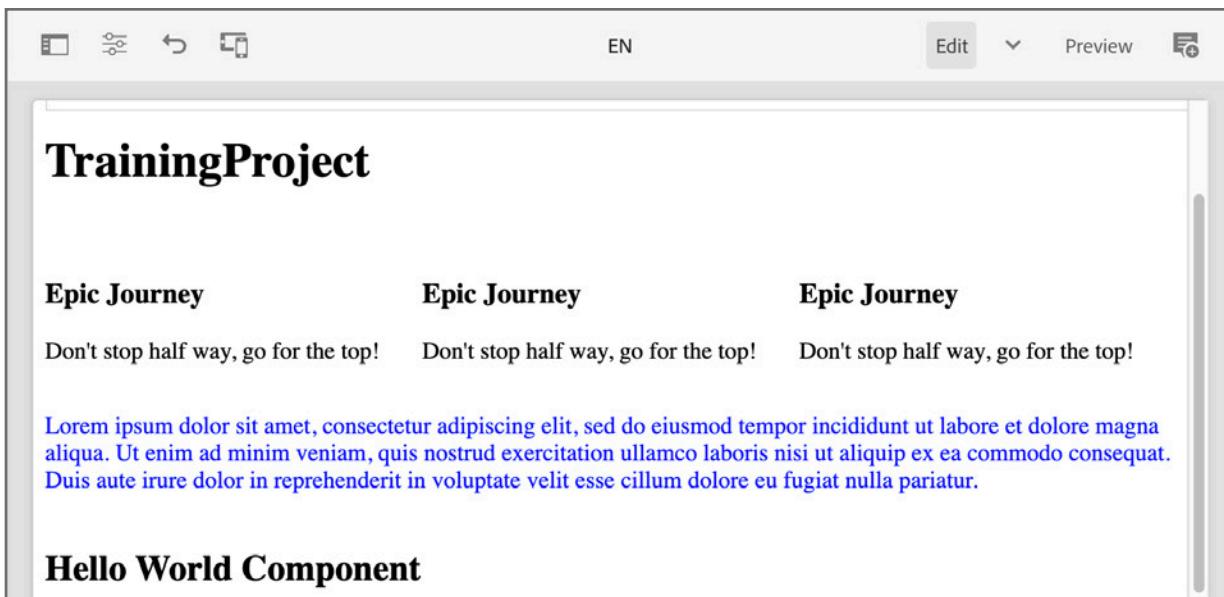


The screenshot shows the CRXDE Lite interface. The left pane displays a tree view of the AEM file system under the path `/apps/training/clientlibs/clientlib-site/components/text/styles`. The `default.less` file is selected and highlighted with a red box. The right pane shows the code editor with the contents of `default.less`:

```
1 /*
2  * This file contains designs that will be applied to
3  * the component without the style system being used.
4  * The classes used here are hard coded into the component HTL script.
5  * This means the template author will not have control
6  * over these styles within the template and they will be
7  * applied by default.
8 */
9
10 /* css extending the WKND design */
11 .text .cmp-text {
12   p {
13     color: red;
14   }
15 }
```

17. In Visual Studio Code, change the font color in `default.less` from **red** to **blue** and save the file.

18. Go back to the browser tab <http://localhost:4502/editor.html/content/training/us/en.html> and drag a text component to the en Page and add some dummy text ,as shown:



The screenshot shows the AEM富文本编辑器界面。顶部有编辑工具栏，显示了EN语言版本，以及Edit、Preview和其它设置按钮。下方是内容区域，有一个标题

TrainingProject

。在内容区中，有一个名为“Hello World Component”的文本组件。该组件内有三个文本块，每个文本块上方都有一个**Epic Journey**的标题。文本内容如下：

Epic Journey
Don't stop half way, go for the top!

Epic Journey
Don't stop half way, go for the top!

Epic Journey
Don't stop half way, go for the top!

中间有一段蓝色的Lorem ipsum假文：

 Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.

Congratulations! You have successfully tested AEM Sync Extension.

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=Yinkai15.aemsync>

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

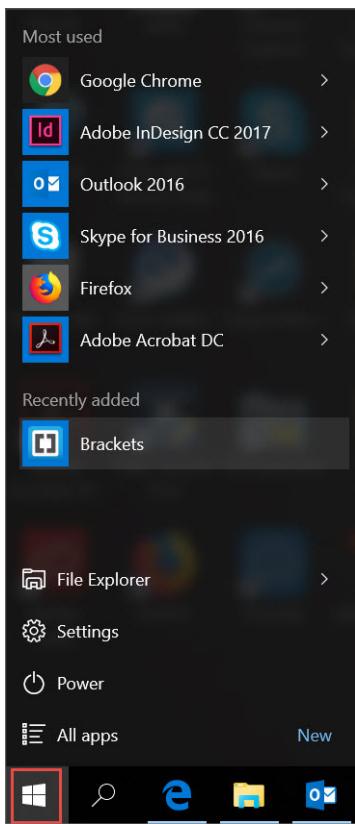
To install Brackets:

1. If you have an Internet connection, 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 the instructor for this course.
2. Based on your Operating System (OS), double-click the **Brackets.Release.1.13.msi** or **Brackets.Release.1.13.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.

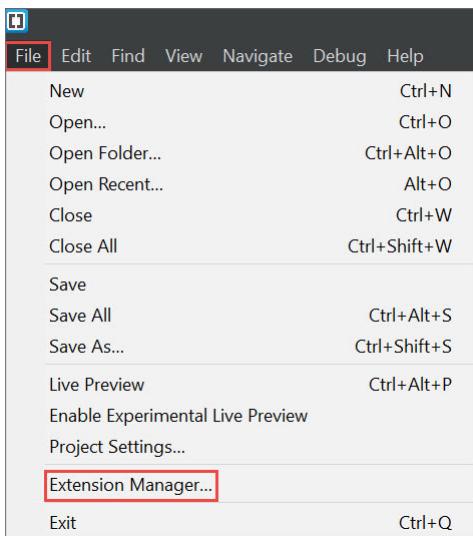


Note: If you are attending a VILT class using ReadyTech, the installation of Brackets has already been performed for you. You may skip ahead to step #2 of Exercise 2 : Make changes to the repository using Brackets.

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.



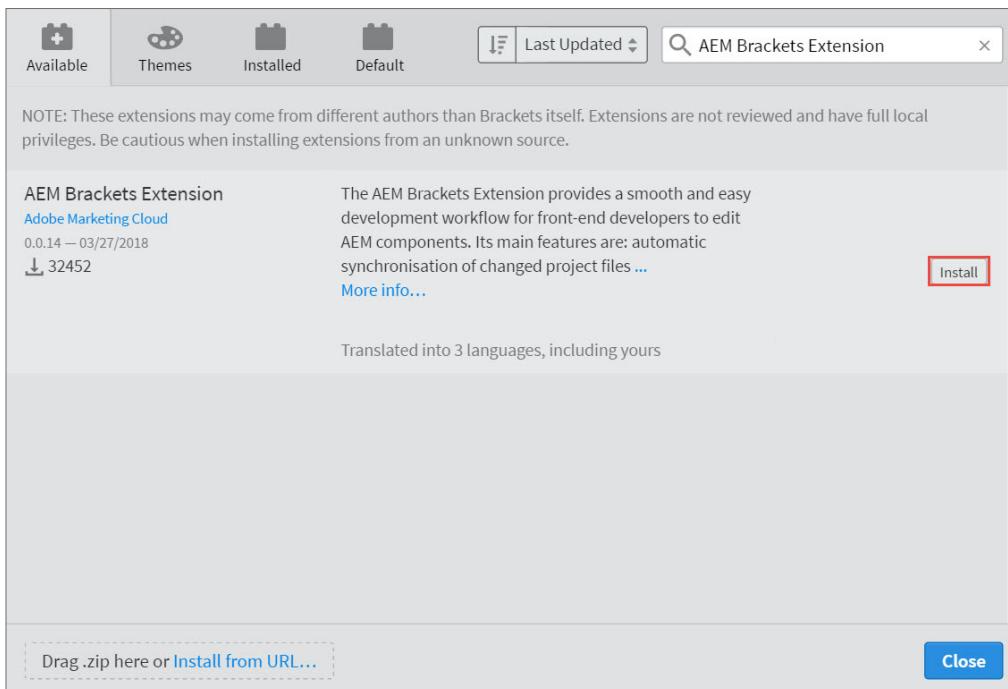
7. Click **File** and select **Extension Manager** from the list, as shown. A new wizard opens.



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**) provided to you by your instructor 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

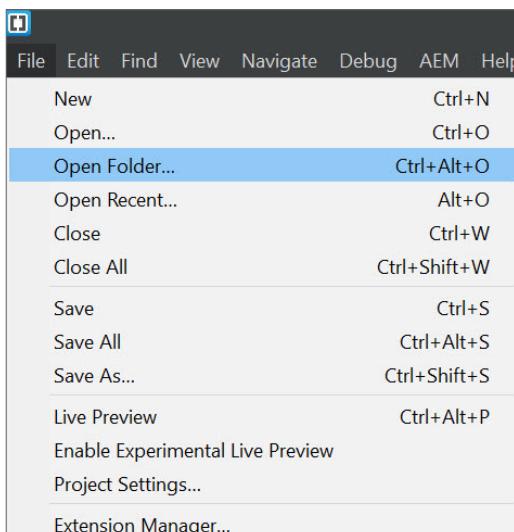
In this exercise, you will edit the contents of your project in Brackets. After completing this task, you can update the package content and share it with team members so they also see the same content in their AEM Services.

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 Wknd pages.

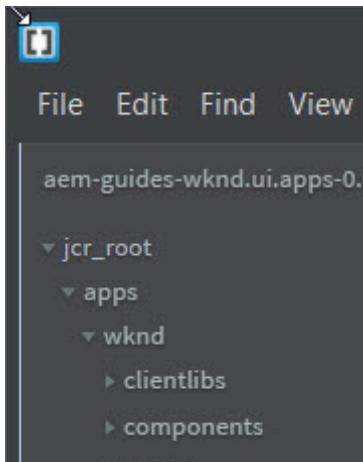
1. Open your AEM author service.

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

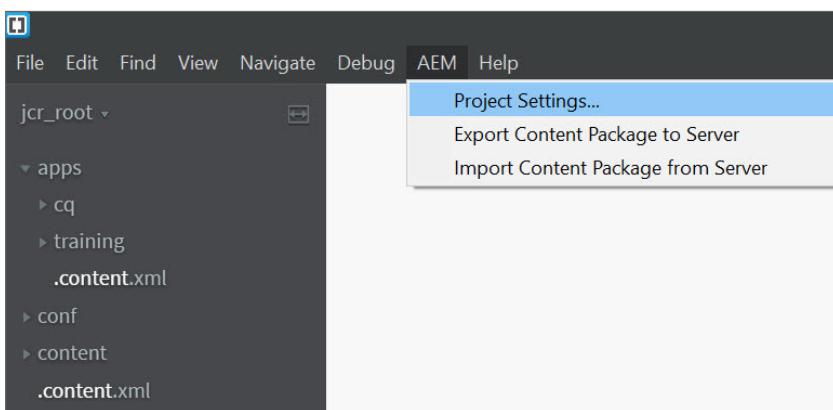
2. Navigate to the **Downloads** folder on your file system and look for the code content package (for example, **aem-guides-wknd.ui.apps-0.0.1-SNAPSHOT.zip**) that you used in the previous exercise.
3. Right-click the **aem-guides-wknd.ui.apps-0.0.1-SNAPSHOT.zip** and select **Extract to aem-guides-wknd.ui.apps-0.0.1-SNAPSHOT** to extract its contents to your local file directory.
4. Open the Brackets instance.
5. Click **File** and select **Open Folder** from the list, as shown. The **Choose a folder** dialog box opens.



6. Locate the **aem-guides-wknd.ui.apps-0.0.1-SNAPSHOT** folder of the unzipped (extracted) package, and click **Select Folder** (or click **Open** on a Mac system). The contents of the folder are listed in the left panel.
7. Select **jcr_root** and then expand the **apps** folder, as shown:



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



9. Provide the following details in their corresponding fields:

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

10. 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.

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

12. Replace the text "Wknd" with **Brackets** around line 21 as shown:



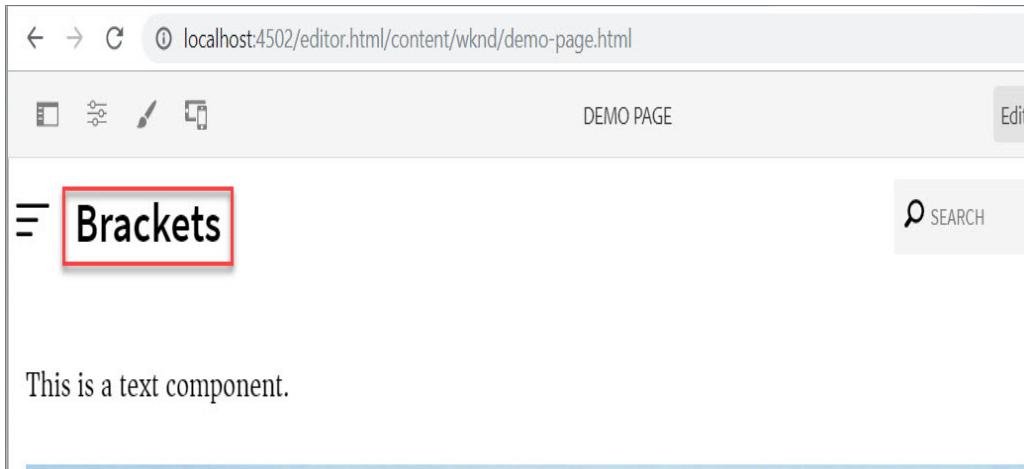
```
10 Unless required by applicable law or agreed to in writing, software
11 distributed under the License is distributed on an "AS IS" BASIS,
12 WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
13 See the License for the specific language governing permissions and
14 limitations under the License.
15 */
16
17 <!--/* Header Component for WKND Site */-->
18 <header class="wknd-header">
19   <div class="container">
20     <a class="wknd-header-logo" href="${properties.rootPath || currentStyle.rootPath @ extension='html'}" href="Brackets">
21       <div class="cmp-search--header" data-sly-resource="${'search' @
22         resourceType='wknd/components/structure/search'}"></div>
23     <div class="cmp-navigation--header" id="header-navbar" data-sly-
24       resource="${'navigation' @
25         resourceType='wknd/components/structure/navigation'}"></div>
26   </div>
27 </header>
```

13. 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 Wknd page that is using the page component you just modified:

14. Navigate to the tab where the AEM author service is running.
15. Open a new tab in your browser, type <http://localhost:4502/editor.html/content/wknd/demo-page.html> in the address bar, and then press **Enter**. The **demo** page opens.
16. Notice the text that you added in **header.html** now appears on the page, as shown:



References

You can use the following links for more information on:

- Creating an Adobe Experience Manager project using Lazybones:
https://helpx.adobe.com/experience-manager/using/aem_lazybones.html
- Creating an Adobe Experience Manager project using Lazybones:
https://helpx.adobe.com/experience-manager/using/aem_lazybones.html

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 (Optional)

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-
  chrome" 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**

The screenshot shows the 'Apache Sling Job Queue Configuration' interface. The 'Apache Sling Log Tracer' configuration is open. Two specific settings are highlighted with a red box: 'Enabled' (checkbox checked) and 'Recording Servlet Enabled' (checkbox checked). Below these, there are descriptive notes and checkboxes for 'Recording Cache Size' (50), 'Recording Cache Duration' (900), 'Compress Recording' (checkbox checked), and 'GZip Response' (checkbox checked). At the bottom, the 'Configuration Information' section shows 'Persistent Identity (PID)' as 'org.apache.sling.tracer.internal.LogTracer' and 'Configuration Binding' as 'Unbound or new configuration'. The 'Save' button at the bottom right is also highlighted with a red box.

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.

```
[Browsersync] Proxying: http://localhost:4502
[Browsersync] Access URLs:
  Local: http://localhost:3000
  External: http://10.0.172.221:3000
  -----
  UI: http://localhost:3001
  UI External: http://localhost:3001
[QR Code] QR code of current url available in browser console using qr() or __browerSync__.qr()
```



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

Exercise 3: Install content locally (Optional)

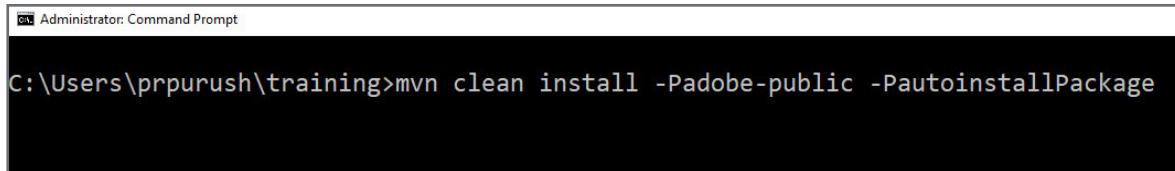
In this exercise, you will install the local Maven project into AEM.

 **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.

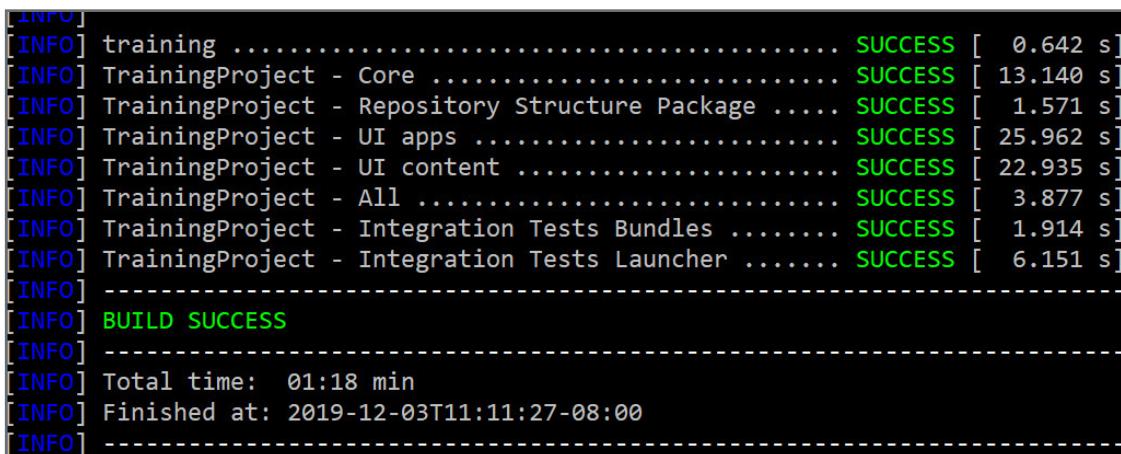
1. Confirm that your AEM service is running.
2. Open a terminal window and navigate to your <AEM project directory>. Run the following command to deploy your AEM project

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



A screenshot of a Windows Command Prompt window titled "Administrator: Command Prompt". The window shows the command \$mvn clean install -Padobe-public -PautoInstallSinglePackage being run in the directory C:\Users\prpurush\training. The output of the command is visible below the command line.

3. Verify the install is successful, as shown:



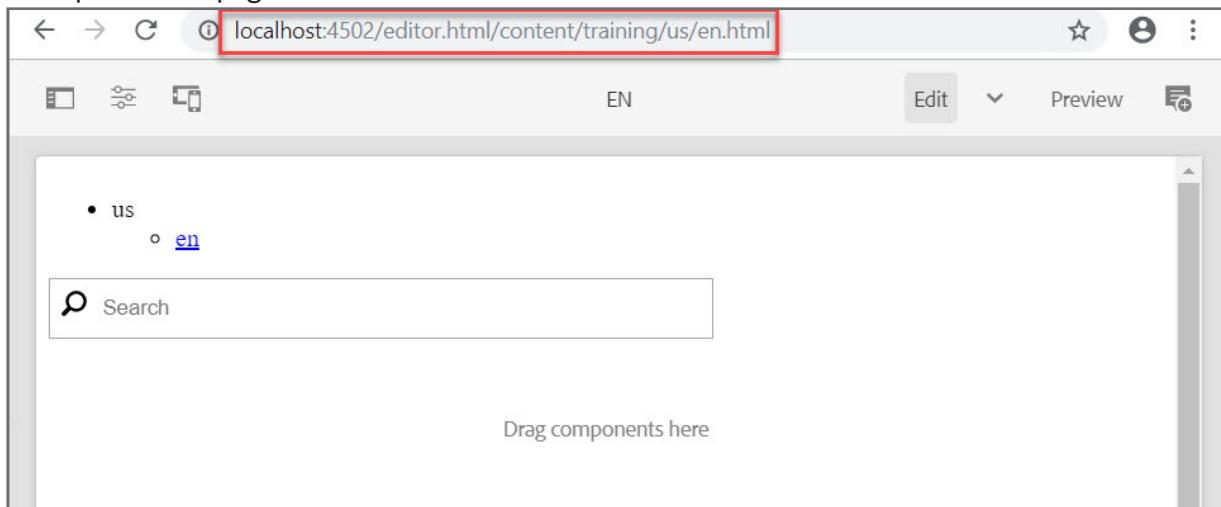
A screenshot of a terminal window showing the Maven build output. The output shows a series of [INFO] messages indicating the successful installation of various components (Core, Repository Structure Package, UI apps, UI content, All, Integration Tests Bundles, Integration Tests Launcher) with their respective execution times. It concludes with a BUILD SUCCESS message and summary statistics.

```
[INFO] [INFO] training ..... SUCCESS [ 0.642 s]
[INFO] [INFO] TrainingProject - Core ..... SUCCESS [ 13.140 s]
[INFO] [INFO] TrainingProject - Repository Structure Package ..... SUCCESS [ 1.571 s]
[INFO] [INFO] TrainingProject - UI apps ..... SUCCESS [ 25.962 s]
[INFO] [INFO] TrainingProject - UI content ..... SUCCESS [ 22.935 s]
[INFO] [INFO] TrainingProject - All ..... SUCCESS [ 3.877 s]
[INFO] [INFO] TrainingProject - Integration Tests Bundles ..... SUCCESS [ 1.914 s]
[INFO] [INFO] TrainingProject - Integration Tests Launcher ..... SUCCESS [ 6.151 s]
[INFO] -----
[INFO] [INFO] BUILD SUCCESS
[INFO] -----
[INFO] [INFO] Total time: 01:18 min
[INFO] [INFO] Finished at: 2019-12-03T11:11:27-08:00
[INFO] -----
```

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 a page you would like to change. 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:

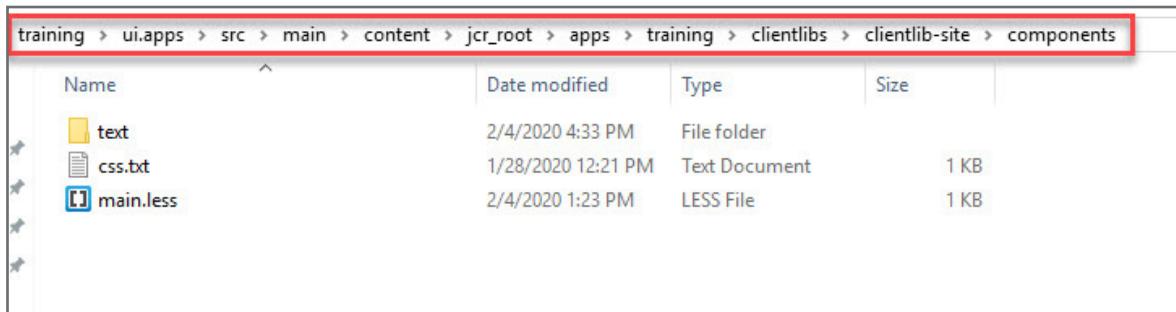
Name	
_MACOSX	2
text	2
css.txt	1
main.less	2

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**

Name	Date modified	Type
clientlib-base	12/3/2019 10:56 A...	File folder
clientlib-dependencies	12/3/2019 10:56 A...	File folder
clientlib-grid	12/3/2019 10:56 A...	File folder
clientlib-site	12/3/2019 10:56 A...	File folder

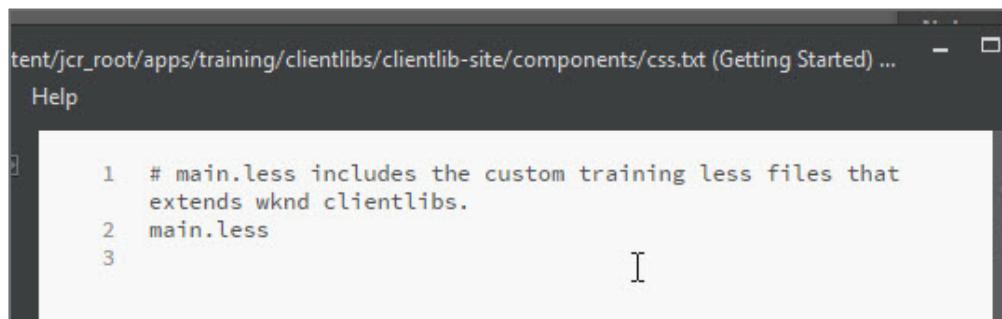
7. Paste the **text** folder from **Exercise_Files_TB > aemfed** into the **clientlib-site/components** folder (create **components** folder if needed):



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

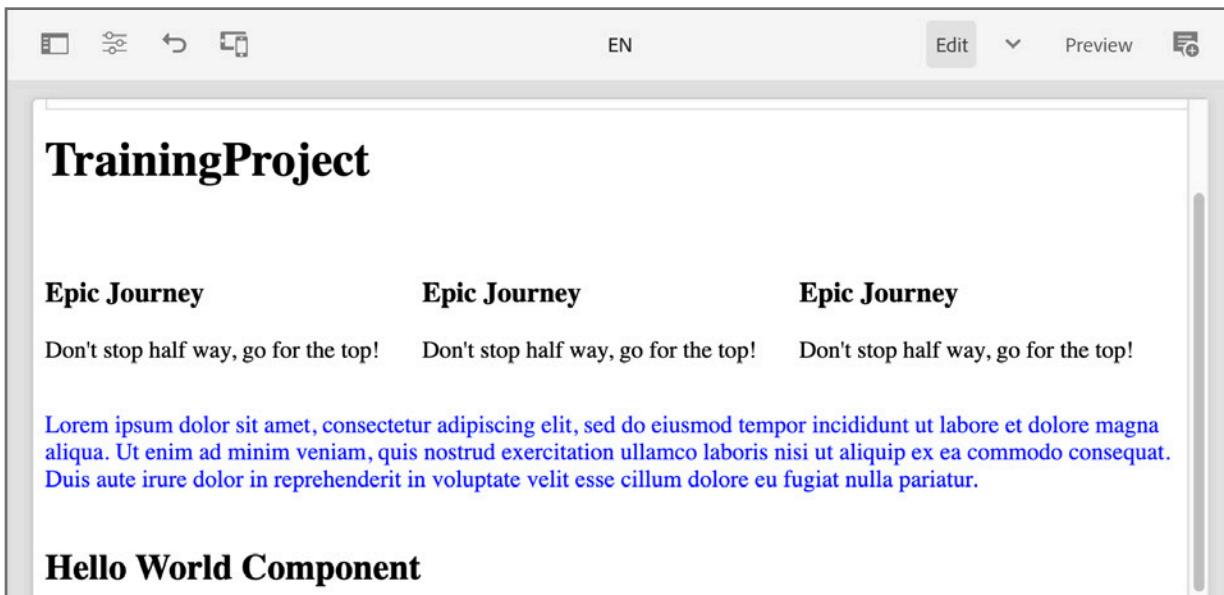
```
[localhost:4502] Special paths were changed, so rebuild clientlib tree
[localhost:4502] Get data from server: 252 ms
[localhost:4502] Process data: 18 ms
[localhost:4502] Clientlib tree: 271 ms
[localhost:4502] Rebuild clientlib tree: 271 ms
[Browsersync] Reloading Browsers...
ADD jcr_root/apps/trainingproject/clientlibs/.DS_Store
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.

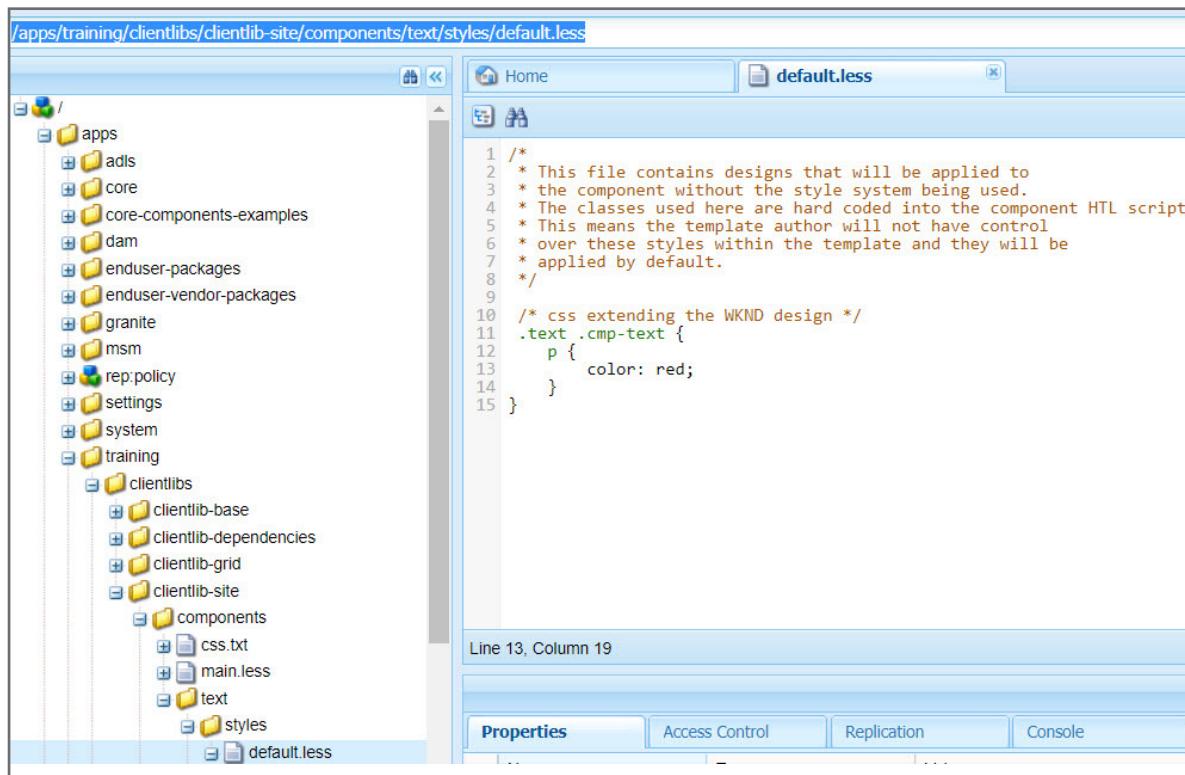
16. Using the IDE of your choice, navigate to **clientlib-site/components/text/styles** and open **default.less**. Update the style to a color of your choice, such as **red**.

> hero	8	*/
> languagenavigation	9	
> page	10	/* css extending the WKND a
< text	11	. text .cmp-text {
< styles	12	p {
{} default.less	13	color: red;
{} text.less	14	}
	15	}
.rss .content.xml		

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
```

- 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-ace6-0a7b02436c93)
[ERROR] LessCompilerImpl: failed to compile less /apps/trainingproject/clientlibs/clientlib-site/main.less: ParseError: Unrecognised input in /apps/trainingproject/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/default.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 is a cloud service allowing customers to build, test and deploy AEM applications hosted by Adobe Managed services. 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 is part of the Adobe Managed cloud services that enables organizations to self-manage Experience Manager in the cloud. 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:

- 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.

- 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.

- 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.

- Autoscaling

Cloud Manager detects the need for additional capacity when the production environment is subject to unusually high load and automatically brings additional capacity online via autoscaling feature. During an autoscaling event, Cloud Manager automatically triggers the autoscaling provisioning process, sends a notification of the autoscaling event, and brings online the additional capacity within minutes. The additional capacity will be provisioned in the production environment, in the same region(s) and matching the same system specifications as the running Dispatcher/Publish nodes.

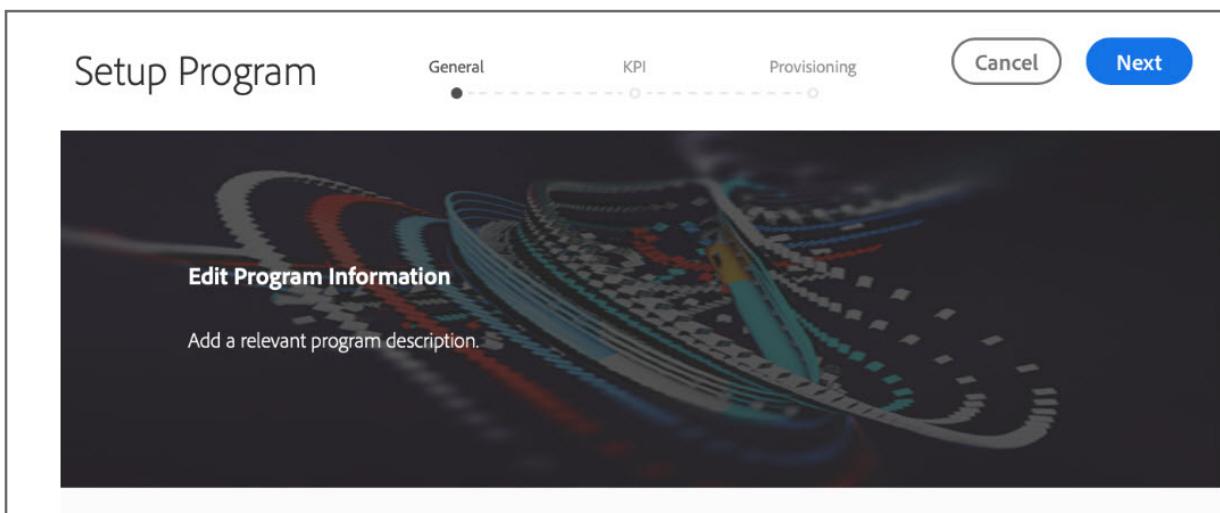
Setup a Program

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.

The screenshot shows two sections: 'AEM Sites' and 'AEM Assets'. Each section contains two KPI definitions with their respective thresholds.

KPI	Threshold	Unit
What's the 95th percentile response time that is acceptable to you? (Required)	<input type="text" value="3"/>	second(s)
How many Page Views per Minute under the peak load? (Required)	<input type="text" value="200"/>	page view(s)/minute
What's the 95th percentile processing time that is acceptable to you? (Required)	<input type="text" value="10"/>	second(s)
How many Assets should be uploaded per minute? (Required)	<input type="text" value="10"/>	number of assets

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.

The screenshot shows the 'Production Environment' and 'Non-Production Environment' sections. In the Production Environment, 'Autoscale is on' is selected, and the 'Max additional Publish-Dispatcher segments allowed' is set to 1. In the Non-Production Environment, 'On-demand scaling policy' is selected, and the 'Max additional Publish-Dispatcher segments allowed' is also set to 1.

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

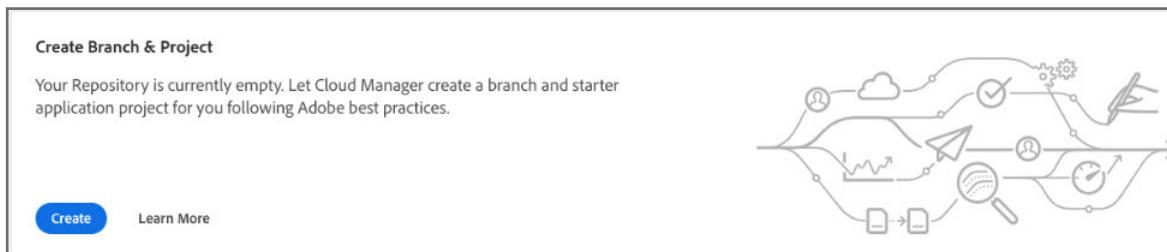
Creating an AEM Application

An empty git repository is provided to customers when they are on-boarded to Cloud Manager. Current Adobe Managed Services (AMS) customers or on-premise AEM customers who are migrating to AMS will generally already have their project code in Git (or another version control system) and will import their project into the Cloud Manager Git repository. New customers will not have existing projects.

Cloud Manager is now 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.

Create a Branch & Project	
Title	New Branch Name
We.Retail	master
Base Maven Group Id	Java Source Package
com.weretail	com.weretail
Base Maven Artifact Id	/apps Folder Name
aem-weretail-project	weretail
Maven Project Version	/content Folder Name
0.0.1-SNAPSHOT	weretail
Maven Project Name	/conf Folder Name
We.Retail	weretail
<input type="button" value="Cancel"/> <input type="button" value="Create"/>	

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.

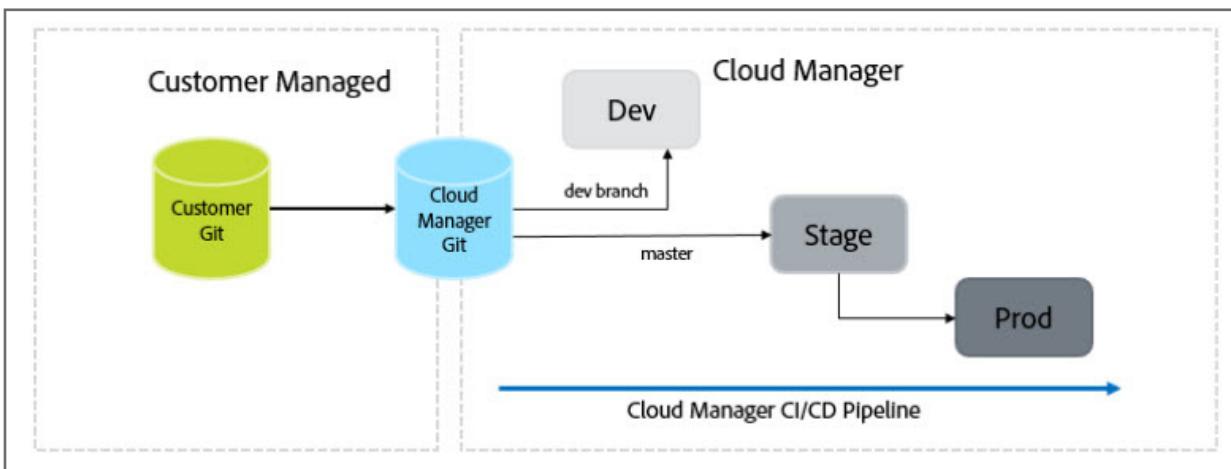
Building Environment Details

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.

As a best practice, you should always use the Cloud Manager's Git Repository, which comes empty without any branches configured or sample projects. To use the Cloud Manager's Git Repository, you will be provided with a private access token that will enable you to use any Git-compatible client to create branches, store and retrieve your code, list the commit history, etc.

On-Premise Repository

In some cases, you will have an existing Git Repository and want to keep using it. In these cases, 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:

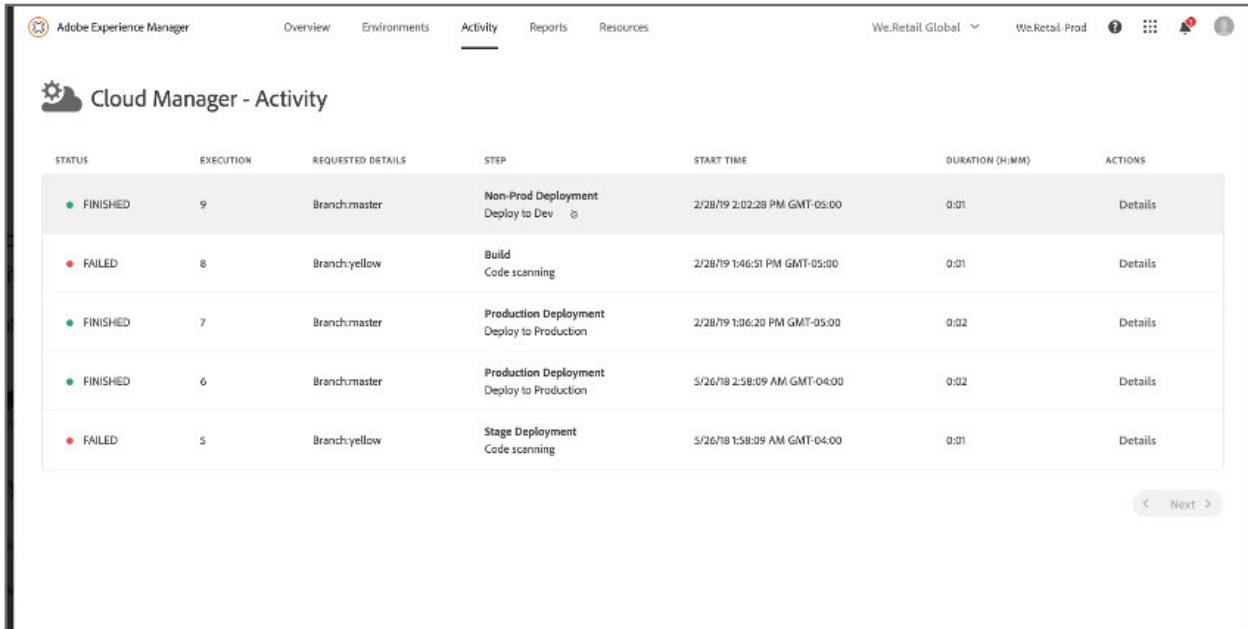
The CI/CD Production Pipeline configuration defines the trigger that will initiate the pipeline, parameters controlling the production deployment and performance test parameters. The CI/CD Production Pipeline is used to build and deploy code through Stage to the Production environment, decreasing time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



The screenshot shows the 'Activity' tab selected in the Cloud Manager UI. The page title is 'Cloud Manager - Activity'. The table lists five pipeline executions:

Status	Execution	Requested Details	Step	Start Time	Duration (H:M)	Actions
FINISHED	9	Branch:master	Non-Prod Deployment Deploy to Dev	2/28/19 2:02:28 PM GMT-05:00	0:01	Details
FAILED	8	Branch:yellow	Build Code scanning	2/28/19 1:46:51 PM GMT-05:00	0:01	Details
FINISHED	7	Branch:master	Production Deployment Deploy to Production	2/28/19 1:06:20 PM GMT-05:00	0:02	Details
FINISHED	6	Branch:master	Production Deployment Deploy to Production	5/26/18 2:58:09 AM GMT-04:00	0:02	Details
FAILED	5	Branch:yellow	Stage Deployment Code scanning	5/26/18 1:58:09 AM GMT-04:00	0:01	Details

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.

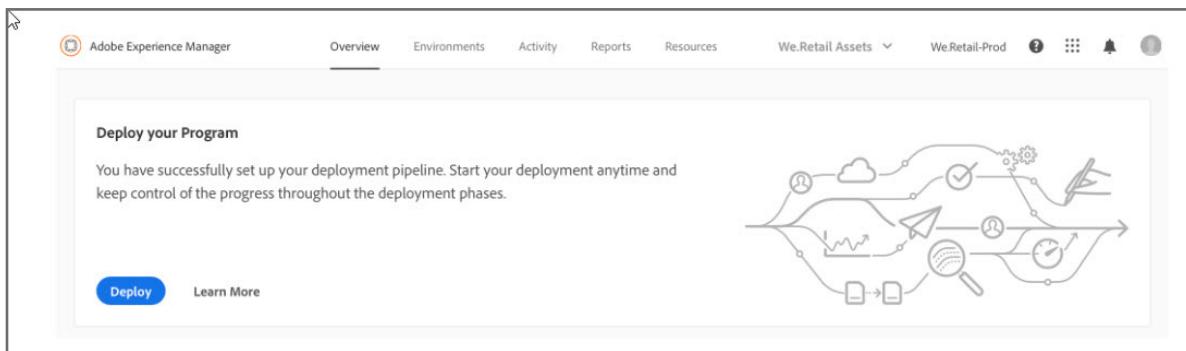
Once you configure your CI/CD Pipeline, you can deploy your code.

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 CI/CD 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:

Rule Key	Description	Type	Severity	Tags
3 squid:S2068	Credentials should not be hard-coded	Vulnerability	Blocker	cert,cwe,owa
4 squid:S2258	"javax.crypto.NullCipher" should not be used for anything other than testing	Vulnerability	Blocker	cwe,owasp-a
5 squid:S2278	Neither DES (Data Encryption Standard) nor DESede (3DES) should be used	Vulnerability	Blocker	cert,cwe,owa
6 squid:S3369	Security constraints should be defined	Vulnerability	Blocker	cwe,jee,owa
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-a
9 squid:S2076	Values passed to OS commands should be sanitized	Vulnerability	Critical	cwe,owasp-a
10 squid:S2077	SQL binding mechanisms should be used	Vulnerability	Critical	cert,cwe,hiber
11 squid:S2078	Values passed to LDAP queries should be sanitized	Vulnerability	Critical	cert,cwe,owa
12 squid:S2089	HTTP referrers should not be relied on	Vulnerability	Critical	cwe,owasp-a
13 squid:S2245	Pseudorandom number generators (PRNGs) should not be used in secure contexts	Vulnerability	Critical	cert,cwe,owa
14 squid:S2254	"HttpServletRequest.getRequestedSessionId()" should not be used	Vulnerability	Critical	cwe,owasp-a
15 squid:S2257	Only standard cryptographic algorithms should be used	Vulnerability	Critical	cwe,owasp-a
16 squid:S2277	Cryptographic RSA algorithms should always incorporate OAEP (Optimal Asymmetric Encryption Padding)	Vulnerability	Critical	cwe,owasp-a
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-a
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_TRAV	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-
48.      * service) that will be modifying the permissions
49.      * @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")){
92.                 //Adds permissions to edit the payload
93.                 logger.info("Adding permissions");
94.                 accessControlManager.setPolicy(payload, acl);
95.                 logger.info("Added modify permissions to " + payload + " for " + userID);
96.             }else if(permission.equals("remove")){
97.                 //Removes permissions to edit the payload
98.                 logger.info("Removing permissions");
99.                 accessControlManager.removePolicy(payload, acl);
100.                logger.info("Removed modify permissions to " + payload + " for " + userID);
101.            }
102.        }
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
File Location	Line Num	Issue	Type	Severity	Effort	Rule
1 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
2 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
3 com.adobe:training.core:src/main/java/com/adobe/training/core/AutoAssignACL.java	108	Do not use Exception.printStackTrace() as Code Sme	Minor	15min		CQRules:CQBP-44---ExceptionPrintStackTrace
4 com.adobe:training.core:src/main/java/com/adobe/training/core/AutoAssignACL.java	48	Catch a list of specific exception subty	Code Sme	Major	15min	squid:S2221
5 com.adobe:training.core:src/main/java/com/adobe/training/core/listeners/ReplicationListener.java	50	Do not use Exception.printStackTrace() as Code Sme	Minor	15min		CQRules:CQBP-44---ExceptionPrintStackTrace
6 com.adobe:training.core:src/main/java/com/adobe/training/core/listeners/ReplicationListener.java	48	"NullPointerException" could be thr Bug		Major	10min	squid:S2259
7 com.adobe:training.core:src/main/java/com/adobe/training/core/listeners/TitlePropertyListener.java	40	Use constant JCR_TITLE from interface	Code Sme	Minor		AEM Rules:AEM-2
8 com.adobe:training.core:src/main/java/com/adobe/training/core/listeners/TitlePropertyListener.java	50	Do not use Exception.printStackTrace() as Code Sme	Minor	15min		CQRules:CQBP-44---ExceptionPrintStackTrace
9 com.adobe:training.core:src/main/java/com/adobe/training/core/ReplicationLogger.java	48	Use try-with-resources or close this "I" Bug		Blocker	5min	squid:S2095
10 com.adobe:training.core:src/main/java/com/adobe/training/core/schedulers/StockImportSchedule	135	HttpClient instances should always ha Bug		Critical	15min	CQRules:ConnectionTimeoutMechanism
11 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
12 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
13 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
14 com.adobe:training.core:src/main/java/com/adobe/training/core/schedulers/StockImportSchedule	58	Make "logger" transient or serializable	Code Sme	Critical	30min	squid:S1948
15 com.adobe:training.core:src/main/java/com/adobe/training/core/servlets/CSVPageCreator.java	61	Make "replicator" transient or serializab	Code Sme	Critical	30min	squid:S00112
16 com.adobe:training.core:src/main/java/com/adobe/training/core/servlets/CSVPageCreator.java	94	Define and throw a dedicated excepti	Code Sme	Major	20min	squid:S2221
17 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:S1948
18 com.adobe:training.core:src/main/java/com/adobe/training/core/servlets/CSVPageCreator.java	50	Make "replicator" transient or serializab	Code Sme	Critical	30min	squid:S1948
19 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
20 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
21 com.adobe:training.core:src/main/java/com/adobe/training/core/servlets/PageCreator.java	73	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	74	Do not use Exception.printStackTrace() as Code Sme	Minor	15min		CQRules:CQBP-44---ExceptionPrintStackTrace
23 com.adobe:training.core:src/main/java/com/adobe/training/core/servlets/SearchServlet.java	44	Don't use format strings that may be e	Vulnerabi	Major	15min	CQRules:CWE-134
24 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
25 com.adobe:training.core:src/main/java/com/adobe/training/core/servlets/TitleSlingServlet.java						

5. Notice the error occurred 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>