

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 Unpublished-rights reserved under the copyright laws of the United States.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.

Contents

| | | |
|----------|--|----|
| Module 6 | Creating a Project using Maven | 3 |
| | Working with Maven | 4 |
| | Working with Node.js and NPM | 5 |
| | Exercise 1: Install build tools (Local only) | 6 |
| | AEM Archetype | 11 |
| | Common modules for an AEM Project: | 12 |
| | AEM Project's Content Package Structure | 13 |
| | Exercise 2: Create an AEM project | 19 |
| | Install to AEM with Maven Profiles | 24 |
| | Exercise 3: Install the project into AEM | 26 |
| | References | 35 |

Creating a Project using Maven

Introduction

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

Objectives

After completing this course, you will be able to:

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

Working with Maven

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

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

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

Working with Node.js and NPM

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

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

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

Exercise 1: Install build tools (Local only)

In this exercise, you will install and configure Maven.

This exercise includes the following tasks:

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



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

Task 1: Install Maven 3.6.x

Windows

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

Mac

1. Open a terminal window and type **mvn -version**. If you get "mvn: command not found" then you need to install Maven.
2. To install Maven, follow this online tutorial <https://crunchify.com/how-to-install-maven-on-mac-os-x-manually-fix-unsupportedclassversionerror-orgapachemavenclimavencli/> or follow the generic steps below:
3. Open a terminal window and type, as shown. The below command will download Maven to / Users/<user>/.

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

4. Type:

```
unzip apache-maven-3.6.1-bin.zip
```

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

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

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

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

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

7. Reload your bash_profile:

```
source ~/.bash_profile
```

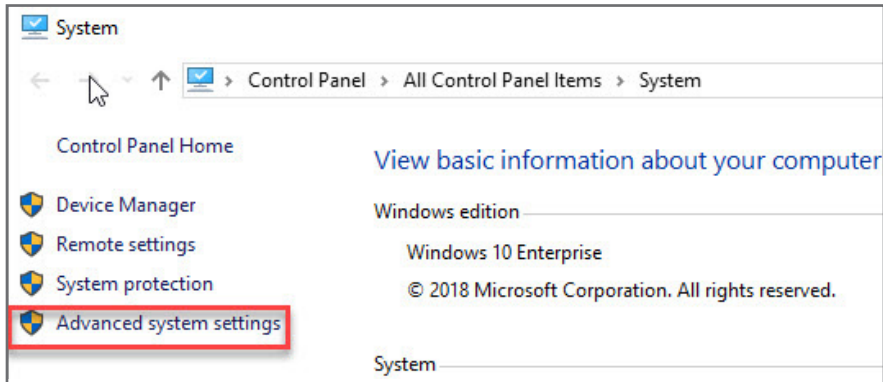
8. Test to see if you can successfully run Maven:

```
mvn -version
```

Task 2: Configure Environment Variables (Windows only)

In this task, you will configure the environment variable.

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



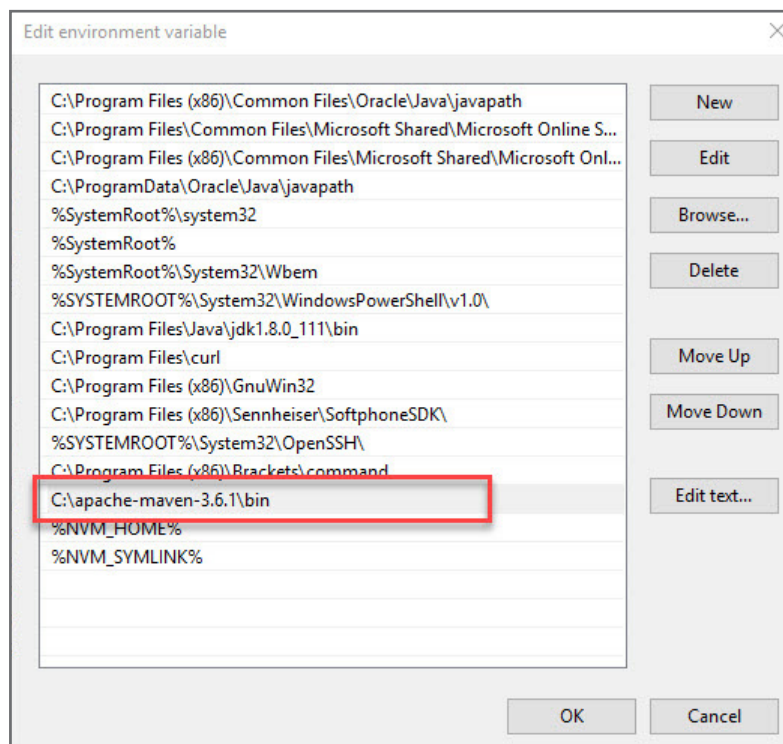
The **System Properties** window opens.

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



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

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



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

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

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

Task 3: Install Node Version Manager (NVM)

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



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

1. Download the latest Long Term Support (LTS) installer, for your operating system, from the node.js org site (or obtain the installer files from your instructor).
<https://nodejs.org/en/download/>
2. Install Node.js.
 - **Windows**
 - › Double-click on the .msi file (example node-v10.16.3-x86.msi) to start the installation and follow the instructions in the installer.
 - **Mac OSX**
 - › Double-click on the .pkg file (example node-v10.16.3.pkg) to start the installation and follow the instructions in the installer.

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

AEM Archetype

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

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

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

- Commerce Enabled: For projects that want to integrate AEM Commerce with commerce solutions like Magento using the Commerce Core Components.
- Example Code: Checkout the HelloWorld component, and the sample models, servlets, filters, and schedulers.
- Open Sourced: If something is not as it should, contribute your improvements!

Common modules for an AEM Project:

- all - A container package that includes all artifacts of a project, including any 3rd party dependencies
- ui.apps - Contains all the immutable code in /apps
- core - Contains the OSGi bundle
- ui.content - Contains all mutable content and configuration
- ui.config - Contains all OSGi configurations for an application
- it.launcher - Used for deploying the sling testing framework
- it.tests - Contains sling tests
- ui.apps.structure - Module that contains project roots
- dispatcher - (required for Managed Services and Cloud Service) Module used to configure the static web server in front of AEM that caches content
- ui.frontend - (optional) Module that contains a front end project built for Webpack, React, or AngularJS and installed as an AEM client library.

AEM Project's Content Package Structure

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

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

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

Mutable vs. Immutable content

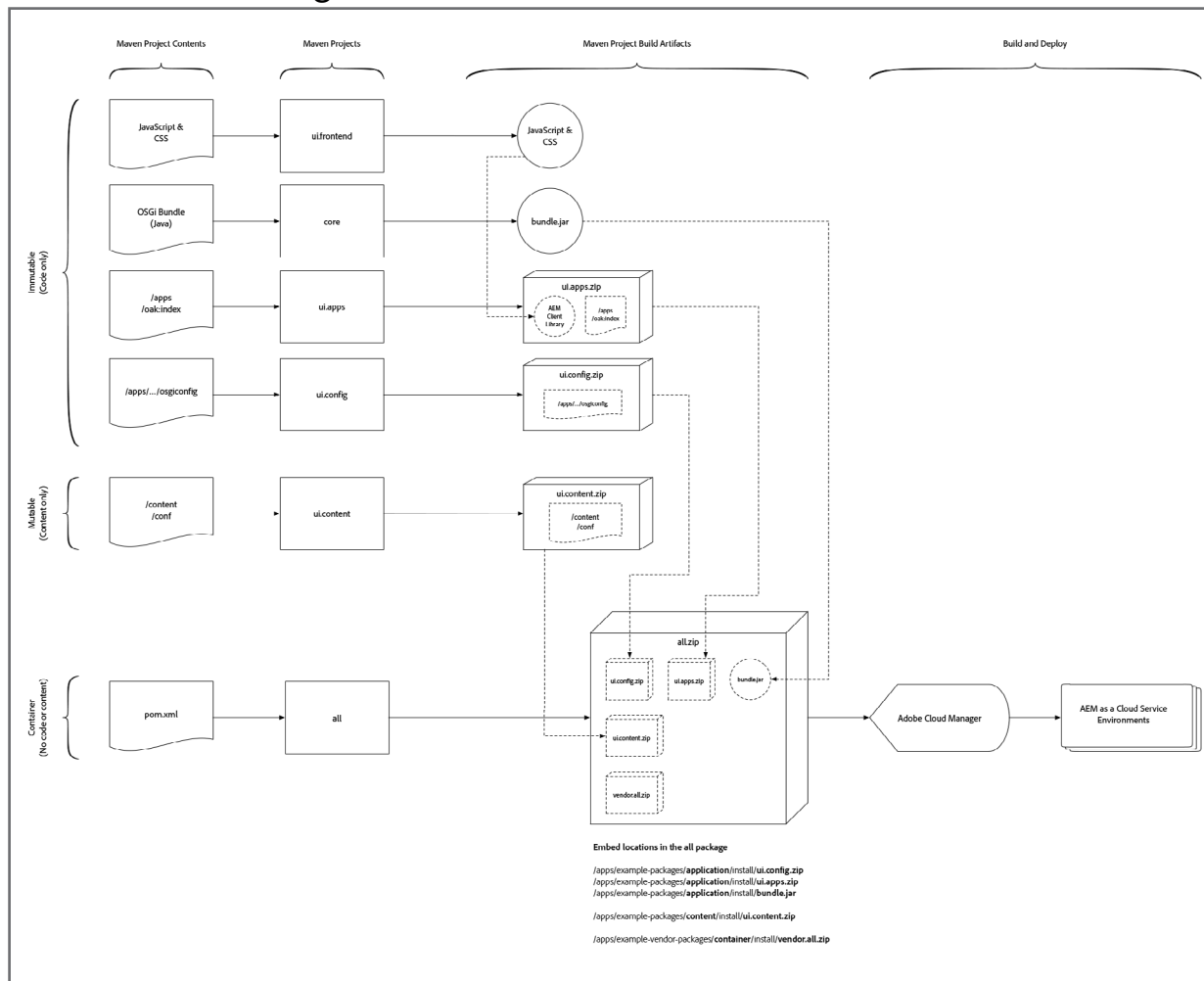
`/apps` and `/libs` are considered immutable areas of Experience Manager as they cannot be changed (create, update, delete) after deployment into the AEM Cloud Service (for example, at runtime). Any attempt to change an immutable area at runtime will fail.

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



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

Recommended Package Structure:



Code Packages / OSGi Bundles

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

- The ui.config package, contains all OSGi configurations:
 - › Organizational folder containing run mode specific OSGi config definitions
 - » /apps/my-app/osgiconfig
 - › Common OSGi configuration folder containing default OSGi configurations that apply to all target AEM as a Cloud Service deployment targets
 - » /apps/my-app/osgiconfig/config
 - › Run mode-specific OSGi configuration folders that contains default OSGi configurations that apply to all target AEM as a Cloud Service deployment targets
 - » /apps/my-app/osgiconfig/config.<author|publish>.<dev|stage|prod>
 - › Repo Init OSGi configuration scripts
 - » Repo Init is the recommended way to deploy (mutable) content that is logically part of the AEM application. The Repo Init OSGi configurations should be places in the appropriate config.<runmode> folder as outlined above, and be used to define:
 - i. Baseline content structures
 - ii. Users
 - iii. Service Users
 - iv. Groups
 - v. ACLs (permissions)
- The all package is a container package that ONLY includes the ui.apps and ui.content packages as embeds. The "all" package must not have any content of its own, but rather delegate all deployment to the repository to its subpackages.

Content Packages

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

Container Packages

- The all package is a container package that ONLY includes deployable artifacts, the OSGi bundle Jar file, ui.apps, ui.config and ui.content packages as embeds. The all package must not have any content or code of its own, but rather delegate all deployment to the repository to its sub-packages or OSGi bundle Jar files.

Packages are now included using the Maven FileVault Package Maven plugin's embedded configuration, rather than the <subPackages> configuration.

For complex Experience Manager deployments, it may be desirable to create multiple ui.apps, ui.config and ui.content projects/packages that represent specific sites or tenants in AEM. If this is done, ensure the split between mutable and immutable content is respected, and the required content packages and OSGi bundle Jar files are embedded as sub-packages in the all container content package.

For example, a complex deployment content package structure might look like this:

- all content package embeds the following packages, to create a singular deployment artifact
 - › common.ui.apps deploys code required by both site A and site B
 - › site-a.core OSGi bundle Jar required by site A
 - › site-a.ui.apps deploys code required by site A
 - › site-a.ui.config deploys OSGi configurations required by Site A
 - › site-a.ui.content deploys content and configuration required by site A
 - › site-b.core OSGi bundle Jar required by site B
 - › site-b.ui.apps deploys code required by site B
 - › site-b.ui.config deploys OSGi configurations required by site B
 - › site-b.ui.content deploys content and configuration required by site B

POM Files

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

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

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

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

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

This API jar file contains:

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

AEM SDK API for Cloud Service

AEM as a Cloud Service uses a dependency called the AEM SDK API. To use the SDK jar file, you must add the following elements to the pom.xml file:

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


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

UberJar for AEM 6.5 and earlier

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

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

```
651      <!-- Adobe AEM Dependencies -->
652      <dependency>
653          <groupId>com.adobe.aem</groupId>
654          <artifactId>uber-jar</artifactId>
655          <version>6.5.9</version>
656          <scope>provided</scope>
657      </dependency>
```

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

Exercise 2: Create an AEM project

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

Prerequisite:

- Maven is installed

This exercise includes the following tasks:

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



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

Task 1: (Optional) Configure the profile for Maven

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

3. Open **settings.xml** using a text editor and replace the Profile settings with the content from **Exercise_Files_TB\Project_using_Maven\adobe-archetype-repo.xml**, as shown:

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

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

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

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

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

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


4. Save the changes.



Note: If you do not have a **settings.xml** file, then copy **Exercise_Files_TB\Project_using_Maven\adobe-archetype-repo.xml** to **\<user>\.m2** directory and rename the file to **settings.xml**.

Task 2: Create an AEM project

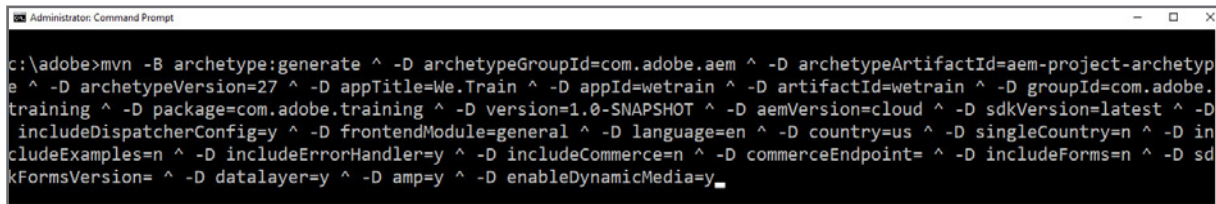
1. Open a **Command Prompt** window by right-clicking the **Command Prompt** app/icon and choose as **Run as administrator**.
2. In your Command Prompt window, navigate to the **adobe** directory which will be your working directory for this training:

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

3. Navigate to **Exercise_Files_TB\Project_using_Maven** and copy the content from the file specified by your instructor. For example: **generate-archetype27-wetrain-cloud.bat** on Windows (For Cloud Service on Mac, use **generate-archetype27-wetrain-cloud.sh**. For 6.5, use **generate-archetype27-wetrain-6.5.bat** on Windows and **generate-archetype27-wetrain-6.5.sh** on Mac):

```
mvn -B archetype:generate ^
-D archetypeGroupId=com.adobe.aem ^
-D archetypeArtifactId=aem-project-archetype ^
-D archetypeVersion=27 ^
-D appTitle=We.Train ^
-D appId=wetrain ^
-D artifactId=wetrain ^
-D groupId=com.adobe.training ^
-D package=com.adobe.training ^
-D version=1.0-SNAPSHOT ^
-D aemVersion=cloud ^
-D sdkVersion=latest ^
-D includeDispatcherConfig=y ^
-D frontendModule=general ^
-D language=en ^
-D country=us ^
-D singleCountry=n ^
-D includeExamples=n ^
-D includeErrorHandler=y ^
-D includeCommerce=n ^
-D commerceEndpoint= ^
-D includeForms=n ^
-D sdkFormsVersion= ^
-D datalayer=y ^
-D amp=y ^
-D enableDynamicMedia=y
```

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



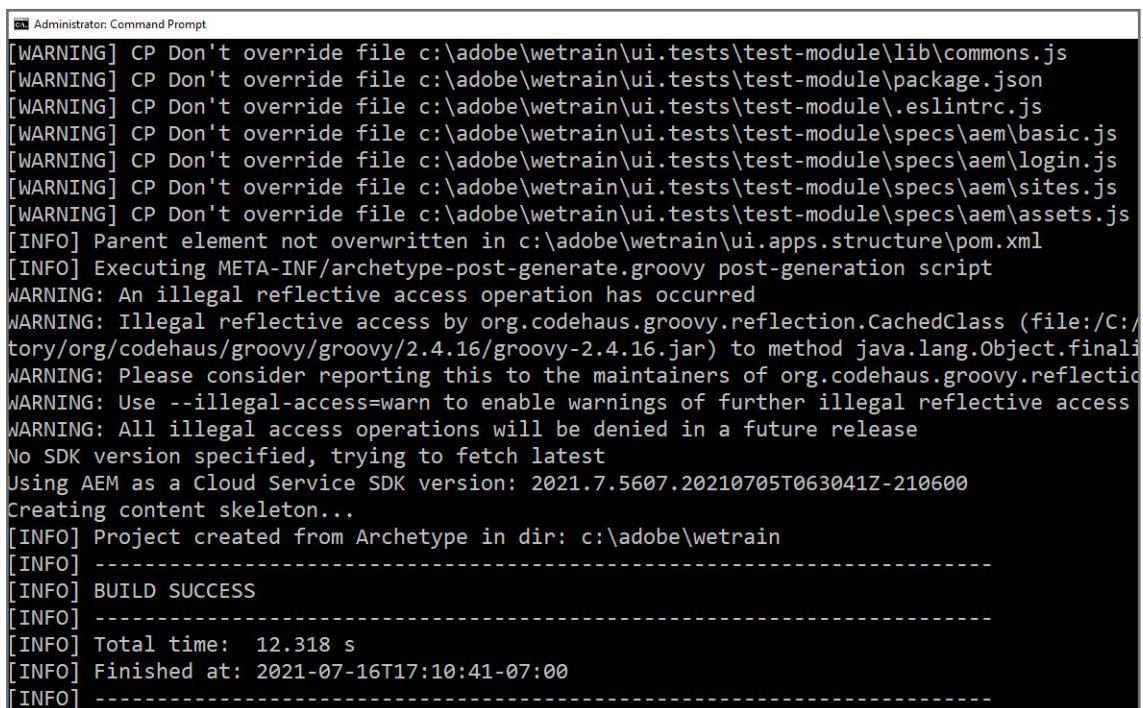
```
c:\adobe>mvn -B archetype:generate ^ -D archetypeGroupId=com.adobe.aem ^ -D archetypeArtifactId=aem-project-archetype ^ -D archetypeVersion=27 ^ -D appTitle=We.Train ^ -D appId=wetrain ^ -D artifactId=wetrain ^ -D groupId=com.adobe.training ^ -D package=com.adobe.training ^ -D version=1.0-SNAPSHOT ^ -D aemVersion=cloud ^ -D sdkVersion=latest ^ -D includeDispatcherConfig=y ^ -D frontendModule=general ^ -D language=en ^ -D country=us ^ -D singleCountry=n ^ -D includeExamples=n ^ -D includeErrorHandler=y ^ -D includeCommerce=n ^ -D commerceEndpoint= ^ -D includeForms=n ^ -D sdkFormsVersion= ^ -D datalayer=y ^ -D amp=y ^ -D enableDynamicMedia=y
```

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



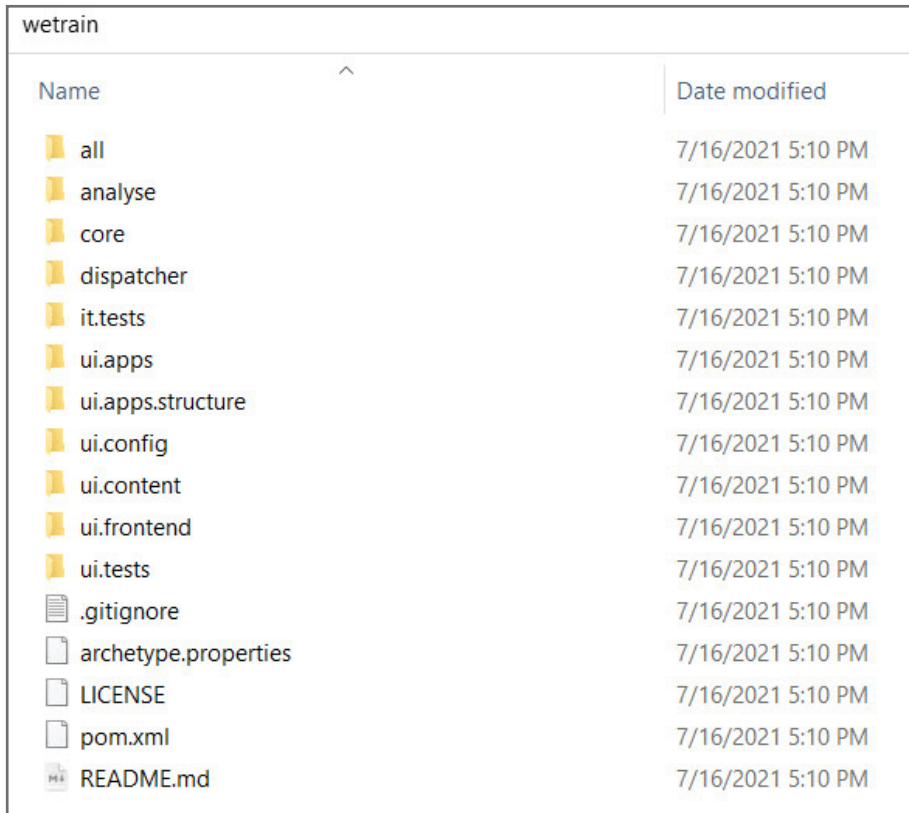
Note: If you are prompted to confirm the parameters type **y** and press <Enter>.

5. Verify Build Success, as shown:



```
[WARNING] CP Don't override file c:\adobe\wetrain\ui.tests\test-module\lib\commons.js
[WARNING] CP Don't override file c:\adobe\wetrain\ui.tests\test-module\package.json
[WARNING] CP Don't override file c:\adobe\wetrain\ui.tests\test-module\eslinttrc.js
[WARNING] CP Don't override file c:\adobe\wetrain\ui.tests\test-module\specs\aem\basic.js
[WARNING] CP Don't override file c:\adobe\wetrain\ui.tests\test-module\specs\aem\login.js
[WARNING] CP Don't override file c:\adobe\wetrain\ui.tests\test-module\specs\aem\sites.js
[WARNING] CP Don't override file c:\adobe\wetrain\ui.tests\test-module\specs\aem\assets.js
[INFO] Parent element not overwritten in c:\adobe\wetrain\ui.apps.structure\pom.xml
[INFO] Executing META-INF/archetype-post-generate.groovy post-generation script
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.codehaus.groovy.reflection.CachedClass (file:/C:/Program Files/Java/jdk-11.0.10/bin/../lib/modules/org.codehaus.groovy/groovy/2.4.16/groovy-2.4.16.jar) to method java.lang.Object.finalize()
WARNING: Please consider reporting this to the maintainers of org.codehaus.groovy.reflection.CachedClass
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
No SDK version specified, trying to fetch latest
Using AEM as a Cloud Service SDK version: 2021.7.5607.20210705T063041Z-210600
Creating content skeleton...
[INFO] Project created from Archetype in dir: c:\adobe\wetrain
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 12.318 s
[INFO] Finished at: 2021-07-16T17:10:41-07:00
[INFO] -----
```

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



The screenshot shows a Windows File Explorer window titled 'wetrain'. The address bar shows the path 'wetrain'. The main area displays a list of files and folders. The columns are 'Name' and 'Date modified'. All files and folders were last modified on 7/16/2021 at 5:10 PM. The files and folders are: 'all', 'analyse', 'core', 'dispatcher', 'it.tests', 'ui.apps', 'ui.apps.structure', 'ui.config', 'ui.content', 'ui.frontend', 'ui.tests', '.gitignore', 'archetype.properties', 'LICENSE', 'pom.xml', and 'README.md'.

| Name | Date modified |
|----------------------|-------------------|
| all | 7/16/2021 5:10 PM |
| analyse | 7/16/2021 5:10 PM |
| core | 7/16/2021 5:10 PM |
| dispatcher | 7/16/2021 5:10 PM |
| it.tests | 7/16/2021 5:10 PM |
| ui.apps | 7/16/2021 5:10 PM |
| ui.apps.structure | 7/16/2021 5:10 PM |
| ui.config | 7/16/2021 5:10 PM |
| ui.content | 7/16/2021 5:10 PM |
| ui.frontend | 7/16/2021 5:10 PM |
| ui.tests | 7/16/2021 5:10 PM |
| .gitignore | 7/16/2021 5:10 PM |
| archetype.properties | 7/16/2021 5:10 PM |
| LICENSE | 7/16/2021 5:10 PM |
| pom.xml | 7/16/2021 5:10 PM |
| README.md | 7/16/2021 5:10 PM |

You have successfully created an AEM project.

Install to AEM with Maven Profiles

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

Profiles in parent POM

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

Profiles in all POM

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

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

- `/apps/<your-project>-packages`
 - › `application/install`
 - » `ui.apps.package.here.zip` (also contains core bundle)
 - › `content/install`
 - » `ui.content.package.here.zip`
- `/apps/<your-project>-vendor-packages` (for example, Core components)
 - › `application/install`
 - » `ui.apps.vendor.package.here.zip`
 - › `content/install`
 - » `ui.content.vendor.package.here.zip`



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

Example Maven Commands:

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

Exercise 3: Install the project into AEM

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

Prerequisite:

- AEM author service is up and running

This exercise includes the following tasks:

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

Task 1: Update the POM file

When you create a new project from the archetype, it automatically sets the SDK or Uberjar version for you. When developing your application, you want to make sure the build in your POM matches the SDK build or 6.5 Service Pack number you're using. Depending how often you download and update your local development environment, you might need to update the build in your POM.

In this task you will update the dependency version in your POM to match either the SDK build if your using Cloud Service or the AEM Service pack on the Uberjar if you're using 6.5.

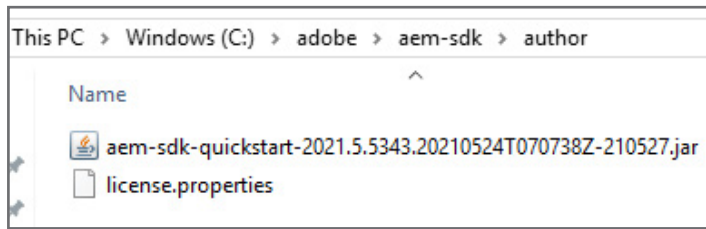
Depending on the type of AEM deployment you have, you will need to update a different dependency:

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

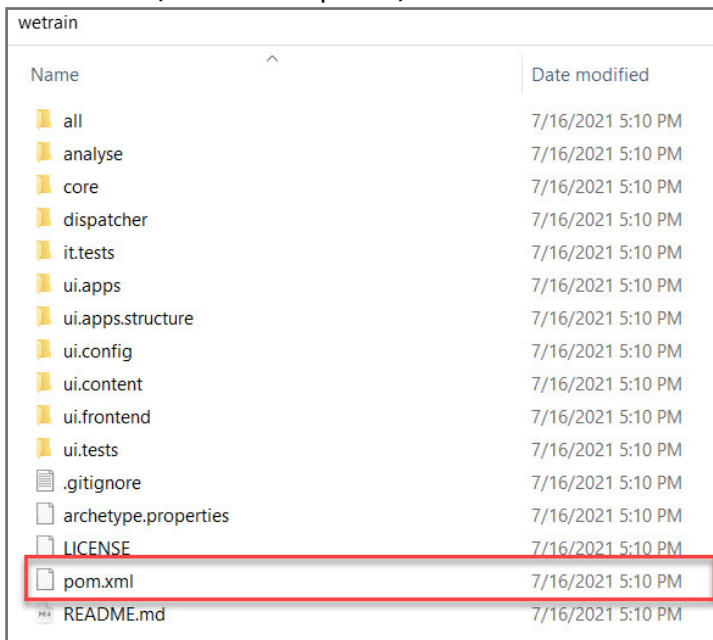
Follow these steps for AEM as a Cloud Service:

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

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



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



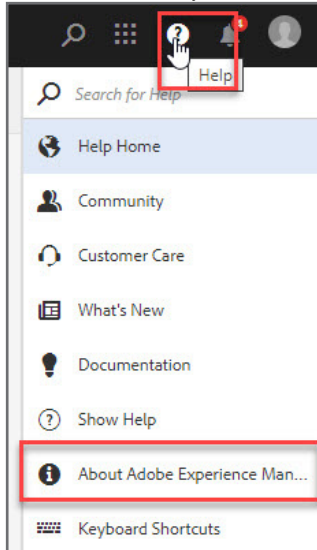
4. Find the property value **<aem.sdk.api>** (near the top) and paste the build number that you copied from the SDK zip, as shown:



5. Save the file.

Follow these steps for AEM 6.5 and lower Only:

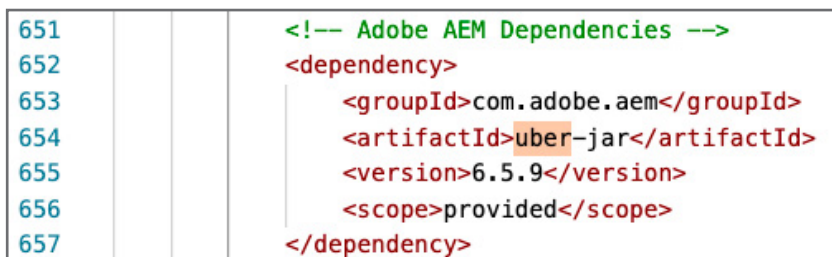
1. If you are unsure about the version of AEM Service Pack you are using, open AEM in the browser: <http://localhost:4502/>
2. Click on the Help icon on the top right corner > About Adobe Experience Manager



3. Take note of the AEM version:



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



Task 2: Install the project into AEM

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



Note: This is the project directory <AEM Project> 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
```

```
Administrator: Command Prompt
c:\adobe\wetrain>mvn clean install -Padobe-public -PautoInstallSinglePackage,
```



Note: The project build may take a few minutes.

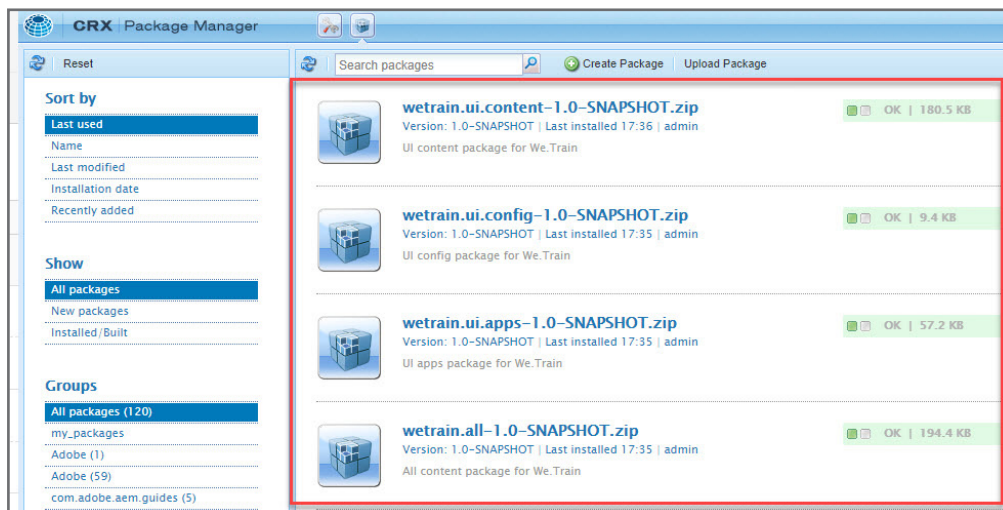
3. Verify the build is success, as shown:


```
Administrator: Command Prompt
[INFO] - Executing Bundle Resources Check [bundle-resources]...
[INFO] - Executing Requirements Capabilities check [requirements-capabilities]...
[INFO] Analyzing feature 'com.adobe.training:wetrain.analyse:slingosgifeature:agg
finished : 0 warnings, 0 errors.
[INFO] -----
[INFO] Reactor Summary for wetrain 1.0-SNAPSHOT:
[INFO]
[INFO] wetrain ..... SUCCESS [ 0.623 s]
[INFO] We.Train - Core ..... SUCCESS [ 22.465 s]
[INFO] We.Train - UI Frontend ..... SUCCESS [01:26 min]
[INFO] We.Train - Repository Structure Package ..... SUCCESS [ 1.959 s]
[INFO] We.Train - UI apps ..... SUCCESS [ 32.025 s]
[INFO] We.Train - UI content ..... SUCCESS [ 27.157 s]
[INFO] We.Train - UI config ..... SUCCESS [ 0.608 s]
[INFO] We.Train - All ..... SUCCESS [ 1.133 s]
[INFO] We.Train - Integration Tests ..... SUCCESS [ 23.745 s]
[INFO] We.Train - Dispatcher ..... SUCCESS [ 0.312 s]
[INFO] We.Train - UI Tests ..... SUCCESS [ 0.454 s]
[INFO] We.Train - Project Analyser ..... SUCCESS [ 43.026 s]
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 04:07 min
[INFO] Finished at: 2021-07-16T17:36:55-07:00
[INFO] -----
```

Task 3: Verify the installed content packages

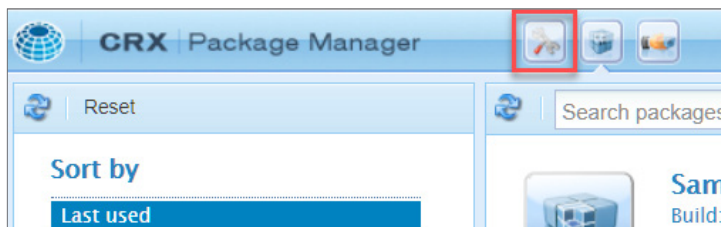
In this task, you will navigate to CRXDE Lite and verify the newly installed content package.

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 <AEM Project> packages, as shown:



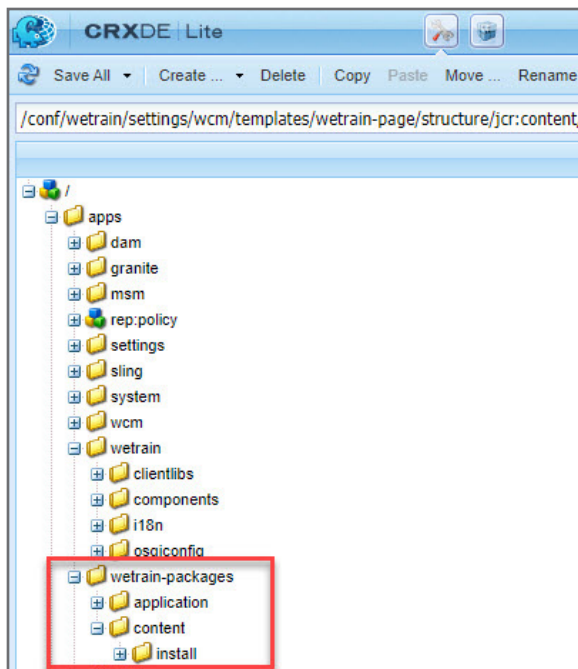
 **Note:** The profile `autoInstallSinglePackage` installed the container package `all`. This `all` package then installed all of the mutable and immutable content packages.

4. Click the Develop icon as shown:

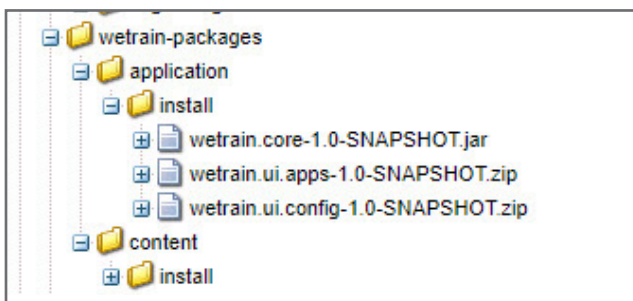


This takes you back to **CRXDE Lite**.


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



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

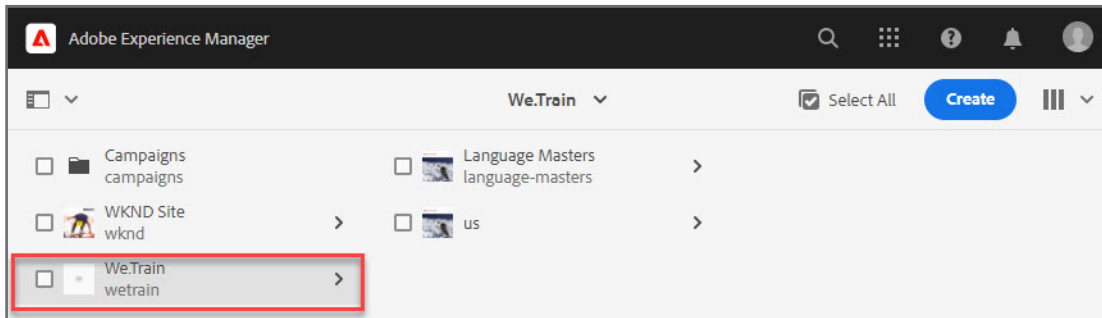


7. Similarly, navigate to the **/apps/<AEM Project>-packages/content/install** folder and notice the install of the **<AEM Project>.ui.content-1.0-SNAPSHOT.zip** which is the immutable content.

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

8. Go back to the browser tab that is open with the author service.
9. Click **Adobe Experience Manager** in the upper left.
10. In the Navigation page area, click **Sites**. The column view is displayed.

11. In the column view, verify the new <AEM Project> site. Navigate to <AEM Project> > us, as shown.



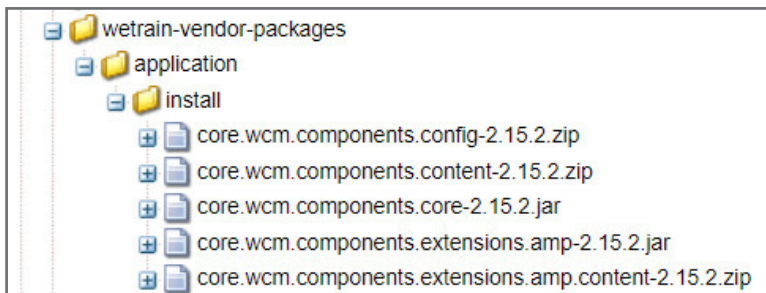
You have successfully installed your project into AEM.


Task 4: Core Components Inclusion (6.5 only)

Scenario

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

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



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

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

2. Open up the **parent pom.xml** and find the property `<core.wcm.components.version>` to view the version that is a part of your project:

```
<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>2.15.2</core.wcm.components.version>
  <bnd.version>3.1.2</bnd.version>
  <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
  <componentGroupName>We.Train</componentGroupName>
</properties>
```

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

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

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

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

References

You can use the following links for more information on:

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