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

02/22/2021

Contents

Cloud Manager Basics

Introduction

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

Objectives

After completing this course, you will be able to:

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

Cloud Manager

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

- Self service Interface

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

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

- CI/CD Pipeline

  CI/CD pipeline helps to speed up the delivery of custom code or updates such as adding new components on the website.
  Through the Cloud Manager UI, customers can configure and kick off their CI/CD pipeline. During this pipeline, a thorough code scan is executed to ensure that only high-quality applications pass through to the production environment.

- Flexible Deployments

  Cloud Manager offers customers flexible and configurable deployment modes so they can deliver experiences according to changing business demands.
  The code is automatically deployed to an environment based on specific events such as code commit. You can also schedule code deployments during specified time frames, even outside business hours.

- Extension and Automation through Adobe IO

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

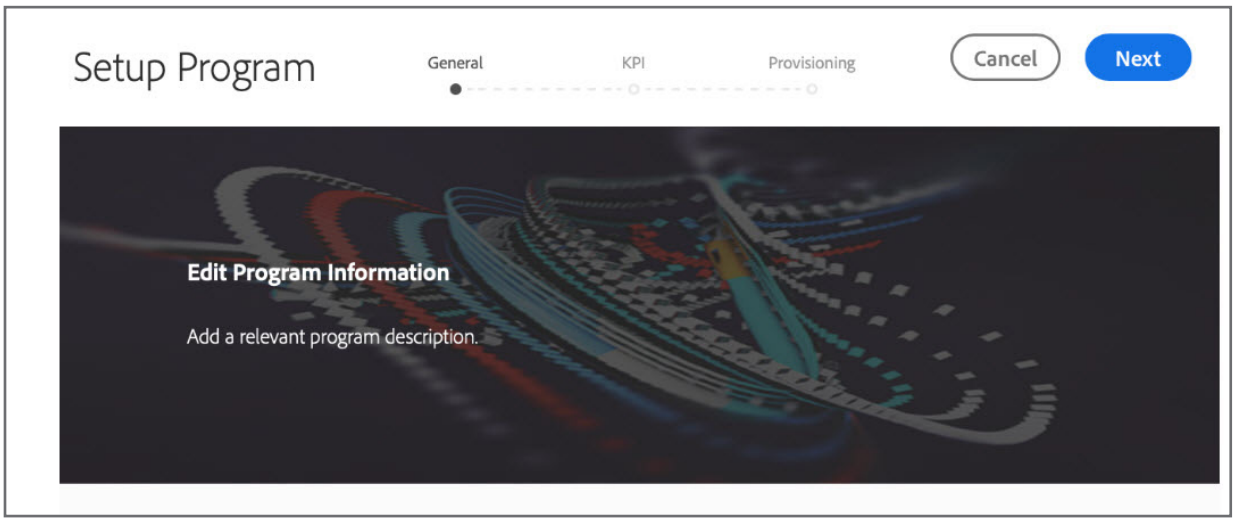
Setup a Program

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

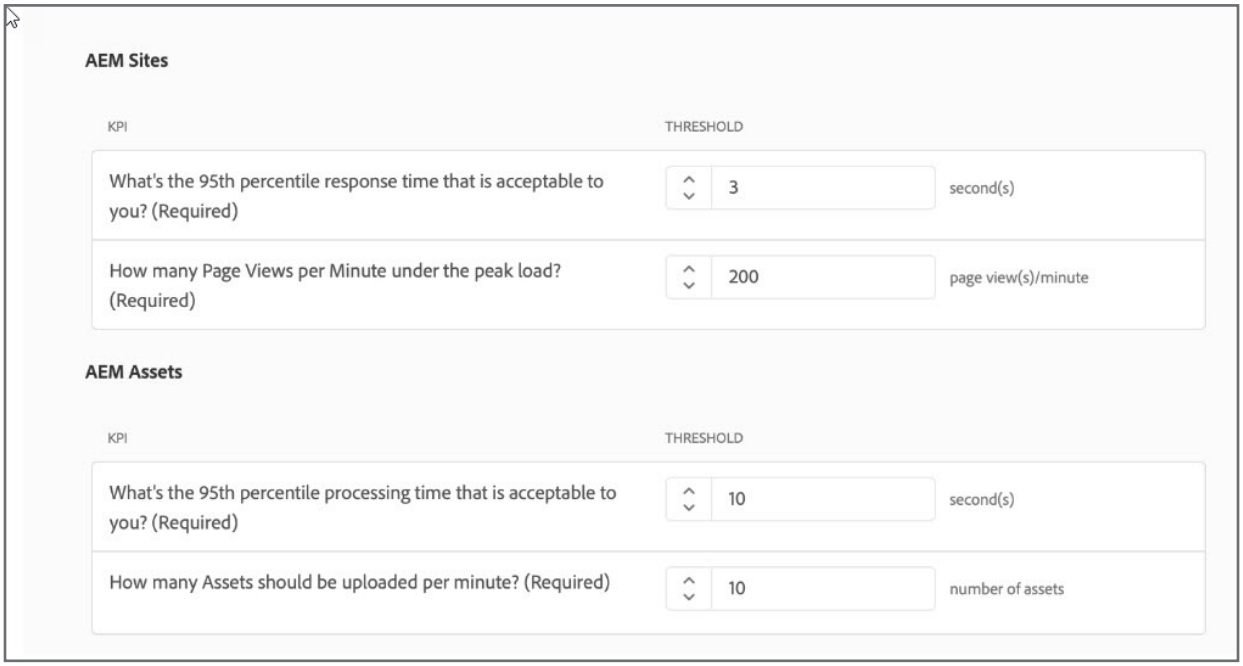
A Program is a high-level entity that contains production and non-production environments which have author, publish and dispatcher services. You can also have program-level configurations, which correspond to your SLA and APIs.

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

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



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



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

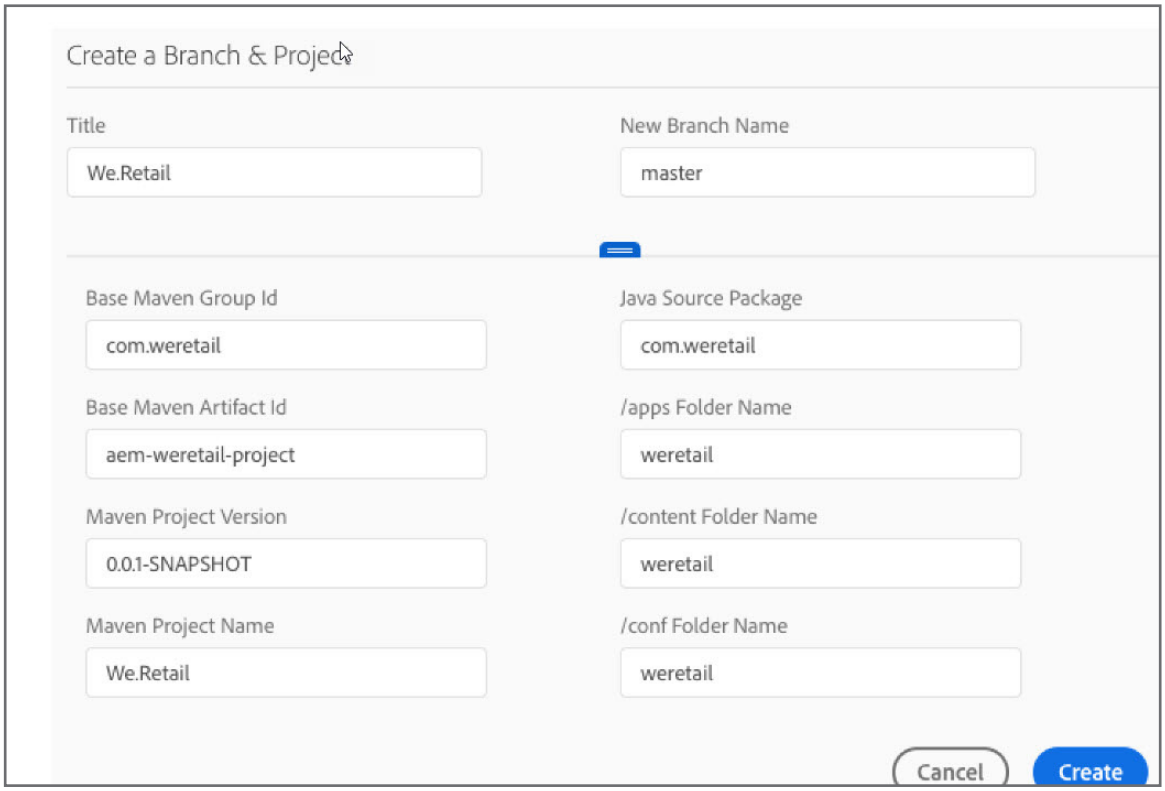
You can also edit the program once the initial program has already been set up.

Creating an AEM Application

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

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

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



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

Setting up Project

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

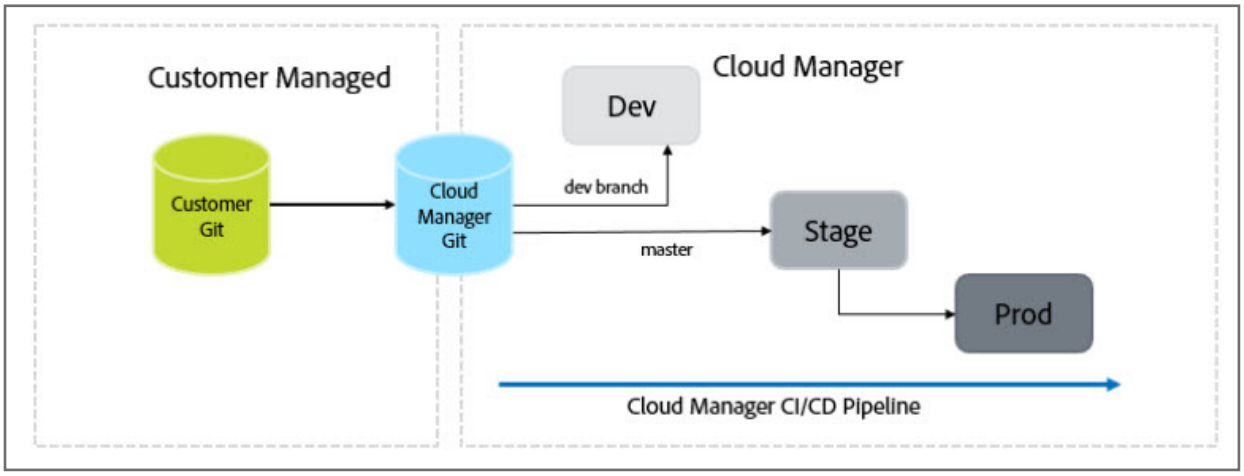
- Projects must be built using Apache Maven.
- There must be a pom.xml file in the root of the Git repository. This pom.xml file can refer to as many submodules (which in turn may have other submodules, and so forth) as necessary.
- You can add references to additional Maven artifact repositories in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

Building Environment Details

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

Managing your code

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



Cloud Manager Repository:

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

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

Customer Managed Repository

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

Integrating Git Repository with Cloud Manager Repository

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

CI/CD Pipeline

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

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

CI/CD Production Pipeline:

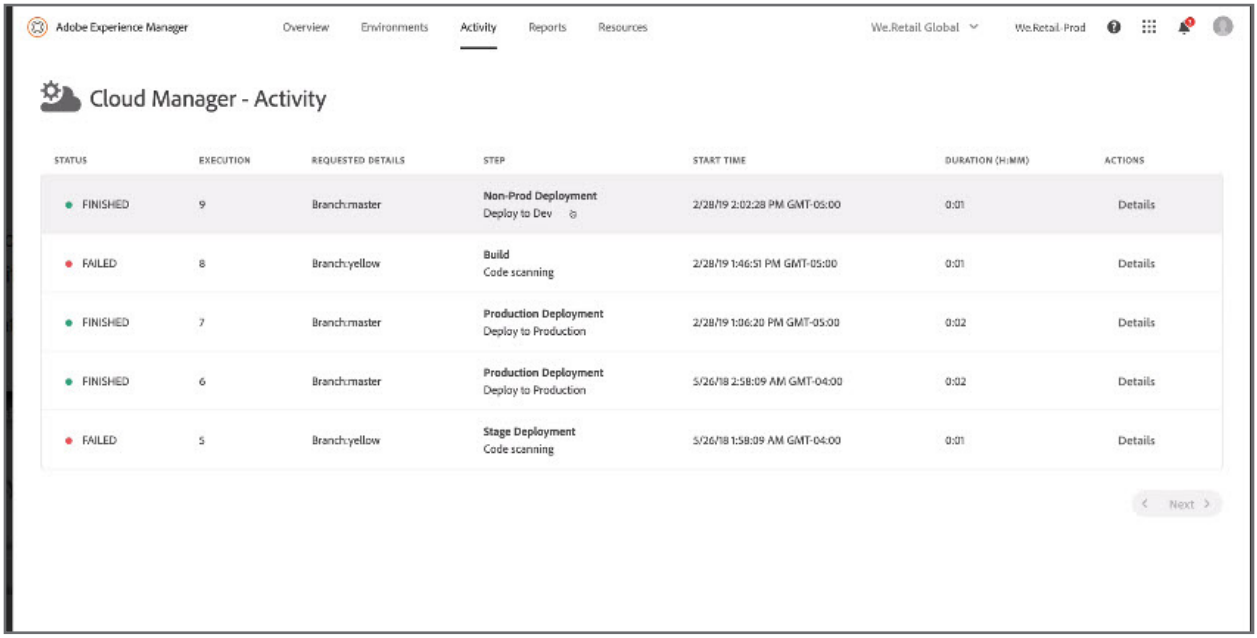
The CI/CD Production Pipeline configuration defines the trigger that will initiate the pipeline, parameters controlling the production deployment and performance test parameters. The CI/CD Production Pipeline is used to build and deploy code through Stage to the Production environment, decreasing time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

CI/CD Non-Production Pipeline:

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

Cloud Manager Activity

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



Pipeline configuration in Cloud Manager

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

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

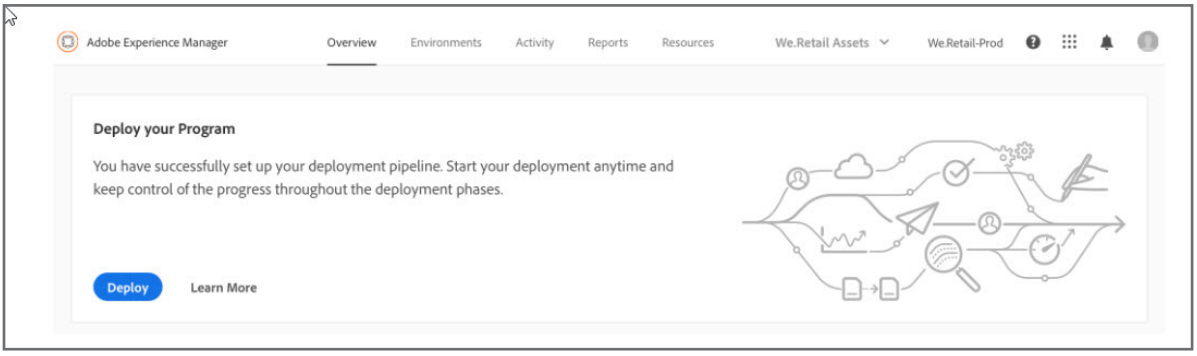
Once you configure your CI/CD Pipeline, you can deploy your code.

The Stage Deployment , involves the following steps:

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

The Production Deployment , involves the following steps:

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



Logs through UI

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

Logs through API

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

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

To tail logs:

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

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

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

SonarQube

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

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

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

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

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

Custom Code Quality rules for Adobe Cloud Manager

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

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

Example of non-compliant vs compliant code:

Non-Compliant Code

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

Compliant Code

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

Exercise 1: Observe the custom code quality rules

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

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

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

```
1.  package com.adobe.training.core;
2.
3.  import javax.jcr.security.AccessControlList;
4.  import javax.jcr.security.AccessControlManager;
5.  import javax.jcr.security.Privilege;
6.  import javax.jcr.Session;
7.
8.  import org.osgi.service.component.annotations.Component;
9.
10. import org.apache.jackrabbit.api.security.user.Authorizable;
11. import org.apache.jackrabbit.api.security.user.UserManager;
12. import org.apache.jackrabbit.commons.jackrabbit.authorization.AccessControlUtils;
13. import org.apache.sling.jcr.base.util.AccessControlUtil;
14. import org.osgi.framework.Constants;
15. import org.slf4j.Logger;
16. import org.slf4j.LoggerFactory;
17.
18. import com.adobe.granite.workflow.WorkflowException;
19. import com.adobe.granite.workflow.WorkflowSession;
20. import com.adobe.granite.workflow.exec.WorkItem;
21. import com.adobe.granite.workflow.exec.WorkflowProcess;
22. import com.adobe.granite.workflow.metadata.MetaDataMap;
23. import com.adobe.granite.workflow.exec.WorkflowData;
24.
25. /**
26.  * This is a project workflow process step. When starting a workflow from a project, you can specify the:
27.  *
28.  * Payload: The page that will have ‹modify› permissions added/removed from its ACL
29.  * User: The user attached to the ACL
30.  *
31.  * This process will add/remove jcr privileges that correlate to ‹Modify, Create, Delete› in the /useradmin
32.  * for the given user on the payload. An argument is used to specify to add or remove ‹modify› permissions.
33.  * The argument must follow permission=<add || remove>
34.  */
35. @Component(service = WorkflowProcess.class,
36.         property = {Constants.SERVICE_DESCRIPTION + "=A sample workflow process implementation.",
37.                 Constants.SERVICE_VENDOR + "=Adobe Digital Learning Services",
38.                 "process.label=Auto Assign Edit Permissions"
39.         })
40.
41. public class AutoAssignACL implements WorkflowProcess{
42.
43.     private final Logger logger = LoggerFactory.getLogger(getClass());
44.
45.     /**
46.      * @param item - Holds the payload and user
47.      * @param workflowSession - Session with service user (workflow-process-service) that will be modifying the permissions
48.      * @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.getPrprincipal(), privileges);
85.
86.             //Based on the process step arguments, permissions are either added or removed
87.             //Assumes arguments are given as: permission=add
88.             String arguments = workflowArgs.get("PROCESS_ARGS", "");
89.             if(arguments.contains("permission")){
90.                 String permission = arguments.split("=")[1];
91.                 if(permission.equals("add")){ //Adds permissions to edit the payload
92.                     logger.info("Adding permissions");
93.                     accessControlManager.setPolicy(payload, acl);
94.                     logger.info("Added modify permissions to " + payload + " for " + userID);
95.                 }else if(permission.equals("remove")){ //Removes permissions to edit the payload
96.                     logger.info("Removing permissions");
97.                     accessControlManager.removePolicy(payload, acl);
98.                     logger.info("Removed modify permissions to " + payload + " for " + userID);
99.                 }
100.            }else{
101.                //do nothing, if the workitem argument is not set
102.                logger.info("No arguments given for workitem");
103.            }
104.
105.            jcrSession.save();
106.        } catch (Exception e) {
107.            logger.error(e.getMessage(), e);
108.            e.printStackTrace();
109.        }
110.    }
111. }
```

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

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

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

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

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

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

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

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

References

You can use the following links for more information on:

- Cloud Manager:
  https://docs.adobe.com/content/help/en/experience-manager-cloud-manager/using/introduction-to-cloud-manager.html
- Key concepts:
  https://docs.adobe.com/content/help/en/experience-manager-cloud-manager/using/overview/key-concepts.html
- Cloud Manager API: https://www.adobe.io/apis/experiencecloud/cloud-manager/docs.html
- Key Features of Cloud Manager:

https://helpx.adobe.com/experience-manager/kt/platform-repository/using/cloud-manager-feature-video-understand.html

- Manage Environments - Cloud Service:

https://docs.adobe.com/content/help/en/experience-manager-cloud-service/implementing/using-cloud-manager/manage-environments.html

- SonarQube: https://docs.sonarqube.org/latest/user-guide/rules/
- Code quality rules: https://docs.adobe.com/content/help/en/experience-manager-cloud-manager/using/how-to-use/custom-code-quality-rules.html
- Functional testing docs: https://experienceleague.adobe.com/docs/experience-manager-cloud-service/implementing/using-cloud-manager/test-results/functional-testing.html
- UI Tests: https://experienceleague.adobe.com/docs/experience-manager-cloud-service/implementing/using-cloud-manager/test-results/ui-testing.html
- Testing documentation: https://docs.adobe.com/content/help/en/experience-manager-cloud-manager/using/how-to-use/understand-your-test-results.html
- AEM Rules for SonarQube: https://github.com/wttech/AEM-Rules-for-SonarQube
- CQ: https://docs.adobe.com/content/help/en/experience-manager-cloud-manager/using/how-to-use/custom-code-quality-rules.html
- Pipelines: https://experienceleague.adobe.com/docs/experience-manager-cloud-manager/using/how-to-use/configuring-pipeline.html
- Pipelines - Non-production: https://experienceleague.adobe.com/docs/experience-manager-cloud-manager/using/how-to-use/configuring-pipeline.html?lang=en#non-production-%26-code-quality-only-pipelines
- Managing Code