Code Testing in Pipelines

Agenda:

- Unit Testing
- Running a Code Quality pipeline of Unit tests failing
- Code Quality Rules
- Running a Code Quality pipeline of SonarQube tests failing
- Tests specific to Production Pipeline

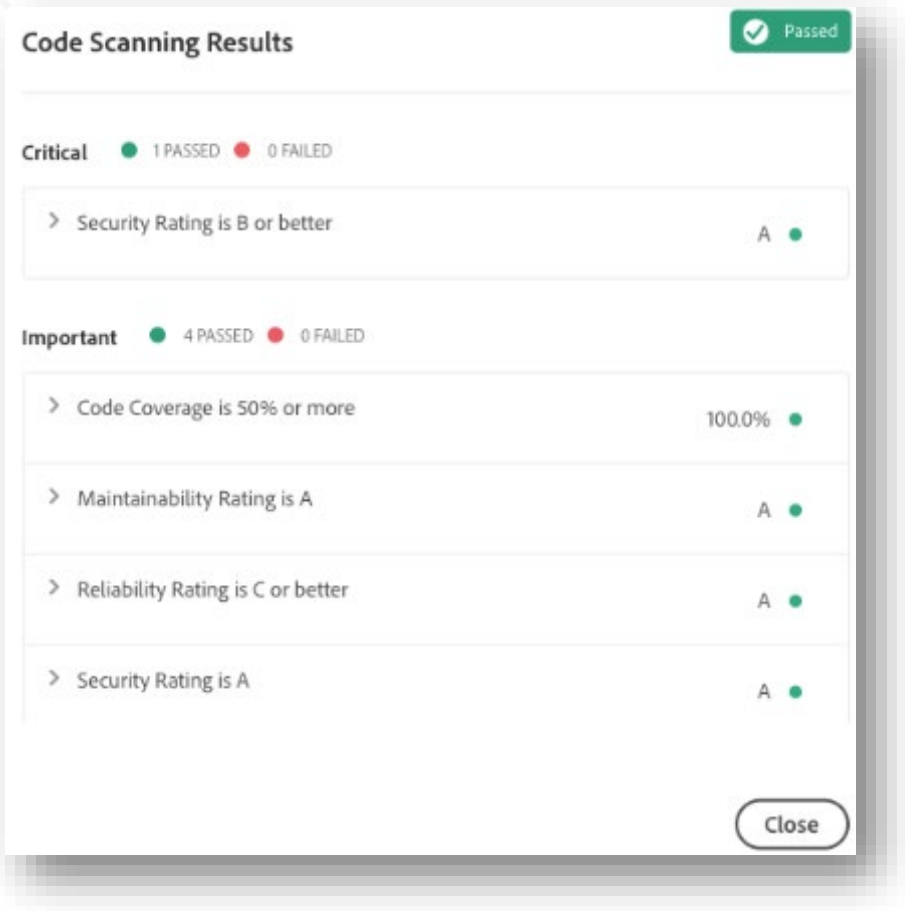
Code scanning

Cloud Manager groups issues into three groups:

1. Code Quality
2. Performance Testing
3. Security Testing

Groups are separated into three severity levels

- **Critical** – fails the pipeline. Must be fixed to continue
- **Important** – Pauses the pipeline. Can be overridden or ignored
- **Information** – No impact on the pipeline execution



Unit testing

- JUnit tests
  - Build-time tests written in Java
  - JUnit tests are located in **core/src/test/java/** in your Maven project
  - Used to test the actual java classes in **core/src/main/java/**
    - **validates the output of a method against expected results**

Unit testing in CICD

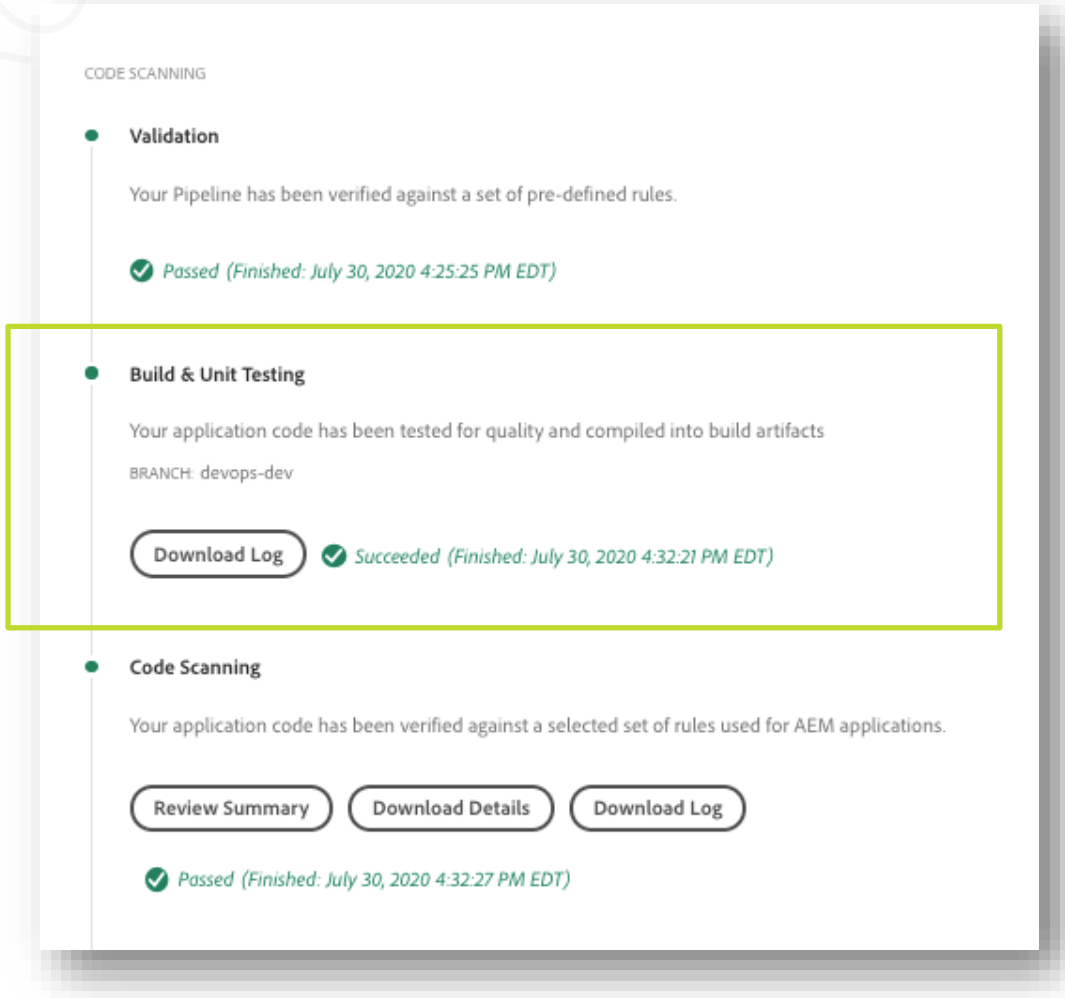
Performed after project validation

- Run using maven
- If unit tests fail, the pipeline fails
- If it passes, Code Scanning occurs to evaluate Code Quality

Code Quality has a metric called Code Coverage

Code Coverage metric is:

- Calculated by unit test coverage
- Considered an Important issue (severity)
- > 50% coverage to pass



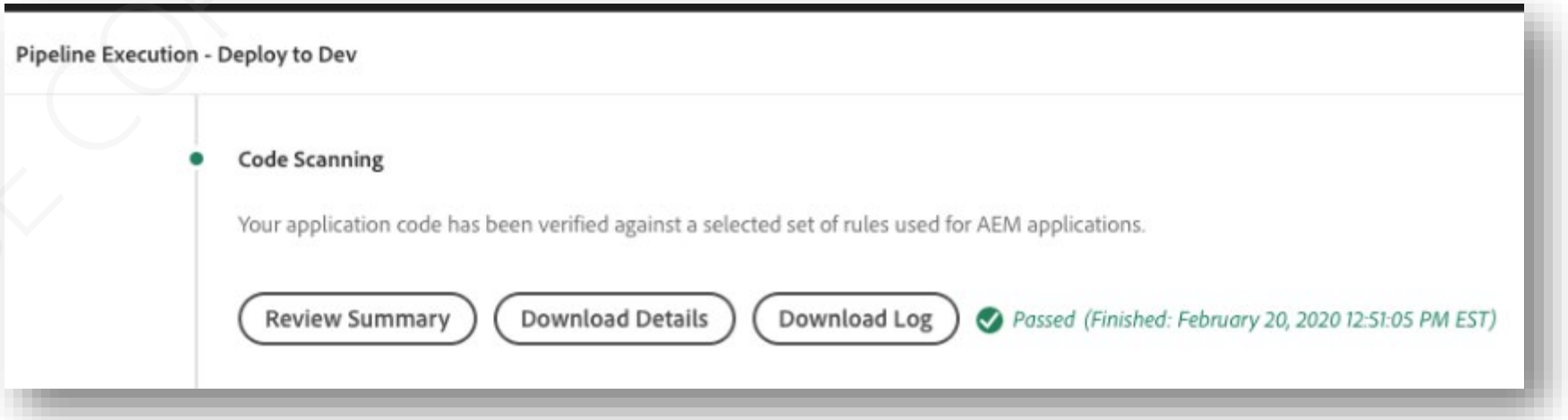
Code scanning

Code will not be loaded onto Adobe Experience Manager (AEM) as a Cloud service if there are **Critical** issues

Best practices are enforced

CSV reports created by SonarQube are extremely useful

- Use SonarQube Rules documentation as help
- Use AEM's custom rules documentation as help



SonarQube

- Standard open source testing platform
- Used to measure and analyze the source code quality
- Four types of rules used:
  - Code Smell (Maintainability domain)
  - Bug (Reliability domain)
  - Vulnerability (Security domain)
  - Security Hotspot (Security domain)

Code quality using SonarQube

Runs on all pipeline types (code quality, development, production)

Groups issues into four categories

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

Rules executed:

- Squid - Standard SonarQube Java rules
- Sonar Plugins
  - AEM – AEM Rules (By Cognifide)
  - CQ – AEMaaCS Rules (By Adobe)

Understand your test results

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,owasp
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,hiber
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 CSV output

File Location	Line Number	I
bundle:pathto/TitlePropertyListener.java	48	A
bundle:pathto/AutoAssignACL.java	106	(
bundle:pathto/AutoAssignACL.java	107	I
bundle:pathto/AutoAssignACL.java	108	I
bundle:pathto/TitlePropertyListener.java	40	(

Exact file, line#, what to fix and how long

Full categorization (Type, Severity, Rule, Tags)

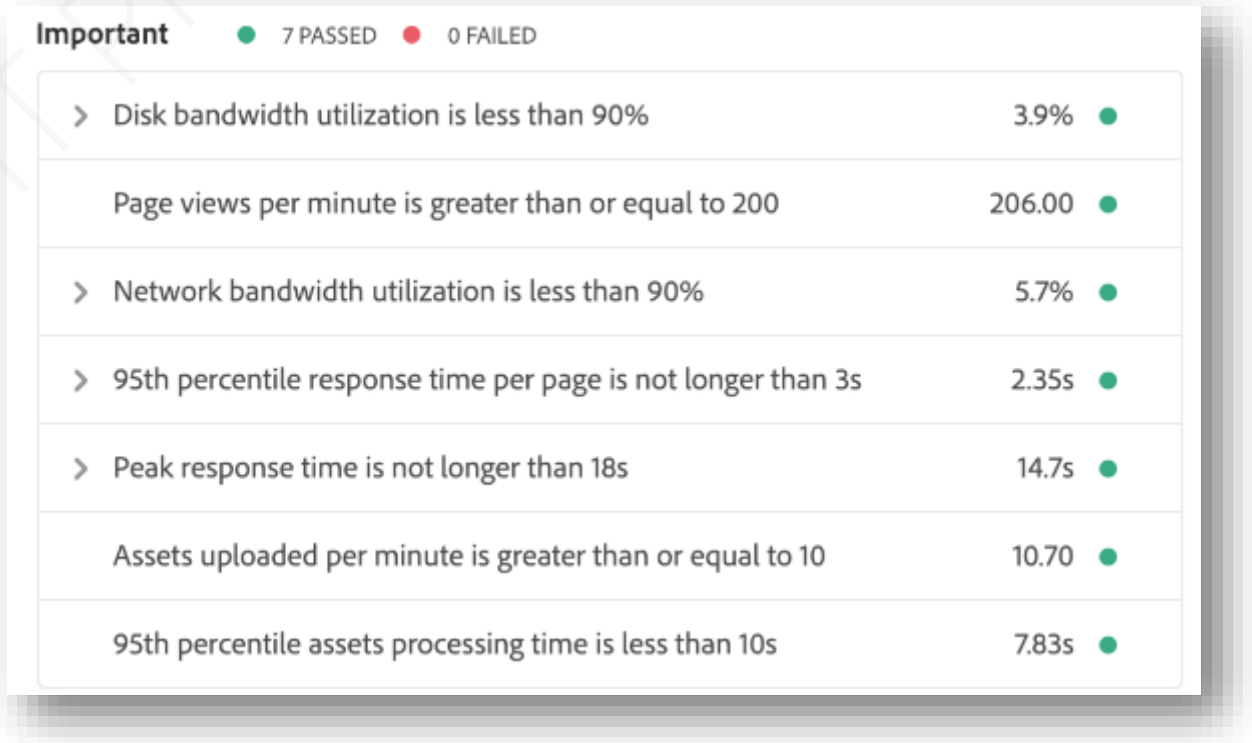
Quick documentation links

Issue	Type
A "NullPointerException" could be thrown; "changedRes" is nullable here.	Bug
Catch a list of specific exception subtypes instead.	Code Smell
Do not use Exception.getMessage() as the first parameter of the Logger.log statement	Code Smell
Do not use Exception.printStackTrace() to print to console	Code Smell
Use constant JCR_TITLE from interface com.day.cq.commons.jcr.JcrConstants instead of hardcoded value.	Code Smell

Severity	Effort	Rule	Tags	Documentation
Major	10min	squid:S2259	cert,cwe	https://www.adobe.com/go/aem_cmcq_s2259_en
Major	15min	squid:S2221	cwe,error-ha	https://www.adobe.com/go/aem_cmcq_s2221_en
Minor	15min	CQRules:CQBP-44---ExceptionGetMessageIsFirstLogParam	cqsoftwareq	https://www.adobe.com/go/aem_cmcq_cqbp-44---exceptiong_en
Minor	15min	CQRules:CQBP-44---ExceptionPrintStackTrace	cqsoftwareq	https://www.adobe.com/go/aem_cmcq_cqbp-44---exceptionp_en
Minor		AEM Rules:AEM-2	aem	https://www.adobe.com/go/aem_cmcq_aem-2_en

Performance and security tests

- Production Pipeline only
  - Runs on Stage Environment
- Performance Testing
  - Based on KPIs on program
  - Loaded content into pipeline
- Security Testing
  - Based on existing AEM health checks
  - Pass/Fail only



Key takeaways

- Unit Testing in CICD
- Performed after project validation
  - Run using maven
  - If unit tests fail, the pipeline fails
- SonarQube
  - Standard open source testing platform
  - Used to measure and analyze the source code quality
- Code quality using SonarQube
- Groups issues into 4 categories
  - Code Smell (Maintainability domain)
  - Bug (Reliability domain)
  - Vulnerability (Security domain)
  - Security Hotspot (Security domain)