

**Hewlett Packard**  
Enterprise

# DevOps Automation

Calculate satisfying items o, p, q and r in  
Attachment 1, Section 2

Feb 15, 2017



---

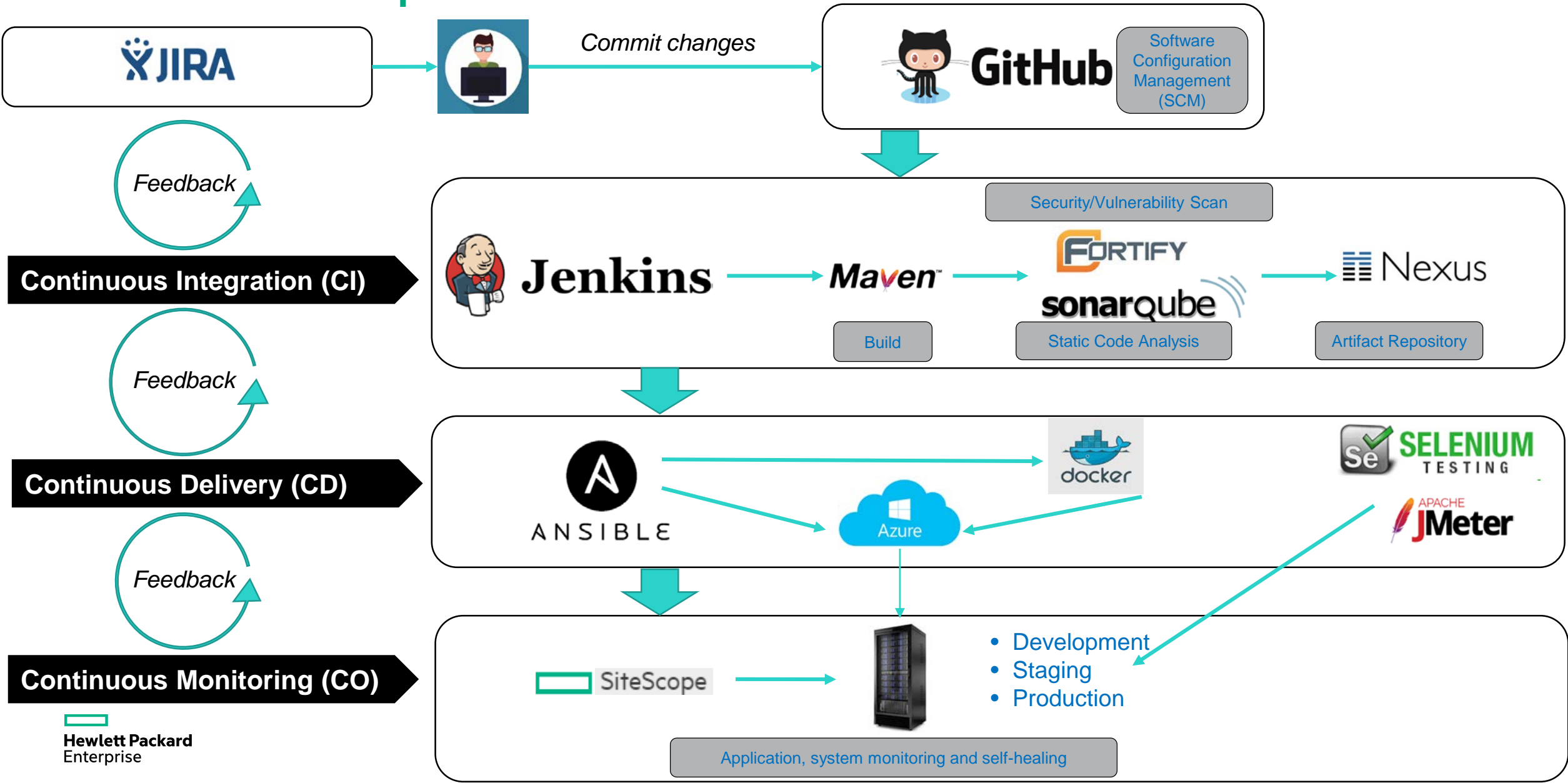
# Contents

The Cal eStore - DevOps Architecture slide pertains all of the following topics in our Technical Approach. Other slides in this deck pertain to the topics under which they are listed below.

- o. Setup or used a continuous integration system to automate the running of tests and continuously deployed their code to their IaaS or PaaS provider
  - Jenkins
  - SonarQube
  - HPE Fortify
  - Nexus
- p. Setup or used configuration management
  - Ansible
  - Nexus
- q. Setup or used continuous monitoring
  - SiteScope
- r. Deployed their software in an open source container, such as docker (i.e., utilized operating-system-level virtualization)
  - docker

The Container Based Deployment slide illustrates how we used these tools together for Continuous Integration (CI) / Continuous Delivery/Deployment (CD) process flow.

# Cal eStore - DevOps Architecture

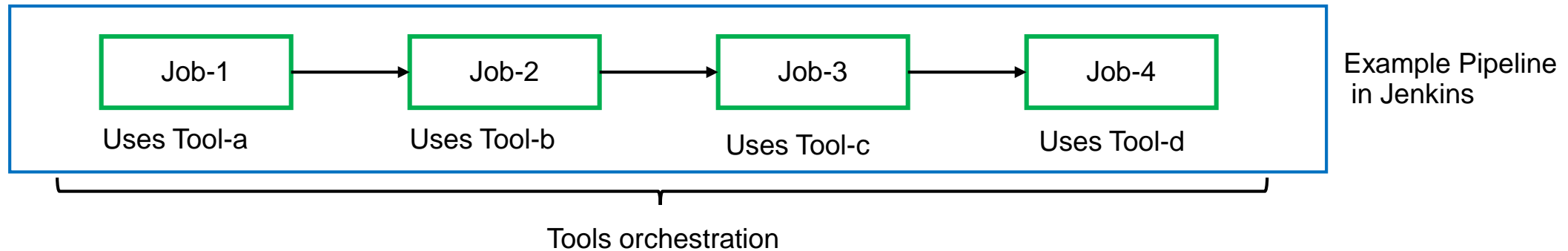


# Jenkins

Version 2.32.1

We used Jenkins, an orchestration tool, to automate the Continuous Integration process flow.

- Each step in the process flow (e.g., Build, Deployment, Testing) is a job in Jenkins that helps to identify issues quickly and easily. Each job uses separate tools to perform desired operations using plugins.



- We configured the jobs to send success or failure notification messages to team members. Messages immediately are sent in to team members in Outlook. The following slide is an example of an email notification received by a member of our team.

# Jenkins Email - Failure Notification

- Failure notification example (The team member clicks the link in the email to research and resolve the problem.)



The team member sees error details similar to those shown in the following example.

```
INFO] -----
INFO] Total time: 3.285 s
INFO] Finished at: 2017-02-24T15:46:11+05:30
INFO] Final Memory: 21M/245M
INFO] -----
ERROR] Failed to execute goal org.apache.maven.plugins:maven-compiler-plugin:3.1:compile (default-compile) on project calEStore: Compilation failure
ERROR] /home/hpuser/main-ca/src/main/java/com/hpe/calEStore/model/User.java:[24,17] cannot find symbol
ERROR] symbol:   class Cool
ERROR] location: class com.hpe.calEStore.model.User
ERROR] -> [Help 1]
ERROR]
ERROR] To see the full stack trace of the errors, re-run Maven with the -e switch.
ERROR] Re-run Maven using the -X switch to enable full debug logging.
ERROR]
ERROR] For more information about the errors and possible solutions, please read the following articles:
ERROR] [Help 1] http://cwiki.apache.org/confluence/display/MAVEN/MojoFailureException
BUILD step 'Execute shell' marked build as failure
mail was triggered for: Failure - Any
ending email for trigger: Failure - Any
ending email to: satyanarayana.raju@hpe.com syed.rah.faiz@hpe.com mohd-abrar.khote@hpe.com sureshbabu@hpe.com nihar.mishra@hpe.com suman-babu.sangaraju@hpe.com
ata.revankar@hpe.com akumar@hpe.com noothan.y-v@hpe.com ravi.g.kumar@hpe.com apoorva.rao@hpe.com kamalesh.r.uppoor@hpe.com
Warning: you have no plugins providing access control for builds, so falling back to legacy behavior of permitting any downstream builds to be triggered
```

# Jenkins Email- Success Notification

- After receiving a failure notification, the developer reviews the error log, resolves the issue, and retests the code. If all issues are resolved, the developer receives a build successful message similar to the following example.



- The team member can also view success details in a log similar to following example.

```
[INFO] Installing /home/hpuser/main-ca/pom.xml to /home/hpuser/.m2/repository/com/hpe/calEStore/1.0.0/calEStore-1.0.0.pom
[INFO] -----
[INFO] BUILD SUCCESS
[INFO] -----
[INFO] Total time: 7.228 s
[INFO] Finished at: 2017-02-25T11:26:06+05:30
[INFO] Final Memory: 23M/207M
```

# Jenkins

This example is an actual pipeline view that displays in Jenkins. For our process, when code is checked in to GitHub, the Jenkins pipeline is triggered. The color of boxes in the display indicate progress.

- Blue – Initial submission
- Yellow – Processing
- Green – Complete



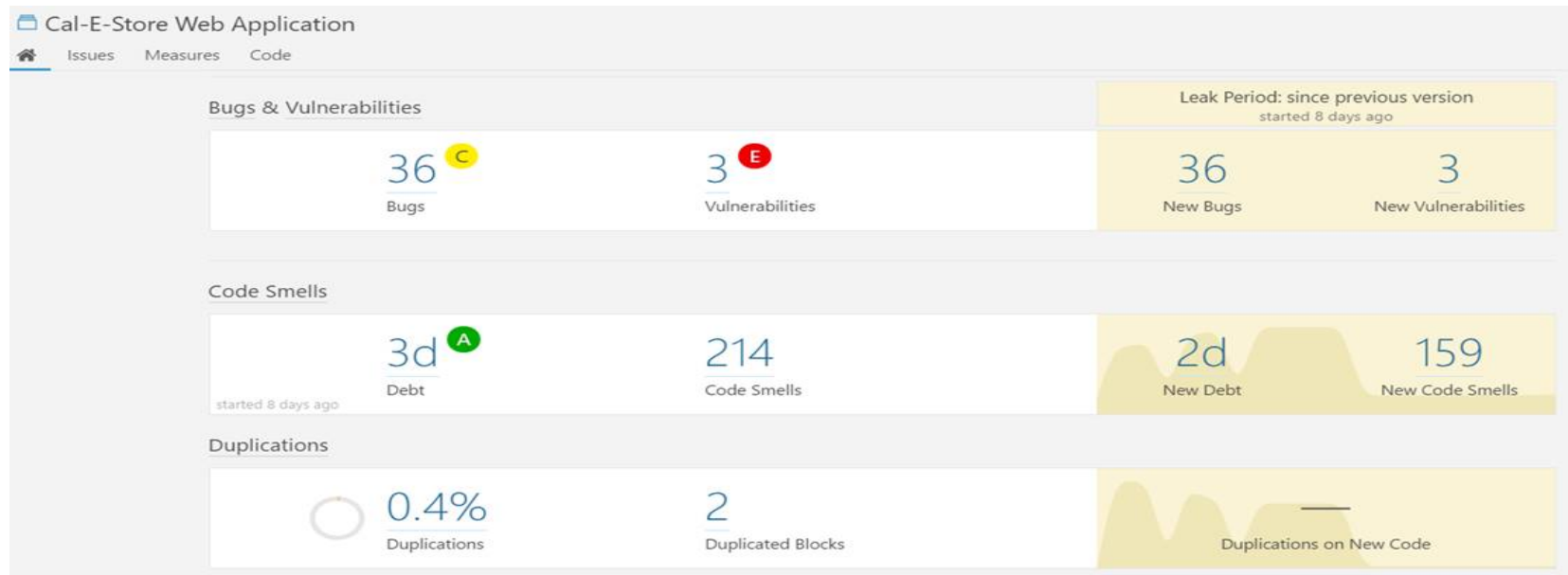
# SonarQube

## Version 6.2

We used SonarQube, a code quality analysis tool, to validate the quality of our code in relation to architecture and design, complexity, duplications, coding smells, and potential bugs.

SonarQube computed our SQALE rating, an application health rating, based on density of technical debt. This is essentially a comparison of the effort to fix the application versus the effort already expended. Ratings in the following example from our development effort indicate an “A Rating” showing that our code quality is healthy.

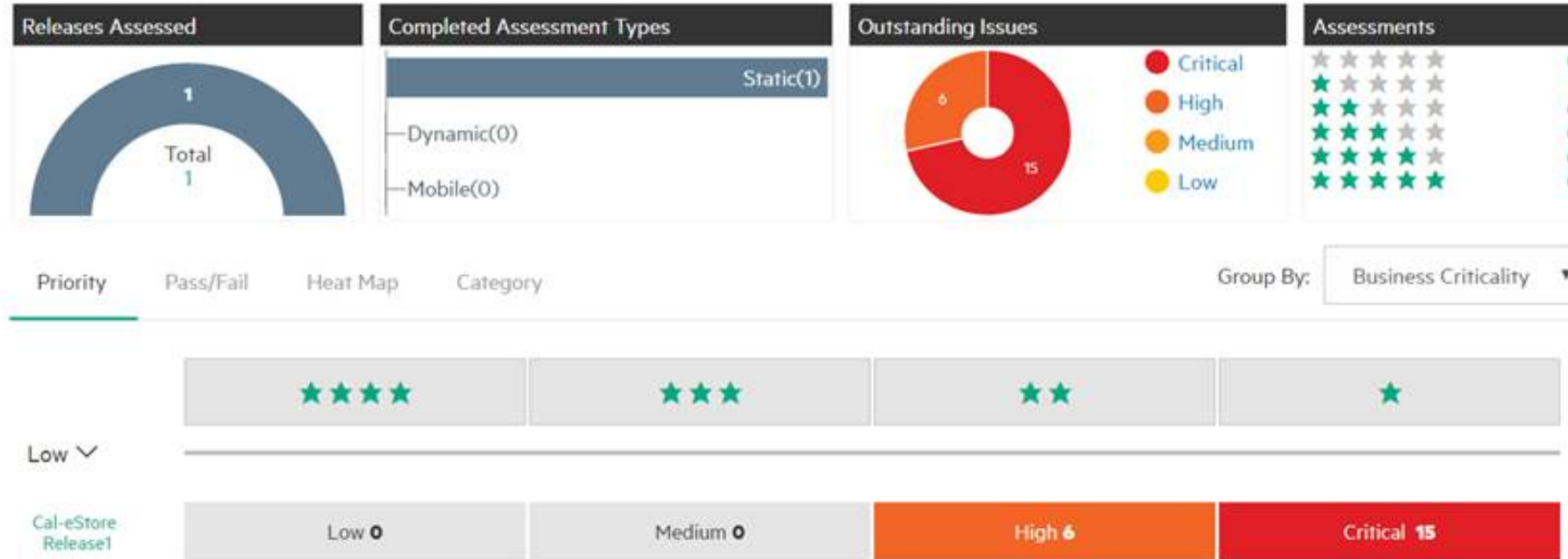
**NOTE:** We maintained a Standard SQALE A Rating throughout the project.





# HPE Fortify Security Scan Tool

HPE Security Fortify is an application security testing tool used to protect software applications from security vulnerabilities. The following example shows statics reported from a vulnerabilities scan within Cal eStore.

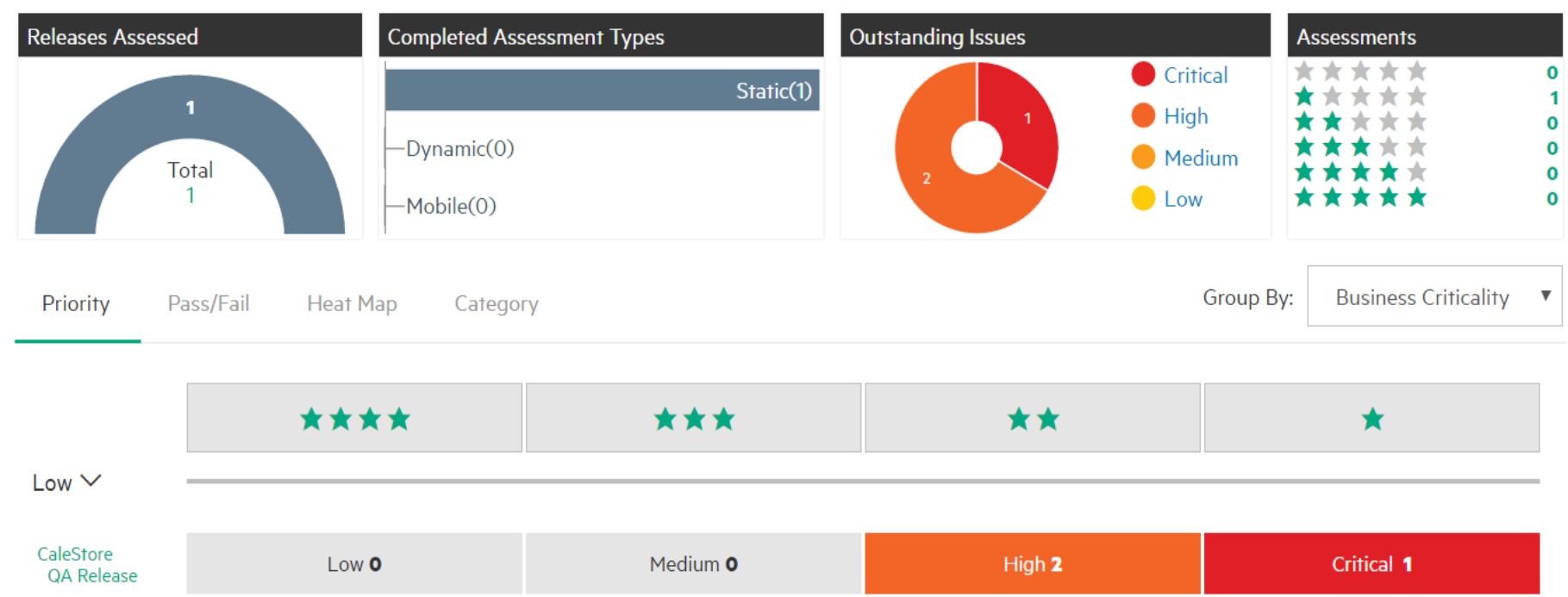


- The tool classifies issues into the following - Critical, High, Medium and Low.
- The Overall security rating for the application is displayed after the scan completes.
- A 1 star rating indicates that there are critical vulnerabilities, 5 stars indicates the application is secure.

In the preceding example, you see the results suggest that there are 15 critical issues and 6 high risk issues.

# HPE Fortify Security Scan Tool

When the web developers clicked on the issues, the tool opened the sections of code where the vulnerabilities or issue existed. The developers corrected the code to eliminate the problems.



After issues were identified and resolved, the developers rescanned to determine if all vulnerabilities and issues had been resolved. When there were no vulnerabilities, the tool displayed a 5 star rating indicating that the application is secure.

# Nexus

## Version 3.2.0

We used Nexus as the central repository for storing build artifacts and configuration files of significant size. With this tool we were able to restore and deploy to a particular version selected from the stored artifacts.

The screenshot shows the Nexus Repository Manager interface. The top navigation bar includes the Nexus logo, version 'OSS 3.2.0-01', a search bar, and user controls for 'admin' and 'Sign out'. The left sidebar has a 'Browse' menu with options: Welcome, Search, Browse, Assets, and Components (which is selected). The main content area is titled 'Components / StateOfCA' and features a table of components. The table has columns for Name, Group, and Version (sorted descending). The current component 'ca' is highlighted, showing versions 125 through 120. A 'Filter' input field is visible on the right side of the table.

Name	Group	Version ↓
ca	StateOfCA	125
ca	StateOfCA	124
ca	StateOfCA	123
ca	StateOfCA	122
ca	StateOfCA	121
ca	StateOfCA	120

### Version:

With Continuous Integration there are frequent changes made in build artifacts. As changes are made Nexus automatically upgrades the version.

In the example you see that our current version is 125.

---

# Ansible

## Version 2.2

Ansible is an open-source configuration management tool that does not require an agent running on the target nodes. This tool uses OpenSSH for connectivity between the master server and slave nodes.

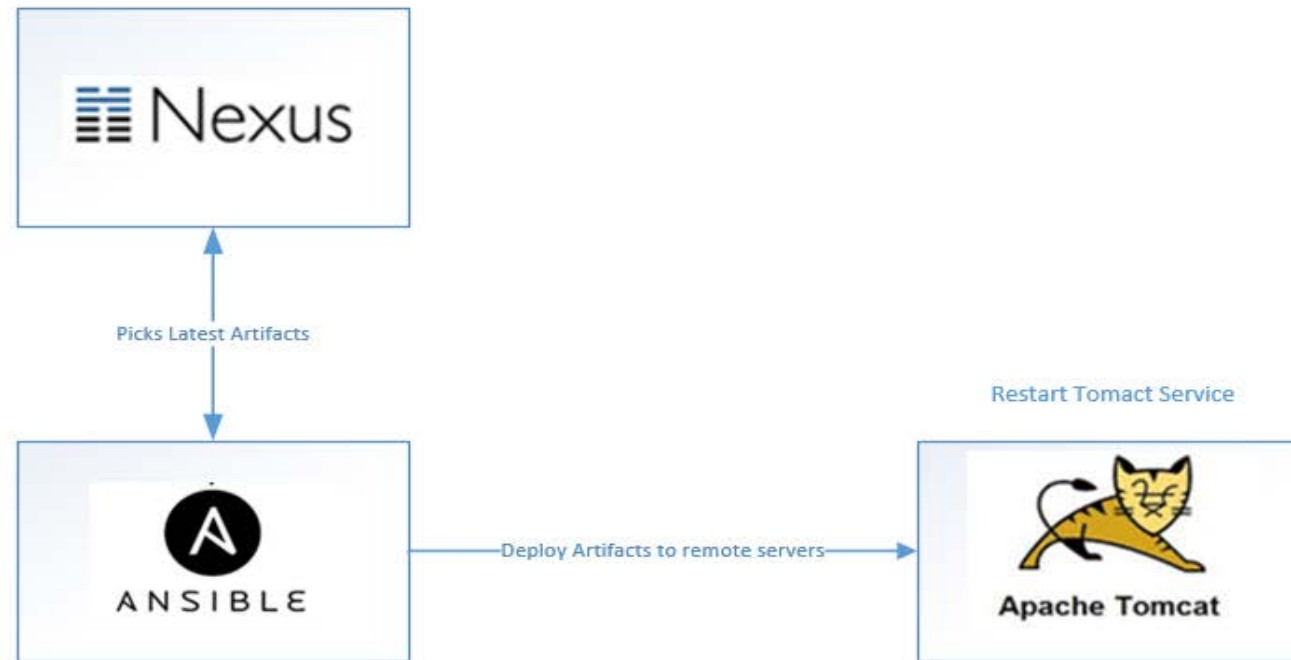
- We used Ansible to deploy and maintain infrastructure as code while creating docker images and containers.
- We wrote scripts in Ansible to perform actions on target nodes.
- When our playbook was executed for modifying the state of a resource on the nodes, Ansible determined if the resource was in a desired state. If the resource was in a desired state, no action was taken; otherwise Ansible modified the resource state to the desired state.



# Deployment Model for Ansible

The following flow illustrates our use of Ansible and how it connected to other tools in our deployment process. You can see that:

- Jenkins uploads the build artifacts (.war file) to Nexus. This invokes Ansible for deployment.
- Ansible picks up the latest artifact from Nexus and deploys it to the “webapps” folder across all target nodes.
- Tomcat service is restarted to complete the process.




# HPE SiteScope

We used HPE SiteScope to:

- Manage end-user status information requests, configure change requests, and for access control.
- Schedule and run different monitors for a particular time period, create alerts, and generate reports.
- Collect and monitor performance and availability information about the system.
- Automatically trigger email notifications.
- Create different type of reports for analysis.

## Summary for url\_check\_http://16.181.237.62:8080/ca

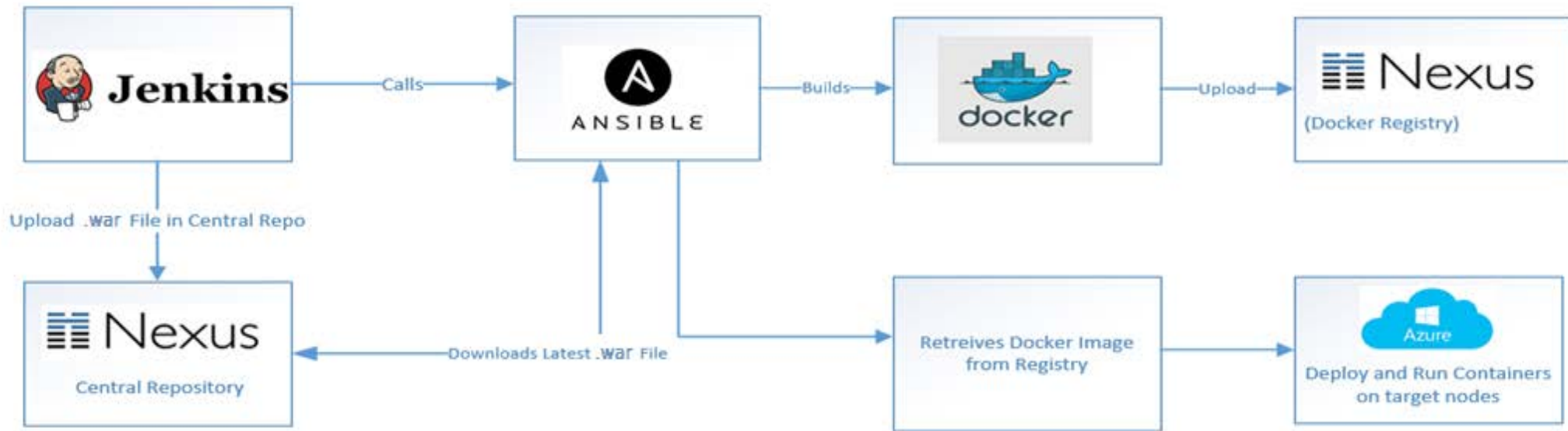
### Uptime Summary

Name	Uptime %	Error %	Warning %	Last
url_check_http://16.181.237.62:8080/ca	100	0	0	

### Measurement Summary

Name	Measurement	Max	Avg	Last
url_check_http://16.181.237.62:8080/ca	content match	0	0	n/a
url_check_http://16.181.237.62:8080/ca	roundtrip time (milliseconds)	230	23.57	15

# Container Based Deployment



- During the CI process, we generated a .war file and uploaded it into Nexus, which acts as a central repository for all of the artifacts.
- During deployment, Ansible was called to download the latest .war file from Nexus to build a docker image and upload that image to a docker registry (in this case Nexus).
- In the final step, Ansible was instructed to select the docker image from Nexus and deploy it to all of the target nodes, start the container, and keep all the environments consistent.