

USING JENKINS INTEGRATIONS TO BRIDGE YOUR MAINFRAME DEVOPS GAP



In modern business, digital innovation is your basis for competitive differentiation. DevOps is vital to that initiative, enabling business velocity with shorter delivery cycles, faster response to business needs and higher-quality deliverables. But DevOps can't be exclusive to your mainstream IT teams—it must include your mainframe teams as well.

By 2019, 64 percent of mainframe-equipped organizations will be running more than half their business-critical workloads on that system of record, according to a Compuware-commissioned Forrester Consulting study. These are applications and data that support 96 percent of new business initiatives, according to a previous study.

So, if the mainframe is core to successful digital transformation, organizations should ensure the platform leverages the same DevOps best practices for development quality, velocity and efficiency that non-mainframe teams are using because, when it comes to supporting rapid digital innovation, the mainframe community's go-to Waterfall development methodology falls flat.

The challenge is understanding how to approach your mainframe teams about making changes to their culture, process and tools to support mainframe DevOps.

Building a Mainframe DevOps Bridge

You need to build a mainframe DevOps bridge, and that takes changing mindsets. Compuware Executive Solutions Architect – DevOps [Rick Slade](#) wrote a great blog post about this—but he points out changing mindsets isn't all you have to worry about.

Supporting mainframe DevOps will require providing those teams with modern, intuitive mainframe tools that support iterative, Agile methods to facilitate faster system delivery without quality sacrifice.

Building a mainframe-inclusive DevOps toolchain is core to bumping your mainframe teams up to the same level as your non-mainframe teams that are using best practices like Agile Development and Continuous Integration/Continuous Delivery.

Compuware-Jenkins Integrations for Mainframe DevOps

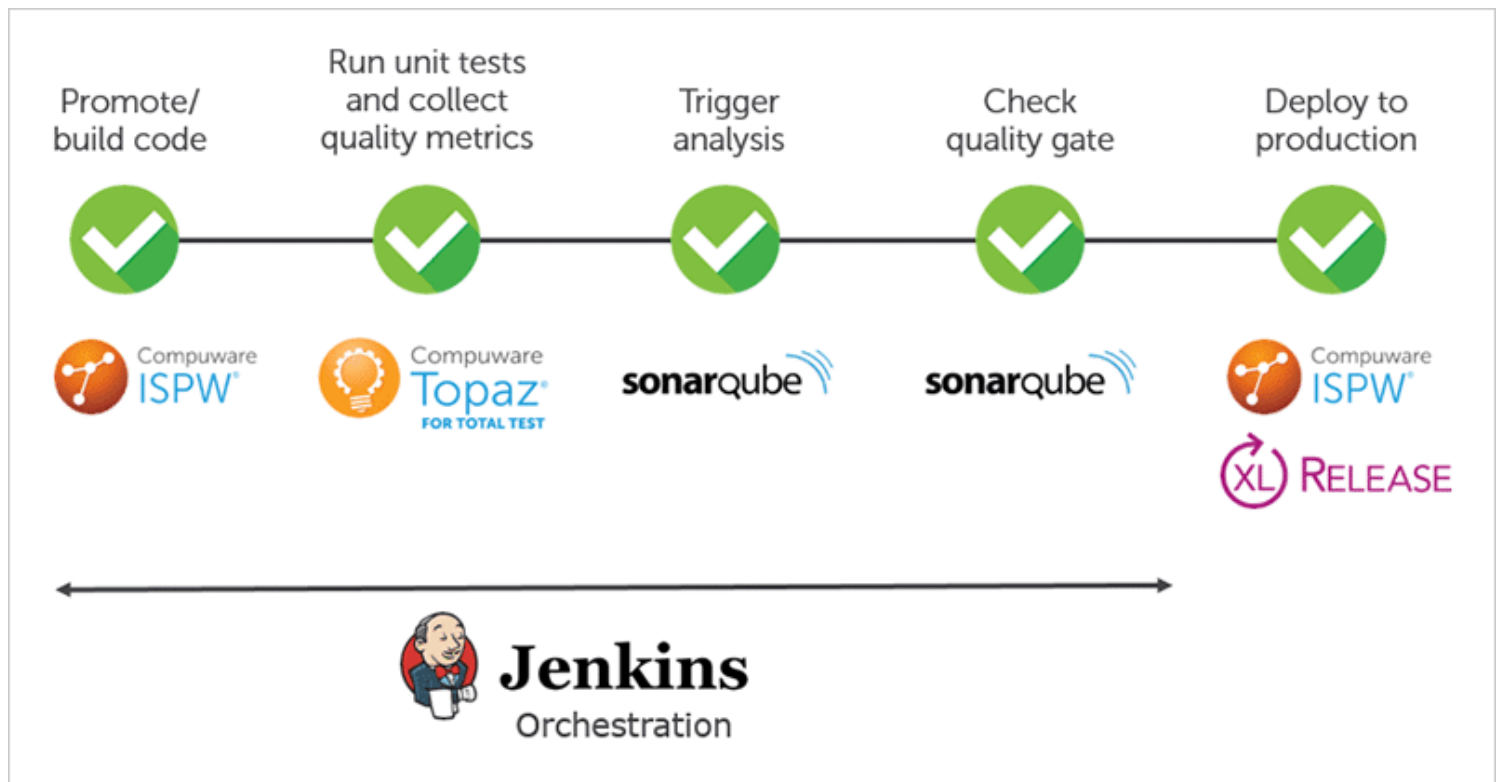
A mainframe-inclusive DevOps toolchain improves the quality and agility of your business software, while at the same time reducing costs. Compuware-Jenkins DevOps integrations are enabling customers to accelerate the software development process through orchestration and automation.

Orchestrate Your DevOps Pipeline

The integration between Jenkins and Compuware ISPW—our Agile source code management and deployment automation software—lets you build a DevOps pipeline in Jenkins to manage code throughout the development lifecycle. After a piece of code is edited and compiled within BMC Compuware's IDE, Topaz Workbench, and ISPW, Jenkins automates the promotion of the code into a staging area within ISPW for testing.

Automatically Trigger Unit Tests

Compuware Topaz for Total Test—our automated unit testing software—can be configured into an existing Jenkins workflow so COBOL unit testing is automatically triggered. ISPW can managed Topaz for Total Test components, enabling the migration of unit tests through the mainframe application release cycle to run specific tests for each level.



Enable Code Coverage

After a Topaz for Total Test unit test executes, code coverage results are published in Jenkins and [SonarSource SonarQube](#) to display which lines of code have or haven't been executed and what percentage of an application has or hasn't been tested and to compare test results to broader cross-platform quality trends and metrics.

The screenshot displays the SonarQube interface for a project named 'CodeCoverage'. The top navigation bar includes 'Overview', 'Issues', 'Measures', 'Code', 'Activity', and 'Administration'. The 'Quality Gate' is shown as 'Passed'. The main dashboard features two large metrics: '4 Bugs' (with a yellow 'C' grade) and '0 Vulnerabilities' (with a green 'A' grade). On the left sidebar, there are three sections: 'Code Smells' showing '2d Debt' (started 19 days ago, with a green 'A' grade), 'Coverage' showing '68.0%' (with a green circle), and 'Duplications' showing '36.5%' (with a red circle). The main content area shows a code editor for a file named 'PLAY/MF_Source / CWBBPGM4.cbl'. The code is a COBOL program with several sections. A tooltip 'Not covered by tests.' is visible over line 383. The code includes sections for processing data, hourly compensation, and overtime calculations.

```
374 ***** WRITTEN AND PROCESSING CONTINUES.
375 *****
376 1000-PROCESS-DATA.
377 IF HOURLY
378     PERFORM 2000-PROCESS-HOURLY
379 ELSE
380     IF SALES
381         PERFORM 3000-PROCESS-SALES
382     ELSE
383     IF MANAGEMENT
384         PERFORM 4000-PROCESS-MANAGEMENT
385     ELSE
386         MOVE ' INVALID EMPLOYEE TYPE ' TO ERROR-LINE
387         WRITE REPORT-RECORD FROM ERROR-LINE.
388     PERFORM 8000-READ-INPUT.
389 *****
390 ***** CALCULATE TYPE H (HOURLY) EMPLOYEE COMPENSATION. ANY
391 ***** EMPLOYEE WITH MORE THAN 40 HOURS RECEIVES OVERTIME COMPUTED
392 ***** AT 1.5 TIMES THEIR HOURLY RATE. ONCE EMPLOYEE COMPENSATION
393 ***** IS CALCULATED, IT IS STORED IN A HOLD TABLE. THE DATA IN
394 ***** THE HOLD TABLE IS USED FOR PRINTING THE EMPLOYEE COMPENSATION
395 ***** REPORT.
396 *****
397 2000-PROCESS-HOURLY.
398     MOVE ZERO TO OT-AMOUNT.
399     IF WA-EMP-HOURS GREATER THAN 40
400         COMPUTE EMP-WAGES = WA-EMP-RATE * 40
401         COMPUTE OT-HOURS = WA-EMP-HOURS - 40
402         COMPUTE OT-AMOUNT = OT-HOURS * (WA-EMP-RATE * 1.5)
403     ELSE
404         COMPUTE EMP-WAGES = WA-EMP-HOURS * WA-EMP-RATE.
405     COMPUTE EMP-COMPENSATION = EMP-WAGES + OT-AMOUNT.
406     ADD EMP-COMPENSATION TO GRAND-TOTAL-EMP.
407     CALL 'CHKXTDATE' USING END-OF-MONTH-SW
408         YRS-OF-SERVICE
409         TODAYS-DATE
410         WA-EMP-HIRE-DATE.
411     PERFORM 5000-STORE-EMPLOYEE-DETAIL.
```

Maintain Continuous Code Quality

Test metrics can be used in SonarSource Quality Gates to determine whether to proceed with a workflow or stop and fix the code before continuing. In the event of a failure, code can be automatically regressed back into a development environment in ISPW to be fixed.

[\(just show failures\)](#) [enlarge](#)

		Promote Code to Stage	Run Total Tests	SonarQube analysis	Quality Gate	Start release in XL Release
Average stage times:		1s	20s	13s	637ms	1s
#27	Jun 21 14:46 No Changes	1s	19s	13s	297ms <small>(paused for 15s)</small>	1s
#26	Jun 21 14:42 No Changes	2s	19s	13s	1s <small>(paused for 38s)</small> failed	
#25	Jun 21 13:43 No Changes	1s	28s	14s	500ms <small>(paused for 14s)</small>	1s

As mainframe workloads increase to support digital initiatives, there's a visible and growing gap between DevOps teams leveraging modern tools and methods and mainframe teams still leveraging the Waterfall status quo. DevOps leaders who understand the value of tools like Jenkins must be the ones to bridge the mainframe DevOps gap. That's much easier to do when you can present those teams with a modern toolset designed to leverage their talents in a new way for higher development quality, velocity and efficiency.